**Andrew Elysee (aelys2)**

High-level discussion with Max Han (mhan36)

CS446

HW2

1. 1. 1. Since , then the probability of getting a vector given , is   
         This makes sense because if , then the probability of that happening is just , but if then the probability of that is so we take the product of all these probabilities to get .
      2. Since we know that each vector was drawn from one of our Bernoulli distributions with parameters , the expression must be the sum of the products of the probabilities that the kth distribution was chosen and the probability of given . So, we get
   2. 4. Using Bayes rule,
      5. Since there is an imaginary expected value sign around the whole expression, , so we get:
   3. 1. To find the value of   that maximizes the E step, we must set the derivative of the E step relative to   to 0.Rearranging to solve for we get:
      2. First, we will set up a Lagrangian, with to enforce the constraint that . Then take the derivative, and set it equal to 0.  
         First we see that our rearranged constraint is

so, solving for and taking the derivative, we that  
This gives us the final piece for our update for :

1. 1. Diagram

      Description automatically generated
   2. This is the partial derivative using only u, but with some manipulation we get:
   3. A forward pass refers to the calculation of the outputs from the input layers, while a backwards pass is the calculation of the change in weights using some gradient descent method.
   4. To make the computation of this derivative easy, we should save the output of .
   5. To make this calculation easier, we should save and , assuming that is implied to be saved.
   6. We should save and from the forward pass. To obtain the result as early as possible and reuse as many results as possible we should do and finally , because we can reuse the results of in and the results of in like I showed in the equation above. This is reverse of the forward pass.
   7. After the first layer the output dimension is 20 X 24 X 24. After the max pooling layer, we get an output of dimension 20 X 12 X 12.
   8. If we undo the max pooling on the 4x4 output, we see that the output of the second convolution layer is 8x8. From this, we can derive that our filters would be of size 5x5 with a stride of 1, because (input size) \* (kernel size) + 1 = 12-5+1 = 8. The channel dimensions is 50.
2. 1. Submitted on gradescope
   2. Submitted on gradescope
   3. In my graphs the Testing and training errors are nearly indistinguishable but looking closely I think that as C increases, the gap between the test and training error seems to increase, this is because the low complexity of the model helps to prevent overfitting.   
      **Plots are below.**

**Also note that the y-axis label is labeled \_\_\_\_ accuracy but should be \_\_\_\_ Error**  
  
  
Plots for C = 1  
Chart, line chart

Description automatically generated

Plots for C = 2  
Graphical user interface, application

Description automatically generated  
Plots for C = 4  
Graphical user interface, application, histogram

Description automatically generated

* 1. For C=64, the testing and training errors curves are significantly smoother this also start off very accurate after the first epoch. Additionally, the gap between the testing and training sets is much more noticeable as our training set gets a lot closer to 0 than our testing error, this could just be because of the scaling of the axis though. In hindsight, I should have overlayed the graphs.  
     Plots for C = 64  
     Graphical user interface, application

     Description automatically generated