

Fast Implementation of ℓ_1 Regularized Learning Algorithms Using Gradient Descent Methods*

Yunpeng Cai[†] Yijun Sun[‡] Yubo Cheng[†] Jian Li[†] Steve Goodison[§]

Abstract

With the advent of high-throughput technologies, ℓ_1 regularized learning algorithms have attracted much attention recently. Dozens of algorithms have been proposed for fast implementation, using various advanced optimization techniques. In this paper, we demonstrate that ℓ_1 regularized learning problems can be easily solved by using gradient-descent techniques. The basic idea is to transform a convex optimization problem with a non-differentiable objective function into an unconstrained non-convex problem, upon which, via gradient descent, reaching a globally optimum solution is guaranteed. We present detailed implementation of the algorithm using ℓ_1 regularized logistic regression as a particular application. We conduct large-scale experiments to compare the new approach with other state-of-the-art algorithms on eight medium and large-scale problems. We demonstrate that our algorithm, though simple, performs similarly or even better than other advanced algorithms in terms of computational efficiency and memory usage.

Keyword: ℓ_1 regularized learning, feature selection, sparse solution, gradient descent

1 Introduction

High-throughput technologies now routinely produce datasets with unprecedented number of features representing each data sample. ℓ_1 regularized learning, due to its ability to produce sparse solutions, has attracted much attention in the last decade. Examples of applications, where this strategy has been shown to be successful, include LASSO [31], generalized LASSO [27], ℓ_1 -SVM [32] and ℓ_1 regularized logistic regression [20].

Despite its attractive properties, the fast implementation of ℓ_1 regularized algorithms for high-dimensional data has long been considered a difficult computational problem since the so-obtained objective function is non-differentiable. Generic methods for non-differentiable convex problems, such as ellipsoid or sub-gradient methods, are typically very slow. In recent years, dozens of algorithms have been developed to deal with medium and large-scale problems, using various advanced optimization techniques. The most commonly used approach is to transform the original problem into one with a convex and differentiable objective function and constraints, and then solve it using standard convex optimization techniques or specified techniques. Due to a large number of constraints, there are only a few methods capable of handling large-scale problems, which mainly fall into two categories. The first category is projected gradient-descent methods (e.g., [1], [5] and [28]). The basic idea is to apply gradient descent methods to minimize an objective function as if there were not constraints, but the infeasible solutions along the gradient direction onto the border of feasible sets. The implementation of these algorithms is very simple. The main drawback, however, is that the projection of infeasible solutions may lead to inaccurate estimates of descending directions, and consequently can significantly slow down the convergence, as is shown in this paper. The methods in the second category employ penalty-based techniques such as interior-point methods [16] to deal with the constraints. Although the interior-point based method is computationally very efficient, its implementation is quite complicated and demands expertise in optimization theories. Moreover, it is a second-order method that takes advantage of the sparseness of a Hessian matrix. If the data matrix is not sparse, which is the case in many practical applications such as cancer studies using microarray data, both computational complexity and memory usage increase. Another popular approach converts the general ℓ_1 learning problem into a series of least-square regression problems, and then solves the so-obtained LASSO model iteratively [19, 27, 18]. These methods require an external LASSO solver and, as we show in our experiments, the

*This work is supported in part by Susan Komen Breast Cancer Foundation under grant No. BCTR0707587. Please address all correspondence, including code requests, to: Dr. Yijun Sun (sunyijun@biotech.ufl.edu)

[†]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32610

[‡]Interdisciplinary Center for Biotechnology Research, University of Florida, Gainesville, FL 32610

[§]Cancer Research Institute, M.D. Anderson Cancer Center, Orlando, FL 32827

performance degrades significantly when the number of relevant features becomes excessively large. Other approaches include path-following methods [22, 25], generalized iterative scaling [13, 23], coordinate descent methods [9, 10] and Gauss-Seidel method [29]. There also exist some approaches that modify objective functions to get an approximate solution to ℓ_1 regularized algorithms in order to cope with high-dimensional data (see, for example, [2]).

The main contribution of this paper is that we show that ℓ_1 regularized learning problems can be easily solved by using gradient-descent techniques. We first transform a convex optimization problem with a non-differentiable objective function into an unconstrained problem. We then prove that if the initial point is properly selected, the solution obtained via gradient descent, when the gradient vanishes, is a global minimizer. We present detailed implementation of the algorithm, using ℓ_1 regularized logistic regression as one particular application. Unlike many existing approaches, the implementation is very easy. Moreover, the proposed algorithm is a generic method that can be readily extended to solve other ℓ_1 regularized learning problems with minor modifications. Although this paper mainly focuses on batch learning, the proposed method can be readily modified to perform online learning. We present some numerical experiments to compare the new approach with other state-of-the-art algorithms on eight medium and large-scale problems. We demonstrate that our algorithm, though simple, performs better than other state-of-the-art algorithms in terms of computational efficiency and memory usage. For example, in one of our simulation studies, it takes only 82 seconds for our algorithm to solve a problem with more than *two million* features, while all other methods fail due to memory limitations.

The rest of the paper is organized as follows. Section 2 describes the main idea of the proposed method. Section 3 presents the detailed implementation of the method. Section 4 presents some numerical experiments to compare the new approach with existing algorithms. Section 5 describes briefly how the proposed method can be used to solve ℓ_1 -SVM where the loss function is non-differentiable and extended for online learning. The paper is concluded in Section 6 with some concluding remarks.

2 Gradient Descent Based Method for Solving ℓ_1 Regularized Problems

This section describes the main idea of the proposed method. Let $\mathcal{D} = \{\mathbf{x}^{(n)}, y_n\}_{n=1}^N$ denote a training dataset, where $\mathbf{x}^{(n)} \in \mathcal{R}^J$ is the n -th pattern and $y_n \in \mathcal{R}$ is the corresponding target value. We seek an optimal solution (\mathbf{w}^*, b^*) to the following ℓ_1 regularized learning

problem:

$$(2.1) \quad \min_{\mathbf{w}, b} f_1(\mathbf{w}, b) = \frac{1}{N} \sum_{n=1}^N L(y_n, \mathbf{w}^T \mathbf{x}^{(n)} + b) + \lambda \|\mathbf{w}\|_1,$$

where $\|\mathbf{w}\|_1 = \sum_j |w_j|$, w_j is the j -th element of \mathbf{w} , $L(\cdot)$ is a loss function, and λ is a regularization parameter that controls the sparseness of the solution. We herein require that $L(\cdot)$ be a convex and twice continuously differentiable function with respect to the second argument. The above formulation encompasses a wide range of learning algorithms, including LASSO [31] and ℓ_1 regularized logistic regression algorithm [20]. If a modified hinge loss is used (see, for example, [24, 4]), (2.1) represents an approximate formulation of ℓ_1 -SVM.

The above formulation has a very appealing property for high-dimensional data analysis. It has been proved in [26] that solving problem (2.1) leads to a globally optimal solution \mathbf{w}^* with at most N non-zero elements. When $N \ll J$, it provides an explicit mechanism to perform feature selection to significantly reduce model complexity. This property, however, comes at a price. Unlike ℓ_2 regularization, $\|\mathbf{w}\|_1$ is a non-differentiable function of \mathbf{w} . The efficient implementation of ℓ_1 regularized formulations poses a computational challenge to the machine learning community. We below demonstrate how a simple gradient descent technique can be used to efficiently solve ℓ_1 regularized learning problems.

Denote $\bar{\mathbf{x}}^{(n)} = [(\mathbf{x}^{(n)})^T, -(\mathbf{x}^{(n)})^T]^T$. Let us consider the following optimization problem:

$$(2.2) \quad \begin{aligned} \min_{\bar{\mathbf{w}}, b} f_2(\bar{\mathbf{w}}, b) &= \frac{1}{N} \sum_{n=1}^N L(y_n, \bar{\mathbf{w}}^T \bar{\mathbf{x}}^{(n)} + b) + \lambda \sum_{i=1}^{2J} \bar{w}_i, \\ \text{s.t. } \bar{\mathbf{w}} &\geq \mathbf{0}. \end{aligned}$$

The following lemma shows that the solution to (2.1) can be recovered from the solution to (2.2).

LEMMA 2.1. *Let $(\bar{\mathbf{w}}^*, b^*)$ be an optimal solution to (2.2) where $\bar{\mathbf{w}}^* = [(\bar{\mathbf{w}}^{*(1)})^T, (\bar{\mathbf{w}}^{*(2)})^T]^T$ and $\bar{\mathbf{w}}^{*(1)}, \bar{\mathbf{w}}^{*(2)} \in \mathcal{R}^J$. Then, $(\bar{\mathbf{w}}^{*(1)} - \bar{\mathbf{w}}^{*(2)}, b^*)$ is an optimal solution to (2.1). Also, if (\mathbf{w}^*, b^*) is an optimal solution to (2.1), then there exist $\bar{\mathbf{w}}^{o(1)}$ and $\bar{\mathbf{w}}^{o(2)}$, so that $\mathbf{w}^* = \bar{\mathbf{w}}^{o(1)} - \bar{\mathbf{w}}^{o(2)}$ and $([(\bar{\mathbf{w}}^{o(1)})^T, (\bar{\mathbf{w}}^{o(2)})^T]^T, b^*)$ is an optimal solution to (2.2).*

The following lemma shows that at least half of the elements of the optimal solution to (2.2) are zero. We will exploit this property in our algorithm implementation.

LEMMA 2.2. *Let $(\bar{\mathbf{w}}^*, b^*)$ be an optimal solution to (2.2) and $\bar{\mathbf{w}}^* = [(\bar{\mathbf{w}}^{*(1)})^T, (\bar{\mathbf{w}}^{*(2)})^T]^T$, then $\forall j \in [J] = [1, \dots, J]$, either $\bar{w}_j^{*(1)}$ or $\bar{w}_j^{*(2)}$ or both are equal to zero.*

The proof of Lemmas (2.1) and (2.2) is straightforward and hence omitted.

The conversion from (2.1) to (2.2) is a standard step that has been previously used in many algorithms (e.g., [28, 5]). Note that (2.2) is a constrained convex optimization problem with a differentiable objective function. In order to use gradient descent, traditional methods usually apply projection or barrier functions to prevent the solution from falling outside the feasible region. In this paper we adopt an essentially different approach, which converts the problem into an unconstrained one.

Let $\bar{w}_j = v_j^2, \forall j \in [2J]$. Then, (2.2) can be rewritten as

$$(2.3) \quad \min_{(\mathbf{v}, b)} f(\mathbf{v}, b) = \frac{1}{N} \sum_{n=1}^N L\left(y_n, \sum_{j=1}^{2J} v_j^2 \bar{x}_j^{(n)} + b\right) + \lambda \sum_{j=1}^{2J} v_j^2.$$

After the above transformation, the objective function of (2.3) is no longer convex, which is usually an undesirable property in optimization, except for some rare cases [6, 7]. In the rest of this section, we show that the transformation is beneficial in the sense that it not only preserves the global convergence property of the original problem, but also enables removal of irrelevant variables.

Taking the derivatives of f with respect to \mathbf{v} and b , respectively, yields

$$(2.4) \quad \begin{aligned} \frac{\partial f}{\partial \mathbf{v}} &= 2 \left(\frac{1}{N} \sum_{n=1}^N \frac{\partial L(y_n, \sum_{j=1}^{2J} v_j^2 \bar{x}_j^{(n)} + b)}{\partial t} \bar{\mathbf{x}}^{(n)} + \lambda \right) \odot \mathbf{v}, \\ \frac{\partial f}{\partial b} &= \frac{1}{N} \sum_{n=1}^N \frac{\partial L(y_n, \sum_{j=1}^{2J} v_j^2 \bar{x}_j^{(n)} + b)}{\partial t}, \end{aligned}$$

where $\partial L(\cdot)/\partial t$ is the derivative of L with respect to the second argument, and \odot is Hadamard operator.

For convenience, we denote $\bar{\mathbf{v}} = [\mathbf{v}^T, b]^T$ and $\mathbf{g} = [(\frac{\partial f}{\partial \mathbf{v}})^T, (\frac{\partial f}{\partial b})^T]$. Let $\bar{\mathbf{v}}^{(k)}$ be the estimates of $\bar{\mathbf{v}}$ in the k -th iteration and $\mathbf{g}^{(k)}$ be the value of \mathbf{g} at $\bar{\mathbf{v}}^{(k)}$. A gradient descent method uses the following updating rule:

$$(2.5) \quad \bar{\mathbf{v}}^{(k+1)} = \bar{\mathbf{v}}^{(k)} - \eta \mathbf{g}^{(k)},$$

where η is determined via a line search.

Since the objective function of (2.3) is not a convex function, a gradient descent method may find a local minimizer or a saddle point. The following theorems show that if the initial point is properly selected, the solution obtained when the gradient vanishes is a global minimizer.

THEOREM 2.1. *Let $f(\mathbf{w}, b)$ be a differentiable convex function of \mathbf{w} and b , where $\mathbf{w} \in \mathcal{R}^J$, $\mathbf{w} \geq \mathbf{0}$, and $b \in \mathcal{R}$. Let $g(\bar{\mathbf{v}}) = f(\mathbf{w}, b)$ where $\mathbf{w} = [w_1, \dots, w_J]^T = [v_1^2, \dots, v_J^2]^T$ and $b = v_{J+1}$. If $\frac{\partial g}{\partial \bar{\mathbf{v}}}|_{\bar{\mathbf{v}}=\bar{\mathbf{v}}^+} = \mathbf{0}$, then $\bar{\mathbf{v}}^+$ is not a local minimizer, but a saddle point or a global minimizer of $g(\bar{\mathbf{v}})$. If the Hessian matrix $H(\bar{\mathbf{v}}^+)$ is positive semi-definite, then $\bar{\mathbf{v}}^+$ is a global minimizer.*

THEOREM 2.2. *For $g(\bar{\mathbf{v}})$ and $\bar{\mathbf{v}}^+$ defined above, if $\bar{\mathbf{v}}^+$ is found through gradient descent with a line search satisfying the following conditions:*

1. *interval condition: a line search splits the section under search into a finite number of intervals,*
2. *descending condition (see the definition below),*
3. *greedy condition (see the definition below);*

and an initial point $\bar{\mathbf{v}}^{(0)}$ satisfying $v_j^{(0)} \neq 0, \forall j \in [J]$, then with probability one, $\bar{\mathbf{v}}^+$ is a global minimizer of $g(\bar{\mathbf{v}})$.

Here we give the definitions of the descending and greedy conditions for a line search:

DEFINITION 2.1. (DESCENDING CONDITION) *Let $g(\bar{\mathbf{v}})$ be an objective function, $\mathbf{g}(\bar{\mathbf{v}})$ be its gradient, $\bar{\mathbf{v}}^{(k)}$ be the solution obtained in the k -th iteration, and $-\mathbf{d}^{(k)}$ be the descending direction, a line search is said to satisfy the descending condition if the chosen step length η satisfies*

$$\mathbf{d}^{(k)T} \mathbf{g}(\bar{\mathbf{v}}^{(k)} - \eta \mathbf{d}^{(k)}) > 0.$$

DEFINITION 2.2. (GREEDY CONDITION) *Given $g(\bar{\mathbf{v}})$ and $\mathbf{g}(\bar{\mathbf{v}})$ defined above, and $\epsilon^{(k)}$ be the length of the intervals at the k -th iteration, a line search is said to satisfy the greedy condition if the step length η chosen satisfies*

$$g(\bar{\mathbf{v}}^{(k)} - \eta \mathbf{d}^{(k)}) \leq g(\bar{\mathbf{v}}^{(k)} - (\eta + \epsilon^{(k)}) \mathbf{d}^{(k)}),$$

or $\bar{\mathbf{v}}^{(k)} - (\eta + \epsilon^{(k)}) \mathbf{d}^{(k)}$ is excluded from the line search; and

$$g(\bar{\mathbf{v}}^{(k)} - \eta \mathbf{d}^{(k)}) \leq g(\bar{\mathbf{v}}^{(k)} - (\eta - \epsilon^{(k)}) \mathbf{d}^{(k)}),$$

or $\bar{\mathbf{v}}^{(k)} - (\eta - \epsilon^{(k)}) \mathbf{d}^{(k)}$ is excluded from the line search.

The interval condition prevents the algorithms from over-exploring along a gradient descent direction. The descending and greedy conditions ensure that a line search approaches a local optimum along the descending direction, but never hits or goes beyond it. With these conditions a gradient descent method can improve its quality step by step and to be immune from misleading gradient information. Golden section search [15] is an example that satisfies the greedy condition and splits the section into finite intervals according to the golden section rule.

We first present the proof of Theorem (2.1).

Proof. For simplicity, we use $\frac{\partial g}{\partial \bar{\mathbf{v}}^*}$ to denote $\frac{\partial g}{\partial \bar{\mathbf{v}}} \big|_{\bar{\mathbf{v}}=\bar{\mathbf{v}}^*}$. Also, we use $\mathbf{A} \succ 0$ and $\mathbf{A} \succeq 0$ to denote that matrix \mathbf{A} is positive definite or semi-definite, respectively.

We examine the properties of the Hessian matrix of $g(\bar{\mathbf{v}})$, denoted as \mathbf{H} . Let $\bar{\mathbf{v}}^+$ be a stationary point of $g(\bar{\mathbf{v}})$ satisfying:

$$(2.6) \quad \frac{\partial g}{\partial \bar{\mathbf{v}}^+} = \left[\frac{\partial f}{\partial w_1^+} 2v_1^+, \dots, \frac{\partial f}{\partial w_J^+} 2v_J^+, \frac{\partial f}{\partial b^+} \right]^T = \mathbf{0},$$

where $\mathbf{w}^+ = [w_1^+, \dots, w_J^+]^T = [(v_1^+)^2, \dots, (v_J^+)^2]^T$ and $b^+ = v_{J+1}^+$.

Note that some elements of $\bar{\mathbf{v}}^+$ may be equal to zero. For simplicity and without loss of generality, assume that the first M elements of $\bar{\mathbf{v}}^+$ belong to $\mathcal{S}_0 = \{v_j^+ : v_j^+ = 0, 1 \leq j \leq J\}$, while the rest $J - M$ elements belong to $\mathcal{S}_{\neq 0} = \{v_j^+ : v_j^+ \neq 0, 1 \leq j \leq J\}$. From Eq. (2.6), we have $\partial f / \partial w_j^+ = 0$ for $v_j^+ \in \mathcal{S}_{\neq 0}$. Then, the Hessian matrix of $g(\bar{\mathbf{v}})$, evaluated at $\bar{\mathbf{v}}^+$, is given by

$$(2.7) \quad \mathbf{H}(\bar{\mathbf{v}}^+) = \begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{pmatrix},$$

where

$$(2.8) \quad \mathbf{A}_1 = \begin{pmatrix} 2 \frac{\partial f}{\partial w_1^+} & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 2 \frac{\partial f}{\partial w_M^+} \end{pmatrix},$$

$$\mathbf{A}_2 = \begin{pmatrix} \mathbf{A}_3 & \mathbf{z} \\ \mathbf{z}^T & \frac{\partial^2 f}{\partial b^2} \end{pmatrix},$$

$$\mathbf{A}_3 = \begin{pmatrix} 4v_{M+1}^+ \frac{\partial^2 f}{\partial w_{M+1}^+} & \dots & 4v_{M+1}^+ v_J^+ \frac{\partial^2 f}{\partial w_{M+1}^+ \partial w_J^+} \\ \vdots & & \vdots \\ 4v_{M+1}^+ v_J^+ \frac{\partial^2 f}{\partial w_{M+1}^+ \partial w_J^+} & \dots & 4v_J^{+2} \frac{\partial^2 f}{\partial w_J^2} \end{pmatrix},$$

$$\mathbf{z} = \left[2v_{M+1}^+ \frac{\partial^2 f}{\partial w_{M+1}^+ \partial b^+} \dots 2v_J^+ \frac{\partial^2 f}{\partial w_J^+ \partial b^+} \right]^T.$$

Here we have used the fact that $\partial f / \partial w_j^+ = 0$ for $v_j^+ \in \mathcal{S}_{\neq 0}$. Since $f(\mathbf{w}, b)$ is a convex function of \mathbf{w} and b , we have

$$\mathbf{B} = \begin{pmatrix} \frac{\partial^2 f}{\partial w_{M+1}^2} & \dots & \frac{\partial^2 f}{\partial w_{M+1} \partial w_J} & \frac{\partial^2 f}{\partial w_{M+1} \partial b} \\ \vdots & & \vdots & \vdots \\ \frac{\partial^2 f}{\partial w_{M+1} \partial w_J} & \dots & \frac{\partial^2 f}{\partial w_J^2} & \frac{\partial^2 f}{\partial w_J \partial b} \\ \frac{\partial^2 f}{\partial w_{M+1} \partial b} & \dots & \frac{\partial^2 f}{\partial w_J \partial b} & \frac{\partial^2 f}{\partial b^2} \end{pmatrix} \succeq 0.$$

It is easy to prove that

$$\mathbf{C} = \begin{pmatrix} 4v_{M+1}^2 & \dots & 4v_{M+1}v_J & 2v_{M+1} \\ \vdots & & \vdots & \vdots \\ 4v_{M+1}v_J & \dots & 4v_J^2 & 2v_J \\ 2v_{M+1} & \dots & 2v_J & 1 \end{pmatrix} \succeq 0.$$

Hence, by Schur product theorem [14], $\mathbf{A}_2 = (\mathbf{B} \odot \mathbf{C}) \big|_{\bar{\mathbf{v}}=\bar{\mathbf{v}}^+}$ is a positive semi-definite matrix. It follows that $\mathbf{H}(\bar{\mathbf{v}}^+) \succeq 0$ if and only if $\mathbf{A}_1 \succeq 0$.

If $\mathbf{H}(\bar{\mathbf{v}}^+)$ is not positive semi-definite, then $\bar{\mathbf{v}}^+$ is a saddle point (Note that it cannot be a maximizer because $g(\bar{\mathbf{v}})$ is convex with respect to v_{J+1} and $\mathbf{H}(\bar{\mathbf{v}}^+)$ cannot be negative semi-definite). If $\mathbf{H}(\bar{\mathbf{v}}^+) \succeq 0$, $\bar{\mathbf{v}}^+$ can be either a saddle point, or a local or global minimizer. We now prove that if $\mathbf{H}(\bar{\mathbf{v}}^+) \succeq 0$, $\bar{\mathbf{v}}^+$ must be a global minimizer.

Let us first consider the following optimization problem:

$$(2.9) \quad \min f(\mathbf{w}, b), \quad \text{subject to } \mathbf{w} \geq \mathbf{0}.$$

Since both the objective function and constraints are convex, the KKT conditions are the sufficient conditions of a global optimal solution. It can be shown that (\mathbf{w}^+, b^+) is a global minimizer of $f(\mathbf{w}, b)$, if for all $j \in \llbracket J \rrbracket$ the following KKT conditions hold simultaneously:

1. $\partial f / \partial b^+ = 0$,
2. $\partial f / \partial w_j^+ = 0$, or $w_j^+ = 0$ and $\partial f / \partial w_j^+ \geq 0$.

Since $\bar{\mathbf{v}}^+$ is a stationary point, by Eq. (2.6),

$$\forall i \in \{i : v_i^+ \in \mathcal{S}_{\neq 0}\}, \partial f / \partial w_i^+ = 0 \text{ and } \partial f / \partial b \big|_{b=v_{J+1}^+} = 0.$$

Moreover, $\mathbf{H}(\bar{\mathbf{v}}^+) \succeq 0$ implies that $\mathbf{A}_1 \succeq 0$. Since \mathbf{A}_1 is a diagonal matrix, it holds that

$$\partial f / \partial w_i^+ \geq 0, \forall i \in \{i : v_i^+ \in \mathcal{S}_{=0}\}.$$

Hence, by the KKT conditions, $(\mathbf{w}^+, v_{J+1}^+)$ and $\bar{\mathbf{v}}^+$ are a global minimizer of $f(\mathbf{w}, b)$ and $g(\bar{\mathbf{v}})$, respectively.

Next, we prove that if the stationary point $\bar{\mathbf{v}}^+$ is found via gradient descent with an initial point $\bar{\mathbf{v}}^{(0)}$ satisfying $v_j^{(0)} \neq 0, 1 \leq j \leq J$, then $\bar{\mathbf{v}}^+$ is a global minimizer of $g(\bar{\mathbf{v}})$ with probability one.

Proof. [Theorem 2.2] Suppose that $\partial g / \partial \bar{\mathbf{v}}^* = \mathbf{0}$ and $\bar{\mathbf{v}}^*$ is a saddle point. Again, we assume that the first M elements of $\bar{\mathbf{v}}^*$ belong to \mathcal{S}_0 , while the following $J - M$ elements belong to $\mathcal{S}_{\neq 0}$. There exists an element $j \in \mathcal{S}_0$ so that $\partial f / \partial w_j^* < 0$ (otherwise $\mathbf{H}(\bar{\mathbf{v}}^*) \succeq 0$ and $\bar{\mathbf{v}}^*$ is a global minimizer). Due to the continuity, there exists $\xi > 0$, such that $\partial f / \partial w_j < 0$ for every

$w_j \in \mathcal{C} = \{w : |w - w_j^*| < \xi\}$. It follows that $\partial g / \partial v_j = 2v_j(\partial f / \partial w_j) < 0$ for $v_j = \sqrt{w_j}$, and $\partial g / \partial v_j > 0$ for $v_j = -\sqrt{w_j}$. That is, a gradient descent method given by $v_j^{(k+1)} \leftarrow v_j^{(k)} - \eta(\partial g / \partial v_j^{(k)})$, drives the solution out of the neighborhood of a saddle point except when (1) the component $v_j^{(k)}$ is set to *exactly* zero, or (2) $v_j^{(k)}$ is outside \mathcal{C} and a line search hits $\bar{\mathbf{v}}^*$ exactly. The latter event cannot happen since gradient $\mathbf{g}(\bar{\mathbf{v}}^*)$ equals to zero at $\bar{\mathbf{v}}^*$, and thus the descending condition does not hold (see Definition 2.1). Instead, a line search will find a solution around $\bar{\mathbf{v}}^*$, and in the subsequent steps the solution will move away from $\bar{\mathbf{v}}^*$. On the contrary, if $\bar{\mathbf{v}}^*$ is a global optimal, $\bar{\mathbf{v}}^{(k+1)}$ will approach it continuously with improved solution quality.

We go on to prove that if $v_j^{(0)} \neq 0$, $v_j^{(k)}$ will be set to exactly zero at a non-stationary point with a zero probability. Let $\bar{\mathbf{v}}^{(k)}$ be the solution obtained in the k -th iteration, $-\mathbf{d}^{(k)}$ be the descending direction, $\bar{\mathbf{v}}^{(k)-} = \bar{\mathbf{v}}^{(k)} - \eta\mathbf{d}^{(k)}$ be a point in the line-search path at which some elements are zeros, and $\mathbf{g}^{(k)-}$ be the gradient at $\bar{\mathbf{v}}^{(k)-}$. Since $\bar{\mathbf{v}}^{(k)-}$ is not a stationary point, $\|\mathbf{g}^{(k)-}\| > 0$.

1. If $\mathbf{d}^{(k)T}\mathbf{g}^{(k)-} \leq 0$, the line search will not reach $\bar{\mathbf{v}}^{(k)-}$.
2. On the other hand, If $\mathbf{d}^{(k)T}\mathbf{g}^{(k)-} > 0$, $-\mathbf{d}^{(k)}$ is also a descending direction at $\bar{\mathbf{v}}^{(k)-}$. Without loss of generality, we assume $\|\mathbf{d}^{(k)}\| = 1$. Due to the continuity, there exists a $\xi_1 > 0$ such that for all $\bar{\mathbf{v}} \in \mathcal{C} = \{\bar{\mathbf{v}} \mid \|\bar{\mathbf{v}} - \bar{\mathbf{v}}^{(k)-}\| < \xi_1\}$, $\mathbf{d}^{(k)T}\mathbf{g}(\bar{\mathbf{v}}) > 0$. This means that for any $\alpha \in (0, 1)$, $g(\bar{\mathbf{v}}^{(k)-} - \alpha\xi_1\mathbf{d}^{(k)}) < g(\bar{\mathbf{v}}^{(k)-})$.

- (a) If the chosen interval length $\epsilon^{(k)} < \xi_1$, the line search has at least one candidate solutions which is correspond with an $\alpha > 0$ above, hence it goes pass $\bar{\mathbf{v}}^{(k)-}$ and moves away from it.
- (b) On the other hand, if $\epsilon^{(k)} \geq \xi_1$, the line search will have only one or two candidate solutions within \mathcal{C} . In this case, the line search has no prior knowledge about the region under search, thus it degenerates to *randomly* selecting one or two $\alpha \in (-1, 1)$ and setting $\bar{\mathbf{v}}^{(k+1)} = \bar{\mathbf{v}}^{(k)-} - \alpha\xi_1\mathbf{d}^{(k)}$. Given an arbitrary bounded probability density distribution, the probability that a single point $\alpha = 0$ is chosen, however, is zero.

Moreover, due to the continuity, there also exists a

$\xi_2 > 0$ such that for all $\bar{\mathbf{v}} \in \mathcal{C}_2 = \{\bar{\mathbf{v}} \mid \|\bar{\mathbf{v}} - \bar{\mathbf{v}}^{(k)-}\| < \xi_2\}$, $\mathbf{g}(\bar{\mathbf{v}})^T\mathbf{g}^{(k)-} > 0$. This mean that a gradient-based search starting in \mathcal{C}_2 will go pass $\bar{\mathbf{v}}^{(k)-}$ following case (2a) stated above, hence $\bar{\mathbf{v}}^{(k)-}$ is not a point of attraction and after a few iterations, case (1) and case (2b) either change to case (2a), or change to the case that the descending direction does not drive any element towards zero. This completes the proof that the saddle points cannot be reached with the designated gradient descent method.

The correctness of the proof of Theorem (2.2) is verified experimentally in Section 4. For the above derivations, we can see that the only price we pay to transform a convex optimization problem with a non-differentiable objective function into an unconstraint one is to double the number of variables. Although the resulting objective function is non-convex, we prove that, via gradient descent, reaching a global minimizer is guaranteed. Since our method mainly relies on gradient descent, we hereafter refer it as Direct Gradient Method, or DGM for short.

3 Implementation Details

We present the detailed implementation of DGM, using ℓ_1 regularized logistic regression as one particular application. However, using DGM to solve other ℓ_1 regularized learning problems is straightforward (see Section 5).

The loss function of ℓ_1 regularized logistic regression is given by

$$(3.10) \quad L(y, a) = \log(1 + \exp(-ya)),$$

where label $y \in \{-1, +1\}$. The gradients of f in Eq. (2.3) with respect to \mathbf{v} and b are given by

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{v}} &= 2 \left(\lambda - \frac{1}{N} \sum_{n=1}^N y_n \sigma \left(-y_n (\bar{\mathbf{w}}^T \bar{\mathbf{x}}^{(n)} + b) \right) \bar{\mathbf{x}}^{(n)} \right) \odot \mathbf{v}, \\ \frac{\partial f}{\partial b} &= -\frac{1}{N} \sum_{n=1}^N y_n \sigma \left(-y_n (\bar{\mathbf{w}}^T \bar{\mathbf{x}}^{(n)} + b) \right), \end{aligned}$$

respectively, where $\sigma(\cdot)$ is the sigmoid function. The gradient descent steps in (2.5) is then applied. In each step, we first apply back-tracking line search [21] to obtain an end point $\bar{\mathbf{v}}^{(e)}$ where $f(\bar{\mathbf{v}}^{(e)}) \leq f(\bar{\mathbf{v}}^{(k)})$, then apply golden section line search on the section between $\bar{\mathbf{v}}^{(k)}$ and $\bar{\mathbf{v}}^{(e)}$.

3.1 Hybrid Conjugate Gradient Because a simple gradient descent method is known to zig-zag in some function contours, we use the Fletcher-Reeves conjugate gradient descent method [8] to enhance the performance

of the algorithm:

$$(3.11) \quad \begin{aligned} \bar{\mathbf{v}}^{(k+1)} &= \bar{\mathbf{v}}^{(k)} - \eta \mathbf{d}^{(k)}, \\ \mathbf{d}^{(1)} &= \mathbf{g}^{(1)}, \\ \mathbf{d}^{(k)} &= \mathbf{g}^{(k)} + \beta \mathbf{d}^{(k-1)}, \quad \forall k > 1, \\ \beta &= \frac{\langle \mathbf{g}^{(k)}, \mathbf{g}^{(k)} \rangle}{\langle \mathbf{g}^{(k-1)}, \mathbf{g}^{(k-1)} \rangle}, \end{aligned}$$

where $\mathbf{d}^{(k)}$ is the conjugate gradient, and $\langle \cdot \rangle$ is the inner product. Note that conjugate gradient method does not ensure that the objective function decreases monotonically. Hence, when $\langle \mathbf{g}^{(k)}, \mathbf{d}^{(k)} \rangle \leq 0$, one usually replaces $\mathbf{d}^{(k)}$ with $\mathbf{g}^{(k)}$ as the search direction to ensure that the algorithm always proceeds in a descending direction. In our implementation, we adopt a hybrid gradient descent scheme. Denote $f^{(k)}$ as the objective function obtained in the k -th iteration and $\angle(\mathbf{a}, \mathbf{b})$ the angle between vectors \mathbf{a} and \mathbf{b} . If $(f^{(k)} - f^{(k-1)})/f^{(k)} < \theta_1$ and $\angle(\mathbf{g}^{(k)}, \mathbf{d}^{(k)}) < \theta_2$, we use $-\mathbf{d}^{(k)}$ as the descending direction, and $-\mathbf{g}^{(k)}$ otherwise. In our implementation, we set $\theta_1 = 0.01$ and $\theta_2 = 5/12\pi$. It should be noted that with the descending condition, the global convergence property also holds for conjugate gradient descent.

In Section 2 we stated that the solution $\bar{\mathbf{w}}^*$ has at most $\min(N, J)$ non-zero elements. We exploit this property to speed up the implementation. Note in (2.4) that if $v_j = 0$, then the gradient will be zero on the j -th element, and v_j will remain zero thereafter. Hence, if some elements of \mathbf{v} are extremely small, the corresponding features can be eliminated from further consideration with a negligible impact on the subsequent iterations and the final solution found. In our implementation, the criterion for eliminating small-valued weights is $w_j < 10^{-10} \|\mathbf{w}\|_\infty$, where $\|\mathbf{w}\|_\infty = \max_j \{w_j\}$. From the discussion in the last section, we can see that if the threshold is small enough that it is always far smaller than ξ_1 in any step, the purging will be safe.

The implementation of DGM is very easy. The pseudo-code is given in Algorithm 1.

3.2 Computational Complexity With conjugate gradient descent, in each iteration, the flops needed to compute gradient is $O(NJ)$, and the memory required is $O(N + J)$, where N is the sample size and J the data dimensionality. For comparison, the interior point method for ℓ_1 logistic regression requires $O(N^2J)$ flops if $J > N$, and $O(NJ^2)$ if $N > J$. Hence, the computational complexity of the interior point method is much higher than DGM, though the interior point method, as a second-order method, usually requires fewer iterations.

Algorithm 1: DGM Algorithm

Input : Data $\mathcal{D} = \{(\mathbf{x}^{(n)}, y_n)\}_{n=1}^N \subset \mathcal{R}^J \times \{\pm 1\}$, parameters θ_1, θ_2 , a stopping criterion
Output: \mathbf{w}, b

- 1 Initialization: Set $\mathbf{v}^{(0)} = \mathbf{1}/\sqrt{(2J)}$, $b^{(0)} = 0$, $t = 0$;
- 2 $\bar{\mathbf{v}}^{(0)} = [(\mathbf{v}^{(0)})^T, b]^T$;
- 3 Compute $f^{(0)}$ using Eq. (2.3);
- 4 **repeat**
- 5 $t = t + 1$;
- 6 Compute $\mathbf{g}^{(t)}$ using Eq. (3.11);
- 7 **if** $t > 1$ *and* $\|f^{(t)} - f^{(t-1)}\|/\|f^{(t)}\| < \theta_1$ **then**
- 8 Compute $\mathbf{d}^{(t)}$ using Eq. (3.11);
- 9 **if** $\angle(\mathbf{g}^{(t)}, \mathbf{d}^{(t)}) \geq \theta_2$ **then**
- 10 $\mathbf{d}^{(t)} = \mathbf{g}^{(t)}$;
- 11 **end**
- 12 **end**
- 13 **else**
- 14 $\mathbf{d}^{(t)} = \mathbf{g}^{(t)}$;
- 15 **end**
- 16 Update $\bar{\mathbf{v}}^{(t)} = \bar{\mathbf{v}}^{(t-1)} - \eta^{(t)} \mathbf{d}^{(t)}$, where $\eta^{(t)}$ is determined via line search;
- 17 **if** $\bar{v}_i^{(t)} < 10^{-5} \|\bar{\mathbf{v}}^{(t)}\|_\infty, \forall i \in [2J]$ **then**
- 18 $\bar{v}_i^{(t)} = 0$;
- 19 **end**
- 20 **until** *stopping criterion* ;
- 21 $\bar{\mathbf{w}}^{(1)} = [(\bar{v}_1^{(t)})^2, \dots, (\bar{v}_J^{(t)})^2]^T$;
- 22 $\bar{\mathbf{w}}^{(2)} = [(\bar{v}_{J+1}^{(t)})^2, \dots, (\bar{v}_{2J}^{(t)})^2]^T$;
- 23 $\mathbf{w} = \bar{\mathbf{w}}^{(1)} - \bar{\mathbf{w}}^{(2)}$;
- 24 $b = \bar{v}_{2J+1}^{(t)}$.

4 Numerical Experiments

We present some numerical experiments to compare DGM with five state-of-the-art methods, namely, interior point method [16], logistic LARS [18], ProjectL1 [28], the ℓ_1 ball projection [5] and Orthant-Wise Limited-memory Quasi-Newton (OWLQN) [1]. These methods cover almost all recently proposed methods on general ℓ_1 regularized learning. They have been extensively tested on a wide variety of data sets and compared against dozens of other methods, including GenLASSO [27], iterated scaling [13], BBR [10] and some traditional gradient-based methods. It is reported that these methods are one or two orders of magnitude faster than existing methods. Hence, we believe that the comparison with these methods is sufficient to demonstrate the effectiveness of our newly proposed method.

4.1 Experiment Setup We apply each algorithm to eight data sets using a specified set of λ values. Each algorithm stops when the achieved objective function is within 10^{-6} precision of the optimal solution. The CPU time of each algorithm is then recorded and compared. The time spent on loading data and writing solutions is excluded from the CPU time.

The interior point method solves (2.1) and its dual problem simultaneously, and computes the duality gap that is used as the upper bound of the precision. The maximum of the dual problem is used as the lower bound of the minimum of the primal problem. In contrast, other methods adopt a different stopping criterion, namely, the difference of the objective functions in two consecutive iterations. To make a fair comparison, in our experiments, for every data set and λ value, we first run the interior point method, which stops when the duality gap is within a desired precision. After that, we set the so-obtained objective function of the dual problem in the interior point method as the target value of the other methods. By using this experimental protocol, we verify that the solution obtained by DGM is indeed a global minimizer.

It should be noted that logistic LARS and ℓ_1 ball projection solve a different problem:

$$\min_{\mathbf{w}, b} \frac{1}{N} \sum_{n=1}^N L(y_n, \mathbf{w}^T \mathbf{x}^{(n)} + b) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \leq C.$$

In order to obtain a comparable solution with other algorithms, after running DGM and the interior point method, we compute the 1-norm of the obtained \mathbf{w} and use it as the upper bound C in logistic LARS and ℓ_1 ball projection.

We use the method described in [16] to determine the proper range of λ . The upper bound of a possible λ can be calculated as:

$$\lambda_{\max} = \left\| \frac{|S^-|}{N} \sum_{n \in S^+} \mathbf{x}^{(n)} - \frac{|S^+|}{N} \sum_{n \in S^-} \mathbf{x}^{(n)} \right\|_{\infty},$$

where $S^+ = \{n | y_n = 1\}$ and $S^- = \{n | y_n = -1\}$. We set λ to be uniformly spaced on a logarithmic scale over interval $[0.001\lambda_{\max}, 0.99\lambda_{\max}]$, which covers the range of interest for most practical applications.

The interior point method and OWLQN are coded in C. Logistic LARS is programmed in matlab and C-mex mixed code with all computationally intensive tasks executed in C, and thus is almost as efficient as pure C program. ProjectionL1 is coded in Matlab only. Hence, we develop both C and Matlab versions of DGM for comparison. Since ℓ_1 ball projection does not have a published code, we implement it in

Table 1: Summary of data sets

Data	No. of features	No. of samples
Colon cancer	2000	62
Leukemia	7129	72
Internet Ads.	1430	2359
Prostate cancer	22291	79
TABM77	1145	291
GSE4922	44932	249
Arcene	10000	200
Linear	10000	400

Matlab by strictly following the instructions provided by [5]. The batch-learning version of the algorithm is implemented. The algorithm needs to specify an initial step length, or determine it via back-tracking line search. We implement both line-search and non-line-search versions, tune the initial step length of the non-line-search version for all data sets and all λ within a wide range of $[10^8, 10^{-8}]$, and report the best results, which outperform the line-search one reported in [5].

The first experiment described below is performed on a personal computer with Intel Core 2 T5500 1.66GHz CPU, 512MB memory, and Linux operating system. The second experiment, due to memory requirements, is run on an Intel Xeon 2.83G server with 8GB memory and Window XP operating system, where the system allows a maximum of 2GB memory.

4.2 Experimental Results We first compare the algorithms on eight medium and large-scale data sets with feature dimensionality ranging from 1,000 to 44,932. Among them, *internet ads*, *leukemia* and *colon cancer* gene expression data have already been used in [16, 18]. The *arcene* data is first used in the NIPS feature selection challenge [11], and *prostate cancer* [30], *ETABM77* [3] and *GSE4922* [12] are three cancer microarray data. The *Linear* data is an artificially generated binary classification problem, with each class having 200 samples characterized by 10^4 features. The first 500 features are drawn from two normal distributions $\mathcal{N}(-1, 1)$ and $\mathcal{N}(1, 1)$, depending on class labels. The rest of the features are drawn from the standard normal distribution, thus providing no discriminant information. The summary of the data is given in Table 1.

In practical applications, cross validation is usually performed over all possible λ values to estimate the optimal regularization parameter λ . We apply the six algorithms to eight data sets, and record in Table 2 the total running time summed over seven λ values uniformly spaced on a logarithmic scale over interval $[0.001\lambda_{\max}, 0.99\lambda_{\max}]$. We observe that in seven out of

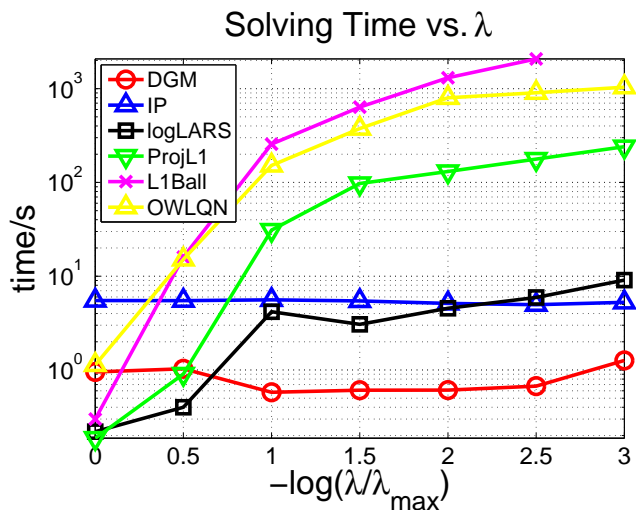


Figure 1: Running time (in seconds) of the six algorithms performed on the *prostate cancer* dataset using different λ values.

eight data sets, the C version of DGM performs the best, and in seven out of eight data sets, the Matlab version of DGM outperforms ProjectionL1. For some data sets (e.g., *arcene* and *prostate cancer*), DGM is one order of magnitude faster than ProjectionL1. ℓ_1 ball projection performs very well on the *Linear* data set, which is consistent with the result reported in [5]. However, in six out of the seven real-world data sets, it performs poorly and cannot reach the targeted precision level in a short time even if we use 10^{-4} as a stop criterion. OWLQN is significantly slower than all other methods on all data sets, because it mainly focuses on reducing the memory usage, though our method is also memory efficient.

Figure 1 depicts the CPU time of the six algorithms performed on the *prostate cancer* data set using different λ values. We can see that when λ is large, logistic LARS, Projection L1 and ℓ_1 ball projection are very fast because the search area is shrunk to a very small region. However, with the decreasing value of λ , the running time of both Projection L1 and ℓ_1 ball projection increases dramatically. In contrast, the computational complexity of both DGM and the interior point method does not change significantly with respect to the regularization parameter. The above observation is consistent over the other seven data sets.

To further demonstrate the scaling property of our method, we compare DGM and the interior point method using a series of artificially generated linear data set with 200 samples, varying the feature dimensionality from 500 to 2.15×10^6 . Figure 2 presents the CPU time

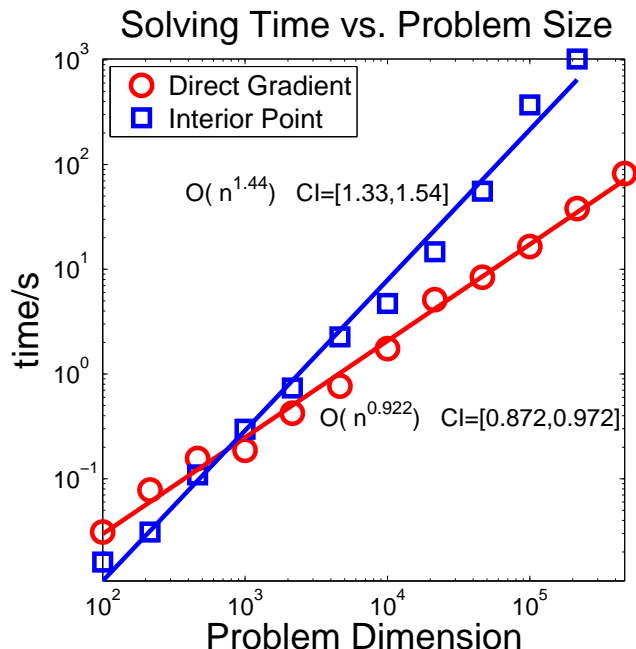


Figure 2: Scalability of DGM and the interior point method performed on data sets with feature dimensionality ranging from $10^{2.66}$ to $10^{6.33}$. The empirical complexity and confidence interval (CI) are also reported.

of DGM and the interior point method as a function of data dimensionality. We observe that the solving time of DGM exhibits a sub-linear growth of $O(n^{0.92})$, while for the interior point method the solving time grows super-linearly ($O(n^{1.44})$) with respect to the sample size, which is consistent with the results reported in [16]. For example, when the data dimension is 10^6 , it takes the interior point method 1017 seconds to solve the problem, while for our method the solving time is only 38 seconds. Moreover, our DGM method uses less memory than the interior point method. For this reason, DGM solves a 2150000-dimensional problem in only 82 seconds using a 2GB memory, while the interior point method fails.

From the above observations, we conclude that DGM, though simple, can achieve competitive computational efficiency compared with the state-of-the-art methods.

5 Extensions of DGM

DGM is a generic method that can be used to solve various ℓ_1 regularized learning problems provided that the loss function is differentiable. The hinge loss used in SVM, however, is a non-differentiable function. We below give a brief discussion on how our method can be used to solve ℓ_1 -SVM. One possible way is to replace the

Table 2: CPU time (in seconds) of the six algorithms performed on the eight data sets. The algorithms stop when the achieved objective function is within 10^{-6} precision of the optimal solution. The results marked with * are obtained by using 10^{-4} as a stop criterion.

Data	Colon	Leukemia	Internet Ads.	Prostate	ETABM77	GSE4922	Arcene	Linear
DGM-C	15	49	226	82	63	313	393	151
Interior Point	33	120	15	384	99	3219	788	1221
Logistic LARS	28	80	1163	274	116	2344	> 24 hr	1866
OWLQN	466	3442	776	32822	641	> 24hr	25543	5861
DGM-Matlab	131	176	665	209	235	959	2237	492
Projection L1	591	1723	491	6783	268	> 24 hr	32194	1821
L1 Ball Proj.	6908*	> 24 hr*	3547	> 24 hr*	24234*	> 24 hr*	> 24 hr*	291

hinge loss with a differentiable function. [4] suggests to use the Huber loss, given by

$$L(y, a) = \begin{cases} 0 & ya > 1 + h, \\ \frac{(1 + h - ya)^2}{4h} & 1 - h \leq ya \leq 1 + h, \\ 1 - ya & ya < 1 - h, \end{cases}$$

where h is a tunable parameter. If h is sufficiently small, SVM using the Huber loss provides the same sparse solution as SVM with the hinge loss [4]. Hence, with minor modifications, DGM described in Section 2 can be directly used to solve ℓ_1 -SVM in the primal domain.

Due to its non-constrained formulation, the proposed method can be readily extended for online learning for applications where both the numbers of features and samples are excessively large [17]. By using the theory of stochastic gradient, the convergence is guaranteed. The experimental results of online DGM is reported elsewhere.

6 Conclusions

In this paper we have proposed a simple yet very efficient method to solve ℓ_1 regularized learning problems. We have conducted large-scale numerical experiments to demonstrate that our method can achieve improved computational efficiency over the state-of-the-art methods. The proposed method can deal with various loss functions such as the least square loss, hinge loss and truncated least square loss, and can be adopted in both batch learning and online learning scenarios. This work provides a new direction for fast implementation of large-scale ℓ_1 regularized learning algorithms.

References

[1] G. Andrew, and J.F. Gao, *Scalable Training of L1-Regularized Log-Linear Models*, Proc. 24th Intl. Conf. Mach. Learn., Corvallis, OR, USA, 2007, pp. 33–40.

[2] S. Balakrishnan, and D. Madigan, *Algorithms for sparse linear classifiers in the massive data setting*, J. Mach. Learn. Res., 9 (2008), pp. 313–337.

[3] M. Buyse, S. Loi, L. van't Veer, G. Viale, M. Delorenzi, A. M. Glas, M. S. d'Assignies, J. Bergh, R. Lidereau, P. Ellis, A. Harris, J. Bogaerts, P. Therasse, A. Floore, M. Amakrane, F. Piette, E. Rutgers, C. Sotiriou, F. Cardoso, and M. J. Piccart, *Validation and clinical utility of a 70-gene prognostic signature for women with node-negative breast cancer*. J. N. Cancer Inst., 98(17) (2006), pp. 1183–1192.

[4] O. Chapelle, *Training a support vector machine in the primal*, Neural Comput., 19 (2007), pp. 1155–1178.

[5] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, *Efficient projections onto the L1-ball for learning in high dimensions*, Proc. 25th Intl. Conf. Mach. Learn., Helsinki, Finland, 2008, pp. 272–279.

[6] Y. G. Evtushenkjo and V. G. Zhadan, *Space-transformation technique: the state of the art*, in Non-linear Optimization and Applications, (1996), pp 101–123.

[7] L. Faybusovich, *Dynamical systems which solve optimization problems with linear constraints*, IMA J. Math. Ctrl. and Info., 8 (1991), pp 135–149;

[8] R. Fletcher, *Practical methods of optimization*, John Wiley, New York, 1997.

[9] J. Friedman, T. Hastie, H. Hoeffling and R. Tibshirani, *Pathwise coordinate optimization*, Ann. Appl. Stat., 1 (2007), pp. 302–332.

[10] A. Genkin, D. Lewis, and D. Madigan, *Large-scale bayesian logistic regression for text categorization*, Technometrics, 49 (2007), pp. 291–304.

[11] I. Guyon, S. R. Gunn, A. Ben-Hur and G. Dror, *Result analysis of the NIPS 2003 feature selection challenge*, 17th Adv. Neu. Info. Proc. Sys., Vancouver, Canada, (2005), pp. 545–552.

[12] A. V. Ivshina, J. George, O. Senko, B. Mow, T. C. Putti, J. Smeds, T. Lindahl, Y. Pawitan, P. Hall, H. Nordgren, J. E. Wong, E. T. Liu, J. Bergh, V. A. Kuznetsov, and L. D. Miller, *Genetic reclassification of histologic grade delineates new clinical subtypes of breast cancer*, Cancer Res., 66(21) (2006), pp. 10292–10301.

- [13] J. Goodman, *Exponential priors for maximum entropy models*, Proc. 42nd Ann. Meet. Asso. Comp. Ling., Barcelona, Spain, 2004, pp. 305–312.
- [14] R. Horn, and C. Johnson, *Matrix analysis*. Cambridge University Press, Cambridge, 1985.
- [15] J. Kiefer, *Sequential minimax search for a maximum*, Proc. 4th Amer Math. Soc., (1953), pp. 502–506.
- [16] K. Koh, S. J. Kim, and S. Boyd, *An interior-point method for large-scale ℓ_1 -regularized logistic regression*, J. Mach. Learn. Res., 8 (2007), pp. 1519–1555.
- [17] J., Langford, L., Li, and T., Zhang, *Sparse Online Learning via Truncated Gradient*, J. Mach. Learn. Res., 10 (2009), pp. 777–801.
- [18] S. I., Lee, H., Lee, P., Abbeel, and A. Y., Ng, *Efficient ℓ_1 regularized logistic regression*, Proc. 21st AAAI Conf. Artif. Intell., Boston, MA, USA, 2006, pp. 1–9.
- [19] J. Lokhorst, *The lasso and generalised linear models*, Tech. Report, Dept. of Stat., Univ. of Adelaide, Australia, 1999.
- [20] A. Y. Ng, *Feature selection, $L1$ vs. $L2$ regularization, and rotational invariance*, Proc. 21st Intl. Conf. Mach. Learn., Banff, Alberta, Canada, 2004, pp. 78–86.
- [21] J. Nocedal, and S. J. Wright, *Numerical optimization*, Springer Verlag, New York, NY, 1999.
- [22] M. Park, and T. Hastie, *$L1$ regularization-path algorithm for generalized linear models*, J. R. Stat. Soc. Ser. B Stat. Meth., 69 (2007), pp. 659–677.
- [23] S. Perkins, and J. Theiler, *Online feature selection using grafting*, Proc. 20th Intl. Conf. Mach. Learn., Washington, DC, USA, 2003, pp. 592–599.
- [24] J. D. Rennie, and N. Srebro, *Fast maximum margin matrix factorization for collaborative prediction*, Proc. 22nd Intl. Conf. Mach. Learn., Bonn, Germany, 2005, pp. 713–719.
- [25] S. Rosset, *Following curved regularized optimization solution paths*, Proc. 17th Adv. Neu. Info. Proc. Sys., Whistler, Canada, 2005, pp. 1153–1160.
- [26] S. Rosset, J. Zhu, and T. Hastie, *Boosting as a regularized path to a maximum margin classifier*, J. Mach. Learn. Res., 5 (2004), pp. 941–973.
- [27] V. Roth, *The generalized LASSO*, IEEE Trans. Neu. Net., 15 (2004), pp. 16–28.
- [28] M. Schmidt, G. Fung, and R. Rosales, *Fast optimization methods for $L1$ regularization: a comparative study and two new approaches*, Proc. 18th Euro. Conf. Mach. Learn., Warsaw, Poland, 2007, pp. 286–297.
- [29] S. Shevade, and S. Keerthi, *A simple and efficient algorithm for gene selection using sparse logistic regression*, Bioinformatics, 19 (2003), pp. 2246–2253.
- [30] A. J. Stephenson, A. Smith, M. W. Kattan, J. Sathagopan, V. E. Reuter, P. T. Scardino and W. L. Gerald, *Integration of gene expression profiling and clinical variables to predict prostate carcinoma recurrence after radical prostatectomy*, Cancer, 104 (2005), pp. 290–298.
- [31] R. Tibshirani, *Regression shrinkage and selection via the LASSO*, J. R. Stat. Soc. Ser. B, 58 (1996), pp. 267–288.
- [32] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, *1-*

norm support vector machines, Proc. 16th Adv. Neu. Info. Proc. Sys., Whistler, Canada, 2002, pp. 49–56.