STATISTICS,OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **1**, December 2013, pp 82–106.
Published online in International Academic Press (www.IAPress.org)

Which is better? Regularization in RKHS vs $\mathbb{R}^m$ on Reduced SVMs

Shuisheng Zhou [1, *]

[1]*School of Mathematics and Statistics, Xidian University, China*

Received 8 October 2013; Accepted 25 November 2013
Editor: Junfeng Yang

Abstract In SVMs community, the learning results are always a combination of the selected functions. SVMs have two mainly regularization models to find the combination coefficients. The most popular model with m input samples is norm-regularized the classification function in a reproducing kernel Hilbert space(RKHS), and it is converted to an optimization problem in $\mathbb{R}^m$ by duality or representer theorem. Another important model is generalized support vector machine(GSVM), in which the coefficients of the hypothesis is norm-regularized in the Euclidean space $\mathbb{R}^m$. In this work, we analyze the difference between them on computing stability, computational complexity and the efficiency of the Newton type algorithms, especially on the reduced SVMs for large scale training problems. Many typical loss functions are considered. Our studies show that the model of GSVM has more advantages than the other model. Some experiments are given to support our analysis.

Keywords SVMs;RSVM;regularizer;representer theorem;Newton algorithms

DOI: 10.19139/soic.v1i1.27

1. Introduction

Based on the Vanpnik and Chervonkis' structural risk minimization principle [1, 2], support vector machines (SVMs) are proposed as the computationally powerful machine learning methods for supervised learning. They are the popular methods in the past 10 more years, and widely used in classification and regression

*Correspondence to: School of Mathematics and Statistics, Xidian University, Xi'an 710071, China
Email: sszhou@mail.xidian.edu.cn

ISSN 2310-5070

problems, such as character identification, disease diagnoses, face recognition, the time serial prediction, etc.

Historically, SVMs are motivated from a geometrical illustration, where an optimal linear classification function with the maximal margin is found in the feature kernel space. In this illustration, this optimal linear function may have an offset and may not always come through the origin. But some research results show that, when a large feature kernel space is considered, this illustrations with offset have some flaws (see [3]) except that cause an additional equality constraint in the dual problem, and the investigation of the generalization performance of SVMs does not suggest that the offset offers any improvement for such kernel space like Gaussian kernels [4]. Roughly speaking, the offset only has advantage on the linear classification problems, where the optimal linear classification function is found in input space directly, and has little advantages on the nonlinear classification with kernel function.

In this work, the nonlinear problem with kernel function is studied. For simplicity, we only take the SVMs without offset into account as [3], [5] and [6] do. Actually, the offset can be added with an extra attribute 1 added to every sample [5].

1.1. Representer theorem

Representer theorem is an important result in learning problem where many samples are acted as the inputs of the problem. It states that the solution (hypothesis) of the learning problem is a linear combination of the input samples (linear problem) or a linear combination of the kernel functions of the input samples (nonlinear problem). This property has important computational implications in kernel version learning problem, because it can transform the problems in a very high or infinite dimensional space into a finite dimensional problems with the size of the input learning data, where the finite combination coefficients are solved according to the input data. Especially, many learning methods for linear problems can be kernelized to solve nonlinear problems by representer theorem. Many learning methods admit this, such as SVMs (Support Vector machines [1, 2]), PCA (Principal Component Analysis [7]), LDA (Fisher linear Discriminant Analysis [8]) etc.

Quantitatively speaking, given an input samples set $T = \{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$ for sample $x_i \in \mathbb{R}^d$ and its label $y_i \in \{-1,1\}$, a kernel learning classification problem is to learn a classification function f in a reproduced kernel Hilbert spaces (RKHS) $\mathbb{H}$ corresponding to a kernel function $k : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ with a good generalized capacity. RKHS $\mathbb{H}$ has the reproducing property that admits $f(x) = \langle f, k(\cdot,x)\rangle_{\mathbb{H}}$ and $\langle k(\cdot,x), k(\cdot,z)\rangle_{\mathbb{H}} = k(x,z)$ for all $f \in \mathbb{H}$ and $x,z \in \mathbb{R}^d$. It may be very high dimensions even infinite dimensions, thus the learning problem on $\mathbb{H}$ can not be solved efficiently. Owing to the representer

theorem, the solution $f \in \mathbb{H}$ to the learning problem can be represented as

$$f(\cdot) = \sum_{j=1}^{m} \alpha_j k(\cdot, x_j). \tag{1}$$

This is a finite linear combination of the basic hypothesis in $\mathbb{H}$. Instead to find an optimal hypothesis f in $\mathbb{H}$, to solve the optimal combination coefficients α_j in (1) is a well-defined finite dimensional problem in $\mathbb{R}^m$.

Not all learning problems admit representer theorem. The first result on representer theorem is introduced in [9, 10], and generalized by [11]. A quantitative versions of the representer theorem are proven in [3, 12]. In [13] the representer theorem is generalized to matrix version with spectral regularization for collaborative filtering, a kind of multi-task learning algorithm. In [14], a necessary and sufficient condition for a learning problem with a differentiable regularizer to admit the representer theorem is given, and is also generalized to a learning problem with matrix samples as inputs [14]. Recently, [15] gives another kind of the necessary and sufficient conditions for a learning algorithm to be kernelizable in the case when the instances are vectors, asymmetric matrices and symmetric matrices, respectively.

1.2. Outline

This paper is organized as following: In Section 2, two related regularization models of SVMs are introduced and many types of loss function and their conjugate function are listed. In Section 3, a simple comparison between two models with least squares loss is given, where the computing stability and the computational complexity are discussed. In Section 4, reduced SVMs with different loss functions are analyzed and the corresponding Newton-type algorithms are discusssed,where SMW (Sherman-Morrison-Woodbury) identity [16] is introduced to decrease the computational complexity of Model 2, but cannot be used to decrease the complexity of Model 1. Section 5 gives some experimental results to support our points. Section 6 concludes the paper.

2. Two Regularization Models for SVMs

There are two mainly regularization forms for SVMs. The first model regularizes the classification function f on RKHS $\mathbb{H}$ named **Model 1** (**M1** for short) as follows:

$$\begin{aligned} \textbf{M1:} \quad & \min_{f\in\mathbb{H},\xi\in\mathbb{R}^m} \ \tfrac{\lambda}{2}\|f\|_{\mathbb{H}}^2 + \textstyle\sum_{i=1}^m L(\xi_i), \\ & s.t. \ \ y_i \langle f, k(\cdot, x_i)\rangle_{\mathbb{H}} + \xi_i = 1, i = 1, 2, \cdots, m, \end{aligned} \tag{2}$$

where λ is the regularization parameter, the loss function $L : \mathbb{R} \to \mathbb{R}_+ \cup \{+\infty\}$ has some typical forms listed in Table 1, $k : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is a kernel function

with the reproducing property. The common used kernel functions are Gaussian kernel $K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$ and polynomial kernel $k(x,z) = \left(x^\top z + 1\right)^d$ with parameter σ and d respectively.

The second regularization form of SVMs is called the generalized support vector machine (GSVM), which was first proposed in [17] with the hinge loss. In [18, 19, 20, 21], GSVM is used to design some kinds of algorithms with the squared hinge loss or the least squares loss. In GSVM model, only the combinations coefficients of the hypothesis f in (1) are norm-regularized in $\mathbb{R}^m$. A general form of GSVM fitting any loss function is given as following **Model 2** (**M2** for short):

$$\begin{array}{ll} \textbf{M2:} & \min_{\beta,\xi\in\mathbb{R}^m} \ \frac{\lambda}{2}\|\beta\|^2 + \sum_{i=1}^m L(\xi_i), \\ & s.t. \ y_i \sum_{j=1}^m \beta_j k(x_i,x_j) + \xi_i = 1, i = 1,2,\cdots,m, \end{array} \tag{3}$$

where we note the combination coefficients be β which may be different with the coefficients α in (1) solved by M1.

At the first glance, optimization problem of M1 (2) is not well-defined, since it will be solved in RKHS $\mathbb{H}$, a very high dimensional, even infinite dimensional, space. However, it can be converted to a finite dimensional problem by the representer theorem [11, 5, 22] or duality [2, 23, 24, 25, 26, 27, 28]. Both models are specified in the following two subsections.

2.1. Regularization Model on $\mathbb{H}$

M1 is the most popular model used in many research papers, such as [2, 5, 22, 23, 24, 25, 26, 27, 28] and the references therein. There are two popular techniques to simply M1 (2) to a well-defined optimization problem. One is the representer theorem and the other is the duality. Both techniques agree that the optimal hypothesis (classification function) is a finite combination of the basic functions in $\mathbb{H}$ as (1).

Duality [29] is used by many researchers to convert (2) as a finite dimensional optimization problem [1, 2, 23, 24, 25, 26, 27, 28], but it always needs a concrete loss function given in advance. Here we give a general dual form.

The Lagrangian dual of M1 in problem (2) is

$$\max_{\gamma} \min_{f\in\mathbb{H},\xi\in\mathbb{R}^m} \ \frac{\lambda}{2}\|f\|_{\mathbb{H}}^2 + \sum_{i=1}^m L(\xi_i) - \sum_{i=1}^m \gamma_i \left(y_i\langle f, k(\cdot,x_i)\rangle_{\mathbb{H}} + \xi_i - 1\right), \tag{4}$$

which is equivalent to

$$\min_{\gamma} \left\{ \ -\min_{f\in\mathbb{H}} \left\{ \tfrac{\lambda}{2}\|f\|_{\mathbb{H}}^2 - \sum_{i=1}^m \gamma_i y_i \langle f, k(\cdot,x_i)\rangle_{\mathbb{H}} \right\} - \sum_{i=1}^m \min_{\xi_i} \{L(\xi_i) - \gamma_i \xi_i\} - \boldsymbol{e}^\top \gamma \ \right\},$$

where $\boldsymbol{e}$ is an appropriate vector whose all entries are 1. Solving the first inner minimization problem above, we get $f(\cdot) = \frac{1}{\lambda}\sum_{i=1}^m \gamma_i y_i k(\cdot,x_i)$, and the second

inner minimization problem on ξ_i is substituted by the **conjugate function** [29] of loss function $L(\cdot)$, denoted as $L^*:\mathbb{R}\to\mathbb{R}_+\cup\{+\infty\}$, which is defined as

$$L^*(v):=\max_u\{uv-L(u)\}=-\min_u\{L(u)-uv\}.$$

Then Lagrangian dual (4) is simplified as the following optimization problem

$$\min_\gamma \frac{1}{2\lambda}\gamma^\top YKY\gamma-\boldsymbol{e}^\top\gamma+\sum_{i=1}^m L^*(\gamma_i), \tag{5}$$

where K is a symmetrical kernel matrix with its component $K_{i,j}=k(x_i,x_j)$, and Y is a diagonal matrix with $y=(y_1,y_2,\cdots,y_m)^\top$ as its diagonal elements. The duality relationship maintains the result (1) induced by the representer theorem. Table 1 lists some popular loss and their conjugate functions.

Table 1. Popular loss functions and their conjugate functions.

Loss Function	Conjugate function
Hinge: $L(u)=\max\{0,u\}$	$L^*(v)=\begin{cases}0, & 0\le v\le 1\\ +\infty, & others\end{cases}$
Huber: $L_\delta(u)=\begin{cases}\max\{0,u\}, & \lvert u\rvert>\delta\\ \frac{(u+\delta)^2}{4\delta}, & \lvert u\rvert\le\delta\end{cases}$	$L^*(v)=\begin{cases}\delta v(v-1), & 0\le v\le 1\\ +\infty, & others\end{cases}$
Logistic: $L_p(u)=\frac{1}{p}\log(1+\exp(pu))$	$L^*(v)=\begin{cases}\frac{1}{p}\log(1-v)^{(1-v)}v^v, & 0\le v\le 1\\ +\infty, & others\end{cases}$
Squared hinge: $L(u)=\frac{1}{2}\max\{0,u\}^2$	$L^*(v)=\begin{cases}\frac{1}{2}v^2, & v\ge 0\\ +\infty, & v<0\end{cases}$
p**-normed:** $L(u)=\frac{1}{p}\lvert u\rvert^p(1<p<\infty)$	$L^*(v)=\frac{1}{q}\lvert v\rvert^q,\ \frac{1}{p}+\frac{1}{q}=1$
Least squares: $L(u)=\frac{1}{2}u^2$	$L^*(v)=\frac{1}{2}v^2$
Absolute: $L(u)=\lvert u\rvert$	$L^*(v)=\begin{cases}0, & \lvert v\rvert\le 1\\ +\infty, & \lvert v\rvert>1\end{cases}$

As for the representer theorem, plugging (1) in (2) and eliminating the equalities constraints, we have

$$\min_{\alpha\in\mathbb{R}^m}\ \frac{\lambda}{2}\alpha^\top K\alpha+\sum_{i=1}^m L\left(1-y_i\sum_{j=1}^m\alpha_jK_{i,j}\right), \tag{6}$$

(6) is called primal SVM in [5] and [22], and some algorithms are given according to the different loss functions.

By the duality technique above, we can prove that the dual of problem (6) has the same form as (5). So (6) and is equivalent to (5) naturally but they have different computational stability since the small parameter λ appears different place.

2.2. *Regularization Model on $\mathbb{R}^m$ – GSVM*

GSVM model (3) also admits the learned classification function as $f(\cdot) = \sum_{j=1}^{m} \beta_j k(\cdot, x_j)$, and it can also be rewritten as the following unconstraint optimization problem

$$\min_{\beta \in \mathbb{R}^m} \quad \frac{\lambda}{2} \|\beta\|^2 + \sum_{i=1}^{m} L\left(1 - y_i \sum_{j=1}^{m} \beta_j K_{i,j}\right). \tag{7}$$

In [17], Mangasarian stated that the regularizer $\|\beta\|^2$ could be replaced by any norm or seminorm function $g(\beta)$. It can easy show that the dual of (3) and (2) has the same duality (5) if $\|\beta\|^2$ is replaced as $\|\beta\|_K^2 := \beta^\top K \beta$.

2.3. *The relationship of two regularization models*

Compared problem (6) and problem (7), it observes that the difference of M1 and M2 is only on the first item of objective function, which is called the regularizer. The former is induced from (2) where the regularizer is L_2 norm of f in RKHS $\mathbb{H}$, while the latter is regularized with L_2 norm of the coefficients β in $\mathbb{R}^m$. All of them can be specified as a optimization problem in finite Euclidean space $\mathbb{R}^m$ by the representer theorem or the duality.

3. Related Models with Least Squares Loss

In this section, the difference of two models is analyzed while the least squares loss function $L(u) = \frac{1}{2}u^2$ is used. At this situation, all models have the closed form solutions and their differences can be compared analytically in the computational stability view.

3.1. *The related models and their relationships*

On the one hand, as soon as the conjugate function of the least squares loss in Table 1 is plugged in the dual problem (5) induced by **M1**, we have

$$\min_{\gamma \in \mathbb{R}^m} \quad \frac{1}{2}\gamma^\top \left(\frac{1}{\lambda}K + \boldsymbol{I}\right)\gamma - y^\top \gamma. \tag{8}$$

Its output classification function is

$$f_\gamma(\cdot) = \frac{1}{\lambda}\sum_{i=1}^{m} \gamma_i^* y_i k(\cdot, x_i) \text{ with } \gamma^* = \lambda\left(K + \lambda \boldsymbol{I}\right)^{-1} y. \tag{9}$$

This model is called least squares Support Vector Machine (LS-SVM), which is first setup in [25] where their model has an extra equality constraint corresponding

to the offset of the classification function. There are numerous applications [30] on it because of its simplicity even though some researches [31] have shown that it has a little bit lower accuracy sometimes.

By the representer theorem, the optimization problem (6) with the least squares loss induced from **M1** is

$$\min_{\alpha\in\mathbb{R}^m} \quad \frac{1}{2}\alpha^\top\left(\lambda K + KK^\top\right)\alpha - y^\top K\alpha. \tag{10}$$

Its output classification function is

$$f_\alpha(\cdot) = \sum_{j=1}^{m} \alpha_j^* k(\cdot, x_j) \tag{11}$$

where α^* solves $(\lambda K + KK^\top)\alpha = Ky$.

On the other hand, with the least squares loss considered in (7), which is induced by **M2**, we have

$$\min_{\beta\in\mathbb{R}^m} \quad \frac{1}{2}\beta^\top\left(KK^\top + \lambda \boldsymbol{I}\right)\beta - y^\top K\beta + \frac{m}{2}. \tag{12}$$

The corresponding output classification is

$$f_\beta(\cdot) = \sum_{i=1}^{m} \beta_i^* k(\cdot, x_i) \tag{13}$$

where $\beta^* = \left(KK^\top + \lambda \boldsymbol{I}\right)^{-1} Ky$.

Model (12) is called Proximal Support Vector Machine (PSVM), first proposed in [21]. It is also generalized to deal with multi-classification problem in [32] and multi-surface classification problem in [33]. Recently, some new results based on it about nonparallel classification hyperplane are reported [34].

Let α^* be the solution of (10), γ^* be the solution of (8) and β^* be the solution of (12). Comparing the related models (10), (8) and (12), we have the following conclusion:

***Proposition** 3.1*
a) For **M1** with least squares loss, problem (10) induced by representer theorem may have multi-solutions (if K is singular) and problem (8) induced by duality has unique solution. However, their output classification functions are the same, namely $f_\alpha(\cdot) = f_\gamma(\cdot)$, and $\frac{1}{\lambda}\gamma^*$ is always one solution of (10).
b) For **M2** with least squares loss, the induced problem (12) always has a unique solution.

For a common kernel matrix K, generally it has $\sigma_{\max}(K) \geq 1 \geq \sigma_{\min}(K)$, where $\sigma_{\max}(K)$ is the maximum eigenvalue of K and $\sigma_{\min}(K)$ is the minimum eigenvalue of K. Then we have $\kappa(\lambda I + K) \leq \kappa(\lambda I + KK^\top) \leq \kappa(\lambda K + KK^\top)$, where $\kappa(A)$ is condition number of a matrix A.

***Proposition** 3.2*
From the computational view, the model derived by duality in (8) is the most stable one and model of GSVMs in (12) is ranked second, and the model derived by representer theorem in (10) is the worst one.

Thus for solving Ml, the optimization problem induced by duality technique is **more simple and stable** than the problem induced by representer theorem for a computational view. This explains why so many researcher focus on it [35, 27, 36, 37, 38, 24, 39]. On the other hand, we have

***Proposition** 3.3*
The advantage of problem (10) over problem (8) is that there may have a sparse solution while K is low rank or is approximated by a low rank matrix.

This is also very important. Such as, compressed sensing [40], a very popular research area, mostly focuses on studying the sparse solution of the system of linear equations or called sparse representation problem.

3.2. Disadvantage of duality models with reduced method

In Proposition 3.2, it shows that the model induced by duality is the most stable one when the whole training kernel matrix is available to train SVMs. However, while the samples number m is larger enough, it is impractical to get the whole kernel matrix to train SVMs based on duality. Fortunately, some practical results [41, 24] and the theory results [3] show that the optimal output function is always a sparsity linear combination of some basic function corresponding to a part of training samples. Namely, in (1), many combination coefficients are 0. Although this is not true when least squared loss is used, it is inspired researcher to consider some reduced methods to get a well approximated solution of SVMs [19, 31, 5, 27, 42] for large scale problems. In those methods, only some basic functions corresponding to a subset $J \subset M$ of input samples are chosen to combine the output classification function f as

$$f(\cdot) = \sum_{j \in J} \alpha_j k(\cdot, x_j), \tag{14}$$

where J is sub-index set random chosen from M with $|J| \leq 0.1|M|$ [19, 31, 42] or well-chosen from M with less cardinal [5, 43, 22].

Only training SVMs on the subset J is not a good choice unless it is just the optimal *support vectors* set, which is always not known in advance. The experimental results in [19] also illustrate that training SVMs only on a random subset is very worse than on the whole set. So a wise method for the reduced SVMs is plugged (14) in M1 (2) or M2 (3) with m losses considered in the objective function and m equality constraints, which is called RSVMs (Reduced SVMs) in [19, 31].

Solving this kind of reduced problem in primal is only a $|J|$ dimensional problem, while the corresponding dual is still m dimensional problem. This is a barrier if we design the quadratic convergence rate algorithms like Newton-type method on dual problem.

Of course there are some efficient algorithms at most with linear convergence based on dual problem [44, 45] such as SMO [24, 36], SVM$^{\text{light}}$ [23, 41]. And there are also some quadratic convergence rate algorithms [46, 28, 27], in which the whole kernel matrix K is pre-computed and factorized approximated as $K \approx GG^\top$ with a low rank G. In this paper, we focus on the Newton-type methods for RSVMs neither the pre-factorized kernel matrix nor the whole kernel matrix.

As a conclusion, training the problem in dual space is not a good choice for reduced SVMs by Newton-type method. So, we only consider the two types primal problems in the rest part of this work. One is regularized in $\mathbb{H}$ but converted to finite optimal problem (15) by (14). The other is the reduced version of GSVM (3) in the following (16).

4. Newton Methods for RSVM with Smooth Losses

In this section, we study training reduced SVMs with Newton-type methods for the primal problems with different loss functions, and the learning results satisfies (14) where with the index set $J \subset M$ is well-chosen or random chosen. Let $r = |J|$ be the reduced set size and it always sets $r \leq 0.1m$.

Its the reduced form of **M1** (6) is as following

$$\min_{\alpha_J \in \mathbb{R}^r} \frac{\lambda}{2} {\alpha_J}^\top K_{JJ} \alpha_J + \sum_{i=1}^{m} L\left(1 - y_i K_{iJ} \alpha_J\right). \tag{15}$$

where $\alpha_J \in \mathbb{R}^r$ is a sub-vector of α consisted by the elements of α in the subset J, K_{IJ} is a sub-matrix of K consisted by all the elements at the rows in subset I and columns in subset J.

And the reduced version of **M2** (GSVM) (7) is:

$$\min_{\beta_J \in \mathbb{R}^r} \frac{\lambda}{2} {\beta_J}^\top \beta_J + \sum_{i=1}^{m} L\left(1 - y_i K_{iJ} \beta_J\right). \tag{16}$$

The only difference between (15) and (16) is also the first item in their objective functions. For simplicity, two related reduced models are rewritten as:

$$\min_{z \in \mathbb{R}^r} \frac{\lambda}{2} z^\top B z + \sum_{i=1}^{m} L\left(1 - y_i K_{iJ} z\right). \tag{17}$$

where $B = K_{JJ}$, $z = \alpha_J$ for **M1** and $B = \boldsymbol{I}$, $z = \beta_J$ for **M2**.

4.1. Setup of the Newton-type algorithms

Here we train RSVMs according to **M1** and **M2** by Newton-type algorithms on different losses. The most popular and meaningful loss is *hinge loss* $L(u) = \max\{0, u\}$, but it is not differentiable and Newton-type methods do not work for it. Some smooth loss functions are adopted to approximate it, including *least squares loss* [21, 25], *squared hinge loss* [5, 46], *Huber loss* [2, 22, 47] and *logistic loss* [19, 42]. Their concrete forms are listed in Table 1.

Given a smooth loss in (17), smooth Newton algorithm or semi-smooth Newton [48] algorithm has quadratic convergence rate. It solves Newton equations to update the current solution iteratively. Specifically, at iteration t, for a given z^t, let $\xi_i^t = 1 - y_i K_{Ji}^\top z^t$ and $I^t = \{i \in M | L(\xi_i^t) > 0\}$. The Hessian matrix H^t and the gradient g^t of the objective function in problem (17) satisfy

$$H^t = \lambda B + K_{JI^t} \Lambda_{I^t}^t K_{JI^t}^\top, \tag{18}$$

$$g^t = \lambda B z^t - \sum_{i \in I^t} y_i L'(\xi_i^t) K_{Ji}, \tag{19}$$

where $\Lambda^t := diag(L_p''(\xi_1^t), \cdots, L_p''(\xi_m^t))$, $\Lambda_{I^t}^t := \Lambda_{I^t I^t}^t$ for short, $L'(\cdot)$ and $L''(\cdot)$ be the first and second derivatives of loss function $L(\cdot)$.

Noting $K_{iJ} z^t = y_i(1 - \xi_i^t)$, the corresponding Newton equation can be written as

$$\left(\lambda B + K_{JI^t} \Lambda_{I^t}^t K_{JI^t}^\top\right) z = \sum_{i \in I^t} y_i \left(L''(\xi_i^t)(1 - \xi_i^t) + L'(\xi_i^t)\right) K_{Ji}. \tag{20}$$

Let $\bar{z}$ be the solution of (20). If the full Newton step is acceptable, then the next $z^{t+1} = \bar{z}$, otherwise $z^{t+1} = (1 - \eta) z^t + \eta \bar{z}$, where η is chosen by an inexact or exact line-search scheme. Experiments show that the full step is often acceptable for those convex problems and the *backtracking Armijo line-search* is pretty well while the full step fails.

***Remark** 4.1*
Our Newton scheme (20) is equivalent to the traditional one $H^t d = -g^t$, where the new solution $z^{t+1} = z^t + \eta \bar{d}$ with Newton direction $\bar{d}$. The new version is consistent with the least squares loss case(see the following equations (21)) and is simple in computing the right hand side for most popular loss(see the right hand side of the following equations (22) and (24) for details).

For a kernel matrix $K \in \mathbb{R}^{m \times m}$, we always have $\kappa\left(\lambda \boldsymbol{I} + K_{JI^t} \Lambda^t K_{JI^t}^\top\right) \leq \kappa\left(\lambda K + K_{JI^t} \Lambda^t K_{JI^t}^\top\right)$. Roughly speaking, we conclude

***Proposition** 4.2*
The model (16) derived from M2 is **more stable** than the model (15) derived from M1 from the computational view.

4.2. Specification of algorithms with different smooth losses

Following, the specification forms of (20) are analyzed according to different smooth loss, including the least squares loss, the squared hinge loss, Huber loss and the logistic loss.

4.2.1. Least squares loss With the least squares loss $L(u) = \frac{1}{2}u^2$, Newton equation (20) is independent to α^t and the closed solution to (17) is obtained by solving the following system of linear equations (21):

$$\left(\lambda B + K_{JM}K_{JM}^\top\right) z \quad = \quad K_{JM}y. \tag{21}$$

If $B = K_{JJ}$ for M1, the coefficients matrix of z may be semi-positive definite but not positive definite. Thus there needs a ridge added to the coefficients matrix as [43] does for the paper [5]. However, if $B = \boldsymbol{I}$ for M2, the coefficients matrix of z is always positive definite. The performance of the classifiers corresponding to two models will be compared experimentally.

4.2.2. Squared hinge loss With the squared hinge loss $L(u) = \frac{1}{2}\max\{0,u\}^2$, the objective function of problem (17) is the piecewise quadratic functions. They have no closed form solutions, but the minimizers can be found by Newton method within finite iterations [49, 20, 5, 46]. Precisely the resulted method should be called semi-smooth Newton method, where the Newton equations (20) are simplified as

$$(\lambda B + K_{JI^t}K_{JI^t}^\top)z = K_{JI^t}y_{I^t}. \tag{22}$$

In [5, 43], they design a complicated procedure to iteratively update the Cholesky factorization of $\lambda B + K_{JI^t}K_{I^tJ}^\top$ with $B = K_{JJ}$ to reduced computational complexity, and the computational complexity of their algorithm per iteration is less than $O(mr^2)$. In their program [43], a ridge is added to the coefficients matrix to overcome the potential singularity for M1. For simplicity, in this paper we use

$$H^t := H^{t-1} + K_{JI_{in}^t}K_{JI_{in}^t}^\top - K_{JI_{out}^t}K_{JI_{out}^t}^\top, \tag{23}$$

to update the coefficients matrix of Newton equations (22) iteratively as [42, 46], where $H^{t-1} = \lambda B + K_{JI^{t-1}}K_{JI^{t-1}}^\top$. And then Newton equations (22) is solved by"\" operator in Matlab, where $I_{in}^t := \{i | i \in I^t, i \notin I^{t-1}\}$, $I_{out}^t := \{i | i \notin I^t, i \in I^{t-1}\}$. Since $|I^t| > |I_{in}^t| + |I_{out}^t|$ always holds [46], the computational complexity of this algorithm per iteration is also less than $O(|I^t|r^2)$, and very less than $O(mr^2)$. This scheme works better in the situation where the reduced set is randomly selected in advanced.

The only difference of two related models is the term of B in (22). GSVM model (M2) is more stable than M1 because its Hessian matrix is a definite positive

matrix all the time and an extra ridge is also added to the coefficients matrix of Newton equations (22) for M1. Their classification performance will be compared experimentally in Section 5.

4.2.3. Huber loss. Huber loss function [50] is defined as $L_\delta(u) := \begin{cases} \max\{0,u\} & if \ \ |u| \geq \delta, \\ \frac{(u+\delta)^2}{4\delta} & if \ \ |u| < \delta, \end{cases}$ for a given smooth parameter δ. It is clear that $\lim_{\delta\to 0} L_\delta(u) = \max\{0,u\}$. In [47], some variations and generalizations are given for the minimax problem.

With Huber loss, the objective function of problem (17) is also a piecewise quadratic function, then their minimizers can be found by Newton method within finite iterations, but the experimental results in Section 5 show that it always needs more iterations than the algorithm with the squared hinge loss. At iteration t, Newton equations (20) are simplified as

$$\left(\lambda B + \tfrac{1}{2\delta} K_{JI^t} K_{JI^t}^\top\right) z = \tfrac{1+\delta}{2\delta} K_{JI^t} y_{I^t} + K_{JI_+^t} y_{I_+^t}. \tag{24}$$

where $I^t = \{i \in M | |\xi_i^t| \leq \delta\}$ and $I_+^t = \{i \in M | \xi_i^t > \delta\}$. Its Hessian matrix $H^t = \lambda B + \frac{1}{2\delta} K_{JI^t} K_{JI^t}^\top$ can also be updated iteratively by the scheme similar as (23):

$$H^t := H^{t-1} + \tfrac{1}{2\delta}\left(K_{JI_{in}^t} K_{JI_{in}^t}^\top - K_{JI_{out}^t} K_{JI_{out}^t}^\top\right), \tag{25}$$

where $I_{in}^t := \{i | i \in I^t, i \notin I^{t-1}\}$, $I_{out}^t := \{i | i \notin I^t, i \in I^{t-1}\}$.

Tune the smoothing parameter δ. We need to tune the smoothing parameter δ. Setting a small δ to solve (24) is not a good choice because it may face a big condition number of Hessian matrix H^t. The difference between the hinge loss and Huber loss is $L_\delta(u) - \max\{x,0\} \leq \frac{\delta}{4}$, and the total approximate error is less than $\frac{|I^0|\delta}{4}$. If a proper smoothing precision ε is available, we can tune $\delta_{\min} = \frac{4\varepsilon}{|I^0|}$. At the beginning of the algorithm, we set a big δ_0 like $\delta_0 = 1$, and reduce it by $\delta_k := 0.1\delta_{k-1}$ while the current solution is good under some criterions (such as $\|g^t\| \leq 1$), and repeat the algorithm until $\delta_k \leq \max\{10^{-4}, \delta_{\min}\}$ and $\|g^t\| \leq \varepsilon$ for a given ε.

4.2.4. Logistical loss Logistical loss is also called the exponential entropy function, which is defined as $L_p(u) := \frac{1}{p}\log(1+\exp(pu))$. In [42], a stable form is given as $L_p(u) := \max\{u,0\} + \frac{1}{p}\log(1+\exp(-|pu|))$ to overcome any potential overflowing. With the Logistical loss $L_p(u)$, (20) is simplified as

$$\left(\lambda B + K_{JI^t} \Lambda_{I^t}^t K_{JI^t}^\top\right) z = K_{JI^t} \Lambda_{I^t}^t K_{JI^t}^\top z^t + \sum_{i \in I_+^t} y_i L_p'(\xi_i^t) K_{Ji}. \tag{26}$$

where $I_+^t := \{i \in M | L_p'(\xi_i^t) \geq \tau\}$ and $I^t := \{i \in M | L_p''(\xi_i^t) \geq \tau\}$ for a tiny number τ like 10^{-10}, $\Lambda^t := diag(L_p''(\xi_1^t), \cdots, L_p''(\xi_m^t))$ and $\Lambda_{I^t}^t := \Lambda_{I^t I^t}^t$. $L'(u)$ and $L''(u)$ are calculated as $L_p'(u) = \frac{\min\{1, \exp(pu)\}}{1+\exp(-p|u|)}, L_p''(u) = \frac{p\exp(-p|u|)}{(1+\exp(-p|u|))^2}$.

Tune the parameter p**.** In order to make the Newton method working well, we should set p moderately such as $p = 10$ at the beginning, then $p := 10p$ if $\|g^t\|$ is small and repeat the algorithm until $p = 10^4$ and $\|g^t\| \leq \varepsilon$ for a given ε.

4.3. SMW identity and the advantage of GSVM

Sherman-Morrison-Woodbury (SMW) identity [16]

$$(A + UDU^\top)^{-1} = A^{-1} - A^{-1}U\left(D^{-1} + U^\top A^{-1}U\right)^{-1}U^\top A^{-1}, \tag{27}$$

can be used to reduce the computational complexity for calculating $(A + UDU^\top)^{-1}$ while A^{-1} is very simple and $D^{-1} + U^\top A^{-1}U$ has a small size.

Based on the analysis above section, Hessian matrices of the corresponding problems always have the form $\lambda B + U^t \Lambda_{I^t}^t U^{t\top}$, where $U^t \in \mathbb{R}^{r\times|I^t|}$ and diagonal matrix $\Lambda_{I^t}^t \in \mathbb{R}^{|I^t|\times|I^t|}$. If $B = \boldsymbol{I}$ for solving M1 (GSVM), we can obtain the solution of Newton equation (20) by SMW identity (27) as

$$\lambda z = b^t - U^t\left(\lambda(\Lambda_{I^t}^t)^{-1} + U^{t\top}U^t\right)^{-1}U^{t\top}b^t \tag{28}$$

where b^t is the right hand side of (20). So while $|I^t| < r$, which always happens for some kinds problem that has a sparse solution, the solution of Newton equation (20) is obtained by (28) with the complexity $O(r|I^t|^2)$, less than $O(r^3)$, where the second part of (28) is computed from right to left. This trick is invalid if $B = K_{JJ}$ for solving the problem induced from M1 (if K_{JJ}^{-1} is per-calculated, the total cost for (28) is $O(r^2|I^t|^2)$ by SMW, very larger than that of M2). As a conclusion, we have

***Proposition** 4.3*
If $|I^t| < r$, Newton-type algorithm based on M2 (GSVM) has **less computational complexity** than the similar algorithm based on M1.

Next we will compare these two models on the classification performance experimentally.

5. Experimental Results

In this section, we perform some experiments to compare the two related models with Newton-type method, and four kinds of popular smooth loss functions are

considered. In Section 5.1, a nonlinearly separable example called the "tried and true" checkerboard dataset [51, 37, 38, 18, 19, 46] is selected to train classification surface with Newton-type methods based on two models, and in Section 5.2, some practical datasets from machine learning repositories [52] are adopted to evaluate the models. All the experiments are run on a Personal Computer with a Intel Core i3 2100 CPU and a maximum of 4Gbytes of memory available for all processes. The computer runs Windows 7 with Matlab 7.10.

5.1. Artificial Data with reduced methods

In this section, we give some sets of experiments on an artificial dataset to compare the performance of two kinds of regularization with reduced methods. A nonlinearly separable example called the "tried and true" checkerboard dataset, first given in [51], which has often been used to show the effectiveness of nonlinear kernel methods [37, 38, 18, 19, 46], is selected to train classification surface with Newton method based on two related models. This checkerboard dataset is generated by uniformly discretized the regions $[0,199]\times[0,199]$ to $200^2 = 40,000$ points, and is labeled two classes "White" and "Black" spaced by 4×4 grids. Training sets are random sampled from the 40,000 checkerboard data with different training sizes m, and the remainder of the points are left in the testing set. Kernel function is chosen as $k(x,y) = \exp(-\gamma\|x-y\|^2)$ with $\gamma = 0.001$. A ridge $\varepsilon \boldsymbol{I}$ with $\varepsilon = 10^{-8}$ is added to all Hessian matrices based on M1.

5.1.1. Comparing M1 and M2 with different smooth losses In this section, we focus on comparing the performance of two models with difference smooth loss functions. Two tables (Table 2 and 3) corresponding with the regularizer $\lambda = 0.1$ and $\lambda = 0.01$ are given, where the averaged test errors, averaged training time, averaged iterations of Newton step and averaged numbers of training samples satisfying $y_i f(x_i) \leq 1$ are listed with standard deviations. All the results are the mean value on 20 random trials, and the standard deviations are given in brackets.

From the results in Table 2 and Table 3, we can get the following conclusions:

- Firstly, the results show that Newton method is very efficient to solve this kind problems. The algorithms are stable for the problems with different size training data since the changes of iteration are small for every algorithms on different size problems. The training time is less than half a minute even the training data size as large as 25,000.
- On the test errors aspect, the results corresponding to M2 are always better than the results corresponding to M1 for the squared hinge loss, logistical loss and Huber loss. The only exception is on the least squares loss, where the test errors corresponding to M1 better than the results of M2, but they are all much worse than others, and we will specify it in Subsection 5.1.2.

Table 2. Comparisons of two regularization with different smooth loss functions. All the results are the mean value on 20 random trials, and the standard deviations are given in brackets. "LS", "SH", "Log" and "Hub" are "Least Squares loss", "Squared Hinge loss", "Logistical loss" and "Huber loss" for short respectively. "M1+LS" means that the model is based on M1 with least squares loss, and others have the similar meaning. ($\lambda = 0.1$)

Size→ Model↓	m=4,000 r=300	m=8,000 r=600	m=15,000 r=1000	m=20,000 r=1000	m=25,000 r=1,000
	Test error(%) on the test sets with $40000 - m$ samples.				
M1+LS	**2.92(0.39)**	**2.11**(0.31)	**1.46(0.27)**	**1.12(0.20)**	**1.05(0.12)**
M2+LS	3.41(0.41)	2.96(0.30)	1.90(0.28)	1.52(0.23)	1.11(0.27)
M1+SH	1.19(0.23)	0.55(0.12)	0.20(0.07)	0.11(0.04)	0.08(0.04)
M2+SH	**0.75(0.17)**	**0.22(0.08)**	**0.06(0.03)**	**0.04(0.02)**	**0.03(0.02)**
M1+Hub	1.80(0.24)	1.06(0.18)	0.53(0.11)	0.30(0.08)	0.22(0.08)
M2+Hub	**1.29(0.20)**	**0.38(0.12)**	**0.13(0.05)**	**0.09(0.03)**	**0.07(0.03)**
M1+Log	1.80(0.24)	1.06(0.18)	0.53(0.11)	0.30(0.08)	0.22(0.08)
M2+Log	**1.29(0.20)**	**0.38(0.12)**	**0.13(0.05)**	**0.09(0.03)**	**0.07(0.03)**
	Training time(s)				
M1+LS	0.04(0.01)	0.22(0.03)	**1.02(0.15)**	1.43(0.21)	**1.60(0.27)**
M2+LS	**0.03(0.01)**	**0.19(0.02)**	1.10(0.17)	**1.39(0.18)**	1.69(0.31)
M1+SH	0.22(0.01)	1.35(0.02)	5.31(0.07)	6.89(0.09)	8.48(0.11)
M2+SH	**0.21(0.01)**	**1.28(0.02)**	**5.22**(0.11)	**6.72**(0.10)	**8.47**(0.21)
M1+Hub	0.74(0.04	3.62(0.25)	12.90(0.66)	15.45(0.80)	18.26(0.87)
M2+Hub	**0.57**(0.05)	**2.60(0.25)**	**8.05(0.66)**	**10.70(0.79)**	**13.01(0.66)**
M1+Log	0.52(0.06)	2.58(0.40)	9.97(1.07)	11.36(0.68)	13.46(1.16)
M2+Log	**0.42(0.05)**	**2.14(0.12)**	**7.56(0.41)**	**9.74(0.61)**	**12.16(0.52)**
	Iterations of Newton step				
M1+LS	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)
M2+LS	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)
M1+SH	10.8(0.6)	12.8(0.4)	14.3(0.6)	15.0(0.1)	**15.5**(0.6)
M2+SH	**10.4(0.5)**	**11.8**(0.6)	**13.4**(0.7)	**14.7**(0.7)	15.7(0.7)
M1+Hub	**84.8(5.7)**	**86.8(9.0)**	**90.0(6.1)**	**89.5(7.9)**	**91.2(7.1)**
M2+Hub	131.8(15.5)	138.9(17.1)	155.6(18.0)	153.8(18.6)	151.8(12.8)
M1+Log	**46.3(4.5)**	**46.8(5.4)**	**49.6(5.4)**	**46.1(4.9)**	**45.5(5.7)**
M2+Log	54.4(8.2)	62.0(6.8)	64.3(11.8)	57.9(8.3)	60.2(12.5)
	Numbers of Support Vectors(training samples satisfied $y_i f(x_i) \leq 1$).				
M1+LS	**2,480(19)**	**4,877(20)**	**9,009**(34)	**11,865**(40)	**14,644**(39)
M2+LS	2,672(20)	5,254(28)	9,716(34)	12,922(39)	16,112(34)
M1+SH	701(17)	1,153(24)	1,757(20)	2,086(22)	2,398(19)
M2+SH	**484**(24)	**641(18)**	**987(15)**	**1,180**(28)	**1,316**(33)
M1+Hub	486(14)	809(18)	1,277(20)	1,573(22)	1,857(22)
M2+Hub	**336**(17)	**440(15)**	**488(11)**	**573(16)**	**658(14)**
M1+Log	486(13)	809(18)	1277(20)	1573(22)	1857(22)
M2+Log	**336**(17))	**440(15)**	**488(11)**	**573(16)**	**657(14)**

- On the training time and iterations aspects, the training time of M1 and M2 is comparable for least squares loss while there needs only one iteration. However, for other three loss, the algorithms based on M2 are always faster than the algorithms based on M1. Especially, for the Huber loss and logistic loss, the iterations of Newton step corresponding to M2 are often very longer than that of M1, but the training time is not. The reason is that SMW identity can be applied to M2 but cannot be applied to M1. Specifically, at the first several iterations, the cost to obtain the corresponding Newton direction for all algorithms is $O(r\max\{|I^t|^2, r^2\}) = O(r|I^t|^2)$ since $|I^t|$ in (18) is

Table 3. Comparisons of two regularization with different smooth loss functions. All the results are the mean value on 20 random trials, and the standard deviations are given in brackets. "LS", "SH", "Log" and "Hub" are "Least Squares loss", "Squared Hinge loss", "Logistical loss" and "Huber loss" for short respectively. "M1+LS" means that the model is based on M1 with least squares loss, and others have the similar meaning. ($\lambda = 0.01$)

Size→ Model↓	m=4,000 r=300	m=8,000 r=600	m=15,000 r=1000	m=2,0000 r=1000	m=25,000 r=1,000
	Test error(%) on the test sets with $40000 - m$ samples.				
M1+LS	**2.39(0.22**	1.67(**0.21**)	**0.89(0.19**)	**0.53(0.12**)	**0.37(0.08**)
M2+LS	3.48(0.31)	2.39(0.33)	1.71(0.23)	1.34(0.18)	1.23(0.17)
M1+SH	0.72(0.14)	0.23(0.05)	0.09(0.03)	0.06(0.02)	0.04(0.02)
M2+SH	**0.35(0.09**)	**0.11(0.03**)	**0.05(0.02**)	**0.04(0.02**)	**0.02(0.02**)
M1+Hub	1.01(0.18)	0.34(0.06)	0.22(0.07)	0.15(0.04)	0.11(0.05)
M2+Hub	**0.52(0.13**)	**0.19(0.05**)	**0.10(0.03**)	**0.07(0.03**)	**0.05(0.02**)
M1+Log	1.01(0.18)	0.34(0.06)	0.22(0.07)	0.15(0.04)	0.11(0.05)
M2+Log	**0.52(0.13**)	**0.19(0.05**)	**0.10(0.03**)	**0.07(0.03**)	**0.05(0.02**)
	Training time(s)				
M1+LS	0.03(0.01)	0.21(0.02)	0.93(**0.03**)	1.21(0.04)	1.47(**0.02**)
M2+LS	**0.03(0.01**)	**0.20(0.02**)	**0.90**(0.04)	**1.17(0.03**)	**1.45**(0.03)
M1+SH	0.23(0.01)	1.46(0.06)	5.69(0.09)	7.49(0.29)	9.17(0.35)
M2+SH	**0.23(0.01**)	**1.43(0.08**)	**5.43(0.06**)	**7.18(0.10**)	**9.13(0.20**)
M1+Hub	**0.76(0.05**)	3.79(0.20)	13.27(0.72)	15.72(0.69)	29.88(2.63)
M2+Hub	0.79(0.09)	**3.47**(0.29)	**10.45**(1.09)	**13.44**(0.91)	**17.01(1.09**)
M1+Log	0.73(0.17)	3.59(0.55)	13.17(1.59)	13.95(1.47)	16.52(2.47)
M2+Log	**0.48(0.04**)	**2.38(0.23**)	**9.24(1.26**)	**13.17(1.12**)	**16.44(2.23**)
	Iterations of Newton step				
M1+LS	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)
M2+LS	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)
M1+SH	14.9(**0.6**)	**16.8(0.5**)	**18.7(0.7**)	**20.6(0.8**)	**21.5(0.8**)
M2+SH	**13.4**(0.8)	16.8(1.3)	22.9(1.3)	24.4(1.6)	26.6(2.2)
M1+Hub	**114.7(9.0**)	**114.3(6.7**)	**116.5(8.7**)	**118.1(7.8**)	**120.2(21.6**)
M2+Hub	218.6(25.4)	264.1(26.0)	275.1(32.6)	273.1(24.2)	280.3(31.4)
M1+Log	**74.2**(20.0)	**72.3(11.2**)	**72.8(13.1**)	**61.5(8.1**)	**62.0(12.9**)
M2+Log	80.6(11.8)	96.2(14.6)	108.6(19.6)	113.5(16.2)	106.7(19.1)
	Numbers of Support Vectors(training samples satisfied $y_i f(x_i) \leq 1$).				
M1+LS	**2,328(19**)	**4,590**(29)	**8,552(34**)	**11,411(36**)	**14,269(28**)
M2+LS	2,604(24)	5,138(27)	9,622(41)	12,826(45)	16,074(56)
M1+SH	357(18)	594(16)	1,028(19)	1,284(15)	1,473(16)
M2+SH	**275**(20)	**322(12**)	**341(9**)	**391(11**)	**452(10**)
M1+Hub	228(14)	366(11)	528(11)	619(11)	703(12)
M2+Hub	**145(10**)	**154(7**)	**186(6**)	**224(6**)	**260(9**)
M1+Log	228(14)	366(11)	528(11)	619(11)	704(10)
M2+Log	**145(10**)	**154**(7)	**186(6**)	**224(6**)	**260(9**)

larger than the reduced size r; and after several iterations, we have $|I^t| < r$, hence the cost to solve Newton equations based on M2 can be reduced to $O(r|I^t|^2)$ by SMW identity, but the cost of the algorithms based on M1 is still $O(r\max\{|I^t|^2, r^2\}) = O(r^3)$.

- Algorithms based on M2 often have less number of training samples satisfied $y_i f(x_i) \leq 1$(low-confidence training samples) than those based on M1 have, where $f(\cdot)$ is the resulted classification function. The only exception is corresponding to least squares loss too, and we will specify it in Subsection 5.1.2 too.

- Compared four types of loss function, the squared hinge loss gets the best generalization errors in all smooth losses. Its training time has only a little longer than least squares loss which has the worst test errors. No matter how large the training set, there are a few iterations(less than $2\log(m)$) of Newton step needed.
- The advantages of the logistical loss and Huber loss are that they always get the least low-confidence training samples than others. Especially while M2 is applied, there always has a small number of low-confidence training samples on the resulted classification function. The reason is that those two losses can converge to hinge loss if $p \to +\infty$(or $\delta \to 0$), but the others losses are not. We also notice that Huber loss has no any advantages over the logistical loss, so we do not recommend the further researches on it.

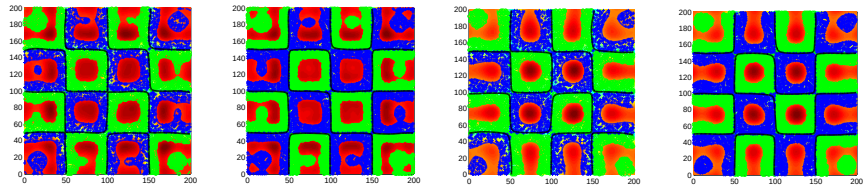
The results in Table 2 and Table 3 show that M2 nearly wins all aspects, so we can conclude that M2 has some advantages over M1. With the same parameters set, the algorithms based on M2 always faster than the algorithms based on M1. The former are always more stable in computing while the latter often need a ridge added to the Hessian matrix as [43] does for the paper [5] to keep the Newton direction well-defined.

Although the data in Table 2 and Table 3 shows that M1 is better than M2 with the *least squared loss* on the two aspects (Test errors and Number of support vectors), in Section 5.1.2 we will show that the resulted classification curves based on M1 have a strange behavior (drawback) but the curves based on M2 hasn't.

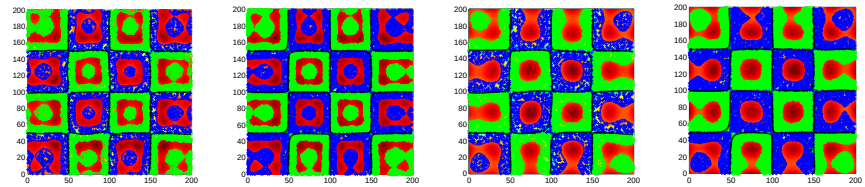
Based on all those experiments, M2 plus squared hinge loss is the best model to train reduced SVM with Newton method, in which the reduced set is random chosen. In Section 5.1.3, we will do more experiments to compare the well-chosen reduced scheme [43, 5] and rand-chosen scheme[19, 31], where M1 and M2 equipped with square hinge loss are considered only.

5.1.2. Drawback of M1 with the least squares loss The experiments in Section 5.1.1 show that M2 has many advantages over M1 almost all aspects except that it equipped with the least squares loss function. Here we give some plots to compare the M1 and M2 with least squares loss in details. M1 equipped with least squares loss is called LS-SVM [25, 30], and M2 equipped with least squares loss is called PSVM [21, 32, 33, 34]. Eight plots with different training data sizes and regularizer paremeters are given in Fig. 1. The regularizer parameters are set as $\lambda = 0.1$ for Fig. 1 (a)-(d) and $\lambda = 0.01$ for Fig. 1 (e)-(h).

Form Fig. 1, it shows that the classification lines of two models corresponding to same size training data are very similar, and hence the test accuracy are very similar too, but the high confidence areas (satisfying $y_i f(x) > 1$ corresponding to the red or dark red areas in the plots) are very different. For M2 (PSVM), the high confidence area is regular (See (c), (d), (g) and (h) in Fig. 1)–The more central the grid, the higher the confidence, and the exception happens only on the corners.



(a) M1($^{\lambda=0.1}_{m=8,000,r=600}$) (b) M1($^{\lambda=0.1}_{m=15,000,r=1000}$) (c) M2($^{\lambda=0.1}_{m=8000,r=600}$) (d) M2($^{\lambda=0.1}_{m=15,000,r=1000}$)



(e) M1($^{\lambda=0.01}_{m=8,000,r=600}$) (f) M1($^{\lambda=0.01}_{m=15,000,r=1000}$) (g) M2($^{\lambda=0.01}_{m=8,000,r=600}$) (h) M2($^{\lambda=0.01}_{m=15,000,r=1000}$)

Fig. 1. Comparison of M1 and M2 with least squares loss on checkerboard problems. The blank line is classification function $f(x) = 0$ and the blue and green lines are support lines corresponding to $f(x) = \pm 1$ respectively. The blue "+" and the green "o" are the training samples satisfying $y_i f(x_i) < 1$ corresponding to $y_i = \pm 1$ respectively. It shows that M1 (Left 4 plots) has some kind of behavior (drawback) on the classification curves.

But for M1 (LS-SVM), the high confidence area is strange on nearly every grids (See(a), (b), (e) and (f) in Fig. 1)–the central of the grids are not always in the high confidence area, especially for the case with the regularizer parameter $\lambda = 0.01$. It shows that M1 has some kind of strange behavior (drawback) on the classification curves.

5.1.3. Comparing two reduced models on random-chosen scheme and well-chosen scheme In this part, we perform some experiments to compare M1 and M2 on the random-chosen (RC) reduced set scheme as in RSVM [19, 31, 42] and well-chosen (WC) reduced set scheme as in [5]. Only the squared hinge loss is considered because it achieves the best performance in the former experiments. The codes for well-chosen reduced set algorithms are gotten from the site [43] which is based on M1. And we made a very minor modification to apply it to M2.

For well-chosen scheme, the reduced set is augmented from empty set iteratively where a well designed technique (see [5] in details) is abided to select some samples which will be put into the reduced set until the set size reached r_{WC}, where r_{WC} is set as 2% of the training size m as the default value in [43] but

limited by upper bound 200. For the comparison propose, we set the reduced size $r_{RC} = 2r_{WC}$ for random-chosen scheme.

In [43], the Matlab code designed by Chapelle is very efficient, where the fast rank 1 updating Cholesky factorization is used to update the Hessian matrix of the problem corresponding to the current solution while one *active sample* is added into or deleted from the active set $I := \{i \in M | 1 - y_i K_{iJ} z^t > 0\}$, and some powerful codes are designed to select the basic samples into reduced set. The total cost per iteration is less than $O(mr_{WC}^2)$. His code is more than 200 lines in Maltab. The total cost of the random-chosen scheme is $O(|I^t| r_{RC}^2)$ per iteration, which is also less than $O(mr_{RC}^2)$. Maybe the cost per iteration of random-chosen scheme is larger than that of well-chosen scheme since $r_{RC} = 2r_{WC}$. However our Matlab code for the random-chosen scheme with squared hinge loss is simple, only about 30 lines.

Some experimental results are reported in Table 4, where two different models with two different reduced schemes are compared on the squared hinge loss situation. The regularizer parameters $\lambda = 0.1$ and $\lambda = 0.01$ are considered.

Table 4. Comparison on two regularization with well-chosen(WC) reduced scheme and random-chosen(RC) reduced scheme with the squared hinge loss functions for different size of training problem($\lambda = 0.1$ (light part) and $\lambda = 0.01$ (dark part)). All the values are averaged on 20 random trials with the standard deviations in brackets.

Train size→	m=4,000	m=8,000	m=15,000	m=20,000	m=25,000
Models↓	r_{RC}=160 r_{WC}=80	r_{RC}=320 r_{WC}=160	r_{RC}=400 r_{WC}=200	r_{RC}=400 r_{WC}=200	r_{RC}=400 r_{WC}=200
	Test errors(%) on the test sets with $40000 - m$ samples.				
M1($^{\lambda=0.1}_{RC}$)	1.14(0.18)	0.50(0.09)	0.20(0.06)	0.13(0.05)	0.08(0.03)
M2($^{\lambda=0.1}_{RC}$)	**1.05(0.21)**	**0.27(0.08)**	**0.12(0.05)**	**0.07(0.03)**	**0.04(0.02)**
M1($^{\lambda=0.1}_{WC}$)	1.32(0.18)	0.61(0.09)	0.24(0.07)	0.16(0.06)	0.10(0.03)
M2($^{\lambda=0.1}_{WC}$)	1.30(0.24)	0.42(0.09)	0.16(0.05)	0.11(0.04)	0.07(0.02)
	Training time(s)				
M1($^{\lambda=0.1}_{RC}$)	0.07(0.00)	0.49(0.02)	1.41(0.02)	1.91(0.03)	2.34(0.03)
M2($^{\lambda=0.1}_{RC}$)	**0.07(0.01)**	**0.46(0.01)**	**1.31(0.03)**	**1.78(0.03)**	**2.20(0.04)**
M1($^{\lambda=0.1}_{WC}$)	0.47(0.06)	1.24(0.03)	2.56(0.05)	3.32(0.10)	4.06(0.09)
M2($^{\lambda=0.1}_{WC}$)	0.44(0.02)	1.25(0.04)	2.58(0.10)	3.33(0.11)	4.07(0.11)
	Test errors(%) on the test sets with $40000 - m$ samples.				
M1($^{\lambda=0.01}_{RC}$)	0.71(0.17)	0.21(0.05)	0.09(0.03)	0.06(0.03)	0.03(0.02)
M2($^{\lambda=0.01}_{RC}$)	**0.50(0.16)**	**0.10(0.03)**	**0.06(0.02)**	**0.04(0.02)**	**0.03(0.01)**
M1($^{\lambda=0.01}_{WC}$)	0.78(0.20)	0.28(0.06)	0.12(0.04)	0.07(0.03)	0.04(0.02)
M2($^{\lambda=0.01}_{WC}$)	0.75(0.18)	0.14(0.05)	0.07(0.03)	0.05(0.02)	0.03(0.02)
	Training time(s)				
M1($^{\lambda=0.01}_{RC}$)	0.08(0.00)	0.52(0.02)	1.51(0.02)	2.02(0.02)	2.49(0.03)
M2($^{\lambda=0.01}_{RC}$)	**0.07(0.00)**	**0.50(0.01)**	**1.46(0.04)**	**1.97(0.04)**	**2.43(0.05)**
M1($^{\lambda=0.01}_{WC}$)	0.46(0.01)	1.21(0.03)	2.40(0.07)	3.05(0.07)	3.78(0.09)
M2($^{\lambda=0.01}_{WC}$)	0.44(0.01)	1.17(0.04)	2.56(0.06)	3.34(0.08)	4.19(0.14)

From the results in Table 4, it shows that the random-chosen scheme is comparable with the well-chosen scheme under our settings: The training time of the random-chosen scheme is less than the training time of the well-chosen scheme while their test errors are comparable. It observes that the algorithms based on M2 with random-chosen scheme win all aspects: For every dataset, they are the faster ones and have the lowest test errors. Taking the simple implemental of the random-chosen scheme into consideration, we can conclude that M2 with random-chosen reduced scheme is the best model in the related models.

5.2. Benchmark data experiments comparison

Four large scale practical data sets of machine learning databases from the site of [52] are adopted to evaluate the related algorithms. These data are also appeared in [5]. For simplicity, Gaussian kernel function $k(x,y) = \exp(-\gamma\|x-y\|^2)$ with the different spread parameters γ is used for all datasets. Kernel spread parameters γ and regularizer parameters λ are roughly chosen by 10-fold cross-validation within $\gamma \in \{2^{-5}, 2^{-4}, \cdots, 2^5\}$ and $\lambda \in \{10^{-5}, 10^{-4}, \cdots, 1\}$. The details of the data sets and the corresponding selected parameters are listed as follows:

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 $\lambda = 1$ and $\gamma = 2^{-4}$.

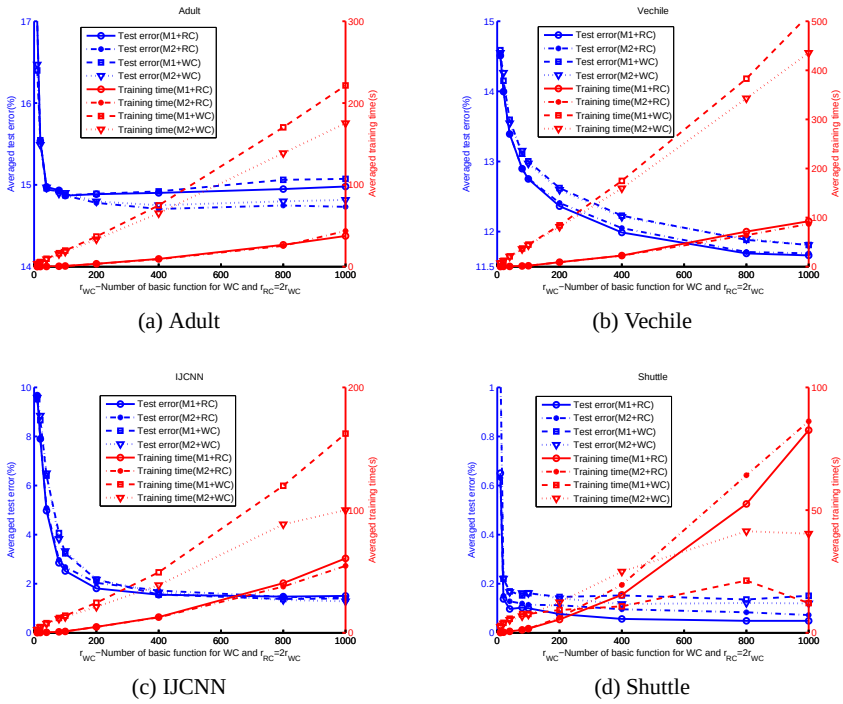
Shuttle—It is a multi-class data set with seven classes including 43,500 training examples and 14,500 test examples. Each example has 9 features. Here a binary classification problem is solved to separate class 1 from the rest, and the parameters used are $\lambda = 10^{-2}$ and $\gamma = 2^4$.

IJCNN—It has 49,990 training examples and 91,701 test examples. Each example is described by 22 features, and the parameters used are $\lambda = 10^{-4}$ and $\gamma = 2^{-1}$.

Vechile—It is the combined SensIT Vehicle in site [52]. It has 78,823 training examples and 19,705 test examples in three classes. Each example has 100 features. Here the binary classification problem is trained to differentiate class 3 from the rest, and the parameters are set as $\lambda = 10^{-2}$ and $\gamma = 2^{-2}$.

Firstly, we compare the random-chosen(RC) scheme with well-chosen(WC) scheme of [5] on the selected datasets, where the reduced size of well-chosen, r_{WC}, is set from 10 to 1000, and the reduced size of random-chosen is set as $r_{RC} = 2r_{WC}$. The plots of the test errors and training time according to different reduced sizes are given in Fig. 2, where the test error plots are according to left $y-$axis(blue) while training time plots are according to right $y-$axis(red). All the values are averaged on 10 trials.

It shows that the difference of the test errors on two schemes is small and it will be diminished further if the parameters(γ and λ) are set finely. However, the training time of two schemes varies. We can classify four datasets according to their test errors: Adult and Vechile are belonging to"hard" dataset(test error



(a) Adult (b) Vechile

(c) IJCNN (d) Shuttle

Fig. 2. Comparison of Well-Chosen(WC) Scheme and Random-Chosen(RC) Scheme on large benchmark data sets.

larger than 10%), IJCNN is the "easy" dataset(test error between 1% to 5%) and the Shuttle is the "very easy" dataset(test error less than 0.5%). For "hard" datasets, the training time of the algorithms based on RC scheme is less than $\frac{1}{4}$ of the training time based on WC scheme(See Figure 2(a) and (b)). For "easy" dataset, the training time of the algorithms based on RC scheme is about $\frac{1}{2}$ of the training time based on WC scheme(See Figure 2(c)). On the contrary, for "very easy" dataset, the training time of the algorithms based on WC scheme is less than the training time based on RC scheme(See Figure 2(d)). For "hard" datasets, the WC scheme of [5] may need to cost more time to select the proper basic functions, hence needs more time to convergence. At this situation, RC scheme is very efficient. Since the "very easy" dataset is scarce, we can conclude that RC scheme is more efficient than WC scheme in most of cases.

In this set of experiments, the difference between M1 and M2 is less. Next we perform more experiments to compare them further. We only consider the RC scheme.

Following we give the experimental result on those 4 data sets with reduced method to compare M1 and M2 further, where only the RC scheme is considered and the reduced size is set roughly as 4%m and is limited by upper bound 2000. The parameters γ and λ are set as above. The results in Table 5 are averaged on 10 random chosen reduced set. The averaged test errors and the training time are given with their standard deviations in brackets.

Table 5. Experimental results on the 4 benchmark data sets from repository [52]. All the values are averaged on 20 random trials with the standard deviations in brackets.

Data Set	Adult	Shuttle	IJCNN	Vechile
	Test error(%)			
M1+RC	14.79(0.08)	**0.07**(0.005)	1.88(0.06)	11.63(0.07)
M2+RC	**14.76**(0.06)	0.12(0.007)	**1.27**(0.18)	**11.61**(0.05)
	Training time(s)			
M1+RC	18.8(0.3)	**38.9**(1.0)	61.5(2.6)	91.9(5.6)
M2+RC	**18.2**(0.5)	45.1(1.7)	**48.9**(1.8)	**85.4**(2.5)

From the data in Table 5, M2 has a few advantages over M1. The algorithms based on M2 are also faster than the algorithms based M1, and the test errors achieved by M2 are better than those achieved by M1 on most data sets. The only exception is on the "easiest" data set "Shuttle" whose test error will be less than 0.1%.

6. Conclusions

There are two main regularization models of SVMs. One, listed as M1 in this paper, is the most popular model where the classification function is norm-regularized in a reproduced kernel Hilbert space. The other, named as M2 in this paper, is GSVM, where only the coefficients of the classification function is norm-regularized in a Euclidean space $\mathbb{R}^m$. All of them are converted to m dimension optimization problem by the duality or the representer theorem. In this paper, we study the difference of two models, where the quadratical convergence Newton algorithms are used to train the models with difference loss functions in primal.

The experimental results in Section 5 reveal that, M2 wins M1 on nearly all aspects, and the classification plots in Figure 1 also show that LS-SVM induced from M1 has some kind of drawback on the classification curves while PSVM induced from M2 has not. It also observes that the random-chosen reduced set scheme [19, 31, 42]is comparable with or sometimes better than the well-chosen reduced set scheme [5] for reduced SVMs with squared hinge loss.

As a conclusion, our studies support that M2 have more advantages over M1, such as simple in computing the Hessian matrix, stable in solving the Newton

direction and fast the algorithm etc. All of those reveal that if training SVMs with reduced method, GSVMs with the random-chosen reduced set are the better choice for common users. This work gives a good explanation of GSVM and is valuable to extend the using of GSVM.

Acknowledgement

This work was supported partly by NNSFC under Grant No. 61072144, 61179040, 61173089, 11101321 and 11101322.

REFERENCES

1. V. N. Vapnik. An overview of statistical learning theory. *IEEE trans on Neural Network*, 10(5):988–999, 1999.
2. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 2000.
3. I. Steinwart. Sparseness of Support Vector Machines. *Journal of Machine Learning Research*, 4:1071–1105, 2003.
4. I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, 2008.
5. S. S. Keerthi, O. Chapelle, and D. Decoste. Building Support Vector Machines with reduced classifier complexity. *J. of Machine Learning Research*, 7:1493–1515, 2006.
6. I. Steinwart, D. Hush, and C. Scovel. Training SVMs without offset. *Journal of Machine Learning Research*, 12:141–202, 2011.
7. B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, 1998.
8. S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. R. Müller. Fisher discriminant analysis with kernels. In E. Wilson Y. H. Hu, J. Larsen and S. Douglas, editors, *Proc. NNSP'99*, pages 41–48. IEEE, 1999.
9. G. S. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1970.
10. G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis Application*, 33:82–95, 1971.
11. Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 416–426. Springer, 2001.
12. E. De Vito, L. Rosasco, A. Caponnetto, M. Piana, and A. Verri. Some properties of regularized kernel methods. *Journal of Machine Learning Research*, 5:1363–1390, 2004.
13. J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *J. of Machine Learning Research*, 10:803–826, 2009.
14. A. Argyriou, C. A. Micchelli, and M. Pontil. When is there a representer theorem? Vector versus matrix regularizers. *J. of Machine Learning Research*, 10:2507–2529, 2009.
15. Manfred K. Warmuth, Wojciech Kotłowski, and Shuisheng Zhou. Kernelization of matrix updates, when and how? In *Proceedings of the 23rd international conference on Algorithmic Learning Theory*, ALT'12, pages 350–364, Berlin, Heidelberg, 2012. Springer-Verlag.
16. G. H. Golub and C. F. V. Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, Maryland, 1996.
17. O. L. Mangasarian. Generalized Support Vector Machine. In A. J. Smola, P. Bartlett, B. Schökopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146. MIT Press, 2000.

18. Y.-J. Lee and O. L. Mangasarian. SSVM: A smooth Support Vector Machine for classification. *Computational Optimization and Applications*, 20(1):5–22, 2001.
19. Y.-J. Lee and Olvi L. Mangasarian. RSVM: Reduced Support Vector Machines. In *CD Proceedings of the SIAM International Conference on Data Mining*, pages 1–17, Chicago, April 2001. SIAM.
20. O. L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):1155–1178, 2002.
21. G. Fung and O. L. Mangasarian. Proximal Support Vector Machine classifiers. In F. Provost and R. Srikant, editors, *Proceedings KDD-2001: Knowledge Discovery and Data Mining, August 26-29, San Francisco, CA*, pages 77–86, New York, 2001. ACM.
22. O. Chapelle. Training a Support Vector Machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.
23. T. Joachims. *SVM^{light}, Support Vector Machine*, 1998.
24. J. C. Platt. Fast training of Support Vector Machines using Sequential Minimal Optimization. In B. Schölkopf, C. J. Burges, and A. J. Smola, editors, *Advances in Kernel Method-Support Vector Learning*, pages 185–208. MIT Press, 1999.
25. J. Suykens and J. Vandewalle. Least square Support Vector Machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
26. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 2000.
27. S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.
28. M. C. Ferris and T. S. Munson. Semismooth Support Vector Machines. *Mathematical Programming*, Ser. B-101:185–204, 2004.
29. S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 7th edition, 2009.
30. J. A. K. Suykens, T. V. Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least squares Support Vector Machines*. World Scientific, River Edge, NJ, 2002.
31. K.-M. Lin and C.-J. Lin. A study on reduced Support Vector Machines. *IEEE Trans. on Neural Networks*, 14(6):1449–1459, 2003.
32. Glenn M. Fung and O. L. Mangasarian. Multicategory proximal Support Vector Machine classifiers. *Mach. Learn.*, 59(1-2):77–97, 2005.
33. O. L. Mangasarian and E. W. Wild. Multisurface proximal Support Vector Machine classification via generalized eigenvalues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):69–74, 2006.
34. S. Ghorai, S. J. Hossain, A. Mukherjee, and P. K. Dutta. Newton's method for nonparallel plane proximal classifier with unity norm hyperplanes. *Signal Processing*, 90(1):93–104, 2010.
35. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2:1–27, 2011.
36. S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.
37. O. L Mangasarian and D. R. Musicant. Successive overrelaxation for Support Vector Machines. *IEEE Transactions on Neural Networks*, 10(5):1032–1037, 1999.
38. O. L. Mangasarian and D. R. Musicant. Lagrangian Support Vector Machines. *Journal of Machine Learning Research*, 1:161–177, 2001.
39. Shuisheng Zhou, Hongwei Liu, Feng Ye, and Lihua Zhou. A new iterative algorithm training SVM. *Optimization Methods and Software*, 24(6):913–932, 2009.
40. Jon Dattorro. *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing USA, 2012.
41. T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, 1999.
42. Shuisheng Zhou, Jiangtao Cui, Feng Ye, Hongwei Liu, and Qiang Zhu. New smoothing SVM algorithm with tight error bound and efficient reduced techniques. *Computational Optimization and Applications*, 56(3):599–618, 2013. DOI 10.1007/s10589-013-9571-6.

43. O. Chapelle. Support vector machines in the primal, 2006.
44. C.-J. Lin. On the convergence of the decomposition method for Support Vector Machines. *IEEE Trans. on Neural Networks*, 12:1288–1298, 2001.
45. C.-J. Lin. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Trans. on Neural Networks*, 13:248–250, 2002.
46. Shuisheng Zhou, Hongwei Liu, Lihua Zhou, and Feng Ye. Semismooth Newton Support Vector Machine. *Pattern Recognition Letters*, 28:2054–2062, 2007.
47. Feng Ye, Hongwei Liu, Shuisheng Zhou, and Sanyang Liu. A smoothing trust-region Newton-CG method for minimax problem. *Applied Mathematics and Computation*, 199(2):581–589, 2008.
48. L. Qi and D. Sun. A survey of some nonsmooth equations and smoothing Newton methods. In A. Eberhard, B. Glover, R. Hill, and D. Ralph, editors, *Progress in Optimization*, volume 30 of *Applied Optimization*, pages 121–146, Dordrecht, 1999. Kluwer Academic Publishers.
49. J. Sun. On piecewise quadratic Newton and trust region problems. *Mathematical programming*, 76:451–467, 1997.
50. P. J. Huber. *Robust Statistics*. John Wiley, New York, 1981.
51. T. K. Ho and E. M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 880–885, Vienna, Austria, 1996.
52. Rong-En Fan and Chih-Jen Lin. LIBSVM Data, 2010.