**Project 2.**

Option 2: Training a Neural Network

COS 314

Taetse Durand

u16168080

24/05/2018

**Experiment 1:**

In this experiment I create a neural network with one output neuron with the purpose of classifying an input pattern in two classes. Namely the class that the user inputs, or a second class which is everything else.

For this experiment I initialize the neural network architecture and control variables to the following values.

* Output neuron amount: 1
  + The neural network only has to classify the training set in 2 classes
* Hidden neuron amount: 30
  + The best amount of hidden neurons determined by trial and error and interpolation as shown in the graphs below. Few hidden neurons are needed as only one letter has its own class, and all the other letters share a class. This is a relative simple mapping and can be mapped relatively accurate using as few as 10 hidden neurons.
* Learning rate: 0.2
  + The optimal learning rate determined by trial and error and interpolation as shown in the graphs below. A medium learning rate delivered better accuracies.
* Momentum: 0.2
  + The optimal momentum determined by trial and error and interpolation as shown in the graphs below.

Other parameters

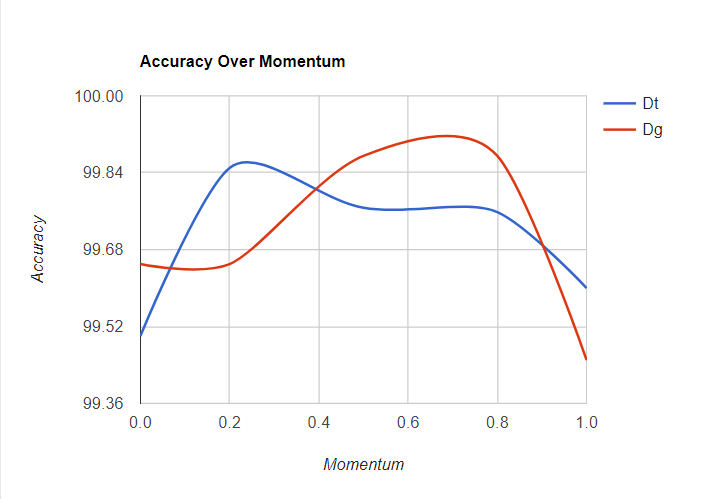
* Max epoch count: 1000
  + Extreme Overfitting occurs after 1000 epochs.
* Desired training accuracy: 99.5
  + A high accuracy with a high generalization ability.

**Impact of the control variables on the training and generalization accuracy.**

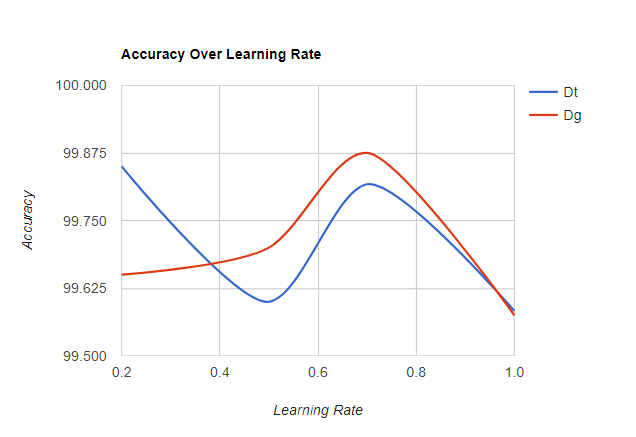
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **# Hidden Neurons** | **Learning Rate** | **Momentum** | **Training Accuracy %** | **Generalization Accuracy %** |
| 30 | 0.2 | **1** | 99.600 | 99.450 |
| 30 | 0.2 | **0.8** | 99.758 | 99.875 |
| 30 | 0.2 | **0.5** | 99.767 | 99.875 |
| 30 | 0.2 | **0.2** | 99.850 | 99.650 |
| 30 | 0.2 | **0** | 99.500 | 99.650 |
| 30 | **0.5** | 0.2 | 99.600 | 99.700 |
| 30 | **0.7** | 0.2 | 99.817 | 99.875 |
| 30 | **1** | 0.2 | 99.583 | 99.575 |
| **1** | 0.2 | 0.2 | 99.200 | 99.375 |
| **10** | 0.2 | 0.2 | 99.717 | 99.675 |
| **40** | 0.2 | 0.2 | 99.558 | 99.575 |
| **80** | 0.2 | 0.2 | 99.617 | 99.600 |
| **100** | 0.2 | 0.2 | 99.625 | 99.575 |

**Accuracy vs the control variables as tested in the table:**

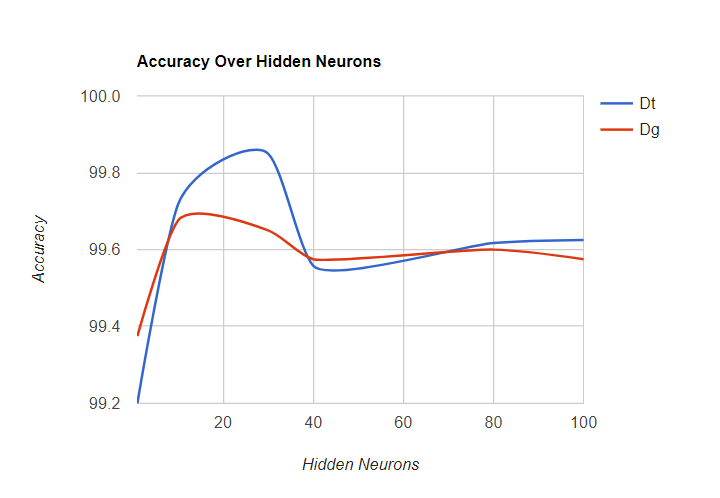
**Momentum:**



**Learning Rate:**

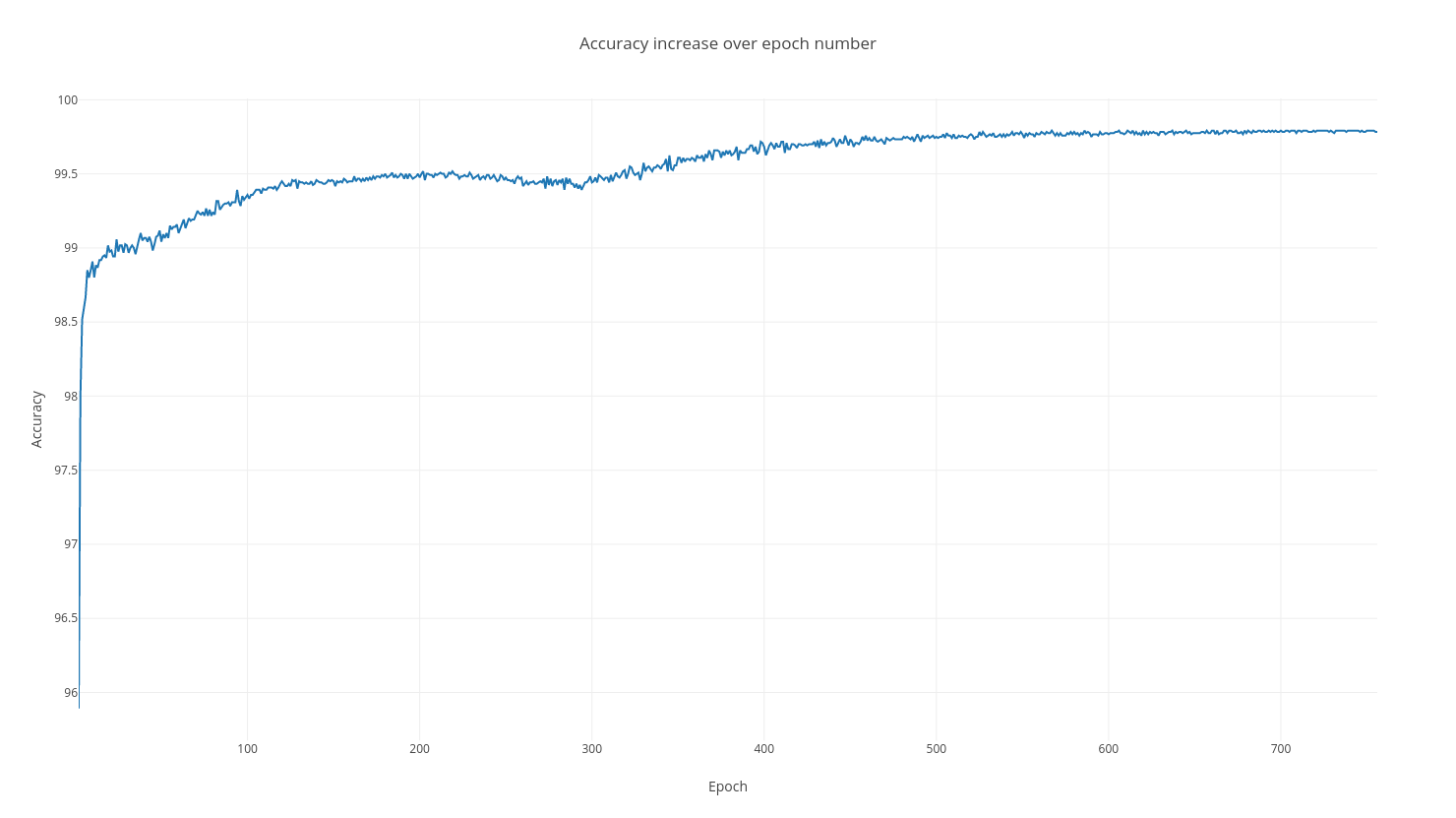


**Hidden Neurons:**



**Learning Profile:**

The learning profile of the neural network using the control variables as mentioned above:



**Experiment 2**

In this experiment I created a neural network again with one output neuron to classify the training set into two classes: vowels and non-vowels. A one as output means the class of the input pattern is a vowel, a zero classifies the pattern as non-vowel.

For this experiment I initialize the neural network architecture and control variables to the following values.

* Output neuron amount: 1
  + The neural network only has to classify the training set into 2 classes
* Hidden neuron amount: 50
  + The best amount of hidden neurons determined by trial and error and interpolation as shown in the graphs below. A higher amount of hidden neurons are needed to correctly map all vowels to one class. Lower hidden neuron counts can be used, but will result in lower generalization accuracies.
* Learning rate: 0.5
  + The optimal learning rate determined by trial and error and interpolation as shown in the graphs below. A high learning rate delivered higher final accuracies and faster training speeds with the increased amount of hidden neurons.
* Momentum: 0.2
  + The optimal momentum determined by trial and error and interpolation as shown in the graphs below. A medium momentum delivered the best accuracies on average.

Other parameters

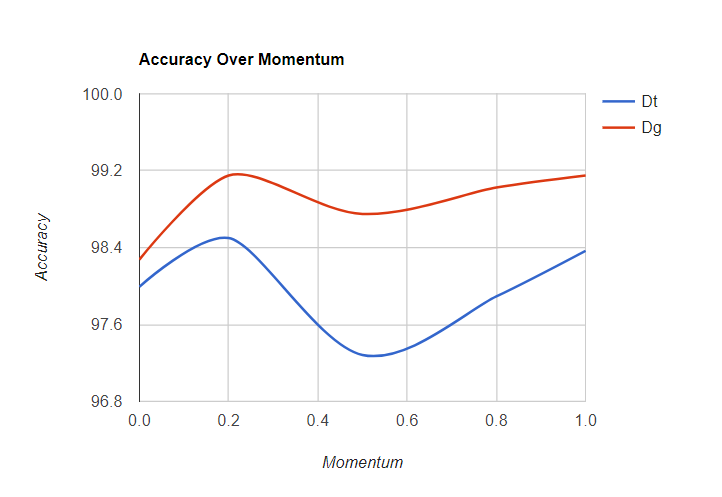
* Max epoch count: 1000
  + Overfitting occurs after 1000 epochs.
* Desired training accuracy: 98
  + A high accuracy with a high generalization ability.

**Impact of the control variables on the training and generalization accuracy.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **# Hidden Neurons** | **Learning Rate** | **Momentum** | **Training Accuracy %** | **Generalization Accuracy %** |
| 50 | 0.5 | **1** | 98.367 | 99.150 |
| 50 | 0.5 | **0.8** | 97.892 | 99.025 |
| 50 | 0.5 | **0.5** | 97.283 | 98.750 |
| 50 | 0.5 | **0.2** | 98.500 | 99.150 |
| 50 | 0.5 | **0** | 97.992 | 98.275 |
| 50 | **0.2** | 0.2 | 97.567 | 98.475 |
| 50 | **0.8** | 0.2 | 97.983 | 98.650 |
| 50 | **1** | 0.2 | 97.883 | 98.975 |
| **100** | 0.5 | 0.2 | 97.733 | 99.075 |
| **70** | 0.5 | 0.2 | 97.925 | 98.700 |
| **30** | 0.5 | 0.2 | 96.967 | 98.875 |
| **10** | 0.5 | 0.2 | 94.850 | 96.400 |
| **1** | 0.5 | 0.2 | 64.550 | 63.750 |

**Accuracy vs the control variables as tested in the table:**

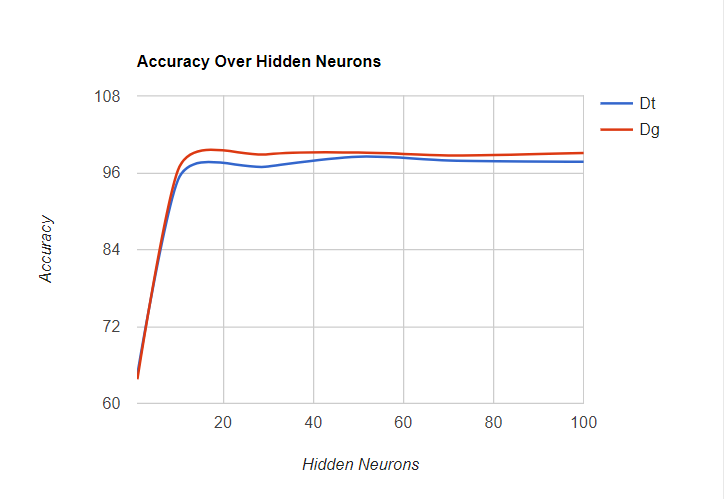
**Momentum:**



**Learning Rate:**

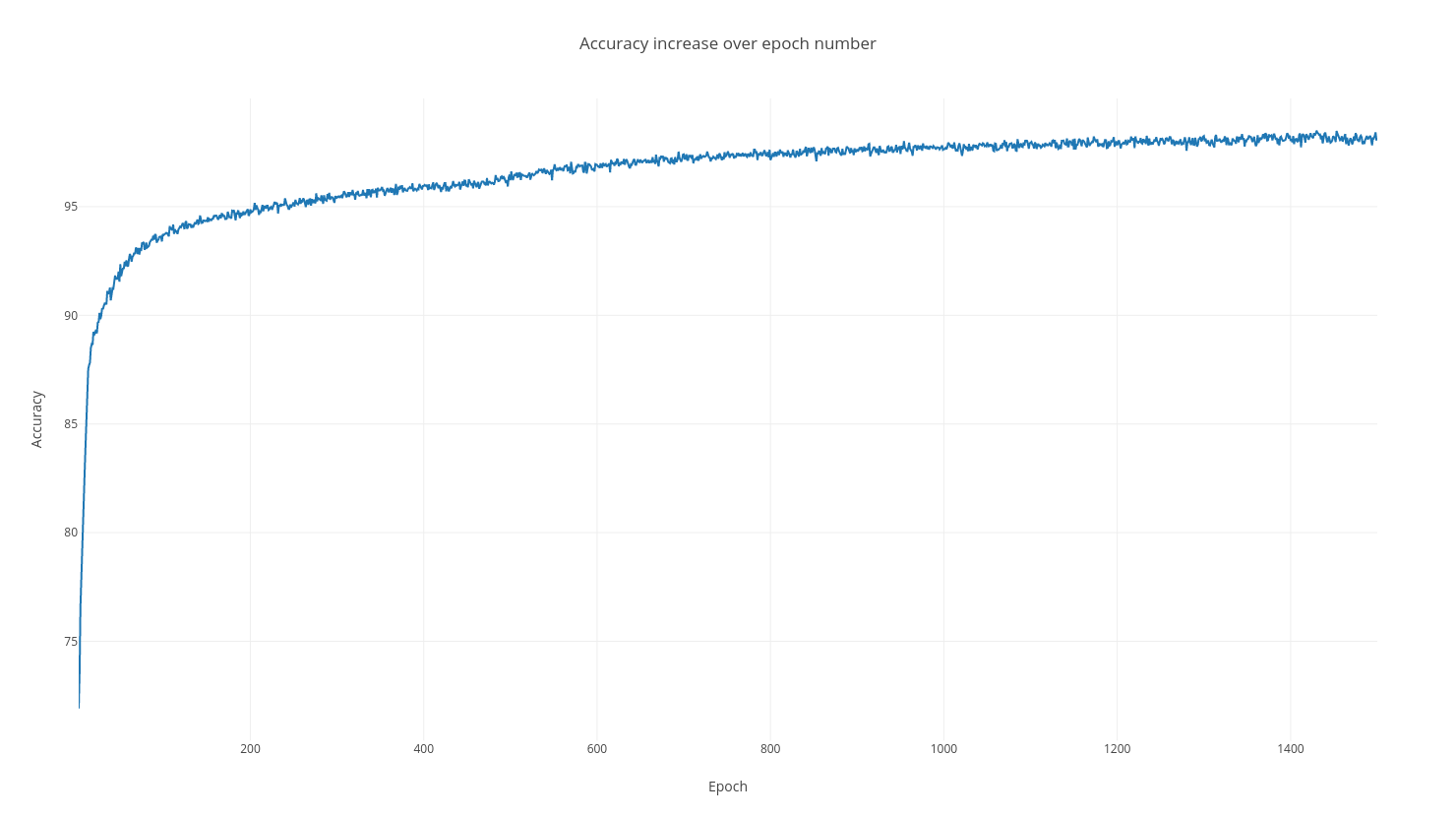


**Hidden Neurons:**



**Learning Profile:**

The learning profile of the neural network using the control variables as mentioned above:



**Experiment 3**

In this experiment I create a neural network with 26 output neurons. One output for each letter in the alphabet. The neural network is trained to recognize each letter of the alphabet and classify them in one of the 26 classes.

For this experiment I initialize the neural network architecture and control variables to the following values.

* Output neuron amount: 26
  + The neural network only has to classify the training set into 26 classes. One class for each letter in the English alphabet.
* Hidden neuron amount: 100
  + The best amount of hidden neurons determined by trial and error and interpolation as shown in the graphs below. A high amount of hidden neurons are needed to correctly map all letters of the alphabet to a separate class. Lower hidden neuron counts can be used, but will result in lower generalization accuracies.
* Learning rate: 0.5
  + The optimal learning rate determined by trial and error and interpolation as shown in the graphs below. A high learning rate delivered higher final accuracies and faster training speeds with the increased amount of hidden neurons.
* Momentum: 0.2
  + The optimal momentum determined by trial and error and interpolation as shown in the graphs below. A medium momentum delivered the best accuracies on average.

Other parameters

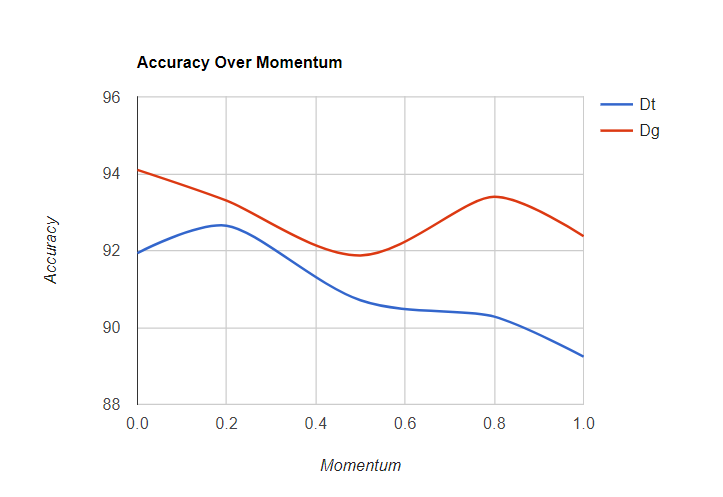
* Max epoch count: 400
  + Overfitting occurs after 400 epochs.
* Desired training accuracy: 90
  + A high accuracy with a high generalization ability.

**Impact of the control variables on the training and generalization accuracy.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **# Hidden Neurons** | **Learning Rate** | **Momentum** | **Training Accuracy %** | **Generalization Accuracy %** |
| 100 | 0.5 | **1** | 89.242 | 92.375 |
| 100 | 0.5 | **0.8** | 90.283 | 93.400 |
| 100 | 0.5 | **0.5** | 90.708 | 91.875 |
| 100 | 0.5 | **0.2** | 92.650 | 93.300 |
| 100 | 0.5 | **0** | 91.933 | 94.100 |
| 100 | **0.2** | 0.2 | 88.850 | 90.525 |
| 100 | **0.8** | 0.2 | 89.683 | 91.900 |
| 100 | **1** | 0.2 | 85.942 | 89.000 |
| **200** | 0.5 | 0.2 | 92.483 | 97.225 |
| **50** | 0.5 | 0.2 | 84.392 | 90.225 |
| **10** | 0.5 | 0.2 | 62.333 | 65.550 |
| **1** | 0.5 | 0.2 | 0.875 | 0.475 |

**Accuracy vs the control variables as tested in the table:**

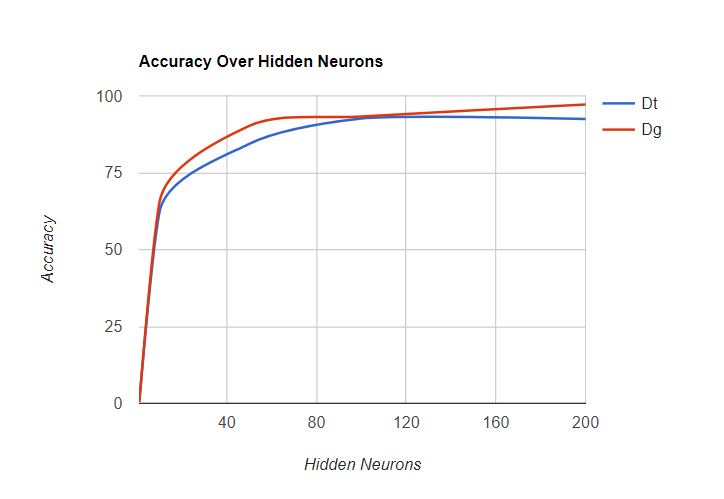
**Momentum:**



**Learning Rate:**

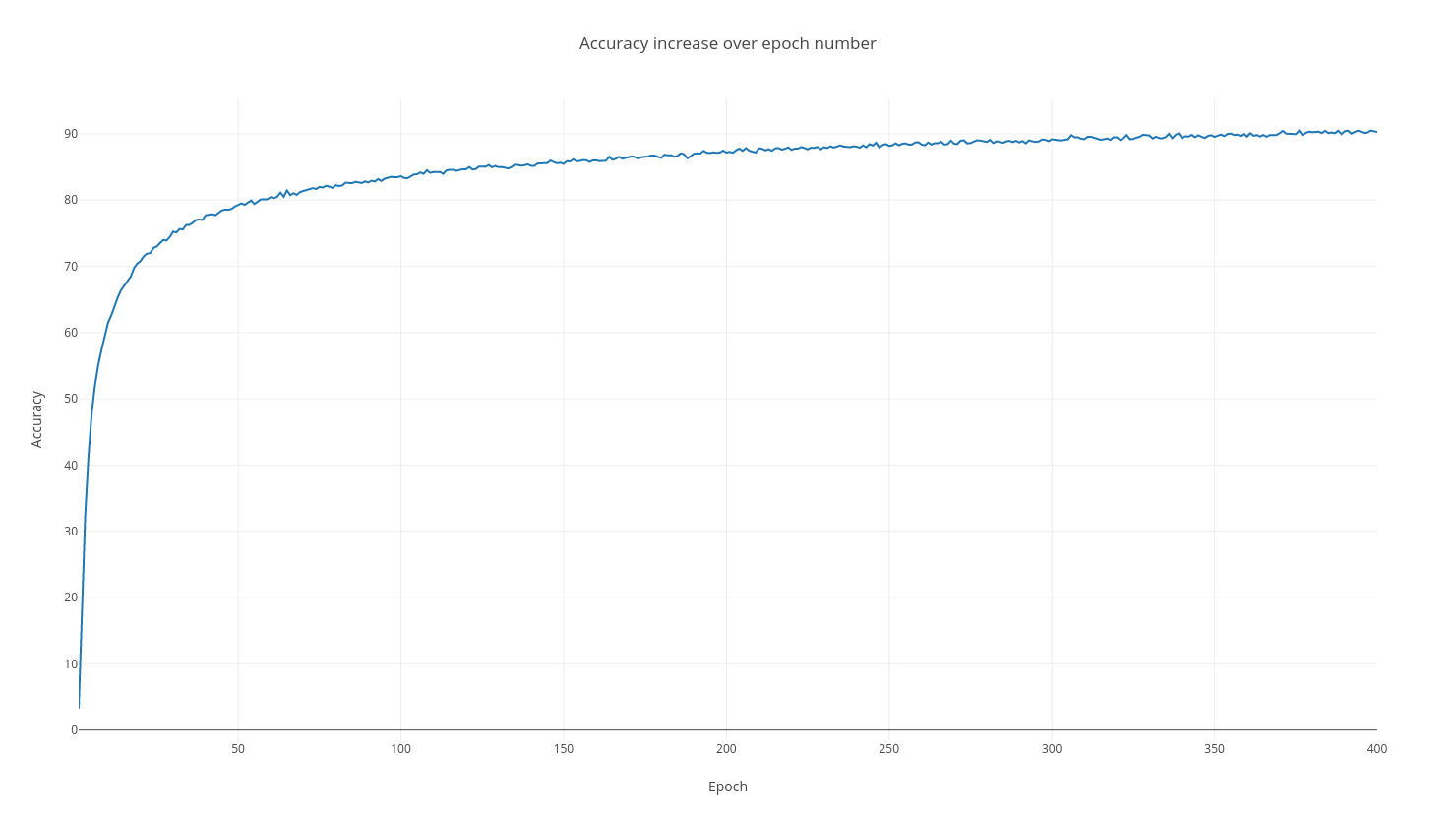


**Hidden Neurons:**



**Learning Profile:**

The learning profile of the neural network using the control variables as mentioned above:



**Instructions:**

To run the project.

Use the provided make file with the following commands:

*make experiment1* to build and run Experiment 1

*make experiment2* to build and run Experiment 2

*make experiment3* to build and run Experiment 3

Each command will run an experiment with the default data file as dataset as given in the specs file.

The control parameters for the neural network are set in the Experiment concrete classes (Experiment1.java, Experiment2.java and Experiment3.java)

An explanation of each parameter is given in these files, so it can be updated.

Output for the experiments are stored in output.txt in the root directory.

The following source files are included:

**Experiment.java**: the abstract Experiment class that implements basic experiment functionality.

**Experiment1.java**: concrete Experiment class. Implements experiment 1’s neural network and functionality.

**Experiment2.java**: concrete Experiment class. Implements experiment 2’s neural network and functionality.

**Experiment3.java**: concrete Experiment class. Implements experiment 3’s neural network and functionality.

**Utils.java**: Utility functions to read the data file and normalize the input training set vectors.

**NeuralNetwork.java**: The neural network class containing all the network matrices and matrix operations.

**InputVector.java**: The input vector class with the target array, input array and classification character.