Logistic Regression with L_2 Regularization

Sandy Wiraatmadja
swiraatm@eng.ucsd.edu

Qiheng Wang
qiw018@cs.ucsd.edu

Abstract

In this paper, we explore logistic regression with L_2 regularization as a binary classification learning model. Given a set of data with different feature values, we want to be able to closely predict what the binary label is for each example. This can be done by training the model on a training set to find the parameters that maximize its log conditional likelihood (LCL). Two optimization methods are analyzed in this paper: Stochastic Gradient Descent (SGD) and Limited-memory BFGS (L-BFGS). The two methods are applied to the Gender Recognition [DCT] dataset from MLcomp [1]. Logistic regression with SGD produces test accuracy averaging at 0.9164, which is just a little higher than L-BFGS test accuracy with an average of 0.9123.

1 Introduction

Machine learning, which is a branch of artificial intelligence, has been growing significantly due to the availability of massive data that can be used in developing and training models. One common problem in machine learning is to train a model that can be used for binary statistical classification. For example, given a data of email messages, some classified as spam and some as non-spam, a model can be trained to learn some distinguishing features between spam and non-spam emails. After the learning process is done and the parameters are chosen, the model can be used on a new data of email messages to label each email as either spam or non-spam. Several classifier learning algorithms have been developed throughout the years, such as k-nearest neighbors, linear regression, or support vector machines [2]. One particular algorithm that we focus on in this paper is logistic regression.

Logistic regression model is widely used in probabilistic classification by fitting the training data to the model and find the parameters that maximizes the log joint conditional likelihood (LCL) of the training set. In this paper, we analyze how to train a logistic regression model for our binary (Bernoulli) label classification problem with two different gradient-based optimization methods for maximizing the LCL. These two algorithms are Stochastic Gradient Descent (SGD) which uses a modified version of a gradient descent method, and Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) which uses the quasi-Newton methods [3]. We use our own implementation of the SGD algorithm in Matlab, while on the other hand, we used Mark Schmidt's Matlab `minFunc` function which implemented L-BFGS [4]. We compare the two models to see how well they perform on the Gender Recognition [DCT] test set from MLcomp.

2 Design and Analysis of Algorithms

Logistic regression uses the principle of maximum log conditional likelihood, where we choose a parameter estimate $\hat{\theta}$ that maximizes the log joint conditional likelihood [5]. This is the sum of the

log conditional likelihood for each training example:

$$\begin{aligned} LCL &= \sum_{i=1}^{n} \log L(\theta; y_i|x_i) = \sum_{i=1}^{n} \log f(y_i|x_i;\theta) \\ &= \sum_{i:y_i=1} \log p_i + \sum_{i:y_i=0} \log(1-p_i) \\ &= \sum_{i=1}^{n} \log(p_i y_i + (1-p_i)y_i). \end{aligned} \tag{1}$$

Here we assume the conditional model

$$p_i = p(Y=1|x;\alpha,\beta) = \sigma(\alpha + \sum_{j=1}^{d} \beta_j x_j) = \frac{1}{1+\exp -[\alpha + \sum_{j=1}^{d} \beta_j x_j]} \tag{2}$$

where α is the intercept. For simplification, we assume that $\alpha = \beta_0$ and we add $x_0 = 1$ for all examples, such that (2) becomes

$$p_i = \frac{1}{1+\exp -[\sum_{j=0}^{d} \beta_j x_j]}. \tag{3}$$

The objective function of the logistic regression model is to find the parameter vector $\hat{\beta}$ such that

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}}\, LCL. \tag{4}$$

Since logistic regression tries to fit the training data into the model to maximize LCL, the problem of overfitting tend to arise. In one extreme example, parameters could be sent to $\pm\infty$ with either a monotonically positive or negative gradient of LCL. The standard method to solve this problem is by introducing regularization to the model, which imposes a penalty on the magnitude of the parameter values. However, there is a tradeoff between minimizing the regularization penalty and maximizing the regularized log joint conditional likelihood (RLCL). With the added regularization penalty, the objective function for the optimization problem becomes

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}}\, RLCL = \underset{\beta}{\operatorname{argmax}}(LCL - \mu||\beta||_2^2) \tag{5}$$

where $||\beta||_2^2 = \sum_{j=1}^{d} \beta_j^2$ is the squared L_2 norm of the parameter vector β of length d. The constant μ is the strength of the regularization which quantifies the trade-off between maximizing likelihood and minimizing parameter values, preventing those parameters from going to infinity. This is called the quadratic or Tikhonov regularization. Note that since each training data provides information about the intercept of the model β_0, there is enough information to avoid overfitting of this parameter. Therefore, β_0 does not need to be regularized [5].

The partial derivative of LCL with respect to parameter β_j is

$$\frac{\partial}{\partial \beta_j} LCL = \sum_i (y_i - p_i) x_{ij} \tag{6}$$

whereas the partial derivative of RLCL with respect to parameter β_j is

$$\frac{\partial}{\partial \beta_j} RLCL = \sum_i (y_i - p_i) x_{ij} - 2\mu\beta_j. \tag{7}$$

The following subsections discuss the different algorithm of the two optimization methods.

2.1 Stochastic Gradient Descent

Since our objective function is a maximization problem, this method should be more appropriately called the stochastic gradient ascent. The parameter values β is changed one step at a time following the gradient until it finally converges to the local maxima point.

The regular gradient descent update rule of the parameter β_j is

$$\beta_j := \beta_j + \lambda \frac{\partial}{\partial \beta_j} RLCL \tag{8}$$

where λ is the learning rate, denoting how big of a step each update should take. However, calculating the partial derivatives per iteration can be time consuming as each requires $O(nd)$ time where n is the training example size and d is the number of features.

To minimize computation, we use stochastic gradient descent method, where we get a random approximation to the partial derivatives by just looking at one randomly chosen example at a time to update β. Thus, we can calculate it in much less time, around $O(d)$ time, independent of the size of the training data. This is extremely useful for very large training set.

By using SGD, the parameter update rule then becomes

$$\beta_j := \beta_j + \lambda[(y_i - p_i)x_j - 2\mu\beta_j] \tag{9}$$

for any randomly chosen i^{th} example. And the update rule for the whole parameter vector $\bar{\beta}$ is

$$\bar{\beta} := \bar{\beta} + \lambda(\bar{y} - \bar{p})^T X \tag{10}$$

This vector update should be done at least once after each epoch of stochastic gradient ascent iteration. An epoch is a complete update for every example in the dataset.

Convergence is reached when the change in the objective value, RLCL, is within a certain threshold which we hold constant at 10^{-3}. Typically, this check is done in the middle of an epoch, which means that the iteration can stop if it reaches convergence before even going through the entire examples. This can save a lot of time when dealing with massive data. However, we decided to do the convergence check after every epoch. The learning rate λ is made smaller by 10% after every epoch. This is done so that convergence is reached faster. Additionally, bigger λ means bigger step and this might cause instability where the objective function cannot reach its true local maxima.

2.2 L-BFGS

L-BFGS is an optimization algorithm of the quasi-Newton method using a limited amount of computer memory[3]. This method is often used for parameter estimation. As mentioned above, for the purpose of this paper, we use Mark Schmidt's `minFunc` function, written in Matlab, that implements the L-BFGS algorithm. This function is a minimizing function. Therefore, for consistency, we modify the objective function (5) from maximizing the regularized LCL to minimizing its negative value.

$$\begin{aligned} \hat{\beta} &= \underset{\beta}{\operatorname{argmin}}(-RLCL) \\ &= \underset{\beta}{\operatorname{argmin}}(-LCL + \mu||\beta||_2^2) \end{aligned} \tag{11}$$

3 Design of Experiments

The stochastic gradient descent and L-BGS methods, both implemented in Matlab, are tested on the Gender Recognition [DCT] dataset from MLcomp, which contains 559 training examples and 239 test examples, with 800 features and a binary label. However, we need to have a separate

set for validation, to choose the hyperparameter values that best fits our model on the training set. Therefore, we split the training set such that 80% of it becomes the training set, consisting of 447 samples, and the other 112 samples will be the validation set.

First, we preprocess the dataset by normalizing the feature values using z-scoring method such that each feature has zero mean and unit variance. The same normalization factors, calculated from the training data, are used in the testing set as well. For simplicity, we also change the label values from -1 to 0, so that the binary label value becomes 0 or 1, instead of -1 or 1. We then add a column of ones in front of both the training and testing dataset, to correspond to the intercept term that is added to the parameter β as seen in (3).

Secondly, we randomize the order of the training set. This is not required for the L-BFGS implementation since it reads the entire training set at every iteration. However, with stochastic gradient descent, the order of the training data becomes more significant. Later data in the epoch tend to have more impact on the model. This randomization of training set is done to guarantee that the data is not originally sorted in any way that might affect the learning process.

The hyperparameters for both models are chosen using the grid search method. For stochastic gradient descent, there are two hyperparameters to be trained, which are the regularization strength μ and the learning rate λ. The grid search for both parameters is done over the range of $\{10^{-5}, 10^{-4}, \ldots, 10^{5}\}$. Different combinations of λ and μ are tried on the training set to train the model and the combination that gives highest accuracy in the validation set is picked. The same grid search is done for the L-BFGS model, however there is only one hyperparameter to be trained here, namely the regularization strength μ. This is done over the same range of $\{10^{-5}, 10^{-4}, \ldots, 10^{5}\}$.

The parameter β, which also includes the intercept term, is initialized to be all zero. It is important to pick a good initial value since both optimization methods find the local extrema point. We tried different values, but zero seems to produce a reasonable β vector. That is why we decided to use zero. Furthermore, we limit the maximum number of epoch for each training to be 100 since the objective function tends to converge before then.

Finally, we do verification to make sure that our algorithms are implemented correctly. We use Carl Edward Rasmussen's Matlab `checkgrad` function [6] which allows us to check the partial derivative of our objective function, given in (7), by comparing it to the finite differences approximation.

4 Results of Experiments

After doing grid search for all the parameters, we find that the optimal values for the stochastic gradient descent model are

$$\lambda = 0.01 \ \texttt{and} \ \mu = 0.01,$$

whereas the optimal hyperparameter value for the L-BFGS model is

$$\mu = 10^{-0.5}.$$

Table 1: Test set accuracy comparison between SGD and L-BFGS model

Model	Min	Max	Mean
SGD	0.894	0.941	0.9164
L-BFGS	0.883	0.937	0.9123

These values are then used to train the parameter β vector from each model. To reduce variance on our result, we train both the stochastic gradient descent and L-BFGS model 100 times, each time reshuffling the training set, to get different parameter β values. The accuracy on the testing set for both models are presented in Table 1. And Figure 1 show the accuracy of the stochastic gradient descent model.

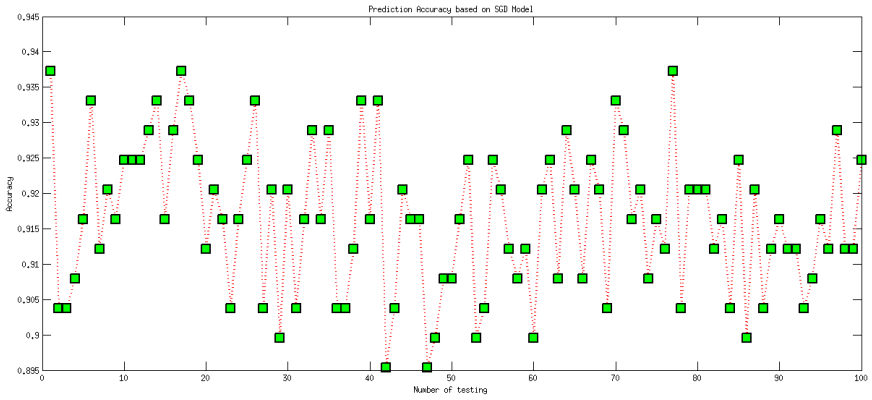

Figure 1: Test set accuracy on stochastic gradient descent model.

5 Findings and Lessons Learned

The Gender Recognition [DCT] dataset provided by MLcomp is too small to make a good prediction. With only 559 examples to be divided for both training and validation set, overfitting problem becomes unavoidable. Figure 2 shows accuracy of training and validation set after training with λ of 0.01 and μ of 0.01. The accuracy of training set is always 1, which clearly indicates overfitting. The accuracy of the validation set has to be interpreted carefully, because bigger training set means smaller validation set. With 500 training examples, the validation set only has 59 examples, which is too small to have a reliable accuracy calculation. This can explain the drop in validation accuracy at that point.

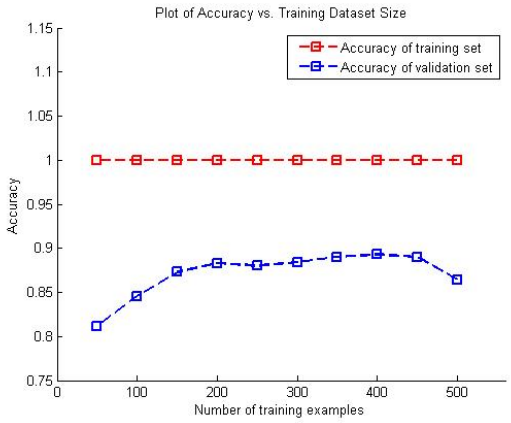

Figure 2: Accuracy of both training and validation set given different size of training examples, with λ of 0.01 and μ of 0.01.

The penalty added by the L_2 regularization can help this overfitting problem. However, there is a trade-off betw Recompile Logs 1 Download een maximizing likelihood and reducing overfitting. From figure 3, we notice that a big value of μ reduces down the accuracy of the training set. Whereas small values of μ seem to give similar accuracy. This is why we pick μ to be 0.01, as it gives a good regularization such that β values are reasonable, without sacrificing the accuracy too much.

By definition, the learning rate λ in stochastic gradient descent affects how fast the function converges. However, since this optimization method only provides a local extrema point, λ can also

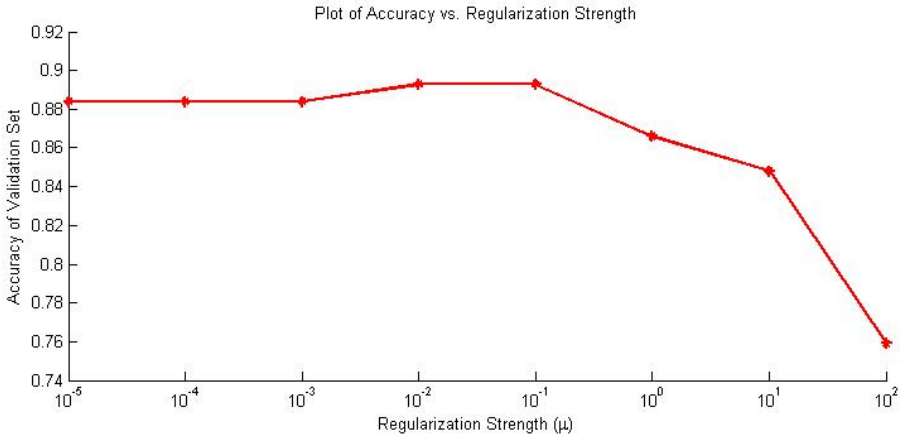

Figure 3: Variance of validation set accuracy with different regularization strength μ, and λ set at 0.01.

impact the accuracy of the trained model. Figure 4 shows how 2 different λ values can result in 2 very different RLCL values. This can happen when the jump step is big enough that it overshoots from one local extrema to the next. However, this jump is not guaranteed to give better result. That is why choosing the λ value is not a trivial matter, as well as choosing a good initial guess of the parameter.

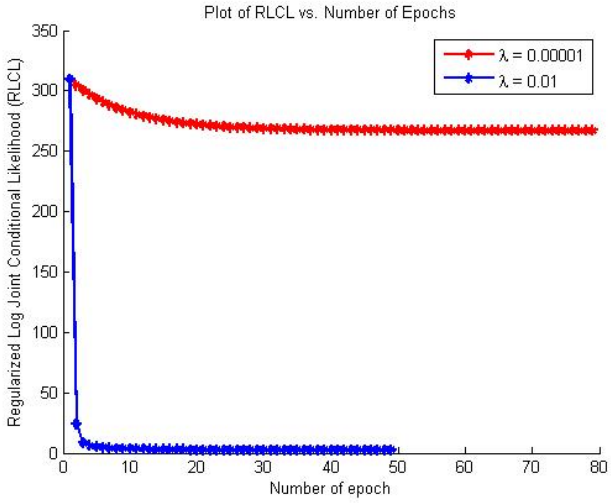

Figure 4: Different convergence RLCL values are observed from different values of λ.

The average time required to train the SGD model until convergence is 0.87 seconds, whereas L-BFGS averages around 0.38 seconds. One reason for the slow convergence of SGD can be attributed to the small value of λ. However, looking at Figure 4, with λ value of 0.01 we can see that by 10 epochs, the RLCL has pretty much converged to a value, but the computation keeps going for 40 more epochs. This shows that our chosen threshold of 10^{-3}, used to check convergence, is too small.

Overall, our SGD model, with a test error of 0.0836, seems to be performing well, even with the limitation in training set size. In fact, we achieve similar accuracy as the `svmlight-linear` model with test error of 0.084 reported in MLcomp website. However, the `svmlight-linear` model performs much faster at 0.16 seconds, around 5x the speed of SGD.

References

[1] Gender Recognition [DCT] dataset. Available at http://mlcomp.org/datasets/1571.

[2] Wikipedia article *Binary classification*. Available at http://en.wikipedia.org/wiki/Binary_classification.

[3] Wikipedia article *Limited-memory BFGS*. Available at http://en.wikipedia.org/wiki/Limited-memory_BFGS.

[4] Schmidt, M. (2012). *minFunc*. Available at http://www.di.ens.fr/m̃schmidt/Software/minFunc.html

[5] Elkan, C. (2014). *Maximum Likelihood, Logistic Regression, and Stochastic Gradient Training*. Available at http://cseweb.ucsd.edu/∼elkan/250B/logreg.pdf.

[6] Rasmussen, C. (2001). *checkgrad.m*. Available at http://learning.eng.cam.ac.uk/carl/code/minimize/checkgrad.m.