Samuel Looper 2019-03-12
1003044181

ROB313 Assignment #3

1. Assignment Objectives

The purpose of this assignment is to evaluate the effectiveness of gradient descent techniques for optimization of estimation parameters. The performance of full batch descent and stochastic gradient descent is compared with respect to a linear regression of data sets supplied in the class material. The loss function error of the two algorithms are compared with respect to the iteration number, and a parameter sweep is performed on the learning rate for each algorithm. Next, gradient descent is used for classification of data sets supplied in class material. In this case we use logistic regression using the Bernoulli likelihood function and a sigmoid estimation function. First, the log likelihood associated with this a classification error is calculated, and analyzed. Next, full batch gradient descent and stochastic gradient descent are used to calculated the maximum likelihood estimate of the weight parameters. For full batch gradient descent the exact negative log likelihood is analyzed with respect to various learning rates. The final results of the classification with optimal parameters are then analyzed for log likelihood and test accuracy.

2. Algorithm Overview

For the linear regression portion, separate functions are created for the full batch and the stochastic gradient descent regression. The parameter sweeps were handled in the main code loop, and helper functions were created to make the code easier to read and remove duplicate calculations for loss function and gradient calculations, as well as root mean squared error. The data sets are parsed into a training set for the regression analysis and plotted in the main code as well.

For the logistic classification, the modular functions from the first part were modified with the help of a set of new helper functions. The input data had to be pre-processed to be compatible with the gradient descent functions, which required a floating point value feature vector set. Otherwise, the code structure followed the same guidelines: parameter sweeps in the main loops, helper functions for readable code, and a main loop where most of the gradient descent occurs.

3. Results

3.1 Gradient Descent for Linear Regression

3.1.1 Full Batch Gradient Descent

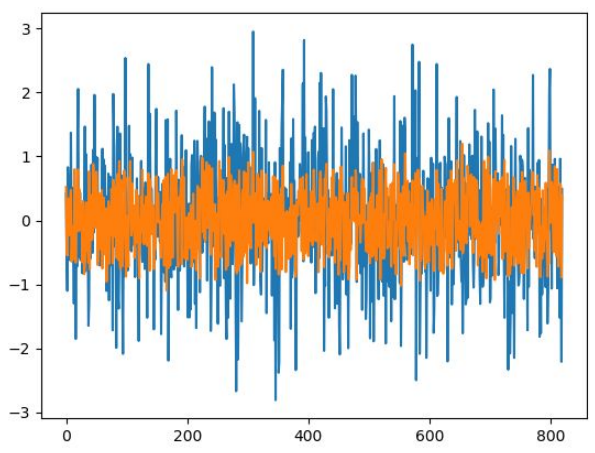

Fig 1 Regression Results

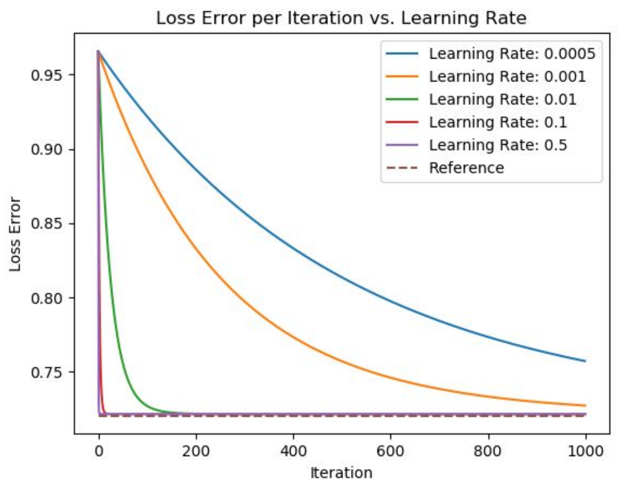

Fig 2 Learning Rate Plot

Optimal Learning Rate: 0.5

3.1.2 Stochastic Gradient Descent

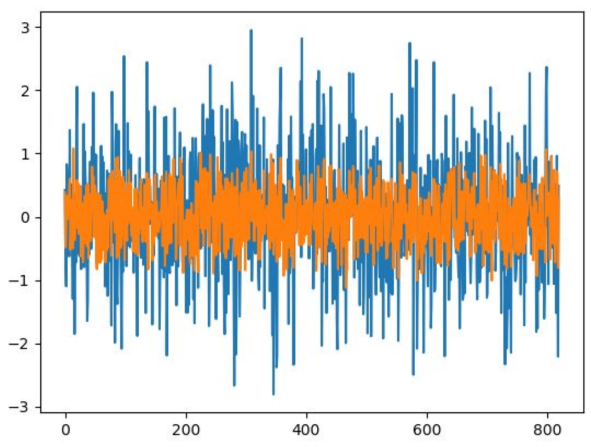

Fig 3 Regression Results

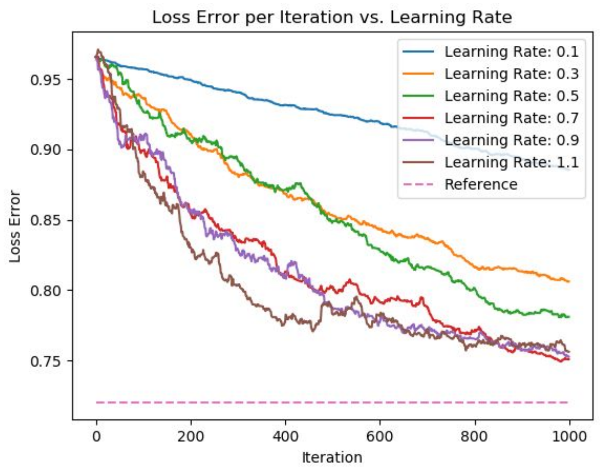

Fig 4 Learning Rate Plot

Optimal Learning Rate: 0.7

3.1.3 Comments and Analysis

After performing the regression using both full batch and gradient descent, we notice some similarities, but also glaring differences between the results. Both show a general convergence towards the reference error from a SVD linear regression, and show similar results in a comparison between the resulting estimates on the PumaDyn32 testing set.

The full batch gradient descent showed clear, monotonous convergence towards the reference error, and is approximately at the reference error mark before the epoch is finished. On the other hand the stochastic gradient descent has local maxima and a more erratic trend, generally converging to error, albeit much slower; the error was still off by about 0.4 after a full epoch. SGD also benefits from higher learning rates, although for values greater than 0.7, the error initially converges faster, but eventually reaches the same performance as the 0.7 learning rate. Otherwise, in both cases, the learning rates were chosen to be lower than the optimum to demonstrate how increasing the rate gradually improves performance. In full batch descent, for learning rates greater than the optimum, the error quickly begins diverging to positive infinity rather than converging at all, thus they were not included in the plot.

3.2 Gradient Descent for Logistic Log-Likelihood Classification

3.2.1 Log Likelihood Edge Case Discussion

The assignment details a possible case where the log likelihood function has an output of 1 (indicating 100% certainty that the feature vector belongs in class 1) but the vector actually belongs to class zero. According to the log likelihood function:

$$\log \Pr(\mathbf{y}|\mathbf{w}, \mathbf{X}) = \sum_{i=1}^{N} y^{(i)} \log\left(\widehat{f}(\mathbf{x}^{(i)}; \mathbf{w})\right) + \left(1 - y^{(i)}\right) \log\left(1 - \widehat{f}(\mathbf{x}^{(i)}; \mathbf{w})\right).$$

we would have the log likelihood probability go to negative infinity. This is due to *$log(1-f(x^{(i)};w))$* going to infinity as *$f(x^{(i)};w)$* goes to 1, and the *$(1-y^{(i)})$* stays at 1. This presence of infinite numbers appears to make this unreasonable behavior, but in the end a log likelihood of negative infinity simply means a probability of zero. In the end, this result would only occur if the sigmoid function goes to 1, which would indicate the argument:

$$\left(w_0 + \sum_{i=1}^{D} w_i x_i\right)$$

is approaching infinity. For this to occur with finite input feature vectors, this would indicate the weights are approaching infinity. This means the problem is overfitted, and a regularization parameter *R(w)* should be included to improve the conditioning of this problem and bound the weight values.

3.2.2 Classification on *Iris Virginica* using Full Batch Descent

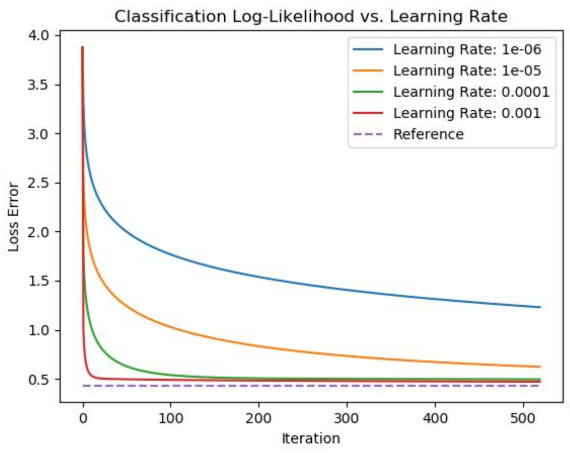

Fig 5: Testing Accuracy Metric Results

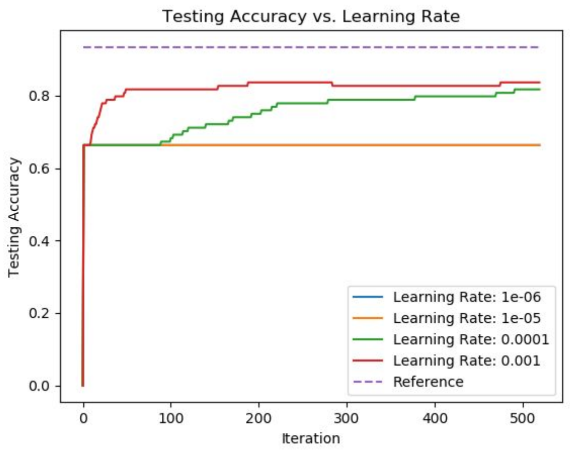

Fig 6: Testing Accuracy Metric Results

To perform classification, the initial feature vector was turned into a single value (0 if the iris wasn't a *Virginica*, 1 if it was). The reference values for our performance metrics were established using the SVD based classification algorithm from the first assignment. Next, the full batch gradient descent algorithm from the first part was modified for classification, and the helper functions were replaced for the new Bernoulli distribution and log likelihood gradient calculations and loss function metrics. This was done with relative ease due as the code was designed to be modular.

Just as in the first part, a parameter sweep was performed on the learning rate, and the two performance metrics were plotted for select representative learning rates. From these plots we learn that 0.001 is the better learning rate, although the behavior several suboptimal learning rates converge with the optimum over time. Having tried other learning rates it was observed that learning rates in the order greater than 10^{-3} lead to quickly diverging log likelihoods and totally unreasonable behavior. Comparing the two performance parameters it looks like log likelihood is a better and more representative performance metric. It is more continuous, ideally converge to zero, and have better defined convergence behavior than the testing accuracy functions.

3.2.3 Classification on *Iris Virginica* using Stochastic Gradient Descent

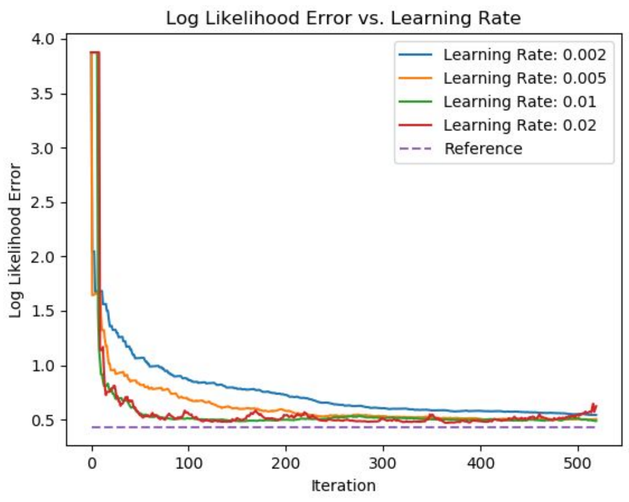

Fig 7: Testing Accuracy Metric Results

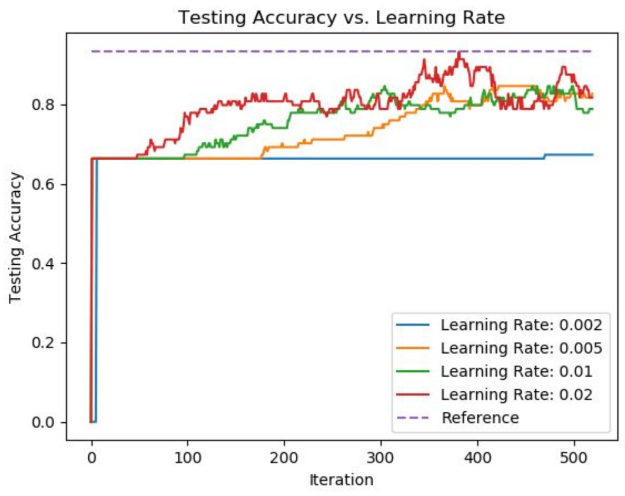

Fig 8: Testing Accuracy Metric Results

Next, the algorithm was further modified to implement a stochastic gradient descent. With the same procedure as the full gradient descent, this modified algorithm applied log likelihood Bernoulli probability to solve the Iris Virginica classification problem. In this case, higher learning rates were generally more effective than they were in the full batch descent, in similar fashion to the linear regression stochastic descent. It is worth noting that stochastic gradient descent is significantly more effective with respect to log likelihood for this classification problem than it was with respect to its loss function error in the previous problem. The performance metrics for this problem suggest an optimal learning rate of 0.01. As in the case of the full batch classification, log likelihood is the more effective performance metric due to its more continuous convergence behavior.