Implementation list

Part one: Created a training table that can generate the input and output dataset from the file automatically. Construct a multilayer feedforward neural network with one hidden layer. The neural network can be trained by using backpropagation. After training, the network will save in the program root directory. The trained network can identify the required destination by analysing the offering inputs.

Part two: Create the method that can make the agent interact with the user, and obtain the information from users. The system will ask the user a few questions, and then generate the input dataset depending on users' answer. The agent will guess user's dream destination by matching the input dataset with the output in training table. Besides, the agent will make an early guess when the user does not answer all questions.

Part three: If the agent cannot give the dream destination to the user based on the input dataset, the user can choose to add a new destination to an agent. Then, the agent will add the new piece information that involves input and new output in the training table. Moreover, the agent will be trained again based on the new training table, then saving the new network in the program root directory.

There are 4420 words in this report excluded the bibliography.

Literature review

Most people may know about the artificial neural network until the artificial intelligence Go player, AlphaGo defeated ShiShi Li who is one of the best human Go players in the world [1]. After that, the artificial intelligence neural network and deep learning become a hotspot in the society. Briefly, the artificial neural network is the network that imitates the human neural network. There are many neurons work with each other in human's brain, constituting the neural network. The neural network can help human think logically. Therefore, the artificial neural network can be regarded as the machine's brain that makes

machine have logic and do reason. Moreover, the machine can “think” faster and more precise than human. That is the reason a human cannot defeat machine in many fields like Go, Chess.

The artificial neural network can be divided into the feedforward neural network and the recurrent neural network. And there are many branches in the feedforward neural network, which is the perceptron, two-layer neural network and the multi-layer neural network [2]. The single layer neural network namely, the perceptron has one input layer and one output layer. There are many units in every layer, and the unit in the input layer can only transfer the data without computing. The output layer in the perceptron will calculate the date and output the result. The perceptron is like the model of logistic regression; it can only solve the simple linear classification task [3]. As for the two-layer neural network, except for the input layer and the output layer, there is another layer between the input layer and the output layer, called the hidden layer. In the two-layer neural network, the hidden layer is responsible for computing data. Normally, the layer’s number of the neural network depends on the number of the computing layer [4]. Therefore, the multi-layer neural network has many computing layers, that can make the network approach more complex function. Compared to the perceptron, the two-layer neural network and the multi-layer neural network can address nonlinear classification tasks very well [5].

Design

Part one

In part one, the program aims to read the data from the existing file. Moreover, based on the date, the agent needs to distinguish the input dataset and the output dataset and obtain them as well. After getting the output dataset and the input dataset, the agent will learn the pattern of these data. As a result, generating a neural network. By the network, the agent can match the output target by a piece of input dataset automatically.

As for the implementation, the first thing is that the program can read the content of a file because the data size used to train the neural net will be quite large. It will be less efficient if all train dataset inputted

in the program manually. The mechanism of the tool that used to read a dataset from the existing file is reading the content of the file and transform the content to the array by I/O stream in the program. The transform rule is the “Yes” corresponds “1”, and the “No” corresponds “0”. Furthermore, the different destination in the data will replace by an array like “{0,0,0,1,0,0}”. After that, the program will acquire an array representing the file data. There are two parts in the array, the input dataset and the output dataset. Therefore, the array needs to divide into two parts. Due to the information of the destination is at the last column of every piece data, the program will put the fore parameters into the input array, and the last parameter of every row will be added into the output array. The input array and the output array will be used to create the neural network. The neural network has three layers which are the input layer, a hidden layer and the output layer. There will be eight input units, and five output unites because there are eight features of destination and five destinations in the dataset of the file. The program will train the network by using the backpropagation algorithm. The training process will keep doing before the error of the training result does not reach the goal number, which is 0.01 in this program. The neural will be saved in the program root directory when the error of training result less than or equals the goal number. Next, the program can generally guess the destination based on the input dataset by the neural network saved in the previous procedure. The agent will load the neural network first, then the input dataset that represents the features of the destination will be added into the neural network to compute. There will be an output array with five values because of five destinations, and the value represents “the possibility of this destination is the user’s goal”. These values computed based on the pattern of the neural network saved advanced. Therefore, the agent will compare these values and get the max value which means it might be the possible goal destination. For example, the output dataset will be like this “{0.909, 0.003, 3.936E-5, 1.566E-8, 0.012}”. Thus the max value is 0.909 which is in the first column. Due to the destination of the file has been added into an array list, the index of the max value in the output array can also be used to get the destination in the array list. In brief, the array that represents the destination

“Spain” is {1,0,0,0,0}; and the max value of in the output array that computed by the neural network is in the first column; that means the possible goal destination is “Spain”.

Part two

Based on part one, the program for part two creates a window to interact with the user. JFrame implemented the window; there are some JLabel components involves the questions, which display to the user. Moreover, there are “Yes” and “No” checkboxes provided to users to select the preference features of the destination. The agent will make an early guess before users complete all questions, and if the guessed destination is wrong, users can submit the selection when they answer all questions, and the destination that conjectured by the agent will show up by a dialogue box. This interacts way will be more directly compared with showing questions by the terminal. Also, the user can enjoy a better interactive experience. As for the implementation, the tool that used to get the input dataset and output dataset is similar to the formal version, as well as the learning class that used to train the neural network. The significant improvement in the window thing. There is a new window class used to implement the interaction between the agent and the user. There are eight features for the destination selection. Thus there will be eight questions need the user to answer. The layout of the window is the absolute layout, and there are eight JLabel and sixteen Checkbox components in the window. Initially, creating these components and adding them in the window. The user will answer the question one by one. Therefore, there is only one question shown in the window because other components are set invisible. Checkboxes are added action listener; the next question will appear in the window when the user checked the previous question. Furthermore, to avoid users can check the “Yes” box and the “No” box at the same time, the checked box will be unchecked when users want to change another answer. Every answer for questions will give a value to the preference array which initialises when the window starts. There will be an early guess when the user completes the fifth question. At that time, the window will transfer the preference array to the agent. Then, the agent will regard preference array as the input array that used to compute

in the network. After that, the guess destination selected by computing result will show up in the window through a dialogue box, which implemented by the object JOptionPane. The content of the dialogue is like "Is your dream destination is Spain?" The user can click the "Yes" button; then the window will close. Alternatively, the user can click the "No" button and continue answering the rest questions. After the user finishes all the questions, they can submit the answer by clicking the button on the button of the window. The button will execute the mouse clicked method. In this function, the window will check that if all questions have been answered and if not, the alert dialogue will show up to notice the user to complete all question. Besides, the user has to give one "Yes" answer among the questions. Otherwise, the dialogue box will also appear. Apart from these two situations, the preference array that represents the user's answer will transfer to the agent to calculate with the neural network. At last, the guessed destination will be shown in the window like "Is your dream destination is Japan?" The user can click "No" to resubmit the answer or close the window by clicking the "Yes" button.

Part three

As for part three, the agent not only can guess the destination depends on the user's preference answer, it also can allow the user to add new destination based on the inputted preference array. Moreover, the new features for the new destination will be used to create a new output dataset, similarly, the input dataset will add into input dataset as well. By the new input dataset and the new output dataset, the agent will re-train the neural network using the new information and update the current network in the program root directory. The following guess for the destination will base on the new neural network. The introduction of the implementation will follow the user's operation order. At first, the interactive way is the same as part two; it uses the JLabel and JCheckbox to show questions and get user's answer. The agent for part three also can make an early guess before the user complements all questions. If the early guess does not conform to user's goal, the user can submit all answer after finishing all questions. The change for part two is that the program will ask that if the user wants to add a new destination when the

final guess destination also does not accord with the user's goal destination. The added window will show up if the user wants to add a new destination. There is a text field in the add window that used to input the new destination for users. Moreover, the adding event will active when the user confirms submitting the new destination. Initially, the agent will add the new destination into the external file used to reserve the destinations. There are only five destinations in this file which is "Spain, Greece, Egypt, Australia and Argentina", before users add new destination into the neural network. Thus, the new destination will enlarge this file continually. However, the destination cannot add to the network if the destination has existed in the network. In this case, the adding event will get the "false" returned value; the program will notify the user that the destination has existed. In the case that the user adds a "real" destination, the destination will add into the destination file, and the file will be used to get the name of the destination. After that, the agent needs to create the new input dataset and the net output dataset, which used to train the neural network. As for new input dataset, the agent will create a new input array that the length will be one larger than the original input array. Then, putting all data of the original input array into the new one, and adding the new features array obtained from the user at the end of the new input array. Next, creating the new output array, the new output array will be one larger than the original output array both in the vertical direction and the horizontal direction, because the array needs to lengthen itself to represent the new destination after initialising the new output array, adding the output data into the new one. Due to the new array has a longer length, the value of the redundant position equals "0". Also, adding the new destination at the end of the new output array. The array that the increasing position will equal to "1" and the other positions equal "0" will be used to represent the new destination. The agent can re-train the neural network using the new input and the new output and save the trained network. Moreover, the next predict will depend on the new network. To improve the neural network continually, as long as there is new data added into the dataset, the agent will update the input dataset and the output dataset

and save them in the files. Therefore, the array in these files can be regarded as the basic array when the user tries to add a new destination into the network.

Example and Testing

Part one

1. Getting data from the CSV file.

   Using the input stream read the content of the file. The parameter “filePath” in this function is the address of the input file. Moreover, the parameter “COLUMN_NUM” is the number of the column in the file. The data in every line will be added into an array first, and then the array will be added into an array list that used to give value to the two-direction array.

```
BufferedReader br = new BufferedReader(new FileReader(new File(filePath)));
String line = "";
ArrayList<String[]> lineList = new ArrayList<String[]>();
while ((line = br.readLine()) != null) {
	StringTokenizer st = new StringTokenizer(line, ",");
	String[] currCol = new String[COLUMN_NUM];
	for (int i = 0; i < COLUMN_NUM; i++) {
		if (st.hasMoreTokens()) {
			currCol[i] = st.nextToken();
		}
	}
	lineList.add(currCol);
}

String[][] str = new String[lineList.size()][COLUMN_NUM];
for (int i = 0; i < lineList.size(); i++) {
	for (int j = 0; j < COLUMN_NUM; j++) {
		str[i][j] = lineList.get(i)[j];
	}
}
```

2. Creating the input array and the output array.

   Initialising the input array and output array by the length of the array obtained in last function. Then making the rule of the transforming way for the destination. After that, dividing the array into input array and the output array. These arrays will be used to train the network.

```
input = new double[str.length - 1][str[0].length - 1];
output = new double[str.length - 1][5];
double[] Spain = { 1, 0, 0, 0, 0 }; double[] Greece = { 0, 1, 0, 0, 0 };
double[] Egypt = { 0, 0, 1, 0, 0 }; double[] Australia = { 0, 0, 0, 1, 0 };
double[] Argentina = { 0, 0, 0, 0, 1 };
```

```
for (int i = 1; i < str.length; i++) {
	for (int j = 0; j < str[i].length; j++) {
		if (str[i][j].equals("Yes")) {
			input[i - 1][j] = 1.0;
		} else if (str[i][j].equals("No")) {
			input[i - 1][j] = 0;
		} else if (str[i][j].equals("Spain")) {
			output[i - 1] = Spain;
		} else if (str[i][j].equals("Greece")) {
			output[i - 1] = Greece;
		} else if (str[i][j].equals("Egypt")) {
			output[i - 1] = Egypt;
		} else if (str[i][j].equals("Australia")) {
			output[i - 1] = Australia;
		} else if (str[i][j].equals("Argentina")) {
			output[i - 1] = Argentina;
		}
	}
}
```

The following codes can test the content and the length of the input array. The output array can be tested by the same way.

```
Training_Table tt = new Training_Table();
tt.getCsvData("trip.csv", 9);
double[][] INPUT = tt.getInput();
for(int i=0;i<INPUT.length;i++) {
	System.out.println(i + " ");
	for(int j=0;j<INPUT[i].length;j++) {
		System.out.print(INPUT[i][j]);
	}
	System.out.println();
}
```

3. Returning the guessed destination.

Finding the max value of the output array and record its index. The index will be used to find the destination in the array list.

```
double max = output.getData(0);
int maxSign = 0;
for (int i = 0; i < output.size(); i++) {
	if (output.getData(i) > max) {
		max = output.getData(i);
		maxSign = i;
	}
}

ArrayList<String> cityArray = new ArrayList<String>();
cityArray.add("Spain"); cityArray.add("Greece");
cityArray.add("Egypt"); cityArray.add("Australia");
```

```
cityArray.add("Argentina");
String destination = cityArray.get(maxSign);
```

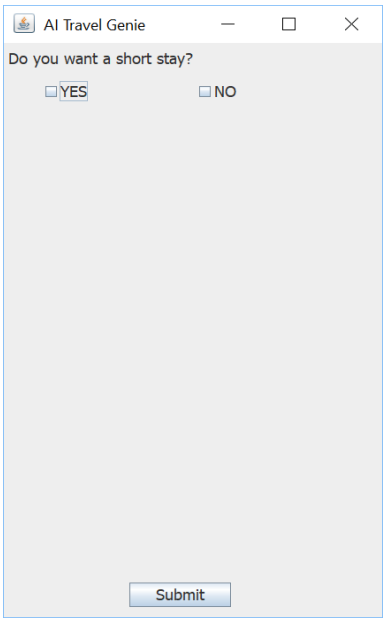
Part two

1. Collecting the user's preference data.

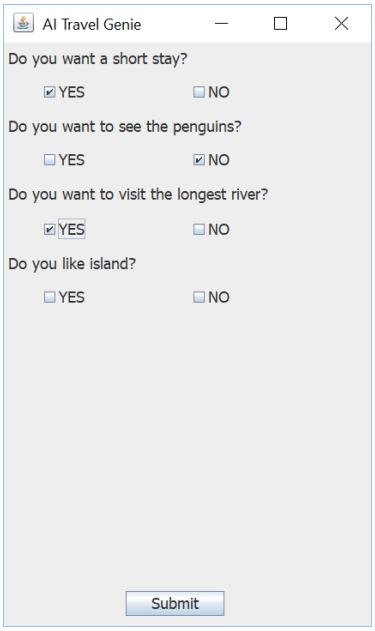
   In this section, it will mainly show the interaction window and some implementations. The initial window will look like the picture 1-1 when the program is running. Due to the Jcheckbox will change the next question's visibility, all questions invisible at first, but the first one. The next question will show up after the current question finished. Besides, once one of two checkboxes is selected, another one will cancel the selected state. The following code is for the case that the user selects "No" answer. Thus, the preference value is "0". If the user selects "Yes", this value will be "1".

```
No_1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) {
	if(No_1.isSelected()) {
		if(Yes_1.isSelected()) {
			Yes_1.setSelected(false);
		}
		preference[1] = 0;
	}

		if(!t3.isVisible()) {
		t3.setVisible(true);
		Yes_2.setVisible(true);
		No_2.setVisible(true);
		}
	}
});
```



1-1

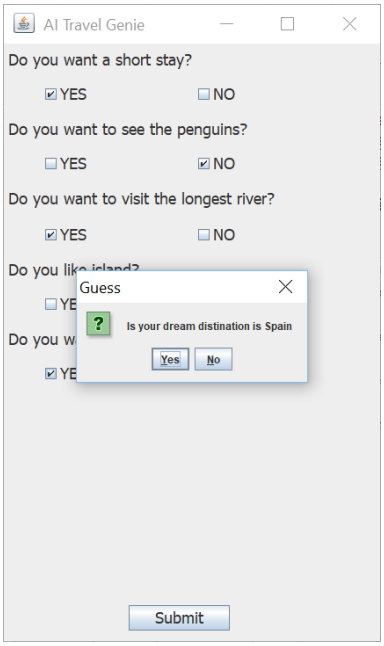


1-2

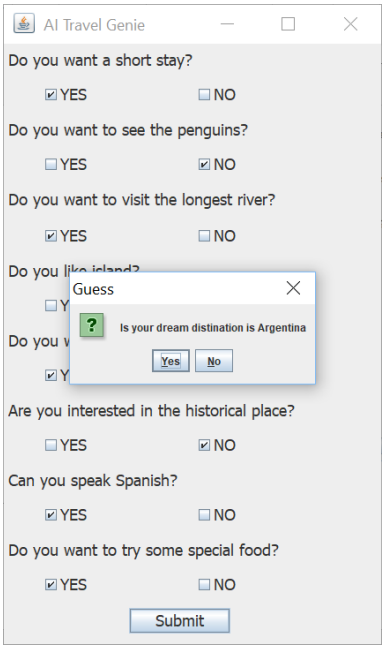
2. The early guess and the final guess.

   The program will make an early guess when the user finishes the fifth question. At this time, the last three positions of the preference array will be “0”. Then, the agent will use this preference array compute with the network, return the guessed destination to the user like the picture 1-3 shows. If the destination does not conform to the user’s goal, the user can click the “No” button and continue completing rest questions. Moreover, clicking the “Submit” button to give the agent the whole preference array to calculate, and the guessed destination will show up. The fowling code is the program’s performance when the user answered the fifth question.

```
agent.guess(preference);
int option= JOptionPane.showConfirmDialog(
Window.this, "Is your dream distination is " + agent.getDestination() , "Guess
",JOptionPane.YES_NO_OPTION);
if(option == JOptionPane.YES_OPTION) {
	dispose();
}else {
	if(!t6.isVisible()) {
	t6.setVisible(true);
	Yes_5.setVisible(true);
	No_5.setVisible(true);
	}
```



1-3



1-4

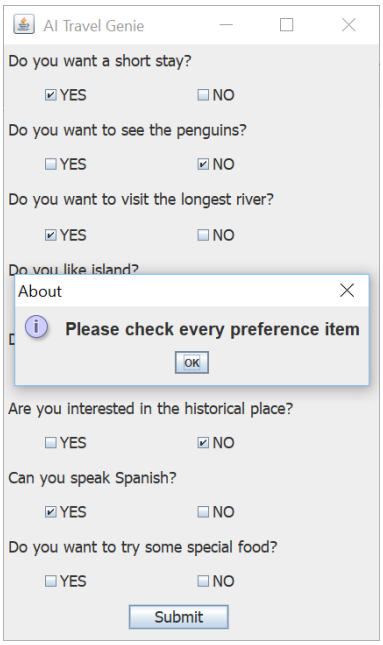
3. The validation for the user's answer.

   To avoid the user submits the answer without completing all questions, the program will test that if all questions have the corresponding answer. The following codes will implement that function. The program will add all checkboxes into an array list, and figure out the number of the checked box. If the number of the box is "8" which means every question has one answer, the agent will execute the guess function. Otherwise, users will get the notice that they should complete all questions like the picture 1-5 shows.

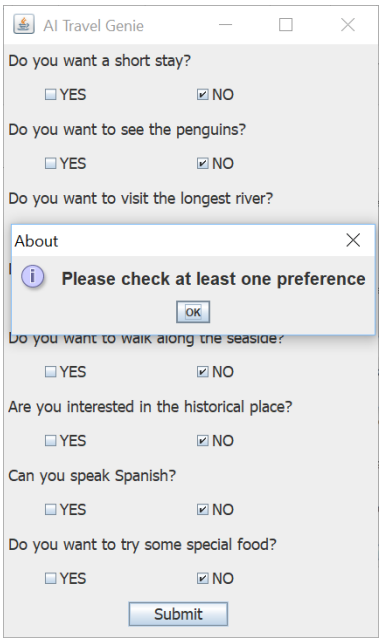
```
int checkNum = 0;
for(int i=0;i<checkList.size();i++) {
	if(checkList.get(i).isSelected()) {
		checkNum++;
	}
}
if(checkNum != 8) {
JLabel label = new JLabel("Please check every preference item");
label.setFont(new Font("Arial", Font.BOLD, 20));
JOptionPane.showMessageDialog(null,label,"About",
JOptionPane.INFORMATION_MESSAGE);
}
```

The user cannot choose “No” answer for all questions as well. Thus, the program will figure out the number of the “No” answer. If all answers are “0”, users will get the message that they need select at least one preference item. The example is the picture 1-6.

```
double test = 0;
for(int i=0;i<preference.length;i++) {
	if(preference[i]==0) {
		test++;
	}
}
if(test == 8) {
	JLabel label = new JLabel("Please check at least one preference");
	 label.setFont(new Font("Arial", Font.BOLD, 20));
	 JOptionPane.showMessageDialog(null, label, "About",
JOptionPane.INFORMATION_MESSAGE);
}
```



1-5



1-6

Part three

1. Loading the input dataset and the output dataset.

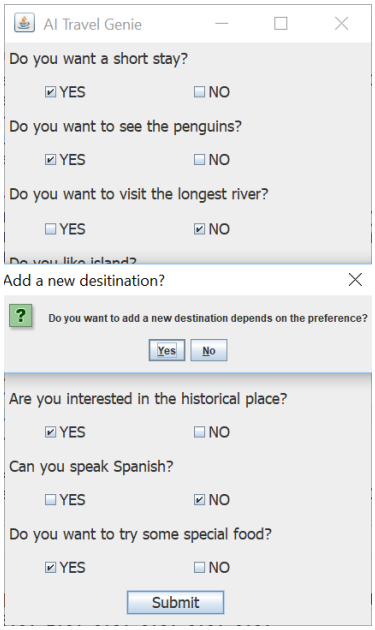
The program will load the input dataset and the output dataset from the external file. The method is similar to the method for getting data from the CSV file. At first, the character string will be added into the array list by input stream. Then, creating a new array based on the length of the array list. Transforming the data format from String to Double and adding it into the two-direction

array. The code shows the example that getting the output array, and the method for input array is similar to this. After that, the program can train the network and guess the destination by the user's preference.

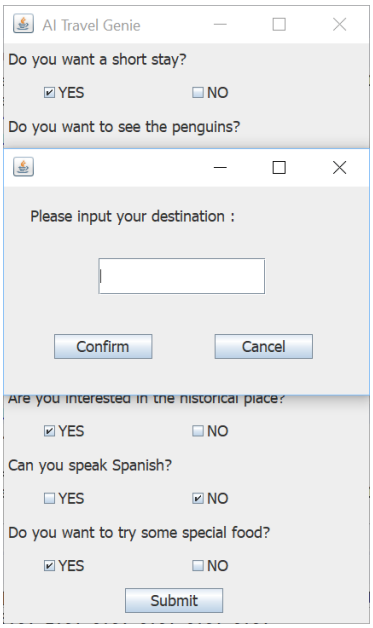
```
public double[][] getNewOutput(String file) throws IOException{
File f = new File(file);
BufferedReader br = new BufferedReader(new FileReader(f));
String line = br.readLine();
ArrayList<String> new_output = new ArrayList<String>();
while (line!=null) {
	new_output.add(line);
	line = br.readLine();
}
double[][] output_array = new
double[new_output.size()][new_output.get(0).length()/4];
for(int i=0;i<new_output.size();i++) {
	String[] piece = new_output.get(i).split(" ");
		for(int j = 0; j<piece.length;j++) {
			output_array[i][j] = Double.valueOf(piece[j]);
		}
	}
	br.close();
	output = output_array;
	return output;
}
```

2. Testing the new destination.

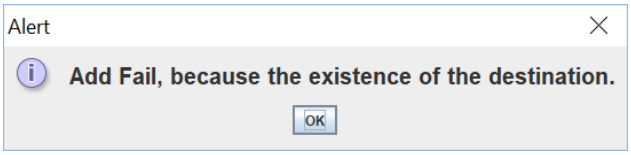
The program will ask users if they want to add a new destination like the picture 1-7, and they can click the "Yes" button to enter the add window like the picture 1-8 shows. Before adding the new destination into the network, the program has to test if the destination got from the user has existed in the neural network. To test this, the program will load the existing destination from the file and store the destination in an array list. Then, if the new destination has existed, the adding event will fail and return the failure message like the picture 1-9. Alternatively, the new destination will be written in the file, and the program continues to create the network by the input array and output array that obtained in the next step.



1-7



1-8



1-9

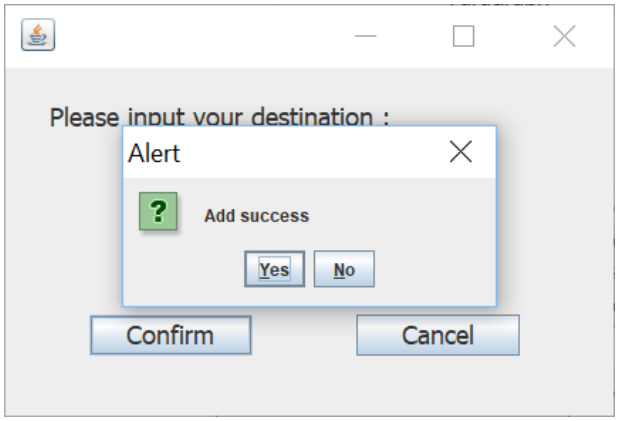
```
BufferedReader bf = new BufferedReader(new FileReader("output.txt"));
String str = bf.readLine();
String[] city = str.split(",");
for(int i=0;i<city.length;i++) {
	cityArray.add(city[i]);
	}
bf.close();

public boolean addNewCity(String city) {
System.out.println("added city "+city);
for(int i=0;i<cityArray.size();i++){
	System.out.print(cityArray.get(i) + " ");
}
if(cityArray.contains(city)) {
	return false;
}else {
	cityArray.add(city);
	try {
	PrintWriter  pw = new PrintWriter(new FileWriter("output.txt", true));
	pw.print(","+city);
	pw.flush();
	} catch (IOException e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
	}
	return true;
	}
```

}

3. Creating new input dataset and new output dataset

The program will add the preference acquired from the user into the original input array to create a new input array. Also, the new output array will be created by combining the new destination and the original output array. The following codes implement these functions. The new input dataset and new output dataset can be used to train the new network, and the user will get the message if the new destination is added successfully like the following picture shows.



1-10

```
public double[][] addNewCity_Input(double[] newInput, double[][] input){
double[][] transferInput = new double[input.length+1][input[0].length];
for(int i=0;i<transferInput.length-1;i++) {
		for(int j=0;j<transferInput[0].length;j++) {
			transferInput[i][j] = input[i][j];
		}
	}
	for(int i=0;i<transferInput[transferInput.length-1].length;i++) {
		transferInput[transferInput.length-1][i] = newInput[i];
	}
	input = transferInput;
	setInput(input);
	return input;
}

public double[][] addNewCity_Output(double[][] output){
double[][] transferOutput = new double[output.length+1][output[0].length + 1];
	for(int i=0;i<transferOutput.length;i++) {
		for(int j=0;j<transferOutput[0].length;j++) {
			if(i==output.length) {
				for(int y = 0;y<transferOutput[0].length;y++) {
					if(y == transferOutput[0].length-1) {
						transferOutput[i][y] = 1;
					}else {
```

```
                                        transferOutput[i][y] = 0;
                                }
                        }
                }else {
                        try {
                                transferOutput[i][j] = output[i][j];
                        } catch (Exception e) {
                                // TODO Auto-generated catch block
                                transferOutput[i][j] = 0;
                        }
                }
        }
    }
    output = transferOutput;
    return output;
}
```

4. Re-star training

   Duo to the weight value will be random at the beginning of the training, sometimes, the training process will last a long time. To avoid the user has a long wait during the training procedure, the program will re-start the training if the network does not complete after 5000-time iteration.

```
int epoch = 1;
do {
        train.iteration();
        epoch++;
        if(epoch>5000) {
                creatTable(INPUT, OUTPUT);
                return;
        }
} while(train.getError() > 0.01);
```

5. Save the input dataset and the output dataset

   The program will save the latest input dataset and the output dataset when the window closed. This is for the updating the programs' database when the user adds new destinations. Thus, the program can load the new database to create the neural network. The array will be written in a file to save by output stream.

```
PrintWriter p_input = new PrintWriter(new FileWriter("new_input.txt",false));
double[][] new_input = agent.getCsv.input;
for(int i=0;i<new_input.length;i++) {
        for(int j=0;j<new_input[i].length;j++) {
                p_input.print(new_input[i][j] + " ");
        }
                p_input.print("\r\n");
}
p_input.flush();
```

```
p_input.close();
```

Evaluation

The program has a good interactive way that makes user can operate it directly. Moreover, the program can load the initial database from the external file automatically rather than inputting it manually. Moreover, with the user adds a new destination into the program, the program can update the dataset as well. By this way, the more data that used to train the neural network, the more accurate guessed destination will be. However, the program cannot allow the user to add a new feature for the destination. Furthermore, the training process of the network will stop when the error less than the 0.01, but sometimes the result might not be accurate enough when the error value only less than 0.01. However, it will consume more computing time if the network wants to get the more precise result.

Running

There three parts in the submitted folder, every part is an individual folder. The part one folder involves two Jar files, Learning1.jar is for training by the original trip file. Moreover, Learning2.jar is trained based on the trip-new file which adding a new destination and a new feature compared to the trip file. The program can be run by the command line like this: java -jar Learning1.jar. Also, the trained network will also be saved in the same folder. As for part two, the program can be run by the command line: java -jar Learning3.jar. The user will get a window during the program is running. Users can select the checkbox for their answer to interact with the program. In terms for the part three, the running command is similar to others, java -jar Learning4.jar. There are three text files in part three, which is used to store the destinations and update the input dataset and the output dataset.

Bibliography

1. Chouard T. The Go Files: AI computer wraps up a 4-1 victory against the human champion. Nature News. 2016 Mar.

2. Bebis G, Georgiopoulos M. Feed-forward neural networks. IEEE Potentials. 1994 Oct;13(4):27-31.
3. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review. 1958 Nov;65(6):386.
4. Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning. Cambridge: MIT press; 2016 Nov 18.
5. Michalski RS, Carbonell JG, Mitchell TM, editors. Machine learning: An artificial intelligence approach. Springer Science & Business Media; 2013 Apr 17.