Multi-Class Logistic Regression and Gradient Descent

Yuelin Liu (260844113), Ziyu An(26767458),
Zilong Wang (260823366)

Abstract -- **In this mini project, we investigated the performance of Softmax regression model [1] and gradient descent optimizer with momentum[2]. The model was implemented with two datasets - the Digits dataset and the Mice dataset. First, we implemented grid searching with 5-fold cross validation to find a good combination of hyperparameters. Second we varied those hyperparameters (learning rates, batch-sizes and momentum parameter beta) to see how the accuracy and run-time of our model changed. Third, we added a termination condition to reduce run-time. Finally, KNN was implemented to compare the performance of our models. For each dataset, we generated a set of hyperparameters with the best predictive accuracy. Our model resulted with over 95% predictive accuracy, close to the result of KNN.**

I. Introduction

A. Task Description

The main goal is to build up a Softmax Regression model and a gradient descent with momentum optimizer. First, we used simple grid search to obtain an acceptable combination in terms of run-time and accuracy. Our own 5-fold cross validation was created and Cross-Entropy was used during grid searching to examine the performance of our model. We varied one of the 3 hyperparameters (learning rates, batch-sizes and momentum) while keeping others constant based on the best combination to analyze its influence on the model's performance, exploring a better set of hyperparameters. Then we added a termination condition and found its impact on the run-time and accuracy. The optimization process stops automatically if the validation error has not been decreasing in the past T iterations. Finally, we compared the performance of our model with the off-the-shelf KNN model.

B. Important Findings

We used cross entropy in the process of descent gradient and assumed cross entropy as validation error. But in order to have a better understanding, we created a predictive accuracy method where we can find the ratio of the number of correct predictions to all samples given weights. **So when we discuss accuracy later, we are referring to predictive accuracy** instead of (1 - cross entropy). The result shows that our model is highly accurate on both data sets. For the digits dataset, it usually reaches over 94% across different hyperparameter combinations, and the highest accuracy is 97%. And the accuracy of mice dataset usually reaches over 97% across different hyperparameter combinations, and the highest accuracy is 99%.

As for run-time, the learning rate is the most significant, if it is smaller than 0.005, each iteration will take 10s in both dataset. The running time of our model on the Mice dataset is longer than the Digits dataset in general since it is more complex; As for accuracy, each hyperparameter seems to have a threshold beyond which the accuracy will not change and show fluctuations. The smaller termination T will reduce our running time while keeping the accuracy mainly unchanged, however, a bigger T does not have such influence.

II. Data process

A. Data acquiring and preprocessing

1. The Digits dataset [3]

There are 1797 instances, 64 features and 10 classes in the Digits dataset, which contains the 1797 8*8 images of handwritten digits from 0 to 9. To understand the Digits dataset, we generated one of the data points which is an (8*8) image with a total of 64 pixels. It can be deduced that 64 features correspond with 64 pixels. 10 classes represent the digits from 0 to 9 and each class has roughly 180 images.

2. The Mice dataset [4]

This dataset contains 1080 instances representing 1080 mices, 77 features (proteins measured in the cerebral cortex) and 8 classes (control and Down syndrome mice exposed to context fear conditioning). In the Mice dataset, 528 instances have 'NaN' values for certain genes, and all 'NaN's are replaced by 0. It is also worth noting that the instances that belong to the same class are grouped together. We found this by printing and visualizing the first 20 labels. In order to avoid overfitting during cross validation, we applied random permutation on the Mice dataset. The goal for this dataset is to assess associative learning of mice.

III. RESULTS

Initially, we assigned a relatively large range for each hyperparameter and narrowed down the range after analyzing the original result. With a few experiments, we chose 2 sets of grid search ranges for Digits and Mice datasets respectively. The best hyperparameters for two datasets are **[learning rate 0.01, batch size 30, beta 0.92]** with accuracy 0.967 for Digits, and **[learning rate 0.03, batch size 15, beta 0.92]** with accuracy 0.990 for Mice dataset.

Using the above combinations of hyperparameters, we implemented 5-fold cross validation on the two datasets, and recorded the accuracy in each fold and the total run-time. For the Digits dataset: the run-time is 7.949 secs, predictive accuracy on the validation set of the 5-fold is [0.928, 0.883, 0.944, 0.933, 0.905], average is 0.92. And for the Mice dataset, runtime is 12.814 secs, accuracy of the 5-fold is [0.977, 0.981, 0.981, 0.986, 0.991], average is 0.98.

Note that the accuracy above is the predictive accuracy, we also recorded Accuracy_CE (refers to the accuracy in terms of cross entropy cost on validation set, calculated by 1 - cross entropy) of the 5-fold: [0.760, 0.535, 0.838, 0.770, 0.652] on Digits and Accuracy_CE of the 5-fold: [0.756, 0.601, 0.843, 0.809, 0.625] on Mice dataset.

In addition, we plotted the cross-entropy cost (Fig. 1) of each optimization step in training. We chose one loss function for digits set and one Accuracy_CE of training and validation curve for mice dataset to represent the result for 5-fold cross validation. (Blue is for training and orange for validation). The final cost is extremely low, at nearly 0, which means the Accuracy_CE on the training set is bigger than 0.99. However, during validation, Accuracy_CE decreased by around 0.2 each fold, dropping to around 0.7-0.8 in the mice dataset. Besides, the convergence was faster in Digits dataset than Mice.

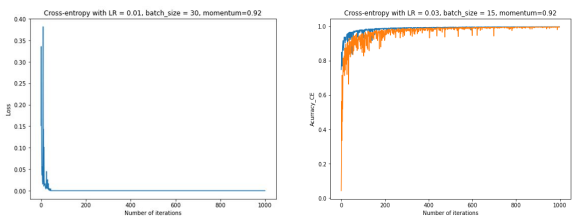

Fig. 1.Corss entropy/Acurracy_CE with best hyperparameter combos vs. Number of iterations on both datasets

While keeping others constant as in the bundles of good hyperparameters, we varied each of the three hyperparameters (learning rates, batch sizes, and momentum parameter beta), still with 5-fold cross validation, and found that the accuracy and runtime behaved quite differently for both datasets. **Now we use predictive accuracy which is more reliable than train and validation curves.**

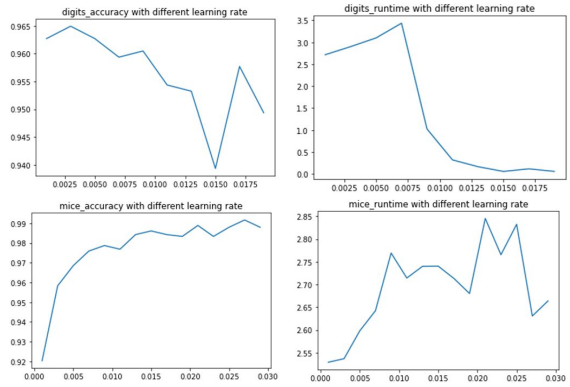

Fig.2. Outcome accuracy, runtime vs. Learning rate on two datasets

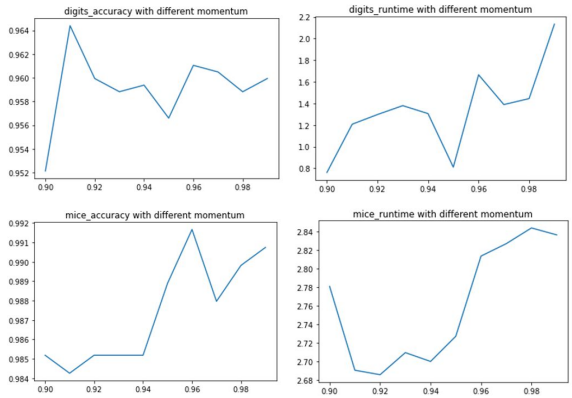

Fig.3. Outcome accuracy, runtime vs. Momentum on two datasets

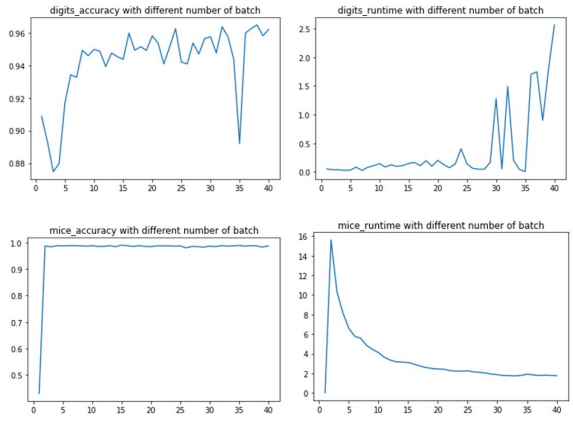

Fig.4.Outcome accuracy, runtime vs. Number of batch on two datasets

It can be seen that in all cases (Fig. 2-4) the accuracy demonstrates fluctuation in a small range.

Different sets have different trends and stability when we vary the hyperparameters. For example, a lower learning rate is good for digits set but it is not for the other set. One significant thing for the mice dataset is that the accuracy remains unchanged after it reaches while the runtime significantly drops in terms of batch size. Therefore we updated the batch sizes for Mice dataset to 30 and reran the 5-fold cross validation on both datasets. The run time for the Mice dataset dropped to 9 secs while the accuracy almost remained constant. However, the runtime of the digits dataset dropped without any changes of hyperparameter spontaneously. It was because the dataset is shuffled every time when creating mini-batches. So the runtime may have to do with the sub parts of the dataset rather than hyperparameters.

Beside the three hyperparameters above, we investigated the impact of the termination condition T on the performance of our model, so we varied T while keeping the other three hyperparameters the same as in the good bundle.

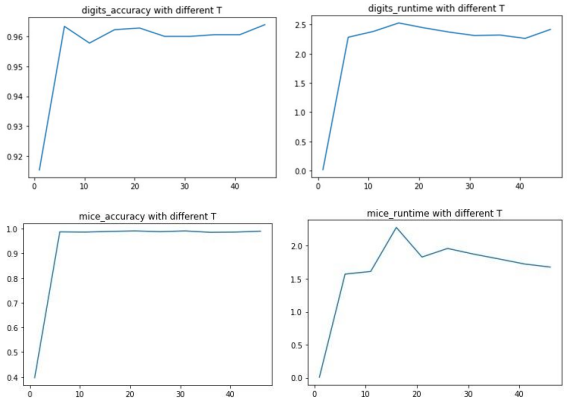

Fig. 5. Outcome accuracy, runtime vs. Termination iteration on two datasets

It can be seen (Fig. 5) that T only has a great impact before a certain value, here 5 on both datasets. After that value, its impact can be ignored. Furthermore, an decrease of T will significantly reduce the running time (from plus 2 to nearly 0 on both datasets) but the accuracy will also decrease slightly(from 0.95 to 0.9 in digits set and from 0.95 to 0.5 in mice dataset).

For the comparison with KNN, our model is slightly inferior in terms of accuracy. Since the only hyperparameter in KNN that dominates is the number of neighbors K, we varied K to see how the accuracy changed. The best result occurred when K= 1 with best_accuracy= 0.995 for Mice dataset and best K= 3 with best_accuracy= 0.988 for Digits dataset, which apparently exceeds our model.

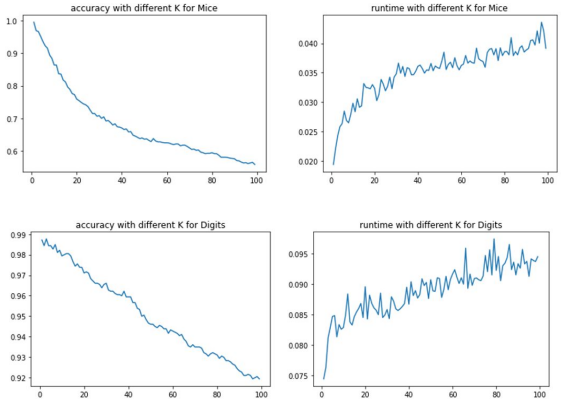

Fig. 6. Outcome accuracy, runtime vs. different K on two datasets

As K increases (Fig. 6), the accuracy keeps a trend of dropping and run-time has a trend of increasing for both dataset, although with normal fluctuations.

IV. Discussion and Conclusion

To conclude our project, we found that the Softmax regression model using gradient descent optimizer with momentum is a powerful categorization model. We achieved over 95% predictive accuracy on both datasets. However, it requires the optimization for hyperparameters to achieve better performance. It may take some time to find out the best hyperparameters. We can use a termination T less than 5, but cost is a decrease of accuracy.

We decided to turn all 'NaN' to 0 in the Mice dataset instead of dropping them because the instances containing 'NaN' accounted for almost half of the dataset. We chose to keep them in order to compute more data. Since every NaN in genes reduces the corresponding classes of behaviour, the result is relatively accurate. On the other hand, not dropping these instances might make our model biased because any $x_i = 0$ leads to $x_i * w_i = 0$, which reduces the $\Sigma x_i * w_i$. [5]

Mice dataset shows a relatively slower run-time than digits dataset. It is possibly because the number of features and classes are both larger in Mice dataset than Digit dataset.

V. Statement of Contributions

Zilong Wang: Implemented momentum, model comparison and contributed to the report.
Yuelin Liu: Implemented the softmax regression, helped in data analysis and report writing. Created k-fold and score methods.
Ziyu An: Implemented the grid search and cross validation, proof reading for the report.

VI. REFERENCES

[1]Wierenga, Rick. Softmax Regression from Scratch in Python, 22 Feb. 2020, rickwierenga.com/blog/ml-fundamentals/softmax.html.

[2]“Multiclass Logistic Regression from Scratch.” Multiclass Logistic Regression from Scratch - The Straight Dope 0.1 Documentation, 01 Nov. 2018, gluon.mxnet.io/chapter02_supervised-learning/softmax-regression-scratch.html.

[3] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[4] Vanschoren, Joaquin. “MiceProtein.” OpenML, 11 Aug. 2017, www.openml.org/d/40966.

[5] Bamtak, “Multi class logistic regression in numpy.”https://github.com/bamtak/machine-learning-implemetation-python/blob/master/Multi%20Class%20Logistic%20Regression.ipynb