Jose Benitez-Santos

MLP Lab

Lab Report

Learning Points

To begin, there are a lot of things I would say I learned about doing backpropagation that I never did until implementing the code. The first being how you are able to assign blame to weights based by level. I was always very confused as to how it was that we would calculate the error or even get the values of the hidden network. Then I realized that once we randomize the weights and biases we will execute a forward function that iterates through the network in order to give us an appropriate 'y' value using the activation function. Along with that I always questioned how it was that we would be able to execute backpropagation with the Yj values that we would get from the second layer. I made the mistake of just taking the dot product of the weights from the input layer and multiplying them by the binary inputs. After further examination and help during office hours I was able to conclude that I actually needed to pass those values through the activation function as well. This alone solved a lot of issues for me. Another problem that I ran into was that I added unnecessary noise to my input values by giving all my bias values a random value between (-1 and 1). This caused my MLP to give me super weird convergence epoch numbers. And finally, the part that I truly struggled with the most was attempting to apply momentum to my network. This part for me was the most confusing and frustrating to grasp my thoughts around. I would say that among all the mistakes I made, the biggest was adding the momentum of all the previous iterations instead of just keeping the momentum from the previous iteration.

Lab Results Discussion

This lab as a whole taught me a lot on various values that can influence the convergence of a network. This being for better or for worse. When running my network without momentum in general, the convergence epochs were significantly larger than those when momentum was applied. In particular the learning rate .005. This had the most drastic change out of all the other values. Other than that, the values started to get relatively closer when the learning rate increased. What this tells me is that this network can use a larger learning rate to accomplish convergence. Yet, applying momentum appears to have a better epoch convergence ratio. This could be very important when working with larger datasets. Along with this, I noticed something that I found very interesting and I decided to leave it in inorder to talk about it. When I applied momentum, for the learning rate .01 I got a horrible epoch value of 673. I was very confused at first as this was the only value that was way off. What I came to realize is that having randomized weights could have dramatically affected the convergence of this rate. In a different setting, having weights that perhaps were all positive or negative, or even smaller could have helped avoid such an outlier.