Fei Ding
Section A
CS 3600
24 November 2019

Note:
This part is not analysis but an overview of my files relating to question 5 through 8.

For codes of question 5 through 8, each main method will produce its output in a file called “output_buffer.txt”. I managed to write all my outputs to a file rather than spitting them out in the command line, and later rename this output file to “q*_output_raw.txt” before executing code for the next question. So be very careful when running the code again on question 8 because whenever some question code executes it might go and overwrite (or append to) whatsoever in “output_buffer.txt”. Just treat “output_buffer.txt” as a place for terminal output. If the file is not there, it will be created after some code executes first.

I have written another code file named “rawToCSV.py” which helps me extract useful information from the programs’ output as .csv files, and later import those .csv into .xlsx spreadsheets. Notice that this file depends python’s *re* module library for regular expression matching, but I could largely have done this by hand if I strictly cannot import any library.

I also changed “NeuralNetUtil.py” file by adding a new function as specified by requirement for question 8, and thus I include this file in my submission.

Questions 5

Related Files:

q5.py: code for answering this question.
q5_pen.txt: code output when executing code, pen data portion.
q5_car.txt: code output when executing code, car data portion.

Output:

Accuracy	Pen Data	Car data
Max	0.903659	0.990000
Average	0.897313	0.981862
Standard Deviation	0.005359	0.009695

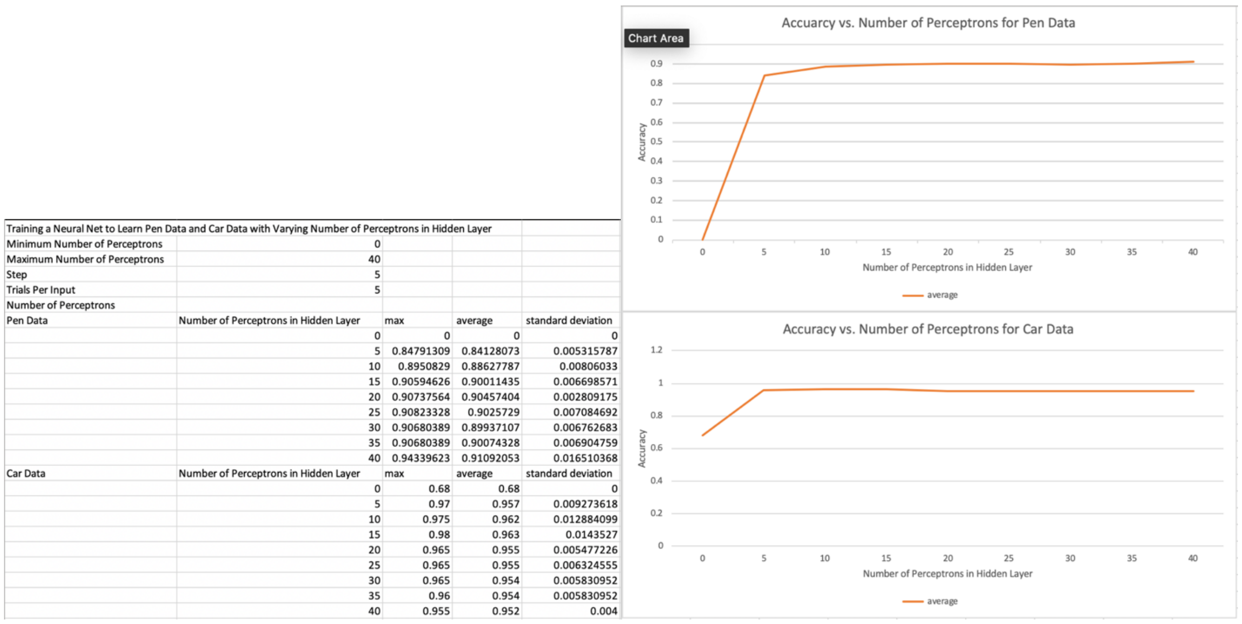
Questions 6

Related Files:

q6.py: code for answering this question.
q6_output_raw.txt: raw output produced by program.
q6_output_csv.csv: converted .csv file for program output.
q6_chart.xlsx: excel spreadsheet which includes table output and chart.

Output:

Here is a screenshot, and they are also visible in the spreadsheet.



Accuarcy vs. Number of Perceptrons for Pen Data
Chart Area
Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
Number of Perceptrons in Hidden Layer
average
Accuracy vs. Number of Perceptrons for Car Data
Accuracy
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
30
35
40
Number of Perceptrons in Hidden Layer
average

Training a Neural Net to Learn Pen Data and Car Data with Varying Number of Perceptrons in Hidden Layer				
Minimum Number of Perceptrons	0			
Maximum Number of Perceptrons	40			
Step	5			
Trials Per Input	5			
Number of Perceptrons				
Pen Data	Number of Perceptrons in Hidden Layer	max	average	standard deviation
	0	0	0	0
	5	0.84791309	0.84128073	0.005315787
	10	0.8950829	0.88627787	0.00806033
	15	0.90594626	0.90011435	0.006698571
	20	0.90737564	0.90457404	0.002809175
	25	0.90823328	0.9025729	0.007084692
	30	0.90680389	0.89937107	0.006762683
	35	0.90680389	0.90074328	0.006904759
	40	0.94339623	0.91092053	0.016510368
Car Data	Number of Perceptrons in Hidden Layer	max	average	standard deviation
	0	0.68	0.68	0
	5	0.97	0.957	0.009273618
	10	0.975	0.962	0.012884099
	15	0.98	0.963	0.0143527
	20	0.965	0.955	0.005477226
	25	0.965	0.955	0.006324555
	30	0.965	0.954	0.005830952
	35	0.96	0.954	0.005830952
	40	0.955	0.952	0.004

Analysis:

For pen data, as the number of perceptrons increase in the hidden layer, the accuracy of the neural network sharply increases first. The neural network even works very well with only 5 perceptrons in the hidden layer, with roughly 85% accuracy. Then, the accuracy stabilizes and changes slowly after we add more into the hidden layer. Also, on the training side not reflected on this figure, more perceptrons in hidden layer definitely has resulted in longer training time.

For car data, the same trend is observed as in the pen data. However, I also spot that the accuracy is also slowly decreasing (though not very noticeable) after it has reached its maximum at the time we have roughly 15 perceptron. I am not sure with this actual number which starts dropping, but clearly the network does not work the best with more and more number of perceptrons, maybe due to the effect of overfitting because we do not have much data in the car data file (7000~ in pen but only 1000~ in car data, so it is reasonable that I do not observe overfitting effect in pen data but a little bit in the car data).

Questions 7

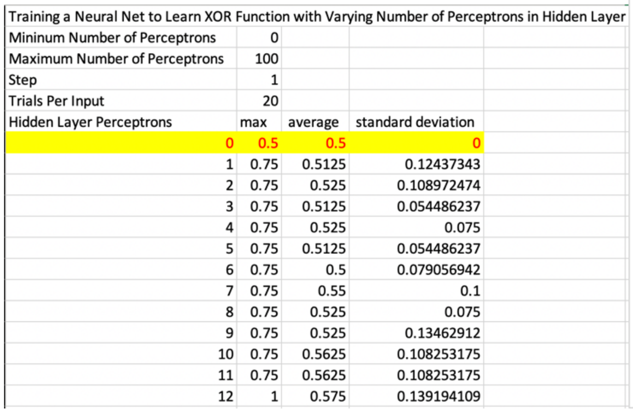
Related Files:

q7.py: code for answering this question.
q7_output_raw.txt & q7_output_raw2.txt: raw output produced by program.
q7_output_csv.csv & q7_output_csv2.csv: converted .csv files for program output.
q7_chart.xlsx: excel spreadsheet which includes table output and chart, not required.

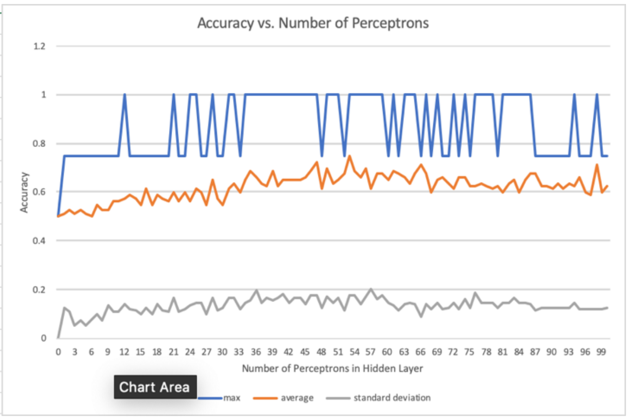
Examples Used for Training:

```
xorTrainData = [([0, 0], [0]), ([0, 1], [1]), ([1, 0], [1]), ([1, 1], [0])]
```

Output:



Training a Neural Net to Learn XOR Function with Varying Number of Perceptrons in Hidden Layer			
Mininum Number of Perceptrons	0		
Maximum Number of Perceptrons	100		
Step	1		
Trials Per Input	20		
Hidden Layer Perceptrons	max	average	standard deviation
0	0.5	0.5	0
1	0.75	0.5125	0.12437343
2	0.75	0.525	0.108972474
3	0.75	0.5125	0.054486237
4	0.75	0.525	0.075
5	0.75	0.5125	0.054486237
6	0.75	0.5	0.079056942
7	0.75	0.55	0.1
8	0.75	0.525	0.075
9	0.75	0.525	0.13462912
10	0.75	0.5625	0.108253175
11	0.75	0.5625	0.108253175
12	1	0.575	0.139194109



To see more, go to the excel spreadsheet.

Analysis:

While training a neural network to learn xor function (with 200 epochs as default), it always works with only 50% accuracy whenever we have zero perceptrons in hidden layer. The first time I hit a 100% accuracy is when I have 12 perceptrons in the hidden layer. As I further increase the number of perceptrons in the single hidden layer, it is more likely I get a neural network that works perfectly (average accuracy).

On the other hand, since we only have 4 non-repeating examples in our dataset, I suppose having 200 epochs is not enough. As I change to a maximum of 20,000 epochs, even training a neural network with only 2 perceptrons in hidden layer will ALWAYS produce 100% accuracy, so the real problem lies in insufficient training rather than a too simple hidden layer because having excessively many perceptrons in hidden layer tends to overfit in other scenarios.

This is the same result as I expect because a network with no hidden layer or only one hidden layer is too simple to learn xor, as it is impossible to cover all cases with a linear function like y = ax1 + bx2 + c , but after we include a hidden layer with more and more perceptrons (even 2 will be enough with extensive training, but more if we train fewer epochs), it becomes likely for the network to learn complex functions like xor by handling non-linear relationships.

Questions 8

Related Files:

q8.py: code for answering this question.
q8_output_raw.txt: raw output produced by program.
q8_output_csv.csv: converted .csv file for program output.
q8_chart.xlsx: excel spreadsheet which includes table output and chart.
q8_readme.txt: instructions on how to run code for this question.
iris.data: the raw dataset file I use for training, downloaded from UCI

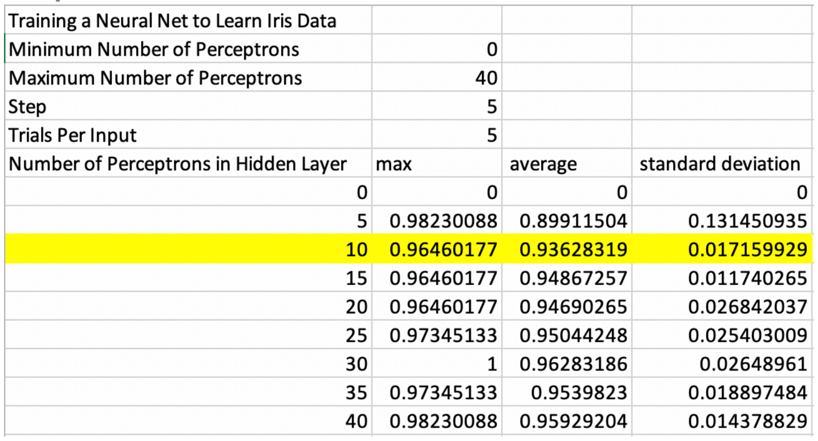
Description:

For this question while exploring the ML datasets on the UCI site, I came across the Iris dataset which is a classification task to determine the type of iris (setosa, versicolor, and virginica) given four features: sepal length (cm), sepal width (cm), petal length (cm), and petal width (cm).

How to Run this Code:

Same as question 5, execute the main method in “q8.py”, and the code itself will take care of calling right functions and doing multiple rounds of training. The output will be in “output_buffer.txt”. If necessary, change “output_buffer.txt” to “q8_output_raw.txt” and run “rawToCSV.py” by uncommenting the correct line and commenting the other lines in the main function. This will produce “q8_output_csv.csv”. Afterwards, the data in the “q8_charts.xlsx” will be updated automatically when choosing to bind data in the spreadsheet. If the data does not update automatically in the spreadsheet due to security warning, click on “Enable Content” on the right.

Output:



Training a Neural Net to Learn Iris Data			
Minimum Number of Perceptrons	0		
Maximum Number of Perceptrons	40		
Step	5		
Trials Per Input	5		
Number of Perceptrons in Hidden Layer	max	average	standard deviation
0	0	0	0
5	0.98230088	0.89911504	0.131450935
10	0.96460177	0.93628319	0.017159929
15	0.96460177	0.94867257	0.011740265
20	0.96460177	0.94690265	0.026842037
25	0.97345133	0.95044248	0.025403009
30	1	0.96283186	0.02648961
35	0.97345133	0.9539823	0.018897484
40	0.98230088	0.95929204	0.014378829

To Avoid overfitting (since we have only 150 examples in the dataset) yet guarantee a decent accuracy, I choose to report the stats with 10 perceptions in the hidden layer. **The max accuracy is about 0.9646, with an average of 0.9363, and standard deviation of 0.0172.**