

University
of
St Andrews

Fourth Practical-CS5011

Artificial Neural Networks - A Ticketing Routing Agent

Submitted by:

180030908

Submission Date: December 20, 2019

1. Introduction:

In this project, I have developed a ticket routing agent using technique of neural network. Some data set was given in tickets.csv file. The main task was to map the request of the user with a specific response team.

In this report, first I will introduce the basic terminologies in neural networks. I will discuss then the design and implementation details of the project that will include the design decisions I took and the steps I covered for implementation of different parts. I will use screenshots to demonstrate that as well. Lastly, I will provide an evaluation of my algorithm and techniques along with different test cases to support the validity of my work.

In this Assignment, I have implemented the complete sections of Basic and Intermediate agent. At the end of the report, I will also few limitations of any part of this implementation as well.

Neural Networks:

Artificial Neural Network were introduced in late 90's. They were designed for classification, clustering, pattern recognition and future predictions. ANNs can generalize data after learning from given input and their relationship, hence predicting unseen data and situations like human brain. [1]

ANNs has three interconnected layers. On the first layer input data is given through neurons which sends it to second hidden layer. From there, output result is passed to the third/output layer. Figure 1 shows architectural structure of the ANN.

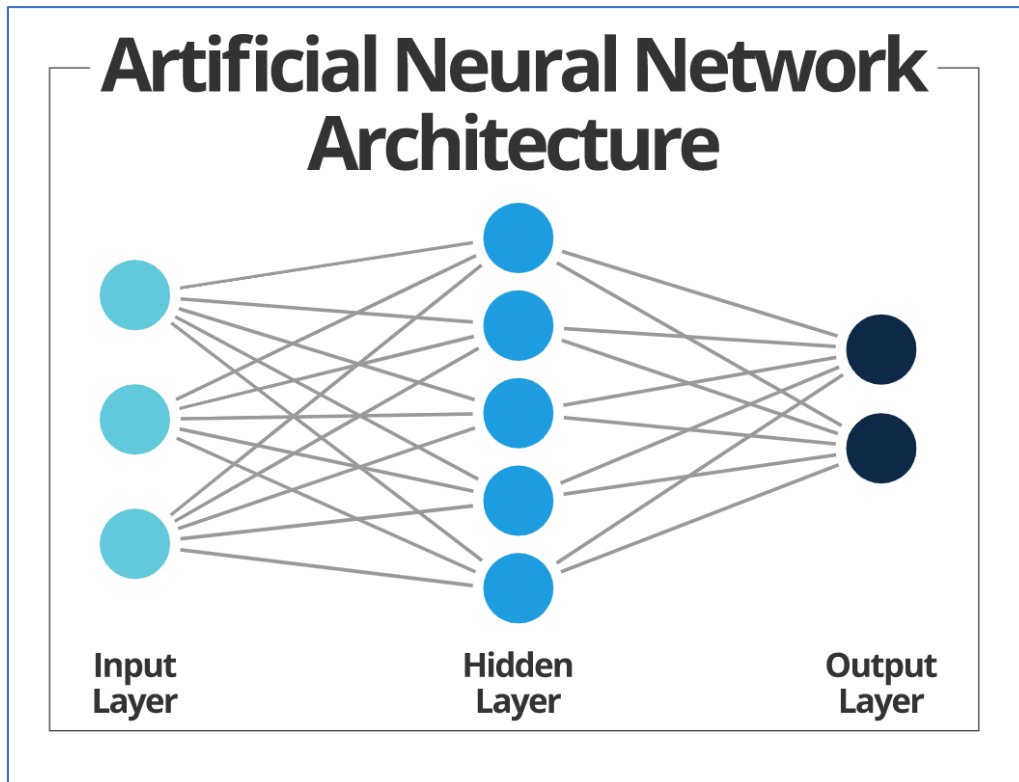


Figure 1: ANN Structure [2]

Instructions to Run the code:

I am unable to run the code from command line as it is giving me class path missing error. Hence this program needs to run through an IDE (intelliJ or Eclipse) from its console. After running the program from console, 'Bas' or 'Int' option should be given to the program to start the execution.

```

1 package machinelearning;
2 import java.io.IOException;
3
4
5
6
7
8 public class Main {
9     static double[] data=new double[9];
10    static double[] AverageInput=new double[] {0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0};
11    static ArrayList<String>ques=new ArrayList<String>();
12    static double answer;
13    static String responseNew;
14    static String file="Data/tickets.xlsx";
15    static String file2="Data/ticketsUpdated.xlsx";
16    static double [] TestData=new double[] {0.0,1.0,1.0,0.0,0.0,0.0,1.0,0.0,0.0}; //selected this data
17
18    @SuppressWarnings("resource")
19    public static void main(String args[]) throws IOException
20    {
21
22        Network1 net =new Network1();
23
24        System.out.println("Select Basic as Bas or Intermediate as Int");
25
26        Scanner scanner1l = new Scanner(System.in);
27        String SearchName=scanner1l.nextLine();
28
29        //String SearchName=args[0];
30        if (SearchName.equalsIgnoreCase("Bas"))
31        {
32
33            DataSet.readData(file);
34            net.creationOfNetwork();
35            net.training();
36            net.saveNetwork();
37            //this is best saved network and not being updated repeatedly
38            net.loadAlreadySavedNetwork("./Data/Network1Saved.eg");
39            net.testData(TestData);
40
41        }
42    }
43 }

```

In intermediate part, user will be giving answers to the agent in 0 (for No) and 1 (for Yes) format to different questions that agent will ask. User can choose and write the name of the response team in the last question if he/she is not satisfied with the assigned response team.

2. Design and Implementation

Before starting development, I designed my solution with PEAS model.

PEAS Model:

Problem: Predict Response team for the user on unseen data after taking input data from the user based on the training that has been given to the agent.

Agent: Software Program

Performance: This determines the level of correctness of the prediction done by the agent. That is agent is determining correctly that

Environment: In-house Help Desk System for IT Services

Actuators: These are the actions performed by the agent so here this include giving predictions to users, asking questions from the users, getting trained on network

Sensors: This is something perceived by the agent and on this basis, different actions are performed. Here it is receiving input from user.

Implementation Strategy:

The first step in the implementation process was the reading of the data from the csv file. For this purpose, I used APACHI POI Library. I converted the given file into .xlsx file for better use within the library.

1. APACHE POI

It is an open source API that make use of Java program for modifying, reading, creating and displaying MS Office files [4] in a memory optimized way.

For this I imported this library within my code first and then using its built-in commands I read all the data from the file and stored it into an Array List.

2. One-Hot Encoding Technique

The second step was the encoding of the data in Array List into 0 and 1 format using one-hot encoding technique so that it can be fed into the neural network.

One hot encoding is a process by which categorical variables are converted into 0,1 form that could be provided to ML algorithms to do a better job in prediction. [3] With this a binary column for each category is created and a sparse matrix or dense array is returned.

I selected one-hot encoding technique for my implementation because it was easier for me to convert the input and output values into binary for this smaller data set. If the data set would have been a bigger one, I might have used some other encoding technique like one-to-n or binary encoding.

Here for first 9 columns of the whole data set that represents different properties of the request is encoded into 0 and 1 for input array. 'Yes' in the data is being encoded to 1 and 'No' is being encoded to 0. The last column of the data represents the name of

different response teams. The value of this column is encoded into 0 and 1 and then stored in the output array. Here I made use of hash map with hard coded values of five different types of response teams. This is mapped with the output data array in a way that a value is 1 for only the particular key to which it represents, and for all other values, the encoding is done to 0 at that particular moment. Following images will give a better understanding of the complete encoding procedure.

	A	B	C	D	E	F	G	H	I	J	K
1	No	No	No	Yes	Yes	Yes	No	Yes	No	Credentials	
2	No	No	Yes	Yes	Yes	No	No	No	No	Credentials	
3	Yes	Yes	No	Yes	No	No	No	No	Yes	Credentials	
4	No	No	No	Yes	Yes	No	No	No	Yes	Credentials	
5	Yes	Yes	No	Yes	Yes	No	No	No	Yes	Credentials	
6	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes	Credentials	
7	Yes	Yes	No	Yes	Yes	No	No	No	Yes	Credentials	
8	No	No	No	Yes	No	No	No	No	Yes	Credentials	

Figure 2: Data in Excel File

```
No-No-No-Yes-Yes-Yes-No-Yes-No-Credentials-
No-No-Yes-Yes-Yes-No-No-No-No-Credentials-
Yes-Yes-No-Yes-No-No-No-No-Yes-Credentials-
No-No-No-Yes-Yes-No-No-No-Yes-Credentials-
Yes-Yes-No-Yes-Yes-No-No-No-Yes-Credentials-
Yes-Yes-No-Yes-Yes-Yes-No-No-Yes-Credentials-
Yes-Yes-No-Yes-Yes-No-No-No-Yes-Credentials-
```

Figure 3: Data Read by the program through APACHE POI

```
0.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0
0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0
1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0
1.0 1.0 0.0 1.0 1.0 0.0 0.0 0.0 1.0
1.0 1.0 0.0 1.0 1.0 1.0 0.0 0.0 1.0
```

Figure 4: Encoding of first 9 rows to Input Data Array

```
Current key: Emergencies
1.00.00.00.00.0
Current key: Credentials
0.01.00.00.00.0
Current key: Datawarehouse
0.00.01.00.00.0
Current key: Networking
0.00.00.01.00.0
Current key: Equipment
0.00.00.00.01.0
```

Figure 5: Encoding for Output Data Array and mapping within HashMap

3. Creation of Network with Encog Library

Since this project is implemented in Java, I used Encog library for network creation, training, saving, loading and lastly testing of the network.

The first step I did here was the creation of the network. For that I give input and output units to the network.

After that I added input, output and hidden layers in the network with Sigmoid function as the activation function. The reason for using sigmoid function here instead of step function is that it provides better continuity in the results as the step function is non-differentiable at $x = 0$ which means there won't be progress in gradient descent during updating the weights and backpropagation will fail [5].

The number of input layer's units in a network is mostly the number of input features of the dataset which is 9 in this case and number of output layer's units is the number of output features which is 5 in this case.

The number of hidden units ideally within input and output units, so I set number of hidden units here to 9. I decided for this number after testing number on different values and then finalized this when I got the most accurate result (More detailed discussion is shown in evaluation section).

Here I also set a value for momentum and learning rate to 0.3 and 0.1 respectively. Momentum and Learning rate are hyperparameters that are used to make the training within a network more efficient by reducing error between actual and targeted output. Momentum further helps the network come out of local minima [6]. Figure 6 shows how momentum helps in speeding up convergence to the minimum by damping oscillations.

Choosing the learning rate and momentum was challenging as if the value is very small, it makes training process very slow and a large value may result in an unstable training process, missing important data points [1].

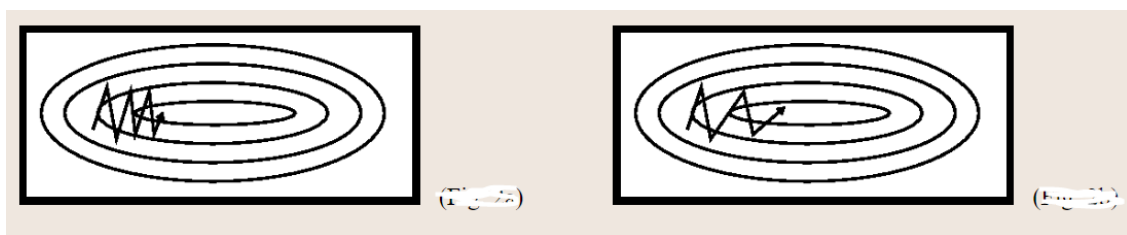


Figure 6: Without Momentum(left) and With Momentum (right) [6]

4. Training of Network with Backpropagation

Backward propagation of errors commonly known as Backpropagation. It is a way to effectively train a neural network through chain rule. It repeatedly adjusts the weights of the connections in the network in order to minimize a measure of the difference between the actual and targeted output. It aims to minimize the cost function by adjusting network's weights. These weights are by default set by the Encog library. The training is done on data until the error is reduced to less than 0.01.

5. Loading, Saving and Testing of Network

For the basic part of the assignment, I trained the system multiple times and the I saved the network with least error and least training time. The saved network file's name is 'Network1Saved.eg'. I then used this saved network for training and predicting in intermediate part of the assignment. However, in basic part, every time a new network is created and saved in Network1.eg' file.

After saving the network, I loaded the saved network for testing the network. For testing, I passed different unseen input data to the network and then obtained the name of the response team in the network and then checked it with actual output to ensure that network is predicting correctly.

6. User Input and Early Predictions

For the intermediate part, I am giving the option to the user to login a request. For that agent is asking different questions from the user and then providing the user with a response team. Here Instead of asking 9 different questions in one go, agent is asking three question in start and then agent provides user with the name of the response team. If user said that it is not correct, agent ask one more question from user and this process goes on till agent asks all questions about all 9 tags. Even after the last question, if user is not satisfied with the assigned response team, then, agent asks user to provide the name himself from the list of different response teams. Agent then add all the responses in the excel sheet and then retrain the network on this new data.

For this part, I am using 'ticketsUpdated.xlsx' file. Again, I am reading and adding rows in the file here through APACHE POI library.

For early predictions, for unseen responses of the users, I took average of yes and no for every tag and the tag with highest yes is set to 1 and set to 0 if the tag has more No values.

	Request	Incident	Web Service	Login	Wireless	Printing	Id Cards	Staff	Student
Yes	79	110	141	94	160	91	107	76	101
No	171	140	109	156	90	159	143	174	149

Request	Incident	Web Service	Login	Wireless	Printing	Id Cards	Staff	Student
0	0	1	0	1	0	0	0	0

Figure 7: Reasoning for early predictions

Code Structure:

In the code, I have four classes. One is main class for dealing with user inputs, one is the basic dataset class for reading data from excel sheets and encoding it into input and output array, third class is Network class where network is created and trained and last one is the class for adding new rows in the excel sheet.

3. Evaluation

To evaluate the system, I have used the error value and training time as bench mark. I have used different values of momentum, learning rate and hidden unit value to find the best value from each where network is providing maximum efficiency. Following table will show network performance on different values. The figures below show the results of the table as well.

Learning Rate	Momentum	Hidden Layer Units	Epochs	Error Value	Figure
0.1	0.3	9	104	0.0099	Figure 8
0.2	0.3	8	176523	0.083200	Figure 9
0.1	0.3	7	84561	0.008299	Figure 10
0.2	0.1	9	5099	0.009595	Figure 11
0.5	0.3	6	120931....	0.19999	Figure 12

The above table clearly shows that only the highlighted row was the most efficient one in terms of training time. In third we see less error but the training time was significantly high than the highlight row.

```
Epoch #77 Error:0.012915657451143515
Epoch #78 Error:0.012780207939114835
Epoch #79 Error:0.012647905074152378
Epoch #80 Error:0.01251855943773086
Epoch #81 Error:0.012391991885926122
Epoch #82 Error:0.012268032915787427
Epoch #83 Error:0.01214652217918906
Epoch #84 Error:0.012027308122687274
Epoch #85 Error:0.01191024773735638
Epoch #86 Error:0.011795206407179768
Epoch #87 Error:0.011682057848259986
Epoch #88 Error:0.011570684133722288
Epoch #89 Error:0.011460975800506494
Epoch #90 Error:0.011352832034008081
Epoch #91 Error:0.011246160924494838
Epoch #92 Error:0.011140879785211819
Epoch #93 Error:0.011036915516080317
Epoch #94 Error:0.01093420498912606
Epoch #95 Error:0.010832695422810977
Epoch #96 Error:0.010732344703249297
Epoch #97 Error:0.010633121602227937
Epoch #98 Error:0.010535005836698627
Epoch #99 Error:0.010437987913770602
Epoch #100 Error:0.010342068710829995
Epoch #101 Error:0.010247258753284989
Epoch #102 Error:0.010153577172615537
Epoch #103 Error:0.010061050353613755
Epoch #104 Error:0.009969710309219545
Trained succesfully.
Saving network
Loading network
Network1 Input = 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0,
Actual Output = 0.046824321780077435, 0.9225106384474675, 0.0011113187808479527, 1.5404709246314975E-4, 0.08420998463581467,
The highest value in the array along with index is : 0.9225106384474675,1
Actual Output after 0 and 1 encoding = 0.0, 1.0, 0.0, 0.0, 0.0,
Your Response will be in Team: Credentials
Expected Output = 0.0, 1.0, 0.0, 0.0, 0.0,
```

Figure 8


```

Epoch #176482 Error:0.08320021048393944
Epoch #176483 Error:0.08320021048255347
Epoch #176484 Error:0.0832002104811675
Epoch #176485 Error:0.08320021047978152
Epoch #176486 Error:0.08320021047839557
Epoch #176487 Error:0.0832002104770096
Epoch #176488 Error:0.08320021047562376
Epoch #176489 Error:0.0832002104742379
Epoch #176490 Error:0.08320021047285198
Epoch #176491 Error:0.08320021047146614
Epoch #176492 Error:0.08320021047008039
Epoch #176493 Error:0.08320021046869458
Epoch #176494 Error:0.0832002104673088
Epoch #176495 Error:0.08320021046592302
Epoch #176496 Error:0.0832002104645373
Epoch #176497 Error:0.08320021046315157
Epoch #176498 Error:0.0832002104617658
Epoch #176499 Error:0.08320021046038018
Epoch #176500 Error:0.08320021045899449
Epoch #176501 Error:0.08320021045760873
Epoch #176502 Error:0.08320021045622318
Epoch #176503 Error:0.0832002104548375
Epoch #176504 Error:0.08320021045345191
Epoch #176505 Error:0.08320021045206637
Epoch #176506 Error:0.08320021045068085
Epoch #176507 Error:0.08320021044929524
Epoch #176508 Error:0.0832002104479097
Epoch #176509 Error:0.08320021044652424
Epoch #176510 Error:0.08320021044513878
Epoch #176511 Error:0.08320021044375321
Epoch #176512 Error:0.08320021044236775
Epoch #176513 Error:0.08320021044098237
Epoch #176514 Error:0.08320021043959692
Epoch #176515 Error:0.08320021043821155
Epoch #176516 Error:0.08320021043682616
Epoch #176517 Error:0.08320021043544089
Epoch #176518 Error:0.08320021043405547
Epoch #176519 Error:0.08320021043267012
Epoch #176520 Error:0.08320021043128487
Epoch #176521 Error:0.08320021042989961
Epoch #176522 Error:0.08320021042851432
Epoch #176523 Error:0.0832002104271291

```

```

import org.encog.ml.data.basic.BasicMLDataSet;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.back.Backpropagation;
import org.encog.persist.EncogDirectoryPersistence;

public class Network1 {

    public int inputUnits;
    public double[][] input_features;
    public double[][] simpleOutput;
    public int hiddenUnits;
    public int outputUnits;
    public double Momentum;
    public double LearningRate;
    public BasicNetwork basicNetwork;
    public BasicNetwork savedNetwork;
    public BasicNetwork loadedNet;
    public MLDataSet trainingSet;
    double value;
    int index;

    public Network1() {

        //creating the basic network
        basicNetwork = new BasicNetwork();
        //Learning rate and momentum are set after running network with different values and c
        LearningRate = 0.2;
        Momentum = 0.3;
    }

    public void creationOfNetwork()
    {
        input_features = DataSet.input; //its always the same as number of input data's featur
        inputUnits=input_features[0].length;
        simpleOutput = DataSet.output; //its always equal to total number of output features
        outputUnits=simpleOutput[0].length;
        System.out.println(inputUnits + " " + outputUnits);
        hiddenUnits =6;
        basicNetwork.addLayer(new BasicLayer(null,false,inputUnits));
        basicNetwork.addLayer(new BasicLayer(new ActivationSigmoid(),true,hiddenUnits));
        basicNetwork.addLayer(new BasicLayer(new ActivationSigmoid(),false,outputUnits));
        basicNetwork.getStructure().finalizeStructure();
    }
}

```

Figure 9

```

Epoch #84530 Error:0.03974467810769143
Epoch #84531 Error:0.03971238374542758
Epoch #84532 Error:0.0396714128951448
Epoch #84533 Error:0.03961835520356459
Epoch #84534 Error:0.03954716362752502
Epoch #84535 Error:0.0394474707767808026
Epoch #84536 Error:0.03929981676889707
Epoch #84537 Error:0.03906380829412637
Epoch #84538 Error:0.03864347058773808
Epoch #84539 Error:0.03776411164827257
Epoch #84540 Error:0.035422285775471075
Epoch #84541 Error:0.02808274643418596
Epoch #84542 Error:0.032298956455976274
Epoch #84543 Error:0.03874542927779895
Epoch #84544 Error:0.0395822403858864
Epoch #84545 Error:0.039651893954796576
Epoch #84546 Error:0.03961973537710278
Epoch #84547 Error:0.0395462427140001
Epoch #84548 Error:0.039432240926880434
Epoch #84549 Error:0.03925354655801556
Epoch #84550 Error:0.03894690565840252
Epoch #84551 Error:0.03833556530448004
Epoch #84552 Error:0.03677993646822483
Epoch #84553 Error:0.03089935598380522
Epoch #84554 Error:0.014111284618857911
Epoch #84555 Error:0.013611600821554114
Epoch #84556 Error:0.014281182308078101
Epoch #84557 Error:0.01470583507417563
Epoch #84558 Error:0.015772283954745
Epoch #84559 Error:0.012770498279199316
Epoch #84560 Error:0.01008866302899517
Epoch #84561 Error:0.008299112871516535
Trained successfully.
Saving network
Loading network
Network1 Input = 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0,
Actual Output = 0.0029644576904359144, 0.9405902742764843, 4.296
The highest value in the array along with index is : 0.940590274
Actual Output after 0 and 1 encoding = 0.0, 1.0, 0.0, 0.0, 0.0,
Your Response will be In Team: Credentials

Expected Output = 0.0, 1.0, 0.0, 0.0, 0.0,

```

```

import org.encog.ml.data.basic.BasicMLDataSet;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.back.Backpropagation;
import org.encog.persist.EncogDirectoryPersistence;

public class Network1 {

    public int inputUnits;
    public double[][] input_features;
    public double[][] simpleOutput;
    public int hiddenUnits;
    public int outputUnits;
    public double Momentum;
    public double LearningRate;
    public BasicNetwork basicNetwork;
    public BasicNetwork savedNetwork;
    public BasicNetwork loadedNet;
    public MLDataSet trainingSet;
    double value;
    int index;

    public Network1() {

        //creating the basic network
        basicNetwork = new BasicNetwork();
        //Learning rate and momentum are set after running network with different values and a
        LearningRate = 0.1;
        Momentum = 0.3;
    }

    public void creationOfNetwork()
    {
        input_features = DataSet.input; //its always the same as number of input data's featur
        inputUnits=input_features[0].length;
        simpleOutput = DataSet.output; //its always equal to total number of output feature
        outputUnits=simpleOutput[0].length;
        System.out.println(inputUnits + " " + outputUnits);
        hiddenUnits =7;
        basicNetwork.addLayer(new BasicLayer(null,false,inputUnits));
        basicNetwork.addLayer(new BasicLayer(new ActivationSigmoid(),true,hiddenUnits));
        basicNetwork.addLayer(new BasicLayer(new ActivationSigmoid(),false,outputUnits));
        basicNetwork.getStructure().finalizeStructure();
    }
}

```

Figure 10

```

Epoch #5068 Error:0.03971175634319405
Epoch #5069 Error:0.039690625450417245
Epoch #5070 Error:0.03966687192534507
Epoch #5071 Error:0.03964007146136251
Epoch #5072 Error:0.03960973206360718
Epoch #5073 Error:0.0395752954291564
Epoch #5074 Error:0.03953612976228315
Epoch #5075 Error:0.03949158168590095
Epoch #5076 Error:0.03944099598696404
Epoch #5077 Error:0.039383815098317274
Epoch #5078 Error:0.03931970350853667
Epoch #5079 Error:0.03924870430450939
Epoch #5080 Error:0.0391713727794426
Epoch #5081 Error:0.03908878558214108
Epoch #5082 Error:0.039002314898192336
Epoch #5083 Error:0.03891316576700484
Epoch #5084 Error:0.03882188717107213
Epoch #5085 Error:0.038728160651359395
Epoch #5086 Error:0.03863092942973448
Epoch #5087 Error:0.038528571956598506
Epoch #5088 Error:0.03841875638084316
Epoch #5089 Error:0.03829775373817346
Epoch #5090 Error:0.038158864008759764
Epoch #5091 Error:0.037988741216829186
Epoch #5092 Error:0.0377573296144548
Epoch #5093 Error:0.03738377215476798
Epoch #5094 Error:0.03657938071777571
Epoch #5095 Error:0.033697707492204486
Epoch #5096 Error:0.019260281625569673
Epoch #5097 Error:0.01418197763405096
Epoch #5098 Error:0.011240234779848535
Epoch #5099 Error:0.0095958873385763
Trained successfully.
Saving network
Loading network
Network1 Input = 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0,
Actual Output = 0.0029644576904359144, 0.9405902742764843, 4.296
The highest value in the array along with index is : 0.940590274
Actual Output after 0 and 1 encoding = 0.0, 1.0, 0.0, 0.0, 0.0,
Your Response will be in Team: Credentials

Expected Output = 0.0, 1.0, 0.0, 0.0, 0.0,
  
```

```

10 import org.encog.ml.data.basic.BasicMLDataSet;
11 import org.encog.neural.networks.BasicNetwork;
12 import org.encog.neural.networks.layers.BasicLayer;
13 import org.encog.neural.networks.training.propagation.back.Backpropagation;
14 import org.encog.persist.EncogDirectoryPersistence;
15
16 public class Network1 {
17
18     public int inputUnits;
19     public double[][] input_features;
20     public double[][] simpleOutput;
21     public int hiddenUnits;
22     public int outputUnits;
23     public double Momentum;
24     public double LearningRate;
25     public BasicNetwork basicNetwork;
26     public BasicNetwork savedNetwork;
27     public BasicNetwork loadedNet;
28     public MLDataSet trainingSet;
29     double value;
30     int index;
31     public Network1() {
32
33         //creating the basic network
34
35         basicNetwork = new BasicNetwork();
36         //Learning rate and momentum are set after running network with different values and
37         LearningRate = 0.2;
38         Momentum = 0.1;
39     }
40
41     public void creationOfNetwork()
42     {
43         input_features = DataSet.input; //its always the same as number of input data's featu
44         inputUnits=input_features[0].length;
45         simpleOutput = DataSet.output; //its always equal to total number of output features
46         outputUnits=simpleOutput[0].length;
47         System.out.println(inputUnits + " , " + outputUnits);
48         hiddenUnits =5;
49         basicNetwork.addLayer(new BasicLayer(null,false,inputUnits));
50         basicNetwork.addLayer(new BasicLayer(new ActivationSigmoid(),true,hiddenUnits));
51         basicNetwork.addLayer(new BasicLayer(new ActivationSigmoid(),false,outputUnits));
52         basicNetwork.getStructure().finalizeStructure();
53         //creating basic network for next run
  
```

Figure 11

```

Epoch #120890 Error:0.1999999994578074
Epoch #120891 Error:0.19999999945780625
Epoch #120892 Error:0.19999999945780517
Epoch #120893 Error:0.19999999945780408
Epoch #120894 Error:0.19999999945780295
Epoch #120895 Error:0.19999999945780186
Epoch #120896 Error:0.19999999945780078
Epoch #120897 Error:0.19999999945779967
Epoch #120898 Error:0.19999999945779856
Epoch #120899 Error:0.19999999945779745
Epoch #120900 Error:0.19999999945779634
Epoch #120901 Error:0.19999999945779526
Epoch #120902 Error:0.19999999945779415
Epoch #120903 Error:0.19999999945779307
Epoch #120904 Error:0.19999999945779195
Epoch #120905 Error:0.19999999945779084
Epoch #120906 Error:0.19999999945778973
Epoch #120907 Error:0.19999999945778862
Epoch #120908 Error:0.19999999945778754
Epoch #120909 Error:0.19999999945778643
Epoch #120910 Error:0.19999999945778535
Epoch #120911 Error:0.19999999945778424
Epoch #120912 Error:0.19999999945778313
Epoch #120913 Error:0.19999999945778202
Epoch #120914 Error:0.19999999945778094
Epoch #120915 Error:0.19999999945777978
Epoch #120916 Error:0.19999999945777874
Epoch #120917 Error:0.19999999945777763
Epoch #120918 Error:0.19999999945777652
Epoch #120919 Error:0.1999999994577754
Epoch #120920 Error:0.1999999994577743
Epoch #120921 Error:0.1999999994577732
Epoch #120922 Error:0.19999999945777208
Epoch #120923 Error:0.19999999945777097
Epoch #120924 Error:0.19999999945776989
Epoch #120925 Error:0.19999999945776878
Epoch #120926 Error:0.19999999945776767
Epoch #120927 Error:0.19999999945776656
Epoch #120928 Error:0.19999999945776548
Epoch #120929 Error:0.1999999994577644
Epoch #120930 Error:0.19999999945776328
Epoch #120931 Error:0.19999999945776216
  
```

```

35 basicNetwork = new BasicNetwork();
36 //Learning rate and momentum are set after running network with different values and
37 LearningRate = 0.5;
38 Momentum = 0.3;
39 }
40
41 public void creationOfNetwork()
42 {
43     input_features = DataSet.input; //its always the same as number of input data's featu
44     inputUnits=input_features[0].length;
45     simpleOutput = DataSet.output; //its always equal to total number of output features
46     outputUnits=simpleOutput[0].length;
47     System.out.println(inputUnits + " , " + outputUnits);
48     hiddenUnits =6;
49     basicNetwork.addLayer(new BasicLayer(null,false,inputUnits));
50     basicNetwork.addLayer(new BasicLayer(new ActivationSigmoid(),true,hiddenUnits));
51     basicNetwork.addLayer(new BasicLayer(new ActivationSigmoid(),false,outputUnits));
52     basicNetwork.getStructure().finalizeStructure();
53     //resetting input weight for next run
54     basicNetwork.reset();
55 }
56
57 public void training()
58 {
59     trainingSet = new BasicMLDataSet(DataSet.input, DataSet.output);
60
61     //Backpropagation(ContainsFlat network, MLDataSet training, double theLearnRate,double
62     Backpropagation train=new Backpropagation (basicNetwork, trainingSet, LearningRate, Mom
63     int epoch = 1;
64     do
65     {
66         train.iteration();
67         System.out.println("Epoch #" + epoch + " Error:" + train.getError());
68         epoch++;
69     } while(train.getError()>0.01);
70     {
71         System.out.println("Trained succesfully.");
72         train.finishTraining();
73     }
74 }
75
76 public void saveNetwork()
77 {
78     System.out.println("Saving network");
79     EncogDirectoryPersistence.saveObject(new File("./Data/Network1.eg"), basicNetwork);
  
```

Figure 12

Program Limitations:

1. I did not use any validation data as given data set was very small but if I had a big data set, it would be an efficient approach to use the validation set for training.

2. Currently at early guessing I am using the average of greater number of yes and no for each tag to decide the random input data that whether it should be 0 or 1, but after retraining with new data, I am not updating this decision. This could have been improved If I had more time and If I had not done this manually as shown in Figure 7.
3. User can only select from the list of present response teams and is not able to suggest new response teams.
4. The program is unable to run with command line instructions as shown below.

```
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\sak\in>cd pictures
C:\Users\sak\in\Pictures>cd MacineLearningAI
C:\Users\sak\in\Pictures\MachineLearningAI>cd src
C:\Users\sak\in\Pictures\MachineLearningAI\src> javac A4Main.java
error: file not found: A4Main.java
Usage: javac <options> <source files>
use --help for a list of possible options

C:\Users\sak\in\Pictures\MachineLearningAI\src>cd..
C:\Users\sak\in\Pictures\MachineLearningAI>cd machineLearning
The system cannot find the path specified.

C:\Users\sak\in\Pictures\MachineLearningAI>cd src
C:\Users\sak\in\Pictures\MachineLearningAI\src>cd machineLearning
C:\Users\sak\in\Pictures\MachineLearningAI\src\machineLearning>javac A4Main.java
A4Main.java:12: error: cannot find symbol
    Network1 net =new Network1();
                    ^
  symbol:   class Network1
  location: class A4Main
A4Main.java:22: error: cannot find symbol
    Network1 net =new Network1();
                    ^
  symbol:   class Network1
  location: class A4Main
A4Main.java:28: error: cannot find symbol
    DataSet.readData(file);
              ^
  symbol:   variable DataSet
  location: class A4Main
A4Main.java:41: error: cannot find symbol
    DataSet.readData(file2);
              ^
  symbol:   variable DataSet
  location: class A4Main
A4Main.java:87: error: package DataSet does not exist
        for (String s: DataSet.hmap.keySet())
                        ^
A4Main.java:98: error: package DataSet does not exist
        for (double r : DataSet.hmap.get(responseNew))
                        ^
A4Main.java:104: error: cannot find symbol
    DataSetUpdate.updatedata(Averageinput, responseNew);
    ^
  symbol:   variable DataSetUpdate
  location: class A4Main
7 errors
```

4. Testing

For testing purposes, I am taking an unseen data set of 5 rows in an excel sheet (I have added it in the data folder). From there I am taking one row and putting it manually into my program after encoding yes to 1 and 0 to no to evaluate the correctness of network.

Testing of Basic Part:

[illegible]

Figure 13: Unseen Data Set for Testing

First Test input: 0.0, 1.0,1.0,1.0,1.0,0.0,1.0,0.0,1.0

Expected Output: Emergencies

Actual output: Emergencies

```

1  terminated> A4Main [Java Application] C:\Program Files\Java\jdk-13\bin\javaw.exe (Dec 20, 2019, 6:59:13 PM)
2
3  0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
4  0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0
5  1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0
6  1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0
7  0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0
8  0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
9  1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
10 1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
11 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
12 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
13 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
14 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
15 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
16 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
17 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
18 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
19 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
20 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
21 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
22 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
23 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
24 1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
25 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
26 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
27 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
28 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
29 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
30 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
31 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
32 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
33 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
34 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
35 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
36 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
37 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0
38 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
39 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
40 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
41 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
42 Loading network
43 Network1 Input = 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0,
44 Actual Output = 0.9726648727964775, 0.0036077023374465776, 0.007119758485922194, 0.0
45 The highest value in the array along with index is : 0.9726648727964775, 0
46 Actual Output after 0 and 1 encoding = 1.0, 0.0, 0.0, 0.0, 0.0,
47 Your Response will be in Team: Emergencies

```

Figure 14

2nd Test input: 0.0, 0.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0

Expected Output: Credentials

Actual output: Credentials

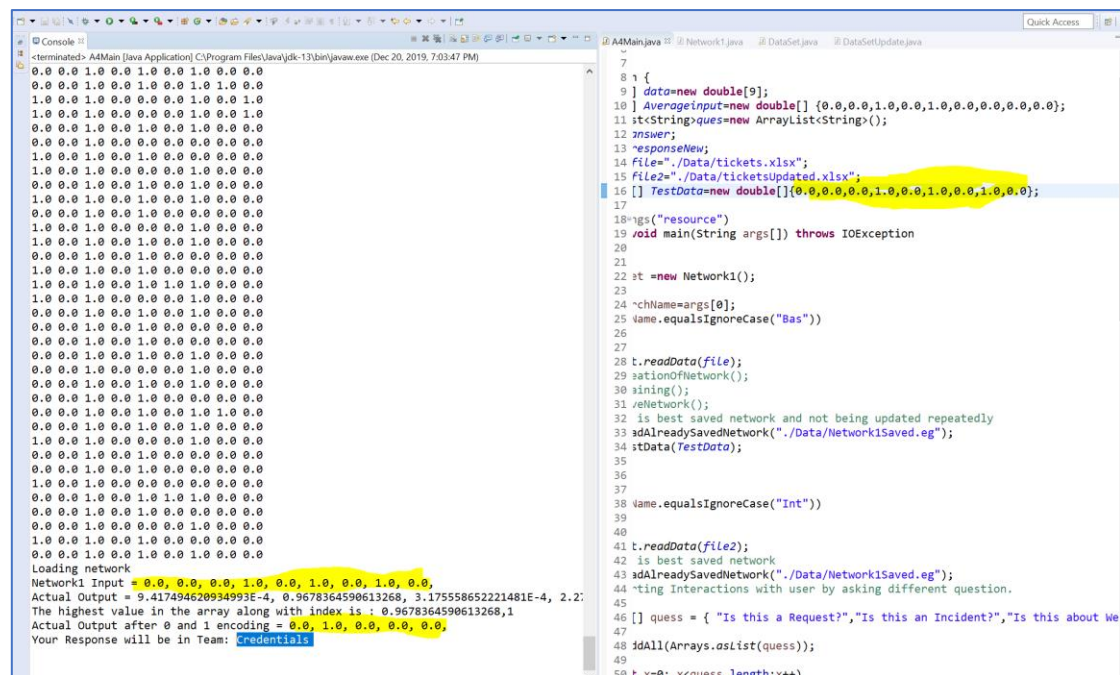


Figure 15

3rd Test input: 0.0, 1.0,1.0,0.0,0.0,0.0,1.0,0.0,0.0

Expected Output: Datawarehouse

Actual output: Datawarehouse

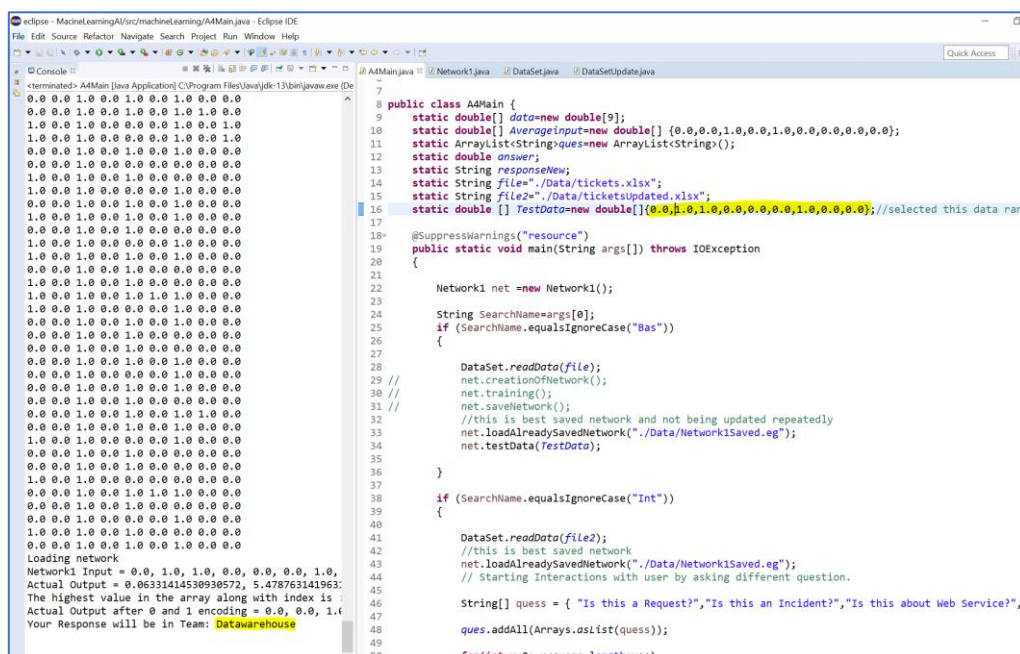


Figure 16

Testing of Intermediate Part:

Below is the training data set before agents take any output from the user.

4. Bibliography

- [1] Stuart Russell and Peter Norvig. (1995) Artificial Intelligence, A Modern Approach, Prentice Hall.
- [2] John A. Flores. (2011) Focus on Artificial Neural Networks
- [3] "What is One Hot Encoding? Why And When do you have to use it?", Hackernoon.com, 2019. [Online]. Available: <https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f>. [Accessed: 15- Dec- 2019]
- [4] "Apache POI - the Java API for Microsoft Documents". [Online]. Available: <https://poi.apache.org/>. [Accessed: 16- March- 2019].
- [5] S. function and J. Schauer, "Step function versus Sigmoid function", Stack Overflow, 2015. [Online]. Available: <https://stackoverflow.com/questions/34469595/step-function-versus-sigmoid-function>. [Accessed: 17- Dec- 2019]
- [6] "Momentum and Learning Rate Adaptation", Cnl.salk.edu. [Online]. Available: <https://cnl.salk.edu/~schraudo/teach/NNcourse/momrate.html>. [Accessed: 12- Dec- 2019]