Machine Learning (course 1DT071) Uppsala University – Spring 2015 Report for Assignment 1 by group 6

Ludvig Sundström and John Shaw

February 9, 2015

1 Task1

For the first task we were asked to create a feed forward neural network (NN) and train it to implement the boolean XOR function. Using tools built into Matlab, we could experiment with different networks and observe the training process. We don't know how the trained neural network finds its solution, but in order to understand how to train the network we somehow need to visualize the training process. Using gradient descent training, we can think of the training process for each problem as finding the lowest point in a landscape. If there exists a global minimum point in this landscape, we want the training algorithm to find it. There can exist local minima though and we don't want the algorithm to get stuck in one, so we want to adjust the parameters of the descent in such a way that we don't get stuck in a local minimum. After a session of trial and error, we found that a learning rate of 2 usually resulted in a well-trained network.

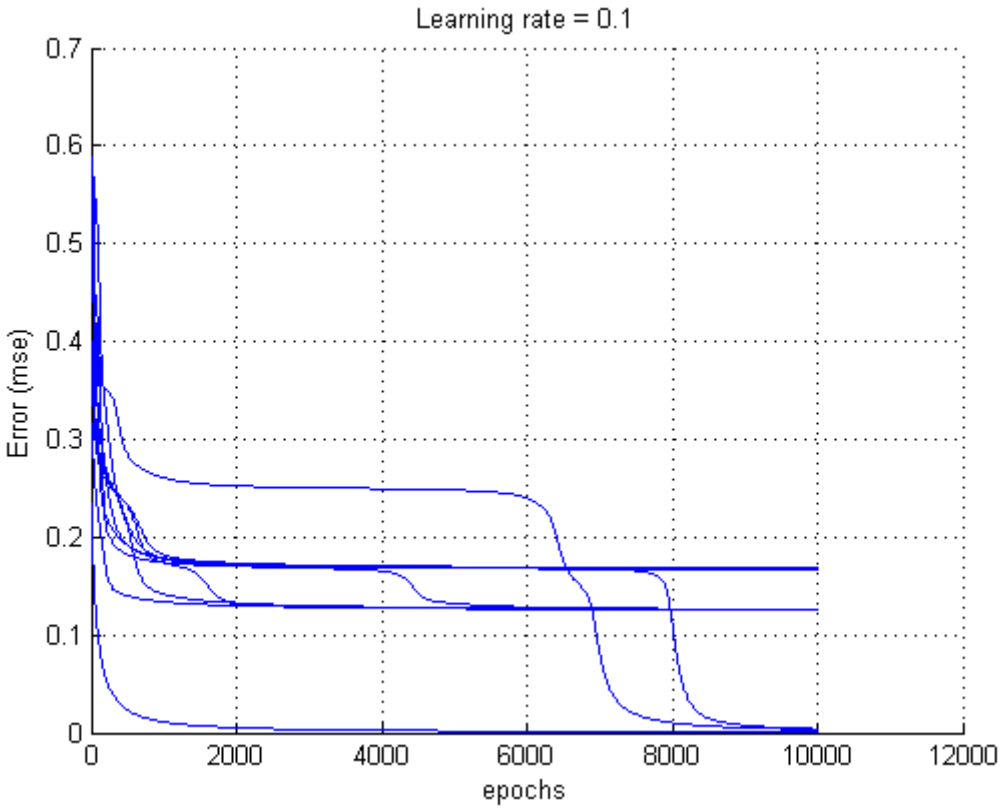
Question 1 & 2

Question 1 & 2 Sometimes, as shown in for example figure {1}, the error plot converges to an value significantly larger than zero. This is because the error rate is too low and we get stuck in local minima. We are taking too small steps, and can't get out of small cavities. A too large learning rate however, will cause the algorithm to take giant leaps through the landscape. Too big steps will make it very hard to find a good solution, and an oscillating behaviour is likely to occur; see figure {3}.

A visualisation of a well trained network for the xor problem is to plot the result of various values as colors, as seen in figure {4}. Here, we can see the classification of the possible input values. We can clearly see the two hyperplanes generated that seperates data.

Here, we also see the hidden node values of possible inputs of the XOR function. Inspecting the plot ranges of the hidden nodes aswell as the output node, we notice that the range for the hidden nodes are [-1, 1] and the range for the output function is [0, 1].

Figure 1: **Plot 1** from 10 Training sessions with a learning rate of 0.1.



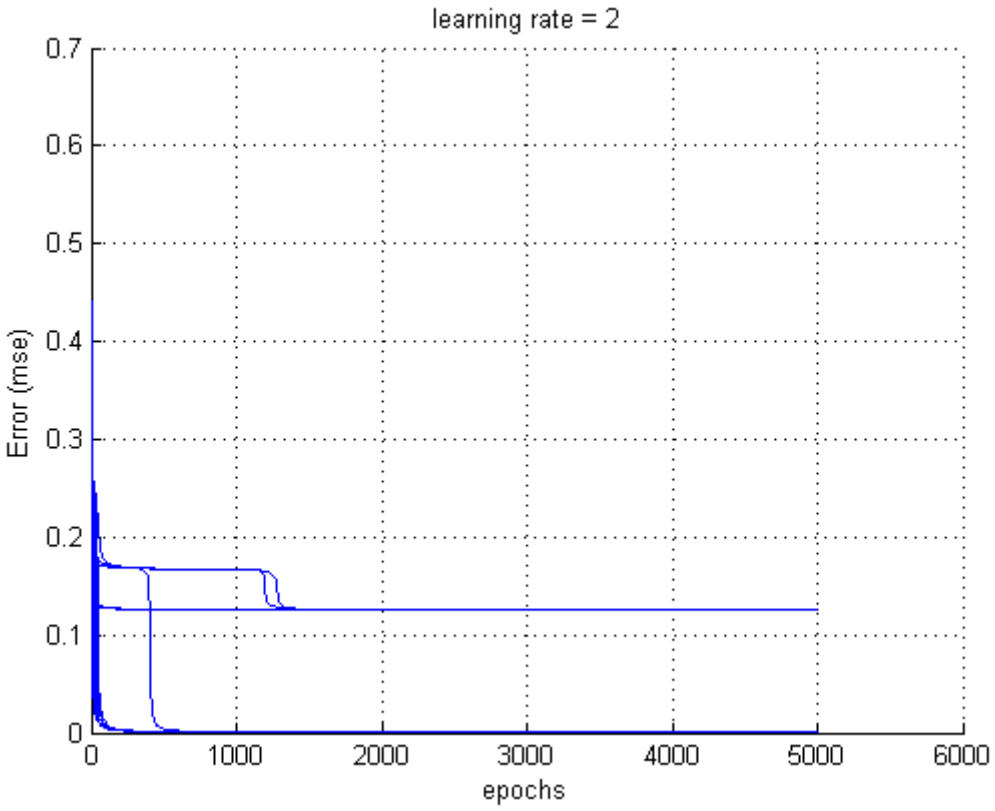
Question 3

Question 3 The reason that the ranges differs is that two different sigmoid functions were used as activation functions used for the network. For the hidden layer, the hyberbolic tangent function (range [-1, 1]) were used. For the output node, the logistic function (range [0, 1]) were used. The authors of this report could not find out weather it makes a difference for the algorithm weather you use a activation function that ranges from [0, 1] or [-1, 1] but it's obviously convinient if the output activation function has the same range as the actual problem output.

Question 4

Question 4A new neural network gets distrubited random weights when initialized. The result of this is that the training ends with different solutions every time the network is initialized and trained and the number of epochs needed to train the network differs aswell.

Figure 2: **Plot 1** from 10 Training sessions with a learning rate of 2.



Question 5

Question 5 Again consider the colored plot of the XOR problem network nodes (figure 4). We already know that this represents a well trained network which means that the output node is infact representing the boolean XOR function. As for the hidden nodes, study the value of the different inputs. For hidden node 1, an input of (0, 0) corresponds to 0 (or atleast a value very close to zero). $(0, 1)$ also yeilds 0, $(1, 0) = 0$ and $(1, 1) = 1$. If this node implemented any boolean function it would be the AND function. In the same way, the hidden node implements the boolean OR function.

Question 6

Question 6 This is veryfied in the truth table below.

Input	boolean XOR	Network implemented XOR
0 0	0	TODO
0 1	1	TODO
1 0	1	TODO
1 1	0	TODO

Figure 3: **Plot 1** from 10 Training sessions with a learning rate of 20.

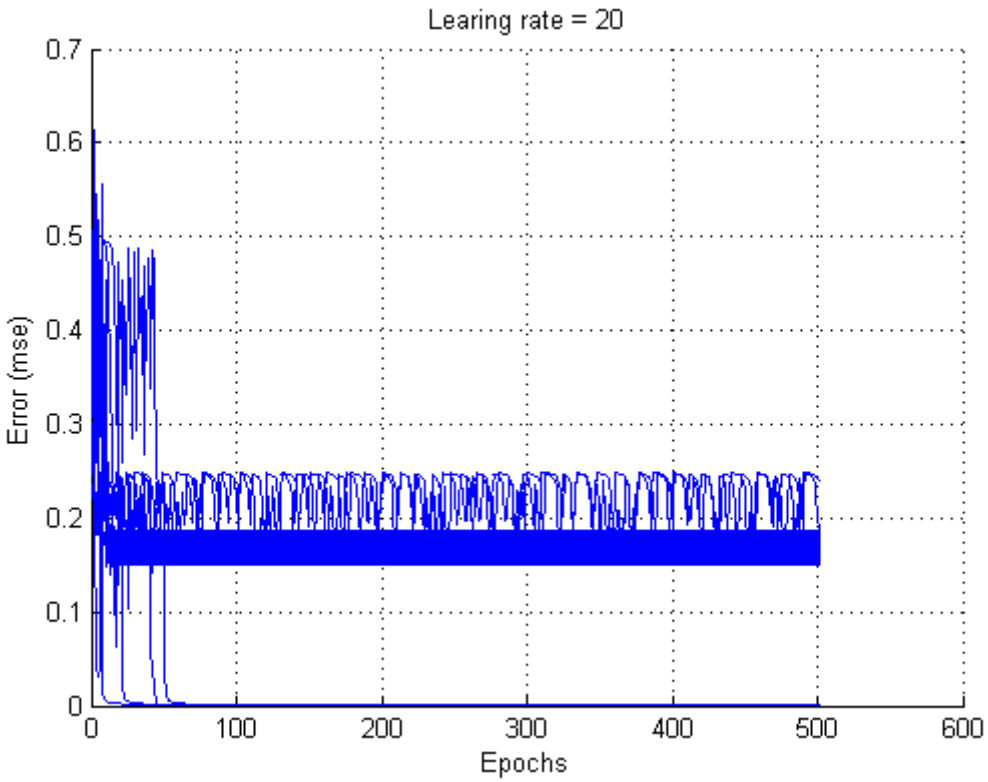
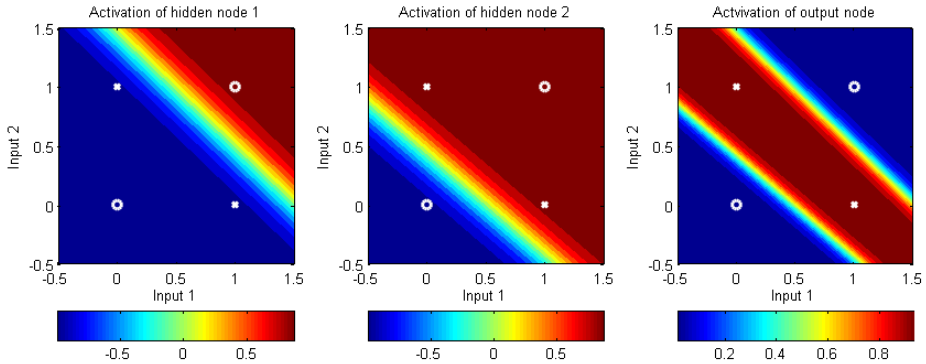


Figure 4: **Plot 2** A good solution found for the XOR problem using back-propagation



Question 7

The biggest difference is the difference in number of epoch needed. Resilient back propagation needs a lot fewer epoch to reach a minima. We think that

Figure 5: **Plot 2** A bad solution found for the XOR problem using back-propagation

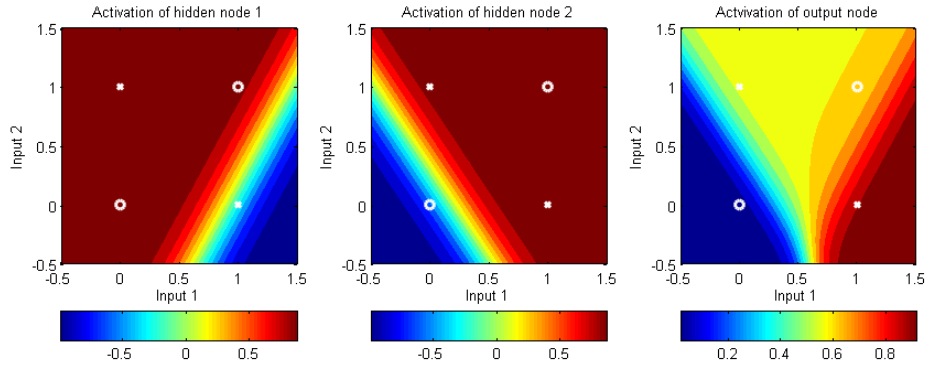
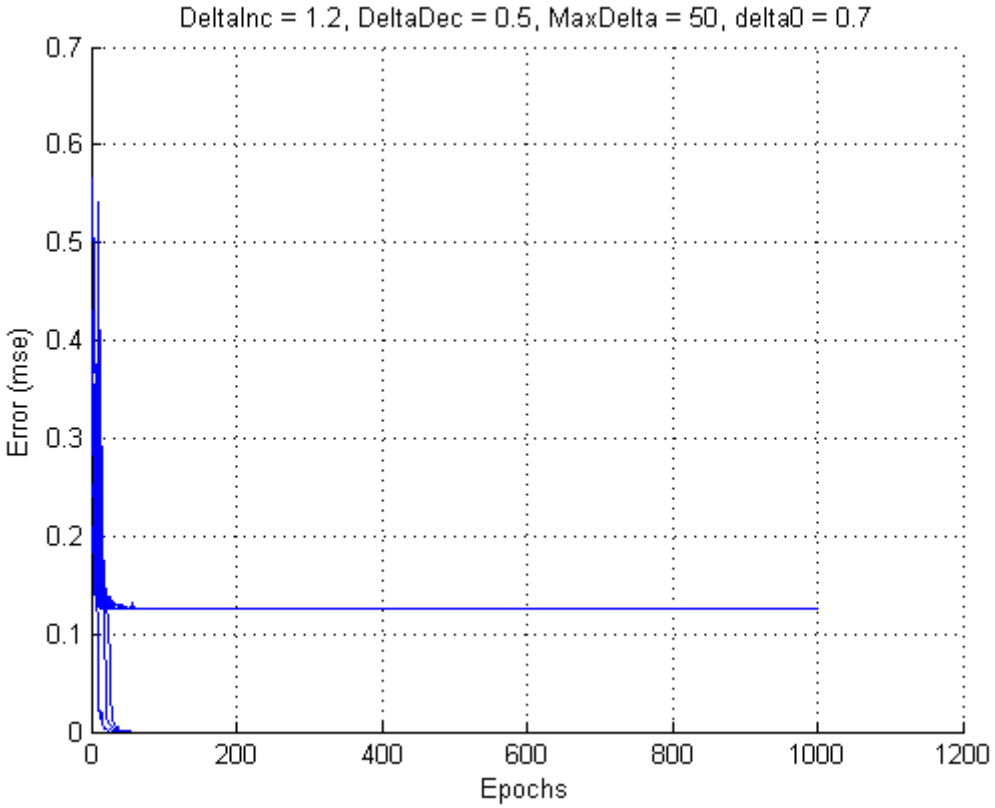


Figure 6: **Plot 3** Using the Resilient Back-propagation algorithm with the variables: $DeltaInc = 1.2, DeltaDec = 0.5, MaxDelta = 50, delta0 = 0.7$



resilient back propagation is better suitable for this problem since it finds a

minimum in about 400 times less epochs compare to the normal back propagation. A interesting observation from using plot_xor is that the way rprop have classified the groups is a mirrored image of the bprop.

2 Task2: Simple function approximation

Question 8

Question 9

Question 10

Question 11

Question 12

Question 13

3 Task3: Classification of wine data

Question 14

Question 15

Question 16

4 Task4: Approximating house prices

Question 17

Question 18

Question 19

5 Task 5: Wrapping up

Question 20

6 Appendix

References

[1] T.H Cormen, C. E. Leiserson, R. L. Rivest, C. Stein *Introduction to Algorithms*, 3rd edition 2009, p. 708-722, 658-659

[2] `http://en.wikipedia.org/wiki/Dijkstra's_algorithm`

[3] http://en.wikipedia.org/wiki/Priority_queue

[4] http://www.geeksforgeeks.org/bipartite-graph/

[5] http://www.personal.kent.edu/~rmuhamma/Algorithms/
MyAlgorithms/GraphAlgor/breadthSearch.htm

Ideas, and general pointers how to solve the various problems have been extracted from these resources. Other then that, we declare that the content of this report is entirely pure, coming purely and directly from the it's authors brains.