A New Regularization Path for Logistic Regression via Linearized Bregman

Jianing V. Shi[1], Wotao Yin[2] and Stanley J. Osher[3]

[1] Department of Electrical and Computer Engineering
Rice University, Houston, TX 77005, USA
[2] Department of Computational and Applied Mathematics
Rice University, Houston, TX 77005, USA
[3] Department of Mathematics
UCLA, Los Angeles, CA 90095, USA

E-mail: jianing.shi@rice.edu

Abstract.

Sparse logistic regression is an important linear classifier in statistical learning, providing an attractive route for feature selection. A popular approach is based on minimizing an ℓ_1-regularization term with a regularization parameter λ that affects the solution sparsity. To determine an appropriate value for the regularization parameter, one can apply the grid search method or the Bayesian approach. The grid search method requires constructing a regularization path, by solving a sequence of minimization problems with varying values of the regularization parameter, which is typically time consuming. In this paper, we introduce a fast procedure that generates a new regularization path without tuning the regularization parameter. We first derive the direct Bregman method by replacing the ℓ_1-norm by Bregman divergence, and contrast it with the grid search method. For faster path computation, we further derive the linearized Bregman algorithm, which is algebraically simple and computationally efficient. Finally we demonstrate some empirical results for the linearized Bregman algorithm on benchmark data and study feature selection as an inverse problem. Compared with the grid search method, the linearized Bregman algorithm generates a different regularization path with comparable classification performance, in a much more computationally efficient manner.

AMS classification scheme numbers: 65, 62, 35

Submitted to: *Inverse Problems*

1. Introduction

1.1. Inverse problems and sparsity

Sparsity has emerged as an important model assumption for inverse problems, especially in compressive sensing, machine learning, statistics and related fields. Minimization of the ℓ_1 norm, due to its sparsity promoting property and convexity, has led to a tremendous amount of algorithm development in the past few years. It fits into the general framework of regularized inverse problems, where we promote sparsity on the latent variables,

$$\arg\min \lambda J(u) + H(u), \tag{1}$$

where $J(u)$ is the regularization term, $H(u)$ is the data fidelity term and λ is a regularization parameter. The regularization term can take the form of ℓ_1 regularization, $J(u) = \|u\|_1$. The ℓ_1 regularization promotes sparsity, based on the hypothesis that only a subset of the independent variable u is informative about the dependent variable y. The data fidelity term models the consistency between independent variable u and dependent variable y. Depending on the application and form of dependent variable y, the fidelity term can take different forms. In general, the fidelity term can be written as

$$H(u) = L(Au, y). \tag{2}$$

- Real data
  If y is real data, the fidelity term typically takes the form of quadratic function. Statistical regression belongs to this form. Compressive sensing is a special case, when A is the sensing matrix.
- Categorical data
  If y is categorical data, the fidelity term takes the form of some loss function. The choice of loss function depends on the decision boundary. Such a form arises often in machine learning, where the goal is to perform classification.

1.2. ℓ_1-regularized logistic regression

Despite the fact that our methodology can be generalized to various fidelity terms, we focus in this paper on the ℓ_1-regularized logistic regression [28], a popular linear decoder in the field of machine learning. The inputs are a set of training data $X = [x_1, \cdots, x_m]^\top \in \mathbb{R}^{m\times n}$, where each row of X is a sample and samples of either class are assumed to be independently identically distributed, and class labels $y \in \mathbb{R}^m$ are of $-1/+1$ elements.

We seek a hyperplane $\{x : w^\top x + v = 0\}$ that separates the data belonging to two classes, where $w \in \mathbb{R}^n$ is a set of weights and $v \in \mathbb{R}$ is the intercept. The ℓ_1-regularized logistic regression applies the ℓ_1-penalty on the weights w:

$$\arg\min_{w,v} \ \lambda J(w) + l_{\text{avg}}(w, v), \tag{3}$$

where $J(w) = \|w\|_1$, and $\lambda > 0$ is a regularization parameter. The empirical loss function is

$$l_{\text{avg}}(w, v) = \frac{1}{m} \sum_{i=1}^{m} \theta\big((w^T x_i + v) y_i\big), \tag{4}$$

where θ is the logistic transfer function, $\theta(z) := \log(1 + \exp(-z))$.

It is well-known that ℓ_1 minimization tends to give sparse solutions. The ℓ_1 regularization results in logarithmic sample complexity bounds (number of training samples required to learn a function), making it an effective learner even under an exponential number of irrelevant features [21, 22]. Furthermore, ℓ_1 regularization also has appealing asymptotic sample consistency for feature selection [36].

Various algorithms exist for solving the ℓ_1-regularized logistic regression, including Gl1ce [19], Grafting [25], GenLASSO [26], and SCGIS [13], IRLS-LARS [9, 18], BBR [12], Glmpath [24], interior point method [17], fixed point continuation (FPC) [16], sparse reconstruction by separable approximation (SpaRSA) [31], hybrid iterative shrinkage (HIS) [27], and accelerated block-coordinate relaxation [30].

1.3. Computing the full regularization path

The regularization parameter λ determines the level of sparsity, which is typically unknown *a priori*. In order to determine an appropriate level of sparsity, one may need to generate a regularization path. In many scenarios, the appropriate level of sparsity can refer to the optimal level sparsity where the corresponding solution gives rise to the best generalization performance for classification. We define two types of regularization paths here:

- **Solution-vs-sparsity**
  Solution path where solution is a function of sparsity level.
- **Solution-vs-lambda**
  Solution path where solution is a function of regularization parameter λ.

One ideally wants to obtain the optimal solution-vs-sparsity path, where each point on the path is the solution that achieves the best classification performance with the given sparsity. However, minimizing or confining the solution sparsity leads to combinatorial problems, which are generally very hard to solve. Therefore in the grid search method, one typically approximates the solution-vs-sparsity path using the solution-vs-lambda path. More specifically, one constructs a regularization path by varying the regularization parameter λ and solving a sequence of minimization problems corresponding to each λ. The optimal λ can be determined via cross validation. However, one needs to solve each minimization accurately. It is not difficult to see the grid search method is time costly. This is especially true when the cardinality of the true solution support is large, since usually the smaller λ is, the lesser sparse solution is and the longer it takes for an algorithm to converge. In this paper, we devise a methodology

to efficiently approximate the solution-vs-sparsity path, using the linearized Bregman algorithm.

Some related work along the line of regularization parameter selection can be found in the Bayesian literature. The Bayesian approach provides an alternative, where the regularization parameter is treated as a parameter of the prior distribution in a hierarchical framework. One strategy is to integrate out the prior parameter to obtain the marginal likelihood, which is used in algorithms such as automatic relevance determination [20], relevance vector machine [29]. Such a Bayesian approach was used for ℓ_1-regularization [10], and ℓ_2-regularization [11].

1.4. Our contribution

We propose a new approach for computing the solution-vs-sparsity path, which is much more computationally efficient than the grid search method while achieving comparable solution quality. Our approach is based on the linearized Bregman algorithm, using all of the intermediate solutions to form the path. The linearized Bregman algorithm is based on solving a sequence of minimization subproblems by introducing the Bregman divergence. Each iteration of the algorithm minimizes the sum of certain Bregman divergence of the ℓ_1 norm, the linearization of the loss function, and a proximity term. The minimizer of each subproblem can be obtained in closed form, resulting in an algebraically simple and computationally efficient algorithm. Unlike the grid search method which solves for the regularization path of (3), our new regularization path does not require tuning the regularization parameter λ and solving each subproblem corresponding to each parameter value. However, like the regularization path of (3), our new regularization path is roughly monotonic in solution sparsity. The linearized Bregman algorithm creates a new regularization path, starting from a big λ, which results in highly sparse solution; as the algorithm proceeds, the sparsity of solution increases, as well as the classification performance on training data. Empirical results demonstrate that the generalization performance of the linearized Bregman algorithm is comparable with the grid search method.

2. Prior art for Bregman related algorithms

2.1. Bregman divergence

The Bregman divergence [2], based on a convex functional $J : \mathbb{R}^n \to \mathbb{R}$, is formally defined by

$$D_J^p(u, v) = J(u) - J(v) - \langle u - v, p \rangle, \quad p \in \partial J(v). \tag{5}$$

For a continuously differentiable functional, such as ℓ_2 norm, there exists a unique element p in the subdifferential and consequently a unique Bregman divergence. For a non-differentiable yet convex functional, such as the ℓ_1 norm, $p \in \partial J(v)$ is an element in the subgradient of J at the point v. It is worth noting that the Bregman divergence is not a distance in the usual sense, since in general $D_J^p(u, v) \neq D_J^p(v, u)$, nor is the triangle

inequality satisfied. However, the Bregman divergence measures the closeness between u and v in the sense that $D_J^p(u,v) \geq 0$ and $D_J^p(u,v) \geq D_J^p(w,v)$ for all points w on the line segment connecting u and v. Moreover, Bregman divergence has the following properties:

- If J is convex, $D_J^p(u,v) \geq 0$;
- If J is strictly convex, $D_J^p(u,v) > 0$ for $u \neq v$;
- If J is strongly convex, there exists a constant $\nu > 0$ such that $D_J^p(u,v) \geq \nu\|u-v\|_2^2$.

Furthermore, the subgradient $p \in \partial J(v)$ is not unique, when $J(v) = \|v\|_1$.

2.2. Earlier work using Bregman divergence

Some earlier work concerning Bregman divergence, logistic regression and Adaboost can be found in [7]. Such a Bregman divergence framework was extended to ℓ_1-regularized logistic regression in [15]. Both applied the Bregman divergence to the logistic loss term.

On the other hand, application of Bregman divergence to the ℓ_1 term was introduced in [23], where an inverse problem of signal and noise decomposition was considered. In that model, $J(u) = \int |\nabla u|$ is the total variation functional of u, and $H(u) = \frac{1}{2}\|Au-y\|_2^2$. The key idea was to solve a sequence of minimization subproblems, where the Bregman divergence was applied to total variation (TV). Such methodology was later applied to wavelet denoising [32] and image deblurring. Recently, the Bregman regularization was applied to the following ℓ_1-regularized basis pursuit problem [35],

$$\min_u \lambda\|u\|_1 + \frac{1}{2}\|Au - y\|_2^2, \tag{6}$$

where the regularization term is the ℓ_1 term $J(u) = \|u\|_1$, and the fidelity term takes the quadratic form. This optimization problem is at the core of compressive sensing.

More recently, a linearized Bregman algorithm was derived in [35], which is simple and fast. Interesting analysis was done in [33] for compressive sensing. Some theoretical results regarding convergence was established in [4, 5]. In all these papers, the authors focused on seeking the solution of (6) with small λ via linearized Bregman algorithm, which is simple and fast. What makes Bregman regularization interesting is its error canceling property of Bregman divergence, when applied to the ℓ_1 norm [34]. The authors were interested in the final solution of the algorithm, instead of the solution path, using the linearized Bregman algorithm. The solution path was documented in the direct Bregman algorithm for TV [23] and the inverse scale space [3].

3. New regularization path for sparse learning

In the sparse learning problem below, we are interested in the full solution path. Computing the full regularization path fits nicely with the cross validation or early stopping procedure in the machine learning community. We first show a new regularization path using the Bregman divergence in Section 3.1, and compare it with

the traditional grid search method. We then describe a revised regularization path generated by the linearized Bregman algorithm, which is much faster, in Section 3.2.

3.1. New regularization path through Bregman divergence

We first introduce a new regularization path using the Bregman divergence, resulting in the direct Bregman method. The difference between the grid search method and the direct Bregman method is illustrated below.

- **Grid Search Method**
  Given a sequence of regularization parameters $\lambda_0 > \lambda_1 > \ldots > \lambda_k > \ldots > \lambda_n$, we solve a sequence of minimization subproblems corresponding to each λ_k,

$$(w^k, v^k) \leftarrow \arg\min_{w,v} \ \lambda_k J(w) + l_{\text{avg}}(w, v), \tag{7}$$

  for $k = 0, 1, \ldots, n$.
- **Direct Bregman Method**
  Given a fixed regularization parameter λ_0, which is large enough so that w^1 will be sufficiently sparse, we solve a sequence of minimization subproblems,

$$(w^{k+1}, v^{k+1}) \leftarrow \arg\min_{w,v} \lambda_0 D_J^p(w, w^k) + l_{\text{avg}}(w, v), \tag{8}$$

  for $k = 0, 1, \ldots, n-1$, with initial conditions $w^0 = \mathbf{0}$, $v^0 = 0$, and $p^0 = \mathbf{0}$.

In the above direct Bregman method, $\{(w^k, v^k)\}$ is a sequence of solutions, and $p^k \in \partial J(w^k)$ is the subgradient of $J(w^k)$, where $J(w^k) = \|w^k\|_1$. By substituting the definition of Bregman divergence, we arrive at

$$(w^{k+1}, v^{k+1}) \leftarrow \arg\min_{w,v} \lambda_0\big(J(w) - J(w^k) - \langle w - w^k, p^k\rangle\big) + l_{\text{avg}}(w, v). \tag{9}$$

One can further simply this expression to

$$(w^{k+1}, v^{k+1}) \leftarrow \arg\min_{w,v} \lambda_0 J(w) - \lambda_0\langle w, p^k\rangle + l_{\text{avg}}(w, v). \tag{10}$$

Fig. 1 illustrates the difference between these two methods. Note in Eqn. (10), the first two terms $\lambda_0 J(w) - \lambda_0\langle w, p^k\rangle$ together determine the contribution of the regularization term. Roughly speaking, the amount of regularization decreases along the new regularization path.

Given that (w^{k+1}, v^{k+1}) satisfies the first-order optimality condition of problem (8),

$$\begin{aligned}
&\mathbf{0} \in \partial\big(\lambda_0 J(w^{k+1}) - \lambda_0\langle w^{k+1}, p^k\rangle + l_{\text{avg}}(w^{k+1}, v^{k+1})\big)\\
&\mathbf{0} = \lambda_0 p^{k+1} - \lambda_0 p^k + \nabla_w l_{\text{avg}}(w^{k+1}, v^{k+1}),
\end{aligned}$$

where p^k is the subgradient of $J(w^k)$, not single-valued due to the non-differentiability of ℓ_1 norm. Hence we arrive at the iterate for updating p^{k+1},

$$p^{k+1} = p^k - \frac{1}{\lambda_0}\nabla_w l_{\text{avg}}(w^{k+1}, v^{k+1}). \tag{11}$$

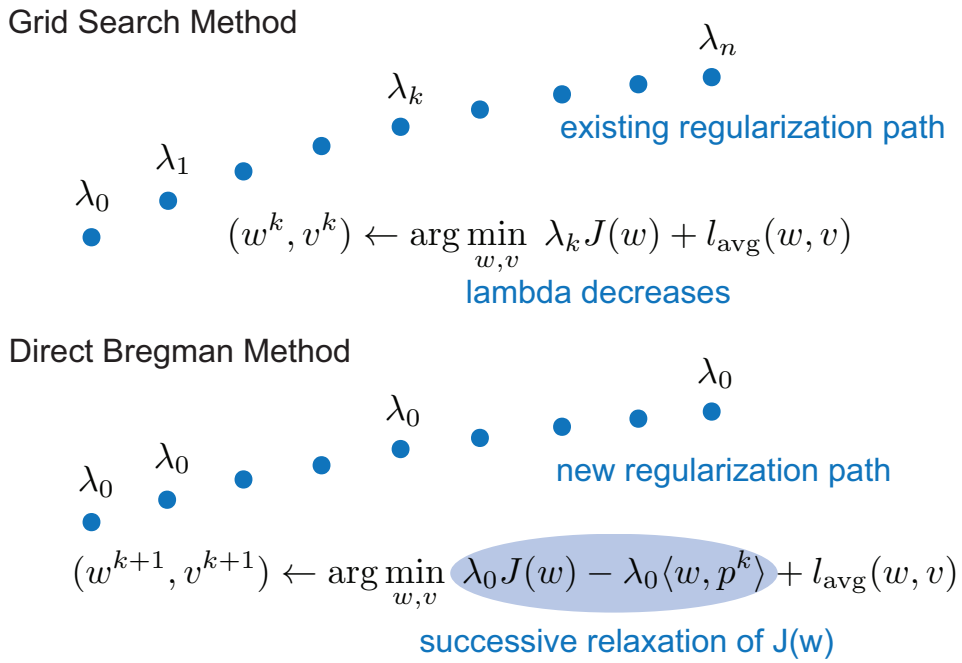


Figure 1. Comparison between the grid search method and the direct Bregman method. These two methods generate different regularization paths. In the grid search method, a sequence of decreasing lambdas are generated as the input to the algorithm, and each subproblems solves for the original optimization problem with each lambda. In the direct Bregman method, lambda is fixed, and each subproblem solves for the Bregman regularized optimization problem.

Algorithm 1 Direct Bregman Method

Input: d ata $X \in \mathbb{R}^{m \times n}$ and label $y \in \mathbb{R}^m$.
Initialize $k = 0$, $w^0 = \mathbf{0}$, $v = 0$, $p^0 = \mathbf{0}$, $\lambda_0 > 0$.
while stopping criterion not satisfied **do**
  $(w^{k+1}, v^{k+1}) \leftarrow \arg\min_{w,v} \lambda_0 J(w) - \lambda_0 \langle w, p^k \rangle + l_{\text{avg}}(w, v)$
  $p^{k+1} \leftarrow p^k - \frac{1}{\lambda_0} \nabla_w l_{\text{avg}}(w^{k+1}, v^{k+1})$
  $k \leftarrow k + 1$
end while

So far, (8) and (11) constitute the direct Bregman method. We summarize it in Algorithm 1.

The direct Bregman procedure generates a regularization path. Compared to the grid search method, where a sequence $\{\lambda_k\}$ controls the regularization path, the direct Bregman procedure generates a regularization path without tuning the regularization parameter λ. This is the key insight of this algorithm, illustrated in Fig. 1. Note the grid search method and direct Bregman method generate two different regularization paths. We will discuss the difference between these two regularization paths from the perspective of inverse scale space in Section 6.

Since each subproblem (10) needs to be solved accurately, the direct Bregman method turns out to be computationally expensive. Therefore we derive a simpler and more efficient algorithm called linearized Bregman in the next section.

3.2. Linearized Bregman algorithm

Recall our goal is to improve the regularization path, especially in terms of computational efficiency. During the direct Bregman procedure, each subproblem needs to be solved accurately. Suppose each subproblem can be solved easily, then our goal is achieved; we do so via linearization. By linearizing the loss function, we can obtain a close-form solution for each subproblem (10). This will result in the linearized Bregman algorithm, as well as a different regularization path.

The linearized version of the direct Bregman algorithm can be derived by approximating the loss function $l_{\text{avg}}(w, v)$ using first-order Taylor expansion at (w^k, v^k), which is $l_{\text{avg}}(w^k, v^k) + \langle \nabla_w l_{\text{avg}}(w^k, v^k), w \rangle$, and adding a proximal term $\|w - w^k\|_2^2/(2\alpha)$ to the objective function,

$$(w^{k+1}, v^{k+1}) \leftarrow \arg\min_{w,v} \lambda_0 D_J^p(w, w^k) + \langle w, \nabla_w l_{\text{avg}}(w^k, v^k) \rangle + \frac{1}{2\alpha}\|w - w^k\|_2^2. \tag{12}$$

Now we can further group w from the last two terms and get

$$(w^{k+1}, v^{k+1}) \leftarrow \arg\min_{w,v} \lambda_0 D_J^p(w, w^k) + \frac{1}{2\alpha}\|w - (w^k - \alpha \nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2. \tag{13}$$

In order to derive the update rule for p^{k+1}, we use the optimality condition for the objective function of the linearized Bregman procedure (13), which leads to

$$p^{k+1} = p^k - \frac{1}{\lambda_0} \nabla_w l_{\text{avg}}(w^k, v^k) - \frac{1}{\lambda_0 \alpha}(w^{k+1} - w^k). \tag{14}$$

Hence the iterates (13) together with (14) constitute the linearized Bregman iterative algorithm, summarized in Algorithm 2.

Algorithm 2 Linearized Bregman Iterative Algorithm

Input: data $X \in \mathbb{R}^{m \times n}$ and label $y \in \mathbb{R}^m$.
Initialize $k = 0$, $w^0 = \mathbf{0}$, $v^0 = 0$, $p^0 = \mathbf{0}$, $\lambda_0 > 0$, $\alpha > 0$.
while stopping criterion not satisfied **do**
  $(w^{k+1}, v^{k+1}) \leftarrow \arg\min_{w,v} \lambda_0 D_J^p(w, w^k) + \frac{1}{2\alpha}\|w - (w^k - \alpha \nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2$
  $p^{k+1} \leftarrow p^k - \frac{1}{\lambda_0} \nabla_w l_{\text{avg}}(w^k, v^k) - \frac{1}{\lambda_0 \alpha}(w^{k+1} - w^k)$
  $k \leftarrow k + 1$
end while

We further simplify the linearized Bregman iterative algorithm below.

Theorem 3.1. *The solution to the original linearized Bregman iterative algorithm, (w^k, v^k), solves a sequence of minimizers (13) along with subgradient update (14). Such an algorithm can be further reduced to updating a sequence of (w^k, v^k, z^k), involving*

$$\begin{aligned} z^{k+1} &= z^k - \frac{1}{\lambda_0}\nabla_w l_{\text{avg}}(w^k, v^k), \\ w^{k+1} &= \lambda_0 \alpha \mathcal{S}(z^{k+1}, 1), \\ v^{k+1} &= v^k - \frac{1}{\lambda_0}\nabla_v l_{\text{avg}}(w^k, v^k), \end{aligned} \tag{15}$$

where $\mathcal{S}$ is the shrinkage operator.

Proof. The objective function for each subproblem for the linearized Bregman iterative algorithm (13) can be reduced as

$$\begin{aligned} &\Rightarrow \arg\min_{w,v} \lambda_0 D_J^p(w, w^k) + \frac{1}{2\alpha}\|w - (w^k - \alpha\nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2 \\ &\Rightarrow \arg\min_{w,v} \lambda_0 J(w) - \lambda_0 J(w^k) - \lambda_0\langle w - w^k, p^k\rangle + \frac{1}{2\alpha}\|w - (w^k - \alpha\nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2 \\ &\Rightarrow \arg\min_{w,v} \lambda_0 J(w) - \lambda_0\langle w, p^k\rangle + \frac{1}{2\alpha}\|w - (w^k - \alpha\nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2. \end{aligned}$$

We hence have the following update,

$$(w^{k+1}, v^{k+1}) \leftarrow \arg\min_{w,v} \lambda_0 J(w) + \frac{1}{2\alpha}\|w - (w^k + \lambda_0\alpha p^k - \alpha\nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2. \tag{16}$$

The updating rule for the subgradient $p^k \in \partial J(w^k)$, where $J(w^k) = \|w^k\|_1$, can be obtained from the first-order optimality condition of (13),

$$\begin{aligned} &\Rightarrow \mathbf{0} \in \partial\big(\lambda_0 D_J^p(w, w^k) + \frac{1}{2\alpha}\|w - (w^k - \alpha\nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2\big) \\ &\Rightarrow \mathbf{0} \in \partial\big(\lambda_0 J(w) - \lambda_0 J(w^k) - \lambda_0\langle w - w^k, p^k\rangle + \frac{1}{2\alpha}\|w - (w^k - \alpha\nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2\big) \\ &\Rightarrow \mathbf{0} \in \lambda_0 p^{k+1} - \lambda_0 p^k + \frac{1}{\alpha}w^{k+1} - \frac{1}{\alpha}(w^k - \alpha\nabla_w l_{\text{avg}}(w^k, v^k)). \end{aligned}$$

Now we have the following update for subgradient,

$$p^{k+1} + \frac{1}{\lambda_0\alpha}w^{k+1} = p^k + \frac{1}{\lambda_0\alpha}w^k - \frac{1}{\lambda_0}\nabla_w l_{\text{avg}}(w^k, v^k). \tag{17}$$

In order to simply these two equations further, we introducing a new variable $z^k = p^k + \frac{1}{\lambda_0\alpha}w^k$. Now Eqn. (17) can be rewritten as

$$z^{k+1} = z^k - \frac{1}{\lambda_0}\nabla_w l_{\text{avg}}(w^k, v^k). \tag{18}$$

Eqn. (16) can be rewritten as

$$\begin{aligned} &\Rightarrow \arg\min_{w,v} \lambda_0 J(w) + \frac{1}{2\alpha}\|w - (\lambda_0\alpha z^k - \alpha\nabla_w l_{\text{avg}}(w^k, v^k))\|_2^2 \\ &\Rightarrow \arg\min_{w,v} \lambda_0\alpha\|w\|_1 + \frac{1}{2}\|w - \lambda_0\alpha z^{k+1}\|_2^2. \end{aligned}$$

Algorithm 3 Linearized Bregman Algorithm

Input: data $X \in \mathbb{R}^{m\times n}$ and label $y \in \mathbb{R}^m$.
Initialize $k = 0$, $w^0 = \mathbf{0}$, $v = 0$, $p^0 = \mathbf{0}$, $\lambda_0 > 0$, $\alpha > 0$.
while stopping criterion not satisfied **do**
$z^{k+1} \leftarrow z^k - \frac{1}{\lambda_0}\nabla_w l_{\text{avg}}(w^k, v^k)$
$w^{k+1} \leftarrow \lambda_0 \alpha \mathcal{S}(z^{k+1}, 1)$
$v^{k+1} \leftarrow v^k - \frac{1}{\lambda_0}\nabla_v l_{\text{avg}}(w^k, v^k)$
$k \leftarrow k + 1$
end while

The above minimization has closed-form solution using shrinkage,

$$w^{k+1} = \mathcal{S}(\lambda_0\alpha z^{k+1}, \lambda_0\alpha) = \lambda_0\alpha\mathcal{S}(z^{k+1}, 1), \tag{19}$$

where the shrinkage operator is also referred to as soft thresholding,

$$\mathcal{S}(z, \alpha) := \text{sgn}(z) \odot \max\{|z| - \alpha, 0\} = \begin{cases} z - \alpha & \text{if } z \in (\alpha, \infty) \\ 0 & \text{if } z \in [-\alpha, \alpha] \\ z + \alpha & \text{if } z \in (-\infty, -\alpha), \end{cases} \tag{20}$$

with $\odot$ denoting element-wise product.

Therefore we arrive at a three line code, summarized in Algorithm 3. □

The linearized Bregman algorithm constructs a well-defined sequence $\{(w^k, v^k, z^k)\}$, essentially a regularization path starting from $w^0 = \mathbf{0}$, $v^0 = 0$, $z^0 = \mathbf{0}$. Following this path, the penalization on the ℓ_1 regularization is weakened due to linearization $\langle w, p^k\rangle$, resulting in less sparse solutions, as if the regularization parameter λ is reduced in model (3). The linearized Bregman algorithm is very straightforward to implement, and only involves matrix multiplication and scalar shrinkage. Again, we note that the regularization paths generated by the grid search method, direct Bregman method, and linearized Bregman method are different. Note the update of subgradient is not computationally explicit in Algorithm 3 due to the introduction of new variable z^k. One attractive property of the linearized Bregman algorithm is that it starts from a big λ and achieves denser solutions without actually tuning λ. The solution to the linearized Bregman algorithm, appears to get closer, in the Bregman sense, to the minimizer of the empirical logistic loss function. Such a result was observed and analyzed formally for image denoising, where the fidelity term is quadratic, in [23].

4. Numerical results

4.1. Forward model for data generation

Consider two Gaussian classes with n dimensions, both with covariance matrices equal to identify. Means of the two classes are $\mu_1 = [c, 0, ..., c, 0, c, 0, ..., 0]^T$, where only k

elements are nonzero and the remaining $n-k$ elements are zero, and $\mu_2 = -\mu_1$. The ground truth of these data is known, i.e. the k dimensions with non-zero means are the informative features for a linear classifier. To introduce noise in the data, we add Gaussian noise with zero mean.

4.2. Convergence

We demonstrate some numerical results for the linearized Bregman algorithm concerning algorithm convergence. In this example, $n = 100$, $m = 100$ (50 samples per class).

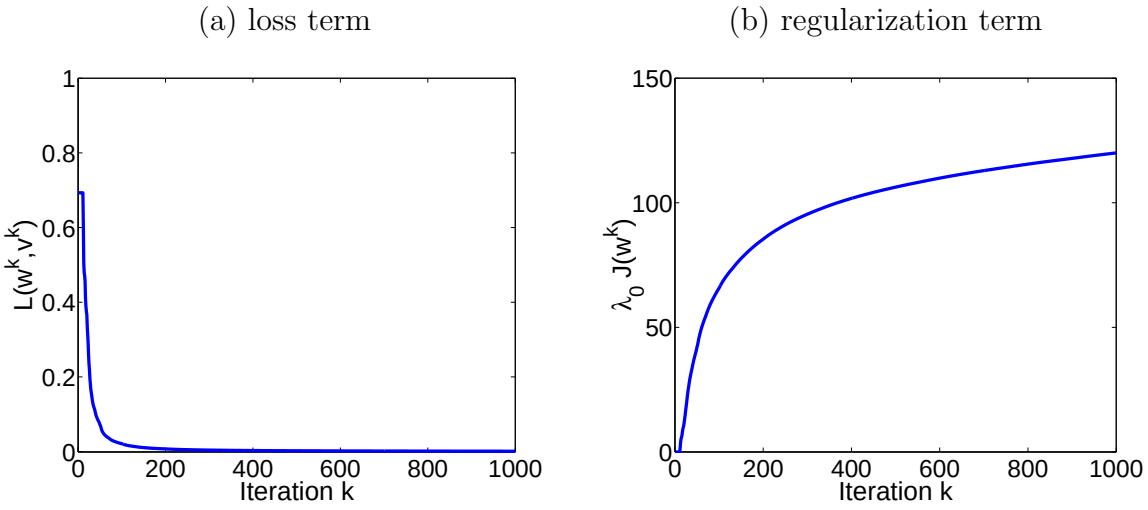


Figure 2. Linearized Bregman algorithm. (a) Logistic loss term $l_{\text{avg}}(w^k, v^k)$ as function of k. (b) Regularization term $\lambda_0 J(w)$ as function of iteration k, where $\lambda_0 = 10$. The data used in this experiment is simulated, with $m = 100$ samples (50 samples per class) and $n = 100$ dimensions.

Fig. 2(a) shows the evolution of loss function $l_{\text{avg}}(w^k, v^k)$ as function of iteration k. This shows that $l_{\text{avg}}(w, v)$ monotonically decreases and converges to the minimum as $k \to \infty$. Fig. 2(b) shows the evolution of the regularization term $\lambda_0 J(w^k)$ as function of k. The solution gets less and less sparse along the regularization path.

Let's denote residual as $\|w^k - w^*\|$, where w^* is the optimal solution where $w^* = \arg\min_w l_{\text{avg}}(w, v)$. We plot $\|w^k - w^*\|$ as a function of iteration k in Fig. 3(a). One can observe empirically that the convergence is at least linear. Fig. 3(b) shows how the solution w^k evolves as function of iteration k. The new regularization path generated by the linearized Bregman algorithm is finely grained, evidenced by the smooth evolution of each component in w^k.

4.3. Computing the full regularization path

As mentioned earlier, practical usage of the ℓ_1-regularized logistic regression for classification problems requires running the algorithm on a regularization path with

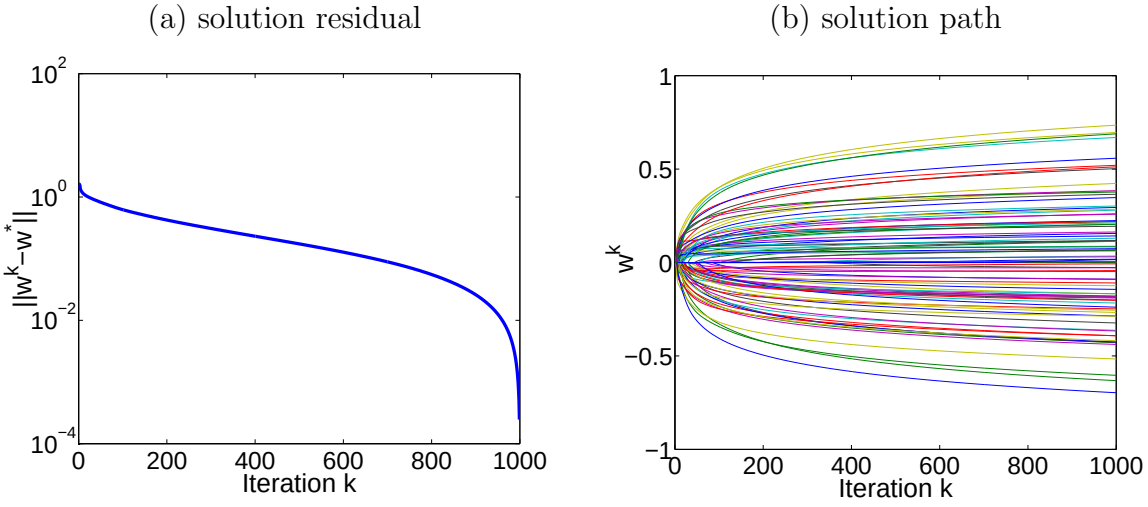


Figure 3. Linearized Bregman algorithm. (a) The residual of solution $\|w^k - w^*\|$ as a function of iteration k. (b) Solution w^k evolves as function of iteration k. One can observe the evolution is very smooth along the regularization path. The data used in this experiment is simulated, with $m = 100$ samples (50 samples per class) and $n = 100$ dimensions.

a sequence of decreasing λ, through the grid search approach. In general, the regularization parameter λ affects the number of iterations to converge for most solvers. Since the linearized Bregman algorithm has first-order accuracy, it is fair to compare it with the fixed point continuation (FPC) algorithm, which is an iterative soft-thresholding algorithm (ISTA) [6, 8, 1] with acceleration by parameter continuation.

The solution and computation time of FPC for each λ are given in Fig. 4. Fig. 4(a) illustrates how the solution component evolves along a regularization path. As λ becomes smaller, the cardinality of the solution support goes up, however the computation time needed for convergence also increases, an apparent computational disadvantage, shown in Fig. 4(b).

We then compare solution paths generated by the linearized Bregman algorithm and the grid search method. We run these two algorithms independently on the same data, and record the accumulated computational time. Fig. 5 contrasts the computational efficiency and quality of solution paths between these two algorithms. Fig. 5(a) and (b) show how the cardinality of solution support evolves as function of accumulated computational time. One can see that linearized Bregman is much more efficient in generating the full solution path. Fig. 5(c) and (d) illustrate the evolution of solution path for both algorithms. One can see the solution path generated by the linearized Bregman algorithm is much more finely grained compared with the grid search method. We note the magnitude of the solutions generated by the linearized Bregman algorithm and the grid search emthod are different. The differences are due to the fact that the linearized Bregman algorithm iteratively applies linear relaxations the ℓ_1 regularization term $J(w)$, and the particular form of ℓ_1-induced Bregman divergence frees nonzero

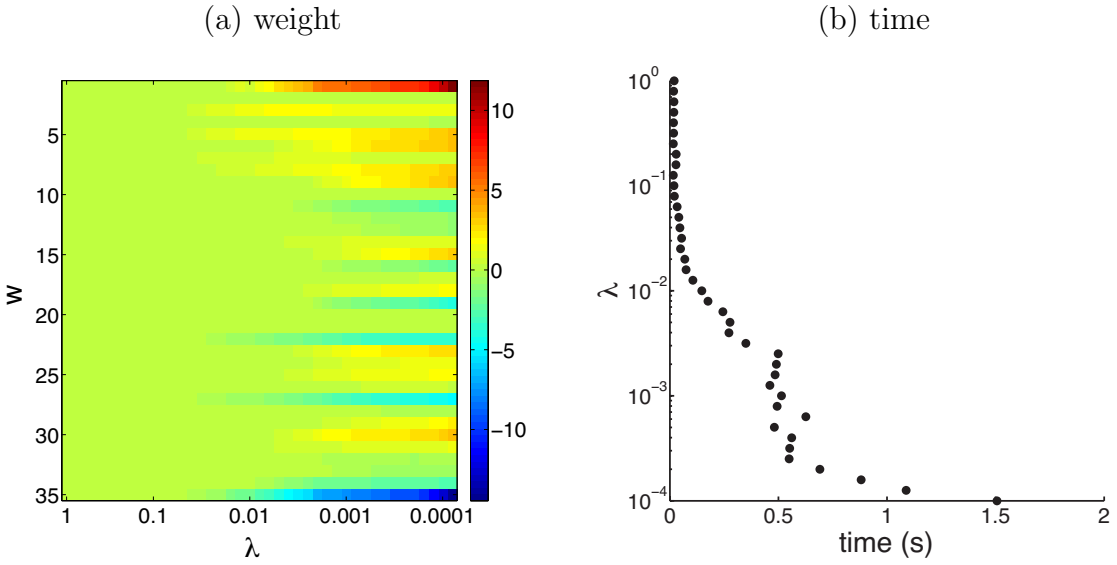


Figure 4. Regularization parameter selection using the grid search method. We show the effect of regularization parameter on FPC algorithm. (a) Solution w evolves along a regularization path, following a geometric progression from 1 to 0.0001. As the λ becomes smaller, the cardinality of the solution goes up. (b) Computation time along such a regularization path, where the smaller λ requires more computation time. Data used in this simulation is the ionosphere data from the UCI repository of machine learning databases.

entries in the last iteration from being penalized in the current iteration, so their values become larger. On the other hand, all entries in the grid search method are penalized equally by ℓ_1 regularization, so the nonzero entries in the solution are smaller due to the ℓ_1 penalty.

The linearized Bregman iterative algorithm can solve a problem starting from a big λ, and create a new regularization path that "effectively" decreases the amount of regularization without tuning λ. More importantly, each iterate of the linearized Bregman is exact. This indicates an important advantage of using the Bregman iterative algorithm.

When one runs the linearized Bregman algorithm, the notion of regularization parameter selection becomes obsolete. The algorithm starts from a big regularization parameter λ and converges to the true solution. In the grid search method, one needs to construct a regularization path varying λ. Therefore, it requires solving a sequence of minimization subproblems with different λ. Such an approach can be extremely time costly. We can thus gain a tremendous amount of speed-up using the linearized Bregman algorithm. Secondly, one can see that linearized Bregman algorithm gives a finer grain regularization path. In the case of grid search method, one can miss the true solution if one does not fine tune λ.

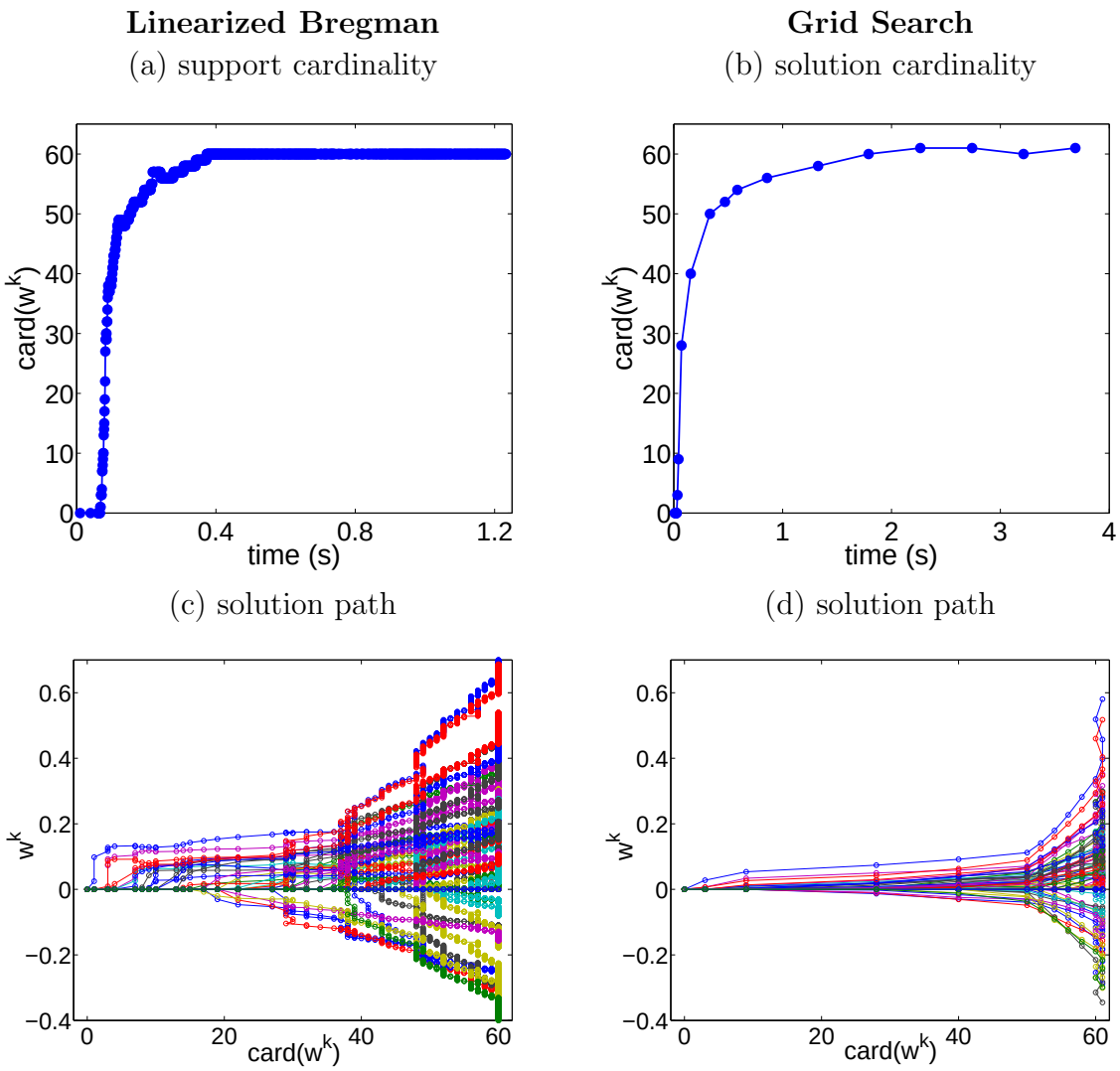


Figure 5. Comparison of solution paths between the linearized Bregman algorithm and the grid search method. (a) & (b) show the evolution of support cardinality plotted against accumulated computation time. We ran the two algorithms independently and recorded computation time, and compare against each other. (c) & (d) show the evolution of solution w^k plotted against the evolution of $\text{card}(w^k)$. One can see the linearized Bregman algorithm generates a solution path much more efficiently and is more finely grained.

4.4. Algorithm scaling

In order to test whether the proposed algorithm is scalable for large-scale data, we compare the linearized Bregman algorithm with the grid search method, on data of different dimensions. We note such a comparison is dependent on data, and how the regularization parameters are chosen for the grid search. In order to make a fair comparison, we simulate data under the same distribution (see Section 4.1). For all cases, we run the linearized Bregman algorithm for 100 time steps, and run the FPC

for 100 different regularization parameters.

In the grid search method, a geometric progression is used for constructing the sequence of regularization parameters. We compute the maximum regularization parameter using

$$\lambda_{\max} = \frac{1}{m}\Big\|\frac{m_-}{m}\sum_{b_i=+1} a_i + \frac{m_+}{m}\sum_{b_i=-1} a_i\Big\|_\infty, \tag{21}$$

where m_- is the number of training samples with label -1 and m_+ is the number of training samples with label $+1$ [17]. $\lambda_{\max}$ is an upper bound for the useful range of regularization parameters, which is the smallest regularization parameter that gives rise to trivial solution. In the linearized Bregman algorithm, we use λ_0 which gives rise to over-smoothed initial solution and stable solution path.

All computation is done using MATLAB R2010a, on a MacBook Pro laptop with Mac OSX 10.6.5, with 2.66 GHz Intel Core i7, and 8 GB 1067 MHz DDR3 memory. Clearly linearized Bregman algorithm is almost 100 times more efficient compared with the grid search method.

n	m	k	Linearized Bregman	Grid Search
10	10	8	0.05 s	14.57 s
100	100	80	0.13 s	33.74 s
1000	1000	800	3.16 s	319.15 s
10000	10000	8000	208.68 s	16085.32 s

Table 1. This table summaries the scaling of both algorithms. We test the computational efficiency of both algorithms for data with different dimensions. The data used in this experiment is simulated, see Section 5.1 for details. Here $n = \#$ dimensions, $m = \#$ samples, and $k = \#$ nonzeros within the dimensions of the original data.

Table 1 is a summary of computational efficiency. We note linearized Bregman induces almost 100 times speedup, compared with the grid search method. Such a result suggests that linearized Bregman can facilitate computational efficiency for large-scale learning problems.

4.5. Generalization performance

In order to prevent overfitting, the optimal level of sparsity is typically not when the classification performance achieves its maximum on the training data, but on testing data. Therefore, it is important to examine the quality of generalization performance of the algorithm. We thus compare the generalization performance for the linearized Bregman algorithm and the grid search method.

This is done using cross validation traditionally; recently early stopping has been quite popular in large-scale learning. The classification performance can be evaluated

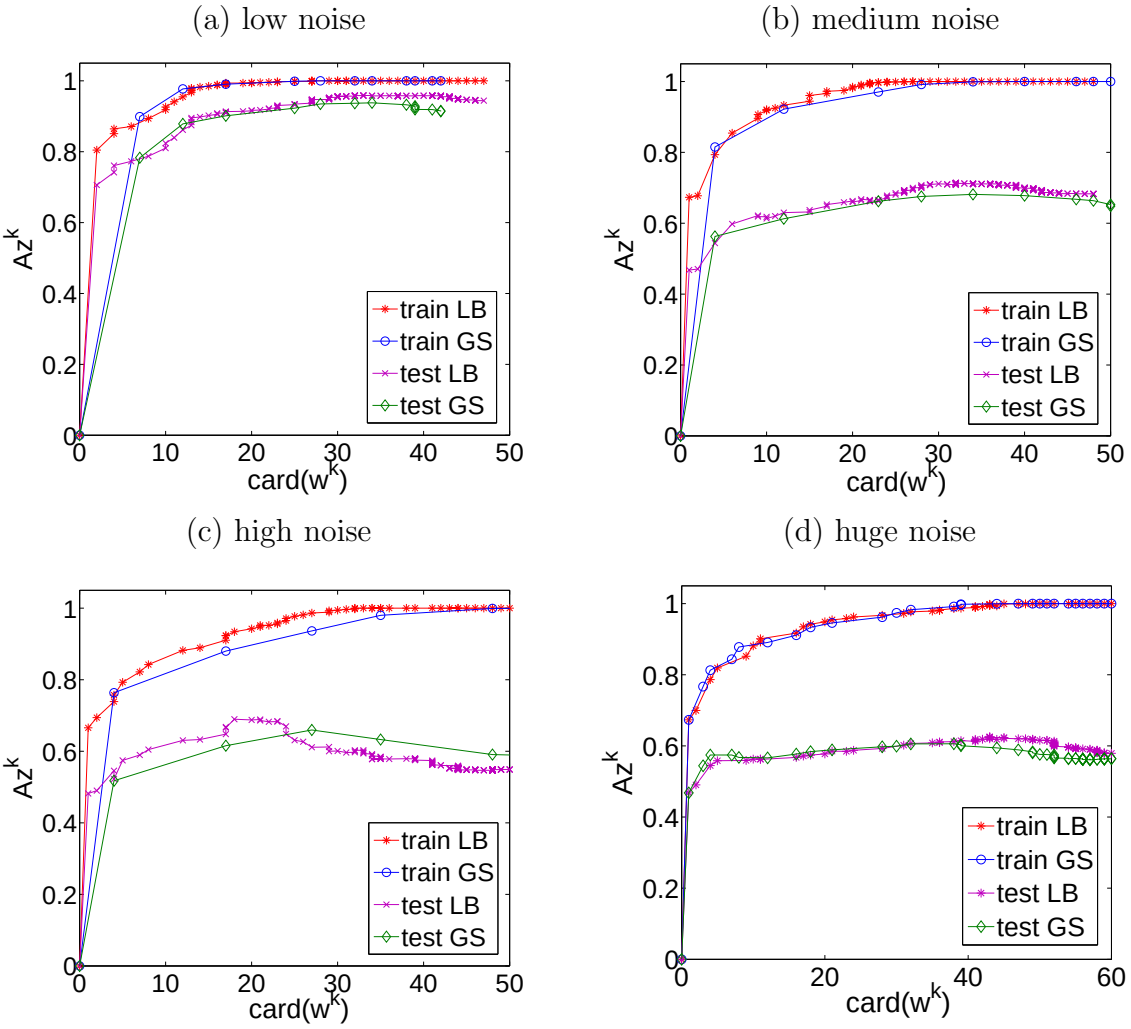


Figure 6. Comparison of generalization performance between the linearized Bregman algorithm (LB) and the grid search method (GS). We run both algorithms on training and testing dataset. The curves are labeled in different colors: (red) training LB, (blue) training GS, (cyan) testing LB, (green) testing GS. In (a)-(d), noise level in the data are increased.

using the receiver operating characteristic (ROC) analysis and K-fold cross validation. In this case, we used two-fold cross validation, similar to early stopping technique in the machine learning community. In signal detection theory, a ROC curve plots the true positive rate versus false positive rate, as the discrimination threshold varies, for a binary classifier system [14]. The area under the ROC curve is termed Az value, $Az \in [0, 1]$. The higher the Az value, the better the classification performance.

Fig. 6 shows the generalization performance for data with varying noise level. We split the data into training and testing sets, and run both linearized Bregman algorithm and the grid search method on them. The generalization performance for the dataset is determined when the testing Az reaches its maximum. The numerical results show that linearized Bregman always achieves comparable, if not better, testing performance

compared with the grid search method. More specifically, the maximum of testing Az value using linearized Bregman algorithm is the same as, or sometimes slightly higher than, that of the grid search method. Fig. 6(a)-(d) shows the consistency of such a result for increasing noise level.

4.6. Algorithm parameters

We note that there are two parameters for the linearized Bregman algorithm, λ_0 and α. Here we study the impact of these two parameters on the algorithm performance in terms of stability and solution path. Fig. 7 illustrates such an idea.

On one hand, α needs to be sufficiently small. When α is too big, the algorithm can become unstable. Fig. 7(b) shows some oscillations in the testing Az curve. Note that α controls the proximal term for linearized Bregman algorithm. This term should be small to make sure the linearization is accurate.

On the other hand, λ_0 needs to be large enough. This is because the step size for gradient descent is $\frac{1}{\lambda_0}$ in Algorithm 3. In Fig. 7(e), we also notice that the solution path becomes more finely grained when λ_0 becomes larger.

5. Feature selection as inverse problem

It is instrumental to test the feature selection behavior of the algorithm, in a manner where the ground truth is known. The forward model for data generation is described in Section 4.1. We set up feature selection as an inverse problem. We run the linearized Bregman algorithm on the training data, and use cross-validation to obtain the generalized classification performance.

We would like to design a stopping criterion for the linearized Bregman algorithm. According to the generalized discrepancy principle, this can be achieved by finding the minimal cardinality in the regularization path that achieves maximum classification performance generalized on the testing data.

$$(w^{opt}, v^{opt}) =: \min_{\mathbf{card}(w)} \max_{Az} Az(w, v, X_{test}, y_{test}), \tag{22}$$

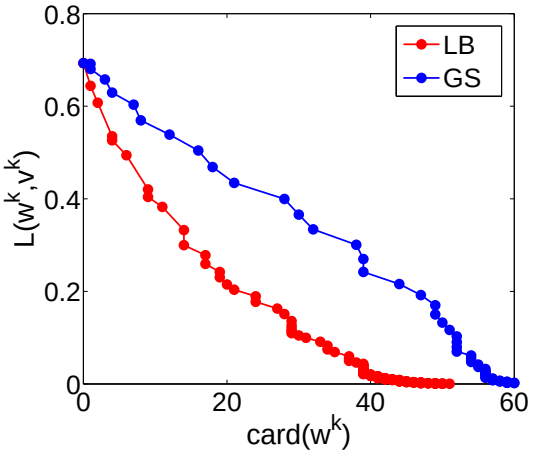
where **card** denotes the cardinality of support, i.e. the number of nonzero elements in a vector, essentially the ℓ_0-norm.

5.1. Feature selection for noise-free data

We first study the case where there is no noise in the input data. As described in Section 5.1, we generate data where dimension $n = 100$, sample numbers $m = 100$, and the number of support in the data is 50. Cross validation is used when we evaluate the classification performance. ROC analysis is employed to compute Az value, which is a measurement of solution quality in learning theory.

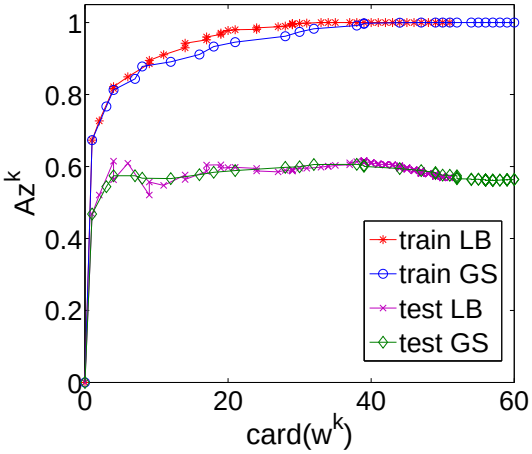
Loss Function

(a) $\lambda_0 = 10$ $\alpha = 0.1$

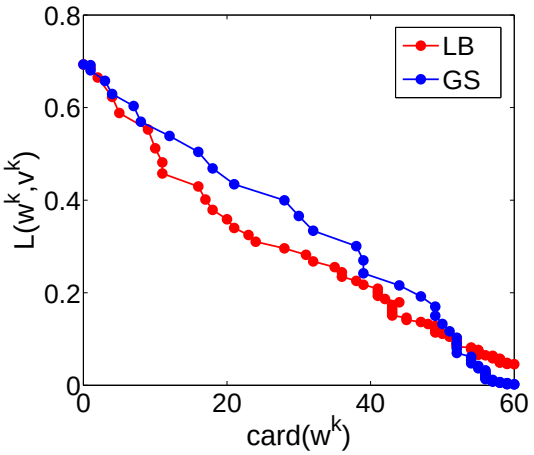


Solution Path

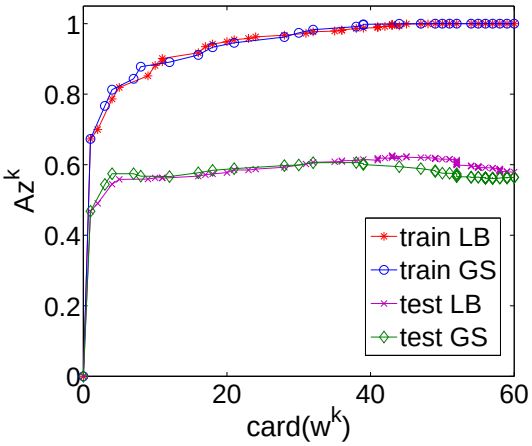
(b) $\lambda_0 = 10$ $\alpha = 0.1$



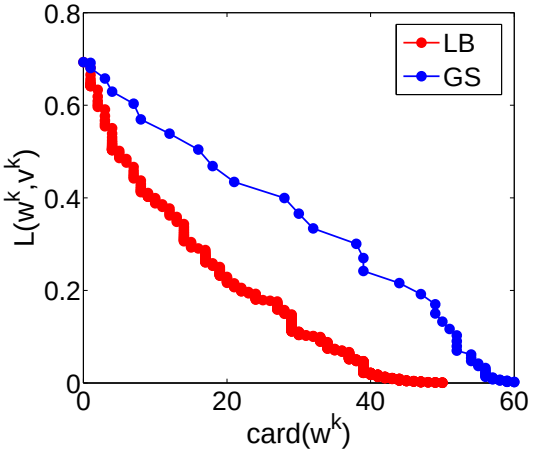
(c) $\lambda_0 = 10$ $\alpha = 0.01$



(d) $\lambda_0 = 10$ $\alpha = 0.01$



(e) $\lambda_0 = 100$ $\alpha = 0.01$



(f) $\lambda_0 = 100$ $\alpha = 0.01$

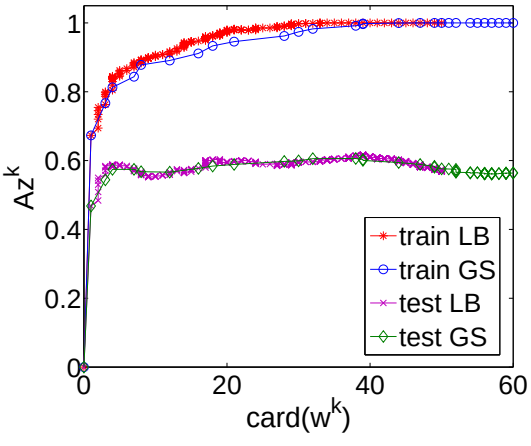


Figure 7. Impact of parameters on algorithm stability and solution path for the linearized Bregman algorithm. Left column compares the loss function against solution cardinality, between linearized Bregman (LB) and grid search (GS) methods. Right column compares the generalization performance. (a) & (b) $\lambda_0 = 10$, $\alpha = 0.1$. (c) & (d) $\lambda_0 = 10$, $\alpha = 0.01$. (e) & (f) $\lambda_0 = 100$, $\alpha = 0.01$.

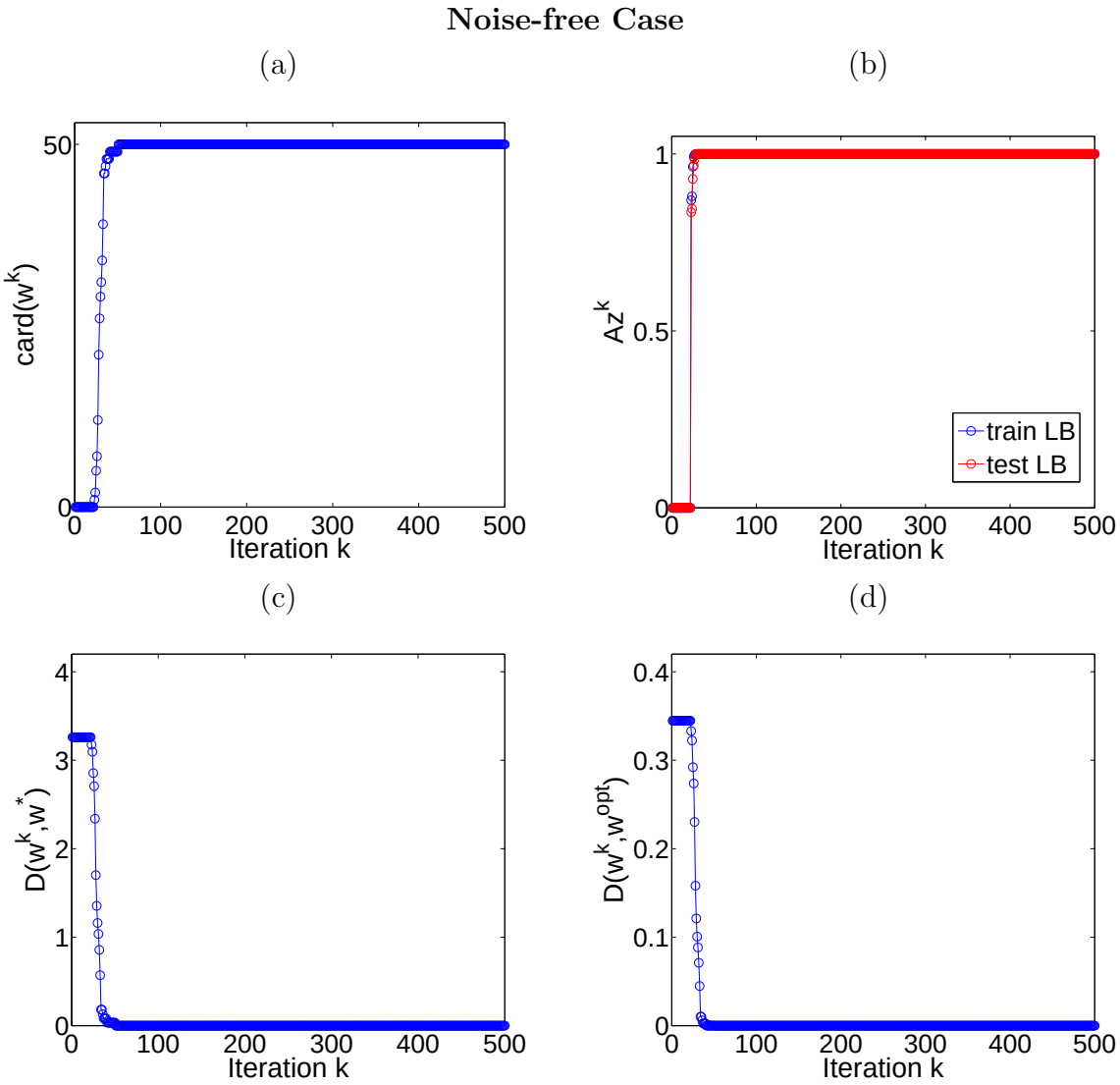


Figure 8. Linearized Bregman algorithm for noise-free data. (a) Cardinality of the solution w_k as a function of iterate k. (b) Classification performance for the linearized Bregman algorithm on both (blue) training data and (red) testing data. (c) Bregman divergence between w^k and the w^*. (d) Bregman divergence between w^k and w^{opt} based on ground truth.

Note when the linearized Bregman algorithm converges, we denote the converged solution as (w^*, v^*). In the numerical experiments below, we will study the following two Bregman divergences, $D(w^k, w^*)$ and $D(w^k, w^{opt})$.

Fig. 8 illustrates the solution quality as a function of iterates k. Linearized Bregman algorithm converges extremely fast, where the cardinality of the solution reaches the true dimension and yielding perfect separation for the testing data. Both training and testing Az values asymptotes perfect classification performance (Az = 1).

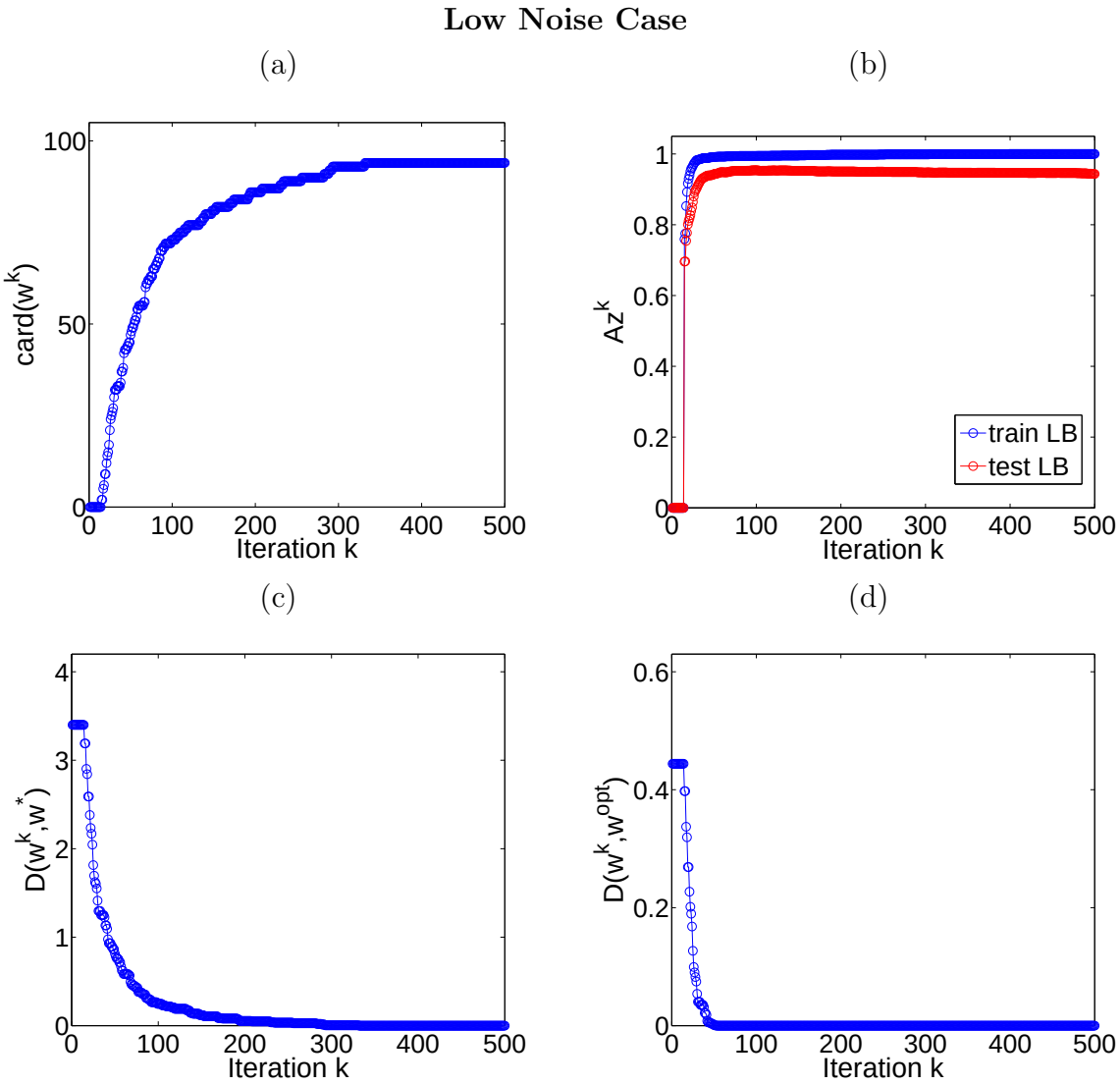


Figure 9. Linearized Bregman algorithm for data with low noise. (a) Cardinality of the solution w_k as a function of iterate k. (b) Classification performance for the linearized Bregman algorithm on both (blue) training data and (red) testing data. (c) Bregman divergence between w^k and the w^*. (d) Bregman divergence between w^k and w^{opt} based on ground truth.

5.2. Feature selection for noisy data

We then test the case where the input data is corrupted by noise, in this case, Gaussian noise. Depending on the amount of noise, the linearized Bregman algorithm appears to have different behaviors.

In the low noise case, linearized Bregman algorithm yields a regularization path, where the solution reaches the optimal solution and eventually asymptotes. Fig. 9 illustrates such an idea. Fig. 9(a) shows along the regularization path, the cardinality of the solution support w^k crosses 50 and asymptotes 94. Fig. 9(b) shows the Az value

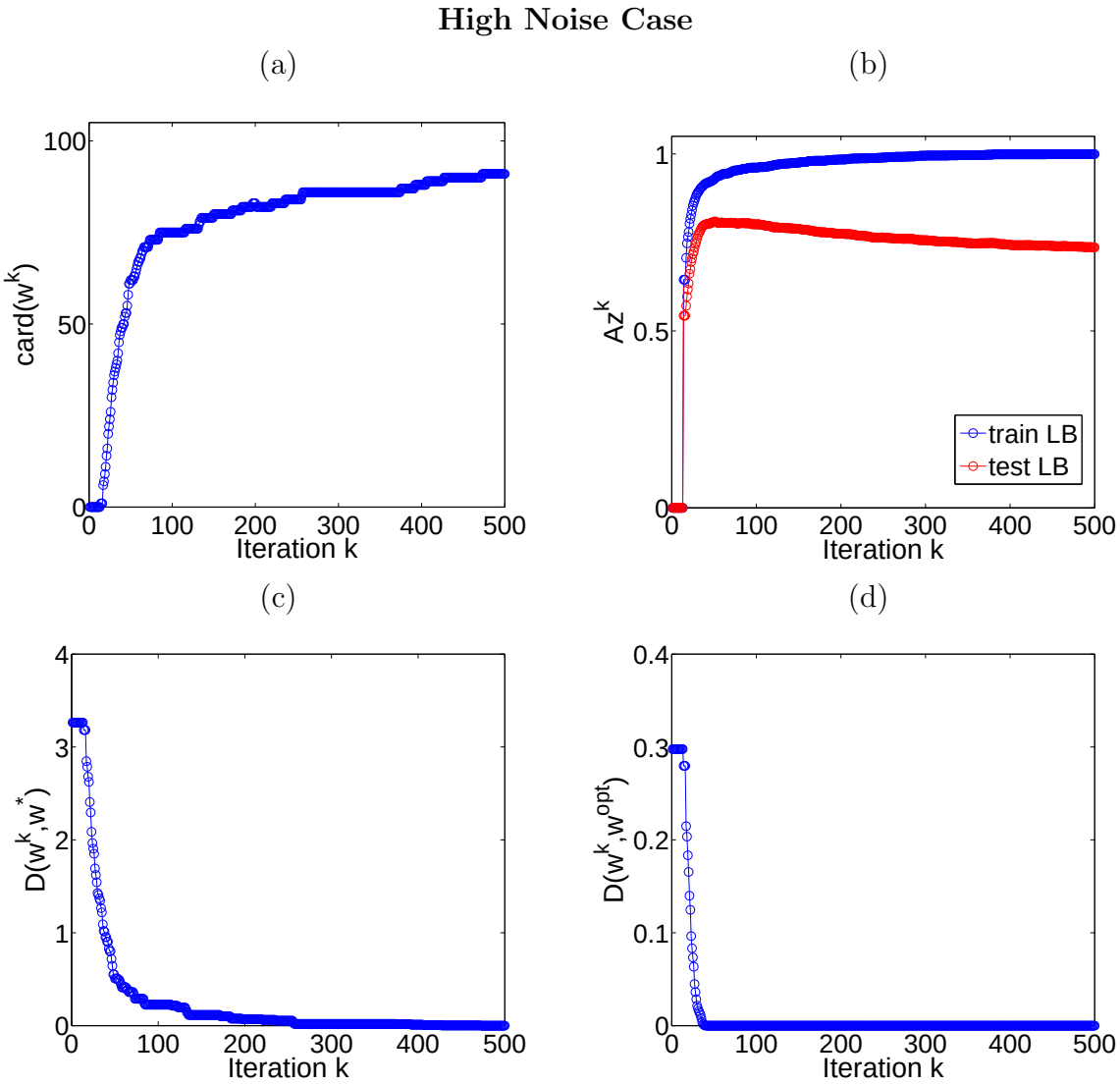


Figure 10. Linearized Bregman algorithm for data with high noise. (a) Cardinality of the solution w^k as a function of iterate k. (b) Classification performance for the linearized Bregman algorithm on both (blue) training data and (red) testing data. (c) Bregman divergence between w^k and the w^*. (d) Bregman divergence between w^k and w^{opt} based on ground truth.

after cross validation (red curve) asymptotes the optimal classification performance. Fig. 9(c) shows when $k \to \infty$, the Bregman divergence between w^k and w^* diminishes. We study the Bregman divergence between each iterative of the solution w_k with the true solution. In the low noise case, we show in Fig. 9(d) that the Bregman divergence between solution w^k and w^{opt}, $D(w^k, w^{opt})$ diminishes after a finite number of iterations.

In the high noise case, linearized Bregman appears to get closer to the true solution and becomes noisy again. Fig. 10(a) shows that the cardinality of the solution starts from 0, passes the true dimension 50 and then goes up. It indicates data overfitting. Fig. 10(b) shows the classification performance has a bell curve (as expected), reaching

its maximum approximately at the iterate where the cardinality of solution recovers the true dimension. Fig. 10(c) shows the Bregman divergence between w^k and w^*. As expected, $D(w^k, w^*)$ diminishes as $k \to \infty$. We also study the Bregman divergence between each iterative of the solution w_k with the true solution. Fig. 10(d) shows the Bregman divergence between solution w^k and w^{opt}, $D(w^k, w^{opt})$ gets smaller and then goes up again (it is too subtle to see on the plot, but can be observed on a log scale). Such a result indicates that the solution w^k gets closer to the true solution, in the sense of Bregman divergence, and becomes noisy again.

6. Solution path explained via inverse scale space flow

There exists a connection between linearized Bregman algorithm and the inverse scale space. Such a connection was examined carefully for the compressive sensing problem, where the fidelity term takes the quadratic form [3].

6.1. Connection with inverse scale space flow

For the direct Bregman method, recall Eqn. (11), the update equation is

$$p^{k+1} = p^k - \frac{1}{\lambda_0}\nabla_w l_{\text{avg}}(w^{k+1}, v^{k+1}). \tag{23}$$

Denote $\Delta t = \frac{1}{\lambda_0}$ and consider it as a time stepping. When taking the limit $\Delta t \to 0$, we can view the above as time evolution and arrive at the following PDE,

$$\frac{dp(t)}{dt} = -\nabla_w l_{\text{avg}}(w(t), v(t)), \quad p(t) \in \partial J(w(t)). \tag{24}$$

Such a PDE is called inverse scale space flow. Note in the asymptotic behavior, we view the direct Bregman iteration as a backward-Euler (implicit) discretization of the above inverse scale space flow.

For the linearized Bregman algorithm, recall Eqn. (14), the update equation is

$$p^{k+1} = p^k - \frac{1}{\lambda_0}\nabla_w l_{\text{avg}}(w^k, v^k) - \frac{1}{\lambda_0\alpha}(w^{k+1} - w^k). \tag{25}$$

This corresponds to the following inverse scale space flow,

$$\frac{dp(t)}{dt} + \frac{1}{\alpha}\frac{dw(t)}{dt} = -\nabla_w l_{\text{avg}}(w(t), v(t)), \quad p(t) \in \partial J(w(t)). \tag{26}$$

In contrast, we view the linearized Bregman iteration as a forward-Euler (explicit) discretization of the inverse scale space flow.

We note that linearized Bregman algorithm is not the first algorithm to compute the full regularization path. The earliest path generating algorithm goes back to the least angle regression (LARS) [9]. The authors proposed an algorithm that solves for a sequence of minimizers to the problem of LASSO,

$$u^k = \arg\min_u J(u) + \frac{t_k}{2}\|Au - f\|_2^2, \tag{27}$$

with $t_0 < t_1 < t_2 < \cdots < t_k$. The asymptotic behavior of the above approaches the following inverse scale space flow,

$$\frac{p(t)}{t} = -A^T(Au(t) - f), \quad p(t) \in \partial J(u(t)). \tag{28}$$

Although this model was not applied to logistic regression, one can potentially generalize the LARS algorithm to ℓ_1-regularized logistic regression,

$$\frac{p(t)}{t} = -\nabla_w l_{\text{avg}}(w(t), v(t)), \quad p(t) \in \partial J(w(t)). \tag{29}$$

Thus far, we can see the regularization paths generated by the direct Bregman method (24), linearized Bregman algorithm (26), and least angle regression (29) are different.

6.2. Behavior of solution path

As mentioned earlier, the solution paths generated by the linearized Bregman algorithm and least angle regression are quite different. Formal analysis of the linearized Bregman algorithm in terms of convergence and solution path is nontrivial, and will be the subject of a future paper. Since linearized Bregman algorithm is a linearized (and accelerated) version to the direct Bregman method, we illustrate the difference between the direct Bregman method and least angle regression through a toy example below.

Consider a simple case where we minimize the following objective function,

$$\arg\min_u \lambda J(u) + \frac{1}{2}\|u - f\|_2^2, \quad J(u) = \|u\|_1. \tag{30}$$

We point out the difference between solution paths for the direct Bregman method and least angle regression by analyzing the corresponding inverse scale space flows:

- Flow for Direct Bregman Method

$$\frac{dp}{dt} = -(u - f), \quad p \in \partial J(u), \tag{31}$$

  starting from $u(0) = 0$, and $p(0) = 0$.

- Flow for Least Angle Regression

$$\frac{p}{t} = -(u - f), \quad p \in \partial J(u), \tag{32}$$

  starting from $u(0) = 0$, and $p(0) = 0$.

Flow for Direct Bregman Method: Formal analysis for the inverse scale space flow Eqn. (31) was discussed in [3] when the right hand side is $-A^\top(Au - f)$. To make it more intuitive, we set $A = I$ and demonstrate the behavior of solution path for the toy example below.

Theorem 6.1. *There exists a sequence of time steps $0 < t_1 < t_2 < \cdots < t_k < \cdots < t_K$, such that the inverse scale space flow Eqn. (31) converges in a finite number of iterations. The number of iterations K is upper bounded by $\|f\|_0$. In addition, the solution converges to $u_k = f$ when $k = K$.*

Proof. We first note $p \in \partial J(u)$ satisfies the following:

$$p_i := \partial J(u) \begin{cases} = -1 & \text{if } u_i < 0 \\ \in [-1, 1] & \text{if } u_i = 0 \\ +1 & \text{if } u_i > 0. \end{cases} \tag{33}$$

Given the inverse scale space flow $\frac{dp(t)}{dt} = f - u(t)$, and initial conditions $u(0) = 0$, $p(0) = 0$, we have

$$p(t) = t(f - u(0)) = tf. \tag{34}$$

Therefore, for $0 < t < t_1$, where $t_1 = \frac{1}{\|f\|_\infty}$, we have

$$\|p(t)\|_\infty = t\|f\|_\infty < t_1\|f\|_\infty = 1. \tag{35}$$

Based on the property of subgradient $p(t) \in \partial J(u)$ mentioned above, it leads to the fact $u(t) = 0$. This result indicates that for large time step up to t_1, $u(t)$ stays zero. In other words, changes in $u(t)$ only occur at t_1 when elements in $p_(t)$ reache $+1$ or -1; otherwise, $p(t)$ behaves linearly in the intermediate time.

Such a result extends to all elements in u. For each element $i \in \text{supp}(f)$, $u_i(t) = 0$ as long as $0 \le t < \frac{1}{|f_i|}$. Once $t \ge \frac{1}{|f_i|}$, the ith element converges to $u_i(t) = f_i$. For each element $i \notin \text{supp}(f)$, it stays constant throughout the flow $u_i(t) = 0$.

With such a conclusion, we can construct the following time iterations $0 < t_1 < t_2 < \cdots < t_k < \cdots < t_K$. Denote the support set of f as $\text{supp}(f)$. We can simply sort the following quantities,

$$\{t_1, t_2, \cdots, t_k, \cdots, t_K\} = \textbf{sort}\{\frac{1}{|f_i|}\}, \text{ for } i \in \text{supp}(f), \tag{36}$$

where **sort** denotes sorting the elements of a set in an ascending order. It is easy to see $K \le \|f\|_0$.

Since $\|f\|_0$ is finite, we have $K \le \|f\|_0 < \infty$. Therefore, the algorithm converges to the solution $u_k = f$ in a finite number of iterations.

□

Flow for Least Angle Regression: Again we explain the behavior of solution path using the toy example where $A = I$.

Theorem 6.2. *For any nontrivial f, the inverse scale space flow for least angle regression does not obtain $u(t_k) = f$ in a finite number of iterations; it converges to $u(t_k) = f$ as $t_k \to \infty$.*

Proof. Upon discretization of Eqn. (32) using $0 < t_1 < t_2 < \cdots < t_k < \cdots < t_K$, we have the following equation at each iteration,

$$J(u) = -t_k \|u - f\|_2^2. \tag{37}$$

One can obtain closed-form solution for $\|u(t_k)\|_1 + t_k \|u(t_k) - f\|_2^2 = 0$,

$$u(t_k) = \mathcal{S}(f, 1/t_k). \tag{38}$$

Note that $\mathcal{S}$ is the shrinkage operator, we have

$$u(t_k) = \begin{cases} f - 1/t_k & \text{if } f \in (1/t_k, \infty) \\ 0 & \text{if } f \in [-1/t_k, 1/t_k] \\ f + 1/t_k & \text{if } f \in (-\infty, -1/t_k). \end{cases} \tag{39}$$

For any nontrivial f, if $f > 0$, then $u(t_k) = f - 1/t_k$; if $f < 0$, then $u(t_k) = f + 1/t_k$.

It is easy to see that as long as $K < \infty$, $u(t_k) \neq f$. In other words, the algorithm does not obtain $u(t_k) = f$ in a finite number of iterations. Only when $t_k \to \infty$, $1/t_k \to 0$, the algorithm converges to $u(t_k) = f$. □

7. Conclusions

We have presented a Bregman regularized model and an efficient linearized Bregman algorithm to generate a solution path for sparse logistic regression. The resulting algorithm accelerates the computation of the regularization path, compared to that of the traditional grid search method. We have tested both algorithms on feature selection problems. The linearized Bregman algorithm achieves comparable classification accuracy on noise-free data, while giving out better classification performance on noisy data. In conclusion, we have found the linearized Bregman algorithm very attractive for seeking the optimal level of sparsity in feature selection problems.

Extension of this work can be applied to support vector machine. We will investigate linearized Bregman algorithm for sparse support vector machine. Many kernel tricks can also be integrated into our framework.

Acknowledgments

Jianing Shi's work is funded in part by NSF DIP-R3D360, ONR SES-R17680, and DARPA MSEE program. Wotao Yin's work is supported in part by ARL and ARO grant W911NF-09-1-0383, NSF grants DMS-0748839 and ECCS-1028790, ONR grant N00014-08-1-1101. Stanley Osher's work is funded in part by DARPA MSEE Program, managed by Dr. Tony Falcone.

References

[1] Bect, J., Laure, B.-F., G. A., and Chambolle, A. A l1-unified variational framework for image restoration. In *8th European Conference on Computer Vision (ECCV)* (2009), vol. 3024, Springer, pp. 1–13.

[2] Bregman, L. M. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics* (1967), 200–217.

[3] Burger, M., Möller, M., Benning, M., and Osher, S. An adaptive inverse scale space method for compressed sensing. Tech. rep., UCLA CAM Report 11-08, 2011.

[4] Cai, J.-F., Osher, S., and Shen, Z. Convergence of the linearized Bregman iteration for ℓ_1-norm minimization. *Math. Comp.* (2009), 2127–2136.

[5] Cai, J.-F., Osher, S., and Shen, Z. Linearized Bregman iterations for compressive sensing. *Math. Comp.* (2009), 1515–1536.

[6] Chambolle, A., DeVore, R. A., Lee, N. Y., and Lucier, B. J. Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Trans. Image Process.* (1998), 319–335.

[7] Collins, M., Schapire, R. E., and Singer, Y. Logistic regression, adaboost and bregman distance. *Machine Learning 48* (2002), 253–285.

[8] Daubechies, I., Defrise, M., and Mol, C. D. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.* (2004), 1413–1457.

[9] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *Annals of Statistics 32(2)* (2004), 407–499.

[10] Figueiredo, M. Adaptive sparseness for supervised learning. *IEEE Trans. Pattern Analysis and Machine Intelligence* (2003), 1150–1159.

[11] Foo, C.-S., Do, C. B., and Ng, A. Y. A majorization-minimization algorithm for (multiple) hyperparameter selection. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning (ICML)* (2009), ACM Press, pp. 321–328.

[12] Genkin, A., Lewis, D. D., and Madigan, D. Large-scale bayesian logistic regression for text categorization. Tech. rep., Rutgers University, 2004. http://www.stat.rutgers.edu/ madigan/BBR/.

[13] Goodman, J. Exponential priors for maximum entropy models. In *Proceedings of the Annual Meetings of the Association for Computational Linguistics* (2004).

[14] Green, D. M., and Swets, J. A. *Signal detection theory and psychophysics.* John Wiley and Sons Inc., 1966.

[15] Gupta, M. D., and Huang, T. S. Bregman distance to ℓ_1 regularized logistic regression. In *IEEE International Conference on Pattern Recognition* (2008), pp. 1–4.

[16] Hale, E., Yin, W., and Zhang, Y. Fixed-point continuation for l1-minimization: methodology and convergence. *SIAM J. Optimization 19(3)* (2008), 1107–1130.

[17] Koh, K., Kim, S.-J., and Boyd, S. An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research* (2007).

[18] Lee, S., Lee, H., Abbeel, P., and Ng, A. Y. Efficient ℓ_1-regularized logistic regression. In *National Conference on Artificial Intelligence (AAAI)* (2006).

[19] Lokhorst, J. The lasso and generalised linear models. Tech. rep., Honors Project, Department of Statistics, University of Adelaide, South Australia, Australia, 1999.

[20] MacKay, D. J. Bayesian interpolation. *Neural Computation* (1992), 415–447.

[21] Ng, A. Y. On feature selection: Learning with exponentially many irrelevant features as training examples. In *International Conference on Machine Learning (ICML)* (1998), pp. 404–412.

[22] Ng, A. Y. Feature selection, l1 vs l2 regularization, and rotational invariance. In *International Conference on Machine Learning (ICML)* (2004), ACM Press, New York, pp. 78–85.

[23] Osher, S., Burger, M., Goldfarb, D., Xu, J., and Yin, W. An iterative regularization

method for total variation based image processing. *SIAM J. Multiscale Modeling and Simulation 4* (2005), 460–489.

[24] Park, M., and Hastie, T. L1 regularized path algorithm for generalized linear models. *J. R. Statist. Soc. B* (2007), 659–677.

[25] Perkins, S., and Theiler, J. Online feature selection using grafting. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)* (2003), ACM Press, pp. 592–599.

[26] Roth, V. The generalized lasso. *IEEE Tran. Neural Networks* (2004), 16–28.

[27] Shi, J., Yin, W., Osher, S., and Sajda, P. A fast hybrid algorithm for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research* (2010), 581–609.

[28] Tibshirani, R. Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc. B 58(1)* (1996), 267–288.

[29] Tipping, M. E. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* (2001), 211–244.

[30] Wright, S. J. Accelerated block-coordinate relaxation for regularized optimization. *SIAM J. Optimization* (2012), 159–186.

[31] Wright, S. J., Nowak, R. D., and Figueiredo, M. Sparse reconstruction by separable approximation. *IEEE Trans. Signal Processing* (2009), 2479–2493.

[32] Xu, J., and Osher, S. Iterative regularization and nonlinear inverse scale space applied to wavelet-based denoising. *IEEE Trans. Image Processing 16(2)* (2007), 534–544.

[33] Yin, W. Analysis and generalization of the linearized bregman method. *SIAM J. Imaging Sciences* (2010), 856–877.

[34] Yin, W., and Osher, S. Error forgetting of Bregman iteration. Tech. rep., Rice University CAAM Report 12-03, 2012.

[35] Yin, W., Osher, S., Darbon, J., and Goldfarb, D. Bregman iterative algorithm for compressed sensing and related problems. *to appear, SIAM J. Imaging Science* (2008).

[36] Zhao, P., and Yu, B. On model selection consistency of lasso. *J. Machine Learning Research 7* (2007), 2541–2567.