Multi-Class Logistic Regression and Gradient Descent

COMP 551: MiniProject 2

Ian Tsai, Zhiying Tan, Teresa Lee

Fall 2020

Abstract

In this project, we trained our self-implemented multi-class logistic regression models for label predictions on three datasets. By implementing our simple grid search, we found the optimal hyperparameters for our model and other compared models. We then compared the training and validation accuracy with other models, namely multi-class logistic regression models with L2 regularization, K-Nearest Neighbors (KNN) classifiers, and decision tree classifiers. Our results seem positive that our model tends to perform better than other classification models.

Introduction

The objective of this project is to investigate the training and validation accuracy of multi-class logistic regression models and compare them with other classifiers. We investigated the performance of all models on three datasets, the Digits dataset, the Iris dataset, and the Vehicle dataset. The project involved with three major tasks: multi-class logistic model with gradient descent and simple grid search implementations, hyperparameters and training/validation curve analysis, and comparison against other classifiers.

Related work include [1], which examine a new classifier based on the multi-class logistic regression model for interval symbolic data where learning set is described by a feature vector. Another related work [2] apply two state-of-art optimization techniques including conjugate gradient (CG) and BFGS to train multi-class logistic regression.

Datasets

The three datasets are from the Scikit-Learn datasets and the OpenML datasets. The digits dataset [3]from Scikit-Learn includes 1797 8×8 images with ten classes, where each class refers to a digit from 0 to 9. The Iris dataset [4] from OpenML is the third version that includes four features and two types of flowers: negative and positive. The vehicle dataset [5] from OpenML contains 18 features and four types of cars. Other than the Iris dataset, the class distributions of the two datasets seems to be uniform (see Appendix Figure 1(1)).There are no missing values in all datasets.

During preprocessing, the labels in all datasets are converted to one-hot vectors. For the digits dataset, the resulting dimension of the class label becomes 1797×10 from 1797×1. For the Iris and vehicle datasets, the resulting dimensions become 150×2 from 150×1 and 846 × 4 from 846 × 1, respectively. Data normalization is applied to each dataset. Each feature is converted to have 0 mean and 1 standard deviation. This enables the features to have similar ranges and scales so that gradient descent would converge faster. Features with larger scales require more gradient descent steps to converge.

Results

1 Hyper-parameters Selection

We implemented a multi-class logistic regression model that takes an optimizer from the gradient descent class. The gradient descent class includes a mini-batch optimization with momentum. Each fitted and predicted label is calculated by a softmax function. We used a simple grid search algorithm to search for the optimal hyper-parameters for the gradient descent class. These hyper-parameters include momentum, learning rate, and batch size. Table 1 shows the best hyperparameters we obtained from our simple grid search.

Result	batch size	learning rate	momentum	lambda	Accuracy
Digits Dataset	100	0.1	0.95	10	96.67±1.46
Iris Dataset	38	0.05	0.90	1	100.00±0.00
Vehicle Dataset	212	0.1	0.99	1	78.23±3.35

Table 1: Optimal Hyper-parameters obtained from Grid Search

For momentum, the range is chosen to be $[0.9, 0.99]$. By letting the range to be larger than 0.9, we could avoid the affect of outliers in each step. However, we cannot take momentum that are larger than 1 or the gradient of current step will not be considered. For batch size, we choose the range between 1 and sample size N to test the performance of mini-batch gradient descent. Adding 1 and N to the training range would also enable us to test the performance of gradient descent and stochastic gradient descent respectively. For learning rate, we choose the range $[0.005, 0.5]$ because if the learning rate is too small, it may take a large amount of steps to converge. If the learning rate is too large, the training process will not converge.

2 Training and Validation Curves

By comparing the training and validation curve (see Appendix Figure 3 (3)), the accuracy of training dataset is generally higher than validation dataset in all three datasets. For the digits dataset, the training and validation curves for momentum and batch size have less than 0.5% gap difference. For the Iris dataset, the training and validation curves for momentum and batch size are the same, which is 100% accuracy. This suggests either the class labels are distinctly different from each other or the model is overfitted. For the vehicle dataset, training and validation curves have similar behaviours and less than 0.5% gap difference. An examination of the termination condition also shows that the accuracy has a wavy-behaviour in the first 200 iterations. Further exploration on identifying this wavy-behaviour is needed.

When considering the learning rate with the range from 0.9 to 0.99, the learning rate curve remains almost unchanged as the learning rate increases. However, as the batch size increases, the variation of accuracy becomes much smaller. Also, as the batch size gets smaller, the training process is more likely to be influenced by the outliers which will lead to some gradient-descending misdirection. As a result, when the batch size increases, the accuracy have fewer variations.

By increasing the momentum, the gradient increases the influence of gradient in the previous steps. Normally speaking, it should gives us a much more stable path towards the global minimum. However, the accuracy of vehicle dataset tends to have more fluctuations. It may imply that the previous step's gradients also have large fluctuations.

3 Iteration vs Accuracy

According to the plot of iteration vs accuracy(see Appendix Figure2(2b)), we can see that at the very beginning, the accuracy increases while the number of iteration increases. However, for all 3 datasets, the accuracy remains almost unchanged after 100 steps of iterations, so we performed an experiment on digits dataset to test the termination condition by setting max iteration at 200 and epsilon $=1e^{-8}$.The termination condition in the gradient descent class is set to the end when the validation error does not decrease after T iterations. We experimented with different choices of T to check the convergence speed on the three datasets by taking max iteration =200 and epsilon $=1e^{-8}$. In general, we observed that the models converge faster as T increases. Table 2 shows the termination conditions of the digits dataset. However, the accuracy remains stable in general.

T	Number of iteration before converge(mean±std)	Validation accuracy(mean±std) %
None	200±0	95.60% ± 0.35
5	20±4	95.06% ± 1.02
10	47±10	95.16% ± 0.71
20	84±20	95.55% ± 0.21

Table 2: The Convergence Speeds of Digits Dataset with different choices of T.

4 L2 Investigation

To investigate any possible overfitting, we added L2 penalties on all models. The optimal lambda value that we found for the three models are 10 (digits), 1(Iris), and 1(vehicle) (Table 1). Adding lambda does not improve the accuracy for the Iris and vehicle dataset. In fact, it has significantly lowered the accuracy of the model for vehicle dataset, which means that further investigation is needed. In the case of digits dataset, we found that adding an L2 regularization increases the validation accuracy and decreases the difference between validation accuracy and training accuracy. The accuracy is 96.32% with L2 regularization while the accuracy is 94.23% without the regularization (See Figure 2(2b) for a detailed comparison between the two models for digit datasets). This might indicate the model for digits datasets is overfitting, whereas the models for other two datasets are not.

5 Model Comparison

Furthermore, we compared the accuracy performances of our models with KNN classifiers and decision tree classifiers. For all three models, we used our implemented simple grid search to find the optimal hyper-parameters. We performed 5-fold cross validation to estimate and compare the modelling performance. Table 3 shows the average accuracy and standard deviation of the multi-class logistic regression models, KNN classifiers, and decision tree classifiers.

Average Accuracy% ± standard deviation%			
Result	KNN	Decision Tree	Our Model
Digits Dataset	98.27± 0.21	79.88± 2.75	96.67±1.46
Iris Dataset	100.00± 0.00	100.00± 0.00	100.00±0.00
Vehicle Dataset	66.23± 1.68	66.59± 3.71	78.23±3.35

Table 3: Average accuracy for KNN, decision tree, and our model on all three datasets

For the digits dataset, KNN has the highest accuracy. For the Iris dataset, all models achieve 100% accuracy and 0 standard deviation. For the vehicle dataset, all models have a relatively lower accuracy then the previous 2 dataset. The Logistic model has the highest overall accuracy. The hyper-parameters for the three KNN classifiers are N-neighbours equal to 10 (digits), 2 (Iris), and 4 (vehicle), which are align with the number of classes in each dataset (10, 2, and 4 respectively). The considered hyper-parameters for a decision tree classifier are maximum number of depth, minimum number of sample leaf, and maximum leaf nodes. The decision tree classifiers for the three models take more iterations and have lower accuracy than the other two models. The result shows that our multi-class logistic regression model performs better than imported KNN and decision tree models in these 3 dataset.

Discussion and Conclusion

The Iris dataset has 100% accuracy as there are only 2 labels in total. However, Iris dataset may contain overfitting problem and we need a much larger dataset to verify our assumption. Digits dataset has good performance with more than 95% accuracy on average but less accurate than KNN model. The vehicle dataset has less than 80% accuracy and the accuracy contains a lot of fluctuation during the training process. It may be caused by the cost function which is not a convex function.

Future improvements: For the vehicle dataset, we should spend some time on feature engineering, selecting the most correlated features for training rather than using all of them. For the Iris dataset, it is better to collect more data and redo the training process. Also, since this dataset includes versions with different class labels, we may need to compare the classifying performances by using different class labels. An unsupervised learning such as K-Mean may help us explore which class label provides better grouping.

There are a lot more hyperparameters rather than batch size, momentum and learning rate which may be used to increase our model's performance. Also, the training time of simple grid search takes quite a long time, using randomly grid search could be a possible approach to reduce run time. Moreover, confusion matrices suggest that the modelling performances are better at certain classes compared to other classes.

Statement of Contribution

All members contributed to the codes and write-up report. Ian Tsai contributed to required analysis. Zhiying Tan contributed to all parts. Teresa Lee contributed to softmax regression and required analysis.

Bibliography

[1] A multi-class logistic regression model for interval data. Accessed Nov. 20, 2020. [Online]. Available: https://www.researchgate.net/publication/224399930_A_multi-class_logistic_regression_model_for_interval_data 1

[2] On optimization of multi-class logistic regression classifier. Accessed Nov. 21, 2020. [Online]. Available: https://www.researchgate.net/publication/272035986_On_Optimization_of_Multi-Class_Logistic_Regression_Classifier 1

[3] scikit learn.org. Scikit-learn digit dataset. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html 1

[4] https://www.openml.org. Openml iris dataset. [Online]. Available: http://scikit-learn.org/stable/datasets/index.html#openml 1

[5] ——. Openml vehicle dataset. [Online]. Available: http://scikit-learn.org/stable/datasets/index.html#openml 1

Supplementary Data and Plots

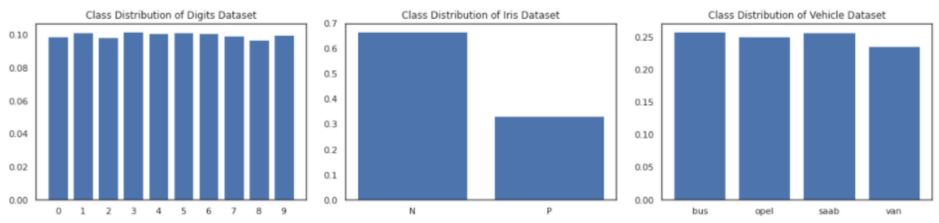
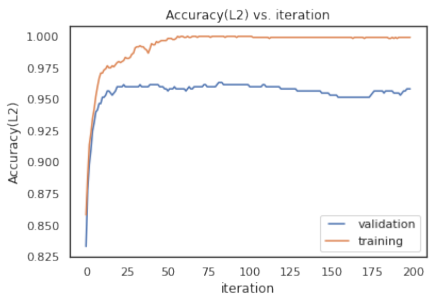
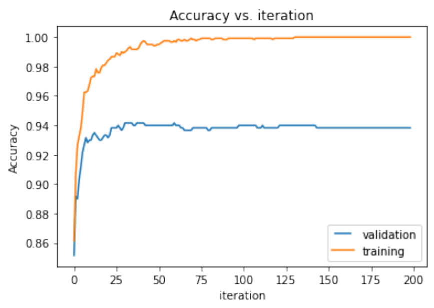


Figure 1: The Class Distribution of Digits, Iris, and Vehicle datasets.



(a) L2 Regularization



(b) None L2

Figure 2: Modelling Performances of our model and our model with L2 penalties (max iteration=200 and epsilon=1e-8).

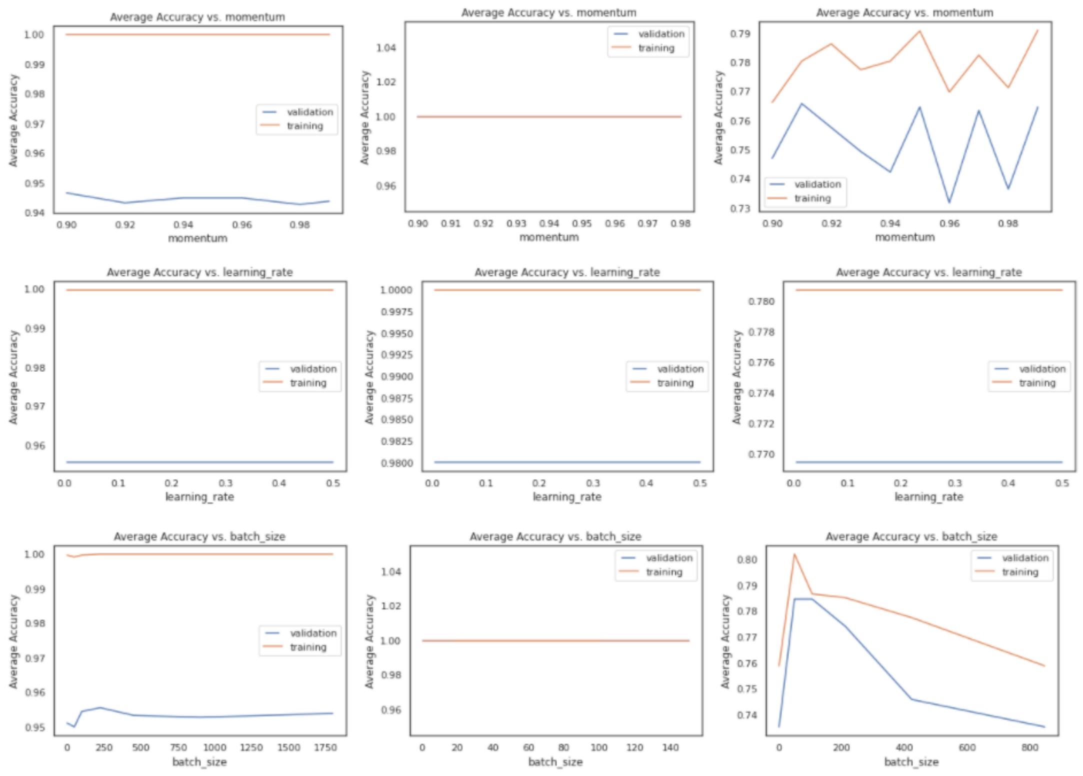


Figure 3: Training (orange) and Validation (blue) Curves for different optimization hyper-parameters in the three datasets (Top to down: momentum, learning rate, and batch size. Left to Right: digits, Iris, and vehicle datasets)

-