Introduction

This assignment concerns the exploration of various types of machine learning models. More specifically, the main models are linear and logistic regression, which are centered on the mean squared error (MSE) and binary cross entropy loss (BCE) respectively. This project also takes into consideration of the effects and differences between batch gradient descent (BGD) and stochastic gradient descent (SGD).

1. Linear Regression

**Loss Function and Gradient**

The linear regression model optimizes based on the mean squared loss function (MSE). In general, MSE takes the squared difference between the correct label and the predicted value of the model to calculate the fit of the model on the dataset that is given to it. The predictive model of the linear regression is given by the equation , where is the weight vector transposed, is the data matrix, and is the bias of the model. The resulting prediction should be a vector that has the length (number of rows) of the dataset that we are using. This would then be used in the general MSE equation, which is displayed below:

For this assignment, we also assigned a

Figure 1: Python Implementation for MSE Loss

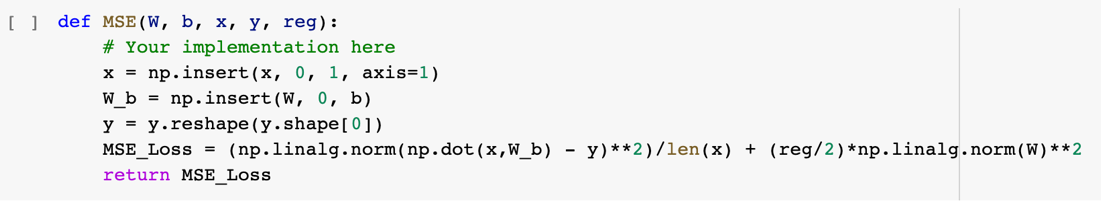
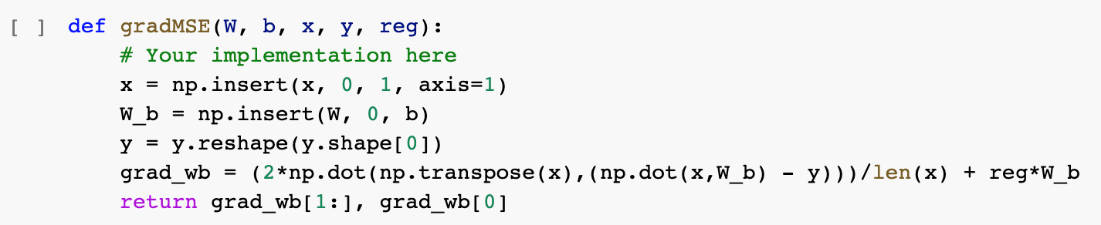


Figure 2: Python Implementation for MSE Gradient



Analytical Expression for MSE Gradient:

**Gradient Descent Implementation**

Figure 3: Python Implementation of MSE Gradient Descent

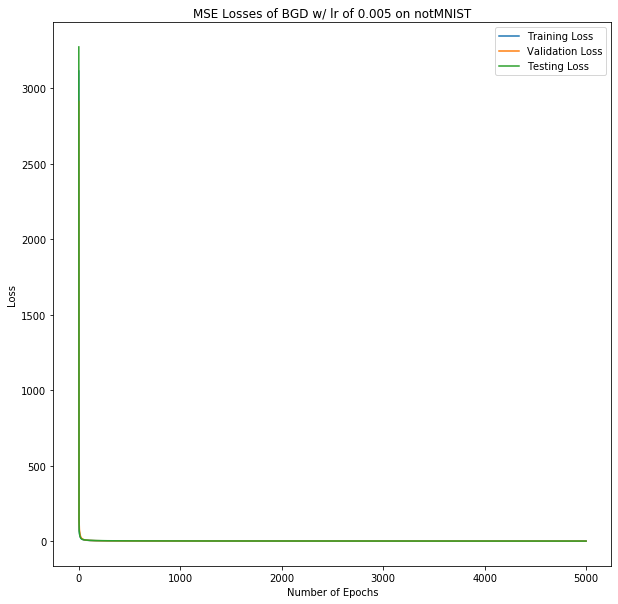
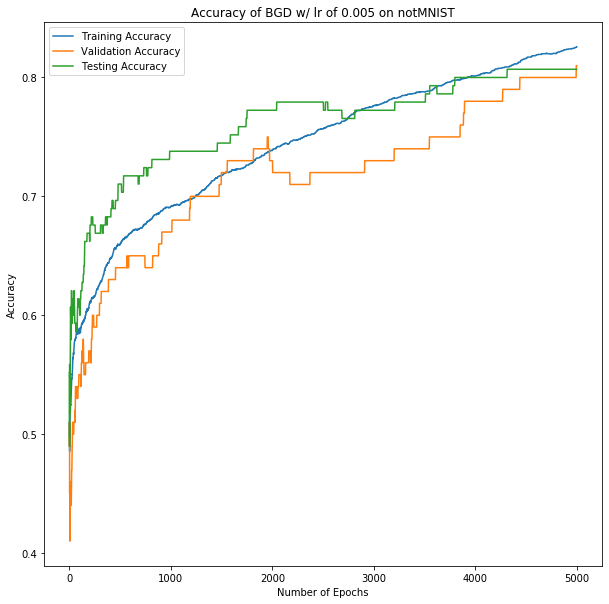
A screenshot of a social media post

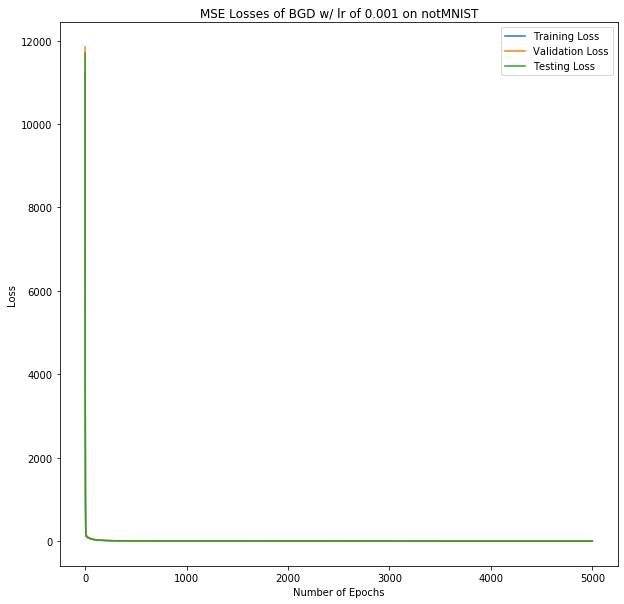
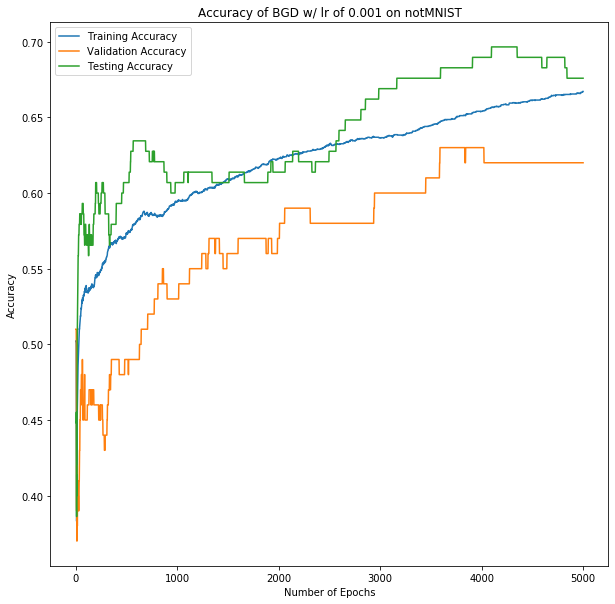
Description automatically generated

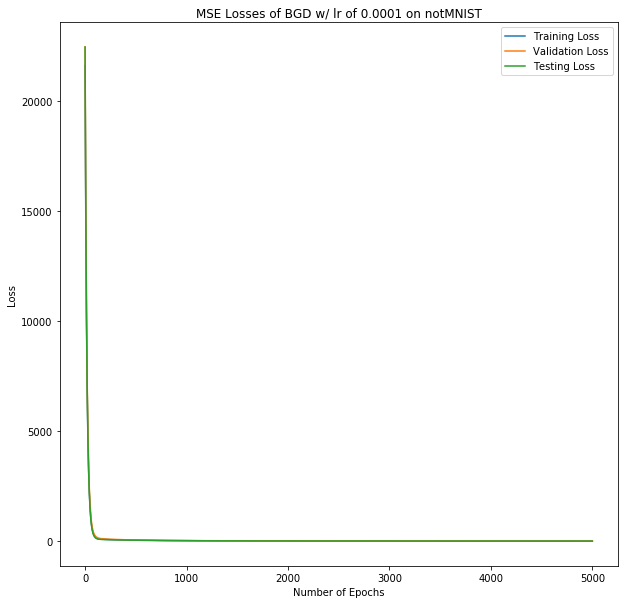
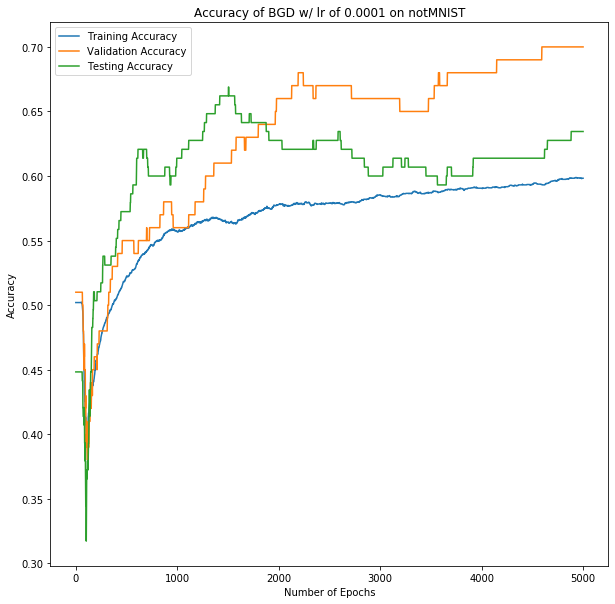
A screenshot of a cell phone

Description automatically generated

**Tuning the Learning Rate**

Figure 4: Training, Validation, and Testing Loss Accuracy, alpha = 0.005

Figure 5: Training, Validation and Testing Loss and Accuracy, alpha = 0.001

Figure 6: Training, Validation and Testing Loss and Accuracy, alpha = 0.0001

As can be seen from Figures 4 through 6, as the training rate alpha decreases, both initial and final training, validation, and testing losses increase, while corresponding accuracies increase. This makes sense as it would intuitively take longer to converge at a lower learning rate. As well, we see worse accuracies for the 0.001 and 0.0001 cases than the 0.005 case. This is likely due to many of the same changes in convergence properties.

**Generalization**

|  |  |  |  |
| --- | --- | --- | --- |
| Regularization | Training Accuracy | Validation Accuracy | Testing Accuracy |
| 0.001 | 0.96 | 0.79 | 0.84 |
| 0.1 | 0.97 | 0.97 | 0.97 |
| 0.5 | 0.96 | 0.97 | 0.96 |

Table 1: Accuracies for BGD MSE training on notMIST dataset for given regularization constants.

**Comparing Batch GD with Normal Equation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Regularization | Training Loss | Training Accuracy | Validation Loss | Validation Accuracy | Testing Loss | Testing Accuracy |
| 0 | 0.0187 | 0.99 | 0.0476 | 0.96 | 0.056 | 0.94 |
| 0.001 | 0.0248 | 0.99 | 0.0536 | 0.96 | 0.063 | 0.94 |
| 0.1 | 0.625 | 0.99 | 0.654 | 0.96 | 0.663 | 0.94 |
| 0.5 | 3.088 | 0.99 | 3.079 | 0.96 | 3.088 | 0.94 |

Table 2: Losses and Accuracies for Normal Equation (Least-Squares) on notMIST dataset for given regularization constants.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Regularization | Training Loss | Training Accuracy | Validation Loss | Validation Accuracy | Testing Loss | Testing Accuracy |
| 0 | 0.3 | 0.83 | 0.39 | 0.81 | 0.37 | 0.81 |
| 0.001 | 0.4 | 0.81 | 0.79 | 0.79 | 0.4 | 0.84 |
| 0.1 | 0.07 | 0.97 | 0.08 | 0.97 | 0.08 | 0.97 |
| 0.5 | 0.04 | 0.96 | 0.05 | 0.97 | 0.05 | 0.96 |

Table 3: Losses and Accuracies for BGD MSE training on notMIST dataset for given regularization constants.

As can clearly be seen upon comparing losses and accuracies between Batch Gradient Descent with MSE and the Normal Equation for any regularization constant, the normal equation performs better in all cases. However, asymptotically, BGD MSE likely runs faster. Where the least squares implementation is O(d3) simply for the matrix inversion, BGD MSE is O(dN) per update, operating much more quickly. (Nominal runtime as well)

1. Logistic Regression

**Loss Function and Gradient**

Figure 7: Cross Entropy Loss Python Implementation

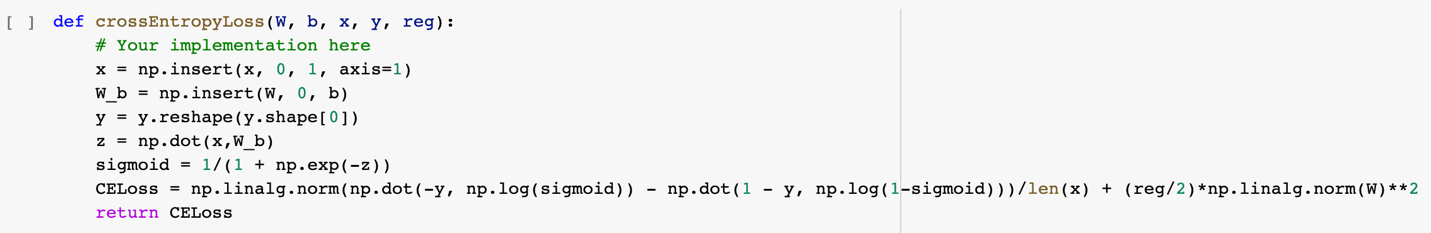


Figure 8: Cross Entropy Gradient Python Implementation

A screenshot of a cell phone

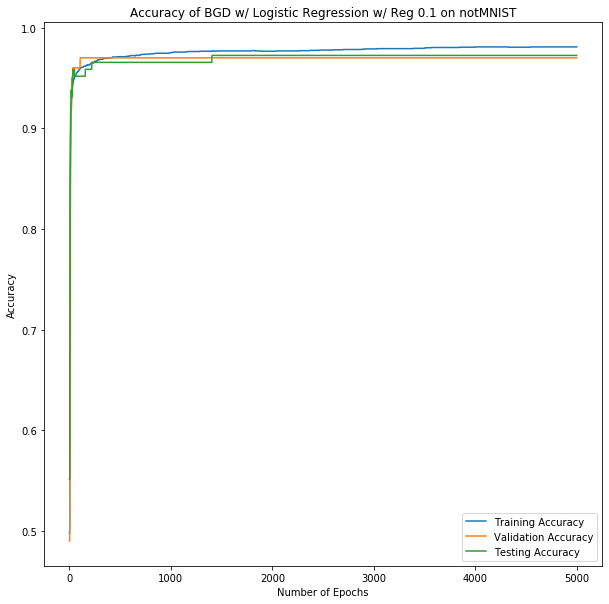
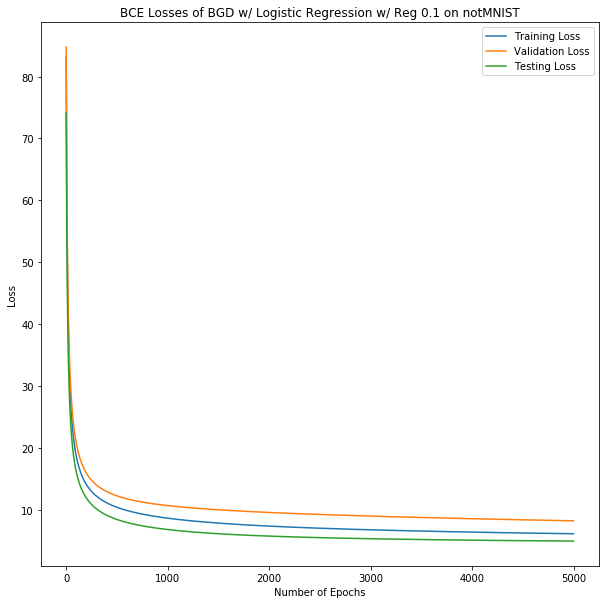
Description automatically generated

Analytical Expressions for Cross Entropy Loss and Gradient

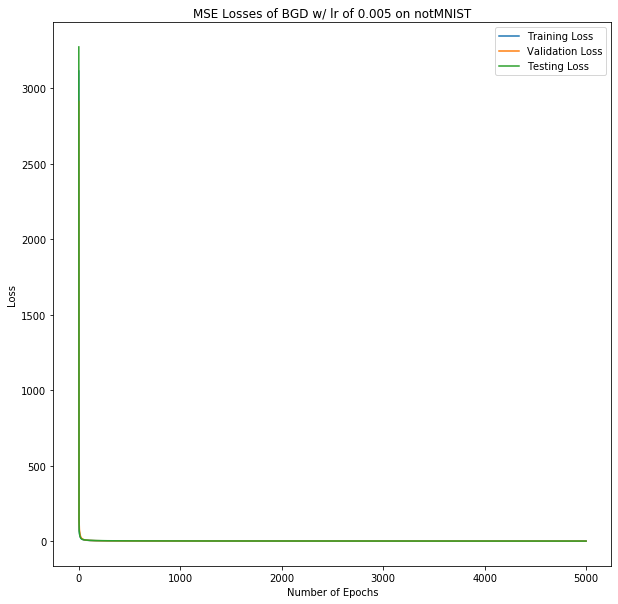
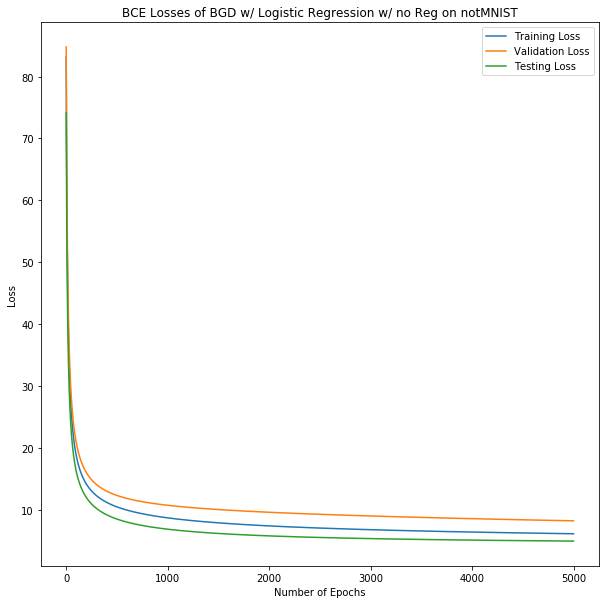
Loss:

Gradient:

**Learning**

****Figure 9: Cross-Entropy Losses Figure 10: Accuracy for Cross-Entropy

**Comparison to Linear Regression. Make sure l=0 for both graphs**

Figure 11: Cross-Entropy Losses Figure 12: MSE Losses

As can be seen by comparing Figures 11 and 12, Cross-Entropy Loss and MSE Loss converge at similar rates. However, the major difference can be seen with the more asymptotic behavior. The MSE case has Testing Loss the largest, but Cross-Entropy has Validation as the larger error. This may imply that Cross-Entropy has a less reliable loss function, but it actually boasts equivalent if not better validation accuracy of 0.98.

1. Batch Gradient Descent vs. SGD and Adam

**Building the Computational Graph**

Figure 13: Python Implementation of buildGraph()

A screenshot of a social media post

Description automatically generated

**Implementing Stochastic Gradient Descent**

Figure 14: MSE Loss and Accuracy for 0.001 Learning Rate and Batch Size of 500 on notMIST

A close up of a logo

Description automatically generatedA close up of a map

Description automatically generated

**Batch Size Investigation**

Figure 15: MSE Loss and Accuracy for 0.001 Learning Rate and Batch Size of 100 on notMIST

A screenshot of a cell phone

Description automatically generatedA close up of a map

Description automatically generated

Figure 16: MSE Loss and Accuracy for 0.001 Learning Rate and Batch Size of 700 on notMIST

A close up of a logo

Description automatically generatedA close up of a map

Description automatically generated

Figure 17: MSE Loss and Accuracy for 0.001 Learning Rate and Batch Size of 1750 on notMIST

A close up of a logo

Description automatically generatedA close up of text on a white background

Description automatically generated

There is a clear correlation between larger batch sizes and lower final validation accuracies as can be seen from Figures 15 through 17. We expect that this is due to the effect of the batch size on the number of batches per epoch. At a batch size of 100, there are 35 batches run per epoch versus only 2 batches for a batch size of 1750. This increase in number of trials increases the rate at which accuracies are going to converge due to the increase in information gained per epoch.

**Hyperparameter Investigation**

Figure 18: MSE Accuracy for and Batch Size of 500 on notMIST

A close up of a map

Description automatically generatedA close up of a map

Description automatically generated

Figure 19: MSE Accuracy for and Batch Size of 500 on notMIST

A close up of a map

Description automatically generatedA close up of a map

Description automatically generated

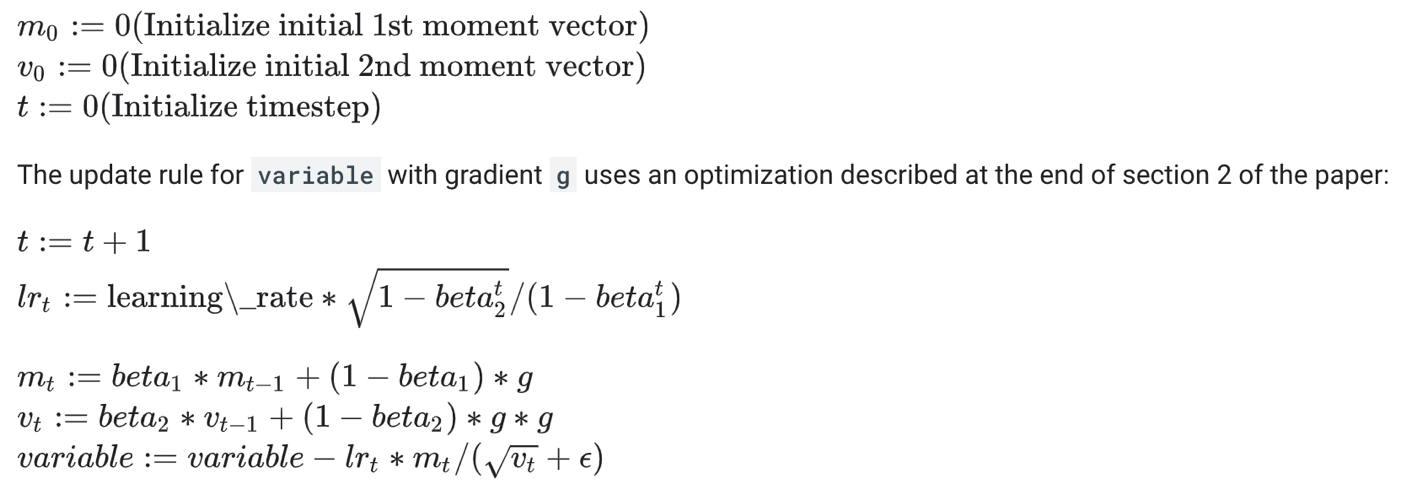
Figure 20: MSE Accuracy for and Batch Size of 500 on notMIST

A close up of a map

Description automatically generatedA close up of a map

Description automatically generated

Figure 21: Adam Initial Conditions and Update Rules [1]



As can be seen in Figure 18, increasing leads to an increase in the rate of convergence of the accuracies, but also similar overall accuracies. This is expected as Adam uses a timestep dependent learning rate, , shown in Figure 21. This time variant learning rate decreases over time, and decreases at a slower rate for higher at a rate proportional to .

We see different effects upon changing . As it increases, we see slower convergence of the accuracies, and also lower validation accuracy. This is also expected as in the Adam update rule. For larger , we would expect a quicker decrease in the learning rate. This is also consistent with early experiments where lower learning rates led to decreases in overall accuracy.

We see that accuracy improved with the increase in , and no noticeable change on the rate of convergence. This outcome is interesting due to the fact that should have the largest impact at early timesteps where is small. As the change in “variable” is proportional to , which in early timesteps is approximately equal to . This value is larger for smaller and coincides with larger changes earlier on. At large timesteps, , for reasonably small , explaining the overall minimal changes to the plots.

**Cross Entropy Loss Investigation**

3.1.2 for Cross Entropy Loss:

Figure 22: BCE Loss and Accuracy for 0.001 Learning Rate and Batch Size of 500 on notMIST

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generated

3.1.3 for Cross Entropy Loss:

Figure 23: BCE Loss and Accuracy for 0.001 Learning Rate and Batch Size of 100 on notMIST

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Figure 24: BCE Loss and Accuracy for 0.001 Learning Rate and Batch Size of 700 on notMIST

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Figure 25: BCE Loss and Accuracy for 0.001 Learning Rate and Batch Size of 1750 on notMIST

A screenshot of a cell phone

Description automatically generatedA close up of a map

Description automatically generated

As can be seen by comparing Figures 22 through 25, increasing batch size reduces the rate of convergence of the accuracy and loss, but minimally affects their final values. The similar final values are not unexpected due to effectiveness of cross-entropy loss minimization pushing accuracy close to 1 for all cases. The decrease in the convergence rate is expected for the same reasons as the MSE loss case.

3.1.4 for Cross Entropy Loss:

Figure 26: MSE Accuracy for and Batch Size of 500 on notMIST

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Figure 27: BCE Accuracy for and Batch Size of 500 on notMIST

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Figure 28: BCE Accuracy for and Batch Size of 500 on notMIST

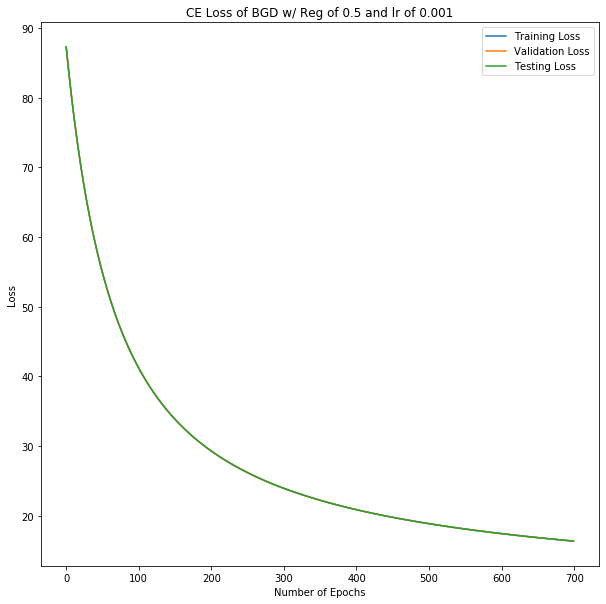
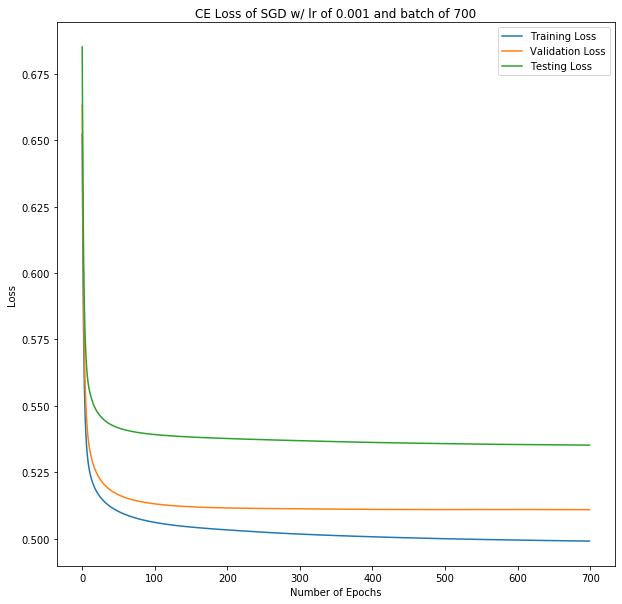
A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

We see the same behavior when varying hyperparameters as was discussed for the MSE loss trials, but with less notable effects overall, with a few minor differences. With the increase in , no decrease in accuracies was seen, likely due to the benefits of BCE. As well, the expected early effects of smaller was more pronounced in this case, where case had a larger initial change than case . As well, BCE was a clearly better performing model than MSE, with consistently higher accuracies for all cases.

**Comparison vs. Batch Gradient Descent**

The graphs of losses for BGD and SGD are displayed below.

At surface value, there is significant difference between the final losses of the training, validation, or testing set for both methods. BGD delivered relatively higher final losses at 16.35 for all three datasets, and SGD gave losses of 0.499, 0.511, and 0.535 for datasets training, validation, and testing respectively. However, this difference has little effect on the accuracy differences between the two models.

The main behavior is the convergence rate of the models. It appears that the SGD converges significantly faster than BGD does. BGD converges slowly and begins to even around the 200 epoch mark, whereas the SGD model starts to plateau around 50 epochs. This efficiency seen in SGD for several reasons. The first one is because the weight updates are based on smaller batches of data, which contributes to the short computation time. Secondly, the multiple weight and bias updates per batch means the updates occur multiple times in SGD vs BGD, which can help the model reach the minima faster. Second, the smaller dataset means that the gradient values are not as generalized as they would be in BGD, which leads to less noise/ambiguity when it comes to the direction of the weight and bias changes, which makes it more likely for it to follow a path that converges to the minima. The convergance comparison can also be seen with the accuracy graph below. BGD reached 96%, 97%, and 95% for training, validation, and testing datasets respectively, and SGD reached 98% for all three datasets.

