Mini Project 3: Classification of Image Data

Group 60: Alvin Chen, Ziqi Li, and Emily Tam

April 01, 2021

Abstract

In this project, we implemented a multilayer perceptron (MLP) model and trained it using mini-batch Stochastic gradient descent (SGD) methods. We explored the accuracy performance of this model when classifying image data from the MNIST dataset. In order to have a better understanding on MLP model, we conducted experiments to analyze the impact of number of hidden layers, types of activation function, regularization, adaptive gradient descent methods, and more on the testing accuracy. We found that increasing depth and width on an MLP model increases accuracy up until the convergence bound, and had surprising results when comparing normalized vs unnormalized training data performance. Finally, we also investigated and demonstrated how adaptive gradient descent methods could speed up convergence significantly.

Introduction

In this mini-project, we implemented multilayer perceptron (MLP) from scratch. The goal of this project was to gain experience with classification of image data using MLP. The dataset we work on is the MNIST dataset, which is a handwritten digits dataset that is widely used for different machine learning and pattern recognition models. It contains a total of 70,000 instances, from which 60,000 are for training and the remainder are for testing [1]. Previous attempts of maximizing performance on this classification task reached an incredible accuracy of .997, however there was significant pre-processing, data augmentation, and normalization prior to training and used deep, convolutional neural networks to achieve such an accuracy (Kayumov et. al, 2019). The best MLP performance to date was achieved by Ciresan et al, who achieved .37% test error on MLP's with depths up to 9. In our own findings using 2-layer MLP with a width of 128 units, we achieved an accuracy of about 0.978, but achieved similar results using 1-layer MLP with the same width. Additionally, we found that the unnormalized dataset had better test accuracy than the normalized dataset (refer to Task 3.4). Finally, in our journey to speed up the convergence (w.r.t. iterations) we implemented Adam (adaptive momentum) optimizer and reached convergence far quicker than plain SGD.

Datasets

The samples for each digit in MNIST is nearly evenly distributed and each image in this dataset is represented by 28x28 pixels such that each pixel has an integer from 0 to 255 representing the color intensity. For data preprocessing, we started with the raw data and vectorized the dataset to 2D: number of data x image pixels. Then, we normalized the data by using mean subtraction and standard deviation. In examining the distribution of the classes, we found that the data were relatively evenly distributed, with the class '1' having the most instances across both training and testing by a fairly large margin.

Results

Task 3.1

In Task 3.1, we created three different models: (1) an MLP with no hidden layers, (2) an MLP with a single hidden layer having 128 units and ReLU activations, (3) an MLP with 2 hidden layers each having 128 units with ReLU activations. We compared the testing and training accuracy of these three models. As shown in Figure 1A, the training accuracy was always slightly higher than the testing accuracy for each model, as expected.

As the number of iterations increased, we found that the model with 2 hidden layers had approximately the same testing accuracy as the model with a single hidden layer at 98%, but it converged slower than the single hidden layer model. The model with no hidden layer did not perform as well as the other two models with a testing accuracy of 92%, which was expected since it is only using the softmax cross entropy loss to perform optimization. We also found that it took longer for the network to converge as the dpeth increased (i.e. the MLP with no hidden layers had a higher testing accuracy early on). Thus, we can see that the network depth has a proportional relationship with the testing accuracy once the network converges. We also see that the non-linear activation functions in the hidden layers help our model to perform better.

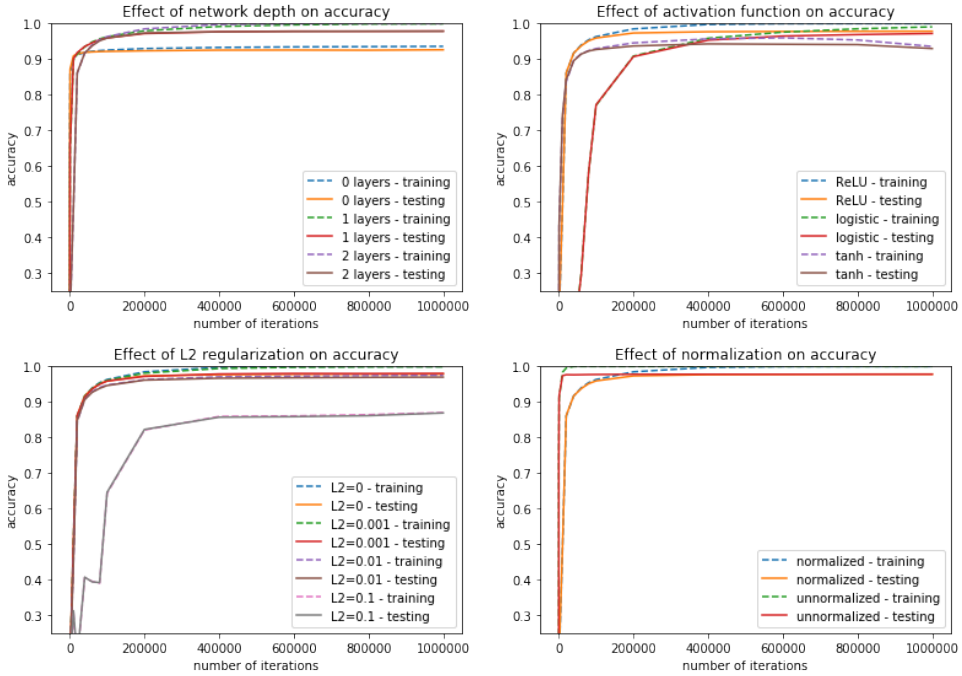

Figure 1: (A) Effect of network depth on accuracy. (B) Effect of activation function on accuracy. (C) Effect of L2 regularization on accuracy. (D) Effect of normalization on accuracy. *Note:* We graphed test and train performance as a function of the number of iterations instead of epochs. However, since number of training samples and batch size were constant throughout experiments, 1 epoch equals 600 iterations (60000 training samples divided by batch size of 100).

Task 3.2

In Task 3.2, we compared three MLP models with 2 hidden layers each having 128 units. However, the activation function of these three models were ReLU, logistic and tanh respectively. As shown in Figure 1B, the model using tanh activation function had lower testing accuracy than the other two models once it converged. Between the ReLU model and the logistic model, the ReLU model converged quicker and had better testing accuracy performance than the logistic model. The logistic model took longer to converge than the other two models. The reason that ReLU activation function performed better than others might be that the ReLU function and its derivative are both monotonic, so it could improve the performance during gradient descent better compared to other activation functions. ReLU has a zero derivative if the unit is inactive, so initialization of weights before training with random values was done to ensure active units at the beginning of optimization.

Task 3.3 (Extended for creativity)

In Task 3.3, we created an MLP with 2 hidden layers each having 128 units with ReLU activations as above. However, this time we added L2 regularization (with 0.001, 0.01 and 0.1 regularization strength) to the cost. As shown in Figure 1C, the model with regularization strength of 0.1 had the lowest testing and training accuracy. The other three models (0, 0.001, 0.01) had relatively similar testing accuracy but the model without L2 regularization performed best overall. The reason may be that adding regularization could cause overshooting during the gradient descent process for our MLP model.

Task 3.4

In Task 3.4, we created an MLP with 2 hidden layers each having 128 units with ReLU activations as above. However, this time we trained it with unnormalized images. As shown Figure 1D, the unnormalized dataset surprisingly converged much faster than the normalized dataset. The reason may be that the relative scales of pixels are already approximately equal (and in range from 0 to 255), so it is not strictly necessary to perform this additional normalization step [2].

Task 3.5

All of the graphs we have presented in this report are graphed over the number of iterations. Any line that stops short indicates that the model had converged according to the conditions set in the gradient descent function.

Creativity

Part 1: Investigate the effect of width (number of units in the hidden layers) on test accuracy

In this experiment, we constructed models with 2 hidden layers and ReLU activations, each having a different number of units in the hidden layer. The list of different widths we used is [16, 32, 64, 128]. Figure 2 shows that the model performs relatively poor when we have less units in hidden layer. As we keep doubling the number of units, the testing accuracy gradually approaches around 98%. This experiment shows that the number of units in hidden layer has a proportional relationship with the testing performance.

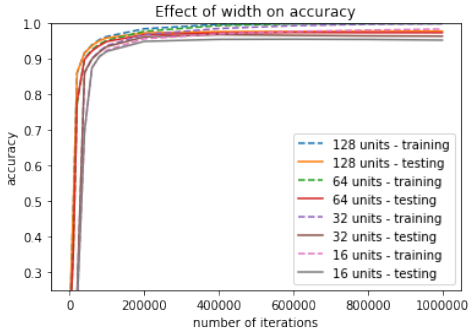

Figure 2: Effect of width on accuracy.

Part 2: Investigate the impact of training set size on the test accuracy

In this experiment, we constructed two layer models with ReLU activation function and 128 units in each hidden layer. We investigated how the change of training data size (10^k, where $k \in \{0, 1, 2, 3\}$) affects the testing accuracy. Figure 3 shows that the test accuracy increases significantly as k increases from 0 to 3. Overall, the number of units in hidden layer was found to be directly proportional to the testing performance.

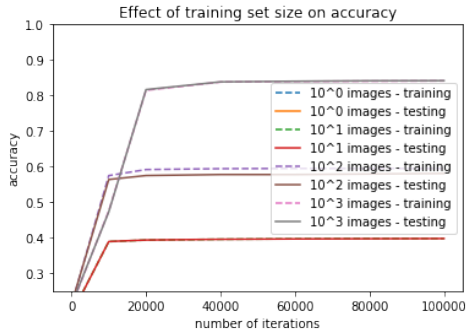

Figure 3: Effect of training set size on accuracy. *Note:* When trained with 10^0 images, testing accuracy was around 9%.

Part 3: Investigating other optimizers

In conducting experiments of Task 3, the vanilla mini-batch stochastic gradient descent performed well, but we noticed that in many trials with different hyperparameters, the maximum iterations was reached before convergence (with an epsilon $\epsilon = 1e - 8$). Additionally, our 2-layer MLP required many iterations to reach 90%+ accuracies, which was a problem we wanted to solve. Recalling that there were optimizers derived in recent years that reach

convergence faster, we decided to test Adam (adaptive momentum), one of the most widely used gradient based optimizers. We implemented Adam using the algorithm proposed by Kingma and Ba (2014) in their original paper. We used the same mini-batch selection method as the SGD optimizer, but updated the weights with the Adam optimizer algorithm. We tested the models using the hyperparameters proposed in the aforementioned paper, setting $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = .999$. All other variables were kept consistent across the two optimizers, such as batch size and activation function.

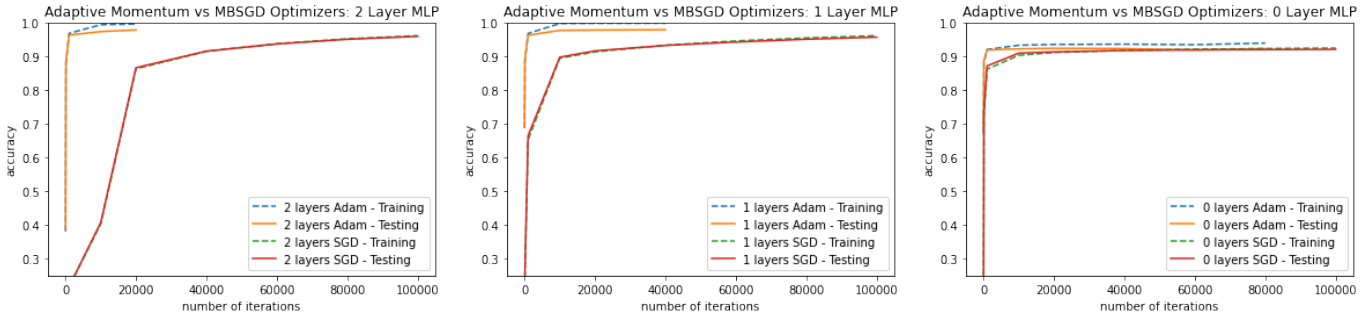

Figure 4: Adaptive momentum vs MBSGD Optimizers for MLP with 2, 1, and 0 hidden layers.

Observe that in the 1 and 2 layer MLP with Adam optimizer, the Adam train and test accuracy line ends early since convergence was reached long before the max iterations variable that we set. In each of the n-layer MLPs, we noted that we reach significantly higher accuracies with a lower number of iterations and consequently reach convergence much faster than the MBSGD optimizer.

Part 4: Impact of optimizer learning rate on the accuracy performance

In this experiment, we constructed two layer models with ReLU activation function and 128 units in each hidden layer. We investigated how the change of learning rate of our MBSGD could have an impact on the testing data accuracy. As shown in Figure 5, as we increase the learning rate from 0.0001 to 0.01, the convergence speed and the accuracy increase accordingly. However, a much higher learning rate could cause overshooting during the gradient descent process. Thus, it was appropriate that we used a learning rate of 0.01 for our model to train the MNIST dataset.

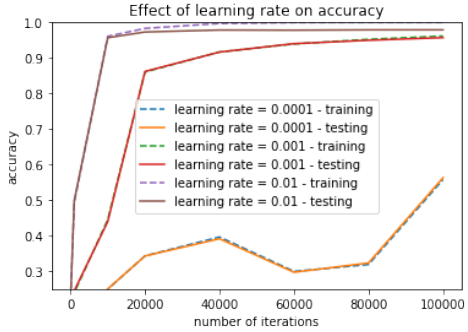

Figure 5: Effect of learning rate on accuracy.

Discussion and Conclusion

In the results of our experiments, we found that the depth of the MLP had a noticeable impact on performance, at least when comparing no hidden layers to 1- and 2-layer MLP. Our 0-layer MLP converged at with about 92% accuracy, whereas both 1 and 2-layer MLPs converged at approximately 98%. Interestingly enough, 1-layer MLP model had around the same accuracy as the 2-layer MLP model and trained slightly faster. Activation function choice had a large impact on the model performance, as we observed that at least for the MNIST dataset, tanh converged to a lower accuracy than ReLU and logistic, and logistic took far longer than ReLU to converge. We

conclude that ReLU performs the best for the MNIST classification task overall. Another interesting observation was that as we increased the L2 regularization strength, we achieved lower testing and training accuracy. The reason may be that adding regularization could cause overshooting during the gradient descent process for the MLP model. Surprisingly, we found that SGD converged on unnormalized MNIST data faster than the normalized set, but again this may be attributed to the already standardized scale of the images. On the other hand, rather predictably, we found that the performance of the model increased as the width of the MLP increased, until performance topped out at around 98% accuracy. MLPs with more than 128 units in hidden layers performed at the same 98% accuracy bound, however took longer to train, as expected.

As we demonstrated in Part 3 of the creativity section, the vanilla mini-batch SGD optimizer is relatively slow in converging when compared to adaptive gradient descent methods and we showed that using an adaptive moment estimation optimizer converged much faster than standard SGD. This can be best visualized in the 2-layer MLP comparison over iterations (refer to Figure 4), where Adam converges before SGD even begins to plateau. However, even Adam has its limitations. Literature argues that Adam does not generalize very well as opposed to SGD [4], and there are even more suggested improvements to the "standard" Adam algorithm, such as Accelerated Adam (AAdam) proposed by Tato and Nkambou (2018) at UQAM, that generalize better and outperform Adam. Future experiments could compare the variants of Adam and compare optimizers on performance and generalization (using SGD and Adam as control). Another experiment we contemplated was the effect of exponential learning rate decay, however in Theorem 4.1 of Kingma and Ba's original Adam paper, the authors hypothesize that the learning rate is already guaranteed to have a square root decay: $\alpha_t = \frac{\alpha}{\sqrt{t}}$. Future experiments could combine Adam or other adaptive optimizer that guarantees learning rate decay and combine it with an exponential time-based learning rate decay and compare training performance. Currently there are no "best practice" guidelines for exponential learning rate decay, so we expect this experiment to be practical and valuable to the deep learning community.

Statement of Contributions

Task 1: Ziqi
Task 2: Alvin, Ziqi, Emily
Task 3 and Creativity: Alvin, Ziqi, Emily
Report: Alvin, Ziqi, Emily

References

[1] Alejandro B. (2019). A Survey of Handwritten Character Recognition with MNIST and EMNIST https://www.researchgate.net/publication/334957576_A_Survey_of_Handwritten_Character_Recognition_with_MNIST_and_EMNIST

[2] CS231n Convolutional Neural Networks for Visual Recognition https://cs231n.github.io/neural-networks-2/datapre

[3] Kingma, D.P., Ba, J. (2015). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.

[4] Zhou, Pan Feng, Jiashi Ma, Chao Xiong, Caiming HOI, Steven Ee, Weinan. (2020). Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning.