Programming 1 Report

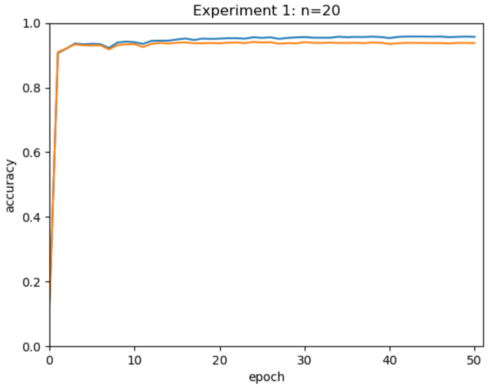
Josiah Vincent

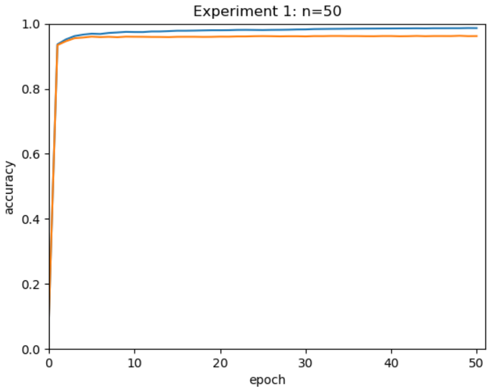
Experiment 1

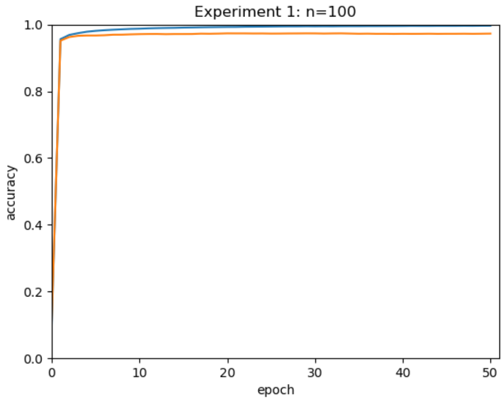
For experiment 1 we are to modify the value of the number of hidden neurons in our hidden layer. We are asked to vary the number (n=20,50,100). It is also important to note that our momentum value is constant at 0.9.

Accuracy Plots:

The blue line is the accuracy over the training set and the orange line is the accuracy over the test set.







Confusion Matrices:

Each follow the same format given in homework1

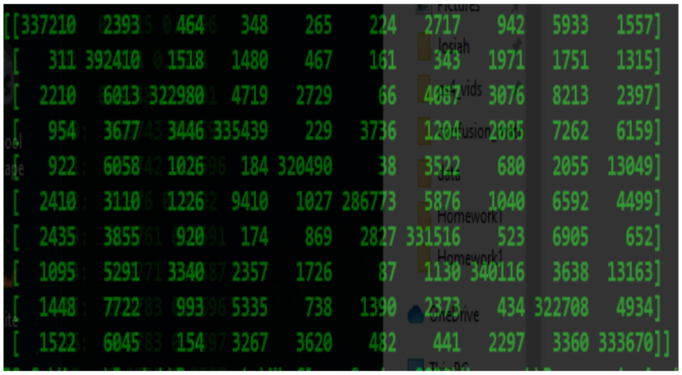
0…   …10

0

…

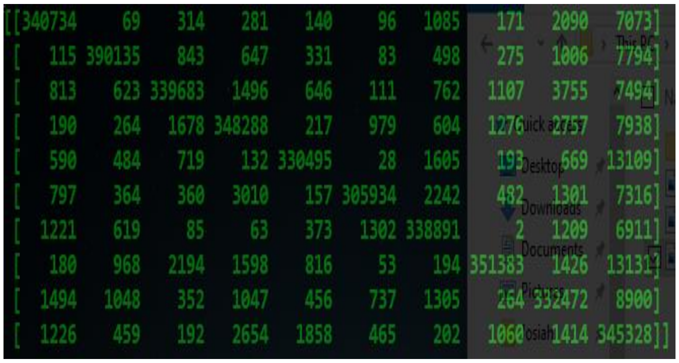
10

N=20



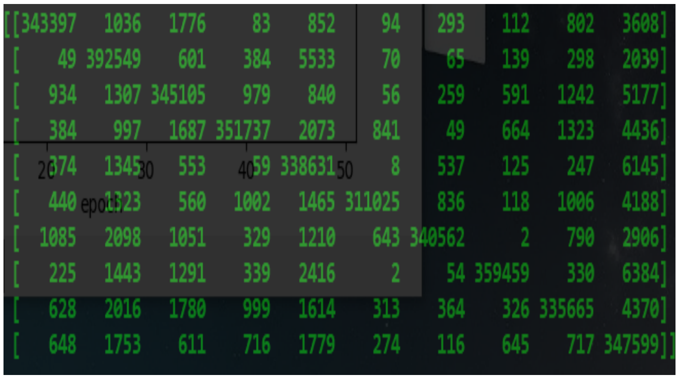
```
[[337210   2393    464    348    265    224   2717    942   5933   1557]
 [   311 392410   1518   1480    467    161    343   1971   1751   1315]
 [  2210   6013 322980   4719   2729     66   4087   3076   8213   2397]
 [   954   3677   3446 335439    229   3736   1204   2085   7262   6159]
 [   922   6058   1026    184 320490     38   3522    680   2055  13049]
 [  2410   3110   1226   9410   1027 286773   5876   1040   6592   4499]
 [  2435   3855    920    174    869   2827 331516    523   6905    652]
 [  1095   5291   3340   2357   1726     87   1130 340116   3638  13163]
 [  1448   7722    993   5335    738   1390   2373    434 322708   4934]
 [  1522   6045    154   3267   3620    482    441   2297   3360 333670]]
```

N=50



```
[[340734     69    314    281    140     96   1085    171   2090   7073]
 [   115 390135    843    647    331     83    498    275   1006   7794]
 [   813    623 339683   1496    646    111    762   1107   3755   7494]
 [   190    264   1678 348288    217    979    604   1276   2757   7938]
 [   590    484    719    132 330495     28   1605    193    669  13109]
 [   797    364    360   3010    157 305934   2242    482   1301   7316]
 [  1221    619     85     63    373   1302 338891      2   1209   6911]
 [   180    968   2194   1598    816     53    194 351383   1426  13131]
 [  1494   1048    352   1047    456    737   1305    264 332472   8900]
 [  1226    459    192   2654   1858    465    202   1060   1414 345328]]
```

N=100



```
[[343397   1036   1776     83    852     94    293    112    802   3608]
 [    49 392549    601    384   5533     70     65    139    298   2039]
 [   934   1307 345105    979    840     56    259    591   1242   5177]
 [   384    997   1687 351737   2073    841     49    664   1323   4436]
 [   374   1345    553     59 338631      8    537    125    247   6145]
 [   440   1323    560   1002   1465 311025    836    118   1006   4188]
 [  1085   2098   1051    329   1210    643 340562      2    790   2906]
 [   225   1443   1291    339   2416      2     54 359459    330   6384]
 [   628   2016   1780    999   1614    313    364    326 335665   4370]
 [   648   1753    611    716   1779    274    116    645    717 347599]]
```

Discussion

(1) How does the number of hidden units affect the final accuracy on the test data?

As you can see by the plots included above, the higher the number of hidden neurons the higher the accuracy. I also noticed less oscillation at the beginning with the number of neurons set higher. You can see that when the number of neurons is 20 there is quite a bit of oscillation at the beginning.
I also went a little further with this and set the number of hidden neurons to 2 and it confirmed my theory of oscillation. If the number of neurons is low than you see rather large oscillations in the data.

(2) How does it affect the number of epochs needed for training to converge?

From the plots I deduce that if the number of hidden neurons is larger than it appears to converge faster. You can see for the value of N=100 that it jumps up closer to its convergence much quicker than N=20. You can still see this in N=50, but it is a little harder to spot.

(3) Is there evidence that any of your networks has overfit to the training data? If so, what is the evidence?

I am not seeing any overfitting here. You can tell because for each of the graphs if you look at the orange(test data) this is data that the machine has not been trained over and it more or less follows the same trend as the training data. There are no big differences so we can deduce the model is not overfitting.

(4) How do your results compare to the results obtained by your perceptron in HW 1?

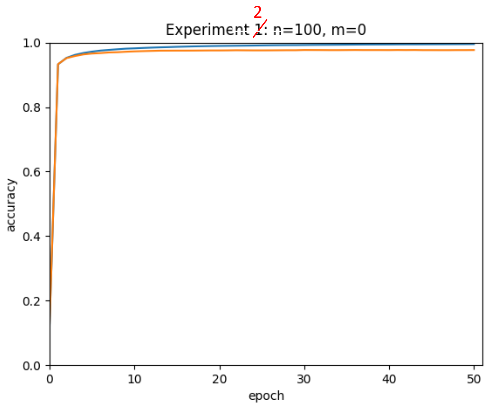
I will be honest and start by saying my homework 1 was rough to say the least. From what I understood about homework 1 the accuracy was much less. We can see that adding a hidden layer and particularly with ones with many hidden neurons it was much more accurate over the data.

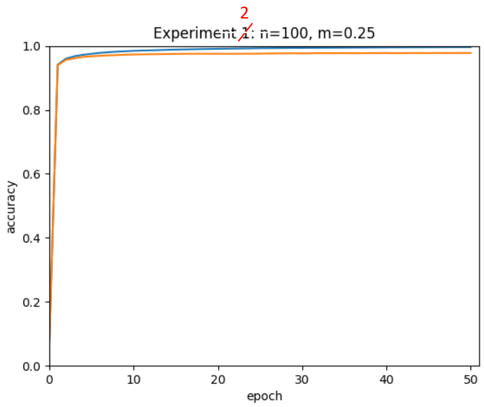
Experiment 2

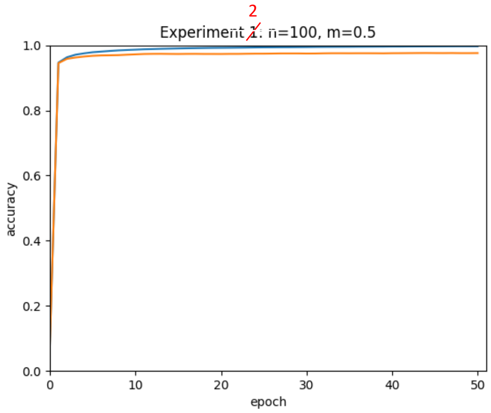
For experiment 2 we were asked to fix the number of neurons in the hidden layer to 100 (n=100) and vary the momentum value (previously 0.9 in experiment 1) to 0, 0.25, and 0.5. The graph from experiment 1 where the momentum value was 0.9 and the number of hidden neurons was 100 will also be included in the plots section (last plot in the section).

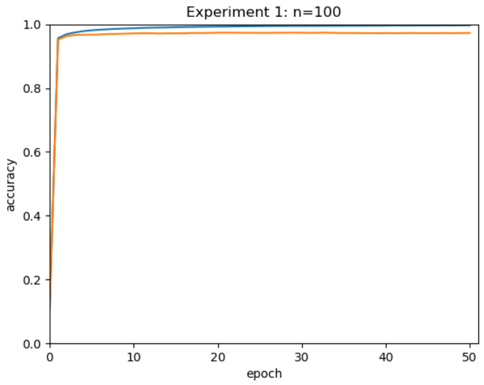
Accuracy Plots

The blue line is the accuracy over the training set and the orange line is the accuracy over the test set.









Confusion Matrices

Each follow the same format given in homework1

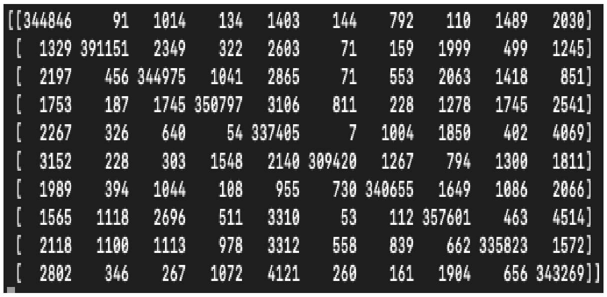
0… …10

0

…

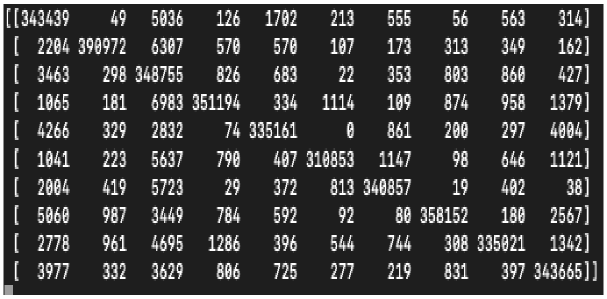
10

Momentum value = 0



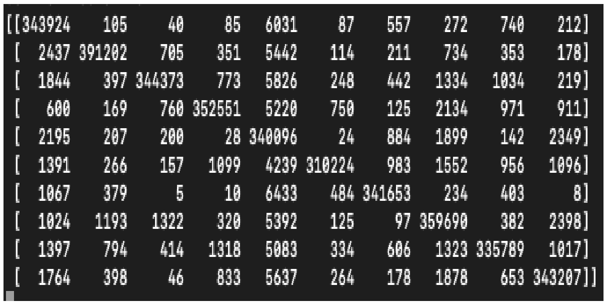
```
[[344846     91   1014    134   1403    144    792    110   1489   2030]
 [  1329 391151   2349    322   2603     71    159   1999    499   1245]
 [  2197    456 344975   1041   2865     71    553   2063   1418    851]
 [  1753    187   1745 350797   3106    811    228   1278   1745   2541]
 [  2267    326    640     54 337405      7   1004   1850    402   4069]
 [  3152    228    303   1548   2140 309420   1267    794   1300   1811]
 [  1989    394   1044    108    955    730 340655   1649   1086   2066]
 [  1565   1118   2696    511   3310     53    112 357601    463   4514]
 [  2118   1100   1113    978   3312    558    839    662 335823   1572]
 [  2802    346    267   1072   4121    260    161   1904    656 343269]]
```

Momentum value = 0.25



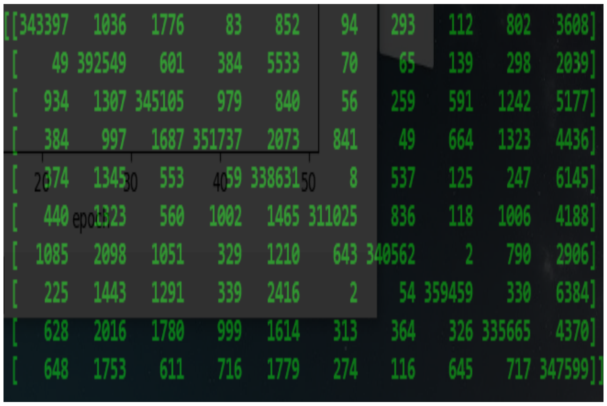
```
[[343439     49   5036    126   1702    213    555     56    563    314]
 [  2204 390972   6307    570    570    107    173    313    349    162]
 [  3463    298 348755    826    683     22    353    803    860    427]
 [  1065    181   6983 351194    334   1114    109    874    958   1379]
 [  4266    329   2832     74 335161      0    861    200    297   4004]
 [  1041    223   5637    790    407 310853   1147     98    646   1121]
 [  2004    419   5723     29    372    813 340857     19    402     38]
 [  5060    987   3449    784    592     92     80 358152    180   2567]
 [  2778    961   4695   1286    396    544    744    308 335021   1342]
 [  3977    332   3629    806    725    277    219    831    397 343665]]
```

Momentum value = 0.5



```
[[343924    105     40     85   6031     87    557    272    740    212]
 [  2437 391202    705    351   5442    114    211    734    353    178]
 [  1844    397 344373    773   5826    248    442   1334   1034    219]
 [   600    169    760 352551   5220    750    125   2134    971    911]
 [  2195    207    200     28 340096     24    884   1899    142   2349]
 [  1391    266    157   1099   4239 310224    983   1552    956   1096]
 [  1067    379      5     10   6433    484 341653    234    403      8]
 [  1024   1193   1322    320   5392    125     97 359690    382   2398]
 [  1397    794    414   1318   5083    334    606   1323 335789   1017]
 [  1764    398     46    833   5637    264    178   1878    653 343207]]
```

Momentum value = 0.9



```
[[343397   1036   1776     83    852     94    293    112    802   3608]
 [    49 392549    601    384   5533     70     65    139    298   2039]
 [   934   1307 345105    979    840     56    259    591   1242   5177]
 [   384    997   1687 351737   2073    841     49    664   1323   4436]
 [   374   1345    553     59 338631      8    537    125    247   6145]
 [   440   1323    560   1002   1465 311025    836    118   1006   4188]
 [  1085   2098   1051    329   1210    643 340562      2    790   2906]
 [   225   1443   1291    339   2416      2     54 359459    330   6384]
 [   628   2016   1780    999   1614    313    364    326 335665   4370]
 [   648   1753    611    716   1779    274    116    645    717 347599]]
```

Discussion

(1) How does the momentum value affect the final accuracy of the test data?

It is hard to see much of a difference from the momentum value that we picked, but when I look at the plots it appears that when you have a lower momentum value that on the test set specifically you get a higher accuracy. It is not by much, but you can see a small increase.

(2) How does it affect the number of epochs needed for training to converge?

Again, it is really hard to tell by looking at the graphs, but it does look as though the higher the momentum value is that it converges slightly faster.

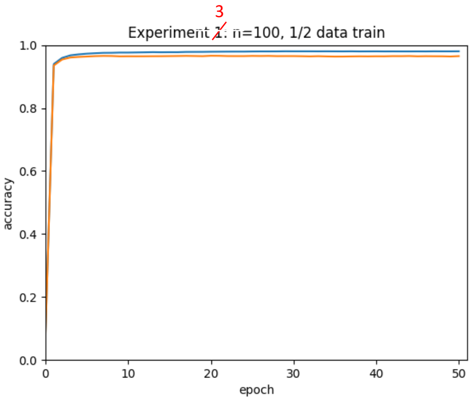
(3) Is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?

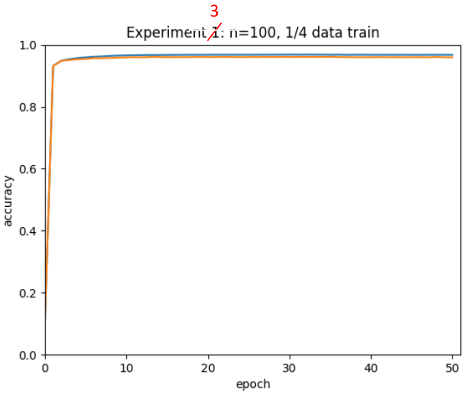
No. Again, both the accuracy on the train and the test sets follow the same trend and are very close in accuracy. There are no major discrepancies an any of the plots and I am not seeing any overfitting to any of the graphs.

Experiment 3

In experiment 3 we were asked to fix the number of hidden neurons to 100 and the momentum value to 0.9. We were also asked to train over half and then a quarter of the training set instead of going over the whole set.

Accuracy Plots





Confusion Matrices

Each follow the same format given in homework1

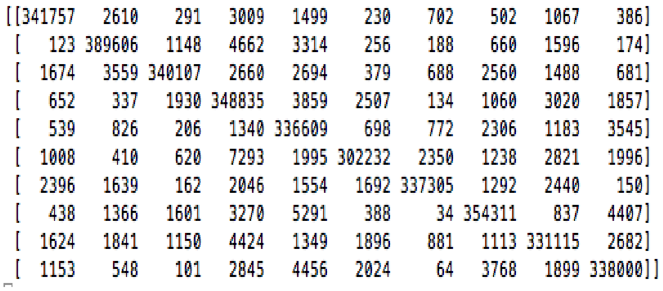
0… …10

0

…

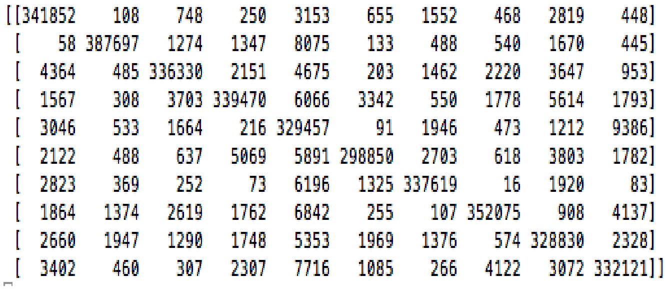
10

Half training set



```
[[341757   2610    291   3009   1499    230    702    502   1067    386]
 [   123 389606   1148   4662   3314    256    188    660   1596    174]
 [  1674   3559 340107   2660   2694    379    688   2560   1488    681]
 [   652    337   1930 348835   3859   2507    134   1060   3020   1857]
 [   539    826    206   1340 336609    698    772   2306   1183   3545]
 [  1008    410    620   7293   1995 302232   2350   1238   2821   1996]
 [  2396   1639    162   2046   1554   1692 337305   1292   2440    150]
 [   438   1366   1601   3270   5291    388     34 354311    837   4407]
 [  1624   1841   1150   4424   1349   1896    881   1113 331115   2682]
 [  1153    548    101   2845   4456   2024     64   3768   1899 338000]]
```

Quarter training set



```
[[341852    108    748    250   3153    655   1552    468   2819    448]
 [    58 387697   1274   1347   8075    133    488    540   1670    445]
 [  4364    485 336330   2151   4675    203   1462   2220   3647    953]
 [  1567    308   3703 339470   6066   3342    550   1778   5614   1793]
 [  3046    533   1664    216 329457     91   1946    473   1212   9386]
 [  2122    488    637   5069   5891 298850   2703    618   3803   1782]
 [  2823    369    252     73   6196   1325 337619     16   1920     83]
 [  1864   1374   2619   1762   6842    255    107 352075    908   4137]
 [  2660   1947   1290   1748   5353   1969   1376    574 328830   2328]
 [  3402    460    307   2307   7716   1085    266   4122   3072 332121]]
```

Discussion

(1) How does the size of the training data affect the final accuracy of the test data?

This is not by much, but not surprisingly if you have more data to train on your accuracy is higher. Also because what we did was take ½ or ¼ of the training set there is a possibility that there may have been more of a particular digit so you may see one be a little more accurate on a digit than the other.

(2) How does it affect the number of epochs needed for training to converge?

Again, it is very difficult to see, but if you have less data to train over it looks like you will converge slightly faster.

(3) Is there any evidence that any of your networks has overfit to the training data? If so, what is that evidence?

No neither have overfit. Something interesting to note is that for ¼ of the training data it appears the accuracy was much more consistent. The accuracy was not higher, but you can see that the test accuracy and train accuracy lines are almost on top of each other. This is closer than any of the others in the whole assignment.

Conclusion

This assignment explored using Multi Layered Perceptron to learn the MNIST dataset and try to determine what handwritten digit was in each image. We varied a some of the parameters of the backpropagation functions (e.g. number of hidden neurons and the momentum value). We also varied how much data we were training on.

What could I do better?

It took me a long time to do the training for each of the examples and I think if I would have utilized more linear algebra termed expression. Doing that would have sped up the training some.