基於遊戲的機器學習入門
作業一實作介紹

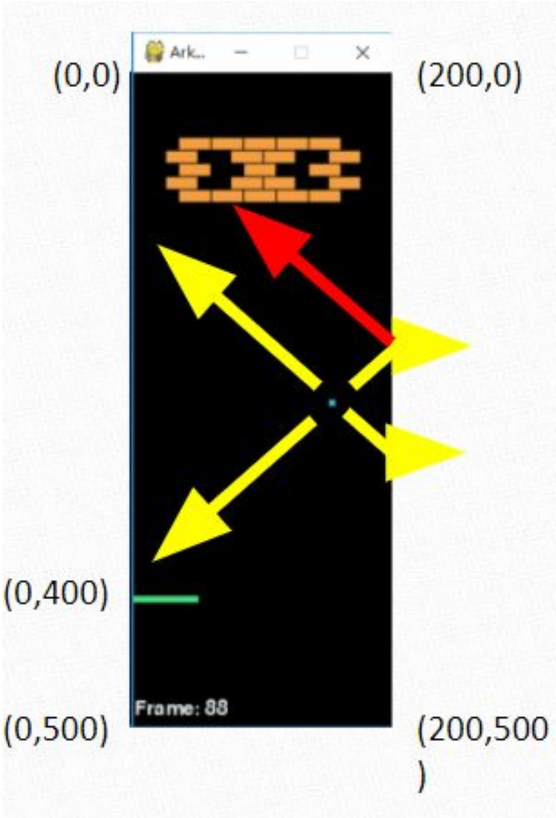
NCBCL

作業流程

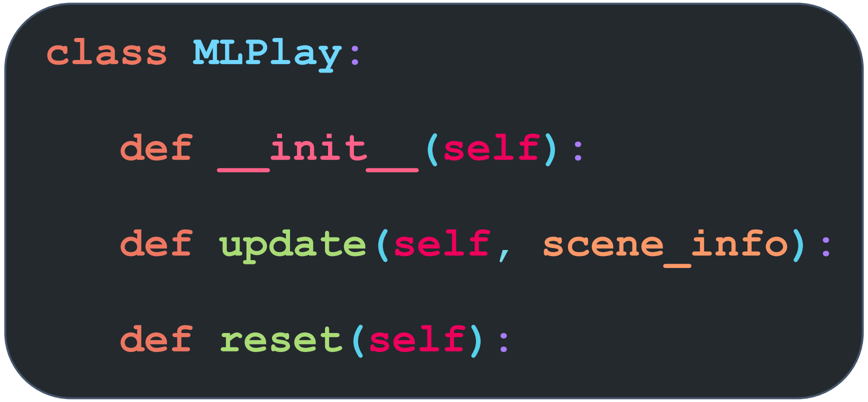
1. 撰寫 Rule code
2. 收集 pickle 檔案 ($ python MLGame.py -r -i you_code.py arkanoid NORMAL 2)
3. 特徵提取 & Train Data
4. 寫 ML code

撰寫 Rule

- 計算斜率，預測球的落點
  - 優點: 如果規則寫得好，資料不須收集太多
  - 缺點: 需要思考...
- 隨機法，讓板子亂動，打到就是好球
  - 優點: 不須思考
  - 缺點: 需收集大量資料且耗時 (絕大部分的資料是無用的)
- 其他你想的到的方法



撰寫 Rule



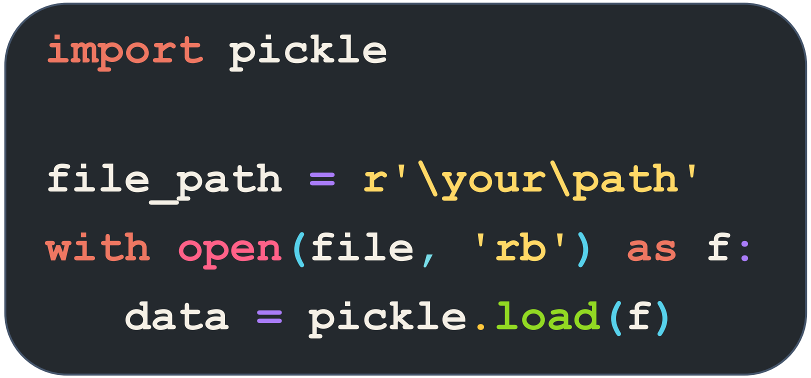
```
class MLPlay:

    def __init__(self):

    def update(self, scene_info):

    def reset(self):
```

Here are the available commands:

- "SERVE_TO_LEFT": Serve the ball to the left
- "SERVE_TO_RIGHT": Serve the ball to the right
- "MOVE_LEFT": Move the platform to the left
- "MOVE_RIGHT": Move the platform to the right
- "NONE": Do nothing
- "RESET": call self.reset()

```
{
    'frame': 10,
    'status': 'GAME_ALIVE',
    'ball': (30, 332),
    'platform': (30, 400),
    'bricks': [(35, 50), (60, 50), (85, 50), (110, 50)
    'hard_bricks': []
  }
```

收集 pickle 檔案



```python
import pickle

file_path = r'\your\path'
with open(file, 'rb') as f:
    data = pickle.load(f)
```

收集 pickle 檔案

data structure

```
# beat 8.x
{
    "record_format_version": 2,
    "ml": {
        "scene_info": [scene_info_0, scene_info_1, ... , scene_info_n-1, scene_info_n],
        "command": [command_0, command_1, ... , command_n-1, None]
    }
  }

# beat 7.x
{
    "scene_info": [scene_info_0, scene_info_1, ... , scene_info_n-1, scene_info_n],
    "command": [command_0, command_1, ... , command_n-1, None]
  }
```

特徵提取 & Train Data

目標:

- 將有用的特徵資料整理出來
- 轉換成二維陣列的資料格式
- 將資料 fit 進 ML model

	特徵1	特徵2	特徵3	特徵4
0				
1				
2				
3				
4				
5				
6				

寫 ML code

步驟:

- 把訓練好的 model load 進來
- 將 scene_info 資訊提取並整理
- 丟進 model predict
- return predict command