Efficient, Scalable L_1-Regularized Logistic Regression

Andrea Klein ANDREAKL@ANDREW.CMU.EDU
Dipan Kumar Pal DIPANP@ANDREW.CMU.EDU

Abstract

We implement a scalable version of L_1 regularized logistic regression designed for use on large, high-dimensional datasets. The primary barrier to scalability relative to the number of features is the need to solve a computationally expensive convex optimization problem. Standard algorithms for this task scale insufficiently well for use with large, real-world datasets. Consequently, we adopt the approach described by Lee et al., who iteratively approximate the objective function via quadratic approximation (Lee et al., 2006). We also empirically explore additional optimizations to improve scalability. Our experiments show that our algorithm can achieve better overall computation time and testing accuracy than the non-parallelized version.

1. Theoretical Background

1.1. A Brief Introduction to Logistic Regression

A quintessential application of supervised machine learning is the classification of N-dimensional feature vectors $x^{(i)} \in R^N$. Here we consider only binary classification, i.e. the assignment of a class label $y^{(i)} \in \{0,1\}$ to each vector. In the logistic regression model, the probability that a vector is a positive instance of a class follows a sigmoid distribution:

$$p(y=1|x,\theta) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

where $\theta \in R^N$ are the model parameters we seek to learn. The maximum a posteriori estimate of these parameters is given by

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

$$\min_\theta \sum_{i=1}^{M} -\log p(y^{(i)}|x^{(i)};\theta) + \beta R(\theta)$$

where M is the total number of training examples and $R(\theta)$ is a regularization term which, for positive values of β, should penalize large values of the model parameters. We adopt L_1 regularization, i.e. $R(\theta) = ||\theta||_1$, which is equivalent to imposing a Laplacian prior $p(\theta) = (\beta/2)^N exp(-\beta||\theta||)$.

In practice, it is generally more convenient to work with the equivalent Lagrangian formulation of this optimization problem,

$$\min_\theta \sum_{i=1}^{M} -\log p(y^{(i)}|x^{(i)};\theta)$$

with the additional requirement that $||\theta||_1 \leq C$ for some fixed constant C.

1.2. Optimizing L_1 Regularized Regression

In either formulation, this optimization problem is resistant to efficient solution by standard means (Lee et al., 2006). In particular, by adding a regularization term to the objective function, we preclude the use of Newton's method, because the objective function is no longer continuously differentiable. It is nonetheless instructive to consider Newton's method, specifically, because the algorithm we ultimately adopt proceeds in an analogous manner.

Newton's method finds stationary points of differentiable functions. Naturally, it has many applications to optimization problems, wherein it is typically necessary to minimize or maximize some objective function. The basic idea is to approximate the function of interest by its second-order Taylor expansion, i.e. a quadratic function, about some point, then take a step toward the root of the quadratic approximation. Repeating this procedure gives a sequence of values that converges toward a stationary point of the actual objective function.

In one dimension, the Taylor expansion f_T to a function f about a new point $x = x_n + \Delta x$, where Δx is the step size, is given by:

$$f_T(x_n + \Delta x) = f(x_n) + f'(x_n)\Delta x + \frac{1}{2} f''(x_n)\Delta x^2$$

This function reaches an extremum with respect to Δx when $f'(x_n) + f''(x_n)\Delta x = 0$, i.e. when the step size satisfies $\Delta x = -\frac{f'(x_n)}{f''(x_n)}$. Hence we derive the following update rule:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

In higher dimensions, the Taylor approximation is given by

$$f_T(x + \Delta x) \approx f(x) + J(x)\Delta x + \frac{1}{2}\Delta x^T H(x)\Delta x$$

where J is the Jacobian matrix and H is the Hessian matrix, so the iterative rule generalizes to

$$x_{n+1} = x_n - [Hf(x_n)]^{-1} f(x_n)$$

with the gradient playing the role of the derivative and the inverse of the Hessian matrix H replacing the reciprocal of the second derivative.

Now recall the optimization we wish to perform in order to determine the parameters θ of our logistic regression model. In the absence of a regularization term, we'd have a simpler maximum likelihood estimate (MLE) problem to solve, namely:

$$\min_\theta \sum_{i=1}^{M} -\log p(y^{(i)}|x^{(i)})$$

From the current position in parameter space $\theta^{(k)}$, we could compute the step direction $\gamma^{(k)}$ via Newton's method, i.e.

$$\gamma^{(k)} = \theta^{(k)} - H^{-1}(F(\theta^{(k)}))F(\theta^{(k)})$$

where $F(\theta^{(k)})$ is the objective function, evaluated at $\theta^{(k)}$. Adopting a step size parameter t to control the impact of the updates, we could then use the following update rule:

$$\theta^{(k+1)} = (1-t)\theta^{(k)} + t\gamma^{(k)}$$

Unfortunately, life is not so simple for us, because we wish to retain the regularization term. In particular, we cannot compute the Newton step term $\gamma^{(k)}$ using the equation above. Instead, we will show that $\gamma^{(k)}$ can be determined by solving a weighted ordinary least squares problem in a manner that does not break down in the presence of a regularization term.

1.3. Adapting the IRLS Algorithm to L_1 Regularized Regression

The Iteratively Reweighted Least Squares algorithm aims to solve problems where the weights of the problem matrices change at every iteration. It is basically Newton's method applied to maximizing the likelihood function.

First, we will rewrite the Hessian and the gradient in a different form. In order to do so, we'll need to make use of the following definitions:

$$X : X = [x^{(1)}x^{(2)} \ldots x^{(M)}]$$

$$\Lambda : \Lambda_{ii} = \sigma(\theta^{(k)T}x^{(i)})\left[1 - \sigma(\theta^{(k)T})\right]$$

$$z : z_i = x^{(i)T}\theta^{(k)} + \frac{\left[1 - \sigma(y^{(i)}\theta^{(k)T}x^{(i)})\right]y^{(i)}}{\Lambda_{ii}}$$

Here, X is the so-called design matrix, Λ is a diagonal matrix, and z is a vector. It can be shown that $H(\theta^{(k)}) = -X\Lambda X^T$ and $(\theta^{(k)}) = X\Lambda(z - X^T\theta^{(k)})$ - see e.g. (Minka, 2003), (Lee et al., 2006). As a result, we have

$$\gamma^{(k)} = (X\Lambda X^T)^{-1}X\Lambda z$$

so we can interpret $\gamma^{(k)}$ as the solution to the following weighted least squares problem:

$$\gamma^{(k)} = \text{argmin}_\gamma ||(\Lambda^{\frac{1}{2}}X^T)\gamma - \Lambda^{\frac{1}{2}}z||_2^2$$

In order to incorporate L_1 regularization, we simply impose the additional constraint that $||\gamma||_1 \leq C$. This formulation can also be written as the well known LASSO problem, for which the Least Angle Regression or LARS is known to be a very efficient solver.

Algorithm 1 IRLS-LARS

```
Initialize $\theta = 0$.
for $k = 1$ to $MaxIterations$ do
    Compute $\Lambda$ and $z$
    Solve for $\gamma$ using LARS
    Using a backtracking line-search, find the value of
    $t$ that minimizes the objective function
    Let $\theta^{(k+1)} = (1-t)\theta^{(k)} + t\gamma^{(k)}$
    Evaluate the objective function at $\theta^{(k+1)}$
    if stopping criterium satisfied then
        Break
    end if
end for
```

2. Algorithm

2.1. L_1-adapted IRLS

L_1-adapted IRLS forms the backbone of our algorithm. The complete IRLS-LARS algorithm is described in pseudocode as Algorithm 1.

Recall that the objective function is given by:

$$\sum_{i=1}^{M} -\log p(y^{(i)}|x^{(i)};\theta)$$

In our implementation, we use the number of iterations as the stopping criterion. However, Lee et al. use the difference between the optimal value of the objective function and the current estimate as their stopping criteria (Lee et al., 2006). They estimate the optimal value by letting the IRLS-LARS routine run for a sufficiently long time. Since our focus is on scalability, we limit our experiments to a maximum number of iterations.

This algorithm is provably guaranteed to converge to a global optimum (Lee et al., 2006).

2.2. The LARS Subroutine

To solve the reweighted least squares subproblem, we use Least Angle Regression (LARS). The LARS solution consists of a curve with a solution for each value of the L_1 constraint of the parameter vector. We thus solve LARS only up to the L_1 constraint. Since LARS has the computational advantage of returning the entire path of coefficients, it is very helpful in cross-validation. However, LARS suffers like any other solver from the problem of noisy features, which can have an appreciable impact in its performance.

LARS also suffers when there exist many correlated features in very high dimensional problems. However, we do not address this problem directly, leaving its investigation to future work. A temporary solution would be to project the data onto a subspace, using techniques such as PCA. However, doing so would be a very expensive process for very large datasets.

Algorithm 2 Parallel SGD$(c^{\cdot}..., c^{m}), T, \eta, w_0, k$

```
for all $i \in 1...k$ parallel do do
    $v_i$ = SGD$(c^{\cdot}..., c^{m}), T, \eta, w_0)$ on client
end for
Aggregate from all machines $v = 1/k \sum_{i=1}^{k} v_i$
```

2.3. Parallelizing IRLS-LARS

Zinkevich, et al. investigate a very simple idea regarding the parallelization of L_2-regularized stochastic gradient descent (Zinkevich et al., 2010). They split the training data into mutiple shards and then train a different model on each shard on a separate machine. The final model is given by the average of the model parameters from all the machines. They show that as the number of machines increases, the difference in the objective function between the averaged model and the model trained on the entire training set decreases. However, they formally prove convergence bounds only for strongly convex functions. We empirically investigate whether this parallelization strategy also works well for the L_1 norm, which is not strongly convex. We present the basic algorithm that Zinkevich, et al. introduce as Algorithm 2.

We follow their approach of averaging the ensemble of models learned on each machine. However, we use IRLS-LARS to learn the models themselves. Lee, et al. show that IRLS-LARS performs well relative to other LASSO solvers for small to medium sized datasets (Lee et al., 2006). We show that IRLS-LARS scales efficiently as we increase the number of shards. We also explore how the runtime scales with the size of the training data and the L_1 constraint.

An advantage of this parallelization approach is that the machines need not interact with one another. Although we do not make use of MapReduce in our implementation, we note that this framework is highly amenable to MapReduce, since the reduce step would simply involve averaging the models.

Zinkevich, et al. prove convergence bounds only for strongly convex functions(Zinkevich et al., 2010). One of the key steps of their proof is to show that the L_2 regularized stochastic gradient weight update rule is a contraction in a complete metric space. A pivotal step in the extension of the proof to general convex functions would be to show that weight update at each iteration for the L_1 regularized logistic regression is a contraction. We leave this to be explored in a later study.

We now describe some of the experiments that we use to characterize the performance of our parallelization framework.

3. Experiments and Results

3.1. Overview of Datasets

The algorithm we describe may be applied to a wide variety of classification problems. For the purposes of testing our classifier, we restrict our attention to the task of identifying whether a given image contains a face. We use a dataset containing 11,000 grayscale images compiled from FDDB (Jain & Learned-Miller, 2010). The dataset includes non-face images which were taken from the backgrounds of those images with faces. The number of faces and non-faces in the dataset are close to equal. Each image in the dataset consists of 24x24 pixels.

As a follow-up, we also work with a larger dataset compiled from the AFLW database (Koestinger et al., 2011) for faces and the MIT (Oliva & Torralba, 2001) database for non-faces. This data includes 49,000 images, with an equal distribution of faces and non-faces, also of size 24x24 pixels.

3.2. Performance of standard IRLS-LARS

Here we explore how IRLS-LARS performs as we increase the training size and the L_1 constraint. We train on 10,000 images of the FDDB dataset and use the last 1,000 images as the validation set. The results are summarized in Fig. 1. We can see that as we provide more variance to the model by relaxing the L_1 constraint, the performance increases. However, as we increase training size, the fixed bias degrades performance.

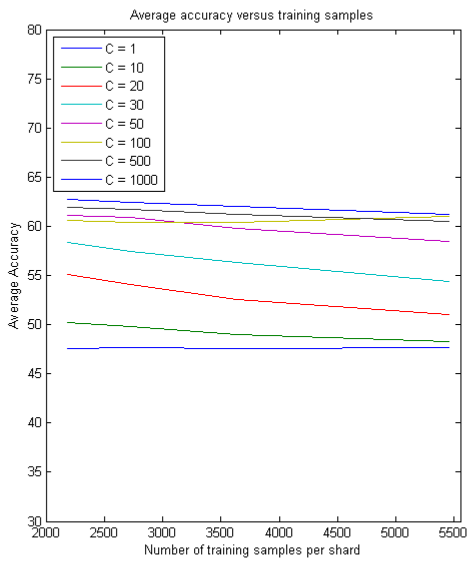

Figure 1. Accuracy of IRLS-LARS vs. training size

3.3. Computational complexity of IRLS-LARS

To understand how IRLS-LARS scales inherently, we set up two simple experiments. We look at wall-clock time as we increase training size and as we increase the L_1 constraint size. For the training size plot, we set the L_1 constraint to 500. The results are summarized in Fig. 2 and Fig. 3.

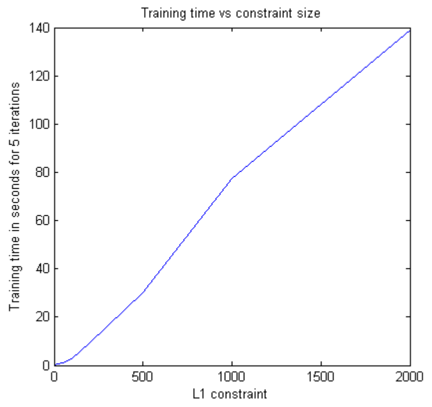

Figure 2. Wall-clock training time vs. training size

We see that IRLS-LARS scales almost linearly. Although this may seem to diminish the need for parallelizing the algorithm, in a later experiment we will explore how IRLS-LARS converges given a particular number of iterations and a constraint size. As we increase the training size, IRLS-LARS requires more iterations to converge, which enables us to gain time

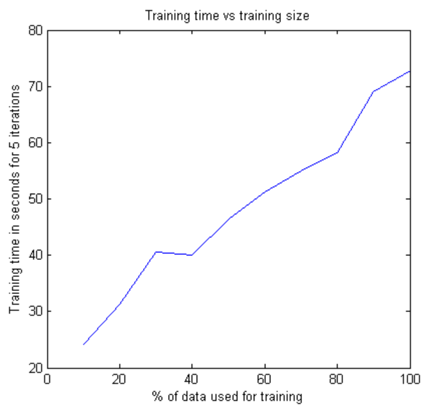

Figure 3. Wall-clock training time vs. constraint size

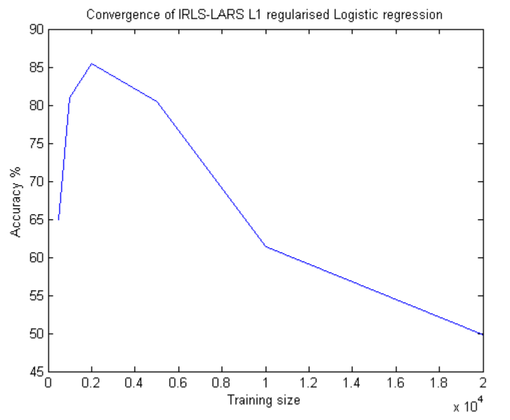

Figure 4. Convergence of IRLS-LARS L_1 regularized logistic regression keeping C = 1000 and 10 iterations

when we parallelize. Also, as we will see, we obtain gains in accuracy as well.

3.4. Convergence of IRLS-LARS

Lee et al. prove that IRLS-LARS is guaranteed to converge to the global optimum of the objective function in a finite number of iterations (Lee et al., 2006). However, they do not explore how many iterations are required for convergence. For a given value of the L_1 constraint, one would expect the number of iterations required for convergence to increase as the training size increases. In this experiment we examine the performance given a fixed number of iterations and the L_1 constraint C. We fix the maximum number of iterations to 10 and C to 1000 and gradually increase the training size.

In a separate experiment, we set the maximum number of iterations to 5 and C to 10,000. We increase the training size gradually. The value of C is chosen with the hope that the model has enough variance to capture the statistics of the data. Fig. 4 and Fig. 5 show the results of these two experiments.

Fig. 5 shows that given enough model variance (high enough C), as we increase training size, the algorithm gets farther from convergence after a fixed number of iterations. This is reflected in its performance which degrades gradually. We make no attempt, however, to find the optimal size of the constraint. It may very well be that we have allowed too much variance. Our intention is to show that upon eliminating high bias, we would expect a decay in performance with respect to training size. The importance of bias in this matter can be more clearly observed in Fig. 4 where we happen to strike a good bias-variance trade off early in the plot.

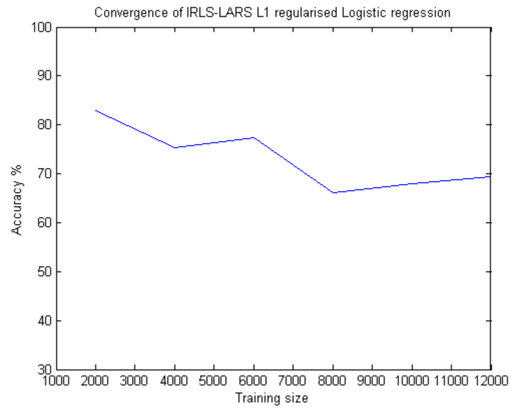

Figure 5. Convergence of IRLS-LARS L_1 regularized logistic regression keeping C = 10000 and 5 iterations

Fig. 4 is more interesting because it shows that for a given value of C, we would expect performance to increase with more data (training size) until the high bias of the constraint starts hindering performance. We see this occuring at the 2,000 mark in the plot. This fact works to our advantage. Since we are looking at scalability, we would typically split the training data into multiple shards. With each shard containing just a fraction of the data, IRLS-LARS requires fewer iterations to converge than if it were run on the entire dataset for a given size of constraint (C). This provides an additional speed-up.

Furthermore, we note that given a smaller shard, a smaller value of C is more likely to capture the desired statistics. A stronger bias in the model helps with generalization when the training set is smaller (Mitchell, 1980). However, this parameter must be determined using cross-validation or problem domain constraints. A smaller C means lesser training time for each shard, resulting in an even faster convergence when measured using a wall-clock.

3.5. Parallelizing L_1 regularized logistic regression using IRLS-LARS

In this experiment, we look at the performance of our implementation of IRLS-LARS in the parallel setting as described in previous sections. We first explore the 11,000 image FDDB dataset. We set aside a validation set of 1,000 images and use the other 10,000 as the training set. We sequentially increase the number of shards and then train multiple models on each shard, averaging them to obtain the final model. Note that for this experiment, the L_1 constraint is limited to 200. The results are shown in Fig. 6.

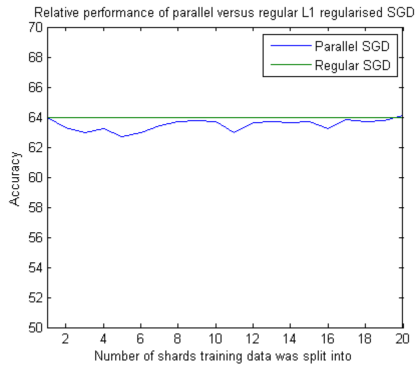

Figure 6. Relative performance of parallel vs. regular L_1 regularised logistic regression

We observe that as we increase the number of shards, i.e. as we increase parallelization, the performance of the parallelized logistic regression approaches that of the regular one. Note that the logistic regression plots are labelled as SGD; however, the algorithm used is IRLS-LARS.

Zinkevich, et al. showed empirically that having more machines reduces the difference between the objective function of the parallelized and the regular SGD (Zinkevich et al., 2010). We saw in the previous section that when using IRLS-LARS, we derive multiple benefits. Not only do we achieve a gain in overall training time through parallelization, we gain in individual training time for each shard as well by allowing for a smaller constraint. Furthermore, having a higher bias (smaller constraint) in the model for the smaller training shards should enable better generalization, resulting in better performance. We conduct the same experiment again but with a different set of parameters. We set the constraint size to 1,000, and we train on 40,000 images with the maximum number of iterations to 10. We use about 9,000 images as a validation set. Fig. 7 showcases the results.

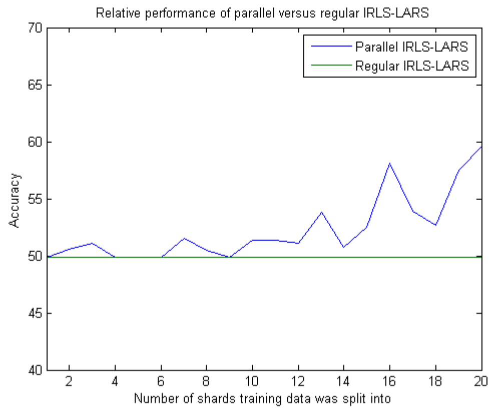

Figure 7. Relative performance of parallel vs regular L_1 regularised logistic regression

Unfortunately, as we observe, the bias is too high to learn directly from the 40,000 training samples in just 10 iterations. The regular logistic regression performs almost as poorly as chance. However, even in this challenging setting, our implementation is able to perform better as the number of shards or machines increases. It seems we achieve better generalization due to high bias coupled with faster convergence for a smaller sized dataset given a fixed number of iterations. This idea is supported by the experiments we performed in the previous subsection.

4. Conclusions

We have introduced a parallelizing framework in the context of L_1 regularised logistic regression using IRLS-LARS. We are able to achieve higher accuracy than sequential logistic regression given a particular constraint size and fixed number of itertions. We also achieve significant gains in training time of the overall model, owing to parallelizing over several machines, allowing for a smaller L_1 constraint size and taking advantage of faster convergence of IRLS-LARS for smaller sized datasets.

Overall, IRLS-LARS seems to be very well suited for parallelization. We emphasize that the framework could easily be implemented as a MapReduce algorithm. In future work, one might also consider developing a theoretical proof for general convex functions in the parallelizing framework described in (Zinkevich et al., 2010).

References

Jain, Vidit and Learned-Miller, Erik. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

Koestinger, Martin, Wohlhart, Paul, Roth, Peter M., and Bischof, Horst. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

Lee, Su-In, Lee, Honglak, Abbeel, Pieter, and Ng, Andrew Y. Efficient L1 Regularized Logistic Regression. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

Minka, Thomas P. A comparison of numerical optimizers for logistic regression. Technical report, 2003.

Mitchell, T. M. The need for biases in learning generalizations. Technical report, Computer Science Department, Rutgers University, New Brunswick, MA, 1980.

Oliva, Aude and Torralba, Antonio. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 2001.

Zinkevich, Martin A., Smola, Alex, Weimer, Markus, and Li, Lihong. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23*, pp. 2595–2603, 2010.