Edward Venator

EECS 484 Assignment 4b: Feedforward Neural Networks with Error Back Propagation

Basic Methodology

For this assignment, I used the MATLAB neural network toolbox to create a two-layer neural network. The first (hidden) layer contains ten neurons and had tansig transfer functions. The second layer used linear transfer functions. I used the default training option (Levenberg-Marquardt) to train this network, using all of the training data (no validation data). For a more detailed discussion of feedforward neural networks and the MATLAB neural network toolbox, see my report for assignment 4a.

I created and trained two different networks. One maps actuator positions to an x-y coordinate pair. The other maps actuator positions onto only an x coordinate. The networks were identical in all other parameters. For each network, I performed ten trials and compared the running time and performance.

Findings and Conclusions

Mapping Onto X-Y Coordinate Pairs

I began by testing the network's ability to map from actuator positions onto X-Y coordinate pairs. I found that the network performed admirably. MATLAB continues to train the network until one of two criteria are met. The first is a time limit; it will not train for more than 1000 iterations of the training patter. This can be adjusted, but I found 1000 to be sufficient. The second criterion is gradient, and is a measure of convergence. If the change in weights is less than 1×10^{-5}, MATLAB ends training.

The network consistently fit to the input data with less than 5×10^{-5} total mean squared error. I consider this a very good fit, about .02% deviation from the target values. Of ten trials performed, only three ran until the maximum of 1000 iterations was reached, and the average training was 548 iterations. A visual inspection of the mapping is shown in figure 1, demonstrating that the fit is indeed excellent. I would not use these results for extrapolation outside of the cluster of known points, but it would be excellent for interpolation between the points.

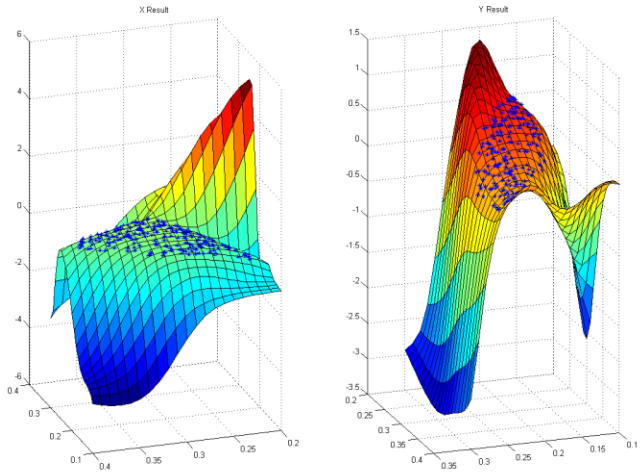

Figure 1: Typical Results from Mapping Actuator Positions Onto X-Y Coordinate Pairs

Mapping Onto an X Coordinate

When only mapping onto one coordinate, I found that the performance was actually marginally worse. The average mean squared error increased from 2.27×10^{-5} to 4.29×10^{-5}, nearly double. However, this is still a very low mean squared error; figure 2 shows that the results are still excellent. This network did train in about half the time. Only one trial ran for the full 1000 iterations, and the mean was 256 iterations. Given the rather quick training times and excellent fits for both networks, I must conclude that one- and two-output neural networks are both acceptable solutions in this situation. One advantage of only mapping onto X is that the X surface appears to me more useful for extrapolation then the XY surface; both results appear to be more or less equally useful for interpolation.

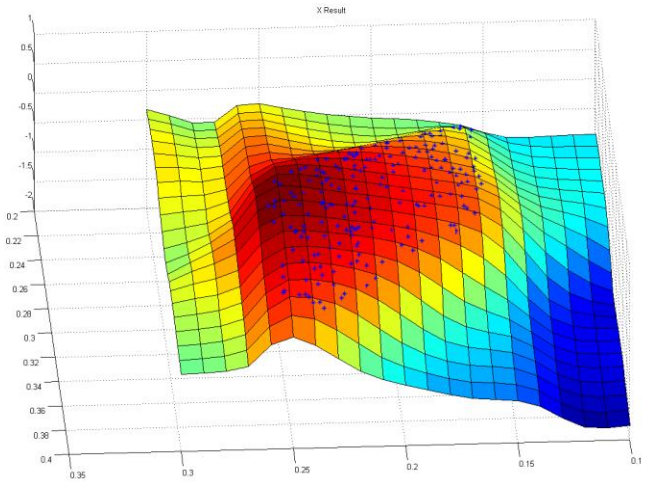

Figure 2: Typical Results from Mapping Actuator Positions Onto X Coordinate

Appendix: Raw Data From Trials

X Only				
Trial	**Epoch**	**Time**	**Performance**	**Gradient**
1	1000	16	1.22E-05	1.35E-04
2	208	3	2.31E-05	9.48E-06
3	290	4	5.02E-06	9.96E-06
4	139	2	2.08E-05	9.80E-06
5	57	0	1.66E-05	9.94E-06
6	81	1	1.78E-05	1.00E-05
7	300	4	5.57E-05	9.89E-06
8	80	1	1.02E-05	9.56E-06
9	251	3	5.67E-06	9.92E-06
10	155	2	2.62E-04	9.01E-06
Mean	256.1	3.6	4.29E-05	2.23E-05
X and Y				
Trial	**Epoch**	**Time**	**Performance**	**Gradient**
1	124	1	1.98E-05	9.88E-06
2	159	2	1.64E-05	9.89E-06
3	96	1	2.73E-05	9.76E-06
4	1000	12	1.71E-05	7.29E-05
5	582	7	1.50E-05	9.98E-06
6	840	10	3.79E-05	5.21E-06
7	599	7	1.82E-05	9.97E-06
8	1000	12	1.42E-05	5.15E-05
9	1000	12	3.83E-05	2.90E-03
10	76	1	2.28E-05	9.63E-06
Mean	547.6	6.5	2.27E-05	3.09E-04