ST.ANDREWS UNIVERSITY COMPUTER SCIENCE

Practice-4 20Q Game

120014670

11/30/2012

Table of Contents

Part-1

Aim of Part-1

This part the of project aims to, design a neurol network like 20Q game using java programming language and Neuroph library. Input output and hidden neurons will be fixed in number. The program asks certain number of questions to guess the animal. It is not able to learn new concepts.

Neural Networks

"An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information." [1] The structure of ANN mirrors the brain. There are highly interconnected processing neurones working in unison to solve specific problems.(see Figure 1) ANNs are able to learn by example just as people do. ANN can be design to use different purposes such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well. [1]

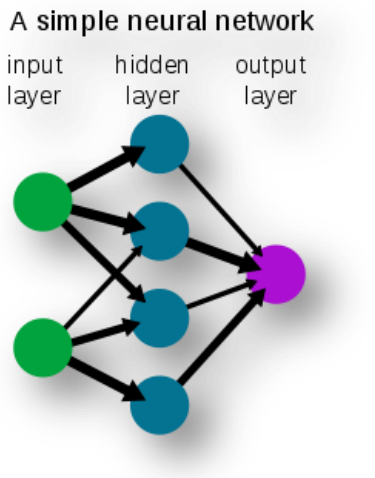

Figure-1 [1]

Design of Neural Network

Generalization and classification is curial to implementing the20Q game with neural network. Generalisation is the ability to train with one data set and then successfully classify independent test sets. [2] The 20Q game is designed to guess animal names in my project. Therefore, questions should be asked so that perceptrons will be able to separate every animal. This part of the report will explain the design decisions for classifying concepts (animals) correctly.

Why Is Multi-Layer Network Is Required?

In figure 2, perceptron is drawing a line across the 2-d input space. Inputs to one side of the line are classified into one category; inputs on the other side are classified into another.

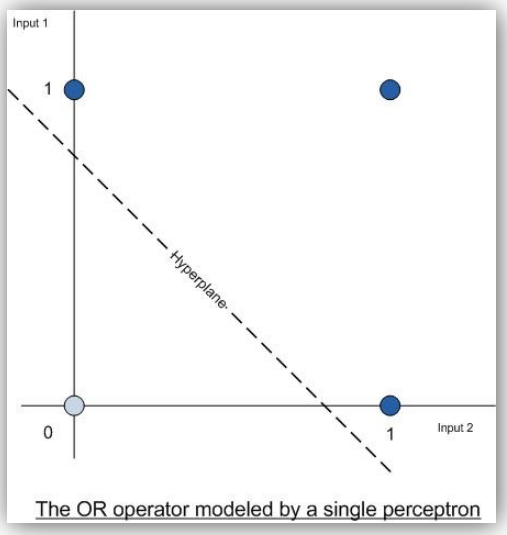

Figure-2 (http://takinginitiative.net)

Zero hidden layers are required to resolve linearly separable data. Implementation of the 20Q game with single hyper plane is not possible. Because, in the 20Q game there are lots of concepts (animals in our case) requiring classification. In this case, design like Figure-3 is required.

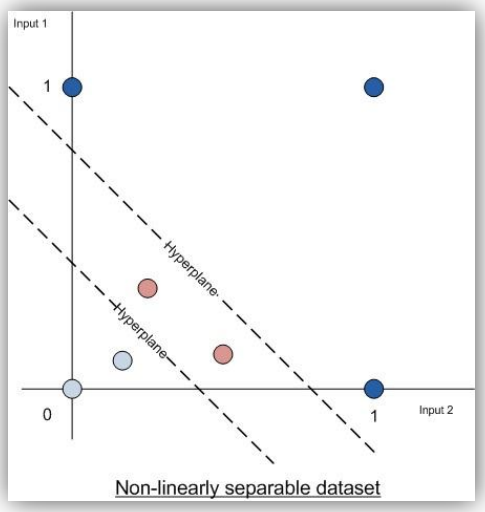

Figure-3 (http://takinginitiative.net)

In figure-3 we need at least two hyper-planes to solve this problem.(Imagine that every dot in graph represents one animal.). This requires us to link up these neurons together, to link them up we'll need shared inputs and outputs in other words we need to use multilayer neural network.

Pre-Processing and Input

In this section displays from Neuroph Studio software will be used in order to justify design decisions. Software initially starts with 8 animal concepts and 8 questions. Every question has two possible answers (yes/no) and every question helps in the classification of the animals. Therefore, each question and animal represented with one neuron.

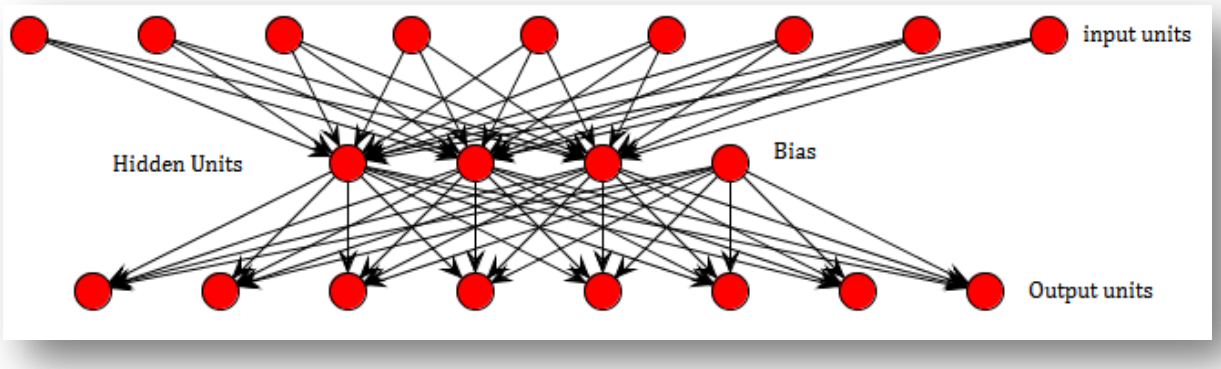

Figure-4 Neuroph Studio initial statement with 8 questions(input units) and 8 animals(output units).

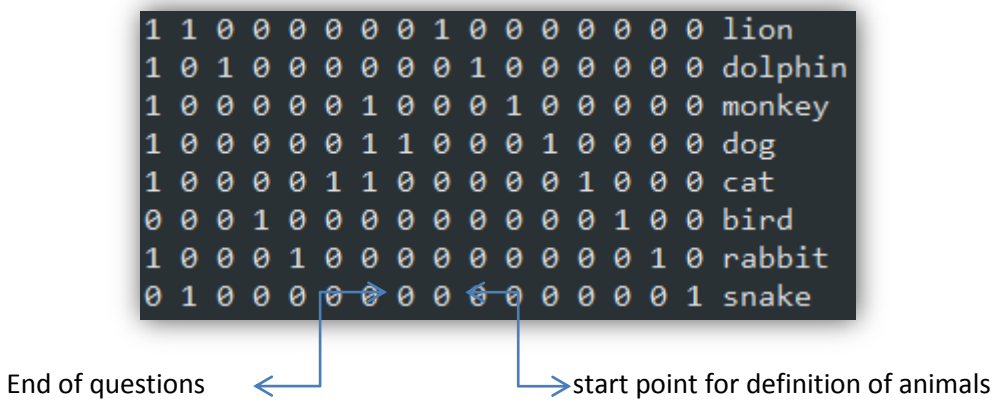

Figure-5 pairings table (animal_training.txt)

Figure-5 shows pairing tables for defining animals and specifying which output unit represents which specific animal. The first 8 bits represents input patterns to define 8 different animals. For instance, in figure 5 in first line users answers "yes" for first and second questions and "no" for the rest of the questions. In this case, network should point out first output neuron. Corresponding name for first output neuron is "lion" according to the table. However, if network is not well trained it can't give corresponding animal names to the user because it will be pointing to another output neuron. This part will be explained more detailed in fallowing sections.

Deciding number of Hidden Units

Determining the number of hidden units is one of the crucial steps of designing a neural network. If an inadequate number of neurons are used, the network will be unable to model complex data, and the resulting fit will be poor and network can't be trained.

If too many neurons are used, the training time may become excessively long, and, worse, the network may over fit the data.

Lots of different papers advocate different approaches to determine number of hidden units in neural network. This part will present these perspectives for determining for hidden layers and try to justify them using Neuroph Studio and project itself.

The first approach advocates that, every problem requires an unique network design. The number of hidden neurons should be determined by a trial and error method at the time of its training. [3]

On the other hand, Phil Picton author of "Neural Networks" devises a formula for calculating the number of appropriate hidden neurons. The Formula is as fallows;

$n \geq \log_2(input)$ [4] Input is number of input neurons and n represents number of output neurons. This method is used in part-1 to calculate number of hidden neurons. However, in Part 2 another approach has been used because this formula does not consider cases of increasing number of output neurons (learning new concepts).

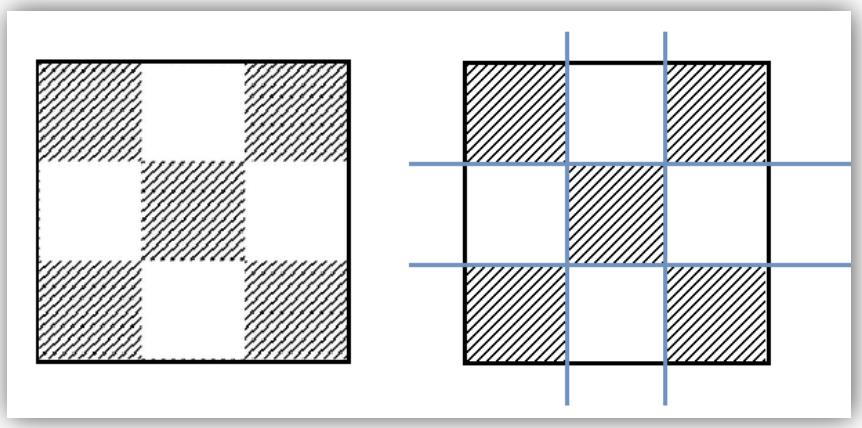

Figure-6 separation of black and white (worse case)

Figure 6 shows and example where Picton's formula could be applied to find how many hidden neurons required to separate black and white in worse case. Figure 6 represents worse case because, in order to separate black and white on board, maximum number of hyper planes required.

According to figure 6: Input number =9; $n \geq \log_2 9$ (round the result) shows that we need at least 4 hidden neurons for the classification of black and white. This example shows that, Picton's formula could be used in this case. In Figure 6 input and output neurons are directly proportional. However, this formula is not applicable for part-2 because input neuron numbers are fix and output neurons are changing with addition of new concepts.

Another approach advocates that the number of hidden neurons should be bit more than average of input and output neuron numbers. Formula for this is;

(Number of inputs + outputs) x 2/3

According to third approach; having a few more hidden neurons than sum of input and output neurons does not have dramatic impact on network and it is possible to prune it later by experience. Additionally, there are some techniques to prune network proposed by Giovanna Castellano. [5] However, I used trial and error approach with this formula to calculate hidden neurons. Detailed information can be found in part 2.

Neuroph Studio Tests

In this part the importance of deciding the number of hidden units will be shown using Neuroph Studio software. All other network parameters remained same. Network designed for 8 questions and 8 animals.

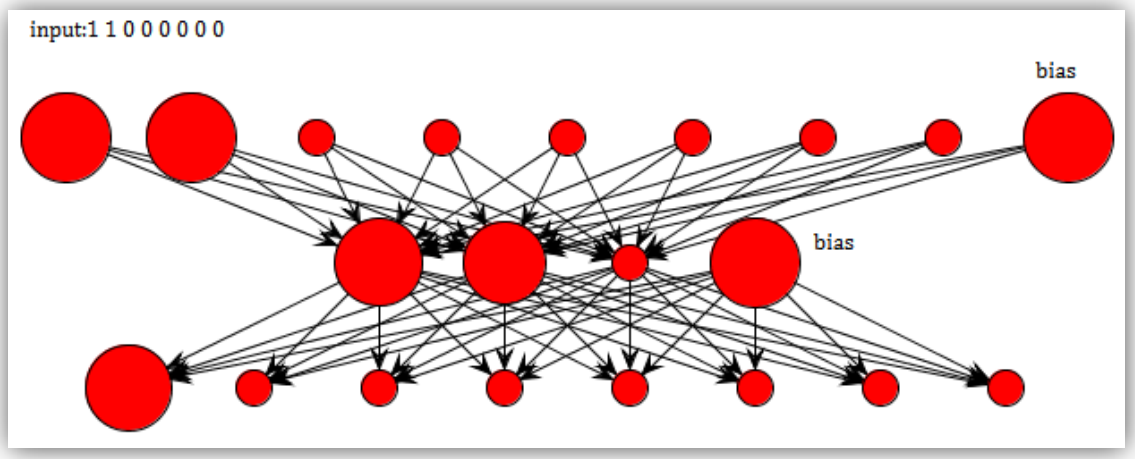

Figure-7

If the network is trained with ideal number of hidden units it can easily classify outputs as it is expected in training set. Figure 7 for input 1 1 0 0 0 0 0 0 network points out the first output. The value for first output is 0.9621 which is really close to 1. This shows that, although network well trained with right amount of hidden units it is not perfect .(error 0.9621-1=-0.0379)

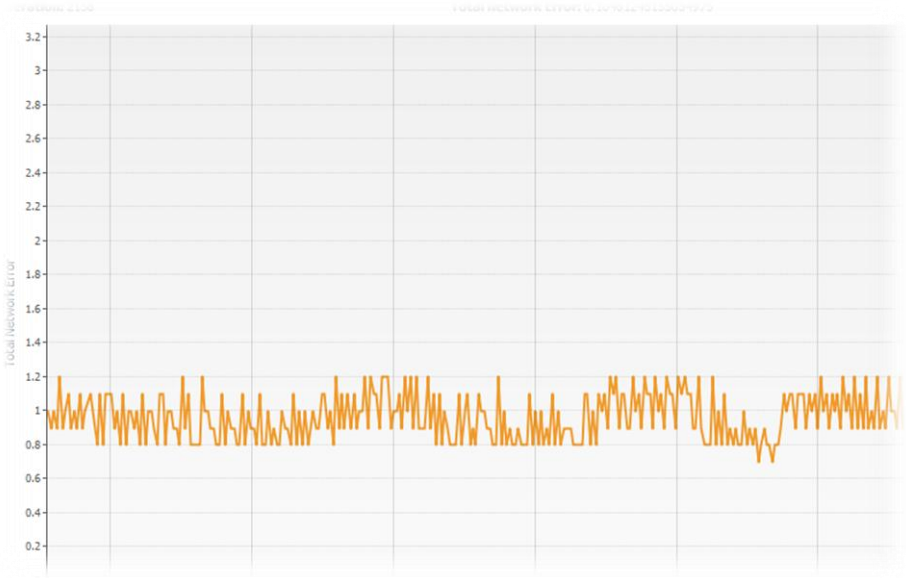

Figure-8 same network attempt to train with 2 hidden units resulting with infinite training period. (Iteration limit = unlimited).

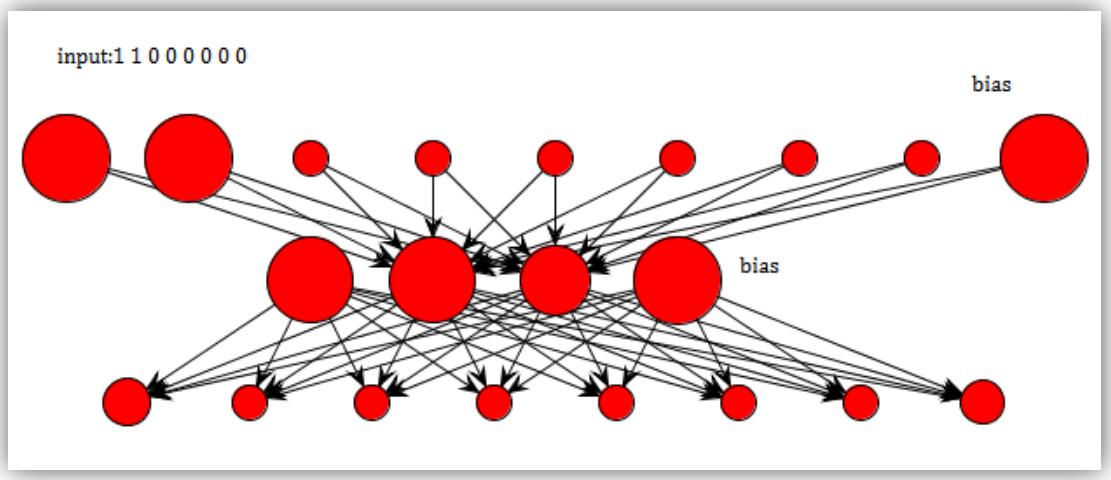

Figure-9 – Network with two hidden neurons(although it looks 3 one of them is not connected)

If there are an inadequate number of hidden units in network, it will be unable to model complex data. Figure 9 shows that all outputs have almost same activation size which disables classification of animals on this network. Thus, numerically error is larger than Figure 7. (-0.7654)

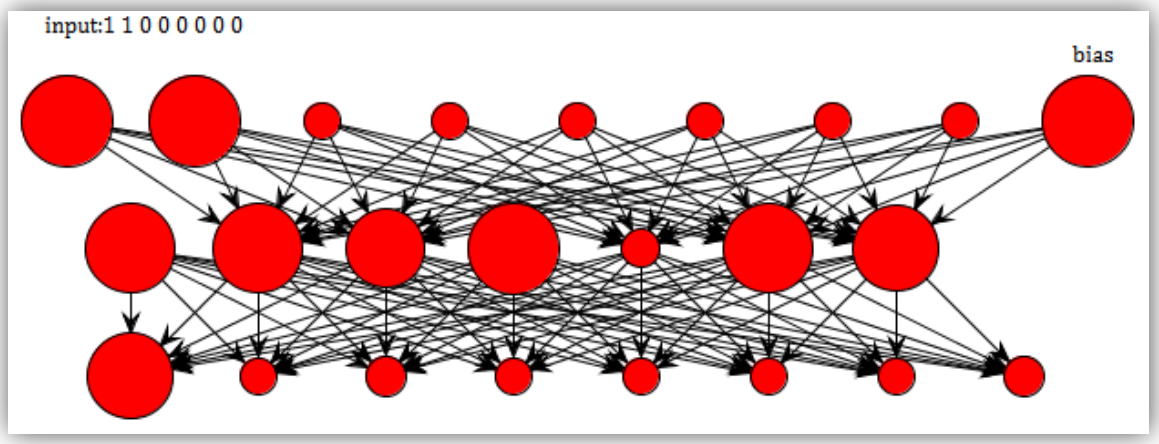

Figure-10 network with 6 hidden units. Activation size looks same with figure 7 however error rate is larger.(error : -0.085)

Deciding number of output units (concepts)

In the case of optimum classification network is able to have 2^n number of outputs (n is the number of input neurons). For example, it is possible to create 3 questions so that it will classify 8 animals perfectly. It can not represent more than 2^n because there will not be any possible input pattern for 9^{th} animal. However, I could not create this optimum case because there are lots of animal possibilities whose classification are really close to each other. Therefore, I have 8 animals which are initially classified according to 8 questions. In order to classify them, more

detailed information need to be asked. This is possible by implementing a decision tree. Decision trees will be discussed in part 2.

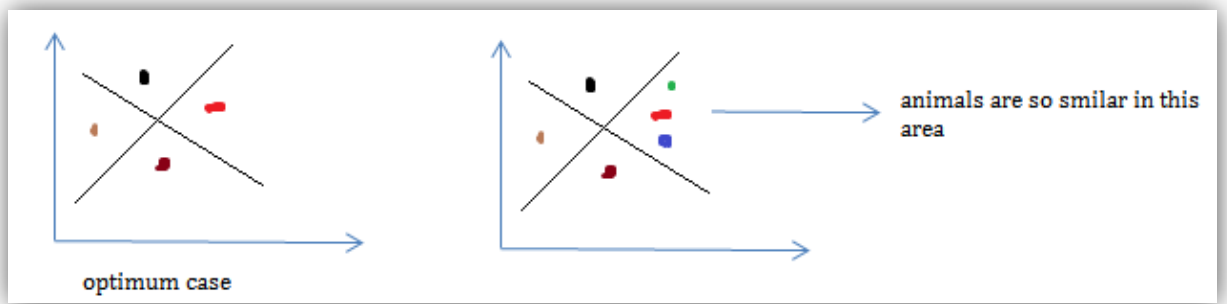

Figure-8 classification of animals . Figure shows optimum situation and non optimum situation. Non optimum situation requires more hyper plane.

Implementation

Classes

Concept – this class created to hold information about animal. Fields are name of animal, corresponding input pattern and output neuron number.

Data – this class includes set of variables which are parameters for project and neural network.

Guess- findmax() method gets network output array as an parameter and choses the maximum value.(this value have to be bigger than threshold also) then guessanimal() method checks whether its first guess of program or not. If it is first guess, it sends index information to main program so that program can display animals name, otherwise guessanimal() checks for other possible values bigger than threshold and send to main class.

Game- this is main class of project. Includes loops to get user input, show results and send data to related methods. Thus, loads weights from existing files.

AnimalNetwork- Includes neural network. Construction of class reads questions.txt and animal_training.txt file, determine number of input and output neurons using initializeNeurons() method and initialize questions calling initializeQuestions() method. Calculates hidden neurons using Picton's formula. Then trains network using its variable parameters according to values in animal_training.txt by calling initializeTrainingSetFromText() method and saves network to project folder. Training method is back-propagation which is provided by neuroph library. askNetwork() method gets answers to questions from user in main class and sends them to network. Network evaluates values and gives output according to its training.

Training

In this part of report I am going to show how training parameters in back-propagation method such as learning rate max error will be determined by using neuroph studio. Thus, behavioural differences with training and without training will be shown using the project.

Back-Propagation

Backpropagation method is the basis for training a supervised neural network. The output is a real value which lies between 0 and 1 based on the sigmoid function. There is a backward pass of error to each internal node within the network, which is then used to calculate weight gradients for that node. The actual outputs are compared with desired outputs in training set and the errors are calculated. The errors are then propagated back to the network in order to adjust the weights. [6]The training process repeats till the desired outputs are obtained or the error reaches an acceptable level.(in our case its iterate whether it reaches max iteration number or desired max error.) Neuroph library provides us to implement backpropagation by calling .learn() method by default. Therefore, it does not require explicit implementation.

Momentum

The momentum used for speeding up training in very flat regions of the error surface suppresses weight fluctuation of the valley shape. Momentum set as 0.7 for part 1. This value is found trial and see methods on Neuroph Studio.

Learning Rate

As a part of back-propagation algorithm the learning rate, is used to adjust the size of weight changes. In other words, learning rate could be considered as step size to adjusting error in each iteration. Learning rate is crucial to optimizing network because it is used for calculating gradient which helps network to reach global minima.(see figure 9)

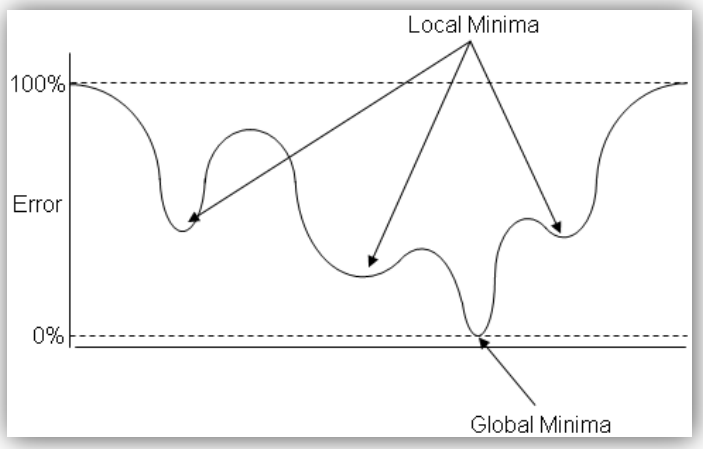

Figure-9 global minima is optimum point(minimum error) for network.

Hessian matrix calculation could be used whereas objective function is quadratic, as in linear models [7]. I use trial and error method to determine my learning rate in part 1. However, in part 2 program is capable of changing its learning rate in certain cases. This will be explained in

fallowing sections. In figure 10 shows a demonstration with learning rate 0.0.4 max error 0.01 (which is a point to stop for training actually) and iteration number is 4000 (after 4000 training will terminate regardless of other parameters.)

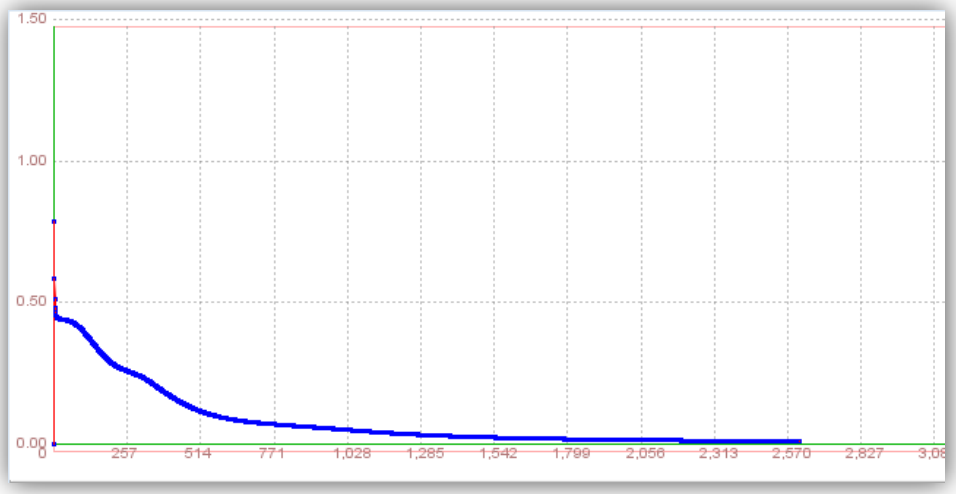

Figure-10 snapshot of error graph from Neruoph Studio

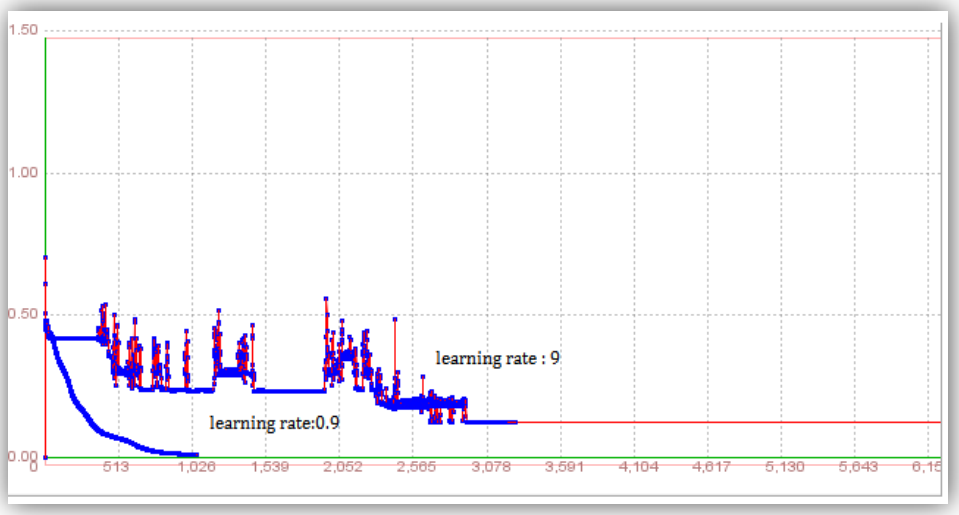

Figure-11 difference between two different learning rate observed with Neuroph Studio. One representation of network continues to training.

In figure 11 it can be seen that, high learning rate makes learning impossible.

For the Shapes in figure 10 and figure 11 its crucial to have smooth lines rather that fluctuation in figure 11. A fluctuation in shape cause training to stuck in local minima (see figure 9).

Maximum Error

Maximum error is a constrain on network to determine when to stop training. It should be set to a reasonable number in order for network to reach maximum performance. If it is set to exceedingly high value, network stops training before reaching global minimum.(see figure 12) If it is set to an unacceptably low number, network will not able to reach the training stoppage point.

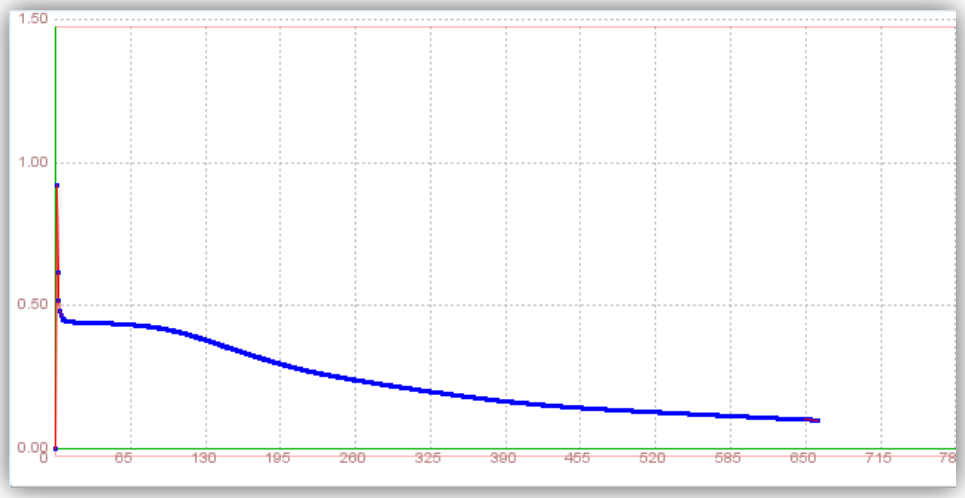

Figure 12 Training stops early (max error parameter 0.10)

Effect of training

This section will observe how network behaves with and without training using project itself.

Monkey is selected as an sample animal.

If variable LEARN in data class set as false and network file is deleted (to prevent project to load weights)

First example with learning ;

```
network is created
number of input units:8
number of output unit:8
number of hidden units:3
learning in progress...
total error:0.009993055
learning finished
save is completed
Is it a mammal? (y/n) y
Is it wild ?  (y/n) n
Does it primarily live in the sea ? (y/n) n
Can it fly ? (y/n) n
Does it have cyclamen ? (y/n) n
Does it purr ? (y/n) n
Is it omnivor ? (y/n) y
Does it barks ? (y/n) n
```

```
Is it monkey? y
Thanks for playing
do you want to play more? (y/n)
```

Same input is given not trained network this time ; Sample animal is monkey.

```
network is created
number of input units:8
number of output unit:8
number of hidden units:3
save is completed
Is it a mammal? (y/n) y
Is it wild ?  (y/n) n
Does it primarily live in the sea ? (y/n) n
Can it fly ? (y/n) n
Does it have cyclamen ? (y/n) n
Does it purr ? (y/n) n
Is it omnivor ? (y/n) y
Does it barks ? (y/n) n
Is it lion?
```

Although the user has entered the same input pattern for both networks, the untrained network was unable to guess correct animal.

I/0

In the first example below, the object concept is 'cat'. The network has not been trained before. Therefore, network will save itself under project folder.

First object is cat;

```
network is created
number of input units:8
number of output unit:8
number of hidden units:3
learning in progress...
total error:0.009998416
learning finished
save is completed
Is it a mammal? (y/n) y
Is it wild ?  (y/n) n
Does it primarily live in the sea ? (y/n) n
Can it fly ? (y/n) n
Does it have cyclamen ? (y/n) n
Does it purr ? (y/n) y
Is it omnivor ? (y/n) y
Does it barks ? (y/n) n
Is it cat?
```

Object is rabbit;

```
number of input units:8
number of output unit:8
```

```
number of hidden units:3
load is completed
Is it a mammal? (y/n) y
Is it wild ?  (y/n) n
Does it primarily live in the sea ? (y/n) n
Can it fly ? (y/n) n
Does it have cyclamen ? (y/n) y
Does it purr ? (y/n) n
Is it omnivor ? (y/n) n
Does it barks ? (y/n) n
Is it rabbit?
```

Third object is reptile;

```
number of input units:8
number of output unit:8
number of hidden units:3
load is completed
Is it a mammal? (y/n) n
Is it wild ?  (y/n) n
Does it primarily live in the sea ? (y/n) n
Can it fly ? (y/n) n
Does it have cyclamen ? (y/n) n
Does it purr ? (y/n) n
Is it omnivor ? (y/n) n
Does it barks ? (y/n) n
Is it rabbit? n
Is it snake?
```

This last attempt, 'reptile', have not previously been introduced to the program. Therefore, the program guesses which animal it is. Program asks "is it rabbit?" if user says n(no) to this question, the program eliminates that possibility and presents its second possible animal to user – which is 'snake'(closer to reptile). The questions in Part-2 are selected to be more discriminative in order to indicate the importance of classification with questions.

Part-2

Aim of part 2

This part of the project aims to extend part 1. The project program in this part is able to:

- Updated classifications which enables program to discriminate animals better.
- Learn new concepts
- Verify Classifications
- More scalability
- Generating questions from attributes
- Adjusting network parameters automatically

Question set-answer pairings

Generating Questions

Unlike Part 1 above, the program in this section is able to create questions from attributes. Therefore, some questions might be grammatically incorrect in natural language. Attributes are listed in attirbutes.txt file with a number which specifies what kind of interrogative particle should be add to generate the question. This feature might be improved such that it will enable the program to ask increasingly specific questions. For instance, let's assume that there is an attribute "extinct". The program in this case asks "is it extinct?" If the user answers as yes to this question, following questions might be asked include "*Was* it fast?" instead of *"Is* it fast?".

Learning new concept

After a series of questions and answers, the program tries to guess the concept. If it cannot manage to guess correctly, it asks the user to enter this new concept. Once the user enters the new concept, the verification process checks for a few criteria (which will be explained in next section) and puts the new concept into table (animal_training.txt). After this, the output units are increased and hidden units are calculated based on the formula explained in Part 1. However, the network also checks whether the new number of hidden units are suitable. This will be discussed in the 'Adjusting Network and Retrain' section below.

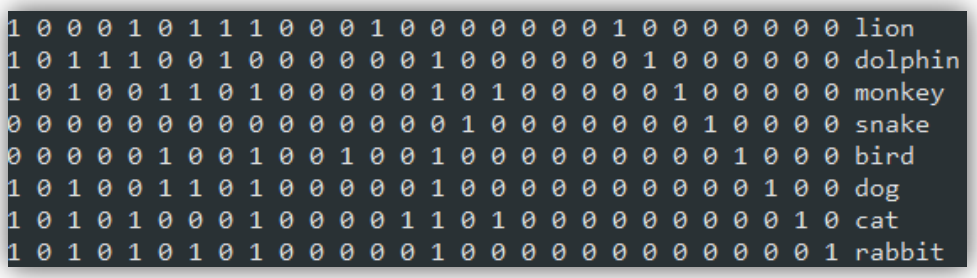

Figure 13- initial table with 8 animals. This part asks 20 questions to have better classification of animals.

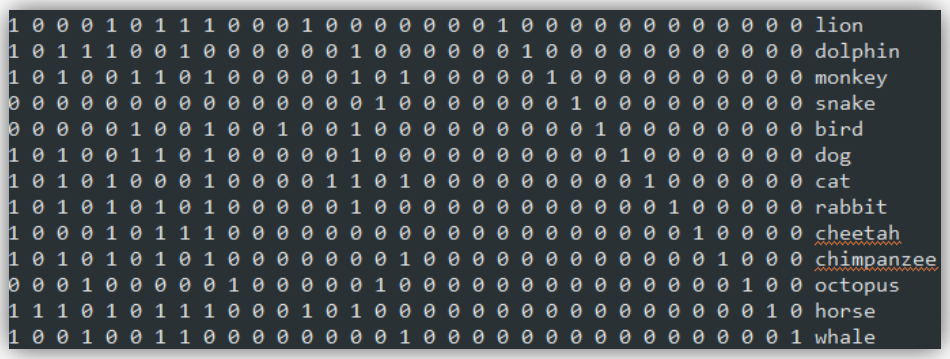

Figure 14- after series of playing 5 concept added to the system.

Classification Verification

Verification completes these tasks;

- Checks whether the user has entered the correct input for a specified animal. Sometimes the user is mistaken about a particular attribute of the intended animal. (e.g. The user might be mistaken whether a particular animal is a ruminant or not) In this case, the program won't able to guess the correct answer and asks user to enter the animal's name. If that animal exists in programs table, the program compares input patterns and notifies the user if it conflicts with her input.
- Change animal name in case the same input pattern is entered to the system. Neural network is confused if the same input pattern is entered for different outputs. This case disables the network from completing its training. In order to prevent this problem, the animal table is changed when the user enters existing input pattern for different output.
- Compare desired output with network output. Verification checks whether network is trained well or not by comparing desired output and the network output after every guess. If the input patterns are the same and the user's output shows conflicts with the system's output, the system detects that there is a issue with training and tries to find correct parameters for training set by applying trial-and-error method. The mechanism will be explained in the 'Adjusting Network and Retrain' section below.

Hidden units

Calculating new Hidden units

Whenever the program learns a new concept, the number of output units will be increased and the number of hidden units need to be updated. A new number of hidden units is calculated by the formula given in Part 1 (Number of inputs + outputs) x 2/3). However, after altering the number of hidden units, parameters such as learning rate requires update, otherwise learning cannot be completed. (Reasons illustrated in Part 1)

Adjusting Network and Retrain

There are two different case for retraining and adjustment;

- In case a problem is detected in the network by the verification system. The program adjusts its network parameters in order to reach the desired output. For example, the learning rate will be altered and the total error will be controlled. If the altered version of the network has less total error than the first version, adjustment will stop. Otherwise alteration will continue.
- When a new concept is taught to the system by an user, the network is required to train itself because the size of the network and training set have changed. In this case, the verification system notifies the network to train itself with new parameters.

Decision Tree

To achieve better classification, response-sensitive questions should be posed to the user. Merely asking 20 fixed questions will not achieve adequate classification of all animals. However, if the system is capable of eliminating some questions depending on previous answers and creating more related questions, the system can classify better. This could be achieved by implementing a decision tree; this method will create detailed questions according to previous answers.

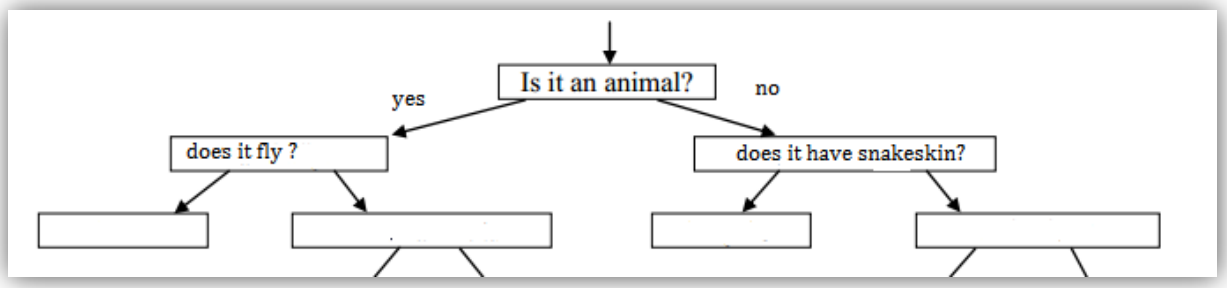

Figure-15 A decision tree which shows questions according to user decisions.

Figure 15 shows an example of a decision tree. If the user answers 'yes' to "Is it mammal?" it will be a wasted question for the network to ask next, "Does it have snakeskin?"

Decision tree is implemented and tested in the project. For each level of tree 2^n (n is depth of tree) questions need to be prepared. Preparing 2^{20} questions is manifestly beyond the scope of the present project. Therefore, I used 20 fixed questions.

Implementation

This part will explain implemented classes in java;

Data – this class includes set of variables which are parameters for project and neural network.

Concept – this class created to hold information about animal. This information is name of animal, corresponding input pattern and output neuron number.

Guess- findmax() method gets network output array as an parameter and choses the maximum value.(this value need to be bigger than threshold) then guessanimal() method checks whether its first guess of program or not. If it is first guess, it sends index information to main program so that program can display animals name, otherwise guessanimal() checks for other possible values bigger than threshold and send to main class.

Game- this is main class of project. Includes loops to get user input, show results and send data to related methods. Thus, loads weights from files if it is exist before.

AnimalNetwork- Includes neural network. Construction of class reads questions.txt and animal_training.txt file, determine number of input and output neurons using initializeNeurons() method and initialize questions calling initializeQuestions() method. Calculates hidden neurons using formula which is mentioned above sections. Then trains

network using its variable parameters according to values in animal_training.txt by calling initializeTrainingSetFromText() method and saves network to project folder. Training method is back-propagation which is provided by neuroph library. askNetwork() method gets answers to questions from user in main class and sends them to network. Network evaluates values and gives output according to its training. reTrain() method trains network in case network does not provide desired output or new concept added to network.

QuestionGenerator- Reads attributes from "attirbutes.txt" file and generates questions using GenerateQuestions() method . Afterwards, initializes these questions into an arraylist.

Questionnode- Represents a node in decision three. Holds the question neighbours and its parent information.

QuestionTree - This method constructs a decision tree from attributes.txt file. Every node includes a question. Tree is initialized when QuestionTree() construct is called.

VerifyClassification- Verifies input, output patterns and update insert text file when new concept learned. VerifyPath() method checks table, input, output patterns and notify network accordingly.

I/0

For this example, the user inputs the pattern for 'horse' but the user claims that it is a pattern for 'dolphin'. The program indeed guesses correctly but is met with a 'no' response. Thus, it presents the closest animal (lion) as a second guess. After the questions and answers, the user claims that it is a 'dolphin', verification class checks that concepts exists and that the input pattern for dolphin in the programs table is different from what the user enters. Therefore, the program assumes that user does not know all the facts about the claimed animal and gives notification.

```
network is created
number of input units:20
number of output unit:16
number of hidden units:25
learning in progress...
total error:0.009996428
learning finished
save is completed
Is it mammal ? (y/n) y
Does it carry ? (y/n) y
Does it hop ? (y/n) y
Lives sea ? (y/n) n
Have fur ? (y/n) y
Have feathers ? (y/n) n
Is it fast ? (y/n) y
Is it big(than_human) ? (y/n) y
Have leg ? (y/n) y
Is it nocturnal ? (y/n) n
Lives snow-covered-areas ? (y/n) n
Does it fly ? (y/n) n
Have mane ? (y/n) y
```

```
Does it purr ? (y/n) n
Is it tamed ? (y/n) y
Does it change-colours ? (y/n) n
Is it omnivor ? (y/n) n
Is it ruminant ? (y/n) n
Is it marsupial ? (y/n)n
Does it horns ? (y/n) n
i am guessing
Is it horse? n
Is it lion? n
I gave up,could you tell me what is it? dolphin
concept is already exsists
verifying paths
mistake in classification, are you sure you have right information about object?
done
```

In the following attempt, network weights will be changed using network. randomizeWeights() method, in order to observer how system detects problem in network and changes its parameters. Thus, network file under project directory is also deleted in order to prevent system to load old values.

```
network is created
number of input units:20
number of output unit:16
number of hidden units:25
learning in progress...
total error:0.009974827
learning finished
save is completed
Is it mammal ? (y/n) n
Does it carry ? (y/n) n
Does it hop ? (y/n) n
Lives sea ? (y/n) y
Have fur ? (y/n) n
Have feathers ? (y/n) n
Is it fast ? (y/n) n
Is it big(than_human) ? (y/n) n
Have leg ? (y/n) n
Is it nocturnal ? (y/n) y
Lives snow-covered-areas ? (y/n) n
Does it fly ? (y/n) n
Have mane ? (y/n) n
Does it purr ? (y/n) n
Is it tamed ? (y/n) n
Does it change-colours ? (y/n) y
Is it omnivor ? (y/n) n
Is it ruminant ? (y/n) n
Is it marsupial ? (y/n) n
Does it horns ? (y/n) n
i am guessing
Is it monkey? n
Is it dog? n
Is it whale? n
I gave up,could you tell me what is it? octopus
concept is already exsists
verfiying paths
```

```
i need to train myself again
randomazing weights
learning in progress...
total error:0.0099959625
learning finished
save is completed
done
```

The following example illustrates how the program will learn the concept “eagle”. After a series of questions, the program makes a few attempts at guessing. The first animal guessed - ‘bird’ – is the closest possibility to eagle. However, although second guessed animal, look bit irrelevant, they have similarities. The program asks the user for the concept. The concept is put into the program’s its table and the program trains itself again.

```
network is created
number of input units:20
number of output unit:16
number of hidden units:29
learning in progress...
total error:0.0099834325
learning finished
save is completed
Is it mammal ? (y/n) n
Does it carry ? (y/n) n
Does it hop ? (y/n) n
Lives sea ? (y/n) n
Have fur ? (y/n) n
Have feathers ? (y/n) y
Is it fast ? (y/n) y
Is it big(than_human) ? (y/n) n
Have leg ? (y/n) y
Is it nocturnal ? (y/n) n
Lives snow-covered-areas ? (y/n) n
Does it fly ? (y/n) y
Have mane ? (y/n) n
Does it purr ? (y/n) n
Is it tamed ? (y/n) y
Does it change-colours ? (y/n) n
Is it omnivor ? (y/n) n
Is it ruminant ? (y/n) n
Is it marsupial ? (y/n) n
Does it horns ? (y/n) n
i am guessing
Is it bird? n
Is it lion? n
I gave up,could you tell me what is it? eagle
i need to learn this concept
this line going to be added to table
0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 eagle
parameters set according to new concept
number of input units:20
number of output unit:17
number of hidden units:30
learning in progress...
total error:0.00996568
learning finished
save is completed
done
```

Check whether program learned concept or not.

```
number of input units:20
number of output unit:17
number of hidden units:30
load is completed
Is it mammal ? (y/n)n
Does it carry ? (y/n)n
Does it hop ? (y/n)n
Lives sea ? (y/n)n
Have fur ? (y/n)n
Have feathers ? (y/n)y
Is it fast ? (y/n)y
Is it big(than_human) ? (y/n)n
Have leg ? (y/n)y
Is it nocturnal ? (y/n)n
Lives snow-covered-areas ? (y/n)n
Does it fly ? (y/n)y
Have mane ? (y/n) n
Does it purr ? (y/n) n
Is it tamed ? (y/n) y
Does it change-colours ? (y/n) n
Is it omnivor ? (y/n) n
Is it ruminant ? (y/n) n
Is it marsupial ? (y/n) n
Does it horns ? (y/n) n
i am guessing
Is it eagle? y
thanks for playing game!
```

Conclusion

Although the present project is not identical to the 20Q.net game, it is designed in such a way that it could scaled up even manipulating text files under project file. Most of the information is filled up from text files dynamically. An enhancement like the decision tree is implemented. The present design of the project enables further developments easily.

Bibliography

[1] Dimitrios Siganos Christos Stergiou. NEURAL NETWORKS. [Online]. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

[2] F.reddy Fierens Paul L. Rosin, "Improving Neural Network Generalisation".

[3] Nibaran Das,Ram Sarkar, Mahantapas Kundu, Mita Nasipuri, Dipak Kumar Basu Subhadip Basu, "An MLP based Approach for Recognition of Handwritten 'Bangla' Numerals".

[4] Phil Picton, *Neurol Networks*.

[5] Anna Maria Fanelli Giovanna Castellano, *An Iterative Pruning Algorithm for Feedforward Neural Networks*.

[6] Robin. http://intelligence.worldofcomputing.net. [Online]. http://intelligence.worldofcomputing.net/machine-learning/learning-by-back-propagation.html#

[7] Cary, NC Warren S. Sarle. Answers provided by other authors as cited below are copyrighted by those authors, who by submitting the answers for the FAQ give permission for the answer to be reproduced as part of the FAQ in any of the ways specified in part 1 of the FAQ. [Online]. ftp://ftp.sas.com/pub/neural/FAQ2.html