New Primal SVM Solver with Linear Computational Cost for Big Data Classifications

Feiping Nie FEIPINGNIE@GMAIL.COM
Yizhen Huang HUANG.YIZHEN@GMAIL.COM
Xiaoqian Wang XIAOQIAN.WANG93@MAVS.UTA.EDU
Heng Huang HENG@UTA.EDU
Computer Science and Engineering Department, University of Texas at Arlington, Arlington, TX, 76019

Abstract

Support Vector Machines (SVM) is among the most popular classification techniques in machine learning, hence designing fast primal SVM algorithms for large-scale datasets is a hot topic in recent years. This paper presents a new L2-norm regularized primal SVM solver using Augmented Lagrange Multipliers, with linear computational cost for Lp-norm loss functions. The most computationally intensive steps (that determine the algorithmic complexity) of the proposed algorithm is purely and simply matrix-by-vector multiplication, which can be easily parallelized on a multi-core server for parallel computing. We implement and integrate our algorithm into the interfaces and framework of the well-known LibLinear software toolbox. Experiments show that our algorithm is with stable performance and on average faster than the state-of-the-art solvers such as SVM^{perf}, Pegasos and the LibLinear that integrates the TRON, PCD and DCD algorithms.

1. Introduction

Because most areas of science, simulations and experiments are flooded with big data, there is an urgent need to develop large-scale data classification techniques. As one of the most widely used classification methods, the fast algorithm to solve Support Vector Machines (SVM) is desired. Given n instance-label pairs (x_i, y_i), $1 \le i \le n$, $x_i \in \Re^d, y_i \in \{-1, +1\}$, the L2-norm regularized SVM in the primal form aims to optimize the following unconstrained problem:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} loss\left(w^T x_i + b, y_i\right) \tag{1}$$

where the support vector $w \in \Re^d$ and the intercept (*i.e.* bias term) $b \in \Re$ are the variables; $loss\,(u, v)$ is a loss function measuring the difference between two scalars $u \in \Re$ and $v \in \Re$; $C \in \Re$ is the weight adjusting the importance between the regularization term $w^T w$ and the loss term $\sum_{i=1}^{n} loss\left(w^T x_i + b, y_i\right)$.

If the loss function is selected to be the hinge loss function $loss\,(u, v) = max(1 - uv, 0)$, the problem becomes the L2-norm regularized L1-norm loss primal SVM, a.k.a L1-primal SVM in some literature:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \left(1 - (w^T x_i + b) y_i\right)_+, \tag{2}$$

where the operator $(u)_+ \overset{def}{=} max(u, 0)$ returns the input scalar $u \in \Re$ unchanged if u is non-negative, and zero otherwise. Such notation is for better readability.

If the loss function is selected to be the squared hinge loss function $loss\,(u, v) = max(1 - uv, 0)^2$, the problem becomes the L2-norm regularized L2-norm loss primal SVM, a.k.a L2-primal SVM:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \left(1 - (w^T x_i + b) y_i\right)_+^2. \tag{3}$$

Primal SVM is attractive, partly due to the fact that it assures a continuous decrease in the primal objective function (Keerthi & DeCoste, 2005). Designing fast primal SVM solvers for large-scale datasets is a hot and important topic in recent years: The method in (Mangasarian & Musicant, 2001) was proposed, but it need compute the inverse of a matrix with size of $(d+1)*(d+1)$ and is slower than later proposed solvers. (Mangasarian, 2002) and (Keerthi & DeCoste, 2005) proposed modified Newton methods to train

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

L2-primal SVM. As Eq.(3) is not 2nd order differentiable, to obtain the Newton direction, they have to use the generalized Hessian matrix (*i.e.* generalized 2nd order derivative), which is not efficient enough. To overcome this limitation, a Trust RegiOn Newton method (TRON) (Lin et al., 2008) was proposed to solve L2-primal SVM and logistic regression. The toolbox SVM^{perf} (Joachims, 2006) used a cutting plane technique to solve L1-primal SVM. (Smola et al., 2008) applied bundle methods, and viewed SVM^{perf} as a special case. (Zhang, 2004) proposed a Stochastic Gradient Descent (SGD) method for primal SVM with any type of loss functions; Pegasos (Shalev-Shwartz et al., 2007) extended Zhang's work and developed an algorithm which alternates between stochastic gradient descent steps and projection steps with better performance than SVM^{perf}. Another stochastic gradient implementation similar to Pegasos was published in (Bottou, 2007). More recently the L2-primal SVM is solved by the PCD (Primal Coordinate Descent) algorithm (Chang et al., 2008) with coordinate descent methods.

All algorithms discussed above are iterative, which update w at each iteration. Based on this understanding, it is naturally to find that, there is a tradeoff between the computational cost spent in each iteration and the number of iterations needed (Lin et al., 2008). Pegasos randomly subsamples a few instances at a time, so its cost per iteration is low, but the number of iterations is high. Contrastively, Newton methods such as the TRON method spends considerable amount of time per iteration, but converges at fast rates. The DCD (Dual Coordinate Descent) algorithm (Hsieh et al., 2008) bypasses the operation to maintain gradients at each iteration in the PCD (Chang et al., 2008), and lowers the algorithmic complexity per iteration from $O(n\bar{d})$ to $O(\bar{d})$ in linear SVM cases, where $\bar{d}$ is the average number of nonzero elements per instance. However, such reduction in complexity does not apply for nonlinear SVM with kernels. Moreover, as shown in our experiments (see §4), the convergence of the DCD may be extremely lengthy for some datasets. In large-scale scenarios, usually an approximate solution of the optimization problem is enough to produce a good model (Lin et al., 2008; Chang et al., 2008; Hsieh et al., 2008). Thus, methods with a low-cost iteration are preferred as they can quickly generate a reasonable model. However, if one specifies an unsuitable stopping condition, such methods may fall into the situation of lengthy iterations. To address these issues, we propose a new SVM solver using Augmented Lagrange Multipliers (ALM) with simple matrix-by-vector multiplication per iteration, linear computational cost, and provable convergence.

The rest of the manuscript is organized as follows: Section 2 presents our proposed algorithm; Section 3 analyzes its optimality and convergence; experimental results are given in Section 4; Section 5 contains the concluding remarks.

2. Proposed Algorithm

For ease of notation and better extensibility, we first unify Eq.(2) and Eq.(3), and generalize them to minimize the objective function of the Lp-primal SVM:

$$obj(w,b) = \frac{1}{2}w^T w + C\sum_{i=1}^{n}\left(1-(w^T x_i+b)y_i\right)_+^p , \qquad (4)$$

where $p \in \Re$ is a constant and typically $1\leq p\leq 2$ for being meaningful. A fundamental difficulty in both L2- and L1-primal SVM is that, their loss functions (*i.e.* hinge loss and squared hinge loss) is piecewise. With this observation and the trick $1-(w^T x_i + b)y_i = y_i y_i-(w^T x_i + b)y_i = y_i(y_i-(w^T x_i + b))$, we introduce auxiliary variables $e_i=y_i-(w^T x_i + b)$, $1\leq i\leq n$, and the minimization of Eq.(4) becomes:

$$\min_{w,b,e_i=y_i-(w^T x_i+b)} \frac{1}{2}w^T w + C\sum_{i=1}^{n}(y_i e_i)_+^p . \qquad (5)$$

Based on ALM (Gill & Robinson, 2012), Eq.(5) is turned into an unconstrained optimization problem by studying the Lagrangian function of Eq.(5) below:

$$L(w,b,e,\lambda)=\frac{1}{2}w^T w+C\sum_{i=1}^{n}(y_i e_i)_+^p+\lambda^T(X^T w+\mathbf{1}b-y+e) \qquad (6)$$

where $X_{d\times n}$=$[x_1,x_2,...,x_n]$, $\mathbf{1}_{n\times 1}$=$[1,1,...,1]^T$, $y_{n\times 1}$=$[y_1,y_2,...,y_n]^T$, $e_{n\times 1}$=$[e_1,e_2,...,e_n]^T$. The last term is the pointwise multiplication of the amount of violation of the n constraints $(w^T x_i+b)-y_i+e_i$=0 with the vector $\lambda \in \Re^n$ consisting of n Lagrangian multipliers.

ALM adds $\frac{\mu}{2}\left\|X^T w + \mathbf{1}b - y + e\right\|^2$ (supplemental term) to Eq.(6): as μ "augments" to infinity, this term forces the n constraints to be satisfied. So the augmented Lagrangian function of Eq.(5) is defined as:

$$\begin{aligned} AL(w,b,e,\lambda,\mu) &= \tfrac{1}{2}w^T w + C\textstyle\sum_{i=1}^{n}(y_i e_i)_+^p + \\ \lambda^T(X^T w+\mathbf{1}b-y+e) &+ \tfrac{\mu}{2}||X^T w+\mathbf{1}b-y+e||^2 . \end{aligned} \qquad (7)$$

Arranging the last two terms in Eq.(7) into a quadratic form leads to:

$$\begin{aligned} AL(w,b,e,\lambda,\mu) &= \tfrac{1}{2}w^T w + C\textstyle\sum_{i=1}^{n}(y_i e_i)_+^p \\ &+\tfrac{\mu}{2}||X^T w + \mathbf{1}b - y + e + \tfrac{\lambda}{\mu}||^2 . \end{aligned} \qquad (8)$$

Note that from Eq.(7) to Eq.(8), we add a term $\frac{\lambda^2}{2\mu}$ that remains constant when performing optimization via the variables w, b and e within a single iteration. As $\mu \to \infty$, this term is very close to zero and thus negligible eventually.

At the k-th iteration, similar to the Iterative Thresholding (IT) method (Wright et al., 2009), the amount of violation of the n constraints is used to update the Lagrangian multiplier vector λ:

$$\lambda_{(k)} = \lambda_{(k-1)} + \mu_{(k)}(X^T w + \mathbf{1}b - y + e)\,. \tag{9}$$

The augmented penalty parameter μ is monotonically non-decreasing over the iterative process. How to determine the augmented penalty parameter series $\mu_{(k)}$ for every iteration k will be discussed in §3. We omit the bracketed subscript k when there is no risk to cause any confusion, and actually we only need a single variable (or array) to store these scalars (or vectors) for algorithmic implementation. Note that the symbols of subscripts and bracketed subscripts refer to quite different meanings in this paper.

Remarks: A merit of the ALM is that the optimal step size to update λ is proven to be the chosen penalty parameter $\mu_{(k)}$, making the parameter tuning much easier than that of the IT (Gill & Robinson, 2012).

Now at each iteration, we can split the updating of w, b, e into two independent portions: minimizing e with w, b fixed and minimizing w, b with e fixed.

When w, b are fixed, the term $w^T w$ is constant and Eq.(8) can be decomposed into n independent single-variable minimization problems w.r.t. e_i:

$$\begin{aligned} e_i &= \arg\min_{e_i} F_i(e_i) \\ &= \arg\min_{e_i} C\,(y_i e_i)_+^p + \frac{\mu}{2}||e_i - (y_i - w^T x_i - b - \frac{\lambda_i}{\mu})||^2 \\ &= \arg\min_{e_i} \gamma\,(y_i e_i)_+^p + \frac{1}{2}(e_i - t_i)^2\,, \end{aligned} \tag{10}$$

where $\gamma = \frac{C}{\mu}$, λ_i is the i-th element of λ, $t_i = y_i - w^T x_i - b - \frac{\lambda_i}{\mu}$ is a constant. Solving Eq.(10) is easy as e_i is the minimizer for the single-variable 2-piece piecewise function $F_i(e_i)$, so we just need to find its minima when $e_i \le 0$ and $e_i > 0$ separately and pick the smaller one. When $y_i e_i \le 0$, $(y_i e_i \le 0)_+^p = 0$, so we only need to pick the smaller between $F_i(0)$ and $F_i(t_i)$. When $y_i e_i > 0$, we need to solve the equation:

$$\frac{\partial F_i}{\partial e_i} = p\gamma(y_i e_i)^{p-1} + e_i - t_i = 0\,. \tag{11}$$

For arbitrary given p and γ, solving Eq.(11) is difficult. But fortunately in our scenario, it is always that $p \ge 1, \gamma > 0$. So $\frac{\partial F_i}{\partial e_i}$ is monotonically increasing w.r.t. e_i, and we can use the well-known binary search method to narrow the possible range of e_i by half via each operation, and obtain an ε-accurate solution in $O(\log 1/\varepsilon)$ time. Particularly, we can write the explicit solution straightforwardly when p=1 or 2.

For L1-primal SVM (p=1), $y_i = \pm 1$:

$$e_i = t_i - y_i\gamma \text{ when } y_i t_i > \gamma;\; e_i = 0 \text{ when } 0 \le y_i t_i \le \gamma;\; e_i = t_i \text{ when } y_i t_i < 0\,. \tag{12}$$

For L2-primal SVM (p=2), $y_i = \pm 1$:

$$e_i = t_i/(1 + 2\gamma) \text{ when } y_i t_i > 0;\; e_i = t_i \text{ when } y_i t_i \le 0 \tag{13}$$

When e is fixed, the term $\sum_{i=1}^n (y_i e_i)_+^p$ is constant, and Eq.(8) becomes an L2-norm regularized Least Square Regression (LSR) problem:

$$G(w, b) = \min_{w,b} \mu^{-1} w^T w + \left\|X^T w + \mathbf{1}b - z\right\|^2\,, \tag{14}$$

where $z = y - e - \mu^{-1}\lambda$ is a constant vector. Eq.(14) can be turned into a standard LSR problem as below, if we set $v = \begin{bmatrix} w \\ b \end{bmatrix}$, $A = \begin{bmatrix} X^T & \mathbf{1} \\ \mu^{-\frac{1}{2}} I & \mathbf{0} \end{bmatrix}$ and $d = \begin{bmatrix} z \\ \mathbf{0} \end{bmatrix}$

$$G(w, b) = G(z) = \min_z \|Az - d\|^2\,, \tag{15}$$

which can be resolved by many standard libraries such as the default LSQR function in MATLAB.

Hereby we finish the illustration of the proposed exact SVM-ALM algorithm and summarize details in Algorithm 1. To be compatible with the existing methods such as (Fan et al., 2008; Chang et al., 2008; Hsieh et al., 2008; Lin et al., 2008), the stopping condition is set to be $\|\nabla obj(w, b)\| \le \epsilon$, where the user-specified parameter ϵ is 0.01 by default, and $\nabla obj(w, b)$ is the gradients of $obj(w, b)$ w.r.t. w.

However, the LSQR used here costs $O(n\bar{d}^2)$ where $\bar{d}$ is the average number of nonzero elements per instance, which is as costly as computing matrix inverse. This is too expensive as we need to afford such computation every iteration. Driven by this consideration and the tradeoff between cost per iteration and the number of iterations as discussed in the introduction, we use an optimal step-size gradient method to update w and b at each iteration.

The gradients of $G(w, b)$ w.r.t. w and b are as below:

$$\begin{aligned} w_g &= \frac{\partial G}{\partial w} = X(X^T w + \mathbf{1}b - z) + \mu^{-1} w\,, \\ b_g &= \frac{\partial G}{\partial b} = nb + \mathbf{1}^T(X^T w - z)\,. \end{aligned} \tag{16}$$

Finding the optimal step-size s is a single-variable quadratic function minimization problem:

$$\min_s \mu^{-1}(w - s w_g)^T (w - s w_g) + \|X^T(w - s w_g) + \mathbf{1}(b - s b_g) - z\|^2 \tag{17}$$

which has the explicit solution

$$\begin{aligned} s &= \frac{(X^T w_g + \mathbf{1}b_g)^T (X^T w + \mathbf{1}b - z) + \mu^{-1} w_g^T w}{(X^T w_g + \mathbf{1}b_g)^T (X^T w_g + \mathbf{1}b_g) + \mu^{-1} w_g^T w_g} \\ &= \frac{w_g^T w_g + b_g^T b_g}{(X^T w_g + \mathbf{1}b_g)^T (X^T w_g + \mathbf{1}b_g) + \mu^{-1} w_g^T w_g}\,. \end{aligned} \tag{18}$$

The last equality is just to simplify the computation of s, and can be verified via substituting w_g and b_g in two denominators with Eq.(16). We prefer the simplified formula, because it saves two matrix-by-vector multiplications.

Algorithm 1 Exact SVM-ALM for Lp-primal SVM

```
**Input:** $p, X, y, \mu_{(1)}, \mu_{(2)}, ..., \mu_{(\infty)}$
Initialize $w = \mathbf{1}, b = 0, \lambda = 0$.
**repeat**
    1. Update $e$ with Eq.(11) or Eq.(12) or Eq.(13).
    2. Update $w, b$ using the LSQR with Eq.(15).
    3. Update $\lambda$ with Eq.(9).
**until** $\|\nabla obj(w, b)\| \leq \epsilon$
```

Algorithm 2 Inexact SVM-ALM for Lp-primal SVM

```
**Input:** $p, X, y, \mu_{(1)}, \mu_{(2)}, ..., \mu_{(\infty)}$
Initialize $w = \mathbf{1}, b = 0, \lambda = 0$.
**repeat**
    1. Update $e$ with Eq.(11) or Eq.(12) or Eq.(13).
    2. Update $w$ by $w - sw_g$, update $b$ by $b - sb_g$, where
       $w_g, b_g, s$ are computed with Eq.(16) and Eq.(18), respectively.
    3. Update $\lambda$ with Eq.(9).
**until** $\|\nabla obj(w, b)\| \leq \epsilon$
```

We summarize the proposed inexact SVM-ALM algorithm in Algorithm 2. At each iteration, Algorithm 2 only needs three matrix-by-vector multiplications with complexity $O(n\bar{d})$, where $\bar{d}$ is the average number of nonzero elements per instance. The several pointwise addition and multiplication between two vectors are with complexity either $O(d)$ or $O(n)$, and can be neglected compared to $O(n\bar{d})$. In large-scale data classifications, the high dimensional features are always reduced by the prescreening procedure, hence $\bar{d}$ is not large. Our new algorithm has linear computational cost *w.r.t.* the number of data instances n.

3. Convergence and Optimality

We first prove some lemmas.

Lemma 1 *Let $\mathscr{H}$ be a real Hilbert space endowed with an inner product $<\cdot,\cdot>$ and a corresponding norm $\|\cdot\|$, and $v \in \partial\|u\|$, where $\partial f(u)$ is the subgradient of $f(u)$. Then $\|v\|^*=1$ if $u\neq 0$, and $\|v\|^* \leq 1$ if $u=0$, where $\|\cdot\|^*$ is the dual norm of $\|\cdot\|$.*

Proof: Because $v \in \partial\|u\|$,

$$\|d\| - \|u\| \geq <v, d-u>, \forall d \in \mathscr{H} \tag{19}$$

If $u\neq 0$, setting $d = 0, 2u$ leads to

$$\|u\| = <v, u> \leq \|u\|\|v\|^* \tag{20}$$

Thus we have $\|v\|^* \geq 1$. On the other side, we have

$$\|d - u\| \geq \|d\| - \|u\| \geq <v, d-u>, \forall d \in \mathscr{H} \tag{21}$$

which leads to

$$<v, \frac{d-u}{\|d-u\|}> \leq 1, \forall d \neq u \tag{22}$$

So $\|v\|^* \leq 1$. Then it can be concluded that $\|v\|^* = 1$.

If $u = 0$, then Eq.(19) is equivalent to

$$<v, d> \leq 1, \forall \|d\| = 1 \tag{23}$$

So $\|v\|^* \leq 1$ by the definition of the dual norm. □

Lemma 2 *The sequence $\{\lambda_{(k)}\}$ in either Algorithm 1 or Algorithm 2 is bounded.*

Proof: From $w_{(k)} = \arg\min_{w,b} AL(w, b, e_{(k)}, \lambda_{(k-1)}, \mu_{(k)})$, $b_{(k)} = \arg\min_{w,b} AL(w, b, e_{(k)}, \lambda_{(k-1)}, \mu_{(k)})$, $e_{(k)} = \arg\min_{e} AL(w_{(k-1)}, b_{(k-1)}, e, \lambda_{(k-1)}, \mu_{(k)})$, we have:

$$\begin{array}{c} 0 \in \partial_w AL(w_{(k)}, b_{(k)}, e_{(k)}, \lambda_{(k-1)}, \mu_{(k)}) \\ 0 \in \partial_b AL(w_{(k)}, b_{(k)}, e_{(k)}, \lambda_{(k-1)}, \mu_{(k)}) \\ 0 \in \partial_e AL(w_{(k)}, b_{(k)}, e_{(k)}, \lambda_{(k-1)}, \mu_{(k)}) \end{array} \tag{24}$$

which indicate:

$$\begin{array}{l} 0 \in \partial\|w_{(k)}\|^2 - \lambda_{(k-1)} - \mu_{(k)}(X^T w + \mathbf{1}b - y + e) \\ 0 \in \partial\|C(y^T e_{(k)})_+\|^p - \lambda_{(k-1)} - \mu_{(k)}(X^T w + \mathbf{1}b - y + e) \end{array} \tag{25}$$

Therefore

$$\lambda_{(k)} \in \partial\|w_{(k)}\|^2, \lambda_{(k)} \in \partial\|C(y^T e_{(k)})_+\|^p . \tag{26}$$

According to Lemma 1, the sequence $\{\lambda_{(k)}\}$ in Algorithm 1 is bounded, because of the fact that, the dual norms of $\|\cdot\|^2$ and $\|\cdot\|^p$ are $\|\cdot\|^2$ and $\|\cdot\|^{\frac{p}{p-1}}$ (Lin et al., 2009), respectively. The boundedness of $\{\lambda_{(k)}\}$ in Algorithm 2 can be proved in the same way. □

Lemma 3 *The sequences $\{w_{(k)}\}$,$\{b_{(k)}\}$,$\{e_{(k)}\}$ in either Algorithm 1 or Algorithm 2 are all bounded, if $\|w_{(k+1)}\|^2 + \|C(y^T e_{(k+1)})_+\|^p + 0.5\mu_{(k+1)}\|X^T w_{(k+1)} + \mathbf{1}b_{(k+1)} - y + e_{(k+1)}\|^2 \leq \|w_{(k)}\|^2 + \|C(y^T e_{(k)})_+\|^p + 0.5\mu_{(k)}\|X^T w_{(k)} + \mathbf{1}b_{(k)} - y + e_{(k)}\|^2$ for every $k > 0$ and $\sum_{k=1}^{\infty} \frac{\mu_{(k+1)}}{\mu_{(k)}^2} < \infty$.*

Proof: As $\|w\|^2 + \|C(y^T e)_+\|^p + 0.5\mu\|X^T w + \mathbf{1}b - y + e\|^2$ is non-increasing as Algorithm 1 iterates, it can be verified that,

$$\begin{array}{l} AL(w_{(k)}, b_{(k)}, e_{(k)}, \lambda_{(k-1)}, \mu_{(k)}) \\ \leq AL(w_{(k-1)}, b_{(k-1)}, e_{(k-1)}, \lambda_{(k-2)}, \mu_{(k-1)}) + \\ \quad 0.5\mu_{(k-1)}^{-2}(\mu_{(k-1)} + \mu_{(k)})\|\lambda_{(k-1)} - \lambda_{(k-2)}\|^2 \end{array} \tag{27}$$

The above inequality can be derived via substituting with Eq.(9) to eliminate $\lambda_{(k-1)}$. So $\{AL(w_{(k)}, b_{(k)}, e_{(k)}, \lambda_{(k-1)}, \mu_{(k)})\}$ is upper bounded, owing to the boundedness of $\{\lambda_{(k)}\}$ and $\sum_{k=1}^{\infty} \frac{\mu_{(k)} + \mu_{(k+1)}}{\mu_{(k)}^2} \leq \sum_{k=1}^{\infty} \frac{2\mu_{(k+1)}}{\mu_{(k)}^2} < \infty$.

Thus, we have

$$\|w_{(k)}\|^2{+}\|C(y^Te_{(k)})_+\|^p \\ =AL(w_{(k)},b_{(k)},e_{(k)},\lambda_{(k-1)},\mu_{(k)})-\frac{\|\lambda_{(k)}\|^2}{2\mu_{(k)}} \quad (28)$$

as upper bounded. Therefore $\{w_{(k)}\}$,$\{e_{(k)}\}$ in Algorithm 1 are both bounded, which leads to the boundedness of $\{b_{(k)}\}$, as $X^Tw{+}\mathbf{1}b{-}y{+}e=0$.

It can be verified that, exactly the same properties hold in Algorithm 2. □

The non-increasing requirement of $\|w\|^2{+}\|C(y^Te)_+\|^p$ $+0.5\mu\|X^Tw+\mathbf{1}b-y+e\|^2$ in Lemma 3 also implies the way to generate the sequence $\{\mu_{(k)}\}$ by setting the upper limit of $\mu_{(k)}$:

$$\begin{aligned}\mu_{(k+1)}=&(0.5\mu_{(k)}\|X^Tw_{(k)}+\mathbf{1}b_{(k)}-y+e_{(k)}\|^2+\|w_{(k)}\|^2\\ &-\|w_{(k+1)}\|^2{+}\|C(y^Te_{(k)})_+\|^p-\|C(y^Te_{(k+1)})_+\|^p)\\ &\div(0.5\|X^Tw_{(k+1)}+\mathbf{1}b_{(k+1)}-y+e_{(k+1)}\|^2)\end{aligned} \quad (29)$$

Because of Eqs.(10,14), we have

$$\begin{aligned}&AL(w_{(k)},b_{(k)},e_{(k)},\lambda_{(k-1)},\mu_{(k)})\\ \le&AL(w_{(k-1)},b_{(k-1)},e_{(k)},\lambda_{(k-1)},\mu_{(k)})\\ \le&AL(w_{(k-1)},b_{(k-1)},e_{(k-1)},\lambda_{(k-1)},\mu_{(k)})\end{aligned}$$

which ensures that $\{\mu_{(k)}\}$ is non-decreasing.

Owing to precision limit, μ cannot increase to infinity in practical implementations of both Algorithm 1 and Algorithm 2, otherwise the significant digits of the terms $\frac{1}{2}w^Tw$ and $C\sum_{i=1}^{n}(y_ie_i)_+^p$ in $AL(w,b,e,\lambda,\mu)$ would be squeezed out by the extremely large term $\frac{\mu}{2}\|X^Tw+\mathbf{1}b-y+e+\frac{\lambda}{\mu}\|^2$. More specifically, μ has a upper limit of 10^5 as an implementation detail. We follow the convention of most existing work by using double-precision floating-point numbers. Using single precision *e.g.* (Bottou, 2007) may reduce the computational time in some situations, but this setting may cause numerical inaccuracy (Chang et al., 2008). An advantage of the ALM is that it converges to the exact optimal solution before μ augments to infinity (Gill & Robinson, 2012). In contrast, strictly speaking the IT (Wright et al., 2009) only finds approximate solutions.

Now we have come to the main results of this section. **Theorem 1** *The solution consisting of the limit of the sequences $\{w_{(k)}\}$,$\{b_{(k)}\}$,$\{e_{(k)}\}$ in Algorithm 1 with Eq.(29) for updating μ, say $(w_{(\infty)}$,$b_{(\infty)}$,$e_{(\infty)})$, is an optimal solution to the Lp-primal SVM problem and the convergence rate is at least $O(\mu_{(k)}^{-1})$ in the sense that $|\|w_{(k)}\|^2{+}\|C(y^Te_{(k)})_+\|^p-obj^*|{=}O(\mu_{(k)}^{-1})$, where obj^* is the minimal value of obj in Eq.(4).*

Proof: As the vital natural property of an ALM algorithm, the following is true:

$$\begin{aligned}&AL(w_{(k)},b_{(k)},e_{(k)},\lambda_{(k-1)},\mu_{(k)}){=}\min_{w,b,e}AL(w,b,e,\lambda_{(k-1)},\mu_{(k)})\\ &\le\min_{w,b,e,X^Tw+\mathbf{1}b-y+e=0}AL(w,b,e,\lambda_{(k-1)},\mu_{(k)})\\ &=\min_{w,b,e,X^Tw+\mathbf{1}b-y+e=0}\|w\|^2{+}\|C(y^Te)_+\|^p+\tfrac{\|\lambda_{(k-1)}\|^2}{2\mu_{(k)}}\\ &=\min_{w,b}\|w\|^2{+}\|C(1-(X^Tw+\mathbf{1}b)y)_+\|^p+\tfrac{\|\lambda_{(k-1)}\|^2}{2\mu_{(k)}}\\ &=obj^*+\tfrac{\|\lambda_{(k-1)}\|^2}{2\mu_{(k)}}\end{aligned} \quad (30)$$

The first equality and second inequality are obvious; the third equality is because of the fact that, when the constraints w.r.t the auxiliary variables e is satisfied, the last term in Eq.(8) degenerates to $||\lambda_{(k-1)}||^2/2\mu_{(k)}$; the fourth equality is obtained just by substituting the constraints, similar to the conversion from Eq.(5) to Eq.(4); the fifth equality is according to the definition in Eq.(4).

In Algorithm 1, it can be verified that,

$$\|w_{(k)}\|^2{+}\|C(y^Te_{(k)})_+\|^p{=} \\ AL(w_{(k)},b_{(k)},e_{(k)},\lambda_{(k-1)},\mu_{(k)})-\tfrac{\|\lambda_{(k)}\|^2}{2\mu_{(k)}} \quad (31)$$

Based on Eq.(30) we have

$$\|w_{(k)}\|^2{+}\|C(y^Te_{(k)})_+\|^p\le obj^*+\frac{\|\lambda_{(k-1)}\|^2}{2\mu_{(k)}}-\frac{\|\lambda_{(k)}\|^2}{2\mu_{(k)}}$$

The proved boundedness of $\{\lambda_{(k)}\}$ in Lemma 2 leads to:

$$obj^*{-}O(\mu_{(k)}^{-1})\le\|w_{(k)}\|^2{+}\|C(y^Te_{(k)})_+\|^p\le obj^*{+}O(\mu_{(k)}^{-1})$$

Note that the range $[obj^*-O(\mu_{(k)}^{-1}),obj^*+O(\mu_{(k)}^{-1})]$ is derived, as the term $O(\mu_{(k)}^{-1})$ may be either positive or negative. Hereby the convergence rate is proved.

When $k\to\infty$, $O(\mu_{(k)}^{-1})$ is negligible, so

$$\|w_{(\infty)}\|^2{+}\|C(y^Te_{(\infty)})_+\|^p\le obj^* \quad (32)$$

According to Eq.(9), the constraints $X^Tw_{(k)}+\mathbf{1}b_{(k)}-y+e_{(k)}=\mu_{(k)}^{-1}(\lambda_{(k)}-\lambda_{(k-1)})$ are satisfied when $k\to\infty$:

$$X^Tw_{(\infty)}+\mathbf{1}b_{(\infty)}-y+e_{(\infty)}=0 \quad (33)$$

Therefore, $(w_{(\infty)}$,$b_{(\infty)}$,$e_{(\infty)})$ is an optimal solution to the Lp-primal SVM problem. □

Theorem 2 *The solution consisting of the limit of the sequences $\{w_{(k)}\}$,$\{b_{(k)}\}$,$\{e_{(k)}\}$ in Algorithm 2 with Eq.(29) for updating μ, say $(w_{(\infty)}$,$b_{(\infty)}$,$e_{(\infty)})$, is an optimal solution to the Lp-primal SVM problem, if $\sum_{k=1}^{\infty}\frac{\mu_{(k+1)}}{\mu_{(k)}^2}<\infty$ and* $\lim_{k\to\infty}\mu_{(k)}(e_{(k+1)}-e_{(k)})=0$.

Note that, unlike Theorem 1 for the exact ALM method, the above statement only guarantees convergence but does not specify the rate of convergence for the inexact ALM method. Although the exact convergence rate of the inexact ALM method is difficult to obtain in theory, extensive numerical experiments have shown that for geometrically increasing μ, it still converges Q-linearly (Gill & Robinson, 2012; Lin et al., 2009).

Proof: Our proof here is based on Theorem 1 by comparing the difference of $\{w_{(k)}\}$,$\{b_{(k)}\}$,$\{e_{(k)}\}$ and $\{\lambda_{(k)}\}$ in Algorithm 1 and Algorithm 2. For distinction purpose, we denote $\{w_{(k)}\}$,$\{b_{(k)}\}$,$\{e_{(k)}\}$ and $\{\lambda_{(k)}\}$ in Algorithm 1 as $\{\hat{w}_{(k)}\}$,$\{\hat{b}_{(k)}\}$,$\{\hat{e}_{(k)}\}$ and $\{\hat{\lambda}_{(k)}\}$ respectively, in this proof.

According to $X^T w_{(k)} + \mathbf{1} b_{(k)} - y + e_{(k)} = \mu_{(k)}^{-1}(\lambda_{(k)} - \lambda_{(k-1)})$ from Eq.(9) and the boundedness of $\{\lambda_{(k)}\}$, we have

$$\lim_{k\to\infty} X^T w_{(k)} + \mathbf{1} b_{(k)} - y + e_{(k)} = 0 \tag{34}$$

So $(w_{(k)}, b_{(k)}, e_{(k)})$ approaches a feasible solution. Further, the boundedness of $\{\lambda_{(k)}\}$ and $\{\hat{\lambda}_{(k)}\}$ leads to:

$$\|e_{(k+1)} - e_{(k)}\| = O(\mu_{(k)}^{-1}\|\hat{\lambda}_{(k+1)} - \lambda_{(k+1)}\|) = O(\mu_{(k)}^{-1})$$

Since $\sum_{k=1}^{\infty} \mu_{(k)}^{-1} \le \sum_{k=1}^{\infty} \frac{\mu_{(k)}+\mu_{(k+1)}}{\mu_{(k)}^2} \le \sum_{k=1}^{\infty} \frac{2\mu_{(k+1)}}{\mu_{(k)}^2} < \infty$, $e_{(k)}$ is a Cauchy sequence, and has a limit $e_{(\infty)}$. Then with Eq.(34), $w_{(k)}$ and $b_{(k)}$ also have their corresponding limits $w_{(\infty)}$ and $b_{(\infty)}$. So $(w_{(\infty)}, b_{(\infty)}, e_{(\infty)})$ is a feasible solution. On the other side, we have the optimality condition:

$$\lambda_{(k)} \in \partial\|w_{(k)}\|^2, \lambda_{(k)} \in \partial\|(y^T e_{(k)})_+\|^p . \tag{35}$$

Thus, by the convexity of norms (for $1 \le p \le 2$) we have:

$$\begin{aligned}
&\|w_{(k)}\|^2 + ||C(y^T e_{(k)})_+||^p \\
&\le obj^* - <\hat{\lambda}_{(k)}, \hat{w}_{(k)} - w_{(k)}> - <\lambda_{(k)}, \hat{e}_{(k)} - e_{(k)}> \\
&= obj^* - \mu_{(k)}^{-1} <\lambda_{(k)}, \lambda_{(k)} - \lambda_{(k-1)}> + \mu_{(k)}^{-1} <\lambda_{(k)}, \hat{\lambda}_{(k)} \\
&\quad - \hat{\lambda}_{(k-1)}> - <\mu_{(k)}(e_{(k)} - e_{(k-1)}), \hat{w}_{(k)} - w_{(k)}>
\end{aligned} \tag{36}$$

The second and third terms approach to zero due to the boundedness of $\{\lambda_{(k)}\}$ and $\{\hat{\lambda}_{(k)}\}$. The last term tends to vanish due to the boundedness of $\{w_{(k)}\}$ and $\{\hat{w}_{(k)}\}$ together with the assumption $\lim_{k\to\infty} \mu_{(k)}(e_{(k+1)} - e_{(k)}) = 0$. So when $k \to \infty$, Eq.(36) becomes

$$\|w_{(\infty)}\|^2 + \|C(y^T e_{(\infty)})_+\|^p \le obj^* . \tag{37}$$

So $(w_{(\infty)}, b_{(\infty)}, e_{(\infty)})$ is an optimal solution to the Lp-primal SVM problem. □

4. Experiments

This paper follows the concepts of reproducible research. All results presented in the manuscript are reproducible using the code and public datasets available online at https://sites.google.com/site/svmalm. All experiments are conducted on an 8-core Intel Xeon X5460 3.16GHz (12M Cache, 1333 MHz FSB) Linux server with 32G memory. For all experiments except in §4.3, we use the default value ϵ=0.01 as in LibLinear. We terminate the algorithms when the objectives' changes are less than 10^{-4}. In our method, we empirically set the maximum iteration number as 100, because in all our experiments our algorithm converges within 100 iterations.

We use 7 popularly adopted benchmark datasets from various sources for performance evaluations: **UCI Forest** (Collobert et al., 2002) ($n = 581,012, d = 54$), **ijcnn1** (Chang & Lin, 2001) ($n = 191,681, d = 22$), **Webpage** (Platt, 1999) ($n = 64,700, d = 300$), **UCI Connect-4** (Frank & Asuncion, 2010) ($n = 67,557, d = 126$), **SensIT Vehicle (acoustic/seismic)** (Duarte & Hu, 2004) (both $n = 98,528, d = 50$), **Shuttle** (Hsu & Lin, 2002) ($n = 58,000, d = 9$), **UCI Poker** (Frank & Asuncion, 2010) ($n = 1,025,010, d = 10$), **Epsilon** (Sonnenburg et al., 2008) ($n = 500,000, d = 2000$). The **Epsilon** dataset has very dense features and was used in many previous large-scale data classifications. The five-fold cross validation is conducted (except in §4.3 when all samples are used for training) as in (Chang et al., 2008).

For multi-class classification, we follow the default one-versus-the-rest strategy in (Chang & Lin, 2011) and (Fan et al., 2008), and simply rely on the existing modules in the LibLinear software toolbox. The average training time is reported.

4.1. How Does Training Time Varies with n?

Fig. 1 shows log-log plots of how the CPU-time used for training increases with respect to n, the number of training samples. Because when n is small the training time is too short to be measured accurately, we run each test for 10 times and report the total training time in Fig. 1.

Lines in a log-log plot correspond to polynomial growth $O(n^l)$, where l corresponds to the slope of the line. It is seen from Fig. 1 that, the training time of both the exact SVM-ALM and the inexact SVM-ALM is roughly linear with respect to n, since the slopes of the lines representing various datasets are very close to 1. Together with the theoretical analysis in §2 that one iteration of the inexact SVM-ALM algorithm costs $O(n\bar{d})$, Algorithm 2 is shown to be a linear computational cost solver for the Lp-primal SVM.

Note that an advantage of our algorithms is that, the training time (and obviously the testing time as well) is completely irrelevant with weight C and norm p.

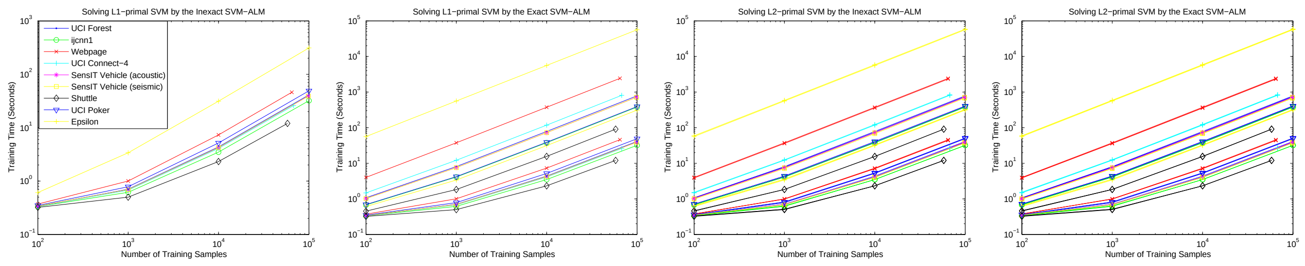

Figure 1. Training time of the proposed exact SVM-ALM (Algorithm 1) and inexact SVM-ALM (Algorithm 2) as a function of n.

4.2. Prediction Accuracy Comparison between Exact and Inexact SVM-ALM Algorithms

A natural drawback of the Inexact SVM-ALM Algorithm is that it still requires μ augments to infinity for obtaining the exact optimal solution, as analyzed in the proof of Theorem 2. This property is similar to the IT algorithms (Wright et al., 2009). However, owing to precision limit as discussed in §2, μ cannot increase to infinity in practical implementations of the Inexact SVM-ALM Algorithm 2. So a potential concern is that the speedup of the Inexact SVM-ALM over the Exact SVM-ALM comes at the expense of prediction accuracies, but this is not the case in fact, as verified experimentally in this subsection.

Fig. 2 shows the difference in terms of prediction accuracy between the classification models produced by the inexact SVM-ALM and the exact SVM-ALM. For better readability, the axis of C is plotted in log-scale, and the difference is shown in terms of percentage points. A positive value indicates that the inexact SVM-ALM has higher prediction accuracy, while a negative value indicates that the exact SVM-ALM performs better. For almost all values of C both algorithms perform almost identically. In particular, there is no indication that the models learned by the inexact SVM-ALM are less accurate. Contrarily, the prediction accuracy of the inexact SVM-ALM may be slightly better than that of the exact SVM-ALM, and such phenomena is reasonable because it has been reported that some implementations of SVM solvers achieve higher accuracy before the objective function reaches its minimal (Chang et al., 2008).

4.3. Training Time Comparison

The proposed Algorithm 2 is compared with the state of the art solvers SVM^{perf}, Pegasos, BMRM (Bundle Method for Regularized Risk Minimization) (Teo et al., 2010) and the LibLinear that integrates the TRON, PCD and DCD algorithms.

The L1-primal SVM cannot be solved by the PCD (Chang et al., 2008), because its objective function Eq.(2) is non-differentiable. Thus the PCD is missing from the test for the L1-primal SVM. As a convention (Joachims, 2006) (Shalev-Shwartz et al., 2007) (Chang et al., 2008) (Hsieh et al., 2008) (Lin et al., 2008), SVM^{perf}, Pegasos and the TRON method are typically only tested for the L1-primal SVM.

Because the TRON, PCD and DCD algorithms do not support the bias term b, we extend each instance by an additional dimension with a large constant $T = 10^3$, as instructed in (Hsieh et al., 2008; Lin et al., 2008). As long as the constant T in the additional dimension is sufficiently large, such conversion is equivalent to supporting the training of the bias term b.

With the same settings as in (Chang et al., 2008) (Hsieh et al., 2008) we compare the L1-SVM and L2-SVM solvers in term of the training time to reduce the objective function $obj(\cdot)$ such that the relative difference of obj to the optimum obj^*, $(obj - obj^*)/|obj^*|$, is within 0.01. In order to obtain the reference solutions, we run TRON with the stopping condition $\nabla obj(w) \leq 0.01$. Since the objective functions are stable under such strict stopping conditions, these solutions are seen to be very close to the ground-truth optima. The results are listed in Tables 2 and 3, from which it is seen that, the proposed algorithm is with stable performance and on average faster than its competitors. The advantage of the propose algorithm is more obvious for large datasets, such as the UCI Forest, SensIT Vehicle, and UCI Poker datasets. The DCD algorithm is not stable, as it may get stuck at some testcases but converges extremely fast at other testcases. When the dimensionality of features increases to 2000 as the Epsilon data, our algorithm still performs well, and is the fastest solver for L1-SVM and the second fastest solver for L2-SVM.

4.4. The Optimal p for Lp-Primal SVM

A natural advantage of our proposed algorithms is that, it can solve the primal SVM with Lp-norm loss functions for any $p \geq 1$. It is not difficult to understand the fact that, it should be coincidental for either p=1 or p=2 to make the prediction accuracy of the Lp-primal SVM the highest among all possible p values.

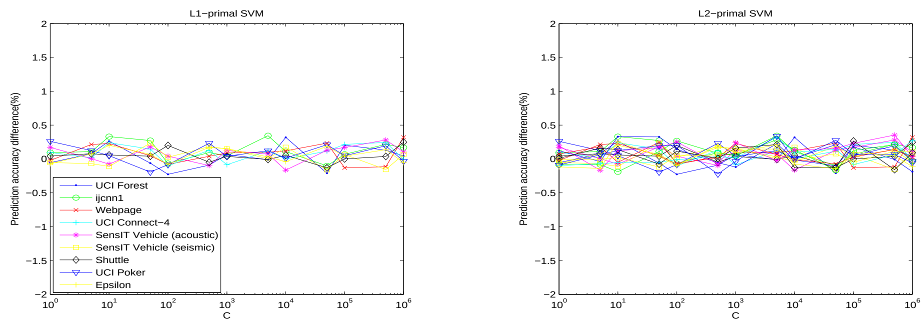

Figure 2. Prediction accuracy difference between the inexact SVM-ALM (Algorithm 2) and the exact SVM-ALM (Algorithm 1) for L1-primal and L2-primal SVMs as a function of C.

Table 1. The training time (seconds) for an L1-SVM solver to reduce $obj(\cdot)$ to within 1% of the optimal value. Though the training time of the proposed algorithms is irrelevant with C, the training time of SVM^{perf}, TRON, PCD and DCD may be affected by C. Following (Chang et al., 2008) and (Hsieh et al., 2008), we set C = 1 for fair comparison. The training time is measured and averaged over 10 runs. The solver with the shortest running time is boldfaced.

Dataset	Our	Pegasos	SVM^{perf}	DCD	BMRM
Forest	**4.1**	74.1	139.2	>500	51.8
ijcnn1	**3.2**	87.9	105.6	7.8	63.5
Webpage	4.6	38.3	62.1	**3.6**	30.2
Connect-4	**2.6**	54.2	122.6	>500	42.9
SensIT (A)	**3.9**	128.7	399.8	17.0	102.5
SensIT (S)	**3.9**	109.3	335.9	11.1	85.2
Shuttle	**1.2**	29.6	66.6	2.2	20.6
Poker	**4.9**	107.4	303.1	>500	80.6
Epsilon	**31.1**	396.4	>500	93.2	315.2

Thus we conduct an interesting experiment showing this phenomenon. Because existing SVM solvers cannot solve the Lp-primal SVM for $p\neq 1$ or 2, we believe that we are the first to report such results in Table 3.

5. Conclusion

This paper proposed a novel linear computational cost primal SVM solver using the ALM algorithm for both the L1-norm and the L2-norm loss functions. To avoid the difficulty of dealing with piecewise loss functions, an auxiliary vector is introduced such that in each iteration, the auxiliary vector and the support vector are alternatively optimized with the direction of Lagrange multipliers. In extensive experiments, our approach is consistently faster than other state-of-the-art solvers. From the methodological perspective, the proposed algorithm is novel and totally different from existing literatures.

Table 2. The training time (seconds) for an L2-SVM solver to reduce $obj(\cdot)$ to within 1% of the optimal value when C = 1, the same as in Table 1. The training time is measured and averaged over 10 runs. The solver with the shortest running time is boldfaced.

Dataset	Our	TRON	PCD	DCD	BMRM
Forest	**3.9**	92.3	10.0	>500	50.6
ijcnn1	**3.2**	7.7	3.4	7.5	64.2
Webpage	4.4	2.2	**0.9**	3.9	32.1
Connect-4	**2.7**	10.4	3.9	>500	39.7
SensIT (A)	**3.9**	27.7	5.3	17.5	99.8
SensIT (S)	**3.7**	28.1	4.9	10.9	86.1
Shuttle	1.2	3.6	**0.9**	2.4	21.1
Poker	**5.1**	59.7	7.1	>500	79.8
Epsilon	32.6	241.9	**16.9**	83.2	329.2

Table 3. Prediction accuracy of L1-SVM, L2-SVM and Lp-SVM, where p is tuned by trying the parameter set $\{$1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2$\}$.

Dataset	L1-SVM	L2-SVM	Lp-SVM	p
Forest	68.1%	65.3%	71.0%	1.3
ijcnn1	67.3%	74.2%	74.6%	1.9
Webpage	57.3%	59.7%	63.4%	1.6
Connect-4	49.3%	44.9%	51.8%	1.2
SensIT (A)	43.5%	45.9%	47.3%	1.8
SensIT (S)	41.6%	42.4%	46.8%	1.6
Shuttle	35.9%	29.7%	36.1%	1.1
Poker	31.5%	33.8%	36.9%	1.7
Epsilon	42.9%	40.3%	44.6%	1.4

Acknowledgments

Corresponding Author: Heng Huang (heng@uta.edu)
This work was partially supported by US NSF IIS-1117965, IIS-1302675, IIS-1344152.

References

Bottou, Leon. Stochastic gradient descent examples, 2007. http://leon.bottou.org/projects/sgd.

Chang, Chih-Chung and Lin, Chih-Jen. Ijcnn 2001 challenge: Generalization ability and text decoding. In *IJCNN*, pp. 1031–1036, 2001.

Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: A library for support vector machines. *ACM TIST*, 2(3):27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.

Chang, Kai-Wei, Hsieh, Cho-Jui, and Lin, Chih-Jen. Coordinate descent method for large-scale L2-loss linear svm. *JMLR*, 9:1369–1398, 2008.

Collobert, R., Bengio, S., and Bengio, Y. A parallel mixture of svms for very large scale problems. *Neural Computation*, 14(5):1105–1114, 2002.

Duarte, M. and Hu, Y. H. Vehicle classification in distributed sensor networks. *JPDC*, 64(7):826–838, 2004.

Fan, Rong-En, Chang, Kai-Wei, Hsieh, Cho-Jui, Wang, Xiang-Rui, and Lin, Chih-Jen. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

Frank, A. and Asuncion, A. Uci machine learning repository, 2010. http://archive.ics.uci.edu/ml.

Gill, Philip E. and Robinson, Daniel P. A primal-dual augmented lagrangian. *Computational Optimization and Applications*, 51(1):1–25, 2012.

Hsieh, Cho-Jui, Chang, Kai-Wei, Keerthi, S. Sathiya, Sundararajan, S., and Lin, Chih-Jen. A dual coordinate descent method for large-scale linear svm. In *ICML*, pp. 408–415, 2008.

Hsu, Chih-Wei and Lin, Chih-Jen. A comparison of methods for multi-class support vector machines. *IEEE TNN*, 13(2):415–425, 2002.

Joachims, Thorsten. Training linear svms in linear time. In *KDD*, pp. 217–226, 2006.

Keerthi, S. Sathiya and DeCoste, Dennis. A modified finite newton method for fast solution of large scale linear svms. *JMLR*, 6:341–361, 2005.

Lin, Chih-Jen, Weng, Ruby C., and Keerthi, S. Sathiya. Trust region newton method for large-scale logistic regression. *JMLR*, 9:627–650, 2008.

Lin, Zhouchen, Chen, Minming, Wu, Leqin, and Ma, Yi. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. 2009.

Mangasarian, O. L. and Musicant, David R. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001.

Mangasarian, Olvi L. A finite newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

Platt, John C. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*, pp. 185–208, 1999.

Shalev-Shwartz, Shai, Singer, Yoram, and Srebro, Nathan. Pegasos: primal estimated subgradient solver for svm. In *ICML*, pp. 807C814, 2007.

Smola, Alex J., Vishwanathan, S. V. N., and Le, Quoc. Bundle methods for machine learning. In *NIPS*, pp. 1377–1384, 2008.

Sonnenburg, Soeren, Franc, Vojtech, Yom-Tov, Elad, and Sebag, Michele. Pascal large scale learning challenge, 2008. http://largescale.ml.tu-berlin.de.

Teo, Choon Hui, Vishwanathan, S.V.N., Smola, Alex, and Le, Quoc V. Bundle methods for regularized risk minimization. *JMLR*, 11:311–365, 2010.

Wright, John, Ganesh, Arvind, Rao, Shankar, and Ma, Yi. Robust principal component analysis: exact recovery of corrupted low-rank matrices by convex optimization. In *NIPS*, pp. 2080–2088, 2009.

Zhang, Tong. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML*, pp. 919–926, 2004.