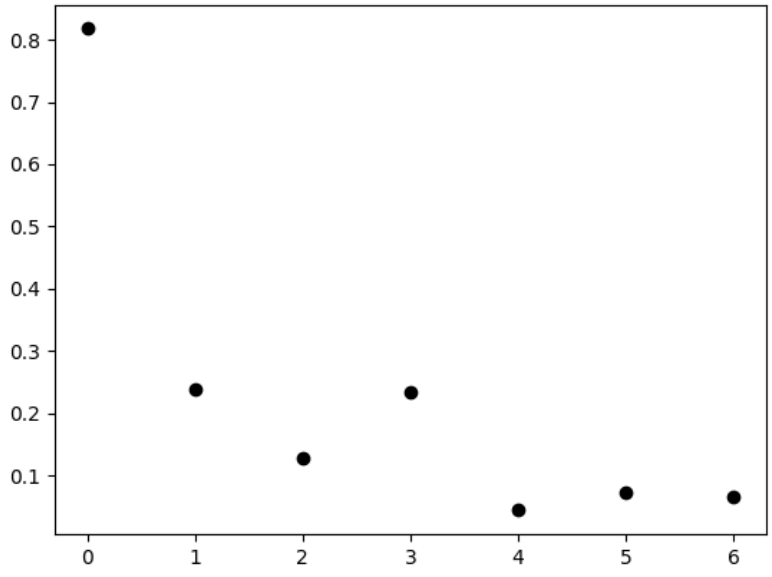
Homework 2

Marshall Grimmett

October 28, 2020

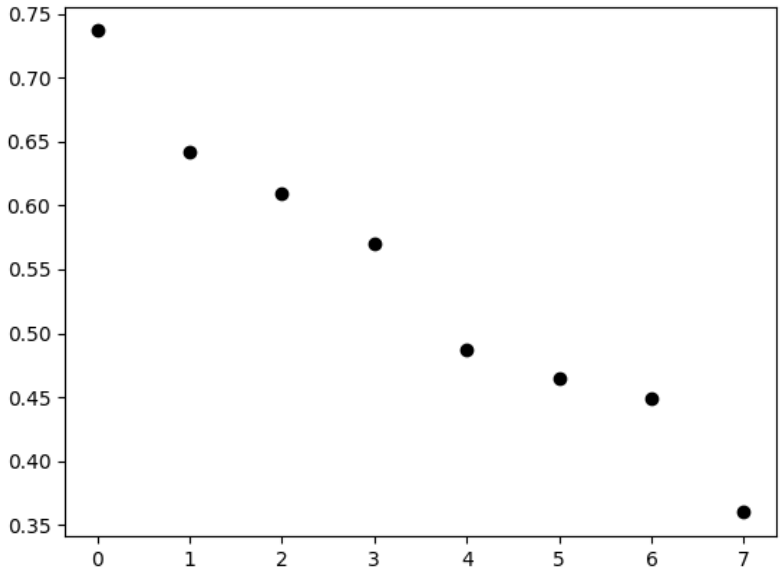
1 Part 1

I started with the initial values of psi and epsilon, but I was only achieving 1 epoch, so I slowly decreased my epsilon until the number of epochs was around 5. This way I could find a breaking point for psi and epsilon. This is because if the psi value is too small then the difference between epochs will be small as well. I landed on an epsilon value of 0.5 with the initial psi value of 0.1. The results are below. The y-axis is the cross entropy error and the x-axis is the number of epochs.



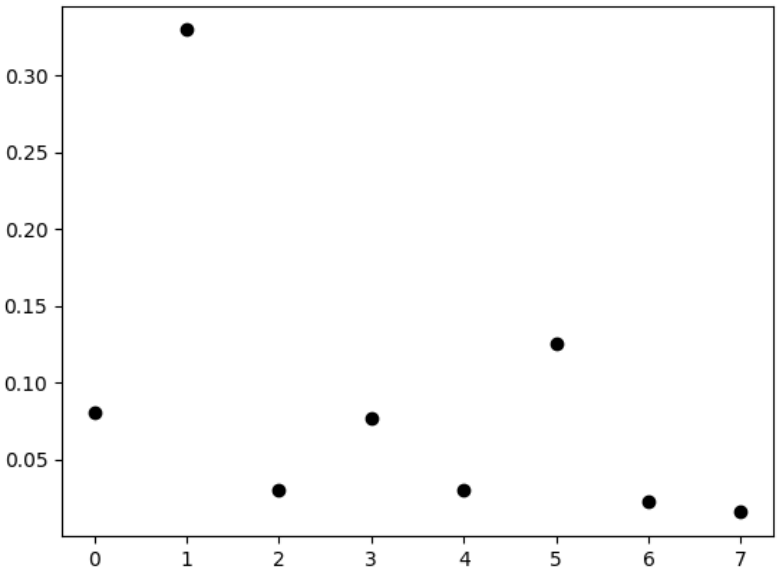
Training average cross entropy error: 0.0001375
Testing average cross entropy error: 2.92e-14

Next, I used psi of 0.01 and epsilon of 0.1.



Training average cross entropy error: 0.0007673
Testing average cross entropy error: 9.06e-10

This got much worse, so I decided to increase psi. I used psi=0.2 and epsilon=0.1



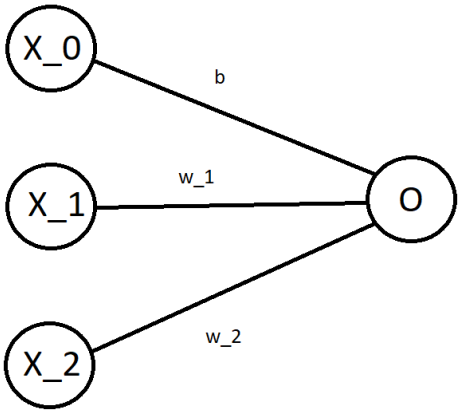
Training average cross entropy error: 3.39e-05
Testing average cross entropy error: 5.33e-17

This obviously got much better. So I decided to test as much as I could After this and I started getting some errors because the values were getting too small. So I decided to stick with my final values from the previous test. In conclusion, a larger step size will make the error jump around a lot more, while a smaller step size has a steady decrease. However, a smaller step size will not converge as quickly. A larger epsilon will oftern make the algorithm stop too soon, and not allow it to discover better optimal values. While too small of an epsilon will never be reached and cause us to miss potentially optimal values.

2 Part 2

2.1 a

Here is the new redesigned MLP with no hidden layer. It consists of just a single input layer and a single output layer.



We can use the old weights to calculate the new weights by substitution.

Since we are using a linear activation function, the output of unit j in the hidden layer is,

$$z_j = c \times (w_j^{h^T} \bar{x}) = c \times (\sum_{i=1}^{n} w_{ij}^h x_i + b_j^h x_0)$$

where $\bar{x}$ is the augmented input vector for some fixed constant scalar c.

More specifically, our z_1 will be,

$$z_1 = c \times ((w_{11}^h x_1 + b_1^h x_0) + (w_{21}^h x_2 + b_1^h x_0)) = c \times (w_{11}^h x_1 + w_{21}^h x_2 + 2b_1^h x_0)$$

And similarily, our z_2 will be,

$$z_2 = c \times (w_{12}^h x_1 + w_{22}^h x_2 + 2b_2^h x_0)$$

The output of unit j in the output layer is,

$$o_j = c \times (w_j^{o^T} \bar{z}) = c \times (\sum_{i=1}^{n} w_{ij}^o z_i + b_j^o z_0)$$

where $\bar{z}$ consists of each neuron in the hidden layer for some fixed constant scalar c.

Similar to above from z_j we can compute o_j

$$o_1 = c \times (w_{11}^o z_1 + w_{21}^o z_2 + 2b_1^o z_0)$$

Since we have no need for the bias term anymore, we can remove it from the equation.

$$o_1 = c \times (w^o_{11} z_1 + w^o_{21} z_2)$$

Now we just have to substitute in for z_1 and z_2 and find the new weights.

$$\begin{aligned}
&o_1 = c \times (w^o_{11} z_1 + w^o_{21} z_2) \\
&= c \times (w^o_{11}(c \times (w^h_{11} x_1 + w^h_{21} x_2 + 2b^h_1 x_0)) + w^o_{21}(c \times (w^h_{12} x_1 + w^h_{22} x_2 + 2b^h_2 x_0))) \\
&= c^2 \times ((w^o_{11} w^h_{11} x_1 + w^o_{11} w^h_{21} x_2 + 2w^o_{11} b^h_1 x_0) + (w^o_{21} w^h_{12} x_1 + w^o_{21} w^h_{22} x_2 + 2w^o_{21} b^h_2 x_0)) \\
&= c^2 \times ((w^o_{11} w^h_{11} + w^o_{21} w^h_{12}) x_1 + (w^o_{11} w^h_{21} + w^o_{21} w^h_{22}) x_2 + (2w^o_{11} b^h_1 + 2w^o_{21} b^h_2) x_0)
\end{aligned}$$

Thus the new weights and bias term are

$$b = 2w^o_{11} b^h_1 + 2w^o_{21} b^h_2 = 2{w^o_1}^T b^h_1$$

$$w_1 = w^o_{11} w^h_{11} + w^o_{21} w^h_{12} = {w^o_1}^T w^h_1$$

$$w_2 = w^o_{11} w^h_{21} + w^o_{21} w^h_{22} = {w^o_1}^T w^h_2$$

2.2 b

Yes it is always possible to represent a neural network that only uses linear neurons without a hidden layer. If you add more neurons to each layer it will not affect the final solution as above, there will only be more weights, but the dot product will remain the same. If you add more layers, then we can think of our new weights as the input for another layer, making our output layer into a hidden layer. So then we just repeat the process, using our new weights to substitute into the new output layer. For example, our new weights from the final equations, would become the vector w^h_1. I.e. $w^h_1 = [w_1, w_2]^T$. There will likely be other neurons added to this layer, which is why I chose w^h_1, I could have also chosen w^h_2.