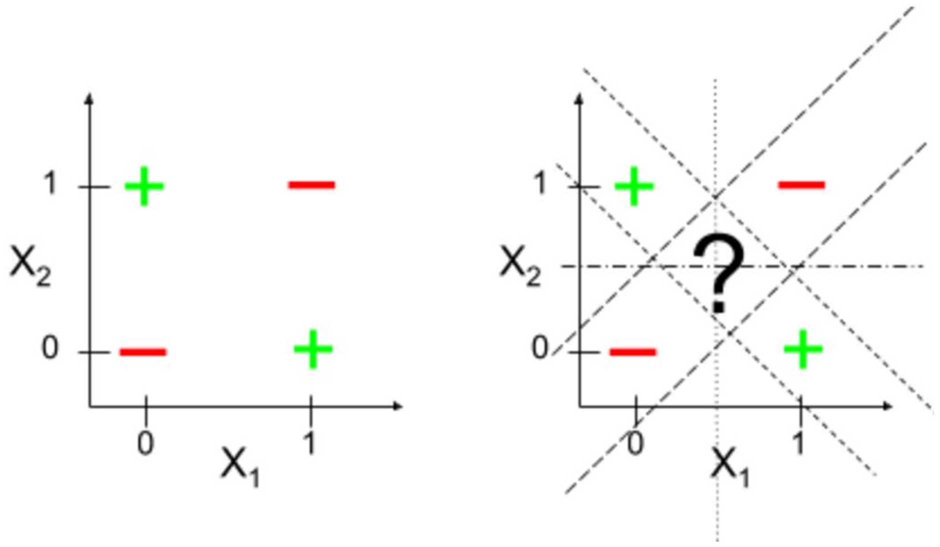
David Josephs
IA651 Applied Machine Learning
Final Project: Perceptron xor

After playing around with the model I came up with what i believe to be the best configuration for training the small neural network to solve the xor problem. The problem with xor is that the space which the perceptron is trying to solve is not linearly separable in a 2 dimensional space. The figure below shows the problem space in a two dimensional space.



I found that the best neural net work that solved the problem was one that had two inputs and one out put. As the xor operation has. In the middle i had a hidden layer that had four inputs/outputs. Playing around with the learning rate solved gave some interesting results. With a learning rate of 0.01 an average of 50,000 rounds was required to have the loss decrease to .05 which i used as the max amount of loss accepted. The neural network was able to correctly solve the problem up to around .20. To make sure that the neural network would properly solve the problem each time it was run I made sure the threshold was low enough that the output would be good. To validate the model I added some test cases to the program. These test cases validated the model that was produced by the neural network.

In future work the sample size of the training set would be increased. This would probably address the cases that I encountered where the neural network would not be able to decrease the loss around 0.55 to .39 it would suddenly be learning with a sedately decreasing amount less and less each time. This suggests that the gradient decent had found a local minimum and was stuck. Usually when running the neural network the it would have slightly less good learning rate when the loss was around 0.45.

This problem was the worse when the n_layer1 was not set to 4. Less or more tended to break the model about half the time. With n_layer1 set to 2 it succeeded about half the time. But setting it too high would not work either. When set to 10 the neural network would not work at all. 4 turned out to be the best value for the value of n_layer1. I also tried to use the random uniform distribution of the generation of the weights of W2 but that also made the model perform sub-optimally. I found that initialing the weights to zero for the second layer produced the best results for classifying the xor problem. While this was true for the second layer it is not true for the first layer. In the first layer not initializing to a random uniform distribution resulted in the model breaking. This may suggest that when the model performs sub-optimally that the initial weights for the first layer are not a good fit for the model. This is an interesting result that I did not expect when I started this project. Interestingly initializing the biases on the first layer to either zero or one did not matter for this neural network.

I believe that this neural network is able to solve the xor problem because it is able to change the problem so that it is not in a 2 dimensional space, but utilizes more than 2 dimensions. This way it is able to find a hyper-plane that is able to separate the two different problem spaces. Even with this insight I am still amazed at what a perceptron or neural net work is able to accomplish. It goes to show that even a small program such as this can be very powerful in accomplishing tasks. The future has so much to discover, and I am looking forward to what is possible with the advances in machine learning technology.