**CSSE 413: Artificial Intelligence**

Neural Networks Lab Manual

Name: \_\_\_Doris Chen, Mashengjun Li\_\_\_\_

**Data Generation**

Done in Google Drive:

<https://docs.google.com/spreadsheets/d/1lPC6O6Tn1r8jy9tCmgXCFq1uQH1aL5fMuvyAsYr-mfo/edit#gid=100272116>

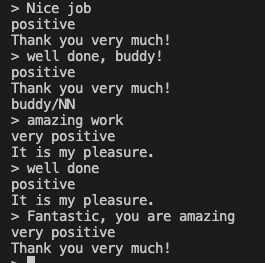
1. [5 pts] Ten inputs appropriate to our domain that are of a positive nature.

2. [5 pts] Ten inputs appropriate to our domain are neutral.

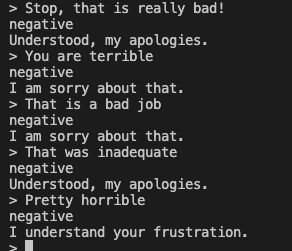
3. [5 pts] Ten inputs appropriate to our domain that are of a negative nature.

**Responding to Sentiments**

6. [5 pts] Please paste a transcript below showing your response to positive sentiments: The robot will reply politely if it detects positive praising.



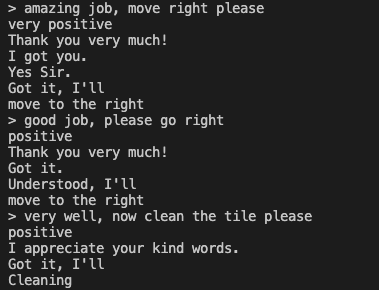
7. [5 pts] Please paste a transcript below showing your response to negative sentiments: If the user is not happy with it, the robot apologizes by acknowledging the human collaborator's pain.



8. [5 pts] Please paste a transcript below showing your response to positive/negative/neutral sentiments that include a command:

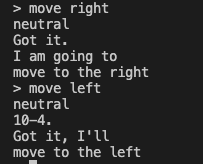
Positive:

The following picture shows the positive conversation between the user and the robot. The three commands in the picture were marked as “very positive”, “positive” and “positive” respectively. The robot replied with words of appreciation. Once it detects the command, it will execute the command.



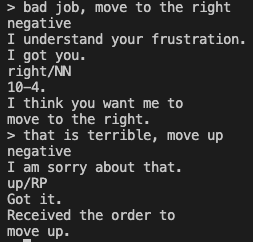
Neutral:

Pretty usual, the commands were in a natural tone and the robot followed the command perfectly.



Negative:

The following picture shows the negative conversation between the user and the robot. The robot felt sorry for “doing a bad job” or “being terrible”, and followed the command as usual.



**Basics**

12. [10 pts] **trainNetwork()** implemented. √

a) [1 pt] Given a *learning rate* of 0.1 and a *threshold* of 0.5, how many training episodes does the network take to learn Boolean **and**? \_\_2\_\_\_

b) [1 pt] Keep the learning rate at 0.1 and lower the threshold until you can learn in one episode. Keep the threshold values to whole 1/10ths. What *threshold* value enabled the network to learn in one episode? \_0.4\_\_

c) [1 pt] Reset the *threshold* to 0.5. Now, modify the *learning rate*, again in 1/10th increments until the network can learn and in one episode. What is the value of the *learning rate* that enables the network to learn in one episode? \_0.2\_\_\_

d) [2 pts] What is the relationship between the *learning rate* and the *threshold* values in the context of the prior two experiments? Please have a look at the weights of the network.

1eps lr=0.1 thr=0.4

1eps lr=0.2 thr=0.5

When we want our model to learn faster, we can either lower our threshold value or increase our learning rate value to make the model learn at the cost of arriving on a sub-optimal final set of weights.

**XOR Experiments**

e) [6 pts] Run XOR.java. Currently, the number of training episodes is to 1,000, the learning rate is set to 0.1 and there is a single hidden layer which has three nodes. Based on the example XOR net we saw in the slides, we should be able to develop a network that implements XOR reliably. As it turns out, this is not the case with the current set-up of the network. The testNetwork() method prints the actual output and the desired output. The printWeights() method shows the weights after learning. Experiment with the learning rate and the number of training episodes to determine values for which the network trains reliably. Add to your report three sets of values with which you were able to train the network efficiently and reliably. Briefly state what you consider to be reliable.

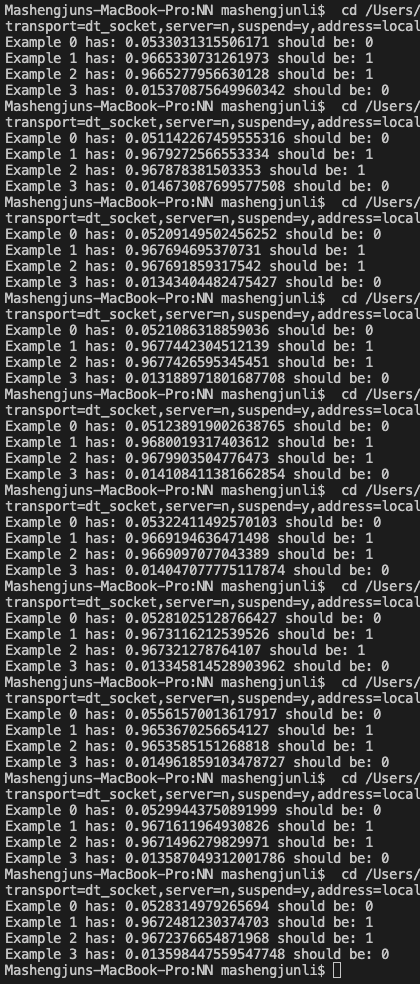
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Set | Learning Rate | Training Episodes | Output for example 1 | Output for example 2 |
| 1 | 0.2 | 50000 | 0.9803315239826986 | 0.9803269463375092 |
| 2 | 0.1 | 40000 | 0.9665330731261973 | 0.9665277956630128 |
| 3 | 0.2 | 45000 | 0.9791943964129664 | 0.9791911028295052 |

**Trials are shown below.**

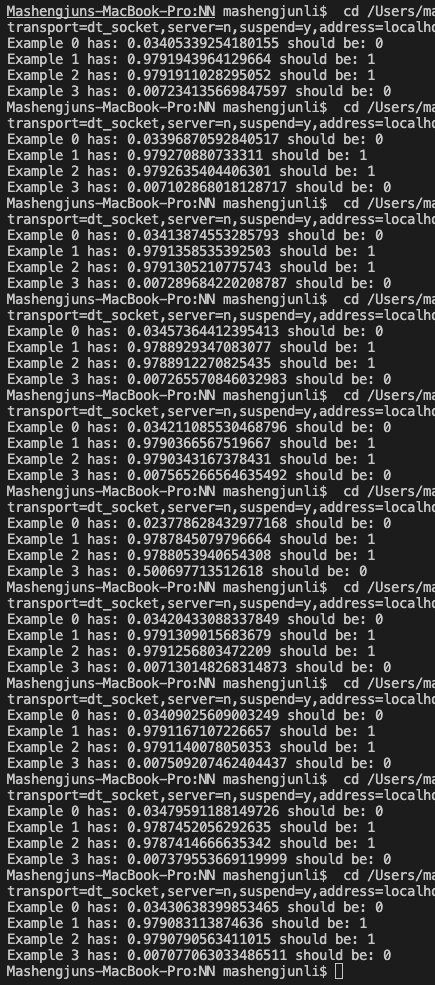
Set 1: 10 trials, all succeededText

Description automatically generated

Set 2: 10 trials, all succeeded



Set 3: 10 trials, 9 out of 10 succeeded



What relationship exists between among the parameters? Answer below.

It appears that the learning rate needs to be inversely proportional with training episodes in order to produce a better and more consistent result. Additionally, the learning rate needs to be relatively small and the training episodes need to be relatively large.

f) [9 pts] Modify the XOR.java file so that the network uses a step activation function rather than a sigmoid activation function. Hint: Recall that the derivative of the step function is not defined for all values, hence, you need to modify trainNetwork(). You should be able to learn XOR in fewer than 20 training episodes. Experiment with the number of training episodes, learning rate, threshold of the activation function and hidden layer size. Below, add three sets of values with which you were able to train the network efficiently and reliably.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Training episodes | Learning rate | Threshold | Hidden layer size |
| 1 | 19 | 0.1 | 0.1 | 10 |
| 2 | 15 | 0.15 | 0.3 | 6 |
| 3 | 10 | 0.19 | 0.4 | 10 |

Briefly state what you consider to be reliable.

We consider a reliable result as having no error, which accurately predicts all the examples for at least 8 out of 10 trials, and failure trials have an error rate(number of different actual and expected output/example number) of 50%-75%.

**Data Curation**

21. [20 points] *convertTextToEmbedding* and *FlattenAndPadding* successfully implemented. **Done**

**Fine-tuning and Evaluating a FFnet**

24. [20 points] Experiment with the various parameters so as to efficiently and reliably train the network. Record at least four experiments in which the hyper-parameters are reasonably different and the accuracy of the evaluation set (val\_acc) is at least 70%.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | #1 | #2 | #3 | #4 |
| Number of hidden layers | 1 | 2 | 3 | 8 |
| Hidden layer size | 10 | 3 | 4 | 8 |
| Batch size | 10 | 10 | 10 | 10 |
| Number of epochs | 100 | 100 | 150 | 200 |
| Learning rate | 0.01 | 0.003 | 0.002 | 0.0000002 |
| val\_acc | 0.71426 | 0.712698 | 0.714286 | 0.714286 |

Subjective evaluation: Looking at the training data and the evaluation accuracy, how well did the network learn?

The network did not learn very well. The best validation accuracy was only around 71%, but its validation accuracy was quite stable once the hyperparameters got tuned and will finally converge no matter how we update the input data by re-running the TextPreprocess.java file to generate a new set of training and validation data.