Neural Comput & Applic (2012) 21:1697–1707
DOI 10.1007/s00521-011-0762-8

ORIGINAL ARTICLE

DDC: distance-based decision classifier

Javad Hamidzadeh · Reza Monsefi · Hadi Sadoghi Yazdi

Received: 5 September 2010 / Accepted: 22 October 2011 / Published online: 17 November 2011

Abstract This paper presents a new classification method utilizing distance-based decision surface with nearest neighbor projection approach, called DDC. Kernel type of DDC has been extended to take into account the effective nonlinear structure of the data. DDC has some properties: (1) does not need conventional learning procedure (as k-NN algorithm), (2) does not need searching time to locate the k-nearest neighbors, and (3) does not need optimization process unlike some classification methods such as Support Vector Machine (SVM). In DDC, we compute the weighted average of distances to all the training samples. Unclassified sample will be classified as belonging to a class that has the minimum obtained distance. As a result, by such a rule we can derive a formula that can be used as the decision surface. DDC is tested on both synthetic and real-world data sets from the UCI repository, and the results were compared with k-NN, RBF Network, and SVM. The experimental results indicate DDC outperforms k-NN in the most experiments and the results are comparable to or better than SVM with some data sets.

Keywords Distance-based classifier · Projection · Kernel space · Nonlinear surface · k-nearest neighbor · Support vector machine

J. Hamidzadeh (✉) · R. Monsefi · H. Sadoghi Yazdi
Department of Computer Engineering,
Ferdowsi University of Mashhad, Mashhad, Iran
e-mail: Javad.Hamidzadeh@stu-mail.um.ac.ir

R. Monsefi
e-mail: monsefi@um.ac.ir

H. Sadoghi Yazdi
e-mail: h-sadoghi@um.ac.ir

1 Introduction

Classification is one of the most important goals of pattern recognition. Classification is the process of determining the label of unlabeled input sample among known labeled samples. Classification can be done based on sample properties. One of these properties is distance. Distance is a numerical description of how far objects are. In the Euclidean space $\Re^n$, the distance between two points is usually given by the Euclidean distance (2-norm distance). Based on the other norms, different distances are used such as 1-, p-, and infinity-norm. In classification, various distances can be used to measure closeness, such as the Euclidean, Mahalanobis distance, or bands distance [1].

In the literature of data classification, there are some methods that classifying based on distance between unclassified input (test) and training samples. One of the classifiers is Minimum Distance Classifier (MDC) [2]. It classifies an unknown sample into a category to which the nearest prototype to the pattern belongs. In this classifier, a Euclidean distance is used as the metric. Senda et al. [3] based on karhunen–loeve expansion omit the redundant calculations of MDC.

In our proposed method, called DDC, similar to k-NN, no explicit training procedure is required. But it presents a decision boundary based on the training set. In the classification phase for determining the label of new sample, on contrary of k-NN, the proposed decision boundary can be used efficiently without any computational cost because it does not need searching time to locate the k-nearest neighbors. Thus, we survey two groups of classifiers as follows: (1) Lazy learning algorithms and (2) Linear decision boundary.

Lazy learning algorithms are learning methods in which no training step is required. Thus, generalization beyond

the training data is delayed until a test sample is ready to be classified. The common used algorithms from this group are 1-NN and k-NN [4]. In this group of algorithms, the entire training data set must be saved in the memory to be used in the test classification phase. In other words, for test classification phase, training data are used as reference data sets. As a result, the main disadvantage of lazy learning for saving reference data sets is a large space requirement [5]. Another disadvantage is those lazy learning methods which are usually slower than nonlazy learning algorithms because they use all the reference data sets, i.e., comparison of each test sample with every prototype in the stored memory makes the method computationally less pleasant [5]. Some of the mentioned algorithms, such as k-NN, required their parameters to be determined by users. More often, the best choice of these parameters depend on the data distribution.

The k-nearest neighbor algorithm (k-NN) is a method to classify the objects based on closest training examples in the feature space. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification [6]. The k-nearest neighbor algorithm is among the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class which includes the nearest prototype sample. If $k = 1$, then the object is simply assigned to the class of its nearest neighbor [6]. Various distances can be used to measure the vicinity (similarity), such as the Euclidean or Mahalanobis distance. Two major weaknesses of k-NN are memory usage and computational time. DDC can overcome these drawbacks. Domeniconi et al. propose an adaptive 1-NN classification method trying to minimize the estimation bias in high dimensions. They estimated a metric for computing neighborhoods based on Chi-squared distance analysis [7]. As a result, they introduced weighted Euclidean distance to overcome heterogeneous input space for high dimensional features. Thus, the neighborhoods are constricted along most influential ones and hence, we can confront with heterogeneous input space for high dimensional features. Vincent et al. [8] discuss the reasons of weak performance of k-NN in comparison with SVM classifier. They have shown geometrically intuition as to why k-NN often performs more poorly than SVM on classification tasks. They also proposed modified k-NN algorithm to overcome this problem.

Nowadays k-NN has been used in various artificial intelligence tasks, such as pattern recognition, data mining, computer vision, and bioinformatics. Various aspects of k-NN development are investigated from algorithm type, computational overheads, and theoretical analysis [9, 10]. To improve the classification performance of k-NN, it has recently been proposed a variety of prototype condensing, distance measure, and weighted NN techniques [11]. Survey of different methods for data reduction can be seen in [12, 13]. Fast nearest neighbor classification has been investigated by efficiently exploiting metric structure and hardware resources [14]. Weighted NN aims to weigh the discriminative capability of different features or different nearest neighbors [15, 16].

Linear decision boundary is used to partition among the sample data sets. Finding a linear decision boundary which separates two classes of pattern vectors is a fundamental problem in pattern recognition. Decision boundaries can be used to enhance results of clustering algorithms [17]. Some of algorithms represent a linear boundary for separating the sample data in the training phase. Takada et al. [18] proposed a geometric algorithm for finding all linear decision boundaries separating two nonoverlapping classes of data sets. LDA, Fisher LDA, Logistic regression, Perceptron, and SVM are some linear discriminative classifier. Most of these algorithms use optimization to describe decision boundary. For example, SVM is the one of these algorithms that suggests an optimal marginal boundary between two classes of data samples [19]. In the training phase, SVM used optimization process to find such an optimal margin between two classes. The result of this optimization is the determination of some data samples for two classes as support vectors. The advantage of this method is memory usage to save only these support vectors for usage in the classification phase.

The kernel approach offers an alternative solution to improve the computational power of linear learning machines by mapping the data into a high dimensional feature space. By using the kernel approach, some linear classifiers can be extended to nonlinear classification. Kai et al. [20] proposed kernel-based nearest neighbor algorithm. They used polynomial and radial basis kernel functions (RBF). Results of kernel-based nearest neighbor algorithm are reasonable. Luxburg et al. [21] proposed distance-based classification with the Lipschitz functions. Their main goal is to propose a decision function which has the small Lipschitz constant. In this way the inverse of the Lipschitz constant can be interpreted as the size of separating margin among classes. Thus, their goal is to propose an algorithm to find the small Lipschitz constant. But there are some problems to implement their algorithm efficiently. Also, there are some practical problems to choose a suitable subspace of the Lipschitz functions. As a result, these problems have remained practically unsolved yet. Kosinov et al. [22] proposed a new approach for finding discriminative linear transformations from inter-observation distances. It is a nonparametric distance-based technique. Their method is used for two and multi-class classifiers and it can be extended to a kernel-based

classification. There are some other works that use different distance metrics [23]. Gaitanis et al. proposed a classifier based on hamming distance metric applicable to binary patterns.

In this paper, we propose two decision surfaces, which are a quadratic surface and kernel type of it; those can be used to classify input objects in binary classification. One of the major differences of k-NN among existing algorithms and DDC is to save memory and computational time for classification phase. In the training phase, the coefficient of the proposed decision surface can be obtained. Then, this decision surface can be used in the classification phase.

The remainder of the paper is organized as follows: In Sect. 2, initially, we formulate a decision surface that is based on distance, and then we present a geometrical interpretation of the new proposed distance-based algorithm. In Sect. 3, a kernel type decision surface has been proposed for nonlinear classification. Section 4 shows experimental results of the proposed method. Section 5 finalizes with our conclusions and future work.

2 The proposed method

In this section, we propose a surface based on distance classifier. For the two given classes, we calculate the average distances of all the training samples. Unclassified sample will be classified as belonging to a class that has the smaller average distance. As a result, by such a rule we can derive a formula that can be used as the decision surface. After which, we present a geometrical view of distance-based classifier, and for a new unclassified sample, average distance to each class is being defined.

2.1 Quadratic decision surface for the distance-based classifier

In this section, we derive a formula for determining the decision boundary between two classes of samples. If $d(x, x_i)$ denoted distance of the test sample to x_i of first class and $\hat{d}(x, x_j)$ denoted distance of the test sample to x_j of second class, we can have the following equation for deriving decision boundary. Operator $\oplus$ denoted as one proper field operator. $\mu_1 d(x, x_1) \oplus \mu_2 d(x, x_2) \oplus \ldots \oplus \mu_{n_1} d(x, x_{n_1}) = \hat{\mu}_1 \hat{d}(x, x_1) \oplus \hat{\mu}_2 \hat{d}(x, x_2) \oplus \ldots \oplus \hat{\mu}_{n_2} \hat{d}(x, x_{n_2})$. In this equation, μ_i is a degree related to number of x_i's neighbors; in other words, it is normalized number of neighbors of a sample of a class in a neighborhood radius. Calculation method of μ vector is shown in Table 1. This degree accommodated for noisy data. DDC is based on distance among unclassified sample X and samples of two classes. Our goal is to determine this boundary in such a way that the average distances from two classes be equal (as shown in (1)).

$$\frac{1}{n_1}\sum_{i=1}^{n_1} \mu_i d(x, x_i) = \frac{1}{n_2}\sum_{j=1}^{n_2} \hat{\mu}_j \hat{d}(x, x_j) \tag{1}$$

In Eq. (1), the average distance of X from all samples in each class is considered. We can derive and determine a decision surface. In (1), n_1 and n_2 are used as the number of training samples for class one and two, respectively. We can derive a quadratic equation as decision surface by Eq. (1). In this way, we can present a quadratic formula as decision surface for classifying new unclassified samples. Therefore, we can improve efficiency of original k-NN by introducing the proposed decision surface for $K = N$ (N is total training samples in each class). Also, we can decrease memory usage. In the k-NN, using large space for saving a large number of samples is another of its drawbacks. By using the proposed decision surface, it is not necessary to use a lot of memory for saving all training samples. Thus, in spite of k-NN, the proposed method is not an online algorithm. The deficiency of using (1) is when there is a big difference between the numbers of samples of two classes. Thus, to eliminate this deficiency, we should normalize the volume of data in both classes. One suggested way to overcome this problem is to divide both sides of (1) by the number of training samples; as a result, Eq. (2) is given.

$$\frac{1}{n_1}\left(\frac{1}{n_1}\sum_{i=1}^{n_1} \mu_i d(x, x_i)\right) = \frac{1}{n_2}\left(\frac{1}{n_2}\sum_{j=1}^{n_2} \hat{\mu}_j d(x, x_j)\right) \tag{2}$$

We use the Euclidean distance (2-norm distance). Thus, by substitution distances with this norm, we have (3).

$$n_2^2 * \sum_{i=1}^{n_1} \mu_i \left\|x - x_i^2\right\| = n_1^2 * \sum_{j=1}^{n_2} \hat{\mu}_j \left\|x - x_j^2\right\| \tag{3}$$

Then, by considering $\left\|x - x_i^2\right\| = (x - x_i)^T (x - x_i)$, after manipulating the equation, we have,

Table 1 μ vector calculation

1. **Distance Matrix:** It is a symmetric matrix of $n*n$ dimensions, where n is the number of class instances. The element in row i and column j of this matrix denotes the distance between sample i and j in their class or $d(x_i, x_j)$.
2. **Neighbor Counter Vector (NCV):** It is a vector that maintain for each sample, the number of other samples within a certain given distance. The element number i of NCV can be computed based on row number i of Distance Matrix.
3. ***μ* vector:** Element number i in μ vector is computed as: NCV(i)/Max(NCV), where Max(NCV) denotes the maximum value in NCV vector. It is obvious that $0 \leq \mu_i \leq 1$.

$$x^T x\left(n_2^2\sum_{i=1}^{n_1}\mu_i\right)-2x^T\left(n_2^2\sum_{i=1}^{n_1}\mu_i x_i\right)+n_2^2\sum_{i=1}^{n_1}\mu_i x_i^T x_i = x^T x\left(n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j\right)-2x^T\left(n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j x_j\right)+n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j x_j^T x_j \quad (4)$$

Thus we have,

$$x^T x\left(n_2^2\sum_{i=1}^{n_1}\mu_i-n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j\right)-2x^T\left(n_2^2\sum_{i=1}^{n_1}\mu_i x_i-n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j x_j\right)+n_2^2\sum_{i=1}^{n_1}\mu_i x_i^T x_i-n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j x_j^T x_j=0 \quad (5)$$

The proposed decision boundary is derived where a, b, and c are defined as in (6) to (8), respectively,

$$a=\left(n_2^2\sum_{i=1}^{n_1}\mu_i-n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j\right) \quad (6)$$

$$b=2\left(n_2^2\sum_{i=1}^{n_1}\mu_i x_i-n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j x_j\right) \quad (7)$$

$$c=n_2^2\sum_{i=1}^{n_1}\mu_i x_i^T x_i-n_1^2\sum_{j=1}^{n_2}\hat{\mu}_j x_j^T x_j \quad (8)$$

$$\mathrm{F(x)}=ax^T x+bx+c=0 \quad (9)$$

A quadratic decision surface is shown in (9). Thus, to classify a test sample such as x, it is sufficient to determine the sign of Eq. (9). Input test sample cannot properly be classified if the sign of decision surface is neither positive nor negative. We define function $G(x)$ as $G(x) = \text{sign}(F(x))$. Thus, classification process can be done based on Eq. (10). Here, ω_1 and ω_2 are class labels.

$$\begin{cases}\text{if } G(x)<0 \text{ then } x\in\omega_1\\ \text{if } \mathrm{G}(x)>0 \text{ then } x\in\omega_2\end{cases} \quad (10)$$

Here, we have proposed a binary classification in (10). In the next section, we present the geometrical interpretation of DDC.

2.2 A geometrical interpretation of distance classifier

In this section, we present the geometrical interpretation of DDC. First of all, some used definitions are addressed, and then based on these definitions, a geometrical interpretation of the proposed method (DDC) is presented.

Definition 2.2.1 The projection line is defined as a drawing line between test sample and its closest sample (1-NN). In Fig. 1, XB is the projection line between test sample $\boldsymbol{x}$ and its closest sample B.

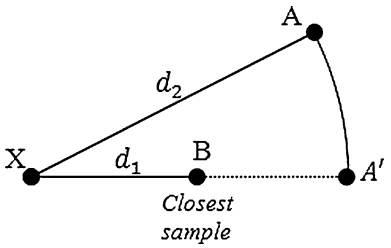

Fig. 1 An equidistance projection of point A onto projection line XB, where $d_2 = XA =$ distance(test_sample,data)

Definition 2.2.2 The equidistance projection of a given sample onto the projection line is such that we draw a circle with the test sample as the center of the circle with the radius equal to ***distance(test_sample,data)*** crossed with the projection line. ***distance(test_sample,data)*** denotes as distance between test sample and the given sample. In Fig. 1, A is a given sample and A' is its equidistance projection of sample A on the projection line XB.

Definition 2.2.3 The average distance of a test sample to a class is defined as the average of equidistance projection of all the class's samples on the projection line.

In Fig. 1, A and B are two given samples of the same class. B is the 1-NN to the test sample X. The sum of distances X from two data A and B is given in (11).

$$d_1+d_2=d_1+XA' \quad (11)$$

All of the data should be projected on the projection line and average of these projected distances is considered the distance of all data to X. In other words, with respect to 1-NN to test sample X, all samples of class projected on the aforementioned projection line passes from X to its 1-NN and average of those projected distances is considered distance X to the class. This projection can be done by considering other samples in addition 1-NN to X. Based on the position of the test sample X, projection line can be different. As shown in Fig. 2, there are different projection lines for different positions of the test sample X because 1-NN for the related X is in a different place.

Figure 3 shows the projected points of samples on projection line that passes through the test sample X and its 1-NN.

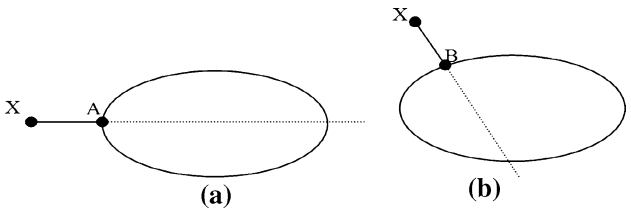

Fig. 2 Different equidistance projection lines based on different X positions **a** A is 1-NN to X **b** B is 1-NN to X

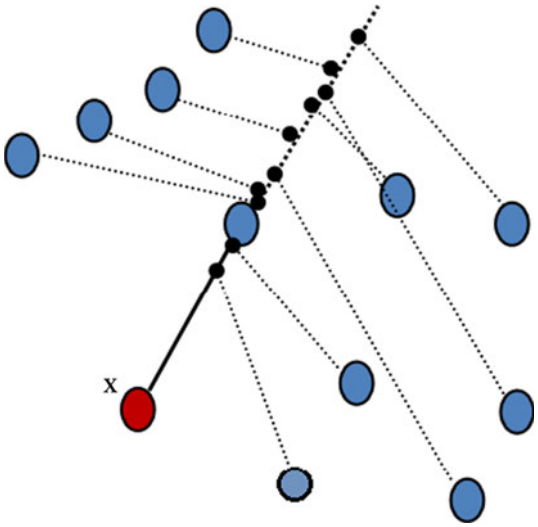

Fig. 3 Projection of samples onto projection line

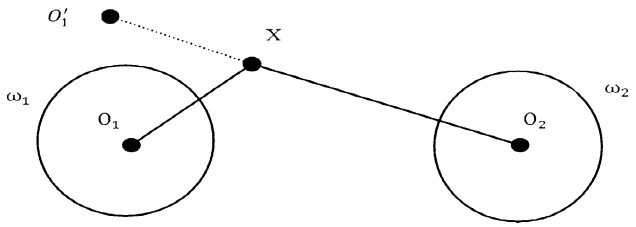

Fig. 4 Distances X from two classes are XO_1 and XO_2, respectively

The questions that may arise here are:-

1. What kinds of effects different lines have on the classification process?
2. What can be the best shape for samples in order to have the best classification process?
3. What is the effect of different numbers of samples in each class?

If data at the two classes are shaped as two spheres with equal radius, there will be similar projections lines regardless of different X positions. And hence there are no problems for different volumes of data for each class. In Fig. 4, XO_1 and XO_2 are represented for average X distances to each of the classes, respectively. As depicted in Fig. 4, X belongs to the class ω_1, because $XO_1 < XO_2$. In other words, X belongs to a class that the average distance is minimized.

If the number of samples in each class is different, then the average distance cannot be satisfactory as a distance metric. As shown in Fig. 5, since radius of class ω_2 is larger than class ω_1, the balance point is absorbed by class ω_2. One approach to overcome this problem is the use of different parameters for normalizing the number of samples in each class. For example, we can use number of samples in each class for normalization. By dividing the average distance to each class by the number of samples in

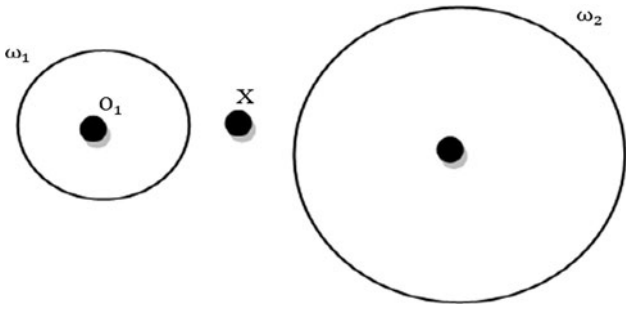

Fig. 5 The balanced point is absorbed by class ω_2

that class, we can reduce the effect of this absorption of the balance point.

3 Kernel type decision surface

Kernel methods are powerful statistical learning techniques, widely applied to various learning algorithms [24, 25]. Kernel methods can be used to transform samples to a high dimensional space. In the high dimensional space, various methods were used to separate samples linearly. A mapping function denoted by φ can be used to transform samples to high dimensional space. In kernel methods, a kernel function can be used to operate without need to compute coordinates at the high dimensional space. In machine learning, this usage of kernel function is known as kernel trick. The kernel trick is a method to use a linear classifier to solve a nonlinear problem by mapping the original observations into a higher-dimensional space to enable us to classify them with the linear classifier. In other words, the linear classifier is subsequently used for linear classification in the new space equivalent to nonlinear classification in the original space. Nonlinear information processing algorithms can be designed by means of linear techniques in implicit feature spaces [26]. By using kernel function, the inner products between the images of the data can be substituted in the feature space. Therefore we have Eq. (12).

$$\mathrm{K}(\mathrm{x},\mathrm{y}) = \langle \varphi(x), \varphi(y) \rangle = \varphi(x)^T \varphi(y) \tag{12}$$

This operation can be done computationally which is more efficient than the explicit computation of the coordinates.

We use φ to map sample features into high dimensional space. Thus, by using Eq. (1) and Euclidean distance in the high dimensional space, Eqs. (13) to (18) are derived to obtain a kernel surface for DDC. We used radial basis function (RBF) kernel as kernel function in DDC. There is one adjustable parameter in RBF kernel.

$$n_2 * \sum_{i=1}^{n_1} \mu_i \|\varphi(x) - \varphi(x_i)\|^2 = n_1 * \sum_{j=1}^{n_2} \hat{\mu}_j \left\|\varphi(x) - \varphi(x_j)\right\|^2 \tag{13}$$

$$n_2 * \sum_{i=1}^{n_1} \mu_i \left[(\varphi(x) - \varphi(x_i))^T (\varphi(x) - \varphi(x_i)) \right] = n_1 * \sum_{j=1}^{n_2} \hat{\mu}_j \left[(\varphi(x) - \varphi(x_j))^T (\varphi(x) - \varphi(x_j)) \right] \quad (14)$$

Since $(\varphi(x)-\varphi(x_i))^T(\varphi(x)-\varphi(x_i)) = \varphi(x)^T\varphi(x) - 2\varphi(x)^T\varphi(x_i) + \varphi(x_i)^T\varphi(x_i)$ and by using (12) we can derive (15).

$$n_2 * \sum_{i=1}^{n_1} \mu_i [k(x,x) - 2k(x,x_i) + k(x_i,x_i)] = n_1 * \sum_{j=1}^{n_2} \hat{\mu}_j [k(x,x) - 2k(x,x_j) + k(x_j,x_j)] \quad (15)$$

We use radial basis kernel as shown in (16).

$$\mathrm{K}(\mathrm{x},\mathrm{y}) = e^{-\frac{\|x-y^2\|}{2\sigma^2}} \quad (16)$$

It is trivial to prove that $K(x, x) = 1$, therefore after making some simplifications, we have (17).

$$n_2 \sum_{i=1}^{n_1} \mu_i - n_2 \sum_{i=1}^{n_1} \mu_i k(x,x_i) = n_1 \sum_{j=1}^{n_2} \hat{\mu}_j - n_1 \sum_{j=1}^{n_2} \hat{\mu}_j k(x,x_j) \quad (17)$$

By substitution of (16) in (17) we can derive (18).

$$n_1 \sum_{j=1}^{n_2} \hat{\mu}_j e^{-\frac{\|x-x_j^2\|}{2\sigma^2}} - n_1 \sum_{j=1}^{n_2} \hat{\mu}_j + n_2 \sum_{i=1}^{n_1} \mu_i - n_2 \sum_{i=1}^{n_1} \mu_i e^{-\frac{\|x-x_i^2\|}{2\sigma^2}} = 0 \quad (18)$$

As a result of using kernel function, we have a nonlinear decision surface as in (18).

A comparison between DDC and SVM hyperplane is presented next. We consider (1) linear separable data and (2) a nonlinear type. The SVM hyperplane formula and DDC for linear separable data are shown in (19) and (20), respectively.

$$f(x) = \sum_{i=1}^{n_1+n_2} \alpha_i y_i x_i x^T + b = \sum_{i=1}^{n_1} \alpha_i x_i x^T + \sum_{i=1}^{n_2} \alpha_i x_i x^T + b \quad (19)$$

$$aX^TX + bX + c = 0 \quad (20)$$

In (19), α_i obtained from optimization process such as Quadratic programming. Some of α_i are zero. For those α_i that are nonzero, x_i are called support vectors and these data must be maintained for classification process. In our decision surface coefficients a, b and c can be easily obtained by Eqs. (6) to (8) without any optimization process. After determining these coefficients, it is not needed to store the training data. In the classification process, Eq. (20) can be used for classifying new unknown sample X. As a result, DDC can save memory usage and time consumption in comparison with standard SVM.

The proposed algorithm is a binary classifier. It is a good idea to extend it to multiclass classifier too. To this end, by using Eq. (13), a discriminate function, $g_j(x)$ can be given as in Eq. (21).

$$g_j(x) = \frac{\sum_{i=1}^{n_j} \mu_i \|\varphi(x) - \varphi(x_i)\|^2}{n_j} = \frac{\sum_{i=1}^{n_j} \mu_i [k(x,x) - 2k(x,x_i) + k(x_i,x_i)]}{n_j} = \frac{2 * \sum_{i=1}^{n_j} \mu_i \left(1 - e^{-\frac{\|x-x_i\|^2}{2\sigma^2}}\right)}{n_j} \quad (21)$$

In Eq. (21), n_j is the number of samples in class j. For classifying test sample X, $g_j(x)$ is computed for all classes (i.e., $j = 1 \ldots$#classes) and then X will be assigned to the class that has the minimum value of $g_j(x)$.

Equations (22) and (23) show SVM and the proposed decision surface, respectively, for classifying nonlinear separable data. As reported for standard SVM (kernel version), most coefficients α_i are nonzero [27, 28]. Hence both methods are the same with respect to the memory usage for storing the learning data. But it is required in SVM to determine α_i from an optimization process and it is time consuming for large amount of data. In DDC, there is no need to do such optimization. Equation (24) shows the one output function in RBF network. In Eq. (24), k, W_{ji}, c_i, and σ_i are used as the number of centers, the weights connect the hidden and output layer, the center and the bandwidth, respectively.

$$f(x) = \sum_{i=1}^{n_1+n_2} \alpha_i y_i K(x_i, x^T) + b = \sum_{i=1}^{n_1} \alpha_i K(x_i, x^T) + \sum_{i=1}^{n_2} \alpha_i K(x_i, x^T) + b \quad (22)$$

$$n_1 \sum_{j=1}^{n_2} \hat{\mu}_j e^{-\frac{\|x-x_j^2\|}{2\sigma^2}} - n_1 \sum_{j=1}^{n_2} \hat{\mu}_j + n_2 \sum_{i=1}^{n_1} \mu_i - n_2 \sum_{i=1}^{n_1} \mu_i e^{-\frac{\|x-x_i^2\|}{2\sigma^2}} = 0 = n_2 \sum_{i=1}^{n_1} \mu_i - n_2 \sum_{i=1}^{n_1} \mu_i K(x,x_i) + n_1 \sum_{j=1}^{n_2} \hat{\mu}_j K(x,x_j) - n_1 \sum_{j=1}^{n_2} \hat{\mu}_j \quad (23)$$

$$f_j(x) = \sum_{j=1}^{k} w_{ji} e^{-\frac{\|x-c_i^2\|}{2\sigma_i^2}} \quad (24)$$

By using kernel function, our proposed kernel surface is similar to RBF networks. An RBF network consists of three layers. These layers are: (1) the input layer, (2) the hidden layer, and (3) the output layer. The input layer is

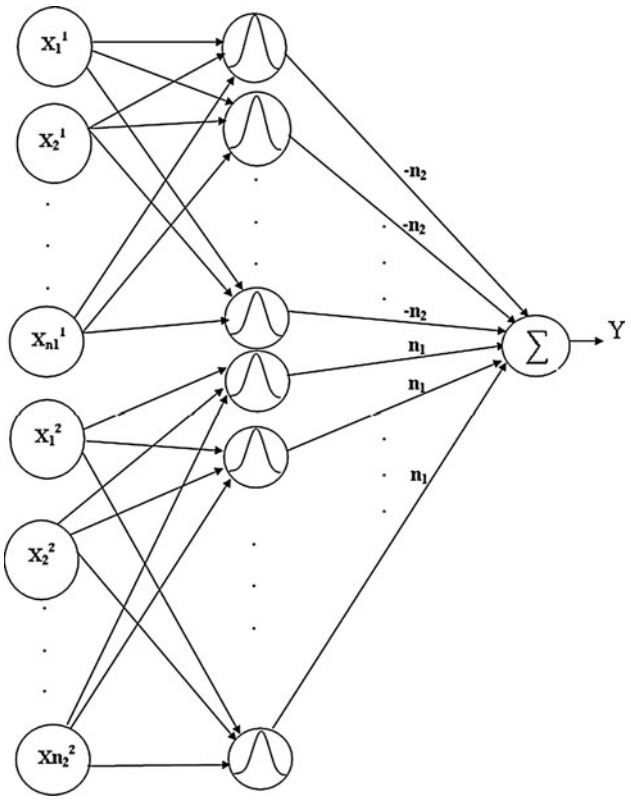

Fig. 6 Kernel decision surface as RBF networks

used to transform the attribute values of input data to each of the units in the hidden layer [29]. Each unit in the hidden layer produces an activation based on its RBF. Thus, the centers of these associated RBFs are samples of DDC. Each unit in the output layer computes a linear combination of the hidden unit activations. The output links of the hidden units are assigned by weight scalars. These weights are the number of samples in DDC. The reaction of RBF networks to a given input is determined by the activation functions assigned to the hidden units and the weights assigned to the output links of the hidden units. Therefore, our proposed kernel surface is similar to RBF networks in which the Gaussian centers are the samples and the weights are the number of samples in each class (see Fig. 6). By the way, $n_2 \sum_{i=1}^{n_1} \mu_i - n_1 \sum_{j=1}^{n_2} \hat{\mu}_j$ is used as the bias. X_i^j is denoted as sample number i of class number j and n_i is used as the number of samples in the class i.

4 Experimental results

In order to validate DDC, the experiments were conducted on both synthetic and real-world data sets. In the following, firstly, we present the results of our experiments on different synthetic data sets and secondly for showing the performance of DDC, we used benchmark data sets from the UCI repository [30]. In all the experiments, tenfold cross-validation is used to evaluate the accuracy of DDC.

4.1 Experimental results on synthetic data sets

We generated synthetic data sets of two classes. We consider the synthetic data sets in Fig. 7. In Fig. 7a, the number of samples in class one and two is 80 and 90, respectively. In Fig. 7b, the number of samples in class one and two is 55 and 110, respectively.

In Fig. 7, there are two classes which are linearly separable. As can be seen in Fig. 7a, our proposed surface decision has been used to classify with the error rate of zero percent.

As the next experiment, we aim to show the effect of high difference between the numbers of samples in each class. As shown in Fig. 7b, we can nevertheless see this difference; the proposed normalizing parameter in Eq. (2) creates no major problem for the proposed decision surface.

In the next experiment, we generate one and two noisy data, respectively, as depicted in Fig. 7c, d. The value of error rate linearly increases with increasing of noisy data. This value of error rate is sensible, because the classifier cannot properly classify the noisy data. Figure 8 shows a two-dimensional data set with 320 samples, whose coordinates are uncorrelated. The samples are separated into two classes, one containing 150 samples and the other 170 samples, both being Gaussian with $\mu_x = -2$, $\mu_y = 0$, $\sigma_x = 2$, $\sigma_y = 1$ and with $\mu_x = 2$, $\mu_y = 0$, $\sigma_x = 1$, $\sigma_y = 2$, respectively. There is some overlap between the two classes. Samples with circle around them are test samples. By using Eq. (18), as decision surface, the boundary between two classes is shown. It is iso-contours with zero value. The error rate for this boundary was 9.82 percent.

In order to classify nonlinear separable data, we performed some other experiments on synthetic data sets. The classes and the results of classification error rate are shown in Fig. 9a, b. For these experiments, we used the Eq. (18) as the decision surface. As shown in Fig. 9, the classifications have been done with error rate of zero percent.

As shown in previous experiments, DDC can be used to classify linear and nonlinear separable data.

4.2 Experimental results on real-world data sets

The experiments reported in this section have been conducted to evaluate the performance (accuracy and efficiency) of DDC in comparison with the alternative data classification methods. The alternative data classification methods involved in the comparison included k-NN, RBF Network, and SVM. For showing the performance of DDC, we used the benchmark data sets from the UCI repository [30]. The selected data sets and their related parameters are

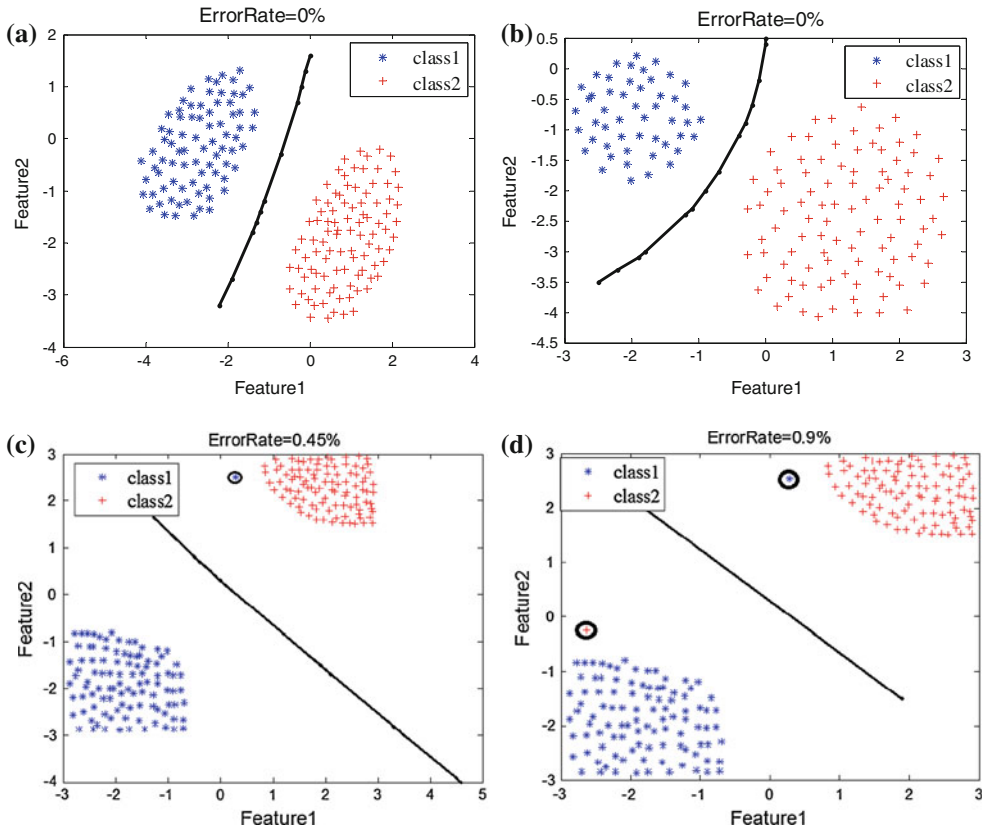

Fig. 7 Using Eq. (9) as decision surface for classifying. in **a** and **b**, decision boundary for separating two classes with equal and different number of samples, respectively, with zero error rate. In **c** and **d**, decision boundary for two classes with one and two noisy samples, respectively

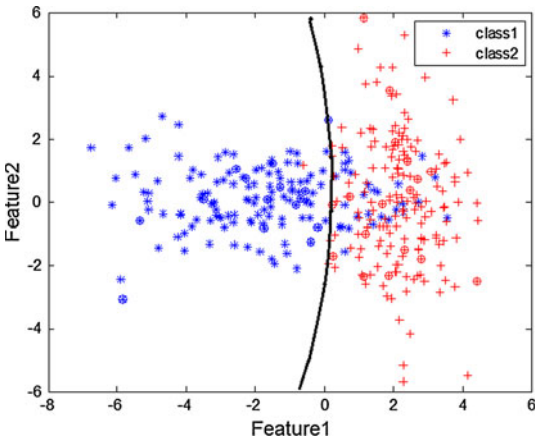

Fig. 8 Using Eq. (18) as decision surface for classifying, with two overlapped data classes

listed in Table 2. In the Table 2, #samples, #features, and #classes denote the number of data samples, the number of the attributes, and the number of the classes, respectively. Note that for data set glass, there is one missing class. That is, in the original application there is one more class but in the data set no samples are with this class.

On each data set, we compared DDC to k-NN, RBF Network, and SVM classifiers. These experiments have been performed in the Matlab R2008a on Intel processor with 2.13GHZ and 4 GB RAM. In these experiments, the SVM in Matlab with the radial basis kernel and the one-against-one practice has been adopted for the SVM and DDC, if the data set contains more than two classes of objects [31, 32]. We also used k-NN in Matlab with Euclidean distance. Tenfold cross-validation has been conducted on the training set to determine the optimal parameter values to be used in the testing phase for the kernel function. For each data set, we estimate the generalized accuracy using different kernel parameters σ in {0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 0.8, 1, 2, 4, 8, 12, 16}. For the SVM, cost parameter C in {0.1, 0.2, 0.4, 0.8, 1, 2, 4, 8, inf} is considered. For k-NN algorithm, we reported the parameter k that has been achieved the lowest error rate. We have performed k-NN with parameter k in {1, 3, 5, …, 19}. We used Gaussian kernel for all algorithms and used the tenfold cross-validation method to estimate the generalization errors of the classifiers.

For DDC and k-NN, we scale data sets. Because we use Euclidean distance, if one of the features of data sets has a very wide range of possible values compared to the other features, it will have a very large effect on the total distance value, and the decisions will be based primarily upon this

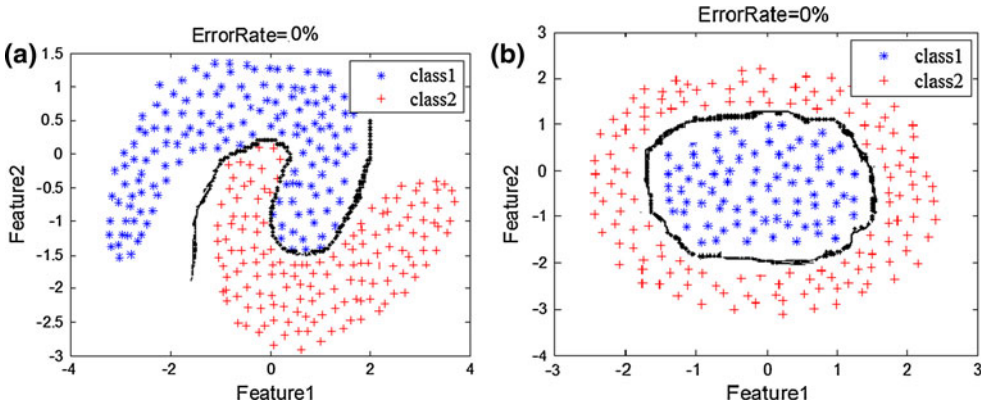

Fig. 9 Using Eq. (18) as decision surface with zero error rates. In **a** and **b,** the decision surface could separate two nonlinear separable classes

Table 2 Data sets

No.	Data set	#samples	#features	#classes
1	Sonar	208	60	2
2	Heart	267	44	2
3	Ionosphere	351	34	2
4	Haberman	306	3	2
5	Liver	345	6	2
6	Diabetes	768	8	2
7	Wdbc	569	30	2
8	Iris	150	4	3
9	Glass	214	9	7
10	Ecoli	336	7	8
11	Segment	2,310	19	7
12	Satimage	4,435	36	6
13	Letter	15,000	16	26
14	Shuttle	43,500	9	7

Table 3 The classification error rates (in percent) and their parameter values for k-NN and DDC classifiers

No.	Data Set	k-NN		DDC		Comparing DDC with k-NN
		K	Rate	$\bar{\sigma}$	Rate	
1	Sonar	1	16.83	0.4	11.06	Better than k-NN
2	Heart	9	20.22	0.75	17.60	Better than k-NN
3	Ionosphere	1	13.68	0.1	5.12	Better than k-NN
4	Haberman	19	24.18	12	24.18	Similar to k-NN
5	Liver	9	31.30	8	32.46	Similar to k-NN
6	Diabetes	7	30.99	4	29.43	Similar to k-NN
7	Wdbc	9	2.46	1	2.99	Similar to k-NN
8	Iris	13	2.0	0.3	2.67	Similar to k-NN
9	Glass	1	26.17	0.1	28.50	Weaker than k-NN
10	Ecoli	5	11.90	0.1	13.39	Weaker than k-NN
11	Segment	3	3.86	2	3.50	Similar to k-NN
12	Satimage	3	9.35	0.8	7.55	Better than k-NN
13	Letter	1	4.32	4	4.33	Similar to k-NN
14	Shuttle	1	2.16	0.8	2.20	Similar to k-NN

single feature. To overcome this, we apply scale factors to the features in the data sets. We scale each feature to have the same standard deviation about its mean.

The experimental results are summarized in Tables 3 and 4. Note that the values in both tables denote the error classification rates (in percent). For multiclass

Table 4 : The classification error rates (in percent) and their parameter values for RBF Network, SVM, and DDC classifiers

No.	Data set	RBF Network	SVM			DDC		**Comparing DDC with RBF Network and SVM**
		Rate	C	б	Rate	б	Rate	
1	Sonar	18.45	0.8	3.2	14.42	0.4	11.06	Better than each two algorithms
2	Heart	20.55	1	4.6	16.85	0.75	17.60	Similar to SVM
3	Ionosphere	6.80	2	4	3.99	0.1	5.12	Between SVM and RBF Network
4	Haberman	26.55	2	2	24.84	12	24.18	Similar to SVM
5	Liver	28.50	2	4	26.96	8	32.46	Weaker than RBF Network and SVM
6	Diabetes	30.45	4	4	27.08	4	29.43	Between SVM and RBF Network
7	Wdbc	5.68	4	2	4.92	1	2.99	Better than each two algorithms
8	Iris	4.67	0.2	2	3.33	0.3	2.67	Similar to SVM
9	Glass	31.84	4	2	28.50	0.1	28.50	Similar to SVM
10	Ecoli	15.80	4	4	12.80	0.1	13.39	Between SVM and RBF Network
11	Segment	6.02	2	1	2.60	2	3.50	Between SVM and RBF Network
12	Satimage	9.75	8	4	8.70	0.8	7.55	Better than each two algorithms
13	Letter	8.84	4	2	2.02	4	4.33	Between SVM and RBF Network
14	Shuttle	2.86	1	8	2.08	0.8	2.20	Similar to RBF Network and SVM

Table 5 : Training time, testing time, and number of support vectors of SVM (Times in ms)

No.	Data set	SVM				K-NN	DDC	
		Training time	Testing time	#SVs	#SVs/training %	Testing time	μ vector calculation time	Testing time
1	Sonar	40.8	2.5	186	89.42	8	1.2	6
2	Heart	1,317.2	2.3	176	65.92	9.5	1.5	7.5
3	Ionosphere	2,836.4	2.9	223	63.53	12.5	1.8	10.4
4	Haberman	55.3	3.2	276	90.20	8.4	1.65	6
5	Liver	188.4	3.3	307	88.99	9.2	1.72	6.8
6	Diabetes	162,481	9.6	646	84.11	24.4	2.10	19
7	Wdbc	16,675	3.7	321	56.41	23.2	2.23	19.6
8	Iris	667.5	8.3	64	36	15.3	0.87	14.49
9	Glass	926.3	38.0	125	58.41	30.1	1.46	31.33
10	Ecoli	5,541.3	50.6	101	30.06	63.5	1.67	64.02
11	Segment	856E5	345.56	896	43.10	98.45	4.53	75.45
12	Satimage	634E7	266.17	2,365	59.26	535.44	6.34	495.38
13	Letter	235E8	483.58	8,945	66.26	716.33	8.23	673.97
14	Shuttle	675E14	654.45	22,679	57.93	938.29	12.34	843.86

classification, we can use classification by pair wise coupling. In this paper, we used the voting approach in comparison with the classifier output approaches for the extension to two-class classifier to multiclass classifier.

As it can be seen, DDC has better performance relative to k-NN and SVM in No. 1 and better results relative to k-NN in No. 1, 2, 3, and 12 and better results relative to SVM in No. 1, 7, and 12. We found DDC from viewpoint of accuracy behave between k-NN and SVM algorithms in the most situations. We also report the training time, testing time for DDC and the SVM in Table 5. We found DDC from viewpoint of training time is similar to k-NN (without training time) and in the testing time is better than k-NN and SVM algorithms. Of course, in *Ecoli* data set, we found SVM give better testing time because of reducing support vectors. So we need a sample reduction procedure for DDC that we will pursuit in the future work.

5 Conclusions and future works

In this paper, a new method for classification without learning phase has been presented. This new method is a distance-based classifier, where we called it DDC. In DDC,

we proposed a decision surface for binary classification process. An input test sample with respect to the decision surface can be assigned to one of the two classes. For multi-class classification, we used classification by pair wise coupling. Here, the voting approaches in comparison with the classifier output approach have been used for the extension of two-class classifier to multi-class classifier.

Results obtained based on both synthetic and benchmark data from the UCI repository show that the accuracy of DDC is reasonable in comparison with the SVM and k-NN algorithms. We found DDC from viewpoint of accuracy behave between k-NN and SVM algorithms in the most situations. Moreover, we excluded training phase in DDC.

To continue this work, we propose the following ideas. To considering other distances such as Mahalanobis, or new distances based on distribution of data for improving performance of DDC. To have a good projection line, it might be a spherical shape for sample data. By using some methods such as kernel, we can have proper shape for data samples in high dimensional space. Data reduction for obtaining decision surface can be considered in the future work.

References

1. Laguia M, Castro JL (2008) Local distance-based classification. Knowl Based Syst 21:692–703
2. Bow ST (2002) Pattern recognition and image preprocessing, 2nd edn. Marcel Dekker, New York
3. Senda S, et al. (1995) A fast algorithm for the minimum distance classifier and its application to kanji character recognition. In: Proceedings of the third international conference on document analysis and recognition, vol 1, pp 283–286
4. Cover TM, Hart PE (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13:21–27
5. Aha DW et al (1991) Instance-based learning algorithms. Mach Lear 6:37–66
6. Duda RO et al (2001) Pattern classification. Wiley Interscience Publication, New York
7. Domeniconi C et al (2002) Locally adaptive metric nearest-neighbor classification. IEEE Trans Pattern Mach Intell 24:1281–1285
8. Vincent P, Bengio Y (2002) K-local hyperplane and convex distance nearest neighbor algorithms, vol 14. The MIT Press, Cambridge
9. Dasarathy BV (1991) Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, Los Alamitos
10. Shakhnarovich G, et al. (2006) (eds) Nearest-neighbor methods in learning and vision: theory and practice. MIT press, Cambridge
11. Lam W et al (2002) Discovering useful concept prototypes for classification based on filtering and abstraction. IEEE Trans Pattern Mach Intell 24:1075–1090
12. Veenman CJ, Reinders MJT (2005) The nearest subclass classifier: a compromise between the nearest mean and nearest neighbor classifier. IEEE Trans Pattern Mach Intell 27:1417–1429
13. Olvera-Lo′pez JA et al (2010) A new fast prototype selection method based on clustering. Pattern Anal Appl 13(2):131–141
14. Herrero JR, Navarro JJ (2007) Exploiting computer resources for fast nearest neighbor classification. Pattern Anal Appl 10:265–275
15. Dudani SA (1976) The distance-weighted k-nearest-neighbor rule. IEEE Trans Syst Man Cybern 6:325–327
16. Zuo W et al (2008) On kernel difference-weighted k-nearest neighbor classification. Pattern Anal Appl 11:247–257
17. Bommanna KR et al (2010) Texture pattern analysis of kidney tissues for disorder identification and classification using dominant Gabor wavelet. Mach Vis Appl 21:287–300
18. Takada Y et al (1994) A geometric algorithm finding set of linear decision boundaries. IEEE Trans Signal Process 42:1887–1891
19. Cortes C, Vapnik V (1995) Support-vector network. Mach Learn 20:273–297
20. Kai Y et al (2002) Kernel nearest neighbor algorithm. Neural Process Letters 15:147–156
21. Luxburg UV, Bousquet O (2004) Distance-based classification with Lipschitz functions. J Mach Lear Res 5:669–695
22. Kosinov S, Pun T (2008) Distance-based discriminant analysis method and its applications. Pattern Anal Appl 11:227–246
23. Gaitanis N, et al. (1993) (eds) Pattern classification using a generalized hamming distance metric. International conference on neural networks
24. Pekalska E, Hassdonk B (2009) Kernel discriminant analysis for positive definite and indefinite kernels. IEEE Trans Pattern Mach Intell 31:1017–1031
25. Li X et al (2009) Kernel-based nonlinear dimensionality reduction for electrocardiogram recognition. Neural Comput Appl 18:1013–1020
26. Ruiz A, Lopez-de-Teruel PE (2001) Nonlinear kernel-based statistical pattern analysis. IEEE Trans Neural Netw 12:16–32
27. Downs T et al (2001) Exact simplification of support vector solutions. J Mach Learn 2:293–297
28. Nefedov A et al (2009) Experimental study of support vector machines based on linear and quadratic optimization criteria. DIMACS Technical Report, no. 2009–18, June 2009
29. Orr MJL (1996) Introduction to radial basis function networks. Center Cognitive Science University Edinburgh, UK, Edinburgh
30. Hettich S, et al. (1998) UCI Repository of machine learning databases. Available: http://www.ics.uci.edu/mlearn/MLRepository.html
31. Hastie T, Tibshirani R (1998) Classification by pairwise coupling. Ann Stat 26(2):451–471
32. Tax DMJ, Duin RPW (2005) Using two-class classifiers for multiclass classification. Pattern Recognition Group, Faculty of Applied Science, Delft University of Technology, Delft