HW1 Short Answers

Note: I used Momentum SGD for all these computations. The default learning rate is 0.01 but with this learning rate my learning was very slow and test accuracy was almost stagnant when I used an SGD optimizer, even though my programs passed all the tests. It got better when I used my MSGD optimizer. When I consulted a T.A about the problem I was told that there might be some scaling going on somewhere. The T.A and I then proceeded to check my code and we did not find any errors. I don’t know why I am having this problem and T.A could not figure it out either so I changed my learning rate to 0.1 and the learning became much better.

1) Playing around with Leaky Relu slopes (same slope for both layers):

Slope 1: 0.1

Accuracy:

- After 1 epoch: 75.4%
- After 5 epochs: 95.6%
- After 10 epochs: 97.2%
- After 15 epochs: 97.5%
- After 20 epochs: 97.9%

Slope 2: 0.5

Accuracy:

- After 1 epoch: 83.6%
- After 5 epochs: 93.9%
- After 10 epochs: 95.9%
- After 15 epochs: 96.6%
- After 20 epochs: 96.7%

Slope 3: 0.01

Accuracy:

- After 1 epoch: 73.1%
- After 5 epochs: 96.2%
- After 10 epochs: 97.5%
- After 15 epochs: 97.9%
- After 20 epochs: 98.2%

Slope 4: 2

Accuracy:

- After 1 epoch: 91.9%
- After 5 epochs: 95.1%
- After 10 epochs: 96.3%
- After 15 epochs: 96.8%

- After 20 epochs: 96.1%

Slope 5: -0.1

Accuracy:

- After 1 epoch: 75.2%
- After 5 epochs: 92.4%
- After 10 epochs: 0.098% (but loss is NaN)
- After 15 epochs: 0.098% (but loss is NaN)
- After 20 epochs: 0.098% (but loss is NaN)

I got the best accuracy after 20 epochs when the slopes were set to 0.01. When the slope was negative the loss initially decreased and then it became NaN. I think this is because all the negative inputs to the function will have positive results, this will cause each output of the LeakyRelu function to have multiple inputs (like a parabola). This should make classification worse. When the slope was greater than 1 the accuracy was less than when the slope was 0.01, the loss also fluctuated more, and the accuracy did not consistently go down. As I increased the slope the accuracy decreased but not considerably. If the slope is 1 the activation function becomes an identity function and does not introduce any non-linearity.

2) After 20 epochs I found the test accuracy to be: 94.6% and the weights to be: 0.185. This is different from the LeakyRelu case where I found the best accuracy from a slope of 0.01.

3) I added more layers but the accuracy went down as a result. Then I decreased the number of layers, changed learning rate to 0.05, increased the number of epochs to 35 and changed every ReLu layer to PReLu layer and found that after 35 epochs I got an accuracy of 98.5%. I could not reach 99%. Finally my network layout is:

```
LinearLayer(28 * 28, 1000),
PReLULayer(1),
LinearLayer(1000, 100),
PReLULayer(1),
LinearLayer(100, 10),
```