Introduction to Neurocomputation

Homework No. 1

Report

Hilal Diab – ID

Ameer Tabon – ID

Part 1:

­­We implemented back-propagation algorithm in Matlab. The code is generic to configure the hidden layers and the network configuration in order to be able to test different configuration and submit architecture with different hidden layers with minimal changes.

We used the wines dataset to train and test the algorithm by normalizing the data and dividing randomly to ten-fold training and validation technique.

The configuration we used for network after trying several and different values with different combinations are:

For No hidden layers:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **# Train** | **Learning Rate** | **# Epochs** | **Drop rate** | **Decreasing rate** | **Computation time (seconds)** | **Training error** | **Success rate %** |
| 1 | 0.9 | 80 | 8 | 0.9 | 236 | 19.5 | 80.2 |
| 2 | 0.8 | 30 | 6 | 0.7 | 86 | 19.6 | 79.5 |
| 3 | 0.8 | 30 | 4 | 0.7 | 86 | 19.6 | 80.4 |
| 4 | 0.7 | 40 | 4 | 0.8 | 115 | 19.8 | 80.6 |
| 5 | 0.6 | 100 | 15 | 0.7 | 288 | 19.4 | 80.3 |
|  |  |  |  |  |  |  |  |

For one hidden layers:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **# Train** | **#Layer1 Neurons** | **Learning Rate** | **# Epochs** | **Drop rate** | **Decreasing rate** | **Computation time (seconds)** | **Training error** | **Success rate %** |
| 1 | 3 | 0.9 | 80 | 8 | 0.9 | 356 | 12 | 88 |
| 2 | 6 | 0.9 | 80 | 8 | 0.7 | 367 | 10.5 | 89.3 |
| 3 | 7 | 0.8 | 30 | 2 | 0.7 | 134 | 19.2 | 80.3 |
| 4 | 10 | 0.7 | 50 | 2 | 0.8 | 236 | 18.7 | 80.8 |
| 5 | 15 | 0.6 | 60 | 3 | 0.7 | 296 | 18.7 | 81.3 |
| 6 | 4 | 0.9 | 100 | 8 | 0.9 | 488 | 10.26 | 88.5 |
| 7 | 24 | 0.9 | 100 | 8 | 0.9 |  |  |  |

The result for two hidden layers:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **# Train** | **#Layer2**  **Neurons** | **#Layer1 Neurons** | **Learning Rate** | **# Epochs** | **Drop rate** | **Decreasing rate** | **Computation time (seconds)** | **Training error** | **Success rate %** |
| 1 | 4 | 3 | 0.9 | 60 | 5 | 0.9 |  |  |  |
| 2 | 6 | 4 | 0.9 | 60 | 5 | 0.9 |  |  |  |
| 3 | 10 | 5 | 0.9 | 60 | 5 | 0.9 |  |  |  |
| 4 | 5 | 10 | 0.9 | 60 | 5 | 0.9 |  |  |  |

Part 2:

The input for the neural network is one dimension array with size 256, meaning, we represented the 16x16 input files with single array of size 256, the first 16 bits are the first line in the input file, the second 16 bits are the second line in the input file, and so on.

We decided to expand the dataset by:

1. Flipping 10% percent of the bits on each file.
2. Rotate each input file 90 degree (Char rotated 90 degree).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Learning Rate | # Epochs | Drop Rate | Decreasing Rate | Computation time (seconds) | Training Error | Success Rate % |
| 0.9 | 80 | 8 | 0.9 | 236 | 19.5 | 80.2 |