Multi-Class Logistic Regression and Gradient Descent

Yiyang Zhou (260765144), Bingchan Ma (260761573), Helen Ren (260846901)

November 22, 2020

Abstract

In this mini project, we applied multi-class logistic regression and implemented mini batch gradient descent with momentum algorithm to solve the optimization problem, we also compared the result against other classification algorithm. We processed and interpreted two datasets: digits dataset and balance-scale dataset. To find the best hyper-parameters, we used grid search technique and 5-cross validation to enumerate all the possible combinations in the defined parameter space, the model performance was then comprehensively evaluated. We found the best hyper-parameters that maximize the validation accuracy for digists and balance-scale dataset respectively. Finally, we compared the result against KNN and Decision Tree classifier. The result suggests that KNN has higher accuracy on both digit dataset and balance-scale dataset.

1 Introduction

Softmax Regression (or multinomial logistic) is a generalization of logistic regression to the case where we want to handle multiple classes[Ng+]. Softmax Regression is a widely used technique due to its high interpretability and efficiency. It is easy to regularize and output well-calibrated predicted probabilities[Don18]. In this project, we first acquired digist dataset from Scikit-Learn and balance-scale dataset from OpenML. In order to solve optimization problem, we then implemented softmax regression model by following the code template released for lectures and constructed mini-batch optimization using gradient descent with momentum and regularization. To analyze the quality of the solution found (in terms of validation accuracy) and the run-time for different hyper-parameters, we used 5-fold cross validation to estimate model performance. We tracked the training and validation accuracy in each iteration and found the best hyper-parameters for digits dataset and balance-scale dataset. Finally, we compared the result with other classifiers, namely KNN and Decision Tree and we found the accuracy of KNN is the highest among those three classifiers.

2 Datasets

We used digits dataset from Scikit-Learn and balance-scale dataset from OpenML. There is no missing values in both datasets. In balance-scale dataset, there are 288, 49 and 288 values in class L, B and R respectively. We mapped these 3 labels to integers when processed the dataset. More details of the two datasets are listed in Table 1.

Dataset Name	Number of Samples	Number of Features	Number of Class	Contain Missing Value
Digits	1797	64	10	No
Balance-scale	625	4	3	No

Table 1: Exploratory Analysis of Digits and Balance-scale Dataset

3 Results

3.1 Discussion of how the multi-class logistic regression performance (e.g., convergence speed) depends on the parameters of optimization

The best hyper-parameters for digits dataset are batch size: 512.00, learning rate: 0.20, momentum: 0.90 and termination step: 60.00 if only considered the highest validation accuracy.These give a 97% accuracy

and an average run time of 0.725s. However, if we take both time and accuracy into consideration, batch size of 256 or 360 might be better choices since the accuracy only has little decrease(accuracy 96.7% for batch size of 256 and 96.9% for batch size of 360) while the time they use is less than half of batch size of 512.(0.28s for batch size of 256 and 0.35s for batch size of 360).
For balance-scale dataset, the best hyper-parameters are batch size: 16.00, learning rate: 0.20, momentum: 0.90 and termination step: 50.00. These provide an 89.92% accuracy and an average run time of 0.051s. Both of the datasets need about a hundred of iterations for all the 5 folds to convergence. Figure 1 and Figure 2 are the plots of validation curves for both datasets. (training accuracy:Figure15,16)

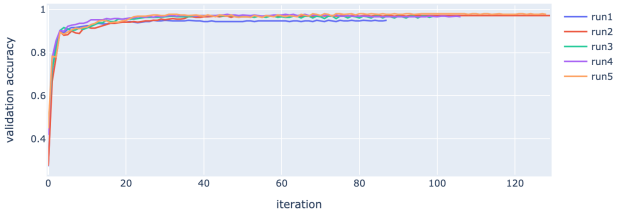

Figure 1: Validation Accuracy with Best Hyper-parameter for Digits Dataset

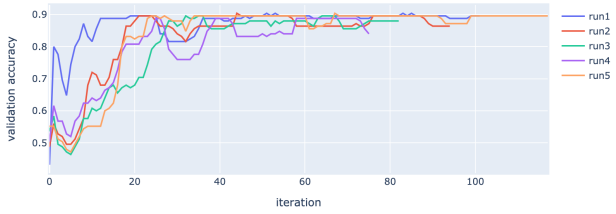

Figure 2: Validation Accuracy with Best Hyper-parameter for Balance-scale Dataset

3.2 Discussion of the training and validation curve for different optimization hyper-parameters

We found that the hyper parameters impact both accuracy and convergence speed. When varying batch size, we found that higher batch size leads to lower convergence speed and may not leads to a significant higher accuracy[Figure3]. When we changed learning rate and kept other hyper-parameter the same, the results suggest slightly higher learning rate leads to a lower run time and higher accuracy[Figure4], thus, we can recover the lost speed from a larger batch size by increasing the learning rate. From Figure5, we found that appropriate momentum can help accelerate training steps. By varying termination condition, we found that as the termination steps increase, the speed of convergence increase significantly while the accuracy do not change a lot when considering termination steps=10,20,30,..,90 and early stopping(i.e. T<20) does not lead to a higher accuracy and hence does not bring many benefits[Figure6]. Figures of performance of Softmax Regression on balance-scale dataset suggest the same findings except that the accuracy changed a lot for some specific termination steps and can be found in Appendix.
We also did some **extensive explorations** in the regularization parameter. We found that the accuracy did not change much when regularization parameter=2e-1,1e-1,1e-2,..., 1e-6,but in most cases having regularization is still better than without(i.e.regularization parameter=0) because it is used to avoid over-fitting on the data. See specific details of regularization parameters for both datasets in appendix.

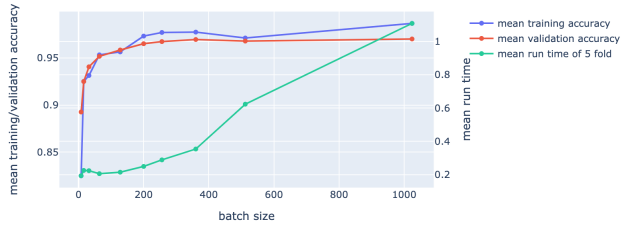

Figure 3: Performance of Softmax Regression when varying batch size on digit dataset

Figure 4: Performance of Softmax Regression when varying learning rate on digit dataset

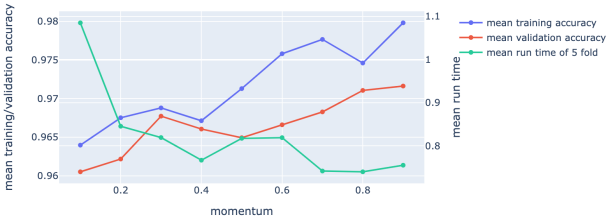

Figure 5: Performance of Softmax Regression when varying momentum on digit dataset

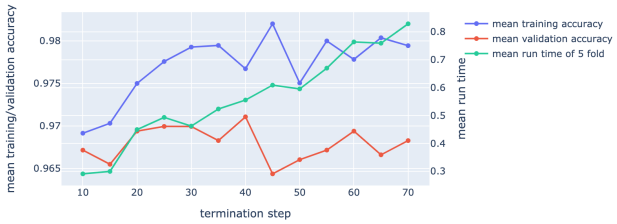

Figure 6: Performance of Softmax Regression when varying termination steps on digit dataset

3.3 Comparison of the Accuracy of other Classifier and Logistic Regression on both datasets

To compare the performance of different classifiers, we used KNN and Decision Tree on the two datasets. We implemented them by using Scikit-Learn implementation package. We found that KNN reaches the highest accuracy on both digit dataset and balance-scale dataset. The detailed result is shown in tables below. We used 5 fold cross validaion to find the optimal parameters of KNN and decision tree.(Figures11,12,13,14 in Appendix) For digits dataset, the optimal depth for decision tree is 15, and the optimal number of neighbors for KNN is 5. For balanced scale dataset, the optimal depth for decision tree is 6, and the optimal number of neighbor for KNN is 17. However, KNN(0.35s for digits and 0.02s for balanced scale) is more expensive in running time comparing with decision tree(0.04s for digits and 0.002s for balanced scale) and slightly better than running time for softmax regression(0.35s for digits set and 0.05s for balanced scale).

Dateset Name	Classifier Name	Accuracy
Digits	Softmax Regression	97.05%
Digits	KNN	98.87%
Digits	Decision Tree	85.86%

Dateset Name	Classifier Name	Accuracy
Balance-scale	Softmax Regression	89.92%
Balance-scale	KNN	90.08%
Balance-scale	Decision Tree	86.20%

4 Discussion and Conclusion

To conclude, in order to have the best performance of a model, it is necessary to find a good combination of hyper-parameters by varying one of these hyper-paramand see how the performance and run time change. We found that higher batch size leads to lower convergence speed and may not leads to a significant higher accuracy but we can recover the lost speed by increasing the learning rate. Using regularization technique is also helpful to avoid overfitting. We found KNN classifier reaches the best accuracy on both digits dataset and balance-scale dataset. In the future research, we may stack the LR model, makes it like a neural network, the accuracy may improve. If we have more time, we would perform the models on more datasets, and analyze the relations between the features of the dataset and the performance of the models. To be more **creative** in this project, we tried more than one other models to perform on the two datasets, namely KNN and Decision Tree, and the result shows that KNN would be slower as it have to keep track of all training data and find the neighbor nodes, unlike decision tree, which supports automatic feature interaction.

5 Statement of Contributions

Everyone dedicated to the project. Bingchan and Helen mainly contibuted to the implementation. Yiyang mainly contributed to the writeup. We are all responsible for quality assurance of the project.

6 Appendix

6.1 Related Figures

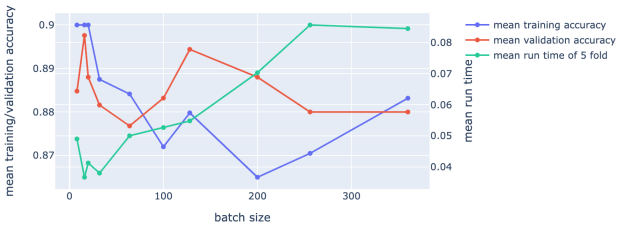

Figure 7: Performance of Softmax Regression when varying batch size on balance-scale dataset

Figure 8: Performance of Softmax Regression when varying learning rate on balance-scale dataset

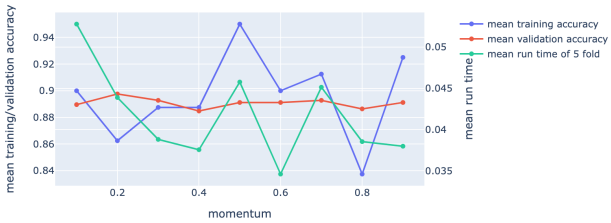

Figure 9: Performance of Softmax Regression when varying momentum on balance-scale dataset

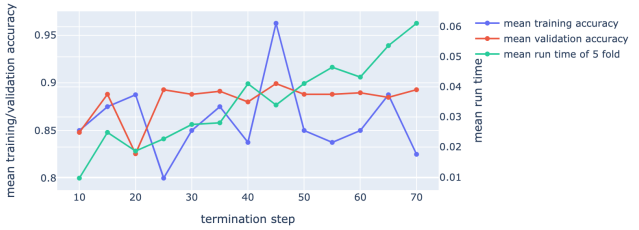

Figure 10: Performance of Softmax Regression when varying termination steps on balance-scale dataset

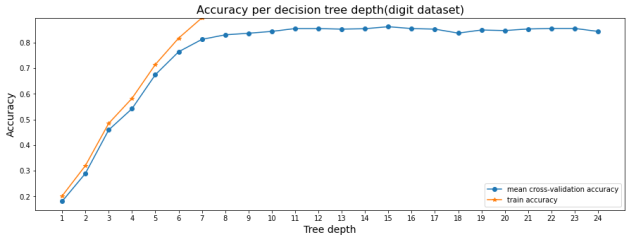

Figure 11: Accuracy per decision tree depth(digits datasets)

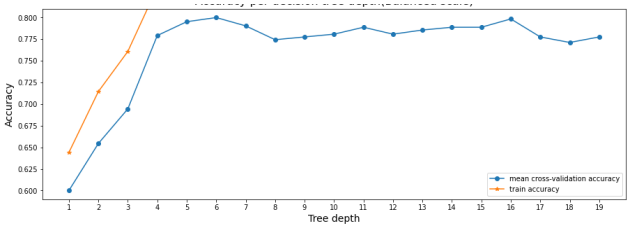

Figure 12: Accuracy per decision tree depth(balance scale)

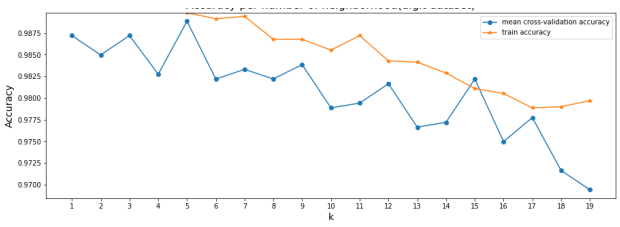

Figure 13: Accuracy per number of neighborhood(digits dataset)

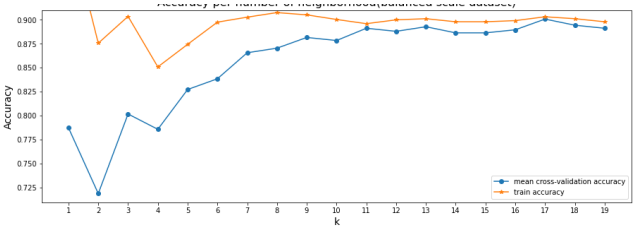

Figure 14: Accuracy per number of neighborhood(balanced scale)

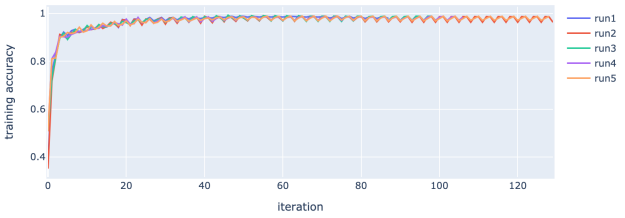

Figure 15: Training Accuracy with Best Hyper-parameter for Digits Dataset

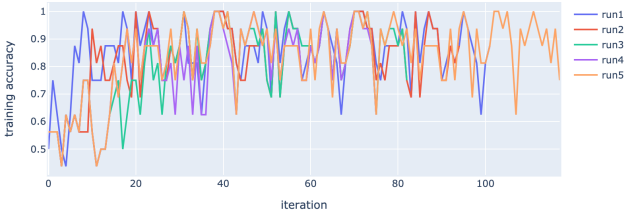

Figure 16: Training Accuracy with Best Hyper-parameter for balance-scale Dataset

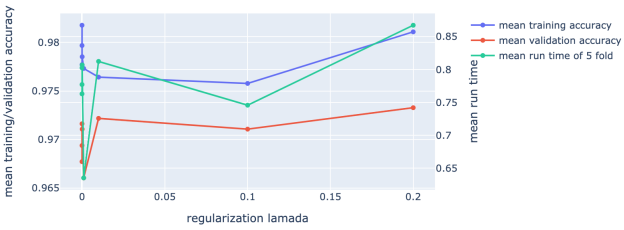

Figure 17: Performance and run time of when varying regularization lamada on digit dataset

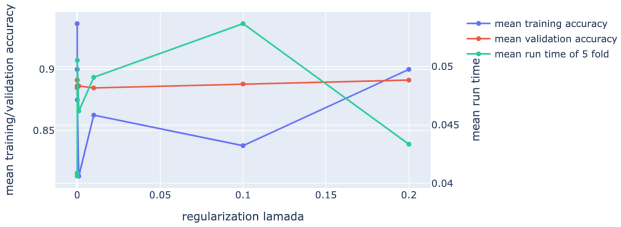

Figure 18: Performance and run time of when varying
regularization lamada on balance-scale dataset

Param	Train	Validation
0.200000	0.981	0.973
0.100000	0.976	0.971
0.010000	0.976	0.972
0.001000	0.977	0.966
0.000100	0.979	0.971
0.000010	0.978	0.969
0.000001	0.980	0.972
0.000000	0.982	0.968

Param	Train	Validation
0.200000	0.900	0.891
0.100000	0.838	0.888
0.010000	0.863	0.885
0.001000	0.812	0.886
0.000100	0.875	0.891
0.000010	0.900	0.891
0.000001	0.900	0.886
0.000000	0.938	0.885

Table 2: Numerical value of training and validation accuracy when varying regularization lamada on digit and balance-scale dataset

References

[Don18] Niklas Donges. *The Logistic Regression Algorithm*. Apr. 23, 2018. https://machinelearning-blog.com/2018/04/23/logistic-regression-101/.

[Ng+] Andrew Ng et al. *Softmax Regression*. http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/.