Support Vector Learning for Fuzzy Rule-Based Classification Systems

Yixin Chen, *Student Member, IEEE*, James Z. Wang, *Member, IEEE*

Yixin Chen is with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA (e-mail: yixchen@cse.psu.edu).

James Z. Wang is with the School of Information Sciences and Technology and the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA (e-mail: jwang@ist.psu.edu).

Abstract

To design a fuzzy rule-based classification system (fuzzy classifier) with good generalization ability in a high dimensional feature space has been an active research topic for a long time. As a powerful machine learning approach for pattern recognition problems, support vector machine (SVM) is known to have good generalization ability. More importantly, an SVM can work very well on a high (or even infinite) dimensional feature space. This paper investigates the connection between fuzzy classifiers and kernel machines, establishes a link between fuzzy rules and kernels, and proposes a learning algorithm for fuzzy classifiers. We first show that a fuzzy classifier implicitly defines a translation invariant kernel under the assumption that all membership functions associated with the same input variable are generated from location transformation of a reference function. Fuzzy inference on the IF-part of a fuzzy rule can be viewed as evaluating the kernel function. The kernel function is then proven to be a Mercer kernel if the reference functions meet certain spectral requirement. The corresponding fuzzy classifier is named positive definite fuzzy classifier (PDFC). A PDFC can be built from the given training samples based on a support vector learning approach with the IF-part fuzzy rules given by the support vectors. Since the learning process minimizes an upper bound on the expected risk (expected prediction error) instead of the empirical risk (training error), the resulting PDFC usually has good generalization. Moreover, because of the sparsity properties of the SVMs, the number of fuzzy rules is irrelevant to the dimension of input space. In this sense, we avoid the "curse of dimensionality." Finally, PDFCs with different reference functions are constructed using the support vector learning approach. The performance of the PDFCs is illustrated by extensive experimental results. Comparisons with other methods are also provided.

Index Terms— Fuzzy systems, statistical learning theory, support vector machines, fuzzy classifier, kernel methods, pattern classification.

1 Introduction

Since the publication of L.A. Zadeh's seminal paper on fuzzy sets [64], fuzzy set theory and its descendant, fuzzy logic, have evolved into powerful tools for managing uncertainties inherent in complex systems. In the recent twenty years, fuzzy methodology has been successfully applied to a variety of areas including control and system identification [27], [30], [48], [57], [65], signal

and image processing [36], [39], [47], pattern classification [1], [17], [20], [26], and information retrieval [8], [34]. In general, building a fuzzy system consists of three basic steps [61]: structure identification (variable selection, partitioning input and output spaces, specifying the number of fuzzy rules, and choosing a parametric/nonparametric form of membership functions), parameter estimation (obtaining unknown parameters in fuzzy rules via optimizing a given criterion), and model validation (performance evaluation and model simplification). There are numerous studies on all these subjects. Space limitation precludes the possibility of a comprehensive survey. Instead, we only review some of those results that are most related to ours.

1.1 Structure Identification and Parameter Estimation

Deciding the number of input variables is referred to the problem of variable selection, i.e., selecting input variables that are most predictive of a given outcome. It is related to the problems of input dimensionality reduction and parameter pruning. Emami *et al.* [14] present a simple method of identifying non-significant input variables in a fuzzy system based on the distribution of degree of memberships over the domain. Recently, Silipo *et al.* [44] propose a method that quantifies the discriminative power of the input features in a fuzzy model based on information gain. Selecting input variables according to their information gains may improve the prediction performance of the fuzzy system and provides a better understanding of the underlying concept that generates the data.

Given a set of input and output variables, a fuzzy partition associates fuzzy sets (or linguistic labels) with each variable. There are roughly two ways of doing it: data independent partition and data dependent partition. The former approach partitions the input space in a predetermined fashion. The partition of the output space then follows from supervised learning. One of the commonly used strategies is to assign a fixed number of linguistic labels to each input variable [56]. Although this scheme is not difficult to implement, it has two serious drawbacks:

- The information in the given data (patterns) is not fully exploited. The performance of the resulting system may be poor if the input space partition is quite distinct from the true distribution of data. Optimizing output space partition alone is not sufficient.
- The scheme suffers from the curse of dimensionality. If each input variable is allocated m fuzzy sets, a fuzzy system with n inputs and one output needs on the order of m^n rules.

Various data dependent partition methods have been proposed to alleviate these drawbacks. Dickerson *et al.* [11] use an unsupervised competitive learning algorithm to find the mean and

covariance matrix of each data cluster in the input/output space. Each data cluster forms an ellipsoidal fuzzy rule patch. Thawonmas *et al.* [50] describe a simple heuristic for unsupervised iterative data partition. At each iteration, an input dimension, which gives the maximum intra-class difference between the maximum and the minimum values of the data along that dimension, is selected. The partition is performed perpendicular to the selected dimension. Two data group representations, hyper-box and ellipsoidal representations, are compared. In [42], a supervised clustering algorithm is used to group input/output data pairs into a predetermined number of fuzzy clusters. Each cluster corresponds to a fuzzy IF-THEN rule. Univariate membership functions can then be obtained by projecting fuzzy clusters onto corresponding coordinate axes.

Although a fuzzy partition can generate fuzzy rules, results are usually very coarse with many parameters to be learned and tuned. Various optimization techniques are proposed to solve this problem. Genetic algorithms [9], [49], [59] and artificial neural networks [22], [24], [60] are two of the most popular and effective approaches.

1.2 Generalization Performance

After going through the long journey of structure identification and parameter estimation, can we infer that we get a good fuzzy model? In order to draw a conclusion, the following two questions must be answered:

- How capable can a fuzzy model be?
- How well can the model, built on finite amount of data, capture the concept underlying the data?

The first question could be answered from the perspective of function approximation. Several types of fuzzy models are proven to be "universal approximators" [28], [38], [58], [63], i.e., we can always find a model from a given fuzzy model set so that the model can uniformly approximate any continuous function on a compact domain to any degree of accuracy. The second question is about the generalization performance, which is closely related to several well-known problems in the statistics and machine learning literature, such as the structural risk minimization [51], the bias variance dilemma [15], and the overfitting phenomena [2]. Loosely speaking, a model, build on finite amount of given data (training patterns), generalizes the best if the right tradeoff is found between the training (learning) accuracy and the "capacity" of the model set from which the model is chosen. On one hand, a low "capacity" model set may not contain any model that

fits the training data well. On the other hand, too much freedom may eventually generate a model behaving like a refined look-up-table: perfect for the training data but (maybe) poor on generalization.

Researchers in the fuzzy systems community attempt to tackle this problem with roughly two approaches:(1) use the idea of cross-validation to select a model that has the best ability to generalize [46]; (2) focus on model reduction, which is usually achieved by rule base reduction [43], [62], to simplify the model. In statistical learning literature, the Vapnik-Chervonenkis (VC) theory [52], [53] provides a general measure of model set complexity. Based on the VC theory, support vector machines (SVM) [52], [53] can be designed for classification problems. In many real applications, the SVMs give excellent performance [10].

1.3 Our Approach

However, no effort has been made to analyze the relationship between fuzzy rule-based classification systems and kernel machines. The work presented here attempts to bridge this gap. We relate additive fuzzy systems to kernel machines, and demonstrate that, under a general assumption on membership functions, an additive fuzzy rule-based classification system can be constructed directly from the given training samples using the support vector learning approach. Such additive fuzzy rule-based classification systems are named the positive definite fuzzy classifiers (PDFC). Using the SVM approach to build PDFCs has following advantages:

- Fuzzy rules are extracted directly from the given training data. The number of fuzzy rules is irrelevant to the dimension of the input space. It is no greater (usually much less) than the number of training samples. In this sense, we avoid the "curse of dimensionality".
- The VC theory establishes the theoretical foundation for good generalization of the resulting PDFC.
- The global solution of an SVM optimization problem can be found efficiently using specifically designed quadratic programming algorithms.

The remainder of the paper is organized as follows. In Section 2, a brief overview of the VC theory and SVMs is presented. Section 3 describes the PDFCs, a class of additive fuzzy rule-based classification systems with positive definite membership functions, product fuzzy conjunction operator, and center of area (COA) defuzzification with thresholding unit. We show that the decision boundary of a PDFC can be viewed as a hyperplane in the feature space induced by the kernel. In Section 4, an algorithm is provided to construct PDFC: first,

an optimal separating hyperplane is found using the support vector learning approach, fuzzy rules are then extracted from the hyperplane. Section 5 demonstrates the experiments we have performed, and provides the results. A description of the relationship between PDFCs and SVMs with radial basis function (RBF) kernels and a discussion on the advantages of relating fuzzy systems to kernel machines are presented in Section 6. And finally, we conclude in Section 7 together with a discussion of future work.

2 VC Theory and Support Vector Machines

This section presents the basic concepts of the VC theory and SVMs. For gentle tutorials of VC theory and SVMs, we refer interested readers to Burges [5] and Müller et al. [35]. More exhaustive treatments can be found in the books by Vapnik [52], [53].

2.1 VC Theory

Let's consider a two-class classification problem of assigning class label $y \in \{+1, -1\}$ to input feature vector $\vec{x} \in \mathbb{R}^n$. We are given a set of training samples $\{(\vec{x}_1, y_1), \cdots, (\vec{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$ that are drawn independently from some unknown cumulative probability distribution $P(\vec{x}, y)$. The learning task is formulated as finding a machine (a function $f : \mathbb{R}^n \to \{+1, -1\}$) that "best" approximates the mapping generating the training set. In order to make learning feasible, we need to specify a function space, $\mathbb{H}$, from which a machine is chosen.

An ideal measure of generalization performance for a selected machine f is expected risk (or the probability of misclassification) defined as $R_{P(\vec{x},y)}(f) = \int_{\mathbb{R}^n \times \{+1,-1\}} \mathrm{I}_{\{f(\vec{x}) \neq y\}}(\vec{x}, y) dP(\vec{x}, y)$ where $\mathrm{I}_A(z)$ is an indicator function such that $\mathrm{I}_A(z) = 1$ for all $z \in A$, and $\mathrm{I}_A(z) = 0$ for all $z \notin A$. Unfortunately, this is more an elegant way of writing the error probability than practical usefulness because $P(\vec{x}, y)$ is usually unknown. However, there is a family of bounds on the expected risk, which demonstrates fundamental principles of building machines with good generalization. Here we present one result from the VC theory due to Vapnik and Chervonenkis [54]: given a set of l training samples and function space $\mathbb{H}$, with probability $1 - \eta$, for any $f \in \mathbb{H}$ the expected risk is bounded above by

$$R_{P(\vec{x},y)}(f) \leq R_{emp}(f) + \sqrt{\frac{h(1 + \ln \frac{2l}{h}) - \ln \frac{\eta}{4}}{l}} \tag{1}$$

for any distribution $P(\vec{x}, y)$ on $\mathbb{R}^n \times \{+1, -1\}$. Here $R_{emp}(f)$ is called the empirical risk (or training error), h is a non-negative integer called the Vapnik Chervonenkis (VC) dimension.

The VC dimension is a measure of the capacity of a $\{+1, -1\}$-valued function space. Given a training set of size l, (1) demonstrates a strategy to control expected risk by controlling two quantities: the empirical risk and the VC dimension. Next we will discuss an application of this idea: the SVM learning strategy.

2.2 Support Vector Machines

Let $\{(\vec{x}_1, y_1), \cdots, (\vec{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$ be a training set. The SVM learning approach attempts to find a canonical hyperplane [1] $\{\vec{x} \in \mathbb{R}^n : \langle \vec{w}, \vec{x} \rangle + b = 0,\ \vec{w} \in \mathbb{R}^n,\ b \in \mathbb{R}\}$ that maximally separates two classes of training samples. Here $\langle \cdot, \cdot \rangle$ is an inner product in $\mathbb{R}^n$. The corresponding decision function (or classifier) $f : \mathbb{R}^n \to \{+1, -1\}$ is then given by $f(\vec{x}) = \operatorname{sgn}(\langle \vec{w}, \vec{x} \rangle + b)$.

Considering that the training set may not be linearly separable, the optimal decision function is found by solving the following quadratic program:

$$\begin{aligned} \text{minimize} \quad & J(\vec{w}, \vec{\xi}) = \frac{1}{2} \langle \vec{w}, \vec{w} \rangle + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i (\langle \vec{w}, \vec{x}_i \rangle + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, \cdots, l \end{aligned} \tag{2}$$

where $\vec{\xi} = [\xi_1, \cdots, \xi_l]^T$ are slack variables introduced to allow for the possibility of misclassification of training samples, $C > 0$ is some constant.

How does minimizing (2) relate to our ultimate goal of optimizing the generalization? To answer this question, we need to introduce a theorem about the VC dimension of canonical hyperplanes [52], which is stated as follows. For a given set of l training samples, let R be the radius of the smallest ball containing all l training samples, and $\Lambda \subset \mathbb{R}^n \times \mathbb{R}$ be the set of coefficients of canonical hyperplanes defined on the training set. The VC dimension h of the function space $\mathbb{H} = \{f(\vec{x}) = \operatorname{sgn}(\langle \vec{w}, \vec{x} \rangle + b) : (\vec{w}, b) \in \Lambda, \|\vec{w}\| \leq A, \vec{x} \in \mathbb{R}^n\}$ is bounded above by $h \leq \min(R^2 A^2, n) + 1$. Thus minimizing the $\frac{1}{2} \langle \vec{w}, \vec{w} \rangle$ term in (2) amounts to minimizing the VC dimension of $\mathbb{H}$, therefore the second term of the bound (1). On the other hand, $\sum_{i=1}^{l} \xi_i$ is an upper bound on the number of misclassifications on the training set [2], thus controls the empirical risk term in (1). For an adequate positive constant C, minimizing (2) can indeed decrease the upper bound on the expected risk.

[1] A hyperplane $\{\vec{x} \in \mathbb{R}^n : \langle \vec{w}, \vec{x} \rangle + b = 0,\ \vec{w} \in \mathbb{R}^n,\ b \in \mathbb{R}\}$ is called canonical for a given training set if and only if $\vec{w}$ and b satisfy $\min_{i=1,\cdots,l} |\langle \vec{w}, \vec{x}_i \rangle + b| = 1$.

[2] A training feature vector $\vec{x}_i$ is misclassified if and only if $1 - \xi_i < 0$ or equivalently $\xi_i > 1$. Let t be the number of misclassifications on the training set. We have $t \leq \sum_{i=1}^{l} \xi_i$ since $\xi_i \geq 0$ for all i and $\xi_i > 1$ for misclassifications.

Applying the Karush-Kuhn-Tucker complementarity conditions, one can show that a $\vec{w}$, which minimizes (2), can be written as $\vec{w} = \sum_{i=1}^{l} y_i \alpha_i \vec{x}_i$. This is called the dual representation of $\vec{w}$. An $\vec{x}_j$ with nonzero α_j is called a support vector. Let $\mathcal{S}$ be the index set of support vectors, then the optimal decision function becomes

$$f(\vec{x}) = \text{sgn}\left(\sum_{i \in \mathcal{S}} y_i \alpha_i \langle \vec{x}, \vec{x}_i \rangle + b\right) \tag{3}$$

where the coefficients α_i can be found by solving the dual problem of (2):

$$\begin{aligned} \text{maximize} \quad & W(\vec{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \vec{x}_i, \vec{x}_j \rangle \\ \text{subject to} \quad & C \geq \alpha_i \geq 0, \ i = 1, \cdots, l, \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned} \tag{4}$$

The decision boundary given by (3) is a hyperplane in $\mathbb{R}^n$. More complex decision surfaces can be generated by employing a nonlinear mapping $\Phi : \mathbb{R}^n \to \mathbb{F}$ to map the data into a new feature space $\mathbb{F}$ (usually has dimension higher than n), and finding the maximal separating hyperplane in $\mathbb{F}$. Note that in (4) $\vec{x}_i$ never appears isolated but always in the form of inner product $\langle \vec{x}_i, \vec{x}_j \rangle$. This implies that there is no need to evaluate the nonlinear mapping Φ as long as we know the inner product in $\mathbb{F}$ for any given $\vec{x}, \vec{z} \in \mathbb{R}^n$. So for computational purposes, instead of defining $\Phi : \mathbb{R}^n \to \mathbb{F}$ explicitly, a function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is introduced to directly define an inner product in $\mathbb{F}$. Such a function K is also called the Mercer kernel [10], [52], [53]. Substituting $K(\vec{x}_i, \vec{x}_j)$ for $\langle \vec{x}_i, \vec{x}_j \rangle$ in (4) produces a new optimization problem

$$\begin{aligned} \text{maximize} \quad & W(\vec{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j) \\ \text{subject to} \quad & C \geq \alpha_i \geq 0, \ i = 1, \cdots, l, \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned} \tag{5}$$

Solving (5) for $\vec{\alpha}$ gives a decision function of the form

$$f(\vec{x}) = \text{sgn}\left(\sum_{i \in \mathcal{S}} y_i \alpha_i K(\vec{x}, \vec{x}_i) + b\right) \quad , \tag{6}$$

whose decision boundary is a hyperplane in $\mathbb{F}$, and translates to nonlinear boundaries in the original space. Several techniques of solving quadratic programming problems arising in SVM algorithms are described in [23], [25], [37]. Details of calculating b can be found in [7].

3 Additive Fuzzy Rule-Based Classification Systems and Positive Definite Fuzzy Classifiers

This section starts with a short description of an additive fuzzy model, based on which binary fuzzy classifiers and standard binary fuzzy classifiers are defined. We then introduce the concept of positive definite functions, and define positive definite fuzzy classifiers (PDFC) accordingly. Finally, some nice properties of the PDFCs are discussed.

3.1 Additive Fuzzy Rule-Based Classification Systems

Depending on the THEN-part of fuzzy rules and the way to combine fuzzy rules, a fuzzy rule-based classification system can take many different forms [29]. In this paper, we consider the additive fuzzy rule-based classification systems (or in short fuzzy classifier) with constant THEN-parts. Although the discussions in this section and Section 4 focus on binary classifiers. The results can be extended to multi-class problems by combining several binary classifiers.

Consider a fuzzy model with m fuzzy rules of the form

$$\text{Rule } j: \quad \text{IF } \mathbf{A}_j^1 \text{ AND } \mathbf{A}_j^2 \text{ AND } \cdots \text{ AND } \mathbf{A}_j^n \text{ THEN } b_j \tag{7}$$

where $\mathbf{A}_j^k$ is a fuzzy set with membership function $a_j^k : \mathbb{R} \to [0,1]$, $j = 1, \cdots, m$, $k = 1, \cdots, n$, $b_j \in \mathbb{R}$. If we choose product as the fuzzy conjunction operator, addition for fuzzy rule aggregation (that is what "additive" means), and COA defuzzification, then the model becomes a special form of the Takagi-Sugeno (TS) fuzzy model [48], and the input output mapping, $F : \mathbb{R}^n \to \mathbb{R}$, of the model is defined as

$$F(\vec{x}) = \frac{\sum_{j=1}^m b_j \prod_{k=1}^n a_j^k(x_k)}{\sum_{j=1}^m \prod_{k=1}^n a_j^k(x_k)} \tag{8}$$

where $\vec{x} = [x_1, \cdots, x_n]^T \in \mathbb{R}^n$ is the input. Note that (8) is not well-defined on $\mathbb{R}^n$ if $\sum_{j=1}^m \prod_{k=1}^n a_j^k(x_k) = 0$ for some $\vec{x} \in \mathbb{R}^n$, which could happen if the input space is not wholly covered by fuzzy rule "patches". However, there are several easy fixes for this problem. For example, we can force the output to some constant when $\sum_{j=1}^m \prod_{k=1}^n a_j^k(x_k) = 0$, or add a fuzzy rule so that the denominator $\sum_{j=1}^m \prod_{k=1}^n a_j^k(x_k) > 0$ for all $\vec{x} \in \mathbb{R}^n$. Here we take the second approach for analytical simplicity. The following rule is added:

$$\text{Rule } 0: \quad \text{IF } \mathbf{A}_0^1 \text{ AND } \mathbf{A}_0^2 \text{ AND } \cdots \text{ AND } \mathbf{A}_0^n \text{ THEN } b_0 \tag{9}$$

where $b_0 \in \mathbb{R}$, the membership functions $a_0^k(x_k) \equiv 1$ for $k = 1, \cdots, n$ and any $x_k \in \mathbb{R}$. Consequently, the input output mapping becomes

$$F(\vec{x}) = \frac{b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k)}{1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k)} \quad . \tag{10}$$

A classifier associates class labels with input features, i.e., it is essentially a mapping from the input space to the set of class labels. In binary case, thresholding is one of the simplest ways to transform $F(\vec{x})$ to class labels $+1$ or -1. In this article, we are interested in binary fuzzy classifiers defined as follows.

Definition 3.1: (Binary Fuzzy Classifier) *Consider a fuzzy system with $m+1$ fuzzy rules where* Rule 0 *is given by (9),* Rule $j, j = 1, \cdots, m$, *has the form of (7). If the system uses product for fuzzy conjunction, addition for rule aggregation, and COA defuzzification, then the system induces a binary fuzzy classifier, f, with decision rule,*

$$f(\vec{x}) = \text{sign}\left(F(\vec{x}) + t\right) \tag{11}$$

where $F(\vec{x})$ is defined in (10), $t \in \mathbb{R}$ is a threshold.

The following corollary states that we can assume $t = 0$ without loss of generality.

Corollary 3.2: *For any binary fuzzy classifier given by Definition 3.1 with nonzero threshold t, there exists a binary fuzzy classifier that has the same decision rule but zero threshold.*

Proof: Given a binary fuzzy classifier, f, with $t \neq 0$. From (10) and (11), we have

$$f(\vec{x}) = \text{sign}\left(\frac{(b_0 + t) + \sum_{j=1}^{m} (b_j + t) \prod_{k=1}^{n} a_j^k(x_k)}{1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k)}\right) \quad ,$$

which is identical to the decision rule of a binary fuzzy classifier with $b_j + t$ as the THEN-part of jth fuzzy rule ($j = 0, \cdots, m$) and zero threshold. □

The membership functions for a binary fuzzy classifier defined above could be any function from $\mathbb{R}$ to $[0, 1]$. However, too much flexibility on the model could make effective learning (or training) unfeasible. So we narrow our interests to a class of membership functions, which are generated from location transformation of reference functions [12], and the classifiers defined on them.

Definition 3.3: (Reference Function, [12]) *A function $\mu : \mathbb{R} \to [0, 1]$ is a reference function if and only if*

- $\mu(x) = \mu(-x)$;
- $\mu(0) = 1$; *and*

- μ *is nonincreasing on* $[0, \infty)$.

Definition 3.4: (Standard Binary Fuzzy Classifier) *A binary fuzzy classifier given by Definition 3.1 is a standard binary fuzzy classifier if for the kth input,* $k \in \{1, \cdots, n\}$*, the membership functions,* $a_j^k : \mathbb{R} \to [0, 1]$*,* $j = 1, \cdots, m$*, are generated from a reference function* a^k *through location transformation, i.e.,* $a_j^k(x_k) = a^k(x_k - z_j^k)$ *for some location parameter* $z_j^k \in \mathbb{R}$*.*

A simple example will be helpful for illustrating and understanding the basic idea of the above definition. Let's consider a standard binary fuzzy classifier with two inputs (x_1 and x_2) and three fuzzy rules (excluding Rule 0)

$$\begin{aligned} \text{Rule 1}: &\quad \text{IF } \mathbf{A}_1^1 \text{ AND } \mathbf{A}_1^2 \text{ THEN } b_1 \\ \text{Rule 2}: &\quad \text{IF } \mathbf{A}_2^1 \text{ AND } \mathbf{A}_2^2 \text{ THEN } b_2 \\ \text{Rule 3}: &\quad \text{IF } \mathbf{A}_3^1 \text{ AND } \mathbf{A}_3^2 \text{ THEN } b_3 \end{aligned}$$

where $a^1(x_1) = e^{-\frac{x_1^2}{4}}$ and $a^2(x_2) = \max(1 - |\frac{x_2}{3}|, 0)$ are reference functions for inputs x_1 and x_2, respectively, a_j^k is the membership function of $\mathbf{A}_j^k$, $j = 1, 2, 3$, $k = 1, 2$. As shown in Figure 1, the membership functions a_1^1, a_2^1, and a_3^1 belong to one location family generated by a^1, the membership functions a_1^2, a_2^2, and a_3^2 belong the other location family generated by a^2.

Corollary 3.5: *The decision rule of a standard binary fuzzy classifier given by Definition 3.4 can be written as*

$$f(\vec{x}) = \text{sign}\left(\sum_{j=1}^{m} b_j K(\vec{x}, \vec{z}_j) + b_0\right) \tag{12}$$

where $\vec{x} = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^n$*,* $\vec{z}_j = [z_j^1, z_j^2, \cdots, z_j^n]^T \in \mathbb{R}^n$ *contains the location parameters of* a_j^k*,* $k = 1, \cdots, n$*,* $K : \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ *is a translation invariant kernel* [3] *defined as*

$$K(\vec{x}, \vec{z}_j) = \prod_{k=1}^{n} a^k(x_k - z_j^k) \quad . \tag{13}$$

Proof: From (10), (11), and Corollary 3.2, the decision rule of a binary fuzzy classifier is

$$f(\vec{x}) = \text{sign}\left(\frac{b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k)}{1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k)}\right) \quad .$$

Since $1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k) > 0$, we have

$$f(\vec{x}) = \text{sign}\left(b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k)\right) \quad . \tag{14}$$

[3] A kernel $K(\vec{x}, \vec{z})$ is translation invariant if $K(\vec{x}, \vec{z}) = K(\vec{x} - \vec{z})$, i.e., it depends only on $\vec{x} - \vec{z}$, but not on $\vec{x}$ and $\vec{z}$ themselves.

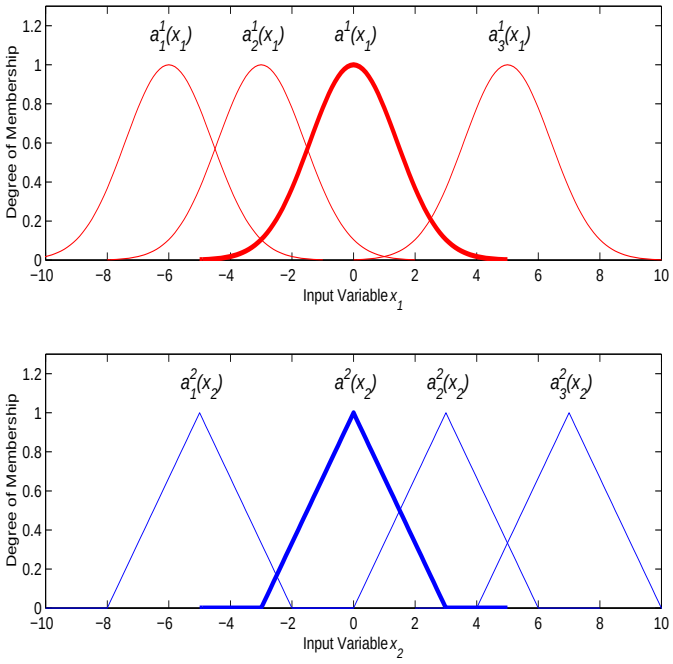


Fig. 1. IF-part membership functions for a standard binary fuzzy classifier. Two thick curves denote the reference functions $a^1(x_1)$ and $a^2(x_2)$ for inputs x_1 and x_2, respectively. $a_1^1(x_1) = a^1(x_1+6)$, $a_2^1(x_1) = a^1(x_1+3)$, and $a_3^1(x_1) = a^1(x_1-5)$ are membership functions associated with x_1. $a_1^2(x_1) = a^2(x_2+5)$, $a_2^2(x_2) = a^2(x_2-3)$, and $a_3^2(x_2) = a^2(x_2-7)$ are membership functions associated with x_2. Clearly, $a_1^1(x_1)$, $a_2^1(x_1)$, and $a_3^1(x_1)$ are location transformed versions of $a^1(x_1)$, and $a_1^2(x_2)$, $a_2^2(x_2)$, and $a_3^2(x_2)$ are location transformed versions of $a^2(x_2)$.

From the definition of standard binary fuzzy classifier, $a_j^k(x_k) = a^k(x_k - z_j^k)$, $k = 1, \cdots, n$, $j = 1, \cdots, m$. Substituting them into (14) completes the proof. □

The decision rule (13) is not merely a different representation form of (11), it provides us with a novel perspective on binary fuzzy classifiers (Section 3.2, 3.3), and accordingly leads to a new design algorithm for binary fuzzy classifiers (Section 4).

3.2 Positive Definite Fuzzy Classifiers

One particular kind of kernel, Mercer kernel, has received considerable attention in the machine learning literature [10], [16], [52], [53] because it is an efficient way of extending linear learning machines to nonlinear ones. Is the kernel defined by (13) a Mercer kernel? Before answering this question, we first quote a theorem.

Theorem 3.6: (Mercer Theorem [10], [32]) *Let $\mathbb{X}$ be a compact subset of $\mathbb{R}^n$. Suppose K is a continuous symmetric function such that the integral operator $T_K : L_2(\mathbb{X}) \rightarrow L_2(\mathbb{X})$,*

$$(T_K f)(\cdot) = \int_{\mathbb{X}} K(\cdot, \vec{x}) f(\vec{x}) d\vec{x}$$

is positive, that is

$$\int_{\mathbb{X}\times\mathbb{X}} K(\vec{x}, \vec{z}) f(\vec{x}) f(\vec{z}) d\vec{x} d\vec{z} \geq 0 \tag{15}$$

for all $f \in L_2(\mathbb{X})$. Then we can expand $K(\vec{x}, \vec{z})$ in a uniformly convergent series (on $\mathbb{X} \times \mathbb{X}$) in terms of T_K's eigen-functions $\phi_i \in L_2(\mathbb{X})$, normalized in such a way that $\|\phi_i\|_{L_2} = 1$, and positive associated eigenvalues $\lambda_j > 0$,

$$K(\vec{x}, \vec{z}) = \sum_{i=1}^{\infty} \lambda_k \phi_i(\vec{x}) \phi_i(\vec{z}) \quad . \tag{16}$$

The positivity condition (15) is also called the Mercer condition. A kernel satisfying the Mercer condition is named a Mercer kernel. An equivalent form of the Mercer condition, which proves most useful in constructing Mercer kernels, is given by the following lemma [10].

Lemma 3.7: (Positivity Condition for Mercer Kernels [10]) *For a kernel $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, the Mercer condition (15) holds if and only if the matrix $[K(\vec{x}_i, \vec{x}_j)] \in \mathbb{R}^{n\times n}$ is positive semi-definite for all choices of points $\{\vec{x}_1, \cdots, \vec{x}_n\} \subset \mathbb{X}$ and all $n = 1, 2, \cdots\cdots$.*

For most nontrivial kernels, directly checking the Mercer conditions in (15) or Lemma 3.7 is not an easy task. Nevertheless, for the class of translation invariant kernels, to which the kernels defined by (13) belong, there is an equivalent yet practically more powerful criterion based the spectral property of the kernel [45].

Lemma 3.8: (Mercer Conditions for Translation Invariant Kernels, Smola *et al.* [45]) *A translation invariant kernel $K(\vec{x}, \vec{z}) = K(\vec{x} - \vec{z})$ is a Mercer kernel if and only if the Fourier transform*

$$\mathcal{F}[K](\vec{\omega}) = \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}^n} K(\vec{x}) e^{-i\langle\vec{\omega},\vec{x}\rangle} d\vec{x}$$

is nonnegative.

Kernels defined by (13) do not, in general, have nonnegative Fourier transforms. However, if we assume that the reference functions are positive definite functions, which are defined by the following definition, then we do get a Mercer kernel (given in Theorem 3.11).

Definition 3.9: (Positive Definite Function [18]) *A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is said to be a positive definite function if the matrix $[f(x_i - x_j)] \in \mathbb{R}^{n\times n}$ is positive semi-definite for all choices of points $\{x_1, \cdots, x_n\} \subset \mathbb{R}$ and all $n = 1, 2, \cdots\cdots$.*

Corollary 3.10: *A function $f : \mathbb{R} \to \mathbb{R}$ is positive definite if and only if the Fourier transform*

$$\mathcal{F}[f](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx$$

is nonnegative.

Proof: Given any function $f : \mathbb{R} \to \mathbb{R}$, we can define a translation invariant kernel $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ as

$$K(x, z) = f(x - z) \quad .$$

From Lemma 3.8, K is a Mercer kernel if and only if the Fourier transform of f is nonnegative. Thus from Lemma 3.7 and Definition 3.9, we conclude that f is a positive definite function if and only if its Fourier transform is nonnegative. □

Theorem 3.11: (Positive Definite Fuzzy Classifier, PDFC) *A standard binary classifier given by Definition 3.4 is called a positive definite fuzzy classifier (PDFC) if the reference functions, $a^k : \mathbb{R} \to [0, 1],\ k = 1, \cdots, n$, are positive definite functions (they do not need to be the same function). The translation invariant kernel (13) is then a Mercer kernel.*

Proof: From Lemma 3.8, it suffices to show that the translation invariant kernel defined by (13) has nonnegative Fourier transform. Rewrite (13) as

$$K(\vec{x}, \vec{z}) = K(\vec{u}) = \prod_{k=1}^{n} a^k(u_k)$$

where $\vec{x} = [x_1, \cdots, x_n]^T$, $\vec{z} = [z_1, \cdots, z_n]^T \in \mathbb{R}^n$, $\vec{u} = [u_1, \cdots, u_n]^T = \vec{x} - \vec{z}$. Then

$$\begin{aligned}
\mathcal{F}[K](\vec{\omega}) &= \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}^n} e^{-i\langle \vec{\omega}, \vec{u} \rangle} \prod_{k=1}^{n} a^k(u_k) d\vec{u} \\
&= \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}^n} \prod_{k=1}^{n} a^k(u_k) e^{-i\omega_k u_k} d\vec{u} \\
&= \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} a^k(u_k) e^{-i\omega_k u_k} du_k
\end{aligned}$$

which is nonnegative since $a^k, k = 1, \cdots, n$, are positive definite functions (Corollary 3.10). □

It might seem that the positive definite assumption on reference functions is quite restrictive. In fact, many commonly used reference functions are indeed positive definite. An incomplete list is given in Table I.

More generally, the weighted summation (with positive weights) and the product of positive definite functions are still positive definite (a direct conclusion from the linearity and product/convolution properties of the Fourier transform). So we can get a class of positive definite

TABLE I

A list of positive definite reference functions and their Fourier transform.

	Reference Function	Fourier Transform
Symmetric Triangle	$\mu(x) = \max(1 - d\,\lvert x\rvert, 0),\ \ d > 0$	$\mathcal{F}[\mu](\omega) = \frac{1}{\sqrt{2\pi}} \frac{\sin^2(\frac{\omega}{2d})}{(\frac{\omega}{2d})^2}$
Gaussian	$\mu(x) = e^{-dx^2},\ \ d > 0$	$\mathcal{F}[\mu](\omega) = \frac{1}{\sqrt{2d}} e^{-\frac{\omega^2}{4d}}$
Cauchy	$\mu(x) = \frac{1}{1+dx^2},\ \ d > 0$	$\mathcal{F}[\mu](\omega) = \sqrt{\frac{\pi}{2d}} e^{-\frac{\lvert\omega\rvert}{\sqrt{d}}}$
Laplace	$\mu(x) = e^{-d\lvert x\rvert},\ \ d > 0$	$\mathcal{F}[\mu](\omega) = \sqrt{\frac{2}{\pi}} \frac{d}{d^2+\omega^2}$
Hyperbolic Secant	$\mu(x) = \frac{2}{e^{dx}+e^{-dx}},\ \ d > 0$	$\mathcal{F}[\mu](\omega) = \frac{1}{d}\sqrt{\frac{\pi}{2}} \frac{2}{e^{\frac{\pi\omega}{2d}}+e^{-\frac{\pi\omega}{2d}}}$
Squared sinc	$\mu(x) = \frac{\sin^2(dx)}{d^2x^2},\ \ d > 0$	$\mathcal{F}[\mu](\omega) = \max(\sqrt{\frac{\pi}{2}}(\frac{1}{d} - \frac{\lvert\omega\rvert}{2d^2}), 0)$

membership functions from those listed above. It is worthwhile noting that the asymmetric triangle and the trapezoid membership functions are not positive definite.

3.3 The PDFC and Mercer Features

Recall the expansion (16) given by the Mercer Theorem. Let $\mathbb{F}$ be an l_2 space. If we define a nonlinear mapping $\Phi : \mathbb{X} \to \mathbb{F}$ as

$$\Phi(\vec{x}) = [\sqrt{\lambda_1}\phi_1(\vec{x}), \cdots, \sqrt{\lambda_k}\phi_i(\vec{x}), \cdots]^T \quad , \tag{17}$$

and define an inner product in $\mathbb{F}$ as

$$\left\langle [u_1, \cdots, u_i, \cdots]^T, [v_1, \cdots, v_i, \cdots]^T \right\rangle_{\mathbb{F}} = \sum_{i=1}^{\infty} u_i v_i \quad , \tag{18}$$

then (16) becomes

$$K(\vec{x}, \vec{z}) = \langle \Phi(\vec{x}), \Phi(\vec{z}) \rangle_{\mathbb{F}} \quad . \tag{19}$$

$\Phi(\vec{x}) \in \mathbb{F}$ is sometimes referred to as the Mercer features. Equation (19) displays a nice property of Mercer kernels: a Mercer kernel implicitly defines a nonlinear mapping Φ such that the kernel computes the inner product in the space Φ maps to. Therefore a Mercer kernel enables a classifier, in the form of (12), to work on Mercer features (which usually reside in a space with dimension much higher than that of the input space) without explicitly evaluating the Mercer features (which is computationally very expensive). The following theorem illustrates the relationship between the PDFCs and Mercer features.

Theorem 3.12: *Given n positive definite reference functions, $a^k : \mathbb{R} \to [0, 1]$, $k = 1, \cdots, n$, and a compact set $\mathbb{X} \subset \mathbb{R}^n$, we define a Mercer kernel $K(\vec{x}, \vec{z}) = \prod_{k=1}^{n} a^k(x_k - z_k)$ where $\vec{x} = [x_1, \cdots, x_n]^T, \vec{z} = [z_1, \cdots, z_n]^T \in \mathbb{X}$. Let $\mathbb{F}$ be an l_2 space, $\Phi : \mathbb{X} \to \mathbb{F}$ be the nonlinear*

mapping given by (17), and $\langle \cdot, \cdot \rangle_{\mathbb{F}}$ *be an inner product in* $\mathbb{F}$ *defined by (18). Given a set of points* $\{\vec{z}_1, \cdots, \vec{z}_m\} \subset \mathbb{X}$*, we define a subspace* $\mathbb{W} \subset \mathbb{F}$ *as* $\mathbb{W} = \text{Span}\{\Phi(\vec{z}_1), \cdots, \Phi(\vec{z}_m)\}$*, and a function space* $\mathbb{H}$ *on* $\mathbb{F}$ *as* $\mathbb{H} = \{h : h(\vec{u}) = \text{sign}(\langle \vec{w}, \vec{u} \rangle_{\mathbb{F}} + b_0),\ \vec{w} \in \mathbb{W},\ \vec{u} \in \mathbb{F},\ b_0 \in \mathbb{R}\}$*. Then we have the following results:*

1. For any $g \in \mathbb{H}$*, there exists a PDFC with* $a^k,\ k = 1, \cdots, n$*, as reference functions such that the decision rule,* f*, of the PDFC satisfies* $f(\vec{x}) = g(\Phi(\vec{x})),\ \forall \vec{x} \in \mathbb{X}$*.*

2. For any PDFC using $a^k,\ k = 1, \cdots, n$*, as reference functions, if* $\vec{z}_j$ *contains location parameters of the IF-part membership functions associated with the* j*th fuzzy rule for* $j = 1, \cdots, m$ *(as defined in Corollary 3.5), then there exists* $g \in \mathbb{H}$ *such that the decision rule,* f*, of the PDFC satisfies* $f(\vec{x}) = g(\Phi(\vec{x})),\ \forall \vec{x} \in \mathbb{X}$*.*

Proof:

1. Given $g \in \mathbb{H}$, we have $g(\vec{u}) = \text{sign}(\langle \vec{w}, \vec{u} \rangle_{\mathbb{F}} + b_0)$. Since $\vec{w} \in \mathbb{W}$, it can be written as a linear combination of $\Phi(\vec{z}_j)$'s, i.e., $\vec{w} = \sum_{j=1}^{m} b_j \Phi(\vec{z}_j)$. Thus $g(\vec{u})$ becomes

$$\begin{aligned} g(\vec{u}) &= \text{sign}\left(\left\langle \sum_{j=1}^{m} b_j \Phi(\vec{z}_j), \vec{u} \right\rangle_{\mathbb{F}} + b_0 \right) \\ &= \text{sign}\left(\sum_{j=1}^{m} b_j \langle \Phi(\vec{z}_j), \vec{u} \rangle_{\mathbb{F}} + b_0 \right) \quad . \end{aligned}$$

Now we can define a PDFC using $a^k,\ k = 1, \cdots, n$, as reference functions. For $j = 1, \cdots, m$, let $\vec{z}_j$ contain location parameters of the IF-part membership functions associated with the jth fuzzy rule (as defined in Corollary 3.5), and b_j be the THEN-part of the jth fuzzy rule. The THEN-part of Rule 0 is b_0. Then from (12) and (19), the decision rule is

$$\begin{aligned} f(\vec{x}) &= \text{sign}\left(\sum_{j=1}^{m} b_j K(\vec{x}, \vec{z}_j) + b_0 \right) \\ &= \text{sign}\left(\sum_{j=1}^{m} b_j \langle \Phi(\vec{x}), \Phi(\vec{z}_j) \rangle_{\mathbb{F}} + b_0 \right) \end{aligned}$$

Clearly, $f(\vec{x}) = g(\Phi(\vec{x})),\ \forall \vec{x} \in \mathbb{X}$.

2. For a PDFC described in the theorem, let b_j be the THEN-part of the jth fuzzy rule, and b_0 be the THEN-part of Rule 0. Then from (12) and (19), the decision rule is

$$f(\vec{x}) = \text{sign}\left(\sum_{j=1}^{m} b_j \langle \Phi(\vec{x}), \Phi(\vec{z}_j) \rangle_{\mathbb{F}} + b_0 \right)$$

$$= \quad \text{sign}\left(\left\langle \sum_{j=1}^{m} b_j \Phi(\vec{z}_j), \Phi(\vec{x}) \right\rangle_{\mathbb{F}} + b_0\right) \quad .$$

Let $\vec{w} = \sum_{j=1}^{m} b_j \Phi(\vec{z}_j)$ and $g(\vec{u}) = \text{sign}(\langle \vec{w}, \vec{u} \rangle_{\mathbb{F}} + b_0)$, then $g \in \mathbb{H}$ and $f(\vec{x}) = g(\Phi(\vec{x}))$, $\forall \vec{x} \in \mathbb{X}$.

This completes the proof. □

Remark 3.13: *The compactness of the input domain $\mathbb{X}$ is required for purely theoretical reason: it ensures that the expansion (16) can be written in a form of countable sum, thus the nonlinear mapping (17) can be defined. In practice, we don't need to worry about it provided that all input features (both training and testing) are within certain range (which can be satisfied via data preprocessing). Consequently, it is reasonable to assume that $\vec{z}_j$ is also in $\mathbb{X}$ for $j = 1, \cdots, m$ because this essentially requires that all fuzzy rule "patches" center inside the input domain.*

Remark 3.14: *Since $g(\vec{u}) = \text{sign}(\langle \vec{w}, \vec{u} \rangle_{\mathbb{F}} + b) = 0$ defines a hyperplane in $\mathbb{F}$, Theorem 3.12 relates the decision boundary of a PDFC in $\mathbb{X}$ to a hyperplane in $\mathbb{F}$. The theorem implies that given any hyperplane in $\mathbb{F}$, if its orientation (normal direction pointed by $\vec{w}$) is a linear combination of vectors that have preimage (under Φ) in $\mathbb{X}$, then the hyperplane transforms to a decision boundary of a PDFC. Conversely, given a PDFC, one can find a hyperplane in $\mathbb{F}$ that transforms to the decision boundary of the given PDFC. Therefore, we can alternatively consider the decision boundary of a PDFC as a hyperplane in the feature space $\mathbb{F}$, which corresponds to a nonlinear decision boundary in $\mathbb{X}$. Constructing a PDFC is then converted to finding a hyperplane in $\mathbb{F}$.*

Remark 3.15: *A hyperplane in $\mathbb{F}$ is defined by its normal direction $\vec{w}$ and the distance to the origin, which is determined by b for fixed $\vec{w}$. According to the proof of Theorem 3.12, $\vec{w}$ and b are defined as $\vec{w} = \sum_{j=1}^{m} b_j \Phi(\vec{z}_j)$ and $b = b_0$, respectively, where $\{\vec{z}_1, \cdots, \vec{z}_m\} \subset \mathbb{X}$ is the set of location parameters of the IF-part fuzzy rules, and $\{b_0, \cdots, b_m\} \subset \mathbb{R}$ is the set of constants in the THEN-part fuzzy rules. This implies that the IF-part and THEN-part of fuzzy rules play different roles in modeling the hyperplane. The IF-part parameters, $\{\vec{z}_1, \cdots, \vec{z}_m\}$, defines a set of feasible orientations, $\mathbb{W} = \text{Span}\{\Phi(\vec{z}_1), \cdots, \Phi(\vec{z}_m)\}$, of the hyperplane. The THEN-part parameters $\{b_1, \cdots, b_m\}$ select an orientation, $\sum_{j=1}^{m} b_j \Phi(\vec{z}_j)$, from $\mathbb{W}$. The distance to the origin is then determined by the THEN-part of Rule 0, i.e., $b = b_0$.*

4 An SVM Approach to Build PDFCs

A PDFC with n inputs and m, which is unknown, fuzzy rules is parameterized by n, possibly different, positive definite reference functions ($a^k : \mathbb{R} \to [0,1],\ k = 1, ...n$), a set of location parameters ($\{\vec{z}_1, \cdots, \vec{z}_m\} \subset \mathbb{X}$) for the membership functions of the IF-part fuzzy rules, and a set of real numbers ($\{b_0, \cdots, b_m\} \subset \mathbb{R}$) for the constants in the THEN-part fuzzy rules. Which reference functions to choose is an interesting research topic by itself [33]. But it is out of the scope of this article. Here we assume that the reference functions $a^i : \mathbb{R} \to [0,1],\ i = 1, \cdots, n$ are predetermined. So the remaining question is how to find a set of fuzzy rules ($\{\vec{z}_1, \cdots, \vec{z}_m\}$ and $\{b_0, \cdots, b_m\}$) from the given training samples $\{(\vec{x}_1, y_1), \cdots, (\vec{x}_l, y_l)\} \subset \mathbb{X} \times \{+1, -1\}$ so that the PDFC has good generalization.

As given in (13), for a PDFC, a Mercer kernel can be constructed from the positive definite reference functions. The kernel implicitly defines a nonlinear mapping Φ that maps $\mathbb{X}$ into a kernel-induced feature space $\mathbb{F}$. Theorem 3.12 states that the decision rule of a PDFC can be viewed as a hyperplane in $\mathbb{F}$. Therefore, the original question transforms to: given training samples $\{(\Phi(\vec{x}_1), y_1), \cdots, (\Phi(\vec{x}_l), y_l)\} \subset \mathbb{F} \times \{+1, -1\}$, how to find a separating hyperplane in $\mathbb{F}$ that yields good generalization, and how to extract fuzzy rules from the obtained optimal hyperplane. We have seen in Section 2.2 that the SVM algorithm finds a separating hyperplane (in the input space or the kernel induced feature space) with good generalization by reducing the empirical risk and, at the same time, controlling the hyperplane margin. Thus we can use the SVM algorithm to find an optimal hyperplane in $\mathbb{F}$. Once we get such a hyperplane, fuzzy rules can be easily extracted. The whole procedure is described by the following algorithm.

Algorithm 4.1: SVM Learning for PDFC

Inputs: *Positive definite reference functions $a^k(x_k)$, $k = 1, \cdots, n$, associated with n input variables, and a set of training samples $\{(\vec{x}_1, y_1), \cdots, (\vec{x}_l, y_l)\}$.*

Outputs: *A set of fuzzy rules parameterized by $\vec{z}_j$, b_j, and m. $\vec{z}_j$ ($j = 1, \cdots, m$) contains the location parameters of the IF-part membership functions of the jth fuzzy rule, b_j ($j = 0, \cdots, m$) is the THEN-part constant of the jth fuzzy rule, and $m + 1$ is the number of fuzzy rules.*

Steps:

1 *Construct a Mercer kernel, K, from the given positive definite reference functions according to (13).*

2 *Construct an SVM to get a decision rule of the form (6):*

1) Assign some positive number to C, and solve the quadratic program defined by (5) to get the Lagrange multipliers $\vec{\alpha}$.

2) Find b (details can be found in, for example, [7]).

3 Extracting fuzzy rules from the decision rule of the SVM:

$b_0 \leftarrow b$

$j \leftarrow 1$

FOR $i = 1$ **TO** l

IF $\alpha_i > 0$

$\vec{z}_j \leftarrow \vec{x}_i$

$b_j \leftarrow y_i \alpha_i$

$j \leftarrow j + 1$

END IF

END FOR

$m \leftarrow j - 1$

It is straightforward to check that the decision rule of the resulting PDFC is identical to (6).

Once reference functions are fixed, the only free parameter in the above algorithm is C. According to the optimization criterion in (2), C weights the classification error versus the upper bound on the VC dimension. Another way of interpreting C is that it affects the sparsity of $\vec{\alpha}$ (the number of nonzero entries in $\vec{\alpha}$) [4]. Unfortunately, there is no general rule for picking C. Typically, a range of values of C should be tried before the best one can be selected.

The above learning algorithm has several nice properties:

- The shape of the reference functions and C parameter are the only prior information needed by the algorithm.
- The algorithm automatically generates a set of fuzzy rules. The number of fuzzy rules is irrelevant to the dimension of the input space. It equals the number of nonzero Lagrange multipliers. In this sense, the "curse of dimensionality" is avoided. In addition, due to the sparsity of $\vec{\alpha}$, the number of fuzzy rules is usually much less than the number of training samples.
- Each fuzzy rule is parameterized by a training sample $(\vec{x}_j, y_j)$ and the associated nonzero Lagrange multiplier α_j where $\vec{x}_j$ specifies the location of the IF-part membership functions,

and $y_j\alpha_j$ gives the THEN-part constant.

• The global solution for the optimization problem can always be found efficiently because of the convexity of the objective function and of the feasible region. Algorithms designed specifically for the quadratic programming problems in SVMs make large-scale training (for example $200,000$ samples with $40,000$ input variables) practical [23], [25], [37]. The computational complexity of classification operation is determined by the cost of kernel evaluation and the number of support vectors.

• Since the goal of optimization is to lower an upper bound on the expected risk (not just the empirical risk), the resulting PDFC usually has good generalization, which will be demonstrated in the coming section.

5 Experimental Results

Using Algorithm 4.1, we design PDFCs with different choices of reference functions [4]. Based on the IRIS data set [3] and the USPS data set [5], we evaluate the performance of PDFCs in terms of generalization (classification rate) and number of fuzzy rules. Comparisons with fuzzy classifiers described in [19] and results in [35] are also provided.

5.1 IRIS Data Set

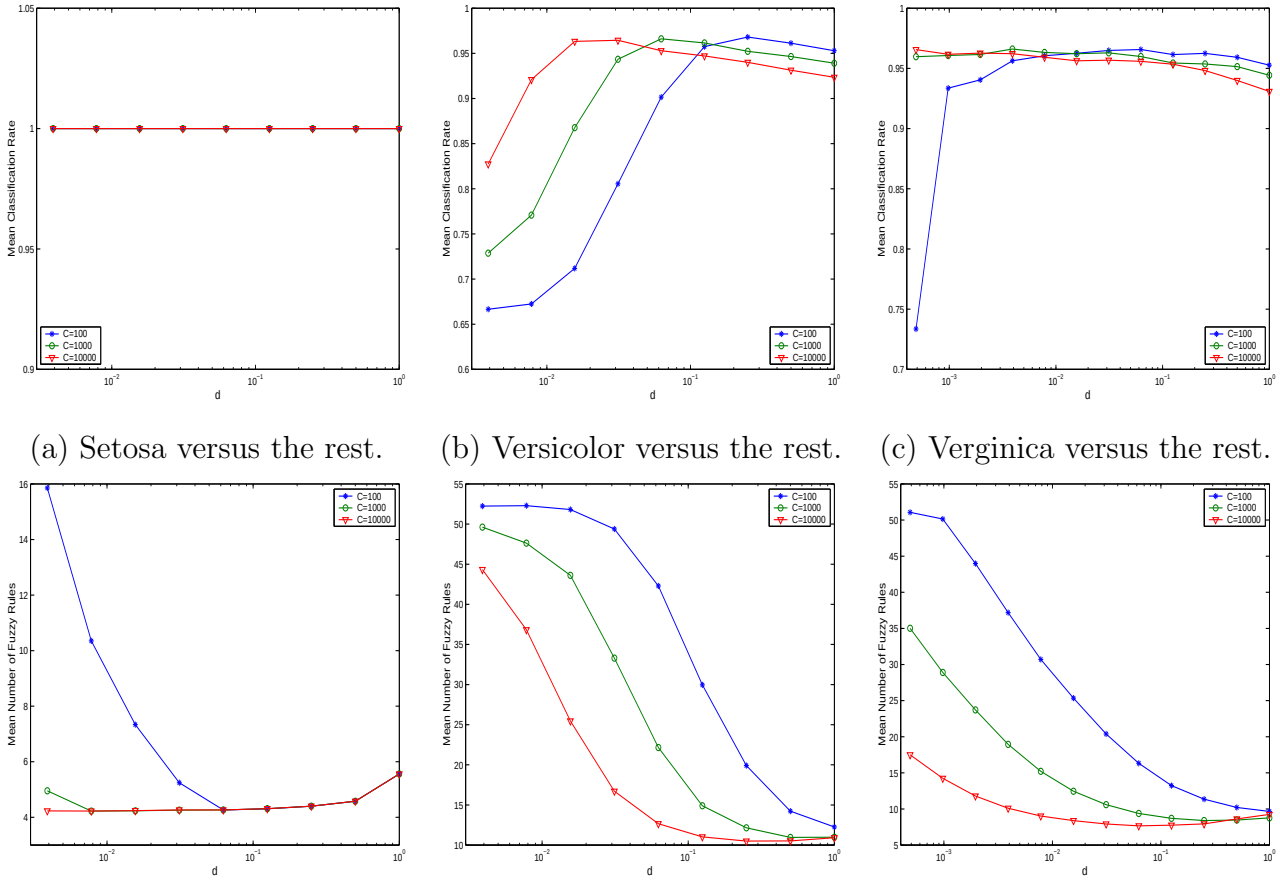
The IRIS data set consists of 150 samples belonging to 3 classes of iris plants namely Setosa, Versicolor, and Verginica. Each class contains 50 samples, and each sample is represented by four input features (sepal length, sepal width, petal length, and petal width) and the associated class label. The Setosa class is linearly separable from the Versicolor and Verginica classes, the latter are not linearly separable from each other. Clearly, this is a multi-class classification problem. But the Algorithm 4.1 only works for binary classifiers. So we design three PDFCs, each of which separates one class from the rest two classes. The final predicted class label is decided by the winner of three PDFCs, i.e., one with the maximum un-thresholded output.

The generalization performance is evaluated via 2-fold cross-validation. The IRIS data set is randomly divided into two subsets of equal size (75 samples). A PDFC is trained 2 times, each time with a different subset held out as a validation set. The classification rate is then defined as the number of correctly classified validation samples divided by the size of the validation set. We repeat the 2-fold cross-validation 200 times using different partitions of the IRIS data

[4] The SVMLight [23] is used to implement the SVMs. This software is available at `http://svmlight.joachims.org`.

[5] The USPS data set is available at `http://www.kernel-machines.org/data`.

set, and compute the mean of the classification rates. This quantity is viewed as an estimation of the generalization performance.



(a) Setosa versus the rest. (b) Versicolor versus the rest. (c) Verginica versus the rest.

(d) Setosa versus the rest. (e) Versicolor versus the rest. (f) Verginica versus the rest.

Fig. 2. Performance of PDFCs in terms of the mean classification rate and the mean number of fuzzy rules for the IRIS data set. (a) and (d) give the mean classification rate and the mean number of fuzzy rules, respectively, of PDFCs designed to separate Setosa class from the other two classes. (b) and (e) give the mean classification rate and the mean number of fuzzy rules, respectively, of PDFCs designed to separate Versicolor class from the other two classes. (c) and (f) give the mean classification rate and the mean number of fuzzy rules, respectively, of PDFCs designed to separate Verginica class from the other two classes.

For all input variables, we use the Gaussian reference function given in Table I. PDFCs are designed for different values of C (in Algorithm 4.1) and d (of the Gaussian reference function). The mean classification rate and the mean number of fuzzy rules for different values of C and d are plotted in Figure 2. Separating the Setosa class from the other two classes is relatively easy since they are linearly separable. Consequently, as shown in Figure 2(a), the PDFCs generalizes perfectly for all values of C and d. Separating the Versicolor (or Verginica) class from the rest two classes requires slightly more efforts. Figure 2(b) and (c) show that the generalization

TABLE II

MEAN CLASSIFICATION RATE r AND MEAN NUMBER OF FUZZY RULES m (FOR MULTI-CLASS CLASSIFIERS). A COMPARISON OF MULTI-CLASS CLASSIFIERS CONSTRUCTED FROM THREE PDFCS AND FUZZY CLASSIFIERS BUILT FROM ISHIBUCHI'S APPROACH USING THE IRIS DATA SET.

	Combining 3 PDFCs ($C = 100$)					Ishibuchi's Approach [19]				
	$d = \frac{1}{16}$	$d = \frac{1}{8}$	$d = \frac{1}{4}$	$d = \frac{1}{2}$	$d = 1.0$	$M = 2$	$M = 3$	$M = 4$	$M = 5$	$M = 6$
r	95.46%	96.22%	96.38%	95.97%	95.55%	91.73%	94.80%	94.53%	94.80%	95.37%
m	62.49	47.5	35.46	28.695	26.69	16	81	256	625	1296

performance depends on the choices of C and d. However, for different values of C, we get very similar generalization performance by picking a proper d value. In Figure 2(b), the maximum mean classification rates for $C = 100$, 1000, and 10000 are 96.81% ($d = \frac{1}{4}$), 96.61% ($d = \frac{1}{16}$), and 96.45% ($d = \frac{1}{32}$), respectively. In Figure 2(c), the maximum mean classification rates for $C = 100$, 1000, and 10000 are 96.57% ($d = \frac{1}{16}$), 96.61% ($d = \frac{1}{256}$), and 96.56% ($d = \frac{1}{2048}$), respectively. Moreover, Figure 2(d), (e), and (f) demonstrate that C affects the number of fuzzy rules. For a fixed value of d, a larger C value corresponds to a smaller mean number of fuzzy rules. This complies with the observation in the SVM literature that the number of support vectors decreases when C is large.

To get the final multi-class classifier, we need to combine three PDFCs (each one is designed to separate one class from the rest two classes). Here we use the following strategy:

- Pick three PDFCs with the same C and d values.
- The predicted class label is given by the PDFC with the maximum un-thresholded output.

This strategy is by no means optimal. But it is very simple, and works very well. The results for $C = 100$, $d = 1$, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, and $\frac{1}{16}$ are summarized in Table II, where we also cite the results reported by Ishibuchi *et al.* [19]. In their approach, input features are normalized to the interval $[0, 1]$, and each axis of the input space is assigned M uniformly distributed fuzzy sets. The rule weights and THEN-part of fuzzy rules are determined by a reward-and-punishment scheme [19]. Clearly, the number of fuzzy rules for such a system is M^4.

From Table II we can see that the classification rates of classifiers built on PDFCs (with a range of d values) are higher than those of the classifiers constructed from Ishibuchi's approach. Moreover, the number of fuzzy rules used by PDFCs is less than that of Ishibuchi's approach (except for $M = 2$ which gives a less favorable classification rate of 91.73%). In addition, for

TABLE III

USPS DATA SET. MEAN CLASSIFICATION RATE $r \pm$ STANDARD DEVIATION AND MEAN NUMBER OF FUZZY RULES m (FOR ONE PDFC) USING DIFFERENT REFERENCE FUNCTIONS.

	Gaussian	Cauchy	Laplace	S-Triangle	H-Secant	Sinc^2
r	$95.2\% \pm 0.3\%$	$95.2\% \pm 0.3\%$	$94.7\% \pm 0.4\%$	$95.0\% \pm 0.3\%$	$95.0\% \pm 0.3\%$	$95.2\% \pm 0.2\%$
m	573	567	685	652	468	391

a PDFC, the number of fuzzy rules is bounded above by the number of training samples since each fuzzy rule is parameterized by a training sample with nonzero Lagrange multiplier. While, using Ishibuchi's approach, the number of fuzzy rules increases exponentially as M^4.

5.2 USPS Data Set

The USPS data set contains 9298 grayscale images of handwritten digits. The images are size normalized to fit in a 16×16 pixel box while preserving their aspect ratio. The data set is divided into a training set of 7291 samples and a testing set of 2007 samples. For each sample, the input feature vector consists of 256 grayscale values.

In this experiment, we test the performance of PDFCs for different choices of reference functions given in Table I. For different input variables, the reference functions are chosen to be identical. Ten PDFCs are designed, each of which separates one digit from the rest nine digits. The final predicted class label is decided by the PDFC with the maximum un-thresholded output. Based on the training set, we use 5-fold cross-validation to determine the d parameter of reference functions and the C parameter in support vector learning (for each PDFC) where C takes values from $\{100, 1000, 10000\}$, and d takes values from $\{\frac{1}{2^n} : n = 2, \cdots, 10\}$. For each pair of d and C, the average cross-validation error is computed. The optimal d and C are the values that gives the minimal mean cross-validation error. Based on the selected parameter, the PDFCs are constructed and evaluated on the testing set. The whole process is repeated 5 times. The mean classification rate (and the standard deviation) on the testing set and the mean number of fuzzy rules (for one PDFC) are listed in Table III. For comparison purpose, we also cite the results from [35]: linear SVM (classification rate 91.3%), k-nearest neighbor (classification rate 94.3%), SVM with Gaussian kernel (classification rate 95.8%), and virtual SVM (classification rate 97.0%).

Note that the Gaussian reference function corresponds to the Gaussian RBF kernel used in

the SVM literature. For the USPS data, all six reference functions achieve similar classification rates. The number of fuzzy rules varies significantly. The number of fuzzy rules needed by the squared sinc reference function is only 68.2% of that needed by the Gaussian reference function. Compared with the linear SVM and k-nearest neighbor approach [35], the PDFCs achieve a better classification rate. SVMs can be improved by using prior knowledge. For instance the virtual SVM [35] performs better than current PDFCs. However, same approach can be applied to build PDFCs, i.e., PDFCs can also benefit from the same prior knowledge.

6 Discussion

6.1 The Relationship between PDFC kernels and RBF Kernels

In the literature, it is well-known that a Gaussian RBF network can be trained via support vector learning using a Gaussian RBF kernel [41]. While the functional equivalence between fuzzy inference systems and Gaussian RBF networks is established in [21] where the membership functions within each rule must be Gaussian functions with identical variance. So connection between such fuzzy systems and SVMs with Gaussian RBF kernels can be established. The following discussion compares the kernels defined by PDFCs and RBF kernels commonly used in SVMs.

The kernels of PDFCs are constructed from positive definite reference functions. These kernels are translation invariant, symmetric with respect to a set of orthogonal axes, and tailing off gradually. In this sense, they appear to be very similar to the general RBF kernels [16]. In fact, the Gaussian reference function defines the Gaussian RBF kernel. However, in general, the kernels of PDFCs are not RBF kernels. According to the definition, an RBF kernel, $K(\vec{x}, \vec{z})$, depends only on the norm of $\vec{x} - \vec{z}$, i.e., $K(\vec{x} - \vec{z}) = K_{RBF}(\|\vec{x} - \vec{z}\|)$. It can be shown that for a kernel, $K(\vec{x}, \vec{z})$, defined by (13) using symmetric triangle, Cauchy, Laplace, hyperbolic secant, or squared sinc reference functions (even with identical d for all input variables), there exists $\vec{x}_1$, $\vec{x}_2$, $\vec{z}_1$, and $\vec{z}_2$ such that $\|\vec{x}_1 - \vec{z}_1\| = \|\vec{x}_2 - \vec{z}_2\|$ and $K(\vec{x}_1, \vec{z}_1) \neq K(\vec{x}_2, \vec{z}_2)$. Moreover, a general RBF kernels (even if it is a Mercer kernel) may not be a PDFC kernel, i.e., it can not be in general decomposed as product of positive definite reference functions. It is worth noting that the kernel defined by symmetric triangle reference functions is identical to the B_n-splines (or order 1) kernel that is commonly used in the SVM literature [55].

6.2 Advantages of Connecting Fuzzy Systems to Kernel Machines

Kernel methods represent one of the most important directions both in theory and application of machine learning. While fuzzy classifier was regarded as a method that "are cumbersome to use in high dimensions or on complex problems or in problems with dozens or hundreds of features (pp. 194, [13])." Establishing the connection between fuzzy systems and kernel machines has the following advantages:

• A novel kernel perspective of fuzzy classifiers is provided. Through reference functions, fuzzy rules are related to translation invariant kernels. Fuzzy inference on the IF-part of a fuzzy rule is equivalent to evaluating the kernel. If the reference functions are restricted to the class of positive definite functions then the kernel turns out to be a Mercer kernel, and the corresponding fuzzy classifier becomes a PDFC. Since Mercer kernel induces a feature space, we can consider the decision boundary of a PDFC as a hyperplane in that space. The design of a PDFC is then equivalent to finding an "optimal" hyperplane.

• A new approach to build fuzzy classifiers is proposed. Based on the link between fuzzy systems and kernel machines, a support vector learning approach is proposed to construct PDFCs so that a fuzzy classifier can have good generalization ability in a high dimensional feature space. The resulting fuzzy rules are determined by support vectors, corresponding Lagrange multipliers, and associated class labels.

• It points out a future direction of applying techniques in fuzzy systems literature to improve the performance of kernel methods. The link between fuzzy systems and kernel machines implies that a class of kernel machines, such as those using Gaussian kernels, can be interpreted by a set of fuzzy IF-THEN rules. This opens interesting connections between fuzzy rule base reduction techniques [43] and computational complexity issues in SVMs [6] and kernel PCA (principal component analysis) [40]:

– The computational complexity of an SVM scales with the number of support vectors. One way of decreasing the complexity is to reduce the number of support-vector-like vectors in the decision rule (6). For the class of kernels, which can be interpreted by a set of fuzzy IF-THEN rules, this can be viewed as fuzzy rule base simplification.

– In kernel PCA [40], given a test point $\vec{x}$, the kth nonlinear principal component, β_k, is computed by $\beta_k = \sum_{i=1}^{l} \alpha_i^k K(\vec{x}, \vec{x}_i)$ where l is the number of data points in a given data set (details of calculating $\alpha_i^k \in \mathbb{R}$ can be found in [40]). Therefore, the computational complexity of computing β_k scales with l. For the class of kernels discussed in this paper, it is not difficult

to derive that β_k can be equivalently viewed as the output of an additive fuzzy system using first order moment defuzzification without thresholding unit. Here $\vec{x}_i$ and α_i^k parameterize the IF-part and THEN-part of the ith fuzzy rule ($i = 1, \cdots, l$), respectively. As a result, fuzzy rule base reduction techniques may be applied to increase the speed of nonlinear principal components calculation.

7 Conclusions and Future Work

In this paper, we exhibit the connection between fuzzy classifiers and kernel machines, and propose a support vector learning approach to construct fuzzy classifiers so that a fuzzy classifier can have good generalization ability in a high dimensional feature space. As future work, we intend to explore in the following directions: 1) The requirement that all membership functions associated with an input variable are generated from the same reference function maybe somewhat restrictive. However, it can be shown that this constraint can be relaxed; 2) The positivity requirement on reference functions can also be relaxed. In that case, the kernel in general will not be a Mercer kernel. But the fuzzy classifiers can still be related to the generalized support vector machines [31]; 3) Although our work focuses on the classification problem, it is not difficult to extend the results to function approximations. Fuzzy function approximation (using positive definite reference functions) is equivalent to support vector regression [55] using the kernel defined by reference functions; 4) Apply fuzzy rule base reduction techniques to reduce computational complexities of the SVM and kernel PCA.

Acknowledgments

The material is based upon work supported by The Pennsylvania State University, the PNC Foundation, the National Science Foundation under Grant No. IIS-0219272, and SUN Microsystems under Grant EDUD-7824-010456-US. The authors would like to thank anonymous reviewers and the associate editor for their comments that led to improvements of the paper.

References

[1] S. Abe and R. Thawonmas, "A Fuzzy Classifier with Ellipsoidal Regions," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 358-368, 1997.

[2] P. L. Bartlett, "For Valid Generalization, the Size of the Weights is More Important Than the Size of the Network," in *Advances in Neural Information Processing Systems 9*, M.C. Mozer, M.I. Jordan, and T. Petsche, (eds.), Cambridge, MA: The MIT Press, pp. 134-140, 1997.

[3] C. L. Blake and C. J. Merz, "UCI Repository of Machine Learning Databases," [http://www.ics.uci.edu/∼mlearn/MLRepository], University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

[4] P. S. Bradley and O. L. Mangasarian, "Feature Selection via Concave Minimization and Support Vector Machines," *Proceedings of the 15th International Conference on Machine Learning*, pp. 82-90, Morgan Kaufmann, San Francisco, CA, 1998.

[5] C. J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[6] C. J.C. Burges and B. Schölkopf, "Improving the Accuracy and Speed of Support Vector Machines," in *Advances in Neural Information Processing Systems 9*, M.C. Mozer, M.I. Jordan, and T. Petsche, (eds.), Cambridge, MA: The MIT Press, pp. 375-381, 1997.

[7] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," [http://www.csie.ntu.edu.tw/∼cjlin/libsvm], 2001.

[8] S.-M. Chen, Y.-J. Horng, and C.-H. Lee, "Document Retrieval Using Fuzzy-Valued Concept Networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 1, pp. 111-118, 2001.

[9] C.-K. Chiang, H.-Y, Chung, and J.-J Lin, "A Self-Learning Fuzzy Logic Controller Using Genetic Algorithms with Reinforcements," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 460-467, 1997.

[10] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.

[11] J. A. Dickerson and B. Kosko, "Fuzzy Function Approximation with Ellipsoidal Rules," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 4, pp. 542-560, 1996.

[12] D. Dubois and H. Prade, "Operations on Fuzzy Numbers," *International Journal of Systems Science*, vol. 9, no. 6, pp. 613-626, 1978.

[13] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Second Edition, John Wiley and Sons, Inc., 2000.

[14] M. R. Emami, I. B. Türksen, and A. A. Goldenberg, "Development of a Systematic Methodology of Fuzzy Logic Modeling," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 346-361, 1998.

[15] S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1-58, 1992.

[16] M. G. Genton, "Classes of Kernels for Machine Learning: A Statistics Perspective," *Journal of Machine Learning Research*, vol. 2, pp. 299-312, 2001.

[17] R. J. Hathaway and J. C. Bezdek, "Fuzzy c-means Clustering of Incomplete Data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 5, pp. 735-744, 2001.

[18] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[19] H. Ishibuchi and T. Nakashima, "Effect of Rule Weights in Fuzzy Rule-Based Classification Systems," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 506-515, 2001.

[20] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Construction of Fuzzy Classification Systems with Rectangular Fuzzy Rules Using Genetic Algorithms," *Fuzzy Sets and Systems*, vol. 65, pp. 237-253, 1994.

[21] J.-S.R. Jang and C. T. Sun, "Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 156-159, 1993.

[22] J.-S.R. Jang and C.-T. Sun, "Neuro-Fuzzy Modeling and Control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378-406, 1995.

[23] T. Joachims, "Making Large-Scale SVM Learning Practical," *Advances in Kernel Methods - Support Vector Learning*, edited by B. Schölkopf, C. J.C. Burges, and A.J. Smola, Cambridge, MA: MIT Press, pp. 169-184, 1999.

[24] N. K. Kasabov, "Learning Fuzzy Rules and Approximate Reasoning in Fuzzy Neural Networks and Hybrid Systems," *Fuzzy Sets and Systems*, vol. 82, no. 2, pp. 135-149, 1996.

[25] L. Kaufman, "Solving the Quadratic Programming Problem Arising in Support Vector Classification," *Advances in Kernel Methods - Support Vector Learning*, edited by B. Schölkopf, C. J.C. Burges, and A.J. Smola, Cambridge, MA: MIT Press, pp. 147-167, 1999.

[26] F. Klawon and P. E. Klement, "Mathematical Analysis of Fuzzy Classifiers," in *Lecture Notes in Computer Science*, vol. 1280, pp. 359-370, 1997.

[27] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, 1995.

[28] B. Kosko, "Fuzzy Systems as Universal Approximators," *IEEE Transactions on Computers*, vol. 43, no. 11, pp. 1329-1333, 1994.

[29] L. I. Kuncheva, "How Good Are Fuzzy If-Then Classifiers," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 30, no. 4, pp. 501-509, 2000.

[30] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I, Part II," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404-435, 1990.

[31] O. L. Mangasarian, "Generalized Support Vector Machines," *Advances in Large Margin Classifiers*, edited by A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, Cambridge, MA: MIT Press, pp. 135-146, 2000.

[32] J. Mercer, "Functions of Positive and Negative Type and Their Connection with the Theory of Integral Equations," *Philosophical Transactions of the Royal Society London*, A209, pp. 415-446, 1909.

[33] S. Mitaim and B. Kosko, "The Shape of Fuzzy Sets in Adaptive Function Approximation," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 637-656, 2001.

[34] S. Miyamoto, "Two Approaches for Information Retrieval Through Fuzzy Associations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 1, pp. 123-130, 1989.

[35] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An Introduction to Kernel-Based Learning Algorithms," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181-202, 2001.

[36] P. J. Pacini and B. Kosko, "Adaptive Fuzzy Frequency Hopper," *IEEE Transactions on Communications*, vol. 43, no. 6, pp. 2111-2117, 1995.

[37] J. C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," *Advances in Kernel Methods - Support Vector Learning*, edited by B. Schölkopf, C. J.C. Burges, and A.J. Smola, Cambridge, MA: MIT Press, pp. 185-208, 1999.

[38] R. Rovatti, "Fuzzy Piecewise Multilinear and Piecewise Linear Systems as Universal Approximators in Sobolev Norms," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 2, pp. 235-249, 1998.

[39] F. Sattar and D. B.H. Tay, "Enhancement of Document Images Using Multiresolution and Fuzzy Logic Techniques," *IEEE Signal Processing Letters*, vol. 6, no. 10, pp. 249-252, 1999.

[40] B. Schölkopf, A. J. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, pp. 1299-1319, 1998.

[41] B. Schölkopf, K.-K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Transactions on Signal Processing*, vol.45, no. 11, pp. 2758-2765, 1997.

[42] M. Setnes, "Supervised Fuzzy Clustering for Rule Extraction," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 4, pp. 416-424, 2000.

[43] M. Setnes and R. Babuška, "Rule Base Reduction: Some Comments on the Use of Orthogonal Transforms," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, vol. 31, no. 2, pp. 199-206, 2001.

[44] R. Silipo and M. R. Berthold, "Input Features' Impact on Fuzzy Decision Process," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 6, pp. 821-834, 2000.

[45] A. J. Smola, B. Schölkopf, and K.-R. Müller, "The Connection Between Regularization Operators and Support Vector Kernels," *Neural Networks*, vol. 11, no. 4, pp. 637-649, 1998.

[46] M. Sugeno and G. T. Kang, "Structure Identification of Fuzzy Model," *Fuzzy Sets and Systems*, vol. 28, pp. 15-33, 1988.

[47] Y. Suzuki, K. Itakura, S. Saga, and J. Maeda, "Signal Processing and Pattern Recognition with Soft Computing," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1297-1317, 2001.

[48] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116-132, 1985.

[49] K. Tang, K. Man, Z. Liu, and S. Kwong, "Minimal Fuzzy Memberships and Rules Using Hierarchical Genetic Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 162-169, 1998.

[50] R. Thawonmas and S. Abe, "Function Approximation Based on Fuzzy Rules Extracted From Partitioned Numerical Data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 4, pp. 525-534, 1999.

[51] V. Vapnik, *Estimation of Dependences Based on Empirical Data (in Russian)*, Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

[52] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

[53] V. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, Inc., New York, 1998.

[54] V. Vapnik and A. Chervonenkis, "On the Uniform Convergence of Relative Frequencies of Events to Theirs Probabilities," *Theory of Probability and its Applications*, vol. 16, no. 2, pp. 264-280, 1971.

[55] V. Vapnik, S. E. Golowich, and A. Smola, "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing," in *Advances in Neural Information Processing Systems 9*, M.C. Mozer, M.I. Jordan, and T. Petsche, (eds.), Cambridge, MA: The MIT Press, pp. 281-287, 1997.

[56] L. Wang and J. M. Mendel, "Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least-Squares Learning," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 807-814, 1992.

[57] L.-X. Wang, *Adaptive Fuzzy Systems And Control: Design and Stability Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1994.

[58] L.-X. Wang, "Analysis and Design of Hierarchical Fuzzy Systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 5, pp. 617-624, 1999.

[59] C.-C. Wong and C.-C. Chen "A GA-Based Method for Constructing Fuzzy Systems Directly from Numerical Data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 6, pp. 904-911, 2000.

[60] S. Wu, M. J. Er, and Y. Gao, "A Fast Approach for Automatic Generation of Fuzzy Rules by Generalized Dynamic Fuzzy Neural Networks," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 578-594, 2001.

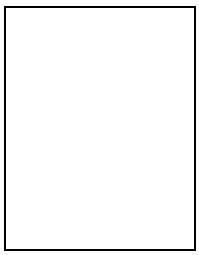
[61] J. Yen, "Fuzzy Logic—A Modern Perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 153-165, 1999.

[62] J. Yen and L. Wang, "Application of Statistical Information Criteria for Optimal Fuzzy Model Construction," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 3, pp. 362-372, 1998.

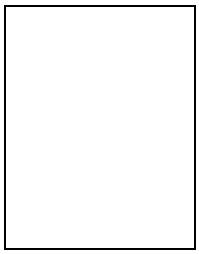
[63] H. Ying, "General SISO Takagi-Sugeno Fuzzy Systems with Linear Rule Consequent are Universal Approximators," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 4, pp. 582-587, 1998.

[64] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338-353, 1965.

[65] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications*, Kluwer Academic Publishers, 1991.



Yixin Chen received the B.S. degree from the Department of Automation, Beijing Polytechnic University, China, in 1995, the M.S. degree in control theory and application from Tsinghua University, China, in 1998, and the M.S. and Ph.D. degrees in electrical engineering from the University of Wyoming, Laramie, WY, in 1999 and 2001, respectively. Since August 2000, he has been a Ph.D student in the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA. He is expected to receive the Ph.D. degree in 2003. His research interests include machine learning, content-based image retrieval, computer vision, precision and fault tolerant robotic control, and soft computing. He is a student member of the IEEE, the IEEE Computer Society, the IEEE Neural Networks Society, and the IEEE Robotics and Automation Society.



James Z. Wang received the Summa Cum Laude bachelor's degree in mathematics and computer science from University of Minnesota (1994), the MSc degree in mathematics and the MSc degree in computer science, both from Stanford University (1997), and the PhD degree in medical information sciences from Stanford University Biomedical Informatics Program and Computer Science Database Group (2000). Since 2000, he has been the holder of the PNC Technologies Career Development Endowed Professorship and an assistant professor at the School of Information Sciences and Technology and the Department of Computer Science and Engineering at The Pennsylvania State University. He has been a visiting scholar at Uppsala University in Sweden, SRI International, IBM Almaden Research Center, and NEC Computer and Communications Research Lab. He is a member of the IEEE.