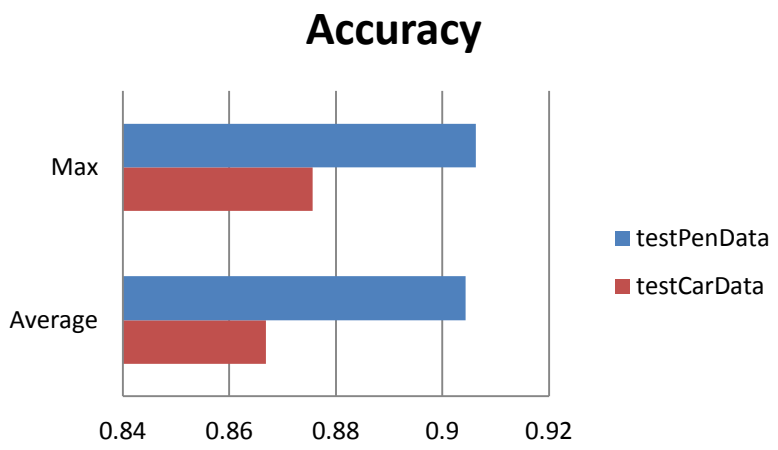
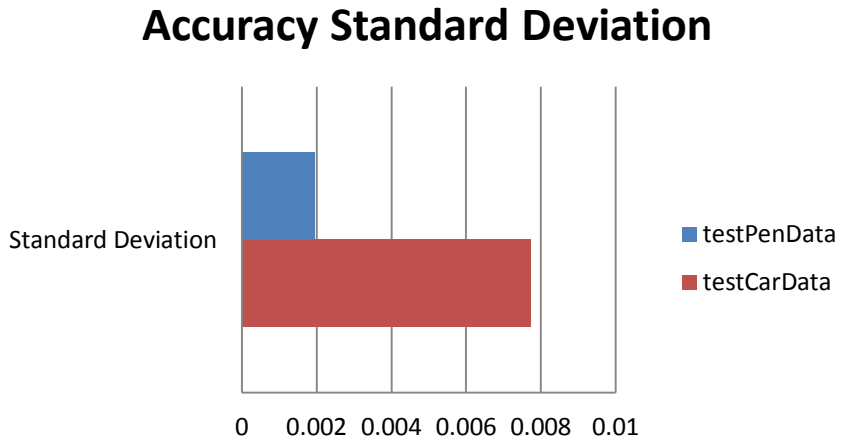
Question 5

testPenData Accuracy	
Max	0.906232132647
Average	0.904345340194
Standard Deviation	0.001969030916

testCarData Accuracy	
Max	0.875654450262
Average	0.866884816754
Standard Deviation	0.007728057288



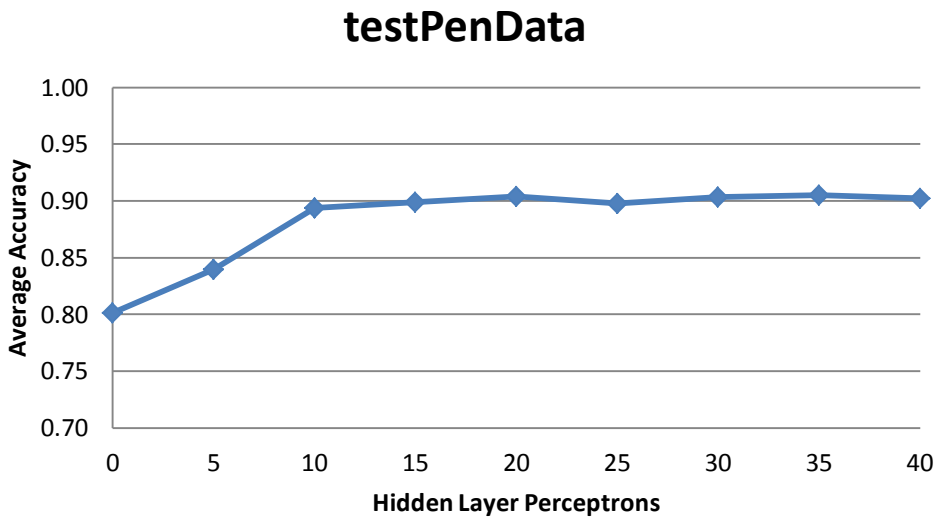


After running 5 iterations of both testPenData and testCarData with default parameters, the above results were produced. The pen data scored higher than the car data in every aspect: higher maximum and average accuracy and lower standard deviation. I originally expected the car data to perform better than pen data because there are significantly more parameters and possible values for each parameter in the pen data than in the car data. I believe this is because the default parameter for testPenData has more hidden layer perceptrons than testCarData has, so adding more perceptrons for the car data may make it perform better.

Question 6

testPenData

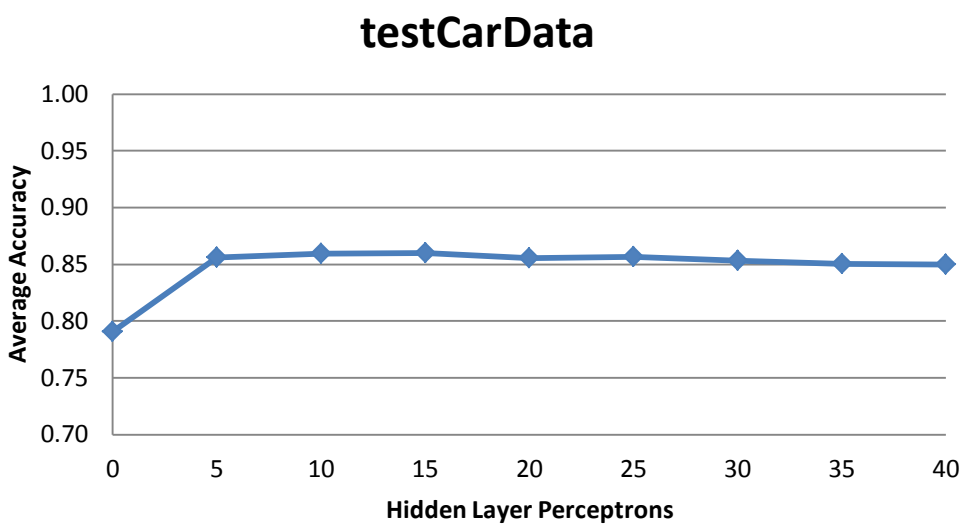
Hidden Layer Perceptrons	Max Accuracy	Average Accuracy	Standard Deviation
0	0.801600914808	0.801200686106	0.000291539137
5	0.853630646083	0.839508290452	0.009441063218
10	0.899085191538	0.894110920526	0.004661120806
15	0.902801600915	0.898913664951	0.005413891624
20	0.906518010292	0.904116638079	0.001360248971
25	0.906232132647	0.897827329903	0.007394456131
30	0.905946255003	0.903201829617	0.002850760632
35	0.908233276158	0.904974271012	0.002913147902
40	0.903945111492	0.902458547742	0.001381709088



When running testPenData, I expected and noticed a very sharp increase until 10 hidden layer perceptrons were used. Afterwards, accuracy just seemed to flat line. There was some variation as more perceptrons were used, but it was consistently hovering at about 90% accuracy. I believe its accuracy had simply reached its asymptote, but I was surprised to see that happen so quickly at only 10 perceptrons.

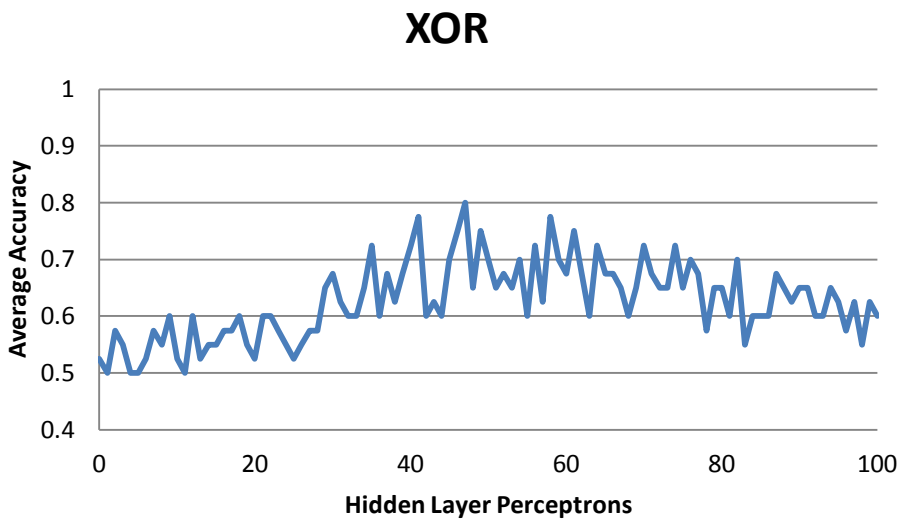
testCarData

Hidden Layers Perceptrons	Max Accuracy	Average Accuracy	Standard Deviation
0	0.791884816754	0.790837696335	0.000523560209
5	0.861910994764	0.856282722513	0.006681805769
10	0.863874345550	0.859554973822	0.004333231769
15	0.861910994764	0.859816753927	0.001125958805
20	0.858638743455	0.855759162304	0.003274868064
25	0.863874345550	0.856675392670	0.005553194617
30	0.861910994764	0.853141361257	0.005608451645
35	0.857329842932	0.850523560209	0.004675542173
40	0.852748691099	0.850000000000	0.002282146044



Running testCarData gave slightly different results. For some reason though, 10 to 15 hidden layer perceptrons gave the best results while additional perceptrons seemed to cause accuracy to decrease. Since it's so slow of decrease, it could just be an anomaly from my test results. However, the decrease in accuracy is very consistent as more perceptrons are used. This result surprised me in that I expect most neural networks to be more reliable as more perceptrons are used.

Question 7



I tested my XOR data set with 0 to 100 hidden layer perceptors, running 10 iterations each time. The graph above plots the average accuracy of the 10 iterations at each number of hidden layer perceptors. I had expected the average accuracy to steadily increase as more perceptors were used, but my results seem to show a maximum at about 45 perceptors. More surprisingly, accuracy appears to be decreasing as more perceptors are used beyond that point. Unfortunately, I expected much more consistent success than I had observed. I am unsure whether or not these results were to be expected or if something is wrong with my implementation of the XOR function.

Full Data:

Hidden Layers Perceptrons	Max Accuracy	Average Accuracy	Standard Deviation
0	0.75	0.525	0.134629
1	0.75	0.5	0.158114
2	0.75	0.575	0.114564
3	0.75	0.55	0.1
4	0.5	0.5	0
5	0.5	0.5	0
6	0.75	0.525	0.075
7	0.75	0.575	0.114564
8	0.75	0.55	0.1
9	1	0.6	0.165831
10	0.75	0.525	0.075
11	0.75	0.5	0.111803
12	1	0.6	0.165831
13	0.75	0.525	0.075
14	0.75	0.55	0.1
15	0.75	0.55	0.15
16	1	0.575	0.160078
17	0.75	0.575	0.114564
18	1	0.6	0.165831
19	0.75	0.55	0.1
20	0.75	0.525	0.075
21	1	0.6	0.2
22	0.75	0.6	0.122474
23	0.75	0.575	0.114564
24	0.75	0.55	0.1
25	0.75	0.525	0.134629
26	0.75	0.55	0.1
27	1	0.575	0.160078
28	0.75	0.575	0.114564
29	1	0.65	0.165831

30	1	0.675	0.195256
31	0.75	0.625	0.125
32	1	0.6	0.165831
33	0.75	0.6	0.122474
34	1	0.65	0.165831
35	1	0.725	0.207666
36	1	0.6	0.165831
37	0.75	0.675	0.114564
38	0.75	0.625	0.125
39	1	0.675	0.195256
40	1	0.725	0.134629
41	1	0.775	0.175
42	1	0.6	0.165831
43	0.75	0.625	0.125
44	0.75	0.6	0.122474
45	1	0.7	0.15
46	1	0.75	0.158114
47	1	0.8	0.15
48	0.75	0.65	0.122474
49	1	0.75	0.193649
50	0.75	0.7	0.1
51	1	0.65	0.165831
52	1	0.675	0.160078
53	1	0.65	0.2
54	1	0.7	0.187083
55	0.75	0.6	0.122474
56	0.75	0.725	0.075
57	1	0.625	0.167705
58	1	0.775	0.175
59	1	0.7	0.187083
60	1	0.675	0.160078
61	1	0.75	0.223607
62	1	0.675	0.195256
63	0.75	0.6	0.122474
64	1	0.725	0.134629
65	1	0.675	0.160078
66	1	0.675	0.160078
67	1	0.65	0.165831
68	0.75	0.6	0.122474
69	1	0.65	0.165831
70	0.75	0.725	0.075
71	0.75	0.675	0.114564
72	0.75	0.65	0.122474
73	0.75	0.65	0.122474
74	0.75	0.725	0.075
75	0.75	0.65	0.122474
76	0.75	0.7	0.1
77	0.75	0.675	0.114564
78	0.75	0.575	0.114564
79	1	0.65	0.165831
80	0.75	0.65	0.122474
81	0.75	0.6	0.122474
82	0.75	0.7	0.1
83	0.75	0.55	0.1
84	0.75	0.6	0.122474
85	0.75	0.6	0.122474
86	0.75	0.6	0.122474
87	1	0.675	0.160078
88	0.75	0.65	0.122474
89	0.75	0.625	0.125
90	0.75	0.65	0.122474
91	0.75	0.65	0.122474
92	0.75	0.6	0.122474
93	0.75	0.6	0.122474
94	0.75	0.65	0.122474
95	0.75	0.625	0.125
96	0.75	0.575	0.114564
97	0.75	0.625	0.125
98	0.75	0.55	0.1
99	0.75	0.625	0.125
100	0.75	0.6	0.122474