Malachi Eberly

Assignment 4

Using pytorch made it a lot easier to work with the data and get results. Once I figured out how to split up the training and testing into two files that worked together, it made it very user-friendly to run train and watch the loss slowly decrease on each epoch, and then run test and see the overall accuracy. Here are my main findings:

1. Having a higher epoch value made the testing accuracy increase and the loss decrease. There was about a 1.5% increase in accuracy from 20 to 40 epochs and around another 1% increase from 40 to 60 epochs.
2. The number of hidden units used had a very minimal effect on the accuracy. Instead of the standard 256 units, I ran once with 100 and once with 400. They both made it to around 15-16 epochs before reaching my cutoff loss value. I found this stopping point by creating a break: **if sum(loss_lt) > 15: break**. The accuracy was within 1% for these values.
3. The learning rate had a significant effect om the accuracy. I ran two separate tests in addition to the standard learning rate. With a rate of .0001, the accuracy was about 10% lower than with a rate of .001. However, with a learning rate of .01, the accuracy was as high as 95.9%.
4. Going pack to point 2, it is still possible to get good results without a hidden layer altogether. I trained the model with the standard 40 epochs and .001 learning rate and found that the test accuracy only was 1.4% lower than that with 256 hidden units. By increasing the learning rate of the slp test to .01, the test accuracy was 1% higher than the mlp test with the standard .001 learning rate.

MLP				
Epochs	Hidden Units	Test Loss	Test Accuracy	Learning Rate
40	256	0.004984638114	90.66481018	1.00E-03
20	256	0.006237411575	88.992836	1.00E-03
60	256	0.004466848389	91.52069855	1.00E-03
15	100	0.007367030071	87.65923309	1.00E-03
16	400	0.006849901691	88.41560364	1.00E-03
40	256	0.01888871952	78.94108582	1.00E-04
40	256	0.001862136422	95.93949127	1.00E-02
SLP				
Epochs	Test Loss	Test Accuracy	Learning Rate	
40	0.006204061068	89.24163818	1.00E-03	
40	0.004466655148	91.620224	1.00E-02	

Figure 1: Data for the tests

The confusion matrices for both the mlp and slp tests showed a clear diagonal:

MLP 40 Epochs, 256 Hidden Units, 1e-3 LR									
955.0	0.0	11.0	3.0	1.0	12.0	15.0	4.0	8.0	11.0
0.0	1108.0	6.0	1.0	3.0	3.0	3.0	15.0	8.0	7.0
4.0	1.0	904.0	20.0	6.0	5.0	4.0	26.0	8.0	4.0
2.0	4.0	17.0	910.0	1.0	42.0	1.0	5.0	29.0	12.0
0.0	1.0	16.0	0.0	908.0	12.0	17.0	9.0	10.0	42.0
7.0	1.0	1.0	28.0	1.0	752.0	16.0	0.0	26.0	14.0
8.0	4.0	14.0	3.0	11.0	17.0	899.0	0.0	12.0	0.0
1.0	1.0	15.0	15.0	2.0	10.0	1.0	934.0	13.0	21.0
3.0	15.0	41.0	23.0	8.0	31.0	2.0	4.0	848.0	6.0
0.0	0.0	7.0	7.0	41.0	8.0	0.0	31.0	12.0	892.0

SLP 40 Epochs, 1e-3 LR									
960.0	0.0	13.0	6.0	1.0	16.0	16.0	2.0	10.0	13.0
0.0	1103.0	8.0	1.0	6.0	7.0	3.0	20.0	9.0	8.0
2.0	2.0	876.0	17.0	5.0	6.0	7.0	30.0	12.0	10.0
3.0	4.0	22.0	895.0	0.0	48.0	2.0	3.0	30.0	11.0
0.0	1.0	17.0	1.0	899.0	16.0	12.0	12.0	10.0	48.0
1.0	2.0	0.0	33.0	1.0	720.0	18.0	0.0	26.0	15.0
6.0	4.0	20.0	6.0	12.0	16.0	894.0	1.0	13.0	0.0
1.0	0.0	22.0	16.0	1.0	10.0	1.0	912.0	13.0	26.0
7.0	19.0	46.0	23.0	8.0	43.0	5.0	5.0	836.0	6.0
0.0	0.0	8.0	12.0	49.0	10.0	0.0	43.0	15.0	872.0

Figure 2: Confusion matrices

Finally, I compared the CPU and GPU running times. I was successfully able to get a significantly quicker time on the GPU than the CPU. Within the epoch loop, I called target.to(dev) and data.to(dev) directly after defining target and data. For three trials of each, I was able to run mlp_train.py an average of 43 seconds faster.

Trial:	1	2	3
Gpu times with standard settings:	175.9927695	177.3105516	174.1426175
Cpu times with standard settings:	222.2154832	215.5783432	219.4350483
Standard settings: 40 epochs, 256 hidden units, 1e-3 learning rate			

Figure 3: Times for GPU and CPU in seconds