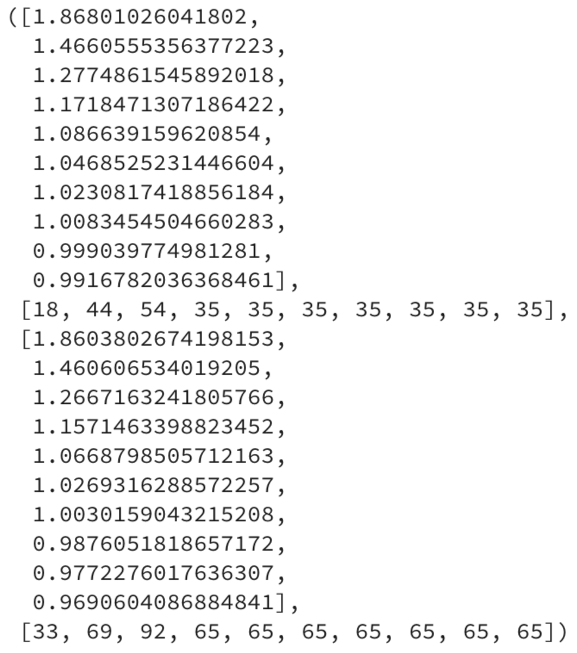
Lavinia Wang
#1473704
CSC578-701

HW 2

In this experiment, I first started changing mini-batch size and eta in scenario 1, where cost function is set to Quatratic and activation function is set to Sigmoid. I incremented the mini-batch size by 1 and left everything else the same, when it's up to 5, I didn't observe any huge difference on accuracy. But when it's larger than 5, accuracy of the first epoch on testing data dropped quite a lot to less than 10, which was around 17. So I felt the size of mini-batch cannot be larger than 5.

Then I reset the parameters to default except eta. In the original code, eta was 1.0 and I wanted to apply Nielsen's rule of thumb, which is changing eta with order of magnitude 1.0, 0.1, 0.01, … So I changed eta to 0.1 and I saw an improvement on accuracy with both training and testing data. In the original code, the output was 65/95 for testing data and 35/55 for testing data. This time, for training data my accuracy was increasing as number of epochs incremented, and reached 70/95 and for testing data, my last epoch reached 44 out of 55. I further decreased eta to 0.01, but the result was not impressing as the learning rate is to low, both accuracy of training and testing went down to 32/95 and 18/55. I tested eta equals 0.05 and 0.5, the results were no better than eta equals 0.1.

With cost function being CrossEntropy, when mini-batch is within [2,5], the accuracy on both training data and testing data are stabilized around 65/96 and 35/55, which is better than the original output, 35/95 for training and 17/55 for testing. The interesting thing is, when I test mini-batch = 10, I have once reached 92/95 on training data and 54/55 on testing data.



```
([1.86801026041802,
  1.4660555356377223,
  1.2774861545892018,
  1.1718471307186422,
  1.086639159620854,
  1.0468525231446604,
  1.0230817418856184,
  1.0083454504660283,
  0.999039774981281,
  0.9916782036368461],
 [18, 44, 54, 35, 35, 35, 35, 35, 35, 35],
 [1.8603802674198153,
  1.460606534019205,
  1.2667163241805766,
  1.1571463398823452,
  1.0668798505712163,
  1.0269316288572257,
  1.0030159043215208,
  0.9876051818657172,
  0.9772276017636307,
  0.9690604086884841],
 [33, 69, 92, 65, 65, 65, 65, 65, 65, 65])
```

Figure 1: cost = CrossEntropy, activation = Sigmoid

When I applied cost = CrossEntropy, hidden layer activation = ReLU, output layer activation = Softmax, my original result is different from support code. I wasn't sure if this result is solid enough to such conclusion: I increased mini-batch to 15 so that I have accuracy of 33/95 for testing and 17/55 for testing. I kept mini-batch to 15 and tested eta = 0.1 and 0.01, with eta = 0.01, I have around 65/95 for training and 35/55 for testing.

When I applied cost = LogLikelihood, hidden layer activation = ReLU, output layer activation = Softmax, the situation is similar. Only when I increased mini-batch to 15 and decreased eta to 0.01 I saw an improvement, which is from 33/95 for testing, 17/55 for testing to 65/95 for training and 35/55 for testing.

When I applied cost = CrossEntropy, activation = Tanh, after I increased mini-batch to 3 and larger, my cost are all 0s and I always have a true divide warning. Only when mini-batch is either 1 or 2 I have valid cost and the warning is still there. This is what made me really confused. If I change eta from 1 to 10, the result remained same but when I decreased eta to 0.1, I have really poor performance on both training and test: 17/95 and 9/55 respectively.

When introduced L2 and L1 regularization, I didn't figure out why my cost function returned all negative numbers. I still get some accuracy change when I change the mini-tach and eta, but I wasn't sure if the computation was correct enough to give me 65/95 accuracy for training and 35/55 for testing when applied L2 regularization. No matter what I changed, I didn't see a huge difference in my output when I applied L1 regularization.

I was very upset and frustrated when I used network "iris-4-20-7-3.dat" for experiment because the dropout function was not perfectly defined. I don't have any accuracy and all my cost function returned negative results. I tried using the previous "iris-423.dat" it worked but still the results were not satisfying. I understand how the mask work conceptually, and how the network architecture is changed when applying this mask. But I didn't figure out how to achieve that by modifying code in feedforward, update mini-batch and backprop. I found a kernel from Kaggle which illustrated the implementation of dropout. The code was concise and intuitive. But Nielsen's code is not well structured, what I mean structured is not the way we talk about network architecture in class.

After this assignment, I felt I have more questions regarding the activation and cost function. In what situation should we apply which function? From stackoverflow I saw someone mentioned that ReLU should not be applied to binary output, which I think makes sense because when $z > 0$, $f(z) = z$. If it's 1 and 0, why not use Sigmoid? Also, for Softmax the result is like probability distribution, what would be the threshold to for classification? If the computed output is 0.5, which class should be assigned?