Computer Science
4P80 - Artificial Neural Networks

Feed Forward Neural Networks

Prepared by: Eduardo Saldana
Instructor: Dave Bockus
Date: March 10, 2023

Abstract

In this report the uses and parameters for Feed-Forward Neural Networks are tested. Also the architecture and explanation of Feed-Forward Neural Networks is explained to show how it works. The different parameters and training sets will be compared to analyze what suits best a Feed-Forward Neural Network given different circumstances. The power of a Feed-Forward Neural Network capacity to classify data will be shown though the results obtained.

Contents

1 Sections

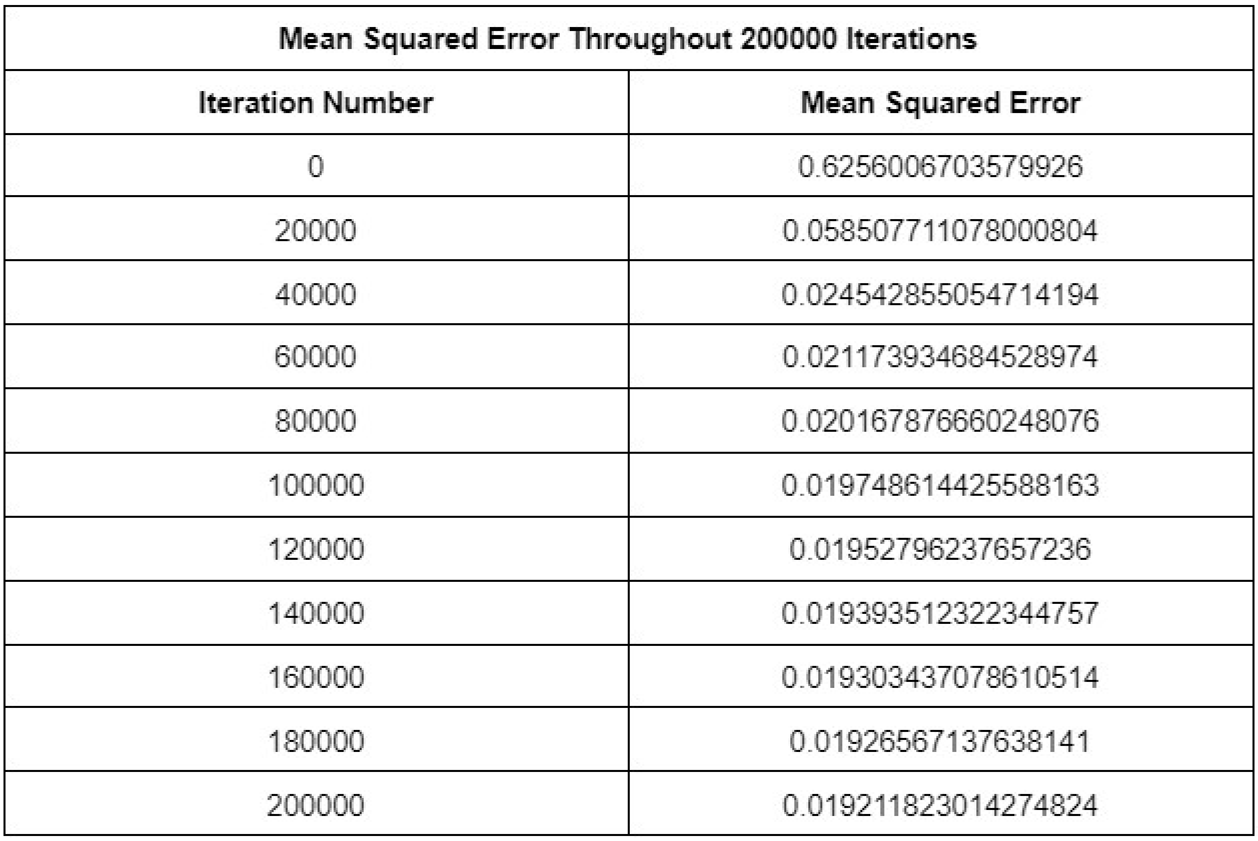
1.1 Introductions

A Feed Forward Neural-Network is a supervised learning algorithm used to classify its inputs. The way is works is by a connected series of layers which are matrices that are initially consisting of randomized values ranging anywhere from -1 to 1 depending on the developers choice. Then the Neural Network is ”trained” meaning it is given the training set that it will use to adjust its values, this input values are multiplied with the following matrix which in turn is then multiplied with the next matrix and so on. This is don’t until it reaches the final layer which only contains one node. The multiplication is done by the Sigmoid function in this case although there are multiple activation functions that work. The reason we can use Sigmoid (or any other activation function) is because the values to be used are first normalized, this meaning that all values to be used are scaled down proportionally to be between 0 and 1. Once it reaches the output node it compares the result that was obtained with what it should have been. This is when the back propagation takes place.
The back propagation works by taking the error result and getting its derivative, then simply multiplying these 2 values. The resulting values will then be added to the previous layer, so for example, in the first instance of back propagation where the output and second to last layer are considered, the layer to have its values modified would be the second to last layer. The values to added are multiplied by the ”alpha” value, this value can also be seen as the learning rate (in this case 0.1) and it is used to control how fast the Neural Network should learn.

1.2 Part A

The architecture for the Neural Network in this case was a 3 layer network, this layers were as following: Input layer, Hidden layer and Output layer. The input layer was initialized with random numbers with as a matrix with its first dimension matching the number data points per input in the file and the second dimension being the number of hidden nodes that we wish to use. The hidden layer which was also initialized with random values in a matrix with its first dimension being the number of hidden nodes and the second dimension being one since it just connects to the output node which is just a 2d array since it will only contain the result.

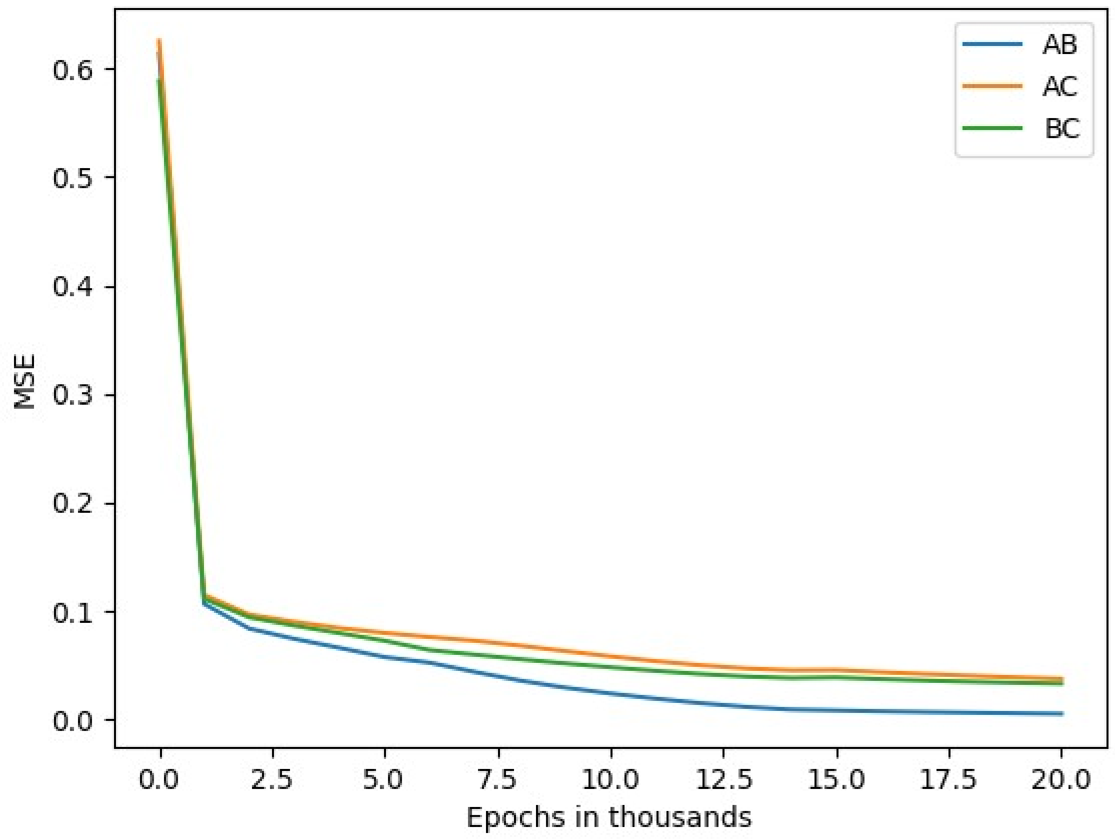
The tests that were run were done so using an alpha of 0.1 since its standard to use a small alpha to allow the Neural Network to find a local minimum. A local minimum is simply a state in which the Neural Network seems to have converged and it reached a point where it seemingly can not get much more accurate. This is not necessarily the global minimum which would be the most accurate possible point the Neural Network can reach.
The alpha was kept static in this case to ensure the Neural Network can find any local minimum even if it meant to finding the global minimum. Here is a table listing the "Mean Squared Error" of the Neural Network when ran with 200000 epochs (iterations).



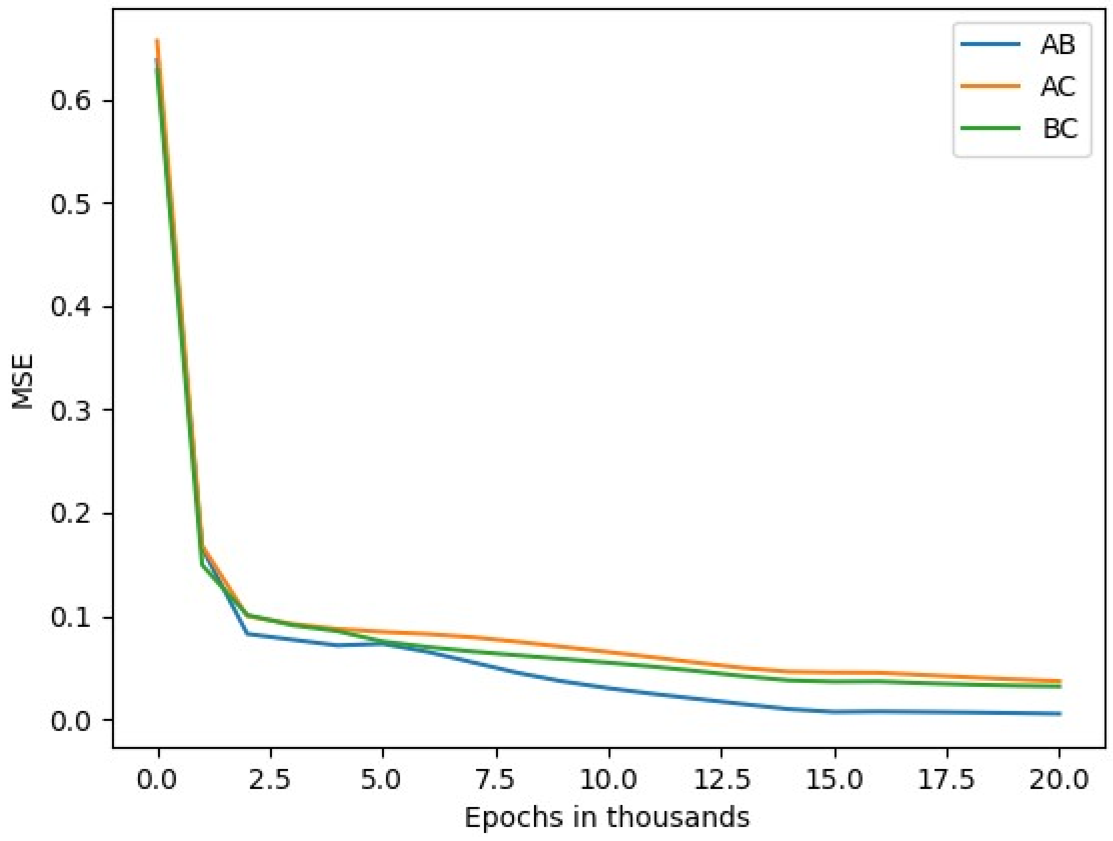
Mean Squared Error Throughout 200000 Iterations	
Iteration Number	**Mean Squared Error**
0	0.6256006703579926
20000	0.058507711078000804
40000	0.024542855054714194
60000	0.021173934684528974
80000	0.020167876660248076
100000	0.019748614425588163
120000	0.01952796237657236
140000	0.019393512322344757
160000	0.019303437078610514
180000	0.01926567137638141
200000	0.019211823014274824

1.3 Part B

For testing purposes in this sections the input data was split into three groups: A, B, C. Each one having a third of the data and roughly the same amount of good and bad motor data in each. Three Neural Networks were created with the following data in each respectively: A+B, B+C, A+C. The purpose of this experiment was to see if the convergence in each of these Neural Networks differ from the original one from earlier where all data was used. The second purpose of this is to see if the Neural Networks are able to properly classify data despite being trained on data that did not include the data to be tested.
When the training sets are split they all end up having similar convergence timings however they all reached different local minimums, noticeably the training set of A+B reached a better local minima than the training sets of B+C or A+C which ended up reaching very similar local minimums. Here are the graphs of the Mean Squared Error of the following training sets using 10 hidden nodes.



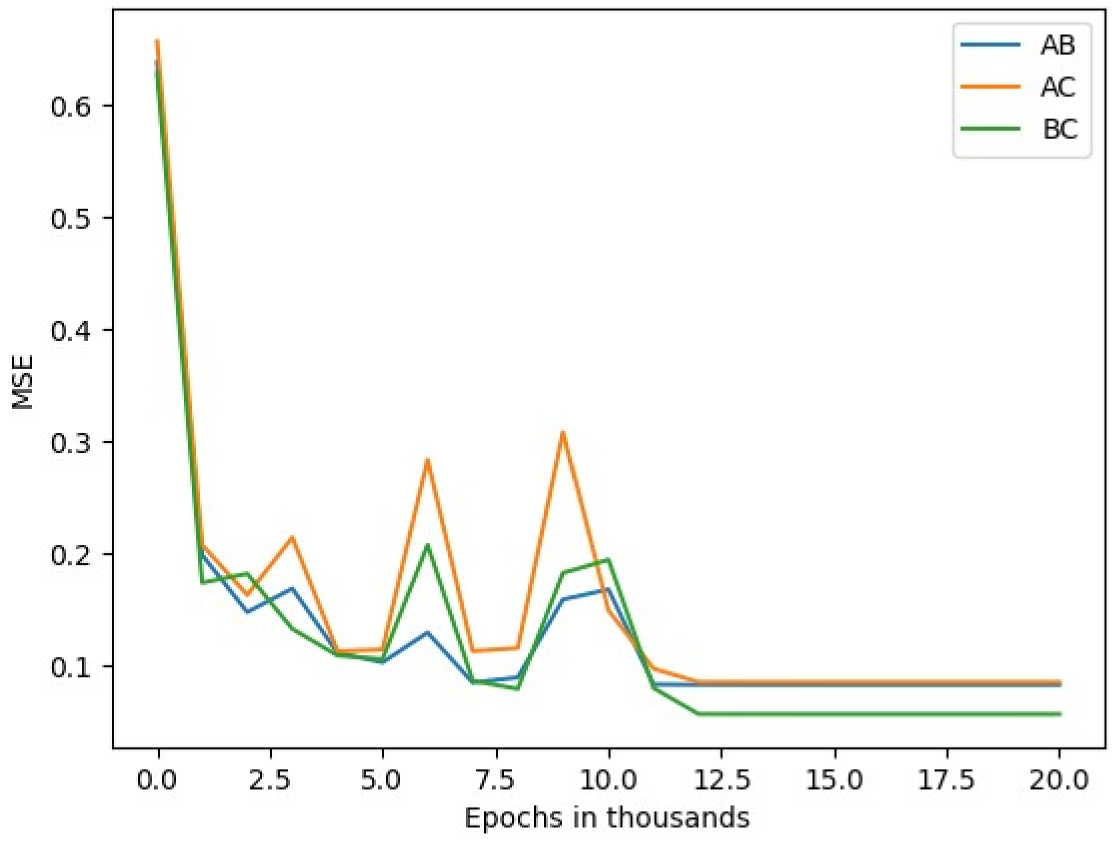
When the same test runs were done with a higher number of hidden nodes, in this case thirty there some noticeable changes. The first change that is noticed is how the code take longer to run since now the matrix multiplication deals with bigger dimensions. Secondly and most importantly the MSE learning curve is noticeably more erratic and not nearly as smooth as when there was a smaller number of hidden nodes. This could be attributed to the Neural Network trying to over fit and being quickly corrected as the epochs go on. As for the final results, they ended up having an average of very similar MSE after the 200000 epochs. Here is the graph of the MSE when using 30 nodes in the hidden layer.



However despite having slightly different MSEs when using a file such as the one with 16 data pieces per input all three Neural Networks were able to correctly classify all the inputs.

1.4 Part C

This section deals with the effects of implementing momentum on the Neural Network and it's ability to learn. The concept of momentum is meant to let the Neural Network move around more in the hope of finding a better local minimum then it would otherwise find and learn at a faster rate. This is to say that it gives the Neural Network to jump around the gradient, it's important to note that this will not always result in a better MSE since it's possible to bounce around and end in a worse local minimum. When momentum was implemented we can see how the learning curve jumped around heavily during the first two thirds, this would represent the Neural Network finding other local minimums and either staying in them because they have a better MSE or leaving that local minimum because it was not good enough to stay in it despite the newly added momentum. In the following graph is more apparent what is meant when talking about erratic jumps while learning.



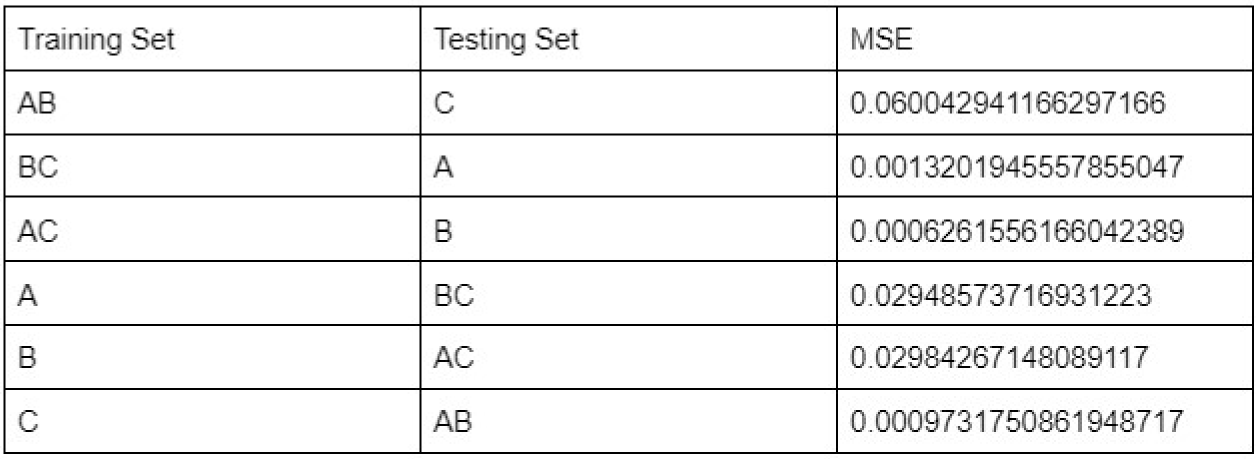
It is very important to note that the reason why momentum is not guaranteed to speed up the learning process or find a better local minimum is because the search itself for the local minimum can be very time consuming and if the criteria to stop the training in a Neural Network is a set number of Epochs there is always a possibility that the training might stop while the Neural Network is still jumping around in the gradient resulting in a poor result.

1.5 Part D

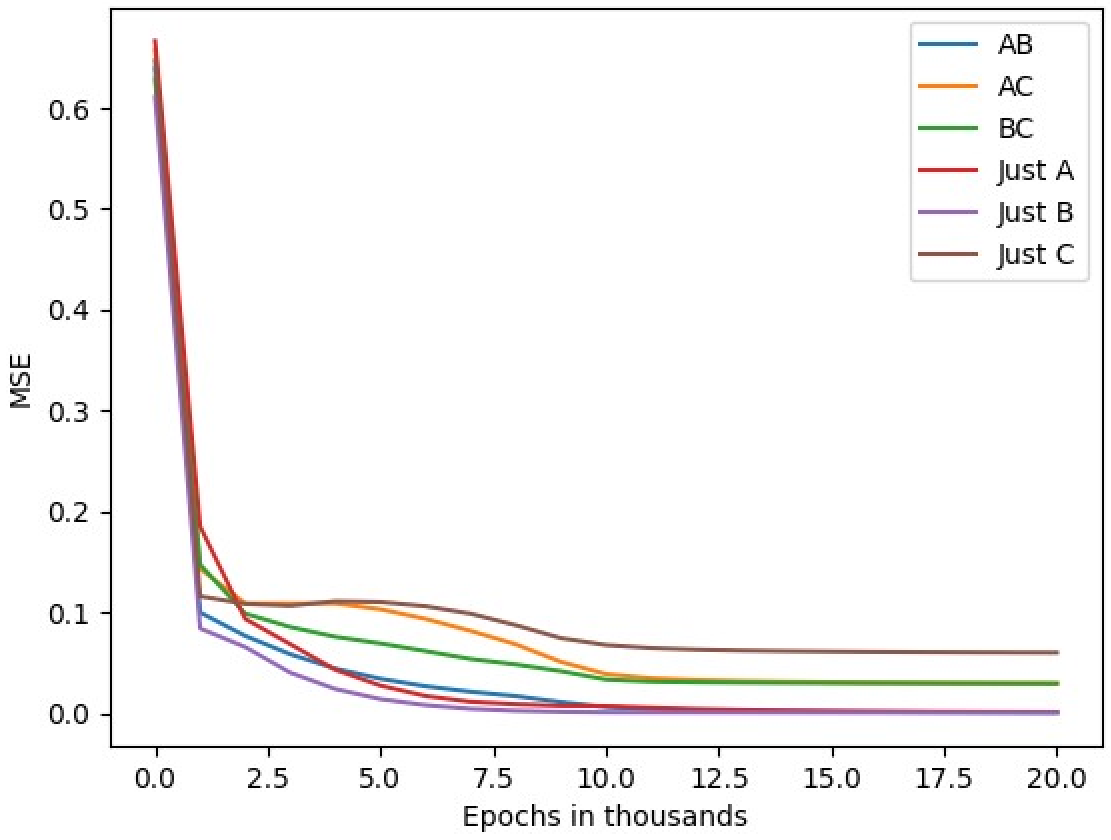
For part D we will be testing different combinations of training sets and testing sets to see how much the end result differs from each other and if the difference is even noticeable. For the test runs the following combinations were used:

- Training Set: AB, Testing Set: C
- Training Set: BC, Testing Set: A
- Training Set: AC, Testing Set: B
- Training Set: A, Testing Set: BC
- Training Set: B, Testing Set: AC
- Training Set: C, Testing Set: AB

After training each Neural Network for 200000 epochs with an alpha value of 0.1. the results showed that the combination of training sets and testing sets can have quite a large impact on the result MSE when the time to test the efficiency of the Neural Networks come. Here is table showcasing the MSE results for different combinations and a graph showing their learning curves.



Training Set	Testing Set	MSE
AB	C	0.060042941166297166
BC	A	0.0013201945557855047
AC	B	0.0006261556166042389
A	BC	0.02948573716931223
B	AC	0.02984267148089117
C	AB	0.0009731750861948717



The best example to show just how much the different combinations vary in results would be to compare the one where the training set was AB and testing set was C to the one where the training set was AC and the testing set was B. The latter one has an MSE with an order of magnitude almost ten times lower than the former one. This goes to show just how important it is to get a sufficiently large and diverse data sample size. Otherwise there will always be a risk of having the data training set be of poor quality whether it is because of data outliers or not a sufficiently diverse sample size which will in turn make the Neural Network biased towards wrong classifications.

1.6 Part E

Extending the number of hidden layers has the advantage of letting the Neural Network adapt better to complex data sets by letting it explore more possible patterns and learn from them. However it also comes with the risk of having the Neural Network learn wrong patterns in the data if the data used is not meant for deeply complex analysis.

2 Conclusion

Feed Forward Neural Networks when implemented correctly are a very potent tool train to classify data. However designing it with a proper architecture is essential. Most of the times this will involve a series of trial and error in which it is essential to try out different combinations and splits of training and testing sets as well as trying out different parameters. These parameters will involve everything from the alpha value, to the number of hidden nodes per layer as well as the number of hidden layers themselves.

Through experimentation this report showcases an example in which a Neural Network is created and trained to classify data and it is able to do so. Additional tests can be run to see the impact on the effectiveness such as trying out different activation functions or redesigning the Neural Network architecture.