NESVM: A Fast Gradient Method for Support Vector Machines

Tianyi Zhou
School of Computer Engineering, Nanyang Technological University, Singapore
Email:tianyi.david.zhou@gmail.com

Dacheng Tao
School of Computer Engineering, Nanyang Technological University, Singapore
Email:dacheng.tao@ieee.org

Xindong Wu[1,2]
[1] *School of Computer Science & Info. Engg., Hefei University of Technology, Hefei, China*
[2]*Department of Computer Science, University of Vermont, Burlington, VT, USA*
Email: xwu@cs.uvm.edu

***Abstract*—Support vector machines (SVMs) are invaluable tools for many practical applications in artificial intelligence, e.g., classification and event recognition. However, popular SVM solvers are not sufficiently efficient for applications with a great deal of samples as well as a large number of features. In this paper, thus, we present NESVM, a fast gradient SVM solver that can optimize various SVM models, e.g., classical SVM, linear programming SVM and least square SVM. Compared against SVM-Perf [1][2] (whose convergence rate in solving the dual SVM is upper bounded by $\mathcal{O}(1/\sqrt{k})$ where k is the number of iterations) and Pegasos [3] (online SVM that converges at rate $\mathcal{O}(1/k)$ for the primal SVM), NESVM achieves the optimal convergence rate at $\mathcal{O}(1/k^2)$ and a linear time complexity. In particular, NESVM smoothes the non-differentiable hinge loss and ℓ_1-norm in the primal SVM. Then the optimal gradient method without any line search is adopted to solve the optimization. In each iteration round, the current gradient and historical gradients are combined to determine the descent direction, while the Lipschitz constant determines the step size. Only two matrix-vector multiplications are required in each iteration round. Therefore, NESVM is more efficient than existing SVM solvers. In addition, NESVM is available for both linear and nonlinear kernels. We also propose "homotopy NESVM" to accelerate NESVM by dynamically decreasing the smooth parameter and using the continuation method. Our experiments on census income categorization, indoor/outdoor scene classification event recognition and scene recognition suggest the efficiency and the effectiveness of NESVM. The MATLAB code of NESVM will be available on our website for further assessment.**

***Keywords*-Support vector machines; smooth; hinge loss; ℓ_1 norm; Nesterov's method; continuation method.**

I. Introduction

Support Vector Machines (SVMs) are prominent machine learning tools for practical artificial intelligence applications [4][5]. However, existing SVM solvers are not sufficiently efficient for practical problems, e.g., scene classification and event recognition, with a large number of training samples as well as a great deal of features. This is because the time cost of working set selection or Hessian matrix computation in conventional SVM solvers rapidly increases with the slight augmenting of the data size and the feature dimension. In addition, they cannot converge quickly to the global optimum. Recently, efficient SVM solvers have been intensively studied on both dual and primal SVMs.

Decomposition methods, e.g., sequential minimal optimization (SMO) [6], LIBSVM [7] and SVM-Light [8], were developed to reduce the space cost for optimizing the dual SVM. In each iteration round, they consider a subset of constraints that are relevant to the current support vectors and optimize the corresponding dual problem on the selected working set by casting it into a quadratic programming (QP) problem. However, they are impractical to handle large scale problems, because their time complexities are super linear in n and the maximization of the dual objective function leads to a slow convergence rate to the optimum of the primal objective function.

Structural SVM, e.g., SVM-Perf [1][2][9], is recently proposed to improve the efficiency of optimization on the dual SVM. It reformulates the classical SVM into a structural form. In each iteration round, it firstly computes the most violated constraint from the training set by using a cutting-plane algorithm and adds this constraint to the current working set, then a QP solver is applied to optimize the corresponding dual problem. The Wolfe Dual of structural SVM is sparse and thus the size of each QP problem is small. It has been proved that the convergence rate of SVM-Perf is upper bounded by $\mathcal{O}(1/\sqrt{k})$ and a lot of successful applications show the efficiency of SVM-Perf. However, it cannot work well when classes are difficult to be separated, e.g., the overlap between classes is serious or distributions of classes are seriously imbalanced. In this scenario, a large C is required to increase the support vectors and thus it is inefficient to find the most violated constraint in SVM-Perf.

Many recent research results [10] show advantages to solve the primal SVM on large scale datasets. However, it is inefficient to directly solve the corresponding QP of the primal SVM if the number of constraints is around the number of samples, e.g., the interior point method for solving the primal SVM. One available solution is to write each of the constraints to the objective function as a hinge loss $\hbar$ and reformulate the problem as an unconstrained one.

Let $X \in \mathbb{R}^{n\times p}$ and $y \in \mathbb{R}^n$ be the training dataset and the corresponding label vector, respectively, where the vector $X_i \in \mathbb{R}^p$ is the i^{th} sample in X and $y_i \in \{1, -1\}$

is the corresponding label. Let the weight vector w be the classification hyper-plane. The reformulated primal problem of classical SVM is given by

$$\min_{w\in\mathbb{R}^p} F(w) = \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n} \hbar\left(y_i X_i, w\right), \tag{1}$$

$$\hbar\left(y_i X_i, w\right) = \max\left\{0, 1 - y_i X_i w\right\}. \tag{2}$$

Since the hinge loss is non-differentiable, first order methods, e.g., subgradient and stochastic gradient methods, can achieve the solution with the convergence rate $\mathcal{O}(1/\sqrt{k})$, which is not sufficiently fast for large-scale problems. Second order methods, e.g., Newton and Quasi-Newton methods, can obtain the solution as well by replacing the hinge loss with differentiable approximations, e.g., $\max\left\{0, 1 - y_i X_i w\right\}^q$ used in [10] or the integral of sigmoid function used in [11]. Although, the second order methods achieve the optimal convergence rate at $\mathcal{O}(1/k^2)$, it is expensive to calculate the Hessian matrix in each iteration round. Therefore, it is impractical to optimize the primal SVM by using the second order methods.

Recently, Pegasos [3], a first order online method, was proposed by introducing a projection step after each stochastic gradient update. It converges at rate $\mathcal{O}(1/k)$. In addition, its computational cost can be rapidly reduced if the feature is sparse, because the computation of the primal objective gradient can be significantly simplified. Therefore, it has been successfully applied to document classification. However, it hardly outperforms SVM-Perf when the feature is dense, which is a frequently encountered situation in artificial intelligence, e.g., computer vision tasks.

In this paper, we present and analyze a fast gradient SVM framework, i.e., NESVM, which can solve the primal problems of typical SVM models, i.e., classical SVM (C-SVM) [12], linear programming SVM (LP-SVM) [13] and least square SVM (LS-SVM) [14], with the proved optimal convergence rate $\mathcal{O}(1/k^2)$ and a linear time complexity. The "NES" in NESVM refers to Nesterov's method to acknowledge the fact that NESVM is based on the method. Recently, Nesterov's method has been successfully applied to various optimization problems [15][16], e.g., compressive sensing, sparse covariance selection, sparse PCA and matrix completion. The proposed NESVM smoothes the non-differentiable parts, i.e., hinge loss and ℓ_1-norm in the primal objective functions of SVMs, and then uses a gradient-based method with the proved optimal convergence rate to solve the smoothed optimizations. In each iteration round, two auxiliary optimizations are constructed, a weighted combination of their solutions is assigned as the current SVM solution, which is determined by the current gradient and historical gradients. In each iteration round, only two matrix-vector multiplications are required. Both linear and nonlinear kernels can be easily applied to NESVM. The speed of NESVM remains fast when dealing with dense features.

We apply NESVM to census income categorization [17], indoor scene classification [18], outdoor scene classification [19] and event recognition [20] on publicly available datasets. In these applications, we compare NESVM against four popular SVM solvers, i.e., SVM-Perf, Pegasos, SVM-Light and LIBSVM. Extensive experimental results indicate that NESVM achieves the shortest CPU time and a comparable performance among all the SVM solvers.

II. NESVM

We write typical primal SVMs in the following unified form:

$$\min_{w\in\mathbb{R}^p} F(w) = R(w) + C \cdot L(y_i X_i, w), \tag{3}$$

where $R(w)$ is a regularizer inducing the margin maximization in SVMs, $L(y_i X_i, w)$ is a loss function for minimizing the classification error, and C is the SVM parameter. For example, $R(w)$ is the ℓ_2-norm of w in C-SVM, $R(w)$ is the ℓ_1-norm of w in LP-SVM, $L(y_i X_i, w)$ is the hinge loss of the classification error in C-SVM, and $L(y_i X_i, w)$ is the least square loss of the classification error in LS-SVM. It is worth emphasizing that nonlinear SVMs can be unified as Eq.3 as well. Details are given at the end of Section 2.3.

In this section, we introduce and analyze the proposed fast gradient SVM framework, i.e., NESVM, based on Eq.3. We first show that the non-differentiable parts in SVMs, i.e., the hinge loss and the ℓ_1-norm, can be written as saddle point functions and smoothed by subtracting respective prox-functions. We then introduce Nesterov's method [21] to optimize the smoothed SVM objective function $F_\mu(w)$. In each iteration round of NESVM, two simple auxiliary optimizations are constructed and the optimal linear combination of their solutions is adopted as the solution of Eq.3 at the current iteration round. NESVM is a first order method and achieves the optimal convergence rate of $\mathcal{O}(1/k^2)$. In each iteration round, it requires only two matrix-vector multiplications. We analyze the convergence rate and time complexity of NESVM theoretically. An accelerated NESVM using continuation method, i.e., "homotopy NESVM" is introduced at the end of this section. Homotopy NESVM solves a sequence of NESVM with decreasing smooth parameter to obtain an accurate approximation of the hinge loss. The solution of each NESVM is used as the "warm start" of the next NESVM in homotopy NESVM and thus the computational time for each NESVM can be significantly saved.

A. Smooth the hinge loss

In SVM and LP-SVM, the loss function is given by the sum of all the hinge losses, i.e., $L(y_i X_i, w) = \sum_{i=1}^{n} \hbar\left(y_i X_i, w\right)$, which can be equivalently replaced by

the following saddle point function,

$$\min_{w\in\mathbb{R}^p}\sum_{i=1}^{n}\hbar\left(y_iX_i,w\right)=\min_{w\in\mathbb{R}^p}\max_{u\in\mathcal{Q}}\langle e-YXw,u\rangle, \quad (4)$$
$$\mathcal{Q}=\left\{u:0\le u_i\le 1,u\in\mathbb{R}^n\right\},$$

where e is a vector full of 1 and $Y=\mathrm{Diag}(y)$. According to [21], the above saddle point function can be smoothed by subtracting a prox-function $d_1(u)$. The $d_1(u)$ is a strongly convex function of u with a convex parameter $\sigma_1>0$ and the corresponding prox-center $u_0=\arg\min_{u\in\mathcal{Q}}d_1\left(u\right)$. Let A_i be the i^{th} row of the matrix A. We adopt $d_1\left(u\right)=\sum_{i=1}^{n}\|X_i\|_\infty u_i^2$ in NESVM and thus the smoothed hinge loss $\hbar_\mu$ can be written as,

$$\hbar_\mu=\max_{u\in\mathcal{Q}}u_i\left(1-y_iX_iw\right)-\frac{\mu}{2}\|X_i\|_\infty u_i^2, \quad (5)$$

where μ is the smooth parameter. Since $d_1(u)$ is strongly convex, u_i can be obtained by setting the gradient of the objective function in Eq.5 as zero and then projecting u_i on $\mathcal{Q}$, i.e.,

$$u_i=\mathrm{median}\left\{\frac{1-y_iX_iw}{\mu\|X_i\|_\infty},0,1\right\}. \quad (6)$$

Therefore, the smoothed hinge loss $\hbar_\mu$ is a piece-wise approximation of $\hbar$ according to different choices of u_i in Eq.6, i.e.,

$$\hbar_\mu=\begin{cases}0, & y_iX_iw>1;\\ \left(1-y_iX_iw\right)-\frac{\mu}{2}\|X_i\|_\infty, & y_iX_iw<1-\mu;\\ \frac{\left(1-y_iX_iw\right)^2}{2\mu\|X_i\|_\infty}, & \text{else.}\end{cases} \quad (7)$$

Fig.1 plots the hinge loss $\hbar$ and smoothed hinge loss $\hbar_\mu$ with different μ. The figure indicates that a larger μ induces a more smooth $\hbar_\mu$ with larger approximation error. The following theorem shows the theoretical bound of the approximation error.

Theorem 1. *The hinge loss $\hbar$ is bounded by its smooth approximation $\hbar_\mu$, and the approximation error is completely controlled by the smooth parameter μ. For any w, we have*

$$\hbar_\mu\le\hbar\le\hbar_\mu+\frac{\mu}{2}\|X_i\|_\infty. \quad (8)$$

According to Eq.6 and Eq.7, the gradient of $\hbar_\mu$ for the i^{th} sample is calculated as:

$$\frac{\partial\hbar_\mu}{\partial w}=\left\{\begin{array}{ll}0, & u_i=0;\\ -\left(y_iX_i\right)^T, & u_i=1;\\ \frac{-\left(y_iX_i\right)^T\cdot\left(1-y_iX_iw\right)}{\mu\|X_i\|_\infty}, & u_i=\frac{1-y_iX_iw}{\mu\|X_i\|_\infty}.\end{array}\right\}$$
$$=-\left(y_iX_i\right)^Tu_i. \quad (9)$$

In NESVM, the gradient of $L(y_iX_i,w)$ is used to determine the descent direction. Thus, the gradient of the sum of the smoothed hinge losses is given by

$$\frac{\partial L(y_iX_i,w)}{\partial w}=\frac{\partial\sum_{i=1}^{n}\hbar_\mu\left(y_iX_i,w\right)}{\partial w}=-\left(YX\right)^Tu.$$

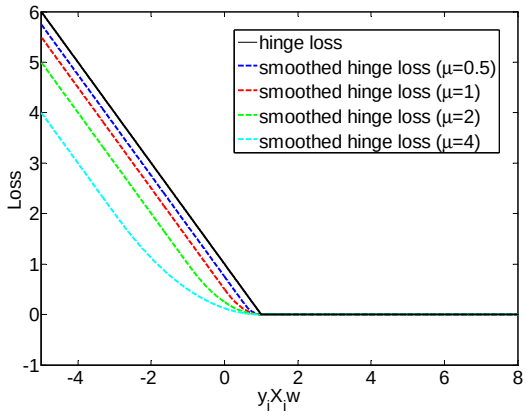

Figure 1. Hinge loss and smoothed hinge loss

In NESVM, the Lipschitz constant of $L(y_iX_i,w)$ is used to determine the step size of each iteration.

Definition 1. *Given function $f(x)$, for arbitrary x^1 and x^2, Lipschitz constant L satisfies*

$$\|\nabla f(x^1)-\nabla f(x^2)\|_2\le L\|x^1-x^2\|_2. \quad (10)$$

Thus the Lipschitz constant of $\hbar_\mu$ can be calculated from

$$\max\frac{\left\|\frac{\partial\hbar_\mu}{\partial w^1}-\frac{\partial\hbar_\mu}{\partial w^2}\right\|_2}{\left\|w^1-w^2\right\|_2}\le L_{\hbar_\mu}. \quad (11)$$

According to Eq.9, we have

$$\frac{\partial\hbar_\mu}{\partial w^1}-\frac{\partial\hbar_\mu}{\partial w^2}=\begin{cases}0, & y_iX_iw>1 \text{ or } <1-\mu;\\ \frac{X_i^TX_i\left(w^1-w^2\right)}{\mu\|X_i\|_\infty}, & \text{else.}\end{cases} \quad (12)$$

Thus,

$$\max\frac{\left\|X_i^TX_i\left(w^1-w^2\right)\right\|_2}{\mu\|X_i\|_\infty\left\|w^1-w^2\right\|_2}\le\frac{\left\|X_i^TX_i\right\|_2}{\mu\|X_i\|_\infty}=L_{\hbar_\mu}. \quad (13)$$

Hence the Lipschitz constant of $L(y_iX_i,w)$ (denoted as L_μ) is calculated as

$$\sum_i L_{\hbar_\mu}\le n\max_i L_{\hbar_\mu}=\frac{n}{\mu}\max_i\frac{\left\|X_i^TX_i\right\|_2}{\|X_i\|_\infty}=L_\mu. \quad (14)$$

B. Smooth the ℓ_1-norm

In LP-SVM, the regularizer is defined by the sum of all ℓ_1-norm $\ell(w_i)=|w_i|$, i.e., $R(w)=\sum_{i=1}^{p}\ell(w_i)$. The minimization of $R(w)$ can be equivalently replaced by the following saddle point function,

$$\min_{w\in\mathbb{R}^p}\sum_{i=1}^{p}\ell(w_i)=\min_{w\in\mathbb{R}^p}\max_{u\in\mathcal{Q}}\langle w,u\rangle, \quad (15)$$
$$\mathcal{Q}=\left\{u:-1\le u_i\le 1,u\in\mathbb{R}^p\right\}.$$

The above saddle point function can be smoothed by subtracting a prox-function $d_1(u)$. In this paper, we choose the

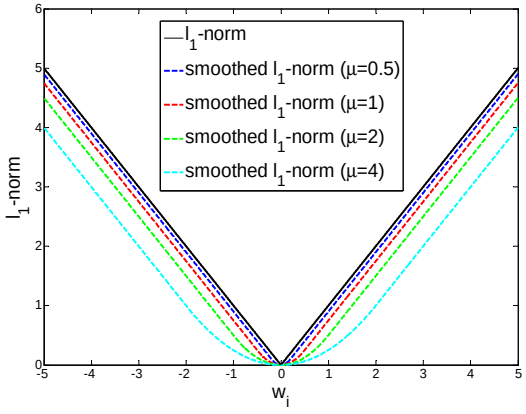

Figure 2. ℓ_1-norm and smoothed ℓ_1-norm

prox-function $d_1(u) = (1/2)\|u\|_2^2$ and thus the smoothed ℓ_1-norm ℓ_μ can be written as,

$$\ell_\mu(w_i) = \max_{u\in\mathcal{Q}} \langle w_i, u_i\rangle - \frac{\mu}{2}u_i^2. \tag{16}$$

Since $d_1(u)$ is strongly convex, u_i can be achieved by setting the gradient of the objective function in Eq.16 as zero and then projecting u_i on $\mathcal{Q}$, i.e.,

$$u_i = \text{median}\left\{\frac{w_i}{\mu}, -1, 1\right\}, \tag{17}$$

where u can also be explained as the result of a soft thresholding of w. Therefore, the smoothed ℓ_1-norm ℓ_μ is a piece-wise approximation of ℓ, i.e.,

$$\ell_\mu = \begin{cases} -w_i - \frac{\mu}{2}, & w_i < -\mu; \\ w_i - \frac{\mu}{2}, & w_i > \mu; \\ \frac{w_i^2}{2\mu}, & \text{else.} \end{cases}$$

Fig.2 plots the ℓ_1-norm ℓ and the smoothed ℓ_1-norm ℓ_μ with different μ. It shows that a larger μ induces a more smooth $\hbar_\mu$ with larger approximation error. The following theorem shows the theoretical bound of the approximation error.

Theorem 2. *The ℓ_1-norm ℓ is bounded by its smooth approximation ℓ_μ, and the approximation error is completely controlled by the smooth parameter μ. For any w, we have*

$$\ell_\mu \le \ell \le \ell_\mu + \frac{\mu}{2}. \tag{18}$$

In NESVM, the gradient of $R(w)$ is used to determine the descent direction. Thus, the gradient of the sum of the smoothed ℓ_1-norm ℓ_μ is

$$\frac{\partial \sum_{i=1}^p \ell_\mu(w_i)}{\partial w} = u. \tag{19}$$

In NESVM, the Lipschitz constant of $R(w)$ is used to determine the step size of each iteration. According to the definition of Lipschitz constant and the second order derivative of ℓ_μ is given by

$$\frac{\partial^2 \ell_\mu(w_i)}{\partial w_i^2} = \frac{1}{\mu}, \tag{20}$$

the Lipschitz constant of the sum of smoothed ℓ_1-norm is given by

$$L_\mu = \max_i \left\{\left|\frac{\partial^2 \ell_\mu(w_i)}{\partial w_i^2}\right|\right\} = \frac{1}{\mu}. \tag{21}$$

C. Nesterov's method for SVM

We apply Nesterov's method [21] to minimize the smoothed primal SVM $F_\mu(w)$. It is a gradient method with the proved optimal convergence rate $\mathcal{O}(1/k^2)$. In its k^{th} iteration round, two auxiliary optimizations are constructed and their solutions are used to build the SVM solution at the same iteration round. We use w^k, y^k and z^k to represent the solutions of SVM and its two auxiliary optimizations at the k^{th} iteration round, respectively. The Lipschitz constant of $F_\mu(w)$ is L_μ and the two auxiliary optimizations are,

$$\min_{y\in\mathbb{R}^p} \langle \nabla F_\mu(w^k), y - w^k\rangle + \frac{L_\mu}{2}\|y - w^k\|_2^2,$$

$$\min_{z\in\mathbb{R}^p} \frac{L_\mu}{\sigma_2} d_2(z) + \sum_{i=0}^{k} \frac{i+1}{2}\left[F_\mu(w^i) + \langle \nabla F_\mu(w^i), z - w^i\rangle\right].$$

We choose the prox-function $d_2(z) = \|z - w^\star\|_2^2/2$ whose strong convexity parameter is σ_2, where $w^\star$ is the prox-center and $\sigma_2 = 1$. The $w^\star$ is usually selected as a guess solution of w.

By directly setting the gradients of the two objective functions in the auxiliary optimizations as zeros, we can obtain y^k and z^k respectively,

$$y^k = w^k - \frac{1}{L_\mu}\nabla F_\mu(w^k), \tag{22}$$

$$z^k = w^\star - \frac{\sigma_2}{L_\mu}\sum_{i=0}^{k}\frac{i+1}{2}\nabla F_\mu(w^i). \tag{23}$$

We have the following interpretation of the above results. The y^k is a solution of the standard gradient descent with step size $1/L_\mu$ at the k^{th} iteration round. The z^k is a solution of a gradient descent step that starts from the guess solution $w^\star$ and proceeds along a direction determined by the weighted sum of negative gradients in all previous iteration rounds. The weights of gradients at later iteration rounds are larger than those at earlier iteration rounds. Therefore, y^k and z^k encode the current gradient and historical gradients. In NESVM, their weighted sum determines the SVM solution after the k^{th} iteration round,

$$w^{k+1} = \frac{2}{k+3}z^k + \frac{k+1}{k+3}y^k. \tag{24}$$

Let ψ_k be the optimal objective value of the second auxiliary optimization, according to [21], we arrive at the following theorem.

Algorithm 1 NESVM

Input: YX, w^0, $w^\star$, C, μ and ϵ
Output: weight vector w
Initialize: $k = 0$
repeat
  Step 1: Compute dual variable u
  Step 2: Compute gradient $\nabla F_\mu(w^k)$
  Step 3: Compute y^k and z^k using Eq.22 and Eq.23
  Step 4: Update SVM solution w^{k+1} using Eq.24
  Step 5: $k = k + 1$
until $|F_\mu(w^{k+1}) - F_\mu(w^k)| < \epsilon$
return $w = w^{k+1}$.

Theorem 3. *For any k and the corresponding y^k, z^k and w^{k+1} defined by Eq.22, Eq.23 and Eq.24, respectively, we have*

$$\frac{(k+1)(k+2)}{4} F_\mu\left(y^k\right) \leq \psi_k. \tag{25}$$

Theorem 3 is a direct result of Lemma 2 in [21] and it will be applied to analyze the convergence rate of NESVM.

A small smooth parameter μ can improve the accuracy of the smooth approximation. A better guess solution w^0 that is close to the real one can improve the convergence rate and reduce the training time.

Algorithm 1 details the procedure of NESVM. In particular, the input parameters are the matrix YX, the initial solution w^0, the guess solution $w^\star$, the parameter C, the smooth parameter μ and the tolerance of termination criterion ϵ. In each iteration round, the dual variable u in smooth parts is first computed, then the gradient $\nabla F_\mu(w)$ is calculated from u, y^k and z^k are calculated from the gradient, and finally w^{k+1} is updated at the end of the iteration round. NESVM conducts the above procedure iteratively until the convergence of $F_\mu(w)$.

NESVM contains no expensive computations, e.g., line search and Hessian matrix calculation. The most computational costs are two matrix-vector multiplications in Steps 1 and 2, i.e., $(YX)w$ and $(YX)^T u$. Since most elements of u are 0 or 1 and the proportion of these elements will rapidly increase with the decreasing of μ, the computation of $(YX)^T u$ can be further simplified. In addition, this simplification indicates that the gradient of each iteration round is completely determined by support vectors in the current iteration round. These support vectors correspond to the nonzero elements in u.

The above algorithm can be conveniently extended to nonlinear kernels by replacing the data matrix X with $K(X, X)Y$, where $K(X, X)$ is the kernel matrix, and replacing the penalty $\|w\|_2^2$ with $w^T K(X, X) w$.

If a bias b in an SVM classifier is required, let

$$w := [w; b] \quad \text{and} \quad X := [X, e], \tag{26}$$

Since the ℓ_2 norm of b is not penalized in the original SVM problem, we calculate the gradient of b according to $\partial F(w)/\partial b = \partial L(y_i X_i, w)/\partial b$ in Algorithm 1, and the last entry of the output solution w is the bias b.

D. Convergence Analysis

The following theorem shows the convergence rate, the iteration number and the time complexity of NESVM.

Theorem 4. *The convergence rate of NESVM is $\mathcal{O}(1/k^2)$. It requires $\mathcal{O}(1/\sqrt{\epsilon})$ iteration rounds to reach an ϵ accurate solution.*

Proof: Let the optimal solution be w^*. Since $F_\mu(w)$ is a convex function, we have

$$F_\mu(w^*) \geq F_\mu(w^i) + \langle \nabla F_\mu(w^i), w^* - w^i \rangle. \tag{27}$$

Thus,

$$\begin{aligned}
\psi_k &\leq \frac{L_\mu}{\sigma_2} d_2(w^*) + \sum_{i=0}^{k} \frac{i+1}{2} \left[F_\mu(w^i) + \langle \nabla F_\mu(w^i), w^* - w^i \rangle \right] \\
&\leq \frac{L_\mu}{\sigma_2} d_2(w^*) + \sum_{i=0}^{k} \frac{i+1}{2} F_\mu(w^*) \qquad (28) \\
&= \frac{L_\mu}{\sigma_2} d_2(w^*) + \frac{(k+1)(k+2)}{4} F_\mu(w^*). \qquad (29)
\end{aligned}$$

According to Theorem 3, we have

$$\frac{(k+1)(k+2)}{4} F_\mu\left(y^k\right) \leq \psi_k \leq \tag{30}$$

$$\frac{L_\mu}{\sigma_2} d_2(w^*) + \frac{(k+1)(k+2)}{4} F_\mu(w^*). \tag{31}$$

Hence the accuracy at the k^{th} iteration round is

$$F_\mu\left(y^k\right) - F_\mu(w^*) \leq \frac{4 L_\mu d_2(w^*)}{(k+1)(k+2)}. \tag{32}$$

Therefore, NESVM converges at rate $\mathcal{O}(1/k^2)$, and the minimum iteration number to reach an ϵ accurate solution is $\mathcal{O}(1/\sqrt{\epsilon})$. This completes the proof. ■

According to the analysis in Section 2.3, there are only two matrix-vector multiplications in each iteration round of NESVM. Thus, the time complexity of each iteration round is $\mathcal{O}(n)$. According to Theorem 4, we can conclude the time complexity of NESVM is $\mathcal{O}(n/k^2)$.

E. Accelerating NESVM with continuation

The homotopy technique used in lasso [22] and LARS [23] shows the advantages of continuation method in speeding up the optimization and solving large-scale problems. In continuation method, a sequence of optimization problems with deceasing parameter is solved until the preferred value of the parameter is arrived. The solution of each optimization is used as the "warm start" for the next optimization. It has been proved that the convergence rate of each optimization is significantly accelerated by this technique, because only a

Algorithm 2 Homotopy NESVM

Input: YX, w^0, $w^\star$, C, μ^0, ϵ and μ^*.
Output: weight vector w.
Initialize: $t = 0$.
repeat
Step 1: Apply NESVM with $\mu = \mu^t$ and $w^0 = w^t$
Step 2: Update $\mu^t = \mu^0/(t+1)$, L_μ and $t := t+1$
until $\mu^t \leq \mu^*$
return $w = w^t$.

few steps are required to reach the solution if the optimization starts from the "warm start".

In NESVM, a smaller smooth parameter μ is always preferred because it produces more accurate approximation of hinge loss or the ℓ_1 norm. However, a small μ implies a large L_μ according to Eq.14 and Eq.21, which induces a slow convergence rate according to Eq.32. Hence the time cost of NESVM is expensive when small μ is selected.

We apply the continuation method to NESVM and obtain an accelerated algorithm termed "homotopy NESVM" for small μ situation. In homotopy NESVM, a series of smoothed SVM problems with decreasing smooth parameter μ are solved by using NESVM, and the solution of each NESVM is used as the initial solution w^0 of the next NESVM. The algorithm stops when the preferred $\mu = \mu^*$ is arrived. In this paper, homotopy NESVM starts from a large μ^0, and sets the smooth parameter μ at the t^{th} NESVM as

$$\mu^t = \frac{\mu^0}{t+1}. \tag{33}$$

Because the smooth parameter μ used in each NESVM is large and the "warm start" is close to the solution, the computation of each NESVM's solution is cheap. In practice, less accuracy is often allowed for each NESVM, thus more computation can be saved. We show homotopy NESVM in Algorithm 2. Notice the Lipschitz constants in Eq.14 and Eq.21 must be updated as the updating of the smooth parameter μ in Step 2.

III. Examples

In this section, we apply NESVM to three typical SVM models, i.e., classical SVM (C-SVM) [12], linear programming SVM (LP-SVM) [13] and least square (LS-SVM) [14][24]. They share an unified form Eq.3, and have different $R(w)$ and $L(y_iX_i, w)$. In NESVM, the solutions of C-SVM, LP-SVM and LS-SVM are different in calculating the gradient item $\nabla F_\mu(w^k)$ and the Lipschitz constant L_μ.

A. C-SVM

In C-SVM, the regularizer $R(w)$ in Eq.3 is

$$R(w) = \frac{1}{2}\|w\|_2^2 \tag{34}$$

and the loss function $L(y_iX_i, w)$ is the sum of all the hinge losses

$$L(y_iX_i, w) = \sum_{i=1}^{n} \hbar\left(y_iX_i, w\right). \tag{35}$$

Therefore, the gradient $\nabla F_\mu(w^k)$ and the Lipschitz constant L_μ in NESVM are

$$\nabla F_\mu(w^k) = w^k - C\left(YX\right)^T u, \tag{36}$$

$$L_\mu = 1 + \frac{Cn}{\mu}\max_i \frac{\left\|X_i^T X_i\right\|_2}{\|X_i\|_\infty}, \tag{37}$$

where u is the dual variable in the smoothed hinge loss and can be calculated according to Eq.6. Thus, C-SVM can be solved by using Algorithm 1 and Algorithm 2.

B. LP-SVM

In LP-SVM, the regularizer $R(w)$ in Eq.3 is

$$R(w) = \|w\|_1 \tag{38}$$

and the loss function $L(y_iX_i, w)$ is the sum of all the hinge losses

$$L(y_iX_i, w) = \sum_{i=1}^{n} \hbar\left(y_iX_i, w\right). \tag{39}$$

Therefore, the gradient $\nabla F_\mu(w^k)$ and the Lipschitz constant L_μ in NESVM are

$$\nabla F_\mu(w^k) = u - C\left(YX\right)^T v, \tag{40}$$

$$L_\mu = \frac{1}{\mu} + \frac{Cn}{\nu}\max_i \frac{\left\|X_i^T X_i\right\|_2}{\|X_i\|_\infty}. \tag{41}$$

where u and v are the dual variables in the smoothed ℓ_1-norm and the smoothed hinge loss, and they can be calculated according to Eq.17 and Eq.6, respectively. μ and ν are the corresponding smooth parameters. They are both updated according to Eq.33 with different initial values in homotopy NESVM. Thus LP-SVM can be solved by using Algorithm 1 and Algorithm 2.

C. LS-SVM

In LS-SVM, the regularizer $R(w)$ in Eq.3 is

$$R(w) = \frac{1}{2}\|w\|_2^2 \tag{42}$$

and the loss function $L(y_iX_i, w)$ is the sum of all the quadratic hinge losses

$$L(y_iX_i, w) = \sum_{i=1}^{n} \left(1 - y_iX_iw\right)^2. \tag{43}$$

Since both the regularizer and the loss function are smooth, the gradient item $\nabla F(w^k)$ and the Lipschitz constant L in NESVM are directly given by

$$\nabla F(w^k) = w^k - 2C\left(YX\right)^T\left(1 - YXw^k\right), \tag{44}$$

$$L = 1 + 2C\max_i\left\{\|X_i\|_2^2\right\}. \tag{45}$$

Steps 1 in Algorithm 1 is not necessary for LS-SVM, and thus LS-SVM can be solved by using Algorithm 1 and Algorithm 2.

IV. Experiments

In the following experiments, we demonstrate the efficiency and effectiveness of the proposed NESVM by applying it to census income categorization and several computer vision tasks, i.e., indoor/outdoor scene classification, event recognition and scene recognition. We implemented NESVM in C++ and run all the experiments on a 3.0GHz Intel Xeon processor with 32GB of main memory under Windows Vista. We analyzed its scaling behavior and the sensitivity to C and the size of dataset. Moreover, we compared NESVM against four benchmark SVM solvers, i.e., SVM-Perf [1], Pegasos [2], SVM-Light [3] and LIBSVM [4]. The tolerance used in stopping criteria of all the algorithms is set to 10^{-3}. For all experiments, different SVM solvers obtained similar classification accuracies and performed comparably to the results reported in respective publications. Their efficiencies are evaluated by the training time in CPU seconds. All the SVM solvers are tested on 7 different C values, i.e., $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$ for 10 times. We show their mean training time in following analysis. In the fist experiment, we also test the SVM solvers on 6 subsets with different sizes.

Five experiments are exhibited, i.e., census income categorization, indoor scene classification, outdoor classification, event recognition and scene recognition. Linear C-SVM models are adopted in the first experiment to train binary classifies. Nonlinear C-SVM models with the RBF kernel $-\|X_i - X_j\|_2^2/p$ are adopted in the rest experiments, wherein p is the number of features. For multiclass classification tasks, the one-versus-one method was adopted. Pegasos is compared with NESVM in the first experiment, because its code is only available to linear C-SVM. In all the experiments, we set the initial solution $w^0 = \mathbf{0}$, the guess solution $w^\star = \mathbf{0}$, the smooth parameter $\mu = 5$ and the tolerance of termination criterion $\epsilon = 10^{-3}$ in NESVM.

A. Census income categorization

We consider the census income categorization on the Adult dataset from UCI machine learning repository [17]. The Adult contains 123 dimensional census data of 48842 Americans. The samples are separated into two classes according to whether their income exceeds $\$50K/yr$ or not. Table 1 shows the number of training samples and the number of test samples in each subset.

Fig.3 shows the scalability of the five SVM solvers on the different C values. The training time of NESVM and Pegasos is slightly slower than the other SVM solvers for small C and faster than the others for large C. In addition, NESVM and Pegasos are least sensitive to C, because the search of the most violated constraint in SVM-Perf, and the working set selection in SVM-Light and LIBSVM will be evidently slowed when C is augmented. However, the main computations of NESVM and Pegasos are irrelevant to C.

Set ID	1	2	3	4	5	6
Training set	1605	2265	3185	4781	6414	11220
Test set	30956	30296	29376	27780	26147	21341

Table I
SIX SUBSETS IN THE CENSUS INCOME DATASET.

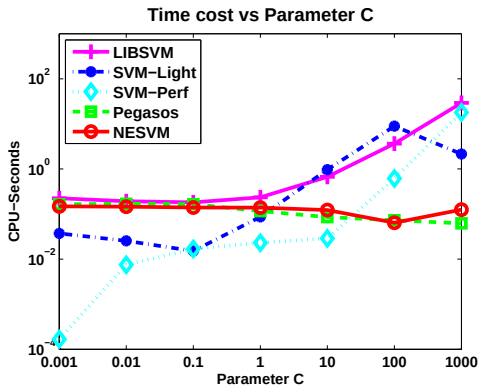

Figure 3. Time cost vs C in census income categorization

Fig.4 shows the scalability of the five SVM solvers on subsets with increasing sizes. NESVM achieves the shortest training time when the number of training samples is less than 5000. Moreover, NESVM and Pegasos are least sensitive to the data size among all the SVM solvers. Pegasos achieves shorter training time when the number of training samples is more than 10000, this is because NESVM is a batch method while Pegasos is an online learning method.

B. Indoor scene classification

We apply NESVM to indoor scene classification on the dataset proposed in [18]. The minimum resolution of all images in the smallest axis are 200 pixels. The sample images are shown in Fig.5. We choose a subset of the dataset by randomly selecting 1000 images from each of the five given groups, i.e., store, home, public spaces, leisure and working place. Gist features of 544 dimensions composed of color, texture and intensity are extracted to represent images. In our experiment, 70% data are randomly selected for training, and the rest for testing.

Fig.6 shows the scalability of four SVM solvers on the different C values. NESVM achieves the shortest training time on different C among all the SVM solvers, because NESVM obtains the optimal convergence rate $\mathcal{O}(1/k^2)$ in its gradient descent. LIBSVM has the most expensive time cost

[1] http://svmlight.joachims.org/svm_perf.html
[2] http://www.cs.huji.ac.il/∼ shais/code/index.html
[3] http://svmlight.joachims.org
[4] http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

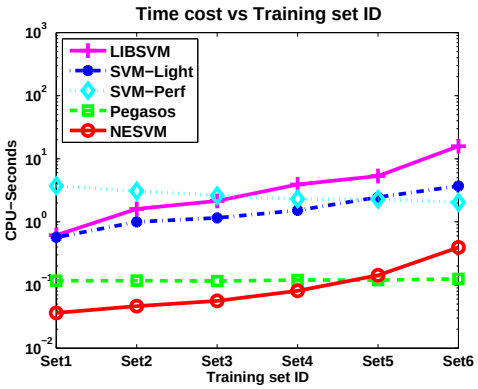

Figure 4. Time cost vs set ID in census income categorization

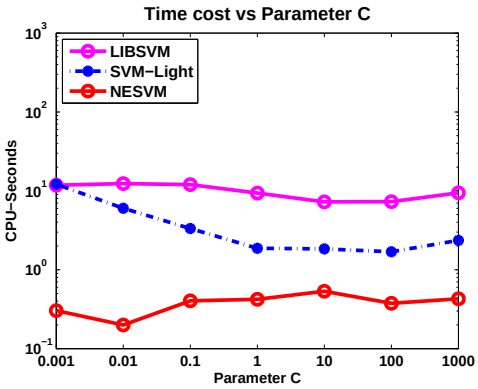

Figure 6. Time cost vs C in indoor scene classification

among all the SVM solvers. In addition, NESVM is least sensitive to C, because the main calculations of NESVM, i.e., the two matrix-vector multiplications, are irrelevant to C. SVM-Light is most sensitive to C. SVM-Perf is not shown in Fig.6 because its training time is much more than the other SVM solvers on all the C (more than 1000 CPU seconds).

C. Outdoor scene classification

We apply NESVM to outdoor scene classification on the dataset proposed in [19]. It contains 13 classes of natural scenes, e.g., highway, inside of cities and office. The sample images are shown in Fig.7. Each class includes 200-400 images, we split the images into 70% training samples and 30% test samples. The average image size is 250×300 pixels. Gist features of 352 dimensions composed of texture and intensity are extracted to represent grayscale images.

Fig.8 shows the scalability of the four SVM solvers on the different C values. NESVM is more efficient than SVM-Light and LIBSVM. It took more than 100 CPU seconds for SVM-Perf on each C, so we do not show SVM-Perf.

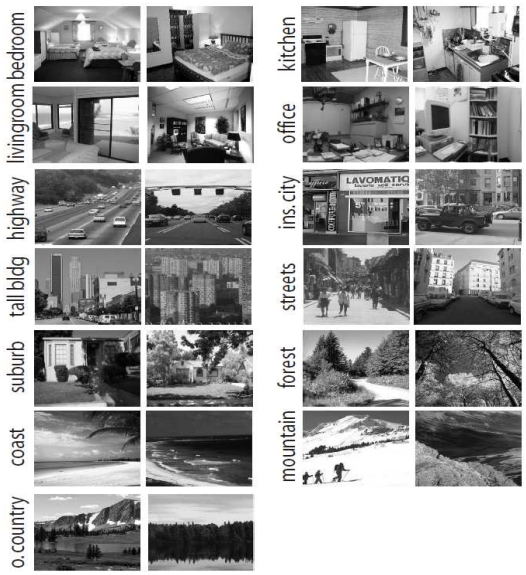

Figure 7. Sample images of outdoor scene dataset

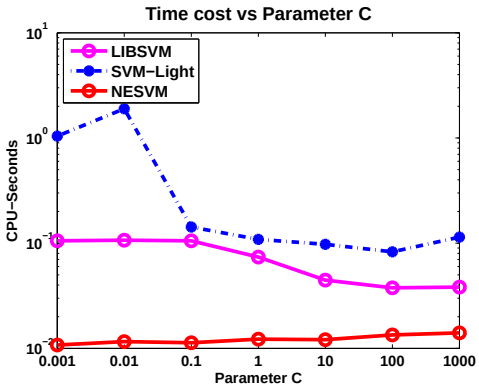

Figure 8. Time cost vs C in outdoor scene classification

D. Event recognition

We apply NESVM to event recognition on the dataset proposed in [20]. It contains 8 classes of sports events, e.g., bocce, croquet and rock climbing. The size of each class varies from 137 to 250. The sample images are shown in Fig.9. Bag of words features of 300 dimensions are extracted according to [20]. We split the dataset into 70% training samples and 30% test samples.

Fig.10 shows the scalability of the four SVM solvers on the different C values. NESVM achieves the shortest training time on different C among all the SVM solvers. SVM-Light and LIBSVM have similar CPU seconds, because both of them are based on SMO. SVM-Perf has the most expensive time cost on different C, because advantages of the cutting-plane algorithm used in SVM-Perf are weakened in the nonlinear kernel situation. NESVM and LIBSVM are

Figure 5. Sample images of indoor scene dataset

Figure 9. Sample images of event dataset

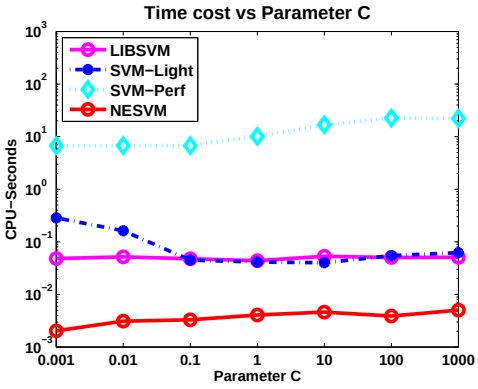

Figure 10. Time cost vs C in event recognition

less sensitive to C than SVM-Perf and SVM-Light.

E. Scene recognition

We apply NESVM to scene recognition on the dataset proposed in [25]. It contains 6 classes of images, i.e., event, program, scene, people, objects and graphics. We randomly select 10000 samples from the scene class and 10000 samples from the other classes and obtain a dataset with 20000 samples. Bag of words features of 500 dimensions are extracted according to [25]. We split the dataset into 50% training samples and 50% test samples.

Fig.11 shows the scalability of the four SVM solvers on the different C values. NESVM achieves the shortest training time on different C among all the SVM solvers. The training time of SVM-Light and LIBSVM similarly increase as the augment of C, because both of them are based on SMO. NESVM and SVM-Perf are less sensitive to C than

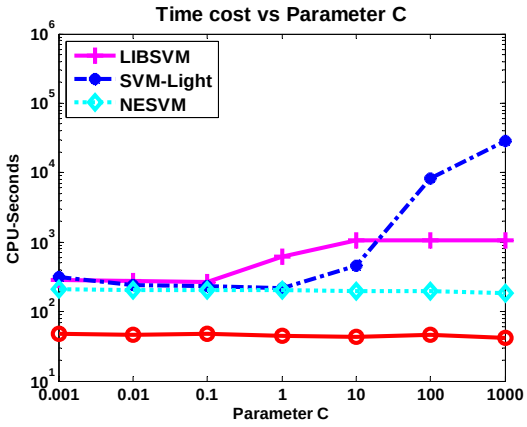

Figure 11. Time cost vs C in scene recognition

LIBSVM and SVM-Light in this binary classification.

V. Conclusion

This paper presented NESVM to solve the primal SVMs, e.g., classical SVM, linear programming SVM and least square SVM, with the optimal convergence rate $\mathcal{O}(1/k^2)$ and a linear time complexity. Both linear and nonlinear kernels can be easily applied to NESVM. In each iteration round of NESVM, two auxiliary optimizations are constructed and a weighted sum of their solutions are adopted as the current SVM solution, in which the current gradient and the historical gradients are combined to determine the descent direction. The step size is automatically determined by the Lipschitz constant of the objective. Two matrix-vector multiplications are required in each iteration round.

We have proposed an accelerated NESVM, i.e., homotopy NESVM, to improve the efficiency of NESVM when accurate approximation of hinge loss or the ℓ_1 norm is required. Homotopy NESVM solves a series of NESVM with decreasing smooth parameter μ, and the solution of each NESVM is adopted as the "warm start" of the next NESVM. The time cost caused by small μ and the starting point w^0 far from the solution can be significantly saved by using homotopy NESVM.

The experiments on various applications indicate that

NESVM achieves the competitive efficiency compared against four popular SVM solvers, i.e., SVM-Perf, Pegasos, SVM-Light and LIBSVM, and it is insensitive to C and the size of dataset. NESVM can be further studied in many areas. For example, it can be sophisticatedly refined to handle sparse features in document classification. Its efficiency can be further improved by introducing the parallel computation. Because the gradient of the smoothed hinge loss and the smoothed ℓ_1 norm is already obtained, NESVM can be further accelerated by extending it to online learning or stochastic gradient algorithms. These will be studied in our future work.

ACKNOWLEDGEMENTS

This research is supported in part by the National Natural Science Foundation of China (NSFC) under award 60828005, in part by the National 973 Program of China under award 2009CB326203, in part by the US National Science Foundation (NSF) under grant CCF-0905337, and in part by the Open Project Program of the State Key Lab of CAD&CG (Grant No. A1006), Zhejiang University.

REFERENCES

[1] T. Joachims, "Training linear svms in linear time," in *The 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006, pp. 217–226.

[2] T. Joachims, T. Finley, and C.-N. Yu, "Cutting-plane training of structural svms," *Machine Learning*, vol. 77, no. 1, pp. 27–59, 2009.

[3] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for svm," in *The 24th Annual International Conference on Machine Learning (ICML)*, 2007, pp. 807–814.

[4] S. Martin, "Training support vector machines using gilbert's algorithm," in *The 5th IEEE International Conference on Data Mining (ICDM)*, 2005, pp. 306–313.

[5] J. Kujala, T. Aho, and T. Elomaa, "A walk from 2-norm svm to 1-norm svm," in *The 9th IEEE International Conference on Data Mining (ICDM)*, 2009, pp. 836–841.

[6] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in kernel methods: support vector learning*, 1999, pp. 185–208.

[7] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[8] K. Morik, P. Brockhausen, and T. Joachims, "Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring," in *The 16th International Conference on Machine Learning (ICML)*, 1999, pp. 268–277.

[9] C.-N. J. Yu and T. Joachims, "Training structural svms with kernels using sampled cuts," in *The 14th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2008, pp. 794–802.

[10] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.

[11] Y.-J. Lee and O. L. Mangasarian, "SSVM: A smooth support vector machine," *Computational Optimization and Applications*, vol. 20, pp. 5–22, 2001.

[12] V. N. Vapnik, *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[13] K. P. Bennett and O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods and Software*, vol. 1, no. 1, pp. 23–34, 1992.

[14] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[15] J. Wang and J. Ye, "An accelerated gradient method for trace norm minimization," in *The 26th International Conference on Machine Learning (ICML)*, 2009.

[16] J. Liu, J. Chen, , and J. Ye, "Large-scale sparse logistic regression," in *The 15th ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, 2009, pp. 547–556.

[17] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[18] A. Quattoni and A.Torralba, "Recognizing indoor scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[19] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 524–531.

[20] L.-J. Li and L. Fei-Fei, "What, where and who? classifying event by scene and object recognition," in *The 10th IEEE International Conference on Computer Vision (ICCV)*, 2007.

[21] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, 2005.

[22] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[23] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2002.

[24] J. Ye and T. Xiong, "Svm versus least squares svm," in *The 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007, pp. 640–647.

[25] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "Nus-wide: A real-world web image database from national university of singapore," in *ACM International Conference on Image and Video Retrieval (CIVR)*, 2009.