Abstract

Based on the geometric interpretation of support vector machines (SVMs), this paper presents a general technique that allows almost all the existing $L_2$-norm penalty based geometric algorithms, including Gilbert’s algorithm, Schlesinger–Kozinec’s (SK) algorithm and Mitchell–Dem’yanov–Malozemov’s (MDM) algorithm, to be softened to achieve the corresponding learning $L_1$-SVM classifiers. Intrinsically, the resulting soft algorithms are to find $\epsilon$-optimal nearest points between two soft convex hulls. Theoretical analysis has indicated that our proposed soft algorithms are essentially generalizations of the corresponding existing hard algorithms, and consequently, they have the same properties of convergence and almost the identical cost of computation. As a specific example, the problem of solving $\nu$-SVMs by the proposed soft MDM algorithm is investigated and the corresponding solution procedure is specified and analyzed. To validate the general soft technique, several real classification experiments are conducted with the proposed $L_1$-norm based MDM algorithms and numerical results have demonstrated that their performance is competitive to that of the corresponding $L_2$-norm based algorithms, such as SK and MDM algorithms.

1. Introduction

SVMs (support vector machines) are a powerful machine learning method based on the statistical learning theory [1–4]. For linearly separable cases, SVMs attempt to optimize the generalization bound by separating data with a maximal margin classifier. This is known as the maximal margin algorithm and can be formulated as a quadratic programming problem. Geometrically speaking, the margin of a classifier is the minimal distance of training points from the decision boundary and the maximal margin classifier is the one with the maximum distance from the nearest patterns to the boundary, called support vectors.

There exists a huge body of literatures and well-known methods on solving quadratic programming problems. However, most mathematical programming approaches require enormous matrix storages and intensive matrix operations. Fortunately, the structure of the SVM optimization problem permits a class of specially tailored algorithms to be constructed to achieve fast convergence and small memory requirement even for large-scale problems. Over the last decade, many algorithms for solving large-scale SVMs have emerged. Generally speaking, these algorithms are mainly based on two kinds of ideas: (1) solving the quadratic programming problems analytically in the dual space, such as Chunking [5], decomposition method [6] and the well-known sequential minimal optimization (SMO, [7]); and (2) solving the nearest points problem (NPPs) in the sample space based on the geometric interpretation. The first geometric interpretation of SVMs can be traced back to the paper for investigating perceptron by Bennett et al. [8], and many important geometric ideas subsequently appeared in Refs. [9–11]. The most significant fact in the geometric interpretation is that problem of SVM classification can be formulated as solving
a NPP between two convex sets. All these geometric explanations not only help us to understand the maximum margin algorithms intuitively, but also motivate us to develop elegant and practical new algorithms. For example, the nearest point algorithm, which carefully combines Gilbert’s algorithm [12] with Mitchell–Dem’yanov–Malozemov’s (MDM) algorithm [13], as well as certain ideas of SMO, was derived by Keerthi et al. [10]. In our paper [14], we incorporated Gilbert’s algorithm with swap projection method developed in Ref. [15] and obtained an interesting algorithm for large-scale problems. Recently, Franc et al. [16] built an iterative algorithm on top of the Schlesinger–Kozinec’s (SK) algorithm [17,18], which finds the maximal margin classifier for a given precision for separable data. Compared with SMO, the requirement for less memory storage allows the geometric algorithms to be applied in large-scale problems, and those geometric approaches should be viewed as another major contribution in solving SVMs.

As pointed out in Refs. [19,10], by using a modified kernel technique, the principle to solve SVMs with L2-norm penalty is essentially the same as the one to solve linearly separable problems. As a result, almost all the geometric algorithms introduced so far are only good for solving SVMs with L2-norm penalty. Note that currently L1-norm penalty SVMs, such as C-SVM in Ref. [20] and ν-SVM in Ref. [21], are very popular among SVM researchers and many have argued that L1-norm SVMs have real advantage over L2-norm SVMs, especially when the redundant noise features are considered [22]. Naturally, several efforts have been made to develop geometric methods for solving L1-norm penalty SVMs.

Since L1-norm penalty SVMs can generally be interpreted as the NPPs between two special convex sets called the soft convex hulls [9,11], a straightforward way for solving them is to apply the geometric approaches directly on the soft convex hulls. Unfortunately, this intuition does not work. The main reason is that each vertex of a soft convex hull is described as a convex combination of several training points and we have to enumerate all such combinations to ensure that all the vertices are searched in each loop of the geometric algorithms, and this will inevitably cause severe problems in terms of computational cost. Recently, we constructed a new convergent algorithm for solving NPPs between two soft convex hulls [23], which captures the nature of kernel SK algorithms in Ref. [16] and is then called soft SK algorithm. Theoretical analysis demonstrates that our soft SK algorithms enjoy the same properties of convergence and almost the identical cost of computation as that of the corresponding hard algorithms. Furthermore, several experiments illustrate their great potentials in solving large-scale problems.

In this paper, we show that the idea developed in Ref. [23] can be extended to a general technique to adapt almost all the available L2 geometrical algorithms to the corresponding L1-norm algorithms. Here we first soften Gilbert’s algorithm to solve NPPs between two soft convex hulls. Since MDM algorithm usually works faster than Gilbert’s algorithm [10], the technique to soften MDM algorithm for solving ν-SVMs is investigated in detail, including its theoretical analysis and experimental verification. The remainder of paper is organized as follows: Section 2 discusses the geometric interpretation of linear ν-SVM; Sections 3–5 first review SK, Gilbert and MDM algorithms, and then show how they can be adapted to solve NPPs between soft convex hulls; results of several real experiments are described in Section 6 and followed by a brief remark on conclusions and future work in Section 7.

2. The geometric interpretation of linear SVMs

In this section, the definition of SVM with L1 and L2-norm, ν-SVM, and their geometric interpretation are introduced. Let \( X = \{ p_i, i \in I_1 \} \) and \( Y = \{ q_i, i \in I_2 \} \) be the two training sets. Let \( y_i \) be the label of \( x_i \in X \cup Y \) and \( y_i = 1 \) or \(-1\).

Basically, a linearly separable SVM is formulated as the following optimization problem:

\[
\begin{align*}
\min \frac{1}{2} \| w \|^2, \\
y_i (w^T x_i + b) \geq 1, & \quad 1 \leq i \leq l. 
\end{align*}
\]

Let \( \overline{X}_1 \) and \( \overline{X}_2 \) denote the closed convex hulls of \( X \) and \( Y \). Assume that \( w_1^* \in \overline{X}_1 \) and \( w_2^* \in \overline{X}_2 \) are the nearest points between \( \overline{X}_1 \) and \( \overline{X}_2 \). From the geometric interpretation of problem (1) in Refs. [8,10,11], the classification problem between \( \overline{X}_1 \) and \( \overline{X}_2 \) is linearly separable if and only if \( \overline{X}_1 \cap \overline{X}_2 = \emptyset \) and the maximal margin linear classifier decided by optimization problem (1) is the hyperplane which passes through the midpoint of the line connecting \( w_1^* \) and \( w_2^* \) with normal vector \( w_1^* - w_2^* \). Generally, we call the hyperplane \( w^T x + b = 0 \) associated with \( w_1 \) and \( w_2 \) if \( w = w_1 - w_2 \) and \( b = -\frac{1}{2} (\| w_1 \|^2 - \| w_2 \|^2) \). In other words, the hyperplane associated with the two nearest points \( w_1^* \) and \( w_2^* \) is the maximal margin classifier.

For problems which allow classification violations, Cortes and Vapnik proposed a soft algorithm by introducing slack variables \( \xi_i \) in constraints [1,2,20] and penalize such violations linearly in the objective function.

\[
\begin{align*}
\min \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{l} \xi_i, \\
\xi_i \geq 0, & \quad y_i (w^T x_i + b) \geq 1 - \xi_i, & \quad 1 \leq i \leq l. 
\end{align*}
\]

In fact, this idea behind the soft SVM was introduced earlier by Bennett and Mangasarian [24]. Note in problem (2), the penalty term \( \sum_{i=1}^{l} \xi_i \) is in terms of L1-norm. It is for this reason that problem (2) is called SVM with L1-norm penalty. Using the suggestion in problem (2), a sum of squared violations in the cost function can be used and the following optimization problem is obtained:

\[
\begin{align*}
\min \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{l} \xi_i^2, \\
\xi_i \geq 0, & \quad y_i (w^T x_i + b) \geq 1 - \xi_i, & \quad 1 \leq i \leq l. 
\end{align*}
\]

Naturally, problem (3) is called SVM with L2-norm penalty. As pointed out in Refs. [19,10], a very nice property of problem (3) is that by doing a simple transformation it can be converted to an instance of problem (1). With this kind of implicit linear separability, all the geometric NPP methods can be directly
Theorem 2.1. Assume that the solution of Eq. (4) satisfies 
\[ \rho > 0, \]  
then the following statements hold:

(i) \( \nu \) is an upper bound on the fraction of margin errors.
(ii) \( \nu \) is a lower bound on the fraction of support vectors.

Clearly, \( \nu \)-SVMs are also in terms of \( L_1 \)-norm penalty. To explain the \( \nu \)-SVM geometrically, the following definition is developed by Crisp and Burges in Ref. [11].

**Definition 2.1.** Let \( 1 \geq \mu \geq 0 \). \( \{ \sum_{i=1}^{n} \lambda_i p_i : \sum_{i=1}^{n} \lambda_i = 1, \mu \geq \lambda_i \geq 0, i = 1, 2, \ldots, n \} \) is called the soft convex hull with constraint \( \mu \) spanned by \( p_1, p_2, \ldots, p_n \).

In Ref. [11], the authors prove that solving an \( \nu \)-SVMs optimization problem is equivalent to finding the nearest points between two soft convex hulls with \( \mu = 2/\nu \) even when the interaction of the two soft convex hulls is not empty. This fact enlightens us to solve \( \nu \)-SVM by using the geometric approaches. In this paper, a soft SVM classifier is regarded as the hyperplane associated with the two nearest points between two soft convex hulls. This viewpoint is the same as the hard one but a little different from that in Refs. [8,11]. In [8,11], the optimal soft hyperplane is the one which connects two nearest points but not necessarily passes through the midpoint. It should be pointed out that the concept of reduced convex hull proposed by Bennett and Bredensteiner in Ref. [8] is the same with Definition 2.1.

As can be seen in Ref. [11], the two soft convex hulls can be made disjoint if \( \mu \) is selected to be small enough. Therefore, the linear separability in \( \nu \)-SVMs can now be guaranteed without using the transformation for SVMs with \( L_2 \)-norm. Obviously, this fact makes it possible to solve \( \nu \)-SVMs using the geometric method. On the other hand, the authors in Ref. [11] define a “nontrivial” solution of \( \nu \)-SVM to any solution with \( \nu \neq 0 \) and establish some results on the choice of \( \nu \). Especially, they proved.

**Theorem 2.2.** A value \( \nu \) exists which will result in a nontrivial solution to the \( \nu \)-SVM classification problem if and only if the intersection of corresponding two soft convex hulls is empty.

Theorem 2.2 indicates that nonempty interaction of the two soft convex hulls may lead to the trivial solutions of \( \nu \)-SVMs. Hence, in this paper, it is natural to assume that the two soft convex hulls are separable with a choice of \( \mu \).

General-purpose algorithms for NPPs usually terminate within a finite number of steps when the allowed precision is achieved. However, such a stopping criteria is not clear enough for SVMs. In order to stop the geometric algorithms effectively and intuitively, the following optimal classifier with a given precision is introduced in Ref. [16].

**Definition 2.2.** Assume that \( w_1 \in \bar{X}_1 \) and \( w_2 \in \bar{X}_2 \). Let \( m(w^* - w) \) be the margin of the maximal margin hyperplane and \( m(w) \) be the margin of the hyperplane associated with \( w_1 \) and \( w_2 \). If \( m(w^*) - m(w) \leq \epsilon \), \( w_1 \) and \( w_2 \) are called the \( \epsilon \)-optimal nearest points and the hyperplane associated with \( w_1 \) and \( w_2 \) is called the \( \epsilon \)-optimal hyperplane.

3. SK algorithm and its softness

In this section, the important idea in Ref. [23] for softening the SK algorithm is clearly described. To this end, we first introduce the SK algorithm in Ref. [16]. Obviously, we must assume \( X_1 \cap X_2 = \emptyset \). SK’s algorithm.

- **Step 1:** Set the vector \( w_1 \) to any vector \( x \in X_1 \) and \( w_2 \) to any vector \( x \in X_2 \). Set the stopping criterion \( \epsilon \).
- **Step 2:** Find a vector \( x_1 \) closest to the hyperplane associated with \( w_1 \) and \( w_2 \) as \( x_1 = \arg \min \{ m(x_i), i \in I_1 \cup I_2 \} \), where

\[
m(x_i) = \frac{(x_1 - w_2, w_1 - w_2)}{\|w_1 - w_2\|}, \quad \text{for } i \in I_1,
\]
\[
m(x_i) = \frac{(x_1 - w_1, w_2 - w_1)}{\|w_1 - w_2\|}, \quad \text{for } i \in I_2.
\]

If the \( \epsilon \)-optimality condition \( \|w_1 - w_2\| - m(x_i) < \epsilon \) holds, then the vector \( w_1 - w_2 \) and \( b = \frac{1}{2}(\|w_1\|^2 - \|w_2\|^2) \) define the \( \epsilon \)-solution. Otherwise go to Step 3.

- **Step 3:** If \( x_1 \in X_1 \), then set \( w_1^{\text{new}} = w_2 \) and compute

\[
w_1^{\text{new}} = w_1(1 - q) + q x_1,
\]

where \( q = \min \left( 1, \frac{(w_1 - w_2, w_1 - x_i)}{\|w_1 - x_i\|^2} \right) \).

Otherwise \( x_1 \in X_2 \), then set \( w_2^{\text{new}} = w_1 \) and compute

\[
w_2^{\text{new}} = w_2(1 - q) + q x_1,
\]

where \( q = \min \left( 1, \frac{(w_2 - w_1, w_2 - x_i)}{\|w_2 - x_i\|^2} \right) \).

Continue with Step 2.

The above SK algorithm is intuitively illustrated in Ref. [16]. Based on the geometric interpretation, a natural idea for solving
According to Definition 2.1, each vertex of the soft convex hull is linearly separable now. The soft SK algorithm in Ref. [23] with constraint is, i.e. to solve a linear programming problem \( \min_{x \in \mathbb{R}^d} \langle \mathbf{w}, x \rangle - \mathbf{b} \) with \( x \in \mathbb{R}^d \) and \( \mathbf{w}, \mathbf{b} \in \mathbb{R}^d \). Obviously, one of the key points of SK algorithm is in Step 2, which one is a real vertex among \( m(\mathbb{C}) \). Let the vector \( x_i \) be the soft convex hull spanned by \( \mathbb{C} \) and \( x_i \) to illustrate this problem. For example, we can assume \( x_1 \) to \( x_2 \) be the soft convex hull spanned by \( \mathbb{C} \) and \( x_1 \) to \( x_2 \) be the soft convex hull spanned by \( \mathbb{C} \). Geometrically, the algorithm iterates until \( x_1 \) and \( x_2 \) become the \( e \)-optimal nearestpoints between the convex hulls of \( \mathbb{C} \) and \( \mathbb{C} \). Essentially, SK algorithm is to find the \( e \)-optimal nearest points between two convex hulls. Let \( \hat{X}_1 \) and \( \hat{X}_2 \) be the soft convex hull spanned by \( \mathbb{C} \) and \( \mathbb{C} \) with constraint \( \mu \), respectively. Denote \([1/\mu]\) as the minimal integer which is not less than \( 1/\mu \) and assume that \( X_1 \) and \( X_2 \) is linearly separable now. The soft SK algorithm in Ref. [23] is described as follows.

**Softened SK algorithm for soft convex hulls.**

- **Step 1:** Set the vector \( w_1 \) to any vector \( x \in \hat{X}_1 \) and \( w_2 \) to any vector \( x \in \hat{X}_2 \). Set the stopping criterion \( e \).
- **Step 2:** Find \( p_1, p_2, \ldots, p_{\lfloor 1/\mu \rfloor} \) in \( \mathbb{C} \) such that
  \[
  m(p_i) = \min_{i \in I} \{ m(p_i), \mu \}.
  \]
  \[
  m(p_i) = \min_{i \in I} \{ m(p_i), \mu \} \setminus \{ m(p_i), m(p_i), \ldots, m(p_i) \},
  \]
  where
  \[
  m(p_i) = \frac{x_i - w_1}{\| w_1 - w_2 \|}.
  \]
- **Step 3:** If \( x_i \in X_1 \), then set \( u_{new} = w_2 \) and compute
  \[
  u_{new} = w_1(1 - q) + x_i,
  \]
  where \( q = \min \left( 1, \frac{x_i - w_1}{\| w_1 - x_i \|^2} \right) \).
  Otherwise \( x_i \in X_2 \), then set \( u_{new} = w_1 \) and compute
  \[
  u_{new} = w_2(1 - q) + x_i,
  \]
  where \( q = \min \left( 1, \frac{x_i - w_1}{\| w_2 - x_i \|^2} \right) \).
- **Continue with Step 2.**

The following remarks are useful in understanding the convergence and computational cost of the soft SK algorithm.

- The only difference between SK algorithm and its softness is Step 2.
- Since the optimization problem \( \min \{ \langle x_i - w_1, w_1 - w_2 \rangle /\| w_1 - w_2 \|, x_i \in X_1 \} \) is linear, it is not difficult to find out that the vector \( x_i \) is the closest point in \( X_1 \) to the hyperplane associated with \( w_1 \) and \( w_2 \). Similarly, the vector \( x_i \) is the closest point in \( X_2 \) to the hyperplane associated with \( w_1 \) and \( w_2 \). Therefore, it can be confirmed that the soft algorithm captures the nature of SK algorithm and is then convergent. So, the above algorithm can find \( e \)-optimal nearest points between two disjoint soft convex hulls.
convex hulls. It is for these reasons that it is called soft SK algorithm.

- The hard soft algorithm can obviously be regarded as a special case of the soft one with \( \mu = 1 \). Intrinsically, their computational principle are the same. So, it can be confirmed that they will have almost the same computational cost. Therefore, it is potential in solving large-scale \( \nu \)-SVM problems.

From above, it is not difficult to find out such an interesting fact, i.e., the solution of a linear optimization problem defined on a soft convex set can be achieved by solving several such linear programming problems on the hard convex set and combining these solutions according to \( \mu \) as that in Eqs. (5) and (6). A direct consequence of this finding is: if an algorithm only described as follows \([12]\):

\[
\text{for simplicity as well as without loss of generality, the algorithms suggested for solving NPPs. In Ref. [10], this algorithm and its softness following Definitions 4.1, 4.2 and Theorem 4.1 can be found in Ref. [12] and they are also intuitively illustrated in Ref. [10].}
\]

**Definition 4.1.** Let \( Q \) be a convex and compact subset of \( R^m \). \( \eta(y) = \max_{x \in Q} \langle x, y \rangle \) is called the support function of \( Q \).

It can be shown that \( \eta(y) \) is a convex and continuous function in \( R^m \). Obviously, \( \{x : \langle x, y \rangle = \eta(y)\} \cap Q \neq \emptyset \).

**Definition 4.2.** Let \( s(y) \) be a function on \( R^m \). \( s(y) \) is called a contact function if \( s(y) \in \{x : \langle x, y \rangle = \eta(y)\} \cap Q \), \( y \neq 0 \) and \( s(0) \in Q \).

As a specific NPP, a minimal norm problem (MNP) is usually described as follows \([12]\): \n
**MNP:** Let \( Q \) be a convex set spanned by \( p_1, p_2, \ldots, p_{n_1} \). Given a convex and compact set \( Q \) in \( R^m \), find a point \( x^* \in Q \) such that \( \|x^*\| = \min_{x \in Q} \|x\| \).

**Gilbert’s algorithm for MNP.**

- **Step 1:** Set \( z_1 \in \tilde{Q} \). Set the stopping criterion \( \varepsilon \).
- **Step 2:** Find \( p_{i1}, p_{i2}, \ldots, p_{i(1/\mu)} \) in \( \{p_1, p_2, \ldots, p_{n_1}\} \) such that

\[
\langle p_{i1}, z_n \rangle = \min \{\langle p_1, z_n \rangle, \langle p_2, z_n \rangle, \ldots, \langle p_{n_1}, z_n \rangle\}\]

\[
\langle p_{ij}, z_n \rangle = \min \{\langle p_1, z_n \rangle, \langle p_2, z_n \rangle, \ldots, \langle p_{n_1}, z_n \rangle\} \langle p_{ij}, z_n \rangle, \langle p_{i1}, \ldots, p_{i(1/\mu)} \rangle, j = 2, 3, \ldots, [1/\mu].
\]

Let \( \tilde{s}(z_n) = \sum_{j=1}^{[1/\mu]} \mu p_{ij} + (1 - \mu([1/\mu] - 1)) p_{i(1/\mu)} \).

Let

\[
z_n = \begin{cases} 
\|z_n - \tilde{s}(z_n)\|^{-2} 
& \text{if } z_n - \tilde{s}(z_n) \neq 0, \\
(z_n - z_n - \tilde{s}(z_n)) 
& \text{if } z_n - \tilde{s}(z_n) = 0,
\end{cases}
\]

for \( n = n + 1 \), go to Step 2.

The following remarks are helpful to understand the nature of Gilbert’s algorithm.

- One of the solutions of linear programming problem \( \eta(y) = \max_{x \in Q} \langle x, y \rangle \) must be obtained in \( \{p_1, p_2, \ldots, p_{n_1}\} \). Therefore, \( s(-z) \) can be assigned to a certain \( p_i \) which satisfies \( \langle p_i, z \rangle = \min \{\langle p_1, z \rangle, \langle p_2, z \rangle, \ldots, \langle p_{n_1}, z \rangle\} \).

\[
z_{n+1} = z_n + z_n(s(-z_n) - z_n) \]

is actually the point on the line \( L(z_n; s(-z_n)) \) with minimum norm.

**Theorem 4.1.** The above Gilbert’s nearest points algorithm converges to the unique solution of MNP.

**Proof.** See Ref. \([12]\). \( \square \)

Let \( Q \) be a soft convex hull spanned by \( p_1, p_2, \ldots, p_{n_1} \) with constraint \( \mu \). Using the interesting idea in Section 2, the above Gilbert’s algorithm is softened as follows.

**Softened Gilbert’s algorithm for MNP.**

- **Step 1:** Set \( z_1 \in \tilde{Q} \). Set the stopping criterion \( \varepsilon \).
- **Step 2:** Find \( p_{i1}, p_{i2}, \ldots, p_{i(1/\mu)} \) in \( \{p_1, p_2, \ldots, p_{n_1}\} \) such that

\[
\langle p_{i1}, z_n \rangle = \min \{\langle p_1, z_n \rangle, \langle p_2, z_n \rangle, \ldots, \langle p_{n_1}, z_n \rangle\}\]

\[
\langle p_{ij}, z_n \rangle = \min \{\langle p_1, z_n \rangle, \langle p_2, z_n \rangle, \ldots, \langle p_{n_1}, z_n \rangle\} \langle p_{ij}, z_n \rangle, \langle p_{i1}, \ldots, p_{i(1/\mu)} \rangle, j = 2, 3, \ldots, [1/\mu].
\]

Let \( \tilde{s}(z_n) = \sum_{j=1}^{[1/\mu]} \mu p_{ij} + (1 - \mu([1/\mu] - 1)) p_{i(1/\mu)} \).

Let

\[
z_n = \begin{cases} 
\|z_n - \tilde{s}(z_n)\|^{-2} 
& \text{if } z_n - \tilde{s}(z_n) \neq 0, \\
(z_n - z_n - \tilde{s}(z_n)) 
& \text{if } z_n - \tilde{s}(z_n) = 0,
\end{cases}
\]

for \( n = n + 1 \), go to Step 2.

The following remarks are useful in understanding the convergence and computational cost of the soft Gilbert’s algorithm.

- From Step 2, it is easy to know \( \langle \tilde{s}(z_n), z_n \rangle = \min_{p \in \tilde{Q}} \langle p, z_n \rangle \).

Hence \( \tilde{s}(z_n) \in \{x : \langle x, z_n \rangle = \eta(z_n)\} \cap \tilde{Q} \), i.e., Step 2 in both hard and soft algorithms accomplishes the same tasks. The only difference is that the former is for the hard convex hulls while the latter is for the soft. Therefore, the soft algorithm captures the nature of Gilbert’s algorithm and is then convergent. So, the above algorithm can find the nearest points between two disjoint soft convex hulls. It is for these reasons that we call it soft Gilbert’s algorithm.

- The soft Gilbert’s algorithm can obviously be regarded as a special case of the hard one with \( \mu = 1 \). In a single iteration step, the soft algorithm is to find \( \min \{p_{i1}, z_n\} \) \( [1/\mu] \) times while the hard is to find \( \min \{p_{i1}, z_n\} \) only once. Intrinsically, their iteration principle is identical and the computational cost in each loop is in the same order. So, it can be confirmed...
that they will have almost the same total computational cost.

Using the same stopping criterion as that in Refs. [10, 16], the above soft Gilbert’s algorithm can be adapted to achieve an ϵ-optimal ν-SVM classifier. Note that the essential idea in softening SK and Gilbert’s algorithm is the same.

5. MDM algorithm and its softness

Unlike SK and Gilbert’s algorithm, MDM algorithm fundamentally uses a combination of several training points as each iterative point in its basic operation. Since there always exists a training point which satisfies the value of objective function severely, the main step in MDM algorithms is to eliminate such a point or lessen its influence from the representation of each iterative point. Based on this idea, as pointed out in Ref. [10], MDM algorithm usually works faster than Gilbert’s algorithm, especially in the end stages. In this section, the important idea in the previous sections will be used to soften MDM algorithms. But there exist extra difficulties in softening MDM algorithms. To introduce MDM for SVMs with L2-norm, we first assume \(\tilde{X}_1 \cap \tilde{X}_2 = \emptyset\).

**MDM algorithm for linearly separable SVMs.**

- **Step 1:** Set \(w_1 = \sum_{i=1}^{l_1} \gamma_i p_i\) and \(w_2 = \sum_{i=1}^{l_2} \beta_i q_i\), where \(1 \geq \gamma_i \geq 0, 1 \geq \beta_i \geq 0, \sum_{i=1}^{l_1} \gamma_i = 1\) and \(\sum_{i=1}^{l_2} \beta_i = 1\). Set the stopping criterion \(\epsilon\).

- **Step 2:** Find a vector \(x_t\) closest to the hyperplane associated with \(w_1\) and \(w_2\) as \(x_t = \arg \min (m(x_t), i \in I_1 \cup I_2)\), where

\[
m(x_t) = \begin{cases} 
\frac{(x_t - w_2, w_1 - w_2)}{\|w_1 - w_2\|} & \text{for } i \in I_1, \\
\frac{(x_t - w_1, w_2 - w_1)}{\|w_1 - w_2\|} & \text{for } i \in I_2.
\end{cases}
\]

If the ϵ-optimality condition \(\|w_1 - w_2\| - m(x_t) < \epsilon\) holds, then the vector \(w_1 - w_2\) and \(b = \frac{1}{2}(\|w_1\|^2 - \|w_2\|^2)\) define the ϵ-solution. Otherwise, let \(z = w_1 - w_2\) and go to Step 3.

- **Step 3:** If \(x_t \in X_1\), find an index \(l_{\text{min}} (l_1 \geq l_{\text{min}} \geq 1)\) such that

\[
\langle -z, p_{l_{\text{min}}} \rangle = \min \{\langle -z, p_i \rangle : \gamma_i > 0, i = 1, 2, \ldots, l_1\}. \tag{7}
\]

Let \(d = s(-z) - q_{l_{\text{min}}} \), \(\hat{z} = z + \gamma_{l_{\text{min}}} d\). Then let \(w_{1}^{\text{new}} = w_1, w_{2}^{\text{new}} = z^{\text{new}} + w_{2}^{\text{new}}\).

Otherwise, find an index \(l_{\text{min}} (l_2 \geq l_{\text{min}} \geq 1)\) such that

\[
\langle -z, q_{l_{\text{min}}} \rangle = \min \{\langle -z, q_i \rangle : \beta_i > 0, i = 1, 2, \ldots, l_2\}. \tag{8}
\]

Let \(d = s(-z) - q_{l_{\text{min}}} \), \(\hat{z} = z + \beta_{l_{\text{min}}} d\). Then let \(w_{1}^{\text{new}} = w_1, w_{2}^{\text{new}} = z^{\text{new}} + w_{1}^{\text{new}}\).

Continue with Step 2.

The MDM algorithm for MNP is intuitively illustrated in Ref. [10]. As the above algorithm uses MDM for MNP as a main step, it can be easily adapted to solve SVMs with L2-norm. To further understand the nature of MDM algorithms, the following remarks are given:

- For efficiency of representation as well as algorithm performance, it is a good idea to eliminate \(t\) satisfying Eq. (7). This is accomplished by letting \(\hat{z} = z + l_{\text{min}} d = z + \gamma_{l_{\text{min}}} s(-z) - \gamma_{l_{\text{min}}} p_{l_{\text{min}}} \) in Step 3.
- Since \(s(-z) \leq \tilde{X}_1, \hat{z}\) is still in \(\tilde{X}_1\).
- Let \(\phi(\hat{z}) = \|s + \epsilon d\|^2\). Since \(\phi'(0) = 2(z, d) = 2(z, s(-z) - 2(z, p_{l_{\text{min}}}) < 0\), it is easy to see moving along \(d\) will decrease \(\|z\|^2\).

\[\hat{z} = z + \gamma_{l_{\text{min}}} d\] can be easily obtained by substituting \(s(-z)\) for \(p_{l_{\text{min}}}\) in the representation of \(z\).

- Since \(z^{\text{new}}\) is the point of minimum norm on the line segment joining \(z\) and \(\hat{z}\), the coefficient of \(p_{l_{\text{min}}}\) and \(\|w_1 - w_2\|\) are decreased simultaneously.

- Geometrically, the algorithm iterates until \(w_1\) and \(w_2\) become the ϵ-optimal nearest points between the convex hulls of \(X_1\) and \(X_2\). Similarly, the above MDM algorithm is to find the ϵ-optimal nearest points between two convex hulls.

To introduce the soft MDM algorithm, we assume that \(\tilde{X}_1 \cap \tilde{X}_2 = \emptyset\).

**Softened MDM algorithm for linear ν-SVMs.**

- **Step 1:** Set \(w_1 = \sum_{i=1}^{l_1} \gamma_i p_i\) and \(w_2 = \sum_{i=1}^{l_2} \beta_i q_i\), where \(\mu \geq \gamma_i \geq 0, \mu \geq \beta_i \geq 0, \sum_{i=1}^{l_1} \gamma_i = 1\) and \(\sum_{i=1}^{l_2} \beta_i = 1\). Set the stopping criterion \(\epsilon\).

- **Step 2:** Find \(p_{l_1}, p_{l_2}, \ldots, p_{l_{[1/\mu]}}\) in \(X_1 = \{p_i, i \in I_1\}\) such that

\[
m(p_{l_1}) = \min \{m(p_i), i \in I_1\}.
\]

\[
m(p_{l_2}) = \min \{m(p_i), i \in I_1\}\backslash \{m(p_{l_1}), m(p_{l_2}), \ldots, m(p_{l_{[1/\mu]}})\}, \quad j = 2, 3, \ldots, [1/\mu], \quad m(p_{l_{j-1}}) = \langle -z, q_{l_{j-1}} \rangle = \min \{m(q_i), i \in I_2\}.
\]

Find \(q_{l_1}, q_{l_2}, \ldots, q_{l_{[1/\mu]}}\) in \(X_2 = \{q_i, i \in I_2\}\). Let \(m(q_{l_1}) = \langle -z, q_{l_1} \rangle = \min \{m(q_i), i \in I_2\}\).

Let \(x_t^2 = \sum_{j=1}^{[1/\mu]} m_{q_{l_j}} + (1 - \mu([1/\mu] - 1)) q_{l_{[1/\mu]}}\).

Find \(x_t^2 = \arg \min (m(x_t^2, m(x_t^2), \ldots, m(x_t^2))\), \(j = 2, 3, \ldots, [1/\mu]\), \(m(q_{l_{[1/\mu]}}) = \langle q_{l_{[1/\mu]}}, w_1 - w_2\rangle\|w_1 - w_2\|^2\).

If the ϵ-optimality condition \(\|w_1 - w_2\| - m(x_t^2) < \epsilon\) holds, then the vector \(w_1 - w_2\) and \(b = \frac{1}{2}(\|w_1\|^2 - \|w_2\|^2)\) define the ϵ-solution. Let \(z = w_1 - w_2\) and go to Step 3.

- **Step 3:** If \(x_t \in X_1\), find an index \(l_{\text{min}} (l_1 \geq l_{\text{min}} \geq 1)\) such that

\[
\langle -z, p_{l_{\text{min}}} \rangle = \min \{\langle -z, p_i \rangle : \gamma_i > 0, i = 1, 2, \ldots, l_1\}. \tag{9}
\]

Find \(p_{l_1}, p_{l_2}, \ldots, p_{l_{[1/\mu]}}\) in \(X_1, p_{l_{[1/\mu]}}\) such that

\[
\langle p_{l_1}, z \rangle = \min \{\langle p_{l_1}, z \rangle, \langle p_{l_2}, z \rangle, \ldots, \langle p_{l_{[1/\mu]}}, z \rangle\}\backslash \{\langle p_{l_1}, z \rangle, \langle p_{l_2}, z \rangle, \ldots, \langle p_{l_{[1/\mu]}}, z \rangle\}, \quad j = 2, 3, \ldots, [1/\mu]. \quad \text{Let } \hat{s}(z) = \sum_{j=1}^{[1/\mu]} \mu p_{l_j} + (1 - \mu([1/\mu] - 1)) p_{l_{[1/\mu]}}\).
Let \( d = \hat{s}(z) - p_{\text{min}} \),
\[
\tilde{z} = z + \gamma_{t_{\text{min}}} d - \left( \sum_{j=1}^{[1/\mu]-1} \gamma_j p_{ij} + \gamma_{[1/\mu]} p_{[1/\mu]} \right) \\
+ \left( \sum_{j=1}^{[1/\mu]-1} \gamma_j \hat{s}(z) + \gamma_{[1/\mu]} \hat{s}(z) \right).
\]

Let \( z_{\text{new}} \) be the point of minimum norm on the line segment joining \( z \) and \( \tilde{z} \).

Then let \( w_{2}^{\text{new}} = w_{2} \), \( w_{1}^{\text{new}} = z_{\text{new}} + w_{2}^{\text{new}} \).

Otherwise, the above same adaption is performed on \( w_{2}^{\text{new}} \).

Then let \( w_{1}^{\text{new}} = w_{1} \), \( w_{2}^{\text{new}} = -z_{\text{new}} + w_{1}^{\text{new}} \).

Continue with Step 2.

The following remarks are useful in understanding the convergence and computational cost of soft MDM algorithms.

- Similarly, one of our main aims in soft MDM algorithms is to eliminate \( t \) satisfying Eq. (9).
- The differences between MDM algorithm and its softness may be steps and soft MDM algorithms are in Steps 2 and 3. However, it is easy to see that the spirit of Step 2 is the same as that in soft SK algorithms.
- Since the NPP is to be solved in the soft hulls, it is natural to use \( \delta(z) \) in Step 2.
- Obviously, \( \delta(z) \) is a combination of some vertexes. So, there may be some vertexes existing in the representation of both \( z \) and \( \delta(z) \). The coefficients of such vertexes in \( z + \gamma_{t_{\text{min}}} d \) may become greater than \( \mu \) after summing. Therefore, we cannot guarantee that \( z + \gamma_{t_{\text{min}}} d \) is contained in the soft convex hull, and this is the main extra difficulty in softening MDM algorithms.
- In the hard convex cases, \( s(z) \) is one of the vertexes and \( z + \gamma_{t_{\text{min}}} d \) can be obtained by substituting \( s(z) \) for \( p_{\text{min}} \) in the representation of \( z \). Since \( \delta(z) \) is a combination of some vertexes, it is natural to replace all such vertexes with \( \delta(z) \) in the expression of \( z \), and this is accomplished by (10).
- Mathematical analysis can prove that \( \tilde{z} \) satisfying Eq. (10) is contained in the soft convex hull (see the following Theorem 5.1). In addition, moving along \( \tilde{z} - z \) will decrease \( \|z\| \) (see the following Theorem 5.2). With the above two desired properties, it can be said that soft MDM algorithms inherit the main advantages of usual MDM algorithms.
- Since \( z_{\text{new}} \) is the point of minimum norm on the line segment joining \( z \) and \( \tilde{z} \), the coefficient of \( p_{\text{min}} \) and \( \|w_{1} - w_{2}\| \) are decreased simultaneously. Therefore, it can be confirmed that the soft algorithm captures the nature of MDM algorithm and is then convergent. So, the above algorithm can find \( \varepsilon \)-optimal nearest points between two disjoint soft convex hulls. It is for these reasons that we call it soft MDM algorithm.
- The hard algorithm can obviously be regarded as a special case of the soft one with \( \mu = 1 \). In a single iteration step, the soft algorithm is to find \( m(x_{j}) \) \([1/\mu]\) times while the hard is to find \( m(x_{j}) \) only once. Intrinsically, their iteration principle is identical and the computational cost in each loop is in the same order. So, it can be confirmed that they will have almost the same total computational cost.

**Theorem 5.1.** In Step 3, \( \tilde{z} = z + t_{\text{min}} d - \left( \sum_{j=1}^{[1/\mu]-1} \gamma_j p_{ij} + \gamma_{[1/\mu]} p_{[1/\mu]} \right) + \left( \sum_{j=1}^{[1/\mu]-1} \gamma_j \hat{s}(z) + \gamma_{[1/\mu]} \hat{s}(z) \right) \) is contained in soft convex hulls.

**Proof.** Without loss of generality, we can assume that \( z = \sum_{i=1}^{l_{1}} \gamma_{i} p_{i} \), \( p_{\text{min}} = p_{i_{1}} \), and \( \hat{s}(z) = \sum_{i=3}^{l_{1}} \lambda_{i} p_{i} \), where \( 0 \leq \gamma_{i} \leq \mu \) \((i = 1, \ldots, l_{1})\), \( 0 < \lambda_{j} \leq \mu \) \((j = 3, \ldots, l_{1} - 1)\), \( \sum_{i=1}^{l_{1}} \gamma_{i} = 1 \) and \( \sum_{i=3}^{l_{1}} \lambda_{i} = 1 \). Then, \( \hat{s}(z) \) can be reformulated as
\[
\tilde{z} = \gamma_{1} p_{1} + \gamma_{2} p_{2} + \sum_{i=3}^{l_{1}} \lambda_{i} p_{i}.
\]

If we introduce \( \hat{s}(z) = \sum_{i=3}^{l_{1}} \lambda_{i} p_{i} \) into expression (11), it is not difficult to verify that \( \tilde{z} \) is contained in soft convex hulls. \( \Box \)

**Theorem 5.2.** In Step 3, let \( \phi'(z) = \|z + \lambda t_{\text{min}} d - \left( \sum_{j=1}^{[1/\mu]-1} \gamma_j p_{ij} + \gamma_{[1/\mu]} p_{[1/\mu]} \right) + \left( \sum_{j=1}^{[1/\mu]-1} \gamma_j \hat{s}(z) + \gamma_{[1/\mu]} \hat{s}(z) \right) \|^{2} \). Then \( \phi'(z) < 0 \).

**Proof.** \( \phi'(z) = 2t_{\text{min}}\langle z, d \rangle - 2\|z\| + \sum_{j=1}^{[1/\mu]-1} \gamma_j \langle p_{ij}, d \rangle + \sum_{j=1}^{[1/\mu]-1} \gamma_j \langle \hat{s}(z), d \rangle \) \) + \( \langle z, \sum_{j=1}^{[1/\mu]-1} \gamma_j p_{ij} + \gamma_{[1/\mu]} p_{[1/\mu]} \rangle \) \( + \langle z, \sum_{j=1}^{[1/\mu]-1} \gamma_j \hat{s}(z) + \gamma_{[1/\mu]} \hat{s}(z) \rangle \).

Based on the definition of \( \hat{s} \) and \( p_{\text{min}} \), \( \langle z, p_{ij} \rangle \geq \langle z, \hat{s}(z) \rangle \) for each \( j \).

Therefore, \( \phi'(z) < 0 \).

Obviously, Theorem 5.2 indicates that moving along \( \tilde{z} - z \) will decrease \( \|z\| \).

At the end of this section, we indicate that in Step 3 of the soft MDM algorithm, \( \hat{s}(z) \) is solved using the important soft idea in this paper. But to ensure several desirable properties of \( z_{\text{new}} \), some other techniques are used. Similarly as that in MDM algorithms in Ref. [10] and SK algorithms in Ref. [16], the proposed soft MDM algorithms can be expressed in terms of dot products. So, kernel technique can be easily incorporated to solve nonlinear \( \nu \)-SVMs. Since the principle is completely the same as that in Refs. [10,16], the details are omitted. It should be also pointed out that there already exist several algorithms for large-scale \( \nu \)-SVMs [25]. In Ref. [25], the authors modified decomposition methods and their computational principle is intrinsically the same as that in Refs. [6,7]. Based on the status of geometric methods in solving SVMs, this paper can be viewed as another important contribution to solve SVMs with \( L_{1} \)-norm.

6. Experiments

In this section, some experiments are conducted to verify the theoretical analysis of soft MDM algorithms. Our soft MDM
Table 1
Quantitative comparison of the algorithms on selected benchmark data sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>Algorithm</th>
<th>Parameters</th>
<th>Precision $\varepsilon$</th>
<th>Class. error (train) (%)</th>
<th>Class. error (test) (%)</th>
<th>Kernel evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thyroid</td>
<td>S–K</td>
<td>$\sigma = 2$</td>
<td>0.01</td>
<td>0.56</td>
<td>4.27</td>
<td>$1.41 \times 10^4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C = 10$</td>
<td>0.005</td>
<td>0.54</td>
<td>4.29</td>
<td>$3.05 \times 10^4$</td>
</tr>
<tr>
<td></td>
<td>MDM</td>
<td>$\sigma = 0.225$</td>
<td>0.01</td>
<td>0.56</td>
<td>4.36</td>
<td>$1.80 \times 10^4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C = 50$</td>
<td>0.005</td>
<td>0.56</td>
<td>4.48</td>
<td>$5.50 \times 10^4$</td>
</tr>
<tr>
<td></td>
<td>Soft-MDM</td>
<td>$\sigma = 0.225$</td>
<td>0.001</td>
<td>0.56</td>
<td>4.48</td>
<td>$5.08 \times 10^4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\mu = 0.2$</td>
<td>0.005</td>
<td>0.09</td>
<td>4.47</td>
<td>$7.93 \times 10^4$</td>
</tr>
<tr>
<td>Heart</td>
<td>S–K</td>
<td>$\sigma = 120$</td>
<td>0.01</td>
<td>23.18</td>
<td>25.12</td>
<td>$5.22 \times 10^5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C = 3$</td>
<td>0.005</td>
<td>23.37</td>
<td>25.44</td>
<td>$1.02 \times 10^6$</td>
</tr>
<tr>
<td></td>
<td>MDM</td>
<td>$\sigma = 16$</td>
<td>0.01</td>
<td>9.92</td>
<td>16.68</td>
<td>$9.03 \times 10^5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C = 0.16$</td>
<td>0.005</td>
<td>9.95</td>
<td>16.65</td>
<td>$9.82 \times 10^5$</td>
</tr>
<tr>
<td></td>
<td>Soft-MDM</td>
<td>$\sigma = 16$</td>
<td>0.01</td>
<td>9.94</td>
<td>16.67</td>
<td>$1.15 \times 10^5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\mu = 0.16$</td>
<td>0.005</td>
<td>9.21</td>
<td>19.36</td>
<td>$1.82 \times 10^5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>9.24</td>
<td>19.21</td>
<td>$2.01 \times 10^5$</td>
</tr>
<tr>
<td>Diabetis</td>
<td>S–K</td>
<td>$\sigma = 20$</td>
<td>0.01</td>
<td>24.79</td>
<td>27.36</td>
<td>$3.62 \times 10^6$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C = 10$</td>
<td>0.005</td>
<td>21.40</td>
<td>24.57</td>
<td>$1.38 \times 10^7$</td>
</tr>
<tr>
<td></td>
<td>MDM</td>
<td>$\sigma = 600$</td>
<td>0.01</td>
<td>21.98</td>
<td>23.65</td>
<td>$3.31 \times 10^7$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C = 9$</td>
<td>0.005</td>
<td>21.50</td>
<td>23.31</td>
<td>$5.01 \times 10^5$</td>
</tr>
<tr>
<td></td>
<td>Soft-MDM</td>
<td>$\sigma = 600$</td>
<td>0.01</td>
<td>25.38</td>
<td>26.30</td>
<td>$7.44 \times 10^5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\mu = 0.035$</td>
<td>0.005</td>
<td>25.40</td>
<td>26.28</td>
<td>$7.32 \times 10^5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
<td>25.39</td>
<td>26.35</td>
<td>$7.63 \times 10^5$</td>
</tr>
</tbody>
</table>

Fig. 1. The relation between the averaged number of iteration and margin on Thyroid. The solid curve is for the soft MDM algorithm ($\sigma = 0.225$, $\mu = 0.2$ and $\varepsilon = 0.01$), while the dashed is for the hard ($\sigma = 0.225$ and $C = 50$ and $\varepsilon = 0.01$).

algorithm is implemented by modifying the kernel SK algorithm of Ref. [16] in Matlab.

As a potential algorithm for solving large-scale problems, the proposed soft MDM algorithm is expected to have comparisons against other geometric methods in terms of computational cost. Obviously, the total computational cost of soft MDM algorithms depends on the convergence rate and computational cost in each step of the loop. We have tested soft MDM algorithms on many data sets and find out that soft MDM is competitive to the hard ones and works faster than SK algorithms. However, it is difficult to precisely analyze the convergence rate of soft MDM algorithms. Compared with SK and Gilbert’s algorithms, MDM algorithms adopt a different iteration strategy. This strategy makes MDM work faster than other algorithms. Since the similar iteration strategy is employed in soft MDM, we confirm that the
convergence rate of soft MDM should be satisfactory. On the other hand, as mentioned in the remarks of Section 5, the computational cost in each single step is in the same order. In a word, the advantage of soft MDM algorithms is completely established on the basis of the hard MDM methods. Note that the total computational cost in experiments is directly affected by the number of kernel evaluation. The reason is that the kernel must be evaluated once the samples are concerned in computation, and this number has been regarded as a benchmark index to judge the speed of kernel algorithms in Ref. [16]. Based on the above discussions, we demonstrate the performance of the algorithms using the number of kernel evaluation and iteration. To make the comparisons against MDM algorithms in Ref. [10] fair, several benchmark data sets are selected.

**Example 1.** These experiments are conducted by employing the training data sets, test data sets and the cross validation strategy provided in http://mlg.anu.edu.au/~raetsch/. The quantitative comparison against SK and MDM algorithms in terms of the number of averaged kernel evaluations is described in Table 1. In all the experiments, RBF kernel $e^{-\|x-y\|^2/2\sigma^2}$ is used.

This example illustrates that the proposed soft MDM algorithm is competitive to MDM algorithm and a little better than SK algorithm of Ref. [16] in some cases.

**Example 2.** In this example, the “margin” is referred to the “temporary margin” in each loop of the algorithms, i.e. the half distance between $w_1$ to $w_2$ in each loop. The relation between the averaged margin and the averaged step number of iteration is also an interesting topic. The comparison between MDM and soft MDM on data set Thyroid can be seen in Fig. 1.

This example illustrates that it needs less number of iterations to achieve the maximal margin by using our soft algorithms.

7. Conclusions

In this paper, we present a general technique which can generalize almost all the available $L_2$-norm SVM algorithms to the corresponding soft version by using $L_1$-norm. Theoretical analysis and experimental results have indicated clearly that all the soft $L_1$-norm SVMs algorithms constructed with the proposed technique have the same properties of convergence and almost the identical cost of computation as that of the corresponding hard algorithms. More specific and practical soft $L_1$-norm SVMs algorithms for real-world large-scale applications and the geometric methods for nonstandard SVMs in Refs. [26,27] will be investigated in the future studies.

References

About the Author—QING TAO received the M.S. degree in mathematics from Southwest Normal University, Chongqing, China, in 1989, and the Ph.D. from the University of Science and Technology of China, Hefei, China, in 1999. From June 1999 to 2001, he was a Postdoctoral Fellow with the University of Science and Technology of China. From June 2001 to 2003, he was a Postdoctoral Fellow with the Institute of Automation, Chinese Academy of Sciences, China. Currently he is a Professor with the Institute of Automation, Chinese Academy of Sciences, and also a Professor with the New Star Research Institute of Applied Technology, Hefei. His research interests are in the area of applied mathematics, neural networks, statistical learning theory, and SVM.

About the Author—GAO-WEI WU was born in 1975 in Jilin City, Jilin province, PR China. He received M.S. degree in Circuit and System from Institute of Semiconductor in 2000. Currently he is a Ph.D. candidate in Institute of Automation, Chinese Academy of Sciences, PR China. His research focuses on neural networks, statistical learning theory and SVM.

About the Author—JUE WANG is a professor in Institute of Automation, Chinese Academy of Sciences, and a IEEE senior member. His research interests include ANN, GA, multi-agent system, machine learning and data mining.