CS3105 Practical 1 Report

170011474

March 2020

1 Overview

This practical requires students to write a 20 Questions Game to guess a concept that a player is currently thinking of. For each question the player can answer "yes" or "no" and the AI would make guess at the end. New concepts and new questions can be added into original data sets and proper re-training processes would be made.

2 Design and Implementation

2.1 Game Logic

Once the game starts, the training set is built, parameters are set, and a neural network model is prepared for further prediction. Questions are read from database into a list and for each answer from players, the answer list would be temporarily filled with `0.5` to indicate a "maybe" state. After a few questions, if the model is confident enough to have a correct guess, then it would directly give its guess without the rest of questions asked. To get the most possible concepts, the `Game` instance maps the output to each concept and orders the map. For example, if the output is `[0.9, 0, 0.1, 0, 0, 0, 0, 0]`, it means that the possibility of the first concept is 90% and the third on is 10%. Ordering the map based on those possibilities and guessing from the most possible concept is the way of producing answers.

In Stage 2, a `threshold` is set up (normally 0.5) to filter those most possible concepts. If all possible concepts greater than 0.5 are not correct answers, then players are asked to add new concepts to enrich the database.

2.2 Concept Table and Question

The question is stored in `Questions.txt` and the concept table is stored in `data.csv`.

```
0 0 1 0 0 0 1 1,1 0 0 0 0 0 0 0,lion
0 1 0 0 0 0 0 1,0 1 0 0 0 0 0 0,cat
...
```

In this case, we only have asked 8 questions, and the answer list corresponds to the first column (`input`). For example, `0 0 1 0 0 0 1 1` means the answer is `no, no, yes, no, no, no, yes, yes`. The second column (`output`) is the corresponding concept in binary encoding format (one hot encoder here). The third column is the concept in human-readable string format.

Since the program is exported as a Jar file eventually, any modification to its own file is forbidden. SQL database is utilised to solve this problem as well as providing more storing functionality. Each time when there is no existing database, contents of `Question.txt` and `data.csv` are parsed and stored into newly created databases respectively as initial state. New concepts added by players would be used to update database rather than those initial files. Therefore, a initial state is guaranteed.

2.3 Configuration and Parameters of Neural Network

The `input_units_num` is the number of questions asked, and `output_units_num` is the number of concepts in the guessing range. For other main parameters, I wrote a simple program to iterate over a possible range to see the best choice and also trial errors are considered to avoid over-fit.

2.3.1 The Number of Hidden Units

As for `hidden_units_num`, an optimal parameter can only be found by using iterating method to try each possible hidden numbers with human intervention. However, in our case, we can use a general way to determine it: `The number of hidden units = (input size + output size) * (2/3)`. Too many hidden units can cause over-fitting and too few hidden units can cause under-fitting. Since we would gradually add more new training samples, the number of hidden units also need changes so it keeps pace with new dataset. In this case, it is tightly related to the number of input and output sizes. Now we have 8 questions and 8 possible concepts, so the hidden unit number is 13.

In order to verify this formula, a trial is made. The target output is `dog` and the original input in training set is `0 1 0 1 0 0 0 1`, but now the trial input is `0 1 0 1 0 0 1 1`. Grid Search is used to use different parameters for the number of hidden units, from 4 to 16. We can see from the result, 13 is actually a good idea.

```
......
hidden units: 10 total error: 0.00434064487523549
trial mse: 9.943546726209405E-4
ratio (trial mse/total error): 22.907994116127604%

hidden units: 11 total error: 5.923457114000121E-10
trial mse: 9.730849574415865E-5
ratio (trial mse/total error): 1.6427652614242706E7%

hidden units: 12 total error: 3.007097167614341E-9
```

```
trial mse: 4.5201426862776525E-6
ratio (trial mse/total error): 150315.8173589607%

hidden units: 13 total error: 5.828317863001897E-16
trial mse: 7.502764234528448E-8
ratio (trial mse/total error): 1.2872949641535374E10%

hidden units: 14 total error: 2.856681752865452E-25
trial mse: 1.2417206273719622E-9
ratio (trial mse/total error): 4.3467236983133645E17%
......
```

2.3.2 The Learning Rate

By using the same Grid Search way, when learning rate is above 0.5, the error appears to be very satisfactory. Those code are located at `BuildModel.java`.

```
......
learning rates: 0.49999999999999994 total error: 1.1656449673608512E-4
trial mse: 9.506131478750703E-5
ratio (trial mse/total error): 81.55254597181192%

learning rates: 0.5499999999999999 total error: 1.290663085850019E-12
trial mse: 4.333036839311993E-6
ratio (trial mse/total error): 3.3572176091627306E8%

learning rates: 0.6 total error: 1.1720555148805442E-12
trial mse: 1.975849669661685E-7
ratio (trial mse/total error): 1.6857987054163244E7%
......
```

3 Stage 2

3.1 When to Guess?

Sometimes the model can guess the correct answer even if only a few questions have been asked. To reduce the gaming time, the model can directly enter the last phase to ask if the player "you are thinking of: xxx" to end the game. To implement this functionality, after each question is asked and an answer is obtained, the `guess_list` will automatically fill the rest answer slots with `0.5` and pass it to the model to be calculated.

If the most possible concept is over `threshold 0.5`, then it is highly probable to be the correct answer. If the player confirms, then game ends. Otherwise, the game continues to ask questions.

```
answer_list.add(answer);
// answers so far to be filled
for(int j = 0; j <= i; j++) {
   guess_list.add(Double.valueOf(answer_list.get(j)));
}
// maybe state for rest of them
for(int j = i + 1; j < question_list.size(); j++) {
   guess_list.add(0.5);
}
```

3.2 Add New Concepts and Questions

If a new concept is added, for example "panda", then other concepts (such as "lion") will change from

```
0 0 1 0 0 0 1 1,1 0 0 0 0 0 0 0,lion
```

to

```
0 0 1 0 0 0 1 1 0,1 0 0 0 0 0 0 0 0,lion
0 1 0 0 0 0 0 1 0,0 1 0 0 0 0 0 0 0,cat
...
{answer for panda},0 0 0 0 0 0 0 0 1, panda
```

By `one hot encoding`, other concepts' binary encoding should append `0`. Besides, other concepts' answer paths should also append a new answer for the new question. After updating databases, the neural network will retrain. If a user is thinking about `computer` but if he denied `computer` when the game asked him, then adding new concept "computer" will fail, since the database already contains `computer` and it would not make it up for player's fault.

3.3 Update Hidden Units Number

Please see Section 2.3.1 to refer the updating.

3.4 Random Questions

The game would ask questions in random order rather than fixed ones. The implementation relies on a sorted tree map `id_answer_map` to associate question id to each answer.

```
Collection<Integer> values = id_answer_map.values();
answer_list = new ArrayList<>(values);
```

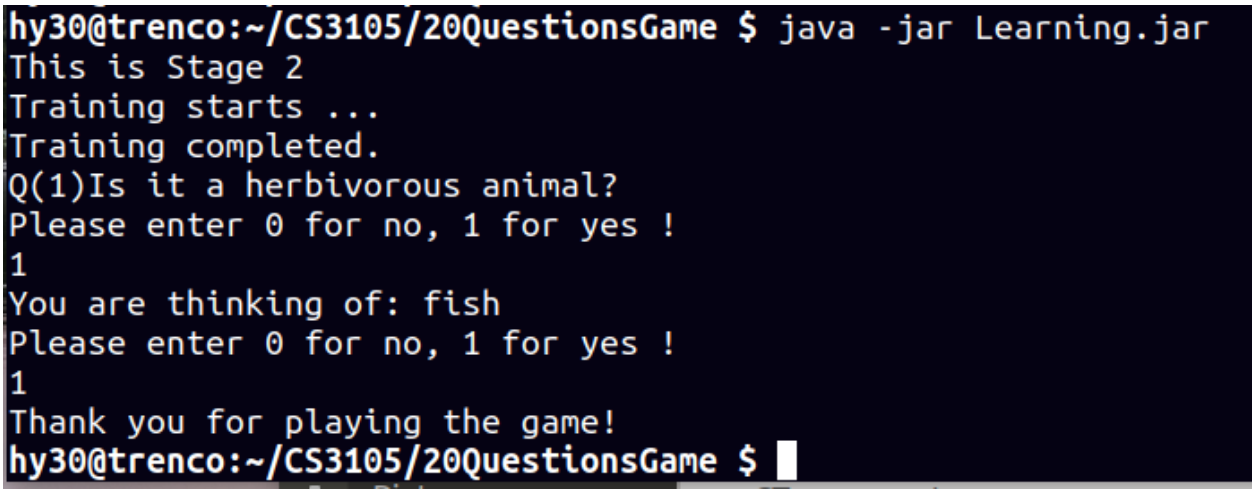

Figure 1: Running in lab machine

4 Running Examples

4.1 Running in Lab Machine

Please see Figure 1: Running in lab machine. **For better format**, I used `lstlisting` element in Latex to show my terminal running results in the following examples.

4.2 Early Guess

I am thinking "bird".

```
Please enter 0 for no, 1 for yes !
Q(1)Is it a herbivorous animal?
0
You are thinking of: snake
Please enter 0 for no, 1 for yes !
0
Q(2)Is it small?
1
You are thinking of: bird
Please enter 0 for no, 1 for yes !
1
Thank you for playing!
```

As we can see from above, the model would automatically use a quick guess.

4.3 Adding New Concepts

I am thinking of "computer".

```
Please enter 0 for no, 1 for yes !
```

```
Q(1)Is it a herbivorous animal?
0
Q(2)Is it small?
1
Q(3)Is it normally dangerous?
0
Q(4)Can it swim?
0
Q(5)Can it fly?
0
Q(6)Does most of them live in water?
0
Q(7)Is it a wildlife?
0
Q(8)Does it make a noise?
1
You win. What is your concept?
computer
Give me a feature about your concept but my guess dont have
Is it electronic?
I need to learn it now
adding new information to database......
adding successful!
Retraining starts ...
Retraining completed.
```

4.4 Questions in Random Order

The first running:

```
Q(1)Is it a wildlife?
Please enter 0 for no, 1 for yes !
0
Q(2)Is it a herbivorous animal?
Please enter 0 for no, 1 for yes !
0
Q(3)Can it fly?
Please enter 0 for no, 1 for yes !
...... // other
```

The second running:

```
Q(1)Is it a wildlife?
Please enter 0 for no, 1 for yes !
0
Q(2)Is it a herbivorous animal?
Please enter 0 for no, 1 for yes !
```

```
0
Q(3)Can it fly?
Please enter 0 for no, 1 for yes !
...... // other
```

4.5 Fault Tolerant

When players input illegal inputs (they enter a string rather than an integer), they will be asked to input again.

```
Q(1)Is it a herbivorous animal?
Please enter 0 for no, 1 for yes !
kjf
Please enter 0 or 1!
Q(1)Is it a herbivorous animal?
Please enter 0 for no, 1 for yes !
98
Q(1)Is it a herbivorous animal?
Please enter 0 for no, 1 for yes !
1
Q(2)Is it small?
Please enter 0 for no, 1 for yes !
```

4.6 Debugging Methods

`print_concept_table` and `print_question_table` are debugging methods printing out databases information in `Stage 2`'s source code. Methods doing grid search on parameters are written in `BuildModel.java` in `Stage 2`. Please refer those functions.

Stage 1		Done?
(i) Pre-processing and Input	Clearly described and justified	Yes
(ii) Hidden Layering	Clearly described and justified	Yes
(iii) Output and Post-Processing	Clearly described and justified	Yes
(iv) Training	Clearly described and justified	Yes
(v) I/O	Selected key examples of question and answer sessions	Yes
(vi) Concept Table	Clear, easy to understand table with main text explanation	Yes
(vii) Documentation for Stage 1	Documentation is clear, appears comprehensive, and shows understanding and effort upon read-through Satisfaction of report requests	Yes
Stage 2		
(i) Concept Table	Clear, easy to understand table with main text explanation	Yes
(ii) Hidden units	Clearly described and justified mechanism for adding units	Yes
(iii) Choosing a question	Clearly described and justified mechanism for effective choices	No
(iv) When to guess	Clearly described and justified mechanism for effective guessing	Yes
(v) I/O	Selected key examples of question and answer sessions	Yes
(vi) Documentation for Stage 2	Documentation is clear, appears comprehensive, and shows understanding and effort upon read-through Satisfaction of report requests	Yes
Answers to Questions	Clearly given, understandable, and satisfactory	Yes

Figure 2: Check Table