Arkanoid

攻略1

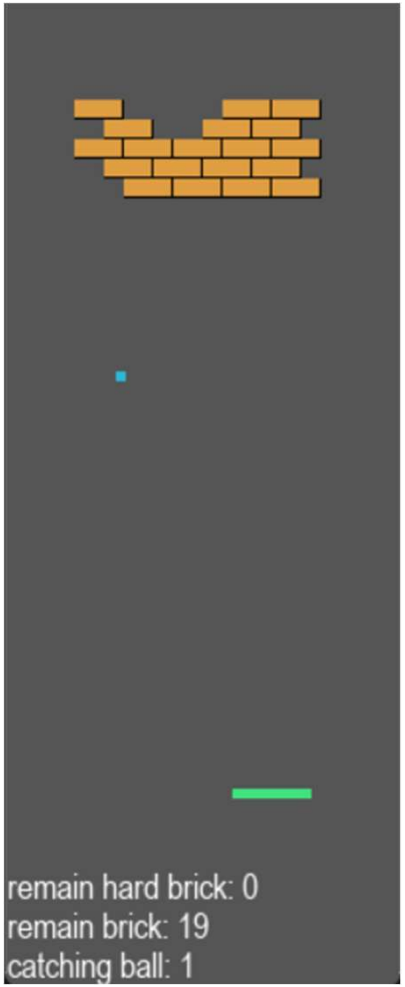
大綱

- 遊戲介紹
- 解題思路
  - 思路說明
    - 解題策略
    - 選擇特徵
    - 選用模型及參數
  - 實作步驟
    - 蒐集特徵
    - 數據清洗
    - 訓練模型
    - 驗證模型
  - 結果分析
  - FAQ

遊戲介紹

遊戲介紹 - 遊戲目標

這是一個經典的打磚塊遊戲，決定平台的移動方向，嘗試接住下墜的球，並打掉所有的磚塊!



遊戲介紹 - 環境介紹

遊戲的座標系統

畫面左上角為(0, 0)，x軸向右遞增，y軸向下遞增。
遊戲提供的座標資料都是該物件左上角的座標。

遊戲物件

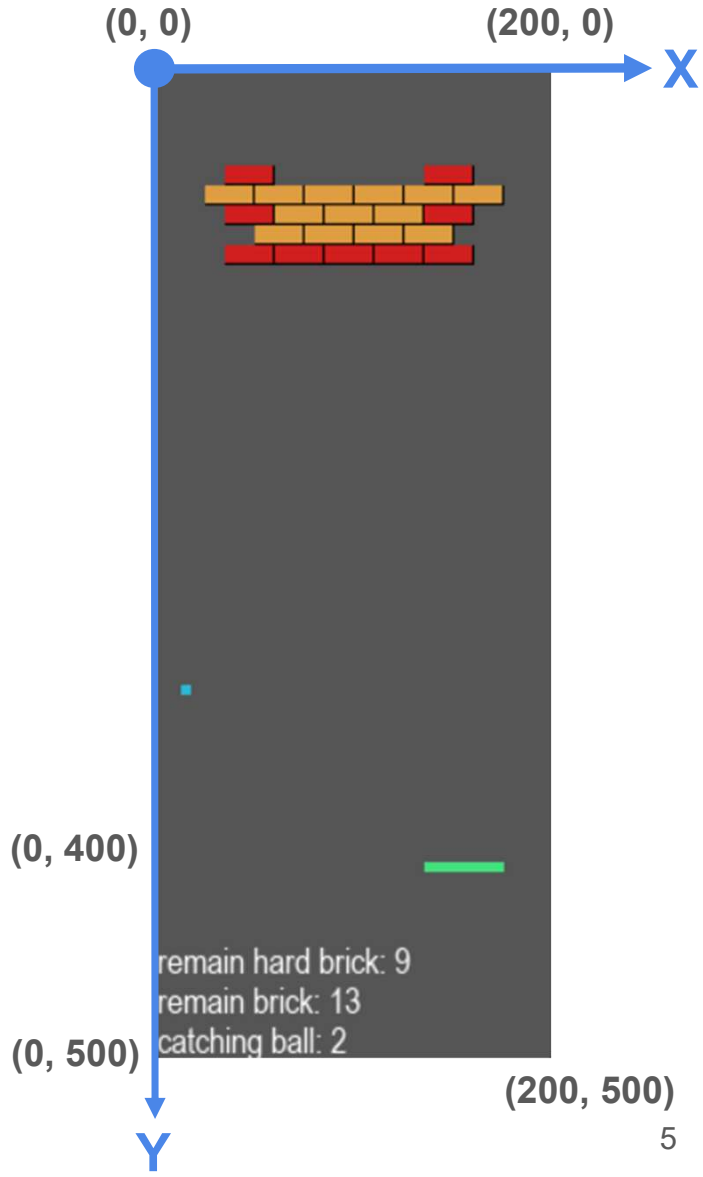
平台固定在y=400之處，長寬為40*5，每幀移動(±5, 0)。

球的邊長為5，每幀移動(±7, ±7)。

磚塊為25*10的長方形，硬磚塊(紅色磚塊)需要打兩次才會消失。

切球機制

在普通難度中，若球與平台碰撞時朝同方向前進，則球速會變成(±10, ±7)，並可以一次打掉硬磚塊。



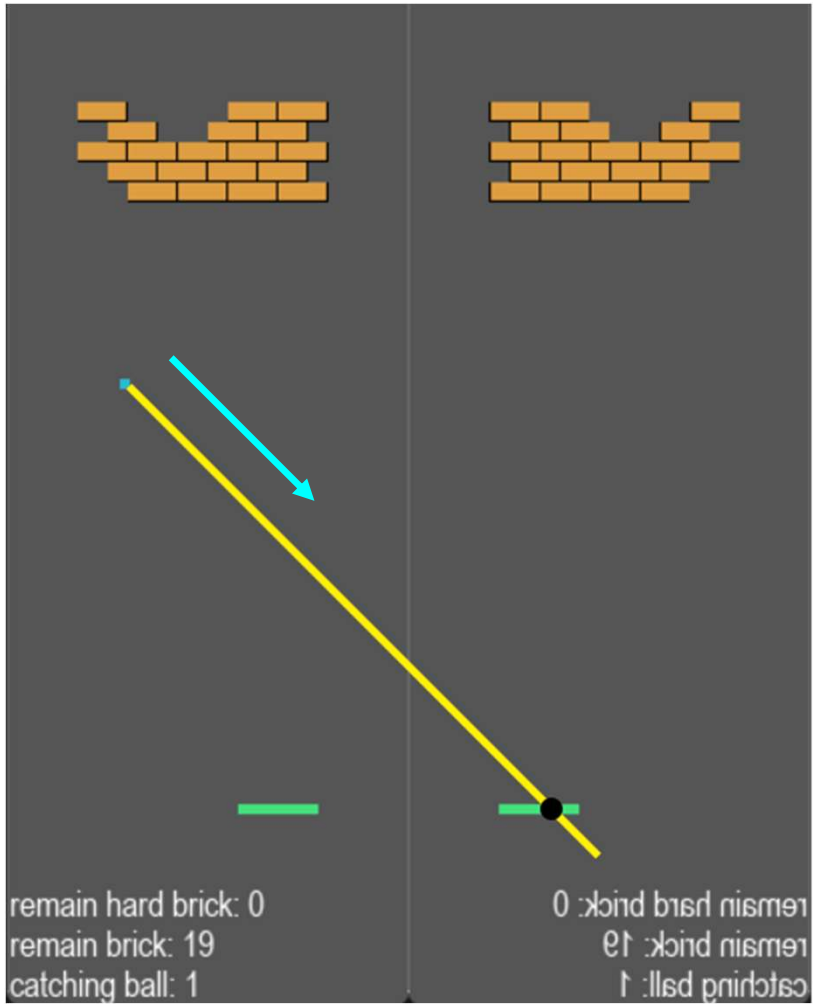
解題思路說明

解題策略

若入射角=反射角，以及球呈直線運動:

透過球的行進速度推算出直線方程式，推算出此直線與平台(y=400)的交叉點，操縱平台朝預測點前進。

註：請注意右幀是左幀的鏡像



解題策略

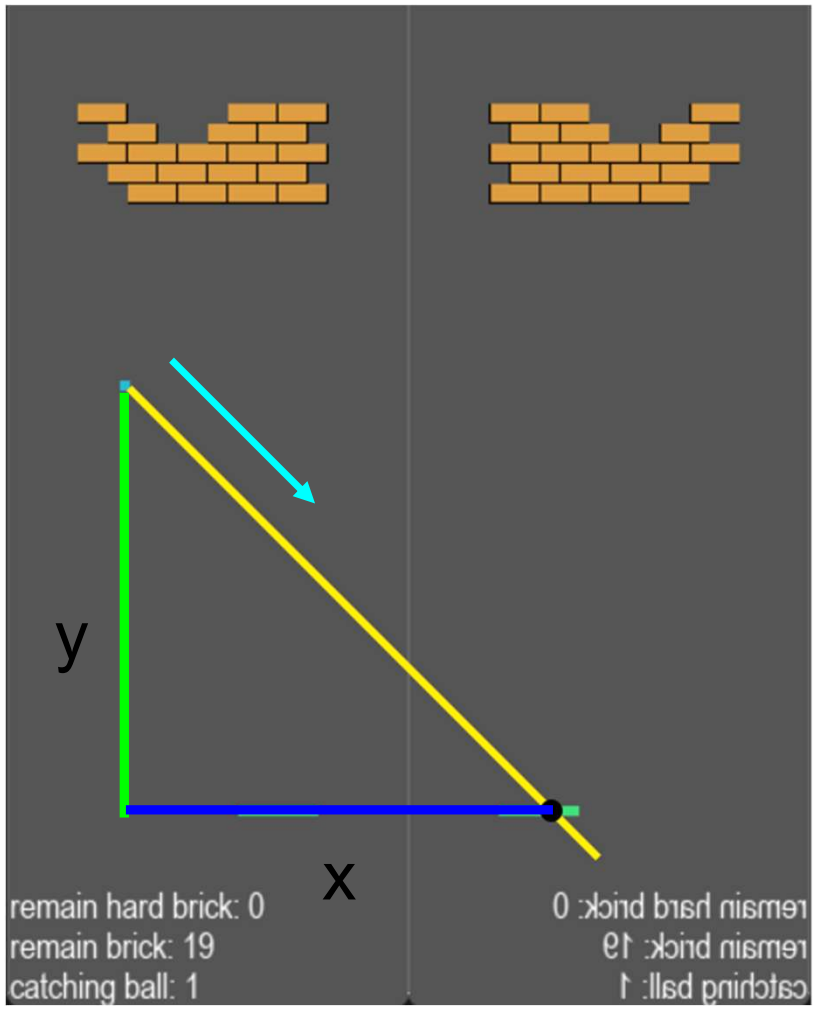
我們以直線方程式y=ax+b進行計算。以球目前的位置為原點(b=0)，故y=ax。

右圖距離y = (平台的y座標 - 球的y座標)

右圖距離x = (預測的x座標 - 球的x座標)

帶入y=ax並移項後可得

預測的x座標 = 球的x座標 + (平台的y座標 - 球的y座標) / 斜率



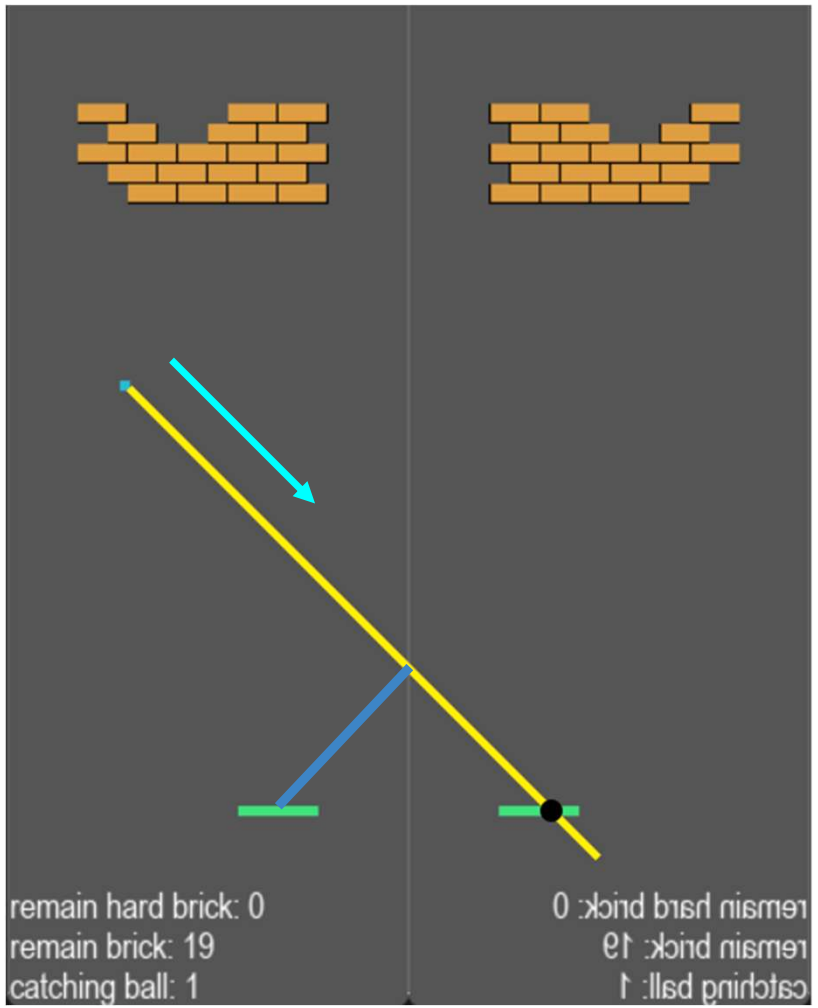
解題策略

在算出預測座標的x後，要依照反彈的次數(x / 200)
來計算出實際落地的座標。

可以發現如果反彈偶數次，
則預測座標為abs(x - 200*撞牆次數)。

若反彈為奇數次，要取鏡像座標，
即(200 - abs(x - 200*撞牆次數))。

得到座標後，就可以操縱平台前往座標，成功接球。



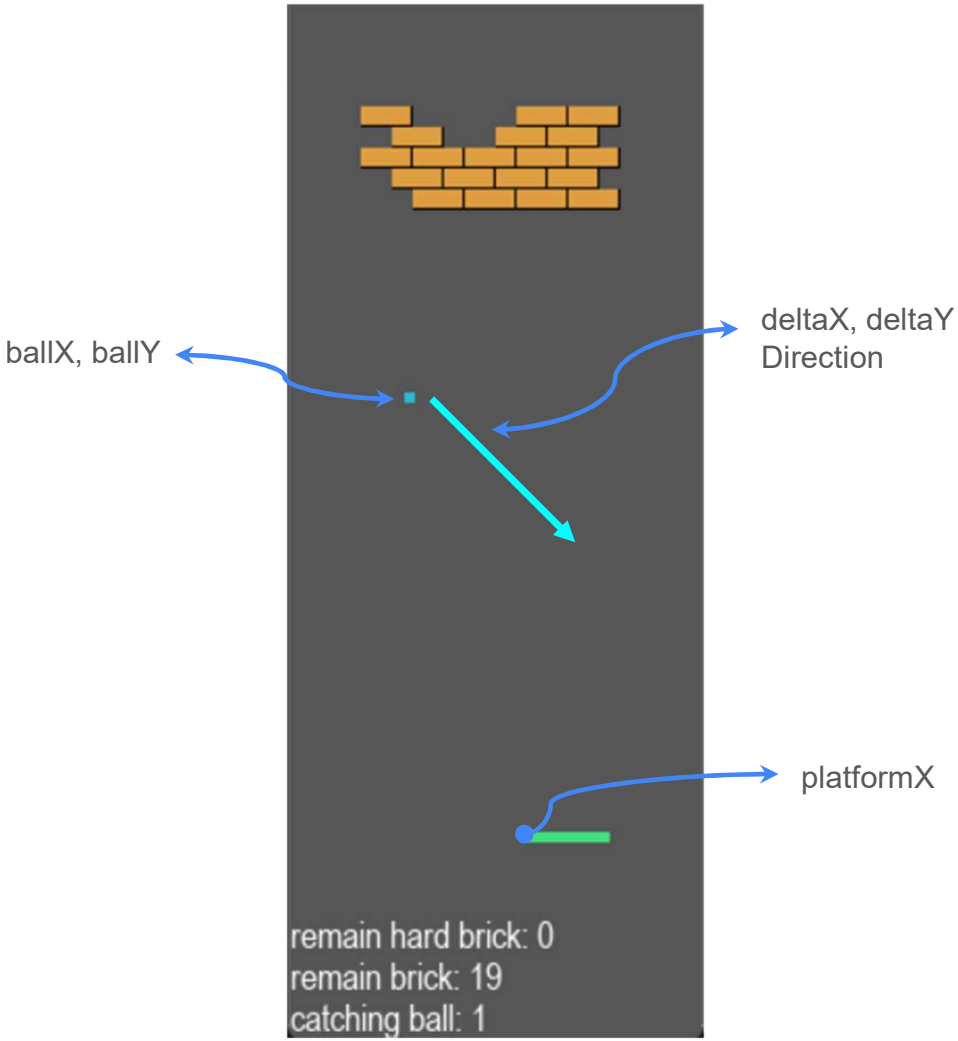
選擇特徵

- 球當前X/Y座標(ballX, ballY)
- 平台當前X座標(platformX)
  - Y方向固定所以不需要儲存

透過暫存上一幀球的位置資訊，可以計算出

- 球的移動方向(Direction)
  - 0(↘),1(↗),2(↙),3(↖)
- 球的X方向及Y方向速度(deltaX, deltaY)

注: 須注意在遊戲中，左上角座標為(0, 0)



選用模型及參數

範例使用模組: K-近鄰演算法(k-nearest neighbors, KNN)

- 這個演算法透過取得最近的k個樣本的類別，來分類預測目標。

- 可以透過迴圈帶入不同的k(neighbors)的數量，比較準確率來得到最佳的參數。
  - 建議可以在k=1~5之間尋找最佳參數。

KNN參考資料: 文章1 影片1 影片2

實作步驟

蒐集特徵

根據上述的策略成功控制遊戲後，我們在reset()中將遊戲資訊及輸出的指令一併儲存至pickle檔案。

透過遊玩不同關卡，我們可以蒐集不同路徑的遊戲資料，增加準確率。

但要注意，如果蒐集過多資料，在訓練時會花較長時間，準確率在訓練後期成長的幅度小，所以要在資料量及準確率之間取得平衡。

建議每關通關2~3次，就可以有足夠的資料了。

數據清洗

提供正確的資料集供機器學習進行訓練是非常重要的，而我們要在蒐集資料的過程中去除錯誤的資料，並避免資料偏袒於，以確保好的機器學習成果。

提供幾個清洗資料的想法:

- 僅儲存表現良好場次
  - 如遊戲結束時，成功通關或剩下1個磚塊才保留紀錄
- 確保各個關卡有差不多數量的資料
  - 例如讓每個關卡都過關固定次數即結束

訓練模型

因為訓練過程與遊戲沒有關連，所以我們會建立一個新的檔案執行。

前處理

在開始訓練前，我們會進行資料的前處理，從pickle檔案(scene_info)中擷取我們需要的特徵，並且計算pickle檔中沒有的特徵，如本篇第10頁提到的移動方向及球的XY變量。

處理特徵後，我們也會將pickle檔案中指令的部分從文字轉換為數字，
分別為-1(向左)，0(不動)，1(向右)。

由於訓練模型僅負責接球，因此我們同時會去除有關發球的動作及資料。

訓練模型

處理完資料後，我們將這些特徵做成一個陣列儲存為特徵資料，並將指令儲存為移動結果，作為特徵(feature)及標籤(labels)來進行學習。

我們會用到sklearn中的train_test_split()進行資料分成訓練資料及測試資料，並使用KNeighborsClassifier()來建立模型，將模型存入pickle中。

驗證模型:

驗證模型的準確率:

我們使用sklearn的accuracy_score來判斷這個模型的準確度。將前面split出的測試資料算出準確分數。

實際測試模型:

複製ml_play_template.py，將模型帶入遊戲中，蒐集與模型相符的特徵後使模型預測出下一個動作。運行遊戲檢視結果。

結果分析

結果分析

預期結果:

- 這個模型可以在高勝率下通關多個關卡。

缺點:

- 若磚塊距離平台太近，將會因為落下時間太短及而無法抵達預測位置。
- 若磚塊在左右牆面上，會因為這方案並未考慮與磚塊的碰撞而預測出錯誤的行進路線。

FAQ

- 在執行策略時，平台會左右抖動?
  - 平台無法剛好抵達預測的座標(平台移動一次是±5)，所以會為了到指定的點而來回移動
  - 可以在決定移動指令的程式碼中增加一點誤差範圍
- 規則寫好但平台動不起來/動的幅度很小?
  - 有可能是忘記在結束時記住這幀球的位置，導致下一幀沒辦法計算正確的落點。

Happy Coding!