11-27-2007

Computational Intelligence Based Classifier Fusion Models for Biomedical Classification Applications

Xiujuan Chen

COMPUTATIONAL INTELLIGENCE BASED CLASSIFIER FUSION MODELS FOR BIOMEDICAL CLASSIFICATION APPLICATIONS

by

XIUJUAN CHEN

Under the Direction of Robert Harrison and Yan-Qing Zhang

ABSTRACT

The generalization abilities of machine learning algorithms often depend on the algorithms' initialization, parameter settings, training sets, or feature selections. For instance, SVM classifier performance largely relies on whether the selected kernel functions are suitable for real application data. To enhance the performance of individual classifiers, this dissertation proposes classifier fusion models using computational intelligence knowledge to combine different classifiers. The first fusion model called T1FFSVM combines multiple SVM classifiers through constructing a fuzzy logic system. T1FFSVM can be improved by tuning the fuzzy membership functions of linguistic variables using genetic algorithms. The improved model is called GFFSVM. To better handle uncertainties existing in fuzzy MFs and in classification data, T1FFSVM can also be improved by applying type-2 fuzzy logic to construct a type-2 fuzzy classifier fusion model (T2FFSVM). T1FFSVM, GFFSVM, and T2FFSVM use accuracy as a classifier performance measure. AUC (the area under an ROC curve) is proved to be a better classifier performance metric. As a comparison study, AUC-based classifier fusion models are also proposed in the dissertation. The experiments on biomedical datasets demonstrate promising performance of the proposed classifier fusion models comparing

with the individual composing classifiers. The proposed classifier fusion models also demonstrate better performance than many existing classifier fusion methods.

The dissertation also studies one interesting phenomena in biology domain using machine learning and classifier fusion methods. That is, how protein structures and sequences are related each other. The experiments show that protein segments with similar structures also share similar sequences, which add new insights into the existing knowledge on the relation between protein sequences and structures: similar sequences share high structure similarity, but similar structures may not share high sequence similarity.

INDEX WORDS

Machine Learning, Bioinformatics, DNA Microarray, Protein Structures and Sequences, Classifier Fusion, Computational Intelligence, Support Vector Machines, Fuzzy Logic, Type-2 Fuzzy Logic, Genetic Algorithms, Classifier Performance Measure, Receiver Operating Characteristics

COMPUTATIONAL INTELLIGENCE BASED CLASSIFIER FUSION MODELS FOR BIOMEDICAL CLASSIFICATION APPLICATIONS

by

XIUJUAN CHEN

A Dissertation Submitted in Partial Fulfillment of Requirements for the Degree of

Doctor of Philosophy

In the College of Arts and Sciences

Georgia Stage University

2007

COMPUTATIONAL INTELLIGENCE BASED CLASSIFIER FUSION MODELS FOR BIOMEDICAL CLASSIFICATION APPLICATIONS

by

XIUJUAN CHEN

Major Professor:	Robert Harrison
	Yan-Qing Zhang
Committee:	Rajshekhar Sunderraman
	Yichuan Zhao

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

December 2007

Acknowledgments

I would like to thank my advisors, first and foremost, Dr. Robert Harrison and Dr. Yan-Qing Zhang, for their valuable and generous guidance and endless support throughout my Ph.D. career and during the process of my dissertation. I am also extremely grateful to the members of my committee, Dr. Rajshekhar Sunderraman, and Dr. Yichuan Zhao for their well-appreciated support and assistance during my graduate study. The dissertation would not have been possible without their helps.

Last, but not least, I would like to thank my family for their strong, patient, and persistent encouragement, understanding, and support during my educational pursuits.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Adaptive Boosting	AdaBoost
Area under the Curve of an ROC	AUC
Center of Sets	COS
Empirical Risk Minimization	ERM
False Positive	FP
False Positive Rate	FPR
False Negative	FN
Footprint of Uncertainty	FOU
Fuzzy Logic System	FLS
Genetic Algorithm	GA
Genetic Fuzzy SVM Classifier Fusion	GFFSVM
Genetic Fuzzy System	GFS
Homology-Derived Secondary Structure of Proteins	HSSP
Membership Function	MF
Protein Sequence Culling Server	PISCES
Receiver Operating Characteristics	ROC
Single Nucleotide Polymorphism	SNP
Structural Risk Minimization	SRM
Support Vector Machine	SVM
True Positive	TP
True Positive Rate	TPR
True Negative	TN

Type-1 Fuzzy SVM Classifier Fusion	T1FFSVM
Type-2 Fuzzy SVM Classifier Fusion	T2FFSVM

Chapter 1 Introduction

Over the past decade, we have witnessed the dramatic progresses in biotechnologies, including genome sequencing technology and DNA chip technology. As a result, many complex biological and medical datasets have increased in volume exponentially, e.g., genome sequences of many species, microarray gene expression data of different cells, protein sequences and structures, single nucleotide polymorphisms (SNPs) in the human genome, and etc. Accordingly, the bulk of research efforts have been shifted to biomedical data analysis, discovery of patterns, functions, and regularities of genes and proteins, and providing assistance and supports for disease diagnosis and biomedical and evolutionary research.

Machine learning and data mining methods have largely been used in biomedical data analysis recently. The basis of machine learning is to program computers using example data or past experience to detect patterns and regularities. It includes various applications such as association, supervised learning, regression, and unsupervised learning. Classification is supervised learning which is capable of predicting class labels of unseen data examples based on existing data. The basic idea of classification can be illustrated in Figure 1.1. Given a set of empirical training data with class labels known: $\{(x_i, y_i)\}$ (i=1…m), where x_i is a data example represented by its feature vector in an input space $\Re$, and y_i is the corresponding class label indicating which class (cancer or normal tissue, for instance) x_i belongs to. A classification algorithm is applied onto the training data to learn patterns. After the training process is finished, the system establishes a mapping relation (called classifier or model) between the input vector and the output label. When a

set of unseen testing data is plugged into the classifier, the classifier will be able to predict whether the testing data examples belong to cancer class or normal class.

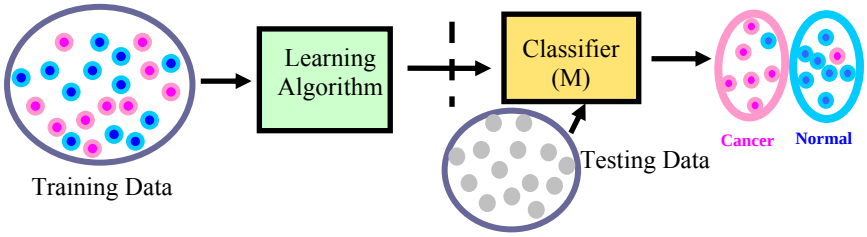


Figure 1.1 The basic idea of classification

In this dissertation, binary classification problems are focused. Multi-class classification problems can be reduced to binary classification problems through several strategies, among which one-against-the-rest strategy (Weston and Watkins, 1998) and pairwise strategy (Hastie and Tibshirani, 1996) are often used. The one-against-the rest schema breaks the original problem into k binary classification problems, where k is the number of classes. While the pairwise strategy creates $\frac{1}{2}k(k-1)$ classifiers and the final decision is determined by a voting scheme such as majority voting.

Classification algorithms have been used widely and successfully in biomedical data analysis. Many experimental results have been reported in the literature by applying classification methods to extract biological functions of genes and proteins, explore the relations between genes or protein patterns and diseases, and help disease diagnosis. For instance, Yao has proposed two neural network based approaches for breast cancer diagnosis (Yao and Liu, 1999); Roth has applied a Bayesian approach to help diagnose Leukemia and Lymphoma (Roth and Lange, 2004); Vlahou has classified ovarian cancer data using decision tree (Vlahou *et al.*, 2003); Ramaswamy has classified datasets

containing multiple tumors using support vector machines (SVMs) with the removal of irrelevant features (Ramaswamy *et al*., 2001).

1.1 Problem: How to Select a Kernel Function in SVM Classification?

When we solve a classification problem using SVM, we all know that we need to select one kernel function. This kernel function defines the feature space where data examples are classified and the separating hyperplane is determined. Kernel functions can directly affect the generalization ability of SVM classification. How to select an appropriate kernel function which fits a particular dataset best is one of the essential issues in SVM classification. One obvious method is brute-force method. That is, exhaustively trying many different kernels and selecting the one which works best. This approach could be extremely time-consuming if the size or the dimension of training data is huge. Are there any other ways through which don't need to do the complete search but can still receive good classification performance? One possible less time-consuming method is to choose several SVM classifiers with different kernels to classify data examples, and then combine the classification results from the different classifiers in a certain way and generate a composite classifier. The resulting classifier is probably expected to have a better performance than each of its composing individual classifiers if the combination methods are effective. In both theory and practice, this combining method seems to outperform the brute force method in terms of both time complexity and classification performance. Indeed, the advantage of combination and complementary provides an effective way to enhance classifier performance (Kittler *et al*., 1998).

1.2 Drawbacks of Single Classifiers

When solving a classification problem, we would like to choose the best classifier. However, the determination of the best classifier is a time-consuming process. This is because a classification algorithm may form different decision functions based on different initialization, different parameter settings, different training sets, or different feature selections. For instance, different initialization may result in different neural network classifiers. Different parameter choices can also result in different classifiers, such as kernel functions mentioned in Section 1.1 and regularization parameter in the SVM algorithm, and the number of neighbors in the *k*-NN algorithm. To obtain the best classifier, the brute-force method as mentioned earlier usually has to be applied to try all the possible initializations, parameters, training sets, and feature selections, which is obviously computationally expensive.

Even if the best classifier is identified, it might not necessarily be an "ideal" choice. A classification algorithm is designed internally based on some classifier performance measure criteria, e.g., training accuracy or complexity of the classifier, and the "best" classifier is selected according to the criteria. Maybe more than one classifier has same training accuracy or meets the criteria. However, the learning algorithm simply selects one classifier and discards others. The discarded classifiers may correctly classify some data examples which are misclassified by the selected best classifier. Potentially valuable information might be lost by discarding the classification results from less-successful classifiers.

1.3 Idea of Combining Multiple Classifiers

We know different classifiers have different knowledge regarding to a problem and may provide complementary information about data examples to be classified. Even poor classifiers may classify some data examples correctly which might be misclassified by some good classifiers. Combining different classifiers in an effective way can achieve better classification results than that from individual classifiers including the best classifier. The reason is that data examples misclassified by different classifiers would not necessarily overlap, which leaves the room for the classifier complementariness (Kittler *et al*., 1998). Figure 1.2 illustrates the idea of combining multiple classifiers. Data examples are classified by multiple classifier hypotheses, h_1, h_2, …, h_L first. The classification results are then combined altogether in a fusion model. How to construct a good classifier fusion model is one of the most active research areas in supervised leaning research. In this dissertation, I will mainly discover ensemble models particularly for combining SVM classifiers by considering their special characteristics.

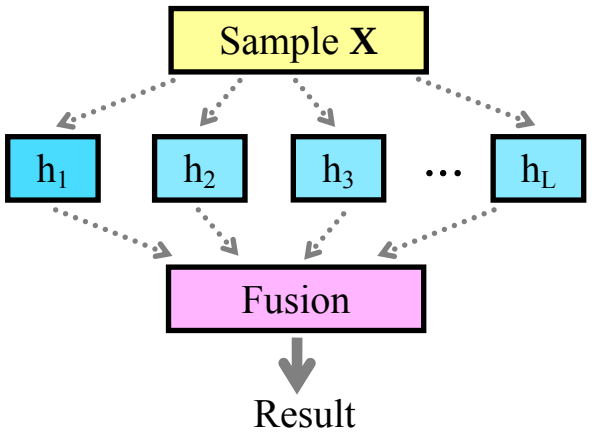


Figure 1.2 Combining multiple classifiers

1.4 Classifier Performance Evaluation

Typically, accuracy (or error, which is 1 minus the accuracy) is the standard criterion to evaluate a classifier performance (Qin, 2005; Ling *et al.*, 2003). In many scenarios, however, accuracy is not enough. For some applications, researchers are interested in more than mere positive / negative classification results. Instead, ranking of examples is more likely to be cared for (Ling *et al.*, 2003; Huang and Ling, 2005). In addition, using accuracy assumes the misclassification costs are equal for all data examples and it would not be necessarily realistic for some real-world applications such as medical diagnosis (Qin, 2005). Accuracy is also not much meaningful when class distribution is skewed or unbalanced since a classifier can simply classify all data examples to the dominant class and still receive a high accuracy (Qin, 2005; Fawcett, 2003). Recently, Receiver Operating Characteristics (ROC) analysis is increasingly recognized in the machine learning and data mining research community and the area under the curve (AUC) of an ROC has been shown to be statistically consistent and more discriminating than accuracy empirically and theoretically (Qin, 2005; Ling *et al.*, 2003; Huang and Ling, 2005).

1.5 Organizations and Contributions

The rapid development of biotechnologies such as genome sequencing technology and microarray DNA chip technology generates large scale biomedical data. Chapter 2 briefly reviews the biological and bioinformatics backgrounds including the concepts of genes, proteins, and microarray technology. The chapter also addresses the current situations by applying computational intelligence algorithms and methods to solve biological problems. SVMs are a powerful and effective supervised learning technique and have been successfully applied in biological and medical domain to discover patterns

and regularities. Chapter 3 will review the basis of SVM theory. Chapter 3 also reviews the existing classifier combination methods which are effective approaches to enhance the performance of weak classifiers. Chapter 4 addresses fuzzy logic concept and fuzzy logic system. Then a fuzzy classifier fusion model (T1FFSVM) to combine SVM classifiers is proposed in the chapter and experiments on biomedical data have been performed and analyzed. The fuzzy MFs in the fuzzy classifier fusion model in Chapter 4 are defined intuitively based on the classification experience. The shapes or the positions of MFs may not be optimal. Genetic Algorithms (GA) provide robust search and learning capabilities in complex space and are ideal for tuning the optimal MFs and discovering optimal FLS accordingly. In Chapter 5, GAs and genetic fuzzy systems (GFS) are first introduced. Then the fuzzy classifier fusion model described in Chapter 4 is optimized using GAs. The genetic fuzzy classifier model (GFFSVM) is tested on biomedical data and compared with the fuzzy classifier model constructed in Chapter 4. Classification accuracy is used to estimate the classifier performance in classifier fusion models defined in Chapter 4 and Chapter 5. However, accuracy is not an ideal performance measure in many cases. Instead, AUC is proven to be a better measure than accuracy or error. Chapter 6 introduces the concepts of confusion matrix, ROC, and AUC of a classifier. And then an AUC-based classifier fusion system is designed. Type-2 fuzzy sets and fuzzy logic system have the capability of handling uncertain and imprecision of a system better than traditional type-1 fuzzy logic. Chapter 7 introduces the basic knowledge of type-2 fuzzy sets and fuzzy systems and then proposes a type-2 based fuzzy classifier fusion model (T2FFSVM). The comparison study between type-1 based and type-2 based fuzzy classifier fusion systems is made. Chapter 7 also compares the proposed fuzzy classifier

fusion models with the existing classifier combination methods addressed in Chapter 3. Chapter 8 studies one interesting biological problem using classifier fusion methods: that is, how protein structures and sequences are related? Can one predict protein sequence by knowing its structure? Chapter 9 will draw the conclusions and also present the future direction of the work.

Most of the dissertation work has been published in the refereed journals and conferences. Specifically, the basic idea of T1FFSVM classifier fusion model by constructing TSK fuzzy model (Chen *et al*., 2005a) was presented at IEEE International Conference on Granular Computing (IEEE-GrC) and its application in bioinformatics has been published in the Journal of Theoretical and Computational Nanoscience (Chen *et al*., 2005b). Genetic fuzzy classifier fusion model (Chen *et al*., 2005c) was proposed at IEEE Congress on Evolutionary Computation (IEEE-CEC) and also published in the Journal of Intelligent & Fuzzy Systems (Chen *et al*., 2007a). Type-2 based classifier fusion model (Chen *et al*., 2006) was presented at IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) and has been published in Applied Soft Computing Journal (Chen *et al*., 2007b). AUC-based classifier fusion model and its applications in bioinformatics (Chen *et al*., 2007c) were presented at International Symposium on Bioinformatics Research and Applications (ISBRA).

Chapter 2 Biological Background and Computational Intelligence in Bioinformatics

The rapid growth of biotechnologies, especially genome sequencing technology and microarray DNA chip technology, generates large scale biomedical data. This chapter briefly reviews the concepts of genes, proteins, and microarray technology. The chapter also addresses the current research situations in which computational intelligence methods have been applied to solve biological problems.

2.1 DNA and Proteins

DNA is a nucleic acid molecule and arranged in chromosomes inside a cell. It contains the genetic information used in the development and functioning of living organisms (Liu, 2002). Human has 23 pairs of chromosomes. "Genome" refers to the collection of all the chromosomes in a cell. Not all segments of the genome carry the genetic instructions. "Genes" are the segments which do hold the genetic instructions. Human is identified to have around 30,000 genes, which only account for about 3% of the human genome. Genes encode the information necessary for making proteins. Proteins are action molecules of a cell and responsible for most of the work in an organism. Gene expression is a process by which the gene's DNA sequence is transcribed to produce a functional protein.

To make a protein molecule, a gene is first transcribed to an RNA via the process of transcription, and then to a protein from an RNA molecule via the translation process as shown in Figure 2.1 (National Health Museum). Transcription is the dominant regulation in the protein synthesis though there are several levels of gene regulation. Messenger

RNA (mRNA) plays an essential role during the transcription. mRNA is an easily degradable molecule and carries coding information to the sites of protein synthesis called ribosome. The amount of certain mRNA copies in a cell roughly reflects the expression level of the corresponding gene (Liu, 2002).

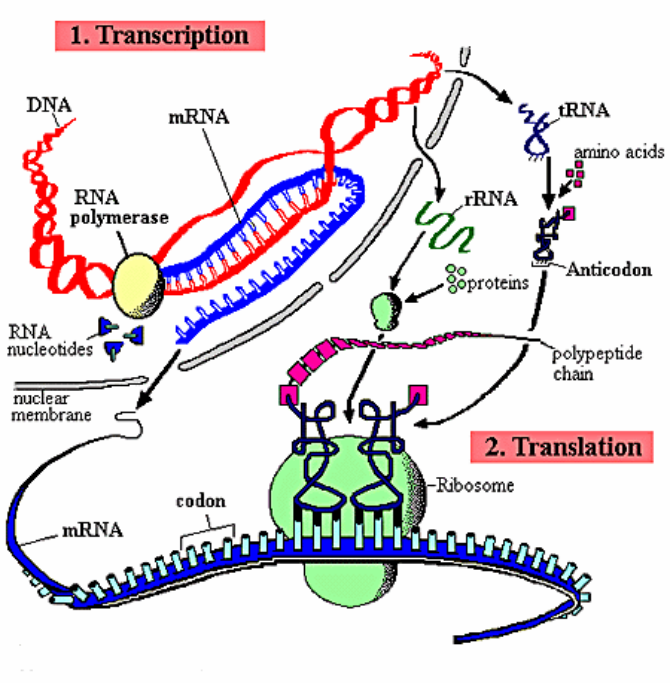


Figure 2.1 Protein Synthesis

2.2 DNA Microarray Technology

DNA microarray is a powerful technology that allows simultaneous measurement of the levels of tens of thousands of mRNA expression. What does a microarray look like? A DNA chip is a collection of gene-specific sequence spots which are deposited or synthesized on a glass slide or a silicon chip in a predetermined spatial order. Each spot has a unique DNA sequence, and thus is a probe for the mRNA encoded by that gene. It

will hybridize only to its complementary DNA strand. To compare the expression levels of two types of cells such as cancer cell versus normal cell, the DNA materials from the two cells are first extracted. Those from one cell type, say, cancer cell, are labeled by fluorescence cy5 (red), and the other cell type by cy3 (green). The microarray is then exposed to the mixture of the two DNA samples for hybridization. When mRNA for a gene is more abundant in the cancer cell than in the normal cell, for example, the array spot corresponding to that gene will show a red color. Since the fluorescence intensities are correlated to the abundance of the corresponding mRNA transcript in the sample, DNA microarray can be used to measure the expression levels of genes in different cells. When more than two types of cells are in consideration, the microarray data often takes the form of a matrix, where each column corresponds to a cell type (e.g., lymphoma cell, leukemia cell, normal cell, etc.) or a treatment (Liu, 2002), and each row corresponds to a gene as shown in Figure 2.2. Thus, through the use of DNA microarrays, one can monitor simultaneously the expression levels of thousands of genes in different types of cells. DNA microarray can be used for gene discovery, disease diagnosis, drug discovery, and toxicological research.

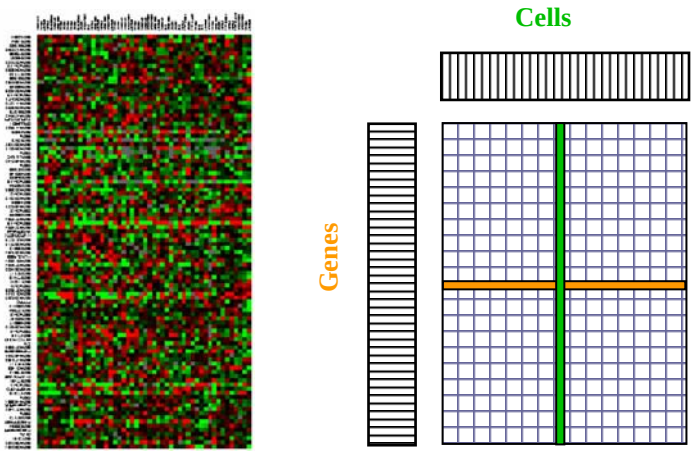


Figure 2.2 Microarray chip and its conceptual view

2.3 Computational Intelligence in Bioinformatics

Computational intelligence methods refer to fuzzy logic, neural networks, and evolutionary algorithms. These methods are able to handle nonlinear separable data, deal with uncertainties and imprecision, search solutions in large spaces, and provide probabilistic or continuous rather than discrete classification results (Rajapakse *et al*., 2007). The advantage of computational intelligence approaches makes them particularly suitable for solving complex biological problems. Recently, the application of computational intelligence methods in bioinformatics has increased. One special section in IEEE/ACM Transactions on Computational Biology and Bioinformatics collects papers that apply computational intelligence knowledge in biology domain (Rajapakse *et al*., 2007).

Fuzzy logic and its theory have been applied on gene expression data to discover the relationship between genes and certain cancers to assist the understanding of the pathobiology of the diseases (Sjahputera *et a*l., 2007). Multiclass classification and

imbalanced class distribution often happen in biological data. Multilayer perception neural networks together with hierarchical decomposition and up-sampling methods have been applied to classify small imbalanced bio-image database (Lerner *et a*l., 2007). Analyzing bio-molecular interaction networks such as metabolic, protein-protein, and protein-DNA networks provides an important way to identify gene and protein functions. Li (Li *et al*., 2007) has constructed a dynamical system using evolutionary algorithms and artificial neural networks to analyze the molecular networks.

Chapter 3 Related Theories

3.1 Support Vector Machines (SVMs)

SVMs have been proved to be a powerful and effective supervised learning technique since they were introduced by Vapnik (Vapnik, 1995). SVMs have been largely and successfully applied in biological and medical domain to analyze microarray gene expression data, identify protein domains or homologies, and detect translation initiation sites.

SVMs employ Structural Risk Minimization (SRM) principle to achieve better generalization ability than conventional machine learning algorithms, such as neural networks, decision trees, and etc, which apply Empirical Risk Minimization (ERM) principle instead.

The goal of SVMs is to find an optimal decision hyperplane to separate classification data into two classes, say, cancer class and normal class. There are many possible separating hyperplane to separate two classes. SVMs select the one which maximizes the distance between the two classes as shown in Figure 3.1. The distance between the two classes is called margin, which is constituted of a small set of training examples, called support vectors, sitting just along the margin. Support vectors are the most informative and critical training data examples for the classification problem. Removal of support vectors could result in a change of the hyperplane. The data examples reside far away from the hyperplane have no influence on the decision of the separating hyperplane.

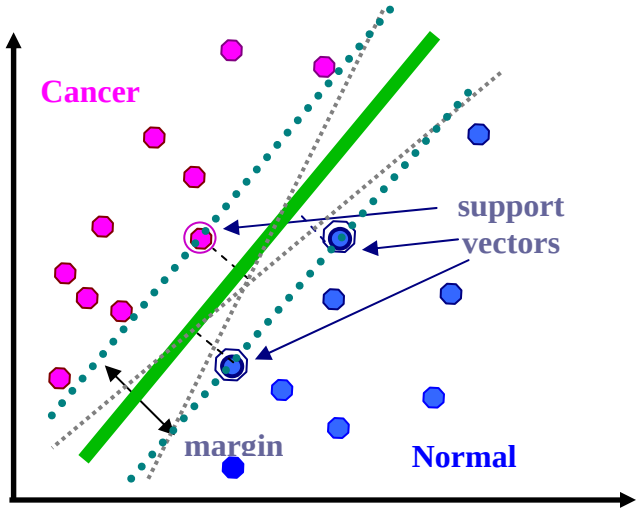


Figure 3.1 Support Vector Machines (SVMs)

Unfortunately, for most real-world applications, there don't exist hyperplanes that successfully and linearly separate two classes in the data's space. To solve the linear inseparability problem, one solution is to map the data from their original space into a higher-dimensional space and determine a separating hyperplane there. This higher dimensional space is called *feature space*, as opposed to *input space* resided by the training data examples. For instance, the data examples in the XOR problem shown in Figure 3.2 cannot be separated in the original two-dimensional input space, no matter how hard we try. But when we transform the data examples into a three-dimensional feature space, we can easily find a separating hyperplane there to separate the two classes.

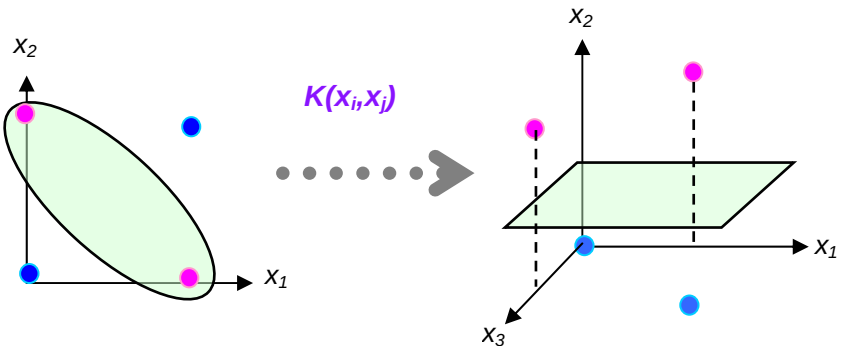


Figure 3.2 XOR problem in input space vs. in feature space

However, mapping the training data examples directly into a higher-dimensional feature space is not only computational expensive but also learning theoretic expensive. The beauty of SVMs relies on that data examples are not explicitly translated from their input space into the high feature space. Instead, the mapping process is done implicitly through a *kernel function*, which returns the dot products between data examples. By applying this kernel trick to define a transformed feature space via a kernel function, SVMs are able to perform efficiently in a nonlinear high-dimensional feature space without being adversely affected by the dimensionality of that space. Indeed, it is possible to work with a feature space of infinite dimension. For example, RBF kernel functions define infinite Hilbert feature spaces as shown in Figure 3.3. Therefore, SVMs can handle not only linear-separable but also nonlinear-separable classification problems efficiently.

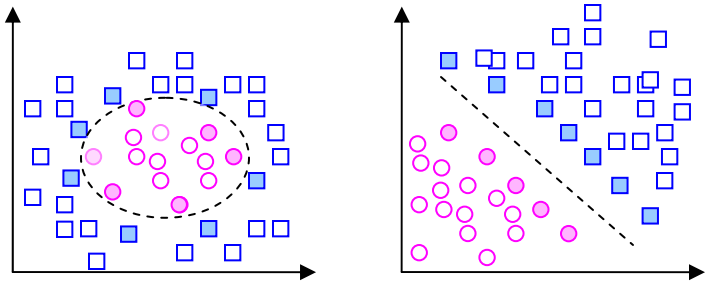


Figure 3.3 RBF kernel functions in its original input space and mapping feature space

The SVM hyperplane is determined by solving a quadratic programming. The decision function of a SVM classifier can be formulated as follows:

$$y = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b\right) \tag{3.1}$$

where $x \in R^n$ is the n-dimensional input vector of a test example, $x_i \in R^n$ is the input vector for the ith training examples, $y_i \in \{-1, 1\}$ is a class label for a binary classification problem, and N is the number of training data examples, and K is a kernel function. α_i and b are the parameters of the model determined during the training of SVM. $\{\alpha_i\}$ are Lagrange multipliers to the following given quadratic problem, one for each training data example.

$$Maximum: W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i \cdot y_j) \tag{3.2}$$

$$Subject\ to: C \geq \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i y_i = 0 \ \ (i = 1..N) \tag{3.3}$$

Of all the training data examples, only support vectors have non-zero value α_i. The upper bound of α_i is defined by a regularization parameter C, which determines the tradeoff

between maximizing margin and minimizing the number of misclassified data examples. It is useful to handle non-separable problems and outliers. The choice of *C* should reflect the knowledge of the noise on the data.

We can see that kernel functions play an important role in SVM classification because kernel functions define the feature spaces in which data examples are classified. The following kernel functions have been used widely and successfully.

Polynomial kernel with degree *d*:

$$K(x_i, x_j) = (1 + x_i \cdot x_j)^d \tag{3.4}$$

Gaussian RBF kernel with tuning parameter σ:

$$K(x_i, x_j) = \exp\left(-\left\|x_i - x_j\right\|^2 / \left(2\sigma^2\right)\right) \tag{3.5}$$

And *sigmoid kernel* with parameter θ:

$$K(x_i, x_j) = \tanh(x_i \cdot x_j - \theta) \tag{3.6}$$

3.2 Combining Classifiers

Classifier combination aims to achieve a higher accuracy than that of individual classifiers. If a set of classifiers is combined in an effective way, the combined classifiers may achieve better performance than the individual classifiers that make them up. This is because that the sets of data examples misclassified by the different individual classifiers would not necessarily overlap. Thus different classifier designs potentially offer complementary information about the data examples to be classified which could improve the performance of the selected classifiers.

One sufficient condition for a combined classifier to be more accurate than any of its individual members is that the individual classifiers should be *diverse* (Hansen and

Salamon, 1990). "Diverse" means that the classifiers make different errors on data examples. To see why diverse is good, image that we have an ensemble of three classifiers: $\{h_1, h_2, h_3\}$ and consider a new data example x. If the three classifiers are not diverse, then when $h_1(x)$ is wrong, $h_2(x)$ and $h_3(x)$ will also be wrong. However, if the errors made by the classifiers are uncorrelated, then when $h_1(x)$ is wrong, $h_2(x)$ and $h_3(x)$ may be correct, so that a majority vote will classify x correctly. It is particularly important that combined classifiers are *different*: such as difference feature sets, different training sets, different classification algorithms, and so on (Kittler *et al*., 1998; Hansen and Salamon, 1990; Ho *et* al., 1994; Ho, 1995; Xu *et* al., 1994).

3.2.1 Approaches to Combining Classifiers

Many strategies can be used to design a classifier combination model (Kuncheva, 2003):

- Different combination schemes. Different classifiers are first created. Then one *combination scheme* is picked to combine the different classifiers.
- Different classifier models. Homogeneous classifiers with different structures, parameters, or initializations, or heterogeneous classifiers can be used to generate different classifiers to be combined.
- Different feature subsets. Different feature selections can be applied to generate different classifiers to be combined.
- Different training sets: Combining classifiers can also come from different groups of training data examples.

3.2.2 Majority Vote and Weighted Majority Vote

Majority vote is one of the simplest and most intuitive methods to combine classification decisions. Given a set of classifier $H = \{h_1, \ldots, h_T\}$ for a binary classification problem such that each individual classifier assigns a data example $x_i \in \mathrm{R}^n$ into a class label w_1 or w_2, h_t: $R^n \rightarrow \Omega$, where $\Omega = \{w_1, w_2\}$. The majority vote method is to assign the class label to $\boldsymbol{x}$ which is supported by the majority of individual classifiers. That is, $h(x) = w_1$ if $sign(\sum_{t=1}^{T} h_t(x))$ is positive or zero, and $h(x) = w_2$ otherwise, where $t = 1..T$. The majority vote is guaranteed to do better than an individual classifier when the classifiers have accuracy greater than 50% (Lam, 2000). If certain individual classifiers are considered better classifiers than the others, weighted majority vote can be employed, where each hypothesis h_t has a weighting factor k_t, that is, $h(x) = w_1$ if $sign(\sum_{t=1}^{T} k_t h_t(x))$ is positive or zero, and $h(x) = w_2$ otherwise.

3.2.3 Bagging

Bagging, introduced by Breiman (Breiman, 1996), is one of the most effective ensemble algorithms. The essential idea of bagging is to combine many weak classifiers to produce a strong aggregated classifier. The individual classifiers are constructed through resampling the original training data set using bootstrap replacement strategy. Each data example $\langle \boldsymbol{x}_i, y_i \rangle$ may appear several times or not at all in a new version of data set, where input $x_i \in R^n$ and output $y_i \in \{+1, -1\}$, $i = 1..N$, for a binary classification problem. Given a training set $D = \{\langle x_i, y_i \rangle\}_{i=1}^{N}$, bagging algorithm is defined as follows:

1) Construct T sets of bootstrap samples D_t (t = 1..T). Each bootstrap sample $D_t = \{\langle x_i^*, y_i^* \rangle\}_{i=1}^N$ is selected randomly from the original data set D with replacement.
2) Train a machine on each D_t and obtain T outputs $h_t(x)$ (t=1,…,T) for each testing data example $\boldsymbol{x}$.
3) Aggregate the classifiers using the majority vote:

$$h(x) = sign(\sum_{j=1}^{B} h_j(x)) \tag{3.7}$$

There are number of variations of the basic bagging algorithms. For instance, "sub-bagging" constructs subsamples with the size less than the original training set size. No replacement bagging samples the training set without replacement. It's proved that bagging algorithms can reduce variance of classifiers, increase the stability of classifiers, and enhance the performance of weak classifiers.

3.2.4 Boosting

Boosting (Schapire, 1990) is also a popular ensemble model for boosting the performance of a weak classifier. Unlike bagging algorithms, boosting *adaptively* changes the distribution of the training set based on the performance of previous classifiers. Boosting includes a family of methods. The most popular one is AdaBoost (**Ada**ptive **Boost**ing) (Freund and Schapire, 1995). Similar to bagging, the AdaBoost algorithm generates a set of classifiers and uses majority voting to make the final decision. Beyond this, the two algorithms differ substantially. The AdaBoost algorithm generates the classifiers sequentially, while Bagging can generate them in parallel. AdaBoost also changes the weights of the training data examples according to the

performance of the previous classifiers so that the next classifier focuses on data examples where the previous classifier failed. Adaptive is the main idea of AdaBoost. Thus, AdaBoost requires a classifier have ability to make weighted learning. Given a training set $D = \{\langle x_i , y_i \rangle\}_{i=1}^{N}$, AdaBoost is defined as follows:

1) Initialize the data distribution as $P_1(i) = 1/N$, where N is the number of training data examples.
2) For $t = 1, \ldots, T$:
   2.1) Train a machine with weights $P_t(i)$ and get the classifier h_t
   2.2) Compute the weighted error of classifier h_t:

$$\varepsilon_t = \sum_{i=1}^{N} P_t(i)[y_i \neq h_t(x_i)] \tag{3.8}$$

   2.3) Compute the importance of classifier h_t:

$$\alpha_t = \tfrac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{3.9}$$

   2.4) Update the distribution:

$$P_{t+1}(i) = P_t(i) \exp(-\alpha_t y_i h_t(x_i)) \tag{3.10}$$

3) The final decision is based on weighted voting:

$$h(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{3.11}$$

By applying re-weighting strategy, data examples identified correctly by the classifier are weighted less and those identified incorrectly are weighted more. Thus, AdaBoost algorithm focuses on remaining errors, directly addresses the representational problem, and is capable of enhancing the performance of a weak classifier (Freund and Schapire, 1996).

3.2.5 Combining Posterior Probability Methods

The classifier fusion models can also be established by combining posterior probability of individual classifiers. Suppose $H = \{h_1, \ldots, h_T\}$ be a set of classifiers and $\Omega = \{w_1, w_2\}$ be class labels for a binary classification problem. The classifier input is a *n*-dimensional feature vector $x_i \in R^n$ and the classifier output is a 2-D vector $h_t(x) = [h_{t,1}(x), h_{t,2}(x)]^T$, where $h_{t,j}(x)$ is the degree that x comes from the class w_j and is supported by classifier $h_t(x)$ (j = 1..2). Without loss of generality, $h_{t,j}(x)$ can be defined within the interval [0,1] (t = 1..T, j = 1..2). We call the classifier outputs "soft labels". $h_{t,j}(x)$ is usually an estimate of the posterior probability $P(w_j|x)$. When the classifiers are applied to the combination methods, the probability of the class label w_j, (j = 1..2), assigned to x can be defined as follows (Shipp and Kuncheva, 2002; Kuncheva, 2003):

$$\hat{P}_j = \Theta(h_{1,j}(x), \ldots, h_{T,j}(x)), (j = 1..2) \tag{3.12}$$

where Θ is the combination method respectively for *maximum*, *minimum*, *average*, *median*, or *product*. The class w_j with maximum $\hat{P}_j$ is the assigned class based on the different rules:

- *Minimum Rule*: First, the minimum of $h_t(x)$ (t=1..T) and the minimum of (1- $h_t(x)$) (t=1..T) are selected. Second, two minimums are compared and the class with the larger value is selected. If the two minimums are same, then the second minimums are compared.
- *Maximum Rule*: Similar to the minimum rule but the maximums are compared.
- *Average Rule*: If the average of $h_{t,1}(x)$ (t=1..T) is greater than the average of $h_{t,2}(x)$ (t=1..T), the class w_1 is selected as the final classifier output. Otherwise, the class

w_2 is selected. In term of a binary classification, that means, the class w_j (j=1..2) is selected if the average of $h_{t,j}(x)$ (t=1..T) is greater than 0.5.

◊ *Median Rule*: Similar to the average method, if the median of $h_{t,1}(x)$ (t=1..T) is greater than 0.5, the class w_1 is selected. Otherwise, the class w_2 is selected.

◊ *Product Rule*: First, the product of $h_t(x)$ (t=1..T) and the product of (1- $h_t(x)$) (t=1..T) are calculated. Second, two products are compared and the class with the larger value is selected.

Table 3.1 illustrates an example of how the combination methods work.

Table 3.1 Combination Methods

Classifier	Support for w_1	Support for w_2	Final Decision
h_1	0.3	0.7	W_2
h_2	0.4	0.6	W_2
h_3	0.8	0.2	W_1
h_4	0.4	0.6	W_1
h_5	0.7	0.3	W_1
MAX	0.8	0.7	W_1
MIN	0.3	0.2	W_1
AVG	0.52	0.48	W_1
PRO	0.02688	0.01512	W_1

Chapter 4 Combining SVM Classifiers Using Fuzzy Logic

4.1 Fuzzy Logic

Fuzzy logic was introduced by Zadeh (Zadeh, 1965) and has demonstrated the powerful framework in manipulating the imprecision in real-world applications. Fuzzy logic and fuzzy rule-based systems have been widely used to model complex systems to capture the uncertainties of human knowledge, such as “speed is high” or “temperature is cold”. In the domain of fuzzy logic, a linguistic variable like “speed” or “temperature” is defined as a set of fuzzy sets like “high” and “low”, or “warm” and “cold” in form of membership functions (MFs), which are used to express different grades of memberships in fuzzy sets.

The basic structure of a fuzzy logic system (FLS) is shown in Figure 4.1. It is usually composed of four components: fuzzifier, fuzzy rule base, fuzzy inference engine, and defuzzifier. Fuzzifier maps crisp inputs into fuzzy sets. Fuzzy rule base is used to represent the fuzzy relationships between input and output fuzzy sets and is expressed in IF-THEN statements. The inference engine combines the fired fuzzy rules and maps crisp inputs into fuzzy output sets. The defuzzifier is used to convert output fuzzy sets into crisp outputs.

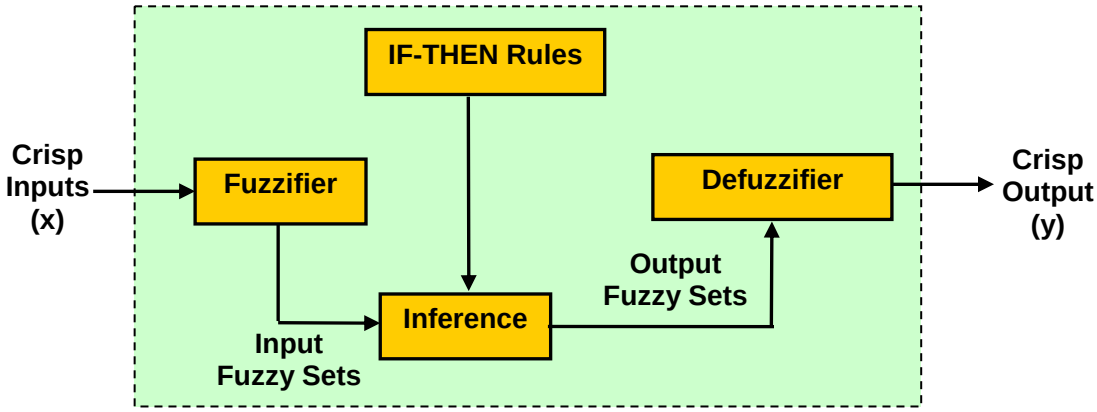


Figure 4.1 The structure of a type-1 FLS

A fuzzy system consists of a set of IF-THEN fuzzy rules. IF-THEN Fuzzy rule base is used to represent the fuzzy relationships between input and output fuzzy sets. Based on the different expression of the consequent part of an IF-THEN fuzzy rule, there are usually two popular models used to construct a FLS: rule-based Mamdani model (Mamdani, 1974) and parametric-based Takagi-Sugeno-Kang (TSK) model (Takagi and Sugeno, 1985).

In the Mamdani-type fuzzy model, the consequence part of an IF-THEN fuzzy rule is defined by linguistic variables as follows (Mamdani and Assilian, 1973):

R_i: IF x_1 is A_1^k and x_2 is A_2^k and ... and x_N is A_N^k, THEN y_1 is B_1^k and y_2 is B_2^k and ... and y_M is B_M^k.

where x_j (j=1 .. N) and y_l (l=1 .. M) are input and output linguistic variables, and A_j^k and B_l^k are fuzzy sets used to define the input and output linguistic variables respectively.

Unlike the Mamdani model, the consequent of an IF-THEN fuzzy rule in the TSK model is crisp and defined as a polynomial function of input values as given (Takagi and Sugeno, 1985):

R_i: IF x_1 is A_1^k and x_2 is A_2^k and ... and x_N is A_N^k , THEN $y_i = f_i(x_1, x_2, \ldots, x_N, \lambda_i)$.

where x_j and A_j^k are defined same as in the Mamdani model, y_i is a function of input linguistic variables, and λ_i is the parameter vector of the function.

All the fired fuzzy rules are invoked using the MFs of inputs. The result of a fuzzy system is calculated by aggregating individual fuzzy rule outputs and eventually defuzzified into a crisp value as the final result of the model.

4.2 Architecture of Fuzzy Multi-SVM Fusion Model

This chapter proposes a fuzzy logic based classifier fusion model for SVM classifiers. The model is designed based on the following observation. For a training data set, when we use different SVMs with different kernel functions to train a set of data examples, different SVM hyperplanes or classifiers are usually formed. When the different classifiers are applied onto a same set of testing data, different decisions might be made. Since different decisions may provide complementary information about the data examples to be classified, when we combine different SVM classifiers altogether in a fusion model, the composite classifier might outperform all its individual base SVM classifiers. Therefore, different kernel functions make individual SVM classifiers *diverse*, which is one of the essential conditions for a composite classifier to outperform its individual classifiers including the best. This idea can be easily observed from the following example. Suppose we have two data examples labeled as data1 and data2.

Data1 is a cancer tissue data example and data2 is a normal data example. SVM1 classifies data1 as a cancer tissue correctly but misclassifies data2 as a cancer tissue. On the contrary, SVM2 misclassifies data1 as a normal tissue but correctly classify data2 as a normal tissue. When these two SVM classifiers are combined, the composite classifier might classify both data correctly. Therefore, the combined classifier might have better performance than the both individual SVM classifiers.

Based on this consideration, the basic SVM fusion model is constructed as shown in Figure 4.2. The system can be divided into two phases. In Phase I, different SVMs are trained and classified to obtain classification decisions for testing data examples and individual SVM classifier accuracies. In Phase II, a SVM fusion model is constructed to combine multiple SVM classifiers. This fusion model will combine the different classification results from different SVM classifiers by take the consideration of the performance of each individual classifier to make the final decision.

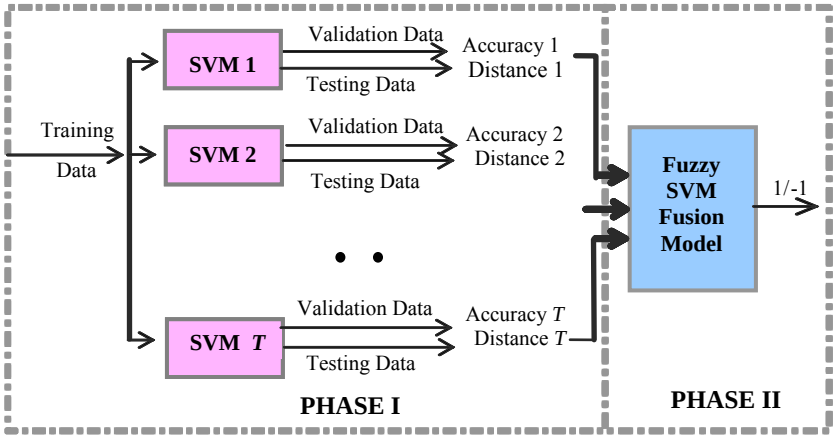


Figure 4.2 Basic architecture of multi-SVM fusion model

Although the model is designed for binary classification, it can also be expanded to solve multi-class classification problems by transforming a multi-class classification problem to several binary classification problems. Many strategies can be applied for the problem transformation, among which one-against-the-rest strategy (Weston and Watkins, 1998) and one-against-one strategy (Hastie and Tibshirani, 1996) are often used. Given a classification problem with k number of classes, the one-against-the-rest schema breaks the original problem into k binary classification problems, while the one-against-one strategy creates $\frac{1}{2}k(k-1)$ classifiers. The classification decision is made by a voting scheme such as majority voting for one-against-one strategy or maximum class voting for one-against-the-rest strategy.

To further clarify the model and the FLS, in the following sections, we will illustrate how to combine, as an example, THREE SVM classifiers to solve a binary classification problem using the proposed model. This process can be easily extended to combine arbitrary number of SVM classifiers or further to solve a multi-class classification problem in general.

4.3 Combining THREE SVM classifiers

4.3.1 Phase I: Training Individual SVM Classifiers

In the first phase, training data are trained by three different SVMs. Three different SVM classifiers could differ in the types of kernel functions, such as polynomial kernel or Gaussian RBF kernel, or in the parameters of the same kernel type, like different degree d in polynomial kernel functions. After three SVM separating hyperplanes are trained, the validation dataset is then classified to obtain accuracies of three individual

SVM classifiers and the testing dataset is classified to obtain distance of each data example to three SVM hyperplanes. Accuracies and distances are then ready for the use in phase II.

4.3.2 Phase II: FLS for Classifier Combination

The SVM fusion model in Phase II in Figure 4.2 is constructed using the knowledge of the traditional fuzzy logic. Here, a layered Mamdani-type fuzzy rule-base system (Mamdani and Assilian, 1973) is established as shown in Figure 4.3 where three SVM classifiers are combined together to form a composite classifier.

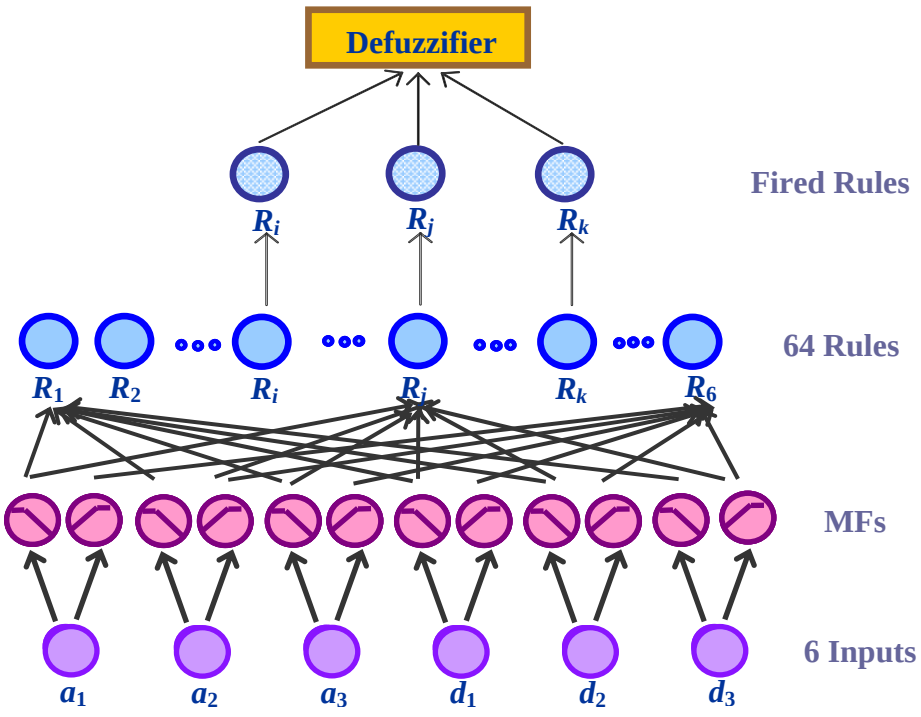


Figure 4.3 The FLS of combining multi-SVM classifiers

The first layer is the input layer, including three accuracy inputs a_i (i=1..3) for three SVM classifiers, and three distance inputs d_i (i=1..3) of each data example to three SVM separating hyperplanes. In the next layer, the fuzzy sets and MFs are specified for all the fuzzy linguistic variables. Each of the six input linguistic variables are represented by two fuzzy sets. The fuzzy system has one output represented by 16 fuzzy sets according to Mamdani fuzzy model (Mamdani and Assilian, 1973), where the consequence of an IF-THEN fuzzy rule is also defined as fuzzy sets. In the next layer, 64 fuzzy rules are formulated and the IF-THEN Fuzzy rule base is used to represent the fuzzy relationships between the input and output fuzzy sets. In the following layer, the firing strength of each fired rule is calculated based on the fuzzy reasoning. The aggregated output is then computed and defuzzified in the final layer. If the aggregated output is greater than or equal to zero, the data example is considered in the positive class and its defuzzification value is 1. Otherwise, it belongs to the negative class and its defuzzification value is -1. The following sections explain the system in further detail.

4.3.2.1 Fuzzy Input and Output MFs

The fuzzy MFs are defined as in Figure 4.4. Each accuracy input has two fuzzy sets: low and high, and each distance input is also represented by two fuzzy sets: negative and positive. Each output is defined by 16 fuzzy sets, one corresponding to one of 16 groups of 64 rule consequences.

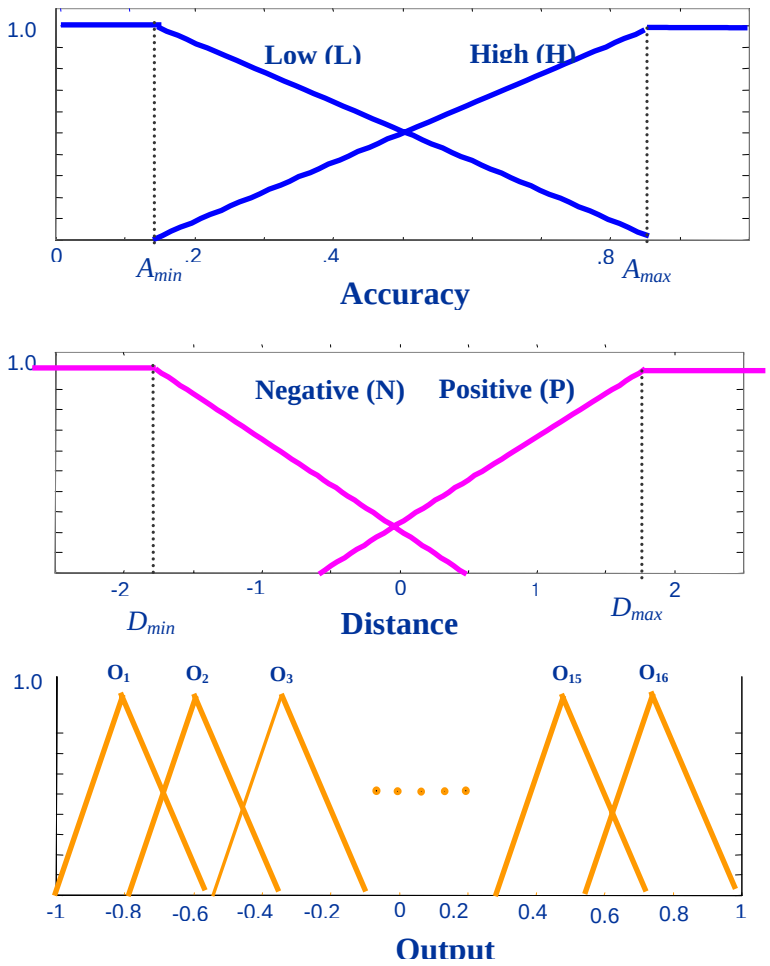


Figure 4.4 The MFs for inputs and output

The domain of accuracy MFs is defined in [A_{min}, A_{max}]. In a binary classification problem, each data example either belongs to the positive class or to the negative class. When a data example is close to the SVM hyperplane, say, when the distance of a data example to the hyperplane is in a certain range, like [-0.5, 0.5], SVMs are sensitive to it. Different SVM classifiers might classify the data example in different classes. Here we consider the data examples within the range partially in the positive class and partially in the negative class. Once the distance of a data example is beyond the range, it will purely

belong to either the negative or positive class. Thus, we define the domain of distance MFs in [D_{min}, 0.5] for negative MF and in [-0.5, D_{max}] for positive MF. The output value is defined in the range [-1, 1]. Defining the output value in other ranges makes no big difference if the defuzzification formula is adjusted accordingly. We will explain why there are 16 output fuzzy sets in the next section. An isosceles triangle is used to represent each output fuzzy set.

4.3.2.2 Fuzzy Rule Base

Based on the six inputs and one output, we define the ith (i = 1...64) fuzzy rule as follows:

R_i: IF a_1 is A_{i1} and a_2 is A_{i2} and a_3 is A_{i3} and d_1 is D_{i1} and d_2 is D_{i2} and d_3 is D_{i3}, THEN g_i is O_i (i = 1...64).

where a_j (j=1..3) and d_j (j=1..3) are input linguistic variables representing SVM accuracy and the distance of a data example to the SVM separating hyperplane; A_{i1}, A_{i2} and A_{i3} are fuzzy sets of accuracy linguistic variable in the universe of discourse {Low, High}; D_{i1}, D_{i2} and D_{i3} are fuzzy sets of distance linguistic variable in the universe of discourse {Negative, Positive}; and O_i is a fuzzy set of the output in $\{O_1...O_{16}\}$.

Since the fuzzy system has three accuracy inputs and three distance inputs, and each accuracy and each distance have two possibilities respectively, there are 2 ^ 6 = 64 exhaustive fuzzy rules in total.

If we make an analysis of these 64 rules, it is not difficult to find that some rules have close relations with some other rules. For example, the following two rules are actually quite similar:

$\boldsymbol{R_m}$: IF a_1 is L and a_2 is L and a_3 is L and d_1 is N and d_2 is N and d_3 is P, THEN g_m is O_m.

$\boldsymbol{R_n}$: IF a_1 is L and a_2 is L and a_3 is L and d_1 is N and d_2 is P and d_3 is N, THEN g_n is O_n.

where L denotes “Low”, P denotes “Positive” and N denotes “Negative” fuzzy set.

The two rules indicate a similar situation in which two SVMs have “Low” accuracies and classify a data example in “Negative” distance, and another one has “Low” accuracy and classifies the data example in “Positive” distance. We group such similar fuzzy rules together and end up with totally 16 groups for the 64 fuzzy rules. The rules in a same group have a similar <accuracy, distance> combination for all three SVMs, just like the example given above. That’s why we define 16 output fuzzy sets since there are 16 groups of similar IF-THEN fuzzy rules. Table 4.1 shows the 64 fuzzy rules and their corresponding fuzzy consequences which are represented by one of 16 fuzzy sets.

Table 4.1 64 Fuzzy rules and their consequent output fuzzy set assignments

Rule No.	*Inputs*						*Output*
	a_1	a_2	a_3	d_1	d_2	d_3	
R1	H	H	H	N	N	N	O_1
R2	L	H	H	N	N	N	
R3	H	L	H	N	N	N	O_2
R4	H	H	L	N	N	N	
R5	L	H	H	P	N	N	
R6	H	L	H	N	P	N	O_3
R7	H	H	L	N	N	P	
R8	L	L	H	N	N	N	O_4
R9	L	H	L	N	N	N	
R10	H	L	L	N	N	N	
R11	L	L	H	N	P	N	
R12	L	L	H	P	N	N	
R13	L	H	L	N	N	P	
R14	L	H	L	P	N	N	
R15	H	L	L	N	N	P	O_5
R16	H	L	L	N	P	N	
R17	H	H	H	N	N	P	
R18	H	H	H	N	P	N	
R19	H	H	H	P	N	N	
R20	L	L	L	N	N	N	O_6
R21	L	L	H	P	P	N	O_7
R22	L	H	L	P	N	P	
R23	H	L	L	N	P	P	
R24	L	L	L	N	N	P	
R25	L	L	L	N	P	N	
R26	L	L	L	P	N	N	
R27	L	H	H	N	N	P	
R28	L	H	H	N	P	N	O_8
R29	H	L	H	N	N	P	
R30	H	L	H	P	N	N	
R31	H	H	L	N	P	N	
R32	H	H	L	P	N	N	
R33	L	L	L	N	P	P	
R34	L	L	L	P	N	P	
R35	L	L	L	P	P	N	
R36	L	H	H	P	N	P	O_9
R37	L	H	H	P	P	N	
R38	H	L	H	N	P	P	
R39	H	L	H	P	P	N	
R40	H	H	L	N	P	P	
R41	H	H	L	P	N	P	
R42	L	L	H	N	N	P	O_{10}
R43	L	H	L	N	P	N	
R44	H	L	L	P	N	N	
R45	L	L	L	P	P	P	O_{11}
R46	L	L	H	N	P	P	
R47	L	L	H	P	N	P	
R48	L	H	L	N	P	P	
R49	L	H	L	P	P	N	O_{12}
R50	H	L	L	P	N	P	
R51	H	L	L	P	P	N	
R52	H	H	H	N	P	P	
R53	H	H	H	P	N	P	
R54	H	H	H	P	P	N	
R55	L	L	H	P	P	P	
R56	L	H	L	P	P	P	O_{13}
R57	H	L	L	P	P	P	
R58	L	H	H	N	P	P	
R59	H	L	H	P	N	P	O_{14}
R60	H	H	L	P	P	N	
R61	L	H	H	P	P	P	
R62	H	L	H	P	P	P	O_{15}
R63	H	H	L	P	P	P	
R64	H	H	H	P	P	P	O_{16}

4.3.2.3 Fuzzy System Output and Defuzzification

The output is calculated by aggregating individual rule contributions:

$$y = \sum_{i=1}^{64} \beta_i g_i \Big/ \sum_{i=1}^{64} \beta_i \tag{4.1}$$

where g_i denotes the output value of the *i*th rule and is represented by the centroid of the gravity of the *i*th isosceles triangle. β_i denotes the firing strength of the *i*th rule. It's defined by product *t*-norm:

$$\beta_i = \prod_{j=1}^{3} \mu_{A_{ij}}(a_j) * \mu_{D_{ij}}(d_j) \tag{4.2}$$

where $\mu_{A_{ij}}(a_j)$ and $\mu_{D_{ij}}(d_j)$ are the membership values of input a_j and d_j (j=1…3) in the fuzzy set A_{ij} and D_{ij}. By choosing *t*-norm "product" instead of other operators such as "min" to calculate the firing strength, the result is more precise since membership grades from all three SVM classifiers are considered.

If the defuzzified result is greater than or equal to 0, we consider the data point in the positive class. Otherwise, it is in the negative class. Therefore, the output is defuzzified as follows:

$$\hat{Z}(y) = \begin{cases} +1, & if \quad y \geq 0 \\ -1, & if \quad y < 0 \end{cases} \tag{4.3}$$

4.4 Experiments on Biomedical Cancer Data

4.4.1 Experimental Design

The experiment is based on cross-validation. A dataset *S* is first divided into *n* subsets. One subset is used as a group of testing data and all the other subsets used as training data. So, there are *n* groups of testing data and training data. Each data example

is either in testing dataset or in training dataset in one group. Next, test individual SVMs *n* times (*n*-fold cross validation). The testing data here will be used as the testing data in Phase II of the system in Figure 4.2. The testing accuracy will be compared with the accuracy from the model in Phase II. The performance of the model is estimated by the average of *n* accuracies from *n* different testing data. If the dataset has been separated into training and testing data, only one fold is necessary in this stage.

The system inputs in Phase II are prepared as follows. Three distance inputs of the input are the distances of each testing data example to the three individual SVM classifier hyperplanes. To obtain the individual SVM classifier performance in the FLS measured by accuracy, we have several choices:

- The first choice is to use testing accuracies as the accuracy inputs in the FLS. But apparently, it's not the right choice since testing accuracies are only used to evaluate how good the model is and in the real applications, the testing accuracies of unseen data are never known ahead.
- The second choice is to use training accuracies as the accuracy inputs in the FLS. Unlike the first choice, training accuracies can be obtained easily by just classifying the training data using the SVM hyperplane formed by the same training data. However, training data cannot correctly reflect the distribution of testing data. It's very common that one SVM has a high training accuracy but low testing accuracy. Therefore, if we use training accuracies as the accuracy inputs of the FLS, the performance of the model will definitely be distorted.
- The third choice is to use the testing accuracies of a different group of data other than the testing data. This group of data plays the role of the validation data in the

model. The validation data are expected to have the similar distribution as the testing data in SVM feature spaces and thus can reflect the testing data well.

In the dissertation, the validation data are prepared and the accuracies of the validation data are used as the three accuracy inputs of the fuzzy system. The validation data are prepared as follows. Each of *n*-fold training data is further divided into *m* subsets (*m*-fold cross validation). One fold of the data is treated as the validation dataset and all the other data as second-level training data. The validation data are classified to get the validation accuracies. The average of *m* validation accuracies from *m*-fold classification will be used as the SVM accuracy inputs in the fuzzy fusion model.

4.4.2 Experimental Data Description

Two datasets from Kent Ridge Biomedical Data Set Repository (Li and Liu, 2003): and one dataset from UCI data mining repository (Merz, 1998) are used to estimate the performance of the fuzzy SVM fusion models. All the three datasets are binary classification data. The data have been normalized in [0, 1] before the classification.

1) *Colon Tumor Dataset:* The dataset is gene expression data. It contains 62 data examples, among which 40 examples are tumor tissues and 22 are normal tissue. Each data example is composed of 2,000 genes as 2,000 features.
2) *Ovarian Cancer Dataset:* Ovarian cancer is one of the deadly diseases among women. This dataset is proteomic data used to identify proteomic patterns in serum that distinguish ovarian cancer from non-cancer samples. It contains 253 data samples, in which 91 samples are non-cancer normal data and the rest 162 examples are ovarian cancer data. Each data example contains 15154 features, indicating the relative amplitude of the intensity at each molecular mass / charge

(M/Z) identity. The data have been normalized in the range [0, 1] before the experiment.

3) *Wisconsin Breast Cancer Dataset*: The dataset is clinical data. It contains 683 data examples, among which 239 examples are malignant data and the rest 444 examples are benign data. Each data example contains 9 features.

4.4.3 Experimental Results

In the first phase, all the datasets are classified in 4-fold cross-validation (n=4) and each training data are further divided into 3-fold (m=3) in order to obtain the validation accuracies needed in the second phase. Table 4.2a-c show the testing and training accuracies of the datasets. The regularization parameter C of the SVMs is set to 1 during the SVM training. Two kinds of kernels: polynomial kernels and RBF kernels are used to train the data. The degree of the polynomial kernels is set to 1...5, and the parameter σ of the RBF kernels is set to 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1},10^{0}, 10^{1}. The data in Phase I of the model are classified using SVM^{light} (Joachims, 1999).

Table 4.2(a) Training and testing accuracies for Colon Tumor Data
(4-fold cross validation, n=4)

Kernels		Training Accuracy (%)					Testing Accuracy (%)				
Polynomial	*d*	1	2	3	4	Avg.	1	2	3	4	Avg.
poly_1	1	91.3	84.8	89.4	85.1	87.6	75.0	93.8	86.7	80.0	83.9
poly_2	2	95.7	93.5	95.7	89.4	93.6	81.3	87.5	86.7	86.7	85.5
poly_3	3	100	97.8	100	100	99.5	75.0	75.0	86.7	80.0	79.2
poly_4	4	100	100	100	100	100	87.5	81.3	66.7	66.7	75.5
poly_5	5	100	100	100	100	100	87.5	81.3	66.7	66.7	75.5
RBF	σ										
rbf_0.0001	0.0001	65.2	67.4	63.8	61.7	64.5	62.5	56.3	66.7	73.3	64.7
rbf_0.001	0.001	65.2	67.4	63.8	61.7	64.5	62.5	56.3	66.7	73.3	64.7
rbf_0.01	0.01	65.2	67.4	63.8	61.7	64.5	62.5	56.3	66.7	73.3	64.7
rbf_0.1	0.1	65.2	67.4	63.8	61.7	64.5	62.5	56.3	66.7	73.3	64.7
rbf_1	1	93.5	93.5	93.6	87.2	92.0	75.0	93.8	86.7	73.3	82.2
rbf_10	10	100	100	100	100	100	62.5	56.3	66.7	80.0	66.4

Table 4.2(b) Training and testing accuracies for Ovarian Cancer data (4-fold cross validation, *n*=4)

Kernels		Training Accuracy (%)					Testing Accuracy (%)				
Polynomial	*d*	1	2	3	4	Avg.	1	2	3	4	Avg.
poly_1	1	100	100	100	100	100	100	100	100	100	100
poly_2	2	100	100	100	100	100	100	100	98.4	100	99.6
poly_3	3	100	100	100	100	100	96.9	100	98.4	98.1	98.4
poly_4	4	100	100	100	100	100	96.9	98.4	98.4	96.3	97.6
poly_5	5	100	100	100	100	100	96.9	98.4	92.1	95.4	95.7
RBF	σ										
rbf_0.0001	0.0001	91.0	88.9	91.1	92.1	90.8	89.1	92.1	87.3	90.5	89.7
rbf_0.001	0.001	99.5	99.5	100	99.5	99.6	98.4	100	98.4	98.4	98.8
rbf_0.01	0.01	100	100	100	100	100	92.1	92.1	84.1	87.3	88.9
rbf_0.1	0.1	100	100	100	100	100	64.1	61.9	63.5	66.7	64.0
rbf_1	1	100	100	100	100	100	64.1	61.9	63.5	66.7	64.0
rbf_10	10	100	100	100	100	100	64.1	61.9	63.5	66.7	64.0

Table 4.2(c) Training and testing accuracies for Breast Cancer data (4-fold cross validation, *n*=4)

Kernels		Training Accuracy (%)					Testing Accuracy (%)				
Polynomial	*d*	1	2	3	4	Avg.	1	2	3	4	Avg.
poly_1	1	97.46	97.85	97.27	96.88	97.37	94.74	95.93	97.66	98.25	96.65
poly_2	2	100	99.80	100	99.61	99.85	90.06	93.02	94.15	93.57	92.70
poly_3	3	100	100	100	100	100	90.06	94.77	93.57	95.32	93.43
poly_4	4	100	100	100	100	100	91.23	93.60	92.40	95.91	93.29
poly_5	5	100	100	100	100	100	90.64	94.19	91.81	95.91	93.14
poly_6	1	100	100	100	100	100	91.81	95.35	91.81	97.08	94.01
poly_8	2	100	100	--	100	--	92.40	94.77	--	97.66	--
poly_10	5	34.96	34.96	--	35.09	--	35.09	34.88	--	34.50	--
RBF	σ										
rbf_0.0001	0.0001	97.85	96.09	95.70	95.52	96.29	91.23	96.51	97.66	98.83	96.06
rbf_0.001	0.001	97.66	97.46	97.07	96.69	97.22	94.15	95.93	97.66	99.42	96.79
rbf_0.01	0.01	98.44	98.05	97.46	96.88	97.71	92.98	95.93	98.25	98.83	96.50
rbf_0.1	0.1	99.80	99.80	99.41	98.83	99.46	91.81	94.19	97.08	98.25	95.33
rbf_1	1	100	100	100	100	100	86.55	84.30	87.13	91.81	87.45
rbf_10	10	100	100	100	100	100	70.76	73.84	71.35	76.61	73.14

The averages of *m*-fold (*m*=3) validation accuracies for each of *n*-fold (*n*=4) testing are shown in Table 4.3. These accuracies will be used in the FLS as the accuracy inputs.

Table 4.3 Average validation accuracies (%) based on *m*-fold validation for each of 4 folds (*n*=4, *m*=3)

Kernels		Colon Tumor					Ovarian Cancer					Breast Cancer				
Polynomial	*d*	1	2	3	4	Avg.	1	2	3	4	Avg.	1	2	3	4	Avg.
poly_1	1	89.03	71.67	80.83	82.92	81.11	99.47	98.94	100	100	99.60	96.49	97.26	95.70	96.10	96.39
poly_2	2	89.03	82.92	85.14	74.58	82.92	99.47	98.41	100	100	99.47	93.76	92.97	93.56	90.45	92.69
poly_3	3	82.36	76.39	85.14	68.06	77.99	98.94	97.89	99.47	97.91	98.55	94.35	94.34	93.36	91.42	93.37
poly_4	4	82.36	78.47	83.06	65.83	77.43	97.36	97.37	97.89	94.77	96.85	94.54	94.14	93.56	92.19	93.61
poly_5	5	80.28	80.56	80.97	61.39	75.80	97.36	96.85	95.26	92.13	95.40	94.54	94.73	92.58	92.20	93.51
RBF	σ															
rbf_0.0001	0.0001	65.28	67.36	63.89	61.67	64.55	80.95	79.42	83.73	76.90	80.25	97.27	95.30	94.73	94.33	95.41
rbf_0.001	0.001	65.28	67.36	63.89	61.67	64.55	95.77	96.84	97.36	97.38	96.84	97.47	97.46	95.70	95.51	96.54
rbf_0.01	0.01	65.28	67.36	63.89	61.67	64.55	85.19	82.58	85.82	80.03	83.41	97.47	96.87	95.51	95.70	96.39
rbf_0.1	0.1	65.28	67.36	63.89	61.67	64.55	64.02	64.69	64.23	63.19	64.03	96.30	95.89	94.92	94.33	95.36
rbf_1	1	86.95	78.20	80.83	80.83	81.70	64.02	64.69	64.23	63.19	64.03	86.94	87.30	86.71	85.19	86.54
rbf_10	10	65.28	67.36	63.89	61.67	64.55	64.02	64.69	64.23	63.19	64.03	67.64	66.22	66.02	66.08	66.49

Next, we select three SVMs and feed the fuzzy SVM fusion system with the validation accuracies shown in Table 4.3 as the three accuracy inputs and with the distances of one testing data example to the three SVM hyperplanes as the three distance inputs. The FLS will generate a crisp output in [-1, 1]. Based on the sign of the output, the data example is determined either in positive or in negative class. For all the testing data examples, we repeat the same procedure and thus obtain the accuracy of the fuzzy fusion model regarding to the entire testing dataset. Since we use 4-fold cross-validation, we have 4 groups of testing data. For the same three selected SVMs, we test the fuzzy fusion model 4 times using 4 different testing data. For instance, Table 4.4 shows one

selection of three SVMs, their validation accuracies used in the model and the model accuracies of each fold of testing data for ovarian cancer data.

Table 4.4 Three selected SVMs, their validation accuracies (%), and the accuracies (%) from the fuzzy fusion model (T1FFSVM)

Data	Fold	SVM1 (poly_1)	SVM2 (poly_3)	SVM3 (rbf_0.01)	Avg.	T1FFSVM
Ovarian Cancer	1	99.47	98.94	85.19	94.53	98.44
	2	98.94	97.89	82.58	93.14	100.0
	3	100.0	99.47	85.82	95.10	98.41
	4	100.0	97.91	80.03	92.65	98.41
	Avg.	99.6	98.55	83.41	93.85	**98.82**

In the experiment, we select six combinations of three SVMs for each dataset and repeat the same test shown in Table 4.4. Table 4.5a-c compare the average testing accuracies from the three individual SVMs with the ones from the fuzzy fusion model by combining the three SVMs. Each row in Table 4.5a-c lists the comparison result for one of the six tests.

Table 4.5(a) Three selected SVMs, their testing accuracies (%) in Table 4.1 and the average model accuracies in 4-fold cross-validation (Colon Tumor)

Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM
1	poly_1	83.9	poly_3	79.2	rbf_0.01	64.7	75.9	83.9	**87.1**
2	poly_1	83.9	poly_2	85.5	rbf_0.1	64.7	78.0	85.5	**88.8**
3	poly_4	75.5	rbf_1	82.2	rbf_0.01	64.7	74.1	82.2	**83.8**
4	poly_1	83.9	poly_3	79.2	poly_5	75.5	79.5	83.9	80.6
5	rbf_0.0001	64.7	rbf_0.01	64.7	rbf_1	82.2	70.5	82.2	77.7
6	poly_5	75.5	rbf_0.001	64.7	rbf_0.1	64.7	68.3	75.5	72.5
Avg.							**74.4**	**82.2**	**81.7**

Table 4.5(b) Three selected SVMs, their testing accuracies (%) in Table 4.1 and the average model accuracies in 4-fold cross-validation (Ovarian Cancer)

Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM
1	poly_1	100	poly_3	98.4	rbf_0.01	88.9	95.8	100.0	98.8
2	poly_1	100	poly_2	99.6	rbf_0.1	64.0	87.9	100.0	99.6
3	poly_4	97.6	rbf_1	64.0	rbf_0.01	88.9	83.5	97.6	**98.0**
4	poly_1	100	poly_3	98.4	poly_5	95.7	98.0	100.0	99.2
5	rbf_0.0001	89.7	rbf_0.01	88.9	rbf_1	64.0	80.9	89.7	**92.2**
6	poly_5	95.7	rbf_0.001	98.8	rbf_0.1	64.0	86.2	98.8	98.4
Avg.							**88.7**	**97.7**	**97.7**

Table 4.5(c) Three selected SVMs, their testing accuracies (%) in Table 4.1 and the average model accuracies in 4-fold cross-validation (Breast Cancer)

Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM
1	poly_1	96.65	poly_3	93.43	rbf_0.01	96.50	95.53	96.65	96.05
2	poly_1	96.65	poly_2	92.70	rbf_0.1	95.33	94.89	96.65	96.35
3	poly_4	93.29	rbf_1	87.45	rbf_0.01	96.50	92.41	96.50	92.54
4	poly_1	96.65	poly_3	93.43	poly_5	93.14	94.41	96.65	94.44
5	rbf_0.0001	96.06	rbf_0.01	96.50	rbf_1	87.45	93.34	96.50	94.88
6	poly_5	93.14	rbf_0.001	96.79	rbf_0.1	95.33	95.09	96.79	96.20
Avg.							94.28	96.62	95.08

4.4.4 Experimental Analysis

From Table 4.5, we can see that in all the six tests, the fuzzy fusion model performs better than the average of three individual SVM classifiers. One more important result is that in some tests, the fuzzy SVM classifier fusion model outperforms the best of its three composing individual SVM classifiers and achieves higher accuracies. The following lists the detailed analysis about the experimental results.

1) Colon tumor dataset: In three of the six tests (Tests 1-3), the fusion model outperforms the best individual SVM classifiers. Even if one or two of the three SVMs

have poor accuracy (64.7%), the fusion model can still achieve high accuracies (87.1%, 88.8%, 83.8%). For example, in Test 1, the testing accuracies of the three SVMs are: 83.9%, 79.22% and 64.7%, while the accuracy from the fusion model is 87.1%. This is a good example to demonstrate that different SVM classifiers can complement each other in the SVM fusion system to achieve a better performance than any of the individual SVM classifiers. In Test 4, and 5, the fusion model doesn't beat the best of the three individual SVM classifiers though it achieves better performance than the average and the second best. The possible reason is that the three SVMs have the same type of the kernel function and the only difference lies in the parameters of the kernel. The three SVMs with a same kernel type may work similarly for the same data and don't have too much information to complement. In Test 6, the fusion model doesn't achieve a better performance than its best individual composing classifier either because two of composing classifiers are not accurate enough or diverse enough.

2) Ovarian cancer dataset: In five of the six tests (Tests 2-6), the fusion model achieves a better performance or a similar performance to its composing individual classifiers. Even if composing one very poor classifier (64.0%), the fusion model still performs well. Test 5 is also an excellent instance to show that the ensemble approach by combining several classifiers could achieve higher performance than any of its individual classifiers (fusion model: 92.2%, three SVMs: 89.7%, 88.9% and 64.0%).

3) Breast cancer dataset: In three of the six tests (Tests 1, 2, 6), the fusion model achieves a similar performance to its composing individual classifiers. In the other three tests, the fusion model doesn't combine the classifiers well and doesn't have a better performance than its composing classifiers. The possible reasons are:

1) The defined FLS or fuzzy MFs are not suitable for the dataset.

(2) Individual classifiers behave similarly each other and are not able to provide much complementary information regarding to the data examples to be classified.

(3) Accuracy measure for classifier performance might not be good for the problem.

Chapter 5 Optimization of Fuzzy Classifier Fusion Model

The MFs in the fuzzy SVM fusion model in Chapter 4 are defined intuitively and manually according to the classification experience. The shapes or the positions of MFs may not be optimal. Genetic Algorithms (GAs) provide robust search and learning capabilities in complex space ideally for tuning optimal MFs and discovering optimal FLS accordingly.

5.1 Genetic Algorithms (GAs)

GAs are optimization algorithms which are inspired by natural evolution. The basic idea is to maintain a population of chromosomes over time through a process of variation and competition (Goldberg, 1989).

The optimization process of GAs starts off with an initial population of chromosomes and advances toward better chromosome by applying genetic operators. The common genetic operators include: *selection*, which selects chromosomes from the previous population based on their fitness to reproduce the next generation; *crossover*, which creates new chromosomes from parts of parents; and *mutation*, which introduces variation into the population by changing selected genes of chromosomes. The process of selection, recombination and mutation is repeated iteratively, generation after generation, until either the required fitness is met or the user-defined number of iterations is reached. A *fitness* or *objective function* must be provided for the problem to be solved. The best chromosome in the final population contains the optimal or near optimal solution to the problem. There are various schemas for each genetic operator,

such as uniform, one-point, two-point or multi-point crossover, random or Gaussian mutation, and roulette wheel or tournament selection.

5.2 Genetic Fuzzy Systems (GFS)

Fuzzy captures uncertainties by defining linguistic fuzzy sets with fuzzy membership functions (MFs) and reasoning fuzzy rules in a rigorous mathematical discipline. However, the success of designing a fuzzy logic system (FLS) largely relies on high-performance fuzzy MFs and fuzzy rules to interpret the expert knowledge. When lack of human expert, rather than choosing fuzzy MFs or defining fuzzy rules in a manual trail-and-error manner, we may seek the assistant from a learning process. GAs provide robust search and learning capabilities in complex space and GA-based fuzzy systems are able to learn and search fuzzy MFs or fuzzy rules efficiently.

A genetic fuzzy system (GFS) is basically a fuzzy system augmented by a learning process based on a genetic algorithm (Magdalena *et al*., 2004). The techniques of adapting fuzzy logic systems with GAs have been exploited by many researchers and genetic fuzzy rule-based systems have been addressed in plenty of papers, where GAs learn or train or tune different components of fuzzy logic systems (Magdalena *et al*., 2004). For instance, some genetic fuzzy rule-based systems may learn and determine the number of IF-THEN fuzzy rules from all possible rules (Herrera *et al*., 1995; Karr, 1991). Other genetic fuzzy systems may tune MFs of a given fuzzy rule set, such as tuning positions or shapes of MFs (Homaifar and McCormick, 1995; Park and Kandel, 1994; Cordon and Herrera, 1997).

5.3 Genetic Fuzzy SVM Fusion Model

The genetic fuzzy SVM fusion model constructed in Figure 5.1 adds one more phase than the basic architecture in Figure 4.3. This additional phase (Phase II) is used to tune the MFs using GAs to obtain the optimal fuzzy MFs.

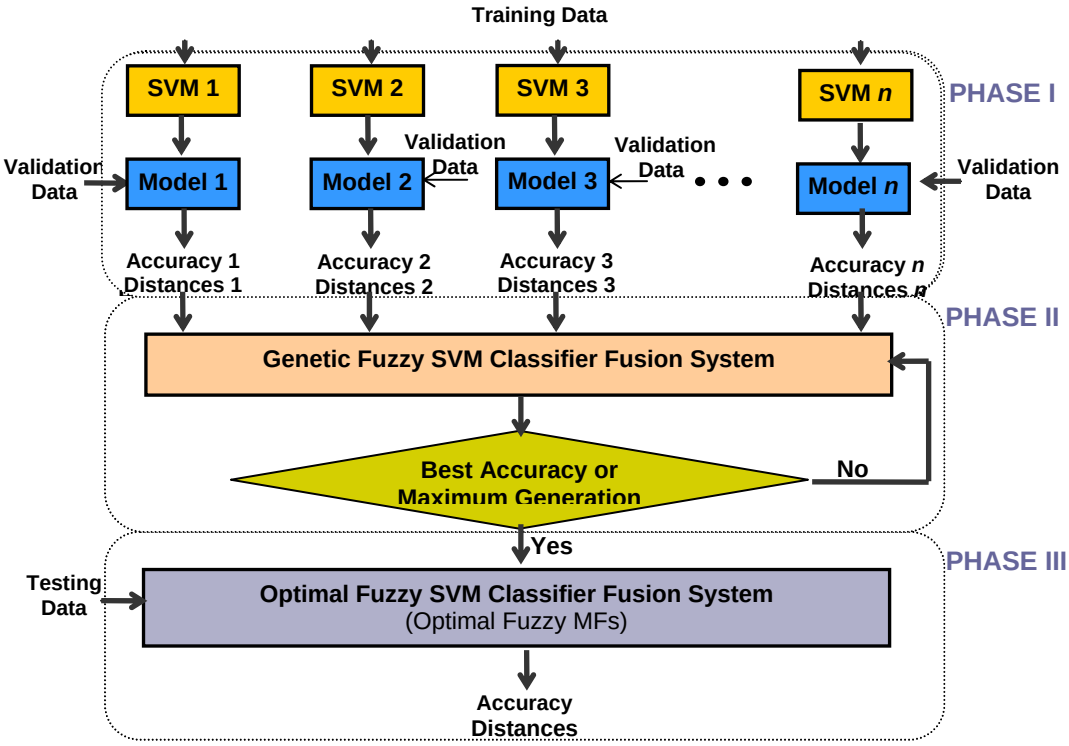


Figure 5.1 Genetic fuzzy SVM classifier fusion model (GFFSVM)

The system is divided into three phases. Three different datasets are used in the system: training data, validation data, and testing data. The training data and validation data are used to construct the fuzzy fusion model. The testing data are merely used to test the performance of the model. In phase I, the training data are trained by different SVMs. And the validation data are classified to obtain individual SVM accuracies and distances

of validation data examples to the SVM hyperplanes. In phase II, a genetic fuzzy fusion model is constructed and the fuzzy MF parameters in the model are tuned according to the accuracies of composite classifiers using *m*-fold cross validation data. By the end of the phase, an optimal fuzzy fusion system has been adapted. Finally, in Phase III, testing data examples are classified using different SVM classifiers and then distance information are plugged into the tuned optimal fuzzy fusion system in Phase II to obtain the final decision. The model accuracies of testing data can be calculated and compared with the testing accuracies from individual base SVM classifiers to estimate the performance of the model.

5.3.1 Tuning Fuzzy SVM Fusion Model

The FLS is similar to the one defined in Figure 4.3 except that the MFs of both input and output variables are not fixed. The control parameters of the MFs will be tuned by GAs. The tuning system is constructed in a cross-validation manner. There are *m* groups of different validation datasets if *m*-fold cross-validation is applied. The objective function of the GAs is to maximize the average model accuracy of all *m* validation data.

The fuzzy rule base, inference engine, system output, and defuzzification are similarly defined as in Chapter 4. The following sections will focus on the difference between, emphasizing on how to tune the MFs using GAs.

5.3.2 Fuzzy Input and Output MFs

All the memberships are defined as triangles shown in Figure 5.2. Again, each accuracy input is represented by two fuzzy sets: low and high. We use two control parameters to determine the shape of each accuracy fuzzy MF. For the fuzzy set “Low”, the two control parameters are labeled as L_l and L_r. At L_l and below, an accuracy input

fully belongs to the fuzzy set "Low". At L_r and above, an accuracy input doesn't belong to the set "Low" at all. Between L_l and L_r, the membership value decreases between 1 and 0 linearly. The membership of the fuzzy set "High" is just similar but the membership value is increased linearly between the two control parameters H_l and H_r. The positions of the control points (L_l, L_r, H_l, H_r) of the memberships will be adjusted by the GA. The initial values of the two left parameters L_l and H_l in the first population of the GA are set to the minimum SVM accuracy of validation data, and two right parameters L_r and H_r are set to the maximum accuracy. Initializing the membership parameters this way can provide the following benefits comparing with initializing the parameters with random numbers within their ranges:

- The MFs still cover all the possible accuracy inputs;
- The MFs are more sensitive to the changes of accuracy inputs;
- The convergence of the genetic process to the optimal values is faster;
- The standard deviation can be kept at a lower level during the tuning process.

Each distance input also has two fuzzy sets: negative and positive. The MFs of the distance fuzzy sets are not fixed. Each membership has two control points (N_l and N_r for "Negative" MF; P_l and P_r for "Positive" MF) used to tune the MFs by the GA. Whenever the distance of a data example is less than N_l, its membership value of the "Negative" fuzzy set is 1. At N_r and above, the distance input doesn't belong to the set "Negative". The membership of "Positive" set is similar.

The output value is defined by 64 fuzzy sets. Here an isosceles triangle is used to represent each linguistic fuzzy set. The positions of the triangles are not fixed and will be tuned by the GA. Since we use the centroid of an isosceles triangle to represent the output

value of each fuzzy rule, only one control point for each MF is needed. We also considered a more complex MF like an arbitrary triangle for the output which needs three control parameters for each MF. Two reasons keep us away from the idea. Generally, a large number of control parameters results in a better model accuracy on the training data, but the data may be overfitting which leads to a poor generalization ability. In addition, more control parameters require more computation complexity to tune and converge to an optimal solution.

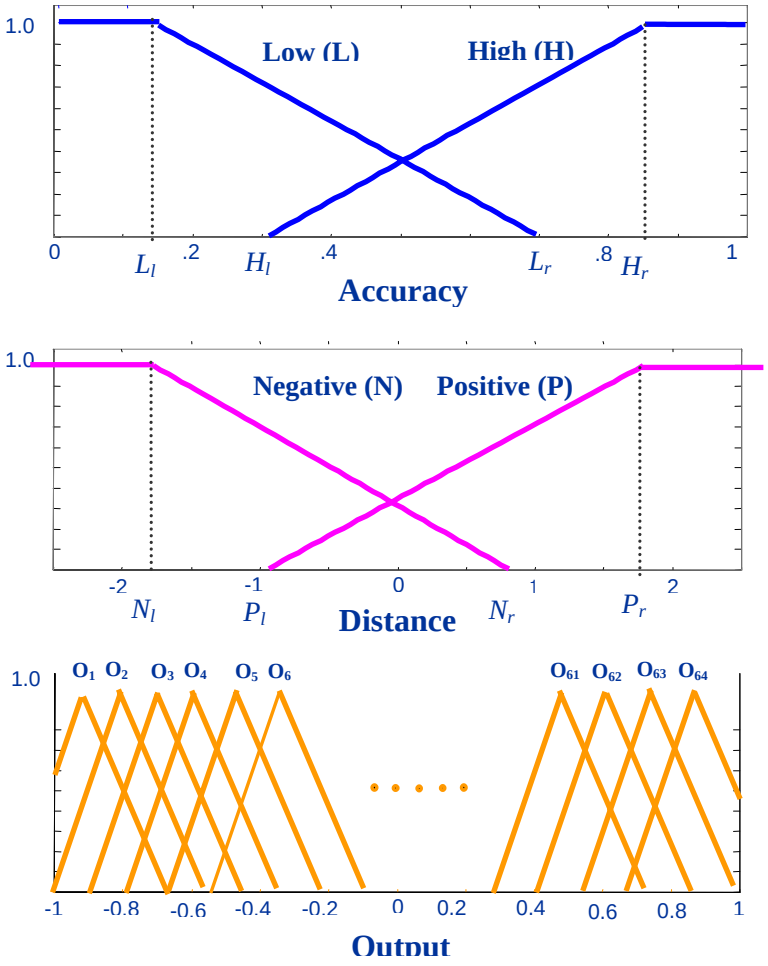


Figure 5.2 The MFs of GFFSVM

In Chapter 4, we discussed that 64 fuzzy rules can be grouped into 16 groups based on the similarity of the fuzzy rules. The rules in a same group have a similar <accuracy, distance> combination from three SVM classifiers as discussed before. Although the rules in a group are similar, we still set their outcome fuzzy sets in the consequence parts of the rules differently and leave the GAs to optimize the MFs of the fuzzy sets for the purpose of searching the MFs in a larger and more flexible space. However, we also consider the similarity and define the same possible ranges for the positions of the fuzzy MFs which fall into in a same group.

5.3.3 Tuning the MFs by GAs

In general, there are two ways to tune fuzzy MFs in general: Pittsburgh approach and Michigan approach (Magdalena *et al*., 2004). Pittsburgh approach is to represent an entire fuzzy rule set as a chromosome and maintain a population of candidate rule sets using genetic operations to produce new generations of rule sets (Smith, 1980). Michigan approach is to represent an individual rule as a chromosome and the whole rule set is represented by the entire population (Holland and Reitman, 1978). We apply Pittsburgh approach to tune the MFs.

5.3.4 Components of a Chromosome

The GA is real-coded. Binary coded GAs can be less efficient. Each chromosome or individual is composed of the 72 membership control parameters. In Figure 5.2 we can see, accuracy has two fuzzy sets: “Low” and “High”. Each is represented by a membership function controlled by two parameters. Likewise, distance has two fuzzy sets: “Negative” and “Positive”, controlled by two parameters for each set as well. Therefore, to tune the input MFs, eight genes are required in each chromosome. Since

only one control parameter is needed to determine the MF of each output fuzzy set and there are 64 output fuzzy sets, we need 64 genes in a chromosome to tune the output MFs, one for each fuzzy set. Totally, we need 72 genes in each chromosome to tune and develop all the MFs by the GA. Figure 5.3 shows the component genes of a chromosome. In summary, each chromosome consists of 72 membership parameters: 4 for tuning accuracy MFs, another 4 for distance MFs, and the rest 64 for the output MFs.

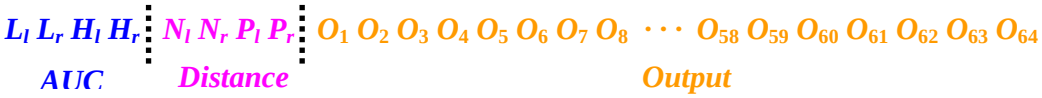


Figure 5.3 Components of a chromosome

As we mentioned earlier, in the initial population, accuracy membership parameters are set to the minimum and maximum SVM accuracy of *m*-fold validation data for left and right control points to ensure a fast convergence of the genetic process. Distance membership parameters are initialized to -1 and +1 for the left and right control points. Each output membership parameter is initialized as a random number within its range. In our fuzzy system, different fuzzy rules have different output fuzzy sets. The entire 64 fuzzy rules and the corresponding 64 output fuzzy sets are categorized into 16 groups based on the similarity of the rules. Here we set a same data range for the fuzzy sets falling in a same group.

In addition, we also consider the meaning of a fuzzy rule when determining its output range. For example, if one rule is defined like:

Rule i: IF a_1 is H and a_2 is H and a_3 is H and d_1 is N and d_2 is N and d_3 is N, THEN g_i is O_i.

where H denotes “High” and N denotes “Negative” fuzzy set.

The rule implies that all three SVMs have “High” accuracies and classify a data example in the “Negative” class. In this case, the centroid of the triangle for the fuzzy set O_i should reside at the left side of the output MFs in Figure 5.2. In other words, the widest range of the fuzzy set O_i should be [-1, 0).

Setting the range of an output membership parameter based on the actual meaning of the fuzzy rule can not only make the fuzzy rule meaningful, but also prevent overtraining or overfitting effectively. If all the output parameters are set to their widest possible range [-1, 1], although a higher fitness value might be achieved, it doesn’t necessarily mean that the tuned model has better performance. In reality, the testing accuracy could be poor when the testing data are applied on the learned fuzzy system. The followings are the details of the GA.

5.3.5 Fitness of the GAs

The fuzzy system is constructed using *m*-fold cross-validation data. Thus, the fitness of the GA is defined to maximize the average accuracy of *m* groups of validation data by processing the fuzzy system *m* times with the same MFs defined in a chromosome. The fitness function is formulized as given:

$$\overline{A} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{N} \left(\left| Z_{ij} + \hat{Z}_{ij} \right| / 2 \right)}{m} \tag{5.1}$$

where *m* is the number of fold of cross validation, *N* is the number of data examples in one set of validation data, $\hat{Z}_{ij}$ is the system output of the *j*th data example in the *i*th cross-validation data, and Z_{ij} is the desired output of the *j*th data example in the *i*th cross-validation data.

5.3.6 Selection and Elitism

Selection schema is standard proportional selection (also referred to as roulette-wheel selection). A chromosome, which contains 72 membership parameters, is selected from the population for the next generation in a way that is proportional to its fitness or the average accuracy of validation data. The higher the accuracy, the greater the chance it will be selected. However, it is not guaranteed that the best combination of MFs goes to the next generation. For this reason, we also apply elitism strategy: the best fuzzy MFs, with which the highest accuracy in the population can be achieved, are copied into the next generation directly without any modification. Elitism can improve the performance of the GA dramatically since it always keeps the best solutions to date.

5.3.7 Crossover Operator

There exist several crossover operators, such as one-point crossover, two-point crossover, multi-point crossover, and uniform crossover. One or multiple point crossover selects one or more crossover points to exchange genes of two intermediate chromosomes to reproduce offspring. Uniform crossover works in a different way that does not select crossover points and allows every locus a potential crossover point. Each gene of offspring chromosomes inherits from either of two parents randomly with a probability of 50%. In this study, we use uniform crossover. Uniform crossover is believed to outperform one or multiple crossover in many applications (Spears and De Jong, 1991]. Figure 5.4 illustrates the different crossover methods.

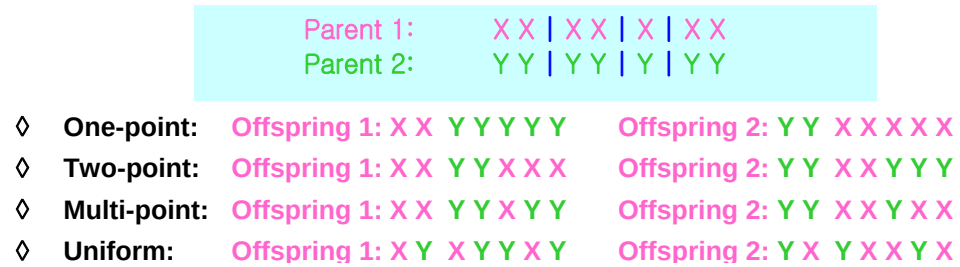


Figure 5.4 Crossover algorithms

5.3.8 Mutation Operator

We use Gaussian mutation in the system. Gaussian mutation modifies each gene of a chromosome by adding a Gaussian distributed random number with a mean of zero to it (Herrera *et al*., 1995). If the updated gene falls outside of the allowed range, the new gene value is cut and set to the boundary value. Gaussian mutation allows smaller alterations to happen more often than larger ones. It is more feasible for many applications than other mutation schemes, such as uniform mutation, which replaces a selected gene with a random number within its range. In the system, the standard deviation of the membership parameters for accuracy, distance and output are set differently since their MFs have the different domains.

5.4 Experiments on Biomedical Data

5.4.1 Experimental Method

The model construction and testing are based on cross-validation manner. Given a dataset *S*, it is first divided into *n* subsets. Each subset is treated as a group of testing data and all the other subsets together form a group of training data. We classify individual SVMs *n* times (*n*-fold cross validation) on each testing dataset. The classification results

in this stage will not be used in the genetic fuzzy fusion model construction in Phase II in Figure 5.1 but used to assess the performance of the model in Phase III in the figure. The accuracy of the testing data from individual SVM classifiers will be compared with the accuracy from the model in Phase III. Since we use *n*-fold cross validation, the performance of the model is estimated by the average of *n* accuracies from *n* different testing datasets.

The input data in Phase II are prepared as follows. Each training dataset above is further divided into *m* subsets (*m*-fold cross validation): one subset as a group of validation data and all the other subsets as a group of training data in the next level. The validation data are treated as the validation data in the model. All *m* group datasets are trained and classified by individual SVMs. The accuracies of validation data from three selected individual SVM classifiers are three accuracy inputs of the genetic fuzzy model. Three distances of each validation data example are the three distance inputs of the model. The model accuracy can be calculated for one validation dataset based on whether the defuzzified output of each data example is equal to the real desired output of the data example. The average accuracy of all *m* different validation datasets will be the fitness value of the genetic fuzzy model.

After we tune the MFs using the GAs, the optimal fusion model is established with the best fuzzy MFs. Testing data are then fed into the constructed optimal model and the decision can be made on the testing data. When the testing data are classified using the tuned optimal fusion model, each accuracy input is the average accuracy of *m* validation accuracies.

5.4.2 Experimental Environments

All the data are classified using SVM^{Light} software (Joachims, 1999) in Phase I. The generalization parameter *C* of individual SVM classifiers is set to 1 when the data are trained. The genetic fuzzy fusion system has been implemented in C. The parameter settings for the GA are as follows: crossover probability of 70%, generation of 200, and population size of 3000. The standard deviations of Gaussian mutation for accuracy, distance and output membership parameters are set to 2, 0.001 and 0.01 respectively. All the code is executed under a PC window system with P4-2.80GHz of CPU and 768MB of RAM.

5.4.3 Experimental Results

Colon tumor dataset and ovarian cancer dataset introduced in Section 4.4 are applied on the genetic fuzzy classifier fusion model. All the datasets are tested by 4-fold cross validation (n=4) and the genetic fuzzy fusion model is constructed by dividing each training dataset further into 3-fold and each fold forms one validation dataset (m=3).

As the experiment shown in Section 4.4, individual SVM classifiers are trained first and the training and testing accuracies are displayed in Table 4.1. The testing accuracies in this table will be compared with the accuracies of the proposed genetic fuzzy classifier fusion model later.

Each of the four training subsets in Table 4.1 is divided into three folds as three validation datasets in Phase II. Three validation datasets coming from the same training dataset will be used in the genetic fuzzy system to tune the fuzzy MFs. To clarify the idea, Table 5.1 shows the SVM accuracies of the validation data coming from the 1st fold training dataset in Table 4.1a for colon tumor dataset.

Table 5.1 Accuracies of 1st fold validation data for Colon Tumor Data (3-fold cross validation, m=3)

Kernels		Training Accuracy (%)				Validation Accuracy (%)			
Polynomial	*d*	1	2	3	Avg.	1	2	3	Avg.
poly_1	1	90.00	90.32	90.32	90.21	93.75	80.00	93.33	89.03
poly_2	2	96.67	96.77	100.0	97.81	93.75	80.00	93.33	89.03
poly_3	3	100.0	100.0	100.0	100.0	93.75	73.33	80.00	82.36
poly_4	4	100.0	100.0	100.0	100.0	93.75	73.33	80.00	82.36
poly_5	5	100.0	100.0	100.0	100.0	87.50	73.33	80.00	80.28
poly_6	6	100.0	100.0	100.0	100.0	81.25	73.33	80.00	78.19
poly_8	8	100.0	100.0	100.0	100.0	75.00	73.33	80.00	76.11
poly_10	10	100.0	100.0	100.0	100.0	68.75	80.00	73.33	74.03
RBF	σ								
rbf_0.0001	0.0001	66.67	64.52	64.52	65.24	62.50	66.67	66.67	65.28
rbf_0.001	0.001	66.67	64.52	64.52	65.24	62.50	66.67	66.67	65.28
rbf_0.01	0.01	66.67	64.52	64.52	65.24	62.50	66.67	66.67	65.28
rbf_0.1	0.1	66.67	64.52	64.52	65.24	62.50	66.67	66.67	65.28
rbf_1	1	93.33	93.55	93.55	93.48	87.50	86.67	86.67	86.95
rbf_10	10	100.0	100.0	100.0	100.0	62.50	66.67	66.67	65.28

Once we have accuracies and data example distances from individual SVM classifiers ready, the next step is to select three base SVM classifiers to be combined in the genetic fuzzy fusion system. Table 5.2 shows one selection of three SVM classifiers and their corresponding accuracies from Table 5.1. These three SVM classifiers will be used to construct the genetic fuzzy system.

Table 5.2 Validation accuracies (%) of three selected SVMs for Colon Tumor

Fold	SVM1 (poly_1)	SVM2 (poly_3)	SVM3 (rbf_0.01)	Avg.
1	93.75	93.75	62.50	83.33
2	80.00	73.33	66.67	73.33
3	93.33	80.00	66.67	80.00
Avg.	89.03	82.36	65.28	78.89

After the genetic fuzzy system is adapted and optimized, the fuzzy model has been developed with the best-fit fuzzy MFs tuned. The testing data in Table 4.1 can then be used in the Phase III to evaluate the performance of the tuned model. The model accuracy on the testing dataset by combining the three selected SVM classifiers in Table 5.2 is shown in the first row of Table 5.3. Since we use *n*-fold (*n*=4) cross validation and have *n* different testing datasets, we test the fuzzy model *n* times by combining the same three individual SVM classifiers and obtain *n* model accuracies for *n* different testing datasets. The model accuracy by combining the three SVMs in Table 5.2 is just one of *n* cases. Table 5.3 shows all *n* (*n*=4) model accuracies of *n* testing data by combining the same three SVM classifiers in Table 5.2. In this table, GFFSVM denotes the proposed genetic fuzzy fusion model. The columns 2-4 show the average accuracies of 3-fold validation data. They are treated as accuracy inputs when the optimized model applies on the testing data to obtain the model accuracies. The last column shows the model accuracies.

Table 5.3 Model accuracies by combining three selected SVMs in Table 5.2

Fold	SVM1 (poly_1)	SVM2 (poly_3)	SVM3 (rbf_0.01)	GFFSVM
1	89.03	82.36	65.28	81.25
2	71.67	76.39	67.36	93.75
3	80.83	85.14	63.89	86.67
4	82.92	68.06	61.67	93.33
Avg.	81.11	77.99	64.55	88.75

Table 5.3 shows one combination of three SVM classifiers. We have tested six combinations of three selected SVMs and the results are shown in Table 5.4. Table 5.3 is

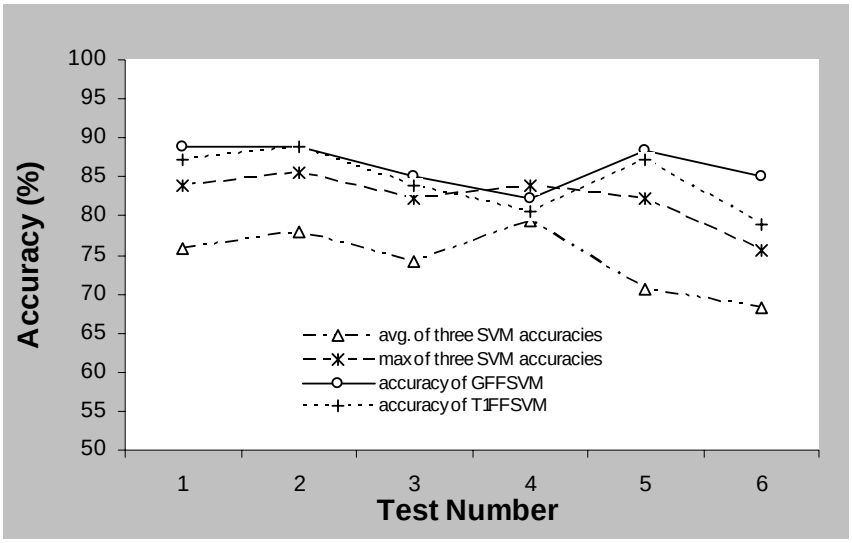
Test 1 in Table 5.4. The first six columns in Table 5.4 are the three SVM classifiers followed by their testing accuracies from Table 4.1a. The next columns show the average accuracies and the maximum accuracies of the three SVM classifiers. The last column shows the model accuracies by combining the corresponding three base SVM classifiers in the columns 2-7.

Table 5.4 Accuracies of individual SVM classifiers, fuzzy fusion model (T1FFSVM), and genetic fuzzy fusion model (GFFSVM) (Colon Tumor)

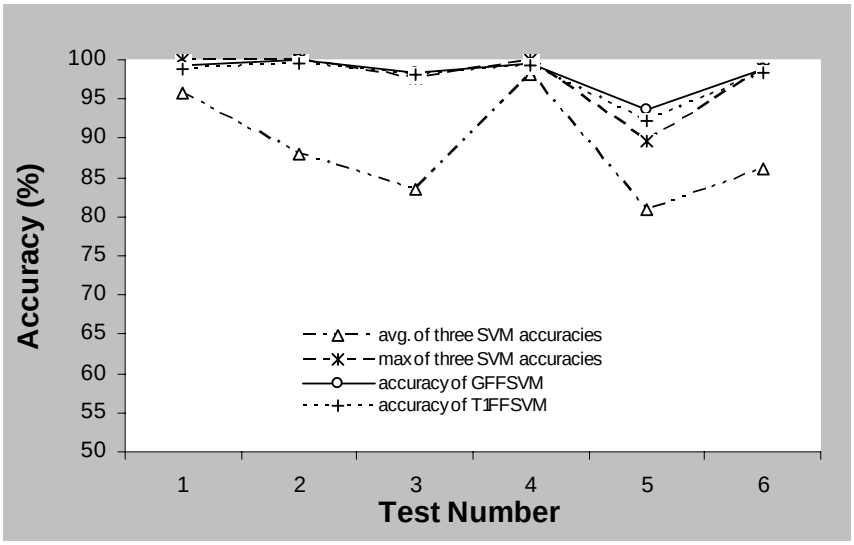
Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM	GFFSVM
1	poly_1	83.9	poly_3	79.2	rbf_0.01	64.7	75.9	83.9	**87.1**	**88.8**
2	poly_1	83.9	poly_2	85.5	rbf_0.1	64.7	78.0	85.5	**88.8**	**88.8**
3	poly_4	75.5	rbf_1	82.2	rbf_0.01	64.7	74.1	82.2	**83.8**	**85.1**
4	poly_1	83.9	poly_3	79.2	poly_5	75.5	79.5	83.9	80.6	82.3
5	rbf_0.0001	64.7	rbf_0.01	64.7	rbf_1	82.2	70.5	82.2	**87.1**	**88.4**
6	poly_5	75.5	rbf_0.001	64.7	rbf_0.1	64.7	68.3	75.5	**78.8**	**85.0**
Avg.							**74.4**	**82.2**	**84.3**	**86.1**

For ovarian cancer data, we repeat exactly the same process above. Table 5.5 lists the results of six tests by combining three selected base SVM classifiers. Just like colon cancer data, ovarian cancer data are also tested in 4-fold cross validation (n=4) and the model is constructed in 3-fold cross validation (m=3). Table 5.5 shows the average model accuracies and individual SVM accuracies of 4-fold testing data.

The curves in Figure 5.5 illustrate the six tests shown in Table 5.4 and Table 5.5 by comparing individual SVM classifiers with the ones from the genetic fuzzy fusion model, and from the fuzzy fusion model designed in Chapter 4 for the both datasets.



(a) Colon Tumor



(b) Ovarian Cancer

Figure 5.5 Comparison of individual SVM classifiers with fuzzy fusion model and genetic fuzzy fusion model

Table 5.5 Accuracies of individual SVM classifiers, fuzzy fusion model (T1FFSVM), and genetic fuzzy fusion model (GFFSVM) (Ovarian Cancer)

Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM	GFFSVM
1	poly_1	100	poly_3	98.4	rbf_0.01	88.9	95.8	100.0	98.8	99.2
2	poly_1	100	poly_2	99.6	rbf_0.1	64.0	87.9	100.0	99.6	100.0
3	poly_4	97.6	rbf_1	64.0	rbf_0.01	88.9	83.5	97.6	**98.0**	**98.4**
4	poly_1	100	poly_3	98.4	poly_5	95.7	98.0	100.0	99.2	99.6
5	rbf_0.0001	89.7	rbf_0.01	88.9	rbf_1	64.0	80.9	89.7	**92.2**	**93.7**
6	poly_5	95.7	rbf_0.001	98.8	rbf_0.1	64.0	86.2	98.8	98.4	98.8
Avg.							**88.7**	**97.7**	**97.7**	**98.3**

5.4.4 Performance Analysis

Since we use cross validation to build the model and also use cross validation to test the model, the performance should be fairly estimated.

From Table 5.4, Table 5.5 and Figure 5.5, we can see that the proposed genetic fuzzy classifier fusion model performs better than any of its composing individual SVM classifiers in most cases. In more detail, during five of six tests on colon tumor data in Table 5.4 and four of six tests on ovarian cancer data in Table 5.5, the composite classifier can beat the best individual SVM classifier or has the same performance as the best. The model in the rest tests has very close performance to the best SVM. Therefore, we have good enough reason to conclude that the proposed genetic fuzzy fusion model performs reliable and robust generalization ability comparing with its individual SVM classifiers.

In particular, for Test 1, 5, 6 on colon tumor data in Table 5.4 and Test 5 on ovarian cancer data in Table 5.5, the accuracies of three selected SVM classifiers are relatively low (<83.9%, 79.2%, 64.7%>, <64.7%, 64.7%, 82.2>, <79.5%, 64.7%, 64.7%>, <89.7%, 88.9%, 64.0%>), while the genetic fuzzy classifier fusion model can achieve higher

accuracies of 88.8%, 88.4%, 85.0%, and 93.7% by combining the three relatively poor individual SVM classifiers. They are good examples to show that weak individual classifiers are able to complement with each other well in the composite classifier to reach a higher performance.

Table 5.4, Table 5.5 and Figure 5.5 also demonstrate that the genetic fuzzy classifier fusion model outperforms the fuzzy classifier fusion model proposed in Chapter 4 in general. The genetic fuzzy system is capable of adapting optimal fuzzy MFs and therefore optimal fuzzy classifier fusion model. In more detail, for all the tests on colon tumor data and ovarian cancer data, the genetic fuzzy fusion model achieves better performance than the fuzzy fusion model. This makes much sense since the initial parameters in the genetic fuzzy fusion system are set in a way as if the initial stage of the genetic fuzzy fusion model is same as the fuzzy fusion model defined in Chapter 4. The experimental results also show that GAs are powerful and robust techniques to tune and adapt an optimal or near optimal FLS.

Chapter 6 Classifier evaluation and AUC-based classifier fusion model

Traditionally, evaluation of a classifier performance is done by minimizing an estimation of a generalization error or some other related measures. However, accuracy or error of a classifier is not necessarily a good one. In fact, when the data distribution is strongly unbalanced, accuracy may be misleading since the *all-positive* or *all-negative* classifier may achieve a very good classification rate. Accuracy cannot maintain ranking data examples to be classified as well, which is often desirable for researchers.

6.1 ROC Analysis for Binary Classification

ROC curve can be a good alternative for model evaluation, since they can make the difference between errors on positive or negative examples. AUC, which is the area under an ROC curve, has been shown to be a better measure than accuracy when assessing classifier performances (Ling *et al.*, 2003; Huang and Ling, 2005).

6.1.1 Confusion Matrix

For a binary classification problem, there are four possible outcomes for a testing example: *true positive (TP)* if the example is positive and classified as positive, *false negative (FN)* if the example is positive but classified as negative, *true negative (TN)* if the example is negative and classified as negative, and *false positive (FP)* if the example is negative but classified as positive. For a set of testing examples, a confusion matrix can be constructed as shown in Table 6.1 to illustrate the classification distribution of testing examples (Fawcett, 2003).

Table 6.1 Confusion matrix of a classifier

	Predicated Positive	Predicated Negative	Total Examples
Actual Positive	(*TP*) True Positives	(*FN*) False Negatives	N^+
Actual Negative	(*FP*) False Positives	(*TN*) True Negatives	N^-

If the number of positives and negatives are denoted by N^+ and N^- respectively, where $N^+ = TP + FN$ and $N^- = FP + TN$, then, the true positive rate (*TPR*) and the false positive rate (*FPR*) are defined as follows.

$$TPR = \frac{TP}{N^+} \qquad FPR = \frac{FP}{N^-} \tag{6.1}$$

And the classification accuracy is defined as:

$$Accuray = \frac{TP + TN}{N^+ + N^-} \tag{6.2}$$

6.1.2 ROC Graph

An ROC graph depicts the trade-off between *TPR* and *FPR* (ROC curve plots *TPR* on the *Y* axis against the *FPR* on the *X* axis) as shown in Figure 6.1. It can also be viewed as a tradeoff between benefits (true positives) and costs (false positives) (Fawcett, 2003).

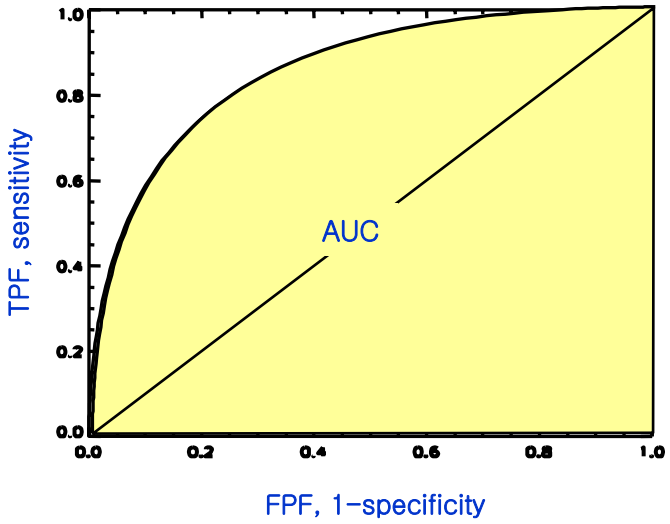


Figure 6.1 An ROC curve

Some classifiers are *discrete classifiers*, such as decision trees, which are designed to produce only a positive/negative class label on each example. Such a discrete classifier can only generate a single confusion matrix. Thus only a single point can be drawn in the ROC graph. However, some other classifiers such as SVMs or neural networks yield a numeric value on each example representing the degree to which an example belongs to a class. This type of classifiers is called *probabilistic classifiers* (Fawcett, 2003). When various decision thresholds are applied to probabilistic classifier outputs to classify data examples (positive if the classifier output is above the threshold, and negative otherwise), different confusion matrixes can be obtained. Therefore, a series of points can be plotted in a ROC plane with pairs of {*FPR*, *TPR*} as their coordinates. Each threshold results in one point on the ROC curve representing the classifier which is generated by using this threshold as the cutoff point. The points (0,0) and (1,1) in the ROC curve represent two default classifiers which always produce negative and positive outputs respectively. We

may think the default classifiers are generated by using $+\infty$ and $-\infty$ as the thresholds. Therefore, an ROC curve of a probabilistic classifier can be viewed as an aggregation of classifiers from all possible decision thresholds (Qin, 2005).

6.1.3 AUC: the Area under the Curve of an ROC

An ROC curve is a two-dimensional depiction of classifier performance and its quality can be summarized in one value by calculating the area under the ROC curve (AUC). AUC represents the probability that one classifier ranks a randomly chosen positive example higher than a randomly chosen negative example (Fawcett, 2003). In other words, AUC depicts the quality of ranking of data examples by the classifier (Qin, 2005). According to Hand (Hand and Till, 2001), AUC can be simply calculated in the following formula:

$$AUC = \frac{\sum_{i=1}^{N^+} r_i - N^+(N^+ + 1)/2}{N^+ N^-} \tag{6.3}$$

where r_i denotes the rank of ith positive example in the ranking list if we arrange the classification results of data examples in ascending order.

To illustrate how to use this formula to calculate the AUC value of a classifier, let's take one example. Suppose we apply two classifiers on a set of testing data with 10 examples. Table 6.2 shows the classification results from the two classifiers and the ranks of data examples in the each classification result. The AUC values of the two classifiers can be calculated as follows:

$$AUC_1 = \frac{(5 + 7 + 8 + 9 + 10) - 5 * 6/2}{5 * 5} = 0.96$$

$$Accuracy_1 = 0.8$$

$$AUC_2 = \frac{(1+6+7+8+9) - 5*6/2}{5*5} = 0.64$$

$$Accuracy_2 = 0.8$$

Table 6.2 Classification ranks of data examples from two classifiers

Classifier 1	-	-	-	-	+	-	+	+	+	+
Rank					5		7	8	9	10
Classifier 2	+	-	-	-	-	+	+	+	+	-
Rank	1					6	7	8	9	

We can see that two classifiers have the same accuracy but different AUC values. Because AUC can maintain the ranks of data examples to be classified and distinguish the different errors from positive or negative data examples, AUC is a better measure than accuracy which makes no difference between positive errors or negative errors.

6.2 Genetic Fuzzy SVM Classifier Fusion Based on AUC

When we construct the classifier fusion models in Chapter 4 and Chapter 5, we use classification accuracy as one of the system parameters to estimate the classifier performance. This accuracy metric might not be appropriate if the data distribution is unbalanced or skewed. Considering AUC is a better classifier performance measure, we may replace the accuracy input in the fusion model with AUC value so that the model may be further improved.

The AUC based classifier fusion model is constructed in the similar way to create the genetic fuzzy classifier fusion model described in Figure 5.1 in Chapter 5 with one

exception: accuracy in the figure will be replaced by AUC. The system also has three phases. In Phase I, training data are trained by different individual SVM classifiers. Validation data are classified to obtain AUC values of individual SVM classifiers and distances of validation data examples to SVM hyperplanes. In Phase II, a GFS is constructed and the fuzzy MFs are tuned by GAs in cross validation manner. Finally, in phase III, testing data are fed into the optimal fuzzy fusion system to make the final decision and get the model AUC and accuracy.

The fuzzy MFs of AUC inputs are defined similar to the ones of accuracy inputs in Chapter 5. Each AUC is also represented by two fuzzy sets: low and high. All the input and output MFs in the system are not fixed and will be tuned by GAs. Corresponding to the change, the accuracies in the fuzzy rules in Chapter 5 will also be replaced by AUC.

6.3 Experiments on the AUC-based Classifier Fusion Model

The experiments reported in Chapter 5 are repeated by using AUC classifier performance measure in the model. Table 6.3 and 6.4 show the experimental results on Colon Tumor data and Ovarian Cancer data respectively.

Table 6.3 AUC and accuracies of individual SVM classifiers, genetic fuzzy fusion model (GFFSVM) in Chapter 5, and AUC-based fuzzy fusion model (Colon Tumor)

Test	SVM1	SVM2	SVM3	Avg. AUC	Max AUC	Avg. Accuracy	Max Accuracy	GFFSVM	Fusion AUC	Fusion Accuracy
1	poly_1	poly_3	rbf_0.01	0.86	0.89	75.9	83.9	**88.8**	**0.92**	**88.8**
2	poly_1	poly_2	rbf_0.1	0.88	0.89	78.0	85.5	**88.8**	0.89	**90.0**
3	poly_4	rbf_1	rbf_0.01	0.85	0.88	74.1	82.2	**85.1**	**0.91**	**85.1**
4	poly_1	poly_3	poly_5	0.83	0.89	79.5	83.9	82.3	**0.91**	**85.2**
5	rbf_0.0001	rbf_0.01	rbf_1	0.87	0.88	70.5	82.2	**88.4**	0.88	**88.8**
6	poly_5	rbf_0.001	rbf_0.1	0.85	0.88	68.3	75.5	**85.0**	**0.89**	**83.3**
Avg.				**0.86**	**0.89**	**74.4**	**82.2**	**86.4**	**0.90**	**86.9**

Table 6.4 AUC and accuracies of individual SVM classifiers, genetic fuzzy fusion model (GFFSVM) in Chapter 5, and AUC-based fuzzy fusion model (Ovarian Cancer)

Test	SVM1	SVM2	SVM3	Avg. AUC	Max AUC	Avg. Accuracy	Max Accuracy	GFFSVM	Fusion AUC	Fusion Accuracy
1	poly_1	poly_3	rbf_0.01	0.99	1.00	95.8	100.0	99.2	1.00	100.0
2	poly_1	poly_2	rbf_0.1	0.95	1.00	87.9	100.0	100.0	1.00	100.0
3	poly_4	rbf_1	rbf_0.01	0.94	1.00	83.5	97.6	**98.4**	1.00	**99.7**
4	poly_1	poly_3	poly_5	0.99	1.00	98.0	100.0	99.6	1.00	100.0
5	rbf_0.0001	rbf_0.01	rbf_1	0.94	0.98	80.9	89.7	**93.7**	**0.99**	**99.2**
6	poly_5	rbf_0.001	rbf_0.1	0.99	1.00	86.2	98.8	98.8	1.00	**99.9**
Avg.				**0.97**	**1.00**	**88.7**	**97.7**	**98.3**	**1.00**	**99.8**

From Table 6.3 and 6.4 we may see that the proposed AUC-based classifier fusion model demonstrates stable and robust classification capabilities. It outperforms the best of three individual SVMs in terms of both AUC and accuracy. It indicates that AUC-based classifier fusion model not only achieves nice AUC performance, but also excellent accuracy at the same time.

From Table 6.3 on colon tumor experiments we can see, in all the six tests, the model accuracy is better than the best accuracy of three SVM classifiers. For example, in Table 5, the model achieves 88% accuracy, but the best accuracy of the three classifiers is only about 82%. For all the six tests, the model achieves better AUC values or the same values as the best AUC of the three individual SVM classifiers.

For Table 6.4 on ovarian cancer experiments we can see, in all the six tests, the model accuracy is better than the best accuracy or similar to the best. For Test 5, the model achieves 99% accuracy, but the best accuracy is only about 90%. For all the six tests, the model also achieves a better AUC or the same AUC as the best AUC of the three individual SVM classifiers.

We can conclude that the classifier fusion model that optimizes AUC measure not only achieves nice AUC performance, but also excellent accuracy as well (Ling and Zhang, 2002). The genetic fuzzy SVM fusion model based on AUC measure produces a combined classifier with the best AUC naturally because of the properties of AUC. This means that the accurate ranking of data examples is maintained and it provides researchers more interpretation of data examples than mere positive or negative classification results.

Chapter 7 Combining SVM Classifiers Using Type-2 Fuzzy Logic

In a FLS, uncertainties are handled by using fuzzy sets, which are represented by MFs. Type-1 fuzzy sets handle the uncertainties by using *precise* and *crisp* MFs and membership grades of type-1 fuzzy sets are any crisp values in [0, 1]. But once the MFs are determined, all the uncertainties will disappear (Liang and Mendel, 2001; Zeng and Liu, 2006). Unlike in the type-1 FLS, the MFs of type-2 fuzzy sets themselves are fuzzy such that membership grades of type-2 fuzzy sets are fuzzy sets in [0, 1] instead of crisp values in [0, 1]. Type-2 fuzzy sets are especially useful to handle the situations where the shapes, positions or other parameters of MFs are uncertain.

To better handle the uncertainties in classification data and in MFs, type-2 fuzzy sets and FLS are applied to construct the SVM fusion model. General type-2 FLS is computationally difficult but the process can be simplified a lot if type-2 fuzzy sets are defined as interval type-2 fuzzy sets. Therefore, we construct one SVM fusion model using interval type-2 FLS in order to achieve better performance than type-1 SVM fuzzy fusion model.

7.1 Type-1 vs. Type-2 Fuzzy Sets and MFs

When we have difficulty to measure the exact value of one object, we know we can apply type-1 fuzzy sets and FLS to solve the problem and obtain a fuzzy set, which is generally more reasonable than a crisp set (Karnik and Mendel, 1998). This is the exact motivation that the concept of fuzzy logic was introduced by Zadeh (Zadeh, 1965). The theory of fuzzy logic has been applied to many real applications to handle the uncertainties associated with FLS inputs and outputs. However, the ability of type-1

fuzzy sets and FLS to handle uncertainties is limited because type-1 fuzzy sets handle uncertainties by defining *precise* and *crisp* MFs. Once the MFs are determined, all the uncertainties will be precisely described (John, 1998; Mendel, 2001; Karnik and Mendel, 1998). It is much likely that the defined type-1 MFs will not be the best choice for a real application. Therefore, the way to define MFs in type-1 FLS restricts the ability of type-1 fuzzy sets and FLS to model and minimize the effect of uncertainties.

A type-2 FLS has the potential to outperform a type-1 FLS because a type-2 fuzzy set is represented by more parameters than a type-1 fuzzy set is (Mendel, 2001). Unlike a type-1 fuzzy set whose MF is defined precisely, the MF of a type-2 fuzzy set is defined *blurrily* and consisted of a set of admissible type-1 MFs called the footprint of uncertainty (FOU) of a type-2 MF (Liang *et al*., 2000; Liang and Mendel, 2000). As a result, the membership grade of a type-2 fuzzy set is a fuzzy set in [0, 1] instead of a crisp value in [0, 1] for a type-1 fuzzy set. Once a type-2 MF is reduced to a type-1 MF, the blurriness of the MF will no longer exist and it becomes a precise MF as defined in a type-1 FLS. Therefore, type-2 fuzzy logic can be viewed as a generalization of type-1 fuzzy logic, and accordingly, type-2 fuzzy sets and MFs can also be considered as an extension of type-1 fuzzy sets and MFs with the increased ability to handle uncertainties existing in MFs and FLS. In the following sections, we will briefly introduce the theory of type-2 fuzzy sets and FLS.

7.2 Structure of Type-2 FLS

The distinction between type-1 and type-2 fuzzy logic is associated with the nature of the MFs, but the structure of a type-2 FLS shown in Figure 7.1 looks similar to that of a type-1 FLS. It also includes four components in general: fuzzifier, fuzzy rule base, fuzzy

inference engine, and output processor. Type-2 fuzzy rules remain the same as in the type-1 FLS and also are expressed in IF-THEN statements but with the antecedents and/or the consequences of rules replaced by type-2 fuzzy sets.

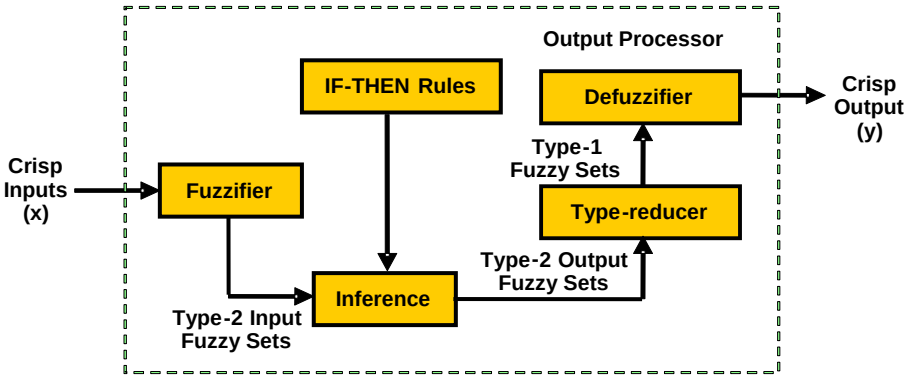


Figure 7.1 The structure of a type-2 FLS

However, there is one significant difference between a type-2 FLS and a type-1 FLS. That is, the output processor of a type-2 FLS needs one additional step: *type-reducer* just before defuzzifier. Type reduction is necessary because, unlike a type-1 FLS whose output is a type-1 fuzzy set, the output of a type-2 FLS is a type-2 fuzzy set which has to be reduced to a type-1 fuzzy set before defuzzifier is able to reduce the type-1 output fuzzy set further into a crisp value.

7.3 Interval Type-2 FLS

7.3.1 Interval Type-2 Fuzzy Sets and MFs

As an extension of an original type-1 fuzzy set, a type-2 fuzzy set, denoted by $\tilde{A}$, is characterized by a type-2 MF $\mu_{\tilde{A}}(x,u)$, where $x \in X$ and $u \in J_x \subseteq [0,1]$, and can be

expressed as (Mendel and John, 2002):

$$\widetilde{A} = \int_{x \in X} \int_{u \in J_x} \mu_{\widetilde{A}}(x,u)/(x,u) \qquad J_x \subseteq [0, 1] \tag{7.1}$$

where $\int \int$ denotes union over all admissible x and u. $J_x \subseteq [0,1]$ is called *primary membership* of x. The primary membership defines the domain of a type-2 fuzzy membership grade. Unlike a type-1 fuzzy set whose membership grade is a crisp value in [0, 1], the membership grade of a type-2 fuzzy set is a type-1 fuzzy set in [0, 1] (Liang, 2000).

The uncertainty in the primary memberships of a type-2 fuzzy set is called FOU, which is defined as the union of all primary memberships and consists of a bounded region (Mendel and John, 2002) as shown in Fig. 1a. The upper bound is called *upper MF* and denoted by $\overline{\mu}_{\widetilde{A}}(x)$. Similarly, the lower bound is called *lower MF* and denoted by $\underline{\mu}_{\widetilde{A}}(x)$. The both upper and lower MFs are type-1 MFs and denote the maximum and minimum membership grade of FOU respectively (Liang, 2000). FOU provides a way to describe the uncertainties in shapes or other parameters of MFs. When the uncertainties of MFs disappear, the type-2 fuzzy sets reduce to type-1 fuzzy sets whose MFs can be precisely determined.

Corresponding to each primary membership, there is a *secondary membership*. The secondary membership is also defined in [0, 1] and used to represent the possibilities of the primary membership (Liang and Mendel, 2000). Many functions can be chosen to define a secondary MF. The name that is used to describe the entire type-2 MF is relevant to the name of its secondary MFs (Mendel, 2001). For example, when the secondary MF are an interval set as shown in Figure 7.2b, that is, when the secondary membership grade are either zero or one ($f_x(u)$=1, $\forall u \in J_x \subseteq [0,1]$ and $f_x(u)$=0, $\forall u \notin J_x$, where J_x is the

primary MF of x and $f_x(u)$ is the secondary MF), the type-2 fuzzy set is called an interval type-2 fuzzy set (Liang, 2000; Liang and Mendel, 2000). Interval secondary memberships reflect a uniform uncertainty at the primary memberships (Mendel, 2001). Due to its characteristic of unity, the interval set can be represented just by its right and left end-points which are located on the upper and lower MFs in FOU. One cardinal advantage of defining interval secondary MFs is that the computation of a type-2 FLS can be significantly simplified (Mendel, 2001; Liang and Mendel, 2000). Specifically, when type-2 fuzzy sets are interval fuzzy sets, the process of type reduction in a type-2 FLS can be calculated in reasonable computational complexity. General type-2 FLS is computationally intensive. Because of its excellent ability to handle uncertainties in real applications and relevantly simple calculation, the interval type-2 FLS has succeeded to solve many real problems on decision making (Liang, 2000; Hagras, 2004; Liang and Mendel, 2001; Zeng and Liu, 2006; Aguero and Vargas, 2005). In the following sections, we will only consider interval type-2 FLS.

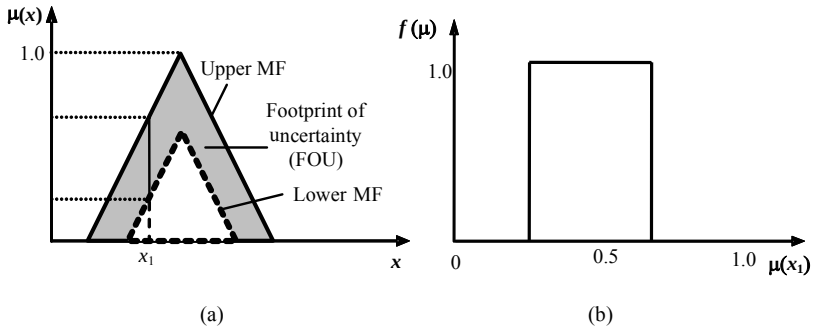


Figure 7.2 (a) FOU and interval type-2 MFs; (b) the secondary memberships are all equal to 1 for an interval type-1 set corresponding to $x = x_1$.

7.3.2 Fuzzy Inference of an Interval Type-2 FLS

Fuzzy inference engine combines fired fuzzy rules and maps crisp inputs into type-2 output fuzzy sets. The antecedents in fuzzy rules are connected by using the *meet* operation, the firing strength of the input fuzzy sets are combined with output fuzzy sets using the extended *sup-star composition*, and multiple rules are combined using the *join* operation (Mendel, 2001).

In our interval type-2 FLS, we use the meet operation under product t-norm, so the firing strength is an interval type-1 set (Mendel, 2001):

$$f^i(\mathbf{x}) = [\,\underline{f}^i(\mathbf{x}),\ \bar{f}^i(\mathbf{x})\,] = [\,\underline{f}^i,\ \bar{f}^i\,] \tag{7.2}$$

where $\underline{f}^i(\mathbf{x})$ and $\bar{f}^i(\mathbf{x})$ can be written in (7.3) and (7.4), where * denotes the product operation:

$$\underline{f}^i(x) = \underline{\mu}_{\tilde{F}_1^i}(x_1) * ..... * \underline{\mu}_{\tilde{F}_p^i}(x_p) \tag{7.3}$$

$$\bar{f}^i(x) = \bar{\mu}_{\tilde{F}_1^i}(x_1) * ..... * \bar{\mu}_{\tilde{F}_p^i}(x_p) \tag{7.4}$$

7.3.3 Type Reduction of an Interval Type-1 FLS

The results from the inference engine are type-2 fuzzy sets. They must be reduced to type-1 fuzzy sets so that defuzzifier can be applied to generate crisp outputs. Type-reducer is the additional step different from the type-1 FLS. In this study, *center-of-sets* (COS) type reducer algorithm developed by Karnik and Mendel (Mendel, 2001; Karnik *et al*., 1999] is used since it requires reasonable computational complexity comparing with expensive centroid type reducer.

COS type reducer can be divided into two phases. The first phase is to calculate the centroids of type-2 fuzzy rule consequences. The second phase is to calculate the reduced fuzzy sets.

1) Calculation of the centroids of rule consequences: Suppose the output of an interval type-2 FLS is represented by interval type-2 fuzzy sets $\widetilde{G}^t$, where $t = 1, \ldots, T$, T is the number of output fuzzy sets. In this stage, we will calculate the centroids of all the T output fuzzy sets first which will be used in the next phase to calculate the reduced fuzzy sets. The centroid of ith output fuzzy set y^t is a type-1 interval set and can be expressed in the following formula (Mendel, 2001; Karnik *et al*., 1999):

$$y^t = [y_l^t, y_r^t] = \int_{\theta_1 \in J_{y1}} \cdots \int_{\theta_z \in J_{yZ}} 1 \Big/ \frac{\sum_{z=1}^{Z} y_z \theta_z}{\sum_{z=1}^{Z} \theta_z} \tag{7.5}$$

where y_l^t and y_r^t are the leftmost and rightmost point of y^t.

Figure 7.3 displays Karnik-Mendel iterative algorithm (Mendel, 2001; Karnik *et al*., 1999) to compute the rightmost point y_r^t for each type-2 output fuzzy set, where Z is the number of discretised points for each output fuzzy set, $J_{y_z} = [L_z, R_z]$, $h_z = (L_z + R_z)/2$, and $\Delta_z = (R_z - L_z)/2$, $z = 1 \ldots Z$. Figure 7.4 shows how to calculate h_z, L_z, R_z and Δ_z needed by the algorithm. The leftmost point y_l^t can be calculated in the similar way except at step 4 in Fig. 3, set $\theta_z = h_z + \Delta_z$ when $z \le e$ and $\theta_z = h_z - \Delta_z$ when $z > e+1$. It has been proved that this iterative procedure can converge in at most Z iterations to find y_l^t or y_r^t (Mendel, 2001).

1. Arrange y_z in ascending order with $y_1 \le y_2 \le \ldots \le y_Z$;
2. Set $\theta_z = h_z$ for z = 1, …, Z; Compute $y' = \dfrac{\sum_{z=1}^{Z} y_z \theta_z}{\sum_{z=1}^{Z} \theta_z}$
3. Find $e \in [1, Z\text{-}1]$ such that $y_e \le y' \le y_{e+1}$;
4. Set $\theta_z = h_z - \Delta_z$ for $z \le e$ and $\theta_z = h_z + \Delta_z$ for $z > e$; Compute $y'' = \dfrac{\sum_{z=1}^{Z} y_z \theta_z}{\sum_{z=1}^{Z} \theta_z}$;
5. Stop if $y'' == y'$; Otherwise, set $y' = y''$ and return to step 3;

Figure 7.3 Karnik-Mendel iterative procedure to calculate y_r^t.

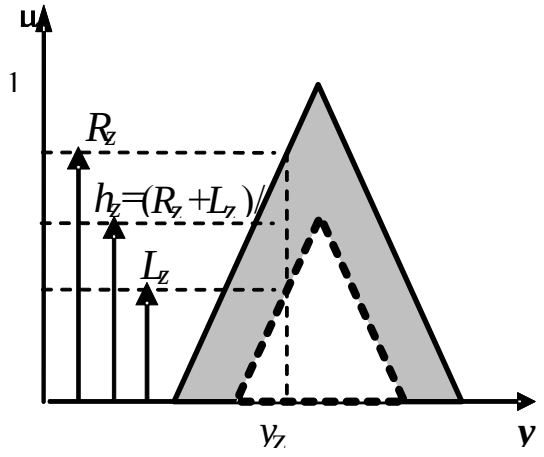


Figure 7.4 Calculation of the parameters needed by each y_z in the procedure in Figure 7.3

2) Calculation of the reduced type-1 fuzzy sets: To compute a type-reduced set, it is sufficient to compute its upper and lower bounds of the reduced set y_l and y_r, which can be expressed as follows:

$$y_l = \frac{\sum_{i=1}^{M} f_l^i y_l^i}{\sum_{i=1}^{M} f_l^i}, \qquad y_r = \frac{\sum_{i=1}^{M} f_r^i y_r^i}{\sum_{i=1}^{M} f_r^i} \tag{7.6}$$

where f_l^i and y_l^i are the firing strength and the centroid of the output fuzzy set of ith rule (i = 1, …, M) associated with y_l respectively; f_r^i and y_r^i are the firing strength and the centroid of the output fuzzy set of ith rule (i = 1, …, M) associated with y_r.

To compute y_r, we use the iterative procedure in Figure 7.5 developed in (Mendel, 2001; Liang and Mendel, 2000). y_l can be computed in the similar way by setting $f_r^i = \bar{f}^i$ for $i \le R$ and $f_r^i = \underline{f}^i$ for $i > R$.

1. Arrange the pre-calculated y_r^i from Fig. 3 in ascending order; i.e. $y_r^1 \le y_r^2 \le \ldots \le y_r^M$;
2. Set $f_r^i = (\underline{f}^i + \bar{f}^i)/2$ for i = 1, …, M; Compute y_r' using Equation (11);
3. Find $R \in [1, M\text{-}1]$ such that $y_r^R \le y_r' \le y_r^{R+1}$;
4. Set $f_r^i = \underline{f}^i$ for $i \le R$ and $f_r^i = \bar{f}^i$ for $i > R$; Compute y_r'' using Equation (11);
5. Stop if $y_r'' == y_r'$; Otherwise, set $y_r' = y_r''$ and return to step 3;

Figure 7.5 Iterative procedure to calculate y_r

The iterative procedure is proved to converge in no more than M iterations to compute y_r and no more than M iterations to find y_l (Liang and Mendel, 2000).

7.3.4 Defuzzification

The final output of type-2 FLS is set to the average of y_r and y_l:

$$y(x) = \frac{y_l + y_r}{2} \tag{7.7}$$

7.4 Type-2 Fuzzy SVM Classifier Fusion Model

7.4.1 Input and Output Interval Type-2 Fuzzy MFs

When we consider combining the classification results from three SVMs, SVM accuracies and distances of one data example to three SVM hyperplanes are reasonable inputs for the type-2 FLS. The output of the type-2 FLS is the combined classification decision and defined in [-1, 1]. If the defuzzified crisp output from the FLS is less than zero, we will consider the data example in the negative class. Otherwise, it is in the positive class. So, we have three accuracy inputs and three distance inputs and one output. All the inputs and the output are defined as interval type-2 fuzzy sets as shown in Figure 7.6.

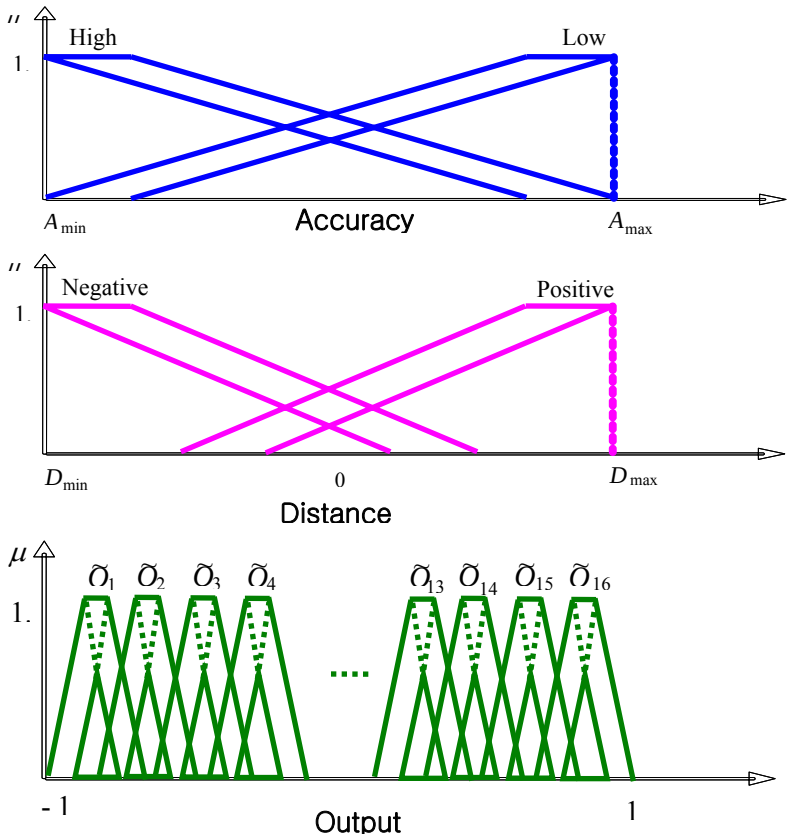


Figure 7.6 Interval type-2 fuzzy MFs

Each accuracy input is represented by two interval type-2 fuzzy sets: high and low, and each distance input is also described by two interval type-2 fuzzy sets: positive and negative. The output is represented by seven interval type-2 fuzzy sets.

1) Type-2 MFs for accuracy inputs: As in type-1 fuzzy fusion model, each accuracy input is represented by two type-2 fuzzy sets. We set the minimum and maximum accuracies as the two bounds of the domains of the accuracy MFs such that the MFs are more sensitive to the changes of accuracy inputs than the ones by setting the domain to the entire range [0%, 100%]. The admissible ranges of the interval type-2 MFs are set to around 2%.

2) Type-2 MFs for distance inputs: Each distance input is represented by two type-2 fuzzy sets. The base point of "Negative" triangle MF is set to around +0.5 and the base point of "Positive" MF is set to around -0.5 as shown in Figure 7.6b. The admissible range of the interval type-2 MFs is set to 0.1~0.3.

3) Type-2 MFs for the output: There are sixteen interval type-2 fuzzy sets to represent the output. The admissible range of the type-2 MFs is set to around 0.1.

7.4.2 Fuzzy Rule Base

Since the system has three accuracy inputs and three distance inputs from three SVM hyperplanes, and each accuracy and each distance have two possibilities respectively, there are 2 ^ 6 = 64 fuzzy rules in total. The *i*th rule is defined as follows (i = 1...64):

IF a_1 is $\widetilde{A}_1^i$ and a_2 is $\widetilde{A}_2^i$ and a_3 is $\widetilde{A}_3^i$ and d_1 is $\widetilde{D}_1^i$ and d_2 is $\widetilde{D}_2^i$ and d_3 is $\widetilde{D}_3^i$,

THEN g_i is $\widetilde{O}^i$ (i = 1...64).

where $\widetilde{A}_1^i$, $\widetilde{A}_2^i$ and $\widetilde{A}_3^i$ in {Low, High}, $\widetilde{D}_1^i$, $\widetilde{D}_2^i$ and $\widetilde{D}_3^i$ in {Negative, Positive}, and $\widetilde{O}^i$ in $\{\widetilde{O}_1, ..., \widetilde{O}_7\}$.

When we decide the consequences of the fuzzy rules, we consider both the accuracy inputs and the distance inputs of three SVMs in the real classification. If all the accuracies of three SVM are high and they all classify one data example as the positive distances, the consequence of the corresponding rule is the output fuzzy set $\widetilde{O}_{16}$, indicating the data example is more likely in the positive class. On the other hand, if all three accuracies are high and all three distances are negative, the consequence of the corresponding rule is the output fuzzy set $\widetilde{O}_1$, indicating the data example is more likely in the negative class. Based on the same consideration, we determine the consequences of other rules. The output fuzzy sets of the fuzzy rule consequences are assigned according to Table 4.1.

7.4.3 Fuzzy Inference and Output Processing

In the inference engine of the type-2 fuzzy SVM fusion model, we use the *meet* under the *product t-norm* operation and the *join* under the *maximum* operation and the extended sup-star composition. We use *center-of-sets* type-reducer algorithm described in Figure 7.3 and Figure 7.5 in Section 7.3 to reduce type-2 output fuzzy sets into type-1 sets. When we calculate the centroid of an output type-2 fuzzy set, we discretise the domain of the output type-2 fuzzy set into 50 points ($Z = 50$). After the type-reduction, the reduced type-1 fuzzy sets will be defuzzified to produce a crisp value in [-1, 1], the domain of the output. If the crisp output is less than zero, we consider the data example in the negative class. Otherwise, it belongs to the positive class.

In our experiments, the iterative procedure shown in Figure 7.3 to compute the centroids of the output type-2 fuzzy sets and the iterative procedure in Figure 7.5 to compute the two bounds of reduced type-1 sets usually converge in less than 5 iterations which are very fast though we set the discrete level to 50 ($Z = 50$) and the total number of rules is 64 ($M = 64$).

7.5 Experiments on Biomedical Cancer Data

The two datasets introduced in Chapter 5 from Kent Ridge Biomedical Data Set Repository (Li and Liu, 2003) have been used to estimate the performance of the type-2 fuzzy SVM fusion model. The data in Phase I of the model are classified using SVM^{light} (Joachims, 1999). The type-2 fuzzy SVM fusion system to combine three SVM classifiers has been implemented in C program. All the software is executed under a PC window system with P4-2.80 GHz of CPU and 768 MB of RAM.

7.5.1 Experimental Results

The selected SVM classifiers have been combined in the proposed type-2 fuzzy classifier fuzzy model for each dataset. And the model accuracies are compared with the testing accuracies in Table 4.2. Table 7.1a-b show the experimental results from the type-2 fuzzy classifier fuzzy model and the comparison results from the type-2 fuzzy fusion model, the individual classifiers, the type-1 based fuzzy fusion model in Chapter 4, and the genetic fuzzy classifier fusion model defined in Chapter 5. The comparison is also illustrated in Figure 7.7.

Table 7.1(a) Accuracies of individual SVM classifiers, fuzzy fusion model (T1FFSVM), genetic fuzzy fusion model (GFFSVM), and type-2 fuzzy fusion model (T2FFSVM) (Colon Tumor)

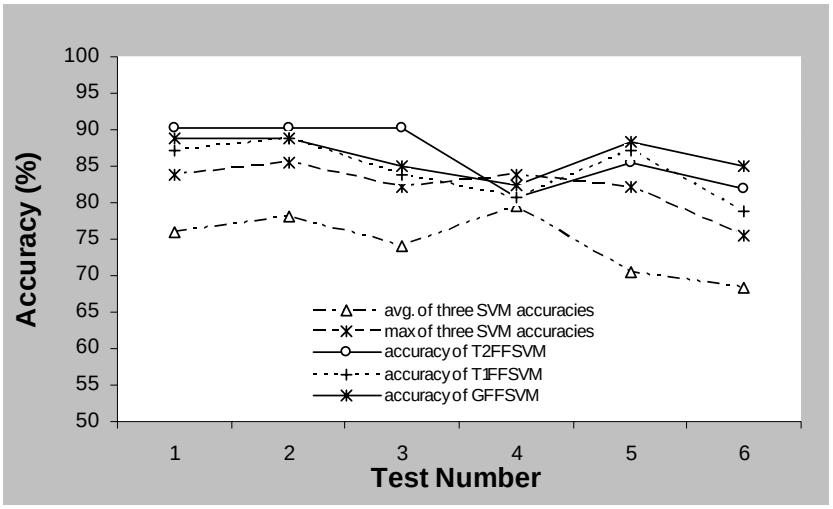
Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM	GFFSVM	T2FFSVM
1	poly_1	83.9	poly_3	79.2	rbf_0.01	64.7	75.9	83.9	**87.1**	**88.8**	**90.3**
2	poly_1	83.9	poly_2	85.5	rbf_0.1	64.7	78.0	85.5	**88.8**	**88.8**	**90.3**
3	poly_4	75.5	rbf_1	82.2	rbf_0.01	64.7	74.1	82.2	**83.8**	**85.1**	**90.3**
4	poly_1	83.9	poly_3	79.2	poly_5	75.5	79.5	83.9	80.6	82.3	80.6
5	rbf_0.0001	64.7	rbf_0.01	64.7	rbf_1	82.2	70.5	82.2	**87.1**	**88.4**	**85.5**
6	poly_5	75.5	rbf_0.001	64.7	rbf_0.1	64.7	68.3	75.5	**78.8**	**85.0**	**82.0**
Avg.							**74.4**	**82.2**	**84.3**	**86.1**	**85.9**

Table 7.1(b) Accuracies of individual SVM classifiers, fuzzy fusion model (T1FFSVM), genetic fuzzy fusion model (GFFSVM), and type-2 fuzzy fusion model (T2FFSVM) (Ovarian Cancer)

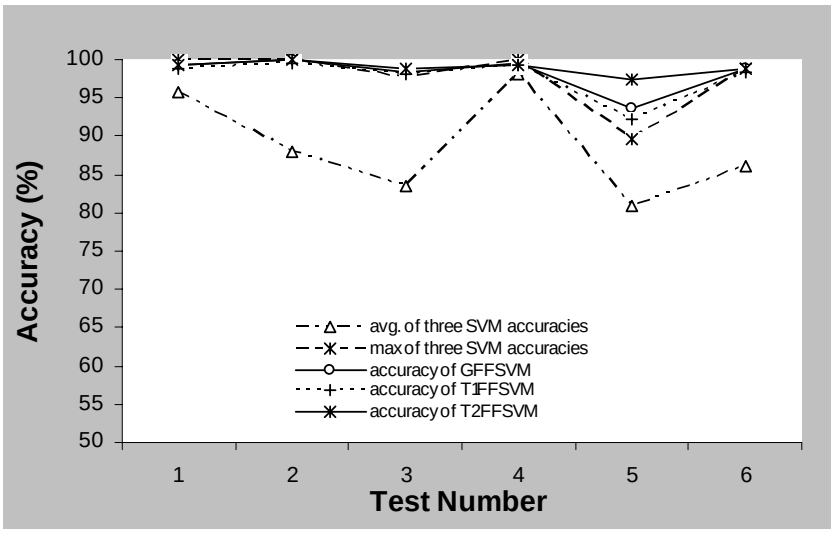
Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM	GFFSVM	T2FFSVM
1	poly_1	100	poly_3	98.4	rbf_0.01	88.9	95.8	100.0	98.8	99.2	99.2
2	poly_1	100	poly_2	99.6	rbf_0.1	64.0	87.9	100.0	99.6	100.0	**100.0**
3	poly_4	97.6	rbf_1	64.0	rbf_0.01	88.9	83.5	97.6	**98.0**	**98.4**	**98.8**
4	poly_1	100	poly_3	98.4	poly_5	95.7	98.0	100.0	99.2	99.6	99.2
5	rbf_0.0001	89.7	rbf_0.01	88.9	rbf_1	64.0	80.9	89.7	**92.2**	**93.7**	**97.3**
6	poly_5	95.7	rbf_0.001	98.8	rbf_0.1	64.0	86.2	98.8	98.4	98.8	98.8
Avg.							**88.7**	**97.7**	**97.7**	**98.3**	**98.9**

7.5.2 Performance Analysis

From Table 7.1 and Figure 7.7, we can see that in most test cases, the type-2 based SVM fusion model outperforms the best of its three composing individual SVMs and achieves higher accuracies. We can also see that in general, the type-2 based fusion model has the better performance than the type-1 based fusion model since the type-2 MFs are characterized by more parameters than type-1 MFs and thus the type-2 based fusion model is able to deal with uncertainties in a better way.



(a) Colon Tumor



(b) Ovarian Cancer

Figure 7.7 Comparison of individual SVM classifiers with fuzzy fusion model, genetic fuzzy fusion model, and type-2 fuzzy classifier fusion model

The following gives the detailed analysis about the experimental results from the type-2 based fusion model.

1) Colon tumor dataset: In five of the six tests (Tests 1-3 and Tests 5-6), the type-2 fusion model outperforms the best individual SVMs. Even if one or two of the three

SVMs have poor accuracy (64.7%), the type-2 fusion model can still achieve high accuracies (90.3%). For example, in Test 3, the testing accuracies of the three SVMs are: 75.5%, 82.2% and 64.7%, while the accuracy from the type-2 model is 90.3%. This is a good example to demonstrate that different SVM classifiers can complement each other in the type-2 SVM fusion system to achieve a better performance than any of the individual SVMs.

2) Ovarian cancer dataset: Again in most tests, the type-2 fusion model outperforms the best individual SVMs or achieves the same performance. For example, In Test 5, the testing accuracies of the three SVMs are: 89.7%, 88.9% and 64.0%, while the accuracy from the type-2 model achieves much higher accuracy of 97.3%.

7.5.3 Comparing with Other Classifier Fusion Methods

The proposed fuzzy classifier fusion models are also compared with the existing classifier fusion methods introduced in Chapter 3, including Majority Vote, Average, Product, Minimum, and Maximum of posterior probabilities on the colon tumor dataset. The comparison results are listed in Table 7.2. We may see from this table that the proposed fuzzy classifier fusion models outperform the most existing classifier fusion methods and demonstrate more stable and robust generalization abilities when comparing with the composing individual classifiers. Among other fusion methods, product and minimum achieve better performance than the other methods.

Table 7.2 Comparison of the fuzzy classifier fusion models with the existing fusion methods

Fusion Methods	Testing Cases						
	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Average
SVM1	poly_1	poly_1	poly_4	poly_1	rbf_0.0001	poly_5	
SVM2	poly_3	poly_2	rbf_1	poly_3	rbf_0.01	rbf_0.001	
SVM3	rbf_0.01	rbf_0.1	rbf_0.01	poly_5	rbf_1	rbf_0.1	
Avg.	75.9	78.0	74.1	79.5	70.5	68.3	74.4
Max	83.9	85.5	82.2	83.9	82.2	75.5	82.2
T1FFSVM	88.8	88.8	85.1	82.3	88.4	85.0	86.1
T2FFSVM	90.3	90.3	90.3	80.6	85.5	82.0	85.9
Average	77.8	76.0	66.3	69.7	64.7	63.0	69.6
Product	86.6	90.2	79.0	86.5	70.7	57.0	78.3
Majority Vote	77.8	76.0	66.3	71.3	65.6	63.0	70.0
Minimum	86.6	90.2	79.0	86.5	70.7	76.1	81.5
Maximum	82.4	85.5	73.0	82.4	71.1	71.3	77.6

Chapter 8 Case Study: Relations between Protein Structures and Sequences

8.1 Protein Structures

Proteins are polymers of amino acids that are synthesized from the gene regions of the genome. Figure 2.1 illustrates the process of a protein synthesis. Proteins fold hierarchically into four different levels of structures in three dimensions:

- ◊ *Primary Structure*: The amino acid sequence of residues in the peptide chain of a protein is called the primary structure.
- ◊ *Secondary Structure*: Due to hydrogen bonds between backbone atoms, local regions of the polypeptide fold into three stable secondary structure, including *alpha helices* and *beta strands*, which are located at the core of a protein, and *coils*, which sit in outer regions.
- ◊ *Tertiary Structure*: Secondary structural elements arrange themselves into the tertiary structure on the level of one whole polypeptide chain in three dimensions.
- ◊ *Quaternary Structure:* If there is more than one polypeptide chain in a complex protein, several peptide chains interact among themselves and develop the quaternary structure.

Knowing protein structures plays an essential role in understanding their biological functions. Scientist and researchers have put a lot of effort in determining protein structures. There are two main techniques to determine protein structures: X-ray crystallography, and Nuclear Magnetic Resonance (NMR). Most of the protein structures available in the PDB database are determined by X-ray crystallography. However, X-ray

crystallography is very expensive. The protein structures in the PDB database only accounts for a small part of the protein family.

The prediction of protein structures based on their primary sequences is one of the most critical research tasks. Multiple sequence alignment has been the main tool to analyze protein sequences and structures for decades. Many computational methods and tools for sequence alignment have been proposed, including ClustalW (Thompson *et al*., 1994), PSI-BLAST (Altschul *et al*., 1997), and etc. The basic idea using the sequence alignment tools is to find the proteins in the current protein database with the high sequence similarity with the target protein and predict the structure of the target protein based on the structures of the similar proteins. It's generally believed that proteins share similar structures if they have similar sequences.

Recent advances in machine learning, statistical learning, and graphical models provide more powerful algorithms and methods in understanding the correlation between protein sequences and structures.

8.2 Protein Sequence-Structure Relations

Protein residues might be mutated, inserted, or deleted during the evolutionary process. However, protein structures and functions are usually conserved. Similar sequences form similar structures. How about the other direction? Do the similar local structures share similar sequences as well? People usually believe that similar protein structures may not share high sequence similarity. This chapter will study this question using machine learning methods and the proposed classifier combination methods.

8.2.1 Generation of Protein Sequence Segments with Certain Structures

Proteins used in this study come from the Protein Sequence Culling Server (PISCES) database (Wang and Dunbrack, 2003), which contains 2290 proteins, among which no proteins share more than 25% sequence identities.

The protein sequence segments are generated by sliding windows with successive residues at the length range from 5 to 14. To study the sequence similarity problem, sequence segments with a desired secondary local structure are extracted from all 2,290 studied proteins. For instance, if we are interested in studying the relation between local structure of alpha helix with a size of 10 and all its corresponding sequences, the sequence segments on an alpha helix with the length of 10 are extracted from the protein sequence database. When we say a sequence segment with helix structure, we mean every amino acid of the sequence segment is located on the same alpha helix local structure, but the two neighbor residues of the sequence segment don't reside on a helix structure. We may represent this helix structure with a length of 10 as $\overline{H}HHHHHHHHHH\overline{H}$, where H denotes alpha helix, and $\overline{H}$ denotes non-helix. Before extracting the sequence segments, the secondary structures are first converted to three category classes based on the following method: *H*, *G*, and *I* to *H*; *B* and *E* to *E*; and all others to *C*. Table 8.1 shows the number of sequence segments extracted from the PISCES database which are satisfied with the certain local structure at the segment length range from 5 to 14, where *H*, *E*, and *C* denote alpha helix, beta strand, and coil structure respectively.

Table 8.1Number of sequence segments with the desired local structures

Window Size *n*	5	6	7	8	9	10	11	12	13	14
$\overline{H}H\cdots H\overline{H}$	932	1,074	1,183	1,167	1,233	1,283	1,266	1,047	1,061	922
$\overline{C}C\cdots C\overline{C}$	5,310	3,486	2,349	1,591	1,265	947	640	504	357	252
$\overline{E}E\cdots E\overline{E}$	3,369	3,064	1,980	1,448	923	589	401	231	166	119

From Table 8.1, we may see that the number of sequence segments with local structure of alpha helix is increased as the slide window size increases and gets its maximum value when the window size is 10. Then it is decreased. The number of sequence segments with local structure of beta strand or coil is decreased from the window size of 5.

8.2.2 Representation of Protein Sequence Segments

The sequence segments extracted are represented by their frequency profile defined in the HSSP (Homology-Derived Secondary Structure of Proteins) database (Sander and R. Schneider, 1991; Dodge *et al*., 1998). The HSSP frequency profile is based on a multiple alignment of the sequence and its structural homologues in the protein database. Each residue of a sequence in the HSSP is represented by the alignment occurrence frequency of each of 20 amino acids in that position. Thus, a sequence segment with the length of L is represented by $20L$ attributes, among which every 20 attributes are for one amino acid in the segment. For example, 200 attributes are used to represent a sequence segment with the length of 10.

8.2.3 Classification of Protein Sequence Segments

To study the sequence similarity problem given similar structures, binary classification methods are applied. Training or testing data are composed of two classes of sequence segments at a certain size: one class from one local structure, such as alpha helix, and the other class either from a different local structure, such as beta strand, coil, partial helix and partial strand, alpha helix with the two neighbors of the segment also in helix, or from all the other structures' combination. The following describes the scenarios considered in discovering whether similar structures share similar sequences:

- ◊ *Case 1 (Helix vs. All Others)*: One class of sequence segments comes from alpha helix structure and the other class comes from all the combinations of the rest other local structures. The purpose of this experiment is to discover whether the sequence segments with a certain structure are similar or have the significant different from sequence segments in other structures.
- ◊ *Case 2 (Helix vs. Helix with Helix Neighbors):* In the previous scenario, the class of sequence segments in the helix structure doesn't include the sequence segments which also have the helix structure but reside in a larger alpha helix structure. In other words, it doesn't include the case in which the neighbors of the sequence segments also have the helix structure. The purpose of this experiment is to discover whether these two classes of segment sequences are similar. If so, it's not proper for the above experiment to include the sequence segments which reside in a larger helix structure. Otherwise, the performance of the experiment would be degraded.

- ◊ *Case 3 (Helix vs. Strand):* One class of sequence segments comes from alpha helix structure and the other class comes from beta strand structure. The purpose of this experiment is to discover whether the sequence segments on helix structure have the significant different from sequence segments on the beta strand structure.
- ◊ *Case 4 (Helix vs. Coil):* One class of sequence segments comes from alpha helix structure and the other class comes from coil structure. The purpose of this experiment is to discover whether the sequence segments on helix structure have the significant different from sequence segments on coil structure.
- ◊ *Case 5 (Strand vs. Coil):* One class of sequence segments comes from beta strand structure and the other class comes from coil structure. The purpose of this experiment is to discover whether the sequence segments with strand structure have the significant different from sequence segments on coil structure.

The sequence segments with window size of 8, 9, and 10 are classified by SVM classifiers. The numbers of features of data examples (here are sequence segments) at the window size of 8, 9, and 10 are 160, 180, and 200 respectively since each residue of a segment is represented by 20 attributes according to the HSSP frequency profile.

8.2.4 Classification Results

The datasets are classified in 4-fold cross-validation (n=4) by SVM^{light} (Joachims, 1999). The regularization parameter C of the SVMs is set to 5 during the training. Two kinds of kernels: polynomial kernels and RBF kernels are applied. The degree of the polynomial kernels is set to 1...5, and the parameter σ of the RBF kernels is set to 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1},10^{0}, 10^{1}. Table 8.2 shows the testing accuracies from 4 folds on the "*Helix vs. Coil*" testing case with the segment length of 9. Tables 8.3-8.5 show the average

testing accuracies in 4-fold cross validation for all the testing cases with the segment length of 8, 9, and 10 respectively.

Table 8.2 Testing accuracy (%) in 4-fold cross-validation (helix vs. coil) (size =9)

Kernels	Testing Accuracy (%)				
	1	2	3	4	Avg.
poly_1	91.05	89.10	91.67	89.42	90.31
poly_2	89.62	87.50	90.71	89.10	89.23
poly_3	89.14	87.82	91.51	89.58	89.51
poly_4	88.50	88.78	91.19	88.94	89.35
poly_5	87.70	88.46	90.87	88.94	88.99
rbf_0.0001	50.64	50.64	50.64	50.64	50.64
rbf_0.001	90.58	88.94	91.03	90.71	90.32
rbf_0.01	91.69	91.03	92.47	90.22	91.35
rbf_0.1	92.33	89.74	91.99	91.03	91.27
rbf_1	84.82	83.33	85.90	82.05	84.03
rbf_10	51.28	51.12	51.28	50.64	51.08

Table 8.3 Average testing accuracy (%) in 4-fold cross-validation (size = 8)

Kernels	Testing Cases				
	Case 1 (H <-> All others)	Case 2 ($\bar{H}H..H\bar{H}$ <-> $HH.HH$)	Case 3 (H <-> E)	Case 4 (H <-> C)	Case 5 (E <-> C)
poly_1	71.98	67.06	83.44	88.98	88.78
poly_2	69.50	64.61	85.09	86.91	86.21
poly_3	69.41	64.82	85.32	86.33	86.51
poly_4	69.46	64.35	85.55	86.22	86.05
poly_5	69.24	64.05	85.55	85.97	85.13
rbf_0.0001	71.43	64.57	55.37	57.69	52.36
rbf_0.001	71.30	65.17	80.19	86.62	87.86
rbf_0.01	72.67	67.74	84.05	89.99	89.37
rbf_0.1	73.10	69.41	86.00	89.92	88.98
rbf_1	69.58	66.24	78.51	80.24	86.31
rbf_10	51.41	50.94	55.75	57.98	52.75

Table 8.4 Average testing accuracy (%) in 4-fold cross-validation (size = 9)

Kernels	Testing Cases				
	Case 1 (H <-> All others)	Case 2 ($\overline{H}H...H\overline{H}$ <-> $HH..HH$)	Case 3 (H <-> E)	Case 4 (H <-> C)	Case 5 (E <-> C)
poly_1	74.69	69.87	81.31	90.31	87.25
poly_2	73.84	67.64	86.69	89.23	85.06
poly_3	74.74	68.62	86.46	89.51	85.42
poly_4	74.94	67.68	85.95	89.35	85.61
poly_5	75.79	67.93	85.11	88.99	84.19
rbf_0.0001	73.76	67.36	57.19	50.64	57.82
rbf_0.001	74.98	66.91	75.47	90.32	80.03
rbf_0.01	75.18	69.83	83.21	91.35	88.39
rbf_0.1	76.64	70.64	85.16	91.27	87.80
rbf_1	71.37	67.97	79.64	84.03	75.23
rbf_10	50.41	50.45	57.19	51.08	57.82

Table 8.5 Average testing accuracy (%) in 4-fold cross-validation (size = 10)

Kernels	Testing Cases				
	Case 1 (H <-> All others)	Case 2 ($\overline{H}H...H\overline{H}$ <-> $HH..HH$)	Case 3 (H <-> E)	Case 4 (H <-> C)	Case 5 (E <-> C)
poly_1	77.71	72.45	82.11	90.45	85.68
poly_2	75.52	67.19	85.53	90.90	84.64
poly_3	75.99	66.99	86.11	90.94	85.03
poly_4	76.16	67.34	85.69	90.40	84.12
poly_5	76.73	66.10	84.40	90.04	81.97
rbf_0.0001	49.47	66.88	68.54	57.54	61.66
rbf_0.001	77.59	68.24	68.54	89.15	64.00
rbf_0.01	79.00	71.75	84.51	92.20	87.31
rbf_0.1	78.92	71.44	85.68	91.97	87.70
rbf_1	71.30	65.90	72.07	79.47	68.17
rbf_10	49.51	50.74	68.54	57.54	61.66

8.2.5 Classification Result Analysis

From Table 8.3-8.5, we may see that for the three window sizes, Test Case 4 (helix vs. coil) has the highest classification accuracies (90-92%), Test Case 5 (strand vs. coil) has the second highest (87-89%), Test Case 3 (helix vs. strand) has the third performance (86%), Test Case 1 (helix vs. all others) has the fourth (73-78%), and the Test Case 5 (helix vs. helix with helix neighbors) has the worst accuracies (69-72%). The following shows the information we can infer from the classification results:

- From the Test Case 4, we can infer that the sequence segments with helix structure have the most significant difference from the sequence segments with coil structure. We can usually identify sequence segments with one structure from the other. This also shows similar structures share high sequence similarity.
- From the Test Case 3 and 5, we can imply that the sequence segments with strand structure have also the significant difference from the sequence segments with either coil structure or helix structure. We can also identify sequence segments with strand structure from these two structures. This also shows similar structures share high sequence similarity, but don't share high sequence similarity with other structures.
- From the Test Case 1 and 2, we can imply that the reason that Test Case 1 doesn't achieve high performance, or in other words, the reason that sequence segments with helix structure are not so significantly different from the segments with all the other structures is that there may exist some segments in the class that comes from all the other local structures but share high sequence similarity with the class that comes from the helix structure. Test Case 2 shows one such case. The fact

that Test Case 2 has lower accuracies than its corresponding Test Case 1 shows that the two classes in Test Case 2 have sort of sequence similarity. We can also infer that the segments with local structure of $\overline{H}H...H\overline{H}$ are not significantly different from the ones with the local structure of $HH...HH$.

8.2.6 Combining the Individual Classifiers

The Test Case 3 and 4 in Table 8.4 at window size of 9 have also been selected as the datasets for testing the proposed fuzzy classifier fusion. Similar to the experiments described in Chapter 5 and 7, these two datasets are applied to the type-1 and type-2 based fuzzy classifier fusion models in cross validation manner. The training dataset in each fold is further divided into second level training and testing datasets. The average testing accuracies in the second level will be used as the accuracy inputs in the fuzzy classifier fusion models. Tables 8.6 and 8.7 show the accuracies from the individual classifiers and also from the two fusion models. We can again see that the proposed fuzzy fusion models perform more stable and more robust than their composing individual classifiers. Even if there is one or more poor classifiers such as SVM 1 in Test 5 and SVM2 in Test 7 (accuracies: 50.64%, 57.19%), both type-1 and type-2 based classifier fusion models can achieve a high accuracy even higher than the best of the individual classifiers.

Table 8.6 Accuracies of individual SVM classifiers, fuzzy fusion model (T1FFSVM), and type-2 fuzzy fusion model (T2FFSVM) (*9helix* vs.*9coils*)

Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM	T2FFSVM
1	poly_1	90.31	poly_3	89.51	rbf_0.01	91.35	90.39	91.35	**91.55**	**91.55**
2	poly_1	90.31	poly_2	89.23	rbf_0.1	91.27	90.27	91.27	91.27	**91.59**
3	poly_4	89.35	rbf_1	84.03	rbf_0.01	91.35	88.24	91.35	90.99	91.27
4	poly_1	90.31	poly_3	89.51	poly_5	88.99	89.60	90.31	**90.63**	**90.71**
5	rbf_0.0001	50.64	rbf_0.01	91.35	rbf_1	84.03	75.34	91.35	91.07	**91.55**
6	poly_5	88.99	rbf_0.001	90.32	rbf_0.1	91.27	90.19	91.27	**91.59**	**91.71**
7	poly_5	88.99	rbf_0.001	50.64	rbf_0.1	84.03	74.55	88.99	**89.11**	**89.67**
Avg.							**85.51**	**90.84**	**90.89**	**91.15**

Table 8.7 Accuracies of individual SVM classifiers, fuzzy fusion model (T1FFSVM), and type-2 fuzzy fusion model (T2FFSVM) (*9helix* vs. *9strands*)

Test	SVM1	Accuracy (%)	SVM2	Accuracy (%)	SVM3	Accuracy (%)	Avg.	Max	T1FFSVM	T2FFSVM
1	poly_1	81.31	poly_3	86.46	rbf_0.01	83.21	83.66	86.46	85.81	86.18
2	poly_1	81.31	poly_2	86.69	rbf_0.1	85.16	84.39	86.69	86.13	86.41
3	poly_4	85.95	rbf_1	79.64	rbf_0.01	83.21	82.93	85.95	85.21	**86.04**
4	poly_1	81.31	poly_3	86.46	poly_5	85.11	84.29	86.46	86.46	**86.60**
5	rbf_0.0001	57.19	rbf_0.01	83.21	rbf_1	79.64	73.35	83.21	**83.30**	**83.95**
6	poly_5	85.11	rbf_0.001	75.47	rbf_0.1	85.16	81.91	85.16	**86.37**	**86.32**
7	poly_5	85.11	rbf_0.001	57.19	rbf_0.1	79.64	73.98	85.11	84.14	**85.67**
Avg.							**80.64**	**85.58**	**85.34**	**85.88**

Chapter 9 Conclusions and Future Work

9.1 Conclusion

In this dissertation, several fusion models have been proposed to combine multiple SVM classifiers. Type-1 fuzzy fusion model for SVMs (T1FFSVM) defined in Chapter 4 is constructed using the traditional fuzzy logic in consideration of classification accuracies and classification results from individual classifiers. This fuzzy classifier fusion model is optimized in Chapter 5 by constructing a genetic fuzzy classification fusion model (GFFSVM) in which the real-coded GAs are applied to tune the input membership parameters and the rule consequences simultaneously in a cross validation manner. The experiments have shown that the GFFSVM outperforms T1FFSVM because unlike the fuzzy MFs intuitively defined in T1FFSVM, GFFSVM provides the powerful searching abilities to find the optimal or near-optimal MFs.

Classifier performance measures are studied in Chapter 6. The AUC-based classifier fusion model is proposed and compared with the accuracy based classifier. The experimental results show that the AUC-based classifier not only achieves nice AUC performance, but also excellent accuracy performance.

The type-2 fuzzy based fusion model (T2FFSVM) proposed in Chapter 7 is constructed by interval type-2 fuzzy sets and interval type-2 FLS. The experimental results demonstrate that the type-2 fuzzy classifier fusion model handles the uncertainties in MFs and classification data better than the model constructed by the traditional type-1 fuzzy logic does.

The experiments also show that the proposed GFFSVM and T2FFSVM are more robust and more reliable than individual SVM classifiers. Different classifiers potentially

offer complementary information about data examples to be classified. Ensemble and fusion model provides a general idea to enhance the performance of a single weak decision. The proposed fuzzy fusion methods also demonstrate better performance than the existing fusion methods including majority vote, minimum, maximum, product, and average.

The case study in Chapter 8 shows the sequence segments with similar protein structures generally have high sequence similarity and are different from the sequences with other structures.

9.2 Future Work

The study on the combination of classifiers is now only at its preliminary stage and is by no mean the final stage of my dissertation work. Although the experiments demonstrate a promising performance comparing with composing individual classifiers and other existing fusion methods, a lot of work could be done to improve the performance of the proposed fusion models:

- ◊ The proposed models have been implemented and tested on combining three SVM classifiers. The system for combining other number of classifiers could be constructed and implemented. The comparison among the fusion models which combine different number of classifiers could be done in terms of both classification performance and time complexity.
- ◊ AUC-based and accuracy based fusion models could be compared and analyzed theoretically and statistically in terms of different class distribution.
- ◊ The proposed classifier fusion models only combine SVM-based classifiers. Other classifiers could also be investigated such as Bayesian classifiers, decision trees,

or neural networks. The models could be compared with SVM-based classifiers regarding to the performance of classifier fusion models.

◊ The dissertation has proposed practically heuristic classifier fusion models. The parameter settings could be further studied in order to provide a general solution to the classifier fusion models. Theoretical study could also be discussed about why combination methods and strategies work well and in what case methods are better than other cases.

Bibliography

- Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J.H., Zhang, Z., Miller, W., and Lipman, D.J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17), 3389-3402.

- Aguero, J. and Vargas, A. (2005). Inference of operative configuration of distribution networks using fuzzy logic techniques-Part I: real-time model. *IEEE Transactions on Power Systems*, 20(3), 1551-1561.

- Breiman, L. (1996). Bagging predictors. *Machine Learning Journal*, 24(2), 123-140.

- Bäck, T., Hoffmeister, F., and Schwefel, H. (1991). A survey of evolution strategies. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 2–9.

- Chen, X.J., Harrison, R., and Zhang, Y.-Q. (2005a). Fuzzy support vector machines for biomedical data analysis. *Proceedings of IEEE International Conference on Granular Computing (IEEE-GrC)*, 131–134, Beijing.

- Chen, X.J., Harrison, R., and Zhang, Y.-Q. (2005b). Multi-SVM fuzzy classification and fusion method and applications in bioinformatics. *Journal of Theoretical and Computational Nanoscience*, 2(4), 514–542.

- Chen, X.J., Harrison, R., and Zhang, Y.-Q. (2005c). Genetic fuzzy fusion of SVM classifiers for biomedical data. *Proceedings of IEEE Congress on Evolutionary Computation (IEEE-CEC)*, 1, 654–659, Edinburgh.

- Chen, X.J., Harrison, R., and Zhang, Y.-Q., and Qiu, Y. (2006). A Multi-SVM fusion model using type-2 FLS. *Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1261–1265, Vancouver.

- Chen, X.J., Li, Y., Harrison, R., and Zhang, Y.-Q. (2007a). Genetic fuzzy classification fusion of multiple SVMs for biomedical data. *Journal of Intelligent & Fuzzy Systems*, 18, 1-15.

- Chen, X.J., Li, Y., Harrison, R., and Zhang, Y.-Q. (2007b). Type-2 fuzzy logic based classifier fusion for support vector machines. *Applied Soft Computing Journal*.

- Chen, X., Zhao, Y.C., Zhang, Y.-Q., and Harrison, R. (2007c). Combining SVM classifiers using genetic fuzzy systems based on AUC for gene expression data analysis. *International Symposium on Bioinformatics Research and Applications (ISBRA)*, 496-505, Atlanta.

- Cordon and Herrera, F. (1997). A three-stage evolutionary process for learning descriptive and approximate fuzzy logic controller knowledge bases from examples. *International Journal of Approximate Reasoning*, 17(4), 369-407.

- Dodge, C., Schneider, R. and Sander, C. (1998). The HSSP database of protein structure–sequence alignments and family profiles. *Nucleic Acids Res*., 26, 313–315.

- Fawcett, T. (2003). ROC graphs: Notes and practical considerations for researchers. *Tech Report HPL-2003-4*, HP Laboratories.

- Freud, Y. and Schapire, R.E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Proceeding of the Second European Conference on Computational Learning*.

- Freund, Y. and Schapire, R.E. (1996). Experiments with a new boosting algorithm. *In Machine Learning: Proceedings of the Thirteenth International Conference*, 148–156.

- Gershon, D. (2002). Microarray technology. An array of opportunities. *Nature*, 416, 885-891.

- Goldberg, D.E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company, Inc.

- Hand, D.J., and Till, R.J. (2001). A simple generalization of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45, 171–186.

- Hagras, H.A. (2004). A Hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Transactions on Fuzzy Systems*, 12(4), 524-539.

- Hansen, L., and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12, 993-1001.

- Hastie, T. and Tibshirani, R. (1996). Classifcation by pairwise coupling. *Technical Report,* Stanford University and University of Toronto.

- Herrera, F., Lozano, M., and Verdegay, J.L. (1995). Generating fuzzy rules from examples using genetic algorithms. *Fuzzy Logic and Soft Computing*.

- Ho, T.K., Hull, J.J., and Srihari, S.N. (1994). Decision combination in multiple classifier systems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(1), 66-75.

- Ho, T.K. (1995). Random decision forests. *Third Int'l Conf. Document Analysis and Recognition*, Montreal, 278-282.

- Holland, J., and Reitman, J. (1978). Cognitive systems based on adaptive algorithms. *Pattern-Directed Inference Systems*, Academic Press.

- Homaifar, and McCormick, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(2), 129-139.

- Huang, J., and Ling, C.X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Trans. Knowl. Data Eng*, 17(3), 299-310.

- John, R. (1998). Type 2 fuzzy sets: An appraisal of theory and applications. International Journal of Uncertainty, *Fuzziness and Knowledge Based Systems*, 6(6), 563–576.

- Joachims, T. (1999). Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning,* MIT-Press. http://svmlight.joachims.org.

- Karnik, N.N. and Mendel, J.M. (1998). Introduction to type-2 fuzzy logic systems. *Proc. 1998 IEEE FUZZ Conf.*, 915-920.

- Karnik, N., Mendel, J., and Liang, Q. (1999). Type-2 fuzzy logic systems. *IEEE Transactions on Fuzzy Systems*, 7, 643–658.

- Karr, C. (1991). Applying genetic to fuzzy logic. *AI Expert*, 6, 26-33.

- Kittler, J., Hatef, M., Duin R., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3).

- Kuncheva, L.I. (2001). Combining classifiers: soft computing solutions. *Pattern Recognition: from Classical to Modern Approaches, World Scientific*, Pal, S.K., Pal, A. (Eds.), 427-451.

- Kuncheva, L.I. (2002). A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2), 281-286.

- Lam, L. (2000). Classifier combinations: implementations and theoretical issues. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, 1857, 78-86, Springer.

- Lerner, B., Yeshaya, J., Koushnir (2007). On the classification of a small imbalanced cytogenetic image database. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2), 204-215.

- Li, J. and Liu, H. (2003). Kent Ridge Biomedical Data Set Repository. http://sdmc.i2r.a-star.edu.sg/rp/.

- Li, W., Liu, Y., Huang, H.-C., Peng, Y., Lin, Y., Ng, W.-K., Ong, K.-L. (2007). Dynamical systems for discovering protein complexes and functional modules from biological networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2), 233-250.

- Liang, Q. and Mendel, J. (2000). Interval type-2 fuzzy logic systems: Theory and design. *IEEE Transactions on Fuzzy Systems*, 8, 535–550.

- Liang, Q., and Mendel, J. (2001). MPEG VBR video traffic modeling and classification using fuzzy technique. *IEEE Transactions on Fuzzy Systems*, 9(1), 183-193.

- Liang, Q., Karnik, N.N. and Mendel, J.M. (2000). Connection admission control in ATM networks using survey-based type-2 fuzzy logic systems. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 30, 329-339.

- Ling, C.X., Huang, J., and Zhang, H. (2003). AUC: a statistically consistent and more discriminating measure than accuracy. *Proc. 18th Int'l Conf. Artificial Intelligence (IJCAI '03)*, 329-341.

- Liu, J.S. (2002). Bioinformatics: microarrays analyses and beyond. *Amstat News*, 59-67.

- Magdalena, L., Cordon, O., Gomide, F., Herrera, F., and Hoffmann, F. (2004). Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets & Systems*, 141(1), 5-31.

- Mamdani, E.H., and Assilian, S. (1973). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7, 1-13.

- Mamdani, E.H. (1974). Application of fuzzy algorithms for control of simple dynamic plant. *IEEE Proceedings*, 121(12), 1585-1588.

- Mendel, J. and John, R. (2002). Type-2 fuzzy sets made simple. *IEEE Transactions on Fuzzy Systems*, 10, 117–127.

- Mendel, J. (2001). *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice-Hall.

- Merz, C.J. and Murphy, P.M. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

- Park, D. and Kandel, A. (1994). Genetic-based new fuzzy reasoning models with application to fuzzy control. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1), 39-47.

- Qin, Z.-C (2005). ROC analysis for predictions made by probabilistic classifiers. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, 5, 3119-3124.

- Rajapakse, J.C., Zhang,Y.-Q., and Fogel, G.B. (2007). Guest editors' introduction to the special section: computational intelligence approaches in computational biology and bioinformatics. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 4(2), 161-162.

- Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang., C.H., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J.P., Poggio, T., Gerald, W., Loda, M., Lander E.S., and Golub, T.R. (2001). Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences of the United States of America*, 98(26), 15149-15154.

- Roth, V., and Lange, T. (2004). Bayesian class discovery in microarray datasets. *IEEE Transactions on Biomedical Engineering*, 51(5), 707 – 718.

- Sander, C., and Schneider, R. (1991). Database of homology-derived protein structures and the structural meaning of sequence alignment. Proteins: *Struct. Funct. Genet.*, 9(1), 56-68.

- Schapire, R.E. (1990). The strength of weak learnability. *Machine learning*, 5(2), 197-227.

- Schena, M., Shalon, D., Davis, R., and Brown, P.O. (1995). Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270, 467-470.

- Shipp, C.A. and Kuncheva, L.I. (2002). Relationship between combination Methods and measures of diversity in combining classifiers. *Information Fusion*, 3(2), 135–148.

- Sjahputera, O., Keller, J.M., Davis, J.W., Taylor, K.H., Rahmatpanah, F., Huidong Shi, Anderson, D.T., Blisard, S.N., Luke, R.H., Popescu, M., Arthur, G.C., Caldwell, C.W. (2007). Relational analysis of CpG islands methylation and gene expression in human lymphomas using possibilistic c-means clustering and modified cluster fuzzy density. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2), 176-189.

- Spears, W. and De Jong, K. (1991). On the virtues of uniform crossover. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 230-236.

- Smith, S. (1980). A learning system based on genetic adaptive algorithms. *Doctoral Dissertation*, Department of Computer Science, University of Pittsburgh.

- Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on System, Man, Cybernetics*, 15(1), 116-132.

- Thompson, J.D., Higgins, D.G. and Gibbson, T.J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*. 22, 4673-80.

- The National Health Museum: http://www.accessexcellence.org/RC/VL/GG/protein_synthesis.html.

- Vapnik, V. (1995). *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

- Vlahou, A., Schorge, J.O., Gregory, B.W., and Coleman, R.L. (2003). Diagnosis of ovarian cancer using decision tree classification of mass spectral data. *Journal of Biomedicine and Biotechnology*, 1(5), 308-314.

- Wang, G., and Dunbrack, R.L. (2003). PISCES: a protein sequence-culling server. *Bioinformatics*, 19(12), 1589-1591.

- Weston, J. and Watkins, C. (1998). Multi-class support vector machines. *Technical Report CSD-TR-98-04*, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England.

- Xu, L., Krzyzak, A., and Suen, C.Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. Systems*, Man, and Cybernetics, 22(3), 418-435.

- Yao, X. and Liu, Y. (1999). Neural networks for breast cancer diagnosis. *Proceedings of the 1999 Congress on Evolutionary Computation*, 3.

- Zeng, J., and Liu, Z. (2006). Type-2 fuzzy hidden Markov models and their application to speech recognition. *IEEE Transactions on Fuzzy Systems*, 14(3), 454-467.

- Zadeh, L. (1965). Fuzzy set. *Information and Control*, 8, 338-35.