DS 340W: Lab Assignment 2

1. Problem, Data and Method

In this lab, we are aiming to build a logistic regression classifier with stochastic gradient descent (SGD) learning to do a classification task on the given dataset. In this lab, the given dataset collects six consecutive days (day 0 and day 5), and data samples are generated by rotating the previous day’s data by 15 degrees counterclockwise. To handle this situation, we will use the incremental learning technique, in which our classifier can evolve the data features over time, to help us have a better result on this classification task.

To implement an incremental learning technique, we have to use stochastic gradient descent (SGD) rather than naïve gradient descent because SGD allows minibatch learning, in which the algorithm can split the training dataset into small batches that are used to calculate model error and update model coefficients (Brownlee, 2019). The difference between SGD and naïve gradient descent is SGD will update the parameters based on the gradient for a single training sample, and Naïve gradient descent will update its parameters based on the gradient for entire training samples. Thus, SGD can make significant progress before it has even looked at all the data.

In scikit-learn implementation of stochastic gradient descent classifier, penalty and alpha are used to control the regularization. For penalty, it is the penalty to be used, and its values could be L2, L1, Elasticnet, or None; each term refers to L2 norm, L1 norm, elastic net regularization, and no penalty, respectively. For alpha, it is a constant that multiplies the regularization.

The equation that to find the model parameters is by minimizing the regularized training error given by

where is a loss function that measures model fit and is a regularization term that penalizes model complexity, and is a non-negative hyperparameter. In this lab, we are using logistic regression for , and then the loss function that is actually minimizing for each of the possible regularization terms becomes

and each regularization equation is below:

L1 Norm:

L2 Norm:

Elastic Net:

Max\_iter, tol, and early\_stopping are the three stopping criteria for learning the logistic regression classifier. For max\_iter, it is the maximum number of passes over the training data, and I used max\_iter = 50000 in this lab. For tol, it is the abbreviation of tolerance, and its value normally is a float; but I set tol = None in this lab, in which the iterations will stop when loss > best\_loss – tol. For early\_stoping, it is used to determine whether to terminate training when the validation score is not improving, and I used its default value (early\_stopping = False) in this lab.

1. Experiments

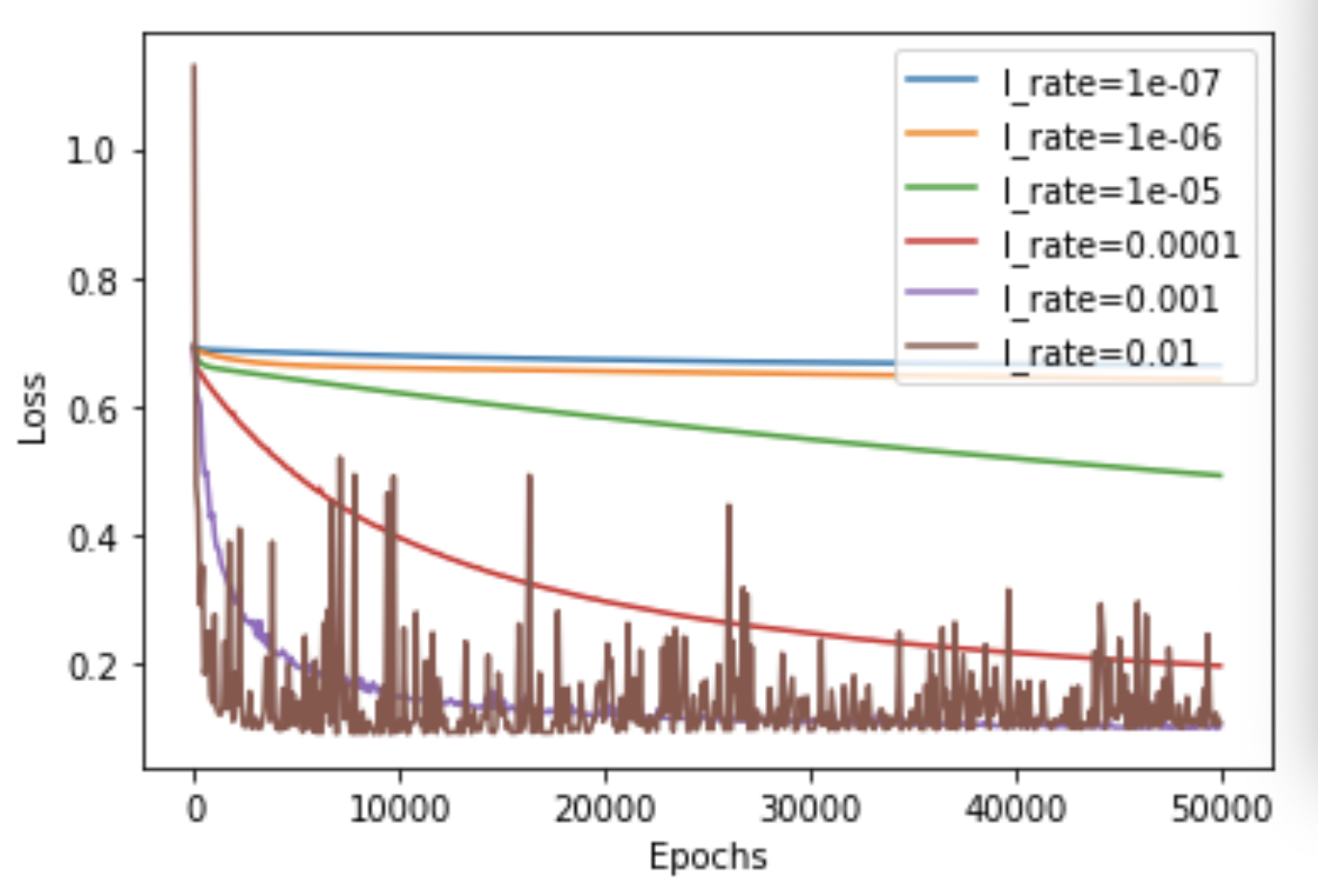
In this lab, the given dataset collects six consecutive days (day 0 and day 5), and data samples are generated by rotating the previous day’s data by 15 degrees counterclockwise. In detail, each data has 200 samples, 2 features, and a class label that value is either 0 or 1. So, the task is to build a logistic regression classifier with SGD to do a classification task. 

Figure 1: The Plot of Training Loss as a Function of the Number of Epochs

In figure 1, we can see that the training loss decreases gradually with the growth of epochs, and that indicates that the number of epochs has a negative correlation with the training loss. Another behavior I observed is that a larger learning rate will cause training loss to decrease rapidly, but a larger learning rate will also make the curve relatively unstable. Based on the results above, I determine 0.001 is the best learning rate among six different learning rates because these two following reasons (1) compare to other smaller learning rates, 0.001 has less training loss after we did 50000 iterations. (2) although training loss of 0.001 is higher than training loss of 0.01 at some points, the overall stability of the former is much better than that of the latter. Thus, I determine 0.001 is the best learning rate.

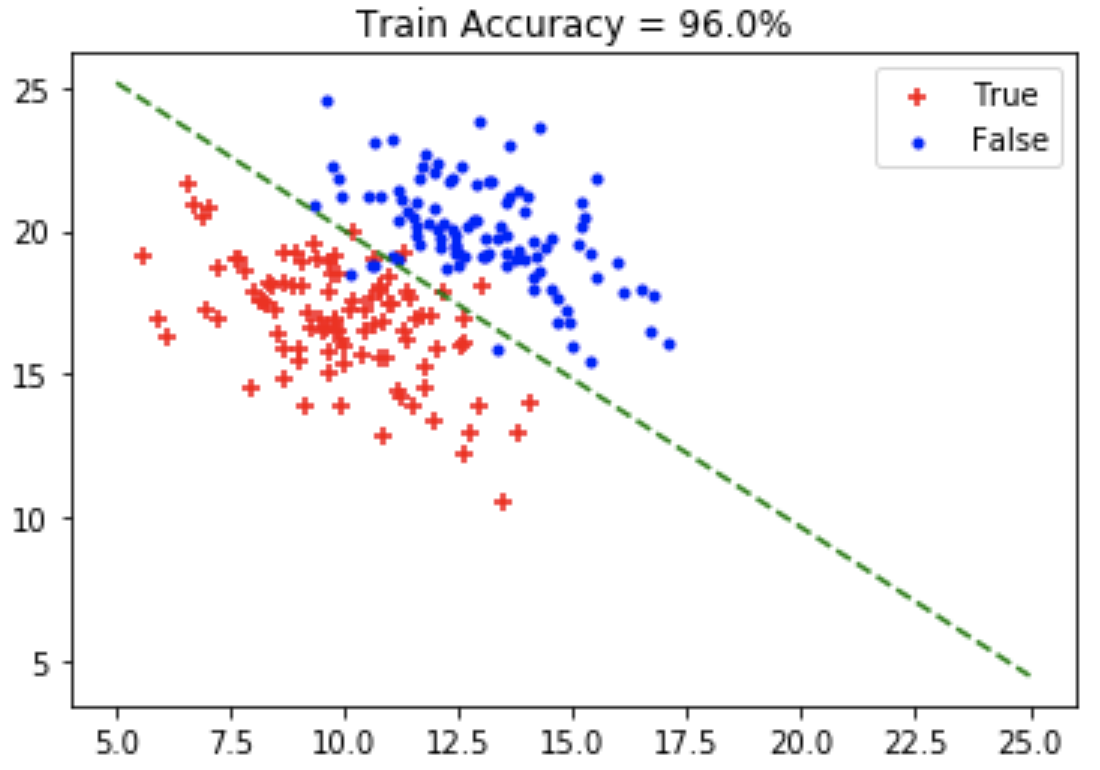


Figure 2: The plot of the data samples in day0.txt and the decision boundary of cls

To build the initial classifier cls, I used these following parameters: (1) loss = log (2) penalty = None (3) max\_iter = 50000 (4) tol = None (5) learning\_rate = constant (6) eta0 = 0.001. For loss, it is the loss function to be used, and I use logistic regression as our loss function because this classification task only has two class labels, and then logistic regression is good at this kind of problem. For penalty, I have introduced this term in the previous section. I set penalty as None because this dataset can be separate by a straight line, and the results are very good; thus, no regularization is needed for this case. For max\_iter, I also described this term in the previous section. I set it as 50000 because this number is good enough to help us train a good classifier. For tol, I set this term as None to get a classifier with a better performance. For learning\_rate, this is the learning rate schedule. And I set this parameter as constant because we want to use our best learning rate (0.001) to train this model. For eta0, it is the initial learning rate for the learning rate schedule, and I used 0.001 in this case because this is the best learning rate we got from previous steps. After the training process, we got a 96% training accuracy of cls on day 0.

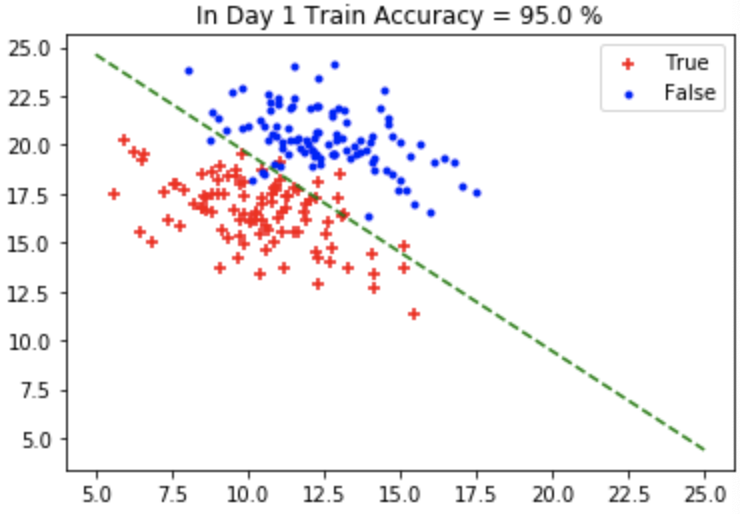


Figure 3: The plot of the data samples in day1.txt and the decision boundary of cls

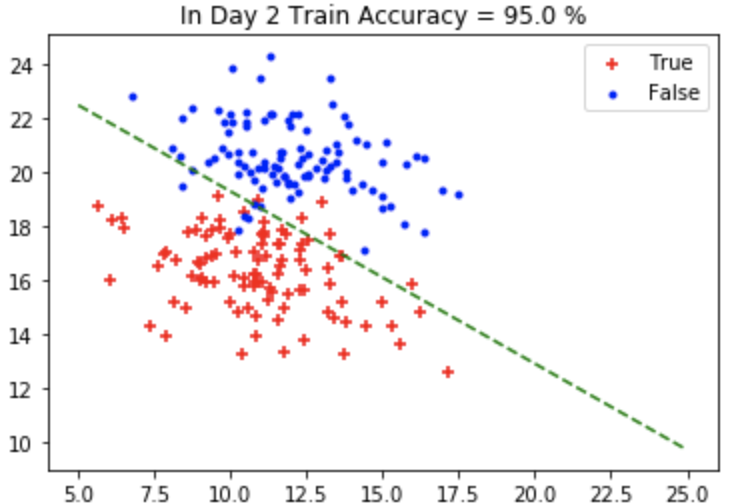


Figure 4: The plot of the data samples in day2.txt and the decision boundary of cls

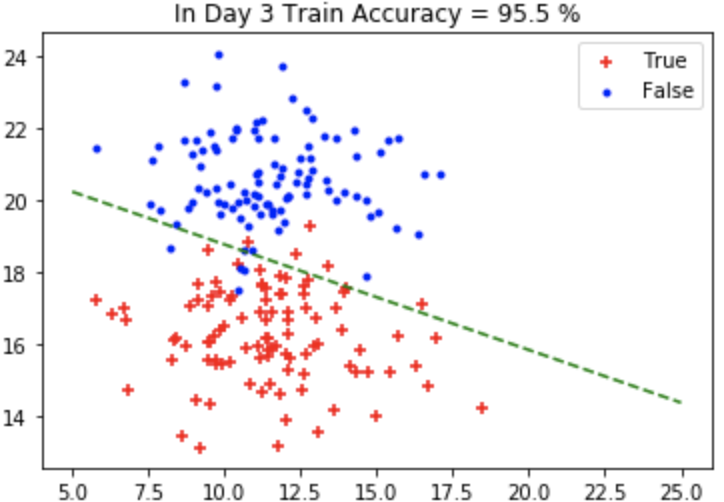


Figure 5: The plot of the data samples in day3.txt and the decision boundary of cls

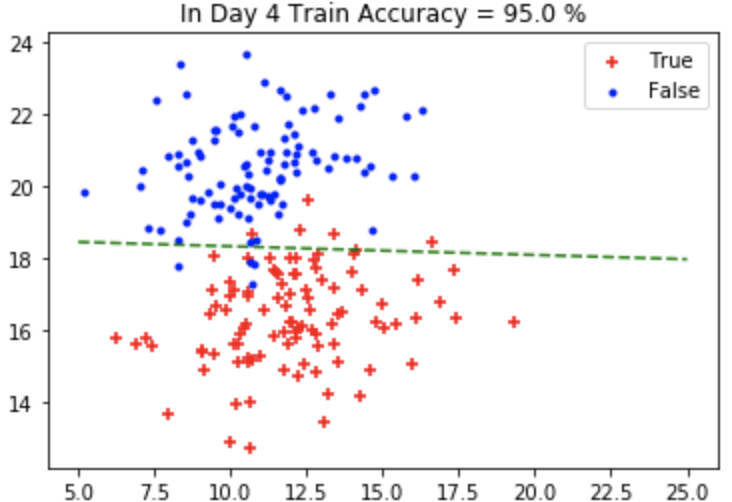


Figure 6: The plot of the data samples in day4.txt and the decision boundary of cls

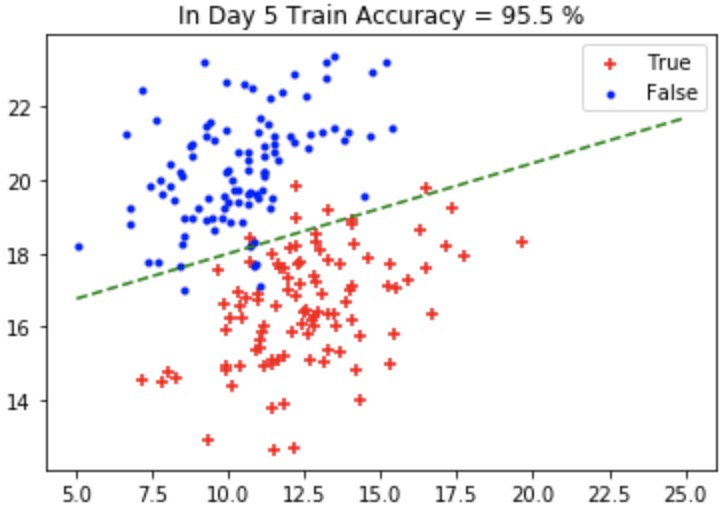


Figure 7: The plot of the data samples in day5.txt and the decision boundary of cls

From day 1 to day 5, the training accuracies are 95%, 95%, 95.5%, 95%, and 95.5%, respectively. I wrote the code that the partial\_fit will terminate if train accuracy great than or equal to 95% because I found that 95% is the minimum training accuracy on day 0 after I trained the classifier multiple times. Thus, 95% is a number that is close enough to maintain roughly the same training accuracy as day 0.

To maintain roughly the same training accuracy as day 0, the cls are needed to train 1, 11, 14, 18, 34 times for day 1 to day 5, respectively. Based on the above results, we can see that the number of epochs is getting larger and larger, which is very easy to understand because the difference between the current data (day i) and the original data (day 0) is also getting bigger. In order to approximately get a 96% training accuracy, cls has to train 50000 times (max\_iter = 50000) in day 0 to get this level. In contrast, cls trained 1, 11, 14, 18, 34 times form day 1 to day 5 to maintain roughly the same training accuracy as day 0; thus, we can say that incremental learning reduced a large amount of the number of epochs, and its efficiency is very significant. In the future, I will use the incremental learning technique to solve similar tasks. For example, we can use this technique to train some dynamic data, so we can reduce unnecessary expenditure and use resources effectively. Overall, this lab made me understand that incremental learning is very practical and useful.

Reference

Brownlee, J. (2019, August 19). A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size. Retrieved February 27, 2020, from https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

Reference on Code

I used the Sklearn, pandas, matplotlib, graphviz, numpy packages to complete the code, so some of the code may very similar.