Fall 2020 COMP 551 Mini Project 2 (Group 15)

Gustavo Alonso Patino Ramirez and **Alexandre Martini Santos** [b] and **Hao Ju**[c]
[a] Student number: 20168196, gustavo.patinoramirez@mail.mcgill.ca
[b] Student number: 260798352, alexandre.martinisantos@mail.mcgill.ca
[c] Student number: 260815992, hao.ju@mail.mcgill.ca

Abstract

In this project, we implemented multi-class logistic regression over 2 different datasets, namely the Digits dataset [2] and the Rats Protein Expression dataset[1], from scratch, using gradient-based optimization. We explored the effect of different hyperparameters over the performance, and found that increasing batch size would generally speaking improve validation accuracy, while accuracy improving was less sensitive by simply increasing learning rate or momentum. If hyperparameters properly selected, the loss and accuracy curve of training and validation will tend to converge with increasing iterations. Having explored the effects of hyperparameters, we then applied the best-performing hyperparameters to the 2 datasets respectively, and compared their performance with other classifiers, KNN and decision trees respectively. In the smaller, easily trainable digits dataset, our model achieved a similar validation accuracy of 0.97 in the test set, compared with 0.977 achieved by KNN with $k = 3$ neighbours; while in the larger Mice Protein Expression dataset with more features, our model achieved a 0.86 accuracy, compared with 0.78 accuracy of the decision tree classifier. This relatively more consistent accuracy implies higher robustness and ubiquitousness in our model.

1 Introduction

In this project, we implemented multi-class logistic regression over 2 different datasets, namely the Digits dataset [2] and the Rats Protein Expression dataset[1], from scratch, using gradient-based optimization algorithms.

We first performed softmax regression over the Digits dataset, with mini-batch optimization using gradient descent with momentum. We tested our optimizer with different values of potential hyperparameters learning rate λ, batch size, and momentum β under 5-fold cross validation. Throughout the process, we found that larger batch size have a higher convergence speed and presents significantly higher accuracy, as it makes each gradient descent update more stable and less stochastic. However, this comes at the cost of higher computational workload, which needs to be taken into account when dealing with large datasets. We also noticed that smaller batch size, when combined with high learning rate and momentum, may lead to unstable validation accuracy (oscillation) apart from the expected lower accuracy. We found that our model performs best with hyperparameters $\lambda = 0.05$, batch size = 1024, and $\beta = 0.5$. We then implemented our model with the optimal hyperparameters as described above, and observed validation loss and validation accuracy steadily converging to a satisfactory level, indicating no need to using early stopping. Eventually, our softmax classification method achieved an accuracy of 0.968 with 5-fold cross validation over the test set in this small, easily trainable dataset, which is very close to the 0.977 accuracy of the KNN model with $k = 3$ neighbors.

We then performed the same process over the larger and more complex Mice Protein Expression dataset, and found the best hyper-parameters to be $\lambda : 0.5$, batch size : 512, $\beta : 0.7$. This set of hyperparameters achieved a 0.86 accuracy with the test set, compared with merely 0.78 accuracy with decision tree algorithm, which indicates a higher ubiquitousness of our model.

2 Datasets

We used two datasets in this project. The first dataset is the digits dataset[2], a set of normalized bitmaps of handwritten digits from a preprinted form, where each one of 1797 data-points is an input matrix of 8x8, and each element has feature integers in the range 0 to 16 and 10 classes. There are no missing values, and all datapoints already processed, ready to analyse.

The second data set [1] consists of 1080 expression levels of 77 proteins or protein modifications that produced detectable signals in the cerebral cortex of mice. The eight classes of mice are described based on characteristics such as genotype, behavior and treatment. Initially the 8 classes are not categorical so in the preprocessing it was necessary to map each of the classes to an entire label. Additionally, the data set contained null values which were replaced by the mean value of the column of the corresponding feature.

3 Results

3.1 Data set 1: digits data set

Initially, we apply the softmax regression method for the multiclass classification problem defined by the data set 1 (digits data set). The hyper-parameters to be optimized in the algorithm are: the learning rate λ, the size of the batch size used in the gradient descent optimization process and the momentum parameter β. Since it is necessary to optimize the three parameters, we build a simulation grid with three parameters for learning rate $\lambda = \{0.5, 0.05, 0.005\}$, three for batch size $= \{128, 512, 1024\}$ (usually power of two), and three for momentum $\beta : \{0.5, 0.7, 0.8\}$. In table1 we can see some results of the optimization process

Learning rate (λ)	batch size	β	run time	Accuracy
0.5	128	0.5	2.13	0.5
0.5	1024	0.8	0.66	0.73
0.05	128	0.5	1.86	0.94
0.05	1024	0.5	0.69	0.97
0.005	128	0.5	2.15	0.94
0.005	1024	0.8	2.15	0.81

Table 1: Performance for different hyper-parameters, running time and accuracy using 5-fold cross validation

calculated using 5 fold cross validation, and we conclude that the best combination of hyper-parameters is defined by $\{\lambda = 0.05, batch\_size = 1024, \beta = 0.5\}$. The complete set of results can be seen in the jupyter notebook annexed to the work. A cru-

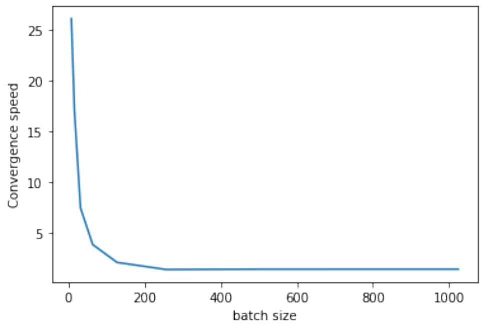

Figure 1: Hyper-parameters: $\lambda : 0.05$, $\beta : 0.5$. Batch run time

cial parameter in the optimization process is the batch size. From Fig.2 we can infer that there is a compromise between batch size and accuracy. At one extreme we see that large batches have a higher convergence speed (see Fig.1) since we are making use of vectorized computation in CPU (the gradient function and soft-max loss calculation were written completely in vectorized form without using for loops). At the other extreme with small batch sizes we get more updates but no vectorization advantage, which leads to a lower convergence speed. The disadvantage of using bigger batch size in larger data sets is that it can be computational memory exhausted, so using this approach we could improve the convergence to the global minimum of the loss function at the expense of computationally more expensive calculations. Simi-

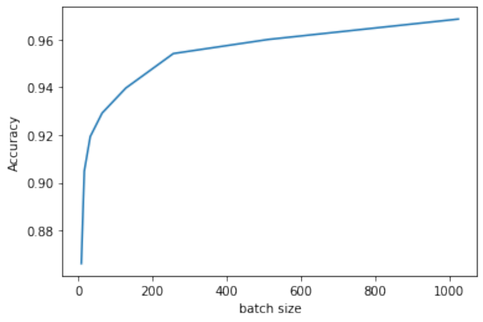

Figure 2: Hyper-parameters: $\lambda : 0.05$, $\beta : 0.5$. Batch accuracy

larly in Fig.2 we see that large batch sizes present a considerably higher accuracy since each gradient descent update is more stable and less stochastic. Fig.3 shows the accuracy of the validation set during the optimization process for each one of the folds.

On one side we have the combination of small batch size, high learning rate and momentum (see Fig.3) which produces unstable values in some folds (red) and low accuracy values in others (blue and yellow) causing a mean accuracy considerably lower than the best accuracy found in the cross-validation process. On

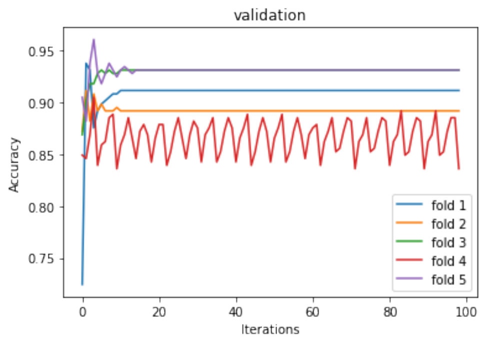

Figure 3: Validation accuracy calculated at each fold. Hyper-parameters: $\lambda : 0.5$, batch size : 128, $\beta : 0.8$.

the other hand, in Fig.4 we have the results calculated using the best hyper-parameters combination found with 5-fold cross validation, remarking that in this case the accuracy in each of the folds is much more homogeneous increasing monotonically and reaching a plateau.

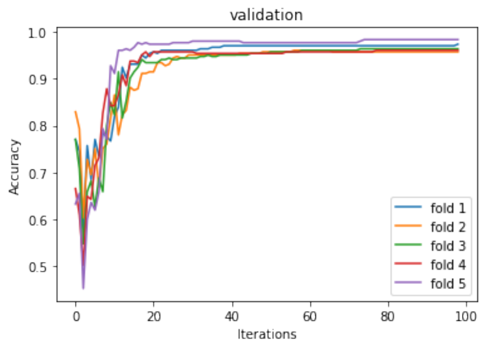

Figure 4: Validation accuracy calculated at each fold. Hyper-parameters: $\lambda : 0.05$, batch size : 1024, $\beta : 0.5$.

In Fig.9, Fig.10,Fig.11, Fig.12 in appendix we can observe the behavior of the loss in validation and training sets in each one of the folds. In general terms the loss presents unstable behaviors in some folds when the hyper-parameters are not optimal and more stable behaviors when calculated in the best combination of hyper-parameters. Globally, we can see that the loss has a decreasing trend reaching small values which is compatible with the high precision values achieved in the validation. In Fig.5 and Fig.6 the loss and accuracy for the validation and training set are depicted. From Fig.5 we see that the loss of validation and training fall monotonically after a certain point so that it was not necessary to use early stopping. Additionally, in Fig.6 the accuracy exhibits a monotonically increasing behavior reaching a maximum value of 0.968. Since we use 5-fold cross validation, the performance of the unseen test data set is similar, reaching a final accuracy of 0.97.
The results obtained with our soft-max classification method were contrasted with the KNN method of the sklearn package. The number of neighbors was optimized using 5-fold cross validation finding that the most convenient value was $k = 3$ neighbors. With

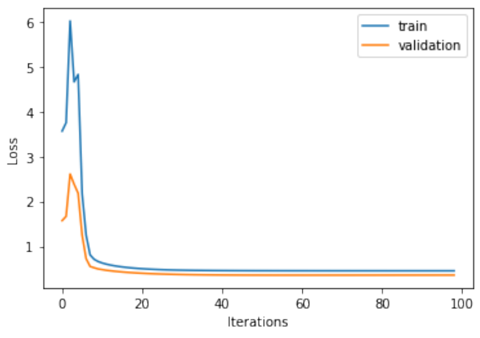

Figure 5: Performance using the best hyper-parameter: λ : 0.05, batch size : 1024, β : 0.5. Train and validation loss

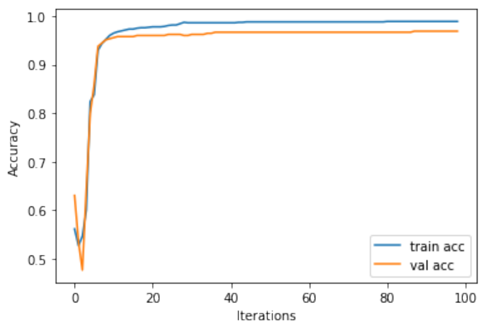

Figure 6: Performance using the best hyper-parameter: λ : 0.05, batch size : 1024, β : 0.5. Train and validation accuracy

this hyper-parameter, the performance in the test set was calculated obtaining an test accuracy of 0.977. This high value is not surprising given that the digits data set is an small and relative easy trainable data set.

3.2 Data set 2: Mice Protein Expression

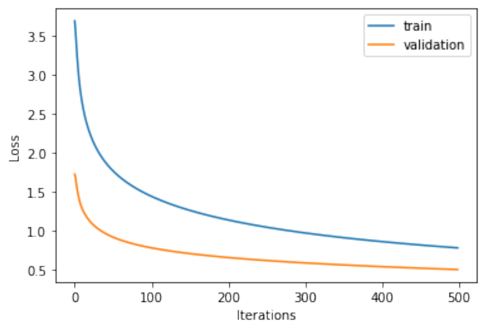

Figure 7: Performance using the best hyper-parameter: λ : 0.5, batch size : 512, β : 0.7. Train and validation loss

For data set 2 we trained the samples with 77 features that represent protein expressions and we tried to predict 8 brain expressions in mice. In Fig.7 and Fig.8 we see the loss and accuracy of the training and validation sets in the training process. Again, as in the case of data set 1, it was not necessary to use early stopping since we did not observe an inflection towards a growing behavior in the loss of validation. The accuracy reached in the unseen test set was 0.86, which is a good result when contrasted with the result obtained with decision tree from sklearn, which obtained a maximum accuracy of 0.78 after calibrating the parameters using 5-fold cross validation.

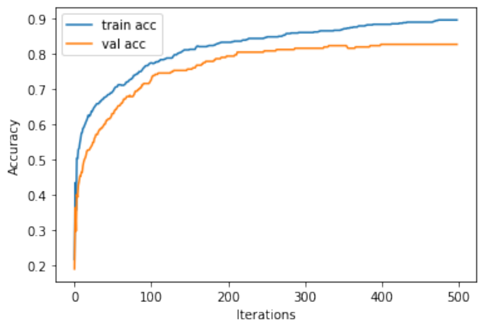

Figure 8: Performance using the best hyper-parameter: λ : 0.5, batch size : 512, β : 0.7. Train and validation accuracy

4 Conclusion

There are a few key takeaways from this projects that we would especially like to emphasize.

- Among the 3 hyperparameters of the optimizer learning rate λ, batch size and momentum β, increasing batch size would typically enhance performance, while increasing learning rate and momentum might not (as for whether performance will be enhanced with increased learning rate and momentum, it depends on the case). However, increasing batch size will significantly increase processing load and time, which needs to be considered with large datasets.
- With properly selected hyperparameters, training and validation curve tend to converge with increasing iterations.
- Performance of softmax regression with gradient descendent optimizer varies between datasets. In small, easily trainable datasets like the Digits dataset, its performance may not be able to exceed other classifiers like KNN, but could still reach a similar validation accuracy; however in larger, more complex datasets, such as our Mice Protein Expression dataset with a total of 77 features and 8 categories, it can still retain a relatively satisfying validation accuracy.

5 Statement of Contributions

Gustavo, Hao and Alexandre contributed equally in the construction of the code the pre-processing of the data and writing the report.

References

[1] C. Higuera, K. J. Gardiner, K. J. Cios, Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome, PLOS ONE 10 (6) (2015) 1–28.

[2] scikit-learn developers, scikit: The digit dataset.
URL `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html`

6 Appendices

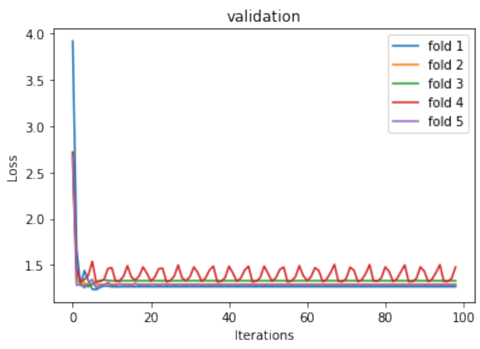

Figure 9: Validation loss calculated at each fold. Hyper-parameters: λ : 0.5, batch size : 128, β : 0.8.

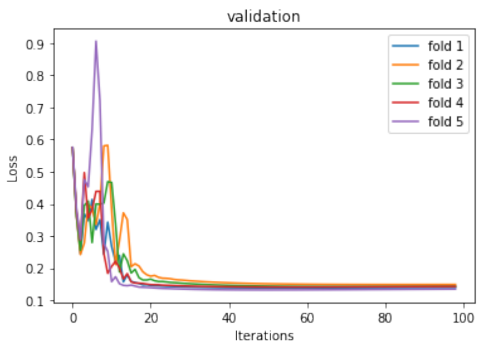

Figure 10: Validation loss calculated at each fold. Hyper-parameters: λ : 0.05, batch size : 1024, β : 0.5

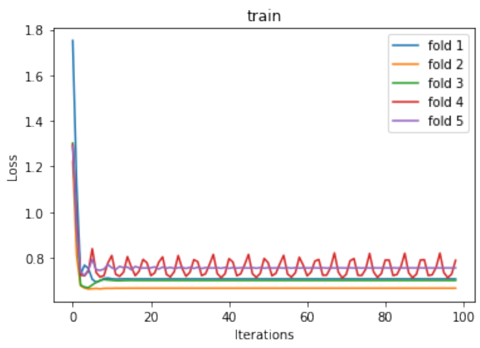

Figure 11: Train loss calculated at each fold. Hyper-parameters: λ : 0.5, batch size : 128, β : 0.8.

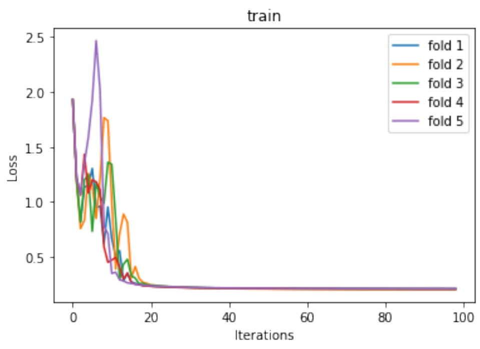

Figure 12: Train loss calculated at each fold. Hyper-parameters: λ : 0.05, batch size : 1024, β : 0.5