Logistic Regression

Tsung-Yi Lin, Chen-Yu Lee
Department of Electrical and Computer Engineering
University of California, San Diego
{tsl008, chl260}@ucsd.edu
January 24, 2013

Abstract

Logistic regression is a technique to map the input feature to the posterior probability for a binary class. The optimal parameter of regression function is obtained by maximizing log likelihood of training data. In this report, we implement two optimization techniques 1) stochastic gradient decent (SGD); 2) limited-memory BroydenFletcherGoldfarbShanno (L-BFGS) to optimize the log likelihood function. We apply logistic regression on classification problems and reproduce the similar experimental result as [1]. We analyze the convergence of SGD methods and show that SGD has the advantage to converage to a reasonably well estimate when learning rate is carefully tuned.

1 Introduction

Logistic regression is a simple and popular techniques to map input features to posterior probability for a binary class. The posterior probability $p(x|y;\theta)$ is a sigmoid function of training feature vector and label (x,y) and parameter θ. We can obtain optimal parameter θ^* of regression function by maximizing the log likelihood function. Because log likelihood function includes the product of nonlinear sigmoid function under iid assumption, we can only find the optimal parameter with numerical methods. The gradient-basded optimization method is a common tool to approach the optimal parameter with the iterative process. In this report, we implement two gradient-based methods 1) SGD and 2) L-BFGS to solve the maximum likelihood problem. We test our implementation by using the same experimental data and setup as [1] which apply logistic regression for binary classification problem. We obtain very similar experimental results as Figure 3 in [1] with both SGD and L-BFG. We analyze the convergence of SGD by measuring classification accuracy and value of objective function over an epoch. The experimental result suggests when learning rate is carefully tuned, SGD can acheive competitive performance as L-BFGS while only using partial amount of data.

2 Algorithm Design And Analysis

In this section, we introduce the principle of logistic regression. The algorithms of SGD and L-BFGS are introduced to optimize the log likelihood function for optimal parameter θ^* of the regression function.

2.1 Logistic Regression

Given a set of training data with binary class labels $\{x_i, y_i | i = 1, 2, ..., n; x_i \in \mathbb{R}^d; y \in \{-1, 1\}\}$ where x_i is the feature vector and y_i is the label of data i. We define the posterior probability of y_i

given x_i to be a sigmoid function:

$$\begin{aligned} p(y_i|x_i;\theta) &= \frac{1}{1+e^{-y_i<\theta,x_i>}} \\ <\theta, x_i> &= \theta^\top[x_i;1], \theta \in \mathbb{R}^{d+1} \end{aligned} \tag{1}$$

The likelihood of training data is measured by the joint probability of training data $p(y_1,...,y_n,\theta|x_1,...,x_n)$. Assume training data is independent and identically distributed and parameter θ is a random variable with zero mean Gaussian distribution and variance σ^2. The joint probability is the product of posterior probability of each data and the prior probability of parameter θ. The optimal parameter θ^* can be computed by maximizing the joint probability:

$$\begin{aligned} \theta^* &= \arg\max_\theta p(y_1,...,y_n,\theta|x_1,...,x_n) \\ &= \arg\min_\theta -logp(y_1,...,y_n,\theta|x_1,...,x_n) \\ &= \arg\min_\theta -log\prod_{i=1}^{n} p(y_i|x_i)p(\theta) \\ &= \arg\min_\theta -\sum_{i=1}^{n} logp(y_i|x_i;\theta) - logG(\theta,0,\sigma^2) \\ &= \arg\min_\theta \sum_{i=1}^{n} log(1+e^{-y_i<\theta,x_i>}) + \frac{1}{2\sigma^2}||\theta||_2^2 \end{aligned} \tag{2}$$

The σ^2 is a hyperparameter that controls the weight of the regularization term. In the following sections, we will use $C = \frac{1}{2\sigma^2}$ for simplicity.

Note that Equation 2 is a nonlinear function that does not have an analytic solution. In the following section, the gradient-based numerical methods are used to optimize the objective function. To compute the gradient, we need to take derivative of minus log likelihood:

$$\begin{aligned} g(X,Y,\theta) &= \frac{\partial \sum_{i=1}^{n} log(1+e^{-y_i<\theta,x_i>}) + C||\theta||_2^2}{\partial\theta} \\ &= \sum_{i=1}^{n} \frac{-y_i x_i (e^{-y_i<\theta,x_i>})}{1+e^{-y_i<\theta,x_i>}} + 2C\theta \\ &= \sum_{i=1}^{n} \frac{-y_i x_i}{1+e^{y_i<\theta,x_i>}} + 2C\theta \end{aligned} \tag{3}$$

Gradient-based methods are iterative process, which update parameter with the partial derivative of objective function times a proper learning rate λ:

$$\theta_{t+1} = \theta_t - \lambda g(X,Y,\theta_t) \tag{4}$$

In the following sections, we introduce two gradient-based numerical methods SGD and L-BFGS to solve the optimization problem.

2.2 Stochastic Gradient Decent

Stochastic gradient decent (SGD) computes and updates gradient with a randomly selected subset called minibatch from the whole training data. The assumption is the expectation of gradient computed from a minibatch equals to the gradient computed from the whole data. At the extreme case, we only use one training data x_i, y_i to compute the gradient:

$$nE[g(x_i,y_i,\theta)] = g(X,Y,\theta) \tag{5}$$

Because using minibatch and batch learning are equivalent in the probability sense, both methods converge to the same result after enough iterations. We illustrates the SGD in Algorithm 1. Note the complexity of stochastic gradient to update once is $O(kd)$ where k is the size of minibatch and d is the dimension of feature vector. The SGD is extremely useful to deal with huge training data [3] since SGD does not require to compute all training data and is able to do online update by partial information. It is also possible that SGD can save the running time in the case partial data can well represent the whole data. We will discuss about convergence of SGD in Discussion Section.

Algorithm 1: Stochastic gradient decent.

```
Data: Training Data X and Label Y
Result: Optimal Parameter $\theta^*$
$\theta \leftarrow \mathbf{0}$;
$\{x_k, y_k\} \leftarrow$ RandomPartition($X$, $Y$) ;
$Loss$ = LossFunction (X, Y, w);
$LossDifference \leftarrow Loss$ ;
while $LossDifference > th$ do
    foreach $x_k$, $y_k$ do
        $\mathbf{g} \leftarrow$ Gradient($x_k$, $y_k$, $\theta$) ;
        $\theta \leftarrow \theta - \lambda\, \mathbf{g}$
    end
    $CurrentLoss$ = LossFunction (X, Y, w);
    $LossDifference \leftarrow CurrentLoss - Loss$ ;
    $Loss \leftarrow CurrentLoss$ ;
end
$\theta^* \leftarrow \theta$
```

2.3 L-BFGS

L-BFGS is a quasi-Newton method that approximates Hessian with few data. [1] The most distinctive part between Newton method and SGD is that SGD requires learning rate decided before training but Newton's method estimates the update by multiplying inverse of Hessian matrix and gradient.

3 Experimental Results

This section illustrate the experimental design and setup to reproduce the results in [1], show the performance, and analyze the characteristic of convergence.

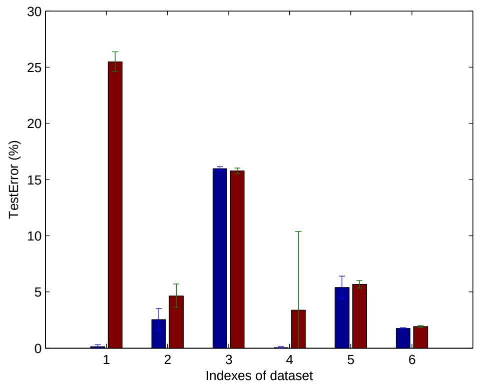
3.1 Experimental Design and Setup

We design two experiments in this report. First, we reproduce the experimental result from [1]. We use the same datasets (Long-Servedio, Mease-Wyner, Adult, Mushroom, USPS-N, and Web) to evaluate performance of SGD and L-BFGS. To test the robustness of the algorithm, label noise is added by randomly choosing 10% of the labels by flipping them in the training set and each dataset is tested with and without label noise. Follow the experimental design in [1], 30% of data is randomly selected as validation set and the rest of the 70% data is used for 10-hold cross validation. The optimal learning rate λ^* and regularization weight $C*$ are chosen by performing a grid search over the parameter space $\{2^{-10,-9,\ldots,10}\}$. For SGD the iterative gradient update stops when the change of the objective function value is less than 10^{-4} or the number of epoch is larger than 1000. The code is implemented in Matlab. We use Mark Schmidt's minFunc [2] to implement L-BFGS algorithm.

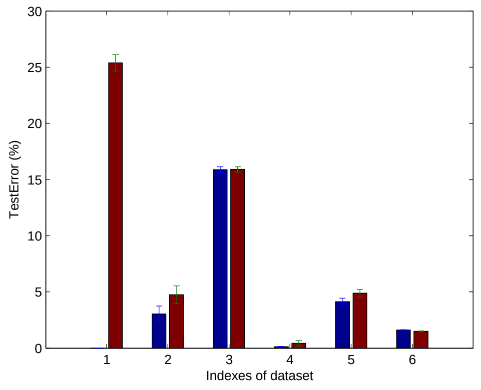
The second experiment tests the robustness of the change of hyperparameters. We analyze the convergence behavior of SGD over an epoch with varying size of minibatch and the learning rate. We follow the same gird search procedure for fixed parameters. The Adult dataset is used for evaluation.

The SGD algorithm has some hyperparameter that we need to set before training. First, we randomly partition training data into minibatches for computing the gradient by using random permutation of the indexes. The same set of minibatches repeat for computation after running an epoch . Second, a minibatch of size 100 is used for each updating iteration in order to overcome the noise in the training set. Third, we find the optimal learning rate λ^* by optimization 10% with different learning rate and pick the best one that converges to the lowest objective function value. The optimal λ^* is used while training on the whole training set. We show the optimal learning rate λ^* and weight of regularization C^* for each dataset at Table 1 for the reference and the reproduction of the experiment.

[1] http://en.wikipedia.org/wiki/Limited-memory_BFGS



(a) SGD.



(b) LBFGS.

Figure 1: The classification error rate of 1a SGD and 1b L-BFGS on six datasets(1 to 6: Long-Servedio, Mease-Wyner, Adult, Mushroom, USPS-N, and Web). The blue bar shows the missing rate with clean label and red bar shows the missing rate with 10% label noise. Two experiments have the same parameter setting except SGD has additional learning rate to tune.

3.2 Classification Performance

The logistic regression can simply apply for binary class classification by thresholding the posterior probabiliy:

$$\begin{aligned} &y = 1, if\ p(y = 1|x; \theta^*) > th \\ &y = 0, otherwise \end{aligned} \tag{6}$$

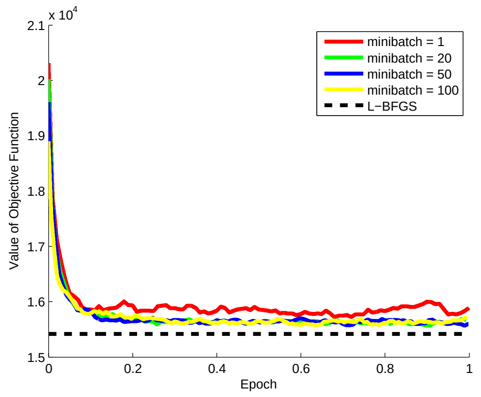
The classification threshold th is determined from cross validation. The 1 shows the error rate using SGD and L-BFGS. Since two optimization methods have exactly same objective function, the classification performances are very similar among these two methods. The SGD method is slightly less stable.

In our experiment, we reproduce the result from [1]. Figure 1 shows the testing error using SGD and L-BFGS. The trend and values of testing error follow Figure 3 in the reference paper which suggests our implementation and experiment setting is reasonable. Note that SGD and L-BFGS produce similar testing error rates because both algorithms share the exactly same convex objective function and try to approach the same global minimum. The performane difference explains that the learning algorithm based on convex loss function are not robust to label noise [1] especially for the synthetic dataset Long-Servedio.

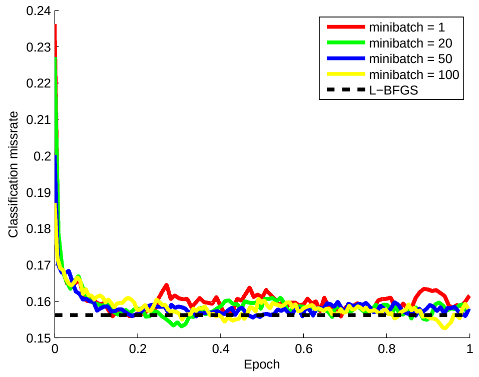
3.3 Convergence Analysis

We analyze the convergence behavior of SGD. We evenly sample 100 time interval during an epoch and output the value of objective function and classification miss rate which are evaluated with the whole training data. We test the robustness for the change of minibatch size. The final outcome of L-BFGS is presented as the reference to indicate the performance of SGD. Figure 2a and 2b show the objective function values drop down at the similar rate regardless the size of minibatch which suggests SGD is not sensitive to minibatch size. The final outcome of L-BFGS outperfroms SGD a little bit after an epoch. However, Figure 2b shows the classification miss rate drops down to competitive performance at 0.2 epoch. The fast convergence of classification suggests that SGD can greatly speed up the process with a marginal decrease of performance. We also find the miss rate of SGD is very close to L-BFGS while the objective function value has a margin. This result indicates that we do not need to optimize objective too hard since the learned model is able to be generalized for validate set well when objective function starts to converge.

Figure 3 shows the convergence behavior of SGD under varying learning rate around the optimal learning rate λ^*. The SGD can never converge (yellow line in Figure 3) if the learning rate is too high. It could also converge too slow (red line in Figure 3) if learning rate is low. The experiment

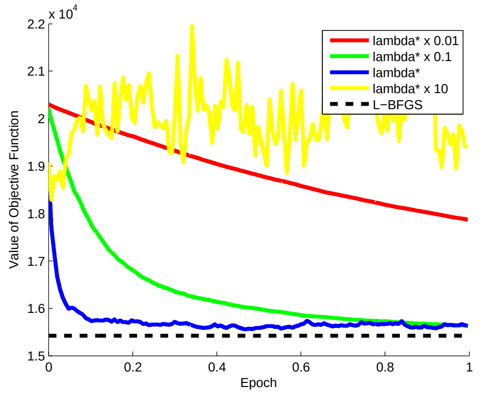


(a) Objective function value.

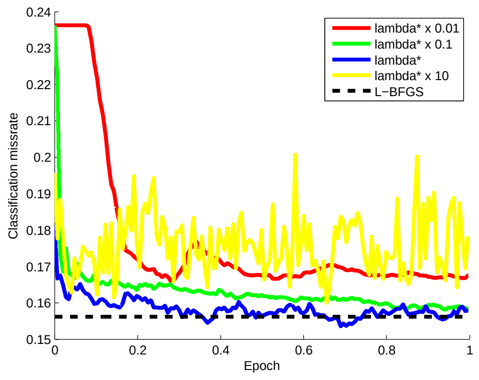


(b) Miss rate.

Figure 2: The convergence behavior of SGD. Figure 2a is the objective function value and 2b is the classification miss rate over an epoch on Adult dataset. The color lines show the convergence with different size of minibatch. Note that we adjust the optimal learning rate with the size of minibatch. The black dash line shows the final outcome of L-BFGS.



(a) Objective function value.



(b) Miss rate.

Figure 3: The convergence behavior of SGD under different learning rate λ and λ^* is the optimal learning rate. Figure 2a is the objective function value and 2b is the classification miss rate over an epoch on Adult dataset. The color lines show the convergence with different learning rates that are around optimal learning rate λ^*. The black dash line shows the final outcome of L-BFGS.

suggests SGD is quite sensitive to learning rate and the working parameter space is within a small region that span order of 2 parameter space.

4 Discussion

In this report, we reproduce the result for logistic regression in Figure 3 of the paper [1] and also analyze the characteristic of convergence of SGD. We find the speed of convergence highly depends on the learning rate λ. If λ is too high SGD would take a long time or never converge to the global optimal value. In this case, it would be safer to conservatively set the learning rate in SGD smaller than regular Gradient Descend algorithm. Second, it is not necessary to go through many iterations in SGD due to the diminishing returns. Figure 2 and 3 show that both algorithms could achieve the accuracy that close to the optimal accuracy within an epoch.

Table 1: Datasets used in our experiment. λ and C are selected by grid search.

Name	Noise	Dimensions	Num. of examples	λ	C
Long-Servedio	0%	21	2000	2^{-20}	2^{-5}
Mease-Wyner	0%	20	2000	2^{-20}	2^{8}
Adult	0%	123	8124	2^{-20}	2^{-3}
Mushroom	0%	112	11000	2^{-12}	2^{-2}
USPS-N	0%	256	48842	2^{-10}	2^{-4}
Web	0%	300	64700	2^{-25}	2^{7}
Long-Servedio	10%	21	2000	2^{-20}	2^{2}
Mease-Wyner	10%	20	2000	2^{-15}	2^{3}
Adult	10%	123	8124	2^{-10}	2^{-3}
Mushroom	10%	112	11000	2^{-12}	2^{-2}
USPS-N	10%	256	48842	2^{-10}	2^{-3}
Web	10%	300	64700	2^{-15}	2^{1}

References

[1] N. Ding and S. V. N. Vishwanathan. t-logistic regression. In *NIPS'10*, 2010.

[2] M. Schmidt. http://www.di.ens.fr/ mschmidt/software/minfunc.html.

[3] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML'04*, 2004.