Neural Networks:

1. Gradient Descent with Momentum (MC) and Learning Rate (LR):
   1. Character Data Set:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Layer Size(s) | LR | MC | Epochs | Perf | Train perf | Val perf | Test perf | Time | function | trainfncn |
| 1 | 40-26-10 | 0.1 | 0.9 | 10000 | 0.0361 | 0.0361 | 0.0361 | 0.036 | 08:04 | fitnet | traingdm |
| 2 | 40-26-10 | 0.1 | 0.9 | 1470 | 0.269 | N/A | N/A | 0.2683 | 02:41 | patternet | traingdm |
| 3\* | 20-20 | 0.1 | 0.9 | 1699 | 0.269 | 0.269 | 0.269 | 0.269 | 01:40 | patternet | traingdm |
| 4 | 20-20 | 0.1 | 0.5 | 1699 | 0.269 | 0.269 | 0.269 | 0.269 | 01:40 | Patternet | traingdm |
| 5 | 20-20 | 0.1 | 0.0 | 1699 | 0.269 | 0.269 | 0.269 | 0.269 | 01:50 | Patternet | traingdm |
| 6 | 20-20 | 0.1 | 0.5 | 1699 | 0.0375 | 0.0375 | 0.0375 | 0.0375 | 00:58 | fitnet | traingdm |
| 7 | 20-20 | 0.1 | 0.9 | 1699 | 0.0372 | 0.0372 | 0.0372 | 0.0371 | 00:58 | fitnet | traingdm |
| 8 | 20-20 | 0.4 | 0.9 | 1699 | 0.0365 | 0.0365 | 0.0365 | 0.0365 | 01:10 | fitnet | traingdm |
| 9 | 20-20 | 0.7 | 0.9 | 7 (stopped auto) | 0.486 | 0.486 | 0.488 | 0.485 | 00:01 | fitnet | traingdm |
| 10 | 20-20 | 0.5 | 0.9 | 1699 | 0.0363 | 0.0363 | 0.0363 | 0.0363 | 01:08 | fitnet | traingdm |
| 11 | 20-20 | 0.01 | 0.9 | 1699 | 0.0544 | 0.0544 | 0.0546 | 0.0542 | 00:56 | fitnet | traingdm |
| 12 | 20-20 | 0.9 | 0.0 | 1699 | 0.0357 | 0.0357 | 0.0356 | 0.0357 | 00:58 | Fitnet | traingdm |
| 13 | 20 | 0.9 | 0.0 | 1699 | 0.0343 | 0.0343 | 0.0342 | 0.0342 | 00:40 | fitnet | traingdm |
| 14 | 20 | 0.01 | 0.9 | 1699 | 0.121 | 0.121 | 0.119 | 0.120 | 00:44 | Fitnet | traingdm |
| 15 | 20 | 0.1 | 0.9 | 1699 | 0.0418 | 0.0418 | 0.0421 | 0.0417 | 00:49 | Fitnet | traingdm |
| 16 | 20 | 0.5 | 0.5 | 1699 | 0.0353 | 0.0353 | 0.0354 | 0.353 | 00:50 | Fitnet | traingdm |
| 17 | 20 | 0.1 | 0.9 | 1699 | 0.2689 | 0.2689 | 0.2689 | 0.2689 | 01:46 | patternet | traingdm |

It can be seen above that 3 layers vs 2 layers does offer much improvement in performance. Which was mentioned in Mitchell Ch. 6 that most nets with 2 hidden layers offer quite good performance. Where having more than 3 is not usually the case. This is expected, since the data set produced good results in other simpler algorithms.

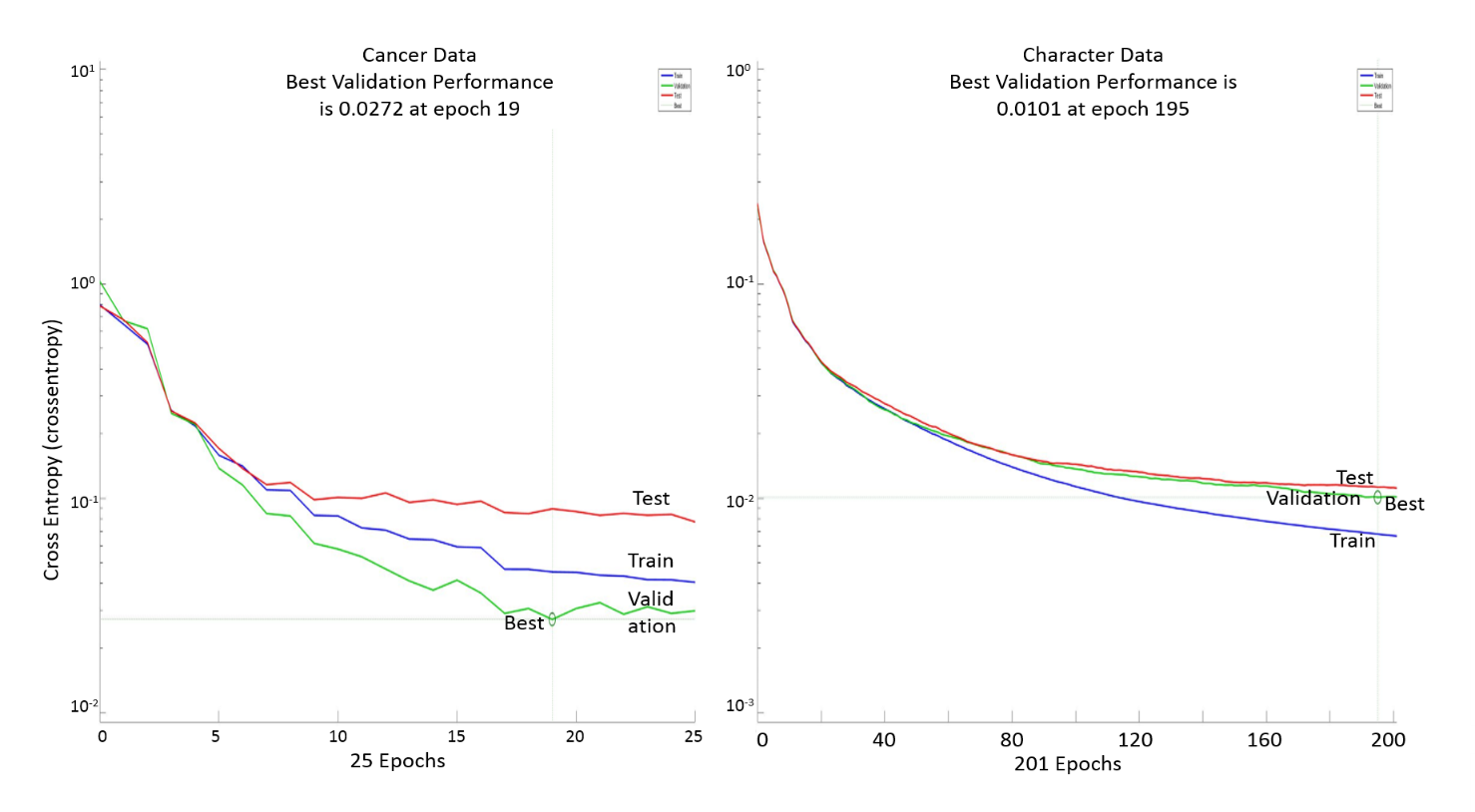
In run 3, even after 10000 epochs without overfitting the CV set, the performance is around 0.268 which is not worth the additional iterations. Moreover, it was noticed that the run time is even higher than the 3 layered network as shown in number 1.

When removing momentum, It was noted that removing the momentum did not have much effect on the performance of the data set. However, it decreased the runtime by around 10 seconds while getting the same results. The main purpose for it, is to allow the network to escape from a local minima. The value of 0.5 had

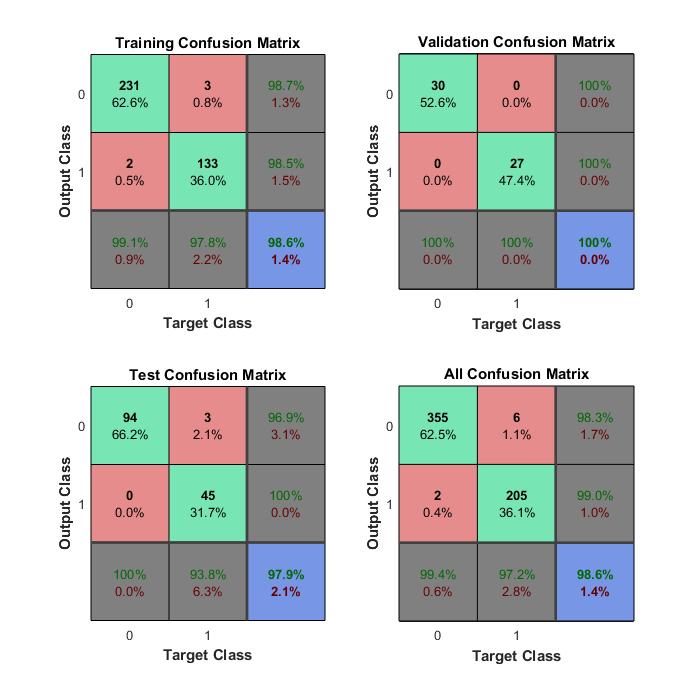
It was noticed that as I increased the learning rate, the performance increased at the fixed epochs of 1699 but it also meant that it might have overshoot where I set the validation check to 6. Such as the case with 9. When I ran that again I got a performance of 0.360 after going through the 1699 epochs. This was made clear when I increased the learning rate to 0.9, I got a score of 0.0356 with a time of 01:01. To overcome this, one can use an adaptive learning rate that is tuned to try using a step that quickens reaching a local optimum such as a L-BFGS algorithm [http://web.stanford.edu/class/cs294a/sparseAutoencoder\_2011new.pdf].

To see how the effect of the number size of hidden layers affects the performance, I decided to use the ‘traingm’ function while changing the Size of hidden layers. Training automatically stops when generalization stops improving, as indicated by an increase in the cross-entropy error of the validation samples which was used as performance metric along the %Error.

|  |  |  |
| --- | --- | --- |
| Size of Hidden Layer | % Error on Cancer Test, validation and testing Data | % Error on Character Test, validation and **testing** Data |
| 2 | 2.71, 1.75, 2.81 (21 Epochs) | 67.98, 68.30, **67.43** (58 Epochs) |
| 8 | 2.16, 0.80, 4.93 (33 Epochs) | 24.04, 24.69, **25.61**(197 Epochs) |
| 12 | 1.35, 0.00, 2.11 (26 Epochs) | 17.13, 16.95, **18.94** (269 Epochs) |
| 25 | 2.17, 0.00, 4.22 (29 Epochs) | 08.03, 10.00, **10.44** (229 Epochs) |
| 39 | 2.71, 10.5, 3.52 (11 Epochs) | 06.51, 7.14, **9.24** (157 Epochs) |
| 50 | 2.17, 0.00, 3.52 (20 Epochs) | 03.89, 7.199, **8.58** (151 Epochs) |
| 70 | 2.71, 0.00, 1.41 (23 Epochs) | 02.04, 07.45, **6.94** (141 Epochs) |
| 85 | 5.42, 1.75, 3.52 (25 Epochs) | 03.23, 07.05, **6.80** (124 Epochs) |
| 100 | N/A | 02.39, 06.09, **6.879** (128 Epochs) |
| 150 | N/A | 1.046, 04.75, **5.48** (172 Epochs) |
| 175 | N/A | 01.73, 05.35, **6.68** (149 Epochs) |
| 200 | 1.89, 0.00, 2.11 (20 Epochs) | 04.83, 07.00, **7.75** (125 Epochs) |
| 300 | N/A | 10.83, 13.15, **13.23** (139 Epoch) |



Best Character @ 39, and best Cancer @ 12

Add cs231n.github.io/neural-networks-1/#arch

Add degree of freedom/noise/ data might be clustered in classes and that is why high number of hidden layers is getting good results