Efficient Elastic Net Regularization for Sparse Linear Models

Zachary C. Lipton Charles Elkan

University of California, San Diego
{zlipton, elkan}@cs.ucsd.edu

May 22, 2015

Abstract

We extend previous work on efficiently training linear models by applying stochastic updates to non-zero features only, lazily bringing weights current as needed. To date, only the closed form updates for the ℓ_1, ℓ_∞, and the rarely used ℓ_2 norm have been described. We extend this work by showing the proper closed form updates for the popular ℓ_2^2 and elastic net regularized models. We show a dynamic programming algorithm to calculate the proper elastic net update with only one constant-time subproblem computation per update. Our algorithm handles both fixed and decreasing learning rates and we derive the result for both *stochastic gradient descent* (SGD) and and *forward backward splitting (FoBoS)*. We empirically validate the algorithm, showing that on a bag-of-words dataset with $260,941$ features and 88 nonzero features on average per example, our method trains a logistic regression classifier with elastic net regularization 612 times faster than an otherwise identical implementation with dense updates.

1 Introduction

Machine learning practitioners are often tasked with learning over large datasets. An abundance of affordable storage, network bandwidth, and the proliferation of data-generating devices, have combined to saddle many organizations with *big data* problems. Document auto-tagging problems frequently contain millions of documents, hundreds of thousands of features, and thousands of labels. When the number of labels and features is large, even the weight matrix of a dense linear model may not fit in main memory on a single workstation. However, for many applications features are sparse and good regularized models can be made compact. Given such sparsity, it is desirable train machine learning models using algorithms that require time that scales only with the number of non-zero features.

Online algorithms, such as stochastic gradient descent, are widely used to train high-dimensional models on large-scale data. In these methods, each example is processed one at a time, updating the model on the fly. When a dataset is sparse and the

objective function is not regularized, this sparsity can be exploited by updating only the weights corresponding to non-zero features. However, to prevent overfitting models to high-dimensional data, it is often necessary to apply a regularization penalty, imposing a prior belief that the model parameters should be sparse and small in magnitude. Problematically, widely-used regularizers such as ℓ_1 (lasso), ℓ_2^2 (ridge), and elastic net ($\ell_1 + \ell_2^2$) destroy the sparsity of the gradient, seemingly requiring an update for each nonzero weight for each example.

In this paper, we develop upon a method for lazily updating weights, first described by [2], [10], and [6]. As each example is processed, we update only those weights corresponding to non-zero features. The model is brought current as needed by lazily applying closed-form constant-time updates whenever a feature becomes non-zero. For sparse data sets, the algorithm runs in time independent of the nominal dimensionality, scaling linearly with the number of non-zero features per example.

To date, only the closed form updates for the ℓ_1, ℓ_∞, and the rarely used ℓ_2 norm have been described. We extend this work by showing the proper closed form updates for the popular ℓ_2^2 and elastic net regularized models. When the learning rate varies (typically decreasing as a function of time), we show that the proper elastic net update can be computed with a dynamic programming algorithm, requiring only one constant-time subproblem computation per update. [1]

Additionally, we implement the system, showing that on a widely studied dataset containing the abstracts of millions of articles from biomedical literature, we can train a logistic regression classifier with elastic net regularization 612 times faster than an otherwise identical model with dense updates.

2 Background and Definitions

We consider a data matrix $X \in \mathbb{R}^{\mathrm{n} \times \mathrm{d}}$ where each row $\boldsymbol{x_i}$ corresponds to one of n examples and each column, indexed by j corresponds to one of d features. We desire a linear model, parametrized by a vector $\boldsymbol{w} \in \mathbb{R}^{\mathrm{d}}$, which minimizes a convex objective function $F(w)$, expressible as $\sum_{i=1}^{n} F_i(w)$, where F returns the loss with respect to the entire corpus X and F_i returns the loss calculated on example $\boldsymbol{x}_i$.

In many datasets, the vast majority of entries x_{ij} are zero-valued. The-bag-of-words representation of text is one such case. We say that such datasets are *sparse*. When features correspond to counts or binary values, as in bag-of-words, we sometimes say say that a zero-valued feature x_{ij} is *absent*. We use p to refer to the average number of nonzero features per example. Naturally, when a dataset is sparse, we prefer algorithms which take time $O(p)$ to those $O(d)$.

[1] Our dynamic programming algorithms for constant-time updates with varying learning rates add time O(1) per stochastic gradient update and have space complexity $O(T)$, where T is the total number of stochastic gradient updates. If this space complexity is too great, that problem can be solved by alotting a space budget and bringing all weights current when the budget is filled. As this cost is amortized across many iterations, it adds negligibly to the total running time.

2.1 Regularization

To prevent overfitting, regularization restricts the freedom of a model's parameters, penalizing their distance from some prior belief. Most widely used regularizers penalize large weights with an objective function of the form

$$F(\boldsymbol{w}) = \mathcal{L}(\boldsymbol{w}) + R(\boldsymbol{w}) \tag{1}$$

Many commonly used regularizers $R(w)$ are of the form $\lambda||\boldsymbol{w}||$ where λ determines the strength of regularization and ℓ_0, ℓ_1, ℓ_2^2, and ℓ_∞ are common choices for the norm. The ℓ_1 regularizer is popular owing to its tendency to produce sparse models. In this paper, we focus on elastic net, a linear combination of ℓ_1 and ℓ_2^2 regularization that has been shown to produce comparably sparse models to ℓ_1 while often achieving superior accuracy [12].

2.2 Stochastic Gradient Descent

Gradient descent is a common strategy to find the optimal parameters $\boldsymbol{w}$. Any gradient descent method requires an initial learning rate η. Then to minimize $F(\boldsymbol{w})$, a number of steps T, indexed by t, are taken in the direction of the negative gradient:

$$\boldsymbol{w}^{(t+1)} := \boldsymbol{w}^{(t)} - \eta \sum_{i=1}^{n} \nabla F_i(\boldsymbol{w}).$$

An appropriately attenuated learning rate ensures both that the optimal parameters will eventually be reached and that the algorithm will yield a value within a distance ϵ of the optimal value for any small value ϵ [2] .

Traditional (or "batch") gradient descent requires a pass through the entire dataset for each update. *Stochastic gradient descent* (SGD) circumvents this problem by updating the model once after visiting each example. With SGD, one determines a number of epochs E indexed by e. Within each epoch, examples are randomly selected one at a time (or in "mini-batches"). For simplicity of notation, w.l.o.g., we will assume the examples are selected one at a time. The gradient is calculated with respect to that example, and the model is updated according to the rule $\boldsymbol{w}^{t+1} := \boldsymbol{w}^t - \eta \nabla_{\boldsymbol{w}^{(t)}} F_i$. Because the examples are chosen randomly, the expected value of this noisy gradient is identical to the true value of the gradient taken with respect to the entire corpus.

Given a continuously differentiable convex objective function $F(\boldsymbol{w})$, stochastic gradient descent is known to converge for learning rates η that satisfy $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$ [1]. Learning rates $\eta_i \propto \frac{1}{t}$ and $\eta_i \propto \frac{1}{\sqrt{t}}$ both satisfy these properties. [2] For many objective functions, such as linear or logistic regression without regularization, the noisy gradient $\nabla_{\boldsymbol{w}} F_i$ is sparse when the input is sparse. In these cases, one needs only to update the weights corresponding to non-zero features in the current example $\boldsymbol{x}_i$. This runs in time $O(p)$, where $p \ll d$ is the average number of nonzero features in an example.

[2] Some common objective functions as those with ℓ_1 regularization, are not differentiable when weights are equal to zero. However, forward backward splitting (FoBoS) [10], offers a principled approach to this problem.

Regularization, however, can ruin the sparsity of the gradient. Consider an objective function of form (Equation 1), where $||\boldsymbol{w}||$ for some norm $||\cdot||$. In these cases, even when the feature $x_{ij} = 0$, the gradient $\nabla_{w_j} F_i$ is nonzero owing to the regularization penalty. A simple and correct solution is to update weights when either the weight or feature is nonzero, but to ignore weights when both are zero. Given feature sparsity and persistent model sparsity throughout training, ignoring all weights $w_j = 0$ corresponding to features $x_{ij} = 0$ provides a substantial benefit. But such an algorithm would still scale with the size of the model, which may be far larger than p. Our algorithm scales in time complexity $O(p)$.

2.3 Forward Backward Splitting

Proximal algorithms are an approach to optimization in which each update consists of solving a convex optimization problem [8]. Forward Backward Splitting (FoBoS) [10] uses proximal algorithms to provide a principled approach to online optimization with non-smooth regularizers. We first step in the direction of the negative gradient of the differentiable loss function. We then update the weight by solving a convex optimization problem, which simultaneously penalizes distance from the new parameters and minimizes the regularization term. Duchi and Singer established the convergence of the Forward Backward method in [10].

First an standard unregularized stochastic gradient step is applied

$$\boldsymbol{w}^{(t+\frac{1}{2})} = \boldsymbol{w}^{(t)} - \eta^{(t)} \nabla_{\boldsymbol{w}} \mathcal{L}_i \tag{2}$$

Then a convex optimization is solved which applies the regularization penalty. For elastic net the problem to be solved is

$$\boldsymbol{w}^{(t+1)} = \text{argmin}_w \left(\frac{1}{2} ||\boldsymbol{w} - \boldsymbol{w}^{(t+\frac{1}{2})}||_2^2 + \eta^t \lambda_1 ||\boldsymbol{w}||_1 + \frac{\eta^t \lambda_2}{2} ||\boldsymbol{w}||_2^2 \right) \tag{3}$$

Note that when $\nabla_{w_j} \mathcal{L} = 0, w_j^{(t+\frac{1}{2})} = w_j^{(t)}$. The problems corresponding to ℓ_1 or ℓ_2^2 separately can easily be derived by setting the corresponding λ to 0.

3 Lazy Updates

We extend the method for lazy updates introduced in [2], [6], and [10] to the case of ℓ_2 and elastic net regularization. The essence of the algorithm for sparse lazy updates is as follows (Algorithm 1). We maintain an array $\boldsymbol{\psi} \in \mathbb{R}^d$ in which each ψ_j stores the index of the last iteration at which each feature was nonzero and a counter k, which stores the index of the current iteration. When processing each example $\boldsymbol{x}_i$, we iterate through the nonzero features x_{ij}, comparing the current iteration index k to ψ_j. For each feature with nonzero weight w_j, we lazily apply the $k - \psi_j$ successive updates in constant time, bringing those weights current. Using the updated weights, we compute the prediction $\hat{y}^{(k)}$ with the current parameters $\boldsymbol{w}^{(k)}$. We then compute the gradient and update the weights. When training is complete, we pass once through all nonzero weights to apply the lazy updates to bring the model current. Provided that we can

apply all lazy updates in $O(1)$ time, our algorithm processes each example in $O(p)$ time regardless of the dimension d.

Algorithm 1 Lazy Updates

Require: $\boldsymbol{\psi} \in \mathbb{R}^d$
  for $t \in 1, ..., T$ **do**
    Sample x_i randomly from the training set
    for j s.t. $x_{ij} \neq 0$ **do**
      $w_j \leftarrow Lazy(w_j, t, \psi_j)$
      $\psi_j \leftarrow t$
    end for
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \nabla_{\boldsymbol{w}} F_i$
  end for

To use the algorithm with a chosen regularizer, it remains only to demonstrate the existence of constant time updates. In the following subsections, we derive the closed form updates for ℓ_1, ℓ_2^2 and elastic net regularization, starting with the simple case when the learning rate η is fixed during each epoch, and extending to the more complicated case when the learning rate is decreased between iterations as a function of time. These results hold for schedule of weight decrease that depend on time, but cannot be directly applied to Adagrad, an algorithm for which each weight has a separate learning rate which is decreased with the inverse of the accumulated sum of squared gradients with respect to that weight.

4 Prior Work

Over the last several years, a large body of work has advanced the field of online learning. Notable contributions include ways of adaptively attenuating the learning rate separately for each parameter such as AdaGrad [3] and AdaDelta [11], using small batches to reduce the variance of the noisy gradient ([7]), and variance reduction methods such as Stochastic Average Gradient (SAG) [9], and Stochastic Variance Reduced Gradient (SVRG) [5].

In 2008 Carpenter described an idea for performing lazy updates for stochastic gradient descent [2]. In his method, they maintain a vector $\boldsymbol{\psi} \in \mathbb{N}^d$, where each ψ_i stores the index of the last epoch in which each weight was last regularized. They then perform periodic batch updates. However, as they acknowledge, the system they describe results in updates that do not produce the same result as applying an update after each time step.

Langford et al. concurrently developed a similar approach for lazily updating ℓ_1 regularized linear models [6]. They restrict their attention to ℓ_1 models. Additionally, they only describe the closed form update when the learning rate η is constant, although they do suggest that an update can be derived when η_i decays as i grows large. We extend this work, deriving the closed form updates for ℓ_2^2, and elastic net regularization. Our algorithms work for both fixed and varying learning rates.

In 2009, Duchi and Singer describe a method they call Forward Backward Splitting (FoBoS) for online optimization of a regularized loss function [4]. Here, after each gradient step to minimize the loss function, the weight is regularized by solving a convex optimization problem. They derive regularization updates for ℓ_1, ℓ_2^2, ℓ_2^2, and ℓ_∞ norms. Further, they share the insight of applying lazy updates, when training on sparse high dimensional data. Their lazy updates hold for all norms ℓ_q for $q \in 1, 2, \infty$, However they do not hold for the commonly used ℓ_2^2 norm. Consequently it also does not hold for mixed norms involving the ℓ_2^2 norm such as the widely used elastic net $(\ell_1 + \ell_2^2)$.

5 Closed Form Lazy Updates for SGD

In this section, we will derive constant-time stochastic gradient updates when processing examples from sparse datasets. Using these, the lazy update algorithm can train linear models with time complexity $O(p)$. For brevity, we will describe the more general case where the learning rate is varied. When the learning rate is constant the algorithm can be easily modified to have $O(1)$ space complexity.

5.1 Lazy SGD Update ℓ_1 Regularization with Attenuated Learning Rate

The closed form update for ℓ_1 regularized models was derived by Singer and Duchi [10]. The constant time lazy update can be applied with:

$$w_j^{(k)} = \text{sgn}(w_j^{(\psi_j)}) \left[|w_j^{(\psi_j)}| - \lambda_1 \left(S(k-1) - S(\psi_j - 1) \right) \right]_+ . \tag{4}$$

where $S(t)$ is a function that returns the partial sum $\sum_{\tau=0}^{t} \eta^{(\tau)}$.

The sum $\sum_{\tau=t}^{t+n-1} \eta^{(\tau)}$ can be computed in constant time, using a dynamic programming scheme. On each iteration t, we compute $S(t)$ in constant time given its predecessor as $S(t) = \eta^{(t)} + S(t-1)$. The base case for this recursion is $S(0) = \eta^{(0)}$. We then cache this value in an array for subsequent constant time lookup.

When the learning rate decays with $1/t$, the terms $\eta^{(\tau)}$ follow the harmonic series, and we do not require dynamic programming to compute the lazy update for $\ell_1 regularization$. Each partial sum of the harmonic series is a harmonic number $H(t) = \sum_{i=1}^{t} \frac{1}{t}$. Clearly $\sum_{\tau=t}^{t+n-1} \eta^{(\tau)} = \eta^{(0)} \left(H(t+n) - H(t) \right)$, where H_τ is the τ_{th} harmonic number. While there is no instantly computable analytic expression to calculate the τ_{th} harmonic number, there exist good approximations.

The $O(T)$ space complexity of this algorithm may seem problematic. However, this easily dealt with by bringing all weights current after each epoch. The cost to do this is amortized across all iterations and is thus negligible.

5.2 Lazy SGD Update for ℓ_2^2 Regularization with Attenuated Learning Rate

We now consider the case of ℓ_2^2 regression with attenuated learning rate. For a given example $\boldsymbol{x}_i$, if a feature $x_{ij} = 0$ and the learning rate is varying, the stochastic gradient update rule for an ℓ_2^2 regularized objective is

$$w_j^{(t+1)} = w_j^{(t)} - \eta^{(t)} \lambda_2 \cdot w_j^{(t)} \tag{5}$$

Considering successive updates, the decreasing learning rate prevents a collecting lazy updates as terms in a geometric series as we could when if learning rate was fixed. However, we can employ a dynamic programming strategy, similar to the system we demonstrated for ℓ_1 with attenuated learning rate.

Lemma 1. *For SGD with ℓ_2^2 regularization, the constant time lazy update to bring a weight current from iteration ψ_j to k is given by*

$$w_j^{(k)} = w_j^{(\psi_j)} \frac{P(k-1)}{P(\psi_j - 1)} \tag{6}$$

where P(t) is the partial product $\prod_{\tau=0}^{t}(1 - \eta^{(\tau)}\lambda_2)$.

Proof. Rewriting the above expressions for lazy updates:

$$\begin{aligned} w_j^{(t+1)} &= w_j^{(t)}(1 - \eta^{(t)}\lambda_2) \\ w_j^{(t+n)} &= w_j^{(t)}(1 - \eta^{(t)}\lambda_2)(1 - \eta^{(t+1)}\lambda_2) \cdot ... \cdot (1 - \eta^{(t+n-1)}\lambda_2) \end{aligned} \tag{7}$$

The products $P(t) = \prod_{\tau=0}^{t}(1 - \eta^{(\tau)}\lambda_2)$ can be cached on each iteration in constant time using the recursive relation

$$P(t) = (1 - \eta^{(t)}\lambda_2)P(t-1)$$

The base case is $P(0) = a_0 = (1 - \eta_0\lambda_2)$. Given cached values $P(0), ..., P(t+n)$, it is then easy to calculate the exact lazy update in constant time:

$$w_j^{(t+n)} = w_j^{(t)} \frac{P(t+n-1)}{P(t-1)}. \tag{8}$$

Our claim follows.

□

As in the case of ℓ_2^2 regularization with fixed learning rate, we needn't worry that the regularization update will flip the sign of our weight w_j, owing to $\forall t \geq 0, P(t) > 0$.

5.3 Lazy SGD Update for Elastic Net Regularization with Attenuated Learning Rate

Finally, we derive the constant time lazy update for SGD with elastic net regularization. recalling that a model regularized by elastic net has an objective function of the form

$$F(w) = \mathcal{L}(w) + \lambda_1 ||w||_1 + \frac{\lambda_2}{2} ||w||_2^2.$$

When a feature $x_j = 0$, the SGD update rule is

$$\begin{aligned} w_j^{(t+1)} &= \text{sgn}(w_j^{(t)}) \left[|w_j^{(t)}| - \eta^{(t)}\lambda_1 - \eta^{(t)}\lambda_2 |w_j^{(t)}| \right]_+ \\ &= \text{sgn}(w_j^{(t)}) \left[(1 - \eta^{(t)}\lambda_2)|w_j^{(t)}| - \eta^{(t)}\lambda_1 \right]_+ \end{aligned} \tag{9}$$

Theorem 1. *To bring a weight $w_j^{\psi_j}$ current to w_j^k using update (Equation 9), the constant time update is*

$$w_j^{(k)} = \text{sgn}(w_j^{\psi_j}) \left[|w_j|^{\psi_j} \frac{P(k-1)}{P(\psi_j - 1)} - P(k-1) \cdot (B(k-1) - B(\psi_j - 1)) \right]_+ \tag{10}$$

Proof. The attenuated learning rate prevents us from working out a simple expansion like **??**. Instead, we can write the following inductive expressions for a few consecutive terms in the sequence

$$w_j^{(t+1)} = \text{sgn}(w_j^{(t)}) \left[(1 - \eta^{(t)}\lambda_2)|w_j^{(t)}| - \eta^{(t)}\lambda_1 \right]_+ \tag{11}$$

Substituting $a_\tau = (1 - \eta^{(\tau)}\lambda_2)$ and $b_\tau = -\eta^{(\tau)}\lambda_1$ gives

$$\begin{aligned} w^{(t+1)} &= \text{sgn}(w_j^{(t)}) \left[a_t |w^{(t)}| + b_t \right]_+ \\ &\dots \\ w^{(t+n)} &= \text{sgn}(w_j^{(t)}) \left[a_{(t+n-1)}(\dots a_{(t+1)} \left(a_t w^t - b_t \right) - b_{(t+1)} \dots) - b_{(t+n-1)} \right]_+ \\ &= \text{sgn}(w_j^{(t)}) \left[|w_j^{(t)}| \prod_{\tau=t}^{t+n-1} a_\tau + \sum_{\tau=t}^{t+n-2} b_i \left(\prod_{q=\tau}^{t+n-2} a_q \right) + b_{(t+n-1)} \right]_+ \end{aligned} \tag{12}$$

The leftmost term, $\prod_{\tau=t}^{t+n-1} a_\tau$ can be calculated in constant time as $P(t+n-1)/P(t-1)$ using cached values from the dynamic programming scheme discussed in the previous section.

To cache the remaining terms, we group the center and right-most terms and apply the following simplifications

$$\begin{aligned} &\sum_{\tau=t}^{t+n-2} b_i \left(\prod_{q=\tau}^{t+n-2} a_q \right) + b_{t+n-1} \\ &= b_t \frac{P(t+n-2)}{P(t-1)} + b_{t+1} \frac{P(t+n-2)}{P(t)} + \dots + b_{t+n-1} \frac{P(t+n-2)}{P(t+n-2)} \\ &= -\lambda_1 P(t+n-2) \left(\frac{\eta^{(t)}}{P(t-1)} + \frac{\eta^{(t+1)}}{P(t)} + \dots + \frac{\eta^{(t+n-1)}}{P(t+n-2)} \right) \end{aligned} \tag{13}$$

We now add a new layer to our dynamic programming formulation. In addition to pre-calculating all values $P(t)$ as we go, we define a partial sum over inverses of partial products

$$B(t) = \sum_{\tau=0}^{t} \frac{\eta^{(\tau)}}{P(\tau-1)}.$$

Given that $P(t-1)$ can be accessed in $O(1)$ at time t, $B(t)$ can now be cached in constant time. The dynamic programming here depends upon the simple recurrence relation

$$B(t) = B(t-1) + \frac{\eta^{(t)}}{P(t-1)}$$

with the base case $B(-1) = 0$ Thus $B(0) = 0 + \eta^{(0)}/1 = \eta^{(0)}$.

Thus, for SGD elastic net with heuristic clipping and attenuated learning rate, the update rule to lazily apply any number n of successive regularization updates in constant time to weight w_j is:

$$w^{(t+n)} = \text{sgn}(w_j^{(t)}) \left[|w_j^{(t)}| \frac{P(t+n-1)}{P(t-1)} - \lambda_1 P(t+n-1) \left(B(t+n-1) - B(t-1) \right) \right]_+ \tag{14}$$

□

6 Lazy Updates for Forward Backward Splitting

Recall the optimization problems for ℓ_1, ℓ_2^2 and elastic net regularization (Section 2.3). They also described the lazy update for ℓ_1 regularization which is the same as the truncated gradient update in (Equation 4). Thus we turn our attention to FoBoS updates for ℓ_2^2 and elastic net regularization.

6.1 Lazy FoBoS Update for ℓ_2^2 Regularization

For ℓ_2^2 regularization, to apply the regularization update we solve the problem from Equation 3 with λ_1 set to 0. Solving for $\boldsymbol{w}^*$ gives the simple update $w_j^{(t+1)} = w_j^{(t)}/(1+\eta^{(t)}\lambda_2)$ whenever $x_{ij} = 0$. Note that this differs from the standard stochastic gradient descent step.

We can store the values $P'(t) = \prod_{\tau=0}^{t} \frac{1}{1+\eta^t \lambda_2}$. Thus constant time lazy update for FoBoS with ℓ_2^2 regularization to bring a weight current at time k from time ψ_j is

$$w_j^k = w_j^{(\psi_j)} \frac{P'(k-1)}{P'(\psi_j - 1)} \tag{15}$$

where $P'(t) = (1+\eta^{(t)}\lambda_2)^1 \cdot P'(t-1)$ with base case $P'(-1) = 1$.

6.2 Lazy FoboS Update for Elastic Net Regularization

Finally, in the case of elastic net regularization via forward backward splitting, we solve the convex optimization problem from Equation 3. This problem also comes apart and can be optimized for each w_j separately. Setting the derivative wrt w_j yields the following optimal solution:

$$w_j^{(t+1)} = \text{sgn}(w_j^{(t)}) \left[\frac{|w_j^{(t)}| - \eta^{(t)}\lambda_1}{\eta^t \lambda_2 + 1} \right]_+$$

Theorem 2. *The exact constant time lazy update for FoBoS with elastic net regularization and attenuated learning rate to bring a weight current at time k from time ψ_j is*

$$w_j^{(k)} = \text{sgn}(w_j^{(\psi_j)}) \left[|w_j^{(\psi_j)}| \frac{P'(k-1)}{P'(\psi_j - 1)} - P'(k-1) \cdot \lambda_1 \left(B'(k-1) - B'(\psi_j - 1) \right) \right]_+ \tag{16}$$

where $B'(t) = B'(t-1) + \frac{\eta^{(t)}}{P(t-1)}$ with base cases $B'(0) = \eta^{(0)}$ and $B'(-1) = 0$.

Proof. We substitute $a_t = (\eta^{(t)}\lambda_2 + 1)^{-1}$ and $b_t = -\eta^{(t)}\lambda_1$. Note that neither a_t nor b_t depends upon w_j. Consider successive updates:

$$\begin{aligned} w_j^{t+1} &= \text{sgn}(w_j^t) \left[a_t(|w_j^{(t)}| + b_t) \right]_+ \\ w_j^{(t+n)} &= \text{sgn}(w_j^t) \left[|w_j^t| \prod_{\beta=t}^{t+n-1} a_\beta + \sum_{\tau=t}^{t+n-1} \left(b_\tau \prod_{\alpha=\tau}^{t+n-1} a_\alpha \right) \right]_+ . \end{aligned} \tag{17}$$

In the first term, $\frac{P'(t+n-1)}{P'(t-1)}$ can be substituted for $\prod_{\beta=t}^{t+n-1} a_\beta$. The second term can be expanded

$$\begin{aligned} \sum_{\tau=t}^{t+n-1} \left(b_\tau \prod_{\alpha=\tau}^{t+n-1} a_\alpha \right) &= \sum_{\tau=t}^{t+n-1} \left(b_\tau \frac{P(t+n-1)}{P(\tau-1)} \right) \\ &= -P(t+n-1) \cdot \lambda_1 \sum_{\tau=t}^{t+n-1} \left(\frac{\eta^{(\tau)}}{P(\tau-1)} \right) \end{aligned} \tag{18}$$

Using the dynamic programming approach, for each iteration t, we can calculate

$$B'(t) = B'(t-1) + \frac{\eta^{(t)}}{P(t-1)}$$

with the base cases $B'(0) = \eta^{(0)}$ and $B'(-1) = 0$. Thus

$$w_j^{(t+n)} = \text{sgn}(w_j^t) \left[|w_j^t| \frac{P'(t+n-1)}{P'(t-1)} - P'(t+n-1) \cdot \lambda_1 \left(B'(t+n-1) - B'(t-1) \right) \right]_+ \tag{19}$$

□

7 Experiments

We do not need justify the usefulness of logistic regression with elastic net regularization. However, to confirm the correctness and speed of our dynamic programming solution, we tested the system on a bag-of-words representation of abstracts from biomedical articles indexed in Medline. Our dataset contained $1,000,000$ examples, 260,941 features and 88.54 nonzero features per document

We prototyped the system in Python. Sparse datasets are represented as lists of dictionaries, where keys correspond to feature indices and values correspond to word counts. We implemented both standard and lazy FoBoS for logistic regression regularized with elastic net. We confirmed on a synthetic dataset that the standard FoBoS updates and lazy updates output identical weights up to 4 significant figures. To make a fair comparison, both systems exploit sparsity when calculating predictions.

Lazy update FoBos performed 612.2 times faster than FoBoS with dense regularization updates. It is important to consider that a pure speedup proportional to the number of zeros to non-zeros in the training data would be $2947.1528\times$. Our additional calculations provide a constant factor slowdown. But overall, when considerably less than one quarter of all features have nonzero values, our algorithm provides a tremendous speedup. While our dynamic programming strategy consumes space linear in the number of iterations, it does not present a significant time penalty. Further, when training one epoch, storing two floating point numbers per example is a modest use of space compared to the data itself.

FoBos Elastic Net w/ Lazy Updates	FoBos Elastic Net w/ Dense Updates
1893 Examples / Second	3.086 Examples / Second

8 Discussion

Many interesting datasets are high-dimensional, and many high-dimensional datasets are sparse. To be fast, algorithms should scale with the number of non-zeros per example, not the nominal dimensionality. We have demonstrated an algorithm for fast learning on linear models with ℓ_2^2 and elastic net regularization, verifying its correctness analytically and performance empirically.

Acknowledgments

This research was conducted with generous support from the Division of Biomedical Informatics at the University of California, San Diego, which has funded the first author via a training grant from the National Library of Medicine.Galen Andrew began evaluating lazy updates for multilabel classification with Charles Elkan in the summer of 2014. His notes provided an insightful starting point for this research. Sharad Vikram provided invaluable help in checking the derivations of closed form updates.

References

[1] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[2] Bob Carpenter. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. *Alias-i, Inc., Tech. Rep*, pages 1–20, 2008.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[4] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

[5] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[6] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. In *Advances in neural information processing systems*, pages 905–912, 2009.

[7] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM, 2014.

[8] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):123–231, 2013.

[9] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

[10] Yoram Singer and John C Duchi. Efficient learning using forward-backward splitting. In *Advances in Neural Information Processing Systems*, pages 495–503, 2009.

[11] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[12] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.