<u>**Assignment 1 Report**</u>

Name: Imaculate Mosha
Student number: MSHIMA001
Date: 27th September 2015.

Introduction

This report concerns the Practical Assignment on Artificial Intelligence and Neural Networks. As specified in the instructions.

- The provided code was modified until the Neural Net learnt the behaviour of the Minesweeper game using data from the textfile (training_data.txt).
- The neural net was tested with 20 Simulations for each environment provided. The outcome measured were the Average number of mines collected per sweeper and the total number of deaths of sweepers per simulation. These numbers were averaged out over the 20 simulations.
- For comparison, similar data was obtained without training the Neural Network(by commenting out the code "neura_net.train()").
- Data was analysed and conclusions were drawn.

Methodology

The following methods of CNeuralNet class were modified using the given pseudo code:

- CNeuralNet(uint inputLayerSize, uint hiddenLayerSize, uint outputLayerSize, double lRate, double mse_cutoff){

```
        1.  initialise instance variables to the passed arguments
        2. Initialise the weights for hidden layer and output layer nodes using
initWeights() function.
        3. Initialise seed to random number.
    }
```

- CNeuralNet::~CNeuralNet() {

```
    //destructor , frees up memory from heap
        Delete all the arrays of instance variables
    }
```

- void CNeuralNet::initWeights(){

```
    //called from the constructor
        Initialise each weight of each hidden and output layer node to a random number.
```

}

- void CNeuralNet::feedForward(const double * const inputs) {
    1. Copy the inputs array to an instance variable
    2. Calculate the output of each hidden layer node using the sigmoid function and the store the hidden layer outputs in an instance variable array.
    3. Use the hidden layer outputs to compute the final outputs from the output layer nodes using the sigmoid function.

  }

- void CNeuralNet::propagateErrorBackward(const double * const desiredOutput){
    1. Create arrays for hidden layer and output layer errors respectively.
    2. Calculate the error for output layer each node using the following function and store them in an array.

       $d_k = (t_k - o_k) * o_k * (1 - o_k)$

       3. Calculate the error for each hidden layer node using the following function

       $d_h = o_h * (1 - o_h) * \sum d_k w_{kh}$

       4. Change each of the output and layer weights with the function

       $w_{ij} = w_{ij} + \eta * d_i * x_{ij}$

       where x represents the input from j to the node i

       5. Free memory used for hidden layer errors and output layer error array.

  }

- double CNeuralNet::meanSquaredError(const double * const desiredOutput){
    1. Computer the their squares of errors of each output layer node by squaring the respective error (t - o).
    2. Sum the mean square errors of all output layer nodes and return their average (sum/outputlayersize)

  }
- void CNeuralNet::train(const double** const inputs,
  const double** const outputs, uint trainingSetSize) {

```
Until the error is acceptable (less than mse_cutoff):
    For each training data:
        1. Feed forward input to the output
        2. Propagate error backwards
        3. Compute the Mean Square Errors
```

}

- uint CNeuralNet::classify(const double * const input){
    1. Feedforward the input
    2. Return the index of the output layer node with maximum output

}

- double CNeuralNet::getOutput(uint index) const{
  return the output at the given index.

}

Observation

Statistics were obtained in three environments which differ in the number of number of mines and number of supermines .

Test Environment 1: 30 minesweepers, 40 mines and 10 super-mines.
Test Environment 2: 30 minesweepers, 25 mines and 25 super-mines.
Test Environment 3: 30 minesweepers, 10 mines and 40 super-mines.

The results were tabulated below

Before training the neural network:

Environment: 1

Table 1: Test Environment 1 before training

Iteration number	Average mines swept	Number of agents destroyed
1	5.76	23
2	7.63	17

3	5.57	24
4	4.3	27
5	4.3	27
6	5.33	28
7	4.17	28
8	7.63	22
9	4.37	29
10	4.47	20
11	5.2	23
12	7.13	18
13	6.1	19
14	7.93	26
15	8.4	19
16	5.5	25
17	6.4	20
18	7.63	18
19	5.13	27
20	6.17	26
Average	5.74	23.3

Environment: 2

Table 2: Environment 2 before training

Iteration number	Average Mines Swept	Number of agents destroyed.
1	3.07	26

2	3.67	24
3	3.57	29
4	2.47	30
5	4.13	21
6	3.2	20
7	4.83	24
8	3.47	21
9	4.77	19
10	3.93	28
11	4.53	22
12	4.33	27
13	3.5	28
14	3.63	26
15	4.73	19
16	4.17	22
17	3.43	21
18	4.63	24
19	2.73	30
20	3.6	24
Average	3.82	24.25

Environment :3

Table 3: Environment 3 before training

Iteration number	Average Mines Swept	Number of agents destroyed.
1	1.07	29
2	1.4	30

3	1.33	27
4	1.73	27
5	0.83	26
6	2.37	23
7	1.8	30
8	1.37	28
9	1.1	29
10	1.47	30
11	1.17	29
12	1.43	30
13	1.4	22
14	1.43	30
15	1.37	26
16	2	29
17	1.9	26
18	1.7	22
19	1.9	29
20	1.23	29
Average	1.5	26.1

After training the neural network:

Environment: 1

Table 4: Environment 1 after training

Iteration number	Average Mines swept	Number of agents destroyed
1	1.9	8
2	1.03	6
3	0.8	4
4	0.6	6
5	1.47	4
6	1.17	2
7	0.8	5
8	0.83	4
9	0.43	1
10	0.77	4
11	1.17	3
12	1.57	6
13	1.2	3
14	0.53	5
15	1.1	2
16	1.07	4
17	1.57	3
18	0.57	6
19	1.1	3
20	0.37	2
Average	1.003	4.05

Environment :2

Table 5: Environment 2 after training

Iteration number	Mines swept	Number of agents destroyed
1	0.43	4
2	0.93	3
3	0.27	2
4	0.13	1
5	0.17	6
6	0.47	1
7	0.3	3
8	0.57	2
9	0.53	9
10	0.23	6
11	0.33	4
12	0.27	2
13	0.63	3
14	1.17	5
15	1.07	4
16	0.7	8
17	0.63	8
18	0.5	4
19	0.53	5
20	0.5	12
Average	0.485	4.6

Environment :3

Table 3: Environment 6 after training

Iteration number	Mines swept	Number of agents destroyed
1	0.13	5
2	0.07	8
3	0.033	5
4	0.067	5
5	0.27	4
6	0.17	4
7	0.13	3
8	0.1	8
9	0.23	5
10	0.13	2
11	0.33	2
12	0.23	4
13	0.13	2
14	0.23	4
15	0.1	5
16	0.13	6
17	0.13	7
18	0.13	5
19	0.3	4
20	0.2	8
Average	0.162	4.8

Analysis

I plotted the graphs of the average values of sweeper's death and number of mines collected for comparison between a trained and untrained neural net. The following graphs were obtained.

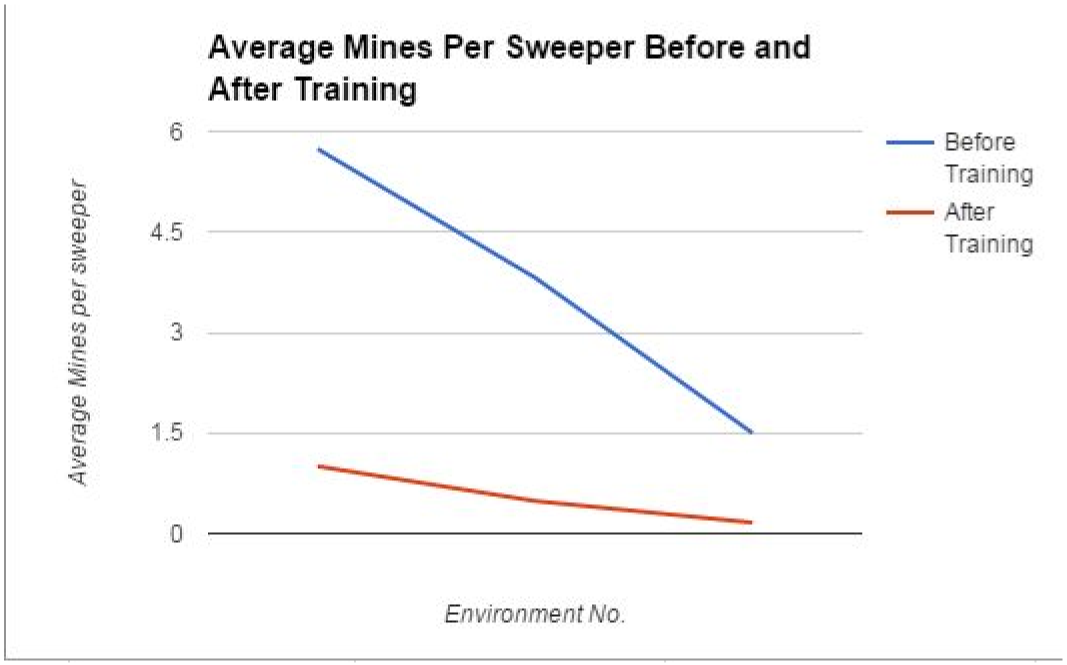

Figure 1: Average mines per sweeper before and after training the neural network

From the graph it is clear that the average number of mines a sweeper collects decreases from test environment 1 to 3. This is explained by the decrease in the number of mines from 40 to 25 to

10. In addition to that the agents collect more mines if they are trained than when they are untrained. This is in contrast to what is expected. The plausible explanation for this is that after training , the agents try as much as possible to collect the mines but most end up cycling around some mines (when approaching at wrong inclination or when a mine and supermine are very close) instead of collecting more. This behaviour is demonstrated in Figure 2 below.

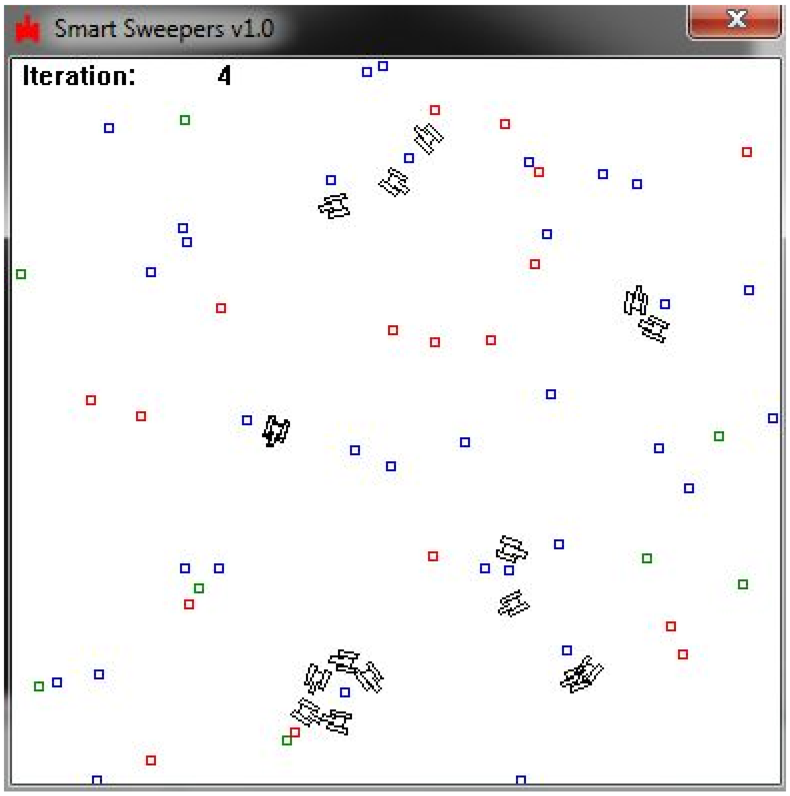

Figure 2 : Sweepers cycling around the mines .

The untrained agents roam about randomly without cycles leading to more mines collected per sweeper.

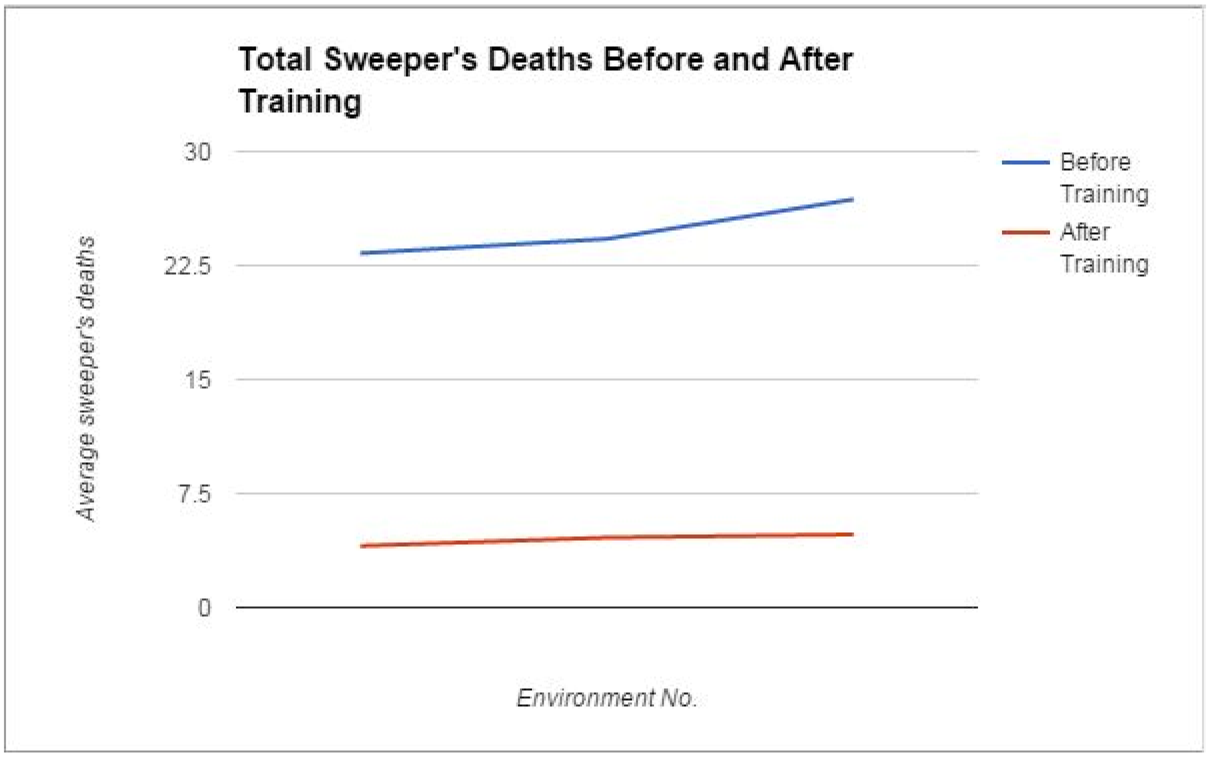

Figure 3: Average deaths per iteration before and after training the neural network

From the graph it is observed that training the neural network decreased the number of deaths of sweepers significantly. This is the expected behaviour as the agents are trained to avoid supermines. Moreover the number of deaths of agents increased gradually from test environment 1 to 3. This is explained by the increase in the number of supermines.

Conclusion:

The neural network learned the correct behaviour, that is to collect as many mines as possible and avoid as many supermines and rocks as possible. On the other hand, training the neural network (agents) resulted in decrease in number of mines collected per sweeper because the sweeper gets stuck in cycles. Training the neural network significantly reduced the number of deaths of sweepers.