

國立交通大學

電機與控制工程研究所

碩士論文

模糊 K -最近相鄰點分類法於蛋白質可溶性預測

**Fuzzy K -Nearest Neighbor Classifier to Predict Protein
Solvent Accessibility**

學 生：施 逸 祥

指 導 教 授：張 志 永

中 華 民 國 九 十 六 年 七 月

模糊 K -最近相鄰點分類法於蛋白質可溶性預測

Fuzzy K -Nearest Neighbor Classifier to Predict Protein

Solvent Accessibility

學 生：施逸祥

Student : Yi-Xiang Shi

指導教授：張志永

Advisor : Jyh-Yeong Chang

國立交通大學

電機與控制工程學系



Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master in

Electrical and Control Engineering

July 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年七月

模糊 K -最近相鄰點分類法於蛋白質可溶性預測

學生:施逸祥

指導教授:張志永博士

國立交通大學電機與控制工程研究所

摘要

蛋白質在生物體中一直扮演著很重要的角色,蛋白質被發現的數量及其結構逐年增加。隨著蛋白質的應用越來越廣泛,待解決的課題也就越來越多。例如:蛋白質二級結構預測問題、蛋白質相對溶劑可接觸性預測問題等。



本篇論文,我們利用修改的模糊 K -最近相鄰點法,混合從 PSI-BLAST 產生的位置加權矩陣,針對蛋白質相對溶劑可接觸性預測問題進行研究。最近 Sim 等人 [31],應用模糊 K -最近相鄰點法於蛋白質可溶性預測有顯著的效果。我們提出改進之模糊 K -最近相鄰點法,應用在三態相對溶劑可接觸性預測和二態相對溶劑可接觸性預測,所得到的實驗結果與近幾年的其它方法比較,有較佳的預測正確率。我們並與歐等人 [52] 所發表的快速輻射半徑基底函數網路演算法做結合。最後,將這兩種方法之結果做資訊融合以有效地提高預測的準確度。六種修正方法包括:(1) 模糊 K -最近相鄰點法、(2) 改進的模糊 K -最近相鄰點法、(3)

快速輻射半徑基底函數網路演算法、(4) 第一種線性相加合併法、(5) 第二種線性相加合併法、以及(6) 信心指數合併法。在大部分條件表現最佳的情況下，我們建議選擇第二種線性相加合併法。



Fuzzy K -Nearest Neighbor Classifier to Predict Protein Solvent Accessibility

STUDENT: YI-XIANG SHI

ADVISOR: JYH-YEONG CHANG

Institute of Electrical and Control Engineering

National Chiao-Tung University



Abstract

Proteins have been played an important role in a creature and the numbers of proteins and their structures have been increased with years. Since protein applications are more widely used, there will be a lot of problems to be solved.

Using a position-specific scoring matrix (PSSM) generated from PSI-BLAST in this thesis, we develop the modified fuzzy k -nearest neighbor method to predict the protein relative solvent accessibility. By modifying the membership functions of the fuzzy k -nearest neighbor method by Sim *et al.* [31], has recently been applied to protein solvent accessibility prediction with excellent results. Our modified fuzzy k -nearest neighbor method is applied on three-state, E, I, and B, and two-state, E, and B, relative solvent accessibility predictions, and its prediction accuracy compares favorably with those by the fuzzy k -NN and QuickRBF approaches. At last, we combine

the prediction results of modified fuzzy k -nearest neighbor method and QuickRBF approach to improve the performance. Six modification approaches include: (1) Fuzzy K -Nearest Neighbor Method, (2) Modified Fuzzy K -Nearest Neighbor Method, (3) QuickRBF, (4) Linear Combination Fusion 1, (5) Linear Combination Fusion 2, and (6) Reliability Index Fusion. We recommend the Linear Combination Fusion 2 approach which has shown the best performance in most cases.



Acknowledgement

I would like to express my sincere appreciation to my advisor, Dr. Jyh-Yeong Chang. Without his patient guidance and inspiration during the two years, it is impossible for me to overcome the obstacles and complete the thesis. In addition, I am thankful to all my lab members for their discussion and suggestion.

Finally, I would like to express my deepest gratitude to my family. Without their strong support and encouragement, I could not go through the two years.



Contents

Abstract (Chinese)	3
Abstract (English)	5
Acknowledgement	7
List of Figures	10
List of Tables	11
Chapter 1. Introduction	12
1.1 Motivation and The Background of This Research.....	12
1.2 Thesis Outline.....	15
Chapter 2. The Data Set Used and Previous Algorithms	16
2.1 Training and Data Set.....	16
2.2 The Definition of Solvent Accessibility.....	18
2.2.1 Static Residue Solvent Accessibility.....	18
2.2.2 Residue Relative Solvent Accessibility.....	20
2.3 PSI-BLAST Profiles.....	24
2.4 <i>K</i> -Nearest Neighbor Algorithm.....	26
2.5 Quick Radial Basis Function Network.....	29
Chapter 3. Protein Relative Solvent Accessibility Prediction	35
3.1 Fuzzy <i>K</i> -Nearest Neighbor Approach.....	35
3.2 Modified Fuzzy <i>K</i> -Nearest Neighbor Approach.....	37
3.3 QuickRBF Approach.....	43
3.4 Fusion Method.....	45

3.4.1 Linear Combination Fusion 1.....	45
3.4.2 Linear Combination Fusion 2.....	47
3.4.3 Reliability Index.....	48
Chapter 4. Experiment and Simulation Results.....	50
4.1 Datasets.....	50
4.2 Results.....	51
4.2.1 Results of Fuzzy <i>K</i> -NN and Modified Fuzzy <i>K</i> -NN Classifiers	51
4.2.2 Results of QuickRBF approach and the Fusion Methods.....	53
4.3 Matthew's Correlation Coefficients of Modified Fuzzy <i>K</i> -NN Approach.....	57
4.4 Comparison with other Approaches.....	62
Chapter 5. Conclusion and Discussion.....	65
References.....	66



List of Figures

Fig. 2.1.	Measure Accessibility.....	19
Fig. 2.2.	Binary Model.....	23
Fig. 2.3.	Raw Profile From PSI-Blast Log File.....	25
Fig. 2.4.	BLOSUM 62 Matrix.....	25
Fig. 2.5.	Simple 2-D case, each instance is described only by two values	27
Fig. 2.6.	General Architecture of Radial Basis Function Network.....	31
Fig. 3.1.	The two-state membership functions.....	39
Fig. 3.2.	The flowchart to calculate 2-state membership values.....	40
Fig. 3.3.	The three-state membership functions.....	41
Fig. 3.4.	The flowchart to calculate 3-state membership values.....	42
Fig. 3.5.	Architecture of QuickRBF method.....	44
Fig. 3.6.	The normalization procedure in Linear Combination Fusion 1.....	46
Fig. 3.7.	The normalization procedure in Linear Combination Fusion 2.....	48

List of Tables

Table 2.1.	Database of Non-Homologous Proteins Used For Seven-Fold Cross Validation.....	17
Table 2.2.	Definition of Solvent Accessibility States.....	21
Table 3.1.	Fusion method : Linear combination.....	47
Table 4.1.	RSA classification accuracy of two kinds Fuzzy K -Nearest Neighbor methods on the RS126 data set with PSI-BLAST pssm profiles.....	52
Table 4.2.	RSA classification accuracy of QuickRBF and the fusion methods.....	54
Table 4.3	Comparison of performance of the six approaches in RSA prediction on the RS126 data set with PSSMs generated by PSI-BLAST.....	56
Table 4.4.	The accuracy tables A of Modified Fuzzy K -NN on each fold and RS126	58
Table 4.5.	Matthew's Correlation Coefficients of the five approaches on RS126.....	60
Table 4.6.	Comparison of performance of Modified Fuzzy K -NN approach with other methods.....	64



Chapter 1. Introduction

1.1 Motivation and the Background of this Research

The solvent accessibility of amino acid residues plays an important role in tertiary structure prediction, especially in the absence of significant sequence similarity of a query protein to those with known structures. The prediction of solvent accessibility is less accurate than secondary structure prediction in spite of improvements in recent researches.

Predicting the three-dimensional (3D) structure of a protein from its sequence is an important issue because the gap between the enormous number of protein sequences and the number of experimentally determined structures has increased [1], [2]. However, the prediction of the complete 3D structure of a protein is still a big challenge, especially in the case where there is no significant sequence similarity of a query protein to those with known structures [3]–[6]. The prediction of solvent accessibility and secondary structure has been studied as an intermediate step for predicting the tertiary structure of proteins, and the development of knowledge-based approaches has helped to solve these problems [7]–[11].

Secondary structures and solvent accessibilities of amino acid residues give a useful insight into the structure and function of a protein [11]–[14]. In particular, the knowledge of solvent accessibility has assisted alignments in regions of remote sequence identity for threading [1], [15]. However, in contrast to the secondary structure, there is no widely accepted criterion for classifying the experimentally

determined solvent accessibility into a finite number of discrete states such as *buried*, *intermediate* and *exposed* states. Also, the prediction accuracies of solvent accessibilities are lower than those for secondary structure prediction, since the solvent accessibility is less conserved than secondary structure [1], although there has been some progress recently.

The prediction of solvent accessibility, as well as that of the secondary structure, is a typical pattern classification problem. The first step for solving such a problem is the feature extraction, where the important features of the data are extracted and expressed as a set of numbers, called feature vectors. The performance of the pattern classifier depends crucially on the judicious choice of the feature vectors. In the case of the solvent accessibility prediction, using evolutionary information such as multiple sequence alignment and position-specific scoring matrix generally has given good prediction results [16], [17]. Once an appropriate feature vector has been chosen, a classification algorithm is used to partition the feature space into disjoint regions with decision boundaries. The decision boundaries are determined using feature vectors of a reference sample with known classes, which are also called the reference dataset or training set. The class of a query data is then assigned depending on the region it belongs to.

Various classification algorithms have been developed. Bayesian statistics is a parametric method where the functional form of the probability density is assumed for each class, and its parameters are estimated from the reference data.

In nonparametric methods, no specific functional form for the probability density is assumed. There are various nonparametric methods such as, for example, neural

networks, support vector machines and nearest neighbor methods. In the neural network methods, the decision boundaries are set up before the prediction using a training set. Support vector machines are similar to neural networks in that the decision boundaries are determined before the prediction, but in contrast to neural network methods where the overall error function between the predicted and observed class for the training set is minimized, the margin in the boundary is maximized.

In the k -nearest neighbor methods, the decision boundaries are determined implicitly during the prediction, where the prediction is performed by assigning the query data the class most matched represented among the k -nearest reference data. The standard k -nearest neighbor rule is to place equal weights on the k -nearest reference data for determining the class of the query, but a more general rule is to use weights proportional to a certain power of distance. Also, by assigning the fuzzy membership to the query data instead of a definite class, one can estimate the confidence level of the prediction. The method employing these more general rules is called the fuzzy k -nearest neighbor methods [18].

Neural network methods are very popular and have been widely used for solvent accessibility prediction [1], [7], [19]–[22], and support vector machines, a recently developed method, shows comparable results to neural network methods [23]–[25]. Bayesian statistics has also been used by Thompson and Goldstein (1996).

The k -nearest neighbor method has been frequently used for the classification of biological and medical data, and despite its simplicity, the performances are competitive compared to many other methods. However, the k -nearest neighbor method has few been applied for predicting solvent accessibility, although it has been

used to predict protein secondary structure [26]–[28].

In this thesis, we apply the modified fuzzy k -nearest neighbor method to the prediction of solvent accessibility where PSI-BLAST [29] profiles are used as the feature vectors. We obtain relatively high accuracy on various benchmark tests.

1.2 Thesis Outline

The organization of this thesis is structured as follows. Chapter 1 introduces the motivation and the background of this thesis. In Chapter 2, we will first introduce the data set and the definition of protein solvent accessibility. Then the k -nearest neighbor algorithm and quick radial basis function network will be described. Moreover, we will propose five different methods to predict protein relative solvent accessibility in Chapter 3. In Chapter 4, the experiment of computer simulation and the results are conducted and compared with other methods. Finally, the conclusion and discussion of this thesis is presented in Chapter 5.

Chapter 2. The Data Set Used and Previous Algorithms

2.1 Training and Data Set

The set of 126 nonhomologous globular protein chains used in the experiment of Rost and Sander [1] and referred to as the RS126 set was used to evaluate the accuracy of the prediction. The proteins in the RS126 data set have less than 25% pairwise sequence identity. This set was used to evaluate different methods of relative solvent accessibility prediction, for example, PHDacc [1] and other methods [23], [30], [31]. In this paper, we performed a sevenfold cross-validation test on this set as defined in Table 2.1 [53]. In order to avoid the selection of extremely biased partitions, the RS126 set was divided into subsets of approximately same composition of each type of RSA state. One subset was chosen as the testing set while the rest was merged into the training set. This procedure was repeated seven times to cover whole RS126 data set.

Table 2.1. The database of non-homologous proteins used for seven-fold cross validation. All proteins have less than 25% pairwise similarity for lengths greater than 80 residues.

Fold_A	256b_A	2aat	8abp	6acn	1acx	8adh	3ait
	2ak3_A	2alp	9api_A	9api_B	1azu	1cyo	1bbp_A
	1bds	1bmv_1	1bmv_2	3blm	4bp2		
Fold_B	2cab	7cat_A	1cbh	1cc5	2ccy_A	1cdh	1cdt_A
	3cla	3cln	4cms	4cpa_I	6cpa	6cpp	4cpv
	1crn	1cse_I	6cts	2cyp	5cyt_R		
Fold_C	1eca	6dfr	3ebx	5er2_E	1etu	1fc2_C	1fdl_H
	1dur	1fkf	1fnd	2fxb	1fxi_A	2fox	1g6n_A
	2gbp	1a45	1gd1_O	2gls_A	2gn5		
Fold_D	1gpl	4gr1	1hip	6hir	3hmg_A	3hmg_B	2hmz_A
	5hvp_A	2i1b	3icb	7icd	1il8_A	9ins_B	1l58
	1lap	5ldh	1gdj	2lhb	1lmb_3		
Fold_E	2ltn_A	2ltn_B	5lyz	1mcp_L	2mev_4	2or1_L	1ovo_A
	1paz	9pap	2pcy	4pfk	3pgm	2phh	1pyp
	1r09_2	2pab_A	2mhu	1mrt	1ppt		
Fold_F	1rbp	1rhd	4rhv_1	4rhv_3	4rhv_4	3rnt	7rsa
	2rsp_A	4rxn	1s01	3sdh_A	4sgb_I	1sh1	2sns
	2sod_B	2stv	2tgp_I	1tgs_I	3tim_A		
Fold_G	1bks_A	1bks_B	1tnf_A	1ubq	2tmv_P	2tsc_A	2utg_A
	2wrp_R	4ts1_A	4xia_A	6tmn_E	9wga_A		

2.2 The Definition of Protein Solvent Accessibility

2.2.1 Static Residue Solvent Accessibility

The native structure of globular proteins exists only in the presence of water, and therefore the analysis of their interactions with water is central to the theory of protein structure. The term “accessible surface area” was introduced by Lee and Richards [32] to quantitatively describe the extent to which atoms on the protein surface can form contacts with water. For a particular protein atom it is defined as the area over which the center of a water molecule can be placed while retaining van der Waals contacts with that atom and not penetrating any other atom. The principal goal is to predict the extent to which a residue embedded in a protein structure is accessible to solvent. Solvent accessibility can be described in several ways [32]–[34]. The most detailed fast method compiles solvent accessibility by estimating the volume of a residue embedded in a structure that is exposed to solvent as shown in Fig. 2.1; note: this method was developed by Lee and Richards [32] and later implemented in DSSP [35]. Different residues have a different possible accessible area.

Studies of solvent accessibility in proteins have led to many new insights into protein structure [32]–[37]. Knowledge of solvent accessibility has proved useful for identifying protein function, sequence motifs, and domains, and for formulating hypotheses about antigenic determinants, site-directed mutagenesis, humanization of antibodies, and on the correctness of designed or experimentally determined protein structures. Furthermore, knowledge of solvent accessibility has assisted alignments in regions of remote sequence identity.

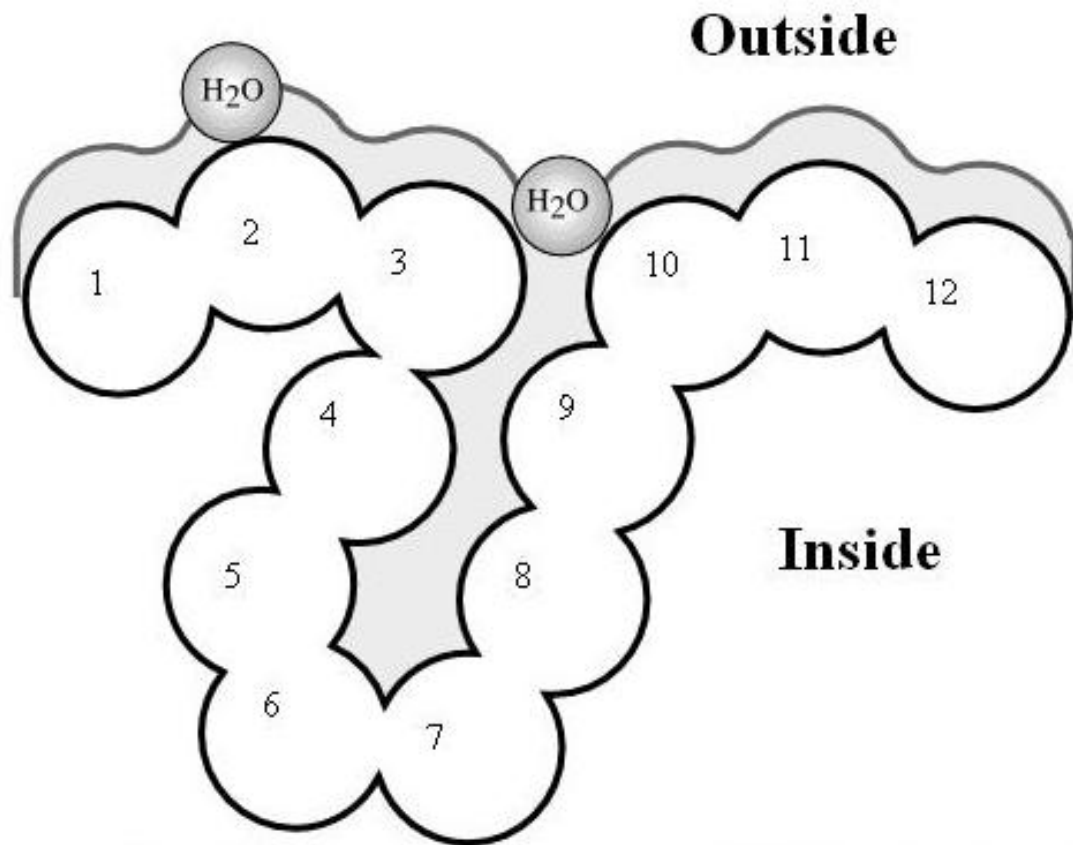


Fig. 2.1. Measure accessibility. Residue solvent accessibility is usually measured by rolling a spherical water molecule over a protein surface and summing the area that can be accessed by this molecule on each residue (typical values range from 0–300 Å²). To allow comparisons between the accessibility of long extended and spherical amino acids, typically relative values are compiled (actual area as percentage of maximally accessible area). A more simplified description distinguishes two states: exposed (here residues numbered 1–3 and 10–12) and buried (here residues 4–9) residues. Since the packing density of native proteins resembles the crystals, values for solvent accessibility provide upper and lower limits to the number of possible inter-residue contacts.

2.2.2 Residue Relative Solvent Accessibility

How can the solvent accessibility of a residue embedded in a 3D structure be cast into a simple number? One simple way is to count the number of water molecules in direct contact with the residue, as estimated by the program DSSP for the first hydration shell. For comparison between amino acids of different sizes, the relative solvent accessibility is a useful quantity as defined in Table 2.2.

Amino acid relative solvent accessibility is the degree to which a residue in a protein is accessible to a solvent molecule. The relative solvent accessibility can be calculated by the formula as follows:

$$RelAcc(\%) = \frac{100 \times Acc}{MaxAcc(\%)}, \quad (2.1)$$

where Acc is the solvent accessible surface area of the residue observed in the 3D structure, given in Angstrom units, calculated from coordinates by the dictionary of protein secondary structure (DSSP) program [35]. The number of water molecules around a residue can be approximated by $Acc/10$, and $MaxAcc$ is the maximum value of solvent accessible surface area of each kind of residue for a Gly-X-Gly extended tripeptide conformation.

Table 2.2. Definition of solvent accessibility states.

- Solvent accessibility:

Acc = solvent accessibility of a residue (given in \AA^2) calculated from coordinates using DSSP [35]. $W \approx Acc/10$, approximates the number of water molecules around the residue.

- Relative solvent accessibility:

$RelAcc = Acc/MaxAcc$, with maximal accessibility (measured in \AA^2) for the amino acids given by the table following (amino acids in one-letter code; B stands for D or N; Z for E or Q, and X for an undetermined amino acid) [37][38].

AA	A	B	C	D	E	F	G	H	I	K	L	M
<i>MaxAcc</i>	106	160	135	163	194	197	84	184	169	205	164	188
AA	N	P	Q	R	S	T	V	W	X	Y	Z	
<i>MaxAcc</i>	157	136	198	248	130	142	142	227	180	222	196	

- Two-state (binary) model for accessibility (B/E) :

	Buried (B)	Exposed (E)
Thresholds to distinguish two states	$RelAcc \leq 0\%$	$RelAcc > 0\%$
	$RelAcc < 5\%$	$RelAcc \geq 5\%$
	$RelAcc < 9\%$	$RelAcc \geq 9\%$
	$RelAcc < 16\%$	$RelAcc \geq 16\%$
	$RelAcc < 25\%$	$RelAcc \geq 25\%$

- Three-state (ternary) model for accessibility (B/I/E) :

	Buried (B)	Intermediate (I)	Exposed (E)
Thresholds to distinguish three states	$RelAcc < 9\%$	$9\% \leq RelAcc < 36\%$	$RelAcc \geq 36\%$

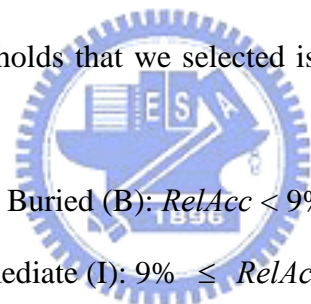
- Measure for evaluation of conservation and accuracy of prediction:

Q_2 percentage of conserved, or correctly predicted, residues in two states defined by thresholds given above.

Q_3 percentage of conserved, or correctly predicted, residues in three states defined by thresholds given above.

RelAcc can hence adopt values between 0% and 100%, with 0% corresponding to a fully buried and 100% to a fully accessible residue, respectively. Different arbitrary threshold values of relative solvent accessibility are chosen to define categories: buried and exposed as shown in Fig. 2.2, or ternary categories: buried, intermediate, or exposed. The precise choice of the threshold is not well defined [1].

We used two kinds of class definitions: (1) buried (B) and exposed (E); and (2) buried (B), intermediate (I), and exposed (E). For the two-state, B and E definition, we chose various thresholds of the relative solvent accessibility such as 25%, 16%, 9%, 5%, and 0%. For the three-state, B, I, and E, description of relative solvent accessibility, one set of thresholds that we selected is the same as those in Rost and Sander [1]:



Buried (B): $RelAcc < 9\%$

Intermediate (I): $9\% \leq RelAcc < 36\%$

Exposed (E): $RelAcc \geq 36\%$

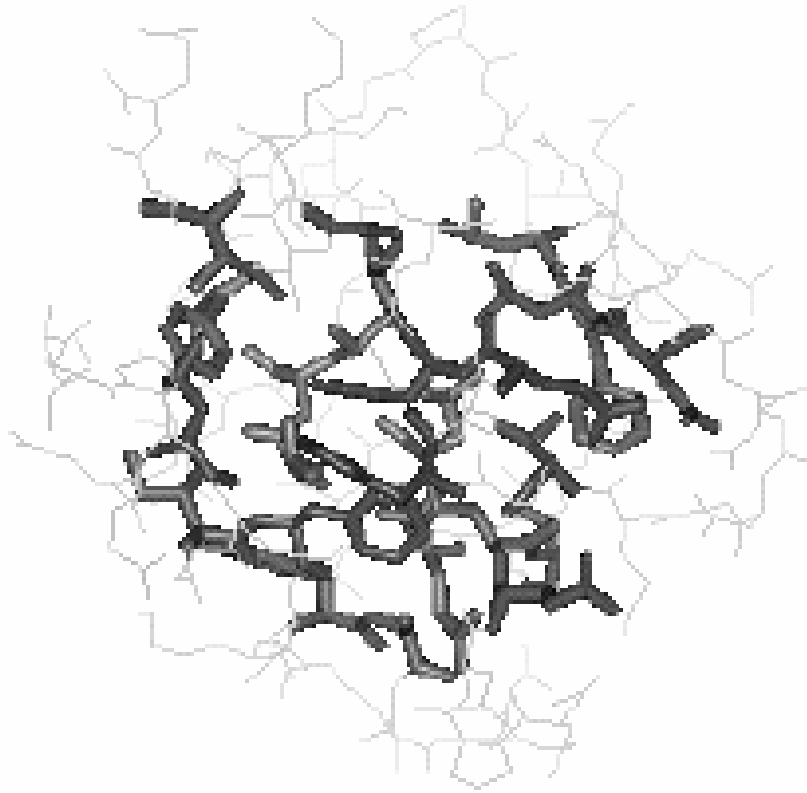


Fig 2.2. Binary model: thick and dark line is buried residues; thin and light line is exposed residues [39].



2.3 PSI-BLAST Profiles

It is well known that evolutionary information in the form of multiple alignments and profiles significantly improves the accuracy of, for instance, secondary structure prediction methods [2], [9], [40]–[42]. This is so because the secondary structure of a family is more conserved than the primary amino acid sequence. Similar effects have been reported for the prediction of contact number and relative solvent accessibility. For relative solvent accessibility, a corresponding increase of 5% has been described both with neural networks [40] and Bayesian methods.

PSI-BLAST [29] generates the profile of a protein in the form of an $N \times 20$ position-specific scoring matrix as shown in Fig. 2.3, where N is the length of the sequence. PSI-BLAST is run with default options, `-j 3`, `-h 0.001`, and `-e 10.0`, and the non-redundant protein sequence database (<ftp://ncbi.nlm.nih.gov/blast/db>) filtered by PFILT [9] to mask out regions of low complexity sequence, the coiled coil regions and transmembrane spans. The BLOSUM62 [43] substitution matrix as shown in Fig. 2.4, is used for PSI-BLAST. These profiles were scaled to the required 0 – 1 range using the standard logistic function:

$$f(x) = \frac{1}{1 + \exp(-x)}, \quad (2.2)$$

where x is the raw profile matrix value.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
S	-1	-3	-1	-2	-3	-2	-2	-2	-3	-4	-4	-2	-3	-4	-3	6	2	-5	-3	-3
I	-2	-2	-4	-4	-3	-3	0	-5	-4	2	4	-1	4	1	-3	-1	-1	-4	-1	0
P	-2	-1	-4	-4	-5	-1	-3	-4	2	-5	-5	-3	-4	-5	8	-2	-2	-5	0	-4
P	-2	-2	-2	-2	-4	0	-1	-4	-2	0	1	-1	2	0	5	-2	-2	-5	-3	0
E	-2	2	-2	2	-4	0	2	-4	-2	2	-2	-3	-2	-2	-1	3	1	-3	1	0
V	-2	-5	-5	-6	-3	-4	-5	-5	-5	3	2	-5	2	4	-5	-4	-3	-3	0	4
K	-2	2	1	0	2	1	1	-4	3	1	-2	3	0	-2	-4	-1	1	-1	1	-1
F	1	-5	-5	-5	-4	-5	-5	-5	-4	1	0	-5	-1	7	-5	-4	-3	-2	2	2
N	-4	-3	7	5	-5	-2	-1	-3	-2	-5	-4	-1	-4	-5	-4	-2	-2	-6	-5	-5
K	-3	6	-1	-3	-5	0	-1	-4	5	-5	-4	2	-3	-5	-4	-2	-2	-5	-3	-5
P	-2	-4	-4	-4	-5	-3	-3	-4	-4	-5	-5	-3	-5	-6	8	-1	-1	-6	-5	-4
F	-4	-5	-5	-6	-5	-5	-5	-5	-3	-2	-2	-5	-2	9	-6	-4	-4	-1	2	-3
V	-3	-4	-5	-6	-3	-3	-5	-6	-1	3	5	-4	2	1	-5	-4	-3	-4	-3	1
F	-3	-5	-5	-6	-1	-4	-5	-5	-4	2	2	-5	3	6	-5	-4	-3	-2	1	2
L	-1	-4	-5	-5	-3	-4	-5	-5	-4	3	3	-4	5	4	-5	-2	-1	-3	-1	1
M	-2	-5	-5	-5	-3	-5	-5	-6	-5	7	1	-5	2	-2	-5	-4	-3	-5	-3	2
I	-3	3	-3	-4	-1	-2	-1	-5	-2	0	0	0	0	3	-4	0	-1	1	4	1
E	-3	-3	0	5	-2	-1	4	-4	5	-5	-4	-2	-4	-5	-2	0	-3	-5	-3	-5
Q	-3	1	2	2	-5	0	3	-2	3	-2	-4	3	-1	-4	-2	-1	1	-1	-2	-1
N	-1	1	3	1	-4	0	1	-2	2	0	-3	1	-1	0	0	0	1	-4	0	-2
T	-1	-3	-1	-3	-3	-3	-3	-2	-1	0	-1	-2	-2	-4	-3	2	6	-5	-4	-1
K	-2	1	2	0	-5	3	0	3	1	-5	-5	2	-4	-3	-1	-1	-3	5	-3	-5
S	0	-2	0	-2	2	-3	-3	-2	0	0	-2	-3	0	-1	-3	5	2	-4	-2	0
P	-1	-4	-4	-2	-3	-3	-4	-5	-5	5	1	-4	-1	-3	4	0	-2	0	-4	2
L	-4	-4	-6	-6	-3	-4	-5	-6	-5	1	6	-5	0	0	-2	-5	-3	-4	-3	0
F	-4	-5	-5	-6	-4	-5	-5	-5	-3	-2	-1	-5	2	9	-6	-4	-4	-1	1	-3
M	-1	-4	-5	-5	1	-3	-4	-5	-4	1	3	-4	7	-2	-5	-4	-3	-4	-2	3
G	1	-4	-2	-3	-4	-4	-4	7	-4	-5	-4	-3	-4	-5	-4	0	-3	-5	-5	-5
K	-2	4	-2	-3	-5	2	-1	-4	1	-3	-4	6	-1	-5	-3	-1	-3	-5	-4	-4
V	-2	-5	-5	-5	-3	-4	-5	-5	-5	3	0	-4	1	0	-4	-2	-4	-3	6	6

Fig. 2.3. Raw profile from PSI-Blast log file



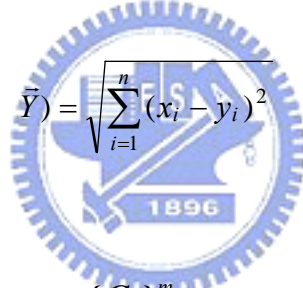
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
	0	-1	1	0	2	1	1	2	1	2	0	0	2	4	1	5	1	2	-2	5	C
		2	0	-2	0	-1	0	0	0	1	0	0	0	1	0	1	-1	1	1	-1	S
C	9		2	-1	-1	-1	0	0	0	0	0	0	-1	0	-1	1	0	1	1	3	T
S	-1	4		2	-2	-1	-1	0	0	-1	-1	-1	1	1	0	-1	0	0	2	1	P
T	-1	1	5		2	-1	-2	-2	-1	0	0	1	1	0	0	1	0	1	1	2	A
P	-3	-1	-1	7		2	0	-1	-2	0	1	1	0	0	-1	0	-1	1	2	4	G
A	0	1	0	-1	4		3	-1	-1	0	0	1	-1	0	-1	0	-1	0	0	0	N
G	-3	0	-2	-2	0	6		2	-1	-1	-1	0	-1	0	0	0	0	2	1	3	D
N	-3	1	0	-2	-2	0	6		1	0	0	2	2	1	-1	0	0	2	2	4	E
D	-3	0	-1	-1	-2	-1	1	6		0	-2	0	1	1	-1	0	0	1	3	3	Q
E	-4	0	-1	-1	-1	-2	0	2	5		2	-1	0	1	0	-1	0	1	2	2	H
Q	-3	0	-1	-1	-1	-2	0	0	2	5		-1	-1	0	-1	1	0	1	3	-4	R
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8		1	-2	-1	1	1	2	3	1	K
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5		-2	-1	-1	0	1	2	4	M
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5		-1	1	0	0	1	3	I
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5		-1	0	-1	1	2	L
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	1	4		0	1	2	4	V	
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	2	2	4		-1	-2	1	F	
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	1	3	1	4		-1	2	Y	
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	-1	6		-1	W	
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3		7	
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11	
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

Fig. 2.4. BLOSUM 62 substitution matrix (Lower) and difference matrix (Upper) obtained by subtracting the PAM 160 matrix position by position. These matrices have identical relative entropies (0.70); the expected value of BLOSUM 62 is -0.52; that for PAM 160 is -0.57.

2.4 K-Nearest Neighbor Algorithm

Nearest neighbor methods are based on learning by analogy. The training samples are described by n -dimensional numeric attributes. Each sample represents a point in an n -dimensional space. In this way, all of the training samples are stored in an n -dimensional pattern space. When given an unknown sample, a k -nearest neighbor classifier searches the pattern space for k training samples that are closest to the unknown sample. The k training samples are the k “nearest neighbors” of the unknown sample. “Closeness” is defined in terms of Euclidean distance, where the Euclidean distance between two points, $\vec{X} = (x_1, x_2, \dots, x_n)$ and $\vec{Y} = (y_1, y_2, \dots, y_n)$ is

$$d(\vec{X}, \vec{Y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.3)$$



Consider the case of m classes $\{C_i\}_{i=1}^m$ and a set of N sample patterns $\{\vec{z}_i\}_{i=1}^N$ whose classification is a priori known. Let \vec{x} denote an arbitrary incoming pattern. The nearest neighbor classification approach classifies \vec{x} in the pattern class of its nearest neighbor in the set $\{\vec{z}_i\}_{i=1}^N$, i.e., if $\|\vec{x} - \vec{z}_j\| = \min_{1 \leq i \leq N} \|\vec{x} - \vec{z}_i\|$ then $\vec{x} \in C_j$ if $\vec{z}_j \in C_j$. This scheme which is basically another type of minimum-distance classification, can be modified by considering the k nearest neighbors to \vec{x} and using a majority-rule type classifier. Its advantage is overcoming class noise in the training set. And the example is shown in Fig. 2.5 :

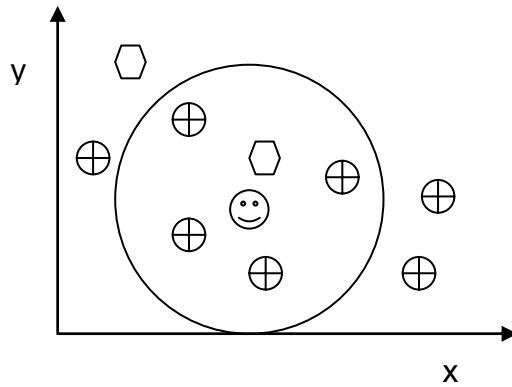


Fig. 2.5. Simple 2-D case, each instance is described only by two values (x , y coordinates). The class is either \oplus or \hexagon .

Inside the circle of Fig. 2.5, we can easily see that the class of simple-NN (1-NN) is \hexagon , and the class of 5-NN is \oplus (\smileyface is the testing data).

Nearest neighbor classifiers are instance-based or lazy-learners in that they store all of the training samples and do not build a classifier until a new (unlabeled) sample needs to be classified. This contrasts with eager learning methods, such as decision tree induction and back propagation, which construct a generalization model before receiving new samples to classify. Lazy learners can incur expensive computational costs when the number of potential neighbors (i.e., stored training samples) with which to compare a given unlabeled sample is large. Therefore, they require efficient indexing techniques. As expected, lazy learning methods are faster at training than eager methods, but slower at classification since all computation is delayed to that time. Unlike decision tree induction and back propagation, nearest neighbor classifiers assign equal weight to each attribute. This may cause confusion when there are many irrelevant attributes in the data.

Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown sample. In this case, the classifier returns the average value of the real-valued labels associated with the k -nearest neighbors of the unknown sample.



2.5 Quick Radial Basis Function Network

Networks based on radial basis functions have been developed to address some of the problems encountered with training multilayer perceptrons: radial basis functions are usually able to converge and the training is much more rapid. Both are feed-forward networks with similar-looking diagrams and their applications are similar; however, the principles of action of radial basis function networks and the way they are trained are quite different from multilayer perceptrons.

A RBFN (radial basis function network) consists of three layers, namely the input layer, the hidden layer and the output layer. The input layer broadcasts the coordinates of the input vector to each of the nodes in the hidden layer. Each node in the hidden layer then produces an activation based on the associated radial basis function. Finally, each node in the output layer computes a linear combination of the activations of the hidden nodes.

For radial basis function networks, each hidden unit represents the center of a cluster in the data space. Input to a hidden unit in a radial basis function is not the weighted sum of its inputs but a distance measure: a measure of how far the input vector is from the center of the basis function for that hidden unit. Various distance measures are used, but perhaps the most common is the well-known Eculidean distance measure.

The Eculidean distance between them is given by

$$D_j = \|\vec{x} - \vec{\mu}_j\| = \sqrt{\sum_i (x_i - y_{ji})^2} \quad (2.4)$$

if \vec{x} is an input vector and $\vec{\mu}_j$ is the location vector of the basis function for hidden node j . The hidden node then computes its outputs as a function of the distance between the input vector and its center. For the Gaussian radial basis function the hidden unit output is

$$h_j(D_j^2) = e^{-D_j^2 / 2\sigma_j^2} \quad (2.5)$$

where D_j is the Euclidean distance between an input vector and the location vector for hidden unit j ; h_j is the output of hidden j and σ_j is a measure of the size of the cluster j (in statistical terms it is called the variance or the square of the standard deviation).

How a RBFN reacts to a given input stimulus is completely determined by the activation functions associated with the hidden nodes and the weights associated with the links between the hidden layer and the output layer. The general mathematical form of the output nodes in an RBFN is as follows:

$$c_r(\vec{x}) = \sum_{j=1}^k w_{rj} \phi(\|x_i - \mu_{ji}\| ; \sigma_j) \quad (2.6)$$

where $c_r(\vec{x})$ is the function corresponding to the r -th output unit (class r) and is a linear combination of k radial basis function $\phi(\cdot)$ with center $\vec{\mu}_j$ and bandwidth component σ_j . Also, \vec{w}_r is the weight vector of class r and w_{rj} is the weight corresponding to the r -th class and j -th center. The general architecture of RBFN is shown as follows.

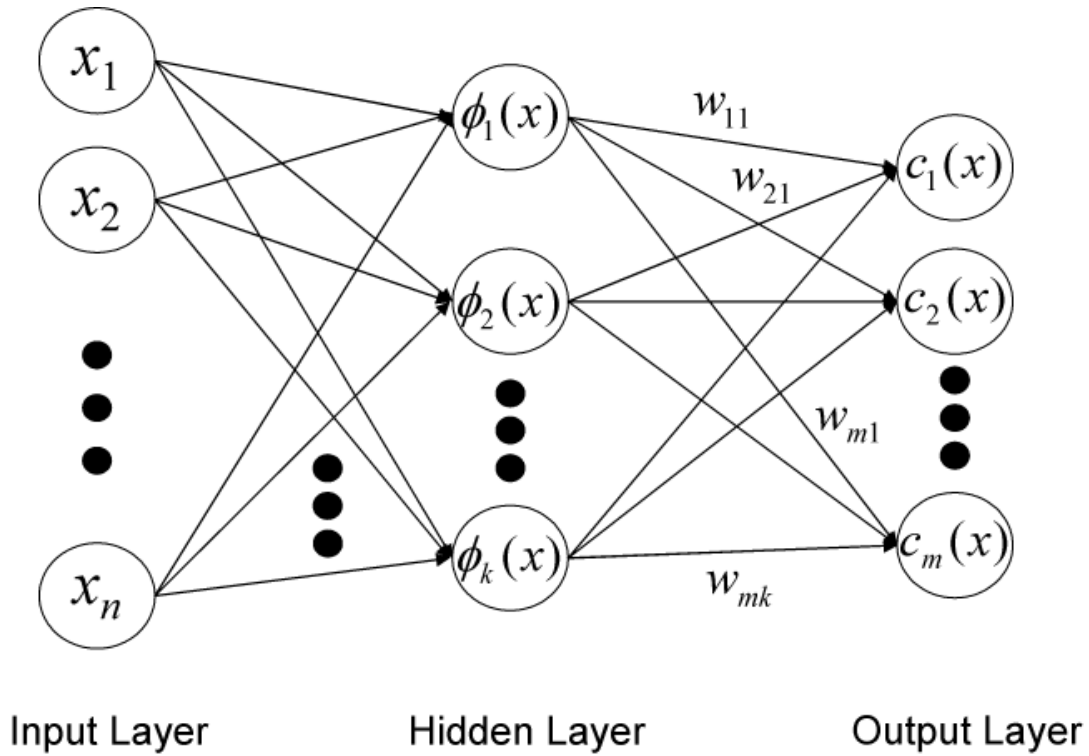


Fig. 2.6. General Architecture of Radial Basis Function Networks.

We can see that constructing an RBFN involves determining the values of three sets of parameters: the centers ($\vec{\mu}_j$), the bandwidths (σ_j) and the weights (w_{rj}), in order to minimize a suitable cost function.

In QuickRBF package, the centers are randomly selected and bandwidth are fixed and set as 5 for each kernel function for conducting the simplest method. The transformation between the inputs and the corresponding outputs of the hidden units is now fixed. The network can thus be viewed as an equivalent single-layer network with linear output units. Then, the LMSE method is used to determine the weights associated with the links between the hidden layer and the output layer.

Assume \mathbf{h} is the output of the hidden layer.

$$\mathbf{h} = [\phi_1(\vec{x}), \phi_2(\vec{x}), \dots, \phi_k(\vec{x})]^T \quad (2.7)$$

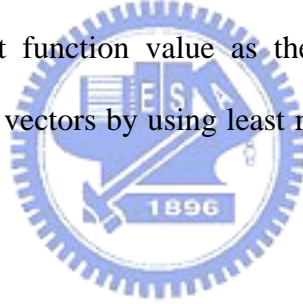
where k is the number of centers, $\phi_1(\vec{x})$ is the output value of first kernel function with input \vec{x} . Then, the discriminant function $c_r(\vec{x})$ of class r can be expressed by the following:

$$c_r(\vec{x}) = \vec{w}_r^T \mathbf{h}, \quad r = 1, 2, \dots, m \quad (2.8)$$

where m is the number of class, and \vec{w}_r is the weight vector of class r . We can show \vec{w}_r as:

$$\vec{w}_r = [w_{r1}(\vec{x}), w_{r2}(\vec{x}), \dots, w_{rk}(\vec{x})]^T \quad (2.9)$$

After calculating the discriminant function value of each class, we choose the class with the biggest discriminant function value as the classification result. We will discuss how to get the weight vectors by using least mean square error method in the following.



For a classification problem with m classes, let \vec{V}_r designate the r -th column vector of an $m \times m$ identity matrix and \mathbf{W} be an $k \times m$ matrix of weights:

$$\mathbf{W} = [\vec{w}_1, \vec{w}_2, \dots, \vec{w}_m] \quad (2.10)$$

Then the objective function to be minimized

$$J(\mathbf{W}) = \sum_{r=1}^m P_r E_r \left\{ \|\mathbf{W}^T \mathbf{h} - \vec{V}_r\|^2 \right\} \quad (2.11)$$

where P_r and $E_r\{\cdot\}$ are the a priori probability and the expected value of class r , respectively.

To find the optimal \mathbf{W} that minimizes J , the gradient of $J(\mathbf{W})$ is set to be zero:

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = 2 \sum_{r=1}^m P_r E_r \{ \mathbf{h} \mathbf{h}^T \} \mathbf{W} - 2 \sum_{r=1}^m P_r E_r \{ \mathbf{h} \} \vec{V}_r^T = [\mathbf{0}] \quad (2.12)$$

where $[\mathbf{0}]$ is a $k \times m$ null matrix. Let \mathbf{K}_r denote the class-conditional matrix of the second-order moments of \mathbf{h} , i.e.

$$\mathbf{K}_r = E_r \{ \mathbf{h} \mathbf{h}^T \} \quad (2.13)$$

If \mathbf{K} denotes the matrix of the second-order moments under the mixture distribution, we have

$$\mathbf{K} = \sum_{r=1}^m P_r \mathbf{K}_r \quad (2.14)$$

Then Eq. (2.12) becomes

$$\mathbf{K} \mathbf{W} = \mathbf{M} \quad (2.15)$$

where

$$\mathbf{M} = \sum_{r=1}^m P_r E_r \{ \mathbf{h} \} \vec{V}_r^T \quad (2.16)$$

If \mathbf{K} is nonsingular, the optimal \mathbf{W} can be calculated by

$$\mathbf{W}^* = \mathbf{K}^{-1} \mathbf{M} \quad (2.17)$$

However, there is a critical drawback of this method. That is, \mathbf{K} may be singular and this will crash the whole procedure. By observing the matrix $\mathbf{h} \mathbf{h}^T$, we are aware of that the matrix $\mathbf{h} \mathbf{h}^T$ is symmetric positive semi-definite (PSD) matrix with rank equal to 1. Since \mathbf{K} is the summation of $\mathbf{h} \mathbf{h}^T$ for each training instance, \mathbf{K} is also a PSD matrix with rank smaller than n . However, PSD matrix may be a singular matrix, so we should add the regularization term to make sure the matrix will be invertible. In the regularization theory, it consists in replacing the objective function as follows:

$$J(\mathbf{W}) = \sum_{r=1}^m P_r E_r \left\{ \left\| \mathbf{W}^T \mathbf{h} - \vec{V}_r \right\|^2 \right\} + \lambda \sum_{r=1}^m \vec{w}_r^T \vec{w}_r \quad (2.18)$$

where λ is the regularization parameter.

Then the Eq. (2.15) becomes

$$(\mathbf{K} + \lambda \mathbf{I}) \mathbf{W} = \mathbf{M} \quad (2.19)$$

If we set $\lambda > 0$, $(\mathbf{K} + \lambda \mathbf{I})$ will be a positive definite (PD) matrix and therefore is nonsingular. The optimal \mathbf{W}^* can be calculated by

$$\mathbf{W}^* = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{M} \quad (2.20)$$

However, the PD matrix has many good properties, and one of them is a special and efficient triangular decomposition, Cholesky decomposition. By using Cholesky decomposition, we can decompose the $(\mathbf{K} + \lambda \mathbf{I})$ matrix as follows:

$$(\mathbf{K} + \lambda \mathbf{I}) = \mathbf{L}\mathbf{L}^T \quad (2.21)$$

where \mathbf{L} is a lower triangular matrix. Then, the Eq. (2.19) becomes

$$(\mathbf{L}\mathbf{L}^T) \mathbf{W} = \mathbf{M} \quad (2.22)$$

Actually, the linear system can be solved efficiently by using back-substitution twice. Finally, we can get the optimal \mathbf{W}_r^* for class r from \mathbf{W}^* , and then the optimal discriminant function $c_r(\vec{x})$ for class r is derived. By using the regularization theory, the optimal weights can be obtained analytically and efficiently.

Chapter 3. Protein Relative Solvent Accessibility Prediction

3.1 Fuzzy K -Nearest Neighbor Approach

The nearest neighbor algorithm is a simple classification algorithm; a query data is classified according to the classification of the nearest neighbor from a database of known classifications. A natural generalization of the nearest neighbor algorithm is the so-called k -nearest neighbor algorithm, where the k -nearest samples are selected and the query data is assigned the class most frequently represented among them. A further extension is to weight the k -nearest samples with a certain power of the distance from the query data. Also, instead of assigning a definite class to the query data, one can calculate the fuzzy membership (see below), which can be used to reflect the confidence level of each nearest neighbor in its prediction. The algorithm incorporating these generalizations is called the fuzzy k -nearest neighbor algorithm [18].

Despite its simplicity, nearest neighbor methods can give competitive performance compared to many other methods. The nearest neighbor methods have been used to predict protein secondary structure [26]–[28] and classify biological and medical data. Also it has been reported that performances of classification were improved by using fuzzy k -nearest neighbor algorithms [44]–[49]. However the k -nearest neighbor method has few been used to predict protein solvent accessibility.

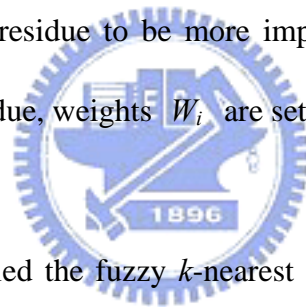
As with Hua and Sun's work [50], the present analysis used the classical local coding scheme of the protein sequences with a sliding window. PSI-BLAST matrix

with n rows and 20 columns can be defined for single sequence with n residues. Each residue is represented using 20 components in a vector, based on the PSSM. Then, each input vector has $20 \times w$ components, where w is a sliding window size.

They constructed a window of size 15 centered on a target residue [9], [23], [24], and use the profile that falls within this window, a 15×20 matrix, as a feature vector. Then, the distance between two feature vectors A and B is defined as

$$D_{AB} = \sum_{i,j} W_i |P_{ij}^{(A)} - P_{ij}^{(B)}|, \quad (3.1)$$

where $P_{ij}^{(A)}$ ($i = 1, 2, \dots, 15; j = 1, 2, \dots, 20$) is a component of the feature vector A, and W_i is a weight parameter. Since it is expected that the profile elements for residues nearer to the target residue to be more important in determining the local environment of the target residue, weights W_i are set to $W_i = (8 - |8 - i|)^2$.



In their work, they applied the fuzzy k -nearest neighbor method to the solvent accessibility prediction. In the fuzzy k -nearest neighbor method, the fuzzy class membership $u_i(x)$ to the class i is assigned to the query data x according to the following equation:

$$u_i(x) = \frac{\sum_{j=1}^k u_i(x^{(j)}) D_j^{-2/(m-1)}}{\sum_{j=1}^k D_j^{-2/(m-1)}}, \quad i = 1, 2, \dots, c, \quad (3.2)$$

where m is a fuzzy strength parameter, which determines how heavily the distance is weighted when calculating each neighbor's contribution to the membership value, k is the number of nearest neighbors, and c is the number of classes. Also, D_j is the distance between the feature vector of the query data x and the feature vector of its j th nearest reference data $x^{(j)}$, and $u_i(x^{(j)})$ is the membership value of $x^{(j)}$ to

the i th class, which is 1 if $x^{(j)}$ belongs to the i th class, and 0 otherwise. The advantage of the fuzzy k -nearest neighbor algorithm over the standard k -nearest neighbor method is quite clear. Modulating the i th neighbor's fuzzy class membership $u_i(x)$ through its percentile distance to the query residue can be considered as the estimate of the probability that the query data belongs to class i , and provides us with more information than a definite prediction of the class for the query data. Moreover, the reference samples which are closer to the query data are given more weight, and an optimal value of m can be chosen along with that for k , in contrast to the standard k -nearest neighbor method with a fixed value of $2/(m-1) = 0$. In fact, the optimal value of k and m are found from the leave-one-out cross-validation procedure, and the resulting value for $2/(m-1)$ is indeed nonzero.

We adopt the optimal values of m and k in [31], which are $(m, k) = (1.33, 65)$ for the 3-state prediction (for both 9% and 36% thresholds) and $(m, k) = (1.50, 40), (1.25, 75), (1.29, 65)$ and $(1.33, 65)$ for the 2-state predictions (for 0, 5, 16, and 25% thresholds, respectively). Moreover, we use $(m, k) = (1.27, 70)$ for the 9% threshold, whose prediction accuracy is slightly higher than other (m, k) values.

3.2 Modified Fuzzy K-Nearest Neighbor Approach

In Sec. 3.1 above, we can see $u_i(x^{(j)})$ is defined as the membership value of $x^{(j)}$ to the i th class, which is 1 if $x^{(j)}$ belongs to the i th class, and 0 otherwise. Here, we modify the definition of $u_i(x^{(j)})$ in Eq. (3.2). It is expected that a neighbor residue close to the threshold $RelAcc$ chosen is not as decisive in determining query

values as a neighbor residue far from the residue's *RelAcc* state. For two-state model for accessibility, as shown in Table 2.2, we have to choose a threshold to distinguish the two states (Buried and Exposed). If we choose a value *Th* (must between 0 and 1) as our threshold, the residues where *Relacc* values range from 0 to *Th* will be classified to the buried state, and others (from *Th* to 100%) will be classified to the exposed state. So the two boundaries of buried class are 0 and *Th*, and the two of exposed one are *Th* and 1. The range lies in $[0, Th]$ for the buried state and $[Th, 1]$ for the exposed state.

It is known that 0 is the minimum for *RelAcc* value, and 1, 100%, is the maximum. That means 0 is the most buried point and 100% is the most exposed one. For each residue of a protein sequence, we can calculate a “buried distance, D_B ” which represents the “distance” from present residue to 0 and a “exposed distance, D_E ” which represents the “distance” from present residue to 1. If the *RelAcc* value of a residue is smaller than *Th*, then we calculate its D_B and D_E values by the following equations:

$$D_B = \frac{RelAcc}{Th},$$

$$D_E = 1 + \frac{Th - RelAcc}{Th}.$$

In contrast, if the *RelAcc* value is larger than *Th*, we calculate the D_B and D_E values by the equation shown below:

$$D_B = 1 + \frac{RelAcc - Th}{1 - Th},$$

$$D_E = \frac{1 - RelAcc}{1 - Th}.$$

In both conditions, if the value of D_B is larger, then the “buried degree” of this

residue should be small. That is, D_B is inversely proportional to the “buried degree.” Similarly, D_E value is also inversely proportional to the “exposed degree.” With this concept in mind, we can use D_B and D_E to calculate membership values $u_i(x^{(j)})$:

$$u_1(x^{(j)}) = u_E(x^{(j)}) = \frac{1/D_E}{1/D_E + 1/D_B},$$

$$u_2(x^{(j)}) = u_B(x^{(j)}) = \frac{1/D_B}{1/D_E + 1/D_B}.$$

Obviously, if we let $u_B(x^{(j)}) = u_E(x^{(j)}) = 0.5$ in both conditions (buried an exposed) to calculate $RelAcc$ value, then we will obtain that $RelAcc = Th$. That means the membership values of both classes at the threshold Th are 50%. The membership functions are shown in Fig. 3.1. The flowchart to calculate 2-state membership values $u_E(x^{(j)})$ and $u_B(x^{(j)})$ is shown in Fig. 3.2.

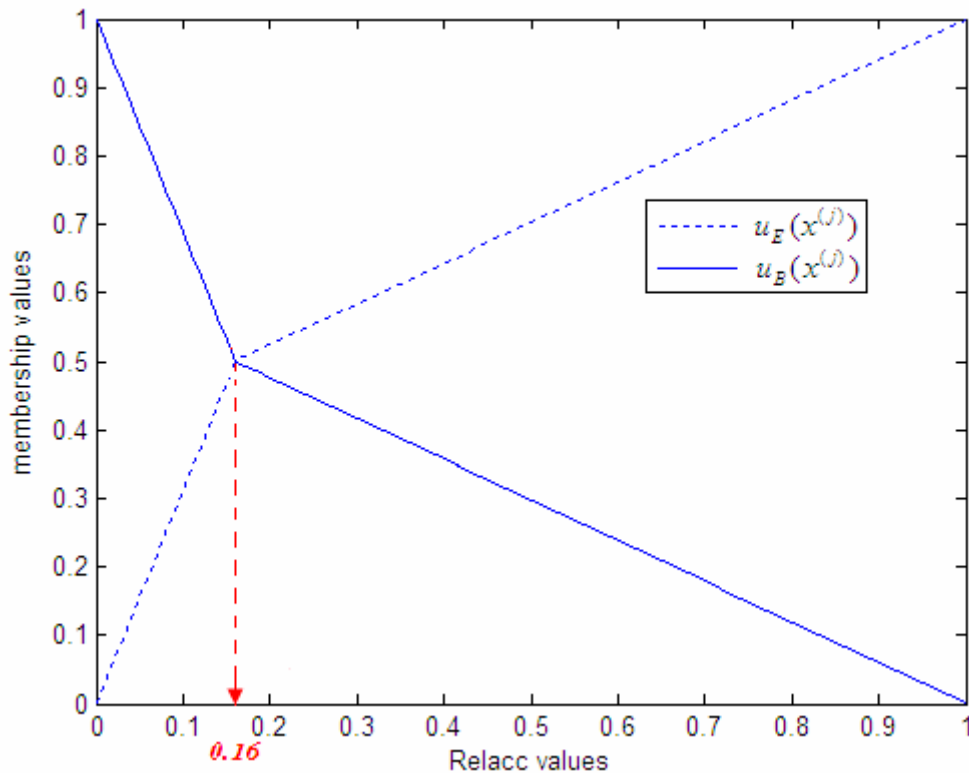


Fig. 3.1. The 2-state membership functions $u_E(x^{(j)})$ and $u_B(x^{(j)})$ with $Th = 16\%$.

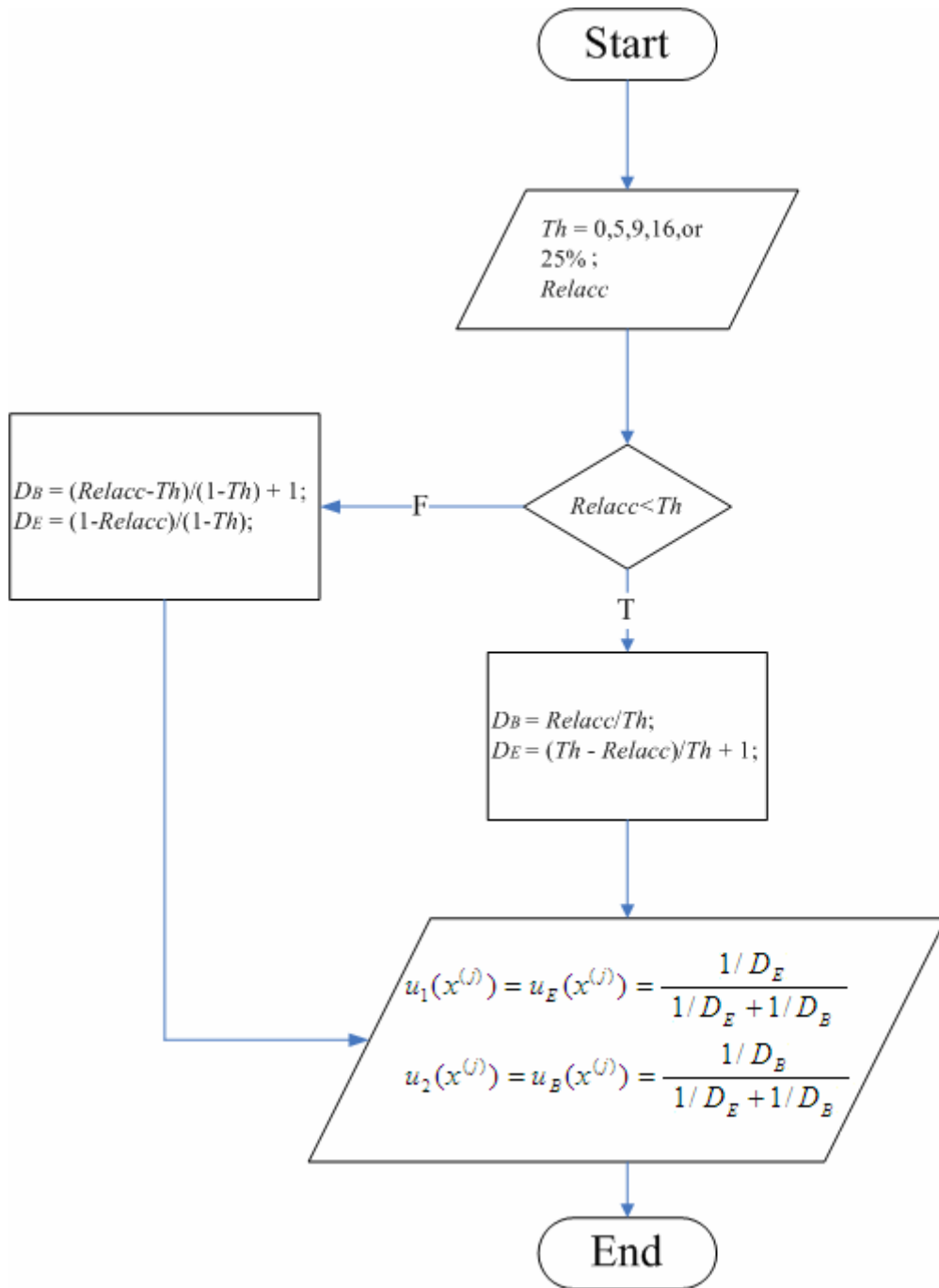


Fig. 3.2. The flowchart to calculate 2-state membership values $u_1(x^{(j)})$ and $u_2(x^{(j)})$.

For the three-state model for accessibility, as shown in Table 2.2, we know that

the boundaries are 0 and 9% for the buried state, 9% and 36% for the intermediate state, 36% and 100% for the exposed state, so the center value of the intermediate class, $\frac{0.09+0.36}{2}$, is 0.225. $u_B(x^{(j)})$ is set to zero when the *RelAcc* value of a residue is greater than 0.225, so we can calculate $u_I(x^{(j)})$ and $u_E(x^{(j)})$ by two-class method given above. In the same manner, we set the $u_E(x^{(j)})$ to zero when *RelAcc* value is smaller than 0.225, and we can calculate $u_I(x^{(j)})$ and $u_B(x^{(j)})$ as above. The three-state membership functions are shown in Fig. 3.3. The flowchart to calculate 3-state membership values is shown in Fig. 3.4.

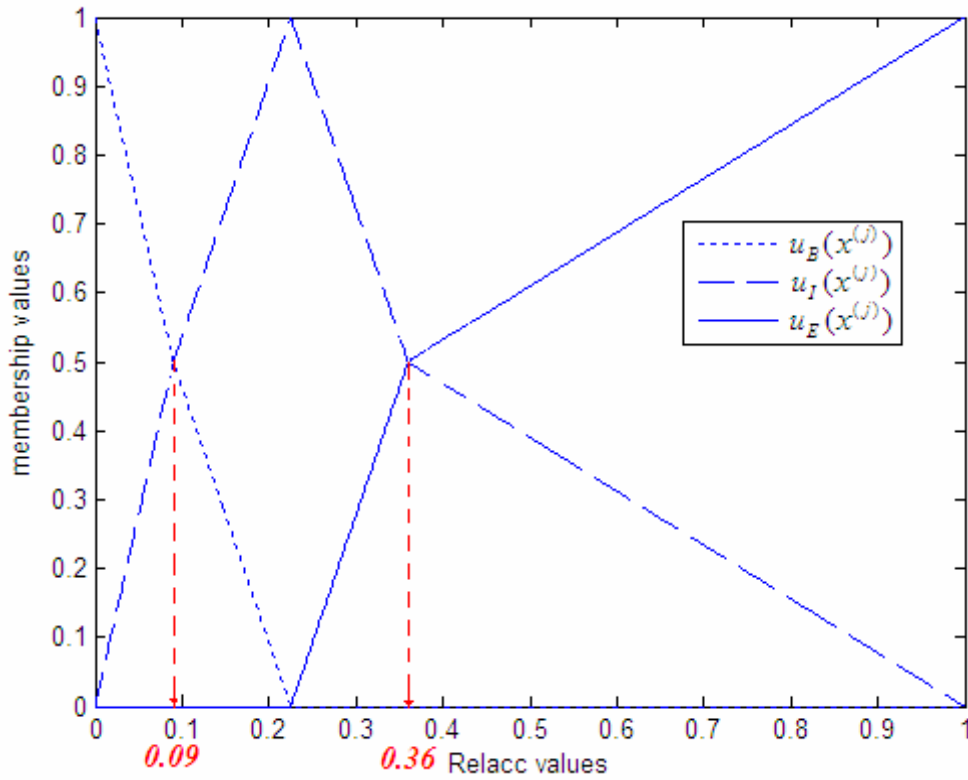


Fig. 3.3. The 3-state membership functions $u_E(x^{(j)})$, $u_I(x^{(j)})$, and $u_B(x^{(j)})$ with $Ths = 9\%$ and 36% .

Therefore, in this setting of membership functions, the values of $u_i(x^{(j)})$ in Eq. (3.2) is not only a crisp value 0 or 1, but a membership degree in $[0, 1]$, and satisfy

$\sum_{i=1}^c u_i(x^{(j)}) = 1$. After this modification, we continue to calculate all the $u_i(x)$ by Eq. (3.2) by the method shown in Sec.3.1, and the maximal $u_i(x)$ class is assumed to be the target class of the residue x .

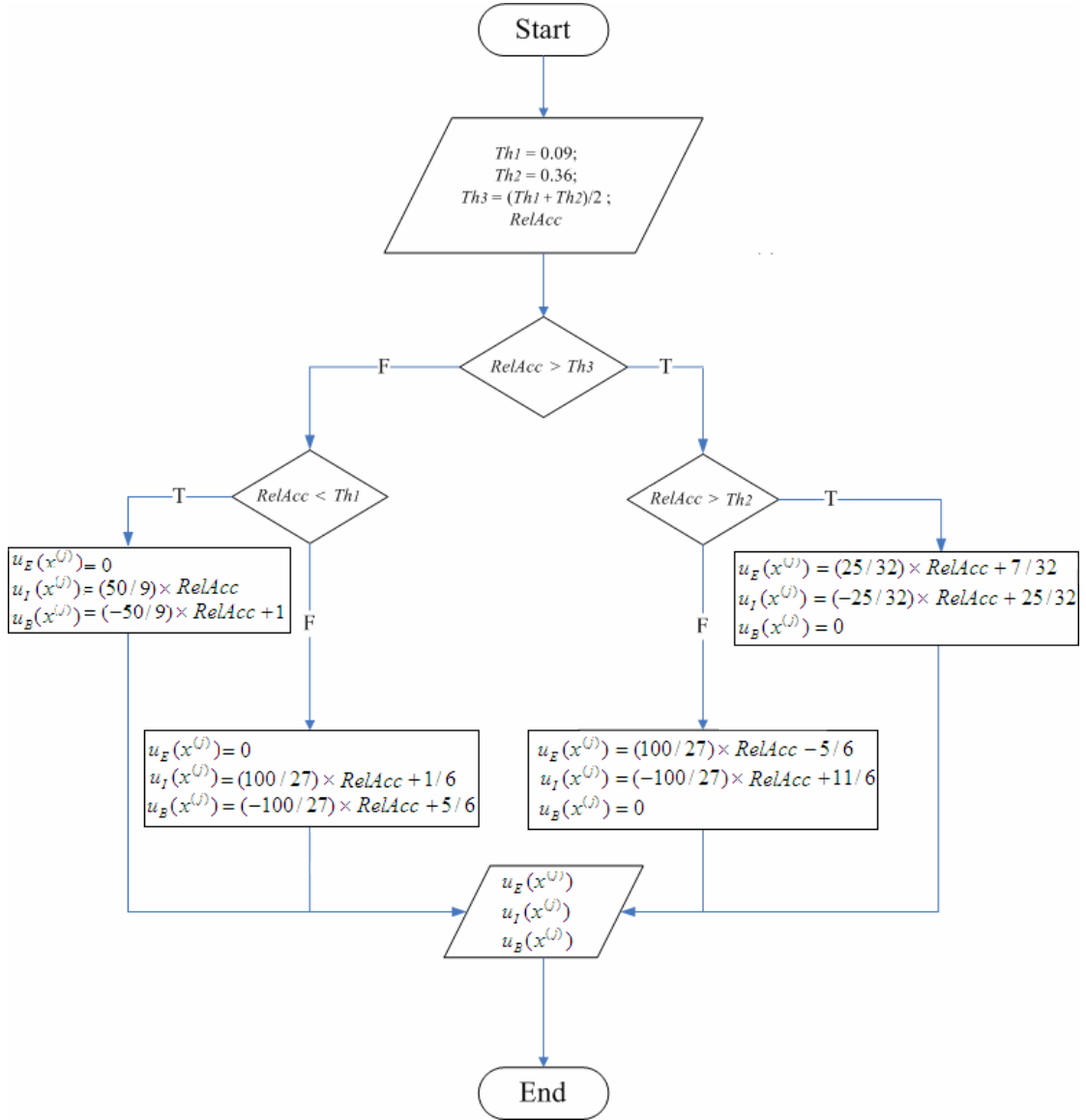


Fig. 3.4. The flowchart to calculate 3-state membership values $u_E(x^{(j)})$, $u_I(x^{(j)})$, and $u_B(x^{(j)})$.

3.3 QuickRBF Approach

A QuickRBF structure used for solvent accessibility prediction system are shown in Fig. 3.5. The QuickRBF classifier classifies each residue of each sequence into the three relative solvent accessibility states, E, I, or B, by using the values of matrices of PSI-BLAST profile as the inputs. The outputs represent the tendency that the residue belongs to that state. The one-against-rest strategy was used for the multiclass classification, so each residue was classified into the state with the largest output value for a QuickRBF approach.



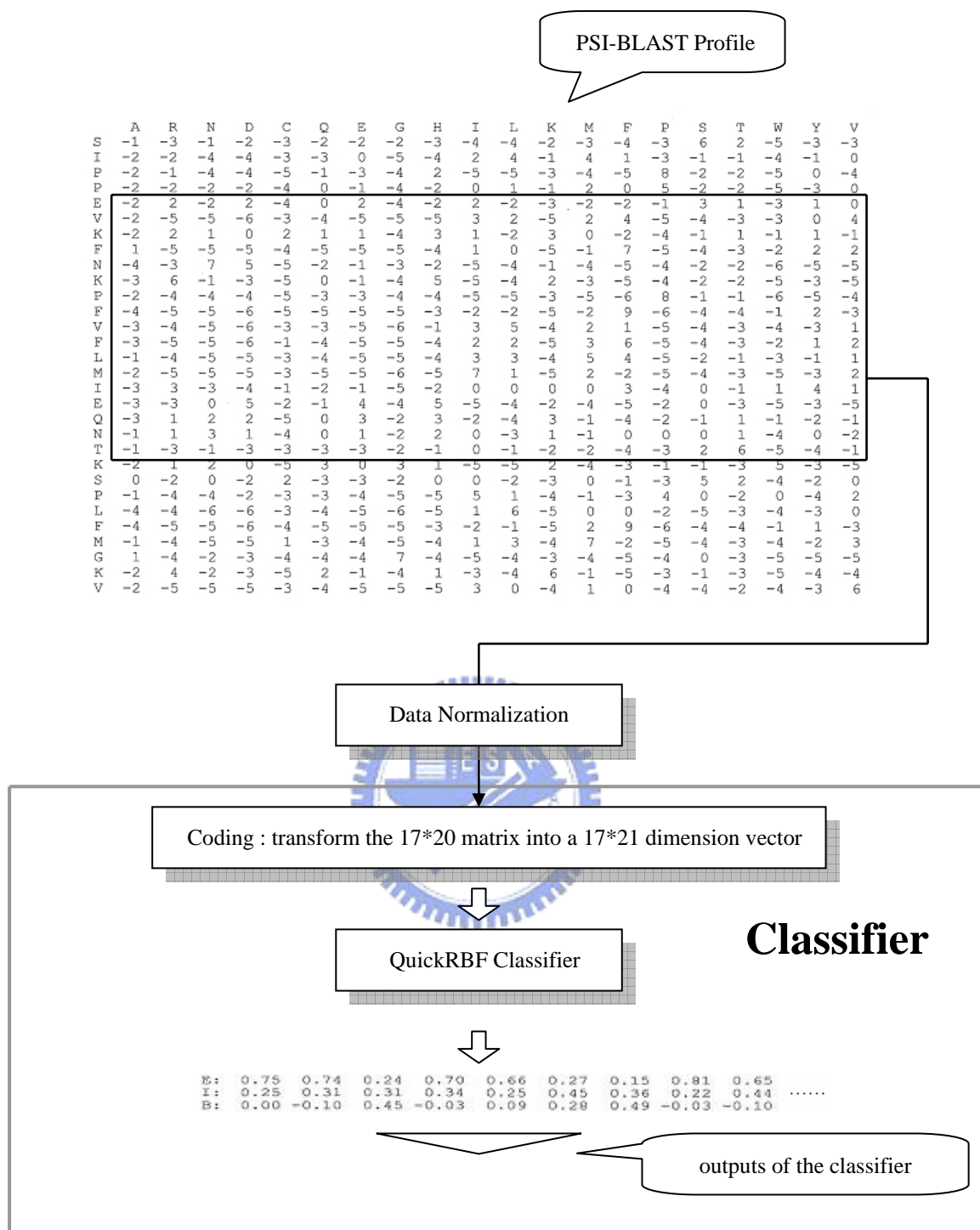


Fig. 3.5. Architecture of QuickRBF method. The system includes two parts: the PSI-BLAST profile, and the classifier. The profile is transformed into a number of 21*17 dimension vectors using the slide-window method. These vectors are input into the QuickRBF classifier. The outputs of the QuickRBF classifier are a number of 3D vectors representing the tendency that the residue belongs to that state. The one-against-rest strategy was used to classify each residue into the state with the largest value.

3.4 Fusion Method

3.4.1 Linear Combination Fusion 1

We have proposed modified fuzzy k -NN approach and QuickRBF approach, both based on the PSSM profiles, to predict the protein relative solvent accessibility. We have found the differences of predictive results between the two approaches are somewhat distinct. This motivates us to design a fusion scheme combining the results of both approaches in order to raise the overall accuracy.

After observing the output values of modified fuzzy k -NN and QuickRBF, we found their dynamic ranges are different. In modified fuzzy k -NN approach shown in Sec. 3.2, the membership values $u_i(x^{(j)})$ of all the two or three classes calculated range between 0 to 1, but the outputs of the QuickRBF approach do not vary in this range. We need to normalize them first before we can make the fusion. Here, we calculated the mean of each class in three-state prediction result for each method, respectively. For each method, these three values (m_E, m_I , and m_B) were used as the “normalization factor.” We divided all the prediction results of each class by its mean to get normalized output. At last, we combined by adding the normalized output of both methods, and the final output class of each residue is assigned to the one with the largest output value. This normalization procedure is shown in Fig. 3.6. The normalization was done similarly in the two-state case.

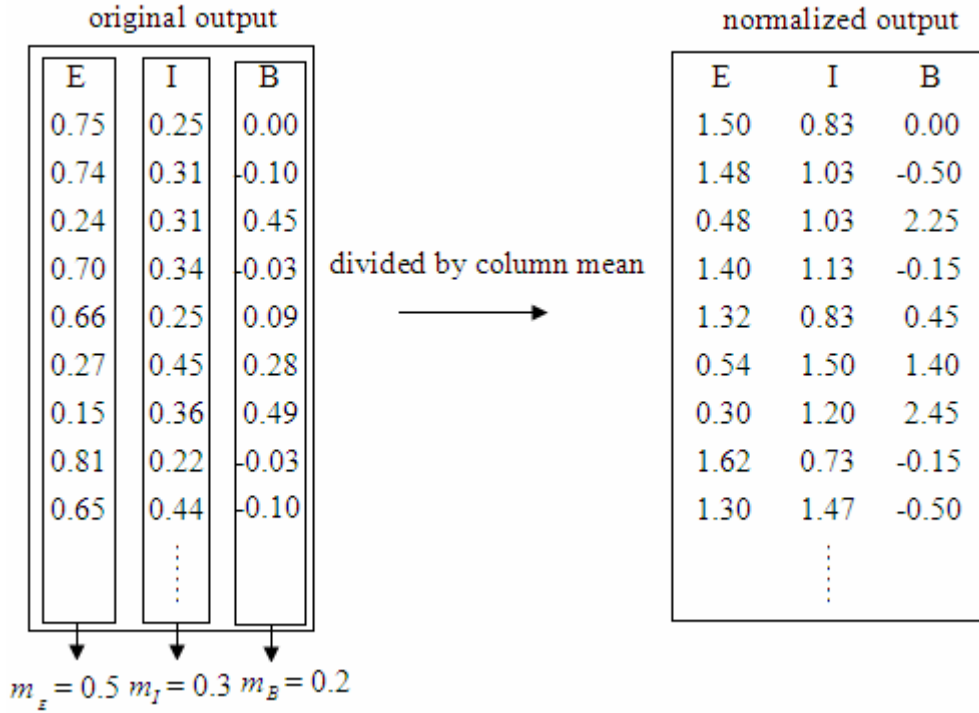


Fig. 3.6. The normalization procedure in Linear Combination Fusion 1.

Referring to Table 3.1, each residue of the present sequence has three target labels denoted as **E**, **I** and **B**. The respective results of each label are the linear combination of results of the normalized QuickRBF and modified fuzzy k -NN. Then the final output class of each residue is assigned to the one with the largest output value. Namely, it applies

$$P_l(x^{(j)}) = \arg \max_l f_l(x^{(j)}) \quad (l \in \{E, I, B\}, j = 1, 2, \dots, m) \quad (3.3)$$

where f_l are the linearly combined values shown below:

$$\begin{aligned} f_E &= E_f(j) = E_1(j) + E_2(j) \\ f_I &= I_f(j) = I_1(j) + I_2(j) \\ f_B &= B_f(j) = B_1(j) + B_2(j) \end{aligned} \quad (3.4)$$

	Modified FKNN			QuickRBF			Fusion Method 1 (Linear Combination)			result
	Exposed	Intermediate	buried	Exposed	Intermediate	buried	Exposed	Intermediate	buried	Class
$x^{(1)}$	$E_1(1)$	$I_1(1)$	$B_1(1)$	$E_2(1)$	$I_2(1)$	$B_2(1)$	$E_f(1)$	$I_f(1)$	$B_f(1)$	$P_1(1)$
$x^{(2)}$	$E_1(2)$	$I_1(2)$	$B_1(2)$	$E_2(2)$	$I_2(2)$	$B_2(2)$	$E_f(2)$	$I_f(2)$	$B_f(2)$	$P_1(2)$
$x^{(3)}$	$E_1(3)$	$I_1(3)$	$B_1(3)$	$E_2(3)$	$I_2(3)$	$B_3(3)$	$E_f(3)$	$I_f(3)$	$B_f(3)$	$P_1(3)$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$x^{(m)}$	$E_1(m)$	$I_1(m)$	$B_1(m)$	$E_2(m)$	$I_2(m)$	$B_3(m)$	$E_f(m)$	$I_f(m)$	$B_f(m)$	$P_1(m)$

Table 3.1. Fusion method : Linear combination.

3.4.2 Linear Combination Fusion 2



We have also developed a different normalization method in this section. We obtained the absolute values of a residue in QuickRBF output data, and summed them up as the “normalization factor.” Then, we divided all the three prediction values E , I , and B of the residue by the normalization factor to obtain their respective normalized outputs. After the normalization, we combined the normalized output by adding the modified fuzzy k -NN and QuickRBF in the same method shown in Sec. 3.4.1. This normalization procedure is shown in Fig. 3.7.

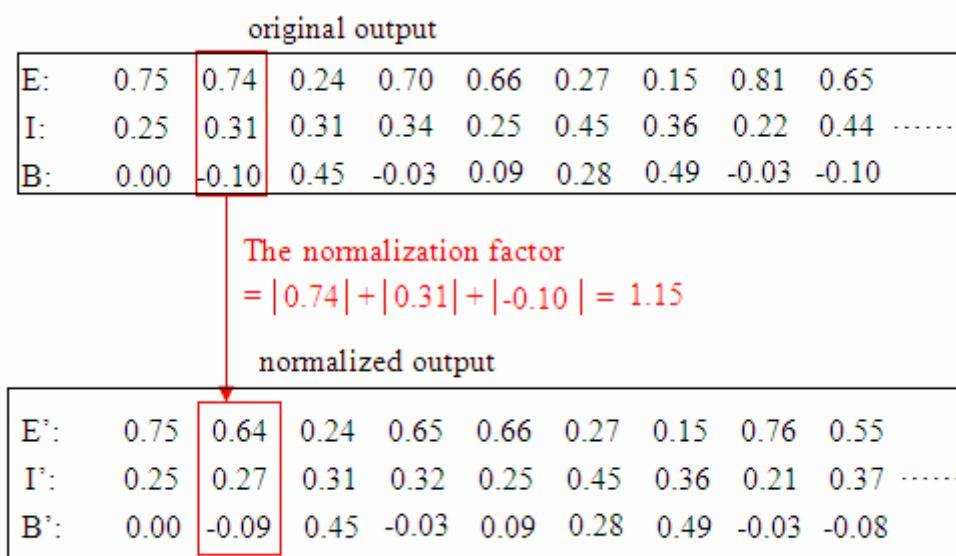


Fig. 3.7. The normalization procedure in Linear Combination Fusion 2.

3.4.3 Reliability Index



The prediction reliability index (RI) was used to access the effectiveness of the approaches for the prediction of the secondary structure of a new sequence. The RI offers an excellent tool for focusing on key regions having high prediction accuracy. Hence, we used reliability index in protein relative solvent accessibility prediction. There are different definitions of the RI. Here we used a definition similar to that proposed by Rost and Sander: $RI = \text{maximal_output}(I) - \text{Second_largest_output}(I)$ [51]. If the value of $RI > 0.9$, then set $RI = 0.9$, so the value of RI is between 0 and 0.9. The prediction accuracy of residues with higher RI values is much better than those with lower RI values. Therefore, the definition of RI reflects the prediction reliability.

In this research, to combine the output from modified fuzzy k -NN and

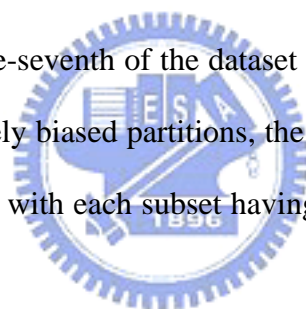
QuickRBF (after normalization shown in 3.4.2), we developed a new classifier design using RI. In this scheme, the classifier with a maximum RI is chosen as the arbiter classifier for the decision of the class. The final class is chosen to the largest output class of this arbiter classifier. For example, if the E/I/B output of the decision function of QuickRBF: **E/I/B**, modified fuzzy k -NN: **E/I/B** are 0.1/0.2/0.7 and 0.3/0.2/0.5, their RI's are 0.5, $0.7 - 0.2$, and 0.2 , $0.5 - 0.3$, respectively. Therefore, the classifier with highest RI, here QuickRBF, is chosen as the arbiter. In this example, the query residue is assigned to be buried since it is the largest class value of the QuickRBF classifier.



Chapter 4. Experiment and Simulation Results

4.1 Datasets

The set 126 nonhomologous globular protein chains used in the experiment of Rost and Sander [1], referred to as the RS126 set, was utilized to evaluate the accuracy of the classifiers. The RS126 dataset contains 23606 residues. Fuzzy K -Nearest Neighbor approaches and QuickRBF approaches were implemented with multiple sequence alignments, and tested on the dataset using a seven-fold cross validation technique to estimate the prediction accuracy. With seven-fold cross validation, approximately six-seventh of the RS126 dataset was selected for training and, after training, the left one-seventh of the dataset was used for testing. In order to avoid the selection of extremely biased partitions, the RS126 set was divided, by [53] previously, into seven subsets with each subset having similar size as shown in Table 2.1.



4.2 Results

4.2.1 Results of Fuzzy K -NN and Modified Fuzzy K -NN Classifiers

Fuzzy k -nearest neighbor approaches are applied on three-state, E, I, and B, and two-state, E and B, relative solvent accessibility predictions. For both classifiers, each residue of sequences is coded as a 20-dimensional vector, which the 20 elements of the vector are the corresponding elements in PSI-BLAST matrix. The window length is 15 and the dimension of the feature vector is 20×15 .

The results and the comparison of the fuzzy k -nearest neighbor approach and modified fuzzy k -nearest neighbor approach on RS126 data set are listed in Table 4.1. On the RS126 data set processed by ourselves, fuzzy k -nearest neighbor approach [31] led to the overall prediction accuracy 58.14% for the three-state prediction with respect to thresholds: 9% and 36%; and 87.93%, 79.18%, 77.59%, 75.35%, 73.49% for the two-state prediction with the thresholds of 0%, 5%, 9%, 16%, and 25%, respectively.

Modified fuzzy k -nearest neighbor approach gave the overall prediction accuracy 58.57% for the three-state prediction with respect to the following two thresholds chosen: 9% and 36%; and 87.93%, 79.84%, 77.76%, 76.34%, 75.26% for the two-state prediction with the chosen thresholds of 0%, 5%, 9%, 16%, and 25%, respectively.

Table 4.1. RSA classification accuracy of ours and previous fuzzy k -nearest neighbor methods on the RS126 data set with PSI-BLAST pssm profiles.

Fuzzy k -NN classifiers [31]

Accuracy: %						
thresholds dataset	3-state (9% ; 36%)	2-state (0%)	2-state (5%)	2-state (9%)	2-state (16%)	2-state (25%)
Fold_A	58.33	86.37	78.63	76.10	75.00	73.97
Fold_B	58.22	88.93	79.84	78.76	76.14	73.17
Fold_C	61.23	87.93	80.11	79.22	77.74	76.18
Fold_D	57.01	88.38	78.28	77.30	74.26	72.65
Fold_E	58.52	88.75	79.90	79.21	76.21	73.15
Fold_F	55.90	88.99	79.54	77.19	73.22	71.20
Fold_G	57.30	86.38	78.09	75.39	74.52	73.66
Average	58.14	87.93	79.18	77.59	75.35	73.49

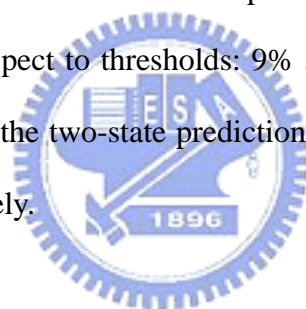
Ours modified fuzzy k -NN classifiers

Accuracy: %						
thresholds dataset	3-state (9% ; 36%)	2-state (0%)	2-state (5%)	2-state (9%)	2-state (16%)	2-state (25%)
Fold_A	59.14	86.37	77.90	77.50	77.27	76.61
Fold_B	59.00	88.93	81.35	78.52	77.03	75.49
Fold_C	60.39	87.93	80.92	78.52	77.71	77.08
Fold_D	57.99	88.38	80.33	77.30	75.40	74.81
Fold_E	59.84	88.75	81.54	77.99	76.21	74.99
Fold_F	56.95	88.99	79.54	77.59	74.31	72.54
Fold_G	55.80	86.38	77.26	76.77	75.72	74.22
Average	58.57	87.93	79.84	77.76	76.34	75.26

4.2.2 Results of QuickRBF approach and the Fusion Methods

For QuickRBF approach, each residue is coded as a 21-dimensional vector, where the first 20 elements of the vector are the corresponding elements in PSI-BLAST matrix and the last element was added in order to allow a window to extend over the N- and the C-terminus. The window length is 17 and the dimension of the feature vector is 21×17 . The number of the centers randomly selected from the training data set is 2000 and the bandwidth is five for each kernel function. The architecture of QuickRBF in the three-state prediction is shown in Fig. 3.5, Sec. 3.3.

QuickRBF approach produced the overall prediction accuracy 60.36% for the three-state prediction with respect to thresholds: 9% and 36%; and 87.76%, 81.15%, 79.06%, 77.64%, 76.17% for the two-state prediction with the thresholds of 0%, 5%, 9%, 16%, and 25%, respectively.



Linear Combination Fusion 1, column mean normalization and then adding, gave the overall prediction accuracy 61.30% for the three-state prediction with respect to thresholds: 9% and 36%; and 71.38%, 75.35%, 77.18%, 77.90%, 76.60% for the two-state prediction with the thresholds of 0%, 5%, 9%, 16%, and 25%, respectively. Obviously, the two-state prediction accuracies of thresholds 0%, 5%, and 9% are lower than those by QuickRBF approach.

Linear Combination Fusion 2, row-wise normalization and then adding, led to the overall prediction accuracy 61.06% for the three-state prediction with respect to thresholds: 9% and 36%; and 87.97%, 81.28%, 79.77%, 78.23%, 76.96% for the two-state prediction with the thresholds of 0%, 5%, 9%, 16%, and 25%, respectively.

Reliability Index Fusion, maximal relative index, gave the overall prediction accuracy 61.09% for the three-state prediction with respect to thresholds: 9% and 36%; and 87.97%, 81.24%, 79.73%, 78.16%, 77.01% for the two-state prediction with the thresholds of 0%, 5%, 9%, 16%, and 25%, respectively. The prediction accuracies of QuickRBF and the fusion methods on RS126 data set are listed in Table 4.2.

Table 4.2. RSA classification accuracies by QuickRBF and the fusion methods.

QuickRBF

accuracy: %						
thresholds dataset	3-state (9% & 36%)	2-state (0%)	2-state (5%)	2-state (9%)	2-state (16%)	2-state (25%)
Fold_A	60.71	86.51	79.40	78.30	77.64	76.38
Fold_B	60.67	89.15	81.35	79.60	78.06	76.38
Fold_C	61.81	87.59	80.92	80.25	78.90	77.17
Fold_D	59.16	88.01	81.29	78.46	76.96	75.51
Fold_E	60.85	88.51	82.76	80.53	78.65	76.07
Fold_F	58.90	88.74	81.13	78.02	76.00	74.27
Fold_G	60.23	85.85	78.35	78.35	77.00	77.26
Average	60.36	87.76	81.15	79.06	77.64	76.17

Linear Combination Fusion 1

of accuracy: %						
thresholds dataset	3-state (9% & 36%)	2-state (0%)	2-state (5%)	2-state (9%)	2-state (16%)	2-state (25%)
Fold_A	62.14	72.08	75.91	77.39	78.46	77.15
Fold_B	61.16	71.15	76.03	77.25	78.22	77.11
Fold_C	63.97	72.89	76.73	78.72	79.27	77.83
Fold_D	59.91	70.47	74.00	75.22	76.91	75.95
Fold_E	62.21	70.74	75.72	78.13	78.61	76.77
Fold_F	59.22	71.02	75.57	76.25	75.60	75.10
Fold_G	59.89	70.99	76.59	77.56	77.79	75.72
Average	61.30	71.38	75.75	77.18	77.90	76.60

Linear Combination Fusion 2

accuracy: %						
thresholds dataset	3-state (9% & 36%)	2-state (0%)	2-state (5%)	2-state (9%)	2-state (16%)	2-state (25%)
Fold_A	61.48	86.53	80.12	78.95	78.51	77.88
Fold_B	61.13	89.45	81.48	80.59	78.60	77.11
Fold_C	63.66	87.73	82.48	81.06	79.22	78.44
Fold_D	60.09	88.30	81.03	79.01	77.71	76.16
Fold_E	61.83	88.61	83.32	80.91	78.75	76.63
Fold_F	59.18	89.07	81.20	79.00	76.54	75.17
Fold_G	59.48	86.27	79.59	78.91	77.90	76.74
Average	61.06	87.97	81.28	79.77	78.23	76.96



accuracy: %						
thresholds dataset	3-state (9% & 36%)	2-state (0%)	2-state (5%)	2-state (9%)	2-state (16%)	2-state (25%)
Fold_A	61.58	86.53	80.17	78.95	78.27	78.04
Fold_B	60.95	89.42	81.40	80.54	78.68	77.06
Fold_C	63.22	87.64	82.45	80.95	79.10	78.55
Fold_D	60.15	88.32	80.88	78.88	77.66	76.28
Fold_E	62.07	88.58	83.14	80.88	78.75	76.73
Fold_F	59.58	89.07	81.05	79.11	76.40	75.03
Fold_G	59.63	86.27	79.85	78.95	77.94	76.70
Average	61.09	87.96	81.24	79.73	78.16	77.01

In summary, the accuracies of six methods are tabulated in Table 4.3.

Table 4.3. Comparison of performance of the six approaches in RSA prediction on the RS126 data set with PSSMs generated by PSI-BLAST.

Comparison of six methods

		accuracy: %				
method \ thresholds	3-state (9% & 36%)	2-state (0%)	2-state (5%)	2-state (9%)	2-state (16%)	2-state (25%)
Fuzzy <i>k</i> -NN	58.14	87.93	79.18	77.59	75.35	73.49
Modified fuzzy <i>k</i> -NN	58.57	87.93	79.84	77.76	76.34	75.26
QuickRBF	60.36	87.76	81.15	79.06	77.64	76.17
Linear Combination Fusion 1	61.30	71.38	75.75	77.18	77.90	76.60
Linear Combination Fusion 2	61.06	87.97	81.28	79.77	78.23	76.96
Reliability Index	61.09	87.96	81.24	79.73	78.16	77.01



4.3 Matthew's Correlation Coefficients of Modified Fuzzy K -NN Approach

Another measure used to evaluate the performance of prediction methods is the Matthew's Correlation Coefficient (MCC). It can be calculated from an accuracy table A by the following equations:

A_{ij} = number of residues predicted to be in type j and observed to be in type i ,

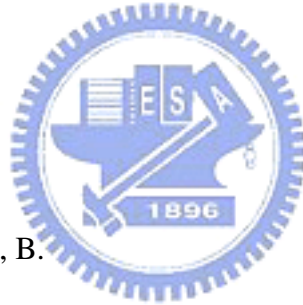
$$MCC_i = \frac{p_i n_i - u_i o_i}{\sqrt{(p_i + u_i)(p_i + o_i)(n_i + u_i)(n_i + o_i)}},$$

$$p_i = A_{ii},$$

$$n_i = \sum_{j \neq i} \sum_{k \neq i} A_{jk},$$

$$o_i = \sum_{j \neq i} A_{ji},$$

$$u_i = \sum_{j \neq i} A_{ij}, \text{ for } i = E, I, B.$$



Also, p_i , n_i , o_i and u_i are the number of true positives, true negatives, false positives and false negatives for class i , respectively. The MCCs have the same value for the two classes in the case of the two-state prediction, i.e. $MCC_E = MCC_B$.

First, the accuracy tables A of modified fuzzy k -NN approach on each fold and the RS126 data set is shown in Table 4.3. Then, the MCCs of five methods on the RS126 data set is shown in Table 4.4. In a similar trend as Table 4.2, MCC's of Linear Combination Fusion 2 usually perform well, although not always the best, in comparison to other approaches.

Table 4.4. The accuracy tables A of modified fuzzy k -NN on each fold and RS126.

3-state (9%; 36%)

	A_{EE}	A_{II}	A_{BB}	A_{EI}	A_{EB}	A_{IE}	A_{IB}	A_{BE}	A_{BI}
Fold_A	1073	525	931	384	67	432	263	253	348
Fold_B	1068	419	699	313	60	442	229	186	289
Fold_C	847	467	778	356	66	314	245	134	257
Fold_D	1003	491	741	336	97	439	266	182	299
Fold_E	737	376	605	288	56	276	177	135	221
Fold_F	755	320	503	229	52	356	159	168	229
Fold_G	724	435	827	375	172	367	402	167	382
RS126	6207	3033	5084	2281	570	2626	1741	1225	2025

2-state (25%)

	A_{EE}	A_{BB}	A_{EB}	A_{BE}
Fold_A	1515	1761	468	532
Fold_B	1456	1341	388	520
Fold_C	1209	1461	420	374
Fold_D	1389	1494	453	518
Fold_E	1021	1132	379	339
Fold_F	1031	979	335	426
Fold_G	880	1098	354	333
RS126	8501	9266	2797	3042

2-state (16%)

	A_{EE}	A_{BB}	A_{EB}	A_{BE}
Fold_A	1942	1362	431	541
Fold_B	1832	1022	399	452
Fold_C	1563	1129	414	358
Fold_D	1790	1116	471	477
Fold_E	1327	861	352	331
Fold_F	1330	729	300	412
Fold_G	1147	871	320	327
RS126	10931	7090	2687	2898

Table 4.4. (continued)

2-state (9%)

	A_{EE}	A_{BB}	A_{EB}	A_{BE}
Fold_A	2294	1014	450	518
Fold_B	2151	758	380	416
Fold_C	1877	844	418	325
Fold_D	2164	811	468	411
Fold_E	1580	659	330	302
Fold_F	1595	554	276	346
Fold_G	1375	675	313	302
RS126	13036	5315	2635	2620

2-state (5%)

	A_{EE}	A_{BB}	A_{EB}	A_{BE}
Fold_A	2588	774	440	474
Fold_B	2369	589	400	347
Fold_C	2104	671	398	291
Fold_D	2412	605	454	383
Fold_E	1771	523	327	250
Fold_F	1782	422	266	301
Fold_G	1558	523	315	269
RS126	14584	4107	2600	2315

2-state (0%)

	A_{EE}	A_{BB}	A_{EB}	A_{BE}
Fold_A	3612	81	40	543
Fold_B	3263	32	34	376
Fold_C	2961	85	49	369
Fold_D	3355	51	23	425
Fold_E	2499	49	37	286
Fold_F	2425	41	26	279
Fold_G	2261	41	17	346
RS126	20376	380	226	2624

Table 4.5. Matthew's Correlation Coefficients of the five approaches on RS126.

3-state (9%; 36%)			
method \ MCC	MCC _E	MCC _I	MCC _B
Fuzzy <i>k</i> -NN	0.439	0.133	0.499
Modified fuzzy <i>k</i> -NN	0.432	0.163	0.485
QuickRBF	0.478	0.138	0.529
Linear Combination Fusion 1	0.491	0.176	0.533
Linear Combination Fusion 2	0.487	0.171	0.533

2-state (25%)	
method \ MCC	MCC _E = MCC _B
Fuzzy <i>k</i> -NN	0.492
Modified fuzzy <i>k</i> -NN	0.505
QuickRBF	0.530
Linear Combination Fusion 1	0.539
Linear Combination Fusion 2	0.543

2-state (16%)	
method \ MCC	MCC _E = MCC _B
Fuzzy <i>k</i> -NN	0.492
Modified fuzzy <i>k</i> -NN	0.514
QuickRBF	0.538
Linear Combination Fusion 1	0.549
Linear Combination Fusion 2	0.550

Table 4.5. (continued)

2-state (9%)	
method \ MCC	$MCC_E = MCC_B$
Fuzzy k -NN	0.470
Modified fuzzy k -NN	0.501
QuickRBF	0.510
Linear Combination Fusion 1	0.529
Linear Combination Fusion 2	0.532

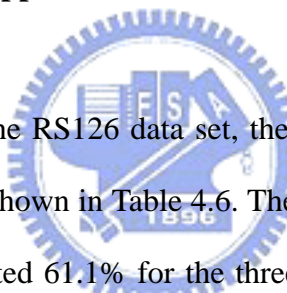
2-state (5%)	
method \ MCC	$MCC_E = MCC_B$
Fuzzy k -NN	0.439
Modified fuzzy k -NN	0.482
QuickRBF	0.472
Linear Combination Fusion 1	0.500
Linear Combination Fusion 2	0.501

2-state (0%)	
method \ MCC	$MCC_E = MCC_B$
Fuzzy k -NN	0.243
Modified fuzzy k -NN	0.243
QuickRBF	0.219
Linear Combination Fusion 1	0.368
Linear Combination Fusion 2	0.237

After comparing the prediction results of three fusion methods, we can find that the two-state prediction accuracies of Linear Combination Fusion 1 are lower than

those by the other two fusion methods with thresholds 0%, 5%, and 9%. In our opinion, this maybe due to the normalization step used in Linear Combination Fusion 1. If the chosen threshold Th is close to zero in two-state modified fuzzy k -NN prediction, then the $u_B(x^{(j)})$ values for most residues will be very low. And hence, m_B will be close to zero, as shown in Fig. 3.6. After dividing all $u_B(x^{(j)})$ values by m_B , the new $u_B(x^{(j)})$ values will become very large, and this will cause the faulty classification after the fusion.

4.4 Comparison with other Approaches



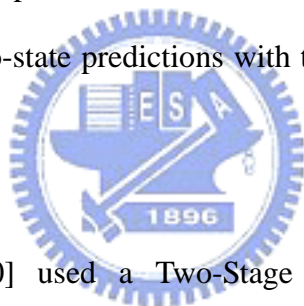
For RSA prediction on the RS126 data set, the performance comparison of our approach to other methods is shown in Table 4.6. The Linear Common Fusion 2 is the best in most cases. It is reported 61.1% for the three-state prediction with respect to thresholds of 9% and 36%; 88.0%, 81.3%, 79.8%, 78.2%, and 77.0% for the two-state predictions with the thresholds of 0%, 5%, 9%, 16%, and 25%, respectively.

Sim *et al.* has led to slightly better prediction accuracies than other methods by fuzzy k -nearest neighbor method using PSI-BLAST profiles on the RS126 data set produced by them. In [31], they reported 63.8% for the three-state prediction with respect to thresholds of 9% and 36%; 87.2%, 82.2%, 79.0%, and 78.3% for the two-state predictions with the thresholds of 0%, 5%, 16%, and 25%, respectively. Using the same method and best parameter settings on our produced RS126 data set, we just obtained 58.1% for the three-state prediction; 87.9%, 79.2%, 75.4%, and 73.5% for the two-state predictions with the thresholds of 0%, 5%, 16%, and 25%,

respectively.

PHDacc [1] used a neural network method using evolutionary profiles of amino acid substitutions derived from multiple sequence alignments, and reported 57.5% for the three-state prediction with respect to thresholds of 9% and 36%; 86.0%, 74.6%, and 75.0% for the two-state predictions with the thresholds of 0%, 9%, and 16%, respectively.

SVMpsi [23] was based on a support vector machine using the position-specific scoring matrix generated from PSI-BLAST, and reported 59.6% accuracy for the three-state prediction with respect to thresholds of 9% and 36%; 86.2%, 79.8%, 77.8%, and 76.8% for the two-state predictions with the thresholds of 0%, 5%, 16%, and 25%, respectively.



Two-Stage SVMpsi [30] used a Two-Stage SVMpsi approach using the position-specific scoring matrix generated from PSI-BLAST. It is reported 90.2%, 83.5%, 81.3%, and 79.4% for the two-state predictions with the thresholds of 0%, 5%, 9%, and 16%, respectively. These prediction accuracies are obtained from their published results.

Table 4.6. Comparison of performance of modified fuzzy k -NN approach with other methods in RSA prediction on the RS126 data set with PSSMs generated by PSI-BLAST.

		accuracy: %				
method \ thresholds	3-state (9% ; 36%)	2-state (0%)	2-state (5%)	2-state (9%)	2-state (16%)	2-state (25%)
Linear Combimation Fusion 2	61.1	88.0	81.3	79.8	78.2	77.0
Fuzzy k -NN (on our dataset)	58.1	87.9	79.2	77.6	75.4	73.5
Modified fuzzy k -NN (on our dataset)	58.6	87.9	79.8	77.8	76.3	75.3
Fuzzy k -NN (on their dataset [31])	63.8	87.2	82.2	—	79.0	78.3
PHDacc	57.5	86.0	—	74.6	75.0	—
SVMpsi	59.6	86.2	79.8	—	77.8	76.8
Two-Stage SVMpsi	—	90.2	83.5	81.3	79.4	—

Fuzzy k -NN (Sim, Kim and Lee, 2005) used fuzzy k -nearest neighbor method [31].

PHDacc (Rost and Sander, 1994) used neural networks [1].

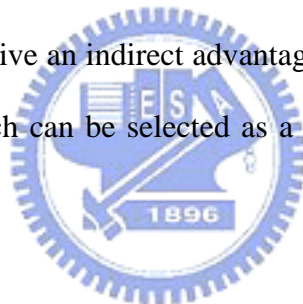
SVMpsi (Kim and Park, 2004) was based on support vector machine [23].

Two-Stage SVMpsi (Nguyen and Rajapakse, 2005) used a two-stage SVM approach [30].

Chapter 5. Conclusion and Discussion

Using PSI-BLAST profiles as feature vectors in this study, we have proposed six approaches, which are Fuzzy K -NN, Modified Fuzzy K -NN, QuickRBF, Linear Combination Fusion 1, Linear Combination Fusion 2, and Reliability Index Fusion, to predict relative solvent accessibility of RS126 data set.

In the future study, we can apply dimensionality reduction technique [54] to reflect the structure existent in the data set. Then we can find more reliable distance metrics faithfully from PSSM table to improve the classification accuracy of our fuzzy k -NN method. Besides, we can apply our method on a larger data set, like CB513. Data set growth can give an indirect advantage to our method. And our better modified fuzzy k -NN approach can be selected as a promising approach for various protein applications.



References

- [1] B. Rost and C. Sander, "Conservation and prediction of solvent accessibility in protein families," *Proteins*, vol. 20, pp. 216–226, 1994.
- [2] M.J. Thompson and R.A. Goldstein, "Predicting solvent accessibility: higher accuracy using Bayesian statistics and optimized residue substitution classes," *Proteins*, vol. 25, pp. 38-47, 1996.
- [3] K. Ginalski and L. Rychlewski, "Protein structure prediction of CASP5 comparative modeling and fold recognition targets using consensus alignment approach and 3D assessment," *Proteins*, vol. 53, pp. 410–417, 2003.
- [4] B. John and A. Sali, "Detection of homologous proteins by an intermediate sequence search." *Protein Sci.*, vol. 13, pp. 54–62, 2004.
- [5] J. Moult *et al.*, "Critical assessment of methods of protein structure prediction (CASP)-round V." *Proteins*, vol. 53, pp. 334–339, 2003.
- [6] C. Sander and R. Schneider, "Database of homology-derived protein structures and the structural meaning of sequence alignment." *Proteins*, vol. 9, pp. 56–68, 1991.
- [7] J.A. Cuff and G.J. Barton, "Application of multiple sequence alignment profiles to improve protein secondary structure prediction." *Proteins*, vol. 40, pp. 502–511, 2000.
- [8] D. Frishman and P. Argos, "Seventy-five percent accuracy in protein secondary structure prediction." *Proteins*, vol. 27, pp. 329–335, 1997.
- [9] D.T. Jones, "Protein secondary structure prediction based on position specific scoring matrices." *J. Mol. Biol.*, vol. 292, pp. 195–202, 1999.

- [10] D. Przybylski and B. Rost, "Alignments grow, secondary structure prediction improves." *Proteins*, vol. 46, pp. 197–205, 2002.
- [11] G. Wohlfahrt *et al.*, "Positioning of anchor groups in protein loop prediction: the importance of solvent accessibility and secondary structure elements." *Proteins*, vol. 47, pp. 370–378, 2002.
- [12] E. Eyal *et al.*, "Importance of solvent accessibility and contact surfaces in modeling side-chain conformations in proteins." *J. Comput. Chem.*, vol. 25, pp. 712–724, 2004.
- [13] S.J. Russell *et al.*, "Stability of cyclic beta-hairpins: asymmetric contributions from side chains of a hydrogen-bonded cross-strand residue pair." *J. Am. Chem. Soc.*, vol. 125, pp. 388–395, 2003.
- [14] M. Totrov, "Accurate and efficient generalized born model based on solvent accessibility: derivation and application for LogP octanol/water prediction and flexiblepeptide docking." *J. Comput. Chem.*, vol. 25, pp. 609–619, 2004.
- [15] B. Rost *et al.*, "Protein fold recognition by prediction-based threading." *J. Mol. Biol.*, vol. 270, pp. 471–480, 1997.
- [16] G. Gianese *et al.*, "Improvement in prediction of solvent accessibility by probability profiles." *Protein Eng.*, vol. 16, pp. 987–992, 2003.
- [17] J. Pei and N.V. Grishin, "Combining evolutionary and structural information for local protein structure prediction." *Proteins*, vol. 56, pp.782–794, 2004.
- [18] J.M. Keller *et al.*, "A fuzzy k-nearest neighbor algorithm." *IEE Trans. Syst. Man Cybern.*, vol. 15, pp. 580–585, 1985.
- [19] R. Adamczak *et al.*, "Accurate prediction of solvent accessibility using neural networks-based regression." *Proteins*, vol. 56, pp. 753–767, 2004.
- [20] S. Ahmad and M.M. Gromiha, "NETASA: neural network based prediction of solvent accessibility." *Bioinformatics*, vol. 18, pp. 819–824, 2002.

- [21] S. Ahmad *et al.*, “Real value prediction of solvent accessibility from amino acid sequence.” *Proteins*, vol. 50, pp. 629–635, 2003.
- [22] G. Pollastri *et al.*, “Prediction of coordination number and relative solvent accessibility in proteins.” *Proteins*, vol. 47, pp. 142–153, 2002.
- [23] H. Kim and H. Park, “Prediction of protein relative solvent accessibility with support vector machines and long-range interaction 3D local descriptor.” *Proteins*, vol. 54, pp. 557–562, 2004.
- [24] Z. Yuan *et al.*, “Prediction of protein solvent accessibility using support vector machines.” *Proteins*, vol. 48, pp. 566–570, 2002.
- [25] Z. Yuan and B. Huang, “Prediction of protein accessible surface areas by support vector regression.” *Proteins*, vol. 57, pp.558–564, 2004.
- [26] A.A. Salamov and V.V. Solovyev, “Protein secondary structure prediction using local alignments.” *J. Mol. Biol.*, vol. 268, pp. 31–36, 1997.
- [27] S. Salzberg and S. Cost, “Predicting protein secondary structure with a nearest-neighbor algorithm.” *J. Mol. Biol.*, vol. 227, pp. 371–374, 1992.
- [28] T.M. Yi and E.S. Lander, “Protein secondary structure prediction using nearest-neighbor methods.” *J. Mol. Biol.*, vol. 232, pp. 1117–1129, 1993.
- [29] S.F. Altschul *et al.*, “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.” *Nucleic Acids Res.*, vol. 25, pp. 3389–3402, 1997.
- [30] M.N. Nguyen and J.C. Rajapakse, “Prediction of protein relative solvent accessibility with a two-stage SVM approach,” *Proteins*, vol. 59, pp. 30–37, 2005.
- [31] J. Sim, S.Y. Kim, and J. Lee, “Prediction of protein solvent accessibility using fuzzy k-nearest neighbor method,” *Bioinformatics*, vol. 21, pp. 2844–2849, 2005.
- [32] B.K. Lee and F.M. Richards, “The interpretation of protein structures: estimation

- of static accessibility,” *J. Mol. Biol.*, vol. 55, pp. 379–400, 1971.
- [33] C. Chothia, “The nature of the accessible and buried surfaces in proteins,” *J. Mol. Biol.*, vol. 105, pp. 1–12, 1976.
- [34] M.L. Connolly, “Solvent-accessible surfaces of proteins and nucleic acids,” *Science*, vol. 221, pp. 709–713, 1983.
- [35] W. Kabsch and C. Sander, “Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features,” *Biopolymers*, vol. 22, pp. 2577–2637, 1983.
- [36] J. Janin, “Surface and inside volumes in globular proteins,” *Nature*, vol. 277, pp. 491–492, 1979.
- [37] G.D. Rose, A.R. Geselowitz, G.J. Lesser, R.H. Lee, and M.H. Zehfus, “Hydrophobicity of amino acid residues in globular proteins,” *Science*, vol. 229, pp. 834–838, 1985.
- [38] C. Sander, M. Scharf, and R. Schneider, “Design of protein structures in: Protein Engineering: A practical Approach,” *Oxford University press*, pp. 89–115, 1992.
- [39] H. Hirakawa and S. Kuhara, “Prediction of Hydrophobic Cores of Proteins Using Wavelet Analysis,” *Genome Inform Ser Workshop Genome Information*, vol. 8, pp. 61–70, 1997.
- [40] B. Rost and C. Sander, “Combining evolutionary information and neural networks to predict protein secondary structure,” *Proteins*, vol. 19, pp. 55–72, 1994.
- [41] G.J. Barton, “Protein secondary structure prediction,” *Curr. Opin. Struct. Biol.*, vol. 5, pp. 372–376, 1995.
- [42] P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, and G. Soda, “Exploiting the past and the future in protein secondary structure prediction,” *Bioinformatics*, vol. 15, pp. 937–946, 1999.

- [43] S. Henikoff and J.G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc. Natl. Acad. Sci.*, vol. 89, pp. 10915–10919, 1992.
- [44] J.C. Bezdek *et al.*, "Review of MR image segmentation techniques using pattern recognition." *Med. Phys.*, vol. 20, pp. 1033–1048, 1993.
- [45] D. Cabello *et al.*, "Fuzzy k-nearest neighbor classifiers for ventricular arrhythmia detection." *Int. J. Biomed. Comput.*, vol. 27, pp. 77–93, 1991.
- [46] Y. Huang and Y. Li, "Prediction of protein subcellular locations using fuzzy k-NN method." *Bioinformatics*, vol. 20, pp. 21–28, 2004.
- [47] K. Leszczynski *et al.*, "Application of a fuzzy pattern classifier to decision making in portal verification of radiotherapy." *Phys. Med. Biol.*, vol. 44, pp. 253–269, 1999.
- [48] H. Seker *et al.*, "A fuzzy logic based-method for prognostic decision making in breast and prostate cancers." *IEEE Trans. Inf. Technol. Biomed.*, vol. 7, pp. 114–122, 2003.
- [49] B. Sokolowska *et al.*, "A fuzzy-classifier system to distinguish respiratory patterns evolving after diaphragm paralysis in the cat." *Jpn. J. Physiol.*, vol. 53, pp. 301–307, 2003.
- [50] S. Hua and Z. Sun, "A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach." *J. Mol. Biol.*, vol. 308, pp. 397–407, 2001.
- [51] B. Rost and C. Sander, "Prediction of protein secondary structure at better than 70% accuracy." *J. Mol. Biol.*, vol. 232, pp. 584–599, 1993.
- [52] Y.Y. Ou, "QuickRBF is an Efficient Construction of Radial Basis Function Networks with the Cholesky Decomposition." <http://csie.org/~yien/quickrbf/>.
- [53] S.K. Riis and A. Krogh, "Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignment." *J. Comput.*

Biol., vol.3, pp. 163–183, 1996.

- [54] S.T. Roweis *et al.*, “Nonlinear Dimensionality Reduction by Locally Linear Embedding.” *Science*, vol. 290, pp. 2323, 2000.

