Artificial neural networks - Lab 1

January 22, 2013

1 One layered neural network with delta rule

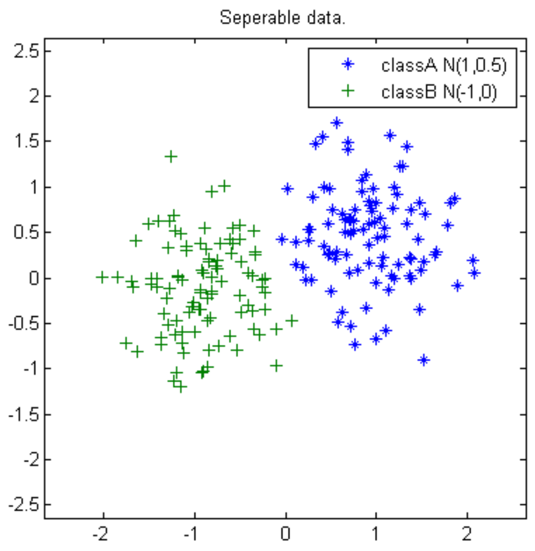
For the one layered network we used the delta rule which minimizes the total error for the classifier, although not in the sense that it minimizes the number of incorrectly classified instances, but that the error/distance from the separating line to the instances becomes minimum. Below you can see the matrix transformations for the

Figure 1: The one layer neural network layout and it's matrix transformation.

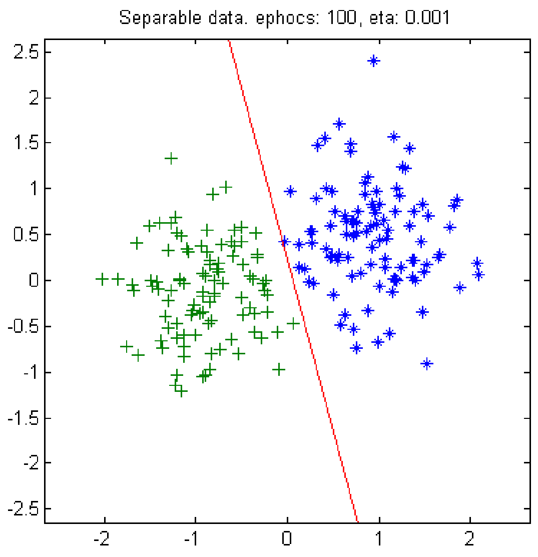
In figure 2(a) you can see the linearly seperable data generated to test the network and in figure 2(b) you can see the separating line that the neural network finds. In figure 2(d) we can also see the way the seperating line converges to it's final values. The steps to adjust the weight matrix decrease as the neural network converges as can be seen by the decreasing angle/distance between the seperating lines.

When we modify the learning rate for the network we find that smaller learning rate means that the network takes longer to converge which inturn means that we will need more epochs before it converges. A larger learning rate has it's own problems though. A too large learning rate will make the weight matrix contain too large numbers for Matlab to be able to handle it. A slightly larger learning rate may however cause the seperating line to oscillate although it still finds the minimum.

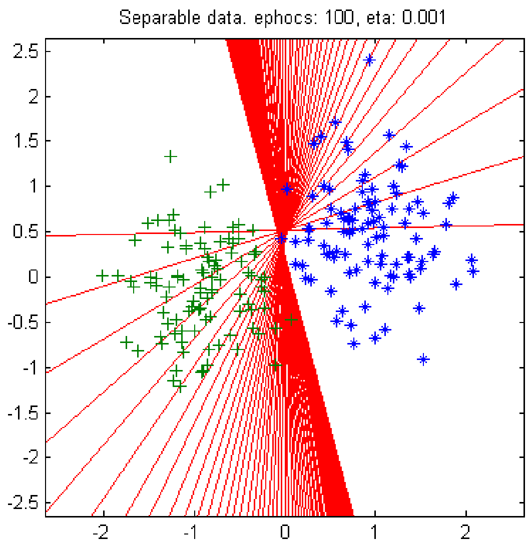
Next we look at nonseperable data to see what happens. It can be seen in figure 4(a) and as we can see it doesn't really manage to seperate the data even though it has minimized it's error. In 4(b) we can see how it converged to it's final value.



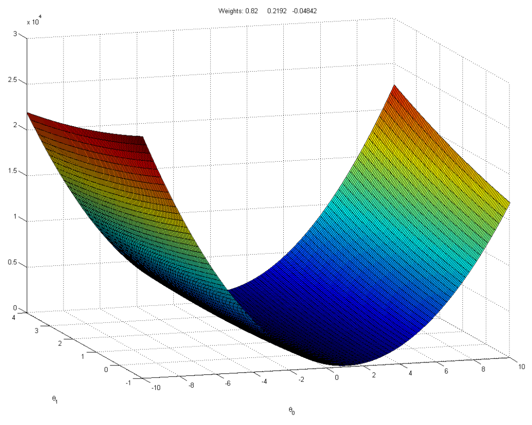
(a) Seperable data



(b) Seperaing line between data

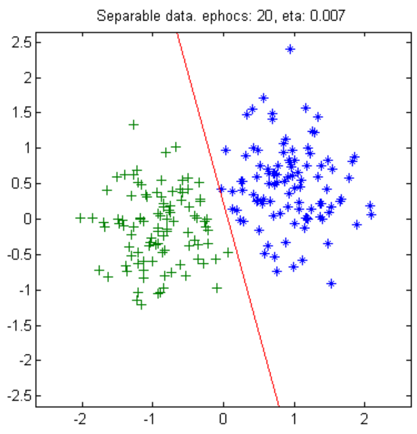


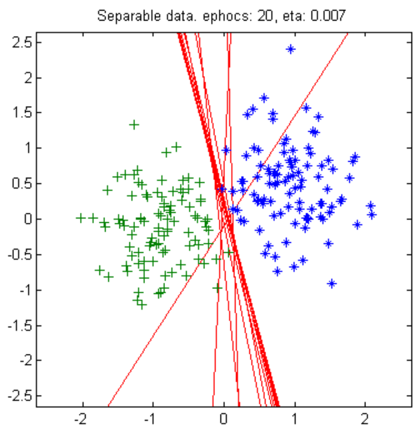
(c) The movement of the seperating line to it's final position.



(d) The error function we're minmizing with the delta rule.

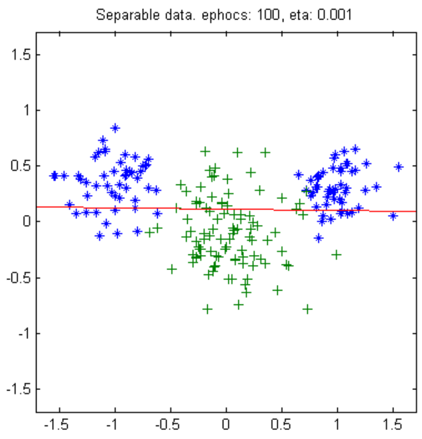
Figure 2: One layer neural network tests runs

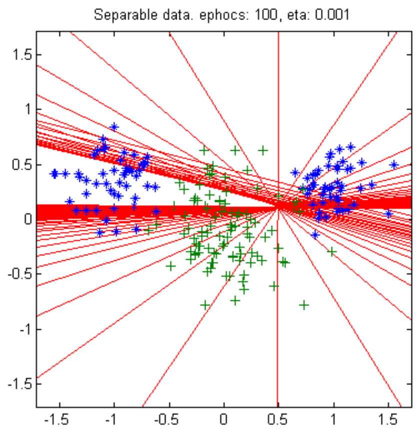




(a) The movement of the seperating line to it's final position. (b) The error function we're minmizing with the delta rule.

Figure 3: Larger eta values cause the weights to oscillate





(a) The movement of the seperating line to it's final position. (b) The error function we're minmizing with the delta rule.

Figure 4: With non-seperable data the weights still converge but we don't get a good classifer.

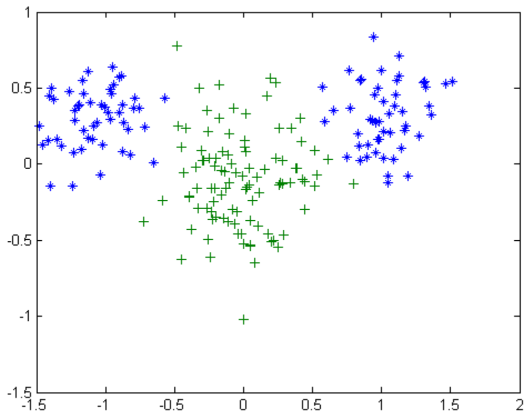
2 Two layered neural network with backprop

Next we do the backprop algorithm for a two layer network network. The layout of the two layer network can be seen in figure 2.

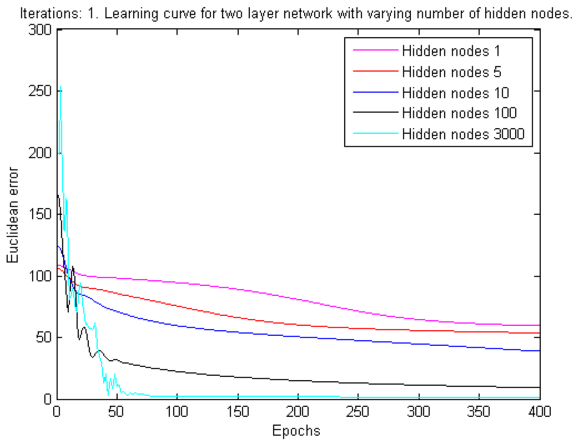
Figure 5: The two layer neural network layout

Next we check how many nodes we need to seperate non linearly seperable data with the two layer network. The results for the euclidean error and the error of missclassified instances can be seen in figure 6(b) and figure 6(c). We can see that as the number of nodes increase both error types are decreasing. It's however really difficult to actually classify all the nodes correctly and we need about 3000 nodes to classify all nodes correctly most of the time.

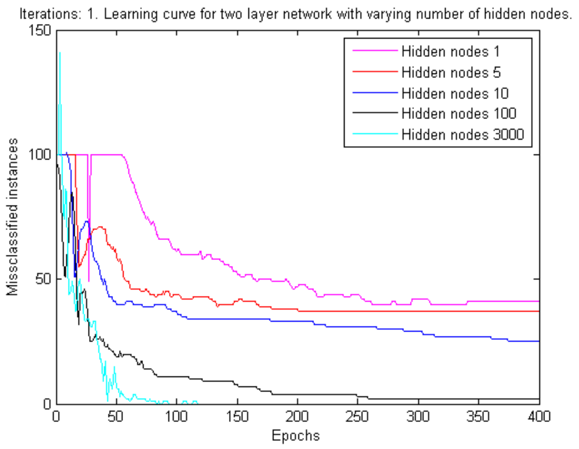
Next we see how the learning rate and momentum impacts the two layer networks learning curve. We see that a highereta means a faster convergance and we also see that a too high or too low momentum makes the weights unstable and causes them to oscillate.



(a) Non seperable data to run twolayer network on.

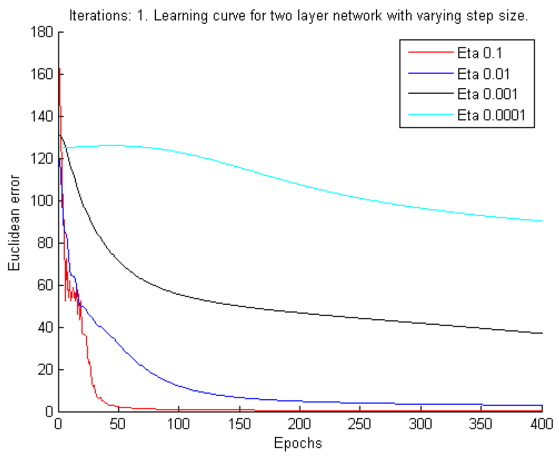


(b) Eucledian error for different amount of nodes.

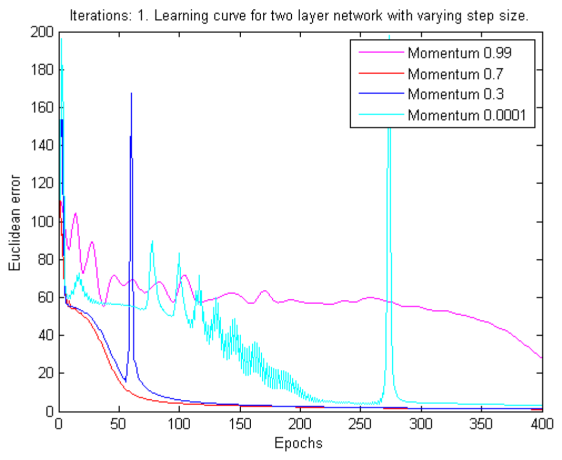


(c) Missclassified instances for different amount of nodes.

Figure 6: Two layer nodes variation



(a) Eta change for two layer network.



(b) Momentum affect on the two layer network.

Figure 7: With non-seperable data the weights still converge but we don't get a good classifer.

3 Two layered neural network used for compression

Now we will show how two layered networks can be used for datacompression. In figure 3 you can see how the neural network looks for compression. For compression the output equals the input, i.e. $O = X$. In figure 3 we can see that WX, the compressed version of X, can be binary coded with three nodes since we have eight patterns.

Figure 8: The two layer neural network layout

O =

$$\begin{array}{rrrrrrrr}
1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1
\end{array}$$

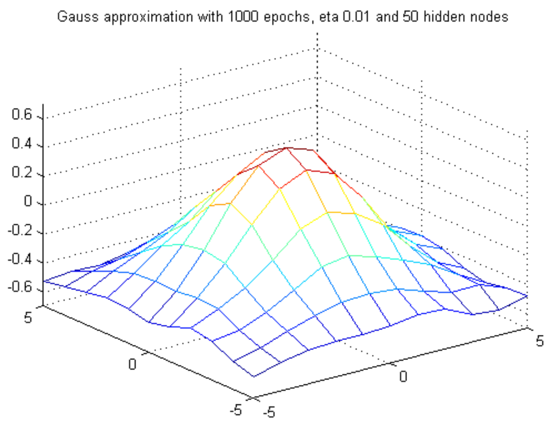
WX =

$$\begin{array}{rrrrrrrr}
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1
\end{array}$$

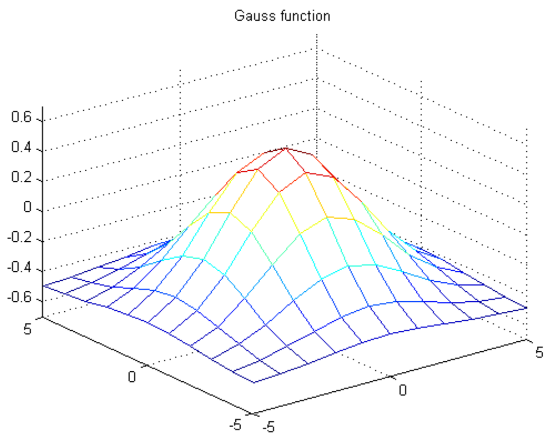
Figure 9: Results from compression network

4 Two layered neural network used for approximation

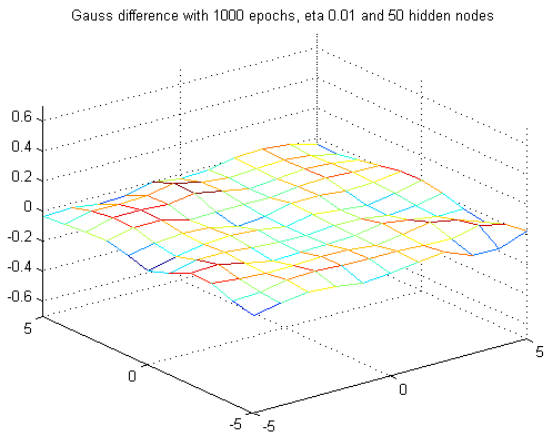
We try and approximate a Gaussian function using a twolayered network. The result can be seen in figure 5.



(a) Approximated gaussian function.



(b) The real gaussian function.

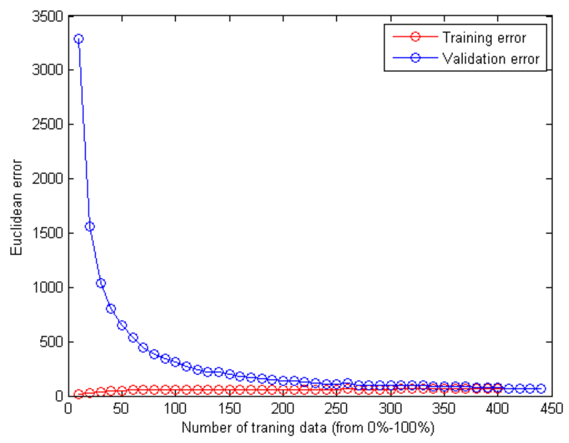


(c) Non seperable data to run twolayer network on.

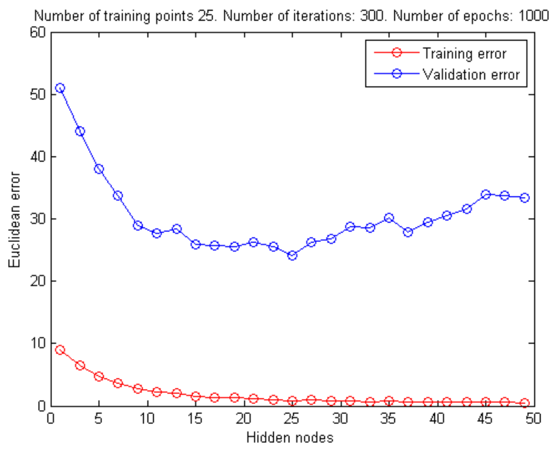
Figure 10: Two layer approximation of Gaussian function.

5 Two layered neural network used for generalisation

Finally we try to see how the generalisation accuarcy of the network depends on the size of the training data aswellas on the number of hidden nodes as seen in figure 11(a) and figure 11(b). We see that the more data you have the better you will classify unseen data. We also see that the more hidden nodes you haev the more the network is prone to overfitting.



(a) Generalization depending on training data size.



(b) Generalization depening on number of hidden nodes.

Figure 11: Two layer approximation of Gaussian function.