# NEAT Module ACW 2008

Contents

[Question 1 1](#_Toc228876608)

[Perceptron 1](#_Toc228876609)

[Solving the Problem 2](#_Toc228876610)

[Generalisation 4](#_Toc228876611)

[Conclusion 5](#_Toc228876612)

[Question 2 6](#_Toc228876623)

[Discussion of Results as the Number of Neurons in Each Layer is Varied 6](#_Toc228876624)

[Discussion on the Effect of a Momentum Term 7](#_Toc228876625)

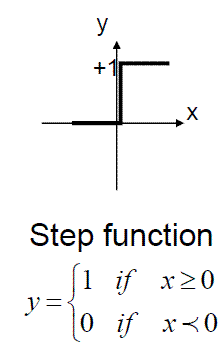
[Conclusions 7](#_Toc228876626)

[Bibliography and Background References 8](#_Toc228876627)

# Question 1

## Perceptron

The perceptron is not capable of learning the data. This is because the data is made up of continuous and real numbers, and a Perceptron (the one I have coded) utilises a Step Function. A Step Function only outputs 0 or 1, therefore a Perceptron is of no use for learning the movements of the mouse, as an example one of the movements of the mouse is *0.100502*. The diagram below demonstrates this

  
This is the code of my Step Function:  
static int Activation(double netSum)

{

if (netSum > 0)

{

return 1;

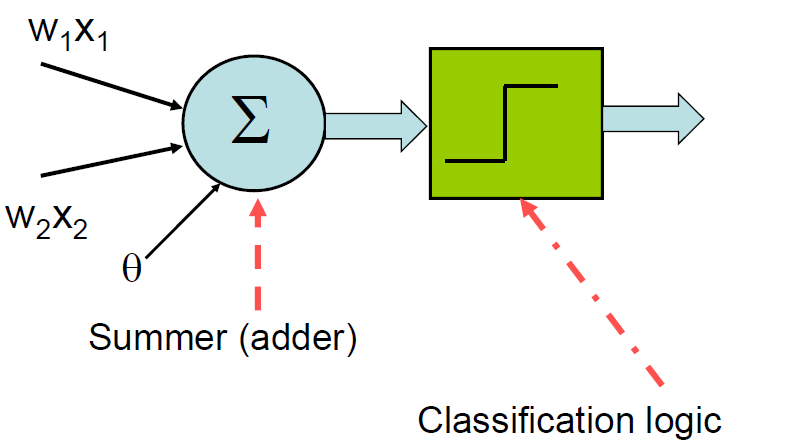
}

else

{

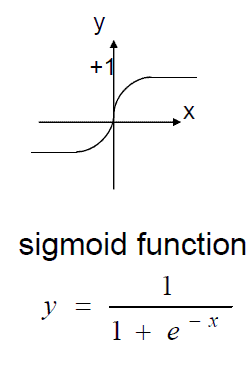
return 0;

}

}  
  
Here is a model of the Perceptron with two inputs:  


|  |  |
| --- | --- |
| **Key** | **What It Means** |
| W1 | Is the weight bound to the first input |
| X1 | Is the value of the first input |
| W2 | Is the weight bound to the second input |
| X2 | Is the value of the second input |
| Summer (adder) | Multiplies the input by the weight for all inputs and keeps the total. |
| Classification logic | In this case this is the Step Function. |

## Solving the Problem

To solve this problem the Step Function must be replaced with a Sigmoid Function. A Sigmoid Function will solve the problem of learning the mouse movements of the mouse, this is because a (Kambhampati) *Sigmoid Function is differentiable everywhere and in fact continuously differentiable*. And because of this reasoning a Sigmoid Function can work with the data it has been given and output numbers between 0 and 1 (something that the Step Function cannot do).  
A Sigmoid Function is depicted below:  


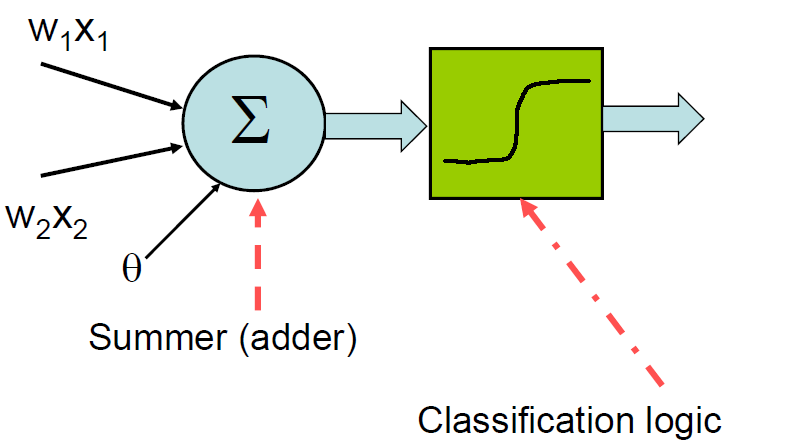
This is my code for the Sigmoid Function:  
inline static double Activation(double netSum)

{

return 1.0 / (1.0 + exp(-netSum));

}

Here is a model of the Neuron (with two inputs) with a Sigmoid Function:



|  |  |
| --- | --- |
| **Key** | **What It Means** |
| W1 | Is the weight bound to the first input |
| X1 | Is the value of the first input |
| W2 | Is the weight bound to the second input |
| X2 | Is the value of the second input |
| Summer (adder) | Multiplies the input by the weight for all inputs and keeps the total. |
| Classification logic | In this case this is the Sigmoid Function. |

## Generalisation

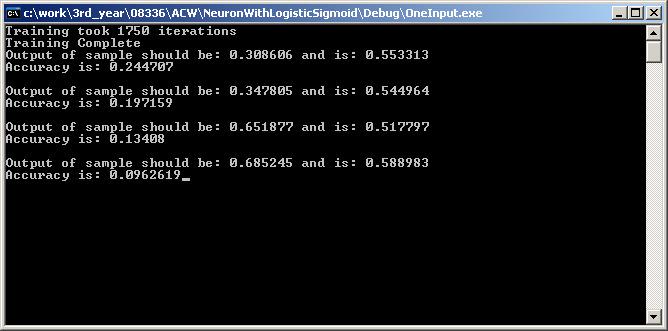
It is not possible for this network to generalise. This is because it is a simple network with the smallest possible amount of nodes. As Hartley states, *this 'sparse' network will be such a simpleton that it will only be able to grasp the simplest facts about the training data set*. One way for the network to posses the ability of generalisation is to increase the network (increasing the nodes), and monitor the generalisation ability closely each time. Hartley suggests that incidentally *when testing on unseen data it is important to use more than a few token unseen data patterns so that a decent number of unseen trades are generated*.

## Conclusion

## Whilst a neuron with a logistic sigmoid function does solve the initial problem that a perceptron with a step function could not, it does so with a varying degree of accuracy. Indeed, leaving the neuron with just one input is simply not accurate enough. To achieve accurate results I have discovered that a ‘sliding window’ effect has to be used; basically a ‘sliding window’ effect uses previous results as inputs. The table below demonstrates this:

|  |  |
| --- | --- |
| Inputs | Result |
| P1, P2, P3, P4 | P5 |
| P2, P3, P4, P5 | P6 |
| P3, P4, P5, P6 | P7 |

## The two pictures below show the difference in accuracy between just using one input as opposed to four inputs *(Using One Input)*



## 

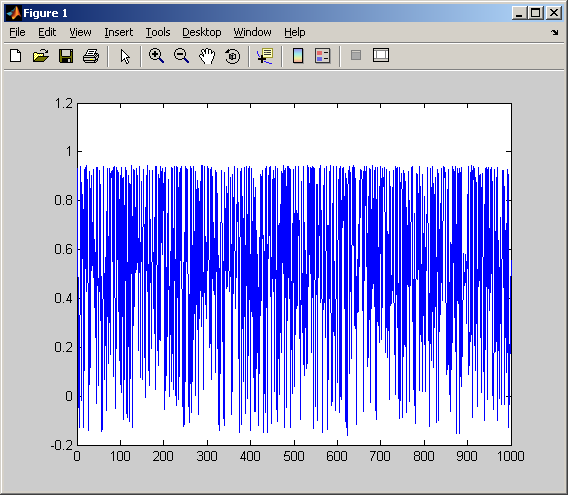
*(Using Four Inputs)*

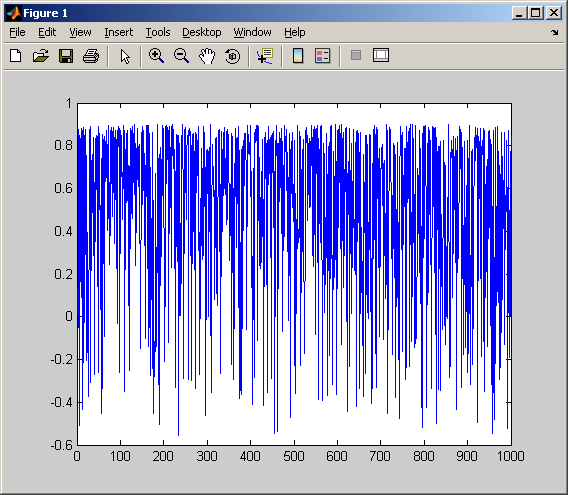
# Question 2

Here is a table of the networks created and trained:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Number of Neurons** | **Number of Layers** | **Momentum Used Yes/No** |
| Network 1 | 6 | 1 | No |
| Network 2 | 6 | 2 | Yes |
| Network 3 | 6 | 2 | No |
| Network 4 | 6 | 1 | Yes |
| Network 5 | 1 | 1 | No |
| Network 6 | 1 | 2 | No |
| Network 7 | 1 | 2 | Yes |
| Network 8 | 1 | 1 | Yes |

## Discussion of Results as the Number of Neurons in Each Layer is Varied

When comparing the results of Networks 5 and 1 it is evident that the classification is better in Network 1 because it has more neurons per layer then what Network 5 does. Therefore, the topology does matter, 1-6-1 vs 1-1-1.  
Network 1  


Network 5  


## Discussion on the Effect of a Momentum Term

When comparing the results of Networks 2 and 3 it is evident that the classification is better in Network 2 because it uses momentum. It is also evident that the learning process is smoother as a result of using momentum and that the (Kambhampati) *error is monotonically decreasing*. Also, as Champard states, *momentum helps to overcome local minima and smooth out the steps to prevent oscillation in areas with high variations*.

## Conclusions

Whilst increasing the learning rate it was discovered that it took less them to train, but the upshot of that was that the results of the learning degraded in comparison to a lower learning rate (0.01).  
The nature of the error became less and less as the number of inputs and the number of hidden nodes increased, the topology of the network that was best was 2-6-1 and was Network 2, and the worse network was Network 5 with a topology of 1-1-1.

# Bibliography and Background References

## Kambhampati, Chandra., 2009, 08336-lecture3-2009.pdf [online]. Nilsson, Nils J,.,1998, Artificial Intelligence: A New Synthesis, San Francisco: Morgan Kaufmann Publishers, Inc. Champandard, Alex J., 2003, AI Game Development, United States of America, New Riders Publishing.

Hartley, Adam., *Some Thoughts On Neural Networks*, Available: <http://www.snapdragonsystems.com/resource/articles/artnn.htm> [Accessed 20th April 2009].