Prabhav Agrawal PID: A53054901
Email: prabhav@eng.ucsd.edu
Soham Shah PID: A53041159
Email: srshah@eng.ucsd.edu

1 Abstract

In this report, we discuss gradient based optimization methods for training logistic regression models using L2 regularization. We also discuss the need for regularization. We then talk about the implementation specifics and techniques used to estimate the hyper parameters. Finally we discuss various problems encountered, lessons learnt and results obtained through the project.

2 Introduction

The project aims to train a Logistic Regression model using L2 Regularization on a Gender Recognition dataset. We use two gradient based optimization methods, Stochastic Gradient Descent (SGD) and L-BFGS. For SGD, we implement the optimizer to train the logistic regression model on the train data and for L-BFGS we use the minFunc library. We use grid search to select the values of the hyper parameters μ and λ that give that maximize the Log Conditional Likelihood on the validation data. Finally, we apply the model to predict the gender on the test set and measure the accuracy using SGD and L-BFGS.

3 Model Analysis

3.1 Logistic Regression

We have used the principle of maximum likelihood to train our model. Given a family of probability distributions, Principle of maximum likelihood picks the model that maximizes the likelihood on the training set. Given the nature of our dataset, logistic regression served as a simple model for binary labels who's probability is linearly dependent on the features. Since maximizing likelihood of a logistic regression model involves solving a system non-linear equations, we look alternate optimization techniques.

3.2 Optimization Techniques

We compare two gradient based optimization techniques. The first one is Stochastic Gradient Descent (SGD) and other is L-BFGS. We will talk about the properties of both these techniques here and compare performance in the following sections. We will also talk about the need for Regularization and how we used L-2 regularization.

3.3 Gradient Based Optimization-Gradient Following

The simplest gradient following technique involves calculating the gradient at each iteration and then updating the parameters based on this value until convergence is achieved. As in the notes, the partial derivative of the log conditional likelihood (LCL) with respect to parameter β_j is:

$$\frac{\partial LCL}{\partial \beta_j} = \Sigma_i (y_i - p_i) x_{ij} \tag{1}$$

where x_{ij} is the value of the jth feature of the ith example. The gradient update of the parameter β_j is

$$\beta_j := \beta_j + \lambda \frac{\partial LCL}{\partial \beta_j} \tag{2}$$

where λ is the step size. The drawback of this approach is that each update involves recalculating the gradient which requires scanning of over the entire dataset (epoch). Instead we explore the performance of a faster gradient optimization method.

3.4 Stochastic Gradient Descent

In this optimization technique, instead of calculating the gradient over all the training examples, we calculate the gradient over a randomly picked example that approximates the true gradient value. Let x_i be the randomly picked example, then the update rule for parameter β_j can be written as:

$$\beta_j := \beta_j + \lambda (y_i - p_i) x_{ij} \tag{3}$$

The major advantage of this approach is time complexity. Each iteration requires only $O(d)$ time as against $O(nd)$ time required for gradient following. Furthermore, the update rule only needs to be applied to those

Prabhav Agrawal PID: A53054901
Email: prabhav@eng.ucsd.edu
Soham Shah PID: A53041159
Email: srshah@eng.ucsd.edu

β values corresponding to non-zero values of the features, reducing the running time of an iteration to $O(nf)$ where f is the average number of non-zero features per example.

3.5 L-BFGS

The well-known Newton's method requires computation of the inverse of the Hessian matrix of the objective function. However, the computational cost for the inverse Hessian matrix is expensive especially when the objective function takes a large number of variables. The L-BFGS method iteratively finds a minimizer by approximating the inverse Hessian matrix by information from last m iterations. This innovation saves the memory storage and computational time drastically for large-scaled problems.

3.6 Need for Regularization

We are training a model to best fit the data, but depending on the nature of the training data we could over-fit. Consider a binary feature j, mathematically speaking the β_j value for a feature could go on increasing with each iteration if all the training examples with feature j are males. Our model will predict every test example with feature j as 1 to be male, irrespective of the value of the other features.
To prevent this over-fitting of data, we use L2 regularization. A penalty dependent on $L2$ norm of β vector is added to the LCL term and we get a model that maximizes Regularized LCL (RLCL) over the train set. As in the notes, this term is given by,

$$\hat{\beta} = argmax_{\beta} LCL - \mu \mid \beta \mid_2^2 \tag{4}$$

and the corresponding update rule is

$$\beta_j := \beta_j + \lambda[(y - p)x_j - 2\mu\beta_j] \tag{5}$$

Although introduction of the penalty implies that in each iteration we must update all the parameters, including those corresponding to zero feature values, our data set is dense and hence performance is not affected by this added complexity.

3.7 Selecting the Value of Hyper-parameters

Hyper-parameters are parameters that control the learning rate of the algorithm. Gradient based optimization methods introduce step size λ as an additional parameter. Furthermore, for regularization we must select a value for μ. Selection of these parameters is done using grid search, refer to section 4 for details.

4 Design of Experiments

We conduct our experiments of SGD and L-BFGS on the Gender Recognition Dataset. There are 559 examples in the training set and 239 in the test set. Each example has a 800 - dimensional feature vector and a binary-valued label associated with it.

4.1 Preprocessing

- **Randomizing:** Input data is read to form a matrix of features X and vectors of labels Y and a random permutation of it is generated
- **Splitting:** We then split the randomized data into two parts: training and validation set in the ratio 4:1 respectively.
- **Feature Scaling:** For every dimension of training data, sample mean and sample standard deviation is calculated as:

$$\mu = \frac{1}{N}\Sigma_{i=1}^{N} X_i \tag{6}$$

$$\sigma = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(X_i - \mu)^2} \tag{7}$$

where μ and σ each represent a 800 dimension vector

All feature vectors in training, validation and test data are scaled as $Z = \frac{X_i - \mu}{\sigma}$ and a vector of 1s with size equal to number of examples is concatenated to each.

Prabhav Agrawal PID: A53054901
Email: prabhav@eng.ucsd.edu
Soham Shah PID: A53041159
Email: srshah@eng.ucsd.edu

4.2 Parameter Calculation

– **Stochastic Gradient Descent:**

– For a particular value of λ and μ, we run the SGD procedure for maximizing RLCL (refer Section 3) to obtain the coefficient vector β after convergence. The criteria for convergence is taken as:

$$\Sigma_{j=1}^{D} \mid \frac{\partial RLCL}{\partial \beta_j} \mid < 10^{-2} \tag{8}$$

– We perform a grid search for selecting the best values of λ and μ We varied λ from 2^{-15} to 2^{+5} and μ from 2^{-15} to 2^{+5} exponentially (i.e. in increasing powers of 2). For a given μ, only those value of λ are selected for which convergence criteria has been satisfied on the training set.

– We obtained (λ, μ) pairs for which convergence criteria has been met from previous step. We then calculate LCL for these pairs on the validation set. Since for a given μ, the LCL is converging to almost same value (within $+-2.0$) , β vector for the (λ, μ) pair for which LCL is maximized on the validation set is selected.

– **L-BFGS:**

– We used a MATLAB implementation of L-BFGS from http://www.di.ens.fr/ mschmidt/Software/minFunc/minFunc.html. We computed LCL, RLCL and partial derivatives and supplied to the L-BFGS function.

– We perform a grid search for selecting best value of μ. For a given value of μ, we find the parameter vector β and select the one with highest LCL on the validation set.

4.3 Testing

We use the β vector obtained from SGD and L-BFGS approach to calculate label probability P as:

$$P = \sigma(\frac{1}{1 + exp(\Sigma_{i=1}^{D} \beta_i x_i)}) \tag{9}$$

If $P > 0.5$, we assign label as 1 otherwise 0

5 Results

5.1 Stochastic Gradient Descent

As described in Section 4, we maximized RLCL for a particular value of λ and μ. Figure 1 displays the variation in RLCL with number of epochs. We can see that a low value of λ leads to slow convergence while a higher value of λ leads to oscillation about the optimum value

From the approach described in Section 4, we obtain a final value (i.e. the one with maximum LCL on validation set) for (λ, μ) pair after performing grid search as $(0.0156, 0.0313)$.

Figure 2 shows the variation of LCL on validation set with μ. We can confirm that a maximum for LCL is achieved in the range selected for μ

Table 1: Accuracy for optimum (λ, μ) pair

Dataset	Accuracy
Train	1.00
Validation	0.8996
Test	0.8661

Table 1 displays the performance of Stochastic Gradient Descent on the three sets. The decrease in accuracies from train to validation to test is expected because LCL is maximized on train set, hyper-parameter are adjusted according to validation set and the test set is used only once for prediction.

Figure 3 illustrates the fact that no regularization (i.e. $\mu = 0.0$) leads to over-fitting on the training set and worse performance on validation and test datasets.

Prabhav Agrawal PID: A53054901
Email: prabhav@eng.ucsd.edu
Soham Shah PID: A53041159
Email: srshah@eng.ucsd.edu

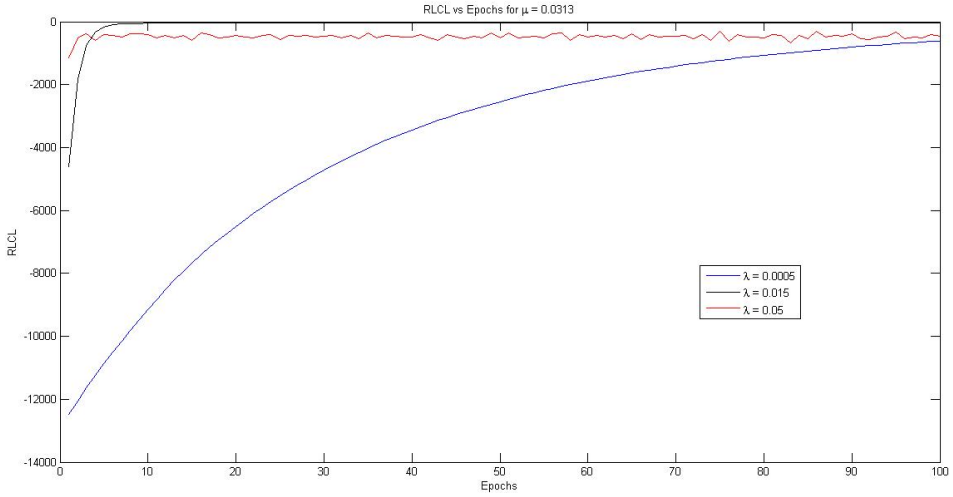

Figure 1: RLCL vs Epochs for $\lambda = 0.0005, 0.0156, 0.05$ for $\mu = 0.0313$

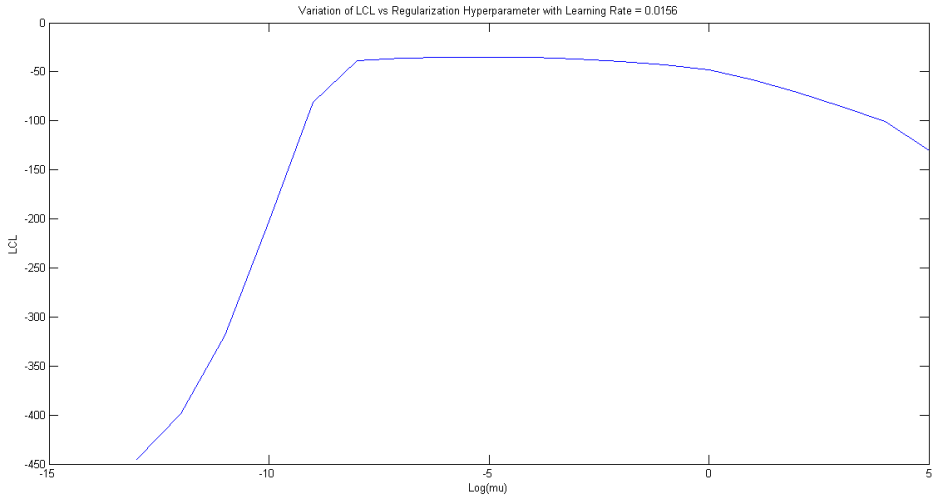

Figure 2: LCL vs μ for $\lambda = 0.0156$ on validation set

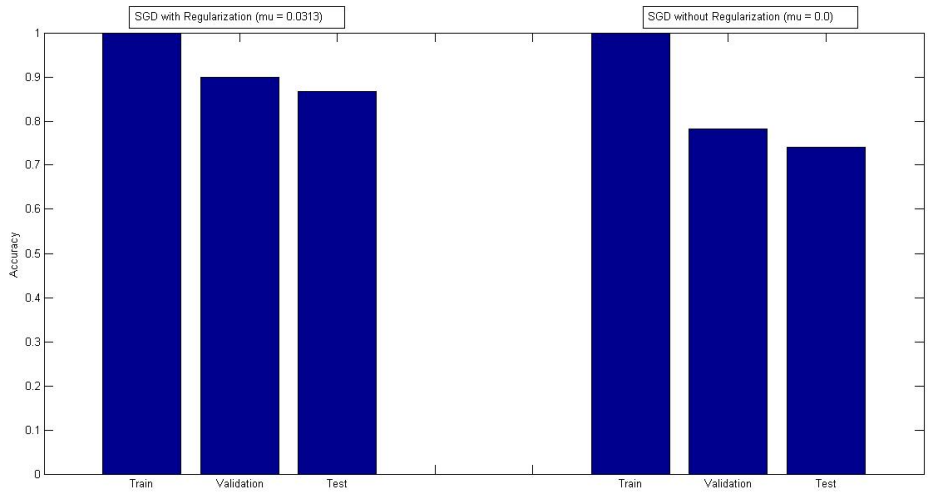

Figure 3: Accuracy on three datasets - with and without regularization

Prabhav Agrawal PID: A53054901
Email: prabhav@eng.ucsd.edu
Soham Shah PID: A53041159
Email: srshah@eng.ucsd.edu

5.2 L-BFGS

Since there is only one hyper-parameter μ in L-BFGS approach, we choose the μ which maximizes LCL on validation set. In this setting, it comes out as $\mu = 8.0$

Figure 4 displays the variation in LCL vs μ. Maximum in the curve confirms that we have selected an appropriate region for μ in grid search.

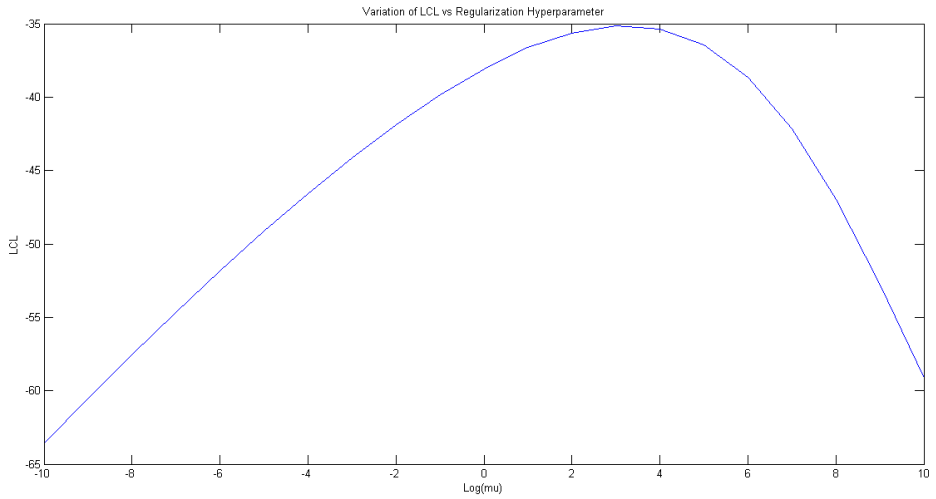

Figure 4: LCL vs $\mu = 8.0$ on validation set

Table 1 displays the performance of L-BFGS on three datasets.

Table 2: Accuracy for optimum (λ, μ) pair

Dataset	Accuracy
Train	1.00
Validation	0.8571
Test	0.8954

In Figure 5, we see very minor decrease in accuracy with regularization as compared to without regularization. This suggests that L-BFGS being a better optimization method than SGD, is less prone to over-fitting.

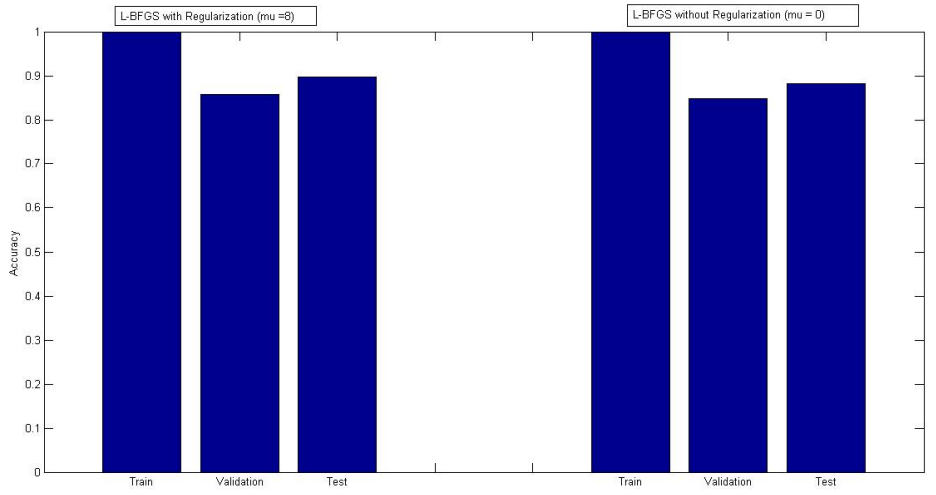

Figure 5: Accuracy on three datasets - with and without regularization

Prabhav Agrawal PID: A53054901
Email: prabhav@eng.ucsd.edu
Soham Shah PID: A53041159
Email: srshah@eng.ucsd.edu

6 Observations and Lessons Learnt

- Applying SGD to different random permutations of the training and validation set, gives different β vector for the same hyper parameters. This can because of SGD not converging perfectly to optimal β values, and changing the permutation gives us a different answer.
- Without feature scaling, we got some of LCL values as $-Inf$. The reason for this is the input data has bigger features values and logistic regression probability values tend to go to 0 (because of underflow) causing the log probability to go to -Inf.
- We experimented with adaptive learning by updating λ values by 0.9λ after every epoch, but for low values of lambda, RLCL was converging to a value which was not optimal. This is because, after a few epochs the learning rate becomes too small such that β values are not updated even when the derivative of RLCL is non-zero.
- To test for convergence, we explored the option of plotting the RLCL values v/s number of epochs, which seemed a good approach because of better visualization . The only drawback was it was manual and grid search required running the algorithm over large number of hyper parameters. Thus, we dropped this idea and settled for our partial-derivative based convergence criterion so that we can automate the process.
- SGD gave a better accuracy on the test data with regularization as compared to without it, but for L-BFGS this improvement was much smaller. We conclude that L-BFGS does a better job dealing with over-fitting as compared to SGD.

References

[1] B. Carpenter, Lazy sparse stochastic gradient descent for regularized multinomial logistic regression.“ *Technical report* (2008)

[2] P. Komarek and A.W. Moore, Making logistic regression a core data mining tool with TR-IRLS. *In proceedings of the Fifth IEEE International Conference on Data Mining* (2005), pp. 685-688

[3] T.P. Minka, A comparison of numerical optimizers for logistic regression. (2001)