Kernel Machines and Additive Fuzzy Systems: Classification and Function Approximation

Yixin Chen* and James Z. Wang†‡
*†Dept. of Computer Science and Engineering, ‡School of Information Sciences and Technology
The Pennsylvania State University, University Park, PA 16802
Email: *yixchen@cse.psu.edu, †jwang@ist.psu.edu

Abstract— **This paper investigates the connection between additive fuzzy systems and kernel machines. We prove that, under quite general conditions, these two seemingly quite distinct models are essentially equivalent. As a result, algorithms based upon Support Vector (SV) learning are proposed to build fuzzy systems for classification and function approximation. The performance of the proposed algorithm is illustrated using extensive experimental results.**

Keywords— **Fuzzy systems, support vector machines, support vector regression.**

I. Introduction

Since the publication of L.A. Zadeh's seminal paper on fuzzy sets [32], fuzzy set theory and fuzzy logic have evolved into powerful tools for managing uncertainties inherent in complex systems. In general, building a fuzzy system consists of three basic steps [29]: structure identification (variable selection, partitioning input and output spaces, and choosing membership functions), parameter estimation, and model validation.

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. Given a set of input and output variables, a fuzzy partition associates fuzzy sets 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. One of the commonly used strategies is to assign fixed number of linguistic labels to each input variable. The partition of the output space then follows from supervised learning. Although this scheme is simple to implement, it has two severe drawbacks:

- The performance of the resulting system may be very bad if the input space partition is quite distinct from the distribution of data. Optimizing output space partition alone is not sufficient.
- It 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. They are basically based on data clustering techniques [6], [20], [25].

Although a fuzzy partition can generate fuzzy rules, results are usually very coarse with many parameters needing to be learned and tuned. Various optimization techniques are proposed to solve this problem. Genetic algorithms [4] and artificial neural networks [11] are two of the most popular and effective approaches.

After going through the long journey of structure identification and parameter estimation, can we infer that we get a good fuzzy model? Conclusions could not be drawn without answering the following two questions:

- 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" [18], [31]. 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 (SRM) [26], the bias variance dilemma [8], and the overfitting phenomena [1]. Loosely speaking, a model, build on finite amount of training data, generalizes the best if the right tradeoff is found between the training 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 [24]; (2) focus on model reduction, which is usually achieved by rule base reduction [30], to simplify the model. In statistical learning literature, the Vapnik-Chervonenkis (VC) theory [27] provides a general measure of model set complexity, and gives associated bounds on generalization. However, no efforts have been made to apply the VC theory and the related techniques to construct fuzzy systems. The work presented here tries to bridge this gap.

The remainder of the paper is organized as follows. In Section II, a brief overview of the VC theory and Support Vector Machines (SVMs) is presented. Section III describes the equivalence of kernel machines and a class of additive fuzzy systems. The SV algorithm is then applied to build fuzzy classifiers and function approximators in Section IV. Section V provides the experimental results. And finally,

we conclude in Section VI together with a discussion of future work.

II. VC Theory and Support Vector Machines

For gentle tutorials, we refer interested readers to Burges [2] and Smola *et al.* [22]. More exhaustive treatments can be found in the book by Vapnik [27].

A. 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 [28]: 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

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

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.

B. Support Vector Machines

We give in this section a brief introduction to SV classification, regression, and function approximation, start with SV classifiers. Let $\{(\vec{x}_1, y_1), \cdots, (\vec{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$ be a training set. The SV approach tries 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 $f(\vec{x}) = \mathrm{sgn}\left(\langle \vec{w}, \vec{x} \rangle + b\right)$. Considering that the training set may not be linearly separable, the optimal decision function is found by solving the following quadratic program:

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

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, 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 $\mathbb{H} = \{f(\vec{x}) = \mathrm{sgn}\left(\langle \vec{w}, \vec{x} \rangle + b\right) : (\vec{w}, b) \in \Lambda, \|\vec{w}\| \leq A, \vec{x} \in \mathbb{R}^n\}$ is bounded above by $h \leq \min\left(R^2 A^2, n\right) + 1$.

Thus minimizing $\frac{1}{2} \langle \vec{w}, \vec{w} \rangle$ in (2) amounts to minimizing the VC dimension of the function space $\mathbb{H}$, 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, 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.

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

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

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

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

[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 $\min_{i=1,\cdots,l} |\langle \vec{w}, \vec{x}_i \rangle + b| = 1$.

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 solving the same optimization problem in $\mathbb{F}$, i.e., find 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 $\mathbb{F}$, i.e., $K(\vec{x}, \vec{z}) = \langle \Phi(\vec{x}), \Phi(\vec{z}) \rangle_{\mathbb{F}}$ where $\langle \cdot, \cdot \rangle_{\mathbb{F}}$ is an inner product in $\mathbb{F}$, and Φ is a nonlinear mapping induced by K. Such a function K is also called the Mercer kernel [5]. Substituting $K(\vec{x}_i, \vec{x}_j)$ for $\langle \vec{x}_i, \vec{x}_j \rangle$ in (4) produces a new optimization problem

$$\text{maximize } W(\vec{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j) \quad (5)$$

$$\text{subject to } C \geq \alpha_i \geq 0,\ i = 1, \cdots, l, \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0.$$

Solving it for $\vec{\alpha}$ gives a decision function of the form

$$f(\vec{x}) = \text{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i K(\vec{x}, \vec{x}_i) + b \right) \quad , \qquad (6)$$

whose decision boundary is a hyperplane in $\mathbb{F}$, and translates to nonlinear boundaries in the original space.

The SV approach can also be applied to regression and function approximation problems by replacing $\sum_{i=1}^{l} \xi_i$ term in (2) with a new loss term $\sum_{i=1}^{l} L^{\epsilon}(\vec{x}_i, y_i, f)$ and adjusting the constraints accordingly. $L^{\epsilon}(\vec{x}, y, f)$ is a linear ϵ-insensitive loss function [2] defined as

$$L^{\epsilon}(\vec{x}, y, f) = (|y - f(\vec{x})| - \epsilon)\ \mathrm{I}_{\{|y - f(\vec{x})| \geq \epsilon\}}(\vec{x}, y, f),$$

i.e., only errors falling outside the interval $[-\epsilon, \epsilon]$ counts. It was shown [5] that the function minimizing (2) with the new loss term has a form

$$f(\vec{x}) = \sum_{i=1}^{l} \alpha_i K(\vec{x}, \vec{x}_i) + b \quad . \qquad (7)$$

To find the coefficients α_i one has to solve the following quadratic program

$$\max\ W(\vec{\alpha}) = \sum_{i=1}^{l} y_i \alpha_i - \epsilon \sum_{i=1}^{l} |\alpha_i| - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j) \qquad (8)$$

$$\text{subject to } C \geq \alpha_i \geq -C,\ i = 1, \cdots, l, \text{ and } \sum_{i=1}^{l} \alpha_i = 0.$$

A detailed discussion on generalization performance of SV regression and function approximation can be found in [22], [5]. Several techniques of solving quadratic programming problems arising in SV algorithms are described in [13], [12], [17].

III. Additive Fuzzy Systems and Kernel Machines

A. Additive Fuzzy Systems

This paper considers additive fuzzy systems (AFS) with constant THEN-parts. Given $m+1$ fuzzy rules of the form

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

$$\text{Rule } j: \quad \text{IF } \mathbf{A}_j^1 \text{ AND } \mathbf{A}_j^2 \ \cdots \text{ AND } \mathbf{A}_j^n \text{ THEN } b_j \quad (10)$$

where $b_0,\ b_j \in \mathbb{R},\ j = 1, \cdots, m,\ k = 1, \cdots, n,\ \mathbf{A}_0^k$ and $\mathbf{A}_j^k$ are fuzzy sets with membership functions $a_0^k(x_k) \equiv 1$ and $a_j^k : \mathbb{R} \to [0, 1]$, respectively, if we choose product as the fuzzy conjunction operator, addition for fuzzy rule aggregation, and first order moment (FOM) defuzzification [3], then the input-output mapping, $F : \mathbb{R}^n \to \mathbb{R}$, of the system becomes

$$F(\vec{x}) = b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k) \qquad (11)$$

where $\vec{x} = [x_1, \cdots, x_n]^T \in \mathbb{R}^n$ is the input. In general, an AFS can have multiple outputs. Here we focus on single-output models. The results derived herein still apply to the multiple-output models by combining several single-output models provided that no coupling exists among outputs.

The membership functions above could be any function from $\mathbb{R} \to [0, 1]$. However, too much flexibility on the model could make effective learning unfeasible. So we narrow our interests to the class of membership functions that are generated from location transformation of reference functions [7].

Definition III.1: (Reference Function, [7]) *A function $\mu : \mathbb{R} \to [0, 1]$ is a reference function if and only if: 1) $\mu(x) = \mu(-x)$; 2) $\mu(0) = 1$; and, 3) μ is non-increasing on $[0, \infty)$.*

Now we can state the relationship between AFS and kernel machines as the following theorem.

Theorem III.2: (AFS and Kernel Machine) *Consider an AFS with $m + 1$ fuzzy rules of the form (9) and (10). Assume that for the kth input, $k \in \{1, \ldots, n\}$, the membership functions $a_j^k : \mathbb{R} \to [0, 1],\ j = 1, \cdots, m$ are generated*

[2] Different loss functions can also be used [22].

[3] Here we use the FOM instead of center of area (COA) defuzzification because the FOM provides a way to handle classification and function approximation in a uniform framework.

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 $z_j^k \in \mathbb{R}$. Then the input-output mapping of the AFS is equivalent to that of a kernel machine

$$F(\vec{x}) = b_0 + \sum_{j=1}^{m} b_j K(\vec{x}, \vec{z}_j) \tag{12}$$

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 for a_j^k, $i = 1, \cdots, n$, $K : \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ is a translation invariant kernel [4] *function defined as*

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

Proof: Using the definition of location transformation, (11) can be equivalently written as (12) with (13) following accordingly. □

B. Positive Definite Fuzzy Systems

Theorem III.2 presents a novel kernel perspective on AFSes. One particular kind of kernel, Mercer kernel, has received considerable attention in the machine learning literature [5], [9], [27] because it is an efficient way of extending linear learning machines to nonlinear ones. Is the kernel defined by (13) a Mercer kernel? A kernel satisfying the Mercer conditions [5] 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 [5].

Lemma III.3: (Positivity Condition for Mercer Kernels [5]) *A kernel $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a Mercer kernel 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 $\{\vec{x}_1, \cdots, \vec{x}_n\} \subset \mathbb{X}$ ($\mathbb{X}$ is a compact subset of $\mathbb{R}^n$) and all $n = 1, 2, \cdots\cdots$.*

For most nontrivial kernels, directly checking the positivity condition in Lemma III.3 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 on the spectral property of the kernel [23].

Lemma III.4: (Positivity Condition for Translation Invariant Kernels [23]) *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 III.7).

Definition III.5: (Positive Definite Function [10]) *A function $f : \mathbb{R} \to \mathbb{R}$ 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 III.6: *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

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

From Lemma III.4, K is a Mercer kernel if and only if the Fourier transform of f is nonnegative. Thus from Lemma III.3 and Definition III.5, we conclude that f is a positive definite function if and only if its Fourier transform is nonnegative. □

Theorem III.7: (Positive Definite Fuzzy Systems, PDFS) *The translation invariant kernel (13) is a Mercer kernel if the reference functions, $a^k : \mathbb{R} \to [0, 1]$, $k = 1, \cdots, n$, are positive definite functions. The corresponding AFS given in Theorem III.2 is named a PDFS.*

Proof: From Lemma III.4, 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

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

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

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 includes: 1) $\mu(x) = \max(1 - d|x|, 0)$ (symmetric triangle); 2) $\mu(x) = e^{-dx^2}$ (Gaussian); 3) $\mu(x) = \frac{1}{1+dx^2}$ (Cauchy); 4) $\mu(x) = e^{-d|x|}$ (Laplace); 5) $\mu(x) = \frac{2}{e^{dx} + e^{-dx}}$ (hyperbolic secant), and, 6) $\mu(x) =$

[4] 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.

$\frac{\sin^2(dx)}{d^2x^2}$ (squared sinc) where $d > 0$ (note that the Gaussian reference function corresponds to the commonly used Gaussian kernel). 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 reference functions from those listed above. It is worthwhile noting that the asymmetric triangle and the trapezoid reference functions are not positive definite.

Remark III.8: A Mercer kernel implicitly defines a nonlinear mapping, $\Phi : \mathbb{X} \to \mathbb{F}$, such that the kernel computes the inner product in $\mathbb{F}$, i.e., $K(\vec{x}, \vec{z}) = \langle \Phi(\vec{x}), \Phi(\vec{z}) \rangle_{\mathbb{F}}$ where $\mathbb{X}$ is the input space, $\langle \cdot, \cdot \rangle_{\mathbb{F}}$ is an inner product in the new feature space $\mathbb{F}$ (its dimension can be infinite). This implies that the input-output mapping (12) of a PDFS can be alternatively viewed as a linear function in the kernel induced feature space.

IV. Support Vector Learning for PDFS

A PDFS with n inputs 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 where m is unknown. Which reference functions to choose is an interesting research topic by itself [15]. Here we assume that the reference functions $a^i : \mathbb{R} \to [0, 1]$, $i = 1, \cdots, n$ are predetermined. Thus the problem is how to extract a set of fuzzy rules ($\{\vec{z}_1, \cdots, \vec{z}_m\}$ and $\{b_0, \cdots, b_m\}$) from training samples so that the PDFS has good generalization ability.

In the previous section, we demonstrate the equivalence (in terms of input-output mapping) between PDFSes and kernel machines. So any learning algorithm for kernel machines can potentially be applied to construct PDFSes. As a universal learning machine for pattern recognition problems, the SV learning method is known to have good generalization ability because it tries to decrease an upper bound on the expected risk by reducing the empirical risk and, at the same time, controlling the VC dimension of the model set [5], [27]. Here we propose using SV learning to build fuzzy classifiers and function approximators.

As shown in Section III, a PDFS is essentially a mapping from the input space to some real numbers. For classification purpose, it is desirable to have class labels as the output. A simple way to extend a PDFS to a binary classifier is to cascade a thresholding stage to the output of the PDFS. The resulting decision function, $f : \mathbb{X} \to \{+1, -1\}$, then becomes

$$f(\vec{x}) = \text{sgn}\left(F(\vec{x})\right) \quad . \tag{14}$$

where $F(\vec{x})$ is the output of the PDFS given by (12). Substituting (12) into (14) gives us an equation very similar to the decision function of an SVM defined in (6). This suggests a connection between support vectors and fuzzy rules. In fact, a PDFS classifier can be constructed by the following algorithm.

Algorithm IV.1: Learning PDFS Classifier

Inputs: *Positive definite reference functions $a^k(x_k)$, $k = 1, \cdots, n$, associated with n input variables, and a set of training samples $\{(\vec{x}_1, y_1), \cdots, (\vec{x}_l, y_l)\} \subset \mathbb{X} \times \{+1, -1\}$ where y_i is the class label associated with $\vec{x}_i$.*

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*
$$y = \text{sign}\left(\sum_{i=1}^{l} y_i \alpha_i K(\vec{x}_j, \vec{x}_i) + b\right)$$
where $\alpha_i \geq 0$, $i = 1, \cdots, l$, are obtained by solving the quadratic program (5).
3. *Extract fuzzy rules from the above SVM decision rule:*

```
b_0 ← b
j ← 1
FOR i = 1 TO l
    IF α_i > 0
        z_j ← x_i
        b_j ← y_i α_i
        j ← j + 1
    END IF
END FOR
m ← j − 1
```

The above algorithm can be modified to construct function approximators. Instead of (5), the quadratic program (8) is solved. Using the SV approach to build PDFSes has several advantages:

- The VC theory guarantees that the resulting classifiers and function approximators can have good generalization.
- The number of fuzzy rules is irrelevant to the dimension of the input space. It is always less than or equal to the number of training samples. In this sense, the "curse of dimensionality" is avoided. In addition, due to the sparsity of support vectors, the number of fuzzy rule is usually much less than the number of training samples.
- 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. Techniques designed specifically for SV algorithms make large-scale training (for example $200,000$ samples with $40,000$ input variables) practical [13], [12], [17].

V. Experimental Results

This section provides two examples to demonstrate the performance of PDFSes.

TABLE I

USPS data set. Mean classification rate $r \pm$ standard deviation and mean number of fuzzy rules m (for one PDFS) using different reference functions.

Reference Function	$r\pm$STD	m
Gaussian	$95.2\% \pm 0.3\%$	573
Cauchy	$95.2\% \pm 0.3\%$	567
Laplace	$94.7\% \pm 0.4\%$	685
Symmetric Triangle	$95.0\% \pm 0.3\%$	652
Hyperbolic Secant	$95.0\% \pm 0.3\%$	468
Squared Sinc	$95.2\% \pm 0.2\%$	391

TABLE II

Function approximation. Average MSE $\pm$ standard deviation and the mean number of fuzzy rules m using different reference functions.

Reference Function	MSE$\pm$STD	m
Gaussian	0.0432 ± 0.0057	123
Cauchy	0.0442 ± 0.0057	122
Laplace	0.0413 ± 0.0044	130
Symmetric Triangle	0.0413 ± 0.0043	130
Hyperbolic Secant	0.0426 ± 0.0055	117
Squared Sinc	0.0422 ± 0.0056	113

A. Classification

The USPS data [5] 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 PDFS classifiers for different choices of reference functions namely the Gaussian, Cauchy, Laplace, hyperbolic secant, and squared sinc. For different input variables, the reference functions are chosen to be identical. Ten PDFS classifiers are designed, each of which separates one digit from the rest nine digits. The final predicted class label is decided by the PDFS with the maximum 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 SV learning (for each PDFS) 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 PDFS classifiers 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 PDFS) are listed in Table I. For comparison purpose, we also cite the results from [16]: linear SVM (classification rate 91.3%), k-nearest neighbor (classification rate 94.3%), 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 [16], the PDFSes achieve a better classification rate. SVMs can be improved by using prior knowledge. For instance the virtual SVM [16] performs better than current PDFSes. However, same approach can be applied to build PDFSes, i.e., PDFCs can also benefit from the same prior knowledge.

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

B. Function Approximation

Consider the function defined by the following equation

$$g(x_1, x_2) = \frac{(5 - x_2)^2}{3(5 - x_1)^2 + (5 - x_2)^2}$$

with x_1, $x_2 \in [0, 10]$. PDFSes are constructed to approximate the above function using different reference functions. The training set contains 200 input-output pairs where the inputs are randomly sampled from $[0, 10] \times [0, 10]$ according to the uniform distribution. A different set of 200 samples is chosen for testing. For the ϵ-insensitive loss function, we let $\epsilon = 0.1$. Based on the training set, we use 5-fold cross-validation to determine the d parameter of the reference functions and the C parameter in SV learning where C takes values from $\{10, 100\}$, and d takes values from $\{\frac{1}{2^n} : n = 1, \cdots, 9\}$. For each pair of d and C, the average cross-validation mean squared error (MSE) is computed. The optimal d and C are the values that gives the minimal mean cross-validation MSE. The whole process is repeated 10 times. The average MSE (and the standard deviation of MSEs) on the test set and the mean number of fuzzy rules are given in Table II. As we can see, different reference functions produce similar approximation errors. The Laplace and symmetric triangle reference functions give slightly better (not statistically significant) performance for this experiment.

VI. Conclusions and Future Work

This paper shows that two seemingly unrelated research areas, fuzzy systems and kernel machines, are closely related. A class of additive fuzzy systems (PDFS) are in essence kernel machines with kernels defined by positive definite reference functions (as a by-product we get a class

of Mercer kernels). The SVM learning approach can be utilized to build PDFSes. This not only avoids the "curse of dimensionality" that occurs in the ordinary fuzzy modeling approach, but also leads to good generalization. The results also imply 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 [21] and the computational complexity issues in kernel PCA [19] and SVMs [3].

The requirement that all membership functions associated with an input variable are generated from the same reference function maybe somewhat restrictive. It looks like that this constraint can be relaxed. 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 system can still be related to the generalized support vector machines [14].

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.

References

[1] 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.

[2] C. J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 1-47, 1998.

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

[4] 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.

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

[6] 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.

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

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

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] A.J. Smola and B. Schölkopf, "A Tutorial on Support Vector Regression," *NeuroCOLT2 Technical Report NC2-TR-1998-030*, Royal Holloway College, London, UK, 1998.

[23] 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.

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

[25] 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.

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

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

[28] 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.

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

[30] 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.

[31] 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.

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