Efficient variable selection in support vector machines via the alternating direction method of multipliers

Gui-Bo Ye
University of California, Irvine

Yifei Chen
University of California, Irvine

Xiaohui Xie
University of California, Irvine

Abstract

The support vector machine (SVM) is a widely used tool for classification. Although commonly understood as a method of finding the maximum-margin hyperplane, it can also be formulated as a regularized function estimation problem, corresponding to a hinge loss function plus an ℓ_2-norm regulation term. The doubly regularized support vector machine (DrSVM) is a variant of the standard SVM, which introduces an additional ℓ_1-norm regularization term on the fitted coefficients. The combined ℓ_1 and ℓ_2 regularization, termed elastic net penalty, has the property of achieving simultaneous variable selection and margin-maximization within a single framework. However, because of the nondifferentiability of both the loss function and the regularization term, there is no efficient method available to solve DrSVM for large-scale problems. Here we develop an efficient algorithm based on the alternating direction method of multipliers (ADMM) to solve the optimization problem in DrSVM. The utility of the method is illustrated using both simulated and real-world data.

1 INTRODUCTION

Datasets with tens of thousands variables have become increasingly common in many real-world applications. For example, in the biomedical domain a microarray dataset typically contains about 20,000 genes, while a genotype dataset commonly includes half of a million SNPs. Regularization terms that encourage sparsity in coefficients are increasingly being used for simultaneous variable selection and prediction (Tibshirani, 1996; Zou and Hastie, 2005).

A widely used strategy for imposing sparsity on regression or classification coefficients is to use the ℓ_1-norm regularization. Perhaps the most well-known example is the least absolute shrinkage and selection operator (lasso) method for linear regression. The method minimizes the usual sum of squared errors while penalizing the ℓ_1 norm of the regression coefficients (Tibshirani, 1996). Due to the nondifferentiability of the ℓ_1 norm, lasso is able to perform continuous shrinkage and automatic variable selection simultaneously. Although the lasso method has shown success in many situations and has been generalized for different settings (Zhu et al., 2003; Lin and Zhang, 2006), it has several limitations. First, when the dimension of the data (p) is larger than the number of training samples (n), lasso selects at most n variable before it saturates (Efron et al., 2004). Second, if there is a group of variables among which the pairwise correlations are very high, the lasso tends to select only one variable from the group and does not care which one is selected.

The elastic net penalty proposed by Zou et al. in Zou and Hastie (2005) is a convex combination of the lasso and ridge penalty, which has the characteristics of both the lasso and ridge regression in the regression setting. More specifically, the elastic net penalty simultaneously does automatic variable selection and continuous shrinkage, and it can select groups of correlated variables. It is especially useful for "large p, small n" problems, where the "grouped variables" situation is a particularly important concern and has been addressed many times in the literature (Hastie et al., 2000, 2003).

The idea of using ℓ_1-norm constraints to automatically select variables has also been extended to classification problems. Zhu et al. (2003) proposed an ℓ_1-norm support vector machine, whereas Wang et al. (2006) proposed a SVM with the elastic net penalty term, which they named doubly regularized support vector machine (DrSVM). By using a mixture of the ℓ_1-norm and the ℓ_2-norm penalties, DrSVM is able to perform automatic variable selection as the ℓ_1-norm

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

SVM. Additionally, it also encourages highly correlated variables to be selected (or removed) together, and thus achieves the grouping effect.

Although DrSVM has a number of desirable features, solving DrSVM is, however, non-trivial because of the nondifferentiability in both the loss function and the regularization term. This is especially problematic for large scale problems. To circumvent this difficulty, Wang et al. (2008) proposed a hybrid huberized support vector machine (HHSVM), which uses a huberized hinge loss function to approximate the hinge loss in DrSVM. Because the huberized hinge loss function is differentiable, HHSVM is easier to solve than DrSVM. Wang et al. (2008) proposed a path algorithm to solve the HHSVM problem. However, because the path algorithm requires tracking disappearance of variables along a regularization path, it is not easy to implement and still does not handle large-scale data well.

Our main contribution in this paper is to introduce a new algorithm to directly solve DrSVM without resorting to approximation as in HHSVM. Our method is based on the alternating direction method of multipliers (ADMM) (Gabay and Mercier, 1976; Glowinski and Marroco). We demonstrate that the method is efficient even for large-scale problems with tens of thousands variables.

The rest of the paper is organized as follows. In Section 2, we provide a description of the SVM model with elastic net penalty. In Section 3, we derive an iterative algorithm based on ADMM to solve the optimization problem in DrSVM and prove its convergence property. In Section 4, we benchmark the performance of the algorithm on both simulated and real-world data.

2 SUPPORT VECTOR MACHINES WITH ELASTIC NET PENALTY

2.1 SVM as regularized function estimation

Consider the classification of the training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$ are the predictor variables and $y_i \in \{-1, 1\}$ is the corresponding class label. The support vector machine (SVM) was originally proposed to find the optimal separating hyperplane that separates the two classes of data points with the largest margin (Vapnik, 1998). It can be equivalently reformulated as an ℓ_2-norm penalized optimization problem:

$$\min_{\beta_0,\beta} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\beta_0 + \mathbf{x}_i^T\beta))_+ + \frac{\lambda}{2}\|\beta\|_2^2, \qquad (1)$$

where the loss function $(1-\cdot)_+ := \max(1-\cdot, 0)$ is called the *hinge loss*, and $\lambda \geq 0$ is a regularization parameter, which controls the balance between the 'loss' and the 'penalty'.

By shrinking the magnitude of the coefficients, the ℓ_2 norm penalty in (1) reduces the variance of the estimated coefficients, and thus can achieve better prediction accuracy. However, the ℓ_2 norm penalty cannot produce sparse coefficients and hence cannot automatically perform variable selection. This is a major limitation for applying SVM to do classification in some high-dimensional data, such as gene expression data from microarrays (Guyon et al., 2002; Mukherjee et al., 1999), where variable selection is essential for both achieving better prediction accuracy and providing reasonable interpretations.

To include variable selection, Zhu et al. (2003) proposed an ℓ_1-norm support vector machine,

$$\min_{\beta_0,\beta} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\beta_0 + \mathbf{x}_i^T\beta))_+ + \lambda\|\beta\|_1, \qquad (2)$$

which do variable selection automatically via the ℓ_1 penalty. However, it shares similar disadvantages as the lasso method for 'large p, small n' problems, such as selecting at most n relevant variables, and disregarding group effects. This is not satisfying for some application problems. In microarray analysis, we almost always have $p \gg n$. Furthermore, the genes in the same biological pathway frequently show highly correlated expression; it is desirable to identify all, instead a subset, of them for both providing biological interpretations and building prediction models.

One natural way to overcome the limitations outlined above is to apply the elastic net penalty to the SVM:

$$\min_{\beta_0,\beta} \frac{1}{n}\sum_{i=1}^{n}(1-y_i(\beta_0+\mathbf{x}_i^T\beta))_+ + \lambda_1\|\beta\|_1 + \frac{\lambda_2}{2}\|\beta\|_2^2, \quad (3)$$

where $\lambda_1, \lambda_2 \geq 0$ are regularization parameters. The model was originally proposed by Wang et al. (2006), and was named doubly regularized SVM (DrSVM). However, to emphasize the role of elastic net penalty, we refer to this model (3) as elastic net SVM or simply ENSVM in the rest of the paper. Due to the properties of the elastic net penalty, the optimal solution of (3) will enjoy both the sparse and the grouping effect the same as the elastic net method in regression.

2.2 RELATED WORK

A similar model has been proposed by Wang et al. (2008) who have applied the elastic net penalty to the huberized hinge function and proposed the HHSVM:

$$\min_{\beta_0,\beta} \frac{1}{n}\sum_{i=1}^{n}\phi(y_i(\beta_0 + \mathbf{x}_i^T\beta)) + \lambda_1\|\beta\|_1 + \frac{\lambda_2}{2}\|\beta\|_2^2, \quad (4)$$

where ϕ is the huberized hinge loss function:

$$\phi(t) = \begin{cases} 0, & \text{for } t > 1, \\ (1-t)^2/2\delta, & \text{for } 1-\delta < t \le 1, \\ 1-t-\delta/2, & \text{for } t \le 1-\delta \end{cases} \quad (5)$$

with $\delta > 0$ being a pre-specified constant. The main motivation for Wang et al. (2008) to use huberized hinge loss function (5) is that it is an approximation of the hinge loss and differentiable everywhere, thereby making the optimization problem easier to solve while at the same time preserving the variable selection feature.

The minimizer of (4) is piecewise linear with respect to λ_1 for a fixed λ_2. Based on this observation, Wang et al. (2008) proposed a path algorithm to solve the HHSVM problem. The path algorithm keeps track of four sets as λ_1 decreases, and calls an 'event' happening if any one of the four sets changes. Between any two consecutive 'events', the solutions are linear in λ_1, and after an 'event' occurs, the derivative of the solution with respect to λ_1 is changed. When each 'event' happens, the algorithm solves a linear system. If the dimension of the data p is large, solving many large-scale linear systems will be required to obtain the solution path. Furthermore, those linear equations are quite different from each other, and there are no special structures involved. As a result, the path algorithm is computational very expensive for large p problems.

3 ALGORITHM FOR ELASTIC NET SVM

The alternating direction method of multipliers (ADMM) developed in the 1970s (Gabay and Mercier, 1976; Glowinski and Marroco) has recently become a method of choice for solving many large-scale problems (Candes et al., 2009; Cai et al., 2009; Goldstein and Osher, 2009). It is equivalent or closely related to many other algorithms, such as Douglas-Rachford splitting (Wu and Tai, 2010), split Bregman method (Goldstein and Osher, 2009) and the method of multipliers (Rockafellar, 1973).

In this section, we propose an efficient algorithm based on ADMM to solve ENSVM in (3) by introducing auxiliary variables and reformulating the original problem.

3.1 DERIVING ADMM FOR ELASTIC NET SVM

Because of the two nondifferentiable terms in (3), it is hard to solve the ENSVM problem directly. In order to derive an ADMM algorithm, we introduce some auxiliary variables to handle the nondifferentiability of the hinge loss and ℓ_1 norm term.

Let $X = (x_{ij})_{i=1,j=1}^{n,p}$ and Y be a diagonal matrix with its diagonal elements to be the vector $y = (y_1, \ldots, y_n)^T$. The unconstrained problem in (3) can be reformulated into an equivalent constrained problem

$$\begin{aligned} \arg\min_{\beta,\beta_0} \frac{1}{n}\sum_{i=1}^{n}(a_i)_+ + \lambda_1\|\mathbf{c}\|_1 + \frac{\lambda_2}{2}\|\beta\|_2^2 \\ s.t. \quad \mathbf{a} = \mathbf{1} - Y\left(X\beta + \beta_0\mathbf{1}\right), \\ \mathbf{c} = \beta, \end{aligned} \quad (6)$$

where $\mathbf{a} = (a_i, \ldots, a_n)^T$ and $\mathbf{1}$ is an n-column vector of 1s.

Note that the Lagrangian function of (6) is

$$\begin{aligned} & L\left(\beta, \beta_0, \mathbf{a}, \mathbf{c}, \mathbf{u}, \mathbf{v}\right) \\ = \; & \frac{1}{n}\sum_{i=1}^{n}(a_i)_+ + \lambda_1\|\mathbf{c}\|_1 + \frac{\lambda_2}{2}\|\beta\|_2^2 \\ & + \langle \mathbf{u}, \mathbf{1} - Y(X\beta + \beta_0\mathbf{1}) - \mathbf{a}\rangle + \langle \mathbf{v}, \beta - \mathbf{c}\rangle, \end{aligned} \quad (7)$$

where $\mathbf{u} \in \mathbb{R}^n$ is a dual variable corresponding to the linear constraint $\mathbf{a} = \mathbf{1} - Y\left(X\beta + \beta_0\mathbf{1}\right)$, $\mathbf{v} \in \mathbb{R}^p$ is a dual variable corresponding to the linear constraint $\mathbf{c} = \beta$, $\langle \cdot, \cdot \rangle$ denotes the standard inner product in Euclidean space. The augmented Lagrangian function of (6) is similar to (7) except for adding two terms $\frac{u_1}{2}\|\mathbf{1} - Y\left(X\beta + \beta_0\mathbf{1}\right) - \mathbf{a}\|_2^2$ and $\frac{u_2}{2}\|\beta - \mathbf{c}\|_2^2$ to penalize the violation of linear constraints $\mathbf{a} = \mathbf{1} - Y\left(X\beta + \beta_0\mathbf{1}\right)$ and $\mathbf{c} = \beta$, thereby making the function strictly convex. That is,

$$\begin{aligned} & \mathcal{L}\left(\beta, \beta_0, \mathbf{a}, \mathbf{c}, \mathbf{u}, \mathbf{v}\right) \\ = \; & L\left(\beta, \beta_0, \mathbf{a}, \mathbf{c}, \mathbf{u}, \mathbf{v}\right) + \frac{\mu_1}{2}\|\mathbf{1} - Y(X\beta + \beta_0\mathbf{1}) - \mathbf{a}\|_2^2 \\ & + \frac{\mu_2}{2}\|\beta - \mathbf{c}\|_2^2, \end{aligned} \quad (8)$$

where $\mu_1 > 0$ and $\mu_2 > 0$ are two parameters. It is easy to see that solving (6) is equivalent to finding a saddle point $(\beta^*, \beta_0^*, \mathbf{a}^*, \mathbf{c}^*, \mathbf{u}^*, \mathbf{v}^*)$ of $\mathcal{L}\left(\beta, \beta_0, \mathbf{a}, \mathbf{c}, \mathbf{u}, \mathbf{v}\right)$ such that

$$\begin{aligned} \mathcal{L}\left(\beta^*, \beta_0^*, \mathbf{a}^*, \mathbf{c}^*, \mathbf{u}, \mathbf{v}\right) &\le \mathcal{L}\left(\beta^*, \beta_0^*, \mathbf{a}^*, \mathbf{c}^*, \mathbf{u}^*, \mathbf{v}^*\right) \\ &\le \mathcal{L}\left(\beta, \beta_0, \mathbf{a}, \mathbf{c}, \mathbf{u}^*, \mathbf{v}^*\right), \end{aligned}$$

for all $\beta, \beta_0, \mathbf{a}, \mathbf{c}, \mathbf{u}$ and $\mathbf{v}$.

We solve the saddle point problem through gradient ascent on the dual problem

$$\max_{\mathbf{u},\mathbf{v}} E(\mathbf{u}, \mathbf{v}), \quad (9)$$

where $E(\mathbf{u}, \mathbf{v}) = \min_{\beta,\beta_0,\mathbf{a},\mathbf{c}} \mathcal{L}\left(\beta, \beta_0, \mathbf{a}, \mathbf{c}, \mathbf{u}, \mathbf{v}\right)$. Note that the gradient $\nabla E(\mathbf{u}, \mathbf{v})$ can be calculated by the following (Bertsekas, 1982)

$$\nabla E(\mathbf{u}, \mathbf{v}) = \begin{pmatrix} \mathbf{1} - Y(X\beta(\mathbf{u}, \mathbf{v}) + \beta_0(\mathbf{u}, \mathbf{v})\mathbf{1}) - \mathbf{a}(\mathbf{u}, \mathbf{v}) \\ \beta(\mathbf{u}, \mathbf{v}) - \mathbf{c}(\mathbf{u}, \mathbf{v}) \end{pmatrix}, \quad (10)$$

with

$$\begin{aligned}&(\beta(\mathbf{u},\mathbf{v}),\beta_0(\mathbf{u},\mathbf{v}),\mathbf{a}(\mathbf{u},\mathbf{v}),\mathbf{c}(\mathbf{u},\mathbf{v}))\\ =\ &\arg\min_{\beta,\beta_0,\mathbf{a},\mathbf{c}}\mathcal{L}\left(\beta,\beta_0,\mathbf{a},\mathbf{c},\mathbf{u},\mathbf{v}\right).\end{aligned}\tag{11}$$

Using gradient ascent on the dual problem (9), Eq. (10) and Eq. (11), we get the method of multipliers (Rockafellar, 1973) to solve (6)

$$\begin{cases}&(\beta^{k+1},\beta_0^{k+1},\mathbf{a}^{k+1},\mathbf{c}^{k+1})\\ &=\arg\min_{\beta,\beta_0,\mathbf{a},\mathbf{c}}\mathcal{L}\left(\beta,\beta_0,\mathbf{a},\mathbf{c},\mathbf{u}^k,\mathbf{v}^k\right)\\ &\mathbf{u}^{k+1}=\mathbf{u}^k+\mu_1(\mathbf{1}-Y(X\beta^{k+1}+\beta_0^{k+1}\mathbf{1})-\mathbf{a}^{k+1}),\\ &\mathbf{v}^{k+1}=\mathbf{v}^k+\mu_2(\beta^{k+1}-\mathbf{c}^{k+1}).\end{cases}\tag{12}$$

The efficiency of the iterative algorithm (12) lies on whether the first equation of (12) can be solved quickly. The augmented Lagrangian function $\mathcal{L}$ still contains nondifferentiable terms. But different from the original objective function (3), the hinge loss induced nondifferentiability has now been transferred from terms involving $1-y_i(\mathbf{x}_i^T\beta+\beta_0)$ to terms involving a_i; and ℓ_1 induced nondifferentiability has now been transferred from terms involving β to terms involving $\mathbf{c}$. Moreover, the nondifferentiable terms involving $\mathbf{a}$ and $\mathbf{c}$ are now completely decoupled, and thus we can solve the first equation of (12) by alternating minimization of $(\beta,\beta_0),\mathbf{a}$ and $\mathbf{c}$,

$$\begin{cases}(\beta^{k+1},\beta_0^{k+1})=\arg\min_{\beta,\beta_0}\mathcal{L}\left(\beta,\beta_0,\mathbf{a}^k,\mathbf{c}^k,\mathbf{u}^k,\mathbf{v}^k\right),\\ \mathbf{a}^{k+1}=\arg\min_{\mathbf{a}}\mathcal{L}\left(\beta^{k+1},\beta_0^{k+1},\mathbf{a},\mathbf{c}^k,\mathbf{u}^k,\mathbf{v}^k\right),\\ \mathbf{c}^{k+1}=\arg\min_{\mathbf{c}}\mathcal{L}\left(\beta^{k+1},\beta_0^{k+1},\mathbf{a}^{k+1},\mathbf{c},\mathbf{u}^k,\mathbf{v}^k\right).\end{cases}\tag{13}$$

For the method of multipliers, the alternate minimization (13) needs to run multiple times until convergence. However, we do not have to completely solve the first equation of (12) since it is only one step of the overall iterative algorithm. We use only one alternation, it is called alternating direction method of multipliers (Gabay and Mercier, 1976). That is, we use the following iterations to solve (6)

$$\begin{cases}(\beta^{k+1},\beta_0^{k+1})=\arg\min_{\beta,\beta_0}\mathcal{L}\left(\beta,\beta_0,\mathbf{a}^k,\mathbf{c}^k,\mathbf{u}^k,\mathbf{v}^k\right),\\ \mathbf{a}^{k+1}=\arg\min_{\mathbf{a}}\mathcal{L}\left(\beta^{k+1},\beta_0^{k+1},\mathbf{a},\mathbf{c}^k,\mathbf{u}^k,\mathbf{v}^k\right),\\ \mathbf{c}^{k+1}=\arg\min_{\mathbf{c}}\mathcal{L}\left(\beta^{k+1},\beta_0^{k+1},\mathbf{a}^{k+1},\mathbf{c},\mathbf{u}^k,\mathbf{v}^k\right),\\ \mathbf{u}^{k+1}=\mathbf{u}^k+\mu_1(\mathbf{1}-Y(X\beta^{k+1}+\beta_0^{k+1}\mathbf{1})-\mathbf{a}^{k+1}),\\ \mathbf{v}^{k+1}=\mathbf{v}^k+\mu_2(\beta^{k+1}-\mathbf{c}^{k+1}).\end{cases}\tag{14}$$

For the first equation in (14), it is equivalent to

$$\begin{aligned}(\beta,\beta_0)\ =\ &\arg\min_{\beta,\beta_0}\frac{\lambda_2}{2}\|\beta\|_2^2+\left\langle\mathbf{v}^k,\beta-\mathbf{c}^k\right\rangle\\ &+\left\langle\mathbf{u}^k,\mathbf{1}-Y(X\beta+\beta_0\mathbf{1})-\mathbf{a}^k\right\rangle\\ &+\frac{\mu_1}{2}\|\mathbf{1}-Y(X\beta+\beta_0\mathbf{1})-\mathbf{a}^k\|_2^2\\ &+\frac{\mu_2}{2}\|\beta-\mathbf{c}^k\|_2^2.\end{aligned}$$

The objective function in the above minimization problem is quadratic and differentiable, and thus the optimal solution can be found by solving a set of linear equations:

$$\begin{pmatrix}(\lambda_2+\mu_2)I+\mu_1X^TX & \mu_1X^T\mathbf{1}\\ \mu_1\mathbf{1}^TX & \mu_1 n\end{pmatrix}\begin{pmatrix}\beta^{k+1}\\ \beta_0^{k+1}\end{pmatrix}$$

$$=\begin{pmatrix}X^TY\mathbf{u}^k-\mu_1X^TY(\mathbf{a}^k-\mathbf{1})-\mathbf{v}^k+\mu_2\mathbf{c}^k\\ \mathbf{1}^TY\mathbf{u}^k-\mu_1\mathbf{1}^TY(\mathbf{a}^k-\mathbf{1})\end{pmatrix}.\tag{15}$$

Note that the coefficient matrix in (15) is a $(p+1)\times(p+1)$ matrix, independent of the optimization variables. For small p, we can store its inverse in the memory, so the linear equations can be solved with minimal cost. For large p, we use the conjugate gradient algorithm (CG) to solve it at each iteration efficiently.

The linear system (15) is very special for large p, small n problems in that X^TX will be a positive low rank matrix with rank at most n. Thus the coefficient matrix in (15) is a linear combination of identity matrix and a positive low rank matrix with rank at most $n+1$. If we use CG to solve the linear system (15), it converges in less than $n+1$ steps (Saad, 2003). In our numerical implementation, we found that CG converges in a few steps much smaller than $n+1$.

For the second equation in (14), it is equivalent to

$$\begin{aligned}\mathbf{a}^{k+1}\ =\ &\arg\min_{\mathbf{a}}\frac{1}{n}\sum_{i=1}^n(a_i)_+\\ &+\frac{\mu_1}{2}\|\mathbf{1}-Y(X\beta^{k+1}+\beta_0^{k+1}\mathbf{1})-\mathbf{a}\|_2^2\\ &+\langle\mathbf{u}^k,\mathbf{1}-Y(X\beta^{k+1}+\beta_0^{k+1}\mathbf{1})-\mathbf{a}\rangle.\end{aligned}\tag{16}$$

In order to solve (16), we need the following Proposition (Ye and Xie, 2010).

Proposition 1 *Let* $s_\lambda(\omega)=\arg\min_{x\in\mathbb{R}}\lambda x_+ +\frac{1}{2}\|x-\omega\|_2^2$. *Then*

$$s_\lambda(\omega)=\begin{cases}\omega-\lambda, & \omega>\lambda\\ 0, & 0\le\omega\le\lambda,\\ \omega, & \omega<0.\end{cases}$$

Note that each a_i is independent of each other in (16) and

$$\begin{aligned} & \frac{\|\mathbf{u}\|_2^2}{2\mu_1} + \frac{\mu_1}{2}\|\mathbf{1} - Y(X\beta^{k+1} + \beta_0^{k+1}\mathbf{1}) - \mathbf{a}\|_2^2 \\ & +\langle \mathbf{u}^k, \mathbf{1} - Y(X\beta^{k+1} + \beta_0^{k+1}\mathbf{1}) - \mathbf{a}\rangle \\ = \; & \frac{\mu_1}{2}\|\mathbf{a} - (\mathbf{1} + \frac{\mathbf{u}}{\mu_1} - Y(X\beta^{k+1} + \beta_0^{k+1}\mathbf{1}))\|_2^2. \end{aligned}$$

Together with Proposition 1, we can then update $\mathbf{a}^{k+1}$ in (16) according to

Corollary 1 *The update of* $\mathbf{a}^{k+1}$ *in* (16) *is equivalent to*

$$\mathbf{a}^{k+1} = \mathcal{S}_{\frac{1}{n\mu_1}}(\mathbf{1} + \frac{\mathbf{u}^k}{\mu_1} - Y(X\beta^{k+1} + \beta_0^{k+1}\mathbf{1})), \quad (17)$$

where

$$\mathcal{S}_\lambda(\omega) = (s_\lambda(\omega_1), s_\lambda(\omega_2), \ldots, s_\lambda(\omega_n))^T, \forall \omega \in \mathbb{R}^n.$$

For the third equation in (14), it is equivalent to

$$\mathbf{c}^{k+1} = \arg\min_{\mathbf{c}} \lambda_1\|\mathbf{c}\|_1 + \langle \mathbf{v}^k, \beta^{k+1} - \mathbf{c}\rangle + \frac{\mu_2}{2}\|\beta^{k+1} - \mathbf{c}\|_2^2. \quad (18)$$

Minimization of $\mathbf{c}$ in (18) can be done efficiently using soft thresholding, because the objective function is quadratic and nondifferentiable terms are completely separable. Let $\mathcal{T}_\lambda$ be a soft thresholding operator defined on vector space and satisfying

$$\mathcal{T}_\lambda(\omega) = (t_\lambda(\omega_1), \ldots, t_\lambda(\omega_p)), \forall \omega \in \mathbb{R}^p, \quad (19)$$

where

$$t_\lambda(\omega_i) = \mathrm{sgn}(\omega_i)\max\{0, |\omega_i| - \lambda\}.$$

Using the soft thresholding operator (19), the optimal solution of $\mathbf{c}$ in (18) can be written as

$$\mathbf{c}^{k+1} = \mathcal{T}_{\frac{\lambda_1}{\mu_2}}\left(\frac{\mathbf{v}^k}{\mu_2} + \beta^{k+1}\right). \quad (20)$$

Finally, by combining (14), (15), (17) and (20) together, we obtain the algorithm ADMM for ENSVM (3) (Algorithm 1). It is a practical algorithm for large p, small n problems and very easy to code.

3.2 Convergence analysis

The convergence property of Algorithm 1 can be derived from the standard convergence theory of the alternating direction method of multipliers (Gabay and Mercier, 1976; Eckstein and Bertsekas, 1992).

Algorithm 1 ADMM for ENSVM (3)

Initialize $\beta^0, \beta_0^0, \mathbf{a}^0, \mathbf{c}^0, \mathbf{u}^0$, and $\mathbf{v}^0$.
repeat
1) Update $\beta^{k+1}, \beta_0^{k+1}$ by solving the following linear equation system:

$$\begin{pmatrix} (\lambda_2+\mu_2)I + \mu_1 X^T X & \mu_1 X^T\mathbf{1} \\ \mu_1\mathbf{1}^T X & \mu_1 n \end{pmatrix}\begin{pmatrix}\beta^{k+1}\\ \beta_0^{k+1}\end{pmatrix}$$

$$= \begin{pmatrix} X^T Y\mathbf{u}^k - \mu_1 X^T Y(\mathbf{a}^k - \mathbf{1}) - \mathbf{v}^k + \mu_2\mathbf{c}^k \\ \mathbf{1}^T Y\mathbf{u}^k - \mu_1\mathbf{1}^T Y(\mathbf{a}^k - \mathbf{1}) \end{pmatrix}$$

2) $\mathbf{a}^{k+1} = \mathcal{S}_{\frac{1}{n\mu_1}}\left(\mathbf{1} + \frac{\mathbf{u}^k}{\mu_1} - Y(X\beta^{k+1} + \beta_0^{k+1}\mathbf{1})\right)$
3) $\mathbf{c}^{k+1} = \mathcal{T}_{\frac{\lambda_1}{\mu_2}}\left(\frac{\mathbf{v}^k}{\mu_2} + \beta^{k+1}\right)$
4) $\mathbf{u}^{k+1} = \mathbf{u}^k + \mu_1(\mathbf{1} - Y(X\beta^{k+1} + \beta_0^{k+1}\mathbf{1}) - \mathbf{a}^{k+1})$
5) $\mathbf{v}^{k+1} = \mathbf{v}^k + \mu_2(\beta^{k+1} - \mathbf{c}^{k+1})$
until
Convergence

Theorem 1 *Suppose there exists at least one solution* (β^*, β_0^*) *of* (3). *Assume* $\lambda_1 > 0, \lambda_2 > 0$. *Then the following property for Algorithm 1 holds:*

$$\lim_{k\to\infty} \frac{1}{n}\sum_{i=1}^n (1 - y_i(\mathbf{x}_i^T\beta^k + \beta_0^k))_+ + \lambda_1\|\beta^k\|_1 + \frac{\lambda_2}{2}\|\beta^k\|_2^2$$

$$= \frac{1}{n}\sum_{i=1}^n (1 - y_i(\mathbf{x}_i^T\beta^* + \beta_0^*))_+ + \lambda_1\|\beta^*\|_1 + \frac{\lambda_2}{2}\|\beta^*\|_2^2.$$

Furthermore,

$$\lim_{k\to\infty} \|(\beta^k, \beta_0^k) - (\beta^*, \beta_0^*)\| = 0,$$

whenever (3) *has a unique solution.*

3.3 Computational cost

The efficiency of Algorithm 1 lies mainly on whether we can quickly solve the linear equations (15). As we have described in Section 3.1, the coefficient of the linear equations (15) has a special structure and thus can be efficiently solved by the conjugate gradient method for 'large p, small n' problems. More specifically, the computational cost for solving (15) is $O(n^2p)$. The number of iterations of Algorithm (1) is hard to predict and it depends on the choice of μ_1 and μ_2. According to our experience, we only need to iterate a few hundred iterations to get a reasonable result by choosing μ_1 and μ_2 correctly.

Similar to our algorithm for (3), the major computational cost in each iteration for HHSVM also comes from solving a linear system. However, the linear system in HHSVM has no special structures. It takes at

least $O(|\mathcal{A}|^2)$ with $|\mathcal{A}|$ being the number of unknown variables. Moreover, $|\mathcal{A}|$ can increase at each iteration. Furthermore, for large scale problems, it usually takes a few thousand steps for the algorithm converges. That's why our algorithm for (3) is much faster than the path algorithm for HHSVM for large scale problems.

4 NUMERICAL RESULTS

In this section, we use time trials on both simulated data as well as real microarray data to illustrate the efficiency of ADMM algorithm for solving elastic net SVM (ENSVM). To evaluate the performance of ADMM for ENSVM, we also compare it with the stochastic sub-gradient method and the path algorithm for HHSVM. Our algorithm and the stochastic sub-gradient method were implemented in Matlab, while HHSVM was implemented in R using the R code provided by the authors in (Wang et al., 2008). All algorithms were compiled on a windows platform and time trials were generated on an Intel Core 2 Duo desktop PC (E7500, 2.93GHz).

The stopping criteria of Algorithm 1 for ENSVM is specified as follows. Let $\Phi(\beta^k, \beta_0^k) = \frac{1}{n}\sum_{i=1}^n (1 - y_i(\mathbf{x}_i^T\beta^k + \beta_0^k))_+ + \lambda_1\|\beta^k\|_1 + \frac{\lambda_2}{2}\|\beta^k\|_2^2$. According to Theorem 1, $\lim_{k\to\infty}\Phi(\beta^k, \beta_0^k) = \Phi(\beta^*, \beta_0^*)$. It is reasonable to terminate the algorithm when the relative change of the energy functional $\Phi(\beta, \beta_0)$ falls below certain threshold δ. Furthermore, Algorithm 1 is solving (6), linear constraints are satisfied when it converges. Therefore, we would expect that $\frac{1}{\sqrt{n}}\|\mathbf{1} - Y(X\beta^k + \beta_0^k\mathbf{1}) - \mathbf{a}^k\|_2 \le \delta$ and $\frac{1}{\sqrt{p}}\|\beta^k - \mathbf{c}^k\|_2 \le \delta$ when we terminate the algorithm. We used $\delta = 10^{-5}$ in our simulation, i.e., we stop Algorithm 1 whenever

$$RelE := \frac{|\Phi(\beta^k, \beta_0^k) - \Phi(\beta^*, \beta_0^*)|}{\max\{1, \Phi(\beta^k, \beta_0^k)\}} \le 10^{-5},$$

$$\frac{1}{\sqrt{n}}\|\mathbf{1} - Y(X\beta^k + \beta_0^k\mathbf{1}) - \mathbf{a}^k\|_2 \le 10^{-5}$$

and

$$\frac{1}{\sqrt{p}}\|\beta^k - \mathbf{c}^k\|_2 \le 10^{-5}.$$

Note that the convergence of Algorithm 1 is guaranteed no matter what values of μ_1 and μ_2 are used as shown in Theorem 1. However, the speed of the algorithm can be influenced by the choices of μ_1 and μ_2 as it would affect the number of iterations involved. In our implementation, we found empirically that choosing $\mu_1 = \frac{100}{n}$ and $\mu_2 \in [25, 100]$ works well for all the problems we tested, though the parameter selecting procedure can certainly be further improved.

4.1 SIMULATION

We consider a binary classification problem in which the sample data are lying in a p dimensional space with only the first 10 dimensions being relevant for classification and the remaining variables being noises. More specifically, we generate n samples with half from $+1$ and the other half from -1 class. For the samples from $+1$ class, they are i.i.d drawn from a normal distribution with mean

$$\mu_+ = (\underbrace{1, \ldots, 1}_{10}, \underbrace{0, \ldots, 0}_{p-10})^T$$

and covariance

$$\Sigma = \begin{pmatrix} \Sigma^*_{10\times 10} & \mathbf{0}_{10\times(p-10)} \\ \mathbf{0}_{(p-10)\times 10} & I_{(p-10)\times(p-10)} \end{pmatrix},$$

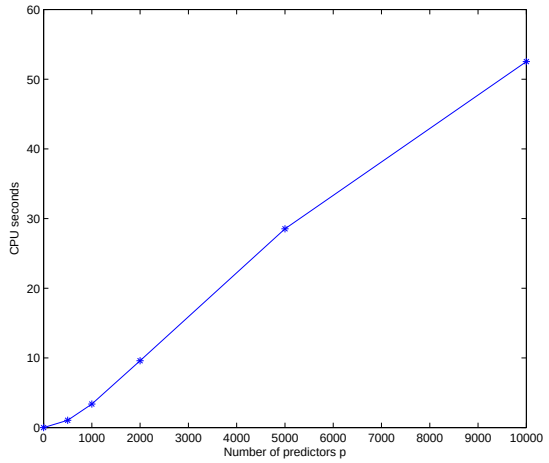
where the diagonal elements of Σ^* are 1 and the off-diagonal elements are all equal to ρ. The -1 class has a similar distribution except that

$$\mu_- = (\underbrace{-1, \ldots, -1}_{10}, \underbrace{0, \ldots, 0}_{(p-10)})^T.$$

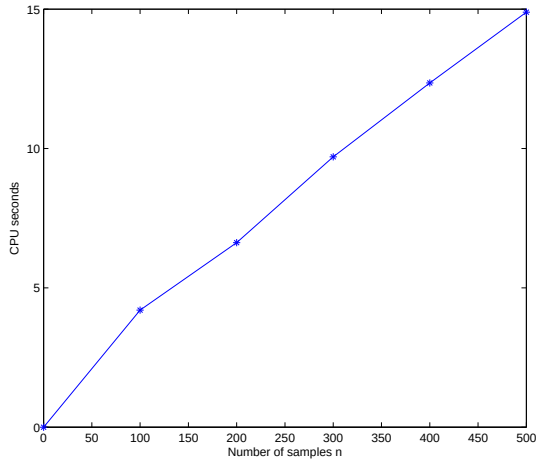
Table 1:
Run times (CPU seconds) of various sizes p and n, different correlation ρ between the features. Methods are ADMM algorithm for elastic-net SVM (ENSVM), path algorithm for HHSVM and stochastic sub-gradient method (SSG). The results for ENSVM and stochastic sub-gradient method are averaged over 25 runs (using 25 different values of λ_1, λ_2) and the ones for HHSVM are averaged over 5 runs (using 5 different values of λ_2).

n, p	Method	$\rho = 0$	$\rho = 0.8$
n=50	ENSVM	0.41	0.31
	HHSVM	3.30	3.19
p=300	SSG	2.35	4.06
n=100	ENSVM	1.19	0.71
	HHSVM	21.65	21.01
p=500	SSG	4.34	4.06
n=200	ENSVM	3.60	3.75
	HHSVM	405.9	390.1
p=1000	SSG	35.40	26.86
n=300	ENSVM	14.73	16.74
	HHSVM	2.07 hours	2.03 hours
p=2000	SSG	123.22	122.84
n=400	ENSVM	48.62	57.15
	HHSVM	$>$ 6 hours	$>$ 6 hours
p=5000	SSG	301.10	290.15
n=500	ENSVM	144.69	170.52
	HHSVM	-	-
p=10000	SSG	785.57	909.92

So the Bayes optimal classification rule depends on $x_1, \ldots, x_{10}$, which are highly correlated if ρ is large. The Bayes error is independent of the dimension p. This simulated data were also used in Wang et al. (2008).



(a)



(b)

Figure 1: CPU times of the ADMM method for ENSVM for the same problem as in Table 1, for different values of n and p. In each case the times are averaged over 10 runs. (a) n is fixed and equals to 300; (b) p is fixed and equals to 2000.

Table 1 shows the average CPU times (seconds) used by the ADMM algorithm, the path algorithm for HHSVM, and the stochastic sub-gradient method. Our algorithm consistently outperforms both the stochastic sub-gradient method and the path algorithm in all cases we have tested. For the data with $n = 300, p = 2000$, the ADMM algorithm is able to achieve 120-fold speedup than the path algorithm. The ADMM algorithm is also significantly faster than the stochastic sub-gradient method, achieving about 5-10 fold speedup in all cases. We should also note that unlike the ADMM method, the objective function in the stochastic sub-gradient method can go up and down, which makes it difficult to design the stopping criteria for the sub-gradient method.

To evaluate how the performance of our algorithm scales with the problem size, we plotted the CPU time that Algorithm 1 took to solve (3) for the data described above as a function of p and n. Figure 1 shows such a curve, where the CPU times are averaged over 10 runs with different data. We note that the CPU times are roughly linear in both n and p.

We also compared the performance of prediction accuracy and variable selection from three different models: ℓ_1-norm SVM (L_1 SVM), HHSVM and ENSVM. The optimal (λ_1, λ_2) pair is chosen from a large grid using 10-fold cross validation. As shown in Table 2 and Table 3, HHSVM and ENSVM are similar in prediction and variable selection accuracy, but both are significantly better than ℓ_1-norm SVM.

Table 2:
Comparison of test errors. The number of training samples is 50. The total number of input variables is 300, with only 10 being relevant for classification. The results are averages of test errors over 100 repetitions on a 10000 test set, and the numbers in parentheses are the corresponding standard errors. $\rho = 0$ corresponds to the case where the input variables are independent, while $\rho = 0.8$ corresponds to a pairwise correlation of 0.8 between relevant variables.

	$\rho = 0$	$\rho = 0.8$
SVM	0.214(0.004)	0.160(0.003)
L_1 SVM	0.143(0.007)	0.160(0.002)
HHSVM	0.133(0.005)	0.143(0.001)
ENSVM	0.111(0.002)	0.144(0.001)

Table 3:
Comparison of variable selection. The setup are the same as those described in Table 2. q_{signal} is the number of selected relevant variables, and q_{noise} is the number of selected noise variables.

	$\rho = 0$		$\rho = 0.8$	
	q_{signal}	q_{noise}	q_{signal}	q_{noise}
L_1 SVM	7.2(0.3)	6.5(1.4)	2.5(0.2)	2.9(1.2)
HHSVM	7.6(0.3)	7.1(1.3)	7.9(0.4)	3.3(2.5)
ENSVM	8.6(0.1)	6.4(0.4)	6.6(0.2)	2.0(0.2)

4.2 GENE EXPRESSION DATA

A microarray gene expression dataset typically contains the expression values of tens of thousands of mRNAs collected from a relatively small number of samples. The genes sharing the same biological pathways are often highly correlated in gene expression (Segal et al.). Because of these two features, it is more desirable to apply the elastic net SVM to do variable selection and classification on the microarray data than the standard SVM or the ℓ_1-SVM (Zou and Hastie, 2005; Wang et al., 2008).

The data we use is taken from the paper published by Alon et al. (1999). It contains microarray gene expression collected from 62 samples (40 colon tumor tissues and 22 from normal tissues). Each sample consists the expression values of $p = 2000$ genes. We applied the elastic net SVM (ENSVM) to select variables (i.e. genes) that can be used to predict sample labels and compared its performance to the path algorithm developed for HHSVM. The results are summarized in Table 4, which shows the computational times spent by different solvers in a ten-fold cross-validation procedure for different parameters λ_1 and λ_2. The ADMM algorithm for ENSVM is consistently many times faster than the path algorithm for HHSVM, with an approximately ten-fold speedup in almost all cases.

Table 4:
Run times (CPU seconds) for different values of the regularization parameters λ_1 and λ_2. The methods are ADMM algorithm for ENSVM and path algorithm for HHSVM.

λ_1	λ_2	10-CV error	ENSVM	HHSVM
0.1	0.2	8/62	8.56	108.2
0.1	0.5	8/62	6.30	107.9
0.05	2	8/62	8.76	109.3
0.05	5	7/62	7.12	109.5

We also tested the prediction and the variable selection functionality of ENSVM with Algorithm 1. Following the method in Wang et al. (2008), we randomly split the samples into a training set (27 cancer samples and 15 normal tissues) and a testing set (13 cancer samples and 7 normal tissues). In training phase, we adopt 10-fold cross validation to tune the parameter λ_1, λ_2. This experiment is repeated 100 times. Table 5 shows the statistics on the testing error and the number of selected genes, in comparison to the statistics of SVM and HHSVM. We note that in terms of testing error, ENSVM is slightly better than HHSVM, which in turn is better than the standard SVM. In terms of variable selection, ENSVM tends to select a smaller number of genes than HHSVM.

Table 5:
Comparison of testing error and variable selection on the gene expression data. Shown are the averages from 100 repetitions and included in the parenthesis are the standard deviations.

	Test error	Number of genes selected
SVM	17.9% (0.69%)	All
HHSVM	15.45%(0.59%)	138.37(8.67)
ENSVM	14.95% (0.53%)	87.7 (7.9)

5 CONCLUSION

In this paper, we have derived an efficient algorithm based on the alternating direction method of multipliers to solve the optimization problem in the elastic net SVM (ENSVM). We show that the algorithm is substantially faster than both the sub-gradient method and the path algorithm used in HHSVM, an approximation of the ENSVM problem (Wang et al., 2006). We also illustrate the advantage of ENSVM in both variable selection and prediction accuracy using simulated and real-world data.

6 ACKNOWLEDGEMENT

The research is supported by a grant from University of California, Irvine.

References

U. Alon, N. Barkai, DA Notterman, K. Gish, S. Ybarra, D. Mack, and AJ Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl Acad. Sci. USA*.

D.P. Bertsekas. Constrained optimization and lagrange multiplier methods. 1982.

J.-F. Cai, S. Osher, and Z. Shen. Split bregman methods and frame based image restoration. *Multiscale Model. Simul.*, 8(2):337–369, 2009.

E.J. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Arxiv preprint arXiv:0912.3599*, 2009.

J. Eckstein and D.P. Bertsekas. On the douglas rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318, 1992. ISSN 0025-5610.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499,

2004. With discussion, and a rejoinder by the authors.

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40, 1976.

R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par penalisation-dualité, d'une classe de problèmes de dirichlet non linéaires. *Rev. Franc. Automat. Inform. Rech. Operat.*

T. Goldstein and S. Osher. The split Bregman method for $L1$-regularized problems. *SIAM J. Imaging Sci.*, 2(2):323–343, 2009. ISSN 1936-4954.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422, 2002.

T. Hastie, R. Tibshirani, M.B. Eisen, A. Alizadeh, R. Levy, L. Staudt, W.C. Chan, D. Botstein, and P. Brown. Gene shaving as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1(2):1–0003, 2000.

T. Hastie, R. Tibshirani, D. Botstein, and P. Brown. Supervised harvesting of expression trees. *Genome Biology*, 2(1):0003.1–0003.12, 2003.

Y. Lin and H. H. Zhang. Component selection and smoothing in multivariate nonparametric regression. *Ann. Statist.*, 34(5):2272–2297, 2006. ISSN 0090-5364.

S. Mukherjee, P. Tamayo, D. Slonim, A. Verri, T. Golub, J. Mesirov, and T. Poggio. Support vector machine classification of microarray data. *CBCL Paper*, 182, 1999.

R. T. Rockafellar. A dual approach to solving nonlinear programming problems by unconstrained optimization. *Math. Programming*, 5:354–373, 1973. ISSN 0025-5610.

Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2003.

M.R. Segal, K.D. Dahlquist, and B.R. Conklin. Regression approaches for microarray data analysis. *J. Comput. Biol.*

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288, 1996.

V. N. Vapnik. *Statistical learning theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons Inc., New York, 1998. A Wiley-Interscience Publication.

L. Wang, J. Zhu, and H. Zou. The doubly regularized support vector machine. *Statistica Sinica*, 16(2):589, 2006. ISSN 1017-0405.

L. Wang, J. Zhu, and H. Zou. Hybrid huberized support vector machines for microarray classification and gene selection. *Bioinformatics*, 24(3):412, 2008.

C. Wu and X.C. Tai. Augmented Lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models. *SIAM Journal on Imaging Sciences*, 3:300, 2010.

G.B. Ye and X. Xie. Split Bregman method for large scale fused Lasso. *Arxiv preprint arXiv:1006.5086*, 2010.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, 2003.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(2):301–320, 2005.