上海科技大学
ShanghaiTech University

本科毕业论文（设计）

题　　目：	基于深度学习模型的蛋白质折叠预测
学生姓名：	方海洲
学　　号：	93320319
入学年份：	2014年
所在学院：	信息科学与技术学院
攻读专业：	计算机科学与技术
指导教师：	范睿 教授

上海科技大学教学事务处制

二 零 一 八 年　六 月

上海科技大学
ShanghaiTech University

THESIS

Subject:	Prediction of Protein Folding by Deep Learning Model
Student Name:	Haizhou Fang
Student ID:	93320319
Year of Entrance:	2014
School:	School of Information Science and Technology
Major:	Computer Science
Advisor:	Prof. Rui Fan

Office of Academic Affairs, ShanghaiTech University
Date: 2018 / 06

摘要

蛋白质折叠问题是生物信息学中最重要的板块之一,而其中最受人关注的毫无疑问是：如何根据给定的氨基酸序列预测其对应的蛋白质结构。一个蛋白质分子的生物学性质与其三维结构存在着密不可分的关系，而蛋白质的天然三维结构正是由氨基酸单体的链式结构折叠而来。然而事实证明，从氨基酸序列直接预测蛋白质空间结构往往效果不佳。因此，本文尝试并改进了目前在深度学习界较为通用的模型，将其运用于预测蛋白质的二级结构中，从而在一级链式结构与三级空间结构中架起一座桥梁。

本文从蛋白质结构数据库（RCSB PDB）中收集到数据后，首先以位置特异性得分矩阵（PSSM）为氨基酸序列编码，随后使用了逻辑回归、卷及神经网络、循环神经网络等模型以及它们的改进版本对数据集进行了学习，最终在测试集上获得了最高84.38%的Q3准确率，以及73.49%的Q8准确率，均为业界领先水平。通过分析实验过程与结果，本文得出结论：在目前主流的深度学习模型中，基于循环神经网络改进版本的模型在蛋白质二级结构预测任务上不仅是准确率最高的，而且是最有改进潜力的。具体实现代码可以参见https://github.com/EmoN-Fang/PSSP。

关键词：蛋白质二级结构预测，位置特异性得分矩阵，深度神经网络

ABSTRACT

The protein folding problem is one of the most important problems in Bioinformatics, and the question which attract most attention from the public must be "How to predict the corresponding protein structure based on a given amino acid sequence?" The biological properties of a protein molecule are inextricably linked with its three-dimensional structure, while the natural three-dimensional structure of a protein is the result of folding from the chain structure of amino acid monomers. However, predicting a protein's spatial structure from its amino acid sequence directly is always ineffective. Therefore, we tried and improved the models that are currently widely used in deep learning area and applied it to predict the secondary structure of proteins, as establishing a bridge between the first-order chain structure and the tertiary space structure .

Data was first collected from the Protein Data Bank (RCSB PDB), then the amino acid sequence were encoded using position-specific scoring matrices (PSSM). After that, logistic regression, convolutional neural networks, recurrent neural network and their updated versions were used to train on the dataset. In the end, this work achieved 84.38% for Q3 accuracy and 73.49% for Q8 accuracy, both of which are slightly surpassed the predictive accuracy of the present state of the art. By analyzing the experimental process and results, we concluded that among most of modern deep learning models, the model based on the improved version of the recurrent neural network not only has the highest accuracy but also has the best potential for further improvement in the protein secondary structure prediction tasks. The source code is available at the github repository https://github.com/EmoN-Fang/PSSP.

Keywords: protein secondary structure prediction, position-specific scoring matrices, deep neural network

Contents

Chapter 1 Introduction

Bioinformatics is an interdisciplinary studies of life sciences, statistics and computer science. It sets molecular biology as the research object while using computers and the internet as research tools. The purpose of this subject is to reveal the complexity of genetic information structure and the fundamental laws of genetic language. Since the 1990s, with the development of various genome sequencing projects and the breakthroughs in the determination of molecular structures, plenty of biological databases have grown rapidly. At the same time, the increment in computing power has given people the opportunity to use statistical and machine simulation methods to explore the links between genomes at different scales. In the early days of bioinformatics research, people focused on large-scale genome sequencing, gene identification and gene discovery. After 2000, with the completion of the sequencing task for human and some other creatures' genomes, the focus of research has gradually shifted from the structure of genome to the function of genes.

Researchers have discovered that protein reflect a significant relationship between protein structure and function at the molecular level. The biological property of a protein is determined by its three-dimensional natural structure, and the natural structure of a protein is determined by its one-dimensional amino acids sequence. Therefore, the protein folding problem has attracted lots of attention in the past decade, which is mainly focusing on the following three points[1]: (i) Physical rules of folding: How do the physicochemical properties of one-dimensional amino acid sequences determine the 3D native structure of proteins? (ii) Folding mechanism: Among millions of possible conformation of a chain, how can proteins be folded so rapidly? (iii) Using computers for prediction: Can we design a computer algorithm that predicts the structure of a protein based on its amino acid sequence?

The most commonly used method in the industry today is the Molecular Dynamics simulation softwares[2]. By calculating the interaction forces between atoms, such as Hydrogen bonds, van der Waals interactions, etc, the combination of these physical forces are described as a kind of force field. Then high performance computers are used to simulate protein folding on the femtosecond(10^{-15} second) timescale. Although this kind of simulation can achieve relatively good results, it often requires a large number of iteration steps and consumes a lot of resources to calculate various interaction forces just for a single certain protein, which is not suitable for experimentation and analysis on a protein dataset.

Therefore, we hope to find a new way to accelerate protein folding without losing much precision. Supervised learning of machine learning (especially deep learning) has achieved amazing results in many end-to-end classification and regression tasks in recent years. Hence, to try deep learning models on the protein folding problem occurs

in my mind naturally. Below we will introduce the basic concept of different levels in protein structure prediction and deep learning.

1.1 Protein structure

In general, we can decompose protein structure into 3 levels: primary structure, secondary structure and tertiary structure. The tertiary structure has both 2-d representation and 3-d representation [3]. In this paper, we mainly focus on protein secondary structure prediction, which uses primary structure as input.

1.1.1 Primary structure

The primary structure of a protein is actually a string of amino acid sequences. There are a total of 20 kinds of amino acids (also called residue), each corresponding to an English letter. Thus, the primary structure is represented by a sequence letters over the 20-letter alphabet. [3]

1.1.2 Secondary structure

The secondary structure of a protein is a 3-d form of local segments of proteins. In other words, before the protein folding to its final tertiary structure, it will form some local structure as an intermediate state. There are two types of secondary structure classification: three structural elements (Q3) or eight structural elements (Q8), according to DSSP (Dictionary of Protein Secondary Structure).

Q3 contains the two main structural elements which behaves very different: alpha helices and beta sheets. The third class is called nonregular coil, which is not a real structural element, but stands for where the typical secondary structural element is missing.

Q8 contains eight types of secondary structure, as DSSP defines [4]:

H = alpha helix
B = residue in isolated beta-bridge
E = extended strand, participates in beta ladder
G = 3-helix (3/10 helix)
I = 5 helix (pi helix)
T = hydrogen bonded turn
S = bend

Similar to Q3, besides the seven basic structural elements, there is the 8th class denoted as C = coil, which means the corresponding secondary structure of the input residues are not belong to any of the above conformations. [G, H, I] belongs to helix class in Q3, [B, E] belongs to sheet class in Q3, and [S, T, C] belongs to nonregular coil class in Q3.

Here we will give an example of how the input and output of protein secondary structure prediction task looks like, which will help readers to understand the problem better.

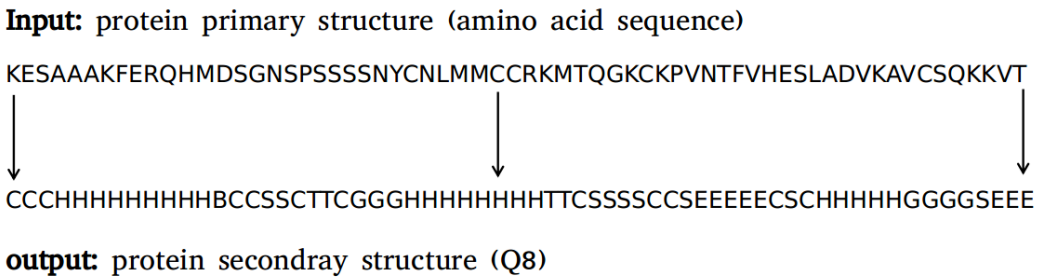


Figure 1.1: An example of protein primary structure and secondary structure

1.1.3 Tertiary structure

A protein's tertiary structure can be represented in both two dimension and three dimension. 3-d representation is easier to understand, just as Figure 1.2[5] shows below:

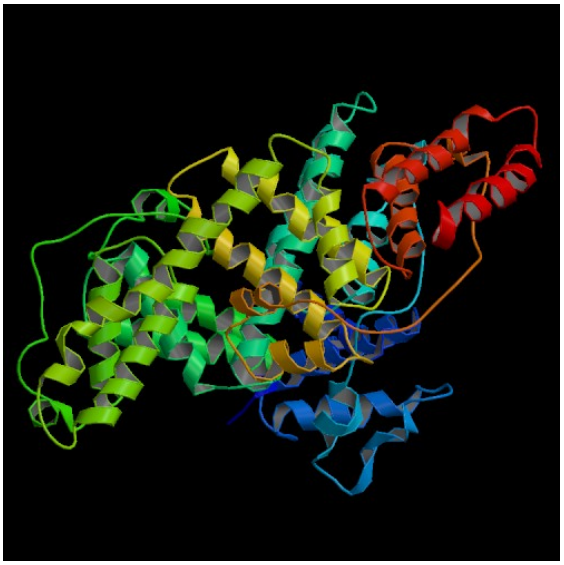
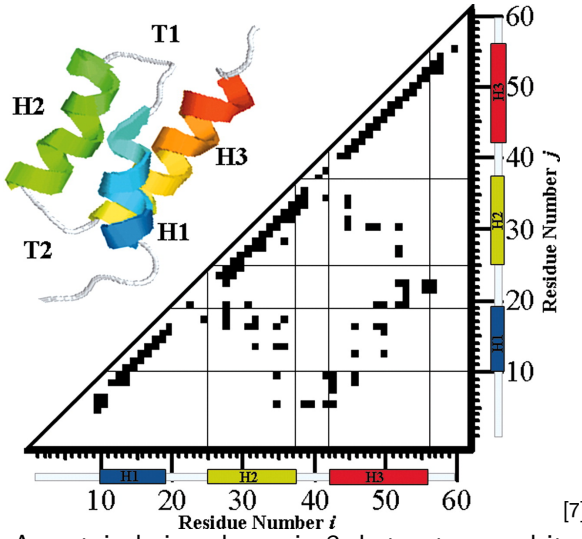

[5]

Figure 1.2: An example of 3-d tertiary structure of a protein

There is one big question exists in 3-d representation. Since there are coordinates stored for every residue in software to visualize the tertiary structure of protein, the whole coordinates will be changed after a simple rotation, while the protein remains the same. Thus, there exist a simpler 2-D representation of protein tertiary structure: distance map[6].

A distance map is a matrix $\mathbf{C}$, which $\mathbf{C}_{i,j}$ stands for the Euclidean distance of the i_{th} residue and j_{th} residue in the 3-d space. Sometimes threshold will be set for the Euclidean distance. If the distance is larger than the threshold, the corresponding entry will reset it to 1, otherwise, the entry will be reset to 0. This new type of map is called contact map, which only contains 0 and 1 inside the matrix. We can see that the contact map of a protein is just a same but simpler representation for the 3-d structure, which means we can use a software to reconstruct them from each other. The example of contact map is shown in Figure 1.3[7], and we will not talk it into details further because it's not related to our task in this paper.



[7]

Figure 1.3: A protein being shown in 3-d structure and its contact map

1.2 Machine Learning

Machine learning is a type of algorithms that automatically analyzes and obtains rules from data, and uses the law to process unknown data. The well-known machine learning architectures includes decision tree, clustering and neural networks, etc. In this paper, we only talk about neural network, which actually belongs to a subcategory of artificial intelligence called "connectionism." As early as the 1940s, connectionism was proposed and attracted many attention. However in the 1970s, due to the downfall of perceptron, even the entire AI community had ushered in a cold winter. In the 1980s, connectionism gradually back in the renaissance after the creation of Hopfiled net and back propagation algorithms. Over the next two decades, the most representative model of connectionism: neural network has gradually become the trend of the entire AI community. It represents knowledge through interconnected networks of simple units. Due to its powerful nonlinear fitting ability, and combining with the rapid increment in computing resources and computing power after the 2000s, neural network has achieved breakthrough results in many tasks.

In the 2012 ImageNet Classification competition, AlexNet[8], a deep neural network which is based on supervised learning, not only won the championship but also surprisingly improved the accuracy by about 10% compared with last year's result. From that time, people's enthusiasm has been completely ignited on applying machine learning to jobs in different areas, such as computer vision, natural language processing, and advertising recommendations, etc. In the following parts, we will briefly introduce the three key ideas in machine learning which is related to the task of protein secondary structure prediction: deep neural network, supervised learning and classification.

1.2.1 Deep neural network

As we mentioned before, neural network is the model for connectionism to represent knowledge, it's called neural network just because it looks similar to the neural network in biology. In the last century, because of the limitation in computing resources, only one layer or two hidden layers are used for jobs like classification or regression. But

in 1998, LeNet-5 proposed by Yann Lecun[9] has shown that with more hidden layers inside the network can achieve an amazing result for digits recognition. From then on, the possibility of deepening the network start to be explored. In general, network with less than three hidden layers will be called shallow neural network because the network is relatively small. When the hidden layer numbers is larger than 3, people tend to call it "deep neural network". After AlexNet[8] got a breakthrough in image classification task in 2012, with its eight hidden layers network which contains around 60 million parameters, people realized that deep neural network have its "magic" in area like computer vision. Although the mathematical theory inside the neural network still can't be proved formally, but it does has an incredible ability in fitting a non-linear function by introducing extreme high dimensions in the computation process. Here's a simple schematic diagram for a shallow neural network and the computation for a neuron inside. One should know that the sigmoid fuction here can be replaced by other activation function such as ReLU (Rectified Linear Unit)[10] for different tasks.

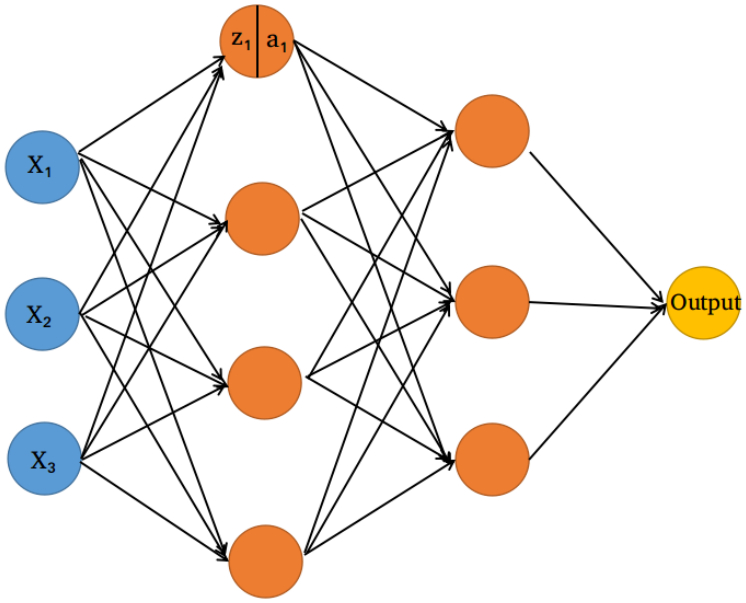


Figure 1.4: A typical 3 layer neural network

$$\begin{aligned} z_1 &= w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b_1 \\ a_1 &= sigmoid(z_1) \end{aligned} \tag{1.1}$$

1.2.2 Supervised learning

Supervised learning is a subclass under machine learning. Here, learning means find a model or pattern from the datasets, for the usage of further tasks like classification of regression. Thus, In supervised learning, each training example contain an input-output pair $(x_i.y_i)$, which x_i is usually a vector which denotes high dimensional input features, and y_i is the true label or output value for the input x_i. In supervised learning, we usually build a model or a neural network, then divide the whole dataset into three parts: training set, validation set and test set. During training process, we

will determine the hyper-parameters first, then train the parameters using the examples from training set. The most common procedure for train a model is pass the input through the model, then compute the loss between our output and the true label/value. Then, we will use back propagation to update the parameters in the model, till our output converges. After training several models, we will use validation set to tune and find the best hyper-parameter for the model. When we make sure that we have find the best model, we will test our model with the examples in the test set, which the accuracy will be considered as our final result.

1.2.3 Classification

Classification is one typical supervised learning problem. In real life, we are doing classification jobs all the time unconsciously. For example, when buying a watermelon, how can one decide whether the watermelon is tasty or not? First, one will check the luster of the watermelon, to see if it is bright green. Then, one will knock it, to check if it will return a thick loud sound. In the end, one will look at the stem of watermelon, to see if it is curled up. If all of the three answers are "yes", it is a "good" melon. In this question, luster, sound of knocking and stem are three input features. On the other hand, "good"/"bad" watermelon is the output of this question once the state of three input features has been settled. In a typical classification problem like this, we will have a certain number of input features, and based on the different values given to the input features, we will output a certain class belongs to the discrete output space. If there are only two possible output classes, then it's a binary classification problem. If there are more than three possible output classes, then it's a multi-class classification problem, such as protein secondary structure prediction problem, since it has three output classes or eight output classes.

1.2.3.1 Softmax

Here we need to talk a little deeper. In multi-class classification problem, usually there only will be one class labeled as 1, while others are labeled as 0. But for the convenience of using back propagation to update the parameters in the model, the non-linear function in output layer is Softmax function[11], which the formula is shown below:

$$p(c_k|\bar{x}) = \frac{exp(a_k(\bar{x}))}{\sum_j exp(a_j(\bar{x}))} \tag{1.2}$$

Here $\bar{x}$ stands for an example's input features, a_k is the equivalent composite function of the neural network before the k^{th} output. And $p(c_k|\bar{x})$ stands for the probability that $\bar{x}$ belongs to the k-th class. We can see that there are two advantage of using Softmax function in the output layer. Firstly, it does normalization on the original output, which locates all the new output between 0 and 1, furthermore, all the new outputs sum up to 1 which means they can be treat as the probability of classification. Secondly, as we mentioned before, Softmax function make it possible for the neural work to do back propagation from the output layer, which will make

the parameters more accurate compared with if we take a simple Max function at the output.

1.2.3.2 Negative log-likelihood function

In classification problem, we generally use the maximum likelihood estimation to construct the loss function. For a input $\bar{x}$, whose corresponding class label is k, our goal is to find such the best parameter θ such that $p(c_k|\bar{x})$ is maximum in multi-class classification. The equation is shown as below:

$$p(c_k|\bar{x}) = \prod_{i=1}^{C} p(c_i|x)^{c_i} = \prod_{i=1}^{C} y_i^{c_i} \tag{1.3}$$

Here c_k is the true class for $\bar{x}$, which is labeled as 1 in the corresponding bit, while other c_i is 0 for i $\neq$ k. C is the total number of classes, and y_i is the Softmax output of i_{th} class. There is one problem in equation (1.3), that is the multiplication may cause the final result to be close to 0. Since we want to maximize $p(c_k|\bar{x})$, the general method is to take the negative logarithm to this function, and transform it into a new equivalent problem: minimizing the negative log-likelihood function.

$$-\log p(c_k|\bar{x}) = -\log \prod_{i=1}^{C} y_i^{c_i} = -\sum_{i=1}^{C} c_i \log(y_i) \tag{1.4}$$

Thus, we have the new function (1.4) as the loss function for our classification problem. Someone might prefer to call it cross entropy loss[12]. They have the same expression, although they are derived from different concepts. We introduced it from maximum likelihood estimation because it's more related to classification.

Chapter 2 Data collecting and processing

2.1 Date collecting

Because we did not have contact with biology research area before this project, it took us a long time to find an available dataset. At first, we downloaded lots of protein sequence data from protein data bank[13] and tried use dssp program to compute the secondary structure for every amino acid sequence which it didnt work in the end. Thus, we have to look for some protein secondary structure file directly. Fortunately, we found a txt file[14] on rcsb-pdb website. The file contains around 0.3 million examples which both amino acid sequence and its corresponding protein secondary structure. Here's an example of how the sequence and secondary structure pair look like in the txt file.

101M:A:sequence
MVLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHP
ETLEKFDRVKHLKTEAEMKASEDLKKHGVTVLTALGA
ILKKKGHHEAELKPLAQSHATKHKIPIKYLEFISEAII
HVLHSRHPGNFGADAQGAMNKALELFRKDIAAKYKEL
GYQG
101M:A:secstr
CCCCHHHHHHHHHHHHHHGGGHHHHHHHHHHHHHHCG
GGGGGCTTTTTCCSHHHHHHCHHHHHHHHHHHHHHHH
HHTTTTCCHHHHHHHHHHHHHTSCCCHHHHHHHHHHHH
HHHHHHCGGGCSHHHHHHHHHHHHHHHHHHHHHHHHT
TCCC

2.2 Data processing

We can see that, the second structure here contains eight classes. Each residue in the sequence has a corresponding secondary structure output. As we said in Chapter 1, the protein secondary structure prediction problem can be treated as a multi-class classification problem. Thus, we split the original txt file into 2 parts: input features and truth label. We gathered all the sequence data together, and stored them as the input features for our model. Similarly, we stored all the secondary structure together as the truth label. All the splitting and storing procedure were done according to the order in the original file, to make sure that all the sequence data and secondary

structure are corresponded to each other after processing. Another thing we did is making a copy of the secondary structure data while replacing [G, H, I] by H, [B, E] by E, and [S, T, C] by C, so that in the new secondary structure file we have only 3 output classes. Thus, with these 2 different secondary structure label files we can do both Q3 classification and Q8 classification. But how to encode these letters properly is still a problem needs to be solved, we will introduce my method in the following parts.

2.2.1 One-hot encoding

The most common method used in encoding is one-hot encoding. What one-hot means is: for every truth label, we encode it a N-bit code, which N is the total classes number. In this N-bit code, only the bit which represents the truth label will be set to 1, all the other bits will be set to 0. For example, in 3-class protein secondary structure classification problem, the truth label has only 3 possibility: H, E and C. Thus, the one-hot encoding for the truth label H is [1, 0, 0], E is [0, 1, 0] and C is [0, 0, 1]. There is one huge advantage of using one-hot encoding for truth label, which is all the vector representations after encoding are in an orthogonal Euclidean space. A certain value of the discrete label (class) corresponds to a point in the Euclidean space, and the distance between any 2 different label are same now. We have been know that the Euclidean distance can represent the similarity between 2 different label. If we use natural number or binary system to encode the labels, then the euclidean distance between 2 labels might vary a lot, while in fact that all the distance should be the same.

We use one-hot encoding for the label space. But for the input space, we use another encoding method, which called PSSM (position-specific scoring matrices).

2.2.2 Position-specific scoring matrices

The concept of PSSM, abbreviation of position-specific scoring matrices[15], came from position weight matrix[16], which is introduced by Gary Stormo in 1982. What it does is converting the amino acid sequence to a matrix based on the probability of a residue occur in the different position of a sequence. So, basically it's a statistical method based a reference dataset. In PSSM, it uses the probability information to get a likelihood score and store it into the corresponding entry. Since there are only 20 type of amino acids, after encoding an amino acid sequence length of n will become a matrix whose shape is $(n, 20)$. This part is purely biology knowledge so we won't introduce it in detail. Readers can search for the relative articles online for better understanding. There are several ways to get the PSSM, one common practice is using the "blast" software, whose full name is "Basic Local Alignment Search Tool" [17]. In this paper, "Patent protein sequence database" was used it as the reference dataset to compute PSSM . So, we transformed the input date of our prediction task from amino acid sequence into position-specific scoring matrix, and stored as the new input. An example of PSSM is shown in Figure 2.1:

In this example, the length of amino acid sequence is 21. So, the shape of the PSSM matrix is $(21, 20)$. As you can see, on x-axis there lists all the 20 possible amino

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
1	G	0	-3	-1	-2	-3	-2	-2	6	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
2	I	-1	-3	-4	-4	-1	-3	-4	-4	-4	5	2	-3	1	0	-3	-3	-1	-3	-2	3
3	V	0	-3	-3	-4	-1	-3	-3	-4	-4	3	1	-3	1	-1	-3	-2	0	-3	-1	4
4	E	-1	0	0	3	-4	2	5	-2	0	-4	-3	1	-2	-4	-1	0	-1	-3	-2	-3
5	Q	-1	1	0	0	-3	6	3	-2	0	-3	-3	1	-1	-4	-2	0	-1	-2	-2	-3
6	C	-1	-4	-3	-4	10	-3	-4	-3	-3	-2	-2	-4	-2	-3	-3	-1	-1	-3	-3	-1
7	C	0	-4	-3	-3	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	0	-1	-3	-3	-1
8	T	0	-1	0	-1	-2	0	0	-2	6	-1	-2	-1	-1	-2	-2	0	3	-3	0	-1
9	S	0	0	2	0	-2	0	0	0	-1	-3	-3	2	-2	-3	-1	3	1	-3	-2	-2
10	I	-1	-1	-2	-3	-2	-2	-2	-3	-3	3	0	-2	0	-1	4	-1	0	-3	-2	2
11	C	-1	-4	-3	-4	10	-3	-4	-3	-3	-2	-2	-4	-2	-3	-3	-1	-1	-3	-3	-1
12	S	1	-1	2	0	-1	0	0	-1	-1	-3	-3	0	-2	-3	-1	4	2	-3	-2	-2
13	L	-2	-3	-4	-4	-1	-3	-3	-4	-3	3	4	-3	2	0	-3	-3	-1	-2	-1	2
14	Y	-2	-2	-3	-4	-3	-2	-3	-4	1	-1	-1	-2	-1	5	-4	-2	-2	2	7	-1
15	Q	-1	0	0	3	-4	5	3	-2	0	-3	-3	1	-1	-4	-2	0	-1	-3	-2	-3
16	L	-2	-2	-4	-4	-1	-2	-3	-4	-3	2	4	-3	2	0	-3	-3	-1	-2	-1	1
17	E	-1	0	0	1	-4	4	5	-2	0	-3	-3	1	-2	-4	-1	0	-1	-3	-2	-3
18	N	-2	-1	6	1	-3	0	0	-1	3	-4	-4	0	-2	-3	-2	0	0	-4	-2	-3
19	Y	-2	-2	-2	-4	-3	-2	-2	-4	2	-2	-1	-2	-1	3	-3	-2	-2	2	8	-1
20	C	0	-4	-3	-3	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	0	-1	-3	-3	-1
21	N	-2	-1	6	1	-3	0	0	-1	0	-4	-4	0	-2	-3	-2	1	0	-4	-2	-3

Figure 2.1: the position-specific scoring matrices of an amino acid sequence

acid type, and on y-axis there lists the input residues. For entry (x, y) in the PSSM, it represents a likelihood of an amino-acid x appearing at position y in a sequence. The larger the score is, the more likely it will appear. The reason why we use pssm encodes the input feature instead of one-hot encoding is that one-hot encoding is quite sparse, which means there is only 1 bit of nonzero information on each line while all the other bits are zero. That's a waste of storage plus we may need a lot of hidden layers to capture the relationship under the sparse representation. Also, one should know that we don't need to make the input features in a orthogonal vector space. On the contrary, we want the input features to be as powerful as they can in representing the property of themselves.

Thus, we used PSSM to encode the input (amino acid sequences) and used one-hot encoding to encode the output label (protein secondary structure)

Chapter 3 Experiments

In this project, we tried three different types of models on the protein secondary structure prediction task, which are: multi-class logistic regression, convolutional neural network and recurrent neural network. Logistic regression is a classic classification model, while the remaining 2 neural networks are the most popular models in deep learning community nowadays. we will describe how we designed the model and chose the hyper-parameters detailedly in the following sections. One more thing needs to be mentioned is that, because the length of amino acid sequence can vary from 20 to 500, for convenience we divided them into different length classes such as 0-50, 50-100, 100-200, etc. And due to the time and computing resource limitation, all the experiments we did is in based on the 50-100 length dataset.

3.1 Logistic Regression

As you may all know, what logistic regression[18] does is projecting the result of a linear transformation into the sigmoid function, which leads the output located between 0 and 1 and therefore can be considered as the probability of belonging to a certain class. In the traditional logistic regression model, it only contain one layer inside the model and has 2 output class. But one layer is not good enough to fit a complicated model like protein secondary structure. Thus, in our model, we add several new layers inside the logistic regression model, and use ReLU function as the activation function before the final layer. Another different point in my model compared with the classic logistic regression is, we used Softmax as the activation function for the output layer because our task is a 3-class/8-class classification problem instead of 2-class. According to the structure of logistic regression, the output will be the probability of the input belongs to a certain class among all secondary structure classes. The equation of forward propagation is shown below:

$$\begin{aligned} z^{[l[} &= w^{[l]} * a^{[l-1]} + b^{[l]} \\ a^{[l[} &= g^{[l]}(z^{[l]}) \end{aligned} \tag{3.1}$$

In equation 3.1, $[l]$ stands for the l^{th} layer inside the network, $z^{[l]}$ is the result of linear transformation, and $a^{[l]}$ is the output of the l^{th} layer. For l is smaller than the total number of layers L, $g^{[l]}$ is chosen to be ReLU. But in the last layer, $g^{[l]}$ is the Softmax function for classification. Now, the input feature size is the only uncertain option.

3.1.1 One-residue-input logistic regression

Since we can only predict one residue's secondary structure class per time, a very natural idea is to use the row data corresponding to a residue in PSSM as the input features, because PSSM has already contained some useful information like the location probability and the amino acid type. The model looks like Figure 3.1. There are around 36000 ami no acid sequence in the dataset. Thus, we choose 30000 sequences randomly for training, which means we have around 30000*(100+50)/2 = 2.25 million training samples. We used the remaining data as test samples.

Here is the training procedure: after inputting a vector length of 20, the model will do forward propagation along with the hidden layers, which use linear transformation and followed by a RelU function to add in the non-linear ability. In the last layer, after the linear transformation we will use Softmax function and negative log-likelihood loss function as we introduced in Chapter 1, to compute the cost between our predicted $\hat{y}$ with the truth label y. Then we will do back propagation which means using gradient descent method to update the parameters of the model till them converge to the optimal state. When training is finished, then we will feed the input of test samples to our model, using the parameters we just trained to get the final prediction. In test time, after getting the Softmax result of last layer, we won't compute the cost again. Instead, we will choose the class which has the biggest prediction value directly as our final prediction label. The test accuracy is the number of correct predictions divided by the total number of test samples.

We set the model to be a three layer model, and the number of neurons in each layer [16, 12, number of classes]. Number of classes can be 3 or 8 depend on the classification problem is Q3 or Q8. In training stage, for the efficiency of learning, we used mini-batch[19] skill of size equals to 1024 and Adam[20] Optimizer, which is an improved version of gradient descent algorithm. Moreover, we set the learning rate to be 0.01 so that it won't take too long for parameters to converge. The number of epoch we chose is 300 for this model.

But when we were training this model, we found that no matter how we tuned my hyper-parameters, the cost between truth label and prediction value stopped decreasing after 50 epochs, and remained in a relatively high value. The best Q3 test accuracy we get from this model is around 56.44%, and the best Q8 test accuracy is 44.92%. At first we thought that there are some mistakes in the computation process, but after we used this model on some other tasks and gained a good result, we began to rethink about the model itself. What supervised learning does is digging through the data to find the pattern, or we can call it the internal rules of the data, then generate a model to help with the prediction task. So it works well when the input features have already contained enough useful information, like an image. Here we only used one row of PSSM as the input feature, which is similar to that we want predict the secondary structure class of a residue just based on its amino acid type. The information we gave the model is obviously not enough, therefore we built the improved version of this model.

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
1	G	0	-3	-1	-2	-3	-2	-2	6	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
2	I	-1	-3	-4	-4	-1	-3	-4	-4	-4	5	2	-3	1	0	-3	-3	-1	-3	-2	3
3	V	0	-3	-3	-4	-1	-3	-3	-4	-4	3	1	-3	1	-1	-3	-2	0	-3	-1	4
4	E	-1	0	0	3	-4	2	5	-2	0	-4	-3	1	-2	-4	-1	0	-1	-3	-2	-3
5	Q	-1	1	0	0	-3	6	3	-2	0	-3	-3	1	-1	-4	-2	0	-1	-2	-2	-3
6	C	-1	-4	-3	-4	10	-3	-4	-3	-3	-2	-2	-4	-2	-3	-3	-1	-1	-3	-3	-1
7	C	0	-4	-3	-3	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	0	-1	-3	-3	-1
8	T	0	-1	0	-1	-2	0	0	-2	6	-1	-2	-1	-1	-2	-2	0	3	-3	0	-1
9	S	0	0	2	0	-2	0	0	0	-1	-3	-3	2	-2	-3	-1	3	1	-3	-2	-2
10	I	-1	-1	-2	-3	-2	-2	-2	-3	-3	3	0	-2	0	-1	4	-1	0	-3	-2	2
11	C	-1	-4	-3	-4	10	-3	-4	-3	-3	-2	-2	-4	-2	-3	-3	-1	-1	-3	-3	-1
12	S	1	-1	2	0	-1	0	0	-1	-1	-3	-3	0	-2	-3	-1	4	2	-3	-2	-2
13	L	-2	-3	-4	-4	-1	-3	-3	-4	-3	3	4	-3	2	0	-3	-3	-1	-2	-1	2
14	Y	-2	-2	-3	-4	-3	-2	-3	-4	1	-1	-1	-2	-1	5	-4	-2	-2	2	7	-1
15	Q	-1	0	0	3	-4	5	3	-2	0	-3	-3	1	-1	-4	-2	0	-1	-3	-2	-3
16	L	-2	-2	-4	-4	-1	-2	-3	-4	-3	2	4	-3	2	0	-3	-3	-1	-2	-1	1
17	E	-1	0	0	1	-4	4	5	-2	0	-3	-3	1	-2	-4	-1	0	-1	-3	-2	-3
18	N	-2	-1	6	1	-3	0	0	-1	3	-4	-4	0	-2	-3	-2	0	0	-4	-2	-3
19	Y	-2	-2	-2	-4	-3	-2	-2	-4	2	-2	-1	-2	-1	3	-3	-2	-2	2	8	-1
20	C	0	-4	-3	-3	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	0	-1	-3	-3	-1
21	N	-2	-1	6	1	-3	0	0	-1	0	-4	-4	0	-2	-3	-2	1	0	-4	-2	-3

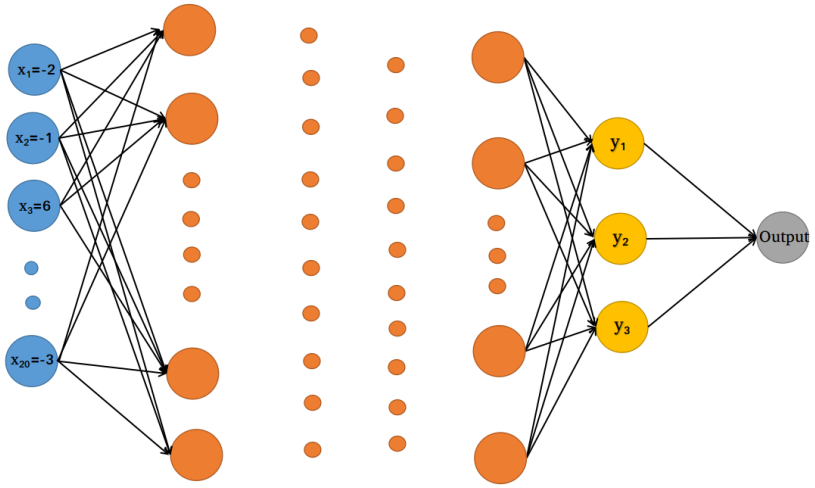


Figure 3.1: The model of one-residue-input logistic regression for 3-class

3.1.2 15-residue-input logistic regression

In the previous model we didn't get a satisfying result, and we believed that's due to the lacking of information in the input features. So, what about making the input feature bigger? Instead of giving only one row of PSSM which is corresponded to residue we want to predict, here we will also feed the residue information which is in the nearby location.

The number of neighbour residues to be included is another hyper-parameter needs to be tune. If we only consider about the adjacent residues, then the input feature is still relatively small. For example a helix structure is usually constructed by more than 5 residues. But if we take into account the residues which are too far, like if the index difference of the 2 residues is more than 20, then the number of parameters in the first several layers will be too large. Another thing we need to mention here is, for the residues which are located at the both ends of the sequences, we will pad the input with zeros when there are no nearby residues, to make sure that all the input features in for training and testing samples will have the same size. Thus, this is another reason for not considering a far neighbour, otherwise we will have to pad lots of zeros at both ends, which might hurt the performance of our model.

The procedure of training and test is similar to the model "one-residue-input logistic regression", except that we have a much larger input in this model. Lets

choose 15 as the number of residues we take into account as an example, which works the best in the test. 15 means we will collect the row of the residue needs to be predicted plus 2 * seven rows of the neighbour residues in both sides from PSSM. So, that will be a sub-matrix of size $(15, 20)$. To meet the requirement of input for logistic regression model, we will flatten it in to a vector of length equals to 300. The model is shown as below in Figure 3.2

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
1	G	0	-3	-1	-2	-3	-2	-2	6	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
2	I	-1	-3	-4	-4	-1	-3	-4	-4	-4	5	2	-3	1	0	-3	-3	-1	-3	-2	3
3	V	0	-3	-3	-4	-1	-3	-3	-4	-4	3	1	-3	1	-1	-3	-2	0	-3	-1	4
4	E	-1	0	0	3	-4	2	5	-2	0	-4	-3	1	-2	-4	-1	0	-1	-3	-2	-3
5	Q	-1	1	0	0	-3	6	3	-2	0	-3	-3	1	-1	-4	-2	0	-1	-2	-2	-3
6	C	-1	-4	-3	-4	10	-3	-4	-3	-3	-2	-2	-4	-2	-3	-3	-1	-1	-3	-3	-1
7	C	0	-4	-3	-3	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	0	-1	-3	-3	-1
8	T	0	-1	0	-1	-2	0	0	-2	6	-1	-2	-1	-1	-2	-2	0	3	-3	0	-1
9	S	0	0	2	0	-2	0	0	0	-1	-3	-3	2	-2	-3	-1	3	1	-3	-2	-2
10	I	-1	-1	-2	-3	-2	-2	-2	-3	-3	3	0	-2	0	-1	4	-1	0	-3	-2	2
11	C	-1	-4	-3	-4	10	-3	-4	-3	-3	-2	-2	-4	-2	-3	-3	-1	-1	-3	-3	-1
12	S	1	-1	2	0	-1	0	0	-1	-1	-3	-3	0	-2	-3	-1	4	2	-3	-2	-2
13	L	-2	-3	-4	-4	-1	-3	-3	-4	-3	3	4	-3	2	0	-3	-3	-1	-2	-1	2
14	Y	-2	-2	-3	-4	-3	-2	-3	-4	1	-1	-1	-2	-1	5	-4	-2	-2	2	7	-1
15	Q	-1	0	0	3	-4	5	3	-2	0	-3	-3	1	-1	-4	-2	0	-1	-3	-2	-3
16	L	-2	-2	-4	-4	-1	-2	-3	-4	-3	2	4	-3	2	0	-3	-3	-1	-2	-1	1
17	E	-1	0	0	1	-4	4	5	-2	0	-3	-3	1	-2	-4	-1	0	-1	-3	-2	-3
18	N	-2	-1	6	1	-3	0	0	-1	3	-4	-4	0	-2	-3	-2	0	0	-4	-2	-3
19	Y	-2	-2	-2	-4	-3	-2	-2	-4	2	-2	-1	-2	-1	3	-3	-2	-2	2	8	-1
20	C	0	-4	-3	-3	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	0	-1	-3	-3	-1
21	N	-2	-1	6	1	-3	0	0	-1	0	-4	-4	0	-2	-3	-2	1	0	-4	-2	-3

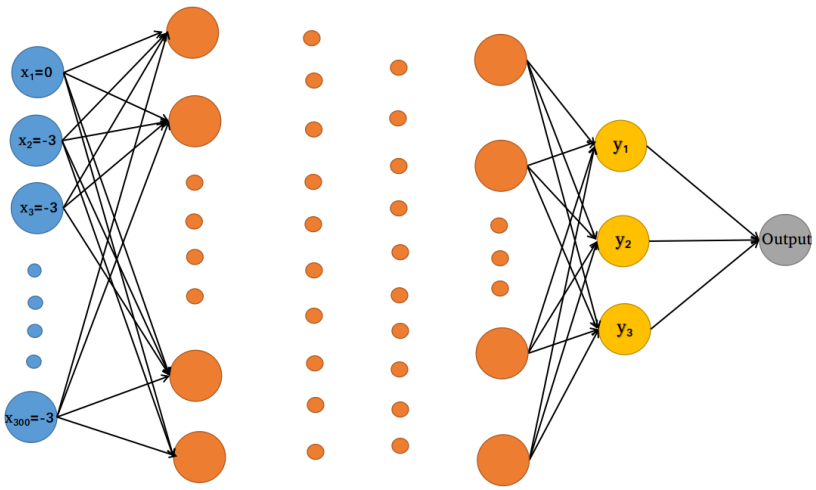


Figure 3.2: The model of 15-residue-input logistic regression for 3-class

As you can see, in the example we are trying to predict the secondary structure for the 10^{th} residue I. For the 15-residue-input logistic regression model, we used not only the row of itself in PSSM, but also the 14 neighbour rows in PSSM, and flattened it into a vector of length 300. Thus, we will do some pre-processing for the input features based on the origin PSSM in both training set and test set. Because the input size is much bigger than the previous one, we built a 4 layer model. The number of neurons in each layer [150, 60, 20, number of classes]. Number of classes can be 3 or 8 depend on the classification problem is Q3 or Q8. Mini-batch size is 512, learning rate is still 0.01, and we set the number of training epoch to be 1000 to ensure that the new model would be fully trained.

The result shows that Our guess in Chapter 3.1.1 is correct, that the poor performance of one-residue-input logistic regression model is caused by insufficient information of input. For test set, using this new model our Q3 accuracy rate reached 82.89%, and the Q8 accuracy rate was 71.00%, which is already close to the best results pub-

lished by others. But there is a upper limit for the performance in this model. Because we manually choose the number of neighbour residues, there may exist some residues located far away but actually have influence on the target residue which are not being considered. Thus, although the 15-residue-input logistic regression model works pretty well, we still want to build some better model which can overcome this problem.

3.2 Convolutional neural network

CNN (Concolutional neural network) is a type of model which is wildly used in computer vision field. In 1989, Yann LeCun was the first to build a CNN and used it in handwritten zip code recognition [21] successfully. But back to that time, the computing power was not big enough to support the huge amount of calculation needed in the process of back propagation. After 2010, with the rise of GPU(Graphics Processing Unit), which is perfect for large-scale matrix multiplication, CNN is finally able to show the world that how capable it is in tasks like image classification, object detection, face recognition, etc. In image classification problem, the input of a CNN model is always an image with is represented by a certain size matrix‘‘ and the output is the label for the input image. That reminds me of our protein secondary structure prediction task.

Since we have already encoded our amino acid sequence with the PSSM, we can choose a sub-matrix which is centered on the target residue to be the input of CNN. Then we just treat it like a image classification problem‘‘‘‘‘‘‘ , and the output of our CNN model is the prediction of the secondary structure label for our target residue. The training procedure is similar to logistic regression. But between every 2 layer, instead of using fully connected structure, we use a square filter doing convolutions on the feature map of the previous layer to generate the next layer. A sketch of the model is given in Figure 3.3.

Here we use a 4 layer CNN model. The structure of network is: $Input->Conv1(8) -> Pool1 -> Conv2(16) -> Pool2 -> Conv3(32) -> Pool3 -> Flatten -> Fc4(3/8) -> Softmax$. The number in parentheses represents the number of channels in the layer. "Conv" represents for a convolution layer, "Pool" represents for a max pooling layer, "Flatten" means that we flattened the feature map into a vector, and "Fc" stands for a fully-connected layer. Because there is no parameter in the max pooling layer, we don't consider it as an independent layer. The filter size of the first layer is (4,4), and in the second and third layer it's (2,2).

Similar to 15-residue-input logistic regression model, we padded rows of 0 at both ends of the PSSM to make sure every target residue can have an input matrix of size (15, 20). Then, we combined each (15, 20) matrix with the truth label of the target residue as a training/test sample. Number of epochs is selected to be 300, learning rate is 0.01 and mini-batch size is 0.01. After the parameters are trained, we test it with test samples. The performance of CNN model is not so good. On test set, its Q3 accuracy is 72.90% and Q8 accuracy is 59.12%.

After trying several hyper-parameter combinations but still not getting the desired results, we stopped the process of tuning the hyper-parameters and started thinking

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
1	G	0	-3	-1	-2	-3	-2	-2	6	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
2	I	-1	-3	-4	-4	-1	-3	-4	-4	-4	5	2	-3	1	0	-3	-3	-1	-3	-2	3
3	V	0	-3	-3	-4	-1	-3	-3	-4	-4	3	1	-3	1	-1	-3	-2	0	-3	-1	4
4	E	1	0	0	3	-4	2	5	-2	0	-4	-3	1	-2	-4	-1	0	-1	-3	-2	-3
5	Q	1	1	0	0	-3	6	3	-2	0	-3	-3	1	-1	-4	-2	0	-1	-2	-2	-3
6	C	1	-4	-3	-4	10	-3	-4	-3	-3	-2	-2	-4	-2	-3	-3	-1	-1	-3	-3	-1
7	C	0	-4	-3	-3	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	0	-1	-3	-3	-1
8	T	0	-1	0	-1	-2	0	0	-2	6	-1	-2	-1	-1	-2	-2	0	3	-3	0	-1
9	S	0	0	2	0	-2	0	0	0	-1	-3	-3	2	-2	-3	-1	3	1	-3	-2	-2
10	I	-1	-1	-2	-3	-2	-2	-2	-3	-3	3	0	-2	0	-1	4	-1	0	-3	-2	2
11	C	-1	-4	-3	-4	10	-3	-4	-3	-3	-2	-2	-4	-2	-3	-3	-1	-1	-3	-3	-1
12	S	1	-1	2	0	-1	0	0	-1	-1	-3	-3	0	-2	-3	-1	4	2	-3	-2	-2
13	L	-2	-3	-4	-4	-1	-3	-3	-4	-3	3	4	-3	2	0	-3	-3	-1	-2	-1	2
14	Y	-2	-2	-3	-4	-3	-2	-3	-4	1	-1	-1	-2	-1	5	-4	-2	-2	2	7	-1
15	Q	-1	0	0	3	-4	5	3	-2	0	-3	-3	1	-1	-4	-2	0	-1	-3	-2	-3
16	L	-2	-2	-4	-4	-1	-2	-3	-4	-3	2	4	-3	2	0	-3	-3	-1	-2	-1	1
17	E	-1	0	0	1	-4	4	5	-2	0	-3	-3	1	-2	-4	-1	0	-1	-3	-2	-3
18	N	-2	-1	6	1	-3	0	0	-1	3	-4	-4	0	-2	-3	-2	0	0	-4	-2	-3
19	Y	-2	-2	-2	-4	-3	-2	-2	-4	2	-2	-1	-2	-1	3	-3	-2	-2	2	8	-1
20	C	0	-4	-3	-3	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	0	-1	-3	-3	-1
21	N	-2	-1	6	1	-3	0	0	-1	0	-4	-4	0	-2	-3	-2	1	0	-4	-2	-3

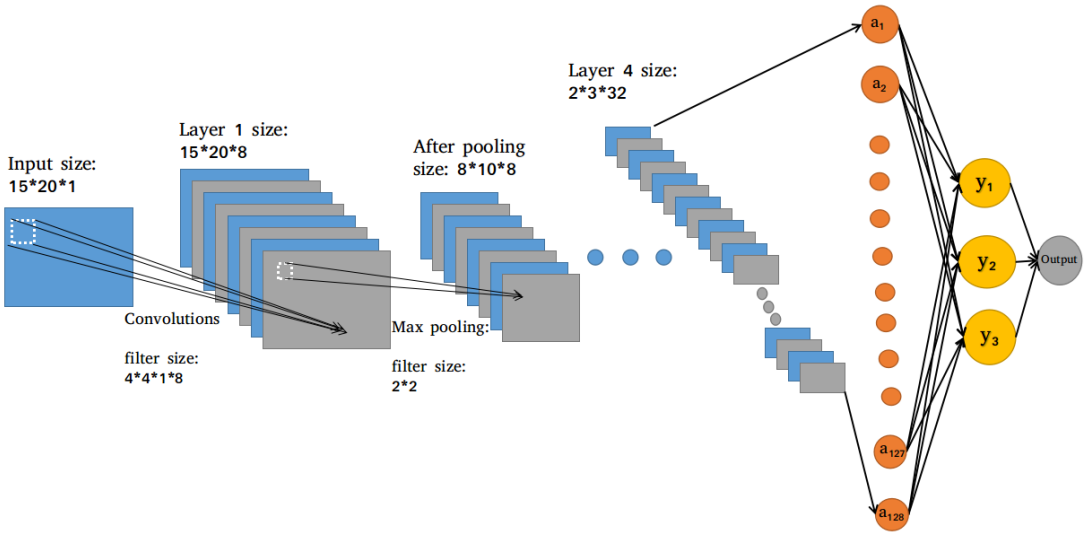


Figure 3.3: The CNN model of 3-class prediction

about the limitations of the CNN model itself. One big reason for CNN working well in computer vision tasks is that the features in different layers contain different scales of useful information. In the first several layers, CNN can detect straight lines or simple curves. Based on the combination of these small details, in the next few layers can detect CNN can detect relatively complex figures such as circles or squares. It is the property of information transferring from small scale to large scale inside the CNN model that makes CNN being so powerful at recognizing complex objects and scenes.

But the PSSM we feed into CNN doesn't has such property. Using a square filter not only can't capture useful information of the residues, but also makes it harder get the entire information of a residue which is stored in one row of PSSM. Moreover, this CNN model can't solved the problem of long-term dependency which occurs in the logistic regression model. Thus, we stopped spending our effort on tuning hyper-parameter of CNN, and started to working on the next model which is more powerful in processing sequence.

3.3 Recurrent neural network

RNN (Recurrent neural network) is a structure that widely used in solving sequence problem, such as speech recognition, machine translation, sentiment classification, etc. In both logistic regression model and CNN model, the input size must be same in@inproceedingsConte2012dmpc, author = Conte, C. and Voellmy, N. R. and Zeilinger, M. N. and Morari, M. and Jones, C. N., booktitle = Proc. American Control Conference , title = Distributed synthesis and control of constrained linear systems, year = 2012

@inproceedingsZeilinger2014pnp, author = Zeilinger, M. N. and Pu, Y. and Riverso, S. and Ferrari-Trecate, G. and Jones, C. N., booktitle = Proc. IEEE Conf. Decision and Control, title = Plug and play distributed model predictive control based on distributed invariance and optimization, year = 2013

@incollectionbengio2012practical, title=Practical recommendations for gradient-based training of deep architectures, author=Bengio, Yoshua, booktitle=Neural networks: Tricks of the trade, pages=437–478, year=2012, publisher=Springer

@Misctimmurphy.org, howpublished = `http://timmurphy.org/2009/07/22/line-spacing-in-latex-documents/`, title = Secondary Structure Files, author = Murphy, Timothy I

@miscpdbsequence, author = Research Collaboratory for Structural Bioinformatics: Rutgers and UCSD/SDSC, title = Structure determined by He and Carter, Protein Data Bank entry 1UOR, howpublished = "`https://www.rcsb.org/pages/download_features#FASTA`", note = "[Online; accessed 20-May-2018]"

@misclstm, author = kijungyoon, title = Differentiable Neural Computer, howpublished = "`https://kijungyoon.github.io/DNC/`", note = "[Online; accessed 20-May-2018]"

@articlehochreiter1997long, title=Long short-term memory, author=Hochreiter, Sepp and Schmidhuber, Jürgen, journal=Neural computation, volume=9, number=8, pages=1735–1780, year=1997, publisher=MIT Press

@miscsstxt, author = kabschSander, title = Protein secondary structure file, howpublished = "`https://cdn.rcsb.org/etl/kabschSander/ss.txt.gz`", note = "[Online; accessed 12-Mar-2018]"

@articlebaldi1999exploiting, title=Exploiting the past and the future in protein secondary structure prediction, author=Baldi, Pierre and Brunak, Søren and Frasconi, Paolo and Soda, Giovanni and Pollastri, Gianluca, journal=Bioinformatics, volume=15, number=11, pages=937–946, year=1999, publisher=Oxford University Press

@articletorrisi2018porter, title=Porter 5: state-of-the-art ab initio prediction of protein secondary structure in 3 and 8 classes, author=Torrisi, Mirko and Kaleel, Manaz and Pollastri, Gianluca, journal=bioRxiv, pages=289033, year=2018, publisher=Cold Spring Harbor Laboratory

@articleviterbi1967error, title=Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, author=Viterbi, Andrew, journal=IEEE transactions on Information Theory, volume=13, number=2, pages=260–269, year=1967, publisher=IEEE

@articlelafferty2001conditional, title=Conditional random fields: Probabilistic models for segmenting and labeling sequence data, author=Lafferty, John and McCallum,

Andrew and Pereira, Fernando CN, year=2001

@bookkindermann1980markov, title=Markov random fields and their applications, author=Kindermann, Ross and Snell, J Laurie, volume=1, year=1980, publisher=American Mathematical Society

@articlesrivastava2014dropout, title=Dropout: A simple way to prevent neural networks from overfitting, author=Srivastava, Nitish and Hinton, Geoffrey and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan, journal=The Journal of Machine Learning Research, volume=15, number=1, pages=1929–1958, year=2014, publisher=JMLR. org

@articlejones1999protein, title=Protein secondary structure prediction based on position-specific scoring matrices1, author=Jones, David T, journal=Journal of molecular biology, volume=292, number=2, pages=195–202, year=1999, publisher=Elsevier

@articlewang2016protein, title=Protein secondary structure prediction using deep convolutional neural fields, author=Wang, Sheng and Peng, Jian and Ma, Jianzhu and Xu, Jinbo, journal=Scientific reports, volume=6, pages=18962, year=2016, publisher=Nature Publishing Group

@articlewang2016raptorx, title=RaptorX-Property: a web server for protein structure property prediction, author=Wang, Sheng and Li, Wei and Liu, Shiwang and Xu, Jinbo, journal=Nucleic acids research, volume=44, number=W1, pages=W430–W435, year=2016, publisher=Oxford University Press

@miscblast, author = National Center for Biotechnology Information, U.S. National Library of Medicine, title = Basic Local Alignment Search Tool, howpublished = "`https://blast.ncbi.nlm.nih.gov/Blast.cgi`", note = "[Online; accessed 12-Mar-2018]"

https://blast.ncbi.nlm.nih.gov/Blast.cgi

@misc3dpicture, author = He and Carter, title = Structure determined by He and Carter, Protein Data Bank entry 1UOR, howpublished = "`http://www.acid-base.org/xraycrystalstructure.html`", year = 1992, note = "[Online; accessed 19-May-2018]"

@articlekingma2014adam, title=Adam: A method for stochastic optimization, author=Kingma, Diederik P and Ba, Jimmy, journal=arXiv preprint arXiv:1412.6980, year=2014

@articlelecun1989backpropagation, title=Backpropagation applied to handwritten zip code recognition, author=LeCun, Yann and Boser, Bernhard and Denker, John S and Henderson, Donnie and Howard, Richard E and Hubbard, Wayne and Jackel, Lawrence D, journal=Neural computation, volume=1, number=4, pages=541–551, year=1989, publisher=MIT Press

@articlebishop2006periodic, title=Periodic Variables, author=Bishop, Christopher M, journal=Pattern recognition and machine learning, volume=1, year=2006, publisher=Springer Science+ Business Media, LLC New York

@inproceedingsglorot2011deep, title=Deep sparse rectifier neural networks, author=Glorot, Xavier and Bordes, Antoine and Bengio, Yoshua, booktitle=Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pages=315–323, year=2011

@articlestormo1982use, title=Use of the ‘‘Perceptron‘‘algorithm to distinguish

translational initiation sites in E. coli, author=Stormo, Gary D and Schneider, Thomas D and Gold, Larry and Ehrenfeucht, Andrzej, journal=Nucleic acids research, volume=10, number=9, pages=2997–3011, year=1982, publisher=Oxford University Press

@articlecox1958regression, title=The regression analysis of binary sequences, author=Cox, David R, journal=Journal of the Royal Statistical Society. Series B (Methodological), pages=215–242, year=1958, publisher=JSTOR

@articledelorenzi2002hmm, title=An HMM model for coiled-coil domains and a comparison with PSSM-based predictions, author=Delorenzi, Mauro and Speed, Terry, journal=Bioinformatics, volume=18, number=4, pages=617–625, year=2002, publisher=Oxford University Press

@articlelecun1998gradient, title=Gradient-based learning applied to document recognition, author=LeCun, Yann and Bottou, Léon and Bengio, Yoshua and Haffner, Patrick, journal=Proceedings of the IEEE, volume=86, number=11, pages=2278–2324, year=1998, publisher=IEEE

@articlede2005tutorial, title=A tutorial on the cross-entropy method, author=De Boer, Pieter-Tjerk and Kroese, Dirk P and Mannor, Shie and Rubinstein, Reuven Y, journal=Annals of operations research, volume=134, number=1, pages=19–67, year=2005, publisher=Springer

@articleML for Protein, title=Machine Learning Methods for Protein Structure Prediction, author=Jianlin Cheng, Allison N. Tegge, Member, IEEE, and Pierre Baldi, Senior Member, IEEE, journal=IEEE Reviews in Biomedical Engineering, year=2008,

@inproceedingskrizhevsky2012imagenet, title=Imagenet classification with deep convolutional neural networks, author=Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E, booktitle=Advances in neural information processing systems, pages=1097–1105, year=2012

@articleitoh2006flexibly, title=Flexibly varying folding mechanism of a nearly symmetrical protein: B domain of protein A, author=Itoh, Kazuhito and Sasai, Masaki, journal=Proceedings of the National Academy of Sciences, volume=103, number=19, pages=7298–7303, year=2006, publisher=National Acad Sciences

@articledill2012protein, title=The protein-folding problem, 50 years on, author=Dill, Ken A and MacCallum, Justin L, journal=science, volume=338, number=6110, pages=1042–1046, year=2012, publisher=American Association for the Advancement of Science

@articlemccammon1977dynamics, title=Dynamics of folded proteins, author=McCammon, J Andrew and Gelin, Bruce R and Karplus, Martin, journal=Nature, volume=267, number=5612, pages=585, year=1977, publisher=Nature Publishing Group

@articlecheng2008machine, title=Machine learning methods for protein structure prediction, author=Cheng, Jianlin and Tegge, Allison N and Baldi, Pierre, journal=IEEE reviews in biomedical engineering, volume=1, year=2008, publisher=IEEE

@articlekabsch1983dictionary, title=Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features, author=Kabsch, Wolfgang and Sander, Christian, journal=Biopolymers, volume=22, number=12, pages=2577–2637, year=1983, publisher=Wiley Online Library

@articlefariselli2001prediction, title=Prediction of contact maps with neural networks and correlated mutations, author=Fariselli, Piero and Olmea, Osvaldo and Valencia, Alfonso and Casadio, Rita, journal=Protein engineering, volume=14, number=11,

pages=835–843, year=2001, publisher=Oxford University Press

@articlehelloworld, title=I love ilana, author=emon fangb, journal=IEEE Reviews in Biomedical Engineering, pages=79-93, year=2012,

@mischochreiter2001gradient, title=Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, author=Hochreiter, Sepp and Bengio, Yoshua and Frasconi, Paolo and Schmidhuber, Jürgen and others, year=2001, publisher=A field guide to dynamical recurrent neural networks. IEEE Press

@inproceedingsLatafat18, author=Latafat, Puya and Bemporad, Alberto and Patrinos, Panagiotis, booktitle = Proc. European Control Conference, title = Plug and Play Distributed Model Predictive Control with Dynamic Coupling: A Randomized Primal-dual Proximal Algorithm, year = 2018 all training samples. But in reality, the length of sentences or other sequence data can vary a lot. Thus, recurrent neural network is a type of model that can solve unfixed length sequence problem, which looks very suitable for protein secondary structure prediction. A basic RNN structure is shown in Figure 3.4, and the formula is listed under the figure.

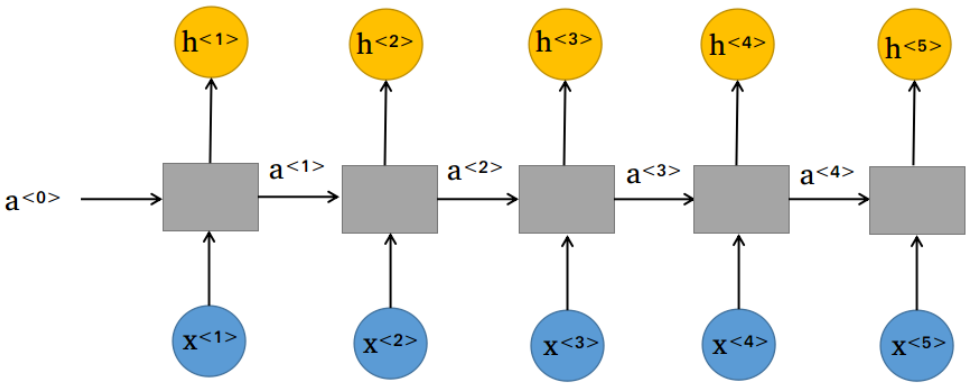


Figure 3.4: The basic RNN structure

$$\begin{aligned} a^{<t>} &= g_1(\, w_a\, [\, a^{<t-1>},\, x^{<t>}]\, + b_a) \\ h^{<t>} &= g_2(\, w_y\, *\, a^{<t>}\, +\, b_y\,) \end{aligned} \tag{3.2}$$

Here $< t >$ means the t^{th} time step. g_1 is usually chosen to be tanh or ReLU function, g_2 is can be sigmoid or Softmax function depend on the task. As you can see in the formula, in the forward process, we not only concern about the the input x in the current step, but also take the activation value of last time step into account. That is the biggest difference between RNN and the previous models, and makes RNN powerful at solving sequence tasks. However, there are two shortcomings in using the standard RNN structure to deal with protein secondary structure prediction problems. Firstly, standard RNN can well reflect the local influence from the nearby residue, but it can not capture the long-pendency because of the vanish gradients[22] problem. Secondly, in standard RNN we only consider the influence from on side, which is usually called previous time step. However, for the amino acid sequence, there is no concept of direction. The adjacent residue of residue from both side should be treated equally. Thus, we decided to try the upgraded version of standard RNN, Bidirectional long short-term memory model.

3.3.1 Deep Bi-LSTM

Bi-LSTM, which is abbreviate of Bidirectional long short-term memory model. A diagram for the neuron in long short-term is shown in Figure 3.5[23]. A neuron in LSTM has two states inside a neuron. h_t is the hidden state which already existed in standard RNN, while c_t is new cell state of a neuron which only pass through between neurons. In order to avoid vanish gradient problem, the cell state is created to store useful information which pass from the neuron located far away to the target neuron. Thus, there are 3 gating variables help controlling the information flow. With this kind of structure, the network is able to store the long term dependency inside the cell state and which improves the performance in long sequence. The details of LSTM can be find in this paper[24].

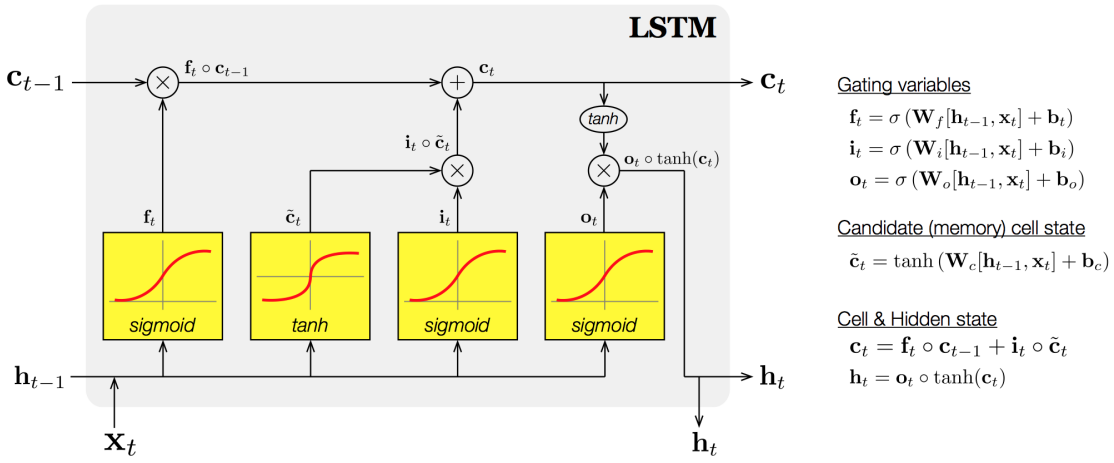


Figure 3.5: A neuron of LSTM [23]

With the knowledge of LSTM, we can build our model now. As we described in the previous part, we want to make the target residue gain information from both side, thus, we will build a Bidirectional LSTM model. And, in case that one layer of Bi-LSTM is not strong enough to process the dependency between residues, we will stack several Bi-LSTM layers together. In addition, after getting the output hidden states of last Bi-LSTM layers, we will add some additional fully connected layers instead of transforming it into 3 classes directly. The diagram of our entire "Deep Bi-LSTM" network is shown in Figure 3.6, and the modification of equation 3.2 due to Bi-direction is shown in equation 3.3. One should know that the parameters in the same layer are shared in different neurons, which is the property of all RNN models.

$$h^{<t>} = g\left(w_y\left[\overrightarrow{a^{<t>}}, \overleftarrow{a^{<t>}}\right] + b_y\right) \tag{3.3}$$

In the diagram, the number inside the square brackets is the number of layer, and the number inside the angle brackets is the index of residue. The number of total index steps is the length of an amino acid sequence. Similarly, what we input to the model is the corresponding row of the a residue in PSSM for every index step. After several Bi-LSTM layer, the output is a vector which length equals to the hidden dimension we

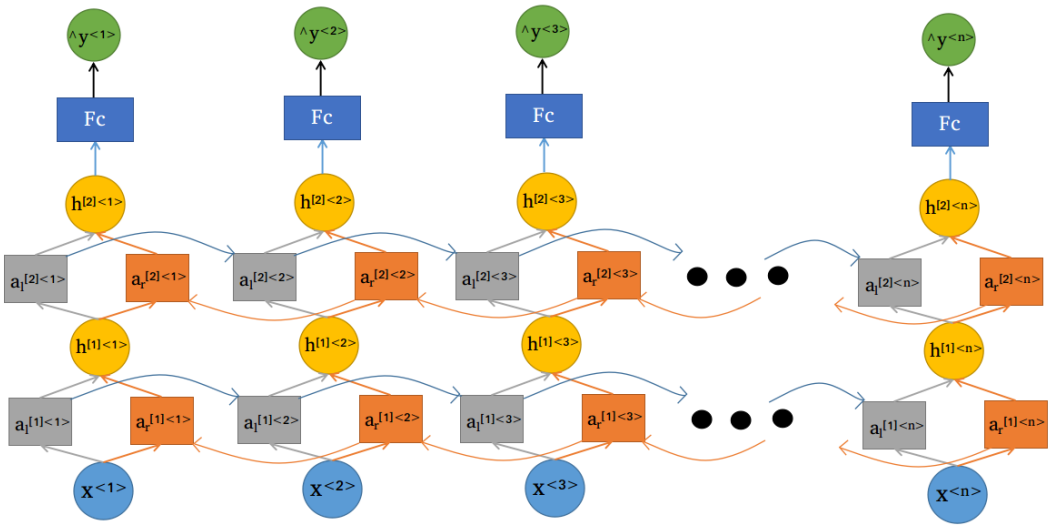


Figure 3.6: The sketch of our Deep Bi-LSTM

set. Then, we connect it with several fully connected layers and which output a vector whose length is the number of secondary structure classes in the end. The remaining procedure is same as model 3.1, Softmax function and negative log-likelihood loss function is used, followed by the back propagation to update the parameters till they converge.

The performance of our deep Bi-lstm model in test set is quite impressive. The best Q3 accuracy is 84.38%, with the hyper-parameters set as: the number of Bi-LSTM layers is 3, hidden dim in LSTM equals to 128, training epochs equal to 300, and a stochastic gradient descent optimizer with learning rate equals to 0.1, momentum equals to 0.8, and weight decay equals to 1^{-6}. The best Q8 accuracy is 73.49%, achieved by a more complex model as follow hyper=parameters: the number of Bi-LSTM layer is 4, hidden dim in LSTM equals to 128, the dimension of additional layer in fully connected layers is 32, the training epochs equal to 1000 and with a dropout [25] rate of 0.2. The hyper-parameter of the optimizer is same as previous one: a stochastic gradient descent optimizer with learning rate equals to 0.1, momentum equals to 0.8, and weight decay equals to 1^{-6}. Both of the Q3 accuracy and Q8 accuracy reached, and even slightly surpassed predictive accuracy of the present state of the art. So, our new goal is to improve it substantially. We learned from a very powerful model in natural language processing''which combined Bi-LSTM and conditional random field together, and used it to our protein secondary structure prediction problem.

3.3.2 Bi-LSTM + CRF

Conditional random field [26], full name of CRF, is a type of discriminative undirected probabilistic graphical model. A probabilistic graphical model is a kind of probabilistic model that expresses the dependencies between random variables in the form of graphs. If the edges in the graph are undirected and the probability distribution of a random variable is only relevant to its neighbors, this kind of probabilistic graphical model is called Markov random field [27]. If there are observations given below each random variable in a Markov random field, and we need to determine the conditional

distribution of this Markov random field given the observation set, then this type of Markov random field is called is called Conditional random field.

Similar to sequence labeling problem in natural language processing, our protein secondary structure prediction problem is a linear sequence structure. Thus, we only need to consider the linear chain CRF. The diagram of an example liner chain CRF is shown in Figure 3.7.

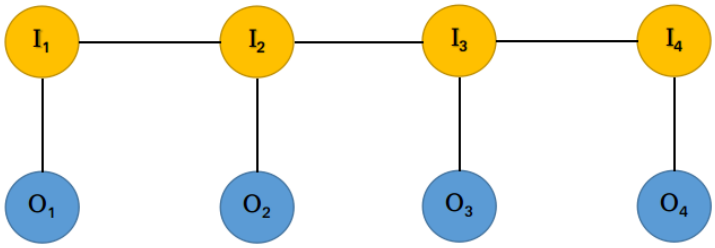


Figure 3.7: linear chain CRF

Here, I is the hidden variable and O is the observation variable. The model of linear chain CRF can be represent as the following equation. $Z(O)$ is the normalization coefficient, $\psi_i(I_i|O)$ is called potential function. And k is the number of random variables in a clique.

$$P(I|O) = \frac{1}{Z(O)} \prod_i \psi_i(I_i|O) = \frac{1}{Z(O)} \prod_i e^{\sum_k \lambda_k f_k(O, I_{i-1}, I_i, i)} \tag{3.4}$$

Moreover, the potential function can be decomposed into two parts as shown in equation 3.5: transition function and state function, we will explain it more in our prediction model.

$$P(I|O) = \frac{1}{Z(O)} \prod_i e^{\sum_i^T \sum_j^J \lambda_j t_j(O, I_{i-1}, I_i, i) + \sum_i^T \sum_l^L \mu_l s_l(O, I_i, i)} \tag{3.5}$$

linear CRF works well on sequence tasks such as Part-of-Speech tagging. Because it can capture the relationship between the adjacency hidden random variables well and avoid some impossible state of output tags (for example verb adjacent to a verb), which is hard for Bi-LSTM to learn. Thus, CRF is like a supplementary tool for Bi-LSTM structure. So, we want to have a try of the combination of these two tools. In our "Bi-LSTM + CRF" model, a CRF layer is connected to the output of output hidden state of Bi-LSTM layer, instead of using Softmax function to classify every residue directly. The diagram is shown in Figure 3.8.

To explain it more clearly, after getting the output of hidden state $h^{<i>}$ from last Bi-LSTM layer, we first did a non-linear transformation to project it into the a vector $v^{<i>}$ which length equals to target size (which equals to the number of prediction classes plus two, including the $start\_tag$ and $stop\_tag$). $v^{<i>}$ can be think as the emission score of residue $x^{<i>}$ projecting to different prediction labels. By stacking all the $v^{<i>}$ vertically, we got a emission matrix P presenting the state function, in which $P_{i,j}$ stands for the non-normalization probability of residue $x^{<i>}$ projecting to $structure_j$. Then we considered the transition function. In CRF layer, there is a transition matrix

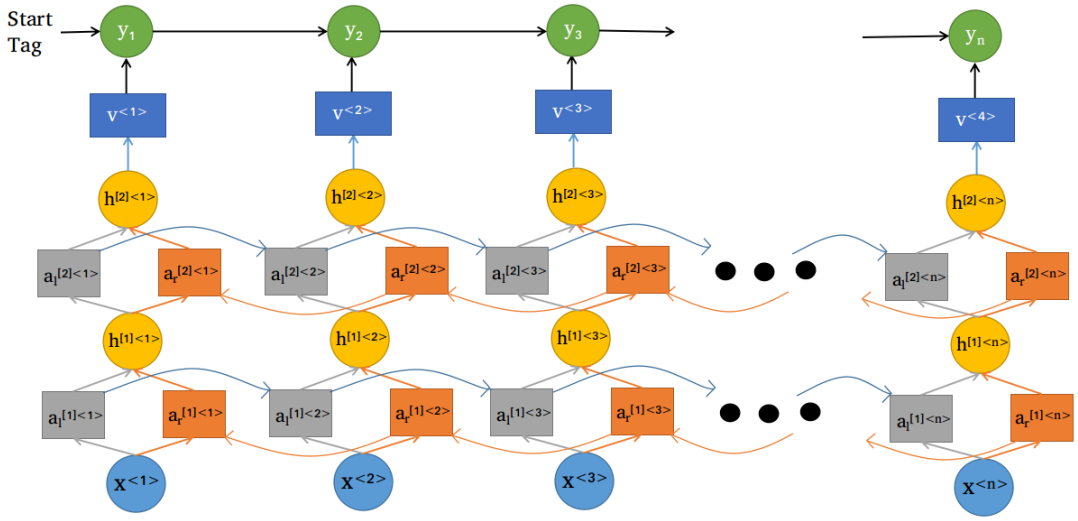


Figure 3.8: Bi-LSTM + CRF

A, in which $A_{i,j}$ represents for the score of transitioning from $structure_i$ to $structure_j$. Thus, for an input amino acid sequence X and its corresponding secondary structure label sequence y, we define a score as below:

$$s(X,y) = \sum_{i=1}^{n} A_{y_i,y_{i+1}} + \sum_{i=1}^{n} P_{i,y_i} \tag{3.6}$$

Using Softmax function, we can define the probability for the true secondary structure sequence as below. Here Y_X represents for all the possible predicting secondary structure, including those which is impossible in reality.

$$p(y|X) = \frac{e^{s(X,y)}}{\sum_{y' \in Y_X} e^{s(X,y')}} \tag{3.7}$$

In training stage, we just need to maximize the probability of $p(y|X)$. Here we can use log-likelihood function again as below:

$$\log(p(y|X)) = \log\left(\frac{e^{s(X,y)}}{\sum_{y' \in Y_X} e^{s(X,y')}}\right) = s(X,y) - \log\left(\sum_{y' \in Y_X} e^{s(X,y')}\right) \tag{3.8}$$

Similarly, we defined the loss function as $-log(p(y|X))$, and used back propagation to learn the parameters in Bi-LSTM and CRF. When training stage is finished, we used the Viterbi algorithm[28] to find the most possible prediction of secondary structure for the amino acid sequences in test set.

However, "BiLSTM+CRF" model did not work so well in protein secondary structure prediction problem. On test set, the best Q3 accuracy we get is 74.26% and the best Q8 accuracy is 59.98%. We think there are 2 main differences exist between natural language processing tasks and our protein prediction task. Firstly, the tag sequence of a sentence varies frequently in every one or two words, which means the transition between tags happen a lot in NLP problem. But in protein secondary structure, the

length of alpha helices or beta sheets is usually longer than 7, which make the score of remaining the current structure is much larger than the score of transiting to other structure in the transition matrix. This phenomenon leads to a reduction of the model's performance caused by when some secondary structure should have been transformed but being ignored in the prediction. Secondly, the length of sentence in NLP is usually less than 30 words. But the length of our amino acid sequence samples are all above 50, which are several times longer than a sentence. This makes the probability of the truth secondary structure relatively small, because there are too many possible state of secondary structure. Thus, the model for our task is much harder to learn compared with a Part-of-Speech tagging problem. Another disadvantage of this model is that it takes too long for every training epoch. When computing the loss of our prediction in training stage, we need compute all possible states as shown in equation 3.8. It costs us several days to train a model just for once, which makes it hard to tune the hyper-parameters and find the best model. Thus we come to the conclusion that "Bi-LSTM + CRF" is not a proper model for protein secondary structure prediction problem.

Chapter 4 Results and discussion

In Chapter 3, we have introduced all the 5 models we built for the protein secondary structure prediction task, based on 3 different structures. Also, we showed their performances on our test set and gave some analyses for each model. In this chapter, we will make a table to show it more clearly, and then give some comparisons between different models to show the pros and cons of each model. Furthermore, we will list some other models which are state-of-the-art to give readers a intuitive feeling about our model's performance.

Table 4.1: Our model's performance on PDB secondary structure file

Model	Q3	Q8
Deep Bi-LSTM	84.38%	73.49%
15-residue-input logistic regression	82.89%	71.00%
Bi-LSTM + CRF	74.26%	59.98%
Convolutional neural network	72.90%	59.12%
One-residue-input logistic regression	56.44%	44.92%

In Table 4.1, our models are listed according to their performance from the best to the worst. To make the description clear, Q3 stands for the 3 classes secondary structure classification and Q8 stands for the 8 classes secondary structure classification. Our training set and test set are both built on the second structure file[14] downloading from protein data bank.

As shown in the table, there are 2 models obviously surpass the other 3 models, "15-residue-input logistic regression" and "Deep Bi-LSTM". We will not analyze "One-residue-input logistic regression" more because it is dominated by 15-residue-input logistic regression and the structure of them are quite similar except for the input.

Firstly, consider model "Convolutional neural network". For every residue, we cut out the $(15, 20)$ sub-matrix from PSSM which centered at the target residue as the input for the model, and treat it like a image classification problem. The main reason of its poor performance is that in PSSM, there is no different scales of information like which exist in a photo. Also, using a square feature detector is not a good idea because the concept of "parameters sharing" in computer vision is not working on a PSSM. Instead, maybe one should use different one-dimension filters on different rows of input, because the residue of different distance should be process separately. We will realize this idea in the future work. In a word, CNN has its limitation on the task

of protein secondary structure prediction. One need to design the structure carefully based on the input feature instead of transferring the powerful models from computer vision tasks.

Secondly, consider model "15-residue-input logistic regression". This is the second best model we built. The high accuracy is quite impressive compared with its simple structure. Input of this model is a vector size of $(300, 1)$. And the procedure of training is straightforward. In forward propagation, there are several fully connected layers followed by a Softmax function, then the negative log-likelihood function is used to compute the loss. Afterwards, gradient descent method is applied in the back propagation till the parameters converge. The time required to train a model is also very short, which took only two or three hours. So we can try many different combinations of the hyper-parameters to get the best model. This is a huge advantage compared to the two remaining Bi-LSTM based models. But the cons of this model is obvious too. First, we have to manually select the number of neighbour residues to be included in the input feature. The only method is to choose a set of numbers and try them one by one. Also, for every different size of input, we need to build a new training/test dataset to meet the constraint of input feature size. These 2 cons make the total time to find the best model much longer than we thought. Generally speaking its a good model, because it proved that a good model does not have to be complicated. Without losing much precision, this model has the simplest structure and smallest number of parameters. But on the other hand, the concept of the neighbor node makes it impossible for this logistic regression model to be an optimal model, for its weakness on capturing long-term influence.

At last, the two model based on Bi-LSTM will be discussed together. Although there is a big gap in their performance, the basic structure of these two model are similar to each other. In both Bi-LSTM based model, the corresponding row vector of a residue in PSSM is the input feature for a time step. And, because of the architecture of LSTM, the number of input features can vary according to the length of the amino acid sequence. Thus, we don't need to pre-process the PSSM again to ensure that all the training/test sample have the same length input. This is the biggest advantage of using RNN models to solve sequence problems, including our protein second structure prediction problem. The difference between these two models occurs on the second half model. In model "Deep Bi-LSTM", the output feature of stacked Bi-LSTM layers is connected to some fully connected layers for further classification. But, in model "Bi-LSTM + CRF", the output feature of stacked Bi-LSTM layers is connected to a CRF layer.

The core idea inside these two models are totally different. "Deep Bi-LSTM" still treats the prediction task as a classification problem. Its core idea is using Bi-LSTM to capture the dependency relationship between target residue and other residues, then store them in the output feature of Bi-LSTM layer. Using this feature, "Deep Bi-LSTM" will do the classification for each residue on each step. The stack of Bi-LSTM layers and additional fully connected layers are just technical method to help improving the result. Model "Bi-LSTM + CRF" is totally different. As we introduced in Chapter 3.3.2. it is a discriminative probabilistic graphical model. In model "Bi-LSTM + CRF", it consider the output feature of Bi-LSTM layer as a observation value, which is used to compute the emission score and update the transition matrix. When computing the

final secondary structure prediction sequence, CRF layer is actually being separated. Given all possible class of every residue, CRF layer is trying to find the a prediction sequence of with the maximum probability, and then assign the prediction labels to the corresponding residue. To sum up, if the length of amino acid sequence is n, and every amino acid has k possible secondary structure classes, then model "Deep Bi-LSTM" consider the secondary structure prediction problem as a k-class classification for n times, while model "Bi-LSTM + CRF" consider the prediction problem as only one classification problem with k^n possible class.

In terms of performance, model "Deep Bi-LSTM" achieved the highest accuracy among all the models, and model " Bi-LSTM + CRF" ranks the third. Basically, both of these two Bi-LSTM-based model works well on the protein secondary structure prediction task, especially model "Deep Bi-LSTM". In fact, performance of model "Deep Bi-LSTM" even slightly surpassed all the other well known models in this task, which is really exciting. And the time for a full training and test procedure of this model only cost three hours in our computer, which is only slower than those two logistic regression models. Thus, if one want to do some protein tertiary structure prediction using the result of secondary structure prediction as an extre input feature, we strongly recommend model "Deep Bi-LSTM" model, because of its high accuracy and acceptable training time.

By contrast, model "Bi-LSTM + CRF" took around two full days to finish the training stage of 300 epochs, and only obtained a medium accuracy. From this perspective, one should never use model "Bi-LSTM + CRF" since there are two alternative model has a better accuracy and a much shorter training time. However, we believe that this type of model has the best potential on protein secondary structure prediction task. There are two main reasons. Firstly, as we know, recurrent neural network is the only deep learning model to process training samples which have different length. And in many tasks, Bi-LSTM is considered to be the best basic structure among all of RNN models. Secondly, probabilistic graphical model is a good complement for Bi-LSTM model, because it considers our prediction problem as a single classification problem on the output space, while Bi-LSTM model still considers it as n separate classification problems.The combination of these two type of models can take advantage from both sides. Thus, we believe that if the probabilistic graphical model is well designed, it can help improving accuracy in the end. Our result shows that the linear chain CRF is not the future of protein secondary structure prediction task.

In the end, we want to show the comparison between our result and some other well-performed models in table 4.2:

Here, we only compared our best model "Deep Bi-LSTM" with others. The experimental data in the table comes from a paper which is published online 2 month ago[29]. Due to time constraints, we are not able to reproduce the models listed in the table. So, we add a $*$ on our model to indicate that the test dataset they used is slightly different between us. To be more specific, their dataset is obtained from the same webstie‘‘protein data bank, but it is a redundancy-reduced version compared with ours. So, although the test set is not exactly the same, we believe it can still reflect how good our model is in general. To whom might be interested in those models listed in the table, here is the corresponding papers: model "Porter 5" comes from paper "Porter 5: state-of-the-art ab initio prediction of protein secondary

Table 4.2: The performance of our model compared with others

Model	Q3	Q8
Deep Bi-LSTM*	**84.38%**	**73.49%**
Porter 5	84.19%	73.02%
SSpro 5 with templates	82.62%	71.91%
PSIPRED 4.01	82.62%	N.A.
RaptorX-Property	82.04%	70.74%
DeepCNF	81.00%	69.76%

structure in 3 and 8 classes" [29], model "SSpro 5 with templates" comes from paper "Exploiting the past and the future in protein secondary structure prediction" [30], model "PSIPRED 4.01" comes from paper "Protein secondary structure prediction based on position-specific scoring matrices1" [31], model "RaptorX-Property" comes from paper "RaptorX-Property: a web server for protein structure property prediction" [32], and model "DeepCNF" comes from paper "Protein secondary structure prediction using deep convolutional neural fields" [33]. From the numbers shown in the table. our model surpassed all the other models of the present state of the art. Considering the error caused by the slightly difference between two datasets, our model is still one of the best models, even we decrease the accuracy by 2% each. Thus, we come to the conclusion that our model "Deep Bi-LSTM" is a state-of-the-art model.

Chapter 5 Conclusion and future work

In this paper, we have tested the three most wildly used neural network architecture on the protein secondary structure prediction task. Based on the basic structures, we built 5 different models. Two of them reached the highest level of this task, and model "Deep Bi-LSTM" achieved 84.38% Q3 accuracy and 73.49% Q8 accuracy, which can be considered a state-of-the-art work. Moreover, we analyzed the pros and cons of each model, which gave people who want to work on this task further a guide on which type of structure they should pay attention to.

In the future, we will continue our research on these three directions: First of all, as shown in table 4.2, all the best models are stuck in the accuracy around 83% now. Thus, we will analyze the misclassified amino acids to find the pattern behind them, and then modify our network manually. Secondly, we will deign a new probabilistic graphical model which fits the protein sequence better, and then combine it with Bi-LSTM. In the end, we will use the secondary structure we predicted as an extra information, to build a new neural network to solve the protein tertiary structure prediction problem.

Bibliography

[1] Ken A Dill and Justin L MacCallum. The protein-folding problem, 50 years on. *science*, 338(6110):1042–1046, 2012.

[2] J Andrew McCammon, Bruce R Gelin, and Martin Karplus. Dynamics of folded proteins. *Nature*, 267(5612):585, 1977.

[3] Jianlin Cheng, Allison N Tegge, and Pierre Baldi. Machine learning methods for protein structure prediction. *IEEE reviews in biomedical engineering*, 1.

[4] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.

[5] He and Carter. Structure determined by He and Carter, Protein Data Bank entry 1UOR. `http://www.acid-base.org/xraycrystalstructure.html`, 1992. [Online; accessed 19-May-2018].

[6] Piero Fariselli, Osvaldo Olmea, Alfonso Valencia, and Rita Casadio. Prediction of contact maps with neural networks and correlated mutations. *Protein engineering*, 14(11):835–843, 2001.

[7] Kazuhito Itoh and Masaki Sasai. Flexibly varying folding mechanism of a nearly symmetrical protein: B domain of protein a. *Proceedings of the National Academy of Sciences*, 103(19):7298–7303, 2006.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[11] Christopher M Bishop. Periodic variables. *Pattern recognition and machine learning*, 1, 2006.

[12] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[13] Research Collaboratory for Structural Bioinformatics: Rutgers and UCSD/SDSC. Structure determined by He and Carter, Protein Data Bank entry 1UOR. `https://www.rcsb.org/pages/download_features#FASTA`.

[14] kabschSander. Protein secondary structure file. `https://cdn.rcsb.org/etl/kabschSander/ss.txt.gz`.

[15] Mauro Delorenzi and Terry Speed. An hmm model for coiled-coil domains and a comparison with pssm-based predictions. *Bioinformatics*, 18(4):617–625, 2002.

[16] Gary D Stormo, Thomas D Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the ''perceptron''algorithm to distinguish translational initiation sites in e. coli. *Nucleic acids research*, 10(9):2997–3011, 1982.

[17] U.S. National Library of Medicine National Center for Biotechnology Information. Basic Local Alignment Search Tool. `https://blast.ncbi.nlm.nih.gov/Blast.cgi`.

[18] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242, 1958.

[19] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[22] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[23] kijungyoon. Differentiable Neural Computer. `https://kijungyoon.github.io/DNC/`.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[26] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[27] Ross Kindermann and J Laurie Snell. *Markov random fields and their applications*, volume 1. American Mathematical Society, 1980.

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

[29] Mirko Torrisi, Manaz Kaleel, and Gianluca Pollastri. Porter 5: state-of-the-art ab initio prediction of protein secondary structure in 3 and 8 classes. *bioRxiv*, page 289033, 2018.

[30] Pierre Baldi, Søren Brunak, Paolo Frasconi, Giovanni Soda, and Gianluca Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.

[31] David T Jones. Protein secondary structure prediction based on position-specific scoring matrices1. *Journal of molecular biology*, 292(2):195–202, 1999.

[32] Sheng Wang, Wei Li, Shiwang Liu, and Jinbo Xu. Raptorx-property: a web server for protein structure property prediction. *Nucleic acids research*, 44(W1):W430–W435, 2016.

[33] Sheng Wang, Jian Peng, Jianzhu Ma, and Jinbo Xu. Protein secondary structure prediction using deep convolutional neural fields. *Scientific reports*, 6:18962, 2016.

Chapter 6 Acknowledgment

I gratefully acknowledge Prof. Rui Fan for giving me helpful advice and suggestions in this entire project. A

grateful to the computing power provided by the UChicago Beagle and RCC allocations.

A heartfelt thanks goes out to my girlfriend Ilana for all your love, support and patience when I was