Pattern Recognition Letters 31 (2010) 1944–1951

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Sparse learning for support vector classification

Kaizhu Huang [a,*], Danian Zheng [b], Jun Sun [b], Yoshinobu Hotta [c], Katsuhito Fujimoto [c], Satoshi Naoi [c]

[a] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[b] Fujitsu R&D Center Co. Ltd., Beijing, China
[c] Fujitsu Laboratories Ltd., Kawasaki, Japan

ARTICLE INFO

Article history:
Received 4 April 2007
Available online 1 July 2010
Communicated by T.K. Ho

Keywords:
Sparse representation
Implementations of L_0-norm
Regularization term
Support vector machine
Kernel methods

ABSTRACT

This paper provides a sparse learning algorithm for Support Vector Classification (SVC), called Sparse Support Vector Classification (SSVC), which leads to sparse solutions by automatically setting the irrelevant parameters exactly to zero. SSVC adopts the L_0-norm regularization term and is trained by an iteratively reweighted learning algorithm. We show that the proposed novel approach contains a hierarchical-Bayes interpretation. Moreover, this model can build up close connections with some other sparse models. More specifically, one variation of the proposed method is equivalent to the zero-norm classifier proposed in (Weston et al., 2003); it is also an extended and more flexible framework in parallel with the Sparse Probit Classifier proposed by Figueiredo (2003). Theoretical justifications and experimental evaluations on two synthetic datasets and seven benchmark datasets show that SSVC offers competitive performance to SVC but needs significantly fewer Support Vectors.

1. Introduction

Support Vector Machines (SVM) (Vapnik, 1995) are well-known learning algorithms covering many interesting topics, such as classification (Burges, 1998), regression (Smola and Schölkopf, 2004), novelty detection (Schölkopf et al., 2000), and density estimation (Vapnik and Mukherjee, 1999). A number of extensions, e.g. (Lanckriet et al., 2002; Huang et al., 2008a,b), have been developed for SVMs in various directions. This paper will only focus on Support Vector Classification (SVC). Many researchers have pointed out that the number of Support Vectors (SV) of SVC is usually large and this results in a substantially slower classification speed than many other approaches. How to reduce the number of SVs but without loss of generalization performance becomes a significant problem both theoretically and practically.

Burges (1996) proposed a reduced set method, which computes an approximation to the SVC decision rule in terms of a reduced set of vectors. A more comprehensive introduction to the reduced set framework, e.g., the reduced set selection and the reduced set construction, can be found in (Schölkopf et al., 1999). The reduced set method is computationally expensive and the local extremum problem exists, although an improvement to its reduction process was given in (Nguyen and Ho, 2005). Downs et al. (2002) proposed a method to recognize and delete some unnecessary SVs that are linearly dependent on other SVs in the feature space, but the reduction is not obvious in order to avoid any possible decline in accuracy. Li and Zhang (2006) presented a method to repeat training SVC on a gradually reduced training set until the decline in the training accuracy is not acceptable or the number of SVs stops decreasing.

The previous methods obtain a sparse decision function by finding an approximation to the solution of SVC or training SVC on the nested subsets of the training set. Unlike them, this paper proposes a new algorithm called Sparse Support Vector Classification (SSVC), which imports an L_0-norm regularization term of parameters into the primal optimization problem and iteratively trains it on the training set until it converges to a highly sparse solution. The L_0-norm regularization term has a hierarchical-Bayes interpretation and it automatically sets the irrelevant parameters exactly to zero.

The proposed novel model is important in the sense that it has various connections with other existing models. More specifically, one variation of the proposed method is equivalent to the zero-norm classifier proposed in (Weston et al., 2003). It is also an extended and more flexible framework in parallel with the Sparse Probit Classifier (SPC) proposed by Figueiredo (2002, 2003); the proposed framework is a general method that is not only applicable for classification, but also can be easily used for feature selection, regression, and kernel density estimation. In the literature, the Relevance Vector Machine (RVM) (Tipping, 2000, 2001) has also been proposed as an extremely sparse model for both regression and classification. As we show in the experiments, the proposed novel model achieves comparable with or even better sparseness than RVM, but with a considerably faster training speed.[1] On the other

* Corresponding author. Fax: +86 10 62545671.
E-mail address: kzhuang@nlpr.ia.ac.cn (K. Huang).

[1] A sequential adding and deleting technique is proposed to speed up RVM (Tipping and Faul, 2003) However, only synthetic data are used to validate the performance.

0167-8655/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2010.06.017

hand, Fung et al. (2002) designed a novel loss function, "pound" function, which is a combination of the L_1-norm and the step function measuring both the magnitude and the presence of any error. The authors then approximated it by a smooth exponential function. The loss function can reduce kernel data dependence and lead to a minimal kernel classifier. However, The model minimizes an approximating function rather than the true L_0-norm term. In this sense, it may not be the true minimal kernel classifier.

The rest of this paper is organized as follows. Section 2 briefly reviews the SVC algorithm. The proposed Sparse Support Vector Classification is described in Section 3. The framework, the implementations, the practical optimization issues, and the theoretical connections with other models will be discussed in turn in this section. Experimental results on some synthetic and real datasets are reported in Section 4. Finally, the conclusion is given in Section 5. The source codes for the proposed algorithm and the comparison approaches are implemented and publicly available at <http://www.enm.bris.ac.uk/staff/xkh/>.

2. Support vector classification

Let us briefly outline the SVC algorithm first. Suppose we are given empirical data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$ with input pattern $\boldsymbol{x}_i \in \mathcal{X}$ and output label $y_i \in \{\pm 1\}$. SVC finds a linear hyperplane $f(\boldsymbol{x}) = \boldsymbol{w}^T\Phi(\boldsymbol{x}) + b (\boldsymbol{w} \in \mathcal{F}, b \in \mathbb{R})$ to separate the positive from the negative examples with the largest soft-margin, where $\Phi(\cdot) : \mathcal{X} \to \mathcal{F}$ denotes a nonlinear mapping from the input space $\mathcal{X}$ into a higher dimensional feature space $\mathcal{F}$. To construct this optimal hyperplane, one solves the following primal problem:

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C1^T\xi \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}^T\Phi(\boldsymbol{x}_i) + b\right) + \xi_i \geqslant 1, \quad \xi_i \geqslant 0,\ \forall i,\ 1 \leqslant i \leqslant l, \end{aligned} \tag{1}$$

where ξ_i are slack variables, $\mathbf{1}$ is a column vector with all the elements equal to 1, $\xi = (\xi_1, \xi_2, \ldots, \xi_l)^T$, and C is a trade-off constant between the margin and the empirical error. This further leads to the dual problem:

$$\begin{aligned} \min_{\alpha} \ & \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{y}^T\boldsymbol{\alpha} = 0, \quad 0 \leqslant \alpha_i \leqslant C,\ \forall i,\ 1 \leqslant i \leqslant l, \end{aligned} \tag{2}$$

where $\boldsymbol{H}$ denotes a symmetric matrix with elements $h_{ij} = y_iy_jk(\boldsymbol{x}_i,\boldsymbol{x}_j)$, $k(\boldsymbol{x}_i,\boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j)$ describes a kernel function, $\boldsymbol{\alpha}$ represents a vector with elements α_i, and $\boldsymbol{y}$ is a vector defined as $(y_1, y_2, \ldots, y_l)^T$. The decision surface then takes the form

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} y_i\alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b. \tag{3}$$

Because the coefficients α_i are Lagrange multipliers, many of them will be equal to zero in the final solution. Those patterns $\boldsymbol{x}_i$ associated with non-zero coefficients α_i are called Support Vectors. The time taken for (3) to predict a class label of a new pattern is proportional to the number of SVs. From the optimization, it can be verified that the training patterns lying in the margin zone $f(\boldsymbol{x}) \pm 1$, and those patterns outside the margin zone but wrongly classified are all SVs. Hence the SVs of SVC are often redundant, especially for a large non-separable training set. To speed up the classification process, it is necessary to reduce the number of SVs.

3. Sparse support vector classification

In this section, we describe the framework of the novel Sparse Support Vector Classification. We first introduce the background on how to achieve the sparse representation. We then provide the framework of achieving sparseness. The model definition of SSVC, the associated implementations, and the optimization techniques will be presented in turn. Finally, we show the connections of our model with other methods.

3.1. Sparse representation

From the dual representation theory, the weight vector can be expressed as $\boldsymbol{w} = \sum_{i=1}^{l}\alpha_i\Phi(\boldsymbol{x}_i)$, where $\alpha_i \in \mathbb{R}$. Note that we do not require the coefficients α_i to be Lagrange multipliers anymore. Then the decision surface (3) can be rewritten as

$$f(\boldsymbol{x}) = \sum_{i=1}^{l}\alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + \alpha_0 = \boldsymbol{\alpha}^T k(\boldsymbol{x}, \cdot), \tag{4}$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_l)^T$ and $k(\boldsymbol{x},\cdot) = (1, k(\boldsymbol{x},\boldsymbol{x}_1), \ldots, k(\boldsymbol{x},\boldsymbol{x}_l))^T$.

To obtain a sparse expansion of kernel terms, i.e., to make most of the coefficients α_i exactly equal to zero, one possible way is to exploit an L_p-norm ($p = 2$, 1, or 0) regularization term.

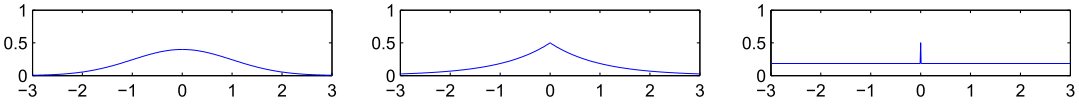
As we know, the L_2-norm $\|\boldsymbol{\alpha}\|_2^2 = \sum_i\alpha_i^2$ assumes a zero-mean Gaussian prior $p(\alpha_i) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\alpha_i^2}{2}\right)$ over each coefficient α_i as shown in Fig. 1(a), which leads to many coefficients approaching to zero but not exactly equal to zero. Hence it yields a non-sparse model. The L_1-norm $\|\boldsymbol{\alpha}\|_1^1 = \sum_i|\alpha_i|$ assumes a zero-mean Laplacian prior $p(\alpha_i) = \frac{1}{2}\exp(-|\alpha_i|)$ over each coefficient α_i as shown in Fig. 1(b), which leads that many coefficients are exactly equal to zero. However, the models achieved via L_1-norm are usually not sparse enough and there still exist some redundant coefficients. Later experimental results also validate this point. In the extreme case, the L_0-norm, defined as $\|\boldsymbol{\alpha}\|_0^0 = \sum_i I_{\{\alpha_i\neq 0\}}(\alpha_i)$, delivers an idea approach for achieving the sparsity. Here I is an indicator function, i.e., if $\alpha_i \in \{\alpha_i \neq 0\}$, then $I = 1$, otherwise $I = 0$. The L_0-norm assumes a zero-mean Pulse prior $p(\alpha_i) \propto \exp(-I_{\{\alpha_i\neq 0\}}(\alpha_i))$ over each coefficient α_i, which is shown in Fig. 1(c). It is straightforward that L_0-norm should lead to a highly sparse model. But L_0-norm also introduces a great difficulty in optimizing a noncontinuous problem.

3.2. Framework of achieving sparseness

In this section, we describe the framework on how to implement the L_0-norm in order to achieve the sparseness. In the following, an iteratively reweighted learning algorithm is proposed to implement L_0-norm in an asymptotical way. At the tth iteration, one optimizes the following problem

$$\boldsymbol{\alpha}_{(t+1)} = \arg\min_{\alpha} Q(\boldsymbol{\alpha}|\boldsymbol{\alpha}_{(t)}) = l(\alpha) + C_\alpha\frac{1}{2}\boldsymbol{\alpha}^T\mathrm{diag}(\boldsymbol{\lambda}_{(t)})\boldsymbol{\alpha}, \tag{5}$$

where $Q(\boldsymbol{\alpha})$ is a predefined objective function, $l(\boldsymbol{\alpha})$ is a loss function, e.g., a hinge loss function used in the standard SVC or an ϵ-insensitive



(a) L_2-norm, Gaussian prior (b) L_1-norm, Laplacian prior (c) L_0-norm, Pulse prior

Fig. 1. The distribution assumptions $p(\alpha_i)$ for each coefficient α_i in the L_p-norm.

loss function used in the Support Vector Regression, C_α represents a trade-off constant, and diag$(\lambda_{(t)})$ is defined as diag$\left(\frac{1}{\alpha_{1,(t)}^2},\ldots,\frac{1}{\alpha_{l,(t)}^2}\right)$.

The above learning can be interpreted as an Expectation–Maximization algorithm from a two-level hierarchical-Bayes model in (Figueiredo, 2002, 2003). That is, in level-1, each parameter α_i has a Gaussian prior $p(\alpha_i|\tau_i)=\frac{1}{\sqrt{2\pi\tau_i}}\exp\left(-\frac{\alpha_i^2}{2\tau_i}\right)$; in level-2, each variance τ_i has a Jeffrey's noninformative hyper-prior $p(\tau_i)\propto\frac{1}{\tau_i}$, for $\tau_i>0$. This interpretation guarantees that the sequence converges to a stationary one $\boldsymbol{\alpha}^*$. As $t\to+\infty$, $\boldsymbol{\alpha}_{(t+1)}^T\text{diag}(\lambda)_{(t)}\boldsymbol{\alpha}_{(t+1)}$ converges to the L_0-norm $\sum_i I_{\{\alpha_i^*\neq 0\}}(\alpha_i^*)$ regularization term asymptotically (Zheng et al., 2006). If $|\alpha_{i,(t)}|$ is small enough, the penalty or weighted term $\frac{1}{\lambda_{i,(t)}}\alpha_i^2$ in (5) always sets $\alpha_{i,(t+1)}$ to zero.

It is noted that, although our model is related to Figueiredo (2002, 2003), there are major differences between them. In a word, Figueiredo (2002, 2003) derived their classifier only within the Bayesian framework, while our model presents a more general framework that can be exploited to achieve sparseness in many kernel methods. By defining a suitable objective function Q, the proposed sparse framework (5) is able to attain feature selection, sparse regression, and sparse kernel density estimation. We will detail this discussion shortly in Section 3.5.2.

Before we summarize the above L_0-norm algorithm, we first present the following optimization problem:

$$\begin{aligned}&\min_{\boldsymbol{\alpha}}\quad\sum_{i=1}^{l}\lambda_i\alpha_i^2+l(\boldsymbol{\alpha})\\&\text{s.t.}\quad g_k(\boldsymbol{\alpha})\leqslant 0,\quad k=1,\ldots,Nc,\end{aligned}\tag{6}$$

where the kernel expansion $f(\boldsymbol{x})=\boldsymbol{\alpha}^Tk(\boldsymbol{x},\cdot)$, $l(\boldsymbol{\alpha})$ is a loss function, $g_k(\boldsymbol{\alpha})$ is a set of constraints, Nc is the constraint number. We summarize the above L_0-norm algorithm called Implementation A as follows:

Implementation A of L_0-norm

1. Set $\lambda=(1,\ldots,1)^T$.
2. Solve the optimization problem (6) and get the solution $\bar{\boldsymbol{\alpha}}$.
3. Update λ by $\lambda_i=\begin{cases}\frac{1}{\bar{\alpha}_i^2} & \text{if}\quad|\bar{\alpha}_i|>\epsilon,\\ \frac{1}{\epsilon^2} & \text{otherwise.}\end{cases}$
4. Go back to 2 until convergence.

We briefly explain the above implementation in the following. At step 1, all variables λ_i are initialized to *1* so that it provides a fair opportunity for each training sample to become a SV. At step 2, the initial feasible solution of the current iteration can be set to the solution of the previous iteration, which is already a sub-optimal solution for the optimization problem. Hence, the training algorithm converges upon the optimal solution after only a small number of iterations, and the training time is mainly spent at the several beginning iterations. At step 3, if $|\bar{\alpha}_i|\leqslant\epsilon$ (ϵ is a very small positive value), the penalty term $\frac{1}{\epsilon^2}\alpha_i^2$ in the optimization problem will set α_i to a smaller value approaching to zero at step 2. When the algorithm converges to a stationary point $\boldsymbol{\alpha}^*$, the regularization term will turn into the L_0-norm, $\sum_{i=0}^{l}\lambda_i|\alpha_i^*|=\sum_{\alpha_i^*=0}\frac{1}{\epsilon^2}0+\sum_{\alpha_i^*\neq 0}\frac{1}{|\alpha_i^*|^2}|\alpha_i^*|^2=\sum_{\alpha_i^*\neq 0}1$.

It is interesting that we can modify the above Implementation A to another implementation (called Implementation B) if we change the adaptive term $\boldsymbol{\alpha}^T\text{diag}(\lambda)_{(t)}\boldsymbol{\alpha}$ from the L_2-norm to L_1-norm. More specifically, we first define the following optimization problem.

$$\begin{aligned}&\min_{\boldsymbol{\alpha}}\quad\sum_{i=1}^{l}\lambda_i|\alpha_i|+l(\boldsymbol{\alpha})\\&\text{s.t.}\quad g_k(\boldsymbol{\alpha})\leqslant 0,\quad k=1,\ldots,Nc,\end{aligned}\tag{7}$$

where the kernel expansion $f(\boldsymbol{x})=\boldsymbol{\alpha}^Tk(\boldsymbol{x},\cdot)$, $l(\boldsymbol{\alpha})$ is a loss function, $g_k(\boldsymbol{\alpha})$ is a set of constraints, Nc is the constraint number.

Implementation B of L_0-norm

1. Set $\lambda=(1,\ldots,1)^T$.
2. Solve the optimization problem (7) and get the solution $\bar{\boldsymbol{\alpha}}$.
3. Update λ by $\lambda_i=\begin{cases}\frac{1}{\bar{\alpha}_i^2} & \text{if}\quad|\bar{\alpha}_i|>\epsilon,\\ \frac{1}{\epsilon^2} & \text{otherwise.}\end{cases}$
4. Go back to 2 until convergence.

Similar to Implementation A, when $t\to+\infty$, $\sum_{i=0}^{l}\lambda_i|\alpha_i^*|=\sum_{\alpha_i^*=0}\frac{1}{\epsilon}0+\sum_{\alpha_i^*\neq 0}\frac{1}{|\alpha_i^*|}|\alpha_i^*|=\sum_{\alpha_i^*\neq r0}1$. Hence it can also achieve sparseness when the algorithm converges.

3.3. Model definition of SSVC

In this section, we describe how to exploit the above novel sparse framework for achieving sparseness in SVC.

The L_0-norm regularization term makes the coefficients α_i shrink to zero aggressively, and therefore only a very few coefficients are non-zero. A consequent algorithm of Sparse Support Vector Classification will be proposed via utilizing the L_0-norm regularization term. We first show how to achieve the SSVC by exploiting Implementation A:

$$\begin{aligned}&\boldsymbol{\alpha}_{(t+1)}=\arg\min_{\boldsymbol{\alpha}}\mathbf{1}^T\boldsymbol{\xi}+C\frac{1}{2}\boldsymbol{\alpha}^T\text{diag}(\lambda)\boldsymbol{\alpha}\\&\text{s.t.}\quad y_i(\boldsymbol{w}^T\Phi(\boldsymbol{x}_i)+b)-1+\xi_i\geqslant 0,\quad\xi_i\geqslant 0,\ \forall i,\ 1\leqslant i\leqslant l,\end{aligned}\tag{8}$$

where C is a trade-off constant tuned by the user, and the elements of diag(λ) are updated at step 3 of Implementation A.

When compared with the original SVC, the above model replaces the regularization term $\frac{1}{2}\|\boldsymbol{w}\|_2^2=\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha}$ ($\boldsymbol{K}$ denotes a symmetric kernel matrix with elements $k_{ij}=k(\boldsymbol{x}_i,\boldsymbol{x}_j)$) with the zero-norm given by the iterative term $\frac{1}{2}\boldsymbol{\alpha}^T\text{diag}(\lambda)\boldsymbol{\alpha}$. The two-norm term, $\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha}$, implying the margin maximization, is used to control the structure complexity. Similarly, the zero-norm, defined as the minimum number of non-zero elements, also aims to reduce the structure complexity.

If one substitutes $\boldsymbol{w}$ with $\sum_i^l\alpha_i\Phi(\boldsymbol{x}_i)$, one can remove $\boldsymbol{w}$ and obtain the following model definition:

3.3.1. Sparse Support Vector Classification A (SSVC$_A$)

The Sparse Support Vector Classification using Implementation A can be achieved by solving the following optimization problem iteratively:

$$\begin{aligned}&\boldsymbol{\alpha}_{(t+1)}=\arg\min_{\boldsymbol{\alpha}}\frac{1}{2}\boldsymbol{\alpha}^T\text{diag}(\lambda)\boldsymbol{\alpha}+C\mathbf{1}^T\boldsymbol{\xi}\\&\text{s.t.}\quad y_i\boldsymbol{\alpha}^Tk(\boldsymbol{x}_i,\cdot)+\xi_i\geqslant 1,\quad\xi_i\geqslant 0,\ \forall i,\ 1\leqslant i\leqslant l,\end{aligned}\tag{9}$$

where C is a positive trade-off constant tuned by the user, and the elements of diag(λ) are updated at step 3 of Implementation A.

The above optimization problem is a convex programming problem, or a typical Quadratic Programming (QP) problem at each iteration. Hence, it can be practically optimized via a sequence of QP steps.

Similarly, we can also exploit Implementation B to achieve the SSVC:

3.3.2. Sparse Support Vector Classification B (SSVC$_B$)

The Sparse Support Vector Classification using Implementation B can be achieved by solving the following optimization problem iteratively:

$$\begin{aligned} &\boldsymbol{\alpha}_{(t+1)} = \arg\min_{\boldsymbol{\alpha}} \boldsymbol{\lambda}^T|\boldsymbol{\alpha}| + C\mathbf{1}^T\boldsymbol{\xi} \\ &\text{s.t.} \quad y_i\boldsymbol{\alpha}^T k(\boldsymbol{x}_i,\cdot) + \xi_i \geqslant 1, \quad \xi_i \geqslant 0,\ \forall i,\ 1 \leqslant i \leqslant l, \end{aligned} \tag{10}$$

where C is a positive trade-off constant tuned by the user, and the elements of $\boldsymbol{\lambda}$ are similarly updated at step 3 of Implementation *B*.

Clearly, this optimization problem is a Linear Programming problem and can also be solved practically. We will discuss how to solve the above two problems more efficiently in the next subsection.

Remarks. It is noted that we can even use a combination form of the two-norm and the zero-norm as the regularization. For example, the objective function of Implementation *A* can be changed to $\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} + C_1\boldsymbol{\alpha}^T\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{\alpha} + C\mathbf{1}^T\boldsymbol{\xi}$ ($C_1 \in \mathbb{R}^+$ is another trade-off parameter), while all the remaining constraints are maintained unchanged. The resulting optimization problem is still a QP problem and can be efficiently solved by the SMO algorithm. The flexibility of choosing different objective functions shows how the proposed sparse framework can be easily integrated into many other kernel methods. Later we will also demonstrate how the proposed zero-norm term can attain a sparse kernel density estimation model. The flexibility is one significant merit of our proposed framework.

3.4. Practical optimization for SSVC

We introduce how to efficiently solve the optimization problem involved in SSVC for Implementation *A* and Implementation *B* in the following.

3.4.1. Optimization of $SSVC_A$

From the Lagrangian of (9), we can obtain its dual problem:

$$\begin{aligned} &\min_{\boldsymbol{\beta}} \quad \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{H}\boldsymbol{\beta} - \mathbf{1}^T\boldsymbol{\beta} \\ &\text{s.t.} \quad 0 \leqslant \beta_i \leqslant C, \quad \forall i,\ 1 \leqslant i \leqslant l, \end{aligned} \tag{11}$$

where the Hessian matrix is $\boldsymbol{H} = \overline{\boldsymbol{K}}^T[\mathrm{diag}(\boldsymbol{\lambda})]^{-1}\overline{\boldsymbol{K}}$ and $\overline{K} = [y_1 k(\boldsymbol{x}_1,\cdot),\ldots,y_l k(\boldsymbol{x}_l,\cdot)]$. This quadratic programming problem lacking a linear equation constraint is slightly different from the formulation (2) of SVC. And it can be optimized by an algorithm similar to Platt's sequential minimal optimization (SMO) (Platt, 1998) or its improvements (Keerthi et al., 2001).

We now show how a modified SMO can be used to solve (11). In the modified SMO algorithm, it chooses one Lagrange multiplier β_i to optimize while fixing the other variables at each iteration. The objective function is $Q(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{H}\boldsymbol{\beta} - \mathbf{1}^T\boldsymbol{\beta}$ and its gradient vector is $\nabla Q = \boldsymbol{H}\boldsymbol{\beta} - \mathbf{1}$. If assuming $\beta_i \leftarrow \beta_i + \Delta$, then we get

$$Q(\beta_i + \Delta) = \frac{1}{2}h_{ii}\Delta^2 + \nabla Q_i\Delta + Q(\beta_i), \tag{12}$$

where h_{ii} is a positive diagonal element of $\boldsymbol{H}$. When $\Delta = -\frac{\nabla Q_i}{h_{ii}}$, a maximal reduction is achieved in the objective function, $Q(\beta_i+\Delta) = Q(\beta_i) - \frac{(\nabla Q_i)^2}{2h_{ii}}$. To make a fast convergence, the chosen Lagrange multiplier β_i should maximize the term $\frac{(\nabla Q_i)^2}{h_{ii}}$.

If $\nabla Q_i > 0$, then $\Delta < 0$ and require $\beta_i > 0$; if $\nabla Q_i < 0$, then $\Delta > 0$ and require $\beta_i < C$. So the optimal i^* is chosen as follows:

$$i^* = \arg\max_i \left\{ \frac{(\nabla Q_i)^2}{h_{ii}} \middle| \nabla Q_i > \tau,\ \beta_i > 0 \text{ or } \nabla Q_i < -\tau,\ \beta_i < C \right\}, \tag{13}$$

where τ is a small tolerance, e.g., $\tau = 0.001$. Then the chosen variable β_{i^*} is optimized analytically, $\beta_{i^*}^{new} \leftarrow \min\left(\max\left(0, \beta_{i^*} - \frac{\nabla Q_i}{h_{ii}}\right), C\right)$. And the gradient is updated by $\nabla Q \leftarrow \nabla Q + \boldsymbol{h}_{i^*}(\beta_{i^*}^{new} - \beta_{i^*})$, where $\boldsymbol{h}_{i^*}$ denotes the i^*th column of $\boldsymbol{H}$.

Until it fails to find i^* or the maximum number of iterations is reached, the optimization algorithm terminates.

After we obtain the optimal solution $\boldsymbol{\beta}$ of (11), the coefficient vector $\boldsymbol{\alpha}$ is updated by $\boldsymbol{\alpha} = [\mathrm{diag}(\boldsymbol{\lambda})]^{-1}\overline{\boldsymbol{K}}\boldsymbol{\beta}$. Let $\bar{\boldsymbol{\alpha}}$ be the solution of the previous iteration. The reciprocals of small λ_i can be avoided by $[\mathrm{diag}(\boldsymbol{\lambda})]^{-1} = \mathrm{diag}(\bar{\alpha}_0^2,\ldots,\bar{\alpha}_l^2)$ without ϵ.

Note that as the iteration t is increased, more and more coefficients α_i shrink to zero and then stay at zero. After each iteration, the coefficients approaching to zero can be pruned so as to decrease the number of the inequality constraints of (9) or reduce the computational time for the Hessian matrix in (11).

3.4.2. Optimization of $SSVC_B$

We now consider how to optimize the SSVC by utilizing Implementation *B*.

We can reformulate (10) into a Linear Programming problem as follows:

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,\boldsymbol{\xi}} \quad \boldsymbol{\lambda}^T(\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) + C\mathbf{1}^T\boldsymbol{\xi} \\ &\text{s.t.} \quad y_i(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T k(\boldsymbol{x}_i,\cdot) + \xi_i \geqslant 1, \quad \alpha_i, \alpha_i^*, \xi_i \geqslant 0,\ \forall i,\ 1 \leqslant i \leqslant l, \end{aligned} \tag{14}$$

where each α_i is substituted by two positive variables α_i and α_i^*. Because $\alpha_i - \alpha_i^* \in \mathbb{R}$ and $\alpha_i\alpha_i^* = 0$ holds for the optimal solution, we get $|\alpha_i - \alpha_i^*| = \alpha_i + \alpha_i^*$.

From the Lagrangian of (14), we can obtain its dual problem

$$\begin{aligned} &\min_{\boldsymbol{\beta}} \quad -\mathbf{1}^T\boldsymbol{\beta} \\ &\text{s.t.} \quad -\mathbf{1} \leqslant \boldsymbol{A}\boldsymbol{\beta} \leqslant \mathbf{1}, \quad 0 \leqslant \beta_i \leqslant C,\ \forall i, \quad 1 \leqslant i \leqslant l, \end{aligned} \tag{15}$$

where $\boldsymbol{A} = [\mathrm{diag}(\boldsymbol{\lambda})]^{-1}\overline{\boldsymbol{K}}$ and $\overline{\boldsymbol{K}} = [y_1 k(\boldsymbol{x}_1,\cdot),\ldots,y_l k(\boldsymbol{x}_l,\cdot)] \in \mathbb{R}^{(l+1)\times l}$. Assuming that $\bar{\boldsymbol{\alpha}}$ be the solution of the previous iteration again, the reciprocals of small λ_i can be avoided by $[\mathrm{diag}(\boldsymbol{\lambda})]^{-1} = \mathrm{diag}(|\bar{\alpha}_0|,\ldots,|\bar{\alpha}_l|)$ without ϵ. The previous Linear Programming problem has $3l+2$ variables, but this one has only l variables.

From the solution $\boldsymbol{\beta}$ of (15), the non-zero elements of $\boldsymbol{\alpha}$ and the bias term b can be obtained by solving a linear equation system $\overline{\boldsymbol{K}}^T_{I_1,I_2}\boldsymbol{\alpha}_{I_1} = \mathbf{1}$. Here I_1 and I_2 are two index subsets, which are defined as $I_1 = \{i|(\boldsymbol{A}\boldsymbol{\beta})_i| = 1\}$ and $I_2 = \{i|0 < \beta_i < C\}$. The sub-matrix $\overline{\boldsymbol{K}}_{I_1,I_2}$ includes the elements at the I_1th rows and I_2th columns of $\overline{\boldsymbol{K}}$. The sub-vector $\boldsymbol{\alpha}_{I_1}$ takes the I_1th elements of $\boldsymbol{\alpha}$, because the other elements of $\boldsymbol{\alpha}$ are zero.

3.5. Connections with other models

In this section, we show that our proposed approach has close connections with some existing models. More specifically, one variation of our proposed framework, i.e., Implementation *B*, is equivalent to the zero-norm classifier proposed in (Weston et al., 2003). Moreover, our proposed sparse framework presents a more general and flexible extension of the Sparse Probit Classifier (Figueiredo, 2002, 2003).

3.5.1. Equivalence between (Weston et al., 2003) and implementation B

Weston et al. (2003) proposed an approximation $\min_{\boldsymbol{\alpha}}\sum_i \ln(\epsilon + |\alpha_i|)$ to the minimization of L_0-norm $\min_{\boldsymbol{\alpha}}\|\boldsymbol{\alpha}\|_0^0$, where $0 < \epsilon \ll 1$ and gave its implementation algorithm for a general kernel-based learning problem as follows:

Implementation in (Weston et al., 2003)

1. Set $\boldsymbol{z} = (1,\ldots,1)^T$.
2. Solve the optimization problem (16) and get the solution $\bar{\boldsymbol{\alpha}}$.
3. Set $\boldsymbol{z} \leftarrow \boldsymbol{z} * \bar{\boldsymbol{\alpha}}$.
4. Go back to 2 until convergence.

$$\min_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{l} |\alpha_i| + l(\boldsymbol{\alpha}), \qquad \text{s.t.} \quad g_k(\boldsymbol{\alpha}) \leqslant 0, \quad k = 1, \ldots, Nc, \tag{16}$$

where the kernel expansion $f(\boldsymbol{x}) = \boldsymbol{\alpha}^T k(\boldsymbol{x}, \cdot)$, $l(\boldsymbol{\alpha})$ is a loss function, $g_k(\boldsymbol{\alpha})$ is a set of constraints, Nc is the constraint number.

Here the operator * denotes the component-wise product between two vectors, i.e., $\boldsymbol{z}^*\boldsymbol{\alpha} = (z_0\alpha_0, \ldots, z_l\alpha_l)^T$. Step 3 is called a multiplicative update for $\boldsymbol{z}$ and step 2 uses it to re-scale the kernel vectors of training data. After it converges, the final solution $f(\boldsymbol{x}) = \boldsymbol{z}^T k(\boldsymbol{x}, \cdot)$ is obtained.

The above implementation uses a novel kernel expansion $f(\boldsymbol{x}) = \boldsymbol{\alpha}^T(k(\boldsymbol{x}, \cdot) * \boldsymbol{z}) = \boldsymbol{z}_{new}^T k(\boldsymbol{x}, \cdot)$ where $\boldsymbol{z}_{new} = \boldsymbol{z}^*\boldsymbol{\alpha}$. In comparison, the implementation B uses a normal kernel expansion $f(\boldsymbol{x}) = \boldsymbol{\alpha}^T k(\boldsymbol{x}, \cdot)$. However, in the dual problem (15) of SSVC$_B$, there is also a re-scaling of the kernel matrix, i.e., $\boldsymbol{A} = \mathrm{diag}(|\bar{\alpha}_0|, \ldots, |\bar{\alpha}_l|)\bar{\boldsymbol{K}}$. If all $|\bar{\alpha}_i|$ are replaced by $\bar{\alpha}_i$ in this formula, the two implementations are exactly equivalent. Our experimental results also validate this equivalence.

3.5.2. Connection withSparse Probit Classifier (Figueiredo, 2002, 2003)

By defining a two-level prior probability model over α, Figueiredo (2002, 2003) proposed a Sparse Probit Classifier model that can adaptively generate a highly sparse solution via an EM process. At each iteration, SPC tries to maximize the following objective function:

$$\begin{aligned} \alpha_{(t+1)} &= \arg\max_{\alpha}\{\log p(\alpha|\boldsymbol{z})\} \\ &= \arg\max_{\alpha}\{\log p(\boldsymbol{z}|\alpha) + \log p(\alpha)\} \\ &= \arg\max_{\alpha}\{-\|\boldsymbol{H}\alpha - \boldsymbol{z}\|^2 - \alpha^T\boldsymbol{\Lambda}\alpha\}, \end{aligned} \tag{17}$$

where $\boldsymbol{H}$ denotes a symmetric matrix with elements $h_{ij} = y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ again, and $\boldsymbol{z}$ defines the data corrupted by a Gaussian distribution. The matrix $\boldsymbol{\Lambda}$, defined as $\mathrm{diag}(1/\tau_1, \ldots, 1/\tau_l)$, is updated by $\mathrm{diag}(|\alpha_{1,(t)}|^{-2}, \ldots, |\alpha_{l,(t)}|^{-2})$ in the Expectation step. The first term of (17) corresponds to the errors between the output of the learned classifier $f(\boldsymbol{x}) = \alpha^T k(\boldsymbol{x}, \cdot)$ and the actual output $\boldsymbol{z}$; the second term represents the prior imposed by the assumption over α, which is exactly the zero-norm term defined in Implementation A of our proposed framework.

There are several major differences between the above SPC method and our proposed novel framework. First and foremost, Figueiredo (2002, 2003) derived the sparse solution only for classification within the Bayesian framework. In contrast, our proposed L_0-norm is more flexible, and can find its applications in many domains. The term $\boldsymbol{\alpha}^T\,\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{\alpha}$ of (9) can be flexibly plugged in most kernel methods. Similar to the optimization presented in Section 3.3, we can just append the sparse term $\boldsymbol{\alpha}^T\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{\alpha}$ to the associated optimization objective function in order to achieve sparseness for certain models. This is not only applicable for classification, it can be also used in feature selection, regression, and kernel density estimation. Hence our proposed method represents a more general framework. For a simple illustration, we show in (18)–(20) how the term can be used to achieve sparseness in kernel density estimation in an iterative way.

3.5.2.1. Sparse Kernel Density Estimation. The Sparse Kernel Density Estimation can be achieved by the following iterative process:

$$\alpha_{(t+1)} = \arg\min_{\alpha}\left\{-\sum_{i=1}^{l}\log p(\boldsymbol{x}_i) + \boldsymbol{\alpha}^T\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{\alpha}\right\}, \tag{18}$$

$$\text{s.t.} \quad \sum_{i=1}^{l}\alpha_i = 1, \tag{19}$$

$$p(\boldsymbol{x}) = \sum_{i=1}^{l}\alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i), \quad \text{and} \quad k(\boldsymbol{x}, \boldsymbol{x}_i) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2\sigma^2}\right). \tag{20}$$

At the tth step, each element of λ is similarly updated by $\lambda_i = |\alpha_{(t)i}|^{-2}$. The final kernel density is given by $p(\boldsymbol{x}) = \sum_{i=1}^{l}\alpha_i^* k(\boldsymbol{x}, \boldsymbol{x}_i)$, where $\boldsymbol{\alpha}^*$ represents the optimal $\boldsymbol{\alpha}$ when the algorithm converges.

Second, our proposed zero-norm framework also motivates many other related models. For example, the variation of Implementation B can also achieve the sparseness, which is equivalent to the model proposed by Weston et al. (2003). On the other hand, arbitrary-norm regularized kernel machines can also be inspired from the proposed framework. In more details, the following iterative process can obtain an arbitrary p-norm regularized kernel machine:

$$\boldsymbol{\alpha}_{(t+1)} = \arg\min_{\boldsymbol{\alpha}} Q(\boldsymbol{\alpha}|\boldsymbol{\alpha}_{(t)}) = \cdots + C_\alpha\frac{1}{2}\boldsymbol{\alpha}^T\mathrm{diag}(\boldsymbol{\lambda}_{(t)})\boldsymbol{\alpha}, \tag{21}$$

where $\mathrm{diag}(\boldsymbol{\lambda}_{(t)})$ is defined as $\mathrm{diag}(|\alpha_{1,(t)}|^{-(2-p)}, \ldots, |\alpha_{l,(t)}|^{-(2-p)})$, and other symbols are similarly defined as before. Furthermore, we can even achieve an infinity-norm regularized kernel machine by replacing the updating of $\mathrm{diag}(\boldsymbol{\lambda}_{(t)})$ with $\mathrm{diag}(0, \ldots, 0, 1/|\alpha_{i_{max},(t)}|, 0, \ldots, 0)$. In comparison, it is not easy to extend SPC to other norm-based regularized models, since this Bayesian method requires the prior probability over α be defined; these prior probabilities might be difficult to be specified.

Remarks. It is interesting to notice that RVM and SPC are actually quite similar in the sense that they assume similar priors over the weight $\boldsymbol{\alpha}$. This can be seen in (22). The only difference is that, SPC uses τ_i as the parameter, while RVM adopts $1/\tau_i$ as the hyper-parameter. This consequently results in different distributional assumption in the second level. In this sense, RVM can also be regarded as one implementation of the L_0-norm.

$$\begin{aligned} \text{SPC}: \text{Level 1}: &\quad p(\alpha_i) = \mathcal{N}(\alpha_i|0, \tau_i) \\ \text{Level 2}: &\quad p(\tau_i) \propto 1/\tau, \\ \text{RVM}: \text{Level 1}: &\quad p(\alpha_i) = \mathcal{N}(\alpha_i|0, 1/\tau_i) \\ \text{Level 2}: &\quad \tau_i : \text{a flat hyper-prior}. \end{aligned} \tag{22}$$

4. Experiments

In this section, we evaluate our proposed sparse models, SSVC$_A$ and SSVC$_B$ against many other competitive models such as the SVC, the Relevance Vector Machine (Tipping, 2000, 2001), the Sparse Probit Classifier (Figueiredo, 2002, 2003), the sparse classifier proposed in (Weston et al., 2003) (for brevity, we call this model as SSVC$_W$), the Linear Programming SVC (LP-SVC) (Kecman and Hadzic, 2000), and the Sparsity-Controlled SVC (SC-SVC) (Drezet and Harrison, 2001). We first present some examples in order to demonstrate the advantages of our proposed framework clearly. We then conduct evaluations on seven benchmark datasets obtained from the UCI machine learning repository (Blake and Merz, 1998).

4.1. Some examples

Some 2D classification examples are given to gain insight into how SSVC works on the separable and non-separable data sets.

The first example is carried out on a separable synthetic set with 40 patterns. The decision boundaries of SVC, SSVC$_A$, and SSVC$_B$ are shown in Fig. 2. Training patterns of the two classes are marked by the hexagram and star points respectively, SVs are marked by the circle points, and the separating hyperplane $f(\boldsymbol{x}) = 0$ and the two support hyperplanes $f(\boldsymbol{x}) = \pm 1$ are plotted by curves. Two observations deserve our attentions. First, the proposed sparse

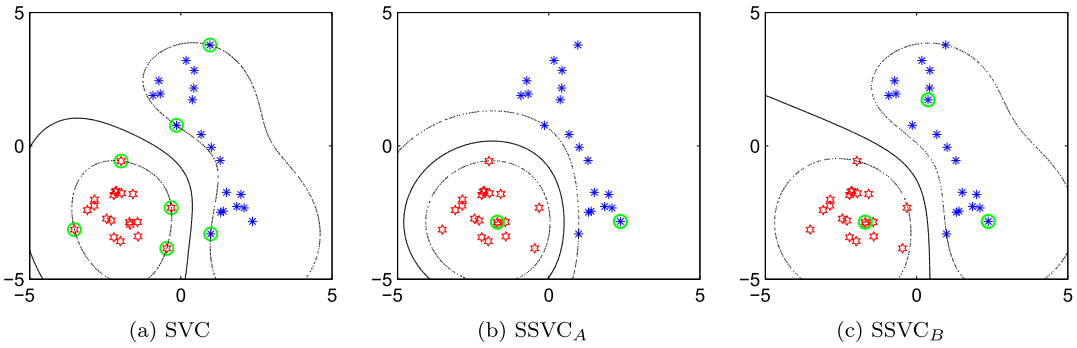


Fig. 2. SVC and SSVCs with Gaussian kernel $\sigma = 2$ and $C = 10$ trained on two separable classes.

SVC can indeed reduce the number of SVs while maintaining the classification accuracy. Concretely, SSVC$_A$ generates 2 and 3 SVs, respectively, while the standard SVC outputs 7 SVs. Second, as observed, the SVs are unnecessarily located on the margins. Instead, they represent some "prototype" points from data. This is because we do not require that the coefficient $\boldsymbol{\alpha}$ should be the Lagrange multipliers. This is a distinct difference between our model and the standard SVC.

The second example is carried out on the Ripley data consisting of 250 patterns. The three training results of SVC, SSVC$_A$, and SSVC$_B$ are shown in Fig. 3. SVC needs 102 Support Vectors, while both SSVC$_A$ and SSVC$_B$ need only 4 SVs. SVC associates SVs with non-zero Lagrange multipliers, so all patterns inside the margin zone and those outside the margin zone but wrongly classified are SVs. SSVCs have no such restriction on SVs and they can yield two quite sparse classifiers.

The third example trains SSVCs with Gaussian kernel on the Ripley data again, but with 1000 patterns, in order to illuminate the convergence speeds of the two training algorithms. From Fig. 4, one can observe that SSVC$_A$ converges after three iterations and SSVC$_B$ converges after nine iterations. For both cases, the number of SVs decreases from the initial 1000 to the final 4 rapidly.

4.2. Evaluations on real data

We compare our SSVC$_A$ with the standard SVC, RVM (Tipping, 2001; Tipping and Faul, 2003), the Sparse Probit Classifier (Figueiredo, 2002, 2003), the sparse classifier (Weston et al., 2003) (SSVC$_W$), the Linear Programming SVC (LP-SVC) (Kecman and Hadzic, 2000), and the Sparsity-Controlled SVC (SC-SVC) (Drezet and Harrison, 2001) in terms of training time, number of Support Vectors or kernel terms, and classification error on seven benchmark datasets. We do not compare SSVC$_B$ with these models, because it is equivalent to (SSVC$_W$) as shown in Section 3.5.2. The data descriptions are summarized in Table 1, where the numbers of

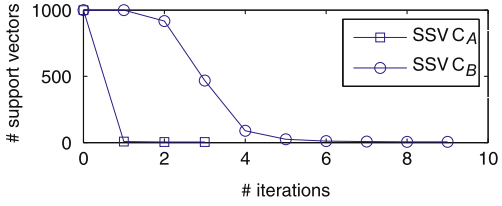


Fig. 4. Number of support vectors decreases rapidly with increasing number of iterations.

samples are constrained to a maximum 1000 due to the time-consuming optimization of the Linear Programming involved in LP-SVC and SSVC$_W$. The maximal numbers of iterations for the training procedures of SSVCs are all set to 50, and the maximal number of iterations for the modified SMO algorithm is set to 9999. If the two-norm distance of $\boldsymbol{\alpha}$ between the two successive iterations is less than 10^{-4}, then the training procedures terminate.

The kernel function used in the experiments is the popular Gaussian kernel. The final experimental results are obtained via 10-fold cross validation. SVC, SSVC$_A$, LP-SVC, and SSVC$_W$ have two parameters C and σ, RVM and SPC have only one parameter σ, while SC-SVC has three parameters. Their parameters are chosen via a 10-fold cross validation process on the training set. The experimental results including the averages and deviations of training times (seconds), number of SVs or kernel terms, and test errors (%) are reported in Table 2.

From Table 2, all the sparse models achieve competitive error rates with the standard SVC in all the datasets. RVM performs slightly worse in Diabetes, B. Cancer and Waveform, and SPC yields marginally worse performance in B. Cancer and German. SVC and SSVCs have two parameters σ and C (a trade-off constant between sparsity and training errors), while RVM and SPC have one parameter σ. Hence, SSVCs are more flexible than RVM and SPC in the sense they can trade off the sparsity with the error rate. Both the

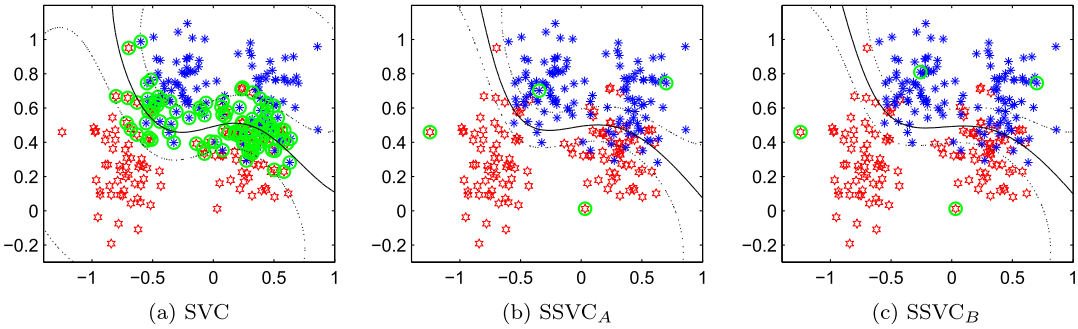


Fig. 3. SVC and SSVCs with Gaussian kernel $\sigma = 0.5$ and $C = 1$ trained on two non-separable classes.

Table 1
Data descriptions: dataset name, dimension, and number of samples.

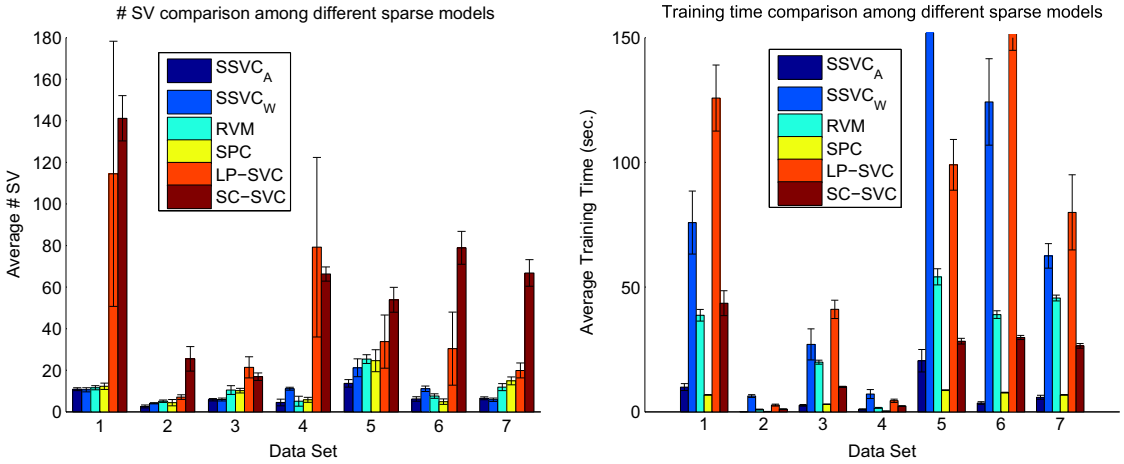
Data set	1	2	3	4	5	6	7
	Banana	Thyroid	Diabetes	B. Cancer	German	Ringnorm	Waveform
Dim	2	5	8	9	20	20	21
Num	1000	215	768	277	1000	1000	1000

Table 2
Experimental results: training times (s), numbers of kernels and test errors (%) obtained by 10-fold cross validation.

	Banana	Thyroid	Diabetes	B. Cancer	German	Ringnorm	Waveform
SVC	0.43 ± 0.10 s 227.5 ± 5.6 10.0 ± 3.9%	0.01 ± 0.00 s 31.0 ± 1.8 4.3 ± 4.2%	0.13 ± 0.00 s 418.4 ± 8.0 22.1 ± 8.6%	0.02 ± 0.00 s 140.3 ± 4.3 24.1 ± 8.6%	0.70 ± 0.03 s 496.6 ± 8.1 23.7 ± 3.2%	0.28 ± 0.02 s 122.6 ± 6.0 2.0 ± 1.7%	0.20 ± 0.01 s 292.2 ± 5.5 9.7 ± 2.3%
$SSVC_A$	9.92 ± 1.39 s 10.9 ± 0.7 10.0 ± 3.6%	0.10 ± 0.02 s 2.6 ± 0.7 4.3 ± 4.7%	2.60 ± 0.35 s 5.8 ± 0.6 22.1 ± 4.9%	0.98 ± 0.22 s 4.5 ± 1.6 24.8 ± 6.8%	20.48 ± 4.49 s 13.7 ± 1.8 23.8 ± 5.2%	3.53 ± 0.53 s 6.2 ± 1.1 2.0 ± 1.3%	5.82 ± 0.80 s 6.5 ± 0.7 9.4 ± 2.2%
$SSVC_W$	75.86 ± 12.63 s 10.6 ± 1.0 9.7 ± 4.0%	6.34 ± 0.56 s 4.1 ± 0.3 3.8 ± 3.0%	27.04 ± 6.23 s 5.9 ± 0.7 22.6 ± 5.1%	7.13 ± 1.80 s 11.1 ± 0.7 24.4 ± 8.6%	505.58 ± 308.77 s 21.2 ± 4.3 23.5 ± 4.7%	124.23 ± 17.33 s 11.1 ± 1.3 1.8 ± 1.1%	62.52 ± 4.91 s 5.8 ± 0.8 9.0 ± 2.1%
RVM	38.72 ± 2.34 s 11.6 ± 1.0 9.9 ± 3.5%	0.92 ± 0.17 s 5.1 ± 0.6 4.3 ± 3.5%	19.91 ± 0.82 s 10.5 ± 2.1 24.2 ± 5.0%	1.59 ± 0.17 s 5.1 ± 2.4 25.2 ± 7.2%	54.11 ± 3.23 s 25.4 ± 2.1 23.2 ± 3.3%	38.98 ± 1.53 s 7.6 ± 1.1 1.8 ± 1.9%	45.62 ± 1.18 s 11.9 ± 1.7 10.5 ± 2.7%
SPC	6.80 ± 0.12 s 12.3 ± 1.5 9.9 ± 4.5%	0.14 ± 0.01 s 4.4 ± 1.5 4.8 ± 3.2%	3.04 ± 0.06 s 10.2 ± 1.1 22.8 ± 5.7%	0.27 ± 0.02 s 5.8 ± 1.2 25.2 ± 6.9%	8.67 ± 0.11 s 24.6 ± 5.3 24.8 ± 5.3%	7.72 ± 0.14 s 4.9 ± 1.3 2.1 ± 1.4%	6.81 ± 0.12 s 14.9 ± 1.9 9.9 ± 3.0%
LP-SVC	125.79 ± 13.24 s 114.5 ± 63.8 9.4 ± 3.8%	2.72 ± 0.39 s 7.1 ± 1.1 5.7 ± 4.4%	41.09 ± 3.67 s 21.4 ± 5.1 22.5 ± 4.8%	4.44 ± 0.65 s 79.2 ± 43.2 25.2 ± 8.3%	99.02 ± 10.15 s 33.8 ± 12.8 23.5 ± 4.9%	161.48 ± 16.58 s 30.4 ± 17.6 1.7 ± 1.3%	79.96 ± 15.05 s 19.9 ± 3.6 9.5 ± 2.6%
SC-SVC	43.55 ± 4.99 s 141.2 ± 10.9 9.6 ± 3.9%	1.08 ± 0.12 s 25.5 ± 5.9 4.8 ± 5.0%	10.01 ± 0.23 s 16.9 ± 1.8 22.9 ± 4.4%	2.31 ± 0.16 s 66.3 ± 3.4 24.1 ± 8.4%	28.32 ± 1.14 s 53.9 ± 6.0 23.5 ± 5.0%	29.82 ± 0.82 s 78.9 ± 7.9 1.8 ± 2.0%	26.45 ± 0.91 s 66.8 ± 6.4 9.8 ± 2.7%

accuracy performance and the sparsity of these sparse methods are sensitive to the parameter σ. Sparsity may fail when a smaller value of σ is used (Weston et al., 2003), while the performance may decrease when a larger value of σ is used.

SVC needs a lot of kernel terms or SVs to construct a separating hyperplane, while the other sparse methods require dramatically fewer ones. This phenomenon can be clearly observed from Table 2. This is the major advantage of the sparse models over the standard SVC. In order to compare the sparsity performance for the various sparse models, we further plot in Fig. 5(a) the number of SVs required for these methods in all the datasets. Although all the sparse models can achieve sparsity, it is obvious that the proposed $SSVC_A$ demonstrates the overall best performance against other sparse models in terms of the sparsity: $SSVC_A$ yields the fewest SVs in Thyroid, Diabetes, B.Cancer, German, while it outputs just slightly more SVs than $SSVC_W$, but still fewer than RVM, SPC, LP-SVC, and SC-SVC in Banana and Waveform. The advantage of $SSVC_A$ can be clearly inspected in the German dataset: $SSVC_A$ generates only 13.7 SVs, while the remaining sparse models outputs over 20 SVs. Furthermore, a t-test shows that $SSVC_A$ generates



(a) Number of SVs (b) Training Time

Fig. 5. Comparisons among various sparse models in terms of the number of SVs and the training time. The proposed $SSVC_A$ demonstrates the best performance in both graphs.

significantly fewer SVs than SPC in all the datasets except Ringnorm at $p = 0.05$. In addition, a t-test shows that the SV number of $SSVC_A$ is just slightly fewer than that of RVM in Banana and B. Cancer, but significantly fewer than that of RVM in the remaining datasets at $p = 0.05$. Another interesting point is that all the L_0-norm based sparse models, including $SSVC_A$, $SSVC_W$, RVM and SPC demonstrate better sparsity than the L_1-norm based sparse models, LP-SVC and SC-SVC. This further justifies the theory mentioned in Section 3.

We examine the training efficiency of all the methods in the following. Clearly, the standard SVC trained by an improved SMO algorithm (Keerthi et al., 2001) costs the least training time among all classifiers. The other sparse models usually exploit iterative approaches to achieve the sparsity, hence they spend more training time than SVC. In order to visually demonstrate the training efficiency of all the sparse methods, we plot the training time required in Fig. 5(b). When inspecting the computational time within these sparse models, one can find that $SSVC_A$ and SPC outperform the remaining methods. More specifically, $SSVC_A$ demonstrates the fast training speed in Thyroid, Diabetes, Ringnorm, and Waveform, while SPC costs the least training time in Banana, B. Cancer, and German. $SSVC_A$ and SPC are faster because $SSVC_A$ can be trained by using the efficient modified SMO algorithm and SPC only needs solving a linear equation system without any constraints at each iteration. In comparison, RVM needs to make a Cholesky decomposition for the Hessian matrix and conduct a matrix inversion whose complexity is of $O(l^3)$. $SSVC_W$ needs to solve a Linear Programming problem with l inequality constraints at each step, which proves to converge relatively slowly. On the other hand, LP-SVC needs to solve a linear programming problem with many constraints, while SC-SVC has to solve a Quadratic Programming problem and proves difficult to use SMO directly. Both the algorithms show relatively slow performance than the proposed sparse learning model. Finally, a t-test show that $SSVC_A$ and SPC are significantly faster than the other sparse models at the confidence level $p = 0.05$.

5. Conclusion

This paper has proposed a sparse learning framework called Sparse Support Vector Classification that can reduce the number of Support Vectors significantly for the standard Support Vector Classification. The proposed novel model exploits the L_0-norm regularization term in order to automatically set the irrelevant parameters exactly to zero. This model is important in that it has close connections with some other existing models. More specifically, one variation of the proposed method is equivalent to the zero-norm classifier proposed in (Weston et al., 2003); it is also an extended and more flexible framework in parallel with the Sparse Probit classifier proposed by Figueiredo (2003). The detailed framework, the implementations of SSVC, and the practical optimization methods have been presented in the paper. Experimental evaluations on both synthetic and benchmark datasets have demonstrated that the proposed sparse framework yields significantly fewer Support Vectors than SVC. When compared with the other sparse models, SSVC has also shown its advantages in terms of both the training speed and the number of Support Vectors. Future work includes further evaluations on more data sets. In particular, we would like to evaluate the proposed algorithm on the data collected by Raetsch.[2] Moreover, it is also interesting to compare more competitive sparse algorithms (e.g., the Fast RVM (Tipping and Faul, 2003)) with the proposed novel algorithm.

[2] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm.

Acknowledgments

The authors thank the anonymous reviewers and the editor for their valuable comments on this paper. This work is partially supported by the Excellent SKL Project of NSFC (No. 60723005), China.

References

Blake, C., Merz, C., 1998. Repository of Machine Learning Databases. University of California, Irvine. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
Burges, C.J.C., 1996. Simplified support vector decision rules. In: Proc. 13th Internat. Conf. Machine Learning. Morgan Kaufmann, San Mateo, CA, pp. 71–77.
Burges, C.J.C., 1998. A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Disc. 2, 121–167.
Downs, T., Gates, K.E., Masters, A., 2002. Exact simplification of support vector solutions. J. Machine Learn. Res. 2, 293–297.
Drezet, P.M.L., Harrison, R.F., 2001. A new method for sparsity control in support vector classification and regression. Pattern Recognition 34, 111–125.
Figueiredo, M., 2002. Adaptive sparseness using Jeffery prior. Advances in Neural Information Processing Systems, vol. 14. MIT Press, pp. 697–704.
Figueiredo, M., 2003. Adaptive sparseness for supervised learning. IEEE Trans. Pattern Anal. Machine Intell. 25, 1150–1159.
Fung, G.M., Mangasarian, O.M., Smola, A.J., 2002. Minimal kernel classifiers. J. Machine Learn. Res. 3, 303–321.
Huang, K., Yang, H., King, I., Lyu, M.R., Chan, L.W., 2008a. The minimum error minimax probability machine. J. Machine Learn. Res. 5, 1253–1286.
Huang, K., Yang, H., King, I., Lyu, M.R., 2008b. Maxi-min margin machine: learning large margin classifiers globally and locally. IEEE Trans. Neural Networks 19, 260–272.
Kecman, V., Hadzic, I., 2000. Support vectors selection by linear programming. In: Proc. Internat. Joint Conf. Neural Networks (IJCNN 2000), Como, Italy, pp. 193–198.
Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K., 2001. Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput. 13, 637–649.
Lanckriet, G., Ghaoui, L.E., Bhattacharyya, C., Jordan, M.I., 2002. A robust minimax approach to classification. J. Machine Learn. Res. 3, 555–582.
Li, Y., Zhang, W., 2006. Simplify support vector machines by iterative learning. Neural Inform. Process. Lett. Rev. 10, 11–17.
Nguyen, D.D., Ho, T.B., 2005. An efficient method for simplifying support vector machines. In: Proc. 22th Internat. Conf. Machine Learning, Bonn, Germany, pp. 617–624.
Platt, J.C., 1998. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, Microsoft Research. Technical Report MSR-TR-98-14.
Schölkopf, B., Mika, S., Burges, C.J.C., Knirsch, P., Müller, K.R., Rätsch, G., Smola, A.J., 1999. Input space versus feature space in kernel-based methods. IEEE Trans. Neural Networks 10, 1000–1017.
Schölkopf, B., Williamson, R., Smola, A., Taylor, J.S., Platt, J., 2000. Support vector method for novelty detection. In: Advances in Neural Information Processing Systems, vol. 12, Denver Colorado, USA, pp. 582–588.
Smola, A.J., Schölkopf, B., 2004. A tutorial on support vector regression. Statist. Comput. 14, 199–222.
Tipping, M.E., 2000. The relevance vector machine. Advances in Neural Information Processing Systems, vol. 12. MIT Press, pp. 652–658.
Tipping, M.E., 2001. Sparse Bayesian learning and the relevance vector machine. J. Machine Learn. Res. 1, 211–244.
Tipping, M.E., Faul, A., 2003. Fast marginal likelihood maximisation for sparse Bayesian models. In: Proc. Internat. Workshop on Artificial Intelligence and Statistics, Key West, Florida.
Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer Verlag.
Vapnik, V., Mukherjee, S., 1999. Support vector method for multivariate density estimation. In: Advances in Neural Information Processing System, vol. 11, Denver, Colorado, USA, pp. 659–665.
Weston, J., Elisseeff, A., Schölkopf, B., Tipping, M., 2003. Use of the zero-norm with linear models and kernel methods. J. Machine Learn. Res. 3, 1439–1461.
Zheng, D.N., Wang, J.X., Zhao, Y.N., 2006. Training sparse MS-SVR with an expectation–maximization algorithm. Neurocomputing 69, 1659–1664.