Int. J. Mach. Learn. & Cyber. (2017) 8:1585–1593
DOI 10.1007/s13042-016-0529-8

ORIGINAL ARTICLE

Cross kernel distance minimization for designing support vector machines

Yujian Li[1] · Qiangkui Leng[1] · Yaozong Fu[1]

Received: 27 May 2015 / Accepted: 6 April 2016 / Published online: 27 April 2016

Abstract Cross distance minimization algorithm (CDMA) is an iterative method for designing a hard margin linear SVM based on the nearest point pair between the convex hulls of two linearly separable data sets. In this paper, we propose a new version of CDMA with clear explanation of its linear time complexity. Using kernel function and quadratic cost, we extend the new CDMA to its kernel version, namely, the cross kernel distance minimization algorithm (CKDMA), which has the requirement of linear memory storage and the advantages over the CDMA including: (1) it is applicable in the non-linear case; (2) it allows violations to classify non-separable data sets. In terms of testing accuracy, training time, and number of support vectors, experimental results show that the CKDMA is very competitive with some well-known and powerful SVM methods such as nearest point algorithm (NPA), kernel Schlesinger-Kozinec (KSK) algorithm and sequential minimal optimization (SMO) algorithm implemented in LIBSVM2.9.

Keywords Cross distance minimization algorithm · Kernel function · Convex hull · Nearest point · Support vector machine

✉ Yujian Li
liyujian@bjut.edu.cn

Qiangkui Leng
qkleng@gmail.com

[1] College of Computer Science, Beijing University of Technology, Beijing 100124, China

1 Introduction

Classification is one of the most widely studied problems within the field of machine learning. As one of the commonest tasks in supervised learning, it is the process of finding a model or function that learns a mapping between the set of observations and their respective classes [1, 2]. As a successful classifier, support vector machines (SVMs) have attracted much interest in theory and practice [3–6].

Respectively, let $X = \{x_i : i \in I\}$ and $Y = \{y_j : j \in J\}$ denote two classes of data points in $\mathbf{R}^n$ ($\mathbf{R}$ stands for the set of all real numbers, n for dimension), and I and J are the corresponding sets of indices. If X and Y are linearly separable, the task of a SVM is to find the maximal margin hyperplane,

$$f(x) = w^* \cdot x + b^*, \tag{1}$$

where (w^*, b^*) is the unique solution of the following quadratic model:

$$\begin{aligned} &\min \frac{1}{2}\|w\|^2 \\ &s.t.\, w \cdot x_i + b \geq 1, i \in I; w \cdot y_j + b \leq -1, \quad j \in J. \end{aligned} \tag{2}$$

It has been shown that $M = 2/||w||$ is the maximal margin, and (w^*, b^*) could be computed from the nearest point pair (x^*, y^*) between the two convex hulls $CH(X)$ and $CH(Y)$ [7, 8],

$$w^* = \frac{2(x^* - y^*)}{\|x^* - y^*\|^2}, \quad b^* = \frac{\|y^*\|^2 - \|x^*\|^2}{\|x^* - y^*\|^2},$$

where (x^*, y^*) is the solution of the following nearest point problem (NPP),

$$\min\|x - y\| \quad s.t.\, x \in CH(X), y \in CH(Y). \tag{3}$$

Cross distance minimization algorithm (CDMA) [9] is an iterative algorithm for computing a hard margin SVM via the nearest point pair between $CH(X)$ and $CH(Y)$. The CDMA is simple to implement with fast convergence, but it works only on linearly separable data. In this paper, we first give a new version of the CDMA with clear explanation of its linear time complexity. Then we extend the new CDMA to solving the non-linear and non-separable problems, in a similar way to generalize the S-K algorithm [10].

This paper is organized as follows. Section 2 describes some related work to the CDMA. Section 3 presents a new version of the CDMA and introduces its kernel version. Section 4 provides experimental results and Sect. 5 concludes the paper.

2 Related work

2.1 Sequential minimal optimization algorithm

The basic idea of sequential minimal optimization (SMO) algorithm is to obtain an analytical solution for subproblems resulting from the working set decomposition [11]. The performance of SMO could be further improved by the maximal violating pair working set selection [12, 13]. LIBSVM [14] is a well-known SMO-type implementation, which solves a linear cost soft margin SVM below:

$$\begin{aligned}&\min \frac{1}{2}\|w\|^2 + C\left(\sum_{i\in I}\xi_i + \sum_{j\in J}\zeta_j\right)\\ &s.t.\ w\cdot\phi(x_i) + b \geq 1 - \xi_i, \quad \xi_i \geq 0, i\in I;\\ &w\cdot\phi(y_j) + b \leq -1 + \zeta_j, \quad \zeta_j \geq 0, j\in J.\end{aligned} \tag{4}$$

In Eq. (4), ξ_i and ζ_j denote the slack variables corresponding to two classes.

2.2 Nearest point algorithm

The most similar approach to our work is the nearest point algorithm (NPA) by Keerthi et al. [7], which combines the Gilbert's algorithm [15] and the Mitchell algorithm [16] to find the nearest points between the two convex hulls $CH(X)$ and $CH(Y)$. In order to deal with the non-linear and non-separable cases, the NPA solves a new nearest point problem that is equivalent to the following quadratic cost soft margin SVM [17] (QCSM-SVM),

$$\begin{aligned}&\min \frac{1}{2}\|w\|^2 + C\left(\sum_{i\in I}\xi_i^2 + \sum_{j\in J}\zeta_j^2\right)\\ &s.t.\, w\cdot\phi(x_i) + b \geq 1 - \xi_i, \quad i\in I;\\ &w\cdot\phi(y_j) + b \leq -1 + \zeta_j, \quad j\in J.\end{aligned} \tag{5}$$

Note that negative values of ξ_i or ζ_j cannot occur at the optimal solution of QCSM-SVM because we may reset $\xi_i = 0$ ($\zeta_j = 0$) to make the solution still feasible and also strictly decrease the total cost if $\xi_i < 0$ ($\zeta_j < 0$). Thus, there is no need to include nonnegativity constraints on ξ_i and ζ_j.With a simple transformation, the QCSM-SVM model can be converted into an instance of hard margin SVM (Eq. 2) with the dimensionality of $n + |I| + |J|$ by mapping $x_i \in X$ to $x_i' \in X'$ and $y_j \in Y$ to $y_j' \in Y'$ as follows:

$$\begin{aligned}x_i' &= \left(x_i, \frac{e_i}{\sqrt{2C}}\right) = [x_i, 0, \ldots, \frac{1}{\sqrt{2C}}, \ldots, 0],\\ y_j' &= \left(y_j, \frac{-e_{|I|+j}}{\sqrt{2C}}\right) = [y_j, 0, \ldots, \frac{-1}{\sqrt{2C}}, \ldots, 0],\end{aligned} \tag{6}$$

where e_k denotes the $n + |I| + |J|$ dimensional vector in which the kth component is one and all other components are zero. Accordingly, w' and b' can be computed as:

$$w' = [w, \sqrt{2C}\xi_1, \ldots, \sqrt{2C}\xi_{|I|}, \sqrt{2C}\zeta_1, \ldots, \sqrt{2C}\zeta_{|J|}], b' = b. \tag{7}$$

The kernel function transformed is given by

$$K'(x_i, x_j) = K(x_i, x_j) + \delta_{i,j}\frac{1}{2C}, \tag{8}$$

where K denotes the original kernel function, $\delta_{i,j}$ is one if $i = j$ and zero otherwise. Thus, for any pair of (x_i, y_j) the modified kernel function is easily computed.

2.3 Schlesinger-Kozinec algorithm

Another closely related approach is Schlesinger-Kozinec (SK) algorithm [10]. The original SK algorithm solves the NPP problem by computing an approximate point pair (x^*, y^*) to construct a ε-optimal hyperplane $w^*\cdot x + b^* = 0$, where $w^* = x^* - y^*$ and $b^* = (\| y^*\|^2 - \| x^*\|^2)/2$. This algorithm starts from two arbitrary points $x^* \in CH(X), y^* \in CH(Y)$ and stops when satisfying the ε-optimal criterion, namely,

$$\| x^* - y^* \| - m(v_t) < \varepsilon,$$

where $v_t \in X\cup Y$, $t = \arg\min_{i\in I\cup J} m(v_i)$, and

$$m(v_i) = \begin{cases}\dfrac{(v_i - y^*)\cdot(x^* - y^*)}{\| x^* - y^* \|}, & v_i \in X, i\in I\\ \dfrac{(v_i - x^*)\cdot(y^* - x^*)}{\| x^* - y^* \|}, & v_i \in Y, i\in J\end{cases}$$

The SK algorithm has been generalized to its kernel version (KSK) [10]. It should be noted that in the SK algorithm, if $v_t = x_t (t\in I)$ violates the ε-optimal criterion, x^*

will be updated by the Kozinec rule $x^{new} = (1-\lambda)x^{*} + \lambda x_t$, with y^{*} fixed and λ computed for minimizing the distance between x^{new} and y^{*}. Otherwise, if $v_t = y_t (t \in J)$ violates the ε-optimal criterion, y^{*} will be updated by the Kozinec rule $y^{new} = (1-\mu)y^{*} + \mu y_t$, with x^{*} fixed and μ computed for minimizing the distance between y^{new} and x^{*}. If $\| x^{*} - y^{*} \| - m(v_t) < \varepsilon$ holds, $w^{*} = x^{*} - y^{*}$ and $b^{*} = (\| y^{*}\|^2 - \| x^{*}\|^2)/2$ gives an ε-optimal solution, which coincides with the optimal hyperplane for $\varepsilon = 0$.

3 Cross kernel distance minimization algorithm

3.1 Cross distance minimization algorithm

Let d stand for the Euclid distance function. In this subsection, we will depict a new version of the CDMA in Algorithm 2, where ε is called “precision parameter”, for controlling the convergence condition.

The original CDMA (OCDMA, see Algorithm 1) is designed to iteratively compute the nearest points of two convex hulls. If (x^{*}, y^{*}) is a nearest point pair obtained by solving the NPP in (3), i.e., $d(CH(X), CH(Y)) = ||x^{*} - y^{*}||$, then it can be proven that their perpendicular bisector is exactly the hard-margin SVM of X and Y [9].

Algorithm 1 OCDMA

```
Input: Two finite sets $X, Y \subseteq \mathbf{R}^n$, precision parameter $\varepsilon$.
1: $x^{*} \in X, y^{*} \in Y$;
2: $x_1 = x^{*}, y_1 = y^{*}$;
3: $x^{*} = \arg\min_z \left\{ d(z, y^{*}) \,\middle|\, \begin{array}{r} z \in X \text{ or } \\ z = x_1 + \lambda(x_2 - x_1), \\ x_2 \neq x_1, x_2 \in X \end{array} \right\}$,
   where $0 < \lambda = \frac{(x_2-x_1)\cdot(y^{*}-x_1)}{(x_2-x_1)\cdot(x_2-x_1)} < 1$;
4: $y^{*} = \arg\min_z \left\{ d(x^{*}, z) \,\middle|\, \begin{array}{r} z \in Y \text{ or } \\ z = y_1 + \mu(y_2 - y_1), \\ y_2 \neq y_1, y_2 \in Y \end{array} \right\}$,
   where $0 < \mu = \frac{(y_2-y_1)\cdot(x^{*}-y_1)}{(y_2-y_1)\cdot(y_2-y_1)} < 1$;
5: if $d(x_1, y_1) - d(x^{*} - y^{*}) \geqslant \varepsilon$, goto Step2;
6: $w^{*} = x^{*} - y^{*}, b^{*} = (\|y^{*}\|^2 - \|x^{*}\|^2)/2$;
Output: $f(x) = w^{*} \cdot x + b^{*}$.
```

The procedure of OCDMA includes two basic blocks. The steps 3–4 are to compute a nearest point pair (x^{*}, y^{*}) between $CH(X)$ and $CH(Y)$. The steps 5–6 are to construct a SVM, $f(x) = w^{*} \cdot x + b^{*}$.

Algorithm 2 CDMA

```
Input: Two finite sets $X, Y \subseteq \mathbf{R}^n$, precision parameter $\varepsilon$.
1: $x^{*} \in X, y^{*} \in Y$;
2: $x_1 = x^{*}, y_1 = y^{*}$;
3: $x^{*} = \arg\min_z \{ d(z, y^{*}) \,|\, z = x_1 + \lambda(x_2 - x_1),\ \lambda = \min\left\{1, \frac{(x_2-x_1)\cdot(y^{*}-x_1)}{(x_2-x_1)\cdot(x_2-x_1)}\right\} > 0, x_2 \neq x_1, x_2 \in X \}$;
4: $y^{*} = \arg\min_z \{ d(x^{*}, z) \,|\, z = y_1 + \mu(y_2 - y_1),\ \mu = \min\left\{1, \frac{(y_2-y_1)\cdot(x^{*}-y_1)}{(y_2-y_1)\cdot(y_2-y_1)}\right\} > 0, y_2 \neq y_1, y_2 \in Y \}$;
5: if $d(x_1, y_1) - d(x^{*} - y^{*}) \geqslant \varepsilon$, goto Step2;
6: $w^{*} = x^{*} - y^{*}, b^{*} = (\|y^{*}\|^2 - \|x^{*}\|^2)/2$;
Output: $f(x) = w^{*} \cdot x + b^{*}$.
```

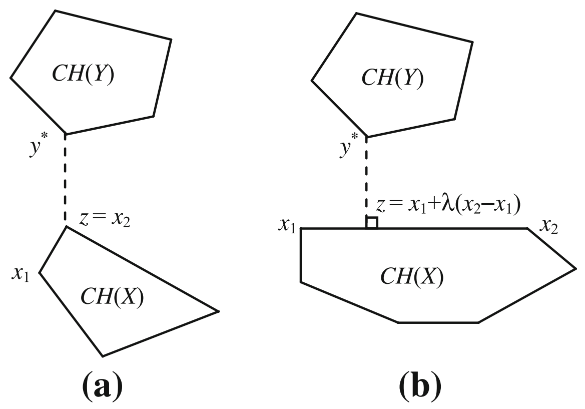

Fig. 1 The geometric meaning of point z: (a) if $\lambda = 1$, then $z = x_2$ is a point in X; (b) if $0 < \lambda < 1$, then $z = x_1 + \lambda(x_2 - x_1)$ is actually the vertical point from y^{*} to the line segment $CH\{x_1, x_2\}$. If x_1 is not the nearest point from y^{*} to $CH(X)$, there must exist another point $z = x_2$ or $z = x_1 + \lambda(x_2 - x_1)$ such that $d(z, y^{*}) < d(x^{*}, y^{*})$

Compared to the OCDMA, the new CDMA can reduce half an amount of distance computations needed in the case $\lambda > 0$ by minimizing $d(z, y^{*})$ instead of $d(x_2, y^{*})$ and $d(z, y^{*})$ as well as $d(x^{*}, z)$ instead of $d(x^{*}, y_2)$ and $d(x^{*}, z)$. The geometric meaning of z point is illustrated in Fig. 1. According to Theorem 5 in Ref. [9], if $x_1 = x^{*}$ is not the nearest point from y^{*} to $CH(X)$, there must exist another point $x_2 \neq x_1$ in X such that $d(z, y^{*}) < d(x^{*}, y^{*})$, where

$$z = x_1 + \lambda(x_2 - x_1), \lambda = \min\left\{1, \frac{(x_2 - x_1)\cdot(y^{*} - x_1)}{(x_2 - x_1)\cdot(x_2 - x_1)}\right\} > 0.$$

Similarly, if $y_1 = y^{*}$ is not the nearest point from x^{*} to $CH(Y)$, there must exist another point $y_2 \neq y_1$ in Y such that $d(x^{*}, z) < d(x^{*}, y^{*})$, where

$$z = y_1 + \mu(y_2 - y_1), \mu = \min\left\{1, \frac{(y_2 - y_1)\cdot(x^* - y_1)}{(y_2 - y_1)\cdot(y_2 - y_1)}\right\} > 0.$$

Therefore, if (x^*, y^*) is not the nearest point pair, the CDMA can always find a nearer one until it converges.

It has been shown that CDMA converges with a rough time complexity of $O(I(\varepsilon)\cdot(|X| + |Y|))$, where $I(\varepsilon)$ is guessed a constant as an open problem [9], representing the total number of running loops for CDMA to converge at ε-precision. In fact, $I(\varepsilon)$ can be clearly bounded by a simple constant, namely, $I(\varepsilon) \le L/\varepsilon$, where L may be estimated as $L = \max_{x\in X, y\in Y}\{\|x - y\|\}$, because we cannot get a cross distance decrease less than ε in each running loop until satisfying the stopping condition for CDMA.

3.2 Cross kernel distance minimization algorithm

Based on the description mentioned above, now we generalize the CDMA to its kernel extension, namely, cross kernel distance minimization algorithm (CKDMA). Given a kernel function $K(x_i, y_j) = \phi(x_i)\cdot\phi(y_j)$, the objective of CKDMA is to minimize the kernel distance $kd(\tilde{x}, \tilde{y})$ between $\tilde{x} \in CH(\phi(X))$ and $\tilde{y} \in CH(\phi(Y))$,

$$\begin{aligned} &\min kd(\tilde{x}, \tilde{y}) = \min\|\tilde{x} - \tilde{y}\| \\ &s.t.\, \tilde{x} \in CH(\phi(X)), \tilde{y} \in CH(\phi(Y)). \end{aligned} \tag{9}$$

For $\tilde{x}^* \in CH(\phi(X))$ and $\tilde{y}^* \in CH(\phi(Y))$, we can express them as,

$$\begin{aligned} \tilde{x}^* &= \sum_{i\in I}\alpha_i\cdot\phi(x_i) = \sum_{i\in I}\alpha_i\cdot\tilde{x}_i, \sum_{i\in I}\alpha_i \ge 0, \tilde{x}_i \in \phi(X), \\ \tilde{y}^* &= \sum_{j\in J}\beta_j\cdot\phi(y_j) = \sum_{j\in J}\beta_j\cdot\tilde{y}_j, \sum_{j\in J}\beta_j \ge 0, \tilde{y}_j \in \phi(Y). \end{aligned} \tag{10}$$

If $\tilde{x}^*$ is not the nearest point from $\tilde{y}^*$ to $CH(\phi(X))$, we can find $t_1 \in I$ to compute a new point $\tilde{x}^{new} = (1-\lambda)\tilde{x}^* + \lambda\tilde{x}_{t_1}$, where $\lambda = \min[1, ((\tilde{x}_{t_1} - \tilde{x}^*)\cdot(\tilde{y}^* - \tilde{x}^*))/((\tilde{x}_{t_1} - \tilde{x}^*)\cdot(\tilde{x}_{t_1} - \tilde{x}^*))]$, such that $\tilde{x}^{new}$ is nearer to $\tilde{y}^*$ than $\tilde{x}^*$, i.e., $\|\tilde{x}^{new} - \tilde{y}^*\| < \|\tilde{x}^* - \tilde{y}^*\|$.

Because $\tilde{x}^{new} = \sum_{i\in I}\alpha_i^{new}\tilde{x}_i = (1-\lambda)\sum_{i\in I}\alpha_i\tilde{x}_i + \lambda\tilde{x}_{t_1} = \sum_{i\in I}[(1-\lambda)\alpha_i + \lambda\delta_{i,t_1}]\tilde{x}_i$, $\tilde{x}^{new}$ can be obtained by updating α_i, namely,

$$\alpha_i^{new} = (1-\lambda)\alpha_i + \lambda\delta_{i,t_1}, \tag{11}$$

where $\delta_{i,j}$ is one if $i = j$ and zero otherwise.

Similarly, if $\tilde{y}^*$ is not the nearest point from $\tilde{x}^*$ to $CH(\phi(Y))$, we can also find a nearer point $\tilde{y}^{new} = (1-\mu)\tilde{y}^* + \mu\tilde{y}_{t_2}, t_2 \in J$ with the adaptation rule below,

$$\beta_j^{new} = (1-\mu)\beta_j + \mu\delta_{j,t_2}. \tag{12}$$

For convenience, we further introduce the following notations:

$$\begin{aligned} A &= \tilde{x}^*\cdot\tilde{x}^* = \sum_{i\in I}\sum_{j\in J}\alpha_i\alpha_j K(x_i, x_j), \\ B &= \tilde{y}^*\cdot\tilde{y}^* = \sum_{i\in I}\sum_{j\in J}\beta_i\beta_j K(y_i, y_j), \\ C &= \tilde{x}^*\cdot\tilde{y}^* = \sum_{i\in I}\sum_{j\in J}\alpha_i\beta_j K(x_i, y_j), \\ \forall i \in I, D_i &= \phi(x_i)\cdot\tilde{x}^* = \sum_{j\in I}\alpha_j K(x_i, x_j), \\ \forall i \in I, E_i &= \phi(x_i)\cdot\tilde{y}^* = \sum_{j\in J}\beta_j K(x_i, y_j), \\ \forall j \in J, F_j &= \phi(y_j)\cdot\tilde{x}^* = \sum_{i\in I}\alpha_i K(x_i, y_j), \\ \forall j \in J, G_j &= \phi(y_j)\cdot\tilde{y}^* = \sum_{i\in J}\alpha_i K(y_i, y_j). \end{aligned}$$

Using these notations to express the CDMA in terms of kernel inner product, it is easy to obtain the CKDMA described in Algorithm 2. Note that sqrt indicates the square root function.

Algorithm 3 CKDMA

```
Input: Two finite sets $X, Y \subseteq \mathbf{R}^n$, precision parameter $\varepsilon$.
1: Initialization: $\alpha_{i_1} = 1, i_1 \in I$; $\beta_{j_1} = 1, j_1 \in J$; $\alpha_i = 0, i \ne i_1$; $\beta_j = 0, j \ne j_1$;
2: Compute: $A, B, C, D_i, E_i, F_j, G_j$; $d^{new} = \mathbf{sqrt}(A + B - 2C)$;
3: $d^* = d^{new}$;
4: $t_1 = \arg\min_i\{\mathbf{sqrt}((1-\lambda_i)^2 A + \lambda_i^2 K(x_i,x_i) + B + 2\lambda_i(1-\lambda_i)D_i - 2(1-\lambda_i)C - 2\lambda_i E_i)\}, \lambda_i = \min\left\{1, \frac{A-C-D_i+E_i}{A-2D_i+K(x_i,x_i)}\right\}, i \in I$;
5: Update: $\alpha_i = \alpha_i(1-\lambda_{t_1}) + \lambda_{t_1}\delta_{i,t_1},\ t_1 \in I$;
6: Update: $A, C, D_i, F_j$;
7: $t_2 = \arg\min_j\{\mathbf{sqrt}(A + \mu_j^2 K(y_j,y_j) + (1-\mu_j)^2 B + 2\mu_j(1-\mu_j)G_j - 2(1-\mu_j)C - 2\mu_j F_j)\}, \mu_j = \min\left\{1, \frac{B-C-G_j+F_j}{B-2G_j+K(y_j,y_j)}\right\}, j \in J$;
8: Update: $\beta_j = \beta_j(1-\mu_{t_2}) + \mu_{t_2}\delta_{j,t_2},\ t_2 \in J$;
9: Update: $B, C, E_i, G_j$;
10: $d^{new} = \mathbf{sqrt}(A + B - 2C)$;
11: if $d^* - d^{new} \geqslant \varepsilon$, goto Step3.;
12: $\widetilde{w}^* = \sum_{i\in I}\alpha_i\cdot\phi(x_i) - \sum_{j\in J}\beta_j\cdot\phi(y_j)$, $\widetilde{b}^* = \frac{B-A}{2}$;
Output: $f(x) = \widetilde{w}^*\cdot\phi(x) + \widetilde{b}^*$.
```

In Step 6 of the CKDMA, A, C, D_i, F_j are updated as follows:

$$A^{new} = (1-\lambda)^2 A + 2(1-\lambda)\lambda D_{t_1} + \lambda^2 K(x_{t_1}, x_{t_1}),$$
$$C^{new} = (1-\lambda)C + \lambda E_{t_1},$$
$$D_i^{new} = (1-\lambda)D_i + \lambda K(x_{t_1}, x_i),$$
$$F_j^{new} = (1-\lambda)F_j + \lambda K(x_{t_1}, y_j).$$

And in Step 9, B, C, E_i, G_j are updated as follows:

$$B^{new} = (1-\mu)^2 B + 2(1-\mu)\mu G_{t_2} + \mu^2 K(y_{t_2}, y_{t_2}),$$
$$C^{new} = (1-\mu)C + \mu F_{t_2},$$
$$E_i^{new} = (1-\mu)E_i + \mu K(y_{t_2}, x_i),$$
$$G_j^{new} = (1-\mu)G_j + \mu K(y_{t_2}, y_j).$$

Moreover, in Algorithm 2, d^* stands for the distance between the nearest points, and d^{new} for its updated value. If $d^* - d^{new}$ is less than ε, the CKDMA will return the result of a linear function in kernel space, namely, $f(x) = \tilde{w}^* \cdot \phi(x) + \tilde{b}^*$, which can be further expressed as a kernel form below,

$$f(x) = \left(\sum_{i\in I}\alpha_i K(x_i, x) - \sum_{j\in J}\beta_j K(y_j, x)\right) + \frac{1}{2}\left(\sum_{i\in J}\sum_{j\in J}\beta_i\beta_j K(y_i, y_j) - \sum_{i\in I}\sum_{j\in I}\alpha_i\alpha_j K(x_i, x_j)\right) = \left(\sum_{i\in I}\alpha_i K(x_i, x) - \sum_{j\in J}\beta_j K(y_j, x)\right) + \frac{1}{2}(B - A)$$

In addition, note that CKDMA can only solve separable problems in a kernel-induced feature space. After using Eq. (6) mentioned in Sect. 2.2, it can be directly applied to non-separable cases, for finding a generalized optimal hyperplane with the quadratic cost function.

Finally, it seems that the update rules of the CKDMA are very similar to that of the KSK algorithm [10]. However, there are at least two notable differences between them: (1) in each iteration the KSK algorithm updates only one of two points, either x^* or y^*, whereas the CKDMA generally updates both of them; (2) Given x^* and y^*, the two algorithms may pick different data points even in the same training set for updating the nearest pair. For example, suppose $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1\}$. In Fig. 2a, the KSK algorithm will choose $x_3 \in X$ to update x^* with $x_{KSK}^{new} = (1-\lambda)x^* + \lambda x_3$. This is because

$$m(x_3) = \|q - y^*\| = \frac{(x_3 - y^*)\cdot(x^* - y^*)}{\| x^* - y^* \|},$$
$$m(x_2) = \|p - y^*\| = \frac{(x_2 - y^*)\cdot(x^* - y^*)}{\| x^* - y^* \|},$$

and $m(x_3) < m(x_2)$ (see Sect. 2.3).

But from Fig. 2, the CKDMA will choose $x_2 \in X$ to update x^* with $x_{CKDMA}^{new} = (1-\lambda)x^* + \lambda x_2$, because $d(y^*, CH\{x^*, x_2\}) < d(y^*, CH\{x^*, x_3\})$ (see Sect. 3.1).

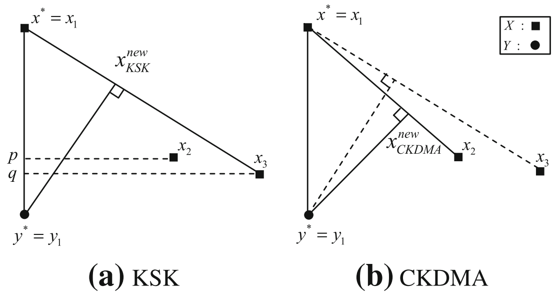

Fig. 2 Different updates of the KSK algorithm and the CKDMA

4 Experimental results

In this section, we conduct a series of numerical experiments to evaluate the classification performances of CKDMA on 17 data sets, which are listed with their code, size and dimensionality in Table 1. All these data sets are selected from the UCI repository [18], but Fourclass from THK96a [19], German.number and Heart from Statlog [20], Leukemia and Gisette from LIBSVM Data [21]. The experiments are divided into two parts. The first part compares the proposed method with KSK, NPA and LIBSVM2.9, and the second part shows the influence of precision parameter ε on CKDMA.

4.1 Comparison with other algorithms

We used 14 data sets in this experiment. The 14 data sets can be seen in Table 2. They are binary problems with features linearly scaled to [−1, 1]. Note that the "training + test" sets have been partitioned for the last four data sets, including mo1, mo2, mo3 and spl. For the sake of homogeneity in results' presentation, we first merge the training and test sets of them. And then, we randomly split each of them into two halves for 10 times, one half for training, the other for testing. We report the average results on all the experiments, which were conducted with RBF kernel function on a Dell PC having I5-2400 3.10-GHz processors, 4-GB memory, and Windows 7.0 operation system. The two parameters, C and γ respectively from the candidate sets $\{2^i | i = -4, -3, \ldots, 3, 4\}$ and $\{2^i | i = -7, -6, \ldots, 4, 5\}$, were determined by 5-fold cross validation on the training sets. The precision parameter ε was chosen as 10^{-5} for CKDMA, 10^{-3} for KSK and NPA, and the default (i.e., 10^{-3}) for LIBSVM 2.9. Note that, in experimental results, the bold values represent higher accuracies, less training time or less support vectors.

From Table 2, it can be seen that CKDMA obtains higher accuracies on six data sets than KSK, six data sets than NPA, and seven data sets than LIBSVM2.9, respectively. Even in the case of lower accuracies, CKDMA can

Table 1 Data sets used in the experiments

Data sets	Code	Size	Dim
Breast cancer	bre	569	30
Fourclass	fou	862	2
German.number	ger	1000	24
Heart	hea	270	13
Ionosphere	ion	351	34
Liver-disorders	liv	345	6
Monks-1	mo1	124 + 432	6
Monks-2	mo2	169 + 432	6
Monks-3	mo3	122 + 432	6
Musk (Version 1)	mus	476	166
Parkinsons	par	195	22
Pima indians diabetes	pim	768	8
Sonar	son	208	60
Splice	spl	1000 + 2175	60
Adult-9	a9a	32561 + 16281	123
Leukemia	leu	38 + 34	7129
Gisette	gis	6000 + 1000	5000

achieve the same level of performance with other algorithms. Moreover, CKDMA generally takes less training time and produces less support vectors than KSK and NPA, but more than LIBSVM2.9.

In addition, we also conducted experiments on three large/high-dimensional data sets (i.e., a9a, leu and gis), with the results described in Table 3. Since it take too long cross validation time to choose C and γ for Libsvm 2.9, we directly set the default values, namely, $C = 1$ and $\gamma = 1/dim$ (0.00813 for a9a, 0.00014 for leu, and 0.00020 for gis). From Table 3, we can get that, on the selected large/high-dimensional data sets, the CKDMA is comparable to the other three classifiers.

Next, we further compare the statistical significance of CKDMA with KSK, NPA, LIBSVM. Referring to [22, 23], we use Bonferroni-Dunn test for comparison over all the selected data sets. Table 4 presents the average ranking of four algorithms, and Fig. 4 provides the corresponding critical difference (CD) diagram. The test results show there are no significant differences between CKDMA and the others.

4.2 Influence of precision parameter

In this subsection we investigate influence of precision parameter ε on performance of CKDMA, with ε set from 10^{-1} to 10^{-6}. We choose eight data sets of different dimensions (i.e., fou, mo1, hea, ger, bre, son, a9a and mus). For mo1 and a9a, we directly use the indicated training and testing sets for conducting experiments. For fou, hea, ger,

Table 2 Comparison of accuracies, training time (ms), and the number of support vectors

Datasets	Accuracies (%)				Training time (ms)				Number of support vectors			
	CKDMA	KSK	NPA	LIBSVM	CKDMA	KSK	NPA	LIBSVM	CKDMA	KSK	NPA	LIBSVM
bre	**97.68 ± 0.8**	97.40 ± 0.9	97.51 ± 1.0	97.37 ± 0.9	19.447	529.407	27.419	**6.957**	77	94	96	**54**
fou	**100.00 ± 0.0**	99.98 ± 0.1	99.98 ± 0.1	99.98 ± 0.1	18.558	120.632	30.161	**9.870**	105	123	111	**92**
ger	75.32 ± 1.9	**75.40 ± 2.0**	75.00 ± 1.8	75.18 ± 1.8	68.660	1619.641	215.068	**33.038**	371	419	426	**282**
hea	82.44 ± 1.9	82.67 ± 1.6	**83.11 ± 2.0**	82.44 ± 2.1	**8.634**	186.724	13.333	11.699	114	117	118	**82**
ion	93.47 ± 1.2	93.41 ± 1.4	**93.81 ± 1.4**	93.01 ± 1.4	15.971	140.350	20.468	**8.854**	104	107	104	**84**
liv	69.9 ± 3.1	68.67 ± 3.6	**70.64 ± 3.2**	68.90 ± 4.0	11.541	418.095	34.832	**5.406**	151	161	161	**128**
mus	88.12 ± 1.2	88.24 ± 1.1	**88.66 ± 1.1**	88.37 ± 1.8	86.074	422.285	149.496	**54.968**	**176**	193	187	177
par	**90.10 ± 2.0**	**90.10 ± 2.1**	90.00 ± 2.2	90.00 ± 2.3	**7.748**	34.813	8.007	10.257	**69**	72	71	70
pim	76.56 ± 1.8	76.38 ± 1.3	76.33 ± 1.4	**76.90 ± 1.6**	28.353	1488.799	102.371	**14.046**	274	339	338	**215**
son	83.14 ± 2.1	83.14 ± 2.3	83.05 ± 2.7	**83.33 ± 2.9**	14.706	128.123	16.227	**8.502**	**81**	84	83	**81**
mo1	91.37 ± 2.88	92.05 ± 2.65	**92.59 ± 2.11**	91.33 ± 2.65	28.085	204.195	82.201	**9.484**	160	165	162	**139**
mo2	89.00 ± 2.07	89.07 ± 1.75	89.73 ± 1.98	**89.87 ± 2.17**	24.770	91.174	57.939	**11.133**	182	214	208	**172**
mo3	95.56 ± 0.72	95.52 ± 0.59	95.56 ± 0.61	**95.81 ± 0.66**	20.095	125.655	35.649	**6.621**	121	138	136	**71**
spl	90.18 ± 0.66	90.33 ± 0.91	**90.42 ± 0.78**	**90.42 ± 0.63**	154.737	757.950	653.110	**98.941**	**744**	1050	995	890

Bold values represent higher accuracies, less training time or less support vectors

Table 3 Comparison of four classifiers on large/high-dimensional data sets

Datasets	Accuracies (%)				Training time (ms)				Number of support vectors			
	CKDMA	KSK	NPA	LIBSVM	CKDMA	KSK	NPA	LIBSVM	CKDMA	KSK	NPA	LIBSVM
a9a	84.66	85.17	**85.23**	84.81	82738.300	850430.000	901374.000	**69742.400**	**10499**	19269	20248	11956
leu	**67.65**	**67.65**	**67.65**	**67.65**	328.368	476.557	361.713	**301.078**	**38**	**38**	**38**	**38**
gis	97.80	97.80	**97.90**	97.70	113918.000	1465460.000	518021.000	**83906.900**	**1293**	2376	2444	1666

Table 4 Average rankings of four algorithms

Algorithm	Ranking
CKDMA	2.5882352941176467
KSK	2.6470588235294112
NPA	2.1176470588235294
LIBSVM	2.6470588235294117

bre, son and mus, we randomly split each of them into two halves only once, one half for training, the other for testing. The corresponding results are shown in Fig. 3, including testing accuracy, training time, and number of support vectors.

From Fig. 3, we can see that as ε gets smaller and smaller, testing accuracies generally tend to increase and become stable. In Fig. 3a, a vertical dashed line marks a good choice of 10^{-5} for CKDMA because it gives rise to satisfactory performance with relatively reasonable training time (see Fig. 3b) and number of support vectors (see Fig. 3c).

5 Conclusions

In this paper, we have explained the linear time complexity for CDMA, and further proposed a new iterative algorithm, i.e., the cross kernel distance minimization algorithm (CKDMA), to train quadratic cost support vector machines using kernel functions. Over CDMA, CKDMA can take advantages to solve non-linear and non-separable problems, preserving the simplicity in geometric meaning. Moreover, it gives a novel way to understand the nature of support vector machines. Since the basic idea of CKDMA is to express the CDMA in kernel-induced feature space, it can be implemented in a similar way to KSK and NPA, but different from LIBSVM. Experimental results show that

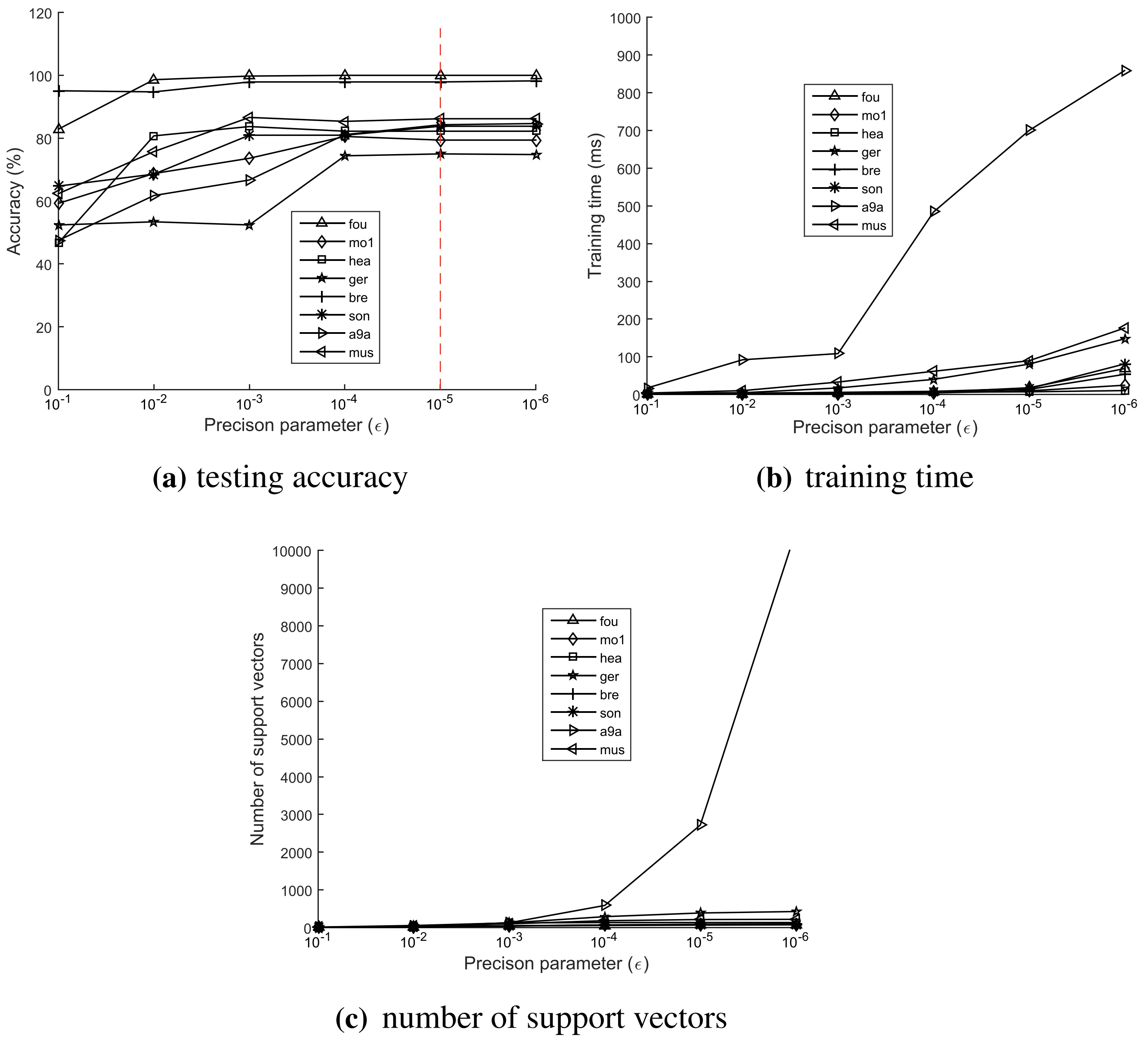

Fig. 3 Experimental results for different precision parameters

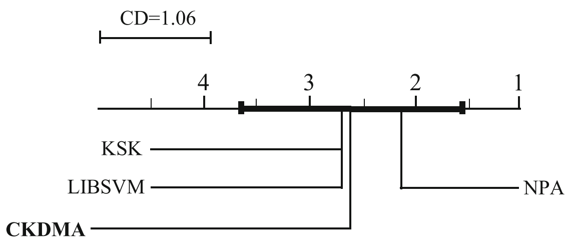

Fig. 4 Comparison of CKDMA against the others with the Bonferroni-Dunn test. All classifiers with ranks outside the marked interval are significantly different ($\alpha = 0.05$) from the control

the CKDMA is very competitive with NPA, KSK and LIBSVM2.9 on the selected data sets. As future work, we would try to introduce large margin clustering [24] for rapid generation of supports vectors.

Acknowledgments This work was supported in part by the National Science Foundation of China under Grant 61175004, the Beijing Natural Science Foundation under Grant 4112009, the Program of Science Development of Beijing Municipal Education Commission under Program KM201010005012, and the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 20121103110029.

References

1. Webb D (2010) Efficient piecewise linear classifiers and applications. Ph. D. Dissertation, The Graduate School of Information Technology and Mathematical Sciences, University of Ballarat
2. Wang X, Ashfaq RAR, Fu A (2015) Fuzziness based sample categorization for classifier performance improvement. J Intell Fuzzy Syst 29(3):1185–1196

3. Ertekin S, Bottou L, Giles CL (2011) Nonconvex online support vector machines. IEEE Trans Pattern Anal Mach Intell 22(2):368–381
4. Wang X, He Q, Chen D, Yeung D (2005) A genetic algorithm for solving the inverse problem of support vector machines. Neurocomputing 68:225–238
5. Wang X, Lu S, Zhai J (2008) Fast fuzzy multi-category SVM based on support vector domain description. Int J Pattern Recognit Artif Intell 22(1):109–120
6. Xie J, Hone K, Xie W, Gao X, Shi Y, Liu X (2013) Extending twin support vector machine classifier for multi-category classification problems. Intell Data Anal 17(4):649–664
7. Keerthi SS, Shevade SK, Bhattacharyya C, Murthy KRK (2000) A fast iterative nearest point algorithm for support vector machine classifier design. IEEE Trans Neural Netw 11(1):124–136
8. Bennett KP, Bredensteiner EJ (1997) Geometry in learning. In: Geometry at work
9. Li Y, Liu B, Yang X, Fu Y, Li H (2011) Multiconlitron: a general piecewise linear classifier. IEEE Trans Neural Netw 22(2):267–289
10. Franc V, Hlaváč V (2003) An iterative algorithm learning the maximal margin classifier. Pattern Recogn 36(9):1985–1996
11. Platt JC (1999) Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges C, Smola A (eds) Advances in kernel methods: support vector learning, MIT Press, Cambridge, p. 185–208
12. Keerthi SS, Shevade SK, Bhattacharyya C, Murthy KRK (2001) Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput 13(3):637–649
13. Shevade SK, Keerthi SS, Bhattacharyya C, Murthy KRK (2000) Improvements to the SMO algorithm for SVM regression. IEEE Trans Neural Netw 11(5):1188–1194
14. Chang CC, Lin CJ (2001) Libsvm: a library for support vector machines (Online). http://www.csie.ntu.edu.tw/cjlin/libsvm. Accessed 10 June 2013
15. Gilbert EG (1966) Minimizing the quadratic form on a convex set. SIAM J Control 4:61–79
16. Mitchell BF, Dem'yanov VF, Malozemov VN (1974) Finding the point of a polyhedron closest to the origin. SIAM J Control 12(1):19–26
17. Friess TT, Harisson R (1998) Support vector neural networks: the kernel adatron with bias and softmargin, Univ. Sheffield, Dept. ACSE, Tech. Rep. ACSE-TR-752
18. Frank A, Asuncion A (2010) UCI machine learning repository (Online). http://archive.ics.uci.edu/ml. Accessed 23 Nov 2013
19. Ho TK, Kleinberg EM (1996) Building projectable classifiers of arbitrary complexity. In: Proceedings of the 13th international conference on pattern recognition, Vienna, p. 880–885
20. Brazdil P, Gama J (1999) Statlog datasets (Online). http://www.liacc.up.pt/ml/old/statlog/-datasets.html. Accessed 25 Apr 2014
21. Lin CJ Libsvm data (Online). https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/. Accessed 11 Aug 2015
22. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
23. Garca S, Herrera F (2009) An extenson on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons. J Mach Learn Res 9:2677–2694
24. Xu L, Hu Q, Hung E, Chen B, Xu T, Liao C (2015) Large margin clustering on uncertain data by considering probability distribution similarity. Neurocomputing 158:81–89