COMP 551 - Mini Project 2
Multi-Class Logistic Regression and Gradient Descent

David Castonguay (260804528), Marco Guida (260803123), Sean Smith (260787775), ECSE and Computer Science McGill University

***Abstract*—The essence of this project to implement multi-class logistic regression (with softmax) and then compare its performance to other common classification algorithms, namely: K-Nearest Neighbours (KNN), Naive Bayes, and SVC. To compare the different classifiers, we used the digits and wine datasets from OpenML. For the digits dataset, KNN achieved the highest validation accuracy, followed by our logistic regression classifier, and then SVC and Naive Bayes, respectively. For the wine dataset, Naive Bayes performed the best, followed by KNN and our logistic regression classifier, which both achieved similar results.**

I. Introduction

The task: Implement logistic regression (with softmax) from scratch, record the training and validation accuracy, and compare these results to those of KNN, Naive Bayes, and SVC. Two program classes were implemented to build the logistic regression model: A logistic regression model *LogisticRegression*, and an optimizer *GradientDescent*. The logistic regression model contains the functionality to train the model using the gradient descent optimizer and some training data, as well as predict the label for unseen data after the model has been fitted. Gradient descent used momentum and mini-batch optimization to accelerate the algorithm, which was especially useful for the digits dataset which contained much more data than the wine dataset.

Once implemented, logistic regression was ran on both datasets using 5-fold cross validation in the training phase. The accuracy and cost (calculated using cross-entropy [1]) of the model was reported on both the training and validation set for both datasets in order to observe the effects of adjusting the model's hyper-parameters. Afterwards, KNN, Naive Bayes, and SVC were trained and ran on both datasets in order to report the accuracy of these common models on the validation set. This was done in order to obtain a quantitative comparison of the different models' ability to correctly classify input data. While the validation accuracy is the primary metric used in this comparison, there are other metrics which could be used but were not quantitatively reported. For instance, our model requires significantly more time to train, potentially making other off-the-shelf classifiers more appealing in practice. A further discussion of the comparison results is discussed later in the report. After running all the tests, a plot is generate to illustrate the performance comparison between different classifiers. While some models performed slightly better than others, in general, all models performed relatively similarly on the validation sets, with the worst performing classifier usually performing within 10 percent of the best classifier (across both datasets).

II. Dataset

Two datasets from Scikit-Learn were used in this project, namely the *Digits Dataset* and the *Wine Dataset*. As the goal of this project is to construct a logistic regression model, using these readily accessible datasets had the advantage of requiring minimal pre-processing of the data. The only pre-processing that was performed was the normalization of the inputs, such that all inputs were ranging between zero and one after processing. The reason for this was due to the softmax equation computing infinity when the exponents were moderately large. This created broken data (i.e. NaNs) in our model and required us to scale down the input values. Further, the targets were one-hot encoded in order to be usable with softmax.

The *Digits Dataset* contained 1797 digit samples, each representing an 8x8 bitmap image (i.e. 64 total features per sample) of a single digit. Prior to normalization, each of the input features of a sample ranged between zero and 16, denoting the color of that specific pixel. The label of a sample was the numerical digit for the bitmap image. The *Wine Dataset*[1] contained 178 wine samples. Each sample corresponded to a particular wine and its

[1]Available at https://www.openml.org/d/187

features represented various aspects of the wine (e.g. alcohol, color intensity). The three possible numerical targets denoted the originating cultivar of a wine sample. Some simple analyses were performed to help us more concretely understand which wine features we can expect to be most important in predicting the wine label. This helped us debug our code to ensure that the model is operating as anticipated.

III. Results

A. Performance

The greatest caveat of our implementation of logistic regression was how long the gradient descent took to converge to an optimum. Some runs took up to several minutes, in comparison to other classifiers that would finish in a few seconds. By adjusting the model's hyper-parameters, the run-time performance could be improved or regressed. As illustrated in Figure 1, some hyper-parameters had a more noticeable impact on the run-time. This namely includes batch size, which directly affected the run-time by varying the amount of data to be trained, and learning rate, which smaller rates took longer to converge. While the run-time curves appear noisy, a general trend is visible for each of the hyper-parameters.

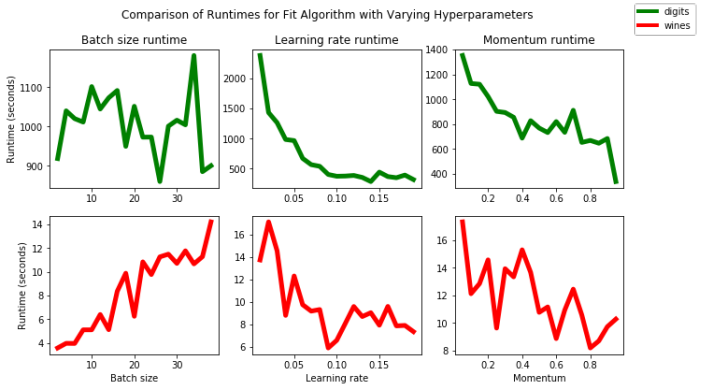

Fig. 1. Run-times for logistic regression while individually modifying hyper-parameters.

B. Hyper-Parameters Optimization

Prior to testing the performance of our model as we varied a single hyper-parameter, we did a simple grid-search to find a combination of hyper-parameters that performed well in general (which we later used for our fixed hyper-parameters as we analyze the results of changing a single hyper-parameter). It was found that a batch size of 30, a learning rate of 0.04, and a momentum of 0.2 was a good starting point for our model on both datasets. More specifically, this combination led to a training and validation accuracy between 90-95% for the *Digits Dataset* and between 85-95% for the *Wine Dataset*. The following sections of our report discuss the results when changing a single hyper-parameter at a time:

Batch size: The batch size is the number of data points from the training set (of a single cross-validation fold) used in gradient descent. As illustrated in Figure 2, a larger batch size resulted in a lower validation cost, and therefore a higher validation accuracy, as expected, for both digits and wine. The training cost and accuracy (not depicted in the figure) had similar results as well. However, the disadvantage to this was that a larger batch size caused the model to be trained much slower. So the optimal batch size for our model would be between 20-25 samples per batch as, by Figure 2, the model can predict new data with over 80% accuracy for both datasets and run around 2 times faster than configurations with a 30+ batch size.

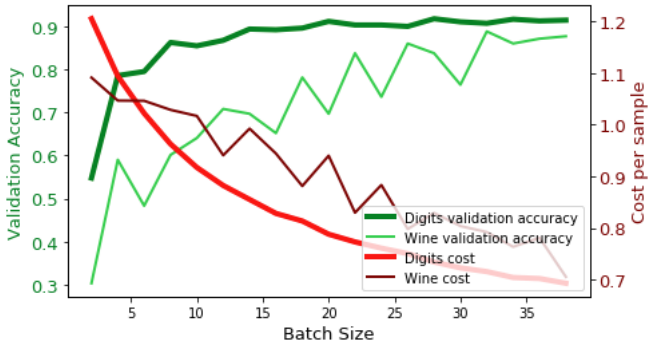

Fig. 2. Logistic regression validation accuracy and cost when only batch size is changing. Learning rate and momentum fixed at 0.04 and 0.2, respectively. The trend lines show that a larger batch size results in a higher validation accuracy.

Learning rate: The learning rate of the model determines the amount that the weights are updated at each step of gradient descent during training. Smaller learning rates require many more epochs in order to find the optimal model weights, whereas larger learning rates update the weights faster but run the risk of never converging to the optimum. As illustrated in Figure 3, the cost per sample decreased and the validation accuracy increased as the learning rate increased. Interestingly one would instead expect the opposite results, in that a lower learning rate finds the better weights during training. However, we suspect that this is the case for us due to the termination condition used during gradient descent, which terminates the algorithm if the cost has not decreased after 25 (our chosen T)

iterations. For small learning rates, this means that most likely gradient descent terminated too early and did not update the weights enough to produce great predictions.

Momentum: Momentum is used to smoothen gradient descent, allowing for a more efficient convergence towards the optimum. As seen in Figure 4, the validation accuracy decreased as the momentum increased, indicating that momentum had a negative effect on the results. A possible reason for this effect is that the term multiplied by the momentum (based on past iterations) may have driven noisy oscillations with higher momentum [2]. In this case our specific termination case would take effect much too early as cost was not necessarily decreasing, resulting in poor weights and bad predictions on new data. Potential solutions may be the use of Adagrad instead as varying learning rates replace the usage of the problematic momentum. Further, we could test variations of momentum-based gradient descent, such as RMSprop and Adaptive Moment Estimation.

C. Classifier Comparison

To better understand our results, we compared our logistic regression classifier to other common classifiers. We chose to compare our results with those of KNN, Naive Bayes, and SVC (image classifier so only useful for digits).

As illustrated in Figure 5, our logistic regression model yielded a very similar validation accuracy to the other classifiers across both datasets (90.9% for digits and 87.6% for wine). The only classifiers that marginally outperformed our model were Naive Bayes for wine (93.8%), KNN for digits (95.7%), and SVC for digits (89.6%). Therefore, logistic regression with softmax demonstrated its ability to perform well in the presence of outliers and noisy data (primarily from wine data) [3].

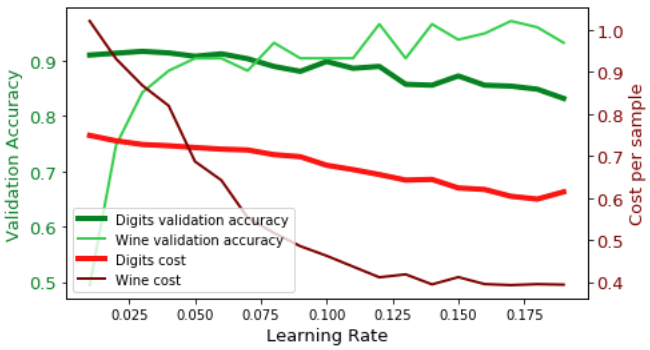

Fig. 3. Logistic regression validation accuracy and cost when only learning rate is changing. Batch size and momentum fixed at 30 and 0.2, respectively. Cost per sample decreases and the validation accuracy increases as the learning rate increases.

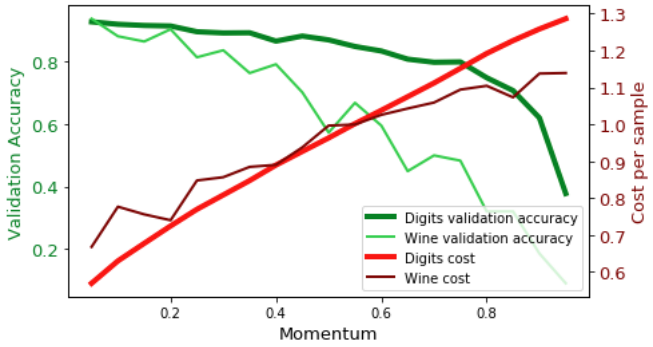

Fig. 4. Logistic regression validation accuracy and cost when only momentum is changing. Batch size and learning rate fixed at 30 and 0.04, respectively. Cost per sample increases and the validation accuracy decreases as the momentum increases.

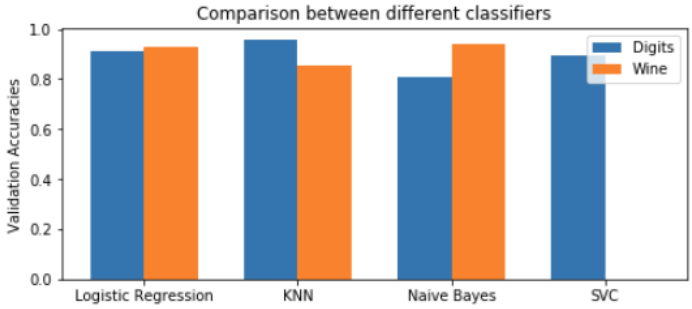

Fig. 5. Comparing validation accuracy between our model and KNN, Naive Bayes, and SVC.

IV. Discussion and Conclusion

The logistic regression classifier along with a mini-batch gradient-descent with momentum, is a classifier that can withstand multiple modifications of its hyper-parameters. Though, as any classifier, there are thresholds to which beyond, the validation accuracy falls exponentially. The optimal combination of hyper parameters is: Learning rate=0.06, Momentum=0.1, batch size=25. This combination achieves maximum validation accuracy without choosing values which result in extremely long run-times. Even though, we tested two fundamentally different datasets, the classifier in question outputs a validation accuracy remarkably similar for both datasets. Compared to other classifiers, the logistic regression can perform just as well, or better in terms of accuracy but at the cost of a much longer run-time.

V. Statement of Contributions

All: report, gradient descent. David: classifier comparison. Marco: logistic regression. Sean: run-time comparison.

REFERENCES

[1] S. Ravanbakhsh, "Logistic and softmax regression," vol. Applied Machine Learning, no. 12, 2020.

[2] R. Bhat, "Gradient descent with momentum." https://towardsdatascience.com/gradient-descent-with-momentum-59420f626c8f, October 2020.

[3] Y. Ren, P. Zhao, Y. Sheng, D. Yao, and Z. Xu, "Robust softmax regression for multi-class classification with self-paced learning," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, p. 2641–2647, AAAI Press, 2017.