Comput Optim Appl (2013) 56:599–617
DOI 10.1007/s10589-013-9571-6

New smoothing SVM algorithm with tight error bound and efficient reduced techniques

Shuisheng Zhou · Jiangtao Cui · Feng Ye · Hongwei Liu · Qiang Zhu

Received: 8 March 2012 / Published online: 6 June 2013

Abstract The quadratically convergent algorithms for training SVM with smoothing methods are discussed in this paper. By smoothing the objective function of an SVM formulation, Lee and Mangasarian [Comput. Optim. Appl. 20(1):5-22, 2001] presented one such algorithm called SSVM and proved that the error bound between the new smooth problem and the original one was $O(\frac{1}{p})$ for large positive smoothing parameter p. We derive a new method by smoothing the optimality conditions of the SVM formulation, and we prove that the error bound is $O(\frac{1}{p^2})$, which is better than Lee and Mangasarian's result. Based on SMW identity and updating Hessian iteratively, some boosting skills are proposed to solve Newton equation with lower computational complexity for reduced smooth SVM algorithms. Many experimental results show that the proposed smoothing method has the same accuracy as SSVM, whose error bound is also tightened to $O(\frac{1}{p^2})$ in this paper, and the proposed boosting skills are efficient for solving large-scale problems by RSVM.

Keywords SSVM · Smoothing algorithm · Error bound · Reduced methods

1 Introduction

Based on the Vapnik and Chervonenkis' structural risk minimization principle [28, 29], Support Vector Machine (SVM) has emerged as a state-of-the-art tool for classification and regression. SVM solves a convex quadratic programming problem, and its solution is not only explicitly defined but also sparse. SVM has been widely used in

S. Zhou (✉) · F. Ye · H. Liu · Q. Zhu
School of Science, Xidian University, Xi'an 710071, China
e-mail: sszhou@mail.xidian.edu.cn

J. Cui
School of Computer Science and Technology, Xidian University, Xi'an 710071, China

many application areas [28, 29], such as character identification, disease diagnoses, time series prediction, etc.

Given a training dataset $\{(x_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \{-1, 1\}$, SVM is to solve the following formulation (see [3, 13] etc.):

$$\min_{\mathbf{w}\in\mathbb{H}, b\in\mathbb{R}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{\sigma}\sum_{i=1}^m \Big(\max\big\{0, 1 - y_i\big(\langle \mathbf{w}, \mathbf{x_i}\rangle_{\mathbb{H}} + \mathbf{b}\big)\big\}\Big)^\sigma, \tag{1}$$

or the formulation with a regularization term $\frac{1}{2}b^2$ of bias b added to the objective function [16, 17, 22, 23]:

$$\min_{\mathbf{w}\in\mathbb{H}, b\in\mathbb{R}} \frac{1}{2}\big(\|\mathbf{w}\|^2 + b^2\big) + \frac{C}{\sigma}\sum_{i=1}^m \Big(\max\big\{0, 1 - y_i\big(\langle \mathbf{w}, \mathbf{x_i}\rangle_{\mathbb{H}} + \mathbf{b}\big)\big\}\Big)^\sigma, \tag{2}$$

with $\sigma = 1$ or 2, tradeoff parameter $C > 0$ and $\mathbf{x}_i := \phi(x_i) \in \mathbb{H}$ for nonlinear classification problems, where $\phi(\cdot)$ maps x_i to a high dimensional, even infinite dimensional, feature space. Owing to the reproduced kernel theory, $\mathbb{H}$ is a reproducing kernel Hilbert space [13, 26] associated with a kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ satisfying $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle_{\mathbb{H}}$. The learning result $\mathbf{w}$ can be represented by a linear combination of the kernel functions:

$$\mathbf{w} = \sum_{i=1}^m \beta_i k(x_i, \cdot), \tag{3}$$

and it is not needed to use the high dimensional map $\phi(\cdot)$ explicitly. By plugging (3) into (1) or (2), the optimal coefficients $\beta := [\beta_1, \beta_2, \ldots, \beta_m]^T \in \mathbb{R}^m$ and b can be solved and the classification surface $\theta(x) := \sum_i \beta_i k(x_i, x) + b = 0$ is obtained.

Many studies have been done on the dual of problem (1) or (2) [1, 7, 14, 22, 23, 25, 30]. For example, the well-known methods, SMO [1, 14, 25] and SVM^{light} [11, 12], are decomposition methods to solve the dual of problem (1). Successive over-relaxation (SOR) [22] and Lagrangian Support Vector Machine (LSVM) algorithms [23] are simple iterative algorithms induced by the KKT conditions of the dual of problem (2), etc. These algorithms do work efficiently in experiments, although their convergence rate is low theoretically (at most linear convergence rate [18, 19, 22, 23]).

Lee and Mangasarian [17] proposed a quadratic training algorithm called SSVM (Smooth SVM) for linear problem, in which the objective function of problem (2) with $\sigma = 2$ is smoothed by the exponential penalty function (Eq. (6) following), and its nonlinear version is based on the Generalized Support Vector Machine [21] listed as the following Eq. (5). However, SSVM cannot be used to solve large-scale nonlinear classification problems because its computational complexity for solving Newton equation is about $O(m^3)$ per iteration (SMW identity (Eq. (21)) can greatly reduce the computing cost of the linear classification problem).

In [6], the kernel matrix K is precalculated and decomposed as $K \approx UU^T$ where $U \in \mathbb{R}^{m\times\widetilde{m}}$ is a full column rank matrix, and a semi-smooth SVM with quadratic convergence rate is proposed. Further, another semi-smooth Newton SVM is proposed with many better experimental results reported in [31]. These algorithms can reach $O(m\widetilde{m}^2)$ computational cost, and will be a good choice if $\widetilde{m} \ll m$. However, these algorithms require the whole kernel matrix to fit in memory, hence they are not efficient for large-scale problems.

Some low-rank methods without the whole kernel matrix precalculated have been studied. In RSVM [16], a reduced set W is selected randomly from the index set $\{1, 2, \dots, m\}$ with $\overline{m} := |W| \le 0.1m$ and (3) is replaced by a compact representation

$$\mathbf{w} = \sum_{i \in W} \beta_i k(x_i, \cdot). \tag{4}$$

Although the reduced set W is a randomly selected small portion of the whole set, many experiments in [16] show that RSVM works extremely well and the accuracy is quite insensitive to the choice of W. Its computational complexity is about $O(m\overline{m}^2)$ per iteration without the whole kernel matrix precalculated. Lin and Lin [20] discuss many kinds of implementations of RSVM in detail and conclude that the reduced methods are an appealing alternate for large-scale problems with a little lower accuracy. Lee and Huang [15] study the RSVM by Nyström approximation theoretically and provide a good understanding of the reduced kernel techniques.

In [13], another low-rank method is proposed to reduce the training complexity in a primal algorithm called PSVM (primal SVM), where the subset W in (4) is well-chosen iteratively to represent $\mathbf{w}$ as (4), and the resulting optimization problem is solved by the Newton method. Its computational complexity is at most $O(md_{max}^2)$ per iteration, where d_{max} is a specified maximum size of subset W. Compared to RSVM, PSVM may not be a better choice for common users because it needs a complicated sub-procedure to maintain W and the corresponding Hessian matrix iteratively.

There are also some efficient SVM solvers proposed in primal space recently. Fletcher and Zanghirati [8] proposed a new model for SVM problem in primal without introducing the penalization parameter C, and gave a new efficient SQP-like algorithm. They observed that the algorithm terminated within a finite number of iterations, but had not proven that. `LapSVM` [24] is a kind of learning algorithm for semi-supervised classification, and `Pegasos` [27] is designed for large datasets by stochastic sub-gradient descent methods.

In this paper, we focus on quadratic convergence algorithms for training SVM in primal space with the smoothing method, and the RSVM scheme is considered to reduce the complexity for large-scale problems. We generalize the results in [16, 17] to derive a new smoothing technique and prove that the error bound between the new smooth problem and the original one is $O(\frac{1}{p^2})$, which is stronger than that of SSVM in [17], where the error bound given is $O(\frac{1}{p})$. We also tighten the error bound of SSVM to $O(\frac{1}{p^2})$ in this paper. Some efficient reduced techniques, such as updating the Hessian matrix iteratively and solving Newton equation by SMW identity, are discussed in the paper to accelerate the algorithms for large-scale problems.

The rest of the paper is organized as follows. Lee and Mangasarian's SSVM is restated in Sect. 2. We derive a new smoothing algorithm called NSSVM and obtain its error bound and convergence rate in Sect. 3. We further summarize the different kernel versions for these smoothing algorithms in Sect. 4, discuss some boosting techniques to reduce its computational complexity in Sect. 5, and tighten the error bound of SSVM in Sect. 6. Experimental results are presented in Sect. 7 to illustrate that the proposed algorithm is comparable to SSVM on generalization errors and training times. Section 8 concludes this paper.

2 SSVM Algorithm

In this section, we restate the Smooth Support Vector Machine (SSVM) proposed by Lee and Mangasarian in [17] and focus on the nonlinear classification problem. The corresponding problem can be rewritten as follows

$$\min_z f(z) := \frac{1}{2} z^T z + \frac{C}{2} \sum_{i=1}^{m} \max\{r_i, 0\}^2, \tag{5}$$

where $z := [\beta^T, b]^T$, $r_i := 1 - H_i z$ with $H_i := y_i[K_i \;\; 1] \in \mathbb{R}^{m+1}$ for nonlinear classification (K_i is the ith row of the kernel matrix K and H_i is the ith row of matrix H). With the solution of (5), the classification surface is $\theta(x) := [k(x_1, x), k(x_2, x), \ldots, k(x_m, x) \;\; 1]z^* = 0$.

In [17], the exponential penalty function $\psi_p(t) := t + \frac{1}{p}\log(1 + \exp(-pt))$, $p > 0$, is applied to smooth the second part of the objective function in (5). Here the following *new* equivalent form of $\psi_p(t)$ is applied

$$\psi_p(t) := \max\{t, 0\} + \frac{1}{p} \log\bigl(1 + \exp\bigl(-p|t|\bigr)\bigr), \tag{6}$$

and then

$$\psi_p'(t) = \frac{\min\{1, \exp(pt)\}}{1 + \exp(-p|t|)}, \qquad \psi_p''(t) = \frac{p\exp(-p|t|)}{(1 + \exp(-p|t|))^2}. \tag{7}$$

Notice that $\log(1 + \exp(a))$ may get an incorrect value if $\exp(a)$ is overflowing for a large positive number a such as $a > 1000$. Our new equivalent form in (6) and its derivatives in (7) can avoid such potential overflowing.

Let $r := (r_1, r_2, \ldots, r_m)^T \in \mathbb{R}^m$ and $\psi_p(r) := (\psi_p(r_1), \psi_p(r_2), \ldots, \psi_p(r_m))^T$. Then the smooth problem of (5) is

$$\min_z f_p(z) := \frac{1}{2} z^T z + \frac{C}{2} \bigl(\psi_p(r)\bigr)^T \psi_p(r). \tag{8}$$

SSVM using Newton algorithm to solve (8) can converge quadratically, where the Newton equation $\nabla^2 f_p(z) d = -\nabla f_p(z)$ is solved per iteration with

$$\nabla^2 f_p(z) = I + C H^T \Lambda_p(z) H \in R^{(m+1)\times(m+1)}, \tag{9}$$

$$\Lambda_p(z) := \operatorname{diag}\Bigl(\bigl(\psi_p'(r_1)\bigr)^2 + \psi_p(r_1)\psi_p''(r_1), \ldots, \bigl(\psi_p'(r_m)\bigr)^2 + \psi_p(r_m)\psi_p''(r_m)\Bigr). \tag{10}$$

Its computational cost per iteration is $O(m^3)$. In Sect. 5, we illustrate that the cost can be reduced by the SMW identity [9]. The following lemma is presented to evaluate the approximate error bound:

Lemma 1 (Theorem 2.2 in [17]) *If $\bar{z}$ be the unique minimizer of the smooth function $f_p(z)$ in (8) and z^* be the unique minimizer the original function $f(z)$ in (5), then we have*:

$$\|\bar{z} - z^*\|_2^2 \le \frac{Cm\log^2 2}{2p^2} + \frac{Cm\eta\log 2}{p}, \qquad f(\bar{z}) - f\big(z^*\big) \le \frac{Cm\log^2 2}{2p^2} + \frac{Cm\eta\log 2}{p}.$$

where $\eta = \max_{1\le i\le m}\{|1 - H_i z^*|\}$.

Lemma 1 shows that the error bound between the smooth problem (8) and the original problem (5) is $O(\frac{1}{p})$. In Sect. 3, we will propose a new smoothing method with a tighter error bound $O(\frac{1}{p^2})$, and will enhance the error bound of SSVM to $O(\frac{1}{p^2})$ in Sect. 6.

3 New proposed smoothing method and its properties

Let r_+ be a vector whose ith element is $\max\{r_i, 0\}$. The necessary and sufficient optimality conditions of problem (5) are

$$\nabla f(z) := z - CH^T r_+ = 0. \tag{11}$$

Newton method cannot be used to solve these piecewise linear equations directly because $\nabla f(z)$ is non-differentiable at some points. Different from SSVM, which smooths the objective function of problem (5), we smooth the piecewise linear equations (11) with (6), and the resulted problem is to solve the following system of nonlinear equations:

$$G_p(z) := z - CH^T \psi_p(r) = 0. \tag{12}$$

It can be solved by any Newton-type method. Next we will investigate the error bound and convergence rate of our new smoothing method.

3.1 Error bound of the new smoothing method

In order to analyze the error bound between the solution of the smooth problem and the original one, we introduce the following minimization problem (13), whose first-order necessary and sufficient optimality conditions are just our smooth nonlinear equations (12).

$$\min_z h_p(z) := \frac{1}{2}z^T z + C\sum_{k=1}^{m} \varphi_p(r_k), \tag{13}$$

where

$$\varphi_p(t) := \int_{-\infty}^{t} \psi_p(u)du, \tag{14}$$

with $\psi_p(u)$ defined by (6). Comparing $f(z)$ in (5) and $h_p(z)$ in (13), it can be observed that the item $\frac{1}{2}\max\{0, r_k\}^2$ is approximated by $\varphi_p(r_k)$ for $k = 1, 2, \dots, m$. The difference between them is analyzed in the following lemma.

Lemma 2 *For $\varphi_p(t)$ as defined in* (14), *we have*

$$0 \le \varphi_p(t) - \frac{1}{2}\max\{0, t\}^2 \le \frac{\pi^2}{6p^2}. \tag{15}$$

Proof First we have $\varphi_p(t)$ is a monotonically increasing function because of $\psi_p(u) \ge 0$, and

$$\varphi_p(0) = \frac{1}{p}\int_{-\infty}^{0} \log\bigl(1 + \exp(pu)\bigr)du = \frac{\pi^2}{12p^2}.$$

If $t < 0$, we have

$$0 \le \varphi_p(t) = \varphi_p(t) - \frac{1}{2}\max\{0, t\}^2 \le \varphi_p(0) = \frac{\pi^2}{12p^2} \le \frac{\pi^2}{6p^2}.$$

If $t \ge 0$, then $\psi_p(t) = t + \frac{1}{p}\log(1 + \exp(-pt))$, and we have

$$\begin{aligned}\varphi_p(t) = \varphi_p(0) + \varphi_p(t) - \varphi_p(0) &= \frac{\pi^2}{12p^2} + \int_0^t \left(u + \frac{1}{p}\log\bigl(1 + \exp(-pu)\bigr)\right)du \\ &= \frac{\pi^2}{12p^2} + \frac{1}{2}t^2 + \frac{1}{p}\int_0^t \log\bigl(1 + \exp(-pu)\bigr)du \\ &\le \frac{\pi^2}{12p^2} + \frac{1}{2}t^2 + \frac{1}{p}\int_0^{+\infty} \log\bigl(1 + \exp(-pu)\bigr)du = \frac{\pi^2}{6p^2} + \frac{1}{2}t^2.\end{aligned}$$

Thus (15) holds for any p and t. □

Then we analyze the difference between $h_p(z)$ (13) and $f(z)$ (5) in Lemma 3.

Lemma 3 *The difference between the smooth problem $h_p(z)$ and the original problem $f(z)$ is bounded by the following inequalities*:

$$0 \le h_p(z) - f(z) \le \frac{Cm\pi^2}{6p^2}.$$

Let z^* be the unique solution of the original problem and $\hat{z}$ be the unique solution of the smooth problem. Using Lemma 3, we explore the error bound between z^* and $\hat{z}$ together with the error bound between $f(z^*)$ and $f(\hat{z})$ in Theorem 1.

Theorem 1 *If z^* is the unique solution of problem* (5) *or* (11), *and $\hat{z}$ is the unique solution of problem* (12) *or* (13), *then the following inequalities hold*:

$$\|\hat{z}-z^*\|_2^2 \le \frac{Cm\pi^2}{6p^2} \quad \textit{and} \quad f(\hat{z})-f(z^*) \le \frac{Cm\pi^2}{6p^2}.$$

Proof Because the eigenvalues of the Hessian matrix of $h_p(z)$ and the eigenvalues of any Clarke generalized Jacobian [4] of $\nabla f(z)$ are all larger than 1 and $\nabla f(z^*) = \nabla h_p(\hat{z}) = 0$, we have

$$f(\hat{z})-f(z^*) \ge \nabla f(z^*)^T(\hat{z}-z^*)+\frac{1}{2}\|\hat{z}-z^*\|_2^2 = \frac{1}{2}\|\hat{z}-z^*\|_2^2,$$

$$h_p(z^*)-h_p(\hat{z}) \ge \nabla h_p(\hat{z})^T(z^*-\hat{z})+\frac{1}{2}\|\hat{z}-z^*\|_2^2 = \frac{1}{2}\|\hat{z}-z^*\|_2^2.$$

Thus

$$\|\hat{z}-z^*\|_2^2 \le h_p(z^*)-f(z^*)-\big(h_p(\hat{z})-f(\hat{z})\big) \le h_p(z^*)-f(z^*) \le \frac{Cm\pi^2}{6p^2}.$$

The second and the last inequalities follow form Lemma 3. Similarly, we have

$$f(\hat{z})-f(z^*) \le h_p(z^*)-f(z^*)-\big(h_p(z^*)-h_p(\hat{z})\big) \le h_p(z^*)-f(z^*) \le \frac{Cm\pi^2}{6p^2}.$$

□

According to Theorem 1, the error bound of our smoothing method is $O(\frac{1}{p^2})$, while the corresponding results of SSVM in [17] are $O(\frac{1}{p})$ as listed in Lemma 1. So the error bound of our new method is better than those of SSVM.

3.2 New smoothing algorithm and its convergence rate

Computing the Jacobian matrix of the nonlinear equations (12), we have

$$\nabla G_p(z) = I + CH^T\widehat{\Lambda}_p(z)H, \tag{16}$$

where

$$\widehat{\Lambda}_p(z) := \operatorname{diag}\big(\psi_p'(r_1), \psi_p'(r_2), \ldots, \psi_p'(r_m)\big). \tag{17}$$

Comparing $\Lambda_p(z)$ in (10) and $\widehat{\Lambda}_p(z)$ in (17), it can be observed that $\nabla G_p(z)$ in (16) is simpler than $\nabla^2 f_p(z)$ as defined in (9) for SSVM. The new smooth problem can be solved by the Newton-type algorithm as follows:

Algorithm 1 New Smoothing SVM Algorithm (**NSSVM**)

```
Step 0. Input data $H$ and parameters $p$, $C$; give an initial point $z^0$; select $\varepsilon > 0$ and
        $\sigma \in (0, 0.5)$. Set $k := 0$.
Step 1. Calculate $g^k = G_p(z^k)$. If $\|g^k\| < \varepsilon$, stops; otherwise go to Step 2;
Step 2. Calculate $d^k$ by solving the Newton system $\nabla G_p(z^k)d = -g^k$;
Step 3. (Armijo line search) Choose $\lambda_k = \max\{1, 2^{-1}, 2^{-2}, \ldots\}$ such that
            $h_p(z^k + \lambda_k d^k) \le h_p(z^k) + \sigma \lambda_k {g^k}^T d^k$;
Step 4. Let $z^{k+1} := z^k + \lambda_k d^k$ and $k := k + 1$. Go to Step 1.
```

The following lemma is necessary to prove the convergence rate of the Algorithm 1.

Lemma 4 *For $\nabla G_p(z)$ defined in* (16), *we have*:

(1) $d^T \nabla G_p(z) d \ge \|d\|_2^2$ *for all d and z*;
(2) $\|\nabla G_p(z^1) - \nabla G_p(z^2)\|_2 \le \kappa \|z^1 - z^2\|_2$ *for all z^1, z^2 and some $\kappa > 0$.*

Proof (1) It is clear because all the eigenvalues of $\nabla G_p(z)$ are larger than 1.

(2) According to (16), we have

$$\begin{aligned}
&\|\nabla G_p(z^1) - \nabla G_p(z^2)\|_2 \\
&\quad = C\|H^T(\widehat{\Lambda}_p(z^1) - \widehat{\Lambda}_p(z^2))H\|_2 \le C\|H\|_2^2 \|\widehat{\Lambda}_p(z^1) - \widehat{\Lambda}_p(z^2)\|_2 \\
&\quad = C\|H\|_2^2 \max_i \{|\psi_p''(\eta_i) H_i(z^1 - z^2)|\} \le C\|H\|_2^2 p \|H(z^1 - z^2)\|_2 \\
&\quad \le Cp\|H\|_2^3 \|z^1 - z^2\|_2 = \kappa \|z^1 - z^2\|_2,
\end{aligned}$$

where η_i is obtained by Lagrange mean value theorem and the second inequality is implied by $\psi_p''(t) \le p$ for any t in (7), and $\kappa := Cp\|H\|_2^3$. □

With the help of Lemma 4, the convergence rate of the new smoothing algorithm can be obtained as in the following Theorem 2. The proof of Theorem 3.2 in [17] can be used to prove Theorem 2 and is thus omitted here.

Theorem 2 *Let $\{z^k\}$ be a sequence generated by Algorithm 1, and $\hat{z}$ be the unique solution of problem* (12). *Then $\{z^k\}$ converges to the unique solution $\hat{z}$* ***quadratically***.

Remark 1 In NSSVM, the objective function $h_p(z)$ is complicated to compute because it contains an integral function $\varphi_p(t)$ (14), which affects the efficiency of the algorithm in Armijo line-search procedure (Step 3). Since the error between $h_p(z)$ and $f(z)$ is $O(\frac{1}{p^2})$ for a large p, we can use $f(z)$ in place of $h_p(z)$ in Armijo line search procedure. Experimental results show that it is a good choice.

4 Kernel versions extension

There are three versions to solve nonlinear classification problem, while the mapping vector $\mathbf{x}_i \in \mathbb{H}$ is defined implicitly by the kernel function as $\langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathbb{H}} = k(x_i, x_j)$. We summarize them here.

(i) According to GSVM [21], as mentioned above, problem (5) and its necessary and sufficient optimality conditions (11) are considered. If the optimal solution is z^*, then the resulting classification surface is $\theta(x) := [k(x_1, x), k(x_2, x), \ldots, k(x_m, x)\ 1]z^* = 0$.

(ii) Stemming from the dual problem of (2) with $\sigma = 2$, we have

$$\min_{0 \le \alpha \in \mathbb{R}^m} \frac{1}{2}\alpha^T \left[D\big(K + ee^T\big)D + \frac{1}{C}I \right] \alpha - e^T \alpha, \tag{18}$$

where $D = \operatorname{diag}(y)$ and $e = [1, 1, \ldots, 1]^T$. Decomposing the kernel matrix as $K = UU^T$ exactly or approximately, Zhou et al. [31] prove that the problem (18) is equivalent to solving the following unconstrained problem

$$\min_{z \in \mathbb{R}^{m+1}} \frac{1}{2} z^T z + \frac{C}{2} r^T r_+, \tag{19}$$

with $r = e - D[U\ \ e]z$. If the optimal solution to (19) is z^*, then the solution to (18) is $\alpha^* = C(e - Hz^*)_+$, and the classification surface is $\theta(x) := \sum_{\alpha_i^* > 0} y_i \alpha_i^* (k(x_i, x) + 1) = 0$.

(iii) Plugging (3) into (2) with $\sigma = 2$ and noting $z = [\beta^T\ \ b]^T$, $r = e - D[K\ \ e]z$, we have

$$\min_{z \in \mathbb{R}^{m+1}} \frac{1}{2} z^T \begin{bmatrix} K & 0 \\ 0 & 1 \end{bmatrix} z + \frac{C}{2} r^T r_+, \tag{20}$$

The resulted classification surface is $\theta(x) := [k(x_1, x), \ldots, k(x_m, x), 1]z^* = 0$.

In all above three approaches, their optimal conditions of the corresponding programming problems are *piecewise linear equations* that can be smoothed by our new method, and our results $O(\frac{1}{p^2})$ on error bound between the approximate solution and the exact optimal solution are achieved. The boost schemes mentioned in Sect. 5 also work for them.

In Sect. 7, we implement the smoothing algorithms based on *case (i)*.

5 Boosting techniques

In this section, we investigate some techniques to boost the performance of these smoothing algorithms. To this end, the SMW identity is adopted to solve Newton equation, and/or some schemes are studied to update the Hessian matrix with low computational complexity.

5.1 Solving Newton equation by SMW identity

If D is a diagonal matrix with smaller size, the following SMW identity [9] can be used to calculate $(I + H^T DH)^{-1}$ with less computational complexity:

$$\left(I + H^T DH\right)^{-1} = I - H^T\left(D^{-1} + HH^T\right)^{-1} H. \tag{21}$$

As soon as $\Lambda_p(z^k)$ in (9) or $\widehat{\Lambda}_p(z^k)$ in (17) has a sparse diagonal, the cost to obtain the corresponding Newton direction based on (21) is much less than $O(m^3)$.

Take NSSVM as an example. For a large smoothing parameter p, $(\widehat{\Lambda}_p(z^k))_{i,i} = \psi'_p(r_i^k) = \frac{\min\{1,\exp(pr_i^k)\}}{1+\exp(-p|r_i^k|)}$ in (17) is numerically zero or very tiny if $r_i^k = 1 - H_i z^k < 0$, and $(\widehat{\Lambda}_p(z^k))_{i,i}$ is significantly positive if $r_i^k \geq 0$, which corresponds to a bounded support vector x_i for the current solution z^k. That is to say, the diagonal elements of $\widehat{\Lambda}_p(z^k)$ are commonly as sparse as the solutions of SVM (the Lagrange multipliers corresponding to input samples x_i). Let $J^k = \{i \mid (\widehat{\Lambda}_p(z^k))_{ii} \geq \tau, i = 1, 2, \ldots, m\}$ with an extremely tiny positive τ. Then $\nabla G_p(z^k)$ can be approximately calculated as follows:

$$\nabla G_p\left(z^k\right) \approx I + CH_{J^k}^T \widehat{\Lambda}_{J^k,J^k} H_{J^k}. \tag{22}$$

where H_{J^k} is a sub-matrix comprised by the rows of H in the index set J^k and $\widehat{\Lambda}_{J^k,J^k}$ is a sub-matrix of $\widehat{\Lambda}_p(z^k)$ corresponding to the rows and columns in J^k. Experimental results show that if $\tau = 10^{-10}$ or less, approximation errors can be neglected.

Based on SMW identity (21) and $\nabla G_p(z^k)$ in (22), Newton direction d^k of NSSVM is obtained as follows:

$$d^k = -g^k + H_{J^k}^T \left(\frac{1}{C}\widehat{\Lambda}_{J^k,J^k}^{-1} + H_{J^k} H_{J^k}^T\right)^{-1} H_{J^k} g^k. \tag{23}$$

Let $\widetilde{m}_k := |J^k|$, which approximate equals the number of support vectors. The computational complexity of the algorithm can be reduced from $O(m^3)$ to $O(m\widetilde{m}_k^2)$ per iteration. Experiments show that $\widetilde{m}_k$ is much smaller than m after several iterations.

The analysis for SSVM is similar as above.

5.2 Updating the reduced Hessian with boosting skills

In order to solve large-scale problems, the reduced techniques have been proposed to accelerate SVM training algorithms. In RSVM [16], a reduced set W in (4) is selected randomly from the index set $\{1, 2, \ldots, m\}$ satisfying $\overline{m} := |W| \leq 0.1m$. If let $\overline{K} \in \mathbb{R}^{m\times\overline{m}}(\overline{m} = |W|)$ be a sub-matrix comprised by the columns of the full kernel matrix K corresponding to the reduced set W, then the formulation of RSVM in [15, 16, 20] is

$$\min_{z\in\mathbb{R}^{\overline{m}+1}} \frac{1}{2}z^T z + \frac{C}{2}r^T r_+, \tag{24}$$

which is very similar to (5). The differences between them are on the dimensions of z and

$$r_i := 1 - \overline{H}_i z \quad \text{with } \overline{H}_i := y_i[\overline{K}_i \ 1],\ i = 1, 2, \ldots, m. \tag{25}$$

RSVM smooths the objective function of (24) by the exponential penalty function (6), while our new algorithm smooths the necessary and sufficient optimality conditions of (24). All algorithms solve Newton equation

$$B^k d^k = -g^k$$

to obtain direction d^k, where $B^k := I + C\overline{H}^T \overline{\Lambda}^k \overline{H} \in \mathbb{R}^{(\overline{m}+1)\times(\overline{m}+1)}$ with a diagonal matrix $\overline{\Lambda}^k \in R^{m\times m}$ defined correspondingly as $\Lambda_p(z^k)$ in (10) for RSVM and $\widehat{\Lambda}_p(z^k)$ in (17) for the new algorithm, in which r and $\overline{H}$ are defined by (25).

There are three methods to further boost the performance of the algorithms:

Scheme I: Based on the sparsity of the diagonal elements of $\overline{\Lambda}^k$, we have

$$B^k := I + C\overline{H}^T_{J^k} \overline{\Lambda}^k_{J^k,J^k} \overline{H}_{J^k}. \tag{26}$$

where $J^k := \{i | \overline{\Lambda}^k_{i,i} \geq \tau, i = 1, \ldots, m\}$ with a tiny positive τ. Let $\widetilde{m}_k := |J^k|$. The total computational complexity per iteration is $O(\widetilde{m}_k \overline{m}^2 + \overline{m}^3) = O(\max\{\widetilde{m}_k, \overline{m}\}\overline{m}^2)$ where $O(\widetilde{m}_k \overline{m}^2)$ is for computing B^k and $O(\overline{m}^3)$ is for solving Newton equation.

Scheme II: B^k in (26) can be updated iteratively as in [31]:

$$B^k = B^{k-1} + C\overline{H}^T \Upsilon^k \overline{H} = B^{k-1} + C\overline{H}^T_{J_0^k} \Upsilon^k_{J_0^k, J_0^k} \overline{H}_{J_0^k}, \tag{27}$$

where $\Upsilon^k = \overline{\Lambda}^k - \overline{\Lambda}^{k-1}$ and $J_0^k = \{i | \Upsilon^k_{i,i} \neq 0, i = 1, \ldots, m\}$. Let $\widehat{m}_k := |J_0^k|$. The total computational complexity is reduced to $O(\widehat{m}_k \overline{m}^2 + \overline{m}^3) = O(\max\{\widehat{m}_k, \overline{m}\}\overline{m}^2)$ per iteration.

Scheme III: If SMW identity (21) is applied to solve the reduced Newton equation, we have

$$d^k = -g^k + \overline{H}^T_{J^k} \left(\frac{1}{C} \left(\Lambda^k_{J^k,J^k} \right)^{-1} + \overline{H}_{J^k} \overline{H}^T_{J^k} \right)^{-1} \overline{H}_{J^k} g^k. \tag{28}$$

The total complexity per iteration is reduced to $O(\overline{m}\widetilde{m}_k^2 + \widetilde{m}_k^3) = O(\max\{\widetilde{m}_k, \overline{m}\}\widetilde{m}_k^2)$.

Experimental results in Sect. 7 show that $\widehat{m}_k \leq \widetilde{m}_k$ holds apparent after several iterations and $\widehat{m}_k$ has a tendency to 0 while $\widetilde{m}_k$ has a tendency to the number of the support vectors. This is to say that Scheme II is better than Scheme I.

In our implements, we compound these cases to get a better one:

Optimal Scheme: If $\widetilde{m}_k > \overline{m}$, Scheme II is chosen with the complexity $O(\widehat{m}_k \overline{m}^2)$; Otherwise, Scheme III is chosen with the complexity $O(\overline{m}\widetilde{m}_k^2)$.

Its computational complexity is often less than $O(m\overline{m}^2)$ given by [20] for RSVM, and is also less than $O(md^2_{max})$ given in [13] for PSVM because $\widehat{m}_k$ and $\widetilde{m}_k$ are often much smaller than m.

6 Tighter error bound for SSVM

Given the same smoothing parameter p, experimental results show that the accuracies of our new method are always very similar to those of SSVM. This is not consistent with the theoretical analysis that the error bound of SSVM is $O(\frac{1}{p})$ in [17], while the error bound of our new method is $O(\frac{1}{p^2})$. In the following we present a new proof to tighten the error bound of SSVM.

The difference between $\psi_p^2(t)$ and $\max\{t,0\}^2$ is analyzed in Lemma 5.

Lemma 5 *If* $\psi_p(t)$ *be defined in* (6), *then the following inequalities holds*:

$$0 \le \psi_p^2(t) - \max\{t,0\}^2 \le \frac{1}{p^2}. \tag{29}$$

Proof The left inequality is obvious and the right inequality is proven as follows.

If $t \le 0$,

$$\psi_p^2(t) - \max\{t,0\}^2 = \psi_p^2(t) = \frac{1}{p^2}\log^2\bigl(1+\exp\bigl(-p|t|\bigr)\bigr) \le \frac{\log^2 2}{p^2} \le \frac{1}{p^2}.$$

If $t > 0$,

$$\begin{aligned}\psi_p^2(t) - \max\{t,0\}^2 &= \frac{1}{p^2}\log^2\bigl(1+\exp(-pt)\bigr) + \frac{2}{p}t\log\bigl(1+\exp(-pt)\bigr)\\ &\le \frac{1}{p^2}\exp(-2pt) + \frac{2}{p}t\exp(-pt) \le \frac{1}{p^2}.\end{aligned}$$

The first inequality holds because $\log(1+a) \le a$ for $a \ge 0$, and the last inequality holds because $\phi(t) := \frac{1}{p^2}\exp(-2pt) + \frac{2}{p}t\exp(-pt)$ is a monotonically decreasing function for $t \ge 0$ (since $\phi'(t) = \frac{2}{p}\exp(-pt)(1-pt-\exp(-pt)) \le 0$). □

A tighter error bound for SSVM can be proven using the same method as Theorem 1.

Theorem 3 *If* z^* *be the unique solution of problem* (5) *or* (11), *and* $\bar{z}$ *be the unique solution of problem* (8), *then the following inequalities hold*:

$$\|\bar{z} - z^*\|_2^2 \le \frac{Cm}{2p^2} \quad \textit{and} \quad f(\bar{z}) - f(z^*) \le \frac{Cm}{2p^2}.$$

By Theorem 3, the error bound of SSVM is also $O(\frac{1}{p^2})$.

7 Experimental results

In this section, we perform some experiments to compare the new algorithm (NSSVM) with SSVM [17] as well as PSVM [13] and also to show the performance of the proposed boosting techniques, and the reported results in [27] are also cited here for comparison. All the experiments are run on a Personal Computer with a Intel Core i3 2100 CPU and a maximum of 4 Gbytes of memory available for all processes. The computer runs Windows 7 with Matlab 7.10. Some Matlab codes of our algorithms are available on the webpage http://web.xidian.edu.cn/sszhou/en/paper.html

7.1 Compare NSSVM and SSVM on checkerboard dataset

This set of experiments is on a non-linearly separable example called "tried and true" checkerboard dataset, first given in [10], which is widely used to show the effectiveness of nonlinear kernel methods [16, 17, 22, 23, 31]. The data is generated by uniformly discretizing the regions $[0, 199] \times [0, 199]$ to $200^2 = 40{,}000$ points, and labeling two classes spaced by 4×4 grids (See [16, 17, 22, 23, 31] for the plots of the dataset). The training set is randomly sampled from the 40,000 data points with different training sizes, and the remainders of the points are left as the testing set. The chosen kernel function is the Gaussian kernel function $K(x_i, x_j) = \exp(-0.001\|x_i - x_j\|^2)$. We set $\varepsilon = 10^{-4}$ to stop the algorithms and $\tau = 10^{-10}$ to prune the diagonal elements of Hessian matrices, and let $C = 10^4$, $p = 10^8$.

Some plots are given in Fig. 1 to show the average changing per iteration of $\widetilde{m}_k = |J^k|$ in (26) and $\widehat{m}_k = |J_0^k|$ in (27). We take the new algorithm as an example, and the results obtained by SSVM are similar. In Fig. 1(a), the plots are obtained on an 8,000 training dataset with the reduced size $\overline{m} = 800$, and the results are the average values of the first 55 iterations on 20 random trials. In Fig. 1(b), the plots are obtained on a 15,000 training dataset with the reduced size $\overline{m} = 1000$, and the results are the average values of the first 65 iterations on 20 random trials. Semi-log plots with logarithmic (base 10) scale for the Y-axis are used.

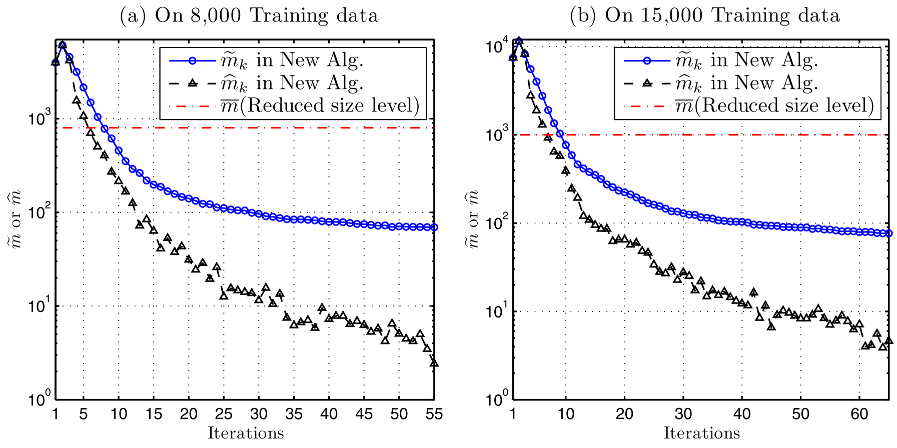

Fig. 1 Average tendency of $\widetilde{m}_k$ and $\widehat{m}_k$ per iteration on 20 random trials. Logarithmic (base 10) scale for the Y-axis is used. The figures show that our optimal scheme is efficient: The computational complexity is $O(\widehat{m}_k\overline{m}^2)$ for the first 5 (or 6) iterations (over the reduced size level) and is $O(\overline{m}\widetilde{m}_k^2)$ for the rest, which is much less than $O(m\overline{m}^2)$ for RSVM in [16, 20] and $O(md_{max}^2)$ for PSVM in [13]

Table 1 Experimental results with the proposed techniques on some medium sized training datasets. All the data are obtained on an average over 20 random trials and the standard deviations are also given in brackets for columns 3–6; $\|\nabla f\|$ (∇f is defined in (11)) is to check the quality of these approximate solutions in term of exact solutions in the 7th column, and the quality of the solutions achieved by the two algorithms is compared in the 8th column, where $\hat{z}$ is computed by NSSVM and $\bar{z}$ is computed by SSVM with Optimal Scheme. "SS" and "NS"are "SSVM" and "NSSVM" for shorts respectively

Data size	Alg.	Test accuracy (%)	Iteration	Training time (s)		$\|\nabla f\|$	$f(\hat{z}) - f(\bar{z})$
				Opt. scheme	Sheme I		
8,000	SS	99.89 (0.05)	60.00 (6.51)	7.96 (0.50)	13.69 (0.95)	2.24e-7	7.31e-11
	NS	99.89 (0.05)	62.20 (6.65)	8.34 (0.44)	14.07 (1.07)	1.55e-7	
10,000	SS	99.92 (0.03)	69.70 (10.1)	14.40 (1.22)	25.97 (2.95)	3.23e-7	7.28e-12
	NS	99.92 (0.03)	69.65 (5.30)	14.76 (0.88)	25.64 (1.61)	1.91e-7	
12,000	SS	99.94 (0.02)	74.20 (8.76)	17.48 (1.17)	29.82 (2.47)	2.11e-7	2.73e-10
	NS	99.94 (0.02)	74.90 (8.32)	17.86 (1.25)	29.90 (2.31)	3.09e-7	
15,000	SS	99.96 (0.02)	76.90 (12.7)	22.41 (2.29)	34.93 (4.08)	2.53e-7	−1.06e-10
	NS	99.96 (0.02)	77.40 (9.96)	22.19 (1.86)	35.07 (3.45)	2.74e-7	
18,000	SS	99.97 (0.02)	80.05 (7.74)	26.92 (1.67)	40.57 (3.14)	1.21e-7	−3.76e-10
	NS	99.97 (0.02)	79.60 (9.59)	26.32 (1.93)	40.14 (3.64)	3.35e-7	

From Fig. 1, it can be observed that $\widehat{m}_k$ is often smaller than $\widetilde{m}_k$. Both $\widehat{m}_k$ and $\widetilde{m}_k$ have a tendency to quickly converge to a small number. $\widetilde{m}_k$ has a tendency to converge to the number of support vectors (near 70) while $\widehat{m}_k$ has a tendency to converge to a very small number less than 10 in the iterative process. They are both much smaller than the reduced size $\overline{m}$ after 6 iterations. It can be concluded that the boosting skills dramatically reduce the complexity of the algorithms: The computational complexity is $O(\widehat{m}_k\overline{m}^2)$ for the first 5 (or 6) iterations (over the reduced size level) and is $O(\overline{m}\widetilde{m}_k^2)$ for the rest (below the reduced size level), which is less than $O(m\overline{m}^2)$ for RSVM in [16, 20] and $O(md_{max}^2)$ for PSVM in [13].

More training sets are randomly sampled from the checkerboard dataset with different training sizes for further experiments, where algorithms with the reduced method and the boosting techniques in Sect. 5 are compared. Table 1 lists the experimental results. The reduced set, with the size $\overline{m} = 0.1m$ but limited by the lower bound 100 and the upper bound 1,000, is randomly selected from the training set. All the data in Table 1 are obtained on an average over 20 random experiments, and the standard deviations are also given in brackets for some columns. In the 5th and 6th columns, the mean training time of the proposed Optimal Scheme in Sect. 5.2 is compared with the mean training time of RSVM [16, 20] only with the sparsity considered as (26), which corresponds to Scheme I in Sect. 5.2. In the 7th column, we list the average value of $\|\nabla f\|$ (∇f is defined in (11)) to check the quality of those approximate solutions in term of the exact solutions of the original problem. In the 8th column, we list the average value of $f(\hat{z}) - f(\bar{z})$ to compare the quality of the solutions achieved by the two algorithms, where $\hat{z}$ is obtained by the new smoothing algorithm and $\bar{z}$ is obtained by SSVM.

The results in Table 1 show that the new algorithm works as well as SSVM [17]. The two algorithms can get the same training accuracies (not listed) and testing accuracies with a little difference in the training time and the number of iterations. It is coherent with our theoretical analysis that the two algorithms have similar error bound. At the same time, according to the 5th and 6th columns, our optimal scheme saves nearly 1/3 of the training time. In the 7th column, it is clear that the obtained solutions are good approximate solutions of original problems, since the mean values of $\|\nabla f\|$ are all much smaller than our stop criterion $\varepsilon = 10^{-4}$ and all ∇f meet the optimality conditions of original problems with tiny errors. The average values of $f(\hat{z}) - f(\bar{z})$ in the last column show that the difference of the two algorithms is also very tiny and clearly they both satisfy inequality $-\frac{Cm}{2p^2} \le f(\hat{z}) - f(\bar{z}) = f(\hat{z}) - f(z^*) - (f(\bar{z}) - f(z^*)) \le \frac{Cm\pi^2}{6p^2}$, which is implied by our theorems on error bound.

7.2 Compare SSVM, NSSVM and PSVM on some large benchmark datasets

This set of experiments is to evaluate the performance of the SSVM, our new NSSVM, as well as the PSVM in [13]. And the reported results in [27] are also cited here to compare with our algorithm. The Matlab codes for PSVM are gotten from the site [2], which are designed for paper [13]. Our codes for SSVM and NSSVM are available on the webpage http://web.xidian.edu.cn/sszhou/en/paper.html.

All algorithms are compared on the training time and the classification accuracies with reduced method, and the iterations of SSVM and our NSSVM are also given (we do not listed the iterations of PSVM since it is not easy to count).

The reduced sets for SSVM and NSSVM is random chosen while the reduced set for PSVM is iteratively updated by some complicate skills (see [13] for details). If we use the same reduced set size, PSVM is much slower than SSVM and NSSVM, but most of time has a little bit higher classification accuracies than SSVM and NSSVM have. So in Table 2, we half the reduced set size of PSVM so that the classification accuracies of three algorithms are comparable.

Eight large benchmark datasets from the site of [5] are adopted. Some of them are also appeared in [13] and [27]. For simplicity, Gaussian kernel function $k(x, y) = \exp(-\gamma\|x - y\|^2)$ with different spread parameters γ is used for all datasets. The kernel spread parameters γ and the tradeoff parameters C are roughly chosen by 10-fold cross-validation. The details of the data sets and the corresponding selected parameters are listed as follows:

- ***USPS3*** and ***USPS8***—USPS is a multi-class data set with 10 classes including 7,291 training samples and 2,007 test samples. Each sample has 256 features. Here USPS3 is used for the task of classifying the digit 3 versus the rest of the classes and USPS8 is used for the task of classifying the digit 8 versus the rest of the classes. The parameters are $C = 40$ and $\gamma = 0.02$.
- ***Adult***—It is the version given by Platt which has 32,561 training examples and 16,281 test examples. Each example has 123 binary features, and the parameters are $C = 1$ and $\gamma = 0.05$.
- ***Shuttle***—It is a multi-class data set with seven classes including 43,500 training examples and 14,500 test examples. Each example has 9 features. Here a binary

Table 2 Experimental results of the related algorithms on some larger size benchmark datasets. All the results are obtained on an average over 20 random trials and the standard deviations are given in brackets. Here "Tr", "Ts", "Dm" and "Rd" are "Training size", "Test size", "Dimension of samples" and "Reduced size" of the input datasets respectively, and "SS", "NS" and "PS" are "SSVM", "NSSVM" and "PSVM" for shorts respectively. Reported results [27] for some sets by other popular SVM solver are listed in the footnotes

	USPS3	USPS8[a]	Adult[b]	Shuttle	IJCNN	MNIST3	MNIST8[c]	Vechile
Tr.	7291	7291	32561	43500	49990	60000	60000	78823
Ts.	2007	2007	16281	14500	91790	10000	10000	19705
Dm.	256	256	123	9	22	784	784	100
Rd.	365	365	1628	2000	2000	2000	2000	2000
	Training time (s)							
SS	**1.04** (0.05)	**1.06** (0.03)	18.94 (0.66)	69.24 (4.27)	59.32 (0.65)	85.33 (0.83)	**85.64** (0.58)	164.09 (7.86)
NS	1.06 (0.06)	1.08 (0.05)	**18.81** (0.52)	**68.83** (4.31)	**58.20** (0.60)	**85.11** (0.81)	86.90 (1.27)	**162.29** (3.24)
PS	11.64 (0.28)	11.27 (0.29)	138.69 (1.03)	103.63 (7.78)	160.39 (4.70)	528.45 (6.17)	568.57 (6.17)	450.00 (3.84)
	Test classification accuracy (%)							
SS	98.49 (0.11)	99.31 (0.08)	**85.31** (0.04)	**99.87** (0.01)	**98.60** (0.09)	99.55 (0.04)	99.36 (0.04)	**88.34** (0.05)
NS	98.49 (0.11)	99.31 (0.08)	**85.31** (0.04)	**99.87** (0.01)	**98.60** (0.09)	99.55 (0.04)	99.36 (0.04)	**88.34** (0.05)
PS	**98.66** (0.07)	**99.36** (0.08)	85.20 (0.05)	99.86 (0.01)	98.49 (0.07)	**99.57** (0.04)	**99.41** (0.05)	88.23 (0.08)
	Iterations							
SS	13.07 (0.62)	17.64 (0.93)	8.43 (0.51)	38.71 (3.79)	14.21 (0.70)	13.21 (0.58)	13.07 (0.47)	9.36 (0.63)
NS	13.07 (0.62)	17.64 (0.93)	8.43 (0.51)	38.79 (3.83)	14.21 (0.70)	13.21 (0.58)	13.07 (0.47)	9.29 (0.47)

[a]Results in [27]: `Pegasos` (99.54 %, 120 s)—means achieve 99.54 % test accuracy with 120 s training time, `SVMlight` (99.54 %, 3.3 s), Optima SVM (99.54 %, not reported).

[b]Results in [27]: `Pegasos` (84.5 %, 30 s), `SVMlight` (84.9 %, 59 s), Optima SVM (85.1 %, not reported).

[c]Results in [27]: `Pegasos` (99.4 %, 4200 s), `SVMlight` (99.42 %, 290 s), Optima SVM (99.43 %, not reported).

classification problem is solved to separate class 1 from the rest, and the parameters used are $C = 10{,}000$ and $\gamma = 2$.

- ***IJCNN***—It has 49,990 training examples and 91,701 test examples. Each example is described by 22 features, and the parameters used are $C = 1{,}000$ and $\gamma = 0.5$.
- ***MNIST3 and MNIST8***—MNIST is a multi-class data set to classify the handwritten digits 0 to 9. It has 60,000 training examples and 10,000 test examples. Each example is described by 28×28 features. Here MNIST3 is the task of classifying digit 3 versus the rest of the classes and MNIST8 is the task of classifying digit 8 versus the other classes. The parameters used are $C = 40$ and $\gamma = 0.02$.
- ***Vechile***—It is the combined SensIT Vechile in [5]. It has 78,823 training examples and 19,705 test examples in three classes. Each example has 100 features. Here the task of classifying class 3 from the rest is trained, and the parameters are set as $C = 1{,}000$ and $\gamma = 0.25$.

The experimental results are listed in Table 2, where the means of abbreviations in the first column are explained in the table caption. All the results are obtained on an average over 20 random trials and the standard deviations are given in brackets, and the best values are in bold.

The results in Table 2 again show that the new algorithm works as well as SSVM [17]. The two algorithms can get the same testing accuracies with a little difference in the training time and the number of iterations, which are coherent with our theoretical analysis that the two algorithms have similar error bound. All the deviations (in brackets) are very small, so the mean values are confident.

The results in Table 2 also show that the smooth algorithms with the proposed boosting skills can achieve test accuracies comparable to PSVM, but save the training time about 2 to 10 times. Notice that the complicate codes of PSVM are more than 200 lines in Maltab while our SSVM and NSSVM are all less than 40 lines: the new smooth algorithms are more suited for the common SVM users than PSVM.

Furthermore, as reported in the footnotes of Table 2, by comparing our results against those obtained in [27] (with a more powerful computer), we can conclude that the smooth algorithms with our boosting skills are faster than some popular SVM solvers with the comparable test accuracies.

By the way, in this set of experiments, we found that Scheme II in Sect. 5.2 is the most useful skill to boost the algorithms. On the last 6 large training datasets with the smaller reduced set size, our Optimal Scheme is equivalent to Scheme II since $\widetilde{m}_k$ is larger than the reduced set size $\overline{m}$.

8 Conclusion

The quadratic convergence rate algorithms for training SVM with smooth method are studied in this paper. In the primal space, SVM can be modeled as an unconstrained problem with a piecewise quadratic objective function, whose optimality condition is a system of piecewise linear equations. We derive a new smoothing, quadratically convergent algorithm by smoothing the piecewise linear equations. The smoothing method is different from SSVM [17], where the objective function is smoothed. We prove that both the algorithms have the same approximation error bound $O(\frac{1}{p^2})$, which is tighter than the result given in [17].

For large-scale nonlinear problems, some reduced methods and boosting techniques are introduced to reduce the computational complexity of these algorithms and to improve the performance of these smoothing methods. Many experimental results support that the new smoothing algorithm is not only as good as SSVM, but also simpler than SSVM on the Hessian matrix. It can also be observed that the smoothing algorithms with the proposed boosting skills are comparable with similar algorithms like PSVM [13], but simpler and easier in implement.

Acknowledgements We gratefully acknowledge the supports of National Natural Science Foundation of China (NNSFC) under Grant Nos. 60603098, 61072144, 61179040, 61173089, 11101321 and 11101322. We also thank the anonymous reviewers and editors for their helpful comments to improve the presentation.

References

1. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**, 1–27 (2011). http://www.csie.ntu.edu.tw/~cjlin/libsvm
2. Chapelle, O.: Support Vector Machines in the primal (2006). http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/primal/
3. Chapelle, O.: Training a Support Vector Machine in the primal. Neural Comput. **19**(5), 1155–1178 (2007)
4. Clarke, F.H.: Optimization and Nonsmooth Analysis. Willey, New York (1983)
5. Fan, R.E., Lin, C.J.: LIBSVM Data (2010). http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
6. Ferris, M.C., Munson, T.S.: Semismooth Support Vector Machines. Math. Program., Ser. B **101**, 185–204 (2004)
7. Fine, S., Scheinberg, K.: Efficient SVM training using low-rank kernel representations. J. Mach. Learn. Res. **2**, 243–264 (2001)
8. Fletcher, R., Zanghirati, G.: Binary separation and training support vector machines. Acta Numer. **19**, 121–158 (2010)
9. Golub, G.H., Loan, C.F.V.: Matrix Computations. Johns Hopkins University Press, Baltimore (1996)
10. Ho, T.K., Kleinberg, E.M.: Building projectable classifiers of arbitrary complexity. In: Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria, pp. 880–885 (1996)
11. Joachims, T.: In: SVM^{light}, Support Vector Machine (1998). http://www.cs.cornell.edu/people/tj/svm_light/
12. Joachims, T.: Making large-scale SVM learning practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods—Support Vector Learning, pp. 169–184. MIT Press, Cambridge (1999)
13. Keerthi, S.S., Chapelle, O., Decoste, D.: Building Support Vector Machines with reduced classifier complexity. J. Mach. Learn. Res. **7**, 1493–1515 (2006)
14. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput. **13**, 637–649 (2001)
15. Lee, Y.J., Huang, S.Y.: Reduced support vector machines: a statistical theory. IEEE Trans. Neural Netw. **18**(1), 1–13 (2007)
16. Lee, Y.J., Mangasarian, O.L.: RSVM: reduced Support Vector Machines. In: CD Proceedings of the SIAM International Conference on Data Mining, pp. 1–17. SIAM, Chicago (2001)
17. Lee, Y.J., Mangasarian, O.L.: SSVM: a smooth Support Vector Machine for classification. Comput. Optim. Appl. **20**(1), 5–22 (2001)
18. Lin, C.J.: On the convergence of the decomposition method for Support Vector Machines. IEEE Trans. Neural Netw. **12**, 1288–1298 (2001)
19. Lin, C.J.: Asymptotic convergence of an SMO algorithm without any assumptions. IEEE Trans. Neural Netw. **13**, 248–250 (2002)
20. Lin, K.M., Lin, C.J.: A study on reduced Support Vector Machines. IEEE Trans. Neural Netw. **14**(6), 1449–1459 (2003)

21. Mangasarian, O.L.: Generalized Support Vector Machine. In: Smola, A.J., Bartlett, P., Schökopf, B., Schuurmans, D. (eds.) Advances in Large Margin Classifiers, pp. 135–146. MIT Press, Cambridge (2000)
22. Mangasarian, O.L., Musicant, D.R.: Successive overrelaxation for Support Vector Machines. IEEE Trans. Neural Netw. **10**(5), 1032–1037 (1999)
23. Mangasarian, O.L., Musicant, D.R.: Lagrangian Support Vector Machines. J. Mach. Learn. Res. **1**, 161–177 (2001)
24. Melacci, S., Belkin, M.: Laplacian support vector machines trained in the primal. J. Mach. Learn. Res. **12**, 1149–1184 (2011)
25. Platt, J.C.: Fast training of Support Vector Machines using Sequential Minimal Optimization. In: Schölkopf, B., Burges, C.J., Smola, A.J. (eds.) Advances in Kernel Method-Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1999)
26. Schölkopf, B., Smola, A.J.: Learning with Kernels-Support Vector Machines, Regularization, Optimization and Beyond. The MIT Press, Cambridge (2002)
27. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: primal estimated sub-gradient solver for SVM. Math. Program. **127**(1), 3–30 (2011)
28. Vapnik, V.N.: An overview of statistical learning theory. IEEE Trans. Neural Netw. **10**(5), 988–999 (1999)
29. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (2000)
30. Zhou, S., Liu, H., Ye, F., Zhou, L.: A new iterative algorithm training SVM. Optim. Methods Softw. **24**(6), 913–932 (2009)
31. Zhou, S., Liu, H., Zhou, L., Ye, F.: Semismooth Newton Support Vector Machine. Pattern Recognit. Lett. **28**, 2054–2062 (2007)