Report for CS4341 project 2

Author: Yihong Zhou (yzhou8), Mengwen Li (mli2)

1. Test ANN with hidden neurons set to 5, change the holdout percentage and the samples used for testing.

a) holdout percentage is 10% and the last 10% data is used for testing:

Run count	Error rate
1	0.25
2	0.25
3	0.3
4	0.25
5	0.3
Average error rate	0.27

```
C:\Users\Mengwen\Desktop\WPI\junior\cTerm\cs4341\project2>python ann.py -h 5 -p 0.1 hw5data.txt
5   0.1   hw5data.txt
error rate is  0.25

C:\Users\Mengwen\Desktop\WPI\junior\cTerm\cs4341\project2>python ann.py -h 5 -p 0.1 hw5data.txt
5   0.1   hw5data.txt
error rate is  0.25

C:\Users\Mengwen\Desktop\WPI\junior\cTerm\cs4341\project2>python ann.py -h 5 -p 0.1 hw5data.txt
5   0.1   hw5data.txt
error rate is  0.3

C:\Users\Mengwen\Desktop\WPI\junior\cTerm\cs4341\project2>python ann.py -h 5 -p 0.1 hw5data.txt
5   0.1   hw5data.txt
error rate is  0.25

C:\Users\Mengwen\Desktop\WPI\junior\cTerm\cs4341\project2>python ann.py -h 5 -p 0.1 hw5data.txt
5   0.1   hw5data.txt
error rate is  0.3
```

Fig. 1. Error rate for running 5 times with the setting above.

```
# Test for hold out last 10%
for i in range(0, len(fileData)):
    if i < int(len(fileData) * (1 - holdOutPerc)):
        trainingSet.append(fileData[i])
    else:
        validationSet.append(fileData[i])
```

Fig. 2. Code used to achieve setting above.

b) holdout percentage is 10% and the first 10% data is used for testing:

Run count	Error rate
1	0.25
2	0.35
3	0.25
4	0.25
5	0.25
Average error rate	0.27

```
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.1
5   0.1   hw5data.txt
error rate is  0.25
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.1
5   0.1   hw5data.txt
error rate is  0.35
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.1
5   0.1   hw5data.txt
error rate is  0.25
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.1
5   0.1   hw5data.txt
error rate is  0.25
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.1
5   0.1   hw5data.txt
error rate is  0.25
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.1
5   0.1   hw5data.txt
error rate is  0.35
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.1
5   0.1   hw5data.txt
error rate is  0.25
```

Fig. 3. Error rate for running 5 times with the setting above.

```
# Test for hold out first 10%
for i in range(0, len(fileData)):
    if i < int(len(fileData) * holdOutPerc):
        validationSet.append(fileData[i])
    else:
        trainingSet.append(fileData[i])
```

Fig. 4. Code used to achieve setting above.

c) holdout percentage is 30% and the last 30% data is used for testing:

Run count	Error rate
1	0.266666666667
2	0.35
3	0.35
4	0.25
5	0.283333333333
Average error rate	0.2999998

```
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.266666666667
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.35
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.35
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.25
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.283333333333
```

Fig. 5. Error rate for running 5 times with the setting above.

```
# Test for hold out last 30%
for i in range(0, len(fileData)):
    if i < int(len(fileData) * (1 - holdOutPerc)):
        trainingSet.append(fileData[i])
    else:
        validationSet.append(fileData[i])
```

Fig. 6. Code used to achieve setting above.

d) holdout percentage is 30% and the first 30% data is used for testing:

Run count	Error rate
1	0.4666667
2	0.1833333
3	0.23333333
4	0.45
5	0.2
Average error rate	0.31

```
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.466666666667
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.183333333333
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.233333333333
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.45
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.3
5   0.3   hw5data.txt
error rate is  0.2
```

Fig. 7. Error rate for running 5 times with the setting above.

```
# Test for hold out first 30%
for i in range(0, len(fileData)):
    if i < int(len(fileData) * holdOutPerc):
        validationSet.append(fileData[i])
    else:
        trainingSet.append(fileData[i])
```

Fig. 8. Code used to achieve setting above.

e) holdout percentage is 50% and the last 50% data is used for testing:

Run count	Error rate
1	0.41
2	0.51
3	0.27
4	0.27
5	0.51
Average error rate	0.394

```
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.44
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.51
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.27
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.27
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.51
```

Fig. 9. Error rate for running 5 times with the setting above.

```
# Test for hold out last 50%
for i in range(0, len(fileData)):
    if i < int(len(fileData) * (1 - holdOutPerc)):
        trainingSet.append(fileData[i])
    else:
        validationSet.append(fileData[i])
```

Fig. 10. Code used to achieve setting above.

f) holdout percentage is 50% and the first 50% data is used for testing:

Run count	Error rate
1	0.23
2	0.48
3	0.21
4	0.39
5	0.52
Average error rate	0.366

```
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.23
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.48
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.21
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.39
Karrenzhous-MacBook-Pro-2:project2 zhoukarren$ python ann.py hw5data.txt -p 0.5
5   0.5   hw5data.txt
error rate is  0.52
```

Fig. 11. Error rate for running 5 times with the setting above.

```
# Test for hold out first 50%
for i in range(0, len(fileData)):
    if i < int(len(fileData) * holdOutPerc):
        validationSet.append(fileData[i])
    else:
        trainingSet.append(fileData[i])
```

Fig. 12. Code used to achieve setting above.

Conclusion:

Six combination of testing data and holdout percentages was used to test our ANN algorithm.

We use 10%, 30% and 50% as our hold out percentages and for each hold out percentage p, we first test our ANN by holding out the last p%, then, test is by holding out the first p% in order to test our algorithm on different test data. Each combination was run five times and the average error rate was calculated.

From the table above, we can tell that when the hold out percentage is the same and the holding out data is different, the average error rate is almost the same. Both 27% for holding out 10%, 29.999% for holding out 30% and 31% for holding out another 31%, 39.4% for holding out 50% and 36.667% for holding out another 50%.

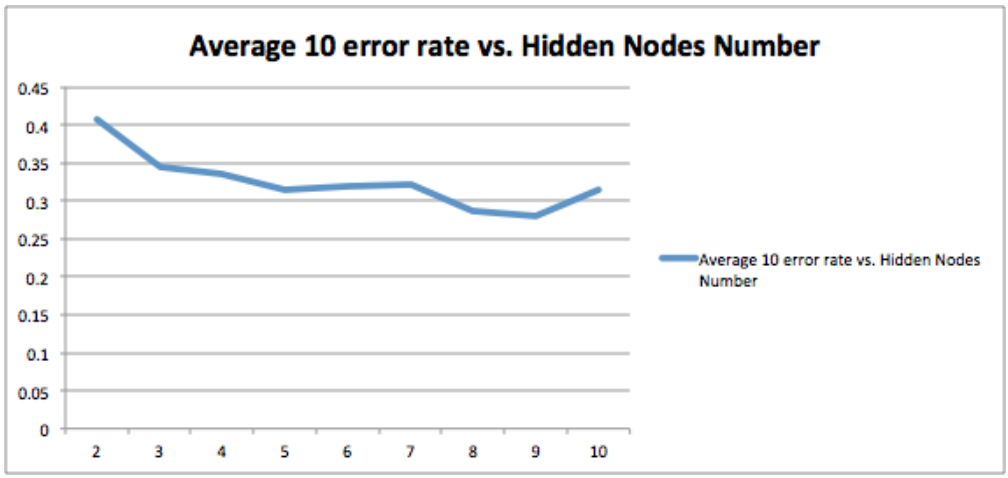
We can also tell that as the holding out percentage increases, the average error rate gradually increases. 27% for holding out 10%, around 30% for holding out 30% and around 37% for holding out 50%. The reason for this might be the ANN is not being trained well enough before we apply the validation set when the hold out percentage is too high.

2. Test ANN with the number of hidden neurons rangin from 2 to 10. Run the algorithm on the first 80% of the training data and test on the remaining 20%.

The table below shows the error rate we get by averaging the error rate of 10 runs for a given hidden node number:

Hidden Nodes Number	Average 10 error rate
2	0.4075
3	0.345
4	0.335
5	0.315
6	0.32
7	0.3225
8	0.2875
9	0.28
10	0.315

The graph below shows the plot we get from the table above. The x-axis is the number of hidden nodes and the y-axis is the error rate.



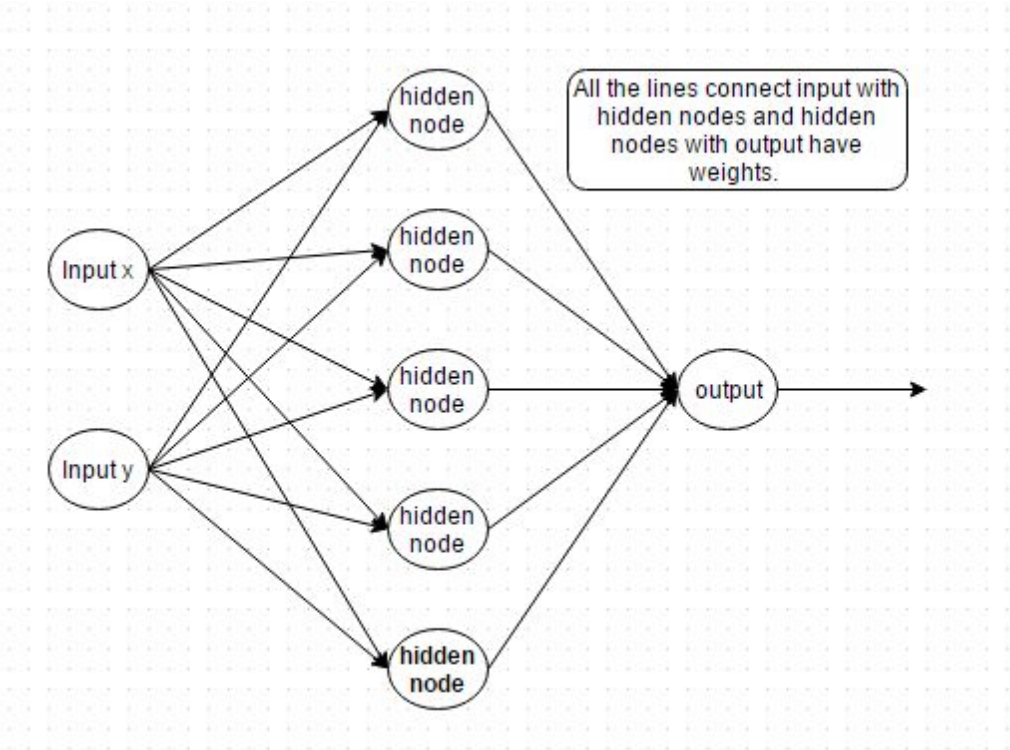
Conclusion:

From the plot above, we can tell that when the number of hidden nodes increaes from 2 to 5, we get a big drop on error rate. Then, from 5 to 7, we get a little increase. Then, from 7 to 9, we get a second big drop. Then, when we use 10 hidden nodes, we get a relative increase on error rate. By increasing hidden node number from 2 to 9, we can tell from the graph that trend of the error rate is decreasing, this is because we have more parameters to adjust with the increasing of hidden nodes, as a result, we achieve higher accuracy. But when we increase the hidden node number to 10, the accuracy rate decreased instead. This may be caused by overfitting. The parameters in this case were tuned too much to fit the training set so it does not perform very well on the validation set.

3.

Discuss any simplifying assumptions that you made while implementing the neural network:

In our neural network, we assume there are two input nodes, one output node and five hidden nodes by default. Both input nodes are connected to all hidden nodes and each hidden node has one output, these outputs are summed up as the input for the output node. The step size “alpha” was set to 0.15 and the training iteration was set to 180. The topology of our neural network implementation is shown in the following figure:



How resistant would your implementation be to changes in the number of input nodes, hidden layers, output nodes and network topology:

In our implementation, we use a “weight” array to record all the weights in our neural network. Also, our calculation for errors was also done in array form. If the number of nodes need to be changed, our array size for “weight” also needs to be changed. If the number of hidden layer is increased, we need to change our “ann3” function to adapt to more layers, we also need to add code in our back propagation function to calculate the error rate for the added layers. Also, when the number of hidden node increases too much, overfitting may occur.