NOVEL MACHINE LEARNING ALGORITHMS FOR DETECTING BIOLOGICAL SIGNALS IN PROTEIN SEQUENCES: INVESTIGATION OF PLASMODESMATA

by

Jiefu Li

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Fall 2021

© 2021 Jiefu Li
All Rights Reserved

NOVEL MACHINE LEARNING ALGORITHMS FOR DETECTING BIOLOGICAL SIGNALS IN PROTEIN SEQUENCES: INVESTIGATION OF PLASMODESMATA

by

Jiefu Li

Approved: _____

Rudolf Eigenmann, Ph.D.
Interim Chair of the Department of Computer and Information Sciences

Approved: _____

Levi T. Thompson, Ph.D.
Dean of the College of Engineering

Approved: _____

Louis F. Rossi, Ph.D.
Vice Provost for Graduate and Professional Education and
Dean of the Graduate College

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Li Liao, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Cathy Wu, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Vijay Shanker, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Jung-Youn Lee, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

First, I would like to give tremendous and heartfelt gratitude to my advisor Dr. Li Liao for his relentless support, guidance, patience and encouragement in the past 5 years. Though he may not notice, there were hundreds of times I was frustrated and even wanted to quit, and it is really his wisdom and patience that have constantly inspired me through this memorable and challenging journey. From Dr. Liao, I understand what advisor really means, and I will for sure, pass what I have learnt to others, as a way to appreciate being so lucky to have Dr. Liao as my advisor.

Second, my dissertation committee members Dr. Cathy Wu, Dr. Vijay Shanker, and Dr. Jung-Youn Lee, I would like to express my profound appreciation for sharing their precious time and expertise, and providing me with insightful comments and suggestions. Also, I would like to give special gratitude to Dr. Gabriel Robles Luna, and Dr. Jung-Youn Lee again. Our collaboration between computation and biology is amazing. Without this collaboration and your wisdom, many interesting problems and great ideas on the computational side would not even be possible.

Moreover, I would like to give my appreciation to all my friends for the wonderful time I spent with you at UD. You have made this journey enjoyable and the memory with you will be there forever.

Finally, and most importantly, my deepest gratitude is to my families, my mother Yu Chen, and my father Zhonghua Li, and my fiancée Ran Hao. Without your continuous support, encouragement, and patience, my journey to Ph.D. would be much harder and can hardly imagine how it could be possible.

TABLE OF CONTENTS

LIST OF TABLES viii
LIST OF FIGURES x
ABSTRACT xii

Chapter

1 INTRODUCTION 1

1.1 Plasmodesmata Targeting Proteins, Signals, and Plasmodesmata Interactions 1
1.2 Hidden Markov Model 6

1.2.1 Basic Concepts 6
1.2.2 Tasks of HMM 8
1.2.3 Training Methods of HMM 9

2 DETECTING DE NOVO PLASMODESMATA TARGETING SIGNALS AND IDENTIFYING PD TARGETING PROTEINS 11

2.1 Introduction 11
2.2 Methods 13

2.2.1 SVM with Dipepetide Features 14
2.2.2 3-States HMM on JMe: PDHMM 16
2.2.3 Combination of SVM with HMM 17

2.3 Results 18

2.3.1 Datasets 18
2.3.2 Performance Metrics 20
2.3.3 Evaluation: SVM Alone 21
2.3.4 Evaluation: HMM Alone 21

2.3.5 Evaluation: Combination of SVM and HMM 21

2.4 Discussion and Conclusion 22

3 TRAINING HIDDEN MARKOV MODELS WITH PARTIAL LABEL DATA 26

3.1 Introduction 26
3.2 Method 28

3.2.1 Constrained Baum-Welch Algorithm 29
3.2.2 Model Selection Based on Partial Label Information 32

3.3 Experiments and Results 33

3.3.1 Synthetic Data 35
3.3.2 Real Data 39

3.4 Discussion and Conclusion 45

4 TRAINING HIDDEN MARKOV MODELS WITH POSITIVE AND NEGATIVE EXAMPLES 50

4.1 Introduction 50
4.2 Method 52

4.2.1 Introduction to HMM 52
4.2.2 Baum-Welch Revisit 54
4.2.3 Reverse Baum-Welch 56
4.2.4 CompHMM 58

4.3 Experiments and Results 59

4.3.1 Training Validation Check 59
4.3.2 Comparison Study on Synthetic Data 60
4.3.3 Applying on Real Data: Plasmodesmata-associated Protein Membership Prediction 63

4.4 Discussion and Conclusion 64

5 A NOVEL METHOD FOR INTER-HELIX CONTACT PREDICTION REFINEMENT USING LOCAL 2D

TOPOLOGICAL INFORMATION **66**

5.1 Introduction 66
5.2 Methods 68

5.2.1 Pitfall of Residue Contact Prediction Refinement and Fuzziness Score 72
5.2.2 2D Contact Structure Model 75
5.2.3 Inter-helix Prediction Refinement 76

5.3 Experiments and Results 79

5.3.1 Dataset 80
5.3.2 2D Contact Structure Model Validation 81
5.3.3 Fuzziness Score Validation 82
5.3.4 Inter-Helix Prediction Refinement and Refinement Selection Scheme 83

5.4 Discussion and Conclusion 85

6 SUMMARY AND FUTURE WORK **89**

6.1 Summary 89
6.2 Future Work 90

BIBLIOGRAPHY **93**

Appendix

A DATASET IN PD LOCALIZATION PREDICTION **99**
B MULTI-PATHES TRANSMEMBRANE PROTEIN DATASET USED IN TRAIN HMM WITH PARTIAL LABEL **102**
C DATASET USED IN INTER-HELIX PREDICTION REFINEMENT **103**
D PERMISSIONS **105**

LIST OF TABLES

1.1 Notations for hidden Markov models 10

2.1 Different types of proteins in dataset A 19

2.2 Comparison for all method in dataset B 22

3.1 Improvements of cBW + model selection, cBW alon vs Scheffer et al, with Fully Connected Initial Transition Matrix for Synthetic Data with Viterbi Algorithm 40

3.2 Improvements of cBW + model selection, cBW alone vs Scheffer et al in Cases of Correct Initial Transition Matrix for Synthetic Data with Viterbi Algorithm 41

3.3 Improvements of cBW + model selection, cBW alon vs Scheffer et al, with Fully Connected Initial Transition Matrix for Synthetic Data with Posterior-Viterbi Algorithm 42

3.4 Improvements of cBW + model selection, cBW alone vs Scheffer et al in Cases of Correct Initial Transition Matrix for Synthetic Data with Posterior-Viterbi Algorithm 43

4.1 Comparisons for the methods of CompHMM, MA, BW, ground truth in different settings 63

5.1 Notations for Inter-Helix Prediction Refinement 70

5.2 Top-L Comparisons of the refinement method (with 2D contact structure model features calculated from ground truth contact map) and DeepHelicon without refinement selection on DATA(101) . . . 82

5.3 Top-L comparisons of the refinement and DeepHelicon without refinement selection on DATA(101) 85

5.4 Top-L comparisons of the refinement results and DeepHelicon with refinement selection (67 out of 101) on DATA(101). 85

5.5 Top-L comparisons of the refinement results and DeepHelicon complement to refinement selection (34 out of 101) on DATA(101) 87

5.6 Comparison of improvements with and without refinement selection scheme 87

A.1 Protein IDs of 360 validate sequences for PD vs none-PD classification - part 1 99

A.2 Protein IDs of 360 validate sequences for PD vs none-PD classification - part 2 100

A.3 Protein IDs of 360 validate sequences for PD vs none-PD classification - part 3 101

B.1 Multi-pathes transmembrane protein dataset (83) 102

C.1 Transmembrane proteins' PDB chain IDs in TRAIN dataset (165) . 103

C.2 Transmembrane Proteins' PDB chain IDs in DATA(101) dataset . . 104

LIST OF FIGURES

1.1 PD served as communication channels cross two cell walls 2

1.2 Topology of PDLPs 3

1.3 Alignments of PDLPs 4

1.4 Sequence comparison of TMD^{PDLPs} in Arabidopsis thaliana 5

1.5 Graphical Representation of HMM 7

1.6 Hidden States' Topology of pHMM 9

2.1 MEME motifs discovered for PDLP JMe regions 12

2.2 3-state PDHMM hidden states connections 17

2.3 Pipeline to combine SVM and HMM 24

2.4 HMM decoding results for PDLP5 and BACK1 25

2.5 ROC for dataset B 25

3.1 Example of constrained Baum-Welch's forward/backward dynamic programming table construction 32

3.2 Comparison results 1 with synthetic data when initialized with fully connected transition matrix 36

3.3 Comparison results 2 with synthetic data when initialized with fully connected transition matrix 37

3.4 Comparison results 3 with synthetic data when initialized with fully connected transition matrix 38

3.5 Multi-pass transmembrane proteins 44

3.6 Topology of 4-state HMM for multi-pass transmembrane prediction 45

3.7 Comparison with real data when initialized with transition matrix of the correct connectivity in Figure 3.6. 46

3.8 Comparison results with real data when initialized with fully connected transition matrix 47

4.1 Log-likelihood track of positive (blue) and negative (black) examples. 60

4.2 Log-likelihood track of the difference between positive and negative examples. 61

4.3 AUC-ROC comparison for CompHMM, BW, MA, and ground truth model 62

4.4 ROC curves of CompHMM, BW, MA for Plasmodesmata-associated protein membership prediction 64

5.1 Diagram of feature extraction. 71

5.2 Fuzziness Score Validation 84

5.3 Pipeline of inter-helix contact prediction refinement. 86

ABSTRACT

With continuous developments in the fields of machine learning and wet-lab technologies for molecular biology, advanced machine learning methods were developed and huge amount of biological data were produced. These developments make applying computational methods on biological problems possible. Great successful applications have been shown in the past decades and lead an important interdisciplinary field named as bioinformatics.

These computational methods assist biologists in dealing with big data, and more importantly, discovering useful knowledge from the biological data. The success of these applications often rely on both the quality and quantity of the data. However, there are increasing situations that neither the quality nor the quantity of the data meets the requirements for traditional machine learning techniques to perform well.

The ongoing research on plasmodesmata-located proteins (PDLPs) is one of the cases when only small number of data is available and little knowledge of the data are known. Plasmodesmata (PD) are membrane-lined intercellular communication channels through which essential nutrients and signaling molecules move between neighboring cells in the plant. This cell-to-cell exchange of molecules through PD is fundamental to the physiology, development and immunity of the plant and is a dynamically regulated cellular process. Once located in PD, biologists believe that PDLPs play their regulatory role via interaction at the transmembrane domain, although currently only very limited experimental data is available about the interaction details. Currently, there are only a limited number of PD-associated proteins have been identified, which prevents any traditional machine learning tools to work efficiently. Moreover, no universal or consensus PD-targeting signal has ever been identified, and the quality of the data is also low, which makes the computational tasks more challenging.

In this dissertation, in order to (i) identify new plasmodesmata targeting proteins, (ii) detect de novo plasmodesmata targeting signals, and (iii) predict PDLP interacting proteins using the transmembrane domain (TMD) as a potential interaction interface, I developed machine learning techniques, which are designed particularly for the PDLP problems with the focus on collaborating biological domain knowledge to mitigate the issues of data quantity and quality.

Chapter 1 introduces the biological problem and computational concepts that related to this dissertation. For the biological problem, the challenges of the research on PDLPs, the current known biological knowledge will be described in details. For the computational concepts, the main focus is the introduction to hidden Markov model (HMM), which is the primary machine learning technique used in my research.

In chapters 2, 3 and 4, I first develop a 3-state HMM named as PDHMM for the task of decoding PD-targeting signals, and by combining PDHMM with a support vector machine to form a decision tree like hybrid classifier for the task of PD-targeting protein predictions. By applying PDHMM and the hybrid classifier on the ongoing research, results of wet-lab experiments have been produced. The majority of predictions have been identified as true positive by wet-lab experiments while some are false positive. Due to the cost of wet-lab experiments, and limitations of both the quantity and quality of the data, enhancing the power of prediction by fully utilizing the wet-lab experimental results is highly desirable in active learning fashion. For the task of detecting de novo PD-targeting signals, based on the standard Baum-Welch algorithm, we develop a novel training algorithm for HMMs when only partially labelled data is available for biological applications. For the task of PD-targeting proteins predictions, algorithm of training HMM with both positive and negative examples is developed. By comparisons with other similar methods, both new developed methods achieve significant improvement cross varies datasets.

In chapter 5, from identifying new PD-targeting proteins and detecting de novo PD-targeting signals, the research focus moved to a more challenging problem: predicting interacting proteins of PDLPs. The research has been concentrated on investigating

inter-helix contact predictions, which are believed to be involved in PD regulation. The developed method leverages the existing state-of-art method's predictions and exploits features that are not fully captured by the existing method with a novel refinement selection scheme. Specifically, with an independent dataset, 2D contact structure models are constructed to extract the features reflecting 2D inter-helix contact patterns, and devise a mechanism to deal with the pitfall of inter-helix prediction refinement, which is an intrinsic challenge to any refinement methods. The cross validation results show that the prediction from the proposed method outperforms the state-of-the-art method by a notable degree even without using the refinement selection scheme. By applying the refinement selection scheme, which selects a subset of sequences to refine, the method outperforms the base method significantly in these selected sequences.

Lastly, future works are discussed in chapter 6 regarding to PD-targeting signals detection for new identified PD-targeting proteins, and future investigation of interactions in PDLPs' transmembrane domains.

Chapter 1

INTRODUCTION

In this chapter, the most related knowledge to this dissertation through chapter 2 to chapter 5 are provided. It contains the introduction to the biological problem of Plasmodesmata and the basic computational concepts of hidden Markov models.

1.1 Plasmodesmata Targeting Proteins, Signals, and Plasmodesmata Interactions

In this section, the basic background of Plasmodesmata will be first introduced. Second, the biological findings related to this dissertation will be described in details. Then, the goals of Plasmodesmata research will be described. Finally, with understanding both the problem and research goals of Plasmodesmata, the challenges of applying computational methods to the Plasmodesmata will be revealed naturally.

Plasmodesmata (PD) are membrane-lined intercellular communication channels through which essential nutrients and signaling molecules move between neighboring cells in the plant as illustrated in Figure 1.1. This cell-to-cell exchange of molecules through PD is fundamental to the physiology, development and immunity of the plant and is a dynamically regulated cellular process. Several types of endogenous proteins, including type-I transmembrane proteins, as well as numerous PD-targeted proteins derived from plant viruses have been identified to associate with PD [43]. However, no universal or consensus PD-targeting signal has ever been discerned nor molecular details are known as to how integral membrane proteins, including one of the best characterized PD-targeted proteins, PDLP5, are targeted to PD-specific membrane domains.

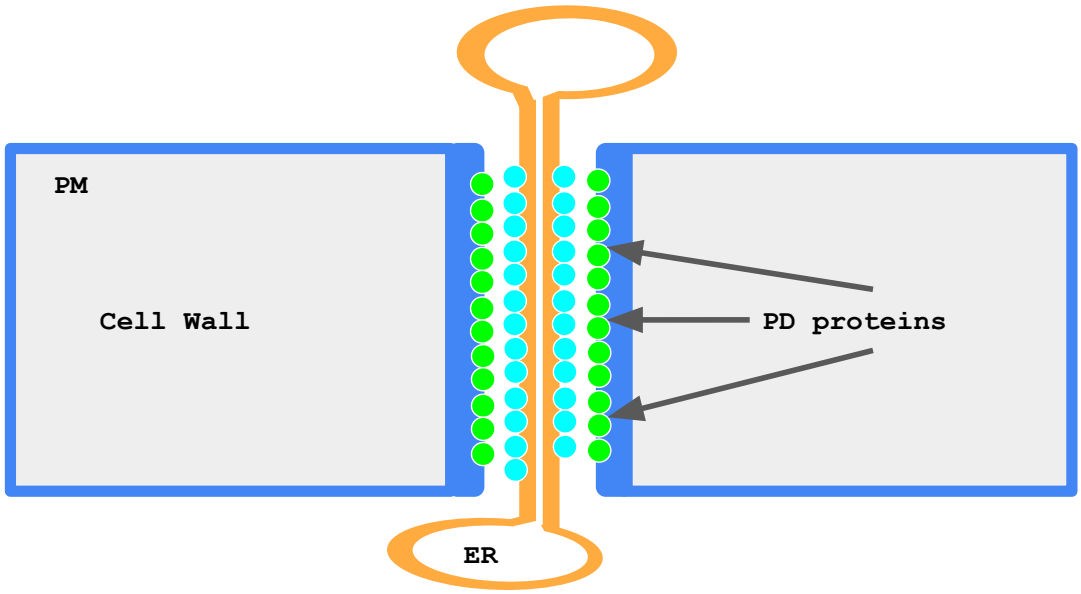

Figure 1.1: PD served as a communication channels cross two cell walls. PM: plasma membrane, ER: endoplasmic reticulum

As the starting of the PD research work, PDLP5 and its seven other family members, all encoded by the Arabidopsis genome, were chosen for investigation. And these PDLPs share a signature topology, as depicted in Figure 1.2. Further experiments (unpublished results from J-Y Lee laboratory) have narrowed PD targeting signals of these eight PDLPs down to a region, called extracellular juxtamembrane domain (JMe), which is located between the DUF26 extracellular domain and the transmembrane domain (TMD). This region spanned about 20-amino acid residues, 9 amino acids downstream of the last conserved Cys residue of the DUF26 domain.

The experimental data (unpublished results from J-Y Lee laboratory) pinpointed that the JMe region of PDLP5 contained a sufficient primary structure for its targeting to PD. Intriguingly, the data also implicated the presence of a second signal outside of the JMe region. However, the multiple sequence alignments of the

Figure 1.2: Topology of PDLPs

eight Arabidopsis PDLP paralogs reveals no hints at the conserved amino acid residues or recognizable patterns shown in Figure 1.3.

In addition to the findings of PD-targeting signals, biological study has shown that the TMD-based dimerization is required for the PD regulating function of PDLP5 [62]. The conserved N-terminal AX_3G motif in PDLP5's TMD region as shown in Figure 1.4, has been verified to be the dimerization motif conferring PD regulation. However, mechanistic details of PD regulation remain unknown. At the current stage, experiments only validated the interaction between PDLP5 and itself and with other PDLP members in their transmembrane domain [62].

The ultimate goal of this plasmodesmata research is to fully understand PD mechanism for biologists. However, this task is impossible with only a limited number of PD targeting proteins, knowing only a few rough regions of PD targeting signals, and little knowledge of protein-protein interactions. Moreover, wet-lab experimental validations require infeasible cost of both time and funding without sufficient information. Therefore, informative computational recommendations for biologists are necessary, and this leads the tasks on computational side:

- Identifying PD targeting proteins.
- Detecting PD targeting signals in PD targeting proteins.
- Predicting the interaction regions of PDLPs and corresponding interacted proteins.

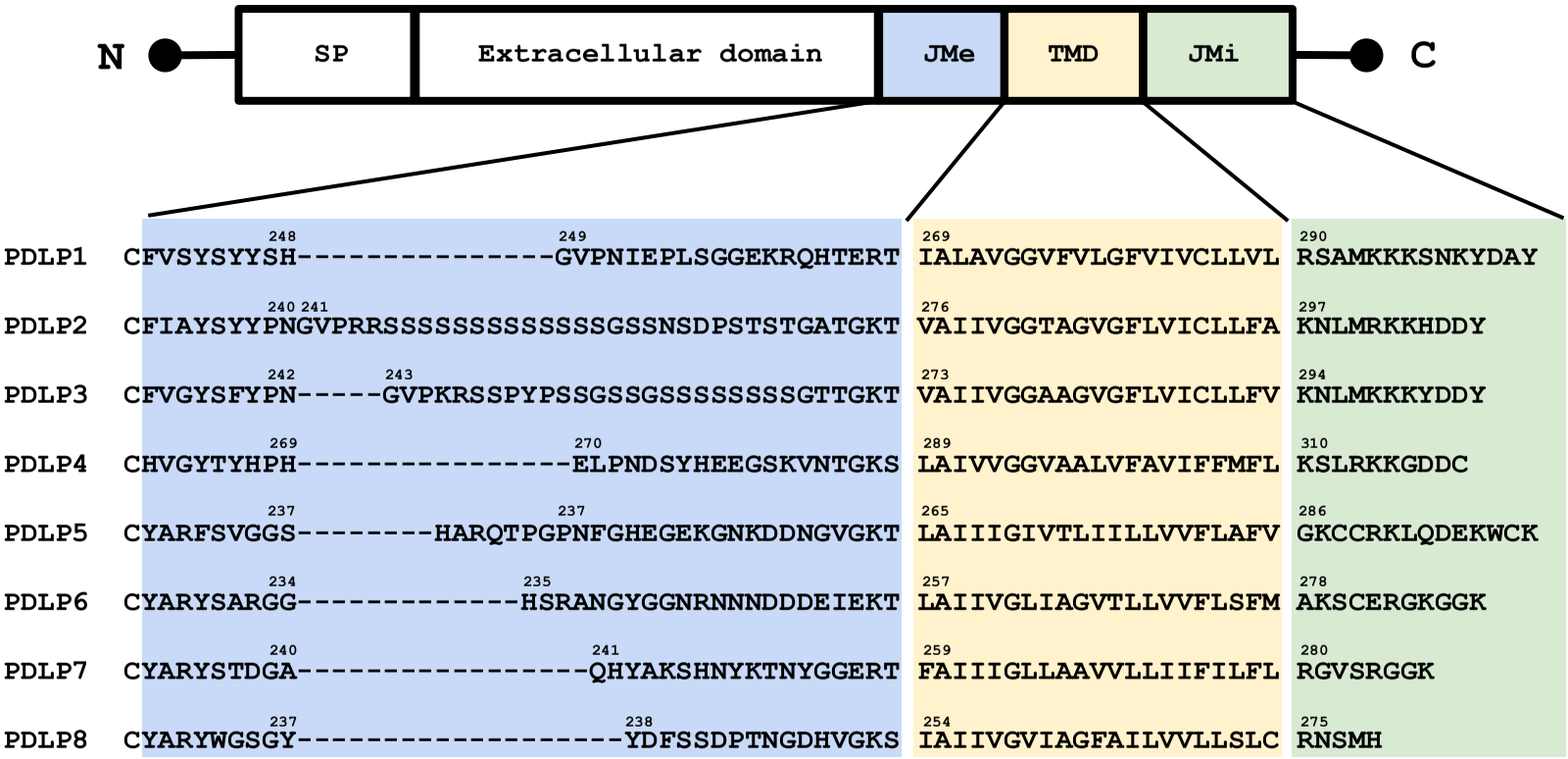

Figure 1.3: Alignments of PDLPs' JMe, TMD, and JMi regions

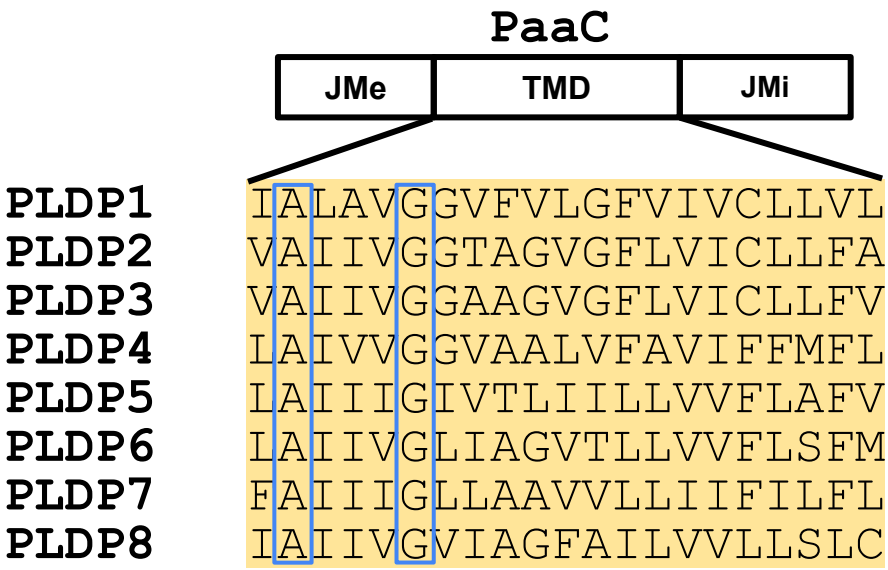

Figure 1.4: A sequence comparison of TMD^{PDLPs} in Arabidopsis thaliana. Blue boxes indicate AX_3G motif.

With these three tasks on computational side, one obvious challenge is lack of enough data, or in other words, the quantity of data is poor: only eight PDLPs are available. The majority of successful machine learning applications rely on large data, and small training prevent any machine learning techniques from learning the true knowledge. More importantly, the quality of data is also important for successful learning. In plasmodesmata problem, prior knowledge is poor and the only knowledge available are:

- PDLPs' JMe regions contain PD targeting signals.
- PDLPs' JMe may contain two independent PD-targeting signals.
- PDLP5's TMD region interacts with itself and other PDLPs for PD regulation.

The information above is important but not enough. Therefore, in this dissertation, in order to overcome the challenges of poor quantity and quality of data, our focus

is designing customized machine learning tools to leverage the known biological domain knowledge, and fully utilize intermediate results from wet-lab experiments to enhance the computational models so that a positive feedback loop between computation and biology can be built.

1.2 Hidden Markov Model

In this dissertation, hidden Markov models are used as primary computational model for both PD-targeting detection, and PD-targeting proteins identification in chapter 2. Moreover, in order to enhance the computational model by utilizing results from wet-lab experiments in active learning fashion, two different hidden Markov models' training algorithms are developed in chapter 3 and chapter 4 to fit different situations. Therefore, an introduction to hidden Markov models is helpful for later references.

1.2.1 Basic Concepts

Hidden Markov model (HMM)[9, 8, 11, 10, 7] is a well known probabilistic model in the field of machine learning. In past decades, HMMs have been successfully applied to many problems in nature language processing and bioinformatics, such as speech recognition [3], biological sequence alignment [25], and CpG island prediction [6].

In general, a HMM consists of a set of states S_i, $i = 1\ to\ N$, and a set of alphabets K that can be emitted from these states with various frequencies; $b_j(k)$ stands for the frequency of letter $k \in K$ being emitted from state S_j. Transitions among states can be depicted as a graph, often referred as model architecture or model structure: each state is represented as a node, and transition from state S_i to state S_j is represented by a directed edge, with a weight a_{ij} being the transition probability. Hereafter, we often refer to a state S_i by its index i. The transition matrix A is an N by N matrix containing all transition probabilities and is a stochastic matrix, namely, each row sums to one: $\Sigma_{j=1}^{N} a_{ij} = 1$. Similar for emission matrix denoted by B, which is an N by K stochastic matrix: $\Sigma_{k=1}^{K} b_j(k) = 1$.

A HMM is often denoted by θ, which stands for collectively all its parameters, namely the emission frequencies $b_j(k)$ and transition probabilities a_{ij}. With transition matrix A and emission matrix B, one can denote a HMM as $\theta = (\pi, A, B)$, where π is called initial probability in dimension of $N \times 1$. Given a sequence of observation O, and its elements $O_t \in K$, where $t = 1..T$, a main assumption of using HMM is that each letter in the sequence is emitted from a state of the model, so correspondingly there is a state sequence, forming a Markov chain, which is hidden from direct observation, hence the name: hidden Markov model. An example of hidden Markov model's graphical representation is demonstrated in Figure 1.5.

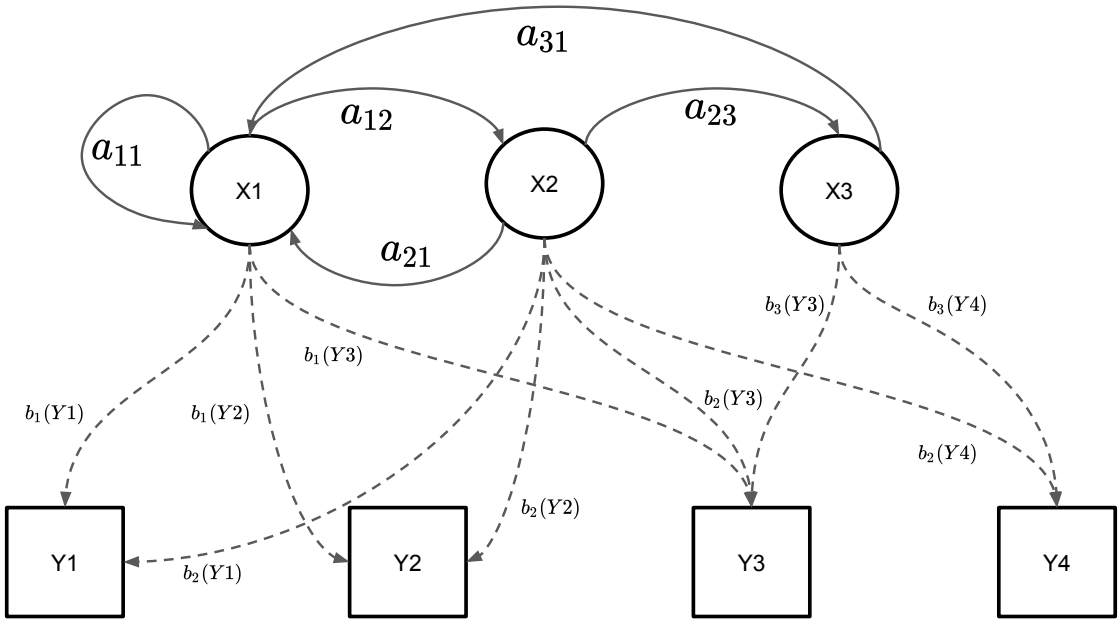

Figure 1.5: Graphical Representation of HMM. 3 hidden states are denoted by circles, 4 alphabets are denoted by squares, transition and emission probabilities are denoted by solid and dash arrows correspondingly.

1.2.2 Tasks of HMM

In machine learning community, HMMs are commonly used for two different tasks. One task is membership prediction: computes the probability of a given sequence belonging to a certain group, which is modelled by the HMM: $P(O|\theta)$. Since this probability is often very small number, people use $log(P(O|\theta))$ to avoid overflow. The other task is called decoding: computes the most probably state sequence (also called hidden path) for the given sequence X^* : $X^* = \text{argmax}_X \, Pr(O, X|\theta)$.

In the field of nature language processing, HMMs are mainly used for decoding. One example is speech recognition: translate human voice (observable sequence) to plain text (hidden path). In the field of bioinformatics, HMMs have been used for both membership prediction and decoding. One successful example is HMMER [25], which is a special case of HMM called profile HMM (pHMM). It is different from general HMM in hidden states' topology as shown in Figure 1.6. HMMER is used for two purposes: sequence alignment and remotely homologous detection. Sequence alignment task uses HMMER for decoding: HMMER computes the insertion, deletion, or mutation state (hidden path) for the given sequence (observable sequence) position wise. Remote homologous detection task uses HMMER for membership prediction: HMMER computes the likelihood score of a sequence belonging to certain protein family.

Both tasks are well studied and straightforward. The membership prediction task, or task of calculating $log(P(O|\theta))$ is done by forward backward procedure. The decoding task is solved by Viterbi algorithm [60] efficiently. In this dissertation, when applying HMMs for plasmodesmata problem, membership prediction task is used for identifying PD targeting proteins, decoding task is used for detecting PD targeting signals in PD targeting proteins.

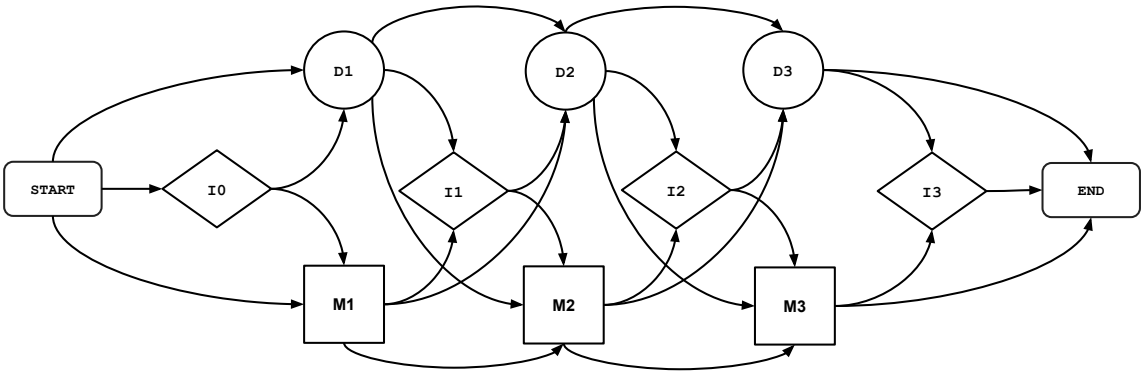

Figure 1.6: Hidden States' Topology of pHMM. Match, deletion, and insertion states are denoted by square, circle, diamond shapes correspondingly.

1.2.3 Training Methods of HMM

Comparing with membership prediction and decoding tasks of HMMs, training HMM is considered as a more challenging problem, and has different methods in different situations. The training of HMM can be described as adjusting model parameters θ to maximize the log-likelihood $\sum_{s=1}^{m} log(Pr(O^s|\theta))$ of observing the given training sequences O^s, where $s = 1...m$ [51].

The major training algorithms of HMM are the following three in general: maximum likelihood, Baum-Welch algorithm, and Viterbi training [34]. Maximum likelihood is used when label information is fully available, and it returns the optimal solution. The term "label" used here can be understood as the hidden path X^* in last section. The latter two algorithms are used when no label information is available. In such situation, both Baum-Welch algorithm and Viterbi training can only provide a local optimal solution. Interested readers can find a gentle introduction and tutorial for hidden Markov models in [51]. In order to enhance the modelling power of HMMs in plasmodesmata problem, we mainly worked on modifying HMM training algorithm

in other situations: only partially labelled data is available (in chapter 3) , utilizing both positive and negative data (in chapter 4). A general HMM notations' description is shown in Table 1.1 for later references.

Table 1.1: Notations for hidden Markov models

Symbols	Explanations
θ	Hidden Markov model: $\theta = (\pi, A, B)$
N	States' number in hidden Markov model.
K	Symbolic Number in hidden Markov model.
A	Transition matrix with dimension $N \times N$.
a_{ij}	Probability of state i transition to state j.
B	Emission matrix with dimension $N \times K$.
$b_j(k)$	Probability of state j emitted from symbol k.
π	Initial probability of states with dimension $N \times 1$.
O^s	The s^{th} sequence with length T^s
X^s	State sequence of O^s
m	Total number of sequences

Chapter 2

DETECTING DE NOVO PLASMODESMATA TARGETING SIGNALS AND IDENTIFYING PD TARGETING PROTEINS

2.1 Introduction

With the background knowledge introduced in chapter 1, the initial work to attempt both problems of PD-targeting signals detection and PD targeting proteins identification is demonstrated in this chapter.

It is well known that proteins after being synthesized have to be transported to their designated cellular location in order to fulfill the biological functions. However, much detail of the transporting mechanisms remain unknown, and subcellular localization prediction is an active research area in bioinformatics [1, 17, 22]. As mentioned in chapter 1, one of the computational task is PD-targeting proteins identification. And this task can be directly transferred to cellular localization prediction task. However, since subcellular localization of Plasmodesmata is not well studied and there is no such existing work for PD localization, machine learning tool of PD localization prediction needs to be done by our own.

In this work, I set out to develop computational approaches to:

- Detect subtle patterns that are associated with PD targeting.
- Identify unknown PD-targeted proteins in Arabidopsis and other species

As mentioned, the second task can be considered as a classification problem, like other subcellular localization prediction problems. Support Vector Machine (SVM) [19] is adopted as a classifier, and dipeptide features is used to characterize proteins sequences. The performance of this straightforward approach is surprisingly good in cross validation experiments.

The first task of detecting PD targeting signal(s) turns out to be more challenging. Using the TMMOD tool [36] with customized training and feature selection, some signals are detected at the vicinity of the last conserved Cys of the DUF in addition to the initially defined JMe region. As for PDLP5, this prediction was consistent with the presence of a second signal. Further phylogenetic analysis of orthologues of PDLPs from other plant species and MEME motif finding [4] also identified similar but slightly stronger patterns in the same location for a subset of PDLPs (consisting of PDLP1 to PDLP4). The MEME motif analysis results are shown in Figure 2.1.

Figure 2.1: MEME motifs discovered for PDLP JMe regions. The red region and green region correspond to α state, β state in PDHMM correspondingly.

These findings from the computational analyses together with the preliminary experimental data for PDLP5 prompted us to hypothesize that: there is the second PD-targeting signal outside of the JMe region might reside at the C-terminal end of the DUF domain. Unlike TMD, which has a clear-cut boundary at both ends from computational predictions, the C-terminal end of the DUF domain is not experimentally defined. This is why the previous wet-lab experiments by J-Y Lee lab have limited JMe as 20 AA adjacent to the N-terminal side of TMD, excluding 9 AA to avoid overstepping into the DUF domain. Based on the new hypothesis that this region likely contains a secondary signal for PD targeting, the JMe is newly defined as the 30-AA

region located between the rightmost conserved Cys at the very C-terminal end of the DUF domain and the N-terminal end of TMD.

Based on the aforementioned hypothesis, a hidden Markov model (HMM) [7, 8, 9, 10, 11] called PDHMM is built to capture the functional structure of the JMe. The model has three states: state α for the left PD signal, state β for the right signal, and state γ for the non-functioning linkers. Using the trained hidden Markov model, the JMe region of the eight PDLPs can be decoded, and the follow-up experiments have verified the two PD signals and their relative positions in PDLP1, PDLP3, PDLP5 and PDLP8, as predicted by the model (unpublished data by J-Y Lee lab). Ongoing experiments are being conducted to verify predicted PD targeting signals for the remaining four PDLP proteins. Furthermore, the model was tested with predicting potential PDLPs in a dataset containing 360 type-I transmembrane proteins, and showed remarkable performance as measured as ROC score in cross validation.

Given the fact that PD targeting signals reside in JMe, a pitfall with the SVM classifier is incidentally discovered. When testing with randomized JMe to establish a baseline. SVM mistakenly classified these synthetic sequences – which are the same of the real PDLPs except for the JMe being randomized. To mitigate this issue, we propose a way to combine SVM and HMM to further improve the classification performance.

The rest of this chapter is organized as follows. In section 2.2, the structure of the HMM, its training, and integration with SVM are described in details. In section 2.3, the results on testing the hidden Markov model and SVM alone and the two methods in tandem are presented correspondingly. Conclusions are presented in the last section.

2.2 Methods

As mentioned in section 2.1, the research focuses are detecting PD targing signals in the sequence of PDLPs and identifying potential novel PDLPs in Arabidopsis and other plant species. In this section, the computational methods developed for these

two tasks will be described in details. The proposed methods are based on two powerful machine learning methods – SVM and HMM. With the understanding the issues of using either SVM or HMM alone, a way of combining these two learning algorithms are described for better classification performance.

2.2.1 SVM with Dipepetide Features

SVM are a type of classifiers that take vectorized inputs and find the optimal separating hyperplane in the vector space with the positive training examples on one side of the hyperplane and negative training examples on the other side [20]. The optimization is to 1) maximize the margin between the hyperplane to the support vectors, namely, these training examples that are closest to the hyperplane, and 2) minimize the penalty incurred from misclassification. For data that are not linearly separable, the kernel technique can be used, which maps the input to a higher dimension space (called feature space), where the data become linearly separable. Once trained, an unseen data point can be mapped to the input space or the feature space, and based on its relative position to the hyperplane, its classification can be correspondingly made: positive if on the positive example side; negative if otherwise.

SVM have been successfully applied to many classification tasks in bioinformatics, including cellular localization prediction, such as MultiLoc2 [15], SlocX [52], APSLAP [53]. For this task, SVM with a linear kernel is adopted. Since the multiple sequence alignment (see Figure 1.3) does not show clear high conservation patterns, alignment-free features is chosen to characterize the proteins for classification, specifically the dipeptide features [12]. In this work, the choice of dipeptide feature is a result of balancing the number of features and the number of training examples: dipeptide features capture more information than single amino acid composition and require less examples than tripeptide features and higher order k-mers, which have a dimension at 8000 and higher and require significantly more training examples.

The dipeptide features are the occurrence frequencies of all 400 possible amino acid pairs in a given protein sequence. Let $x_{dipeptide}$ denote the dipeptide feature, then

$x_{dipeptide}(i,j)$ is the frequency of ith and jth amino acid as a neighboring pair to appear in the protein sequence. It is calculated as following:

$$x_{dipeptide}(i,j) = \frac{count([aa_i, aa_j])}{\sum_{i=1}^{20}\sum_{j=1}^{20} count([aa_i, aa_j])} \tag{2.1}$$

As a common practice, the dipeptide features are normalized as follows:

$$x_{dipeptide-norm} = \frac{x_{dipeptide} - \overline{x_{dipeptide}}}{std(x_{dipeptide})} \tag{2.2}$$

Although SVM with dipeptide features can be a powerful classifier, as shown in literature [15, 52, 53], it is worth noting that dipeptide occurrence frequency captures global features of the sequence as a whole and is hence not suitable for picking up subtle features from within short regions that are nevertheless important to the protein's functions. And this issue is exacerbated with insufficient amount of positive training data. In such cases, many dipeptide pairs may have zero counts in Eq. 2.2.1 – are these zeros real or will they become non zero should enough training examples be available? Consequently, SVM trained on these dipeptide features can be susceptible to overfitting and thus does not generalize well on unseen data.

In this work, only eight PDLPs are available and the JMe region that is known to contain PD-targeting signal is a very short region (around 30 amino acids long) as compared with the whole sequence (up to 700 amino acids long). As such, special attention should be paid to alleviate the aforementioned issues of training SVM with dipeptide features, for otherwise it may give rise to false positive predictions for the sequences that have similar dipeptide features but do not contain PD-targeting signals. Note that these issues are not unique only for SVM but for any classifiers that rely on features from full length sequences. For comparison, random forest classifier is also

trained on dipeptide features, and the performances between the two classifiers are comparable, with SVM being slightly better.

2.2.2 3-States HMM on JMe: PDHMM

In contrast to the dipeptide approach of capturing more global information, the hidden Markov model designed here is focused on JMe region in order to detect PD-targeting signals that the web-lab mutagenesis experiments have suggested.

Based on the signature topology as shown in Figure 1.2, the JMe region of a PDLP, or a potential PDLP such as type I-transmembrane protein, can be easily extracted by two steps procedure:

1. Finding protein's transmembrane region via various sophisticated tools such as TMDOCK [47] and TMMOD [36].

2. Extracting 30 amino acids upstream of the transmembrane domain in step 1.

The hidden Markov model has the 3 states, denoted as α, β, and γ. State α stands for PD-targeting signal A, and state β stands for PD-targeting signal B. State γ stands for everything else in the JMe region but the PD-targeting signals. The hidden states transition connections of these 3 states are shown in Figure 2.2. The direct edge between state α and state β is to allow the case in which there are no linker residues between the two PD signals. Since it is not known for any given residue, which of the three states it is – in other words, the training data are unlabeled – the HMM cannot be trained with counting as in a typical maximum likelihood approach, or with a multiple sequence alignment as in typical profile hidden Markov models for protein family classification. Instead, Baum-Welch algorithm is adopted, which is an expectation maximization approach and does not require the hidden states to be labeled in the training data [10].

After the model is trained, it is used for two tasks:

1. Classifying PDLPs from a set of 360 type I proteins in Arabidopsis thaliana.

2. Decoding the JMe region, i.e., marking out each residue as α, β or γ state.

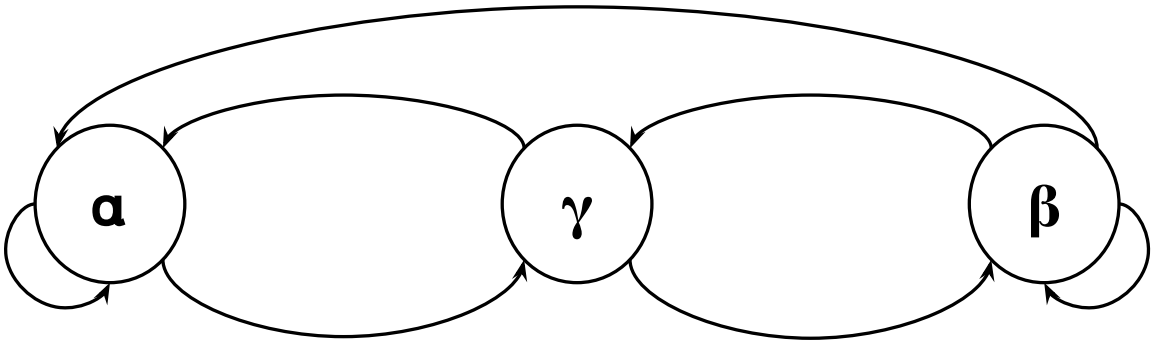

Figure 2.2: 3-state PDHMM hidden states connections

It is worth to note that, because the 3-state HMM focuses on modeling with JMe, the model loses the global picture of PDLP sequences. For example, not every protein sequence with valid JMe region can be considered as PD-targeted proteins. Furthermore, because HMM is not a discriminative model, the boundary between predicted positive and predicted negative is not readily given by the model and hence can be fuzzy. In the next subsection, we proposed a way to combine SVM and HMM, where a mechanism to impose a threshold on HMM prediction score is devised.

2.2.3 Combination of SVM with HMM

From the two previous subsections, it can be noticed that SVM is good at capturing characteristic amino acid composition for full length sequences with dipeptide features, but less effective for short sequences like JMe region. This can potentially lead to false positive in predicting some protein sequences as PD targeting even though these proteins do not contain PD-JMe region. The issue becomes more apparent when testing the trained SVM to make de novo prediction of PD-targeted proteins in large dataset, in which more proteins may contain dipeptide features or even overall topology

similar to that of the real PDLPs. On the other hand, because the HMM focuses on the JMe region exclusively, it loses other features of PDLPs, e.g., it is found experimentally that a mutant without a proper JMi can not even be synthesized.

As such, the shortcomings of SVM and HMM both can give false positive predictions, but for different situations. The SVM tends to give false positive predictions for the sequences with none-PD JMe and PD like dipeptide feature, whereas HMM tends to give false positive predictions for the sequences with PD-JMe but none-PD's other domains because HMM only focuses on JMe region. Therefore, it is sensible to combine SVM and HMM in a complementary way that can overcome their shortcomings with their advantages. An option is to use decision tree on the predictions from SVM and HMM alone to choose a better one. However, due to the limited number of data samples, training decision tree by traditional method did not work. Fortunately, with the understanding of SVM and HMM in this particular task, a simple decision tree is proposed and the decision boundary for each node can be calculated properly, without use of a large training set.

The structure of combining SVM and HMM through decision tree to make final prediction is shown in Figure 2.3. The node of SVM's decision boundary is naturally defined by the support vectors in the trained SVM. The decision boundary of HMM node is slightly tricky to define. Since we have the trained HMM model, synthetic JMe sequences can be generated. Given a large number of synthetic JMe sequences, their log-likelihoods follow a Gaussian distribution. By traditional statistic convention, 95% confidence interval is used, and the lower bound of the interval is picked as the decision boundary of HMM node.

2.3 Results

2.3.1 Datasets

Dataset A contains 360 type-I membrane proteins in Arabidopsis thaliana, including the 8 PDLP sequences, all retrieved from Uniport [18]. These protein names are also recorded in Appendix A for readers' reference. From Uniport, labels of protein

Table 2.1: Different types of proteins in dataset A

protein localization	endoplasmic reticulum	Plasma	PDLP	Vacuole	Vesicle
number of proteins	17	322	8	7	6

cellular localizations and their transmembrane domain are extracted, which account for 7 different types of membrane proteins, including endoplasmic reticulum membrane, Endosome membrane, Golgi membrane, Plasma membrane, Vacuole membrane, Vesicle membrane, and the PD membrane. In this work, for our purpose, the eight PDLPs are grouped into the other class (PD) and the rest are lumped into one class (none-PD).

In order to train the HMM for the JMe region, a procedure described in section 2.2 is applied to all sequences in the dataset to extract a valide JMe region defined as 30 amino acids upstream of the transmembrane domain identified by TMMOD. This procedure eliminates Endosome and Golgi membrane proteins from the dataset, as their JMe region is shorter than 30 amino acids. Table 2.1 lists each cellular localizations and the number of proteins in dataset A.

To test the robustness of SVM and HMM in handling false positives, as described in the section 2.2, eight synthetic sequences, which are the 8 PDLP sequences but with the original JMe region being replaced with random residues, are added to the dataset. Since JMe region contains PD-targeting signal, it is almost certain that randomly replacing JMe region will lead to none-PD proteins. In other words, these eight synthetic sequences are negative data. Dataset A plus these eight synthetic sequences give rise to dataset B. Note that these eight synthetic sequences will be only used for testing.

As there are only 8 PDLP sequences, the leave-one-out (or equivalently 8 fold) cross validation scheme is adopted to ensure the maximum possible number of training examples to train the models. Specifically, each one of the 8 PDLP sequences is reserved as a positive test example once, and the remaining 7 PDLPs are used as positive training

examples. The 352 none-PD sequences, as the negative examples, are randomly split into 8 subsets of equal size (44 sequences). One negative subset is picked to combine with the positive testing example to form the test set (45 sequences); and the remaining 7 negative subsets are merged together to form the negative training set. Note that, for the HMM, no negative training examples are needed. When dataset B is used, the whole process is the same, except that the synthetic none-PD sequences are repeatedly used as testing data for each fold.

2.3.2 Performance Metrics

To test the trained hidden Markov model θ, the test examples from the 8 fold cross validation are combined and ranked by their prediction score $P(x|\theta)$, which is the likelihood for sequence x to be emitted from the model, calculated from the Forward algorithm. If a threshold is set for the prediction score, test examples with score above the threshold are classified as positive – they are true positive (TP) if their ground truth label is positive; they are false positive (FP) if otherwise. Similarly, test examples with score below the threshold are classified as negative – they are true negative (TN) if their ground truth label is negative; they are false negative (FN) if otherwise. We use receiver operating characteristic (ROC) curve and ROC score, which is the area under the ROC curve, to evaluate the performance. The ROC curve plots the true positive rate against the false positive rate at the threshold sliding down the ranked list of test examples [27]. ROC curve starts (0,0) and goes to (1,1) in a monotically manner. The perfect classifier has ROC score 1.0, and a random classifier has ROC score 0.5. When a natural choice of threshold is available, like the distance to the the separating hyperplane in SVM, we also use the precision and recall associated with that threshold to evaluate the performance.

2.3.3 Evaluation: SVM Alone

In this experiment, SVM is trained and tested with linear kernel on dipeptide features extracted from full length sequences. The ROC score from 8 fold cross validation is 1.0 for dataset A but drops to 0.8837 for dataset B, because of misclassifying the synthetic none-PD as PD, which confirms the concern that SVM alone can be susceptible to the overfitting issue. As comparison, the performance from a RF classifier is: ROC score = 0.9984 for dataset A and ROC score = 0.9177 for dataset B.

2.3.4 Evaluation: HMM Alone

In this experiment, the 3-state HMM, PDHMM is trained as described in the section 2.2. The trained PDHMM is then tested with 8 fold cross validation. The ROC score is 0.93 for dataset A and is 0.9408 for dataset B. Unlike SVM, HMM's performance remains about the same for both datasets, confirming that HMM is more robust with false positives.

For the decoding task, the standard Viterbi algorithm [30] is used to scan the sequence against the model, trained with PDLP5 and ten of its orthologues, to annotate which residues belong to which of the three states, α, β or γ. So far, there is no ground truth available yet to directly evaluate the triple state annotation within JMe, except for PDLP5 and BAK1, the latter of which is experimentally confirmed as non PD targeting, see Figure 2.4. It is very encouraging that the experiments for PDLP5 validated the existence of two PD targeting signals and their delineation is consistent with the annotation made by the model. More web-lab experiments are planned to validate model annotation of other PDLP paralogs.

2.3.5 Evaluation: Combination of SVM and HMM

From the experimental results in subsection 2.3.3 and 2.3.4, it is clear that SVM is susceptible to the pitfall of misclassifying the synthetic none-PD, whereas HMM is not affected. Also, by comparing the ROC score of SVM and HMM in dataset A, SVM has better performance and it naturally gives a clear decision boundary. In this

Table 2.2: Comparison for all method in dataset B

Method	ROC score	precision	recall
SVM with dipeptide feature	0.8837	0.1014	0.8750
HMM with JMe residues	0.9408	not applicable	not applicable
Combining SVM with HMM	0.9997	1.0000	0.8750

experiment, we only focus on dataset B to show that by combining SVM with HMM via decision tree, performance can be improved, as compared to either SVM or HMM alone.

Figure 2.5 shows the ROC curve for the method of combining SVM with HMM (red) and the comparison with HMM (green) and SVM (blue) alone. The ROC score of combining SVM with HMM is 0.9997. Moreover, the precision/recall and ROC score comparison for all the three method for dataset B are shown in Table 2.2.

2.4 Discussion and Conclusion

This chapter presented computational approaches based on machine learning techniques to solve a very challenging biological problem: detecting plasmodesmata targeting signals and identifying novel PDLP sequences. The challenges arise from lack of clear sequence patterns, insufficient amount data, and unbalanced dataset. Without addressing these challenges, a straightforward application of standard machine learning techniques can lead to unreliable prediction, as demonstrated with using SVM on dipeptide features. In order to overcome these challenges, domain specific knowledge is closely incorporated into hidden Markov model design, and devised a pipeline to leverage the predictive power of different models to reduce false positives. As a result, we are able to detect de novo PD targeting signals, verified by wet-lab experiments, and to classify PDLPs with remarkably high accuracy.

It is worth noting that, in this study, some common practices is adopted to avoid overfitting, such as the multi-fold cross validation scheme and use of a simple linear kernel versus a more powerful kernel in SVM. While the performance from cross

validation as compared with training error does not indicate overfitting, given the small positive training examples in this study, it is difficult to know how well the trained classifiers will generalize to a large data set or data from different genomes, especially in detecting de novo PD proteins, which are actually being investigated in the web-lab experiments. On the decoding task with PDHMM, half of the predicted PD targeting signals have already been verified to be correct in the web-lab experiments. Moreover, an online server is deployed based the methods in this chapter to assist biologists discovering new PDLP members. With new discovered PDLP members, an improved classifier can be built, which leads a positive feedback cycle of PDLP prediction and new PDLP members discovery.

With the work done in this chapter, we are able to predict PD targeting signals among PDLPs, and PD targeting proteins candidates. Such prediction results were evaluated via wet-lab experiments, and the results of evaluations are encouraging. However, the success is not enough due to the cost of wet-lab experiments and limited data. It is urgent to improve current machine learning tools. One way is retrain machine learning models with new results in an active learning fashion. Therefore, in order to fully utilize the results gained so far, in chapter 3, training HMM with partial label data is developed to utilize wet lab experimental results of PD targeting signals, and training HMM with both positive and negative data is developed to utilize negative PD targeting proteins in wet-lab experiments, which is shown in chapter 4.

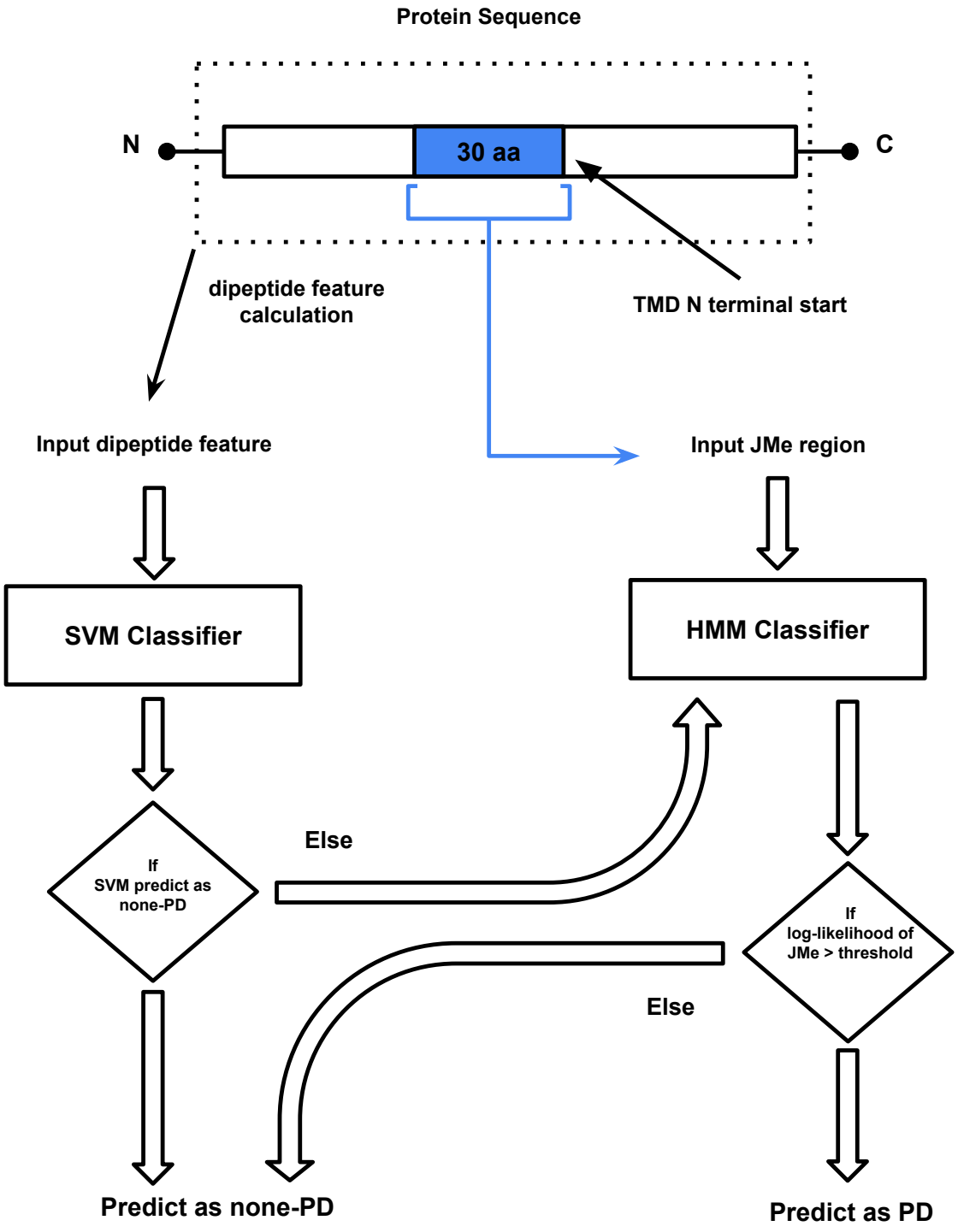

Figure 2.3: Pipeline to combine SVM and HMM.

PDLP5 JMe sequence:	YARFSVGGSHARQTPGPNFGHEGEKGNKDDNGVGKT
PDLP5 HMM decoding:	αααααααααααγγγγγγγγγββββββββββββββ
BAK1 JMe sequence:	NTKLTPLPASPPPPISPTPPSPAGSNRITGAI
BAK1 HMM decoding:	γγγγγγγγαγγγγγγγγγγγγγαγγγγγγγαγ

Figure 2.4: HMM decoding results for PDLP5 and BAK1. Red color and green color in the sequence represent state α, state β regions correspondingly. The region without color refers to state γ.

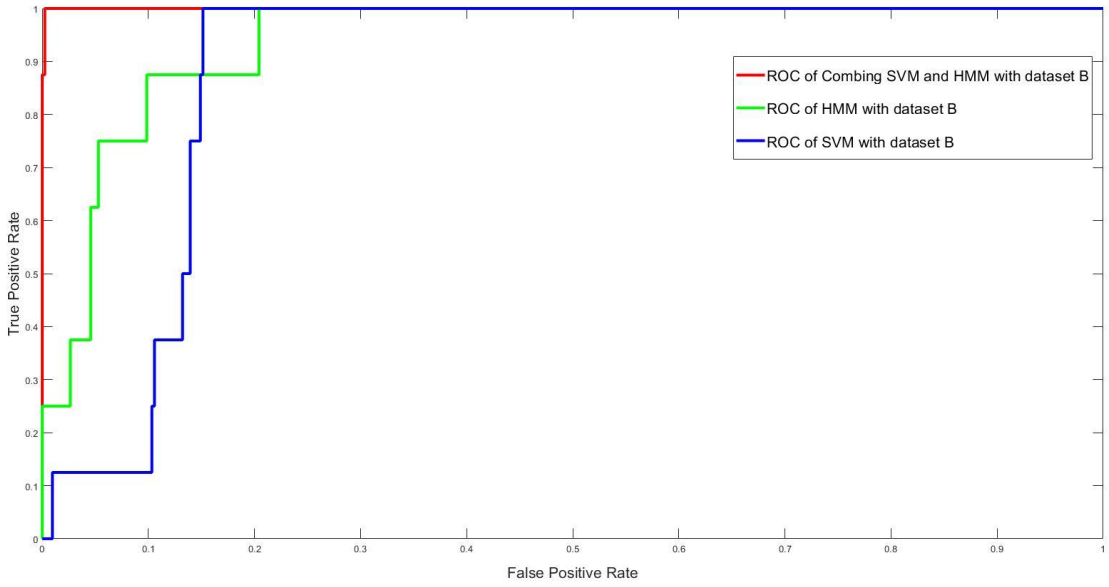

Figure 2.5: ROC for combining SVM and HMM with dataset B (red), with ROC score = 0.9997, and comparing with using HMM alone (green), SVM alone (blue)

Chapter 3

TRAINING HIDDEN MARKOV MODELS WITH PARTIAL LABEL DATA

3.1 Introduction

Hidden Markov model [9, 8, 11, 10, 7] is a well known probabilistic model in the field of machine learning, suitable for detecting patterns in sequential data, such as plain texts [65, 32, 39], biological sequences [14, 63], and time series data in the stock market [28, 26]. For all these applications, successful learning and modeling depends, to a large degree, on the amount and, more importantly, the quality of the data. In text mining problem, though the data amount is huge, careful labelling tasks consume massive human labor [54]. In biological sequence analysis, discovering de novo signal remains challenging because a precise full labeling via wet-lab experiments demand even more resources and time, and hence it is considered unfeasible in general. Therefore, research is necessary in data handling with different labelling quality for applied machine learning community. In this chapter, the focus is designing a Baum-Welch-algorithm based learning method for HMMs to handle the biological problems when only partial labeling is available in training data.

This work is inspired by research on detecting de novo plasmodesmata targeting signals in Arabidopsis Plasmodesmata-located proteins (PDLPs) as shown in last chapter. PDLPs are type I-transmembrane proteins, which are targeted to intercellular pores called plasmodesmata that form at the cellular junctions in plants [42]. In last chapter's study, by building a 3-state HMM named as PDHMM, the presence of two different plasmodesmata targeting signals (named as α and β) were predicted in the juxta membrane region of PDLPs. While all the predicted signals were successfully verified in wet-lab experiments so far, some predicted signals contain residues that do

not conform to the true signal. Because both the cost and time are high for wet-lab experiments, an improved HMM would be highly desirable. However, due to the limitation in the number of the training examples – Arabidopsis genome encodes only eight PDLP members, further improvements of the model can be hardly achieved. It would require to fully utilize the current wet-lab experimental results to train the model, i.e., by labeling the residues that have been already shown to be either part of the signals or not part of the signals, given that labels are not available for all the residues due to limited experimental results.

In a related work by Tamposis et al, a semi-supervised approach is developed to handle a mixture of training sequences that contains a subset of fully labelled sequences, with the remaining sequences having no labels at all or partial labels [58]. Their method uses the fully labeled sequences to train the parameters for HMMs and then use Viterbi algorithm to predict the missing labels followed by training the model again with the predicted labels. This process is iterated until a convergence condition is met. Instead, this work is specifically interested in situations where no fully labeled sequences are available and often the partial labeling is also sparse. In the text mining field, HMM training algorithm of handling partial label was developed especially for active learning purposes and designed to fit into text mining special situation: no label scenario, or in other words, no meaningful label can be assigned [54]. However the unit of observation in text mining and information retrieval is a word, instead of a single letter, corresponding to individual amino acid residue as in biological sequences. So, in order to deal with the partial labeling aforementioned, a novel Baum-Welch based HMM training algorithm is designed here to leverage partial label information with techniques of model selection through partial labels. Besides the difference in the observation unit, the proposed algorithm differs from [54] primarily in how to calculate the expected value for a given partial label at a given position: the proposed method sums over hidden state paths that must be subject to constraints anywhere given partial labels in the training sequence. In contrast, in [54] the expected value for a given partial label at a given position is calculated by summing over paths that are

only constrained at the position being considered, and anywhere else in the sequence the hidden paths are free to go through all possible states (labels) even at positions where partial labels are given. Moreover, this difference affects how the expected value for a transition is calculated, regardless whether the transition happens to involve one partial label, two partial labels, or no partial labels at all. The comparison between the proposed method and the method described in [54] showed that the proposed method outperformed in both synthetic and real data for decoding task in biological problems.

The rest content of this chapter is organized as follows. First, the relevant background knowledge of HMM is briefly reviewed. Then, the proposed method of training HMM when only partial label sequences are available is described in details. This is followed with experiments and results to examine and demonstrate the modelling power of the novel algorithm. Discussion and conclusion are given at the end.

3.2 Method

As introduced in previous section, when no labels are available, Baum-Welch algorithm is typically used to train HMM and Viterbi training is sometimes used for speed and simplicity; when all label information is given, training HMM is straight forward by maximum likelihood approach. Currently, training HMM with partial label is mainly studied in the field of text mining, with a particular focus on active learning problems, such as the work done in [54], which is compared with the proposed method in section 3.3.

The proposed method is a novel approach to this partial label training problem with modification of Baum-Welch algorithm (called constrained Baum-Welch algorithm) and a model selection technique, which helps the algorithm leverage available information and improve the training and performance in decoding task. In the next two subsections, constrained Baum-Welch algorithm, the model selection method, and how to combine the two for model training will discussed in details respectively.

3.2.1 Constrained Baum-Welch Algorithm

The standard Baum-Welch algorithm is an Expectation-Maximization approach to maximizing likelihood when the system contains latent variables, which are the state sequences for hidden Markov models when training sequences are not labeled. The constrained Baum-Welch algorithm (cBW) is similar to the standard Baum-Welch algorithm except that the training sequences are partially labeled, which imposes the constraints on the possible hidden state paths in calculating the expectation. Standard Baum-Welch algorithm is divided into E-step and M-step. cBW's M-step is identical to standard Baum-Welch's. The difference is the E-step, computing forward and backward matrices. The formulas of computing α, β, γ are shown as following respectively.

Given the model $\theta = (\pi, A, B)$. Then, the initial values of forward matrix α for one given training sequence $O = (O_1, ..., O_T)$ is computed as follows.

$$\alpha_i(1) = \pi b_i(O_1) \tag{3.1}$$

After calculating the initial values of α, by dynamic programming, the remaining values at any position for any state are calculated recursively by summing over the possible state paths $X = (X_1, ..., X_T)$, allowed by the model, that lead to the point whose α value is being calculated. However, since we now have partial labels for the training sequence O, care must be taken to satisfy the constraints at each position O_t imposed by the partial label, $L(O_t) \in S \cup \{0\}$, where a value zero means no label available. Specifically,

$$\alpha_i(t+1) = \begin{cases} b_i(O_{t+1}) \sum_{j=1}^{N} \alpha_j(t) a_{ji}, & \text{if } L(O_{t+1}) = 0 \text{ or } i \\ 0 & \text{if } L(O_{t+1}) \neq 0 \text{ and } L(O_{t+1}) \neq i \end{cases} \tag{3.2}$$

In the above equation, the first case is when position O_{t+1} is either unconstrained (0) or constrained to be state i by the partial label. In such a case, the α value is computed in the same way as the standard Baum-Welch algorithm, though the actual value can still be affected by partial labels at earlier positions via recursion. The second case is

when the position t+1 is constrained by the partial label to be a state other than i. In this case, $\alpha_i(t+1) = 0$. This latter case is what makes the algorithm different from the standard Baum-Welch algorithm in order to "honor" the partial labels.

The backward matrix β is initialized as the following.

$$\beta_i(T) = 1 \tag{3.3}$$

Then, similarly, a recursive procedure is applied for the remaining of backward matrix.

$$\beta_i(t) = \begin{cases} \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(O(t+1)), & \text{if } L(O_t) = 0 \text{ or } i \\ 0 & \text{if } L(O_t) \neq 0 \text{ and } L(O_t) \neq i \end{cases} \tag{3.4}$$

Note that, while the α is calculated the same way as the modified Forward algorithm in [2] but the β is calculated differently from their modified Backward algorithm. After the calculations of α and β, then we can calculate γ variable, where $\gamma_i(t)$ is the probability of observing the training sequence O from all possible state paths that are allowed by hidden Markov model θ as constrained by the partial labels and go through state i at position t. $\gamma_i(t)$ is computed as follows.

$$\begin{aligned} \gamma_i(t) = P(X(t) = i|\theta, O) &= \frac{P(X(t) = i, O|\theta)}{P(O|\theta)} \\ &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)} \end{aligned} \tag{3.5}$$

where the last equal sign holds because $P(O|\theta) = \sum_{j=1}^{N} \alpha_j(t)\beta_j(t)$. The next step is to compute $\xi_{ij}(t)$, which is the probability of of observing the training sequence O from all possible state paths that are allowed by hidden Markov model θ as constrained by the partial labels and go through state i at positive t and transition to state j at position $t+1$:

$$\begin{aligned} \xi_{ij}(t) &= \frac{P(X(t) = i, X(t+1) = j, O|\theta)}{P(O|\theta)} \\ &= \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(O(t+1))}{P(O|\theta)} \end{aligned} \tag{3.6}$$

Finally, with γ, ξ, and the initial probability π^* the M-step is to update the transition matrix A^* emission matrix B^* can be computed as follows.

$$\pi(i)^* = \gamma_i(1) \tag{3.7}$$

$$a^*_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{3.8}$$

$$b^*_i(o_k) = \frac{\sum_{t=1}^{T-1} \gamma_i(t) I_{O(t)=o_k}}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{3.9}$$

where $I_{O(t)=o_k}$ stands for indicator function, which equals to 1 if $O(t) = o_k$, and 0 otherwise. Then, for the case of multiple sequences, each sequences indexed by s, total number of sequences of m, The only changing is the updating of π^*,A^*, and B^* as follows.

$$\pi(i)^* = \frac{\sum_{s=1}^{m} \gamma_i^s(1)}{m} \tag{3.10}$$

$$a^*_{ij} = \frac{\sum_{s=1}^{m} \sum_{t=1}^{T^s-1} \xi_{ij}^s(t)}{\sum_{s=1}^{m} \sum_{t=1}^{T^s-1} \gamma_i^s(t)} \tag{3.11}$$

$$b^*_i(o_k) = \frac{\sum_{s=1}^{m} \sum_{t=1}^{T^s-1} \gamma_i^s(t) I_{O^s(t)=o_k}}{\sum_{s=1}^{m} \sum_{t=1}^{T^s-1} \gamma_i^s(t)} \tag{3.12}$$

The procedure above is repeated till either the $\sum_s^m log(P(O^s|\theta))$ converge or reaching maximum iteration numbers set by the user. As mentioned in the section 3.1, a key difference between the proposed method and [54] lies in the E-step for calculating the expected value for a given emission or transition. The proposed method handles the partial label constraints recursively for the α and β, whereas [54] calculates α and β without using the partial labels and only uses the partial labels in resetting γ at each partial labeled position independently, as if partial labels elsewhere would have no effect for the position being considered. Since E-step in Baum-Welch algorithm invokes forward and backward algorithms, which are essentially a dynamic programming to more efficiently calculate the likelihood: $Pr(O|\theta) = \Sigma_{X \in \Gamma} Pr(O, X|\theta)$ with Γ being the set of all hidden paths, and hence should give the same result when the likelihood is

computed by exhaustively summing over probability for all possible hidden state paths. Therefore, the partial labels would restrict the possible hidden state paths, $Pr(O|\theta) = \Sigma_{X \in \Gamma'} Pr(O, X|\theta)$ with Γ' being the set of all hidden paths constrained by partial labels and such constraints should be handled recursively in the dynamic programming. Figure 3.1 shows an example for the forward/backward dynamic programming table construction. Another advantage of our method comparing with the method in [54] is that the proposed training method can keep the topology of the initial transition and emission guesses for the model as standard Baum-Welch does. In other words, if prior knowledge is available for the model topology, the proposed training method for partial label data can keep the knowledge to the end of training.

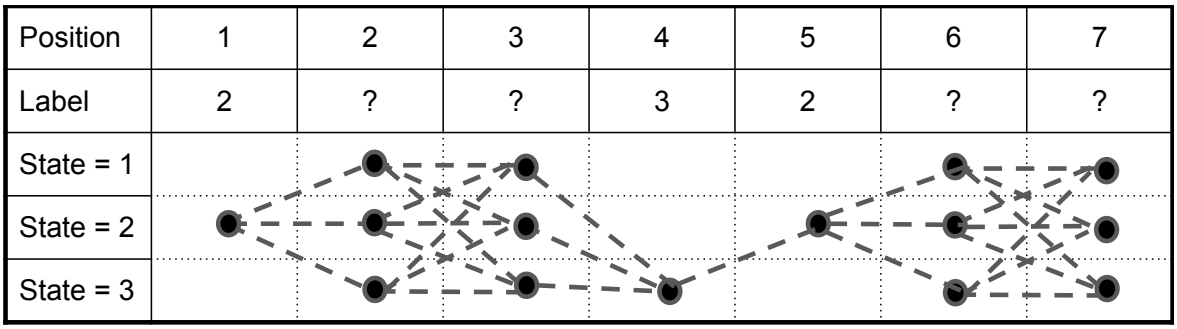

Figure 3.1: Example of constrained Baum-Welch's forward/backward dynamic programming table construction. Position 1 to 2 is the case of labelled position to unlabelled position. Position 3 to 4 is the case of unlabelled position to labelled position. Position 4 to 5 is the case of labelled position to labelled position. Dashed lines indicate state transitions.

3.2.2 Model Selection Based on Partial Label Information

The second part of the proposed method is model selection based on partial label information. The rationale is straightforward: while the constrained Baum-Welch algorithm increases the log-likelihood of the given training sequences (with partial labels), iteration after iteration monotonically as ensured by EM approach, there is no direct guarantee that the increased log-likelihood would necessarily lead to higher

decoding accuracy. Therefore, at each iteration of constrained Baum-Welch algorithm, decoding accuracy for the partially labelled training sequence can be calculated and factored into model selection.

Specifically, after reaching convergence condition or maximum number of iterations, the total number of iteration is Q and the i^{th} iteration's model and the corresponding log-likelihood can be denoted as θ_i and $\sum_s^m Log(P(O^s|\theta_i))$ respectively and let the decoding accuracy denote as $Accuracy(\theta_i, O, X)$. The final model returned by the algorithm can be expressed as:

$$\underset{\theta^*}{\operatorname{argmin}} Pr(O|\theta^* \equiv \{\underset{\theta_i \in \theta_{1\ldots Q}}{\operatorname{argmax}} Accuracy(\theta_i, O, X)\}) \tag{3.13}$$

Notice that θ^* is a set of models in general. Finally, combining the constrained Baum-Welch and the model selection described above, the overall algorithm of the proposed method is given in Algorithm 1. In next section, Table 3.2 will show the usefulness of both this model selection method and the ability of keeping correct topology of cBW method.

3.3 Experiments and Results

In this section, experiments are set up using both real biological data and synthetic data to test the proposed method for decoding task and compared the results with those from using the method in [54]. It has been reported that [37, 24] posterior decoding in general performs better than Viterbi algorithm. So, in order to evaluate how our training method can impact on decoding, decoding on the testing sequences with the trained model using both the standard Viterbi algorithm [51] and Posterior-Viterbi algorithm described in [24] are carried out, and the accuracy was computed by comparing the predicted label with the ground truth label at each position to determine the number of correct predictions:

$$Accuracy = \frac{\#of\ correct\ predicted\ labels}{\#of\ total\ labels}$$

Algorithm 1: Constrained Baum-Welch with model selection based on decoding accuracy.

Data: sequences: O, partially label X
Result: bestA,bestB
initialization: $\theta = (\pi, A, B)$;
bestAccuracy $\leftarrow 0$;
while *not reaching maximum iteration nor convergent* **do**
  calculate α,β by Eq. 3.1 to 3.4 ;
  calculate γ, ξ by Eq. 3.2.1 to 3.2.1 ;
  update π^*,A^*,B^* by Eq. 3.7 to 3.9 ;
  $X^* \leftarrow$ ViterbiDecoding(π^*,A^*,B^*,O) ;
  myAccuracy $\leftarrow$ Accuracy(X^*,X) ;
  if *myAccuracy* $>$ *bestAccuracy* **then**
    bestAccuracy $\leftarrow$ myAccuracy ;
    $\pi \leftarrow \pi^*$;
    bestA $\leftarrow A^*$;
    bestB $\leftarrow B^*$;
  end
end

The results of these experiments show that the proposed method outperforms Scheffer et al's method in model training, as evidenced in the improved decoding accuracy, regardless which decoding algorithm is used. Specifically, on average, decoding accuracy improves by 33% with Viterbi algorithm, 36% with Posterior-Viterbi algorithm in real data, and improves by 7.35% to 14.06% with Viterbi algorithm, 7.08% to 13.89% with Posterior-Viterbi algorithm in synthetic data with significant p-values. Note that, in two cases when the sequences are either almost fully labelled (95%) or very sparsely labelled (5%), the differences between various algorithms are insignificant. This phenomenon is no surprising though, as it is expected that the benefit from making good use of partial labels diminishes when labels are extremely sparse, which makes the various algorithms converge to Baum-Welch algorithm, or when sequences are almost fully labelled, which makes the various algorithms converge to the maximum likelihood. Therefore, our evaluations are divided into two settings for synthetic data. Setting 1 has partial label information from 5% to 95%. Setting 2 has partial label information from 10% to 90%.

3.3.1 Synthetic Data

The method described in [54] is mainly focused on handling text mining problems using synthetic data. To make the comparison fair, experiments using synthetic data are also performed , which allowed us to observe the proposed method's different performance in different situations. In the experiments with synthetic data, the data is generated from ground truth HMMs, which are also generated randomly with predefined connections. For each experiment, the size for initial guess of transition and emission matrices are identical to the corresponding ground truth model.

The number of symbols in hidden Markov model is fixed to be 20 to mimic the 20 amino acids in protein sequences. To test how model complexity may impact the training, three different numbers of states: 3, 5, and 7, are chosen. Moreover, different levels of training sample size were also considered as an experimental variable. Each experiment (with fixing state number and training examples) was evaluated for different levels of partial label and repeated for 50 times, and the corresponding paired p-values were also calculated to assess the statistical significance of the performance difference between our method and the other method. Since the proposed method can maintain the topology of initial guess of transition matrix, experiments were divided into two different groups. One was initialized the transition matrix with the same connectivity as the ground truth model, and the other was initialized with fully connected transition matrix.

Three sets of experimental results with fully connected transition matrix as initial guess are shown in Figure 3.2 to Figure 3.4. Additional results are shown in Table 3.1 to Table 3.4 for comparison.

Conducted using different numbers of states, training examples, and different decoding algorithms, the results show that the proposed method outperforms the method by Scheffer et al by 7.08% to 14.06% across different percentage of unlabelled data, with significant p-value (< 0.05) for majority of the experiments. While both methods achieve a performance closer to that of the ground truth model as the level of partial labels increases, the improvement of our method over the method of Scheffer et al's is

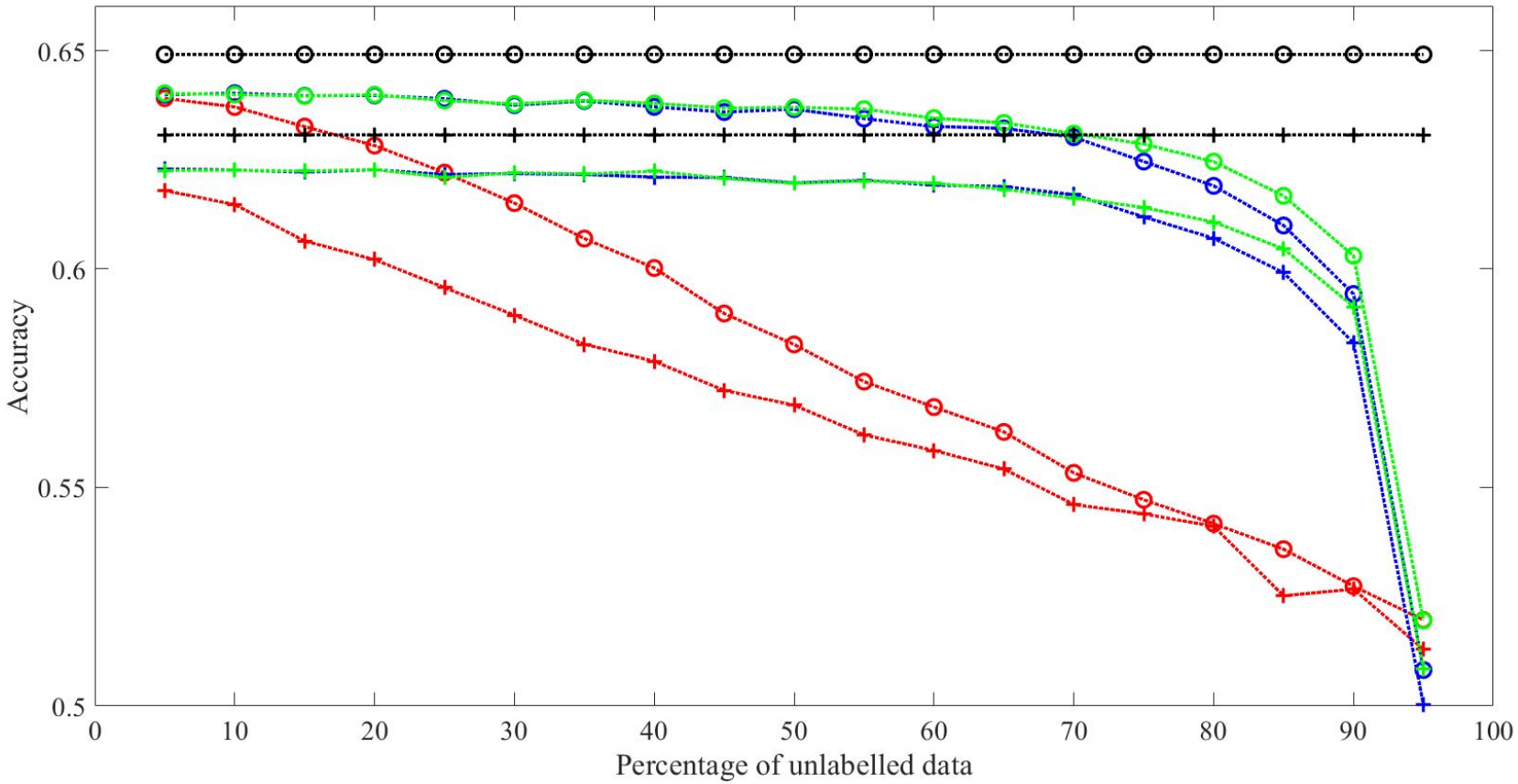

Figure 3.2: Comparison results 1 with synthetic data when initialized with fully connected transition matrix. state number = 3 and training sample size = 1600. **Training Methods:** Ground truth model – black; cBW – green; cBW+model_selection – blue; Scheffer et al – red. **Decoding Methods:** Viterbi – cross tick mark; Posterior-Viterbi – circle tick mark.

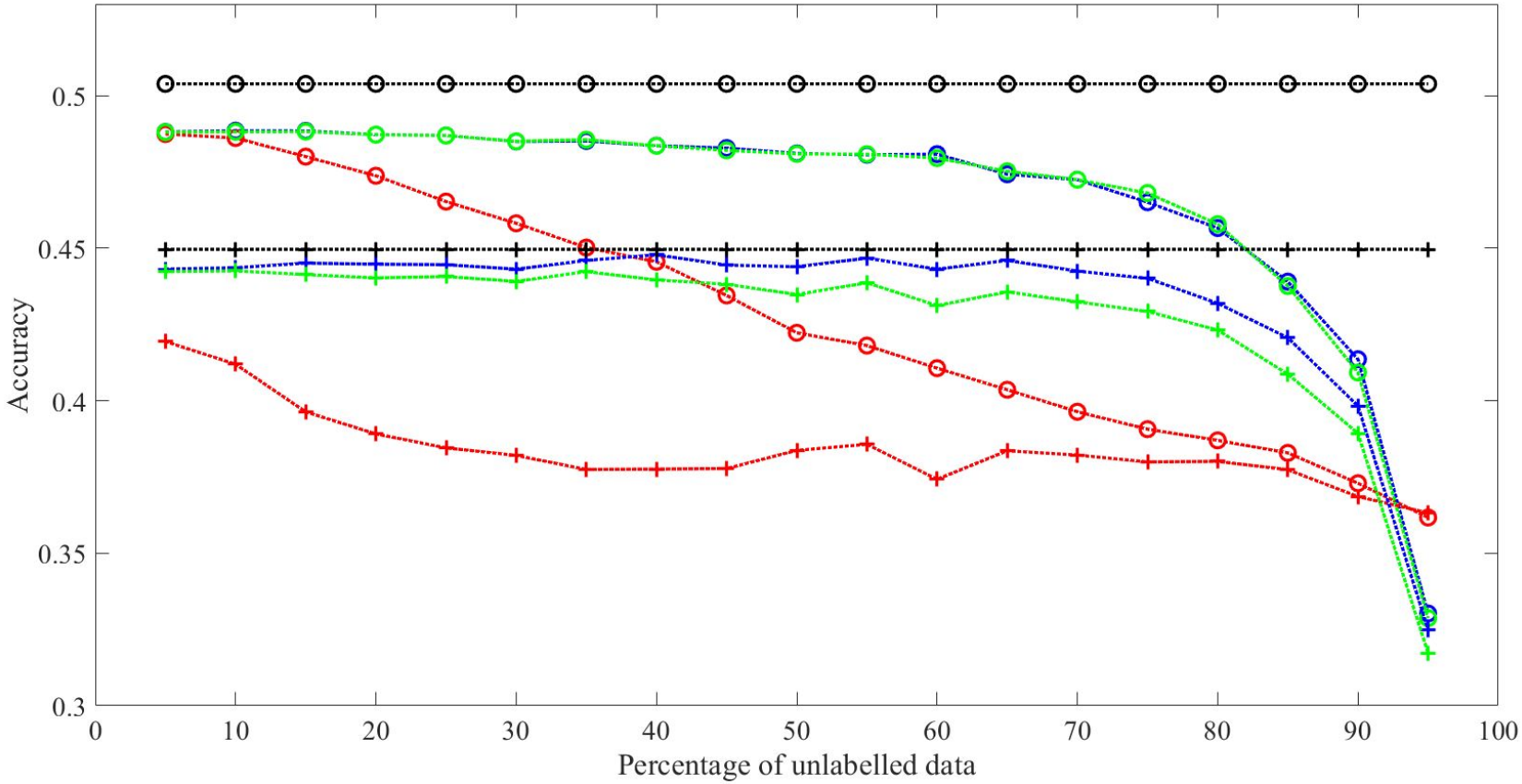

Figure 3.3: Comparison results 2 with synthetic data when initialized with fully connected transition matrix. State number = 5 and training sample size = 1600. **Training Methods:** Ground truth model – black; cBW – green; cBW+model_selection – blue; Scheffer et al – red. **Decoding Methods:** Viterbi – cross tick mark; Posterior-Viterbi – circle tick mark.

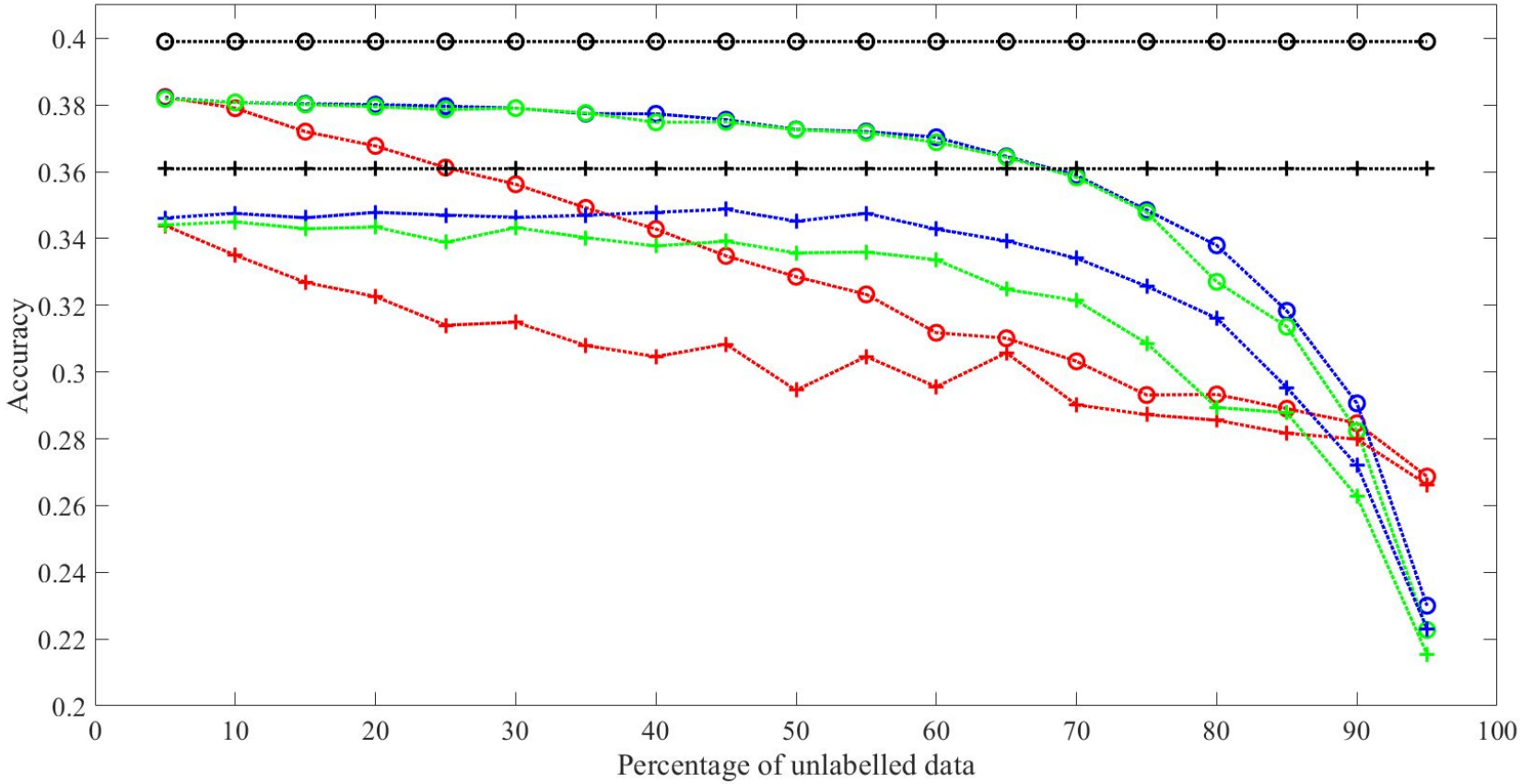

Figure 3.4: Comparison results 3 with synthetic data when initialized with fully connected transition matrix. State number = 7 and training sample size is 1600. **Training Methods:** Ground truth model – black; cBW – green; cBW+model_selection – blue; Scheffer et al – red. **Decoding Methods:** Viterbi – cross tick mark; Posterior-Viterbi – circle tick mark.

more pronounced when partial labels are sparse, namely the level of unlabelled data is high, as shown in the X-axis of the Figures. For example, in Figure 3.2, with Viterbi decoding, at the level of 70% unlabelled data, i.e., 30% partial labels, our method reaches an accuracy of 62%, which is 98% of the ground truth model accuracy, whereas Scheffer et al's reaches accuracy of 54%, which is 85% of the ground truth model accuracy. Similar trends hold true for Figure 3.3 and Figure 3.4 when the model has 5 and 7 states respectively regardless of the decoding algorithm used.

3.3.2 Real Data

For the real biological data, data from [36] is adopted. The data contains 83 multi-pass transmembrane proteins with complete label information. For readers' reference, these 83 transmembrane protein names are list in Appendix B. The topology of multi-pass transmembrane protein is shown in Figure 3.5. The label for each sequence contain three different values: **i**, **o**, **M**. They stand for the region of protein sequence inside, outside cell membrane, and the transmembrane domain respectively. While much more sophisticated hidden Markov models have been used for modeling transmembrane protein topology [36, 56, 38, 29], a simple HMM is used in this study to primarily evaluate the new training algorithm for partial labels. The architecture of the HMM is shown in Figure 3.6, in which a redundant **M'** node is introduced as a simple mechanism to avoid a state path, such as **iiiimmmmiii** or **oooommmoooo**, that does not correspond to the real topology of transmembrane protein, in which a membrane domain has to be flanked by **i** on one side and **o** on the other side. Therefore, the transition matrix is 4 by 4, corresponding to the four states. Note that the amino acid emission frequencies for the transmembrane state are calculated by lumping together counts or expectation from both M and M' states. Two different experiments are set up with different initial conditions:

1. Transition matrix has correct zeros as ground truth model.

2. Transition matrix is fully connected.

Table 3.1: Improvements of cBW + model selection, cBW alon vs Scheffer et al, with Fully Connected Initial Transition Matrix for Synthetic Data with Viterbi Algorithm

number of state / training samples	Average improvements of cBW+model selection in setting 1 / 2	Average p-value of cBW+model selection in setting 1 / 2	Average improvements of cBW alone in setting 1 / 2
3 /1600	7.35% / 8.29%	2.1E-02 / 6.2E-05	7.61% / 8.49%
3 /2000	7.99% / 8.82%	3.9E-02 / 2.6E-09	8.31% / 9.00%
3 /2400	8.47% / 9.13%	2.8E-03 / 2.4E-10	8.71% / 9.18%
3 /2800	8.51% / 9.24%	5.8E-03 / 6.9E-10	8.65% / 9.24%
5 /1600	12.97% / 14.75%	7.8E-05 / 3.3E-05	11.13% / 12.82%
5 /2000	14.63% / 16.32%	2.5E-03 / 1.2E-06	12.65% / 14.11%
5 /2400	14.69% / 16.54%	7.8E-04 / 6.2E-06	12.73% / 14.50%
5 /2800	15.56% / 17.22%	1.6E-02 / 1.3E-07	13.72% / 15.22%
7 /1600	8.61% / 10.42%	3.5E-02 / 1.1E-02	5.56% / 7.20%
7 /2000	10.56% / 12.40%	6.4E-03 / 6.2E-03	7.87% / 9.52%
7 /2400	11.35% / 13.21%	8.2E-03 / 7.8E-03	8.71% / 10.43%
7 /2800	12.16% / 14.06%	1.0E-03 / 1.1E-04	9.68% / 11.41%

Table 3.2: Improvements of cBW + model selection, cBW alone vs Scheffer et al in Cases of Correct Initial Transition Matrix for Synthetic Data with Viterbi Algorithm

number of state / training samples	Average improvements of cBW+model selection in setting 1 / 2	Average p-value of cBW+model selection in setting 1 / 2	Average improvements of cBW alone in setting 1 / 2
3 /1600	8.07% / 8.57%	2.3E-04 / 4.7E-07	8.11% / 8.56%
3 /2000	8.58% / 9.05%	1.9E-06 / 6.5E-09	8.63% / 9.03%
3 /2400	8.93% / 9.24%	1.6E-07 / 1.0E-09	8.97% / 9.20%
3 /2800	8.87% / 9.31%	1.3E-08 / 1.5E-09	8.94% / 9.26%
5 /1600	11.99% / 13.24%	1.7E-02 / 4.5E-06	11.76% / 13.08%
5 /2000	13.07% / 14.20%	4.1E-02 / 3.4E-06	12.87% / 14.11%
5 /2400	13.22% / 14.59%	2.0E-02 / 1.2E-05	12.94% / 14.35%
5 /2800	13.89% / 15.20%	4.1E-02 / 1.6E-07	13.85% / 15.16%
7 /1600	7.85% / 9.37%	6.7E-02 / 3.5E-02	6.04% / 7.34%
7 /2000	9.75% / 11.28%	5.6E-03 / 4.1E-03	7.93% / 9.32%
7 /2400	10.50% / 12.10%	1.8E-02 / 1.9E-02	8.99% / 10.53%
7 /2800	11.39% / 12.95%	1.4E-03 / 1.9E-04	9.75% / 11.29%

Table 3.3: Improvements of cBW + model selection, cBW alon vs Scheffer et al, with Fully Connected Initial Transition Matrix for Synthetic Data with Posterior-Viterbi Algorithm

number of state / training samples	Average improvements of cBW+model selection in setting 1 / 2	Average p-value of cBW+model selection in setting 1 / 2	Average improvements of cBW alone in setting 1 / 2
3 /1600	7.08% / 8.02%	3.6E-02 / 1.9E-04	7.49% / 8.35%
3 /2000	7.93% / 8.57%	9.5E-03 / 3.9E-05	8.32% / 8.88%
3 /2400	8.21% / 8.85%	1.7E-03 / 2.5E-05	8.55% / 9.03%
3 /2800	8.43% / 9.15%	1.8E-03 / 1.1E-06	8.82% / 9.36%
5 /1600	9.13% / 10.62%	2.4E-02 / 3.2E-03	9.08% / 10.58%
5 /2000	10.32% / 11.68%	1.4E-02 / 6.2E-05	10.38% / 11.76%
5 /2400	11.26% / 12.74%	1.1E-02 / 2.0E-06	11.29% / 12.84%
5 /2800	12.48% / 13.76%	9.8E-03 / 2.1E-08	12.56% / 13.86%
7 /1600	8.40% / 10.08%	6.0E-02 / 2.5E-02	7.77% / 9.50%
7 /2000	10.22% / 11.96%	3.2E-02 / 1.3E-04	9.82% / 11.65%
7 /2400	11.10% / 12.68%	8.1E-03 / 1.5E-05	10.60% / 12.31%
7 /2800	12.18% / 13.89%	1.6E-04 / 8.4E-08	11.96% / 13.77%

Table 3.4: Improvements of cBW + model selection, cBW alone vs Scheffer et al in Cases of Correct Initial Transition Matrix for Synthetic Data with Posterior-Viterbi Algorithm

number of state / training samples	Average improvements of cBW+model selection in setting 1 / 2	Average p-value of cBW+model selection in setting 1 / 2	Average improvements of cBW alone in setting 1 / 2
3 /1600	7.46% / 8.01%	2.8E-02 / 1.7E-02	7.64% / 8.11%
3 /2000	8.09% / 8.52%	3.0E-02 / 1.6E-02	8.25% / 8.60%
3 /2400	8.32% / 8.67%	3.7E-02 / 5.9E-03	8.49% / 8.73%
3 /2800	8.58% / 8.99%	4.9E-02 / 1.2E-03	8.79% / 9.09%
5 /1600	9.52% / 10.62%	1.1E-01 / 5.1E-02	9.45% / 10.59%
5 /2000	10.53% / 11.55%	6.0E-02 / 8.2E-03	10.47% / 11.63%
5 /2400	11.51% / 12.58%	2.0E-02 / 3.5E-04	11.44% / 12.58%
5 /2800	12.49% / 13.55%	1.7E-02 / 8.0E-06	12.55% / 13.61%
7 /1600	8.75% / 10.19%	2.9E-02 / 2.6E-02	8.27% / 9.77%
7 /2000	10.50% / 11.99%	2.1E-02 / 4.2E-03	9.82% / 11.31%
7 /2400	11.15% / 12.47%	6.9E-02 / 7.8E-04	10.69% / 12.11%
7 /2800	12.36% / 13.75%	5.2E-02 / 1.1E-05	12.05% / 13.57%

The reason for performing experiments for condition (2) is that the method in [54] cannot enforce initial zeros to remain zeros during the training, therefore, condition (2) gives more fair comparison of the two methods when no prior knowledge is available. The HMM is trained by these two different methods in a 10 fold cross validation scheme. Different levels of unlabelled data in training examples are actuated by selecting locations randomly to be unlabelled for each sequence. Since no ground truth model is available, maximum likelihood method with fully labelled training data is used to mimic the role of the ground truth model in experiments with synthetic.

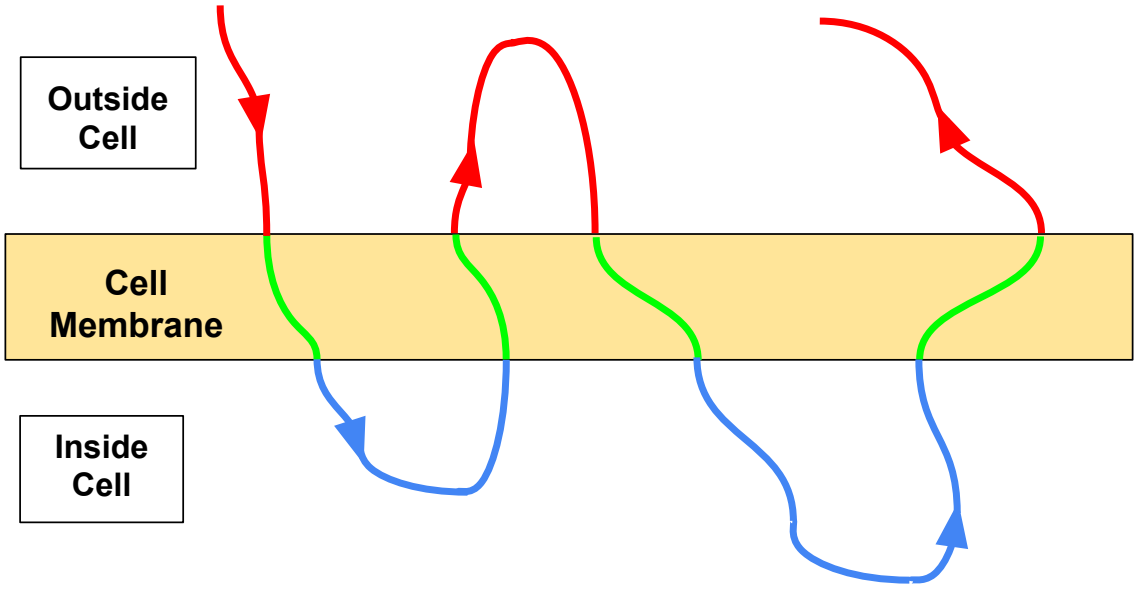

Figure 3.5: Multi-pass transmembrane proteins. The red lines represents protein sequence outside of cell membrane, the blue lines represents protein sequence inside of cell membrane, and green line represents transmembrane domain of the protein sequence.

For condition (1), the result shown in Figure 3.7 demonstrates that the proposed method (constrained Baum-Welch with model selection) outperforms other method (Scheffer et al) by 33.59% with Viterbi Algorithm and 36.16% with Posterior-Viterbi

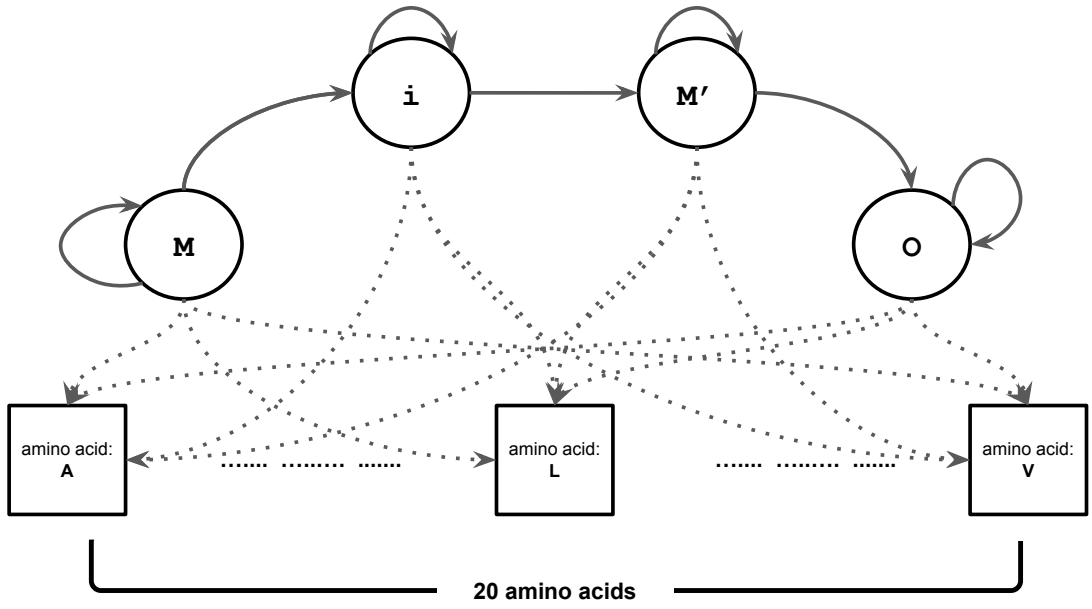

Figure 3.6: Topology of 4-state HMM for multi-pass transmembrane prediction. States **I** and **O** represent inside and outside cell membrane respectively. Both **M** and **M'** stand for transmembrane domain, the redundant **M'** is used to avoid direct connection between state **i** and **o**, which is impossible.

algorithm. For condition (2), the result shown in Figure 3.8 attests that the proposed method outperforms other method by 33.20% with Viterbi Algorithm and 36.32% with Posterior-Viterbi algorithm. For both conditions, the performance of the proposed method with or without model selection technique and maximum likelihood are very close.

3.4 Discussion and Conclusion

From the results of experiments with synthetic data in Table 3.1 to Table 3.4, they show:

- Constrained Baum-Welch algorithm with or without model selection technique achieve significant better performance than Scheffer et al in [54]

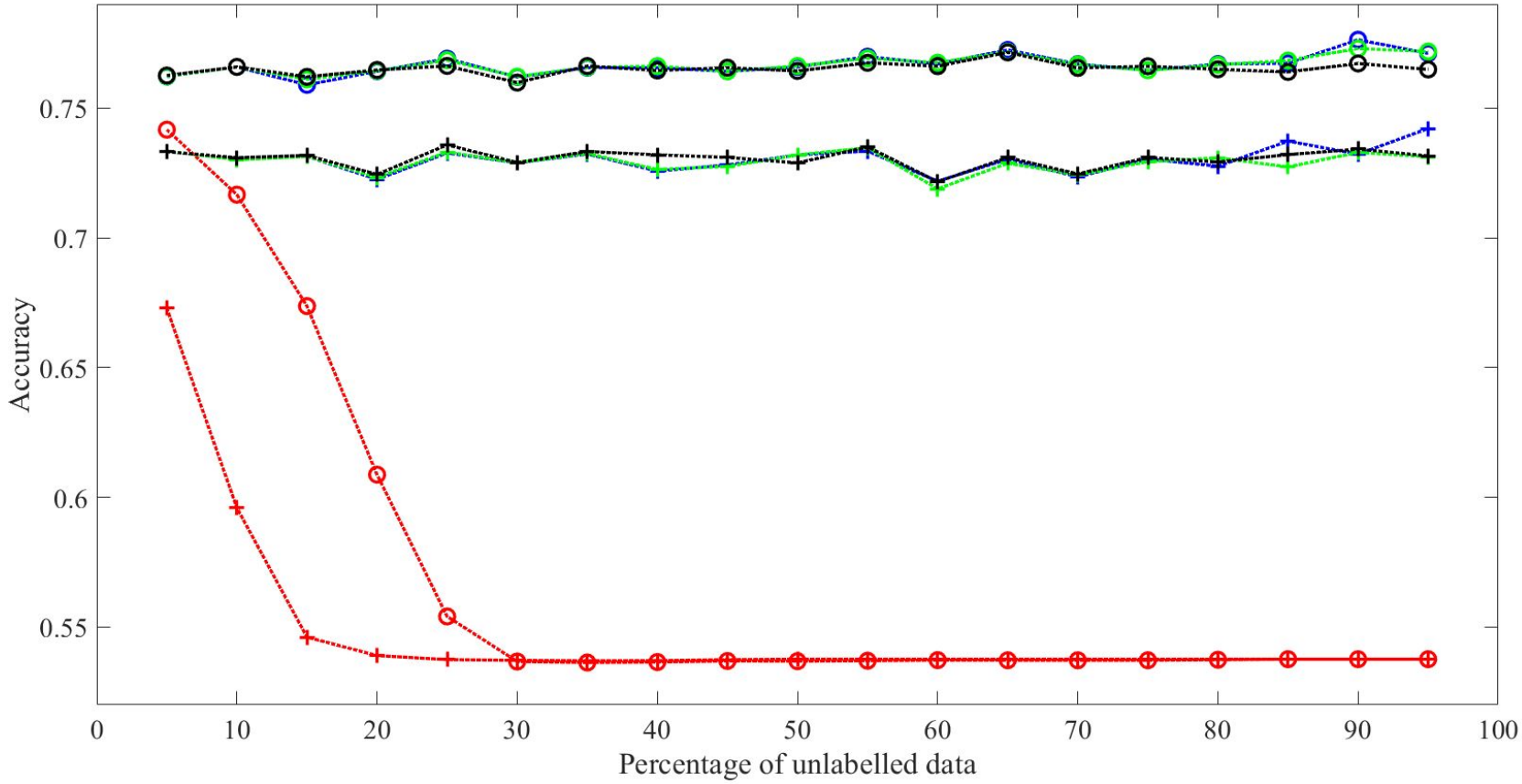

Figure 3.7: Comparison with real data when initialized with transition matrix of the correct connectivity in Fig 3.6. **Training Methods:** ML – black; cBW – green; cBW+model_selection – blue; Scheffer et al – red. **Decoding Methods:** Viterbi – cross tick mark; Posterior-Viterbi – circle tick mark..

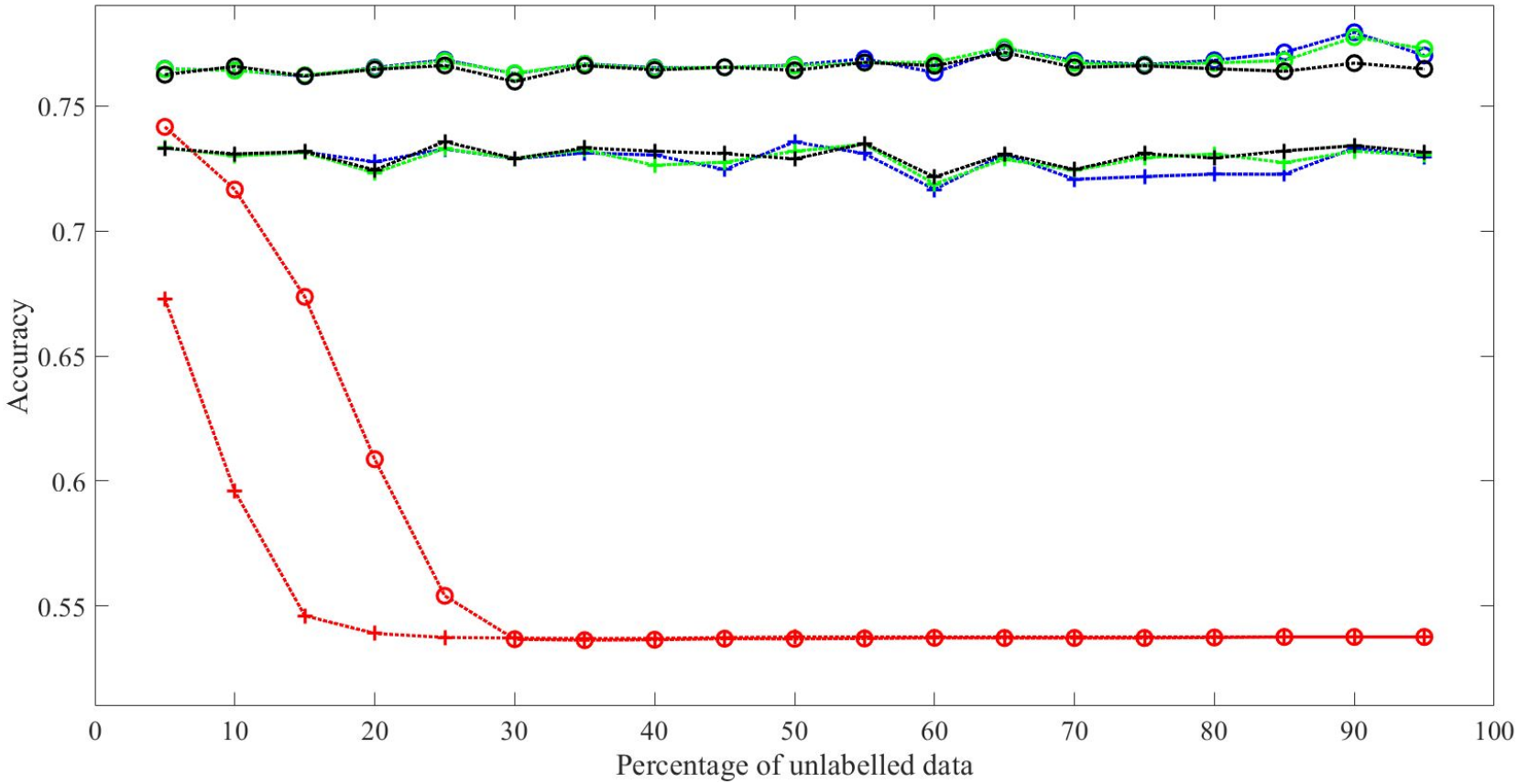

Figure 3.8: Comparison results with real data when initialized with fully connected transition matrix. **Training Methods:** ML – black; cBW – green; cBW+model_selection – blue; Scheffer et al – red. **Decoding Methods:** Viterbi – cross tick mark; Posterior-Viterbi – circle tick mark.

- Constrained Baum-Welch benefit from having correct topology (comparisons between 3rd and 4th columns in Table 3)
- Constrained Baum-Welch algorithm performs better when model selection technique used, especially when the task is hard (comparisons between 2nd column in Table 3.1 and 3rd column in Table 3.2, comparisons between 2nd and 4th columns in Table 3.2).
- Disregarding the training methods, Posterior-Viterbi always outperforms standard Viterbi.

From the results of experiments with real data, performance of constrained Baum-Welch with or without model selection are very close to maximum likelihood approach across different percentages of partial label. However, the performance of Scheffer et al's drops dramatically after the percentage of unlabelled data is greater than 10. The reason behind this is the method done by Scheffer et al cannot enforce the correct topology even the initial guess is correct. For this problem in particular, have a HMM with correct topology is key for higher accuracy.

Moreover, there are a few points worth mentioning for the benefits of those who may consider using this method for their applications. First, the ability of keeping correct topology makes cBW method compatible with more complex HMM, such as profile HMMs. However, as a trade-off, the training time can significantly increase. Second, model selection technique, although optional, is highly recommended to be used with Posterior-Viterbi instead of standard Viterbi for best decoding performance. Lastly, the proposed method is designed especially for the task of detecting de novo targeting signals, which assumes no fully labelled sequence is available in general. For the cases with relaxing constrain: some fully labelled sequences are available, our method is not the only choice, interested readers may also consider methods in [58].

In this work, by modifying the standard Baum-Welch algorithm, a novel training method is developed, which, along with a simple model selection scheme, enables leveraging the partial labels in the data to improve the training of hidden Markov

models. Compared with a similar method, the proposed method achieved significant improvements in training hidden Markov models as evidenced by better performance in decoding both synthetic data and the real biological sequence data.

Chapter 4

TRAINING HIDDEN MARKOV MODELS WITH POSITIVE AND NEGATIVE EXAMPLES

4.1 Introduction

As a well known statistical tool for modelling sequential data, hidden Markov models (HMM) [9, 8, 11, 10, 7] have been successfully applied to many problems in the fields of nature language processing and bioinformatics, such as speech recognition [3, 49, 16], biological sequence alignment [55, 25, 23], and CpG island prediction [21, 6]. These great successful applications of HMMs in different domains, like most methods in machine learning, their successes rely to a large degree on both the quantity and quality of data. Meanwhile, there are still many situations where both quantity and quality of the data are poor, preventing any machine learning tools to fully capture the knowledge of the data, especially for ongoing biological research [43]. And the fact that HMMs use only positive examples for training exacerbates the issue of limited training data, by leaving out negative examples with no use. To tackle this issue, in this chapter, a novel Baum-Welch based training algorithm is proposed to utilize both positive and negative data to train HMMs in order to enhance the membership prediction task for biological sequences when the number of positive examples is limited.

This work is inspired by the research on detecting novel Plasmodesmata-located proteins (PDLPs) [43] as shown in chapter 2. PDLPs are type-I transmembrane proteins, which are targeted to intercellular pores called Plasmodesmata that form at the cellular junctions in plants [42]. At the beginning of this research, only 8 PDLPs have been verified in Arabidopsis thaliana experimentally, and their PD-targeting signal regions have been narrowed down to extracellular juxtamembrane domain (JMe). In chapter 2, a 3-state HMM called PDHMM was designed to model PDLPs' JMe region

for detecting de novo PD-targeting signal and identifying novel PD proteins. By wet-lab experiments, several predicted PD-targeting signals have been successfully verified. However, identifying de novo PD proteins still remains a challenge: for those proteins ranked highly by PDHMM as PD-targeting candidates, some are found to be none-PD as reported in the literature and many others are still unknown. Due to the high cost of wet-lab experiments, an improved HMM for PD proteins prediction has become even more imperative for this ongoing research. One way to improve the model is having more data, and due to the limited number of positive data, an HMM training algorithm of utilizing both positive and negative data is urgently needed.

In this chapter, a novel training method is developed to utilize both positive and negative data for training a "comprehensive" HMM, or CompHMM for short. Hereafter, without confusion, CompHMM is used to refer to both the training method and the model thus trained. CompHMM train the HMM in a discriminative fashion: the positive and negative examples are separated in its measure space. In contrast, conventional training algorithms for HMMs, such as Baum-Welch, Viterbi training, and maximum likelihood, are considered as a positive-only approach, because they use only positive data for training. As such, HMMs belong to a category of models, called generative models in the field of machine learning, which means that the trained model can be used to generate synthetic examples with varied likelihood of being a positive example, but cannot give a clear decision boundary, although in practice 50% likelihood may be used as a de facto threshold. On the other hand, classifiers such as support vector machines that give a clear decision boundary are often referred to as discriminative model. Note that, although trained in a discrinimative manner, the trained CompHMM is a hidden Markov model nonetheless, and hence does not give a clear decision boundary. While there has been efforts to remedy the positive-only training limitation of HMMs by combining it with another discriminative classifier such as SVM [33][27], the goal here is to modify the training within HMMs so that both positive and negative examples can be used. The most relevant work to the proposed method was the one carried out by Mamitsuka [48]. This method, named as "MA", also

attempts to have a discriminative manner training for HMM but is based on gradient descent technique developed in [5]. One apparent advantage of CompHMM over MA method is that fewer hyper-parameters to select in CompHMM. In MA method, there are four hyper-parameters, and CompHMM only has one, which will be defined in section 4.2.4. Besides having fewer hyper-parameters, based on experimental results from both synthetic and real data, the proposed method outperformed standard Baum-Welch and MA method significantly.

The rest of this chapter is organized as the following. In the section 4.2, a brief review of HMMs is first given, primarily to introduce the notations and revisit the standard Baum-Welch algorithm. What follows is a key step in this new method, called reverse Baum-Welch. Then the whole procedure of CompHMM is presented and explained. In section 4.3, detailed descriptions are given for synthetic data generation and the setup of experiments for testing the methods with both synthetic and real data. Then, comparisons of CompHMM with the standard Baum-Welch and MA methods are given. Last, discussion and conclusion are given at the end.

4.2 Method

In this section, a brief introduction to HMM is given first. Then, with understanding of the basic problems and algorithms in HMM, CompHMM will be described by three steps: revisiting Baum-Welch algorithm, introducing reverse Baum-Welch, and combining standard and reverse Baum-Welch to implement CompHMM.

4.2.1 Introduction to HMM

HMMs are a well known tool in machine learning field for modeling sequential data, mainly to discover underlying structures and patterns. One of the concepts in modeling sequential data with HMMs is to define hidden states corresponding to those underlying structures/patterns, besides the data, which is called "observable". One example can be people's activities (observable) and their emotions (hidden) on each day. Another example from biology can be a genomic sequence (observable) and

whether a nucleotide is in a gene (hidden state) or an intergenic region (another hidden state). The core assumption of modeling data in such a way is that the observation is emitted from the hidden states by some probabilities and the a hidden state can transition to another hidden state with certain probability, and the path of state to state transitions forms a Markov chain. And as the name of HMM suggested, it is designed to capture this observable and hidden relationship of sequential data.

A hidden Markov model is referred to by its parameters, denoted as θ collectively, which contains three sets of parameters $\theta = (\pi, A, B)$. π is called initial probability over the N states of the model, from which the hidden Markov chain starts with, in other words, π's element π_i stands for the probability of being state i at time $t = 1$. A is called transition matrix with dimension of $N \times N$ whose elements denoted as $a_{i,j}$ gives the probability of transition from state i to state j. B is called emission matrix with dimension of $N \times K$ whose elements denoted as $b_j(k)$, which stands for the probability of being at state j and emitted observation (symbol) k, where K is the size of alphabets (or symbols) of the observable data.

With a HMM θ and a sequence of observation O whose elements are $O_t \in K$, where $t = 1..T$. There are three tasks in data modeling with HMMs. One task is to compute $Pr(O|\theta)$, namely, the probability that sequence O is a member of the sequence family described by HMM θ. This probability, also called likelihood when considered as a function of θ, is computed by the so-called Forward algorithm efficiently. The second task is to train the HMM so that for m sequences, $\sum_{s=1}^{m} Pr(O^s|\theta)$ is maximized. When the sequence of hidden states or hidden path is given, this is done simply by maximum likelihood approach, namely by counting the numbers of various transitions and emissions, followed by proper normalization. When only observation is available, one popular method is Baum-Welch algorithm. The third task is called decoding task. This task is to find the most probable hidden state sequence X^* for a given observation sequence O and model θ: $X^* = \text{argmax}_X \, Pr(O, X|\theta)$. This task is carried out using Viterbi algorithm and is not the focus of this paper. For readers who are interested in knowing more about HMM, a comprehensive introduction can be found

in [51]. In this work, the focus is on the second task as discussed above. In the second task, Baum-Welch algorthim trains the HMM so that a particular set of sequences' summed probability is maximized, when only observation sequences, not the hidden state path, are available. Let this particular set of sequences called positive set. The assumption is that these sequences share commonalities in a way that they can be collectively described by a hidden Markov model. As such, the training is to fix the model parameters θ to maximize the likelihood, which is the principle of maximum likelihood. When the data are unlabeled, namely, the hidden state paths are latent, the technique for maximizing the likelihood is called Expectation-Maximization, see next subsection for details. After the training, one can use the trained HMM to calculate the log-likelihood of any given sequence and use this log-likelihood as a score metric to determine how likely the sequence belongs to this positive set. The focus of this chapter is to train the HMM discriminatively so that the model can separate the target positive set from some negative sets and gain additional differentiating power. Table 1.1, introduce general notations used throughout this chapter with explanations.

4.2.2 Baum-Welch Revisit

Since CompHMM is built based on Baum-Welch algorithm, a revisit of it will be helpful as the first step. We will not go into much details of Baum-Welch algorithm but only the parts that are necessary.

Baum-Welch algorithm is an expectation-maximization (EM) algorithm, which has two steps: expectation (E) and maximization (M) steps. The E-step's calculation needs two matrices with dimension of $N \times T$ given HMM's state number N, and a sequence with length T. These two matrices are called forward and backward matrices and denoted as α and β respectively. Forward matrix's element $\alpha_i(t)$ stands for the probability of the symbol O_t being emitted from state i at time t given the hidden Markov model θ and the sub-sequence before time t: $O_{q=1\ldots t}$. Backward matrix's element $\beta_i(t)$ stands for the probability of the symbol O_t being emitted from state i

at time t given model θ and the sub-sequence after time t: $O_{q=t+1\ldots T}$. They are all calculated recursively by forward-backward procedure as follows.

Baum-Welch algorithm starts with initial guesses of transition and emission matrices A and B. Then for each state $i \in N$, the value of α at time $t = 1$ is initialized by π:

$$\alpha_i(1) = \pi b_i(O_1) \tag{4.1}$$

With the above initialization, the remaining values of α are calculated recursively:

$$\alpha_i(t+1) = b_i(O_{t+1}) \sum_{j=1}^{N} \alpha_j(t) a_{ji} \tag{4.2}$$

Similar to α's recursion calculation, β's calculation is backward of the time and initialized with all ones at time $t = T$:

$$\beta_i(T) = 1 \tag{4.3}$$

Then, the remaining values of β are calculated backward recursively:

$$\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(O(t+1)) \tag{4.4}$$

With α and β, we can calculate the posterior state probabilities γ whose element $\gamma_i(t)$ stands for the probability of the symbol O_t visit state i at time t given the full sequence O.

$$\begin{aligned} \gamma_i(t) = P(X(t) = i|\theta, O) &= \frac{P(X(t) = i, O|\theta)}{P(O|\theta)} \\ &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)} \end{aligned} \tag{4.5}$$

And also $\xi_{ij}(t)$, which stands for the summed probability of all possible hidden paths with the restrict that at time t the symbol O_t is emitted from state i and at time t+1 O_{t+1} is emitted from state j.

$$\begin{aligned}\xi_{ij}(t) &= \frac{P(X(t)=i, X(t+1)=j, O|\theta)}{P(O|\theta)} \\ &= \frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(O(t+1))}{P(O|\theta)}\end{aligned} \tag{4.6}$$

At this point, the expectation information is captured in γ and ξ. The next step is M-step: maximization.

In case of multiple sequences, the updated probability of state i transit to state j, which denoted as a^*_{ij}, and updated probability of observing symbol o_k given at state i, which denoted as $b^*_i(o_k)$ are calculated as the follows.

$$a^*_{ij} = \frac{\sum_{s=1}^{m}\sum_{t=1}^{T^s-1}\xi^s_{ij}(t)}{\sum_{s=1}^{m}\sum_{t=1}^{T^s-1}\gamma^s_i(t)} \tag{4.7}$$

$$b^*_i(o_k) = \frac{\sum_{s=1}^{m}\sum_{t=1}^{T^s-1}\gamma^s_i(t)I_{O^s(t)=o_k}}{\sum_{s=1}^{m}\sum_{t=1}^{T^s-1}\gamma^s_i(t)} \tag{4.8}$$

Where $I_{O^s(t)=o_k}$ is 1 if $O^s(t) = o_k$, and is zero otherwise. Replace the elements of A and B by a^*_{ij}, $b^*_i(o_k)$ correspondingly. Repeat above steps (Eq. 4.1 to Eq. 4.8) and updating of A and B until $\sum_{s=1}^{m} Pr(O^s|\theta)$ converges with certain threshold or reaches the maximum iterations set by the user.

4.2.3 Reverse Baum-Welch

The reverse task of training HMM can be described as finding HMM parameters θ^- so that $\sum_{s=1}^{m^-} Pr(O^{-s}|\theta^-)$ is minimized with m^- number of sequences denoted as O^{-s}. Our reverse Baum-Welch is not precisely defined for such a task as the goal, rather it is defined a procedure to incorporate negative examples in order to combine with the parameters trained by standard Baum-Welch algorithm so that the training can be "comprehensive", namely, taking into account of both positive and negative examples.

The E-step is identical to standard Baum-Welch: calculations of α, β, γ, and ξ. And we denoted them as α^-, β^-, γ^-, and ξ^- correspondingly in order to differentiate reverse Baum-Welch from the standard Baum-Welch. The difference is in M-step:

$$a_{ij}^{-*} = \frac{C - \sum_{s=1}^{m^-} \sum_{t=1}^{T^s-1} \xi_{ij}^{-s}(t)}{\sum_{s=1}^{m^-} \sum_{t=1}^{T^s-1} \gamma_i^{-s}(t)} \tag{4.9}$$

$$b_i^{-*}(o_k) = \frac{C - \sum_{s=1}^{m^-} \sum_{t=1}^{T^s-1} \gamma_i^{-s}(t) I_{O^{-s}(t)=o_k}}{\sum_{s=1}^{m^-} \sum_{t=1}^{T^s-1} \gamma_i^{-s}(t)} \tag{4.10}$$

Where C is a constant defined as $C = \sum_{s=1}^{m^-} Length(O^{-s})$. The constant C makes sure that a_{ij}^{-*} and $b_i^{-*}(o_k)$ are all positive values. Because of the usage of C, a_{ij}^{-*} and $b_i^{-*}(o_k)$ need to be normalized to stand for probabilities as the ones in Eq. 4.8:

$$\overline{a_{ij}^{-*}} = \frac{a_{ij}^{-*}}{\sum_{q=1}^{N} a_{iq}^{-*}} \tag{4.11}$$

$$\overline{b_i^{-*}(o_k)} = \frac{b_i^{-*}(o_k)}{\sum_{o_q=1}^{K} b_i^{-*}(o_q)} \tag{4.12}$$

Then the iterations part and stopping criterion are same as standard Baum-Welch algorithm.

The rationale of reverse Baum-Welch is simple. One way to interpret it is to think of the constant C as a uniform pseudo count for emissions and transitions, and the training effect from a negative example O^- for each iteration is to negatively "engrave" this background, namely subtracting C by the actual counts from O^-. This is in contrast to standard Baum-Welch algorithm, which can be considered as "embossing" on an flat empty background. Or one can understand reverse Buam-Welch as there are sequences O^* as opposite of O^- implicitly, with the initial guess of/ last updated model, sequences O^* lead to the updated $\overline{a_{ij}^{-*}}$ and $\overline{b_i^{-*}(o_k)}$ with standard Baum-Welch algorithm, and try to fit model parameters θ^- with O^*.

There are a few points worthy to emphasize. First, it is not necessary to develop a reverse version of Baum-Welch algorithm to find θ^- to minimize $\sum_{s=1}^{m^-} Pr(O^{-s}|\theta^-)$; such minimization can be achieved using gradient descent technique [5]. Rather, the reverse Baum-Welch is a part of CompHMM, not meant to be used stand-alone for the aforementioned minimization. Second, the reverse Baum-Welch cannot guarantee

a monotonous decrease of $\sum_{s=1}^{m^-} Pr(O^{-s}|\theta^-)$, the reason of this directly follows reverse Baum-Welch's second interpretation: the imaged sequences O^* are different for each iteration.

4.2.4 CompHMM

Equipped with the standard Baum-Welch and reverse Baum-Welch, it is now quite straightforward to describe CompHMM. First, let us define the task. Given m^+ number of positive sequences denoted as O^+, and m^- number of negative sequences denoted as O^-. The task is to find a HMM denoted as θ so that $\sum_{s=1}^{m^+} Pr(O^{+s}|\theta) - \sum_{s=1}^{m^-} Pr(O^{-s}|\theta)$ is maximized. Such a HMM is CompHMM, since it is trained comprehensively, taking into account of both positive and negative examples.

With initial parameters θ and positive sequences O^+, obtain its updated transition and emission probabilities:

$$a_{ij}^{*+} = \frac{\sum_{s=1}^{m^+} \sum_{t=1}^{T^s-1} \xi_{ij}^{+s}(t)}{\sum_{s=1}^{m^+} \sum_{t=1}^{T^s-1} \gamma_i^{+s}(t)} \tag{4.13}$$

$$b_i^{*+}(o_k) = \frac{\sum_{s=1}^{m^+} \sum_{t=1}^{T^s-1} \gamma_i^{+s}(t) I_{O^{+s}(t)=o_k}}{\sum_{s=1}^{m^+} \sum_{t=1}^{T^s-1} \gamma_i^{+s}(t)} \tag{4.14}$$

Then, with θ and negative sequences O^-, using reverse Baum-Welch to obtain its updated transition and emission probabilities: a_{ij}^{-*} and $b_i^{-*}(o_k)$. Finally, with a hyperparameter r, chosen by user in range of $(0, 1)$, is used as a weighting factor to combine "embossing" effect from positive examples and "emgraving" effect from negative examples. So the final updating equation is:

$$a_{ij}^* = \frac{r \times a_{ij}^{*+} + (1-r) \times a_{ij}^{*-}}{\sum_{j=1}^{N} r \times a_{ij}^{*+} + (1-r) \times a_{ij}^{*-}} \tag{4.15}$$

$$b_i^*(o_k) = \frac{r \times b_i^{*+}(o_k) + (1-r) \times b_i^{*-}(o_k)}{\sum_{o_k=1}^{K} r \times b_i^{*+}(o_k) + (1-r) \times b_i^{*-}(o_k)} \tag{4.16}$$

For each iteration, update θ according to Eq. 4.15 and Eq. 4.16. Repeat this procedures to update θ, until $\sum_{s=1}^{m^+} Pr(O^{+s}|\theta) - \sum_{s=1}^{m^-} Pr(O^{-s}|\theta)$ converges with certain

tolerance or reaches the maximum iteration set by users. Note that for membership prediction task, the scoring metric for HMM is the log-likelihood of $P(O|\theta)$, which decided majorly by emission matrix B, therefore, it is reasonable to just make $a^*_{ij} = a^{*+}_{ij}$. The experiments in section 4.3 adopted this method.

Even though due to similar reason of reverse Baum-Welch, CompHMM cannot guarantee monotonous increase of $\sum_{s=1}^{m^+} Pr(O^{+s}|\theta) - \sum_{s=1}^{m^-} Pr(O^{-s}|\theta)$ for each iteration, it happens in majority iterations. By experiments, only its few beginning iterations show decrease or oscillation behaviors.

4.3 Experiments and Results

In this section, in order to have a intuitively understanding of the CompHMM, the training validation check of CompHMMs will be presented first. Then, a series of experiments of both synthetic and real data will be performed.

4.3.1 Training Validation Check

The primary goal here is to set up experiment to monitor CompHMM's training phase in order to validate that training procedure indeed increases the separation of positive and negative examples in their measured space, or equivalently, $\sum_{s=1}^{m^+} Pr(O^{+s}|\theta) - \sum_{s=1}^{m^-} Pr(O^{-s}|\theta)$ is maximized.

In the experiment, two hidden Markov models are randomly initialized with state number of 3, alphabetic number of 20. One of them served as positive model, the other served as negative model. Each of them generates 50 sequences, and each sequence has a length of 30. Together they form a dataset containing 50 positive sequences and 50 negative sequences. The initial guess of transition and emission matrices were generated randomly with same size as the positive/negative model. For training, CompHMM is set to have a maximum number of 1500 iterations, tolerance of convergence set to be $1e-6$, and hyper-parameter $r = 0.5$. The experiment shows that the algorithm reaches convergence at 635^{th} iteration. The tracking of $log(P(O^+|\theta))$,

$log(P(O^-|\theta))$ are shown in Figure 4.1, and the tracking of $log(P(O^+|\theta)) - log(P(O^-|\theta))$ is shown in Figure 4.2 as the following.

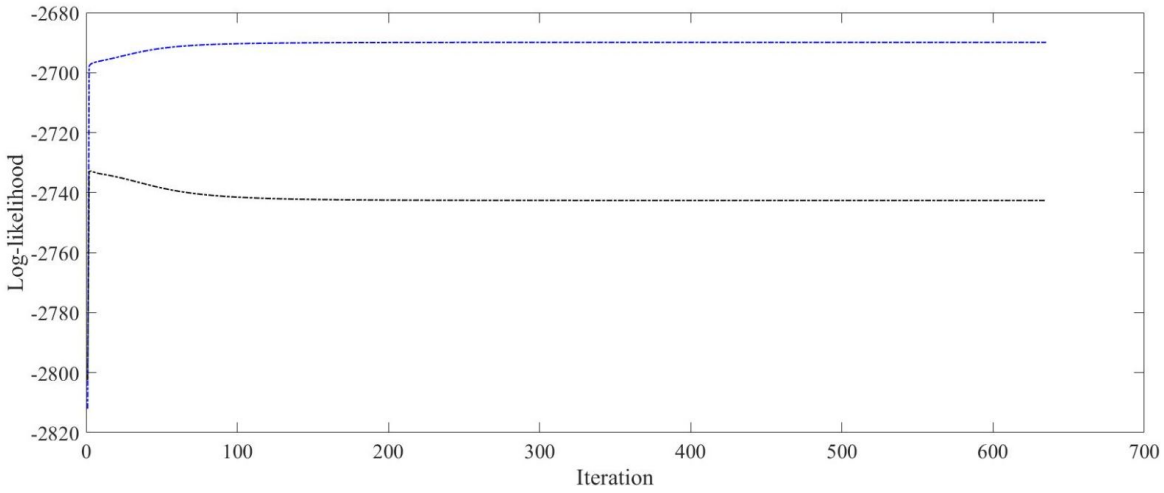

Figure 4.1: Log-likelihood track of positive (blue) and negative (black) examples.

The above results indicate that CompHMM meets the designing purpose, which utilizes both positive and negative data and separates them gradually iteration by iteration, measured by log-likelihood.

4.3.2 Comparison Study on Synthetic Data

After the training validation check of CompHMM, in this subsection, a series of experiments were designed with synthetic data to test CompHMM's performance on membership prediction task.

For each trial of experiments, the synthetic data is generated by 1 randomly generated positive model, and 5/10 negative models, which are generated from positive model by injecting random noise. The injected noise level is varied in order to control the difficulty levels of membership prediction, which are "calibrated" with the positive model being used as ground truth. That is, as the injected noise increases, it is expected to see that the ground truth model has better performance in membership prediction, i.e., recognizing its own sequences out of the sequences generated by the negative

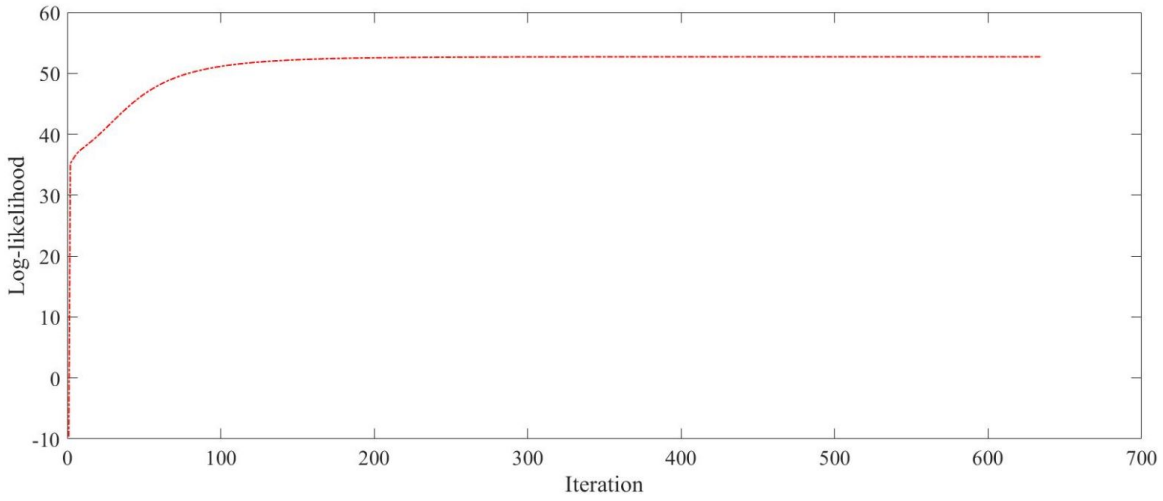

Figure 4.2: Log-likelihood track of the difference between positive and negative examples.

models. The synthetic data contains 120 sequences in total, all with length of 30. 50% of these sequences are used as training and the others are used as testing. Among both training and testing sets, the numbers of positive and negative sequences are equal.

Different training difficulty levels are also tested in terms of the model complexity. The state number is chosen to be 3, 5, and 7, all with alphabetic length of 20 to mimic amino acids in biological field. When state number is fixed, the state number of positive model, negative models, and learning models are the same. The prediction results are evaluated using AUC-ROC as evaluation metric [27]. The sequences in a test set is ranked by their log-likelihood score from the trained model, and a sliding threshold goes from the top to the bottom of the ranked list, one sequence on a time. At each time, the sequences at and above the threshold are predicted as positive and sequences below the threshold are predicted as negative. The predictions are checked with the ground truth label to be determined as true positive or false positive. The ROC curve is a plot of the true positive rate as a function of the false positive rate as the sliding threshold scans through the list. AUC-ROC is the area under the ROC curve, ranging from 0 to 1. A random classifier with no differentiating power would receive a AUC-ROC value of 0.5, and the higher the value the more differentiating

power a classifier has, with AUC-ROC of 1.0 for a perfect classifier.

In this study, each experiment is repeated for 30 times to mitigate random effects. Figure 4.3 shows the performances of various methods from experiments with the synthetic data generated by state number of 5, and 10 negative models. As it can be seen, as the "easiness" increases – negative data easier to be differentiated from the positive data, the performance increases for the ground truth model, as expected. Across the whole range of "easiness", the proposed method performs closely to the ground truth model and significantly outperforms the standard Baum-Welch and the MA method.

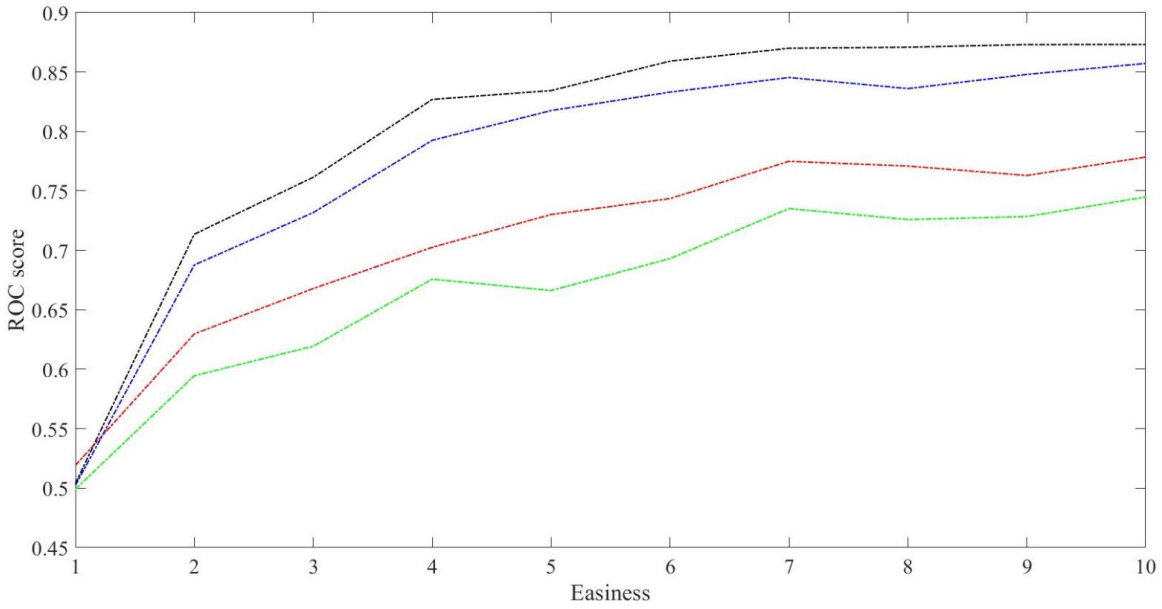

Figure 4.3: AUC-ROC comparison for CompHMM (blue), BW (red), MA (green), and ground truth model (black) for state number = 5, with 10 negative models in different easiness level for membership prediction.

The complete results of the synthetic experiments are summarized in Table 4.1. Like Figure 4.3, Table 4.1 shows that:

1. CompHMM performs strictly better than both BW and MA, and closely to ground truth model.

2. MA performs actually worse than standard BW with the synthetic data especially when negative sequences come from an ensemble of many different negative models, a situation that is more closely mimic to the reality.

Table 4.1: Comparisons for the methods of CompHMM, MA, BW, ground truth in different settings

# of state / # of negative models	CompHMM	MA	BW	ground truth
3/5	0.7752	0.6683	0.7080	0.7987
3/10	0.7758	0.6551	0.7065	0.8003
5/5	0.7121	0.6012	0.6049	0.7455
5/10	0.7100	0.5963	0.6071	0.7446
7/5	0.6549	0.5601	0.5453	0.6922
7/10	0.6590	0.5649	0.5563	0.6990

4.3.3 Applying on Real Data: Plasmodesmata-associated Protein Membership Prediction

To further test the proposed method, CompHMM has been applied to real data. Recall, in previous work [43] shown in chapter 2, a 3-state HMM called PDHMM is used to model Plasmodesmata-located proteins (PDLP)' juxta membrane region, which contains the Plasmodesmata targeting signals, to perform PD vs none-PD classification. Later, 13 additional Plasmodesmata-associated proteins are noticed in literature. Plasmodesmata-associated proteins (PAP) are PD-targeting proteins but have different topology from PDLPs and are very likely from different gene families. Therefore PDHMM, trained on PDLPs, is expected to have lesser performance of predicting PAPs than predicting PDLPs. This can be a good case study of the improvements gained by applying CompHMM.

In this experiment, 8 PDLPs are used as positive sequences for training, and 13 additional Plasmodesmata-associated proteins are used as positive sequences for testing. There are also 365 other type-I transmembrane proteins served as negative

sequences for both training and testing. A 20 folds cross validation performed, and each fold only splits the 365 other type-I transmembrane proteins into negative training set and negative testing set. Positive training's 8 PDLPs and positive testing's 13 Plasmodesmata-associated proteins are used repeatedly for each fold.

The graphic result is shown in Figure 4.4. The AUC-ROC of CompHMM, MA, and BW are 0.7125, 0.6254, 0.6005 respectively, which once again shows ComHMM's significant improvement over both MA and BW.

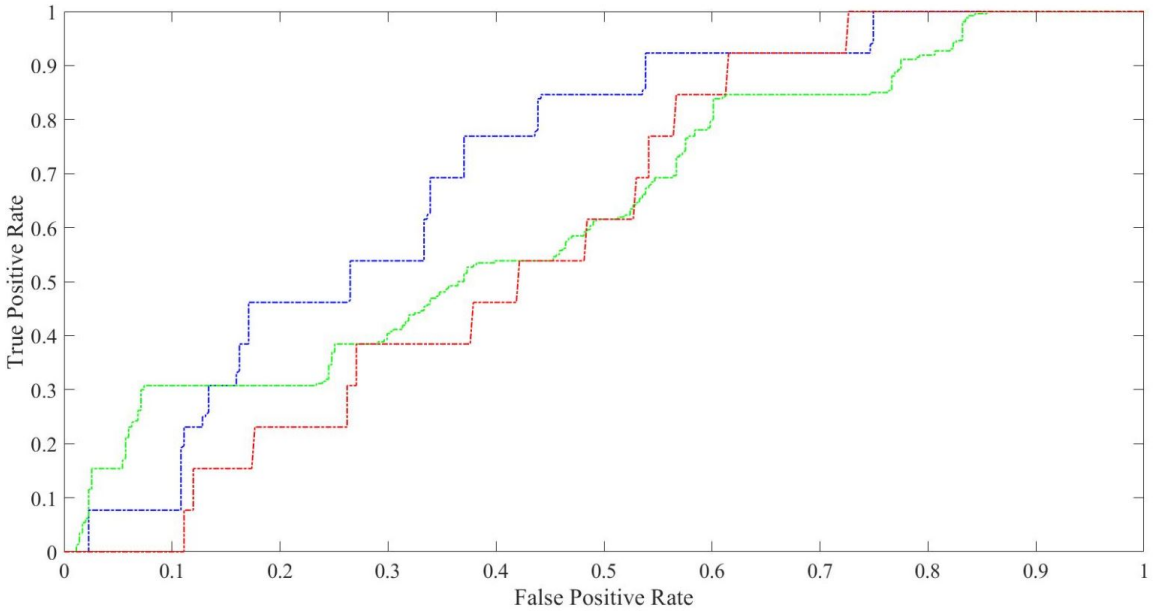

Figure 4.4: ROC curves of CompHMM (blue), BW (red), MA (green) for Plasmodesmata-associated protein membership prediction.

4.4 Discussion and Conclusion

In conclusion, based on the standard Baum-Welch algorithm, I developed a novel training method for HMM that enables us to utilize both positive and negative examples at the same time to form a comprehensive HMM classifier. Experiments in membership prediction task with synthetic data and real data show that the proposed method significantly outperformed both the standard Baum-Welch algorithm

and the MA method, which is a gradient-descent based method to use both positive and negative examples to train HMMs.

While the improvements gained by utilizing negative examples shown in this study are remarkable, a few issues remain to be further investigated in the future. First, the effect of hyper-parameters r, which is used as a weighting factor between positive and negative examples, needs to be studied. Second, as mentioned in section 4.2.3, convergence of CompHMM cannot be guaranteed like standard Baum-Welch algorithm. Therefore, future study of CompHMM will focus on how to improve the algorithm so that the convergence is guaranteed or what can be done practically to mitigate the issue when its convergence cannot be guaranteed theoretically. Moreover, CompHMM will be deployed to the ongoing research of discovering de novo Plasmodesmata-located proteins.

Chapter 5

A NOVEL METHOD FOR INTER-HELIX CONTACT PREDICTION REFINEMENT USING LOCAL 2D TOPOLOGICAL INFORMATION

5.1 Introduction

With the dramatic growth of biological data by continuous developments of wet-lab technologies, various well structured and maintained databases, such as GenBank, Uniprot, and PDB, were developed, are growing and open access to the world. The developments of these databases encourage computer scientists to apply machine learning techniques to biology problems with big data, which becomes an indiscipline field, namely bioinformatics. Among different biology problems, an important one is protein structure prediction. Many successful methods have been developed, and such success majorly replies on deep learning. An example is the recent most notable work AlphaFold2 [35], which is an end-to-end learning and predicts accurate 3D models of proteins. While noticing these success, a sub problem of protein structure prediction still remains challenge, which is inter-helix contact prediction.

Inter-helix contact prediction focus on α-helical integral membrane proteins. Such proteins is important for studying cellular functions. However, membrane proteins' structures are very hard to verified through crystallography [64]. As reported in [13], only 2% to 3% integral membrane proteins have been verified experimentally. Currently, the state-of-art method is DeepHelicon [57]. It is also a deep learning based method, which takes co-evolutionary features as input to the network, and outputs 2D contact matrix. With noticing these success, the trend of applying deep learning techniques in protein structure prediction is clear. On the other hand, while deep learning based methods achieved advanced developments and encouraging performance, not limited in protein structure prediction task, the computational costs of deep learning

increased dramatically [59]. As reported in [41], disregarding the models training, making accurate predictions takes roughly $1K per sequence for AlphaFold. With all these success using deep learning techniques for protein structure prediction, computational cost will become a future challenge at some point, and the computational barrier for the research groups with limited computational power has already been noticed [41].

With no surprise, the more accurate protein structure prediction method we have, the harder to improve in the future. To gain future improvements of protein structure prediction with deep learning, one straightforward and appealing way is to taking the intermediate feature/ final output from existing work as input to new deep neural network, and the training procedure ideally extract useful new features in order to have a better prediction at the end. However, in such way, the size of deep neural network and input features generally increase, which leads the computational barrier higher. Along with the computational barrier, another potential problem is the well known intrinsic nature of deep learning based method, which commonly refers to "black box". This problem prevents researchers from diagnosing the models efficiently, especially when models perform well. In such scenario, to improve the deep learning based model is much more challenging than the original developments.

To tackle the computational barrier issue, I proposed a low computational cost refinement method, which leverages the existing state-of-art method's predictions and exploits features that are not fully captured by the existing method with a novel refinement selection scheme. With realizing the "black box" nature of deep learning based method, the pipeline of the proposed method can be also considered as a "probe" to other deep learning based methods to indicate which features are not fully captured yet, and shed light to better model in the future. Specifically, with an independent dataset, 2D contact structure models are built to extract the features reflecting 2D inter-helix topological patterns, and devise a mechanism to deal with the pitfall of inter-helix prediction refinement, which is an intrinsic challenge to any refinement methods. The cross-validation results show that the prediction from the proposed method outperforms

the state-of-the-art method by a notable degree even without using the refinement selection scheme. By applying the refinement selection scheme, which selects a subset of sequences to refine, the proposed method outperforms the base method significantly in the selected sequences.

The work presented in this chapter is inspired by the PD research problems as described from chapter 2 to chapter 4. In previous chapters, the focuses are developing machine learning algorithms for the tasks of PD targeting signals detection and PD proteins identification. In order to understand PD mechanism comprehensively, PD interactions also needs to be investigated as biologists believe that PDLPs play their regulatory role via interaction at the transmembrane domain [62]. Therefore, the research in this chapter focused on investigating inter-helix contact predictions.

The rest of chapter is organized as the following. First, the proposed method will be described in details by introducing (i) the pitfall of residue contact prediction refinement and fuzziness score, which is a quantitative partial solution to the pitfall. (ii) 2D contact structure model with its feature calculations, (iii) inter-helix prediction refinement. Second, with the understanding of both the proposed method and the pitfall, three experiments will be given. The first experiment is a validation experiment which shows the usefulness of 2D contact structure models' feature for the task of inter-helix prediction refinement. The second experiment examines the idea of fuzziness score as a partial solution to the pitfall. The third experiment will demonstrate the refinement of DeepHelicon's prediction with and without refinement selection scheme. Last, discussion and conclusion are given at the end.

5.2 Methods

In this section, the proposed method, or more specifically, a framework of refining inter-helix prediction results via 2D contact structure model and refinement selection scheme will be described. Before diving into the details, it is helpful to take a global view of the proposed method.

First, the proposed method is designed to refine inter-helix contact prediction results from other method, and achieved better performance evaluated by top-L metric, which is the commonly accepted evaluation criterion by the community. The details of top-L performance evaluation will be described in section 5.3. Second, the successful of this method replies on two key ideas:

- The usage of 2D contact structure models, which are used to generate simple but useful features to capture residue contact 2D topological information.
- A scheme of refinement selection via the idea of fuzziness score, which partially solves the pitfall of residue contact prediction refinement, and be able to select a proper subset of other method's predictions to refine without knowing the ground truth.

In addition to the above preview of the method, it is also helpful to define the major notations used throughout this chapter. Given a sequence numbered as k with length of L_k, a $L_k \times L_k$ binary matrix RC_k is used to denote the contact matrix of the sequence. Each element of RC_k is denoted by $RC_k(i,j) \in \{0,1\}$, which represents non-contact/contact information of the residue pair $< i,j >$ in sequence k. Since the method relies on other method's prediction results, it is worthy to emphasis that RC_k and $RC_k(i,j)$ are always referring to ground truth information since their values are either 0 or 1. In contrast, $\widetilde{RC_k}$ and $\widetilde{RC_k}(i,j)$ are used to refer predicted contact information by other inter-helix contact prediction methods, and their values are from 0 to 1.

Suppose there are total number of K sequences with ground truth information (i.e., RC_k are available for all k). Then, for each inter-helix residue pair in sequence k, a $(2n+1) \times (2n+1)$ matrix can be extracted from RC_k, and denoted by $Fc_k^n(i,j)$, whose elements are $RC_k(a,b)$, where $|i-a| \leq n$ and $|j-b| \leq n$. In the cases of a or b out of the boundary or $< a,b >$ is not an inter-helix pair, 0 values are assigned. Again, since $Fc_k^n(i,j)$ is extract from RC_k, it is a binary matrix. When such matrix extracted from predicted contact map, $\widetilde{Fc_k^n}(i,j)$ is used. The lower case letter "c" in

$Fc_k^n(i,j)$ and "$\widetilde{Fc_k^n}(i,j)$" stands for "including center". In the procedures of building the 2D contact structure model and calculating its features, centers of $Fc_k^n(i,j)$ and $\widetilde{Fc_k^n}(i,j)$ needs to be excluded, $F_k^n(i,j)$ and $\widetilde{F_k^n}(i,j)$ are used to denote such center excluded $Fc_k^n(i,j)$ and $\widetilde{Fc_k^n}(i,j)$ correspondingly. To describe $F_k^n(i,j)$ and $\widetilde{F_k^n}(i,j)$ in a more formal manner, their elements are extracted from $RC_k(a,b)$ and $\widetilde{RC_k}(a,b)$ where $0 < |i-a| \leq n$ and $0 < |j-b| \leq n$. Since they cannot form square matrices, they are reshaped as arrays with size of $(2n+1)^2 - 1$. A graphical representation with $n = 1$ of above procedure is shown in Figure 5.1. For readers reference, the summary of the notations introduced above and several others appeared later in this section are shown in Table 5.1.

Table 5.1: Notations for Inter-Helix Prediction Refinement

Symbols	Explanations
L_k	The length of k^{th} sequence.
RC_k	A $L_k \times L_k$ binary matrix represents the k^{th} sequence's 2D contact map.
$RC_k(i,j)$	RC_k's element, which represents contact/non-contact of residue pair $< i,j >$ of the k^{th} sequence.
$\widetilde{RC_k}$	A $L_k \times L_k$ matrix represents the k^{th} sequence's predicted 2D contact map whose values are from 0 to 1.
$\widetilde{RC_k}(i,j)$	$\widetilde{RC_k}$'s element, which represents contact/non-contact estimation of residue pair $< i,j >$ of the k^{th} sequence. .
$Fc_k^n(i,j)$	$(2n+1) \times (2n+1)$ matrix extracted from RC_k, centered at $< i,j >$.
$F_k^n(i,j)$	A $(2n+1)^2 - 1$ length array, reshaped from $Fc_k^n(i,j)$ by excluding the center.
$\widetilde{Fc_k^n}(i,j)$	$(2n+1) \times (2n+1)$ matrix extracted from $\widetilde{RC_k}$, centered at $< i,j >$
$\widetilde{F_k^n}(i,j)$	A $(2n+1)^2 - 1$ length array, reshaped from $\widetilde{Fc_k^n}(i,j)$ by excluding the center.
$CSM_n(+/-)$	2D contact structure model with parameter n.
$FzScore(*)$	Fuzziness score, which defines the fuzziness level of predictions.
$feature_k^{n+/-}(i,j)$	2D contact structure model's feature extracted from sequence k residue pair $< i,j >$ with parameter n.

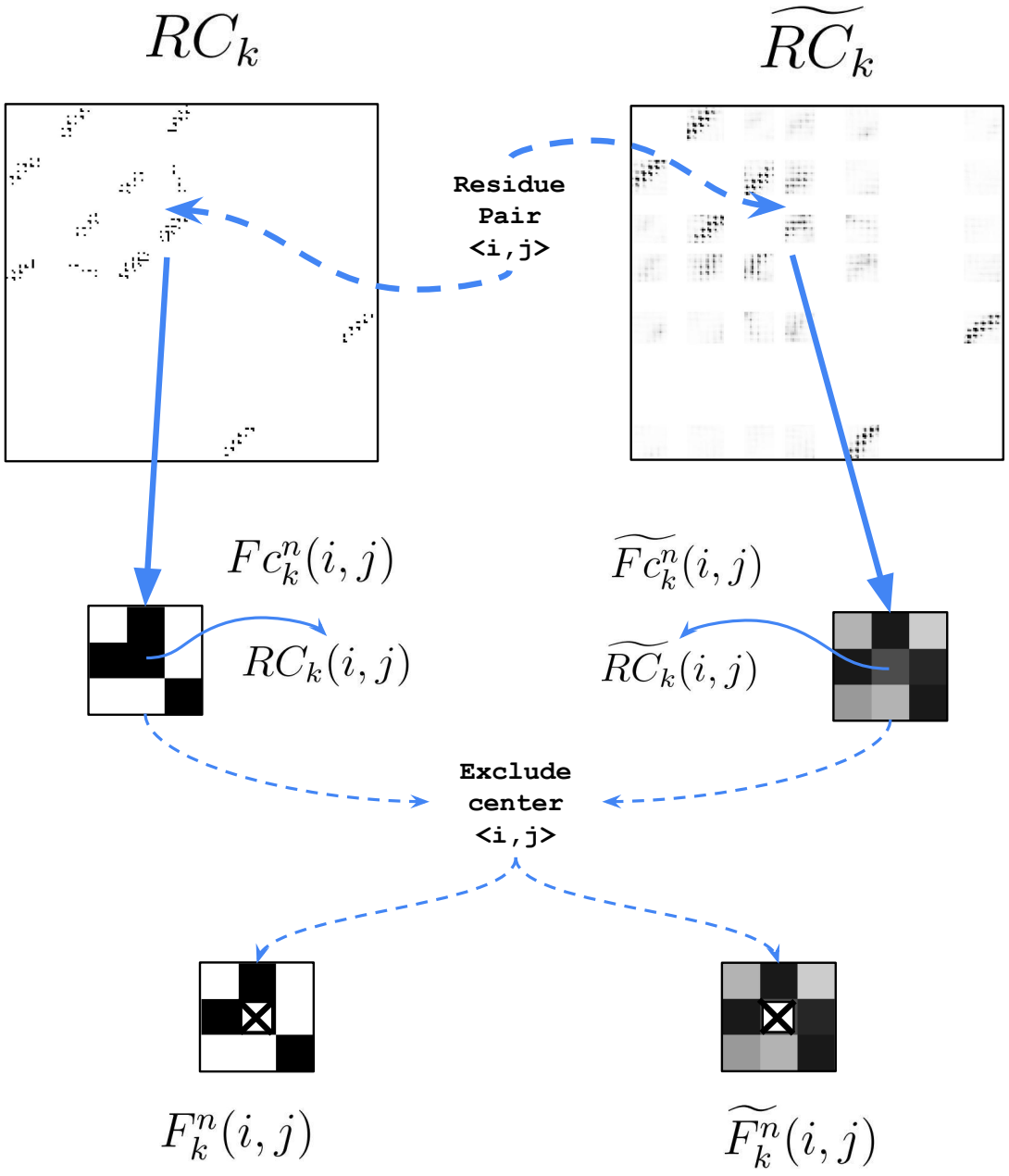

Figure 5.1: Diagram of $Fc_k^n(i,j)$, $\widetilde{Fc_k^n}(i,j)$, $F_k^n(i,j)$, $\widetilde{F_k^n}(i,j)$ extraction from RC_k and $\widetilde{RC_k}$ with $n = 1$.

The rest of this section is organized as the follows. First, the pitfall of residue contact prediction refinement and the idea of fuzziness score will be introduced. Then, 2D contact structure model and its feature calculation will be described in details. Last, with understanding of the 2D contact model with its feature calculation and the idea of fuzziness score, the procedure of training and testing of the proposed method will be given.

5.2.1 Pitfall of Residue Contact Prediction Refinement and Fuzziness Score

Before introducing the technical details of 2D contact structure model and refinement model's training and testing procedures, it is necessary to illustrate the challenges of the refinement, which is described as the pitfall of residue contact prediction refinement in this content.

Though the focus of this chapter is refinement of inter-helix prediction, the pitfall applies to general residue contact prediction also since inter-helix contact is just a subset of residue contact. When one considers refining other results, it automatically suggests to use output of other method as input either directly or indirectly to new machine learning models, and output an improved results. Even though the task of residue contact prediction perfectly filing into this setting (i.e. using predicted residue contact map as input, with ground truth label to train classifier and expect better predictions in testing comparing with origin results), the task is non-trivial as illustrated by the following examples.

Suppose in one sequence denoted as $*$, residue pair $< i, j >$ is correlated with its neighbour $< k, l >$ by conditional probabilities:

$$\begin{aligned} &Pr(RC_*(i,j) = 1 | RC_*(k,l) = 1) \\ &\quad = Pr(RC_*(k,l) = 1 | RC_*(i,j) = 1) = x \end{aligned}$$

Then, given one method's estimations of residue pairs $< i, j >$ and $< k, l >$:

$$\begin{aligned} Pr(RC_*(i,j) = 1 | Method) &= \widetilde{RC_*}(i,j) = y \\ Pr(RC_*(k,l) = 1 | Method) &= \widetilde{RC_*}(k,l) = z \end{aligned}$$

With assuming that the quality of above estimations and the conditional probabilities are same i.e. there are no difference between $Pr(RC_*(i,j) = 1|Method)$ and $Pr(RC_*(i,j) = 1|RC_*(k,l) = 1)$ if they are equal. Moreover, $Pr(\cdot) = 0.5$ indicates no informative estimation between contact versus non-contact. A question can be asked is in which conditions the correlated information between $< i, j >$ and $< k, j >$ as two conditional probabilities will be beneficial to use without considering the ground truth information $RC_*(i,j)$ and $RC_*(k,l)$. If considering $Pr(RC_*(i,j) = 1|RC_*(k,l) = 1)$ and $Pr(RC_*(i,j) = 1|RC_*(k,l) = 1)$ together as the refinement model, the question is equivalent to in which conditions the refinement model should be applied. To answer this question with above problem settings, the following simple examples are introduced.

First, assuming the refinement model is perfect as in the following form, and it is given that the residue pair $< k, j >$ is contact:

$$\begin{aligned} &Pr(RC_*(i,j) = 1|RC_*(k,l) = 1) \\ &\quad = Pr(RC_*(k,l) = 1|RC_*(i,j) = 1) = 1 \\ &RC_*(k,l) = 1 \end{aligned}$$

In this case, it is straightforward that applying the conditional probabilities/ refinement model is always preferred since it gives certain prediction on $RC_*(i,j)$ unless $RC_*(i,j)$ is known.

Then, releasing the assumption of $RC_*(k,l)$: only an estimation of it is given: $Pr(RC_*(k,l) = 1|Method) = \widetilde{RC_*}(k,l) = z$. In such case, applying the conditional probabilities/ refinement model on the original estimation:

$$\begin{aligned} &\widetilde{RC_*(i,j)}_{refine} \\ &= \frac{Pr(RC_*(i,j) = 1|RC_*(k,l) = 1) \times Pr(RC_*(k,l) = 1|Method)}{Pr(RC_*(k,l) = 1|RC_*(i,j) = 1)} \\ &= Pr(RC_*(k,l) = 1|Method) = \widetilde{RC_*(k,l)} = z \end{aligned}$$

With above released assumption, it is favorable to apply the conditional probabilities/ refinement model when $|z - 0.5| > |y - 0.5|$. The reason is that both z and y are estimations from the same method, and when ground truth label is unavailable, the only way to rank the estimations' quality is to see which one has more confidence, which measure as close to 0 or 1 with 0.5 as cutoff. One extreme case can be $z = 0.999$ and $y = 0.499$, clearly, z is an better estimation.

If further releasing the assumption, the refinement model is not perfect i.e. $Pr(RC_*(i,j) = 1|RC_*(k,l) = 1) \neq Pr(RC_*(k,l) = 1|RC_*(i,j) = 1)$. In such condition, $\widetilde{RC_*(i,j)}_{refine} = r \times z$, where $r = Pr(RC_*(i,j) = 1|RC_*(k,l) = 1)/Pr(RC_*(k,l) = 1|RC_*(i,j) = 1)$, is a positive number either greater than 1 or small than 1. Then, applying the conditional probabilities/refinement model gives better estimation when $|rz - 0.5| > |y - 0.5|$.

In real scenario, conditions of applying refinement model is much more complex than these simple cases, and cannot be generalized in such manners. For examples, $< i, j >$ is generally correlated with many neighbors of it and estimations' qualities of these neighbours can vary a lot, the correlations between $< i, j >$ and its neighbours may not be modelled as conditional probabilities with single condition, and even may not be able to be modelled in form of conditional probabilities. Though there are big discrepancy between real scenario and above simple ones, it is still worthy and helpful to pinpoint important facts from above examples as the pitfall of residue contact prediction refinement:

- Even with perfect refinement model, applying refinement model may not be the preferred option.
- The conditions of applying refinement model depends on:
  - The quality of refinement model.
  - The relative qualities between the refining target and support information to the refinement model from original predictions.

With realizing the pitfall of residue contact prediction refinement, a quality measure of predictions called fuzziness score is proposed and used to avoid the pitfall partially. In fuzziness score, it is assumed that other methods' estimations are in range from 0 to 1 and 0.5 is the cutoff between positive and negative predictions. The fuzziness score for a particular estimation $\widetilde{RC_*(i,j)}$ is calculated as follows:

$$fzScore(\widetilde{RC_*(i,j)}) = \min(1 - \widetilde{RC_*(i,j)}, \widetilde{RC_*(i,j)}) \tag{5.1}$$

When $\widetilde{RC_*(i,j)} = 0.5$, then $fzScore(\widetilde{RC_*(i,j)}) = 0.5$ to indicate the highest fuzzy level. When $\widetilde{RC_*(i,j)} = 1$ or 0, $fzScore(\widetilde{RC_*(i,j)}) = 0$ to indicate the lowest fuzzy level.

The hypothesis of inter-helix contact refinement, which is verified experimentally in section 5.3, is that given estimations $\widetilde{RC_*}(i,j)$ and $\widetilde{F_*^n}(i,j)$, only when the fuzziness score of $\widetilde{F_*^n}(i,j)$ is less than the fuzziness score of $\widetilde{RC_*}(i,j)$ in a certain degree, the refinement model is more likely to be useful to give a better estimation of $RC_*(i,j)$. In practice, the fuzziness score of $\widetilde{F_*^n}(i,j)$ is just the average of each element's fuzziness score as follows.

$$fzScore(\widetilde{F_*^n}(i,j)) = \frac{\sum_{0<|a-i|\leq n, 0<|b-j|\leq n} fzScore(\widetilde{RC_*}(a,b))}{(2n-1)^2 - 1} \tag{5.2}$$

5.2.2 2D Contact Structure Model

With a global view of the proposed method and its notations introduced in the beginning of this section, the procedure of building the 2D contact structure model is straightforward. To build 2D contact structure models, a training set with ground truth label is needed. Given K number of training sequences, $F_k^n(i,j)$ can be extracted from RC_k for all k and all inter-helix residue pair $< i,j >$. A 2D contact structure model has one parameter n. n is used to define the model's length as $(2n+1)^2 - 1$, which is same as the length of $F_k^n(i,j)$. One 2D contact structure model contains

two sub-models, one for contact inter-helix residue pairs, one for non-contact inter-helix residue pairs, and are denoted by $CSM_n(+)$ and $CSM_n(-)$ respectively. The calculations of 2D contact structure model with parameter n is shown as the following.

$$CSM_n(+) = \frac{\sum_{k=1}^{K} \sum_{i,j} F_k^n(i,j) \cdot RC_k(i,j)}{\sum_{k=1}^{K} \sum_{i,j} RC_k(i,j)} \tag{5.3}$$

$$CSM_n(-) = \frac{\sum_{k=1}^{K} \sum_{i,j} F_k^n(i,j) \cdot (1 - RC_k(i,j))}{\sum_{k=1}^{K} \sum_{i,j} (1 - RC_k(i,j))} \tag{5.4}$$

Explaining 2D contact structure model by words, $CSM_n(+/-)$ is an array with length of $(2n+1)^2 - 1$. The elements of $CSM_n(+)$ represent the contact frequencies position wise of all contact inter-helix residue pairs' neighbours, which are bounded by $(2n+1) \times (2n+1)$ box. In contrast, the elements of $CSM_n(-)$ are the contact frequencies position wise of all non-contact inter-helix residue pairs' neighbours. With $CSM_n(+/-)$, its corresponding features of other method's predicted inter-helix results can be calculated by $\widetilde{F_k^n}(i,j)$ as the following.

$$feature_k^{n+}(i,j) = \widetilde{F_k^n}(i,j) \cdot CSM_n(+)^T \tag{5.5}$$

$$feature_k^{n-}(i,j) = \widetilde{F_k^n}(i,j) \cdot CSM_n(-)^T \tag{5.6}$$

Here, $\widetilde{F_k^n}(i,j)$ and $CSM_n(+/-)$ are in dimension of $[(2n+1)^2 - 1] \times 1$, therefore, $feature_k^{n+/-}(i,j)$ are in dimension of 1×1. The values of $feature_k^{n+/-}(i,j)$ can be interpreted as a probability-like scores, which indicate how likely $RC_k(i,j) = 1$ or 0 given $CSM_n(+/)$ and inter-helix residue pair $< i,j >$'s neighbours contact estimations $\widetilde{F_k^n}(i,j)$.

5.2.3 Inter-helix Prediction Refinement

With introduction to the notations, pitfall of residue contact refinement, and detail descriptions of 2D contact structure model with its feature calculation, the procedure of inter-helix prediction refinement can be easily described as follows in the order of feature preparation, model training and testing, and refinement selection.

First, with $CSM_n(+/-)$ built by independent training dataset with $n = 1...N$, features to the refinement model, which is chosen as random forests, need to be defined. The feature to random forests classifier of sequence k, at inter-helix residue pair$< i, j >$, with parameter N, is denoted as $RF_feature_k^N(i,j)$ in the following form:

$$\begin{aligned} RF_feature_k^N(i,j) = concat[&feature_k^{1+}(i,j), feature_k^{1-}(i,j), \\ &feature_k^{2+}(i,j), feature_k^{2-}(i,j), \\ &..... \\ &feature_k^{N+}(i,j), feature_k^{N-}(i,j), \widetilde{Fc_k^N}(i,j)] \end{aligned} \tag{5.7}$$

Then, with $RF_feature_k^N(i,j)$ ready for each inter-helix residue pair of each sequence, training and testing procedures are standard if one decides to refine all predictions disregarding the pitfall. Moreover, since the proposed method is designed to refine other's predictions, this automatically assumes the predictions are trustworthy in a certain degree. Therefore, in practice, users can choose only refining top-$(S \times L)$ inter-helix residue pairs in training and testing, where S is an integer and L is the total length of the helices in a particular sequence. In experiments, S is chosen to be 3.

Last, as the pitfall of residue contact prediction refinement stated, there is no guarantee that the refinement model can give better prediction even the refinement model is perfect. Therefore, it is users' option to decide whether to apply our refinement selection techniques.

The refinement selection scheme have two different settings. One is selecting particular residue pairs. The other is selecting particular sequences. The first setting gives a clear statistical meaningful boundary between refine vs no refine residue pairs. However, the gain from the first setting does not reflect to top-L evaluation clearly. Therefore, a second refinement setting is offered to only refine a subset of testing sequences.

The first setting of refinement selection is the following. For each inter-helix residue pair $< i, j >$ and its neighbours bounded by a $(2N + 1) \times (2N + 1)$ box, a

pair fuzziness scores can be calculated to measure the fuzziness level of $\widetilde{RC_*}(i,j)$ and $\widetilde{F_*^N}(i,j)$. Their difference is denoted by $\partial fzScore_*^N(i,j)$ and calculated as:

$$\partial fzScore_*^N(i,j) = fzScore(\widetilde{RC_*}(i,j)) - fzScore(\widetilde{F_*^N}(i,j)) \tag{5.8}$$

When $\partial fzScore_*^N(i,j)$ is higher than a threshold, it indicates that $\widetilde{RC_*}(i,j)$ is more fuzzy than $\widetilde{F_*^N}(i,j)$, and applying refinement model with the neighbours of $< i,j >$ as input is more likely to give better prediction at $< i,j >$. The threshold of $\partial fzScore_*^N(i,j)$ can be determined by the following five steps.

1. With trained refinement model and an independent dataset, calculate refined and original predictions' absolute errors as $refine_Error_*(i,j)$ and $Error_*(i,j)$ for each $< i,j >$.

2. Define $\partial Error_*(i,j) = Error_*(i,j) - refine_Error_*(i,j)$

3. Sort all $\partial fzScore_*^N(i,j)$ in ascending order, and sort all $\partial Error_*(i,j)$ in same order of $\partial fzScore_*^N(i,j)$.

4. With sorted $\partial fzScore_*^N(i,j)$ as x-axis, do cumulative summation of the sorted $\partial Error_*(i,j)$ as y-axis. Plot the curve.

5. Find the curve's minimum point and corresponding value in x-axis, which is the threshold.

Then, with the threshold, refinement method applies to residue pair $< i,j >$ when the corresponding $\partial Error_*(i,j)$ is greater than the threshold. Experiment of this first setting is demonstrated in section 5.3.3.

For the second setting of refinement selection, the procedures are similar. Since the setting selects sequences instead of residue pairs, fuzziness score for each sequence need to be defined. It is denoted as $\overline{fzScore_k}$, which is the average of $fzScore(\widetilde{F_k^N}(i,j))$ for the top-$(S \times L)$ inter-helix residue pairs in sequence k. And the similar five steps procedure is shown as follows.

1. With trained refinement model and an independent dataset, for each sequence k in the dataset, evaluate refined and original predictions by a performance metric as $refine_Perform_k$ and $Perform_k$.

2. Define $\partial Perform_k = refine_Performr_k - Perform_k$

3. Sort all $\overline{fzScore_k}$ in ascending order, and sort all $\partial Perform_k$ in same order of $\overline{fzScore_k}$.

4. With sorted $\overline{fzScore_k}$ as x-axis, do cumulative summation of the sorted $\partial Perform_k$ as y-axis. Plot the curve.

5. Find the curve's minimum point and corresponding value in x-axis, which is the threshold.

Here, one requirement of performance metric is that $Perform_k$ needs to be a scalar, and higher value indicates better performance. In this work, Matthews correlation coefficient is chosen as performance metric for this second setting of refinement selection scheme. A brief explanation for the second setting of refinement is the following: with high $\overline{fzScore_k}$, the origin prediction of sequence k is more likely to be worse than the ones with low $\overline{fzScore_k}$, and is more likely to gain performance improvement by applying the refinement model, and the corresponding $\partial Perform_k$ is expected to be higher (i.e. greater than 0). Experiment of the second setting is shown in section 5.3.4

5.3 Experiments and Results

In this section, the dataset used will be described in details first. Then, three sets of experiments and their corresponding results will be shown. The first experiment is served as validation purpose for 2D contact structure model and demonstrates the upper bound of this refinement method. The second experiment will validate the hypothesis of the pitfall as described in last section, and examine the first setting of refinement selection scheme with DeepHelicon's results. The third experiment will

show the results of the proposed method refining DeepHelicon's predictions with and without refinement selection technique.

Before experiments and results, the evaluation criterion need to be explained clearly first. For the problem of inter-helix predictions, top-L evaluation criterion is the commonly accepted metric. The term L refers to the total length of particular sequence's α helices. In general, top L/1, L/2, L/5, and L/10 are chosen as decision boundary's cutoff. For example, with a given multi-path transmembrane proteins, the total length of its α-helices is X, the top-L/5 evaluation means ranking each prediction scores in descending order, and considering the top $X/5$ predictions as predicted contact inter-helix residue pairs, remaining as predicted non-contact inter-helix residue pair. For comparison purpose, top-L evaluation is adopted with L/1, L2, L/5, L/10 to evaluate precision, recall, F_1 score, $F_{0.35}$ score, and Matthews correlation coefficient (MCC). With multiple sequences in the testing set, average of precision, recall, F-scores, and MCC are computed as final results. For readers convince, the evaluation of F-scores and MCC are the followings.

$$F_\beta = (1+\beta)^2 \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{5.9}$$

$$MCC = \frac{TP \cdot TN + FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{5.10}$$

5.3.1 Dataset

The datasets are from the work done by DeepHelicon [57]. There are three different datasets: TRAIN, TEST, and PREVIOUS. They contains 165, 57, and 44 multi-path transmembrane proteins respectively. TRAIN and TEST datasets were extracted from PDBTM database [40] by DeepHelicon group. PREVIOUS is a combined dataset from TMhhcp [61] and MemConP [31].

Here, the TRAIN dataset is used as the independent dataset to build 2D contact structure model with different parameters n. And combing the TEST and PREVIOUS datasets as 101 sequences in total to evaluate the proposed method by 10 fold cross

validation. The combined dataset is called DATA(101) hereafter. Moreover, both TRAIN and DATA(101) are recorded in Appendix C for readers' reference.

5.3.2 2D Contact Structure Model Validation

In this experiment, the main goal is to demonstrate the usefulness of 2D contact structure model and its features in task of inter-helix prediction. In addition to the purpose of validation, this experiment also computes the upper bound with the given data. Again, this experiment and later two used TRAIN dataset to build 2D contact structure models, and used DATA(101) to perform 10 fold cross validation. The classifier used is random forests with 500 trees. The hyper-parameter N is chosen to be 7. The feature in this experiment is slightly different from the one in Eq. 5.7. As described in Eq. 5.7, the $RF_feature_k^N(i,j)$ is concatenation of $\widetilde{Fc_k^N}(i,j)$ and 2D contact structure models' features $feature_k^{n+/-}(i,j)$ with $n = 1...N$. Recall in Eq. 5.5, only estimation of the neighbours of residue pair $< i,j >$ is used to calculate 2D contact structure models' features. Here, instead of using the estimations $\widetilde{F_k^n}(i,j)$, ground truth information $F_k^n(i,j)$ is used to calculate the feature according to Eq. 5.5 as the following.

$$feature_k^{n+}(i,j) = F_k^n(i,j) \cdot CSM_n(+)^T$$
$$feature_k^{n-}(i,j) = F_k^n(i,j) \cdot CSM_n(-)^T$$

The results are shown in Table 5.2. With 2D contact structure model's feature calculated from ground truth contact map, top-L performances are strictly better than the original results cross different Ls. This validates the usefulness of applying 2D contact structure model to such task. However, in reality, ground truth contact map is not available and only an estimation of it is given. In such case, the performance will drop but still give better results comparing with the original inter-helix contact predictions as shown later in this section.

Table 5.2: Top-L Comparisons of the refinement method (with 2D contact structure model features calculated from ground truth contact map) and DeepHelicon without refinement selection on DATA(101)

L	Methods	Precision	Recall	F_1	$F_{0.35}$	MCC
L/1	Our Method	**74.73%**	**53.35%**	**59.99%**	**70.50%**	**61.10%**
	DeeppHelicon	62.14%	44.86%	50.41%	58.80%	50.76%
L/2	Our Method	**90.02%**	**34.06%**	**47.34%**	**74.10 %**	**53.45%**
	DeeppHelicon	76.04%	28.21%	39.81%	62.63%	44.62%
L/5	Our Method	**96.43%**	**15.06%**	**25.15%**	**57.69%**	**36.61%**
	DeeppHelicon	84.98%	12.70%	21.68%	50.71%	31.72%
L/10	Our Method	**98.35%**	**7.86%**	**14.19%**	**41.06%**	**26.63%**
	DeeppHelicon	88.79%	6.69%	12.30%	36.68%	23.55%

5.3.3 Fuzziness Score Validation

The goal of this experiment is to validate the hypothesis described in section 5.2.1, and test the procedures 5-steps procedure described in section 5.2.3 for the first setting of refinement selection. The setting of this experiment is same as the previous one expect that the evaluation criterion is absolute error, which measure performance per residue pair, instead of top-L, which measures performance per sequence, and feature used are all calculated from DeepHelicon's predictions.

Let X to denote the collection of original prediction results from DeepHelicon, X_r to denote the collection of inter-helix prediction refinement results, and Y denote the ground truth label. X, X_r, Y are arrays with length equal to total inter-helix residue pairs in DATA(101). Then, define:

$$Error = |X - Y|$$

$$Error_r = |X_r - Y|$$

$$\partial Error = Error - Error_r$$

Then, let ∂e be an element of $\partial Error$. When ∂e is greater than 0, original error is greater than the refinement's, which is the favourable case. Otherwise, it indicates refinement model should not be used for the residue pair. Plotting curve by follow the

5-steps procedure described in section 5.2.3 with $N = 7$, the results are shown in Figure 5.2. The curve is in a "smile" shape, and the red "X" indicates the minimum point of the curve (the threshold). The left region of the threshold goes down almost monotonically indicating majority ∂e is less than 0. The right region of the threshold goes up almost monotonically indicating majority ∂e is greater than 0. This validates the hypothesis, and examines the 5-steps procedure in first setting of refinement selection scheme.

5.3.4 Inter-Helix Prediction Refinement and Refinement Selection Scheme

This experiment contains four parts. First part is the evaluation without the second setting of refinement selection scheme. The second part shows the evaluation results when refinement selection is used, which selects 67 out of 101 sequences to refine. The third part demonstrates the complement results of the second part, which shows the performance of the rest 34 sequences. The last part is a comparison of the improvements in first and second part. This comparison highlights the method's true performance gain, which is diluted by the unpreferred sequences when refinement selection scheme is not used.

Again, the experiment used random forests with 500 trees, training and testing on top-$3 \times L$ inter-helix residue pairs in 10 fold cross validation. Here, the performance metric described in section 5.2.3 is the average of MCC cross different Ls. Since this experiment is more complex than the early two, the overall pipeline is shown in Figure 5.3 to readers' convenience.

The four parts' results of this experiment are shown in Table 5.3 to Table 5.6 correspondingly. In Table 5.3, the proposed method gains around $1\% - 2\%$ improvements in precision and around 1% improvements in recall without the selection scheme. In Table 5.4, for the 67 selected sequences, the improvements of the proposed method is higher. Moreover, even for the 34 unpreferred sequences, the method performs almost as same as DeepHelicon's results as shown in Table 5.5. In Table 5.6, the comparison results highlight that both first and second parts of this experiment gain improvements

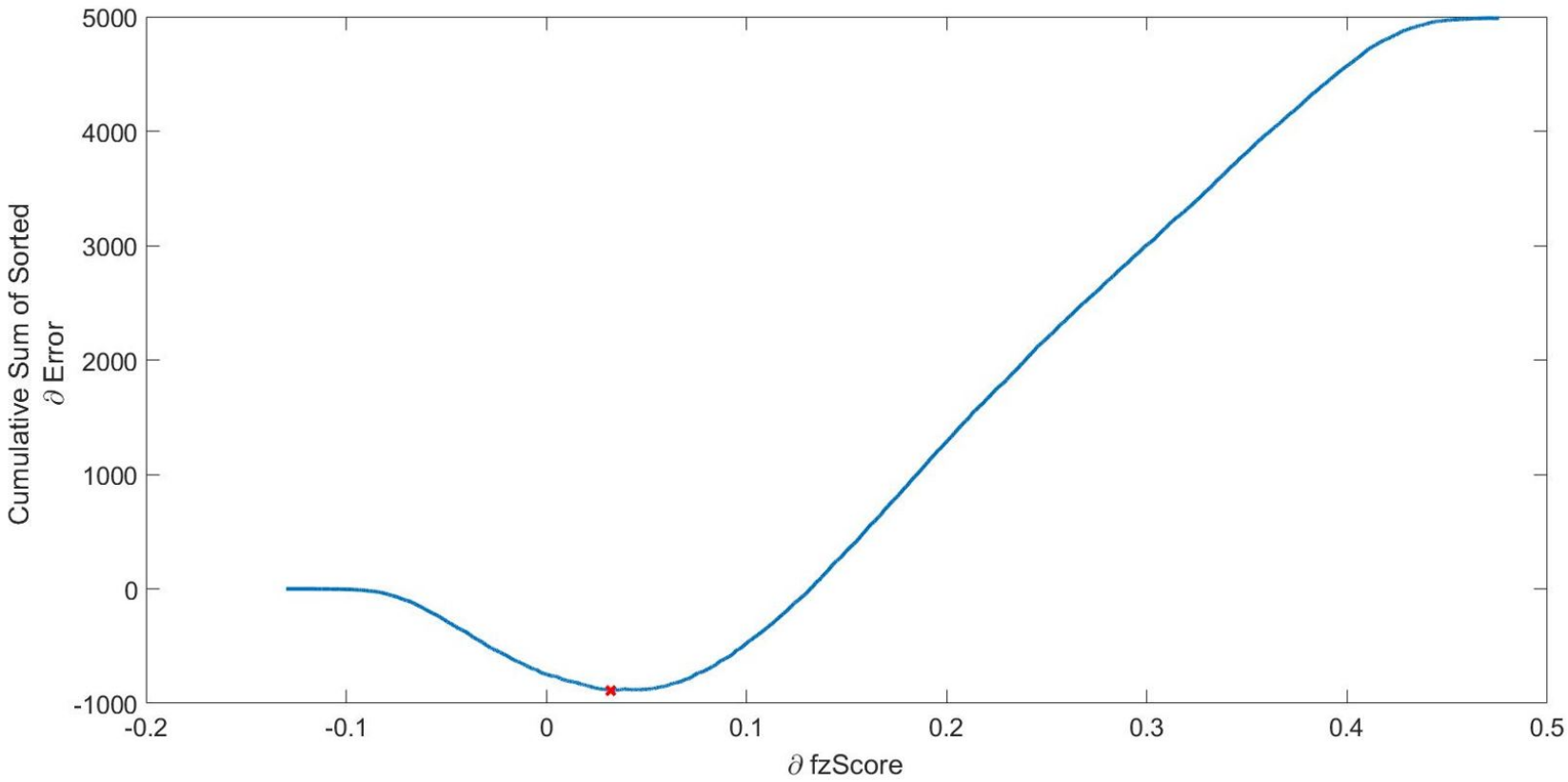

Figure 5.2: Fuzziness Score Validation

in precision, recall, F_1, $F_{0.35}$, and MCC cross different Ls, and show the differences between the true improvements and diluted improvements of the proposed method.

Table 5.3: Top-L comparisons of the refinement and DeepHelicon without refinement selection on DATA(101)

L	Methods	Precision	Recall	F_1	$F_{0.35}$	MCC
L/1	Our Method	**64.10%**	**45.46%**	**51.26%**	**60.42%**	**51.89%**
	DeeppHelicon	62.14%	44.86%	50.41%	58.80%	50.76%
L/2	Our Method	**77.71%**	**28.89%**	**40.39%**	**63.74 %**	**45.52%**
	DeeppHelicon	76.04%	28.21%	39.81%	62.63%	44.62%
L/5	Our Method	**87.01%**	**13.49%**	**22.53%**	**51.87%**	**32.79%**
	DeeppHelicon	84.98%	12.70%	21.68%	50.71%	31.72%
L/10	Our Method	**89.48%**	**6.77%**	**12.40%**	**36.75%**	**23.70%**
	DeeppHelicon	88.79%	6.69%	12.30%	36.68%	23.55%

Table 5.4: Top-L comparisons of the refinement results and DeepHelicon with refinement selection (67 out of 101) on DATA(101).

L	Methods	Precision	Recall	F_1	$F_{0.35}$	MCC
L/1	Our Method	**63.90%**	**44.31%**	**50.37%**	**60.02%**	**51.08%**
	DeeppHelicon	61.65%	42.95%	48.78%	57.97%	49.36%
L/2	Our Method	**76.96%**	**28.08%**	**39.29%**	**62.77%**	**44.55%**
	DeeppHelicon	75.00%	26.66%	37.87%	61.00%	42.97%
L/5	Our Method	**86.60%**	**13.26%**	**22.03%**	**51.09%**	**32.32%**
	DeeppHelicon	84.53%	12.25%	20.86%	49.44%	30.97%
L/10	Our Method	**89.88%**	**6.70%**	**12.26%**	**36.52%**	**23.61%**
	DeeppHelicon	88.79%	6.51%	11.94%	35.77%	23.14%

5.4 Discussion and Conclusion

There are several points worthy for readers attention. First, the machine learning method used here is not necessary to be random forest. Any method is proper

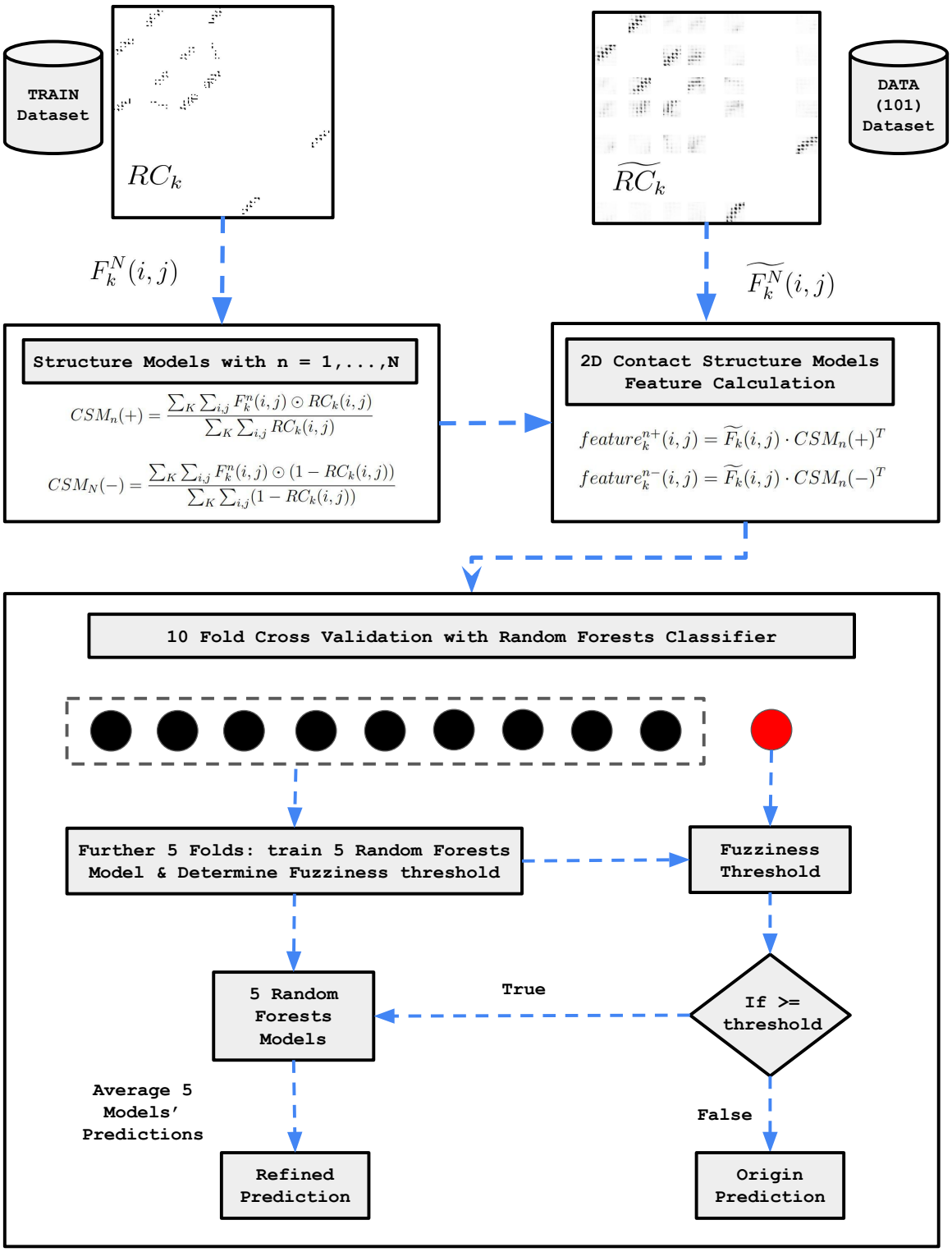

Figure 5.3: Pipeline of inter-helix contact prediction refinement.

Table 5.5: Top-L comparisons of the refinement results and DeepHelicon complement to refinement selection (34 out of 101) on DATA(101)

L	Methods	Precision	Recall	F_1	$F_{0.35}$	MCC
L/1	Our Method	**63.82%**	**46.56%**	**51.93%**	**60.37%**	**52.34%**
	DeeppHelicon	63.61%	46.37%	51.71%	60.16%	52.22%
L/2	Our Method	**78.59%**	**30.04%**	**41.90%**	**64.95 %**	**46.80%**
	DeeppHelicon	78.55%	30.01%	41.87%	64.92%	46.78%
L/5	Our Method	**86.64%**	**13.65%**	**23.00%**	**52.39%**	**33.09%**
	DeeppHelicon	86.53%	13.51%	22.80%	52.15%	32.90%
L/10	Our Method	88.29%	6.85%	12.54%	36.85%	23.67%
	DeeppHelicon	**89.05%**	**6.92%**	**12.66%**	**37.19%**	**23.90%**

Table 5.6: Comparison of improvements with and without refinement selection scheme

L	Methods	Precision	Recall	F_1	$F_{0.35}$	MCC
L/1	with selection	**2.2456%**	**1.3601%**	**1.5903%**	**2.0594%**	**1.7215%**
	without selection	1.5790%	0.9784%	1.1444%	1.4569%	1.2281%
L/2	with selection	**1.9634%**	**1.4135%**	**1.4233%**	**1.7637%**	**1.5819%**
	without selection	1.4023%	0.9874%	1.0085%	1.2572%	1.1165%
L/5	with selection	**2.0764%**	**1.0019%**	**1.1676%**	**1.6519%**	**1.3543%**
	without selection	1.4339%	0.7089%	0.8381%	1.1807%	0.9619%
L/10	with selection	**1.0987%**	**0.1888%**	**0.3240%**	**0.7427%**	**0.4716%**
	without selection	0.3893%	0.0936%	0.1575%	0.3321%	0.2090%

as long as its outputs can be interpreted as probability. Second, for refinement selection scheme's first setting, as mentioned in section 5.3, the left/right region decreases/increases only almost monotonically even though it looks like monotonically in Fig 5.2. This implies the first setting cannot perfectly separate the cases of refine vs non-refine. Third, in the second setting of refinement selection scheme, MCC is not the only choice, but others such as AUC-ROC, does not perform well when top-L evaluation is used. Last, it is encouraged to change the features to proper others, and replace the DeepHelicon to interested ones with refinement selection scheme, as a

way to probe the interesting deep learning methods with candidate features for future improvements. Here, the usage of refinement selection scheme is highly recommended since the contribution of an important but not fully captured/missing feature can be easily diluted by improper testing cases.

In conclusion, a low computational cost, genetic enough method is proposed for inter-helix contact prediction refinement. The proposed method shows notable improvements by top-L evaluation criterion. The success is achieved by exploiting features that are not fully captured by the current state-of-art method, and the development of refinement selection scheme via the idea of fuzziness score, which offers a partial solution to the intrinsic challenge of any refinement method. In addition to current work, there are several components of the proposed method can be improved further. The first component is the 2D contact structure model. At current stage, the developed 2D contact structure model with its feature calculation is still simple, which can lead losing important spatial information of inter-helix patterns. Second, as mentioned, the refinement selection scheme is only a partial solution. To investigate the challenge of refinement further both practically and theoretically, more experiments cross different domains are necessary.

Chapter 6

SUMMARY AND FUTURE WORK

6.1 Summary

In summary, the main research goal of this dissertation is to develop machine learning techniques to detect signals in protein sequences that are involved in subcellular localization and protein interactions in general and to apply these techniques to aid biologists studying Plasmodesmata proteins in specific. This research contains three topics:

- Identifying PD targeting proteins.
- Detecting PD targeting signals.
- Understanding PD interaction in TMD.

Due to the limitation of data quantity, and more importantly the data quality, existing computational tools cannot handle these tasks well, which motivates the development of novel machine learning techniques with the focus on leveraging known biological knowledge and relevant information. Specifically, these novel tools/algorithms are listed as follows.

- A hybrid classification model for PD targeting proteins identification, which is shown in chapter 2, and published in [44].
- Train HMM with partial label data, which is shown in chapter 3, and published in [46].
- CompHMM: train HMM with both positive and negative examples, which is shown in chapter 4, and published in [45]

- A novel refinement method for inter-helix contact prediction, which is described in chapter 5, and will be submitted to Bioinformatics for publication.

Along with the developments of these novel machine learning techniques on computational side, many encouraging achievements have also been made on biological side as well. These include but not limited to identifying multiple new PD proteins, successfully discovering majority PD-targeting signals in PDLPs, and correcting false predicted PD-targting signals in PDLP7.

In addition to computational tasks in PD localization and interaction, the developed machine learning techniques also have great potential in other domains, where identification of subtle signals with limited training examples is required. For instance, the two novel training methods for HMMs can be applied whenever sequential data is involved, such as natural language processing, communication system, and even in financial market predictions. The novel refinement method can be applied for contact map reconstruction for general protein structure prediction rather than just for inter-helix contact.

6.2 Future Work

First, the study of PD interactions is still in its infancy. The refinement method described in chapter 5 is a first attempt at leveraging existing data and methods to predict inter-helix contact, which as biologists believe plays a key role in PD regulation. Although the method has shown encouraging improvement in prediction accuracy, at current stage, it is not yet ready to apply the method to predict PD-associated proteins and their interacting partner, due to the following:

- Majority of data-driven machine learning approaches – including the one used in the refinement method – focus on the contact inside one protein but not cross different proteins.

- Ab initio methods such as TMDOCK [47] and PREDDIMER [50] consider protein folding only in helices but not other domains, with an implicit assumption that

two helices interact with each other, although in reality that assumption is not given.

In order to apply the refinement method for PD interactions:

- More biological knowledge is needed, such as amino acid composition, which can be captured and leveraged by 2D contact structure model.
- Hacking inter-helix prediction methods properly is preferred as a way to mimic helix contact cross different proteins.

Second, as more PD proteins were identified, key residues of PD-targeting signals have not yet been discovered successfully, even using the improved PDHMM. One reason is that the newly discovered PD proteins have very different JMe regions comparing with the JMe in PDLPs as biologists verified that some low ranked proteins by PDHMM are actually PD-associated proteins. To overcome this challenge, using PDHMM as template but with new data and collaborating with additional biological knowledge is necessary. For example, as discovered PD-associated proteins have very different JMe and multiple sequences alignment technique cannot reveal conserved patterns, building multiple PDHMMs with training on different clusters of JMe is necessary. Also, preliminary sequence analysis either from other computational tools or biological domain knowledge is essential in PDHMM topology construction step. With multiple PDHMMs training on different JMe regions ready, PD-targeting signals decoding can be performed. With a larger set of decoding results for PD-associated proteins, the searching space of the key residues of PD-targeting signals can be narrowed down dramatically.

Lastly, the model construction pipeline of PDHMM has potentials to be modified and extended for other subcellular localization predictions as well since wet-lab experiments showed that by PD-targeting signal deletion/mutation, PDLPs located from PD to other locations, such as ER and PM. Then, with deeper biological insight of both PD-targeting and other localization targeting signals, an unified computational

framework for multi-class subcellular localization prediction can be developed, which can not only be used as computational prediction tool but also severed as decoding tool to reveal important signals and exploit useful knowledge for different subcellular localizations.

BIBLIOGRAPHY

[1] José Juan Almagro Armenteros, Casper Kaae Sønderby, Søren Kaae Sønderby, Henrik Nielsen, and Ole Winther. Deeploc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, 33(21):3387–3395, 2017.

[2] Pantelis G Bagos, Theodore D Liakopoulos, and Stavros J Hamodrakas. Algorithms for incorporating prior topological information in hmms: application to transmembrane proteins. *BMC bioinformatics*, 7(1):189, 2006.

[3] Lalit Bahl, Peter Brown, Peter De Souza, and Robert Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 49–52. IEEE, 1986.

[4] Timothy L Bailey, Charles Elkan, et al. Fitting a mixture model by expectation maximization to discover motifs in bipolymers. 1994.

[5] Pierre Baldi and Yves Chauvin. Smooth on-line learning algorithms for hidden markov models. *Neural Computation*, 6(2):307–318, 1994.

[6] A Barazandeh, MR Mohammadabadi, M Ghaderi-Zefrehei, and H Nezamabadipour. Predicting cpg islands and their relationship with genomic feature in cattle by hidden markov model algorithm. *Iranian Journal of Applied Animal Science*, 6(3):571–579, 2016.

[7] Leonard Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, 3:1–8, 1972.

[8] Leonard E Baum and John Alonzo Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, 1967.

[9] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[10] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[11] Leonard E Baum and George Sell. Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, 27(2):211–227, 1968.

[12] Manoj Bhasin and GPS Raghava. Eslpred: Svm-based method for subcellular localization of eukaryotic proteins using dipeptide composition and psi-blast. *Nucleic acids research*, 32(suppl_2):W414–W419, 2004.

[13] Roslyn M Bill, Peter JF Henderson, So Iwata, Edmund RS Kunji, Hartmut Michel, Richard Neutze, Simon Newstead, Bert Poolman, Christopher G Tate, and Horst Vogel. Overcoming barriers to membrane protein structure determination. *Nature biotechnology*, 29(4):335–340, 2011.

[14] Ewan Birney. Hidden markov models in biological sequence analysis. *IBM Journal of Research and Development*, 45(3.4):449–454, 2001.

[15] Torsten Blum, Sebastian Briesemeister, and Oliver Kohlbacher. Multiloc2: integrating phylogeny and gene ontology terms improves subcellular protein localization prediction. *BMC bioinformatics*, 10(1):274, 2009.

[16] Rupali S Chavan and Ganesh S Sable. An overview of speech recognition using hmm. *International Journal of Computer Science and Mobile Computing*, 2(6):233–238, 2013.

[17] Kuo-Chen Chou, Hong-Bin Shen, et al. Recent advances in developing web-servers for predicting protein attributes. *Natural Science*, 1(02):63, 2009.

[18] UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47(D1):D506–D515, 2019.

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

[20] Nello Cristianini and John Shawe-Taylor. *Support Vector Machines and other kernel-based learning methods*. Cambridge, 2004.

[21] Nilanjan Dasgupta, Simon Lin, and Lawrence Carin. Sequential modeling for identifying cpg island locations in human genome. *IEEE Signal Processing Letters*, 9(12):407–409, 2002.

[22] Pierre Dönnes and Annette Höglund. Predicting protein subcellular localization: past, present, and future. *Genomics, proteomics & bioinformatics*, 2(4):209–215, 2004.

[23] Sean R Eddy. Accelerated profile hmm searches. *PLoS computational biology*, 7(10):e1002195, 2011.

[24] Piero Fariselli, Pier Luigi Martelli, and Rita Casadio. A new decoding algorithm for hidden markov models improves the prediction of the topology of all-beta membrane proteins. *BMC bioinformatics*, 6(4):1–7, 2005.

[25] Robert D Finn, Jody Clements, and Sean R Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic acids research*, 39(suppl_2):W29–W37, 2011.

[26] Aditya Gupta and Bhuwan Dhingra. Stock market prediction using hidden markov models. In *2012 Students Conference on Engineering and Systems*, pages 1–4. IEEE, 2012.

[27] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[28] Md Rafiul Hassan and Baikunth Nath. Stock market forecasting using hidden markov model: a new approach. In *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, pages 192–196. IEEE, 2005.

[29] Sikander Hayat, Christoph Peters, Nanjiang Shu, Konstantinos D Tsirigos, and Arne Elofsson. Inclusion of dyad-repeat pattern improves topology prediction of transmembrane β-barrel proteins. *Bioinformatics*, 32(10):1571–1573, 2016.

[30] Jerold Heller and Irwin Jacobs. Viterbi decoding for satellite and space communication. *IEEE Transactions on Communication Technology*, 19(5):835–848, 1971.

[31] Peter Hönigschmid and Dmitrij Frishman. Accurate prediction of helix interactions and residue contacts in membrane proteins. *Journal of structural biology*, 194(1):112–123, 2016.

[32] Eva Lorenzo Iglesias, A Seara Vieira, and Lourdes Borrajo. An hmm-based oversampling technique to improve text classification. *Expert Systems with Applications*, 40(18):7184–7192, 2013.

[33] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493, 1999.

[34] B-H Juang and Lawrence R Rabiner. The segmental k-means algorithm for estimating parameters of hidden markov models. *IEEE Transactions on acoustics, speech, and signal Processing*, 38(9):1639–1641, 1990.

[35] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Kathryn Tunyasuvunakool, Olaf Ronneberger, Russ Bates, Augustin Žídek, Alex Bridgland, et al. High accuracy protein structure prediction using deep learning.

Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book), 22:24, 2020.

[36] Robel Y Kahsay, Guang Gao, and Li Liao. An improved hidden markov model for transmembrane protein detection and topology prediction and its applications to complete genomes. *Bioinformatics*, 21(9):1853–1858, 2005.

[37] Lukas Käll, Anders Krogh, and Erik LL Sonnhammer. An hmm posterior decoder for sequence feature prediction that includes homology information. *Bioinformatics*, 21(suppl_1):i251–i257, 2005.

[38] Lukas Käll, Anders Krogh, and Erik LL Sonnhammer. Advantages of combined transmembrane topology and signal peptide prediction—the phobius web server. *Nucleic acids research*, 35(suppl_2):W429–W432, 2007.

[39] Vandana Korde and C Namrata Mahender. Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, 3(2):85, 2012.

[40] Daniel Kozma, Istvan Simon, and Gabor E Tusnady. Pdbtm: Protein data bank of transmembrane proteins after 8 years. *Nucleic acids research*, 41(D1):D524–D529, 2012.

[41] Elodie Laine, Stephan Eismann, Arne Elofsson, and Sergei Grudinin. Protein sequence-to-structure learning: Is this the end (-to-end revolution)? *arXiv preprint arXiv:2105.07407*, 2021.

[42] Jung-Youn Lee, Xu Wang, Weier Cui, Ross Sager, Shannon Modla, Kirk Czymmek, Boris Zybaliov, Klaas van Wijk, Chong Zhang, Hua Lu, et al. A plasmodesmata-localized protein mediates crosstalk between cell-to-cell communication and innate immunity in arabidopsis. *The Plant Cell*, 23(9):3353–3373, 2011.

[43] Jiefu Li, Jung-Youn Lee, and Li Liao. Detecting de novo plasmodesmata targeting signals and identifying pd targeting proteins. In *International Conference on Computational Advances in Bio and Medical Sciences*, pages 1–12. Springer, 2019.

[44] Jiefu Li, Jung-Youn Lee, and Li Liao. Detecting de novo plasmodesmata targeting signals and identifying pd targeting proteins. In *Computational Advances in Bio and Medical Sciences: 9th International Conference, ICCABS 2019, Miami, FL, USA, November 15–17, 2019, Revised Selected Papers*, volume 12029, page 1. Springer Nature, 2020.

[45] Jiefu Li, Jung-Youn Lee, and Li Liao. A novel algorithm for training hidden markov models with positive and negative examples. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 305–310. IEEE, 2020.

[46] Jiefu Li, Jung-Youn Lee, and Li Liao. A new algorithm to train hidden markov models for biological sequences with partial labels. *BMC bioinformatics*, 22(1):1–21, 2021.

[47] Andrei L Lomize and Irina D Pogozheva. Tmdock: an energy-based method for modeling α-helical dimers in membranes. *Journal of molecular biology*, 429(3):390–398, 2017.

[48] Hiroshi Mamitsuka. A learning method of hidden markov models for sequence discrimination. *Journal of Computational Biology*, 3(3):361–373, 1996.

[49] Ara V Nefian, Luhong Liang, Xiaobo Pi, Liu Xiaoxiang, Crusoe Mao, and Kevin Murphy. A coupled hmm for audio-visual speech recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–2013. IEEE, 2002.

[50] Anton A Polyansky, Anton O Chugunov, Pavel E Volynsky, Nikolay A Krylov, Dmitry E Nolde, and Roman G Efremov. Preddimer: a web server for prediction of transmembrane helical dimers. *Bioinformatics*, 30(6):889–890, 2014.

[51] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[52] Malgorzata Ryngajllo, Liam Childs, Marc Lohse, Federico M Giorgi, Anja Lude, Joachim Selbig, and Björn Usadel. Slocx: predicting subcellular localization of arabidopsis proteins leveraging gene expression data. *Frontiers in plant science*, 2:43, 2011.

[53] Vijayakumar Saravanan and PTV Lakshmi. Apslap: an adaptive boosting technique for predicting subcellular localization of apoptosis protein. *Acta biotheoretica*, 61(4):481–497, 2013.

[54] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[55] Johannes Söding. Protein homology detection by hmm–hmm comparison. *Bioinformatics*, 21(7):951–960, 2005.

[56] Erik LL Sonnhammer, Gunnar Von Heijne, Anders Krogh, et al. A hidden markov model for predicting transmembrane helices in protein sequences. 1998.

[57] Jianfeng Sun and Dmitrij Frishman. Deephelicon: Accurate prediction of inter-helical residue contacts in transmembrane proteins by residual neural networks. *Journal of Structural Biology*, 212(1):107574, 2020.

[58] Ioannis A Tamposis, Konstantinos D Tsirigos, Margarita C Theodoropoulou, Panagiota I Kontou, and Pantelis G Bagos. Semi-supervised learning of hidden markov models for biological sequence analysis. *Bioinformatics*, 35(13):2208–2215, 2019.

[59] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.

[60] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[61] Xiao-Feng Wang, Zhen Chen, Chuan Wang, Ren-Xiang Yan, Ziding Zhang, and Jiangning Song. Predicting residue-residue contacts and helix-helix interactions in transmembrane proteins using an integrative feature-based random forest approach. *PloS one*, 6(10):e26767, 2011.

[62] Xu Wang, Gabriel Robles Luna, Cecilia Noemi Arighi, and Jung-Youn Lee. An evolutionarily conserved motif is required for plasmodesmata-located protein 5 to regulate cell-to-cell movement. *Communications biology*, 3(1):1–12, 2020.

[63] Kyoung-Jae Won, Adam Prügel-Bennett, and Anders Krogh. Training hmm structure with genetic algorithm for biological sequence analysis. *Bioinformatics*, 20(18):3613–3619, 2004.

[64] Yan Xia, Axel W Fischer, Pedro Teixeira, Brian Weiner, and Jens Meiler. Integrated structural biology for α-helical membrane protein structure determination. *Structure*, 26(4):657–666, 2018.

[65] Kwan Yi and Jamshid Beheshti. A hidden markov model-based text classification of medical documents. *Journal of Information Science*, 35(1):67–81, 2009.

Appendix A

DATASET IN PD LOCALIZATION PREDICTION

These 360 type-I transmembrane proteins are all retrieved from Uniport [18]. It is named as dataset A and used in chapter 2 for cross validation of PD vs none-PD.

Table A.1: Protein IDs of 360 validate sequences for PD vs none-PD classification - part 1

IMK3_ARATH	ERL2_ARATH	ERECT_ARATH	ERL1_ARATH
HERK_ARATH	FERON_ARATH	CLV1_ARATH	BAME1_ARATH
B120_ARATH	BAME2_ARATH	BAME3_ARATH	CALX1_ARATH
BRL3_ARATH	MDIS1_ARATH	MDIS2_ARATH	LRK62_ARATH
LRK14_ARATH	LRK16_ARATH	LRK17_ARATH	LRKS1_ARATH
LRK19_ARATH	LRKS4_ARATH	LRK22_ARATH	NIK2_ARATH
EMS1_ARATH	P24B3_ARATH	P24DB_ARATH	P24B2_ARATH
P24D8_ARATH	P24D5_ARATH	P24D6_ARATH	P24D3_ARATH
P24D4_ARATH	P24D7_ARATH	P24D9_ARATH	LRK31_ARATH
LRL15_ARATH	LRK51_ARATH	LRK55_ARATH	LRK92_ARATH
LRL21_ARATH	LRK41_ARATH	LRL24_ARATH	LRL25_ARATH
LRK56_ARATH	LRK58_ARATH	LRK32_ARATH	LRK43_ARATH
LRK44_ARATH	LRK52_ARATH	LRK53_ARATH	LRK71_ARATH
LRK21_ARATH	LRK42_ARATH	LRK54_ARATH	LRK81_ARATH
LRKS6_ARATH	LRKS7_ARATH	LRL27_ARATH	LRL28_ARATH
LRK57_ARATH	LRK82_ARATH	LRK61_ARATH	LRK11_ARATH
LRK64_ARATH	LRK13_ARATH	LRK91_ARATH	LRK59_ARATH
LRK12_ARATH	LRK72_ARATH	LRK15_ARATH	LRK18_ARATH
LRKS5_ARATH	LRL11_ARATH	LRL22_ARATH	MEE39_ARATH
NIK1_ARATH	LRK63_ARATH	LRL12_ARATH	NIK3_ARATH
NCRK_ARATH	P24DA_ARATH	SOL1_ARATH	PRK5_ARATH
RKF1_ARATH	RKF3_ARATH	PSKR1_ARATH	SD17_ARATH
SD31_ARATH	SERK2_ARATH	SSRA_ARATH	SERK4_ARATH

Table A.2: Protein IDs of 360 validate sequences for PD vs none-PD classification - part 2

NICA_ARATH	PAM74_ARATH	PDLP8_ARATH	PDLP2_ARATH
PDLP3_ARATH	PDLP6_ARATH	PDLP7_ARATH	PXC2_ARATH
PXC1_ARATH	PXC3_ARATH	PXL1_ARATH	PSRK_ARATH
RCH1_ARATH	PDLP1_ARATH	PDLP4_ARATH	PDLP5_ARATH
PRK1_ARATH	RKS1_ARATH	PRK6_ARATH	PR5K_ARATH
OST1B_ARATH	RPK1_ARATH	OST1A_ARATH	OST48_ARATH
PEPR2_ARATH	RGI3_ARATH	RCH2_ARATH	LK110_ARATH
LK111_ARATH	PEPR1_ARATH	Y2451_ARATH	Y2579_ARATH
Y2921_ARATH	Y1448_ARATH	WAKLR_ARATH	Y2393_ARATH
WAKLN_ARATH	Y5977_ARATH	Y3037_ARATH	Y5815_ARATH
Y1133_ARATH	Y1144_ARATH	Y2899_ARATH	Y2913_ARATH
Y2923_ARATH	Y5838_ARATH	Y2241_ARATH	Y2289_ARATH
Y2685_ARATH	Y3148_ARATH	Y4230_ARATH	Y4245_ARATH
Y5182_ARATH	Y5371_ARATH	Y2242_ARATH	Y4361_ARATH
TPST_ARATH	Y5241_ARATH	Y5248_ARATH	Y5392_ARATH
Y5457_ARATH	Y5344_ARATH	Y2165_ARATH	Y2317_ARATH
Y3804_ARATH	Y5188_ARATH	Y5374_ARATH	Y5487_ARATH
THE1_ARATH	Y1130_ARATH	Y1136_ARATH	Y1105_ARATH
TMK1_ARATH	Y1124_ARATH	Y1135_ARATH	Y1137_ARATH
Y1141_ARATH	Y1142_ARATH	Y1143_ARATH	Y5900_ARATH
Y5639_ARATH	Y5694_ARATH	Y4052_ARATH	Y4119_ARATH
Y4291_ARATH	Y4312_ARATH	Y4539_ARATH	VSR1_ARATH
VSR4_ARATH	Y1639_ARATH	Y1668_ARATH	Y1755_ARATH
Y1972_ARATH	Y1523_ARATH	Y1497_ARATH	Y1649_ARATH
Y1643_ARATH	Y1743_ARATH	Y2214_ARATH	Y5169_ARATH
Y5189_ARATH	Y5393_ARATH	Y5389_ARATH	Y5410_ARATH
Y5537_ARATH	Y1680_ARATH	Y1684_ARATH	Y1864_ARATH

Table A.3: Protein IDs of 360 validate sequences for PD vs none-PD classification - part 3

Y4372_ARATH	Y5005_ARATH	Y5129_ARATH	Y5181_ARATH
Y5343_ARATH	Y5524_ARATH	Y5573_ARATH	Y5596_ARATH
Y5597_ARATH	Y5613_ARATH	Y5614_ARATH	Y1146_ARATH
Y1148_ARATH	Y1421_ARATH	Y2027_ARATH	Y3471_ARATH
Y3475_ARATH	Y3463_ARATH	Y3565_ARATH	Y1699_ARATH
Y2232_ARATH	Y2278_ARATH	Y2239_ARATH	Y1571_ARATH
VSR2_ARATH	WAK4_ARATH	WAKLF_ARATH	WAKLI_ARATH
WAKLJ_ARATH	Y1570_ARATH	Y1677_ARATH	Y1634_ARATH
Y1155_ARATH	Y1150_ARATH	Y1675_ARATH	Y1661_ARATH
Y1723_ARATH	Y1765_ARATH	WAKLL_ARATH	WAK3_ARATH
VSR6_ARATH	WAKLC_ARATH	Y1518_ARATH	Y1756_ARATH
WAKLP_ARATH	WAK2_ARATH	WAKLA_ARATH	Y1534_ARATH
Y1561_ARATH	Y1614_ARATH	VSR5_ARATH	Y1063_ARATH
WAKLM_ARATH	WAKLB_ARATH	WAKLQ_ARATH	WAKLE_ARATH
VSR3_ARATH	WAKLD_ARATH	WAKLG_ARATH	WAK1_ARATH
WAKLK_ARATH	WAKLO_ARATH	WAK5_ARATH	WAKLH_ARATH
CRN_ARATH	CE101_ARATH	DEX1_ARATH	CALX2_ARATH
CLV2_ARATH	BRL1_ARATH	BRI1_ARATH	BRL2_ARATH
ANX1_ARATH	BAK1_ARATH	FEI1_ARATH	EFR_ARATH
HSL2_ARATH	FLS2_ARATH	HSL1_ARATH	FEI2_ARATH
IRE1A_ARATH	IRK_ARATH	RPK2_ARATH	RLP26_ARATH
RLP5_ARATH	RLP11_ARATH	RLP54_ARATH	RLP56_ARATH
RLK6_ARATH	RLP50_ARATH	RLP53_ARATH	RMR4_ARATH
RP24L_ARATH	SIRK_ARATH	SBIR1_ARATH	SERK5_ARATH
SD113_ARATH	SERK1_ARATH	SD18_ARATH	RLP30_ARATH
RLP31_ARATH	RLP33_ARATH	RLP35_ARATH	RLP37_ARATH
RLP9A_ARATH	RLP46_ARATH	RMR2_ARATH	RLP27_ARATH
RLP2_ARATH	RLP36_ARATH	RLP39_ARATH	RLP45_ARATH
RLP47_ARATH	RLP14_ARATH	RLK5_ARATH	RLP6_ARATH
RLP12_ARATH	RMR6_ARATH	RLP32_ARATH	RLP55_ARATH
RLP19_ARATH	RMR1_ARATH	RLK0_ARATH	RLP43_ARATH
RLP51_ARATH	RLK1_ARATH	RLP29_ARATH	RLP38_ARATH
RLP41_ARATH	RLP42_ARATH	RLP44_ARATH	RLP48_ARATH
RLP57_ARATH	RMR5_ARATH	SD16_ARATH	SD129_ARATH
SD22_ARATH	SD11_ARATH	SD25_ARATH	RMR3_ARATH
Y1112_ARATH	TMM_ARATH	TDR_ARATH	TMKL1_ARATH
Y1343_ARATH	Y1357_ARATH	Y1341_ARATH	ACCR4_ARATH
ACCR2_ARATH	ACCR3_ARATH	ALE2_ARATH	ACCR1_ARATH
ACR4L_ARATH	ANX2_ARATH	HAP2_ARATH	GSO2_ARATH
GSO1_ARATH	MIK1_ARATH	MIK2_ARATH	IMK2_ARATH

Appendix B

MULTI-PATHES TRANSMEMBRANE PROTEIN DATASET USED IN TRAIN HMM WITH PARTIAL LABEL

These 83 sequences in multi-pathes transmembrane protein dataset are from the work done in [36], and used to test HMM with partial label data in chapter 3.

Table B.1: Multi-pathes transmembrane protein dataset (83)

5H2A_CRIGR	5HT3_HUMAN	5HT3_MOUSE	CYOA_ECOLI
CYOC_ECOLI	HEMA_PI4HA	IGGB_STRSP	LEP_ECOLI
SECY_BACSU	A1AA_HUMAN	AA1R_CANFA	AA2A_CANFA
CYOB_ECOLI	CYOE_ECOLI	IG1R_HUMAN	IL2B_HUMAN
MALF_ECOLI	MPRD_HUMAN	EDG1_HUMAN	GLPA_HUMAN
GLP_PIG	GPT_CRILO	HG2A_HUMAN	MELB_ECOLI
OPPB_SALTY	UHPT_ECOLI	VNB_INBBE	CYOD_ECOLI
GP1B_HUMAN	HEMA_CDVO	HEMA_MEASH	IL2A_HUMAN
MOTB_ECOLI	OPPC_SALTY	VMT2_IAANN	4F2_HUMAN
BAC1_HALS1	BACH_HALSS	BACR_HALHA	CEK2_CHICK
ITA5_MOUSE	LACY_ECOLI	TRSR_HUMAN	1PRCM
A2AA_HUMAN	CXB1_HUMAN	CXB1_RAT	CXB1_XENLA
GLPC_HUMAN	MAGL_MOUSE	TRBM_HUMAN	A4_HUMAN
EGFR_CHICK	EGFR_DROME	EGFR_HUMAN	IMM1_ECOLI
MYP0_HUMAN	SUIS_HUMAN	TCR1_ECOLI	1PRCH
ADT_RICPR	FCE2_HUMAN	LECH_HUMAN	LECI_MOUSE
PT2M_ECOLI	RIB1_RAT	SECE_ECOLI	OPS1_CALVI
OPS2_DROME	OPS3_DROME	OPS4_DROME	OPSB_HUMAN
OPSD_BOVIN	OPSG_HUMAN	OPSR_HUMAN	1PRCL
C561_BOVIN	GMCR_HUMAN	IMMA_CITFR	NEP_HUMAN
NGFR_HUMAN	PIGR_HUMAN	TCB1_RABIT	

Appendix C

DATASET USED IN INTER-HELIX PREDICTION REFINEMENT

The PDB chains in Table C.1 is from DeepHelicon group [57], and used as independent dataset for 2D contact structure model construction in chapter 5.

Table C.1: Transmembrane proteins' PDB chain IDs in TRAIN dataset (165)

1aig L	1h7c A	1jb0 A	1kqf C	1p49 A	3abk B	3aqp A
1pw4 A	2a06 P	2ahy A	2axt A	2bhw A	3abk C	3ayf A
2c3e A	2cfp A	2e74 B	2e74 A	2f93 B	3abv D	3aou A
2jkv A	2pri A	2vl1 A	2yev B	3a3y A	3abv C	3aoa A
3b44 A	3dh4 A	3nym A	3qnq A	3tx3 A	3waj A	3wo6 A
3d31 C	3egw C	3wvf A	3zcc A	4a01 A	4a2n B	4a82 A
3det A	3g6b A	4a97 A	4ain A	4aps A	4aw6 B	4bem A
3jyc A	3m71 A	4al0 A	3h90 A	4avm A	4bbj A	4bpd A
4cad C	4g7v S	4k1c A	4wd7 A	5a63 C	5bzb A	5d91 A
4d1a A	4gd3 A	4khz F	4wis A	5aex A	5c65 A	5da0 A
4dji A	4gx0 A	4m64 B	4ymk A	5aji A	5c6p A	5dir A
4dw0 A	4hkr A	4mnd A	4zr1 A	5aww Y	5cgc A	5djq A
4ev6 A	4ikp A	4phz A	5a1s A	5aym A	5ckr A	5djq C
4ezc A	4iu8 A	4quv A	5a43 A	5azb A	5ctg A	5doq B
4f4c A	4jkv A	4rdq A	5a44 A	5bw8 D	5d0y A	5duo A
4fz0 A	4jta B	4u1w A	5a63 B	5bw8 C	5d3m D	5ec5 A
5edl A	5h1q A	5l8r G	5ogl A	5v7p A	6b3j R	6bhu A
5egi A	5hwx A	5lki A	5sv0 B	5v8k A	6bat A	6bml A
5eiy A	5iji A	5lwy A	5t4d A	5w3s A	6bcj B	6bw5 A
5eke A	5irx A	5m87 A	5t77 A	5x5y F	6c96 A	6ezn E
5er7 B	5iwk A	5mrw A	5tcx A	5y78 A	6eti A	6ezn B
5ezm A	5iws A	5n6h A	5tj6 A	5yi2 B	6ezn H	6ezn F
5fgn A	5jwy A	5nv9 A	5u73 A	5z96 A	6ezn C	5zdh A
5gko A	5khn B	5oge A	5ul0 A			

DATA(101) is used as cross validation dataset in chapter 5. DATA(101) is a combined dataset of TEST and PREVIOUS datasets used in [57], where TEST dataset was extracted from PDBTM database by DeepHelicon group and PREVIOUS is a combined dataset from TMhhcp [61] and MemConP [31].

Table C.2: Transmembrane Proteins' PDB chain IDs in DATA(101) dataset

1jb0 L	2abm A	2axt B	2axt Z	2bs2 C	2zuq A	3abk A
2a06 C	2acz C	3b4r A	3o7p A	3mp7 A	3ux4 A	3tui A
2a65 A	2acz D	3wdo A	4a4m A	4bw5 A	4dnt A	4dxw A
4f35 B	4fc4 A	4he8 D	4he8 F	4j05 A	4kpp A	4mes A
4oqy A	4p79 A	4pgr A	4phz B	4phz K	4q2e A	4qtn A
4rp8 A	4ryi A	4tqu M	4xks A	4yms D	5a8e A	5b57 A
5c6n A	5doq A	5guf A	5guw B	5jki A	5kbw A	5l26 A
5o0t A	2rh1 A	2zxe A	2zy9 A	3b9w A	3c02 A	3ddl A
5x5y G	2w2e A	3eam A	3gia A	3hd6 A	3k3f A	3kly A
5xjj A	2wsc 2	3gd8 A	3m7l A	3m73 A	3qe7 A	3rko L
5xu1 M	2wsw A	3rvy A	3t9n A	3tij A	3ukm A	3usi A
6awf C	2xq2 A	3v5u A	4czb B	4hyg A	4ikw A	4m5b A
6awf D	2yev A	4q2g B	4r0c B	4twd A	4u1x C	4wd8 B
6bar A	2yvx A	2r6g G	2r6g F	2nq2 A	2jln A	2cfq A
6cb2 A	2z73 A	1xqf A				

Appendix D

PERMISSIONS

Chapter 3 was published in BMC bioinformatics in 2021 with title "A new algorithm to train hidden Markov models for biological sequences with partial labels".

The copyright link is the following:

https://bmcbioinformatics.biomedcentral.com/submission-guidelines/copyright