Deep Reinforcement Learning for Tic Tac Toe game

โดย

นาย เมธานนท์ แก้วกระจ่าง รหัส 62050214

นาย เอกสิทธิ์ บุตรดา รหัส 62050251

เสนอ

อาจารย์ อัคเดช อุดมชัยพร

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา Neural Network and Deep Learning

ภาควิชา วิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2564

Abstract

Reinforcement Learning เป็นการแก้ปัญหาของเกม โดยจะเห็นได้ตามข่าวสารด้านเทคโนโลยี เช่น Alpha Go ก็เป็นการใช้ Reinforcement Learning ในการแก้ปัญหา ซึ่งมี Algorithms ต่างๆ มากมายให้เลือกใช้ รายงานฉบับนี้จะเลือกใช้ Deep Learning มาใช้ในการแก้ปัญหาของการเล่น Tic Tac Toe

Introduction

Tic Tac Toe มีการเดินทั้งหมด 9! รูปแบบ หรือ เท่ากับ 362880 รูปแบบ โดยจะเป็นการนับความน่าจะเป็นของผู้เล่นทั้งสองฝั่งรวมกัน

เช่น ผู้เล่นคนแรก ในตาแรกจะมีทางเลือกคือ 9 ช่อง

ผู้เล่นคนที่สอง ในตาแรกจะมีทางเลือกคือ 8 ช่อง ซึ่งสุดท้าย จะมีรูปแบบทั้งหมดคือ 9! รูปแบบ

สิ่งที่คาดหวังจากการทำ Deep Reinforcement Learning ก็คือ การได้ model เพื่อทำนายผลลัพธ์ของรูปแบบทั้งหมด เพื่อสามารถนำไปแปลงเป็น application ในการตัดสินใจเล่นต่อไป

Method and Design

1. Dataset

การเตรียม Dataset จะเขียน CSV โดยการสุ่ม State ของเกมมาจำนวนหนึ่ง เพื่อใช้ในการ train และ test ซึ่งจะเก็บไฟล์ CSV ไว้ใน Github เพื่อความง่ายในการ Import

ข้อมูลจะมีทั้งหมด 82 columns โดยที่ 1 - 81 คือ ข้อมูล state ของเกม และ column ที่ 82 เป็นผลลัพธ์ของเกม ซึ่งมีที่มาจากการเล่นในแต่ละตา ซึ่งแต่ละตาจะเดินได้ 9 ช่อง

และมีทั้งหมด 9 ตา

ค่า X ที่เก็บ 0 = ช่องว่าง 1 = ผู้เล่น 1 ลงช่องนั้น 2 = ผู้เล่น 2 ลงช่องนั้น

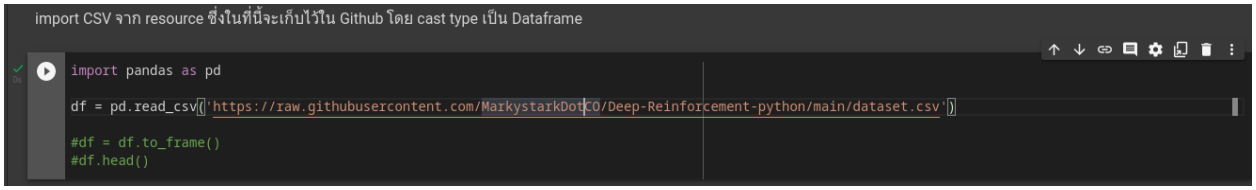
ค่า Y ที่เก็บ 0 = เสมอ 1 = ผู้เล่น 1 ชนะ 2= ผู้เล่น 2 ชนะ

ตัวอย่าง

1	2	3	9	ผลลัพธ์
000000001	0000000021	000002021	……..111022021	1

2.Train Model

2.1 Import ไฟล์ CSV เข้ามา ด้วย Pandas

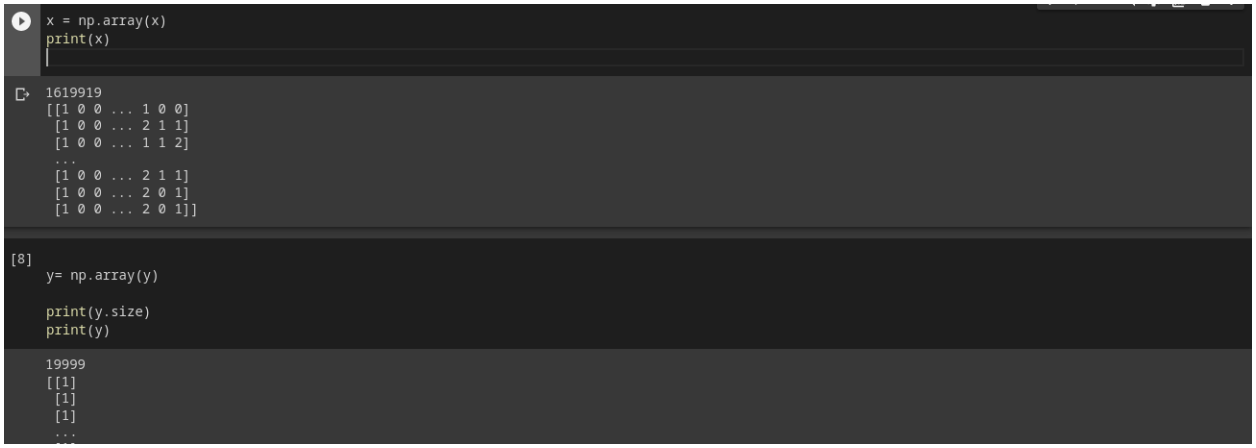


2.2 Import Numpy เพื่อใช้ในการ transform data จาก dataframe ไปเป็น numpy array

2.3 แบ่งข้อมูลก้อนใหญ่ ออกเป็น x และ y

ซึ่ง y จะเป็น column สุดท้าย และ x เป็น column แรก จนถึง column ที่ 81

2.4 แปลงข้อมูลจาก dataframe เป็น numpy array



```
x = np.array(x)
print(x)
```

```
1619919
[[1 0 0 ... 1 0 0]
 [1 0 0 ... 2 1 1]
 [1 0 0 ... 1 1 2]
 ...
 [1 0 0 ... 2 1 1]
 [1 0 0 ... 2 0 1]
 [1 0 0 ... 2 0 1]]
```

```
[8]
y= np.array(y)

print(y.size)
print(y)
```

```
19999
[[1]
 [1]
 [1]
 ...
```

2.5 ตั้งค่า parameter ของ model

-Convolutional Layer .ใช้แค่ 2 layers เพื่อความเร็วในการ train

-Activation Function จะใช้เป็น Relu

-Pooling Layerจะใช้ Max Pooling

-activation function ตัวสุดท้าย จะเป็น softmax เพราะ output สุดท้ายจะเป็นผลเฉลยของเกม

ได้แก่ 0 คือ เสมอ, 1คือ ผู้เล่น 1 ชนะ , 2 คือ ผู้เล่น 2 ชนะ

```
from keras.models import Sequential
from keras.layers import Dense, Conv1D, Flatten, MaxPooling1D
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.15)
model = Sequential()
model.add(Conv1D(64, 2, activation="relu", input_shape=(81,1)))
model.add(Dense(16, activation="relu"))
model.add(MaxPooling1D())
model.add(Conv1D(32, 2, activation="relu", padding='same'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(3, activation = 'softmax'))

model.compile(loss = 'sparse_categorical_crossentropy',
     optimizer = "adam",
              metrics = ['accuracy'])

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_3 (Conv1D)           (None, 80, 64)            192

 dense_3 (Dense)             (None, 80, 16)            1040

 max_pooling1d_2 (MaxPooling  (None, 40, 16)           0
 1D)

 conv1d_4 (Conv1D)           (None, 40, 32)            1056

 max_pooling1d_3 (MaxPooling  (None, 20, 32)           0
 1D)

 flatten_1 (Flatten)         (None, 640)               0

 dense_4 (Dense)             (None, 3)                 1923

=================================================================
Total params: 4,211
Trainable params: 4,211
Non-trainable params: 0
_________________________________________________________________
532/532 [==============================] - 2s 3ms/step - loss: 1.3303e-08 - accuracy: 1.0000
```

-ทำการ fit model โดยใส่ข้อมูลทั้ง train และ test เข้าไป

-ทำการวัด accuracy & confusion matrixเพื่อการวัดประสิทธิภาพของ model

```
model.fit(xtrain, ytrain, batch_size=16,epochs=100, verbose=0)

acc = model.evaluate(xtrain, ytrain)
print("Loss:", acc[0], " Accuracy:", acc[1])

pred = model.predict(xtest)

pred_y = pred.argmax(axis=-1)

cm = confusion_matrix(ytest, pred_y)
print(cm)

print(" y test is", ytest[1])
print("y predicted is ", pred[1])
```

```
Loss: 1.3303131929376377e-08  Accuracy: 1.0
[[ 451    0    0]
 [   0 1770    0]
 [   0    0  779]]
 y test is [1]
y predicted is  [2.2779438e-14 1.0000000e+00 1.2786604e-12]
```

- ทดสอบกับข้อมูลใน row ที่ 1075 จะพบว่า ค่าผลลัพธ์จริง คือ 0 (เสมอ) และค่าที่ทำนาย (predict) คือ 0 เช่นกัน แสดงว่าการทำนายถูกต้อง

```
[ ]  test_index = 1075

     print("x test is   :::", xtest[test_index])
     print(" y test is   :::", ytest[test_index])

     max_value = np.max(pred[test_index])

     predited_y = np.argmax(pred[test_index])
     print("y predict is    :::", predited_y)
```

```
x test is   ::: [1 0 0 0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 1 1 2 0 0 0 0 0 0 1 1 2 0 0 0 0 0 2 1
 1 2 0 0 0 0 1 2 1 1 2 0 2 0 0 1 2 1 1 2 0 2 1 0 1 2 1 1 2 2 2 1 0 1 2 1 1
 2 2 2 1 1 1 2]
 y test is   ::: [0]
y predict is    ::: 0
```

-นำข้อมูล x ที่ใช้ test มาจำลองเป็นกระดาน จะพบว่าเป็นไปตามการเสมอกันจริงๆ

```
[ ]  test = xtest[test_index]
     table = test[:81]

     #print(table)
     write =""

     raws = ""
     raw = 71
     for i in range(9):
       #print(table[0,raw])
       raw = raw+1
       write = write +"     "+str(table[raw])
       raws = raws+"     "+str(table[raw])
       if i ==2 or i==5 or i==8:
         write = write +"\n"

     print(write)
     #print(raws)
```

```
    1    1    2
    2    2    1
    1    1    2
```

Conclusion

Model สามารถทำนายผลลัพธ์ได้ตรงตามที่ต้องการ แต่ยังไม่สามารถ confirm ได้ว่าจะสามารถทำนายได้ถูกในทุกๆ State เพราะข้อมูลที่นำมา train มีเพียง 19,999 rows จากทั้งหมด 362880 rows