Protein Secondary Structure Prediction based on Neural Network Models and Support Vector Machines

Jaewon Yang
Departments of Electrical Engineering, Stanford University
jaewony@stanford.edu

Abstract **The prediction of protein secondary structure is an important step in the prediction of protein tertiary structure. Protein tertiary structure prediction is of great interest to biologists because proteins are able to perform their functions by coiling their amino acid sequences into specific three-dimensional shapes (tertiary structure). Therefore, this subject is of high importance in medicine (e.g. drug design) and biotechnology. Instead of costly and time-consuming experimental approaches, effective methods have been developed continuously. The secondary-structure prediction approaches in use today can be categorized into three groups: neighbor-based, model-based, and metapredictor-based approaches. The model-based approaches employ sophisticated machine learning techniques such as neural networks, hidden markov models, and support vector machines to learn a predictive model trained on sequences of known structure. With the help of growing databases and the evolutionary information available from multiple-sequence alignments, resources for secondary structure prediction became abundant. However, this paper focused on single- sequence prediction in order to compare algorithmic efficiency and to save computational-time. In this paper, the neural network and the support vector machine based algorithms will be compared.**
Keywords: protein structure prediction/ secondary structure/ neural network/back-propagation/ support vector machines

I. INTRODUCTION

Protein structure prediction is one of the most important goals pursued by bioinformatics and theoretical chemistry. This subject is of great interest to biologists because proteins are able to perform their functions by coiling their amino acid sequences (primary structure) into specific three-dimensional shapes (tertiary structure) – this process is called protein folding. In other words, the linear ordering of amino acids forms secondary structure, arranging secondary structures yields tertiary structure. Therefore, protein structure prediction is of high importance in medicine (e.g. drug design) and biotechnology (e.g. the design of novel enzymes).

A number of factors exists that make protein structure prediction a very difficult task. Two main problems are that the number of possible protein structures is extremely large, and that the physical basis of protein structural stability is not fully understood. In this sense, the techniques such as spectroscopy and far-ultraviolet (far-UV, 170-250 nm) circular dichroism for structure prediction are time-consuming and expensive. However, due to the increase in computer power and especially new algorithms, much progress is being made to overcome these problems [1]. Research in computational structure prediction concerns itself mainly with predicting secondary structure from known experimentally determined primary structure. This is due to the relative ease of determining primary structure and the complexity involved in tertiary structure.

The secondary-structure prediction approaches in today can be categorized into three groups: neighbor-based, model-based, and metapredictor-based [2]. The neighbor-based approaches predict the secondary structure by identifying a set of similar sequence fragments with known secondary structure; the model-based approaches employ sophisticated machine learning techniques to learn a predictive model trained on sequences of known structure, whereas the metapredictor -based approaches predict based on a combination of the results of various neighbor and/or model-based techniques.

Historically, the most successful model-based approaches, such as PSIPRED [4] were based on neural network (NN) learning techniques [5]. However, in recent years, secondary structure prediction algorithms based on support vector machines have been developed and have been showing good performance [7]. In this paper, these two successful methods will be compared.

II. METHOD

A. Database

a. Definition

No unique method of assigning residues to a particular secondary-structure exists, although the most widely accepted protocol is based on the DSSP algorithm. DSSP uses the following structural classes: H (α-helix), G (3_{10}-helix), I (π-helix), E (β-strand), B (isolated-β bridge), T (turn), S (bend), and – (other). In this paper, the reduction scheme that converts this eight-state assignment to three states by assigning H the helix state (H), E to the strand state (E), and the rest (I,T,S and −) to a coil state (C). This is the simplest format used in structure databases.

b. Training and testing sets[3]

Cross-validation appears to remove the problem of a limited data set for training and test. However, artificially high accuracies can be obtained if the set of proteins used in cross-validation show sequence similarity to each other. Accordingly, cross-validation sets must be pruned stringently to remove internal sequence similarities, but if it is not possible, then a completely independent test set must be used. Therefore, in this paper, the hold-out cross validation technique, where test proteins are removed from the training set, was used. The training data set is the CB396 and the RS126 set is used for testing. The 11 pairs which showed up homologies in the CB396 set were removed from the RS126 set, and protein chains of <30 residues were also removed. In conclusion, 382 proteins (61455 residues) from CB396 set for training and 115 protein chains (21755 residues) from RS126 set for testing were used.

Secondary structure fractions		
Total number of residues for training = 61455		
Total number of residues for testing = 21755		
	Train set(%)	**Test set(%)**
H(α-helix)	31.87	28.46
E(β-sheet)	21.56	21.15
–(others)	46.56	50.37

Table 1. Proteins in training and testing set

c. Performance measures

There are many ways to access the performance of the method for predicting secondary structures. The most commonly used measure is a simple success rate, Q_3, which is the percentage of correctly predicted residues on three types of secondary structures:

$$Q_3 = \sum_{(i=H,E,C)} \frac{\text{correctly predicted}_i}{\text{observed}_i} \times 100(\%)$$

B. *Neural Network*

The neural network is trained using the supervised learning method. Here the training process is finding the appreciate value for the weight of each layer in the network to maximize accuracy of prediction. In supervised learning, a training data set is encoded into feature vectors combined with correct class labels, such as helix, sheet, or coil. The PSIPRED method by Jones (1999) is a successful approach for predicting secondary structure [4]. In PSIPRED, a two-stage neural network was used based on the position-specific scoring matrices generated by PSI-BLAST. This paper is based on the algorithm of PSIPRED, but instead of applying PSSM (Position-specific Scoring Matrices) into input, single sequence prediction method is used in order to focus on the algorithm and to avoid expensive computational time.

a. Neural network and properties

A feedforward network is composed of two or more layers of processing units. The first is the input layer, the last is the output layer, and all the other layers between are termed hidden layers. The state of each unit has a real value in the range between 0 and 1. The all input units (I) that form an input vector are determined by an input window of amino acid residues through an input coding scheme. Starting from the hidden layer (h) and moving toward the output layer (O), the state of each hidden unit i in the network is determined by:

$$O_i = \sum_j v_{ij} \cdot \frac{1}{1+\exp(-h_j)} \tag{1}$$

$$h_i = \sum_j w_{ij} \cdot I_j \tag{2}$$

The back-propagation learning algorithm can be used in networks with hidden layers to find a set of weights that performs correct mapping between sequences and structures. Starting with an initial set of randomly assigned numbers, the weights are altered by gradient descent to minimize the error between the desired and the actual output vectors.

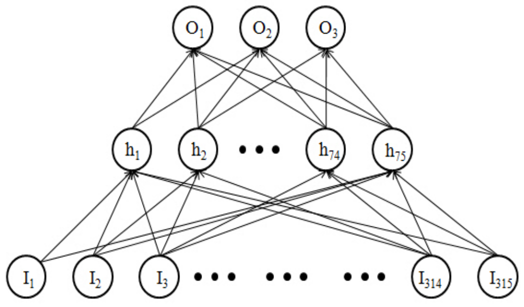


Figure 1. The 1st neural network design

b. Network design

The neural network was trained with different sequence and structural information using a sliding window scheme which was also used in the SVM^{multiclass} and SVM^{hmm}. In the sliding window method, a window becomes one training pattern for the predicting structure of the residue at the center of the window. In many input-encoding methods, orthogonal encoding was used. Each residue has a unique binary vector, such as (1,0,0,…),(0,1,0,…),(0,0,1,…), … ,(0,…,0,1) in orthogonal encoding. Each binary vector is 21-dimensional that means 21 amino acids. In this method, the weights of all residues in a window are assigned to 1.

For a given input and set of weights, the output of the network will be a set of numbers between 0 and 1. The secondary structure chosen was the output unit that had the highest activity level; this was equivalent to choosing the output unit that had the least mean square error with the target output.

The performance of the network on the training and testing sets depends on many variables, including, the number of training examples, the number of hidden units, and the size of window. The ability of a network to extract higher order features from the training set depends on the layer of hidden units and the types of encoding scheme.

C. *SVM^{hmm} and SVM^{multiclass}*

Among the many machine learning approaches, support vector machine (SVM) methods are the most recent to be used for structure prediction. The paper [7] designed classifiers for the three cluster problems based on the binary classifiers generated by SVMs. However, in this paper, the generalized multi-class SVMs were used [10]. Unlike the case of multiclass classification where output space with interchangeable, arbitrarily numbered labels, structured output spaces are considered in generalized multiclass SVMS. I used SVM^{light} which is widely used software implementations of SVM. SVM^{hmm} and SVM^{multiclass} are applications using SVM^{light} for generalized multiclass classification.

$$\text{SVM: } \min_{w,\xi} \ \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \ , \text{s.t.} \forall i, \ \ \xi_i \geq 0 \quad (3)$$

$$\text{s.t.} \forall i, \forall y \in \mathcal{Y}: w^T \delta\Psi_i(y) \geq \ \Delta(y_i, y) - \ \xi_i \quad (4)$$

$$\text{where } \delta\Psi_i(y) \ \equiv \Psi(x_i, y_i) - \ \Psi(x_i, y) \quad (5)$$

SVM^{multiclass} is an implementation of the multi-class SVM described in [8]. For a training set $(x_1, y_1) \cdots (x_n, y_n)$ with labels y_i in i = 1, 2, 3 in the secondary structure prediction. For linear kernels, this is very fast and runtime scales linearly with the number of training examples. Non-linear kernels are not (really) supported. The loss function $\Delta(y_i, y)$ is the number of misclassified tags.

SVM^{multiclass}:

$$\min \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_i \xi_i \quad (6)$$

$$\text{s.t. for all y : } \ x_1 \cdot w_{yi} \geq x_1 \cdot w_y + \ 100\Delta(y_1, y) - \xi_1 \quad (7)$$

$$\vdots$$

$$x_1 \cdot w_{yi} \geq x_1 \cdot w_y + \ 100\Delta(y_1, y) - \xi_1 \quad (8)$$

SVM^{hmm} is an implementation of structural SVMs for sequence tagging [9]. Given an observed input sequence $x = (x_1, x_2, \cdots, x_{21})$ of feature vectors, the model predicts a tag sequence $y = (\ y_1, \ y_2, \cdots, y_{21})$ according to the following optimization problem. For the given training examples $(x^1, y^1), (x^2, y^2), \cdots, (x^n, y^n)$, the feature vector is $x^j = (x_1^j, \cdots, x_{21}^j)$ with their correct tag sequence, $y^j = (y_1^j, \cdots, y_{21}^j)$. For the secondary structure prediction, only one entry has 1 and others are zero by orthogonal encoding.

SVM^{hmm}:

$$\min \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_i \xi_i \quad (9)$$

$$\text{s.t. for all y:}$$

$$\sum_{i=1,2,\ldots,21}\left(x_i^1 \cdot w_{y_i^1}\right) + \ \varphi_{trans}(y_{i-1}^1, y_i^1) \cdot w_{trans}$$

$$\geq \ \sum_{i=1,2,\ldots,21}\left(x_i^1 \cdot w_{y_i} \ \right) + \ \varphi_{trans}(y_{i-1}, y_i \ ) \cdot w_{trans} + \ \Delta(y^1, y) \quad (10)$$

$$\vdots$$

$$\sum_{i=1,2,\ldots,21}\left(x_i^n \cdot w_{y_i^1}\right) + \ \varphi_{trans}(y_{i-1}^n, y_i^n) \cdot w_{trans}$$

$$\geq \ \sum_{i=1,2,\ldots,21}\left(x_i^n \cdot w_{y_i} \ \right) + \ \varphi_{trans}(y_{i-1}, y_i \ ) \cdot w_{trans} + \ \Delta(y^1, y) \quad (11)$$

a. Parameter optimization

In generalized multi-class SVMs, we need to select a kernel function and the regularization parameter C. The primal formulation of the generalized soft-margin SVMs maximize margin and minimize training error simultaneously by solving the previous optimization problem (3)-(5).

b. Dynamic programming for cutting-plane

The key challenge in solving the quadratic problems for the generalized multi-class SVM learning is the large number of margin constraints. If the length of each amino acid is L and there are n training data, we have exponential number of constraints, $n3^L$ because each amino acid can be one of three classes.

However, only a much smaller subset of constraints needs to be explicitly examined by using cutting-plane. The cutting-plane using dynamic programming (Viterbi algorithm) aims at finding a small set of active constraints that ensures a sufficiently accurate solution. This method can reduce the number of constraints to a polynomial-sized subset of constraints that the corresponding solution fulfills all constraints with a precision of ε. In other words, the remaining exponentially many constraints are guaranteed to be violated by no more than ε, without the need for explicitly adding them to the optimization problem.

III. RESULTS

A. Neural Network

a. Training and testing with increasing the number of iteration.

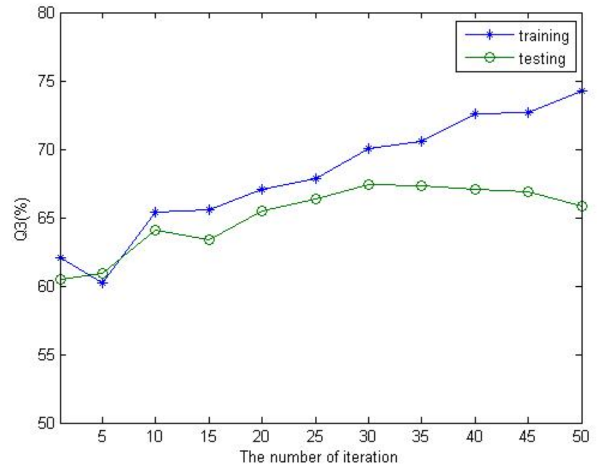


Figure 2. Learning curve for real proteins with 75 hidden units and 15 window size.

Further training improved the performance of the networks with hidden units on the training set, but performance on the testing set did not improve but tended to decrease. This result is an indication that memorization of the detail of the training set is interfering with the ability of the network to generalize. The peak performance for a network with 15 window size and 75 hidden units was Q_3 = 67.42% after 30 iterations.

b. Dependence on the number of hidden units

Hidden Units	Q_3(%)
0	61.39
25	63.68
50	65.64
75	67.42
100	66.35
125	66.91

Table 2. Dependence of testing success on hidden units

Table 2 shows that the peak performance on the testing set depends on the number of hidden units. Also, it is shown that having more hidden units is not always good because it can cause high-variance problem as mentioned in the previous section.

c. Dependence on the size of window

Window Size	Q_3(%)
1	50.37
3	54.04
5	59.51
7	63.17
9	65.40
11	65.09
13	66.61
15	67.04
17	66.67
19	66.64
21	66.72

Table 3. Dependence of testing success on window size

Table 3 shows the dependence of testing accuracy rate on the size of the input window with 75 hidden units. The result shown in Table 3 indicates that when the size of the window was small the performance on the testing set was reduced, probably because information outside the window is not available for the prediction. When the size of the window was increased, the performance reached a maximum at around 15 window size. For larger window sizes, the performance deteriorated, probably because of the effects of extra weights that could not contain any information about the secondary structure of the center. Thus, irrelevant weights can interfere with the performance of the network.

d. 2nd network

	1st network	2nd network
Q_3(%)	67.42	67.45

Table 4. Testing accuracy of the 1st and 2nd network

In PSIPRED, a second network is used to filter successive outputs from the main network. As only three possible inputs are necessary for each amino acid position, the network has an input network comprising just 63 input units. For this network, a smaller hidden layer of 75 units were used instead of 60 units of the PSIPRED. Q3 performance was similar but showed a little improvement.

B. SVM^{hmm} and SVM^{multiclass}

Non-linear kernels are not supported in the SVM^{hmm} and SVM^{multiclass}. Without the optimization of kernel parameters, I focused on selecting the optimal parameter C which plays a critical role. Common practice is to choose the parameter that maximizes the accuracy by using hold-out cross validation method.

C	Q_3(%) (W=1)	Q_3(%) (W=17)	# SV(W=17)
2	50.37	61.83	4
4	47.24	61.83	4
6	47.24	61.83	4
8	47.24	**63.28**	4
10	47.24	63.13	3
12	47.24	63.13	3
14	47.24	59.63	4
16	47.24	59.63	4
18	42.95	59.63	4
20	42.95	59.78	4

Table 5. Dependence of the number of support vector and testing success on C(Regularized Parameter) (**SVM^{multiclass}**)

The optimal window size of the encoding scheme for SVM^{multiclass} and SVM^{hmm} was obtained by testing the accuracy for the various window sizes, and it was shown that 17 for SVM^{multiclass} and 19 for SVM^{hmm} are optimal window size. The interpretation of the result is similar with that of the neural networks. Also this shows that the local sequence environment of a residue substantially determines its secondary structure. It was considered the fact that residues far apart in sequence but close in three dimensions can have the tertiary interactions.

C	Q_3(%) (W=1)	Q_3(%) (W=19)	# SV (W=19)
2	59.26	64.81	20
4	59.88	64.91	35
6	60.31	64.94	44
8	59.87	64.97	38
10	60.21	65.12	46
12	59.87	65.07	49
14	59.76	65.09	53
16	60.18	64.95	57
18	60.07	65.03	46
20	59.84	**65.22**	59
22	60.08	65.07	62
24	60.12	64.93	72

Table 6. Dependence of the number of support vector and testing success on C(Regularized Parameter) (**SVM^{hmm}**)

Interestingly, the performance of SVM^{hmm} is better than that of SVM^{multiclass}. Also, we can see that SVM^{hmm} has much more support vectors than SVM^{multiclass} does. Based on the algorithm of HMM(Hidden-Markov Model), SVM^{hmm} considers the transition probabilities between hidden-state which means secondary structure in this problem, but this information is not involved in SVM^{multiclass}. Intuitively, the secondary structure of an amino acid are correlated with that of the neighborhood sequences, not far from it. This concept is very similar to the reason why we take window to input feature vector.

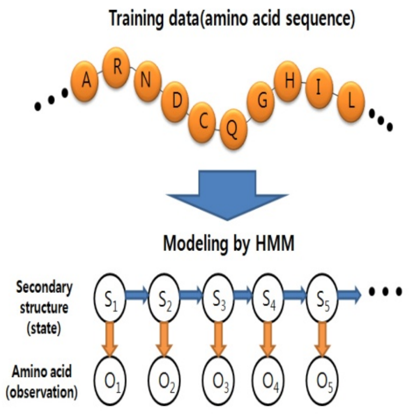
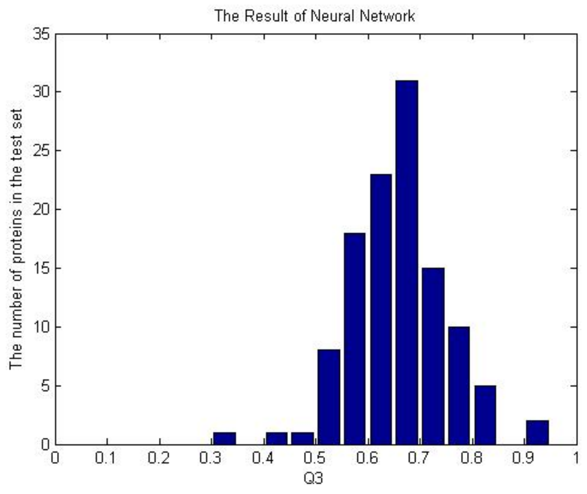


Figure 3. Hidden-Markov Model(HMM) for protein secondary structure prediction

C. Compare

Training with SVMs has crucial advantages including much faster convergence than neural networks (NNs). The SVMs tend not to overfit and the ability to find the global optimum by solving the dual problem of quadratic convex function minimization. However, the neural networks approach suffers from the local minima, determination of appropriate structure of neural networks and too many parameters.



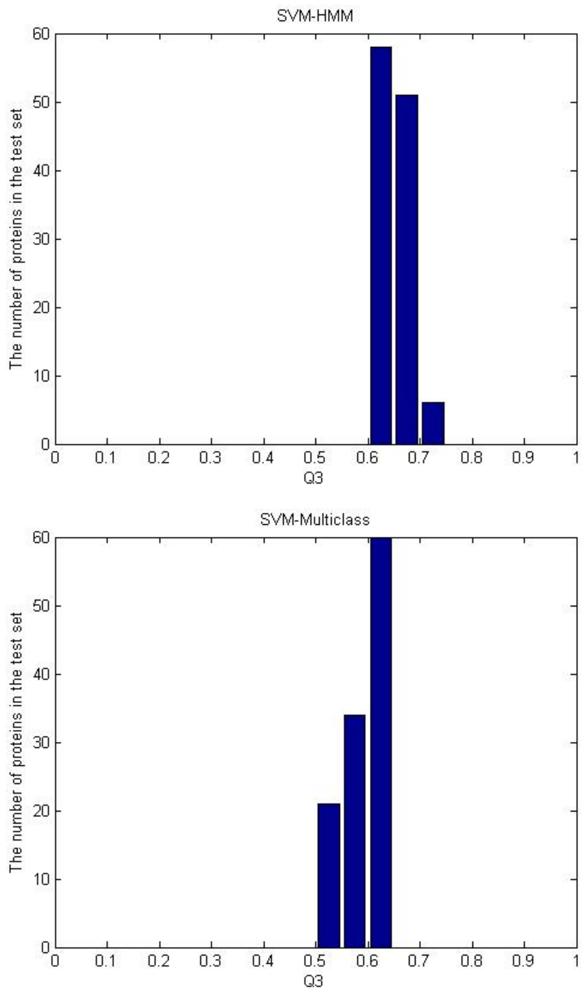


Figure 4. Bar graphs showing the distribution of Q_3 scores

As we can see, the performance of the neural network outperforms the SVMs. In the neural network, I used non-linear network by using sigmoid function which attributed to increase the accuracy because this problem is not linear. However, the SVMs are still linear because kernels cannot be applied. As a result of it, the performance of SVMs was worse than that of NNs.

Methods	Q_3(%)
Non-linear NNs with 75 hidden units	67.42
SVM^{hmm}	65.22
SVM^{multiclass}	63.28
Linear NNs with no hidden units	62.50

Table 7. The best performance of linear NNs and SVMs

In a different point of view, the overall accuracy of SVMs is less than the nonlinear neural network, but the variance of accuracy is smaller than that of the neural network. This means that the average accuracy of SVMs is more reliable.

However, the prediction levels of all three methods I used in this paper are not sufficient to compare with the result using multiple sequence alignments. I will discuss about the ways to improve the accuracy of both approaches in the next section.

IV. Discussion and Future Work

The highest accuracy is 67.42% of the neural network, but it is not high enough. I think that there are two weak points for this low accuracy: multiple sequence alignments and kernel methods. For example, one recent study adopted frequency profiles with evolutionary information as an encoding scheme for SVM [6], and another approach is to use incorporated PSI-BLAST PSSM profiles as an input vector [4][7]. Both of them showed the improvement of accuracy. And based on the result of these studies, the success of SVM methods depends on the proper choice of encoding profiles and kernel function.

A. *Encoding Profile*

In frequency matrix encoding, the frequency of occurrence of 21 amino acid residues at each position in the multiple sequence alignment is calculated for each residue. And in PSSM coding the individual profiles were used to reflect detailed conservation of amino acids in a family of homologous proteins. The previous study reported that by using reliable local alignment, the prediction accuracy could be improved. Therefore, if multiple sequence alignment methods are applied, the accuracy could be higher.

B. *Kernel Methods*

The choice of kernel function is critical to the success of SVM. By applying the kernel method, a nonlinear classifier can be built. Most studies adopted the radial basis function (RBF) kernel [7]. Therefore, applying kernels methods could increase the performance. However, we have still a remaining problem to handle huge training data sets, which prevents from applying kernels to SVMs.

V. Acknowledgement

Specially thanks to Chung(Tom) Do, Ph.D candidate in the Department of the Computer Science, in Stanford. Without his advice and guideline, I was not able to make a progress on the final project. I really appreciate his efforts to help me out.

References

[1] Hae-Jin Hu, Robert W. Harrison, Phang C, Tai, and Yi Pan, "Current Methods For Protein Secondary-Structure Prediction Based on Support Vector Machines", Ch.1, Knowledge Discovery in Bioinformatics: Techniques, Methods, and Applications, 2007.

[2] Hae-Jin Hu, Robert W. Harrison, Phang C, Tai, and Yi Pan, "Protein Structure Prediction using String Kernels", Ch.8, Knowledge Discovery in Bioinformatics: Techniques, Methods, and Applications, 2007.

[3] James A. Cuff and Geoffrey J. Barton, "Evaluation and Improvement of Multiple Sequence Methods for Protein Secondary Structure Prediction", Proteins, 1998.

[4] David T. Jones, "Protein Secondary Structure Prediction Based on Position-specific Scoring Matrices", JMB, 1999.

[5] Ning Qian and Terrence J. Sejnowski, "Predicting the Secondary Structure of Globular Proteins Using Neural Network Models", JMB, 1988.

[6] S. Hua and Z. Sun, "A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach", JMB, 2001.

[7] H. Kim and H. Park, "Protein secondary structure prediction. based on an improved support vector machines approach", Protein Eng. 2003.

[8] K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs, JMLR, 2001.

[9] Y. Altun, I. Tsochantaridis, T. Hofmann, "Hidden Markov Support Vector Machines", International Conference on Machine Learning (ICML), 2003.

[10] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support Vector Learning for Interdependent and Structured Output Spaces", ICML, 2004.

[11] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Large Margin Methods for Structured and Independent Output Variables", JMLR, 2005

[12] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", IEEE (1989)

Appendix

A. *Proteins in training and testing set*

Amino acid fractions		
Total number of residues = 61455		
Amino Acid	**Train set**	**Test set**
A(Alanine)	0.0875	0.2473
R(Arginine)	0.0465	0.1315
N(Asparagine)	0.0471	0.1330
D(Aspartic acid)	0.0597	0.1686
C(Cysteine)	0.0148	0.0418
E(Glutamic acid)	0.0607	0.1715
Q(Glutamine)	0.0371	0.1048
G(Glycine)	0.0781	0.2207
H(Histidine)	0.0218	0.0617
I(Isoleucine)	0.0553	0.1564
L(Leucine)	0.0861	0.2432
K(Lysine)	0.0579	0.1637
M(Methionine)	0.0200	0.0565
F(Phenylalanine)	0.0390	0.1102
P(Proline)	0.0461	0.1303
S(Serine)	0.0606	0.1712
T(Threonine)	0.0591	0.1671
W(Tryptophan)	0.0149	0.0423
Y(Tyrosine)	0.0372	0.1052
V(Valine)	0.0687	0.1941
Others	0.0009	0.0027

B. *The result of Neural Network for each test data*

Protein	#residues	Q_{alpha}	Q_{beta}	Q_{coil}	Q_3(%)
1acx.all	107	0	0.3829	0.85	0.6448
1azu.all	125	0.3571	0.5	0.7594	0.648
1bbpa.all	172	0.7058	0.5060	0.8611	0.6744
1bds.all	42	0	0.1818	0.9032	0.7142
1bmv1.all	184	0.6250	0.3733	0.7128	0.5706
1bmv2.all	373	0.6842	0.3257	0.7252	0.5817
1cbh.all	35	0	0.25	0.9130	0.6857
1cc5.all	82	0.6929	0	0.8372	0.7682
1cdta.all	59	0	0.2592	0.7812	0.5423
1crn.all	45	0.2105	0.5	0.9090	0.5777
1csei.all	62	0.6363	0.5789	0.75	0.6774
1eca.all	135	0.5360	0	0.7105	0.5851

1etu.all	176	0.6153	0.4444	0.7741	0.6363
1fc2c.all	42	0.6190	0	0.6666	0.6428
1fdlh.all	217	0	0.3425	0.8256	0.5852
1fdx.all	53	0	0	0.7954	0.6603
1fkf.all	106	1	0.5121	0.8793	0.7452
1fnd.all	295	0.8235	0.3950	0.7123	0.6508
1fxia.all	95	0.1250	0.6666	0.8166	0.7157
1gd1o.all	333	0.7397	0.5263	0.6787	0.6486
1gdj.all	152	0.6111	0	0.5681	0.5986
1gp1a.all	182	0.5714	0.5517	0.7927	0.7032
1hip.all	84	0.9	0.2222	0.7230	0.6904
1il8a.all	70	0.9375	0.6666	0.75	0.7714
1l58.all	163	0.7196	0.5	0.7619	0.7116
1lap.all	480	0.6741	0.4313	0.78	0.6666
1lmb3.all	86	0.8813	0	0.4814	0.7558
1mrt.all	30	0	0	0.7666	0.7666
1ovoa.all	55	0.4	0.0833	0.8787	0.6181
1paz.all	119	0.8823	0.5454	0.6206	0.6302
1ppt.all	35	1	0	0.8823	0.9428
1pyp.all	279	0.6944	0.4642	0.6930	0.6702
1r092.all	254	0.2222	0.4606	0.6538	0.5708
1rbp.all	173	1	0.2317	0.7051	0.5028
1rhd.all	292	0.7037	0.4062	0.6648	0.6472
1s01.all	274	0.4756	0.5106	0.8	0.6532
1sh1.all	47	0	0	0.9090	0.4255
1tnfa.all	151	0	0.2461	0.6976	0.5033
1ubq.all	75	0.5	0.625	0.7692	0.68
1wsya.all	247	0.8373	0.2727	0.8021	0.7489
1wsyb.all	384	0.7482	0.4461	0.7558	0.7005
256ba.all	105	0.7160	0	0.875	0.7523
2aat.all	395	0.7851	0.4	0.6622	0.6810
2ak3a.all	225	0.6534	0.375	0.8152	0.68
2alp.all	197	0.875	0.3269	0.7176	0.5177
2cab.all	255	1	0.1898	0.7924	0.6196
2ccya.all	126	0.7444	0	0.9444	0.8015
2cyp.all	292	0.5671	0.125	0.8239	0.6678
2fox.all	137	0.78	0.7586	0.7931	0.7810
2fxb.all	80	0.4615	0.5	0.6981	0.625
2gbp.all	308	0.6693	0.5789	0.7795	0.6980
2gcr.all	172	0	0.4492	0.6699	0.5813
2glsa.all	467	0.7152	0.3375	0.7983	0.6937
2gn5.all	86	0	0.5	0.6829	0.6744
2hmza.all	113	0.6285	0	0.6279	0.6283
2i1b.all	152	0	0.3478	0.7469	0.5657
2ltna.all	180	0	0.4805	0.6893	0.6
2ltnb.all	46	0.25	0.2333	0.5833	0.3260
2mev4.all	57	0	0	0.6226	0.5789
2or1l.all	62	0.775	0	0.9545	0.8387
2paba.all	113	0.625	0.3389	0.8260	0.5575
2phh.all	390	0.7794	0.2636	0.7638	0.6282
2rspa.all	114	0.5	0.4042	0.8524	0.6491
2sns.all	140	0.9615	0.6071	0.6279	0.6857
2sodb.all	150	0	0.5555	0.7916	0.7066
2stv.all	183	0.7222	0.4268	0.5903	0.5300
2tgpi.all	57	0.625	0.4285	0.9142	0.7543
2tmvp.all	153	0.6406	0.2857	0.5243	0.5620
2tsca.all	263	0.6216	0.2615	0.8145	0.6235
2utga.all	69	0.6666	0	0.8333	0.7101
2wrpr.all	103	0.6463	0	0.8095	0.6796
3ait.all	73	0	0.3428	0.6578	0.5068
3b5c.all	84	0.6666	0.5789	0.6363	0.6309
3blm.all	256	0.7191	0.5217	0.7933	0.7187
3cd4.all	177	0	0.5243	0.7368	0.6384
3cla.all	212	0.8166	0.2131	0.7472	0.6132
3cln.all	142	0.8977	0.25	0.8478	0.8450
3gapa.all	207	0.8437	0.3333	0.5737	0.6328
3hmga.all	327	0.4	0.3416	0.7582	0.5779
3hmgb.all	174	0.6629	0.0869	0.3870	0.4885
3icb.all	74	0.9767	0	0.9032	0.9459
3pgm.all	229	0.8695	0.6	0.6896	0.7379
3rnt.all	103	0.3529	0.4642	0.8620	0.6699
3tima.all	248	0.7113	0.6578	0.7345	0.7137
4bp2.all	116	0.4285	0.25	0.8135	0.6120
4cpai.all	36	0	0	0.8333	0.6944
4gr1.all	460	0.6590	0.4414	0.7327	0.6413
4pfk.all	318	0.7686	0.6949	0.76	0.7515
4rhv1.all	272	0.6666	0.4337	0.7683	0.6617
4rhv3.all	235	0.6666	0.4637	0.7814	0.6808
4rhv4.all	39	0	0	0.7428	0.6666
4rxn.all	53	0	0.375	0.8	0.7358
4sdha.all	144	0.5940	0	0.5348	0.5763
4sgbi.all	50	0	0.4166	0.8947	0.78
4ts1a.all	316	0.8	0.2333	0.6946	0.7025
4xiaa.all	392	0.7341	0.4285	0.7771	0.7270
5cytr.all	102	0.4761	0	0.7333	0.6274
5er2e.all	329	0.4615	0.3154	0.7532	0.5319
5ldh.all	332	0.5967	0.5483	0.6045	0.5963
5lyz.all	128	0.5789	0.25	0.7560	0.6718
6acn.all	753	0.6026	0.5075	0.7909	0.6852
6cpa.all	306	0.5840	0.48	0.7972	0.6666
6cpp.all	404	0.6444	0.2195	0.8142	0.6782
6cts.all	428	0.6428	0	0.7764	0.6869
6dfr.all	153	0.5806	0.2888	0.7532	0.5816
6hir.all	48	0	1	0.7857	0.8125
6tmne.all	315	0.6694	0.4038	0.7586	0.6666
7cata.all	497	0.5839	0.4084	0.7785	0.6720
7icd.all	413	0.7070	0.4133	0.6906	0.6464
7rsa.all	123	0.9090	0.2439	0.8	0.6341
8adh.all	373	0.4523	0.4823	0.6911	0.5898
9apia.all	338	0.7961	0.1517	0.6910	0.5443
9apib.all	35	0	0.5294		0.7714
9pap.all	211	0.4693	0.5277	0.8174	0.6872
9wgaa.all	170	0.1875	0.375	0.9420	0.8176