Mini-batch Gradient Descent Softmax Regression and Hyperparameter Optimization

COMP 551 Mini Project 2

Mike Gao - 260915701
Isaac Meadowcroft-Grijalva - 260752377
Linhui Huang - 260831346

ABSTRACT

In this project, we analyze how the performance of *softmax regression* depends on the method's parameters of optimization. To gain a good set of the parameters of optimization, we first do a grid search across different *gradient descent* batch sizes, learning rates and momentums. Using these estimates we then plot how the training accuracy, validation accuracy, and runtime of *softmax regression* changes as we vary each parameter individually. Lastly, we compare the performance of *softmax regression* with *decision trees* to draw insight on which method is preferable when using real-world data. In this analysis, we use three open-source datasets: a *digits* dataset that aggregates 8x8 pixel images of digits; an *Iris* dataset that aggregates measurements from three types of Iris flowers; and a *letters* dataset that aggregates images of computer-generated letters and numerical features associated with each letter.

INTRODUCTION

This study analyzes how the performance, in terms of run time, training accuracy and validation accuracy, of *softmax regression* depends on the batch size, learning rate and momentum of the *gradient descent* algorithm used in the method's optimization. Our analysis is done on three open-source datasets: the *digits* dataset[1] available on Scikit-Learn which contains 1797 8x8 pixel images of hand-written digits classified by digit; the *Iris* dataset[2] provided by OpenML which contains 150 samples of Iris flowers (classified as either Iris setosa, Iris versicolor or Iris virginica) along with each sample's sepal length, sepal width, petal length and petal width; and the *letters* dataset[3] provided by OpenML which classifies 20,000 images of computer-generated letters according to 16 primitive numerical attributes extrapolated from the images. In our analysis, we first do a grid search to find a good combination of *gradient descent* parameters. Using these estimates we then plot how training and validation accuracies vary over iterations, and how the performance of *softmax regression* varies across the different optimization batch size, learning rate and momentum parameters. Lastly, we compared the accuracy of *softmax regression* with *decision trees* to characterize the better method for real-world data.

Datasets

Since the datasets were already preprocessed, we imported them from sklearn.datasets and used them directly.

Scikit-Learn Digits Dataset

This dataset contains 1797 samples of 8x8 pixel images of hand-written digits. It has ten classes for each of the digits from 0 to 9. Each digit/class has roughly 180 instances and each instance has 64 features arising from the instance's 64 pixel image.

OpenML Iris Dataset

This dataset contains 150 samples of Iris flowers classified as either Iris setosa, Iris versicolor or Iris virginica. Each class of flowers has 50 instances and each instance aggregates the sepal length, sepal width, petal length and petal width of the sample flower in centimeters.

OpenML Letters Dataset

This dataset contains 20,000 instances, each representing an image of a computer-generated letter, and 26 classes, each representing one of the letters of the alphabet. It was created by randomly distorting images of each of the letters of the alphabet in various fonts into files of 20,000 unique stimuli and then converting these stimuli into 16 primitive numerical attributes. Each letter in the dataset has roughly 770 instances, each containing 16 features corresponding to the datapoint's 16 primitive numerical attributes.

RESULTS

The termination condition of our mini-batch *gradient descent* algorithm was to end the optimization if the mean squared validation error had not been decreasing for 20 iterations, so it returns the model with highest validation accuracy rather than the model at the last step of gradient descent. To check how convergence for gradient descent varies as learning rate changes, we also implemented a check which concludes the run is convergent if the L2 distances of gradients of successive iterations stay less than 1 for more than 5 iterations after achieving minimum mean squared validation error. It was reported as the percentage of convergent runs among the 5 runs in 5-fold cross-validation.

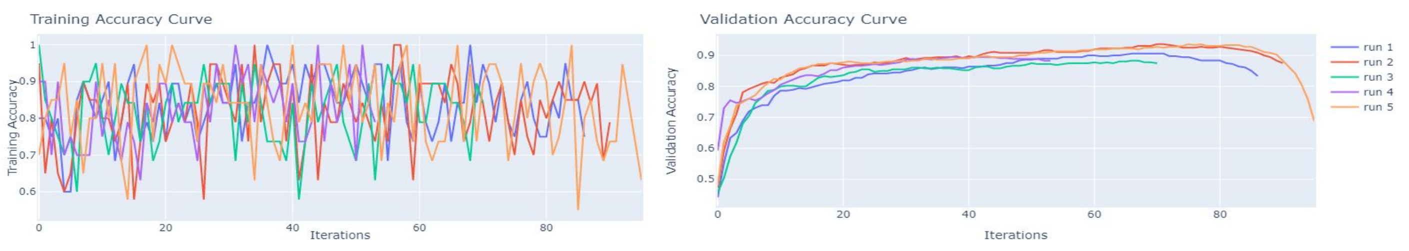


Figure 1: digits dataset gradient descent Implementation with optimal parameters, 5-fold cross-validation (Left: validation accuracy, Right: training accuracy)

In order to find a good set of mini-batch gradient descent hyperparameters, we implemented a method that performs a grid search across different combinations of the *gradient descent* parameters and returns the parameters associated with the maximum accuracy. Here, the accuracy associated with each *gradient descent* implementation was estimated using the average validation accuracy of five-fold cross validation. Table 1 shows the vector of batch sizes, the vector of learning rates and the vector of momentums that our grid search algorithm searched over for each of the three datasets. Table 2 shows the optimal batch size, learning rate and momentum parameters obtained from our grid searches for each of the three datasets.

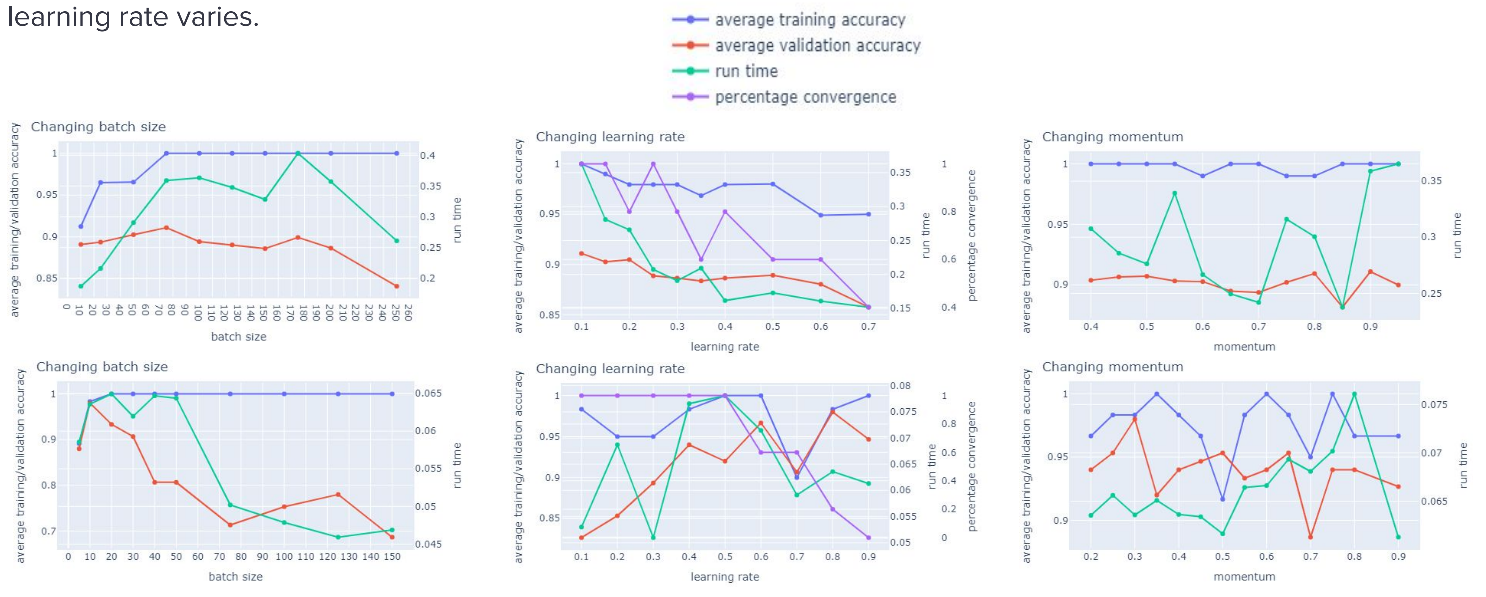
Dataset	Batch Sizes	Learning Rates	Momentums
Digits	`[10, 25, 50, 75, 100, 125, 150, 175, 200, 250]`	`[0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.6, 0.7]`	`[0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95]`
Iris	`[5, 10, 20, 30, 40, 50, 75, 100, 125, 150]`	`[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]`	`[0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.9]`
Letters	`[5, 10, 25, 50, 75, 100, 150, 200]`	`[0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5]`	`[0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.98]`

Table 1: The vectors of batch size, learning rate and momentum parameters that were grid searched over for each dataset

Dataset	Maximum Validation Accuracy	Batch Size	Learning Rate	Momentum	Run Time
Digits	0.9110	75	0.10	0.90	0.362820s
Iris	0.9800	10	0.80	0.30	0.083408s
Letters	0.5112	5	0.40	0.80	5.447611s

Table 2: Optimal batch size, learning rate and momentum parameters, with the corresponding maximum validation accuracy and run time obtained from grid searches of digits, iris, and letters datasets

Having found a good combination of parameters for each dataset, we then explored how the run time and performance of our algorithm changed as we varied each parameter individually. Figure 2 visualizes how the training accuracy, validation accuracy and run time of our *gradient descent* algorithm varied across either different batch sizes, learning rates, or momentums, while the other 2 parameters remained constant. Training and validation accuracies were reported as the average accuracy of the training or validation set from each run in 5-fold cross-validation. The plots in the middle also show how convergence of gradient descent changes as learning rate varies.



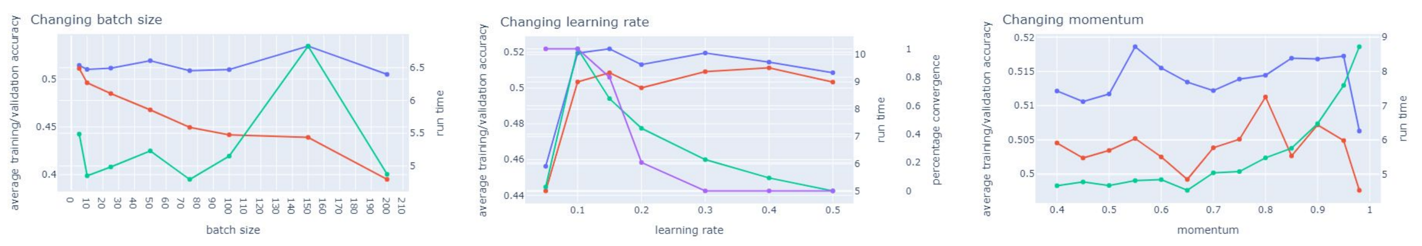


Figure 2: Variation of training accuracy, validation accuracy and run time of gradient descent across different batch sizes, learning rates or momentums, and variation of convergence across different learning rates (first row: digits dataset, second row: Iris dataset, third row: letters dataset)

In addition, we trained a *decision trees classifier* on the three datasets. The average validation accuracies of decision trees with different max depths (k) were reported for 5-fold cross-validation and plotted in Figure 3. The maximum validation accuracy of the *digits* dataset was 0.8247 at k = 17, the maximum validation accuracy of the *Iris* dataset was 0.9667 at k = 10, and the maximum validation accuracy of the *letters* dataset was 0.8558 at k = 36.

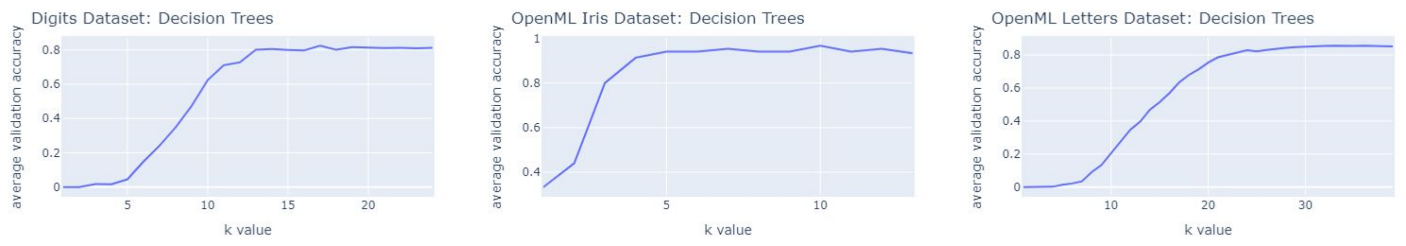


Figure 3: Average validation accuracies of decision trees with different max depths (k), 5-fold cross-validation (left: digits dataset, center: Iris dataset, right: letters dataset)

Discussion

As shown by Figure 2, validation accuracies are consistently lower than training accuracies because the model has the tendency to overfit the training data. Training accuracies were especially high with smaller datasets such as the *Iris* dataset. One interesting trend observed in the batch size plots is that the validation accuracy decreases in all datasets as batch size increases. The observation that increasing batch size on a fixed learning rate increases validation error and decreases validation accuracy is well documented. Indeed, Hoffer, Hubara, and Soudry's 2017 paper[4] establishes that, without modifying the learning rate, "the move from a small-batch (SB) to a large-batch (LB) indeed incurs a substantial generalization gap." This "generalization gap" is defined by the paper to be the observation that "when using large batch sizes there is a persistent degradation in generalization performance" meaning an increase in validation/generalization error. Another noteworthy observation is that, across all datasets, an increase in learning rate results in quicker runtime. This is to be expected since a larger learning rate results in larger steps in the direction of the gradient. These larger steps result in quicker convergence or, if the learning rate is too large, result in the *gradient descent* algorithm overshooting the minimum and diverging. The purple "percentage convergence" plot in Figure 2 demonstrates that convergence decreased as the learning rate became very large. In comparison with decision trees, softmax regression with mini-batch gradient descent achieved higher validation accuracies for the *digits* dataset and the *Iris* dataset, while decision trees had higher validation accuracy for the *letters* dataset. This indicates that depending on the dataset, either method may be preferable.

Conclusion

In this project, we analyzed how the performance of *softmax regression*, optimized with mini-batch gradient descent, depends on the method's parameters of optimization and we compared the performance of *softmax regression* and *decision trees*. In the process, we made many insightful observations. We observed that validation accuracies were consistently lower than training accuracies, no matter how we varied the optimization parameters, due to the natural tendency of *gradient descent* optimized *softmax regression* (and machine learning algorithms in general) to overfit the data. In agreement with Hoffer, Hubara, and Soudry's 2017 paper[4], we observed that across all three datasets increasing batch size on a fixed learning rate decreases validation accuracy. Furthermore, our plots showed that, across all three datasets, increasing learning rate results in quicker runtime. This is to be expected since a larger learning rate results in larger steps in the direction of the gradient and, therefore, quicker halting due to convergence or divergence. Finally, we observed that *softmax regression* had better validation accuracy than *decision trees* on two out of the three datasets indicating that depending on the dataset, either method may be preferable.

Attribution

Mike implemented *softmax regression* and *cross validation*. Linhui implemented *decision trees*, grid search and generated plots for the parameters of optimization. Isaac reviewed code and wrote the writeup.

Bibliography

[1] “The Digit Dataset.” n.d. scikit-learn 0.23.2 documentation. https://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html?highlight=digits.

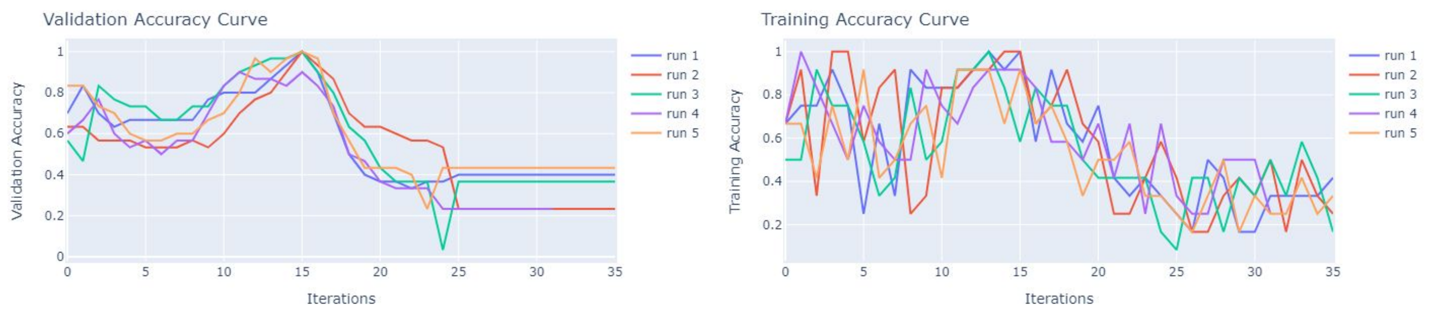
[2] “iris dataset.” n.d. OpenML. https://www.openml.org/d/61.

[3] “letter dataset.” n.d. OpenML. https://www.openml.org/d/6.

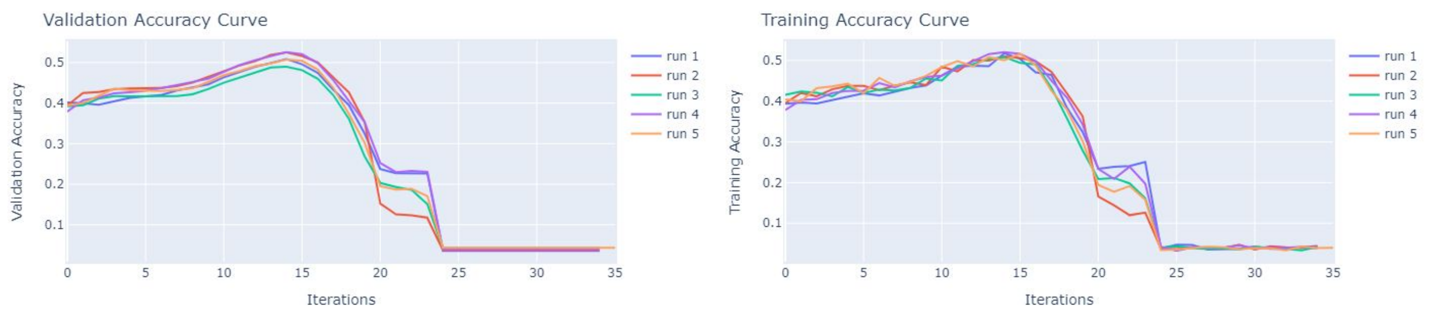
[4] Advances in Neural Information Processing Systems 30 2017; pages 1729-1739;

http://papers.nips.cc/paper/6770-train-longer-generalize-better-closing-the-generalization-gap-in-large-batch-training-of-neural-networks

Appendix



Appendix 1: Iris dataset gradient descent implementation with optimal parameters, 5-fold cross-validation (left: validation accuracy, right: training accuracy)



Appendix 2: Letters dataset gradient descent implementation with optimal parameters, 5-fold cross-validation (left: validation accuracy, right: training accuracy)