

# Alternating direction method of multipliers for regularized multiclass support vector machines

Yangyang Xu<sup>1</sup>, Ioannis Akrotirianakis<sup>2</sup>, and Amit Chakraborty<sup>2</sup>

<sup>1</sup> Rice University, Houston, TX, USA  
yangyang.xu@rice.edu

<sup>2</sup> Siemens Corporate Technology, Princeton, NJ, USA  
(ioannis.akrotirianakis, amit.chakraborty)@siemens.com

**Abstract.** The support vector machine (SVM) was originally designed for binary classifications. A lot of effort has been put to generalize the binary SVM to multiclass SVM (MSVM) which are more complex problems. Initially, MSVMs were solved by considering their dual formulations which are quadratic programs and can be solved by standard second-order methods. However, the duals of MSVMs with regularizers are usually more difficult to formulate and computationally very expensive to solve. This paper focuses on several regularized MSVMs and extends the alternating direction method of multiplier (ADMM) to these MSVMs. Using a splitting technique, all considered MSVMs are written as two-block convex programs, for which the ADMM has global convergence guarantees. Numerical experiments on synthetic and real data demonstrate the high efficiency and accuracy of our algorithms.

**Keywords:** Alternating Direction Method of Multipliers, Support Vector Machine, Multiclass classification, Elastic Net, Group lasso, Supnorm

## 1 Introduction

The linear support vector machine (SVM) [6] aims to find a hyperplane to separate a set of data points. It was originally designed for binary classifications. Motivated by texture classification and gene expression analysis, which usually have a large number of variables but only a few relevant, certain sparsity regularizers such as the  $\ell_1$  penalty [4], need to be included in the SVM model to control the sparsity pattern of the solution and achieve both classification and variable selection. On the other hand, the given data points may belong to more than two classes. To handle the more complex multiclass problems, the binary SVM has been generalized to multiclass classifications [7].

The initially proposed multiclass SVM (MSVM) methods construct several binary classifiers, such as “one-against-one” [1], “one-against-rest” [2] and “directed acyclic graph SVM” [17]. These models are usually solved by considering their dual formulations, which are quadratic programs often with fewer variables and can be efficiently solved by quadratic programming methods. However, these MSVMs may suffer from data imbalance (i.e., some classes have much fewer data

points than others) which can result in inaccurate predictions. One alternative is to put all the data points together in one model, which results in the so-called “all-together” MSVMs; see [14] and references therein for the comparison of different MSVMs. The “all-together” MSVMs train multi-classifiers by solving one large optimization problem, whose dual formulation is also a quadratic program. In the applications of microarray classification, variable selection is important since most times only a few genes are closely related to certain diseases. Therefore some structure regularizers such as  $\ell_1$  penalty [19] and  $\ell_\infty$  penalty [23] need to be added to the MSVM models. With the addition of the structure regularizers, the dual problems of the aforementioned MSVMs can be difficult to formulate and hard to solve by standard second-order optimization methods.

In this paper, we focus on three “all-together” regularized MSVMs. Specifically, given a set of samples  $\{\mathbf{x}_i\}_{i=1}^n$  in  $p$ -dimensional space and each  $\mathbf{x}_i$  with a label  $y_i \in \{1, \dots, J\}$ , we solve the constrained optimization problem

$$\min_{\mathbf{W}, \mathbf{b}} \ell_G(\mathbf{W}, \mathbf{b}) + \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \phi(\mathbf{W}) + \frac{\lambda_3}{2} \|\mathbf{b}\|_2^2, \text{ s.t. } \mathbf{W}\mathbf{e} = \mathbf{0}, \mathbf{e}^\top \mathbf{b} = 0 \quad (1)$$

where

$$\ell_G(\mathbf{W}, \mathbf{b}) = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^J I(y_i \neq j) [b_j + \mathbf{w}_j^\top \mathbf{x}_i + 1]_+$$

is generalized hinge loss function;  $I(y_i \neq j)$  equals *one* if  $y_i \neq j$  and *zero* otherwise;  $[t]_+ = \max(0, t)$ ;  $\mathbf{w}_j$  denotes the  $j$ th column of  $\mathbf{W}$ ;  $\mathbf{e}$  denotes the vector of appropriate size with all *ones*;  $\|\mathbf{W}\|_1 = \sum_{i,j} |w_{ij}|$ ;  $\phi(\mathbf{W})$  is some regularizer specified below. Usually, the regularizer can promote the structure of the solution and also avoid overfitting problems when the training samples are far less than features. The constraints  $\mathbf{W}\mathbf{e} = \mathbf{0}, \mathbf{e}^\top \mathbf{b} = 0$  are imposed to eliminate redundancy in  $\mathbf{W}, \mathbf{b}$  and are also necessary to make the loss function  $\ell_G$  Fisher-consistent [16]. The solution of (1) gives  $J$  linear classifiers  $f_j(\mathbf{x}) = \mathbf{w}_j^\top \mathbf{x} + b_j$ ,  $j = 1, \dots, J$ . A new coming data point  $\mathbf{x}$  can be classified by the rule  $\text{class}(\mathbf{x}) = \arg\max_{1 \leq j \leq J} f_j(\mathbf{x})$ .

We consider the following three different forms of  $\phi(\mathbf{W})$ :

$$\text{elastic net: } \phi(\mathbf{W}) = \frac{1}{2} \|\mathbf{W}\|_F^2, \quad (2a)$$

$$\text{group Lasso: } \phi(\mathbf{W}) = \sum_{j=1}^p \|\mathbf{w}^j\|_2, \quad (2b)$$

$$\text{supnorm: } \phi(\mathbf{W}) = \sum_{j=1}^p \|\mathbf{w}^j\|_\infty, \quad (2c)$$

where  $\mathbf{w}^j$  denotes the  $j$ th row of  $\mathbf{W}$ . They fit to data with different structures and can be solved by a *unified* algorithmic framework. Note that we have added the term  $\frac{\lambda_3}{2} \|\mathbf{b}\|_2^2$  in (1). A positive  $\lambda_3$  will make our algorithm more efficient and easier to implement. The extra term usually does not affect the accuracy of classification and variable selection as shown in [14] for binary classifications. If  $\lambda_3$  happens to affect the accuracy, one can choose a tiny  $\lambda_3$ . Model (1) includes as special cases the models in [16] and [23] by letting  $\phi$  be the one in (2a) and (2c) respectively and setting  $\lambda_1 = \lambda_3 = 0$ . To the best of our knowledge, the regularizer (2b) has not been considered in MSVM before. It encourages group sparsity of the solution [22], and our experiments will show that (2b) can give similar results as those by (2c). Our main contributions are: (i) the development

of a unified algorithmic framework based on the ADMM that can solve MSVMs with the three different regularizers defined in (2); (ii) the proper use of the Woodbury matrix identity [13] which can reduce the size of the linear systems arising during the solution of (1); (iii) computational experiments on a variety of datasets that practically demonstrate that our algorithms can solve large-scale multiclass classification problems much faster than state-of-the-art second order methods.

We use  $\mathbf{e}$  and  $\mathbf{E}$  to denote a vector and a matrix with all *ones*, respectively.  $\mathbf{I}$  is used for an identity matrix. Their sizes are clear from the context.

The rest of the paper is organized as follows. Section 2 gives our algorithm for solving (1). Numerical results are given in section 3 on both synthetic and real data. Finally, section 4 concludes this paper.

## 2 Algorithms

In this section we extend ADMM into the general optimization problems described by (1). Due to lack of space we refer the reader to [3] for details of ADMM. We first consider (1) with  $\phi(\mathbf{W})$  defined in (2a) and then solve it with  $\phi(\mathbf{W})$  in (2b) and (2c) in a unified form. The parameter  $\lambda_3$  is always assumed positive. One can also transform the MSVMs to quadratic or second-order cone programs and use standard second-order methods to solve them. Nevertheless, these methods are computationally intensive for large-scale problems. As shown in section 3, ADMM is, in general, much faster than standard second-order methods.

### 2.1 ADMM for solving (1) with $\phi$ defined by (2a)

Introduce auxiliary variables  $\mathbf{A} = \mathbf{X}^\top \mathbf{W} + \mathbf{e}\mathbf{b}^\top + \mathbf{E}$  and  $\mathbf{U} = \mathbf{W}$ , where  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ . Using the above auxiliary variables we can equivalently write (1) with  $\phi(\mathbf{W})$  defined in (2a) as follows

$$\begin{aligned} \min \quad & \frac{1}{n} \sum_{i,j} c_{ij} [a_{ij}]_+ + \lambda_1 \|\mathbf{U}\|_1 + \frac{\lambda_2}{2} \|\mathbf{W}\|_F^2 + \frac{\lambda_3}{2} \|\mathbf{b}\|_2^2 \\ \text{s.t.} \quad & \mathbf{A} = \mathbf{X}^\top \mathbf{W} + \mathbf{e}\mathbf{b}^\top + \mathbf{E}, \quad \mathbf{U} = \mathbf{W}, \quad \mathbf{W}\mathbf{e} = \mathbf{0}, \quad \mathbf{e}^\top \mathbf{b} = 0. \end{aligned} \quad (3)$$

The augmented Lagrangian<sup>3</sup> of (3) is

$$\begin{aligned} \mathcal{L}_1(\mathbf{W}, \mathbf{b}, \mathbf{A}, \mathbf{U}, \mathbf{H}, \mathbf{A}) = & \frac{1}{n} \sum_{i,j} c_{ij} [a_{ij}]_+ + \lambda_1 \|\mathbf{U}\|_1 + \frac{\lambda_2}{2} \|\mathbf{W}\|_F^2 + \frac{\lambda_3}{2} \|\mathbf{b}\|_2^2 + \langle \mathbf{A}, \mathbf{W} - \mathbf{U} \rangle \\ & + \frac{\mu}{2} \|\mathbf{W} - \mathbf{U}\|_F^2 + \langle \mathbf{H}, \mathbf{X}^\top \mathbf{W} + \mathbf{e}\mathbf{b}^\top - \mathbf{A} + \mathbf{E} \rangle \\ & + \frac{\alpha}{2} \|\mathbf{X}^\top \mathbf{W} + \mathbf{e}\mathbf{b}^\top - \mathbf{A} + \mathbf{E}\|_F^2, \end{aligned} \quad (4)$$

where  $\mathbf{H}, \mathbf{A}$  are Lagrange multipliers and  $\alpha, \mu > 0$  are penalty parameters. The ADMM approach for (3) can be derived by minimizing  $\mathcal{L}_1$  alternatively with

<sup>3</sup> We do not include the constraints  $\mathbf{W}\mathbf{e} = \mathbf{0}$ ,  $\mathbf{e}^\top \mathbf{b} = 0$  in the augmented Lagrangian, but instead we include them in  $(\mathbf{W}, \mathbf{b})$ -subproblem; see the update (5a).

respect to  $(\mathbf{W}, \mathbf{b})$  and  $(\mathbf{A}, \mathbf{U})$  and updating the multipliers  $\mathbf{\Pi}, \mathbf{A}$ , namely, at iteration  $k$ ,

$$(\mathbf{W}^{(k+1)}, \mathbf{b}^{(k+1)}) = \underset{(\mathbf{W}, \mathbf{b}) \in \mathcal{D}}{\operatorname{argmin}} \mathcal{L}_1(\mathbf{W}, \mathbf{b}, \mathbf{A}^{(k)}, \mathbf{U}^{(k)}, \mathbf{\Pi}^{(k)}, \mathbf{A}^{(k)}), \quad (5a)$$

$$(\mathbf{A}^{(k+1)}, \mathbf{U}^{(k+1)}) = \underset{\mathbf{A}, \mathbf{U}}{\operatorname{argmin}} \mathcal{L}_1(\mathbf{W}^{(k+1)}, \mathbf{b}^{(k+1)}, \mathbf{A}, \mathbf{U}, \mathbf{\Pi}^{(k)}, \mathbf{A}^{(k)}), \quad (5b)$$

$$\mathbf{\Pi}^{(k+1)} = \mathbf{\Pi}^{(k)} + \alpha(\mathbf{X}^\top \mathbf{W}^{(k+1)} + \mathbf{e}(\mathbf{b}^{(k+1)})^\top - \mathbf{A}^{(k+1)} + \mathbf{E}), \quad (5c)$$

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \mu(\mathbf{W}^{(k+1)} - \mathbf{U}^{(k+1)}), \quad (5d)$$

where  $\mathcal{D} = \{(\mathbf{W}, \mathbf{b}) : \mathbf{W}\mathbf{e} = \mathbf{0}, \mathbf{e}^\top \mathbf{b} = 0\}$ . The updates (5c) and (5d) are simple. We next discuss how to solve (5a) and (5b).

**Solution of (5a):** Define  $\mathbf{P} = [\mathbf{I}; -\mathbf{e}^\top] \in \mathbb{R}^{J \times (J-1)}$ . Let  $\hat{\mathbf{W}}$  be the submatrix consisting of the first  $J-1$  columns of  $\mathbf{W}$  and  $\hat{\mathbf{b}}$  be the subvector consisting of the first  $J-1$  components of  $\mathbf{b}$ . Then it is easy to verify that  $\mathbf{W} = \hat{\mathbf{W}}\mathbf{P}^\top$ ,  $\mathbf{b} = \mathbf{P}\hat{\mathbf{b}}$  and problem (5a) is equivalent to the unconstrained optimization problem

$$\min_{\hat{\mathbf{W}}, \hat{\mathbf{b}}} \frac{\lambda_2}{2} \|\hat{\mathbf{W}}\mathbf{P}^\top\|_F^2 + \langle \mathbf{A}^{(k)}, \hat{\mathbf{W}}\mathbf{P}^\top \rangle + \frac{\lambda_3}{2} \|\hat{\mathbf{b}}^\top \mathbf{P}^\top\|_2^2 + \frac{\mu}{2} \|\hat{\mathbf{W}}\mathbf{P}^\top - \mathbf{U}^{(k)}\|_F^2 + \langle \mathbf{\Pi}^{(k)}, \mathbf{X}^\top \hat{\mathbf{W}}\mathbf{P}^\top + \mathbf{e}\hat{\mathbf{b}}^\top \mathbf{P}^\top \rangle + \frac{\alpha}{2} \|\mathbf{X}^\top \hat{\mathbf{W}}\mathbf{P}^\top + \mathbf{e}\hat{\mathbf{b}}^\top \mathbf{P}^\top - \mathbf{A}^{(k)} + \mathbf{E}\|_F^2. \quad (6)$$

The first-order optimality condition of (6) is the linear system

$$\begin{bmatrix} \alpha \mathbf{X}\mathbf{X}^\top + (\lambda_2 + \mu)\mathbf{I} & \alpha \mathbf{X}\mathbf{e} \\ \alpha \mathbf{e}^\top \mathbf{X}^\top & n\alpha + \lambda_3 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{W}} \\ \hat{\mathbf{b}}^\top \end{bmatrix} = \begin{bmatrix} (\mathbf{X}\mathbf{e} - \mathbf{A}^{(k)} + \mu\mathbf{U}^{(k)}) \mathbf{P}(\mathbf{P}^\top \mathbf{P})^{-1} \\ \mathbf{e}^\top (\alpha \mathbf{A}^{(k)} - \mathbf{\Pi}^{(k)} - \alpha \mathbf{E}) \mathbf{P}(\mathbf{P}^\top \mathbf{P})^{-1} \end{bmatrix}, \quad (7)$$

where  $\mathbf{e} = \alpha \mathbf{A}^{(k)} - \mathbf{\Pi}^{(k)} - \alpha \mathbf{E}$ . The size of (7) is  $(p+1) \times (p+1)$  and when  $p$  is small, we can afford to directly solve it. However, if  $p$  is large, even the iterative method for linear system (e.g., preconditioned conjugate gradient) can be very expensive. In the case of ‘‘large  $p$ , small  $n$ ’’, we can employ the *Woodbury matrix identity* (e.g., [13]) to efficiently solve (7). In particular, let  $\mathbf{D} = \operatorname{block\_diag}((\lambda_2 + \mu)\mathbf{I}, \lambda_3)$  and  $\mathbf{Z} = [\mathbf{X}; \mathbf{e}^\top]$ . Then the coefficient matrix of (7) is  $\mathbf{D} + \alpha \mathbf{Z}\mathbf{Z}^\top$ , and by the *Woodbury matrix identity*, we have  $\mathbf{P}(\mathbf{P}^\top \mathbf{P})^{-1} = [\mathbf{I}; \mathbf{0}] - \frac{1}{J}\mathbf{E}$  and

$$(\mathbf{D} + \alpha \mathbf{Z}\mathbf{Z}^\top)^{-1} = \mathbf{D}^{-1} - \alpha \mathbf{D}^{-1} \mathbf{Z}(\mathbf{I} + \alpha \mathbf{Z}^\top \mathbf{D}^{-1} \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{D}^{-1}.$$

Note  $\mathbf{D}$  is diagonal, and thus  $\mathbf{D}^{-1}$  is simple to compute.  $\mathbf{I} + \alpha \mathbf{Z}^\top \mathbf{D}^{-1} \mathbf{Z}$  is  $n \times n$  and positive definite. Hence, as  $n \ll p$ , (7) can be solved by solving a much smaller linear system and doing several matrix-matrix multiplications. In case of large  $n$  and  $p$ , one can perform a proximal gradient step to update  $\mathbf{W}$  and  $\mathbf{b}$ , which results in a proximal-ADMM [8]. To the best of our knowledge, this is the first time that the Woodbury matrix identity is used to substantially reduce<sup>4</sup> the computational work and allow ADMM to efficiently solve large-scale multiclass SVMs. Solve (7) by multiplying  $(\mathbf{D} + \alpha \mathbf{Z}\mathbf{Z}^\top)^{-1}$  to both sides. Letting  $\mathbf{W}^{(k+1)} = \hat{\mathbf{W}}\mathbf{P}^\top$  and  $\mathbf{b}^{(k+1)} = \mathbf{P}\hat{\mathbf{b}}$  gives the solution of (5a).

<sup>4</sup> For the case of  $n \ll p$ , we found that using the Woodbury matrix identity can be about 100 times faster than preconditioned conjugate gradient (pcg) with moderate tolerance  $10^{-6}$  for the solving the linear system (7).

**Solution of (5b):** Note that  $\mathbf{A}$  and  $\mathbf{U}$  are independent of each other as  $\mathbf{W}$  and  $\mathbf{b}$  are fixed. Hence we can separately update  $\mathbf{A}$  and  $\mathbf{U}$  by

$$\mathbf{A}^{(k+1)} = \underset{\mathbf{A}}{\operatorname{argmin}} \frac{1}{n} \sum_{i,j} c_{ij} [a_{ij}]_+ + \frac{\alpha}{2} \|\mathbf{X}^\top \mathbf{W}^{(k+1)} + \mathbf{e}(\mathbf{b}^{(k+1)})^\top + \frac{1}{\alpha} \mathbf{\Pi}^{(k)} + \mathbf{E} - \mathbf{A}\|_F^2$$

$$\mathbf{U}^{(k+1)} = \underset{\mathbf{U}}{\operatorname{argmin}} \lambda_1 \|\mathbf{U}\|_1 + \frac{\mu}{2} \|\mathbf{W}^{(k+1)} + \frac{1}{\mu} \mathbf{\Lambda}^{(k)} - \mathbf{U}\|_F^2.$$

Both the above problems are separable and have closed form solutions

$$a_{ij}^{(k+1)} = \mathcal{T}_{\frac{c_{ij}}{n\alpha}} \left( \left( \mathbf{X}^\top \mathbf{W}^{(k+1)} + \mathbf{e}(\mathbf{b}^{(k+1)})^\top + \frac{1}{\alpha} \mathbf{\Pi}^{(k)} + \mathbf{E} \right)_{ij} \right), \quad \forall i, j, \quad (8)$$

$$u_{ij}^{(k+1)} = \mathcal{S}_{\frac{\lambda_1}{\mu}} \left( \left( \mathbf{W}^{(k+1)} + \frac{1}{\mu} \mathbf{\Lambda}^{(k)} \right)_{ij} \right), \quad \forall i, j, \quad (9)$$

where

$$\mathcal{T}_\nu(\delta) = \begin{cases} \delta - \nu, & \delta > \nu, \\ 0, & 0 \leq \delta \leq \nu, \\ \delta, & \delta < 0, \end{cases}$$

and  $\mathcal{S}_\nu(\delta) = \operatorname{sign}(\delta) \max(0, |\delta| - \nu)$ . Putting the above discussions together, we have Algorithm 1 for solving (1) with  $\phi$  defined in (2a).

---

**Algorithm 1:** ADMM for (1) with  $\phi(\mathbf{W})$  in (2a)

---

**Input:**  $n$  sample-label pairs  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ .

**Choose:**  $\alpha, \mu > 0$  and  $(\mathbf{W}_0, \mathbf{b}_0, \mathbf{A}_0, \mathbf{U}_0, \mathbf{\Pi}_0, \mathbf{\Lambda}_0), k = 0$ .

**while not converge do**

Solve (7); let  $\mathbf{W}^{(k+1)} = \hat{\mathbf{W}}\mathbf{P}^\top$  and  $\mathbf{b}^{(k+1)} = \mathbf{P}\hat{\mathbf{b}}$ ;  
Update  $\mathbf{A}^{(k+1)}$  and  $\mathbf{U}^{(k+1)}$  by (8) and (9);  
Update  $\mathbf{\Pi}^{(k+1)}$  and  $\mathbf{\Lambda}^{(k+1)}$  by (5c) and (5d);  
Let  $k = k + 1$

---

## 2.2 ADMM for solving (1) with $\phi$ defined by (2b) and (2c)

Firstly, we write (1) with  $\phi(\mathbf{W})$  defined in (2b) and (2c) in the unified form of

$$\min_{\mathbf{W}, \mathbf{b}} \ell_G(\mathbf{W}, \mathbf{b}) + \lambda_1 \|\mathbf{W}\|_1 + \sum_{j=1}^p \lambda_2 \|\mathbf{w}^j\|_q + \frac{\lambda_3}{2} \|\mathbf{b}\|_2^2, \quad \text{s.t. } \mathbf{W}\mathbf{e} = \mathbf{0}, \mathbf{e}^\top \mathbf{b} = 0, \quad (10)$$

where  $q = 2$  for (2b) and  $q = \infty$  for (2c). Introducing auxiliary variables  $\mathbf{A} = \mathbf{X}^\top \mathbf{W} + \mathbf{e}\mathbf{b}^\top + \mathbf{E}$ ,  $\mathbf{U} = \mathbf{W}$ , and  $\mathbf{V} = \mathbf{W}$ , we can write (10) equivalently to

$$\begin{aligned} \min \quad & \frac{1}{n} \sum_{i,j} c_{ij} [a_{ij}]_+ + \lambda_1 \|\mathbf{U}\|_1 + \sum_{j=1}^p \lambda_2 \|\mathbf{v}^j\|_q + \frac{\lambda_3}{2} \|\mathbf{b}\|_2^2 \\ \text{s.t.} \quad & \mathbf{A} = \mathbf{X}^\top \mathbf{W} + \mathbf{e}\mathbf{b}^\top + \mathbf{E}, \quad \mathbf{U} = \mathbf{W}, \quad \mathbf{V} = \mathbf{W}, \quad \mathbf{W}\mathbf{e} = \mathbf{0}, \quad \mathbf{e}^\top \mathbf{b} = 0. \end{aligned} \quad (11)$$

The augmented Lagrangian of (11) is

$$\begin{aligned} \mathcal{L}_2(\mathbf{W}, \mathbf{b}, \mathbf{A}, \mathbf{U}, \mathbf{V}, \mathbf{\Pi}, \mathbf{\Lambda}, \mathbf{\Gamma}) = & \frac{1}{n} \sum_{i,j} c_{ij} [a_{ij}]_+ + \lambda_1 \|\mathbf{U}\|_1 + \sum_{j=1}^p \lambda_2 \|\mathbf{v}^j\|_q + \frac{\lambda_3}{2} \|\mathbf{b}\|_2^2 \\ & + \langle \mathbf{\Pi}, \mathbf{X}^\top \mathbf{W} + \mathbf{e}\mathbf{b}^\top - \mathbf{A} + \mathbf{E} \rangle + \frac{\alpha}{2} \|\mathbf{X}^\top \mathbf{W} + \mathbf{e}\mathbf{b}^\top - \mathbf{A} + \mathbf{E}\|_F^2 \\ & + \langle \mathbf{\Lambda}, \mathbf{W} - \mathbf{U} \rangle + \frac{\mu}{2} \|\mathbf{W} - \mathbf{U}\|_F^2 + \langle \mathbf{\Gamma}, \mathbf{W} - \mathbf{V} \rangle + \frac{\nu}{2} \|\mathbf{W} - \mathbf{V}\|_F^2, \end{aligned} \quad (12)$$

where  $\boldsymbol{\Pi}, \mathbf{A}, \boldsymbol{\Gamma}$  are Lagrange multipliers and  $\alpha, \mu, \nu > 0$  are penalty parameters. The ADMM updates for (11) can be derived as

$$(\mathbf{W}^{(k+1)}, \mathbf{b}^{(k+1)}) = \underset{(\mathbf{W}, \mathbf{b}) \in \mathcal{D}}{\operatorname{argmin}} \mathcal{L}_2(\mathbf{W}, \mathbf{b}, \mathbf{A}^{(k)}, \mathbf{U}^{(k)}, \mathbf{V}^{(k)}, \boldsymbol{\Pi}^{(k)}, \mathbf{A}^{(k)}, \boldsymbol{\Gamma}^{(k)}) \quad (13a)$$

$$(\mathbf{A}^{(k+1)}, \mathbf{U}^{(k+1)}, \mathbf{V}^{(k+1)}) = \underset{\mathbf{A}, \mathbf{U}, \mathbf{V}}{\operatorname{argmin}} \mathcal{L}_2(\mathbf{W}^{(k+1)}, \mathbf{b}^{(k+1)}, \mathbf{A}, \mathbf{U}, \mathbf{V}, \boldsymbol{\Pi}^{(k)}, \mathbf{A}^{(k)}, \boldsymbol{\Gamma}^{(k)}) \quad (13b)$$

$$\boldsymbol{\Pi}^{(k+1)} = \boldsymbol{\Pi}^{(k)} + \alpha(\mathbf{X}^\top \mathbf{W}^{(k+1)} + \mathbf{e}(\mathbf{b}^{(k+1)})^\top - \mathbf{A}^{(k+1)} + \mathbf{E}) \quad (13c)$$

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \mu(\mathbf{W}^{(k+1)} - \mathbf{U}^{(k+1)}), \quad (13d)$$

$$\boldsymbol{\Gamma}^{(k+1)} = \boldsymbol{\Gamma}^{(k)} + \nu(\mathbf{W}^{(k+1)} - \mathbf{V}^{(k+1)}). \quad (13e)$$

The subproblem (13a) can be solved in a similar way as discussed in section 2.1. Specifically, first obtain  $(\hat{\mathbf{W}}, \hat{\mathbf{b}})$  by solving

$$\begin{bmatrix} \alpha \mathbf{X} \mathbf{X}^\top + (\nu + \mu) \mathbf{I} & \alpha \mathbf{X} \mathbf{e} \\ \alpha \mathbf{e}^\top \mathbf{X}^\top & n\alpha + \lambda_3 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{W}} \\ \hat{\mathbf{b}}^\top \end{bmatrix} = \begin{bmatrix} (\mathbf{X} \boldsymbol{\Xi} - \mathbf{A}^{(k)} - \boldsymbol{\Gamma}^{(k)} + \mu \mathbf{U}^{(k)} + \nu \mathbf{V}^{(k)}) \mathbf{P} (\mathbf{P}^\top \mathbf{P})^{-1} \\ \mathbf{e}^\top (\alpha \mathbf{A}^{(k)} - \boldsymbol{\Pi}^{(k)} - \alpha \mathbf{E}) \mathbf{P} (\mathbf{P}^\top \mathbf{P})^{-1} \end{bmatrix}, \quad (14)$$

where  $\boldsymbol{\Xi} = \alpha \mathbf{A}^{(k)} - \boldsymbol{\Pi}^{(k)} - \alpha \mathbf{E}$  and then let  $\mathbf{W}^{(k+1)} = \hat{\mathbf{W}} \mathbf{P}^\top$ ,  $\mathbf{b}^{(k+1)} = \mathbf{P} \hat{\mathbf{b}}$ . To solve (13b) note that  $\mathbf{A}, \mathbf{U}$  and  $\mathbf{V}$  are independent of each other and can be updated separately. The update of  $\mathbf{A}$  and  $\mathbf{U}$  is similar to that described in section 2.1. We next discuss how to update  $\mathbf{V}$  by solving the problem

$$\mathbf{V}^{(k+1)} = \underset{\mathbf{V}}{\operatorname{argmin}} \sum_{j=1}^p \lambda_2 \|\mathbf{v}^j\|_q + \frac{\nu}{2} \|\mathbf{W}^{(k+1)} + \frac{1}{\nu} \boldsymbol{\Gamma}^{(k)} - \mathbf{V}\|_F^2 \quad (15)$$

Let  $\mathbf{Z} = \mathbf{W}^{(k+1)} + \frac{1}{\nu} \boldsymbol{\Gamma}^{(k)}$ . According to [22], the solution of (15) for  $q = 2$  is

$$\left(\mathbf{v}^{(k+1)}\right)^j = \begin{cases} \mathbf{0}, & \|\mathbf{z}^j\|_2 \leq \frac{\lambda_2}{\nu} \\ \frac{\|\mathbf{z}^j\|_2 - \lambda_2/\nu}{\|\mathbf{z}^j\|_2} \mathbf{z}^j, & \text{otherwise} \end{cases}, \forall j. \quad (16)$$

For  $q = \infty$ , the solution of (15) can be computed via Algorithm 2 (see [5] for details). Putting the above discussions together, we have Algorithm 3 for solving (1) with  $\phi(\mathbf{W})$  given by (2b) and (2c).

### 2.3 Convergence results

Let us denote the  $k$ th iteration of the objectives of (3) and (11) as

$$F_1^{(k)} = F_1(\mathbf{W}^{(k)}, \mathbf{b}^{(k)}, \mathbf{A}^{(k)}, \mathbf{U}^{(k)}), \quad F_2^{(k)} = F_2(\mathbf{W}^{(k)}, \mathbf{b}^{(k)}, \mathbf{A}^{(k)}, \mathbf{U}^{(k)}, \mathbf{V}^{(k)}), \quad (17)$$

and define

$$\begin{aligned} \mathbf{Z}_1^{(k)} &= (\mathbf{W}^{(k)}, \mathbf{b}^{(k)}, \mathbf{A}^{(k)}, \mathbf{U}^{(k)}, \boldsymbol{\Pi}^{(k)}, \mathbf{A}^{(k)}), \\ \mathbf{Z}_2^{(k)} &= (\mathbf{W}^{(k)}, \mathbf{b}^{(k)}, \mathbf{A}^{(k)}, \mathbf{U}^{(k)}, \mathbf{V}^{(k)}, \boldsymbol{\Pi}^{(k)}, \mathbf{A}^{(k)}, \boldsymbol{\Gamma}^{(k)}). \end{aligned}$$

**Theorem 1.** *Let  $\{\mathbf{Z}_1^{(k)}\}$  and  $\{\mathbf{Z}_2^{(k)}\}$  be the sequences generated by (5) and (13), respectively. Then  $F_1^{(k)} \rightarrow F_1^*$ ,  $F_2^{(k)} \rightarrow F_2^*$ , and  $\|\mathbf{X}^\top \mathbf{W}^{(k)} + \mathbf{e}(\mathbf{b}^{(k)})^\top + \mathbf{E} - \mathbf{A}^{(k)}\|_F, \|\mathbf{W}^{(k)} - \mathbf{U}^{(k)}\|_F, \|\mathbf{W}^{(k)} - \mathbf{V}^{(k)}\|_F$  all converge to zero, where  $F_1^*$  and  $F_2^*$  are the optimal objective values of (3) and (11), respectively. In addition, if  $\lambda_2 > 0, \lambda_3 > 0$  in (3), then  $\mathbf{Z}_1^{(k)}$  converges linearly.*

The proof is based on [10,8] and due to the lack of space we omit it.

---

**Algorithm 2:** Algorithm for solving (15) when  $q = \infty$

---

Let  $\tilde{\lambda} = \frac{\lambda_2}{\nu}$  and  $\mathbf{Z} = \mathbf{W}^{(k+1)} + \frac{1}{\nu} \mathbf{\Gamma}^{(k)}$ .  
**for**  $j = 1, \dots, p$  **do**  
    Let  $\mathbf{v} = \mathbf{z}^j$ ;  
    **if**  $\|\mathbf{v}\|_1 \leq \tilde{\lambda}$  **then**  
        Set  $(\mathbf{v}^{(k+1)})^j = \mathbf{0}$ .  
    **else**  
        Let  $\mathbf{u}$  be the sorted absolute value vector of  $\mathbf{v}$ :  $u_1 \geq u_2 \geq \dots \geq u_J$ ;  
        Find  $\hat{r} = \max \left\{ r : \tilde{\lambda} - \sum_{t=1}^r (u_t - u_r) > 0 \right\}$   
        Let  $v_{ji}^{(k+1)} = \text{sign}(v_i) \min \left( |v_i|, (\sum_{t=1}^{\hat{r}} u_t - \tilde{\lambda}) / \hat{r} \right), \forall i$ .

---

### 3 Numerical results

We now test the three different regularizers in (2) on two sets of synthetic data and two sets of real data. As shown in [19] the  $L_1$  regularized MSVM works better than the standard “one-against-rest” MSVM in both classification and variable selection. Hence, we choose to only compare the three regularized MSVMs. The ADMM algorithms discussed in section 2 are used to solve the three models. Until the preparation of this paper, we did not find much work on designing specific algorithms to solve the regularized MSVMs except [19] which uses a path-following algorithm to solve the  $L_1$  MSVM. To illustrate the efficiency of ADMM, we compare it with Sedumi [18] which is a second-order method. We call Sedumi in the CVX environment [12].

---

**Algorithm 3:** ADMM for (1) with  $\phi(\mathbf{W})$  in (2b) and (2c)

---

**Input:**  $n$  sample-label pairs  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ .  
**Choose:**  $\alpha, \mu, \nu > 0$ , set  $k = 0$  and initialize  $(\mathbf{W}_0, \mathbf{b}_0, \mathbf{A}_0, \mathbf{U}_0, \mathbf{V}_0, \mathbf{\Pi}_0, \mathbf{A}_0, \mathbf{\Gamma}_0)$ .  
**while not converge do**  
    Solve (14); let  $\mathbf{W}^{(k+1)} = \hat{\mathbf{W}}\mathbf{P}^\top$  and  $\mathbf{b}^{(k+1)} = \mathbf{P}\hat{\mathbf{b}}$ ;  
    Update  $\mathbf{A}^{(k+1)}$  and  $\mathbf{U}^{(k+1)}$  by (8) and (9);  
    Update  $\mathbf{V}^{(k+1)}$  by (16) if  $q = 2$  and by Algorithm 2 if  $q = \infty$ ;  
    Update  $\mathbf{\Pi}^{(k+1)}$ ,  $\mathbf{A}^{(k+1)}$  and  $\mathbf{\Gamma}^{(k+1)}$  by (13c), (13d) and (13e);

---

#### 3.1 Implementation details

All our code was written in MATLAB, except the part of Algorithm 2 which was written in C with MATLAB interface. We used  $\lambda_3 = 1$  for all three models. In our experiments, we found that the penalty parameters were very important

for the speed of ADMM. By running a large set of random tests, we chose  $\alpha = \frac{50J}{n}$ ,  $\mu = \sqrt{pJ}$  in (4) and  $\alpha = \frac{50J}{n}$ ,  $\mu = \nu = \sqrt{pJ}$  in (12). Origins were used as the starting points. As did in [21], we terminated ADMM for (3), that is, (1) with  $\phi(\mathbf{W})$  in (2a), if

$$\max \left\{ \frac{|F_1^{(k+1)} - F_1^{(k)}|}{1 + F_1^{(k)}}, \frac{\|\mathbf{W}^{(k)} - \mathbf{U}^{(k)}\|_F}{\sqrt{pJ}}, \frac{\|\mathbf{X}^\top \mathbf{W}^{(k)} + \mathbf{e}(\mathbf{b}^{(k)})^\top + \mathbf{E} - \mathbf{A}^{(k)}\|_F}{\sqrt{nJ}} \right\} \leq 10^{-5},$$

and ADMM for (11), that is, (1) with  $\phi(\mathbf{W})$  in (2b) and (2c), if

$$\max \left\{ \frac{|F_2^{(k+1)} - F_2^{(k)}|}{1 + F_2^{(k)}}, \frac{\|\mathbf{X}^\top \mathbf{W}^{(k)} + \mathbf{e}(\mathbf{b}^{(k)})^\top + \mathbf{E} - \mathbf{A}^{(k)}\|_F}{\sqrt{nJ}}, \frac{\|\mathbf{W}^{(k)} - \mathbf{U}^{(k)}\|_F}{\sqrt{pJ}}, \frac{\|\mathbf{W}^{(k)} - \mathbf{V}^{(k)}\|_F}{\sqrt{pJ}} \right\} \leq 10^{-5}.$$

In addition, we set a maximum number of iterations  $maxit = 5000$  for ADMM. Default settings were used for Sedumi. All the tests were performed on a PC with an i5-2500 CPU and 3-GB RAM and running 32-bit Windows XP.

**Table 1.** Results of different models solved by ADMM and Sedumi on a five-class example with synthetic data. The numbers in parentheses are standard errors.

Models	ADMM					Sedumi				
	Accuracy	time	CZ	IZ	NR	Accuracy	time	CZ	IZ	NR
elastic net	0.597(0.012)	0.184	39.98	0.92	2.01	0.592(0.013)	0.378	39.94	1.05	2.03
group Lasso	0.605(0.006)	0.235	34.94	0.00	3.14	0.599(0.008)	2.250	33.85	0.02	3.25
supnorm	0.606(0.006)	0.183	39.84	0.56	2.08	0.601(0.008)	0.638	39.49	0.61	2.21

### 3.2 Synthetic data

The first test is a five-class example with each sample  $\mathbf{x}$  in a 10-dimensional space. The data was generated in the following way: for each class  $j$ , the first two components  $(x_1, x_2)$  were generated from the mixture Gaussian distribution  $\mathcal{N}(\boldsymbol{\mu}_j, 2\mathbf{I})$  where for  $j = 1, \dots, 5$ ,

$$\boldsymbol{\mu}_j = 2[\cos((2j - 1)\pi/5), \sin((2j - 1)\pi/5)],$$

and the remaining eight components were independently generated from standard Gaussian distribution. This kind of data was also tested in [19,23]. We first chose best parameters for each model by generating  $n = 200$  samples for training and another  $n = 200$  samples for tuning parameters. For elastic net, we fixed  $\lambda_2 = 1$  since it is not sensitive and then searched the best  $\lambda_1$  over  $\mathcal{C} = \{0, 0.001, 0.01 : 0.01 : 0.1, 0.15, 0.20, 0.25, 0.30\}$ . The parameters  $\lambda_1$  and  $\lambda_2$  for group Lasso and supnorm were selected via a grid search over  $\mathcal{C} \times \mathcal{C}$ . With the tuned parameters, we compared ADMM and Sedumi on  $n = 200$  randomly generated training samples and  $n' = 50,000$  random testing samples, and the whole process was independently repeated 100 times. The performance of the compared models and algorithms were measured by accuracy (i.e.,  $\frac{\text{number of correctly predicted}}{\text{total number}}$ ), running time (sec), the number of correct zeros (CZ), the number of incorrect zeros (IZ) and the number of non-zero rows (NR). We counted CZ, IZ and NR from the truncated solution  $\mathbf{W}^t$ , which was obtained

from the output solution  $\mathbf{W}$  such that  $w_{ij}^t = 0$  if  $|w_{ij}| \leq 10^{-3} \max_{i,j} |w_{ij}|$  and  $w_{ij}^t = w_{ij}$  otherwise. The average results are shown in Table 1, from which we can see that ADMM produces similar results as those by Sedumi within less time. Elastic net makes slightly lower prediction accuracy than that by the other two models.

**Table 2.** Results of different models solved by ADMM and Sedumi on a four-class example with synthetic data. The numbers in the parentheses are corresponding standard errors.

Models	ADMM							Sedumi						
	Accuracy	time	IZ	NZ1	NZ2	NZ3	NZ4	Accuracy	time	IZ	NZ1	NZ2	NZ3	NZ4
Correlation $\rho = 0$														
elastic net	0.977(0.006)	0.27	13.8	37.6	36.9	36.8	37.0	0.950(0.013)	3.75	11.0	40.2	40.0	39.5	40.4
group Lasso	0.931(0.020)	0.46	30.4	33.7	33.4	33.2	33.2	0.857(0.022)	12.13	40.5	31.8	31.6	31.8	31.7
supnorm	0.924(0.025)	0.52	32.6	36.6	36.1	36.4	36.2	0.848(0.020)	13.93	46.6	34.2	33.8	33.7	33.5
Correlation $\rho = 0.8$														
elastic net	0.801(0.018)	0.19	24.1	29.6	29.7	30.6	29.6	0.773(0.036)	3.74	15.7	35.4	36.3	36.0	35.7
group Lasso	0.761(0.023)	0.38	64.0	21.4	21.2	21.3	21.2	0.654(0.023)	12.30	89.7	17.3	17.6	17.5	17.3
supnorm	0.743(0.023)	0.45	63.1	34.1	34.0	33.9	34.2	0.667(0.016)	14.01	79.8	35.3	35.3	35.3	35.2

The second test is a four-class example with each sample in  $p$ -dimensional space. The data in class  $j$  was generated from the mixture Gaussian distribution  $\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), j = 1, 2, 3, 4$ . The mean vectors and covariance matrices are  $\boldsymbol{\mu}_2 = -\boldsymbol{\mu}_1, \boldsymbol{\mu}_4 = -\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_4 = \boldsymbol{\Sigma}_3$ , and

$$\boldsymbol{\mu}_1 = (\underbrace{1, \dots, 1}_s, \underbrace{0, \dots, 0}_{p-s})^\top, \quad \boldsymbol{\mu}_3 = (\underbrace{0, \dots, 0}_{s/2}, \underbrace{1, \dots, 1}_s, \underbrace{0, \dots, 0}_{p-3s/2})^\top,$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} \rho \mathbf{E}_{s \times s} + (1 - \rho) \mathbf{I}_{s \times s} & \\ & \mathbf{I}_{(p-s) \times (p-s)} \end{bmatrix},$$

$$\boldsymbol{\Sigma}_3 = \begin{bmatrix} \mathbf{I}_{\frac{s}{2} \times \frac{s}{2}} & & \\ & \rho \mathbf{E}_{s \times s} + (1 - \rho) \mathbf{I}_{s \times s} & \\ & & \mathbf{I}_{(p-\frac{3s}{2}) \times (p-\frac{3s}{2})} \end{bmatrix}.$$

This kind of data was also tested in [20,21] for binary classifications. We took  $p = 500, s = 30$  and  $\rho = 0, 0.8$  in this test. As did in last test, the best parameters for all models were tuned by first generating  $n = 100$  training samples and another  $n = 100$  validation samples. Then we compared the different models solved by ADMM and Sedumi with the selected parameters on  $n = 100$  randomly generated training samples and  $n' = 20,000$  random testing samples. The comparison was independently repeated 100 times. The performance of different models and algorithms were measured by prediction accuracy, running time (sec), the number of incorrect zeros (IZ), the number of nonzeros in each column (NZ1, NZ2, NZ3, NZ4), where IZ, NZ1, NZ2, NZ3, NZ4 were counted in a similar way as that in last test by first truncating the output solution  $\mathbf{W}$ . Table 2 lists the average results, from which we can see that the elastic net MSVM tends to give best predictions. ADMM is much faster than Sedumi, and interestingly, ADMM

also gives higher prediction accuracies than those by Sedumi. This is probably because the solutions given by Sedumi are sparser and have more IZs than those by ADMM.

**Table 3.** Original distributions of SRBCT and leukemia data sets

Data set	SRBCT					leukemia			
	NB	RMS	BL	EWS	total	B-ALL	T-ALL	AML	total
Training	12	20	8	23	63	19	8	11	38
Testing	6	5	3	6	20	19	1	14	34

### 3.3 Real data

This subsection tests the three different MSVMs on microarray classifications. Two real data sets were used. One is the children cancer data set in [15], which used cDNA gene expression profiles and classified the small round blue cell tumors (SRBCTs) of childhood into four classes: neuroblastoma (NB), rhabdomyosarcoma (RMS), Burkitt lymphomas (BL) and the Ewing family of tumors (EWS). The other is the leukemia data set in [11], which used gene expression monitoring and classified the acute leukemias into three classes: B-cell acute lymphoblastic leukemia (B-ALL), T-cell acute lymphoblastic leukemia (T-ALL) and acute myeloid leukemia (AML). The original distributions of the two data sets are given in Table 3. Both the two data sets have been tested before on certain MSVMs for gene selection; see [19,23] for example.

Each observation in the SRBCT dataset has dimension of  $p = 2308$ , namely, there are 2308 gene profiles. We first standardized the original training data in the following way. Let  $\mathbf{X}^o = [\mathbf{x}_1^o, \dots, \mathbf{x}_n^o]$  be the original data matrix. The standardized matrix  $\mathbf{X}$  was obtained by

$$x_{gj} = \frac{x_{gj}^o - \text{mean}(x_{g1}^o, \dots, x_{gn}^o)}{\text{std}(x_{g1}^o, \dots, x_{gn}^o)}, \quad \forall g, j.$$

Similar normalization was done to the original testing data. Then we selected the best parameters of each model by three-fold cross validation on the standardized training data. The search range of the parameters is the same as that in the synthetic data tests. Finally, we put the standardized training and testing data sets together and randomly picked 63 observations for training and the remaining 20 ones for testing. The average prediction accuracy, running time (sec), number of non-zeros (NZ) and number of nonzero rows (NR) of 100 independent trials are reported in Table 4, from which we can see that all models give similar prediction accuracies. ADMM produced similar accuracies as those by Sedumi within less time while Sedumi tends to give sparser solutions because Sedumi is a second-order method and more accurately solves the problems.

The leukemia data set has  $p = 7,129$  gene profiles. We standardized the original training and testing data in the same way as that in last test. Then we rank all genes on the standardized training data by the method used in [9]. Specifically, let  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$  be the standardized data matrix. The relevance

**Table 4.** Results of different models solved by ADMM and Sedumi on SRBCT and Leukemia data sets

Data	Models	ADMM				Sedumi			
		Accuracy	time	NZ	NR	Accuracy	time	NZ	NR
SRBCT	elastic net	0.996(0.014)	1.738	305.71	135.31	0.989(0.022)	8.886	213.67	96.71
	group Lasso	0.995(0.016)	2.116	524.88	137.31	0.985(0.028)	42.241	373.44	96.27
	supnorm	0.996(0.014)	3.269	381.47	114.27	0.990(0.021)	88.468	265.06	80.82
Leukemia	elastic net	0.908(0.041)	1.029	571.56	271.85	0.879(0.048)	30.131	612.16	291.71
	group Lasso	0.908(0.045)	2.002	393.20	150.61	0.838(0.072)	76.272	99.25	44.14
	supnorm	0.907(0.048)	2.211	155.93	74.60	0.848(0.069)	121.893	86.03	41.78

measure for gene  $g$  is defined as follows:

$$R(g) = \frac{\sum_{i,j} I(y_i = j)(m_g^j - m_g)}{\sum_{i,j} I(y_i = j)(x_{gi} - m_g^j)}, \quad g = 1, \dots, p,$$

where  $m_g$  denotes the mean of  $\{x_{g1}, \dots, x_{gn}\}$  and  $m_g^j$  denotes the mean of  $\{x_{gi} : y_i = j\}$ . According to  $R(g)$ , we selected the 3,571 most significant genes. Finally, we put the processed training and testing data together and randomly chose 38 samples for training and the remaining ones for testing. The process was independently repeated 100 times. Table 4 tabulates the average results, which show that all three models give similar prediction accuracies. ADMM gave better prediction accuracies than those given by Sedumi within far less time. The relatively lower accuracies given by Sedumi may be because it selected too few genes to explain the diseases.

## 4 Conclusion

We have developed an efficient unified algorithmic framework for using ADMM to solve regularized MSVS. By effectively using the Woodbury matrix identity we have substantially reduced the computational effort required to solve large-scale MSVMS. Numerical experiments on both synthetic and real data demonstrate the efficiency of ADMM by comparing it with the second-order method Sedumi.

## References

1. Bishop C.: Pattern Recognition and Machine Learning, Springer-Verlag, New York (2006).
2. Bottou L., Cortes C., Denker J.S., Drucker H., Guyon I., Jackel L.D., LeCun Y., Muller U.A., Sackinger E., Simard P., et al.: Comparison of classifier methods: a case study in handwritten digit recognition. In Proceedings of the 12th IAPR International Conference on Pattern Recognition, volume 2, pages 77–82 (1994).
3. Boyd S., Parikh N., Chu E., Peleato B., and Eckstein J.: Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine Learning, 3(1):1–122 (2010).

4. Bradley P.S. and Mangasarian O.L.: Feature selection via concave minimization and support vector machines. In Proceedings of the Fifteenth International Conference of Machine Learning (ICML'98), pages 82–90 (1998).
5. Chen X., Pan W., Kwok J.T., and Carbonell J.G.: Accelerated gradient method for multi-task sparse learning problem. In Proceedings of the Ninth International Conference on Data Mining (ICDM'09), pages 746–751. IEEE (2009).
6. Cortes C. and Vapnik V.: Support-vector networks. *Machine learning*, 20(3):273–297 (1995).
7. Crammer K. and Singer Y. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2: 265–292 (2002).
8. Deng W. and Yin W.: On the global and linear convergence of the generalized alternating direction method of multipliers. *Rice technical report TR12-14* (2012).
9. Dudoit S., Fridlyand J., and Speed T.P.: Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American statistical association*, 97(457):77–87 (2002).
10. Glowinski R.: *Numerical methods for nonlinear variational problems*. Springer Verlag (2008).
11. Golub T.R., Slonim D.K., Tamayo P., Huard C., Gaasenbeek M., Mesirov J.P., Coller H., Loh M.L., Downing J.R., Caligiuri M.A., Bloomfield C.D., and Lander E.S.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537 (1999).
12. Grant M. and Boyd S.: *CVX - Matlab software for disciplined convex programming, version 2.1*. <http://cvxr.com/cvx> (2014).
13. Hager W.W.: Updating the inverse of a matrix. *SIAM Review*, 31:221–239 (1989).
14. Hsu C.W. and Lin C.J.: A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425 (2002).
15. Khan J., Wei J.S., Ringnér M., Saal L.H., Ladanyi M., Westermann F., Berthold F., Schwab M., Antonescu C.R., Peterson C., et al.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673–679 (2001).
16. Lee Y., Y. Lin, and Wahba G.: Multicategory support vector machines. *Journal of the American Statistical Association*, 99(465):67–81 (2004).
17. Platt J.C., Cristianini N., and Shawe-Taylor J.: Large margin dags for multiclass classification. *Advances in neural information processing systems*, 12(3):547–553 (2000).
18. Sturm J.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653 (1999).
19. Wang L. and Shen, X.: On  $L_1$ -norm multiclass support vector machines. *Journal of the American Statistical Association*, 102(478):583–594 (2007).
20. Wang L., Zhu J., and Zou, H.: Hybrid huberized support vector machines for microarray classification and gene selection. *Bioinformatics*, 24(3):412–419 (2008).
21. Ye G.B., Chen Y., and Xie X.: Efficient variable selection in support vector machines via the alternating direction method of multipliers. In Proceedings of the International Conference on Artificial Intelligence and Statistics (2011).
22. Yuan M. and Lin, Y.: Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67 (2006).
23. Zhang H., Liu Y., Wu Y., and Zhu J.: Variable selection for the multicategory svm via adaptive sup-norm regularization. *Electronic Journal of Statistics*, 2:149–167 (2008).