Manasbi Parajuli

Professor Amruth Kumar

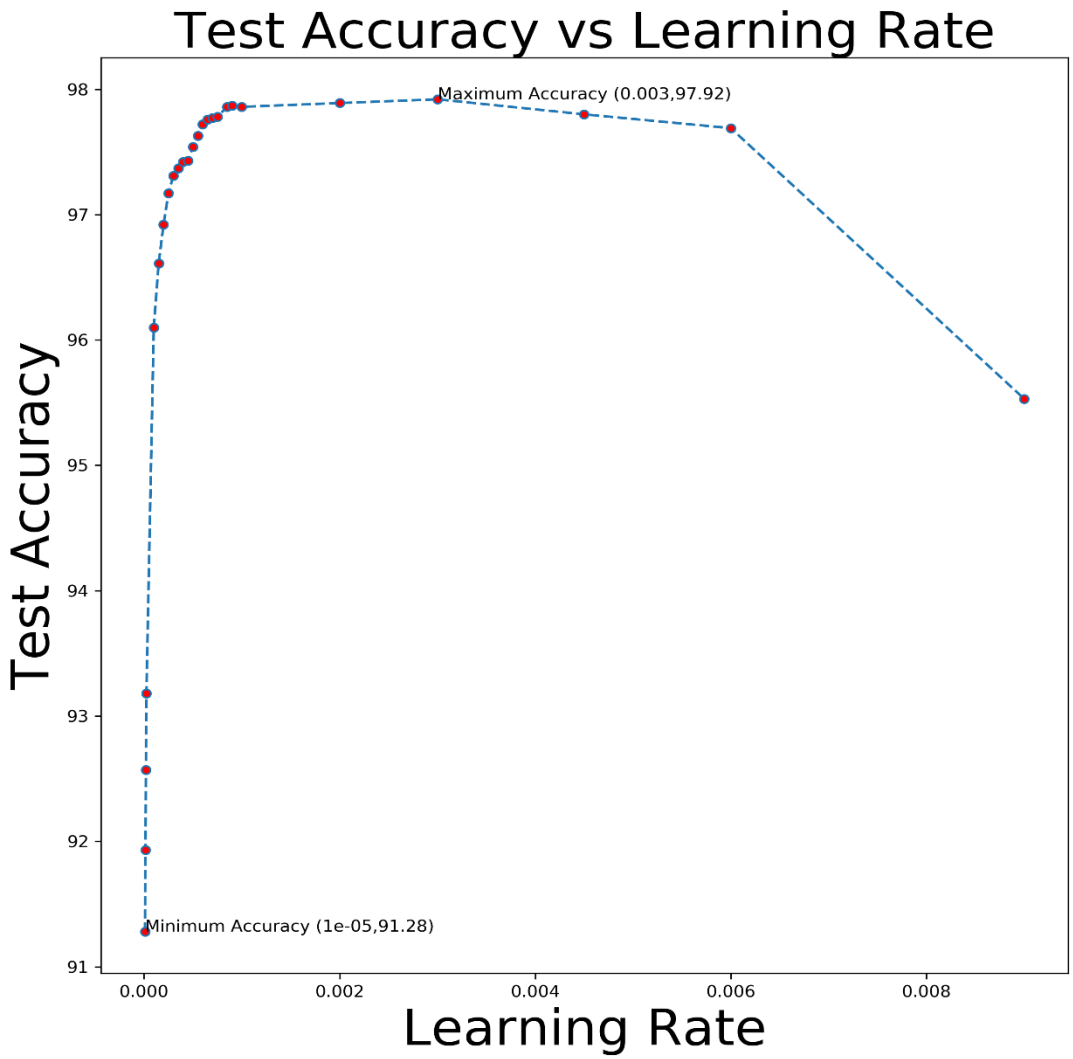
AI

4/29/2018

Project 5 Data Analysis

1) Analysis of changing learning rate in our multi-layer perceptron model

SN	Learning Rate	Train Accuracy (%)	Valid Accuracy (%)	Test Accuracy (%)
1	0.001	99.67	98.1	97.86
2	0.002	99.76	98.22	97.89
3	0.003	99.76	98.22	97.92
4	0.0045	99.6	98.36	97.80
5	0.006	99.55	98.34	97.69
6	0.009	96.85	96.24	95.53
7	0.0001	96.69	96.94	96.10
8	0.0002	98.13	97.58	96.92
9	0.0003	98.73	97.80	97.31
10	0.0004	99.06	98.02	97.42
11	0.0005	99.28	97.98	97.54
12	0.0006	99.43	98.10	97.72
13	0.00075	99.58	98.06	97.78
14	0.00015	97.57	97.38	96.61
15	0.00025	98.48	97.60	97.17
16	0.00035	98.93	97.92	97.37
17	0.00045	99.19	98.00	97.43
18	0.00055	99.36	98.08	97.63
19	0.00065	99.51	98.04	97.77
20	0.0007	99.55	98.04	97.77
21	0.00085	99.63	98.06	97.86
22	0.0009	99.65	98.04	97.87
23	0.00001	90.61	93.18	91.28
24	0.000015	91.69	93.88	91.93
25	0.00002	92.54	94.36	92.57
26	0.000025	93.14	94.66	93.18

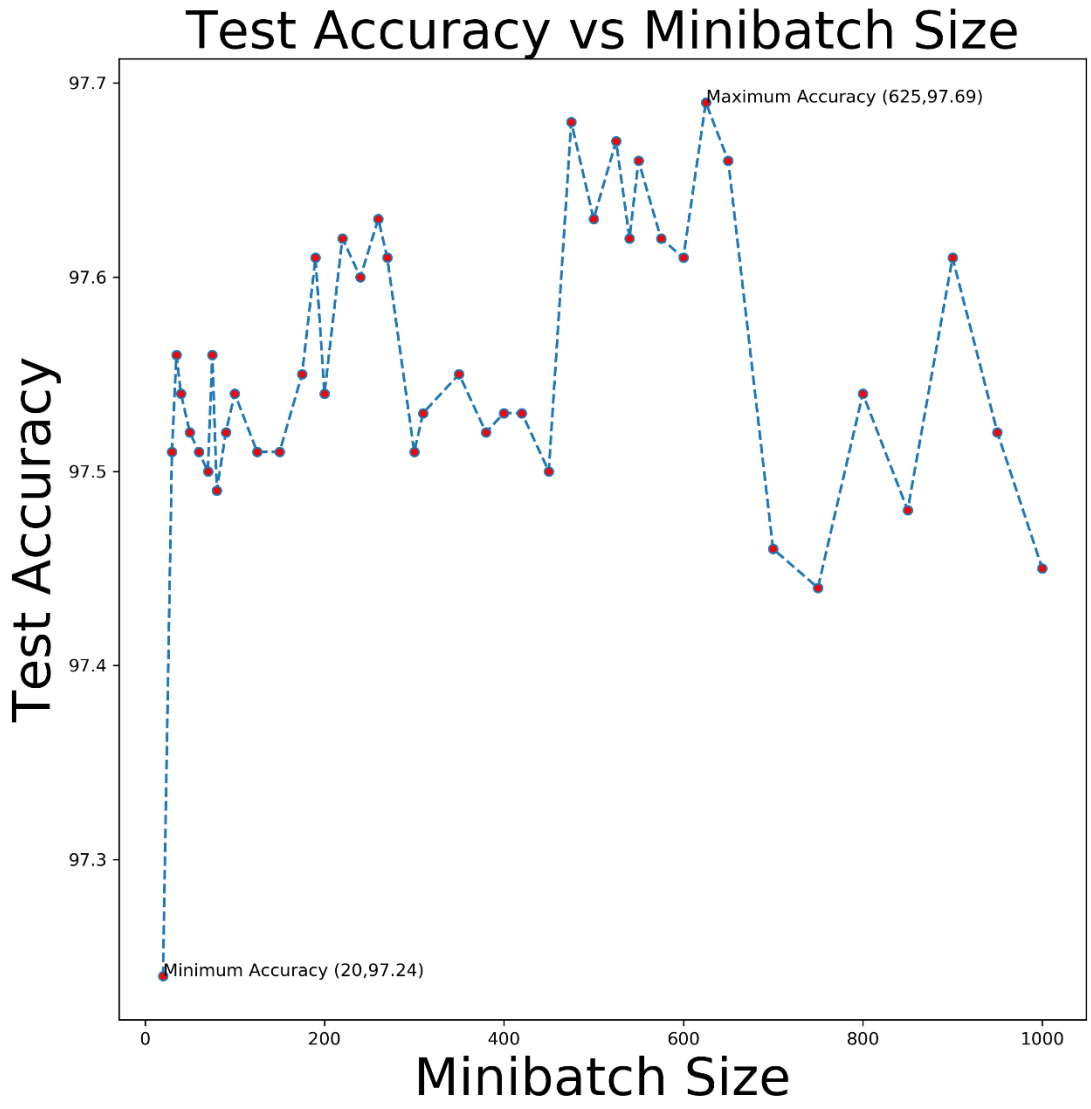


After running 26 different values of the learning rate, I found that the local maxima occurred when learning rate equals 0.003 with 97.92% accuracy. Any values of learning rate less than 0.003 and greater than 0.003 decreased the test accuracy. Also, when I set learning rate to 10^{-5}, the test accuracy was poor yielding 91.28% accuracy which suggests that this low value of learning rate is overfitting our data.

In the program, I only altered the values of the learning rate while the values for the rest of the hyperparameters stayed the same as in the original program.

2) Analysis of changing minibatch size in our multi-layer perceptron model

SN	Minibatch Size	Train Accuracy (%)	Valid Accuracy (%)	Test Accuracy (%)
1	20	98.15	97.88	97.24
2	35	98.73	98.00	97.56
3	50	98.96	98.12	97.52
4	75	99.17	98.10	97.56
5	90	99.24	98.00	97.52
6	100	99.28	97.98	97.54
7	150	99.39	98.08	97.51
8	200	99.46	97.96	97.54
9	260	99.48	98.02	97.63
10	310	99.49	97.98	97.53
11	350	99.51	97.92	97.55
12	420	99.50	97.88	97.53
13	500	99.52	98.08	97.63
14	550	99.50	97.96	97.66
15	700	99.47	97.94	97.46
16	30	98.61	97.98	97.51
17	40	98.81	98.08	97.54
18	60	99.07	98.14	97.51
19	70	99.14	98.06	97.50
20	80	99.18	98.10	97.49
21	125	99.34	98.04	97.51
22	175	99.43	97.94	97.55
23	190	99.45	98.00	97.61
24	220	99.48	97.96	97.62
25	240	99.48	98.04	97.60
26	270	99.47	98.04	97.61
27	300	99.49	98.04	97.51
28	380	99.48	97.88	97.52
29	400	99.47	97.94	97.53
30	450	99.49	97.96	97.50
31	475	99.48	97.98	97.68
32	525	99.51	98.02	97.67
33	540	99.47	97.86	97.62
34	575	99.46	98.10	97.62
35	600	99.47	98.04	97.61
36	625	99.47	98.02	97.69
37	650	99.47	98.08	97.66
38	750	99.41	97.80	97.44
39	800	99.45	97.88	97.54
40	850	99.41	97.76	97.48
41	900	99.39	97.84	97.61
42	950	99.39	97.70	97.52
43	1000	99.39	97.78	97.45

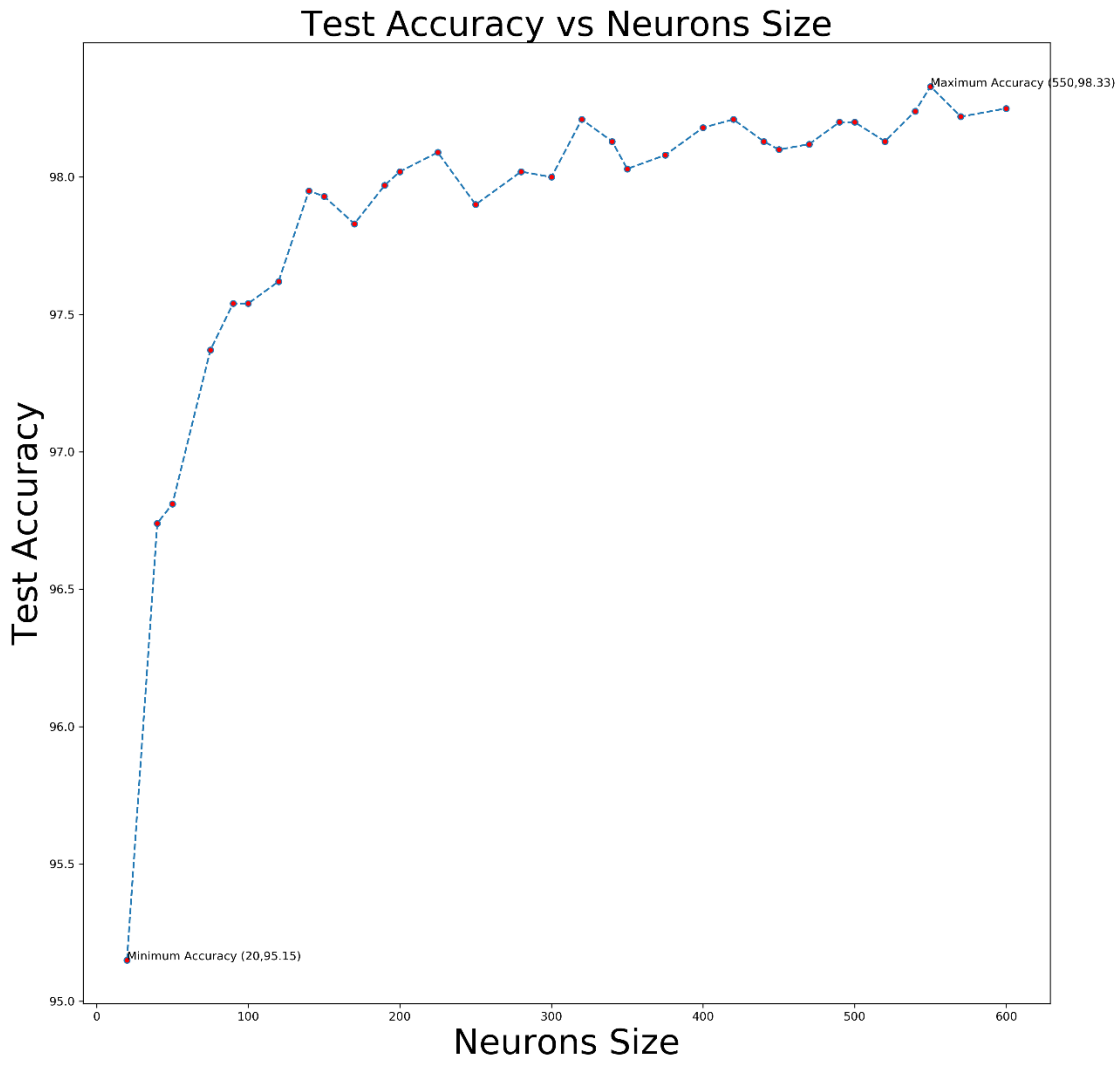


From our analysis of running 43 different values of minibatch size, the minibatch size of 625 yields the highest accuracy at 97.69%. Interestingly, all the minibatch sizes gave me similar test accuracy at 97% with only the decimal values changing. This suggests that there is less effect on our accuracy whatever the value of our minibatch sizes. We can say with 97% confidence that any value of minibatch sizes between 20 and 1000 will not affect the accuracy of our model.

In the program, I only altered the values of the minibatch size while the values for the rest of the hyperparameters stayed the same as in the original program.

3) Analysis of changing neurons size in our multi-layer perceptron model

SN	Neurons	Train Accuracy (%)	Valid Accuracy (%)	Test Accuracy (%)
1	20	96.53	96.16	95.15
2	40	98.18	97.32	96.74
3	50	98.44	97.52	96.81
4	75	99.00	97.80	97.37
5	90	99.16	97.96	97.54
6	100	99.28	97.98	97.54
7	150	99.53	98.36	97.93
8	200	99.57	98.58	98.02
9	250	99.62	98.38	97.90
10	300	99.69	98.32	98.00
11	350	99.70	98.74	98.03
12	400	99.71	98.60	98.18
13	450	99.74	98.56	98.10
14	500	99.75	98.58	98.20
15	550	99.77	98.62	98.33
16	600	99.78	98.50	98.25
17	120	99.40	98.00	97.62
18	140	99.50	98.22	97.95
19	170	99.51	98.40	97.83
20	190	99.62	98.24	97.97
21	225	99.63	98.60	98.09
22	280	99.66	98.46	98.02
23	320	99.72	98.64	98.21
24	340	99.70	98.40	98.13
25	375	99.68	98.56	98.08
26	420	99.76	98.64	98.21
27	440	99.74	98.64	98.13
28	470	99.72	98.62	98.12
29	490	99.77	98.66	98.20
30	520	99.78	98.60	98.13
31	540	99.77	98.66	98.24
32	570	99.77	98.50	98.22

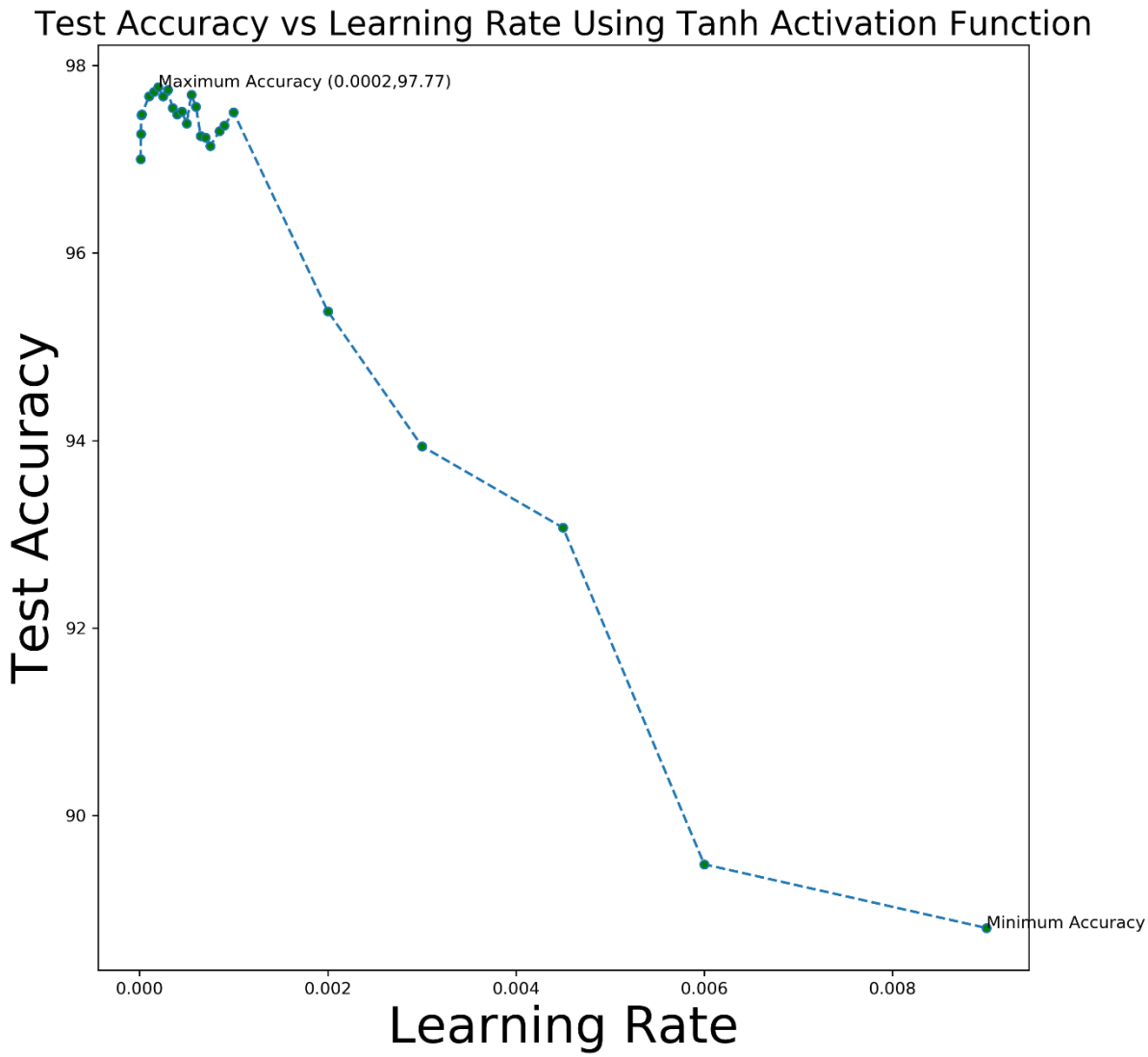


With 32 different values of neurons in the first layer, I found that we got a maximum test accuracy of 98.33% when we had 550 neurons in the first layer. This was also the local maxima as any values less than 550 as our number of neurons gave me a lower accuracy and any values greater than 550 gave a lower values of test accuracy. Also, less number of neuron sizes meant less accuracy. For example, we achieved a minimum test accuracy of 95.15% when we had 20 neurons in the first layer. Moreover, with neuron sizes of greater than 200 in the first layer, we can see clusters above 98% mark from the graph. This suggests that optimal value of neurons sizes is greater than 200.

In the program, I only altered the values of the number of neurons in the first layer while the values for the rest of the hyperparameters stayed the same as in the original program.

4) Analysis of changing our activation function to tanh function and the effect in the learning rate in our multi-layer perceptron model

SN	Learning Rate	Sigmoid Function Test Accuracy (%)	Tanh Function Test Accuracy (%)
1	0.001	97.86	97.50
2	0.002	97.89	95.38
3	0.003	97.92	93.94
4	0.0045	97.80	93.07
5	0.006	97.69	89.48
6	0.009	95.53	88.80
7	0.0001	96.10	97.67
8	0.0002	96.92	97.77
9	0.0003	97.31	97.74
10	0.0004	97.42	97.48
11	0.0005	97.54	97.38
12	0.0006	97.72	97.56
13	0.00075	97.78	97.14
14	0.00015	96.61	97.72
15	0.00025	97.17	97.67
16	0.00035	97.37	97.55
17	0.00045	97.43	97.51
18	0.00055	97.63	97.69
19	0.00065	97.77	97.25
20	0.0007	97.77	97.23
21	0.00085	97.86	97.30
22	0.0009	97.87	97.36
23	0.00001	91.28	97.00
24	0.000015	91.93	97.27
25	0.00002	92.57	97.47
26	0.000025	93.18	97.48



To analyze the accuracy in the learning rate, I fed the model with two different activation functions: sigmoid/logistic and tanh function. I used scikit's MLPClassifier to calculate the accuracy score of my model when using tanh function. Interestingly, I obtained maximum test accuracy when the learning rate was 0.0002 with 97.77% using tanh function as compared to 0.003 with 97.92% using sigmoid function. Similarly, we get the minimum accuracy 88.80% when our learning rate was 0.009 which contrasts with what we got using sigmoid function as the learning rate of 10^{-5} had yielded the minimum test accuracy of 91.28%. In other words, there is a sharp decline in the curve towards the right side of our graph when we used tanh function rather than the observable left-hand side in our sigmoid curve.

In the program, I only altered the values of the learning rate and used tanh as the activation function while the values for the rest of the hyperparameters stayed the same as in the original program.

5) Analysis of changing the topology from one to two hidden layers in our multi-layer perceptron model

SN	First Hidden Layer	Second Hidden Layer	Test Accuracy (%)
1	100	50	97.55
2	150	100	97.20
3	200	150	97.76
4	250	200	97.11
5	300	250	97.70
6	350	300	97.00
7	400	350	97.67
8	450	400	97.44
9	500	450	97.94
10	550	500	97.02

With just 10 different values as the number of neurons in our two hidden layers, we observe that the test accuracy fluctuated around the 97% mark. We achieved the maximum test accuracy of 97.94% when we had 500 neurons in the first hidden layer and 450 neurons in the second hidden layer. I had used scikit's MLPClassifier to set the number of neurons in the hidden layers.