1 Introduction

This report describes the designing, building and fine-tuning of a multi-layer neural network which will be used in a robot that does groceries. The design of this Artificial Neural Network (ANN) is a feed-forward neural network with back-propagation algorithm. The robot would be able to recognize 7 different types of products, such as fruit, meat and candy. The robot can base these conclusions on 10 features, such as roundness, colour, weight, etc.

2 Method

# 2.1. Input and output

There were a couple of architectural decisions to be made before coding the neural network could start. Based on the data that was given, it was immediately clear this neural network would consist of 10 inputs. However, for the number of output neurons this was less evident. Since a total of 7 states was necessary for the output, one could argue that working with binary output would make 3 outputs sufficient. This was first considered to be the most effective method, because there were fewer output states thus increasing the chance of getting it right when calculating. Ironically, however, this method proved inadequate due to a high error rate, and so the number of output neurons was increased to 7. In this architecture, there should be only one active node after each calculation, where each node represents one of the 7 states.

# 2.2. Perceptron

Each node in the neuron is nothing more than a perceptron, which multiplies all inputs with weights, and then sums the results. Mathematically:

This output is fed into the activation function, which in our case is a sigmoid function (see Chapter 2.2. Hidden layers):

The principle of the perceptron can be seen graphically in Figure 1.

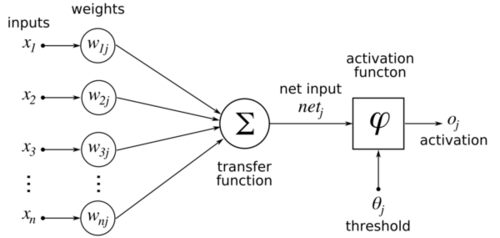


Figure 1: Perceptron

# 2.3. Hidden layers

It was soon thought that making code that was flexible to the number of neurons in a hidden layer would take some extra time to code, but was also the most convenient. Due to time constraints, the number of hidden layers was initially fixed at two layers. Later, however, two extra scripts were made to facilitate one or three hidden layers as well. To update the weights of the output layer, a backpropagation algorithm was implemented:

Where:

Where:

In these functions, means the weight between node ‘j’ from the previous layer, and node ‘k’ from the layer for which the weights are calculated. is the learning rate, which increases the changes in updates at larger values, but also makes it more likely to surpass an optimum. is called the momentum constant, which has a stabilizing effect on training by making the changes less susceptible to spikes in the learning surface. is the error, which is nothing more than the desired output minus the actual output.

The sigmoid function was used as an activation function for the perceptron (see Chapter 2.2. Perceptron) because it is relatively easy to differentiate compared to a step function. The gradient can now be written as:

Because the error of a hidden layer is impossible to compute (one does not know the desired output), the following function is used for the gradient of hidden layers:

The summation is nothing more than a sum of all weights on the output of the hidden neuron multiplied by the gradient of the node they are connected to.

# 2.4. Overview

An overview of the network can be found in Figure 2. The number of hidden layers in this figure can vary, as well as the number of nodes per hidden layer. Note that the amount of nodes in a layer can only go as high as the number of nodes in the previous layer (except for the output layer).

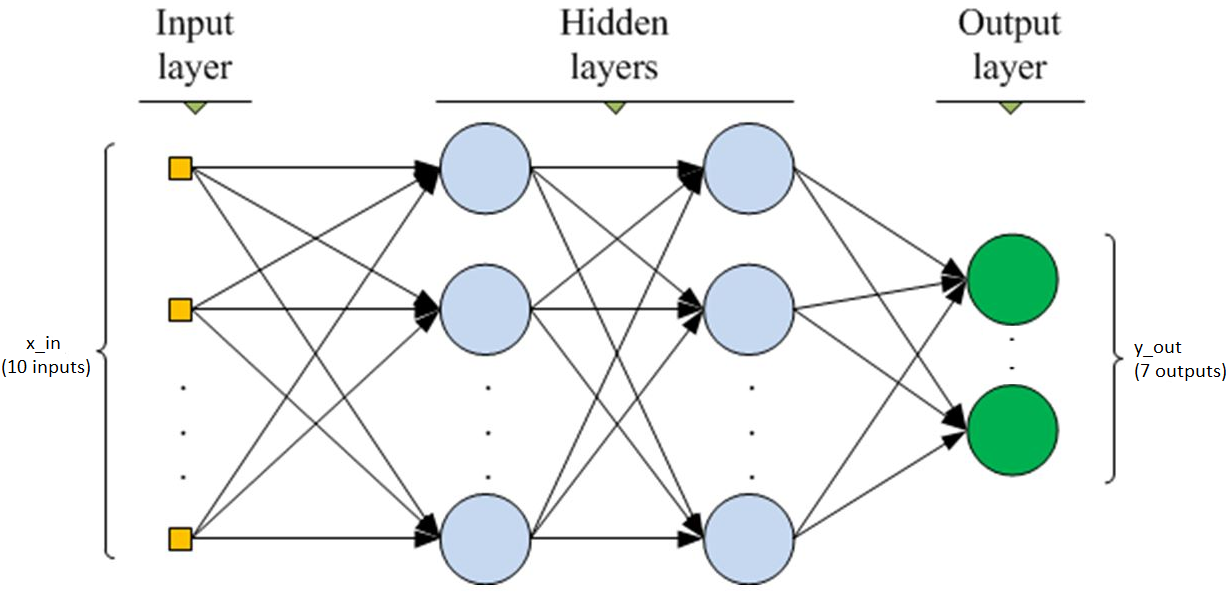


Figure 2: Schematic of the ANN

3 Results

# 1.1. Document Structure

# 1.2. Cover and Title Page

4 Conclusion

# 1.1. Document Structure

# 1.2. Cover and Title Page

5 Discussion

# 1.1. Document Structure

# 1.2. Cover and Title Page