COMP 551: Mini Project 2

Andrei Dragoi[a], **Weiming Guo**[a], **and Viet Hoang**[a]

[a]Department of Computer Science, McGill University, Montreal, Quebec, Canada

Softmax regression, also known as multi-class logistic regression, is a generalization of the commonly used logistic regression model. It is not limited to binary classifications, and it is able to classify categorical data. For this project, we implemented our own classifier using softmax regression. We compared its performance to another classifier, decision trees, and we found that our classifier is as accurate as decision trees. We also analyzed the effects of hyper-parameters (batch size, learning rate, and momentum) on the performance of classification. Higher batch sizes causes an increase in runtime without improvements in accurracy, and high momentum decreases accuracy. Both high and low learning rates result in a decrease in performance.

Logistic regression is one of the most widely used classifiers in academia. Unfortunately, it is only able to classify binary variables and fails in categorical classification problems. Softmax regression, a generalization of logistic regression, addresses this problem. The main objective of this project was to implement a softmax regression model that could be used across a variety of datasets and whose performance would be competitive against other classifiers. Three datasets were used to test the accuracy of the model. The first two dealt with images of handwritten digits, and the third dataset consisted of satellite images of the earth's surface, with the true soil type representing the label of this dataset. The motivation behind choosing a second digits dataset was to determine whether our model can replicate the same level of accuracy when presented with similar data. The third dataset was chosen to present a different and more novel problem than digit classification. To fit the softmax regression model to the data, we determined the optimal parameters for the model by minimizing the cross-entropy cost function. We minimized the cost function using stochastic gradient descent (SGD) with mini-batches and momentum. We found that our implementation produced very accurate classifications with all three datasets having at least 75% accuracy. We also analyzed the effects of the hyper-parameters (mini-batch size, learning rate, momentum) on the performance of the models. Increasing mini-batch size past 10 did little to improve accuracy while also worsening the runtime. Learning rates that were too low (10^{-4}) or too high (>1) resulted in poorer classification. Finally, high momentum (>0.9) slightly reduced accuracy. When comparing our model with decision trees we found the two to have comparable accuracies with our model being better for the 2 digit datasets and the decision tree slightly outperforming our model in the satellite image dataset.

Datasets

None of the datasets used required extensive pre-processing, in large part due to a lack of any missing values. The first of the digits dataset was taken from the scikit learn library, and contained around 1800 total samples. Each sample was an 8x8 image of a handwritten digit (0-9), where each of the 64 parameters of the model represented the greyscale intensity of a pixel. The second digits dataset from the Simeion Research Center was similarly structured with regards to the parameters, except the inputs were 28x28 images of handwritten digits, meaning each of the 1600 data points had 256 attributes. The third dataset came from the University of California Irvine and had just under 6500 data points, each of which was a 3x3 satellite image of the earth's surface with 36 attributes in total. The label of each data point was the true soil type of the region that was photographed. The only pre-processing all these datasets required was one hot encoding each label.

Results

Hyper-parameter optimization. We performed a grid-search to determine a good combination of hyper-parameters. In this grid-search, we tracked the validation accuracy at each iteration of gradient descent. If the accuracy has not increased for the 20 iterations, we terminated gradient descent early and returned the set of weights that produced the highest validation accuracy. Below is a table detailing the results of the grid search.

Dataset	Batch size	Learning rate	Momentum	Mean accuracy	Std. Dev.
Digits	128	0.01	0.5	91%	2.2
Semeion	128	0.1	0.5	80%	1.8
Satimage	16	0.5	0.5	76%	1.6

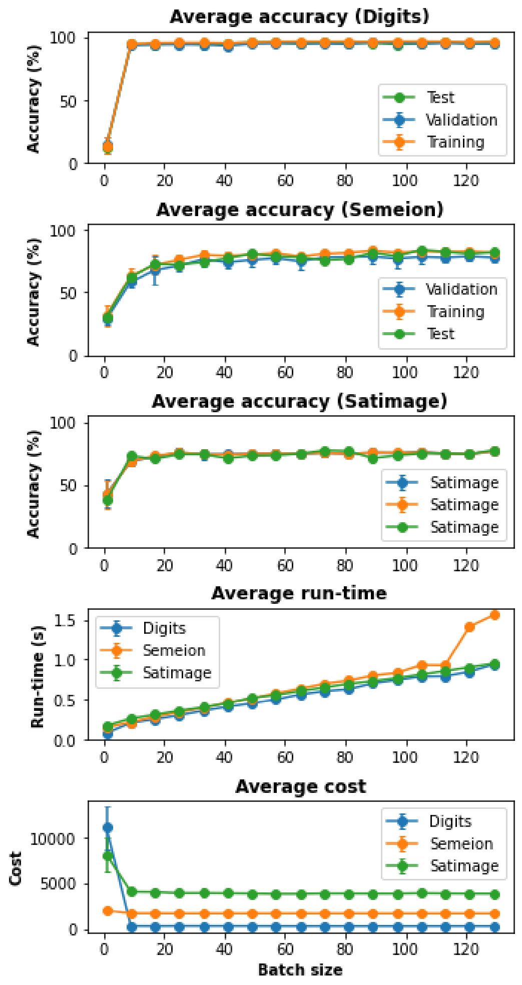

Fig. 1. The first 3 plots show the accuracy with respect to batch size. The third plot shows runtime and last one is the cost (cross-entropy). Error bars represent standard deviation. For each dataset, the 20% of the data was removed for the test set. The remainder was used for 5-fold cross-validation.

Analysis of batch size. We were interested in the effect that batch size has on the performance of the model. Figure 1 shows how certain metrics of performance vary with batch size when the other optimal values are held constant. From Figure 1, we see that the test, validation, and training accuracy remains constant and high for nearly all batch sizes. There appears to be a plateau in accuracy and increasing the batch size has little effect. This is further supported by the fact that the cost does not go any lower as you increase the batch size. From the figure, we also see that the run-time increases with batch size. Therefore, it would be more beneficial to keep the batch size down as that improves run-time without sacrificing accuracy. Only models with a batch size of 1 have poor accuracy. It is likely that at this batch size, the algorithm does not converge, and this is evidenced by an extremely high cost that does not seem to be minimized. A possible explanation for poor convergence at small batch sizes is that the learning rate is not small enough to reduce the oscillations. A learning rate that is too high for the batch size will result in the gradient descent algorithm to frequently overshoot the minimum of the cost function, resulting in poor accuracy and convergence.

Analysis of learning rate. Figure 2 shows the level of performance of the model with respect to the learning rate (other hyper-parameters are fixed). We see that at very low learning rates (on the order of 10^{-4}), the accuracy is slightly lower. This likely due to the learning rate being so low that the model does not have enough time to converge before the maximum number of iterations (1000) is reached. At high learning rates the accuracy drops tremendously, likely because the model keeps overshooting the global minimum and is unable to converge. This is supported by the very high cost values at high learning rates. Run-time appears to be mostly unaffected by learning rate. Learning rate isn't involved in the calculation of the gradient, which where run-time is affected the most.

Analysis of momentum. Figure 2 shows how performance is affected by changes in momentum. Momentum appears to have little effect on the accuracy, which remains quite high throughout. Accuracy does decrease slightly as momentum approaches 1. This could be due to the learning rate being too high. As the algorithm approaches the minimum of the cost function, a high learning rate combined with a high momentum will cause the algorithm to overshoot. This is supported by an increase in cost that results when the momentum is close to 1. The same way a ball doesn't stop immediately when it reaches the bottom of a valley, a model with high momentum will be difficult to halt. If we implemented a scheduled learning rate, it would decrease the size of each subsequent step, counteracting the effects of momentum.

Comparison against other classifiers. We compared the performance of our classifier against scikit-

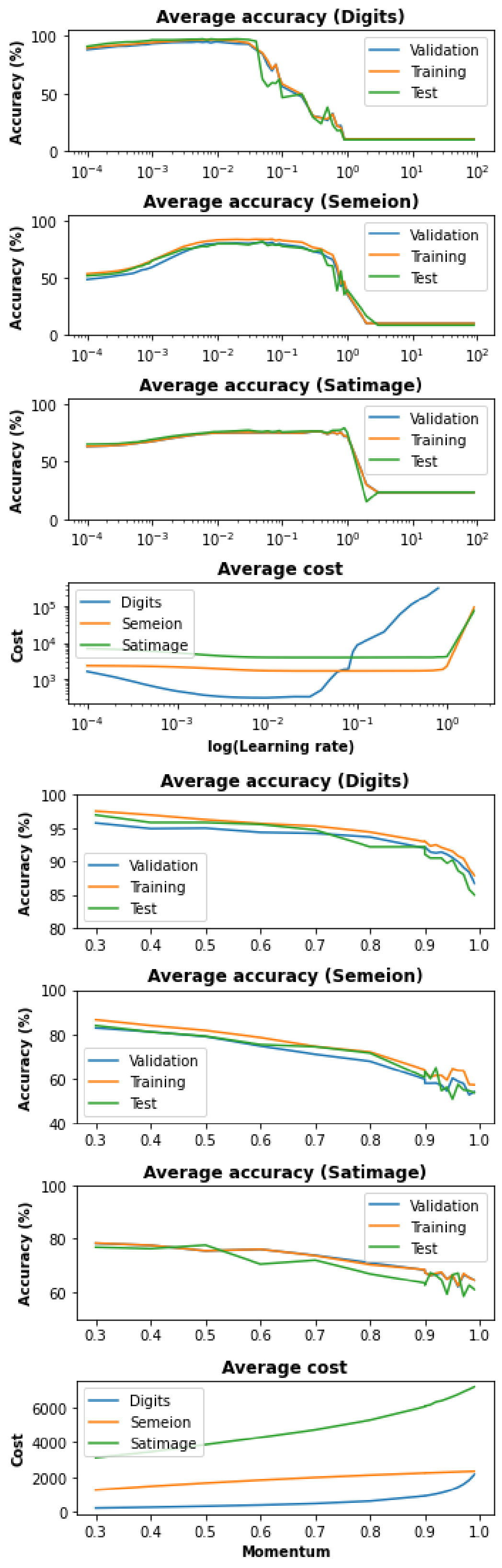

Fig. 2. (From top to bottom) The first four plots show how certain metrics of performance vary with respect to the log of learning rate. The last four is the same but for momentum. Standard deviation bars are omitted for visual clarity. Run-time was not included, because we found that it did not change as you vary learning rate or momentum. For each dataset, the 20% of the data was removed for the test set. The remainder was used for 5-fold cross-validation

learn's decision tree classifier. We ran a grid search on the hyper-parameters (max tree depth, minimum number of leaves, etc.) to find the optimal combination. Figure 3 shows the validation accuracies of different datasets. There is a small difference in accuracy between our implementation of softmax regression and decision trees. Our model is marginally better than decision trees for the Digits and Semeion dataset and worse for the Satimage dataset.

Discussion and Conclusion

Our implementation of softmax regression produces accurate classifications when comparied to scikit-learn's decision tree. We chose the Semeion dataset to demonstrate that our classifier is able replicate the same level of accuracy when presented with a dataset similar to Digits. Satimage was selected to show that our model can handle different datasets with different objectives for classification (i.e not classifying digits). One thing that we did not explore in this paper is how the interaction between hyper-parameters can affect accuracy. For example, in many cases, a mini-batch size of 1 will diverge except when the learning rate is small enough and the momentum is large enough. Our analysis (holding 2 hyper-parameters constant and varying the third) assumes that the hyper-parameters operate independently, which we cannot necessarily assume.

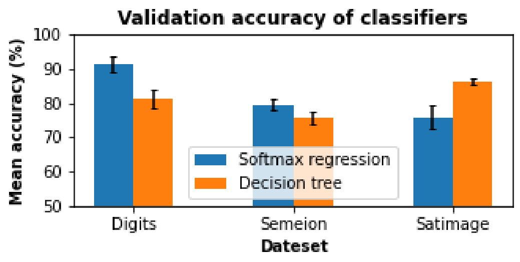

Fig. 3. The bar graph shows the mean validation accuracy of softmax regression and decision trees for all three datasets. Error bars represent standard deviation.

Statement of Contributions. All three members contributed equally to the project and were involved in all steps of the project. Andrei preprocessed the data and implemented GradientDescent.Weiming optimized the parameters and compared with other classifiers. Viet implemented SoftmaxRegression and the early Termination Condition.

References

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.

Semeion Research Center of Sciences of Communication, via Sersale 117, 00128 Rome, Italy Tattile Via Gaetano Donizetti, 1-3-5,25030 Mairano (Brescia), Italy.