Expert Systems with Applications 38 (2011) 4530–4539

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# Shell fitting space for classification

Mostafa Ghazizadeh Ahsaee *, Hadi Sadoghi Yazdi, Mahmoud Naghibzadeh

*Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran*

ARTICLE INFO

*Keywords:*
Shell fitting space
Distance based transformation space
Fitting
Classification

ABSTRACT

In this paper, a shell fitting space (SFS) is presented to map non-linearly separable data to linearly separable ones. A linear or quadratic transformation maps data into a new space for better classification, if the transformation method is properly guessed. This new SFS space can be of high or low dimensionality, and the number of dimensions is generally low and it is equal to the number of classes. The SFS method is based on fitting a hyper-plane or shell to the learning data or enclosing them into a hyper-surface. In the proposed method, the hyper-planes, curves, or cortex become the axis of the new space. In the new space a linear support vector machine (SVM) multi-class classifier is applied to classify the learn data.



## 1. Introduction

Classification is an important research area with a wide range of applications. Nonlinear discriminant functions (NDF) are useful in training a system to recognize specific patterns and now many applications are based on this method. Neural network and support vector machine are preeminent mathematical tools of NDF. Support vector machines (Vapnik, 1995) are very popular and powerful in learning systems because of the utilization of kernel machine in linearization, providing good generalization properties, their ability to classify input patterns with minimized structural misclassification risk and finding acceptable separating hyper-plane between two classes in the feature space.

The result of applying kernels allows the algorithm to fit the maximum-margin hyper-plane in the transformed feature space. The transformation may be non-linear and the transformed space may be high dimensional; thus though the classifier is a hyper-plane in the high-dimensional feature space it may be non-linear in the original input space. If the used kernel is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results.

Of course kernel methods (KMs) (Abe, 2005; Huang, Kecman, & Kopriva, 2006; Scholkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004) map input space into a high dimensional feature (HDF) space that may be helpful in linearization. As we know some kernels were proposed for this purpose, namely polynomials, Gaussians, and splines (Friedman, 1991), but these kernels do not guaranty linearization in HDF. This problem motivates us for presentation of new space in which patterns can be classified by linear classifier, we name it shell fitting space (SFS) because we use the concept of shell fitting for the creation of the new space.

Support vector machines (SVM) and its variants (Abe, 2005; Lin & Wang, 2002; Sadoghi Yazdi, Effati, & Saberi, 2007; Wang, 2005; Wu, Jie-Chi, & Lee, 2007) is a particular instance of KMs. But it has some weaknesses as follows.

Slow training (compared to neural network) due to computationally intensive solution to QP problem especially for large amounts of training data ⇒ needs special algorithms.

- The kernel to be used is not deterministic and it changes for each data set and finally a large feature space is produced (with many dimensions).
- Slow classification for the trained SVM.
- Generates complex solutions (normally > 60% of training points are used as support vectors), especially for large amounts of training data.
- Difficult to incorporate prior knowledge.

Our proposed approach is not to expand the original space into a new space with many dimensions in comparison with kernel methods. In SFS the mapping is done to an $m$-dimensional space; where $m$ is the number of classes.

The rest of this paper is as follows. Section 2 is devoted to the development of our method, Section 3 explains how the new method works on example datasets, Section 4 is devoted to the application of the method on real datasets. In Section 5 we test our method with a simple linear classifier on SFS data and compare its accuracy results with multi-class SVM results on input space data. Conclusions are made in Section 6.

* Corresponding author.
*E-mail addresses:* GhazizadehAhsaee.most@stu-mail.um.ac.ir (M. Ghazizadeh Ahsaee), h-sadoghi@um.ac.ir (H. Sadoghi Yazdi), naghibzadeh@um.ac.ir (M. Naghibzadeh).

0957-4174/$ - see front matter © 2010 Published by Elsevier Ltd.
doi:10.1016/j.eswa.2010.09.127

## 2. Shell Fitting Space formulation

Definitions: $\{x_{ij} \in [x_{11}, x_{21}, \ldots, x_{K_1 1}, x_{12}, \ldots, x_{K_2 2}, \ldots, x_{1j}, x_{2j}, \ldots, x_{k_j j}, \ldots], j = 1, \ldots, m\}$ is the $i$th sample with $n$ dimensions of class $j$.

$C_j$ is the fitted curve, hyperplane, or surrounded cortex or (shell) to the set $\{(X_{ij}, y_j), i = 1, \ldots, k_j\}$ of data, where $y_j$ is the $j$th label of the training data and $k_j$ shows the number of samples with label $j$. In general our mapping is done from a space with $m$ patterns (classes) of data in $n$-dimensional space to $m$-dimensional space by the following notation:

$$\varphi : X = \{x_1, x_2, \ldots, x_n\} \in R^n \rightarrow \varphi(X) \in R^m$$

where $\phi$ is a function that depends on distance of data to each pattern's fitted curve, hyperplane or surrounded cortex or (shell).

This transformation can be seen in Fig. 1, where $X$ is a feature in the input space and $D = \{d_1, d_2, \ldots, d_m\}$ is the feature space element and $c_i$ is the class $i$.

### 2.1. Dealing with hyperplane

When a hyperplane is fitted to a set of data the distance of the point $X_i$ from that plane is calculated with the following formula:

Suppose that the fitted hyperplane is $y$ in (1)

$$y = w_1 x_{1i} + w_2 x_{2i} + \cdots + w_n x_{ni} \tag{1}$$

So the distance from $y$ is obtained as (2)

$$d(X_i, C) = \frac{w_1 x_{1i} + w_2 x_{2i} + \cdots + w_n x_{ni}}{\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}} \tag{2}$$

For instance in $y = w_1 x_1 + w_2 x_2$ we have the shape shown in Fig. 2a.

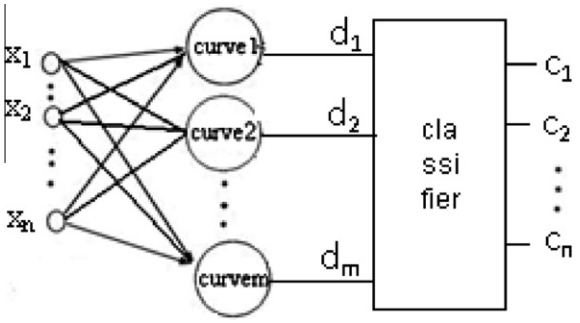



**Fig. 1.** RBF-like transformation space.

### 2.2. Dealing with a hypersphere

When a hypersphere is fitted to a set of data it can be formulated as:

$$(x_1 - a_1)^2 + (x_2 - a_2)^2 + \cdots + (x_n - a_n)^2 = r^2 \tag{3}$$

the distance between $X_i$ and $C$ is calculated by (4)

$$d(X_i, C) = \left( \left| r - \sqrt{x_{1i}^2 + x_{2i}^2 + \cdots + x_{ni}^2} \right| \right) \tag{4}$$

where $r$ is as follows:

$$r = \sqrt{(x_1 - a_1)^2 + (x_2 - a_2)^2 + \cdots + (x_n - a_n)^2} \tag{5}$$

For example, consider two points $(x_1, y_1)$ and $(x_2, y_2)$ in Fig. 2. The distances $d_1$ and $d_2$ are calculated as shown in Fig. 2b.

### 2.3. Dealing with a polynomial in two-dimensional space

When a polynomial is fitted to a set of data, points obtained by minimizing Eq. (6) are examined to see which one corresponds to the actual global minimum. To do so, Eq. (6) is evaluated for each of these points and the one that gives a lower value is selected

$$r = \sqrt{(x - x_1)^2 + (y - y_1)^2} \tag{6}$$

where $y$ is as follows:

$$y = w_0 + w_1 x^1 + w_2 x^2 + \cdots + w_n x^n \tag{7}$$

For example suppose:

$$y = w_0 + w_1 x^1 + w_2 x^2 + \cdots + w_6 x^6$$

then we have:

$$r(x) = \sqrt{(x - x_1)^2 + (w_0 + w_1 x + w_2 x^2 + \cdots + w_6 x^6 - y_1)^2} \tag{8}$$

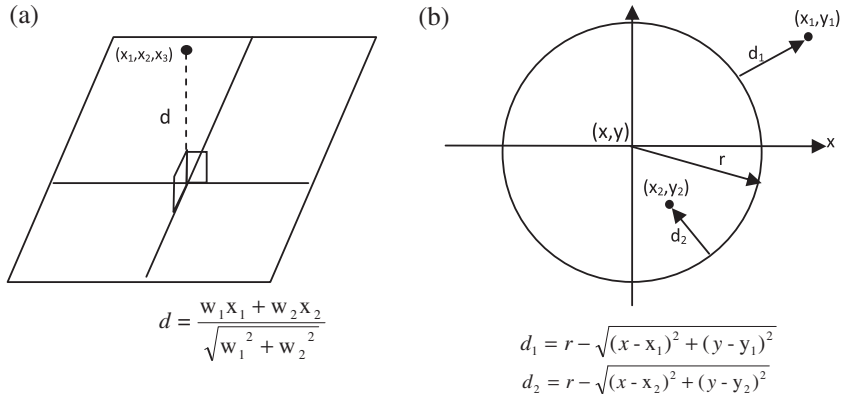
The points that minimize $r$ are calculated by setting the derivatives of $r$ equal to zero.

Generally, by setting the derivatives of r equal to zero we have simple formulas as (9)

$$\partial r / \partial x = x - x_0 + (w_n x^n + \ldots + w_1 x + w_0 - y_1) * (n w_n x^{n-1} + (n-1) w_{n-1} x^{n-2} + \cdots + w_1) = 0 \tag{9}$$

Then for each answer $x_i$ we find $r(x_i)$ from (8) and at the end the minimum distance is:

$$Min \quad \{r(x_i) | x_i \in \{answers\}\}$$




$$d = \frac{w_1 x_1 + w_2 x_2}{\sqrt{w_1^2 + w_2^2}}$$

$$d_1 = r - \sqrt{(x - x_1)^2 + (y - y_1)^2}$$

$$d_2 = r - \sqrt{(x - x_2)^2 + (y - y_2)^2}$$

**Fig. 2.** (a) Distance from a hyperplane and (b) distance from a hypersphere.

### 2.4. Dealing with a line in n-dimensional space

Generally an equation for a line in $n$-dimensional space can be written as follows:

$$L(x,y,\ldots,z): \begin{cases} w = a_1 x + b_1 \\ y = a_2 x + b_2 \\ \cdots \\ z = a_n x + b_n \end{cases} \tag{10}$$

So the distance between a point $(x_0,y_0,\ldots,z_0)$ and $(x,y,\ldots,z)$ on $L$ can be calculated with the formula below:

$$d(w,x,\ldots,z) = \sqrt{(w-w_0)^2 + (x-x_0)^2 + \cdots + (z-z_0)^2} \tag{11}$$

Considering Eq. (10), the distance can be expressed as:

$$d(x) = \sqrt{(a_1 x + b_1 - w_0)^2 + (x-x_0)^2 + \cdots + (a_n x + b_n - z_0)^2} \tag{12}$$

Once again, we like to minimize (12). Therefore, its derivatives are set to zero and the $x$ values for which $d(x)$ is minimized are calculated

$$\partial d/\partial x = (a_1 x + b_1 - w_0)a_1 + (x - x_0) + \cdots + (a_n x + b_n - z_0)a_n = 0 \tag{13}$$

Then for each solution $x_i$, $d(x_i)$ is evaluated to find the actual global minimum distance, that is:

$$Min \quad \{d(x_i) | x_i \in \{answers\}\}$$

### 2.5. Dealing with a curve in n-dimensional space

If the number of features is more than two, we can use neural network-based methods like adaptive neural-fuzzy inference system (ANFIS) for this purpose. Detailed explanation can be followed from Wu et al. (2007) and a brief description is given in the Appendix.

Steps of transformation using ANFIS are:

(a) Learn ANFIS the classes of data.
(b) Evaluate new data by the system.
(c) Calculate the distance with a subtraction operation; evaluation result is subtracted from the main value of data as distance. You can see the outputs in Section 3.

### 2.6. Dealing with a voluminous classes in n-dimensional space

#### 2.6.1. SVDD one-class classification

Three general approaches are proposed to resolve the one-class classification problem (Tax, 2001). The most straightforward method to obtain a one-class classifier is to estimate the density of the training data and to set a threshold on this density. Several distributions can be assumed, such as a Gaussian or a Poisson distribution. The most popular density models are the Gaussian model, the mixture of Gaussians, and the Parzen density (Bishop, 1995; Parzen, 1962). In the second method a closed boundary around the target set is optimized. $K$-centers, nearest neighbor method and support vector data description (SVDD) are examples of the boundary methods (Tax & Duin, 2004; Ypma et al., 1998). Reconstruction methods are another one-class classification method which have not been primarily constructed for one-class classification, but rather to model the data. By using prior knowledge about the data and making assumptions about the generating process, a model is chosen and fitted to the data. Some types of reconstruction methods are: the $k$-means clustering, learning vector quantization, self-organizing maps, PCA, a mixture of PCAs, diabolo networks, and auto-encoder networks.

To obtain data description for a group of $N$ target objects in input space, we try to find a sphere with minimum volume which encloses all or most of these target objects. The problem of finding the minimum hyper-sphere represented by a center "$a$" and radius "$R$" can be formulated into:

$$F(R,a) = R^2 + C\sum_i \xi_i \tag{14}$$

$$s.t. \quad \|x_i - a\|^2 \leqslant R^2 + \xi_i, \quad \forall i, \xi_i \geqslant 0 \tag{15}$$

The parameter $C$ gives the tradeoff between the volume of the description and the errors. The free parameters, $a$, $R$ and $\xi_i$, have to be optimized, taking the constraints (15) into account. Constraints (15) can be incorporated into formula (14) by introducing Lagrange multipliers and constructing the Lagrangian:

$$L(R,a,\alpha_i,\gamma_i,\xi_i) = R^2 + C\sum_i \xi_i - \sum_i \alpha_i\{R^2 + \xi_i - (\|x_i\|^2 - 2a.x_i + \|a\|^2)\} - \sum_i \gamma_i \xi_i \tag{16}$$

with the Lagrange multipliers $\alpha_i \geqslant 0$ and $\gamma_i \geqslant 0$. Setting partial derivatives to 0 gives these constraints:

$$\frac{\partial L}{\partial R} = 0: \quad \sum_i \alpha_i = 1 \tag{17}$$

$$\frac{\partial L}{\partial a} = 0: \quad a = \frac{\sum_i \alpha_i x_i}{\sum_i \alpha_i} = \sum_i \alpha_i x_i \tag{18}$$

$$\frac{\partial L}{\partial \xi_i} = 0: \quad C - \alpha_i - \gamma_i = 0 \tag{19}$$

From the last equation $\alpha_i = C - \gamma_i$ and because $\alpha_i \geqslant 0$, $\gamma_i \geqslant 0$, Lagrange multipliers $\gamma_i$ can be removed when we demand that

$$0 < \alpha_i < C \tag{20}$$

Resubstituting (17)–(19) into (16) results in:

$$L = \sum_i \alpha_i (x_i \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \tag{21}$$

By definition, $R^2$ is the distance from the center of the sphere "$a$" to (any of the support vectors on) the boundary. Support vectors which fall outside the description ($\alpha_i = C$) are excluded. Therefore:

$$R^2 = (x_k \cdot x_k) - 2\sum_i \alpha_i (x_i \cdot x_k) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \tag{22}$$

Because we are able to give an expression for the center of the hypersphere "$a$", we can test if a new object "$z$" is accepted by the description. For that, the distance from the object "$z$" to the center of the hypersphere "$a$" has to be calculated. A test object $z$ is accepted when this distance is smaller than or equal to the radius:

$$\|z - a\|^2 = (z \cdot z) - 2\sum_i \alpha_i (z \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \leqslant R^2 \tag{23}$$

#### 2.6.2. Using SVDD for our distance based method

Here we used one SVDD for each class of a dataset (for example 2 SVDD for 2 classes). Then to check if a data belongs to a class, its distance from cortex of each SVDD classes is calculated and this distance is used as a main criterion to classify the data. In other words, a data belongs to a class, if its distance to a voluminous class' cortex is less than the other classes' in kernel space. The distance is calculated by (24). As stated above, like hypersphere-but in kernel space, the transformation is done

$$d = (z \cdot z) - 2\sum_i \alpha_i (z \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) - R^2 \quad (24)$$

## 3. Experimental results

First the proposed circle fitting space transformation method is now demonstrated using simple data as an illustrating example, and at the end, our method will be tested using some well known datasets. In the following discussion, we assume that the label for each train data is known, in advance. Here our method has been implemented and tested on MATLAB (MathWork Inc.).

### 3.1. Operation on synthetic data

We describe our approach by some examples. At first we introduce our synthetic data as shown in Fig. 3a–i.

In Fig. 3a there are two classes of linear shape which are linearly separable. But in other figures (b–i) are not linearly separable and these figures are divided into three categories: (1) sphere-based figures (b–f), (2) sinuous function (Fig. 3g): more complex classes which are not easily separable. (3) Fig. 3h and i: are used in learning ANFIS.

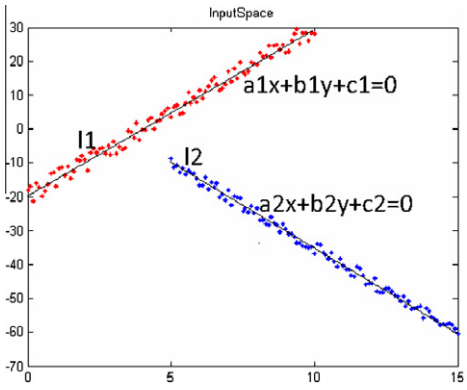



**Fig. 4.** Fitting a line to each of data classes.

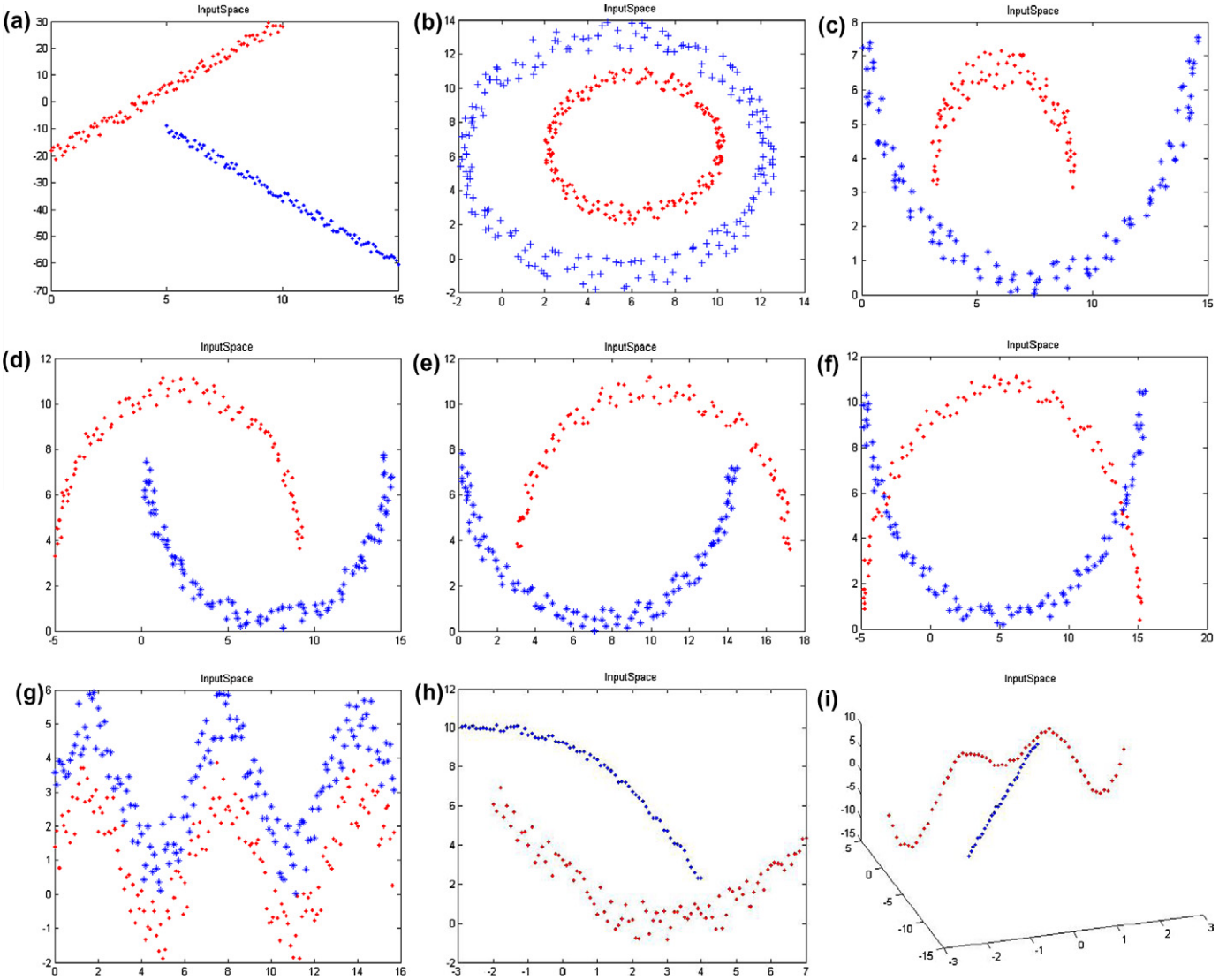



**Fig. 3.** (a) Two linear classes; (b) circle classes; (c) half-circle classes; (d) half-circle classes; (e) half-circle classes; (f) half-circle classes interfere each other; (g) sin-function; (h) two classes for ANFIS and (i) two classes for ANFIS in 3-D space.

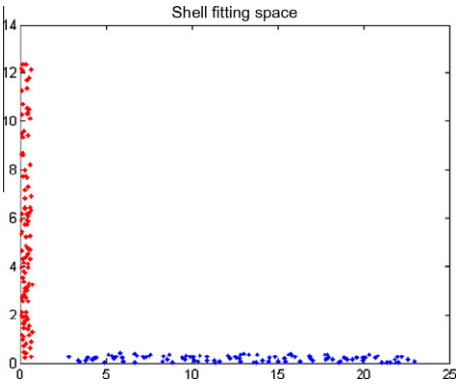



**Fig. 5.** Transformation space of Fig. 4.

Consider two data classes in Fig. 3a that are linear classes.

Firstly a line is fitted to each class of data with least square error fitting method. The result is seen in Fig. 4.

And then to map them to a new space we use a formula like the following to compute the distance of each data $(x_0, y_0)$ to the lines. Here $d(X_0, l_1)$ stands for the distance of point $X_0 = (x_0, y_0)$ from line $d_1$

$$d(X_0, l_1) = \frac{|a_1 x_0 + b_1 y_0 + c_1|}{\sqrt{a_1^2 + b_1^2}}$$

$$d(X_0, l_2) = \frac{|a_2 x_0 + b_2 y_0 + c_2|}{\sqrt{a_2^2 + b_2^2}}$$

We use these distances to map data to a new 2-dimensional space like Fig. 5 which is linearly separable.

As the second example, consider Fig. 3b. These classes are examples of hypersphere but in a 2D-space. Curve fitting is done first and the distance between each circular curve and dataset elements is calculated. The results are depicted in Fig. 6a and b. Horizontal axis shows the distance between each dataset element and the internal circle and the other axis is the distance between each dataset element and the external circle.

### 3.2. Transformation of circle classes

Our proposed method is applied to a data series in 2-dimensional space (two classes with two features) and the results are shown in Figs. 7–11.

We have tested our method on four kinds of half-circle (Fig. 3c–f). The two half-circles may even interfere each other (Fig. 3f). The results of curve fitting and calculating distance between each dataset element and each fitted data are depicted in Figs. 7–10.

### 3.3. Transformation of more complex classes

Data distribution may seem so complex, but this method tries to separate these two classes of data (Fig. 11).

### 3.4. Transformation using ANFIS

Other experimental results are those which use ANFIS to fit a curve (Figs. 12 and 13). Here we have used two classes in 2- and 3-dimensional space.

The two ANFIS were used to fit curves to the data set. As shown in Fig. 12, the result is linearly separable.

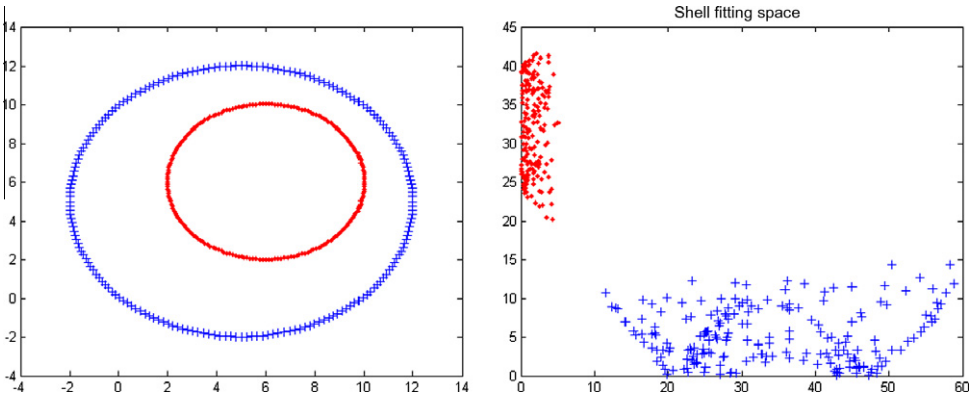



**Fig. 6.** (a) Fitted spheres and (b) Output of mapping.

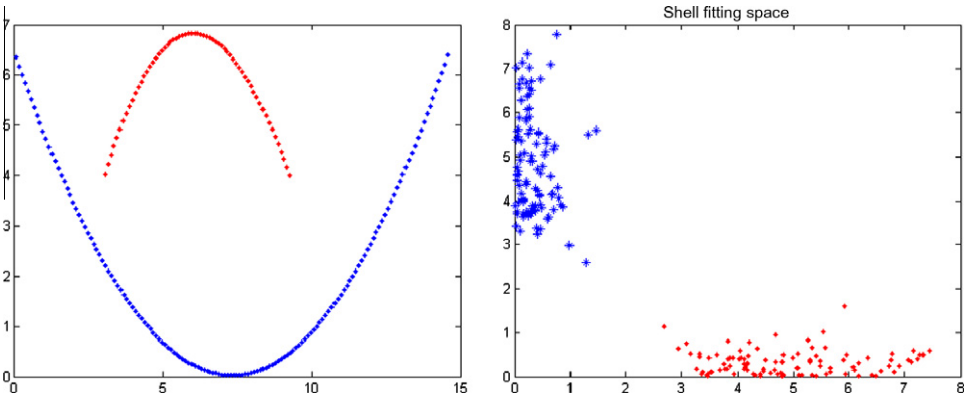



**Fig. 7.** Fitting a half-circle to each of data classes.

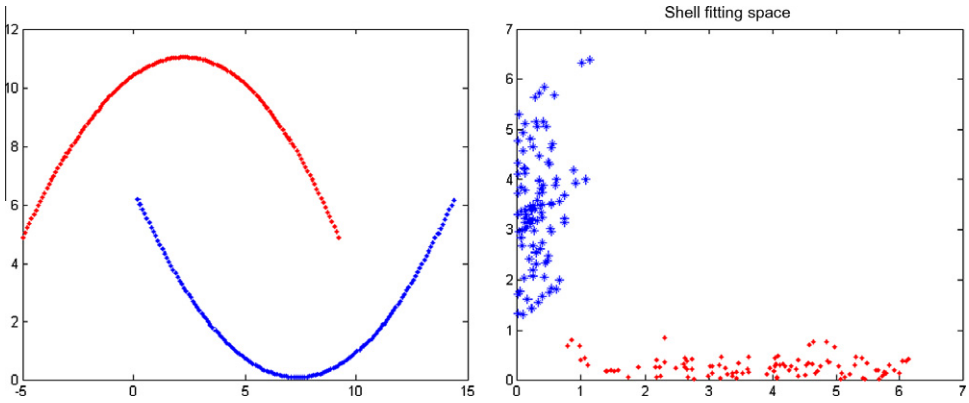



**Fig. 8.** Fitting a half-circle to each of data classes.

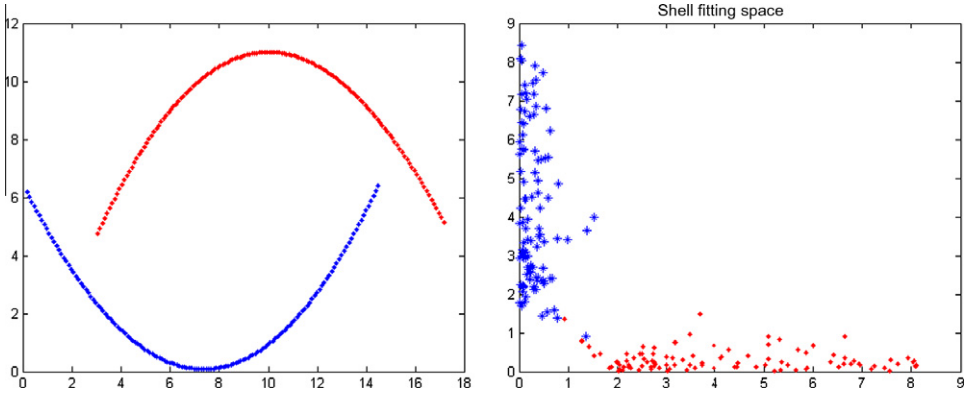



**Fig. 9.** Fitting a half-circle to each of data classes.

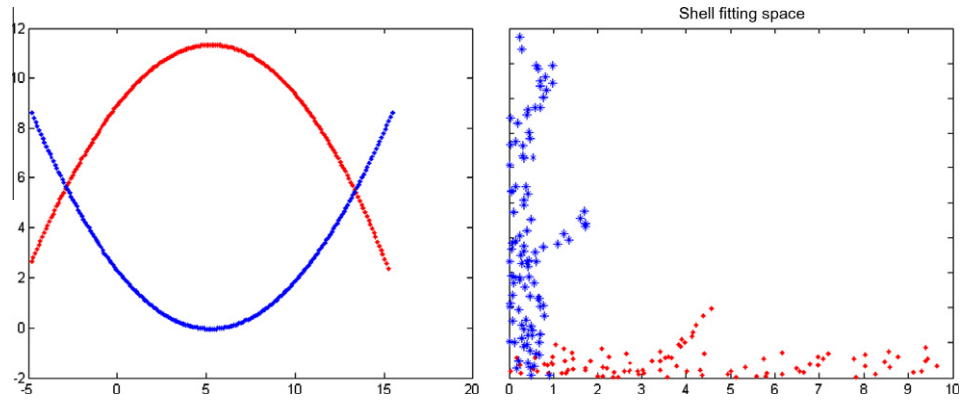



**Fig. 10.** Fitting a half-circle to each of data classes.

### 3.5. Transformation using SVDD

As stated before, SVDD is a one-class classification used in classification of voluminous data. At this point, a dataset is shown in Fig. 14a and the result of our proposed method is depicted in Fig. 14b.

## 4. Experiments on Datasets

In this section our proposed method is used to classify some datasets. These datasets are from UCI Machine Learning Repository. Datasets used here are breast-cancer-wisconsin dataset, Iris dataset, Ionosphere dataset, Transfusion dataset and heart dataset with 11, 5, 35, 15, 4, and 14 attributes with 2, 3, 2, 3, 2, and 2 classes of data in sequence.

### 4.1. Circle fitting on datasets

As we said in previous sections, circle fitting is used in classification of like-circle data. We show here that our method is good to classify breast-cancer-wisconsin and Iris datasets and to some extent to classify ionosphere dataset using circle fitting. Our method calculated the distance of each data in the dataset to circles fitted

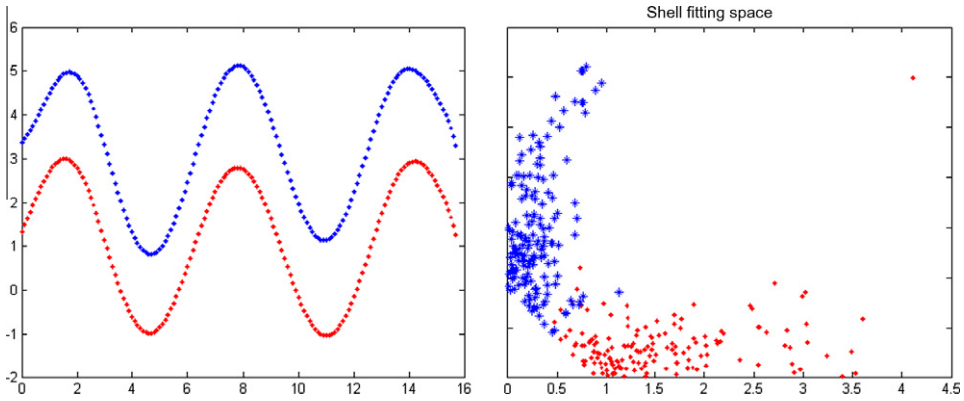



**Fig. 11.** Fitting a curve to each of data classes.

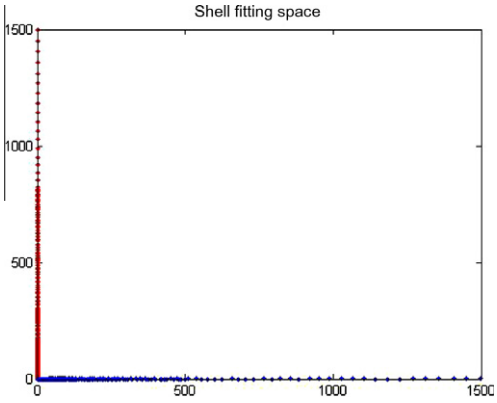



**Fig. 12.** Fitting a curve to a data set using ANFIS and transform them to a new linearly separable space.

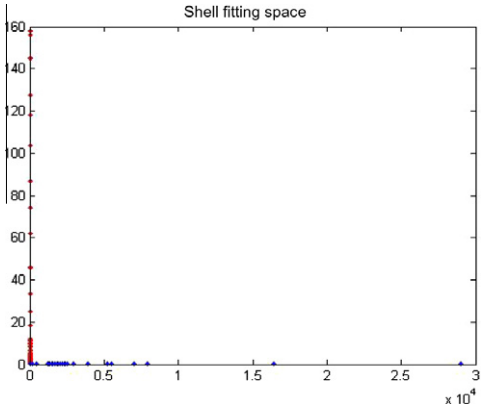



**Fig. 13.** Curves have been fitted to data sets of two classes and the result of transformation has been shown in right figure.

on data and the classes are identified. We can see the result in Figs. 15–17.

We can see from Figs. 15 and 17 that data may have interference in their distribution or may not be well fitted to a hypersphere.

### 4.2. Using SVDD on dataset

Here we used one SVDD for each class of iris dataset (3 SVDD for 3 classes). Then we calculated distances from cortex of each SVDD class. The result of classification is depicted in Fig. 18.

For other datasets (breast-cancer-wisconsin dataset, Ionosphere dataset, Wine dataset, Transfusion dataset and heart dataset), the same process is followed and the result shows, Figs. 19–22, that SVDD is good to be used as a tool for our distance based classification.

## 5. Accuracy of our method in comparison with multi-class SVM

To show the performance of our method, we first transformed the data of each dataset to the proposed shell fitting space and we then used a simple linear classification (Adaline) to classify the transformed data. In this stage, 50% of the data in each dataset was used for the training purpose. Kernel-based SVM classifier was used on the original data to compare the accuracy of our classification results. Amongst the available kernels the one with the best performance for each dataset was used in the training stage. Both systems were tested for each test data of each dataset. The comparison results of the mean values of 30 runs of both methods follow. It is worth mentioning that in each run of each dataset the data are randomly divided into training class and test class.

Adaline classifier uses a $W$, weight vector, to separate classes with a hyper-plane where $W$ is obtained from (25):

$$W = DX(XX')^{-1} \tag{25}$$

In (25) $D$ is the vector of the desired label for each training data of the class of training data and $X$ is the vector of training samples. We used rbf, linear, polynomial,...,kernels for SVM classifier and selected the best results for the comparison purpose.

Accuracy (AC) is calculated for a dataset using the following definition:

$$AC = (\text{number of true classifications}) \\ /(\text{number of true and false classifications})$$

Or,

$$AC = \frac{\text{number of true positives} + \text{true negatives}}{\text{number of true positives} + \text{true negatives} + \text{false positives} + \text{false negatives}} \tag{26}$$

The comparison results are shown in Table 1. As you can see, in our method only the hyper-sphere fitting to Wine dataset and heart dataset has led to a lower accuracy. This shows that, if the fitting shape is not selected well, the result might not be desirable.

## 6. Conclusions

In this paper we introduced a new transformation space which is easier than the other space transformations to classify input

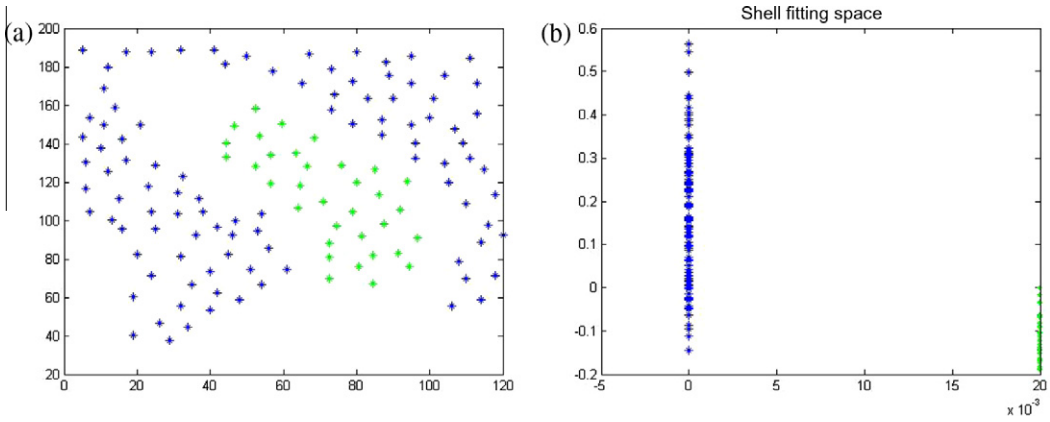



**Fig. 14.** (a) Input dataset and (b) result of classification using distance based method.

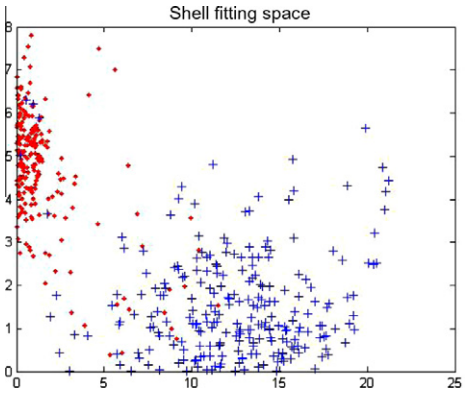



**Fig. 15.** Result of circle fitting on breast-cancer-wisconsin dataset.

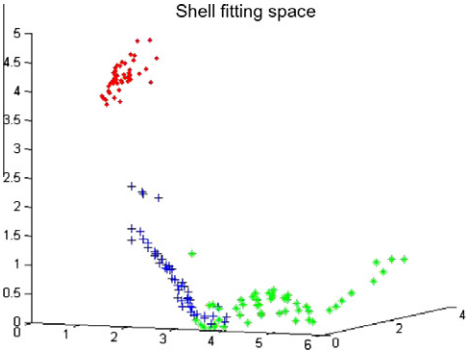



**Fig. 16.** Result of circle fitting on Iris dataset.

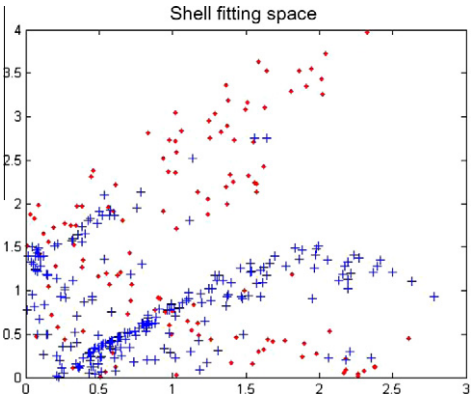



**Fig. 17.** Result of circle fitting on ionosphere dataset.

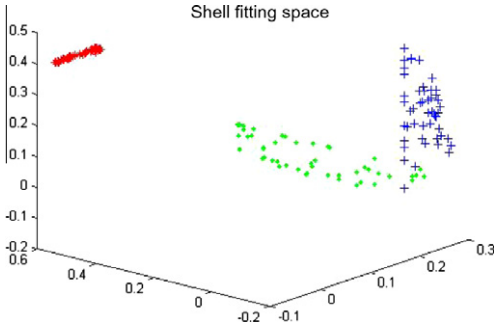



**Fig. 18.** Result of SVDD on Iris dataset.

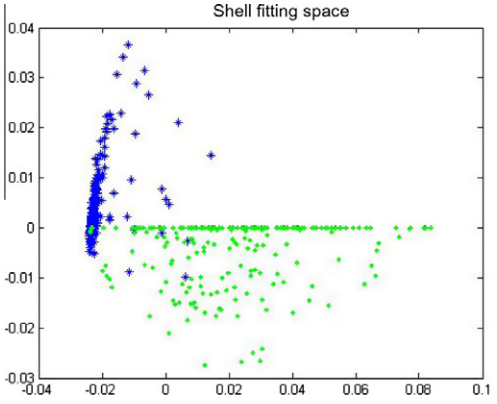



**Fig. 19.** Result of SVDDD on breast-cancer-wisconsin dataset.

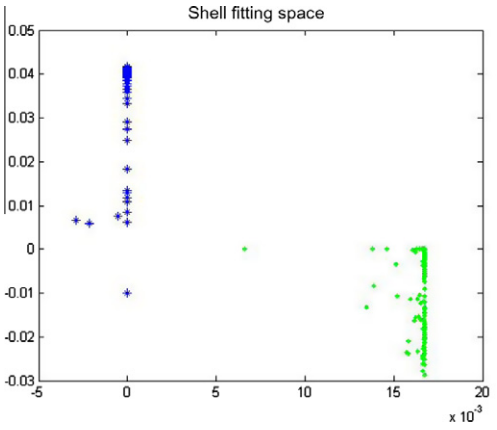



**Fig. 20.** Result of SVDDD on Ionosphere dataset.

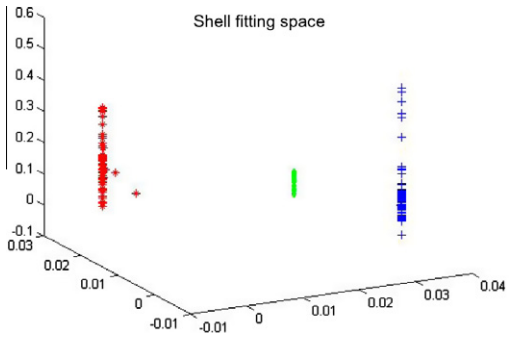



**Fig. 21.** Result of SVDDD on Wine dataset.

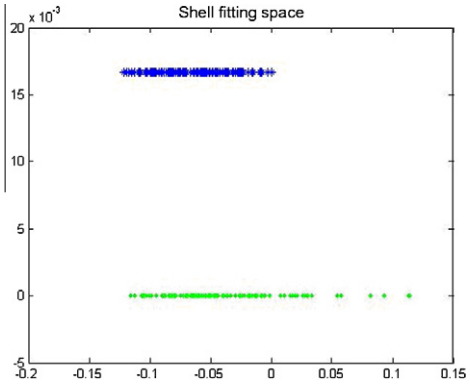



**Fig. 22.** Result of SVDDD on Ionosphere dataset.

**Table 1**
Comparison of our method accuracy with multi-class SVM (mean accuracy in 30 runs).

| Mean accuracy in 30 runs | Our method (circle fitting) (%) | Our method (using SVDD) (%) | Multi-class SVM (%) |
|---|---|---|---|
| Cancer | 96.76 | 97.25 | 96.38 |
| Heart | 53.86 | 100 | 77.38 |
| Wine | 30.80 | 100 | 76.55 |
| Iris | 93.44 | 96.46 | 95.20 |
| Ionosphere | 72.27 | 99.72 | 85.89 |

data. The main idea was to use distance of data to a line, curve, or hyperplane fitted to each class' data or shells enclosing the classes. The results show that this space transformation works well for collections of data.

## Appendix. Adaptive neuro-fuzzy inference system (ANFIS) architecture (Jang, 1993)

A typical architecture of ANFIS is shown in Fig. 23 for modeling of function, in which a circle indicates a fixed node, and a square indicates an adaptive node. For simplicity, we consider two inputs $x$, $y$ and one output $z$ in the fuzzy inference system (FIS). The ANFIS used in this paper implements a first-order Sugeno fuzzy model. Among the many fuzzy inference systems, the Sugeno fuzzy model is the most widely used due to its high interpretability and computational efficiency, and built-in optimal and adaptive techniques. For example for a first-order Sugeno fuzzy model, a common rule set with two fuzzy if-then rules can be expressed as (27) and (28)

$$\text{Rule 1}: \text{If } x \text{ is } A_1 \text{ and } y \text{ is } B_1, \text{ then } z_1 = p_1x + q_1y + r_1 \tag{27}$$

$$\text{Rule 2}: \text{If } x \text{ is } A_2 \text{ and } y \text{ is } B_2, \text{ then } z_2 = p_2x + q_2y + r_2 \tag{28}$$

where $A_i$, $B_i$ ($i = 1,2$) are fuzzy sets in the antecedent, and $p_i$, $q_i$, $r_i$ ($i = 1,2$) are the design parameters that are determined during the training process. As in Fig. 23, the ANFIS consists of five layers.

Layer 1, every node $i$ in this layer is an adaptive node with a node function like (29)

$$\begin{aligned} O_i^1 &= \mu_{A_i}(x), \quad i = 1,2 \\ O_i^1 &= \mu_{B_i}(y), \quad i = 3,4 \end{aligned} \tag{29}$$

where $x$, $y$ are the input of node $i$, $\mu_{A_i}(x)$ and $\mu_{B_i}(y)$ and can adopt any fuzzy membership function (MF). In this paper, Gaussian MFs which are defined by (30) are used

$$gaussian(x, c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \tag{30}$$

where $c$ is center of Gaussian membership function and $\sigma$ is standard deviation of this cluster.

Layer 2, every node in the second layer represents the ring strength of a rule by multiplying the incoming signals and forwarding the product as (31)

$$O_i^2 = w_i = \mu_{A_i}(x)\mu_{B_i}(y), \quad i = 1 \tag{31}$$

Layer 3, the $i$th node in this layer calculates the ratio of the $i$th rule's ring strength to the sum of all rules' rings strengths:

$$O_i^3 = \varpi_i = \frac{\omega_i}{\omega_1 + \omega_2}, \quad i = 1,2 \tag{32}$$

where $\varpi_i$ is referred to as the normalized ring strengths.

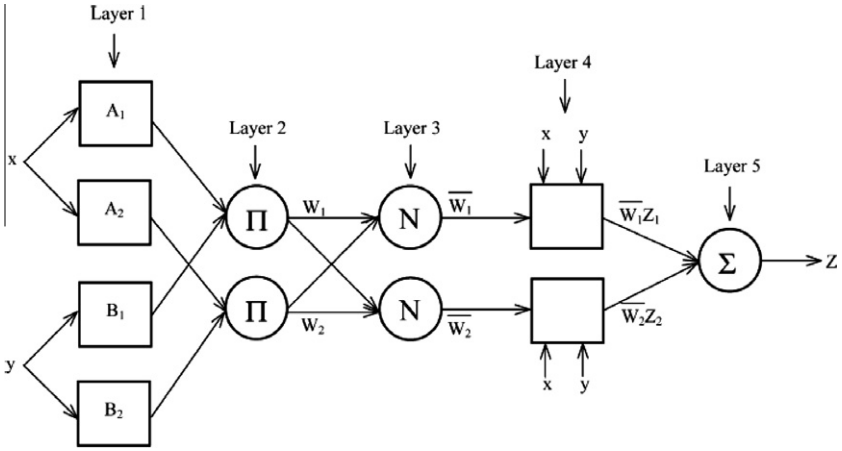



**Fig. 23.** ANFIS architecture.

Layer 4, the node function in this layer is represented by (33).

$$O_i^4 = \tilde{\omega}_i Z_i = \tilde{\omega}_i (p_i x + q_i y + r_i), \quad i = 1, 2 \tag{33}$$

where $\varpi_i$ is the output of layer 3, $\{p_i, q_i, r_i\}$ and is the parameter set. Parameters in this layer are referred to as the consequent parameters.

Layer 5, the single node in this layer computes the overall output as the summation of all incoming signals like (34)

$$O_1^5 = \sum_{i-1}^{2} \varpi_i z_i = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2} \tag{34}$$

It is seen from the ANFIS architecture that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters:

$$z = (\tilde{\omega}_1 x) p_1 + (\tilde{\omega}_1 y) q_1 + (\tilde{\omega}_1) r_1 + (\tilde{\omega}_2 x) p_2 (\tilde{\omega}_2 y) q_2 + (\tilde{\omega}_2) r_2 \tag{35}$$

The hybrid learning algorithm (Jang, 1993) and (Jang, Sun, & Mizutani, 1997) combining the least square method and the back propagation (BP) algorithm can be used to solve this problem. This algorithm converges much faster since it reduces the dimension of the search space of the BP algorithm. During the learning process, the premise parameters in layer 1 and the consequent parameters in layer 4 are tuned until the desired response of the FIS is achieved. The hybrid learning algorithm has a two-step process. First, while holding the premise parameters fixed, the functional signals are propagated forward to layer 4, where the consequent parameters are identified by the least square method. Second, the consequent parameters are held fixed while the error signals, the derivative of the error measure with respect to each node output, are propagated from the output end to the input end, and the premise parameters are updated by the standard BP algorithm.

## References


Abe, S. (2005). *Support vector machines for pattern classification*. Springer-Verlag London Limited.

Abe, S. (2005). *Advances in pattern recognition*. Springer-Verlag London Limited. ISSN: 1617-7916.

Bishop, C. (1995). *Neural networks for pattern recognition*. Walton Street, Oxford OX2 6DP: Oxford University Press.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics, 19*, 1–141.

Huang, T., Kecman, V., & Kopriva, I. (2006). *Kernel based algorithms for mining huge data sets*. Berlin, Heidelberg: Springer-Verlag.

UC Irvine Machine Learning Repository. <http://archive.ics.uci.edu/ml/>

Jang, J.-S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics, 23*(3), 665–685.

Jang, J.-S. R., Sun, C. T., & Mizutani, E. (1997). *Neuro–Fuzzy and Soft Computing: A computational approach to learning and machine intelligence*. Englewood Cliffs, NJ: Prentice-Hall.

Lin, C.-f., & Wang, S.-d. (2002). Fuzzy support vector machine. *IEEE Transactions on Neural Networks, 13*(2), 464–471.

Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics, 33*, 1065–1076.

Sadoghi Yazdi, H., Effati, S., & Saberi, Z. (2007). The probabilistic constraints in the support vector machine. *Applied Mathematics and Computation, 194*(2), 467–479.

Scholkopf, B., & Smola, A. J. (2002). Learning with kernels. *Massachusetts Institute of Technology*.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

Tax, D. M. J. (2001). *One-class classification: concept learning in the absence of counter-examples* (Vol. 65). Netherlands: Technische Universiteit Delft.

Tax, D. M. J., & Duin, R. P. W. (2004). *Support vector data description*. *Machine learning* (vol. 54). Kluwer Academic Publishers (pp. 45–66).

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.

Wang, L. (Ed.). (2005). *Support Vector Machines: Theory and Applications*. Berlin Heidelberg: Springer-Verlag. ISSN: 1434-9922.

Wu, C., Jie-Chi, Y., Lee, Y. (2007). An approximate approach for training polynomial kernel SVMs in linear time. In *45th Annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions, June 2007* (pp. 65–68).

Ypma, R.D. (1998). Support objects for domain approximation. In *Proceedings of international conference on artificial neural networks (ICANN98). Skovde, Sweden.*