MISSION #12

Artificial Neural Networks (SIMBot)

Due: Week 15

Introduction:

Now, after you have learned something about using GA which is used for creating fuzzy rules for you. This assignment is about learning how to apply an approach of ANN for a brain of robot in order to control the SIMBot with the same task. You may know that ANN needs training data to adjust their weights during the learning period. Fortunately, for you, I have already prepared the training data. The training data are collected from recording all sensory inputs (both IR sensors and smell sensors) and the outputs (Turn and Move) during executing a fuzzy control SIMBot on the same environment. There are 3 set of training data. You may use all data sets or only a few sets. You may notice about how many training data are needed to get the good brain. ☺

  - Training data have been conversed in the normal form of [0.0, 1.0]
  - There are 9 inputs and 2 outputs.
  - Nine inputs are 8 IR sensors and 1 smell sensor.
- The IR sensors have its possible values between 0 and 100.
- The Smell sensor has its possible values between -180 and 180.
- The outputs are Turn and Move, which are ranged between [-90, 90] and [-10, 10] respectively.
- The ANN codes, we have added all library classes into the project. You may need to look inside for its coding and try to find which functions you may need to change or adapt.
- At start, you need to configure the structure of ANN before using. You may use different settings.

There are 3 main code files: the data collection code, the training code, and the testing code.

1) The data collection code: you need to create data set by showing how to move by yourself. The simulation is run and record every step the movements which are controlled by pressing **wasd** keys. The data are recorded only when the robot moves, if it stays still, no date is recorded. You may have to do it many times for collecting all possible movements that need to be trained. In every 4000 steps, separated files are recorded in which you can decide to keep or use it or not. The final data have to save into one file named “history_all.csv”. You have to put them up into one file by yourself as many as you can.
   - This shows some parts of training data.

```
ir0,ir1,ir2,ir3,ir4,ir5,ir6,ir7,angle,turn,move
100,46.56854249492384,100,32.42640687119281,20.0,32.42640687119277,20.0,32.42640687119277,46.73570458892837,0,5
100,32.42640687119277,20.0,100,70.0,32.42640687119277,20.0,32.42640687119277,49.864514437760505,0,5
100,100,100,32.42640687119293,100,32.42640687119277,20.0,32.42640687119277,56.309932474020215,5,0
100,100,97.07558450817119,100,100,100,100,100,34.19167716816156,5,0
100,100,77.52132233408457,100,100,100,100,100,-7.805205425787221,0,5
100,100,100,100,100,100,100,100,-7.9414875159599205,0,5
100,57.20286602974577,100,100,100,100,100,100,35.68793848235046,0,5
100,100,100,47.1947191000093,100,74.22663713730022,49.557226276323696,74.22663713730026,27.119389315826993,0,5
71.79324911373982,100,100,100,100,100,74.22663713730022,49.557226276323696,74.22663713730026,34.57978334082776,0,5
40.500903455150876,100,100,100,100,100,100,46.32643879163029,112.20636470167646,5,0
100,100,100,100,100,100,45.75639103708276,43.77551154089719,11.998815119281943,0,5
100,100,100,100,100,100,43.424852746307785,100,12.376954711861742,0,5
100,92.60863218827691,100,100,100,95.0452803762273,100,100,14.653132866738247,5,0
100,83.275243813717,100,100,100,100,100,100,9.653132866738275,5,0
100,100,100,100,42.86686974682059,100,100,100,-102.05496717488148,-5,0
100,100,100,100,42.26257809230038,70.99674358929342,100,100,-97.05496717488148,-5,0
```

```
#!/usr/bin/python3

from pysimbotlib.core import PySimbotApp
```

```
from kivy.logger import Logger

from kivy.config import Config
# Force the program to show user's log only for "info" level or more. The info log will be
 disabled.
Config.set('kivy', 'log_level', 'info')

import random

if __name__ == '__main__':
    app = PySimbotApp(enable_wasd_control=True, save_wasd_history=True, max_tick=4000, sim
ulation_forever=True)
    app.run()
```

2) The ANN training code: there are 7 steps. The first step is to define the scaling function, which is used to converse all training data into the normal form of [0.0, 1.0]. Then, by using **pandas** library, data are read from a file named "history_all.csv" and also scale them into normal form. All data are loaded into a couple of long list as input list and output list. The input list consists of 9 fields of the IR sensors and a smell sensor. To define ANN structure, we use **keras** library. Basic setup is define for you. It is up to you to alter the setting. Before to start training, we will check whether there is existing model to start with. If there is one, the pre-trained model is loaded. Next, the step 5, training process is started. You must select how to train these huge data. Then, the final model is saved as a file named "assignment5_model.h5". We also use **matplotlib** library to generate the learning curve. Your task to make sure that your training model is sufficiently trained.

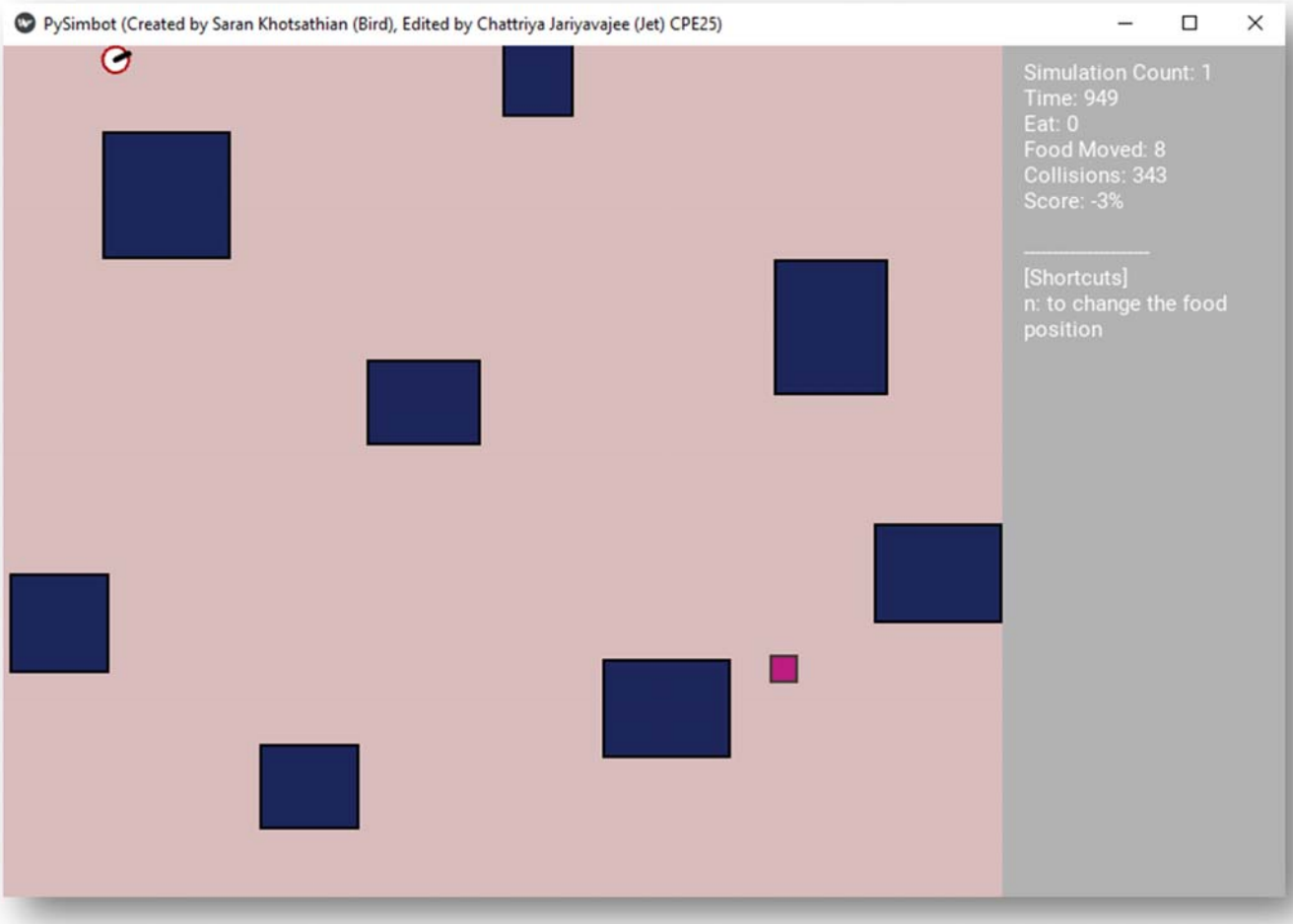
```
# 1. define scaling function
from typing import Tuple
def scale(data, from_interval: Tuple[float, float], to_interval: Tuple[float, float]=(0, 1
)):
    from_min, from_max = from_interval
    to_min, to_max = to_interval
    scaled_data = to_min + (data - from_min) * (to_max - to_min) / (from_max - from_min)
    return scaled_data


# 2. read data
import pandas as pd
dataframe = pd.read_csv('history_all.csv', sep=',')
dataframe.iloc[:, 0:8] = scale(dataframe.iloc[:, 0:8], from_interval=(0, 100), to_interval
=(0,1))
dataframe.iloc[:, 8] = scale(dataframe.iloc[:, 8], from_interval=(-
180, 180), to_interval=(0,1))
dataframe.iloc[:, 9:] = scale(dataframe.iloc[:, 9:], from_interval=(-
5, 5), to_interval=(0,1))


# 3. select input and output of the ANN
x = dataframe.iloc[:, :9]
y = dataframe.iloc[:, 9:]


# 4. define ANN architecture, loss and optimizer
from keras.models import Sequential
```

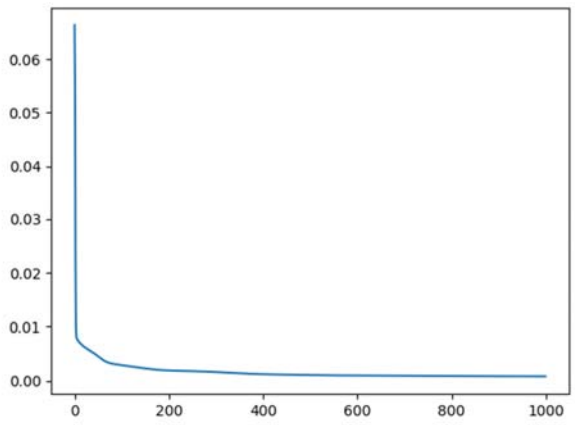
```
from keras.layers import Dense
model = Sequential()
model.add(Dense(32, activation='sigmoid', input_shape=(9,)))
model.add(Dense(16, activation='sigmoid'))
model.add(Dense(2, activation='sigmoid'))
model.compile(optimizer='sgd', loss='mean_squared_error')
import os
if os.path.isfile('./assignment5_model.h5'):
    model.load_weights('assignment5_model.h5')

# 5. train ANN model
history = model.fit(x, y, batch_size=1000, epochs=1000, shuffle=True)

# 6. save model for later usage
model.save('assignment5_model.h5')


# 7. plot the loss value of the training
from matplotlib import pyplot as plt
plt.plot(history.history['loss'])
plt.show()
```

3) The ANN testing code: there are 3 steps. The first step is to define the scaling function, which is to be used to store current sensory data into the normal form. Then, these input data are feed into the trained model that you created from the training data. The trained model is stored in a file named "assignment5_model.h5". The output are then conversed into turn and move data. The turn and move output are applied to the Simbot in turn. In simulation, we give you 5000 steps, so that the simulation robot has to use the trained model in controlling the movements for getting food and avoiding hitting obstacles. The highest score is expected as possible.

```
#!/usr/bin/python3

from pysimbotlib.core import PySimbotApp, Simbot, Robot, Util
from kivy.logger import Logger
from kivy.config import Config
from keras.models import load_model
import random
import numpy as np

# # Force the program to show user's log only for "info" level or more. The info log will
be disabled.
# Config.set('kivy', 'log_level', 'debug')
Config.set('graphics', 'maxfps', 10)

# 1. define scaling function
from typing import Tuple
def scale(data, from_interval: Tuple[float, float], to_interval: Tuple[float, float]=(0, 1
)):
    from_min, from_max = from_interval
    to_min, to_max = to_interval
    scaled_data = to_min + (data - from_min) * (to_max - to_min) / (from_max - from_min)
    return scaled_data
```

```python
# 2. create robot for testing the model.
class NNRobot(Robot):

    def __init__(self, **kwarg):
        super(NNRobot, self).__init__(**kwarg)
        self.model = load_model('assignment5_model.h5')

    def update(self):
        # read sensor value
        ir_values = self.distance()  # [0, 100]
        angle = self.smell()         # [-180, 180]

        # create 2D array of size 1 x 9
        sensor = np.zeros((1, 9))

        # set the values of the input sensor
        for i, ir in enumerate(ir_values):
            sensor[0][i] = scale(ir, (0, 100), (0, 1))

        ## use this line if the training data is from Jet
        sensor[0][8] = scale(angle, (-180, 180), (0, 1))

        # inference
        output = self.model.predict(sensor)

        # scale the output back
        answerTurn = scale(output[0][0], (0, 1), (-90, 90))
        answerMove = scale(output[0][1], (0, 1), (-10, 10))

        # perform the robot movement
        self.turn(int(answerTurn))
        self.move(answerMove)

        if self.stuck:
            Deg = random.randint(-10, 10)
            self.turn(Deg)
            self.move(-5)


# 3. start the simulation
if __name__ == '__main__':
    app = PySimbotApp(robot_cls=NNRobot,
                      num_robots=1,
                      theme='default',
                      interval = 1.0/60.0,
                      simulation_forever=False)
    app.run()
```


PySimbot (Created by Saran Khotsathian (Bird), Edited by Chattriya Jariyavajee (Jet) CPE25)
Simulation Count: 1
Time: 949
Eat: 0
Food Moved: 8
Collisions: 343
Score: -3%
[Shortcuts]
n: to change the food position


0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
200
400
600
800
1000