Diandra Prioleau
27 November 2018
EEL 5840 – Fundamentals of Machine Learning
HW06

In Figure 1, the multi-layer perceptron (MLP) achieved a high accuracy of 100% and resulted in a better performance, shown by the confusion matrices, using error backpropagation and the following parameters: 5 nodes in the hidden layer; a learning rate of 0.01; and 10,000 iterations. Error backpropagation was used to update the weights of the hidden layer and the output layer. As shown in the confusion matrices in Figure 1, before updating the weights, the MLP had a high number of false negatives and an accuracy of 50%; however, after updating the weights using error backpropagation with the values set for the parameters, stated above, the accuracy increased to 100%. The highest accuracy was achieved by using error backpropagation, discussed previously, ensuring the learning rate was small enough to avoid oscillating across the local optima to smoothly converge, and setting the number of iterations to a large enough value to make sure that the weights have enough updates to converge to a local optimum. In addition, there needed to be enough nodes in the hidden layer so that the MLP has enough information to accurately classify each input. If there is not a sufficient number of nodes, decision boundaries necessary to classify each instance will not be included and will negatively affect performance and accuracy.

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

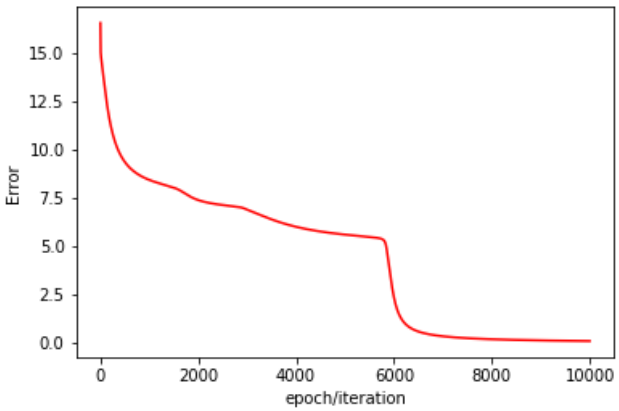

Figure 1 Error per Iteration

In Figures 2-5, all the parameters are the same as those for the MLP in Figure 1 except for the learning rate. In Figure 2, the learning rate is 0.5. In Figure 3, the learning rate is 0.1. In Figure 4, the learning rate is 0.001. In Figure 5, the learning rate is 0.0001. As the learning rate decreases, shown below, the accuracy of the MLP increases. This is because the learning rate controls the step size when updating weights. Therefore, as the learning rate decreases, the step size decreases causing the weights to change slower. Since the number of iterations is fixed at 10,000, when the learning rate is 0.5, shown in Figure 2, the error increases. Since the learning rate is large, the weights are oscillating between local optima causing the error, show in Figure 2, to not converge smoothly to a local optimum. However, as the learning rate is decreased, the weights are slowly moving towards the weights that decreases error, causes error begins to converge smoothly to a local optimum, and increase performance and accuracy, shown in Figures 1 and 3. However, if the learning rate is too small, it can still affect performance. In Figures 4 and 5, the learning rate is 0.001 and 0.0001, respectively. As can be seen in Figure 4, the MLP converges to a local optimum slower than in Figure 1. In Figure 5, the MLP converges to a local optimum much slower than both the MLPs in Figures 1 and 4 and causes a decrease in performance and accuracy. This occurs because the learning rate controls the step size when updating the weights. Since the learning rate is really small, it causes the MLP to take much longer, in terms of the number of iterations, to converge to a local optimum in which error is as small as possible.

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[40. 20.]
 [20. 40.]]
Percentage Correct:  66.66666666666666
```

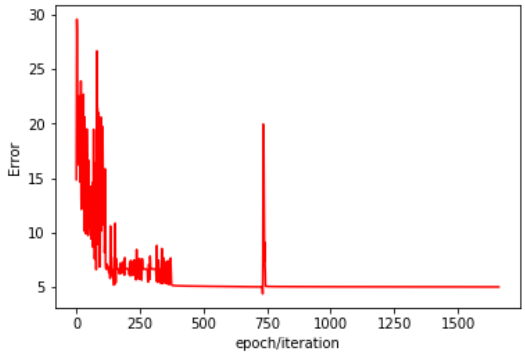

Figure 2 Error when Learning Rate is 0.5

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[40.  0.]
 [20. 60.]]
Percentage Correct:  83.33333333333334
```

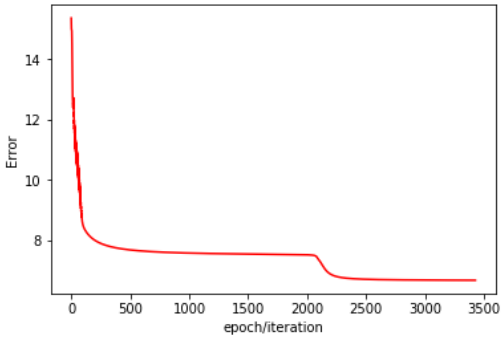

Figure 3 Error when Learning Rate is 0.1

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

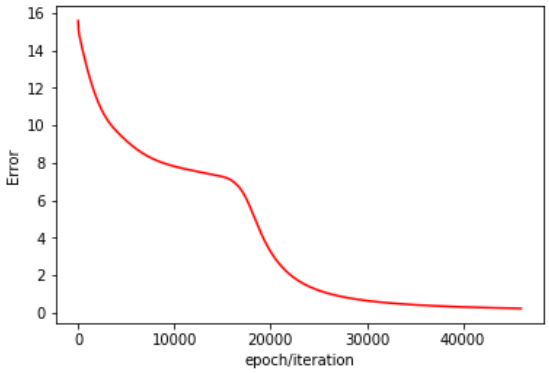

Figure 4 Error when Learning Rate is 0.001

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[40.  0.]
 [20. 60.]]
Percentage Correct:  83.33333333333334
```

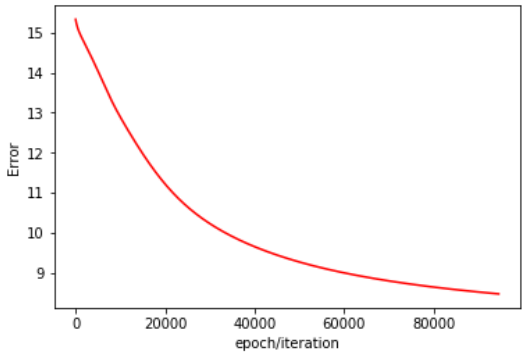

Figure 5 Error when Learning Rate is 0.0001

In Figures 6-8, all the parameters are the same as those for the MLP in Figure 1 except for the number of iterations. In Figure 6, the number of iterations is 100. In Figure 7, the number of iterations is 1000. As the number of iterations increases, shown below, the accuracy of the MLP increases and the number of false negative and false positives decreases. This is because the number of iterations controls the number of times the weights are updated. If the number of iterations is too small and the MLP has not begun to converge to a local optimum, the performance of MLP will not be as good. Therefore, as the learning rate decreases, the step size decreases causing the weights to change slower. Since the learning rate is fixed at 0.01 when the number of iterations is 100, shown in Figure 4, the error has not begun to converge to a local optimum. However, as the number of iterations is increased, more time is given for the weights to be updated such that the error converges to a local optimum and to increase performance and accuracy, shown in Figures 1 and 7. In Figure 8, the number of iterations is 100,000; however, the graph stops after close to 10,000 iterations. This occurs because in the mlptrain function, the loop is being controlled by the number of iteration and the change in error. Since the change in error reaches the threshold of 0.00001, the training stops. This helps to avoid iterations that are not necessary to improving the accuracy of the MLP and improves performance by limiting unnecessary computations for updating the weights of the MLP.

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[20. 20.]
 [40. 40.]]
Percentage Correct:  50.0
```

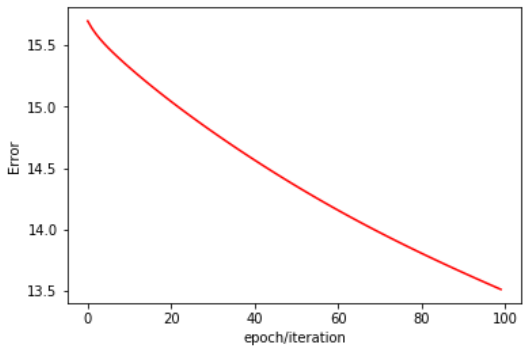

Figure 6 Error when # of Iterations = 100

```
Before Updating Weights
Confusion matrix is:
[[60. 60.]
 [ 0.  0.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[40.  0.]
 [20. 60.]]
Percentage Correct:  83.33333333333334
```

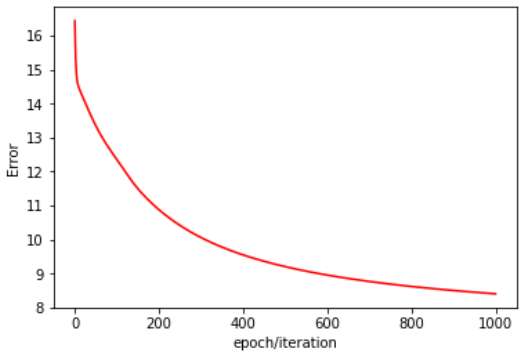

Figure 7 Error when # of Iterations = 1000

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

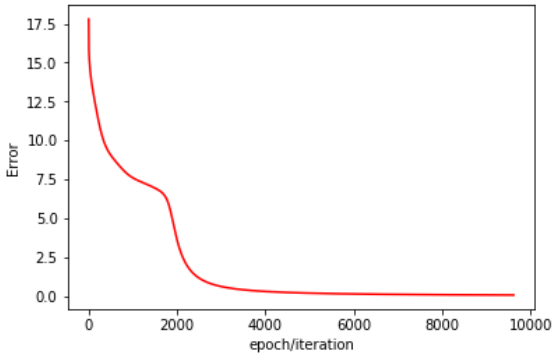

Figure 8 Error when Number of Iterations is 100,000

In Figures 9-12, all the parameters are the same as those for the MLP in Figure 1 except for the number of hidden nodes. In Figure 9, the number of hidden nodes is 1. In Figure 10, the number of hidden nodes is 3. In Figure 11, the number of hidden nodes is 7. In Figure 12, the number of hidden nodes is 9. As shown below, the accuracy of performance of the MLP does not change as the number of hidden nodes increases from 1 to 3. It is not until the number of hidden nodes is changed to 5, shown in Figure 1, that the highest accuracy is achieved. In addition, the confusion matrices indicate that the MLPs in Figures 9 and 10 are having some difficulties with predicting instances as negative when they are positive (false negative). While choosing 7 and 9 hidden nodes, shown in Figures 11 and 12, respectively, resulted in high accuracy, it has not clear whether using such large numbers of hidden nodes will improve performance since there is not a validation set to confirm. It may result in overfitting if using those large number of hidden nodes is causing the MLP to fit only to the training data and to not be able to accurately predict new data. Therefore, since higher performance and accuracy was achieved when choosing 5 as the number of hidden nodes, along with the selected values for the other parameters given in Figure 1, it was chosen.

```
Before Updating Weights
Confusion matrix is:
[[60. 60.]
 [ 0.  0.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

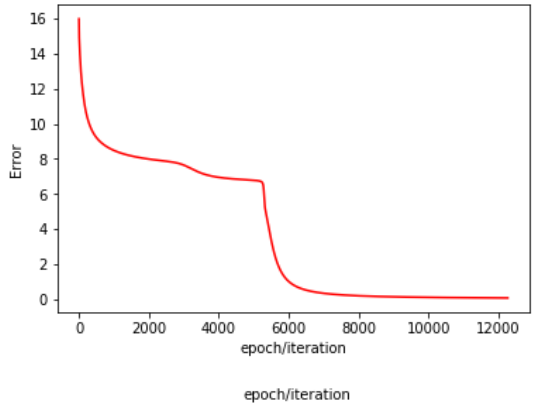

Figure 9 Number of Hidden Nodes is 1

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

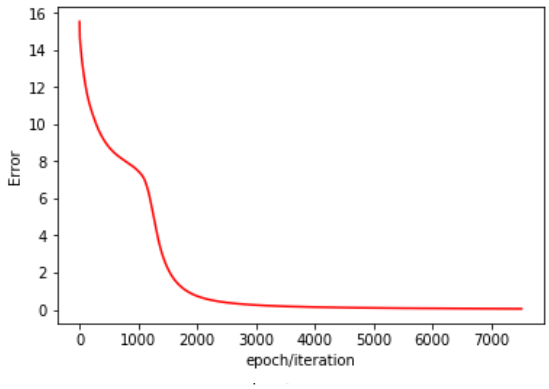

Figure 10 Number of Hidden Nodes is 3

```
Before Updating Weights
Confusion matrix is:
[[60. 60.]
 [ 0.  0.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

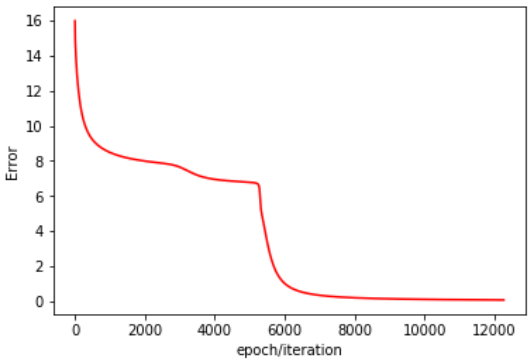

Figure 11 Error when Number of Hidden Nodes is 7

```
Before Updating Weights
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
After Updating Weights
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

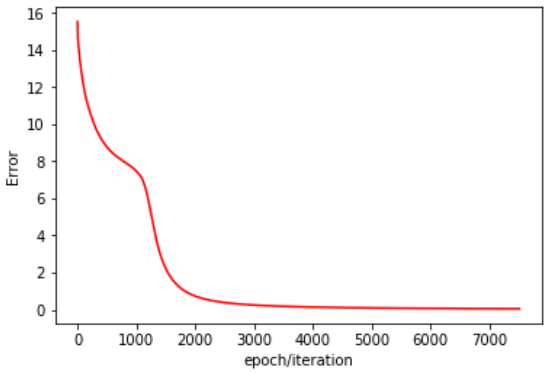

Figure 12 Error when Number of Hidden Nodes is 9