Lawrence Hsu

CS 545

Dr. Mitchell

Experiment One

Description: In experiment one, the number of hidden nodes was changed while momentum and the learning rate were fixed. The initial number of hidden nodes was 20 and then increased to 50 and 100 for following trials. Learning rate was fixed at 0.1 and momentum was fixed at 0.9. After each epoch, the accuracy of the training and test set were calculated and a confusion matrix was created after training was complete

As the number of the hidden nodes increase, the number of false negatives and false positives decreases as shown by hidden nodes of 50 and 100. It seems there is a limit on how many nodes will decrease the number of misclassifications. 1) As the number of hidden nodes increases, so does the accuracy. However, the increase in accuracy is very small (from 20 to 50: increase was 2%, 50 to 100: 1% increase).2) Seems like the number of hidden nodes affect how fast the training set converges to an extent. At 20 nodes, it doesn't seem like it would converge anytime soon as it seems was steadily increasing. At 50 and 100 nodes, I couldn't tell but roughly around 30 epoch the curves started to converge. 3)There was no overfitting. It may look like overfitting as the test curve isn't following the training curve; however, that is due to scale of the y axis. Secondly, the accuracy should be decreasing as the accuracy of the training increases, which I see no indication in the graphs. 4) Compared to the perceptrons, the accuracy on neural network is a lot higher and stable compare to the perceptron where the accuracy is always oscillating.

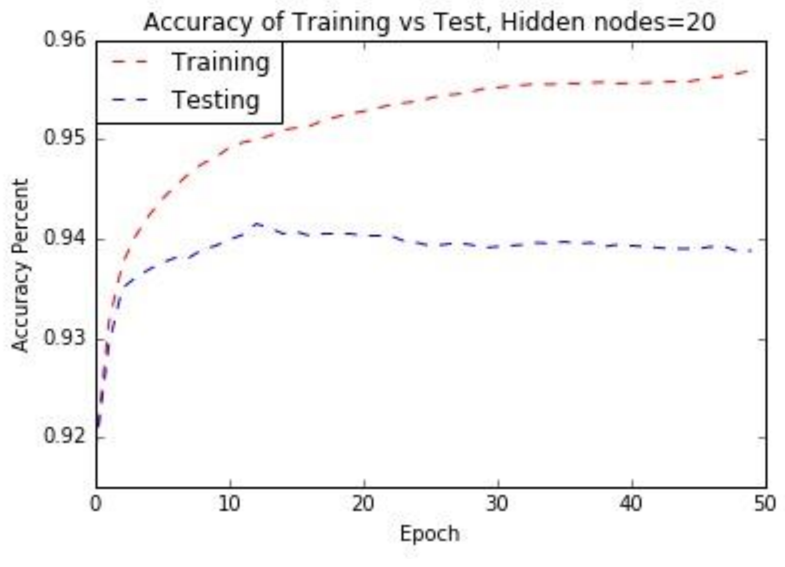


Fig1. Plot of accuracy of training vs test set of 20 hidden nodes with fixed 0.9 momentum

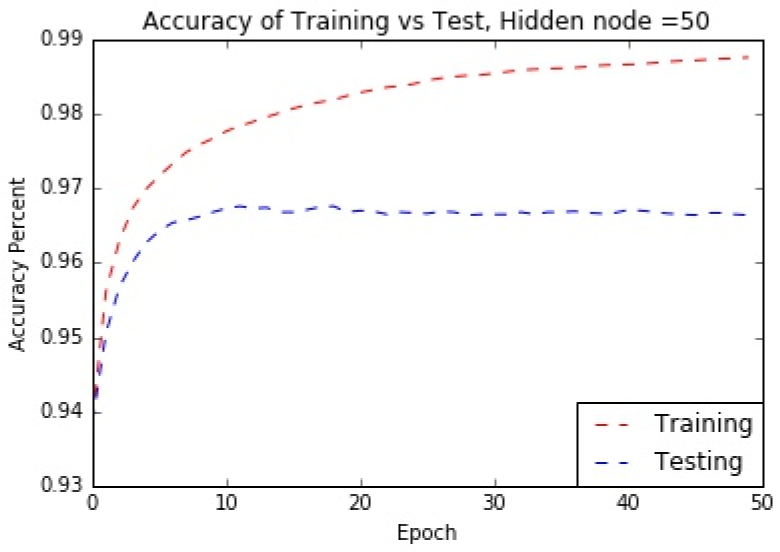


Fig 2. Plot of accuracy of training vs test set of 50 hidden nodes with fixed 0.9 momentum

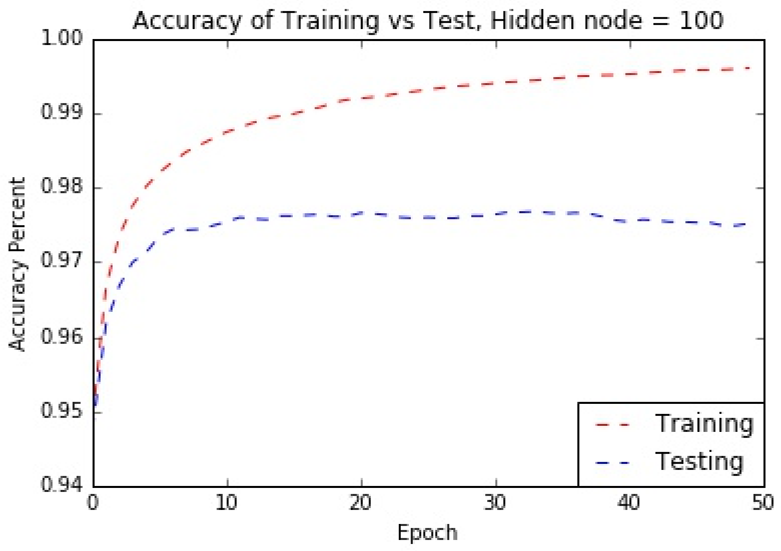


Fig 3. Plot of accuracy of training vs test set of 100 hidden nodes with fixed 0.9 momentum

```
Predicted    0     1    2    3    4    5    6    7    8    9
Actual
0          962     0    0    2    1    0    8    1    6    0
1            0  1113    4    3    0    1    4    1    9    0
2            2     5  934   20   11    4   15    6   33    2
3            4     0    8  944    0   20    3    8   15    8
4            1     2    3    1  914    0   12    0    6   43
5            6     1    2   25    5  791   16    2   34   10
6           16     3    1    1    3    8  917    0    9    0
7            0     5   15   15    7    0    2  947   11   26
8            9    10    1    8    8    6    9    2  916    5
9            5     8    0   10   13    4    2    5   12  950
```

Fig 4. Confusion matrix of 20 hidden nodes

```
Predicted    0     1    2    3    4    5    6    7    8    9
Actual
0          969     0    1    0    1    1    2    1    4    1
1            0  1120    3    3    0    1    2    2    4    0
2            6     2  989    9    2    0    3    8   11    2
3            1     0    5  973    0    5    0   10   11    5
4            1     0    0    1  946    0    8    1    2   23
5            6     0    0    8    1  847   15    2    7    6
6           11     4    0    1    3    7  926    0    5    1
7            1     4   12    6    0    0    0  988    4   13
8            8     1    2    6    3    3    5    5  938    3
9            3     7    1    5   12    1    1    6    5  968
```

Fig 5. Confusion matrix of 50 hidden nodes

```
Predicted    0     1    2    3    4    5    6    7    8    9
Actual
0          969     0    1    0    1    1    2    1    4    1
1            0  1120    3    3    0    1    2    2    4    0
2            6     2  989    9    2    0    3    8   11    2
3            1     0    5  973    0    5    0   10   11    5
4            1     0    0    1  946    0    8    1    2   23
5            6     0    0    8    1  847   15    2    7    6
6           11     4    0    1    3    7  926    0    5    1
7            1     4   12    6    0    0    0  988    4   13
8            8     1    2    6    3    3    5    5  938    3
9            3     7    1    5   12    1    1    6    5  968
```

Fig 6. Confusion matrix of 100 hidden nodes

Experiment 2:

In this experiment, learning rate and number of hidden nodes were kept fixed while momentum changed. Learning rate was fixed at 0.1 and number of hidden nodes was fixed at 100. The following momentum values were tested: 0, 0.25, and 0.5. After each epoch, the accuracy of the training and test set were calculated and a confusion matrix was created after training was complete.

The confusion matrixes of the different momentum values don't seem to be very different from one another. There isn't many false positives or false negatives. 1) Momentum doesn't seem to make a difference on the accuracy as the accuracy on the test set for each value hovers around 97%. I don't think it would make a difference because the objective of the momentum parameter is to speed up the process of reaching the correct target value. That's if the change of weight is going in the right direction. 2) It generally seems like it decreased the amount of epoch needed for the training set to converge. I couldn't tell from 0.25 and 0.5 but from 0.5 to 0.0, I could see that 0.5 converge quicker than the 0.0.

3) There was no overfitting. It may look like overfitting as the test curve isn't following the training curve; however, that is due to scale of the y axis. Secondly, the accuracy should be decreasing as the accuracy of the training increases, which I see no indication in the graphs.

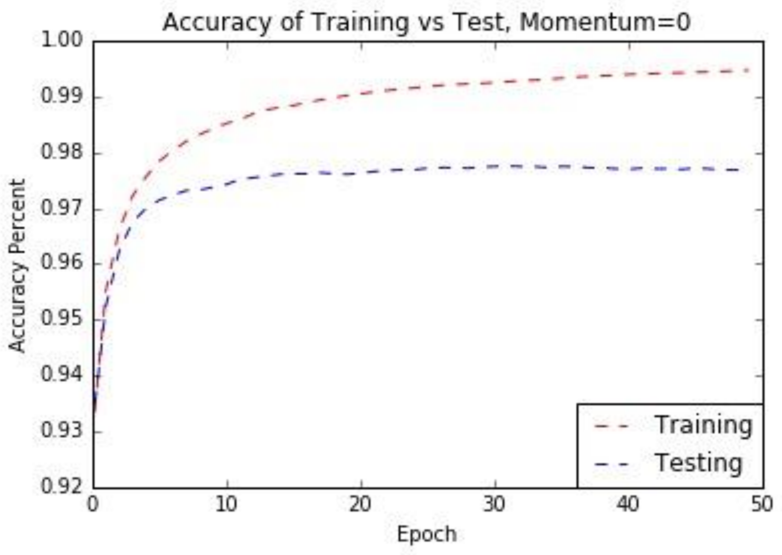


Fig 7. Plot of accuracy of Training vs Test, with fixed 100 hidden nodes and 0 momentum

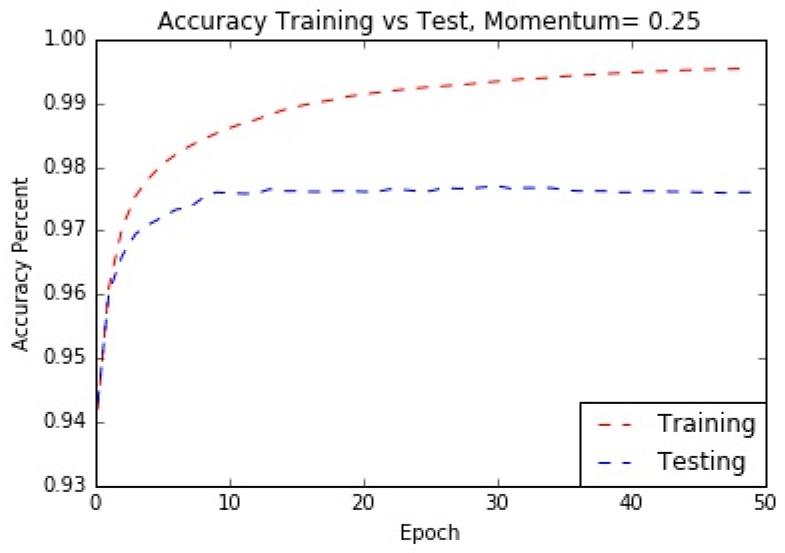


Fig 8. Plot of accuracy of Training vs Test, with fixed 100 hidden nodes and 0.25 momentum

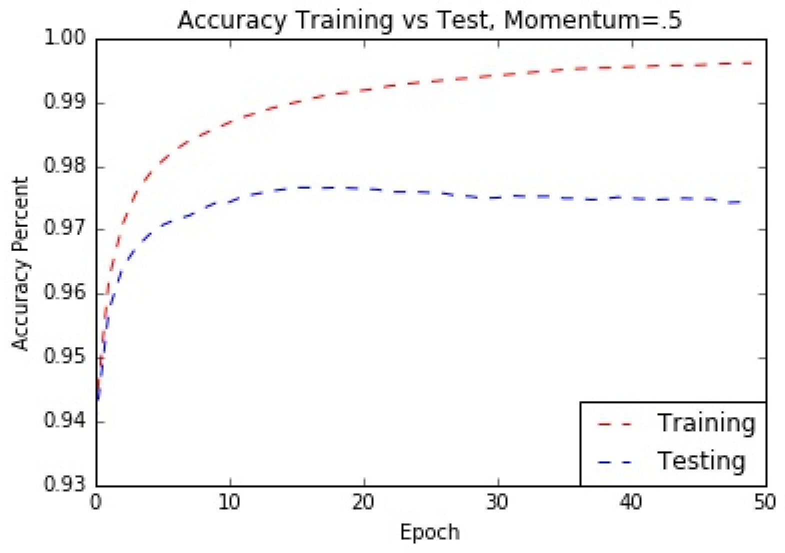


Fig 9. Plot of accuracy of Training vs Test, with fixed 100 hidden nodes and 0.5 momentum

```
Predicted    0     1     2    3    4    5    6    7    8    9
Actual
0          971     0     0    1    1    0    3    1    3    0
1            0  1126     3    1    0    1    2    0    2    0
2            2     5  1005    4    2    0    3    5    5    1
3            0     0     3  991    0    4    0    3    3    6
4            1     0     0    0  957    0    6    1    2   15
5            2     1     0    9    0  860    6    3    7    4
6            5     4     1    1    2    4  937    0    4    0
7            1     3    10    3    3    0    1  995    2   10
8            4     1     1    4    4    4    1    4  949    2
9            2     6     0    7    8    1    1    3    6  975
```

Fig 10. Confusion matrix of 100 hidden nodes momentum =0

```
Predicted    0     1     2    3    4    5    6     7    8    9
Actual
0          971     0     1    0    0    1    3     1    3    0
1            0  1126     0    3    0    1    3     0    2    0
2            6     3  1001    1    1    0    5     8    7    0
3            1     0     3  996    0    2    0     3    3    2
4            2     0     2    0  956    0    4     1    1   16
5            4     0     1   13    0  853    9     2    6    4
6            6     3     1    1    1    7  934     0    4    1
7            0     3    12    2    0    0    0  1001    3    7
8            4     0     3    3    4    4    1     3  948    4
9            4     7     0    5   11    1    0     4    3  974
```

Fig 11.confusion matrix of 100 hidden nodes momentum=0.25

```
Predicted    0     1     2    3    4    5    6    7    8    9
Actual
0          973     0     0    0    1    0    1    1    4    0
1            0  1121     4    1    1    1    1    2    4    0
2            4     4  1003    4    0    0    3    5    7    2
3            0     1     6  986    0    4    0    5    5    3
4            2     1     0    1  956    0    5    0    1   16
5            3     0     1    7    1  860    4    3    7    6
6            9     3     0    1    0    8  930    0    5    2
7            0     4    14    1    5    0    0  991    5    8
8            4     1     2    3    3    2    1    3  950    5
9            5     4     0    5   11    0    0    4    7  973
```

Fig 12. Confusion matrix of 100 hidden nodes momentum=0.5

Experiment 3

Description: In this experiment, learning rate, momentum, number of hidden nodes were fixed while the number of training examples that were used varied. Learning rate was set to 0.1, and momentum was set to 0.9, number of hidden nodes was set to 100. The number of training examples were reduced to half or quarter for each trial. After each epoch, the accuracy of the training and test set were calculated and a confusion matrix was created after training was complete .

There isn't many false negatives or false positive when the training set is reduced to half or quarter. 1) The accuracy just didn't grow after the first few epoch. This occurred in both the half and quarter set. 2) It cut down on how many epoch was needed for training to converge as you can see in the quarter set roughly round 15 epoch, you don't see much change in the curve.3) There was no overfitting. It may look like overfitting as the test curve isn't following the training curve; however, that is due to scale of the y axis. Secondly, the accuracy should be decreasing as the accuracy of the training increases, which I see no indication in the graphs.

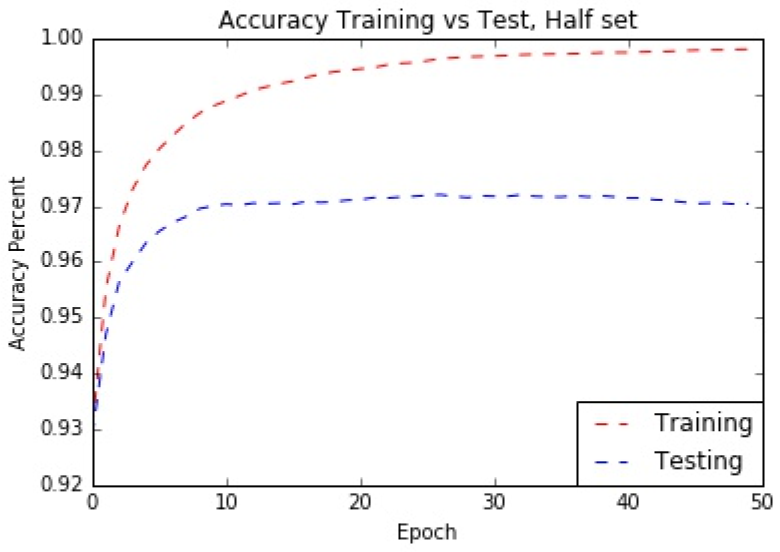


Fig 13. Plot of accuracy of training vs test set, half of training set; momentum and hidden nodes fixed

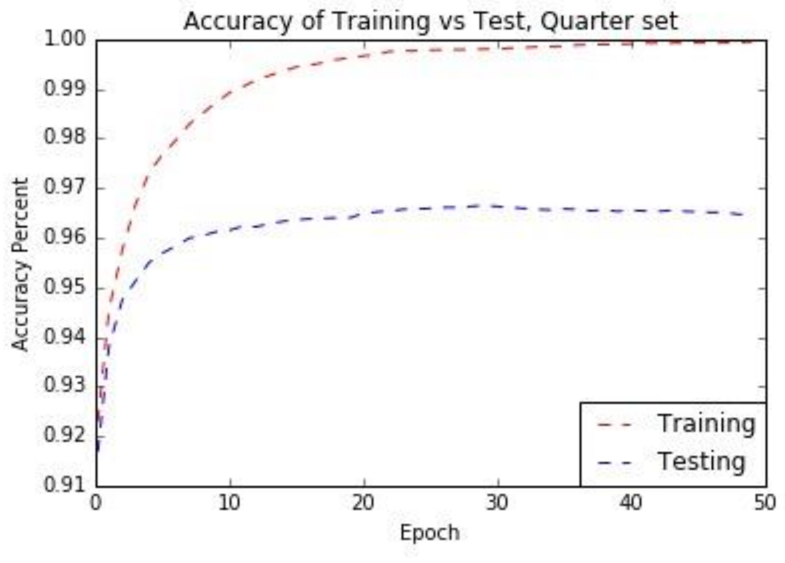
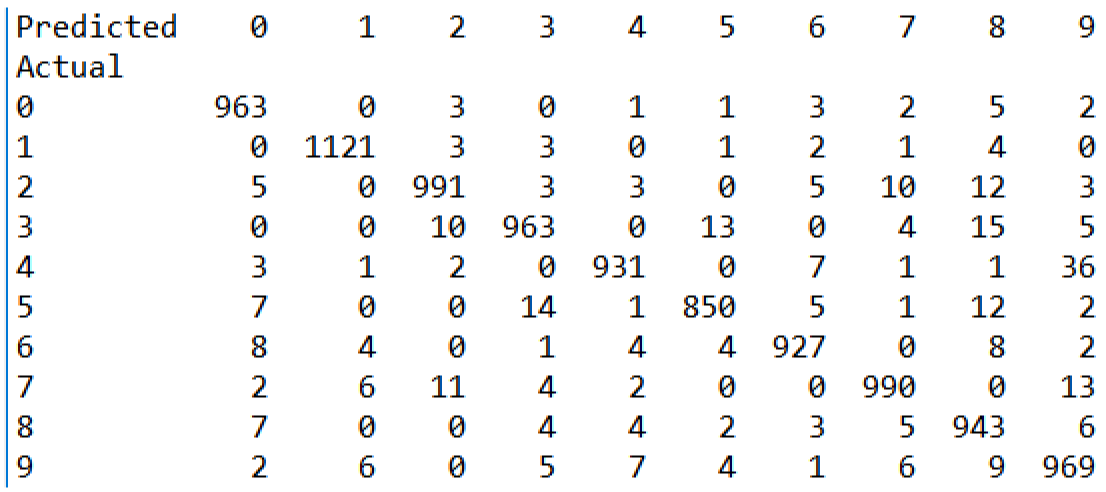


Fig 14. Plot of accuracy of training vs test set, quarter of training set; momentum and hidden nodes fixed

```
Predicted     0     1     2     3     4     5     6     7     8     9
Actual
0           962     1     3     1     0     0     7     1     2     3
1             0  1118     1     5     0     1     4     1     5     0
2             8     1   996     6     2     0     1     8     6     4
3             1     0     1   983     0     7     0     4     9     5
4             1     0     1     0   959     0     5     0     3    13
5             3     1     0    13     0   850     8     2    11     4
6             9     3     0     1     1     8   928     1     6     1
7             1     6     9     7     1     0     0   991     4     9
8             5     1     3     2     4     5     2     4   944     4
9             4     4     0     3     9     4     1     5     6   973
```

Fig 15. Confusion matrix half population



```
Predicted    0     1    2    3    4    5    6    7    8    9
Actual
0          963     0    3    0    1    1    3    2    5    2
1            0  1121    3    3    0    1    2    1    4    0
2            5     0  991    3    3    0    5   10   12    3
3            0     0   10  963    0   13    0    4   15    5
4            3     1    2    0  931    0    7    1    1   36
5            7     0    0   14    1  850    5    1   12    2
6            8     4    0    1    4    4  927    0    8    2
7            2     6   11    4    2    0    0  990    0   13
8            7     0    0    4    4    2    3    5  943    6
9            2     6    0    5    7    4    1    6    9  969
```

Fig 16. Confusion matrix of quarter population

Instructions for each experiment:

- For all experiments and their trials:
    - Use the fxn figure() to create the graph
    - Use fxn cm() to create confusion matrix
        - The variables that are used in the confusion matrix are: test_pred1(predicted values of test set),ttline2(actual values of test set)
- Experiment One:
- For all trials
    - hweight=hweight+dhidden+(momentum[3]*prehid)
    - weight=weight+dinput+(momentum[3]*preout)
    - 20 nodes
        - Codes for weight should look like this:
            - hweight=weightgen(10,21)
            - weight=weightgen(20,785)
    - 50 nodes
        - Codes for weight should look like this:
            - hweight=weightgen(10,51)
            - weight=weightgen(50,785)
    - 100 nodes
        - Codes for weight should look like this:
            - hweight=weightgen(10,101)
            - weight=weightgen(100,785)
- Experiment two:
    - For all trials
        - Codes for weight should look like this:
            - hweight=weightgen(10,51)
            - weight=weightgen(50,785)
        - momentum 0
            - hweight=hweight+dhidden+(momentum[0]*prehid)

        - weight=weight+dinput+(momentum[0]*preout)
    - momentum 0.25
        - hweight=hweight+dhidden+(momentum[1]*prehid)
        - weight=weight+dinput+(momentum[1]*preout)
    - momentum 0.5
        - hweight=hweight+dhidden+(momentum[2]*prehid)
        - weight=weight+dinput+(momentum[2]*preout)
- Experiment three
  - For all trials
    - Codes for weight should look like this:
        - hweight=weightgen(10,101)
        - weight=weightgen(100,785)
        - hweight=hweight+dhidden+(momentum[3]*prehid)
        - weight=weight+dinput+(momentum[3]*preout)
    - half set
        - for x in range(len(ori2)/2):…
        - for z in range(len(ori2)/2):…
    - quarter set
        - for x in range(len(ori2)/4):…
        - for z in range(len(ori2)/4):…