#Project2 Report

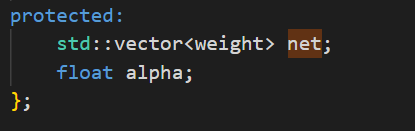
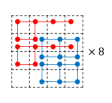
0816169 陳伯庭

1. Implementation

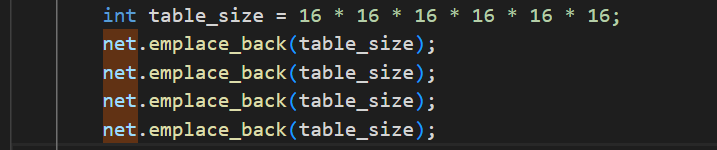
# N tuple Network

這次的project是用TD(0) afterstate learning加上n tuple network去訓練我們的agent。看似很難，但其實spec上面助教都幫我們把步驟寫得很清楚了，所以我就按照spec上面的講解去設計。

首先要先設計我們自己的tuple network。我這次設計了三種，分別是8 x 4、8 x 4 x 6、8 x 8 x 6。我就拿8 x 4 x 6來說明好了。N tuple network主要是用來衡量afterstate value。我在agent這個class底下開了一個變數叫做net來儲存weight table。

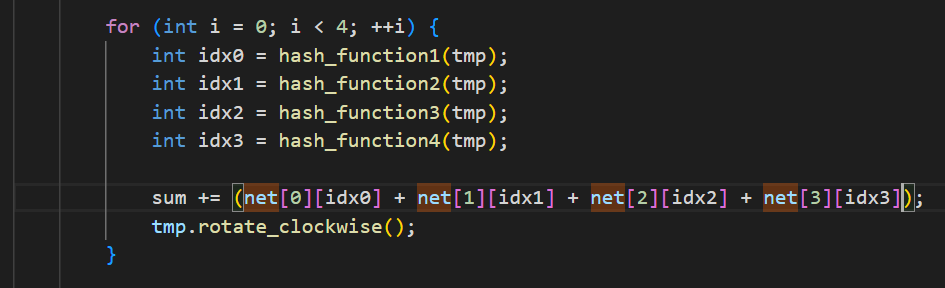
圖(一) 圖(二)

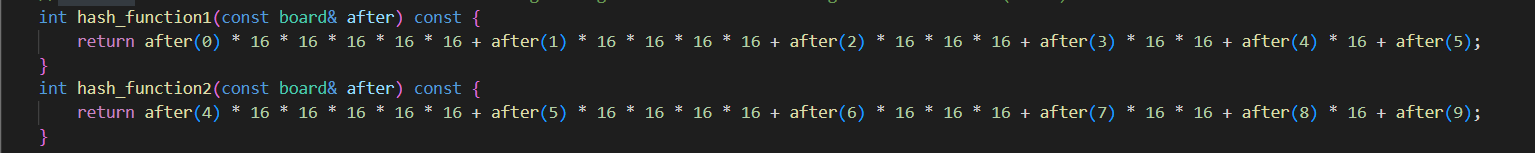
從圖二可以看到8x4x6每次總共有4個形狀要儲存，再加上我用了share weights所以最後只要開一個大小為4的list，分別去存不同形狀的就夠了。

圖(三)

那每一個index的weight table size，我參考了世界紀錄，發現他只到了12288，是第14個數字，我就決定取第16個數字來當作我的最大值，代表一個tile有16種可能。

接下來到了我們要怎麼mapping不同狀態的下的tuple。我用hash\_function這個自定義function去extract feature跟encode index。



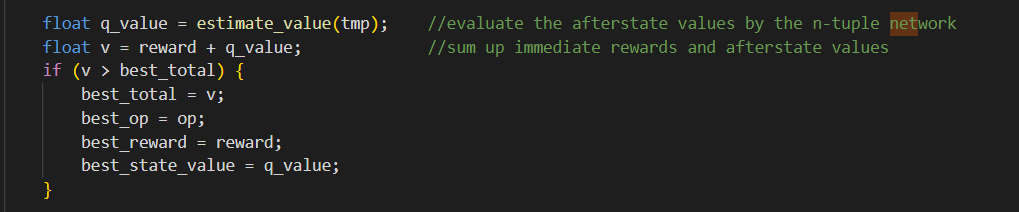
圖(五)

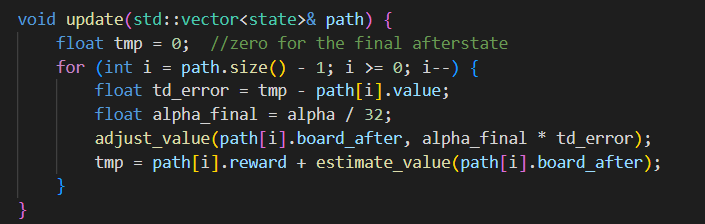
從圖(五)可以看到我們在不同的hash\_function裡面會用不同index的tile。這是根據圖(二)去取index的。

最後根據8x4x6 tuple network的定義，我們用了rotate\_clockwise產生四個方向的組合，再加上reflect\_horizontal後rotate\_clockwise的四種，共八種去計算跟調整weight。

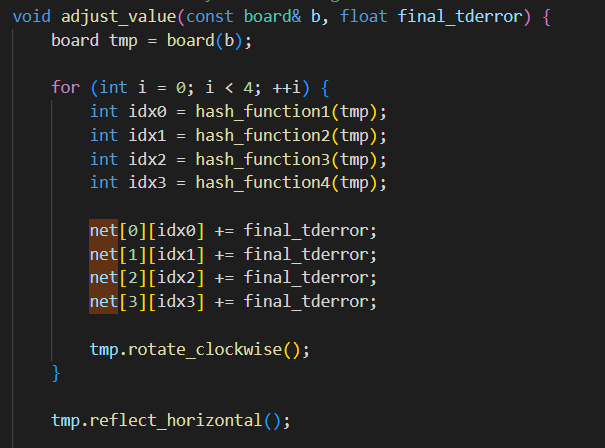
# TD learning

那有了weight table，我們就可以根據TD learning來去train他了。TD learning跟n tuple的關係主要在兩個部分，更新weight table跟選擇下一步。選擇的部分，他會取產生immediate reward加上我們用n tuple network去衡量的afterstate value的最大值。

Immediate reward我們就用助教提供的，afterstate value我用了estimate\_value這個自定義的function去計算。Estimate\_value我將四種形狀、八個方向在weight table裡面的值全部加起來。

最後是update的部分

Update會用到reward加上我們計算出來的afterstate value，減去實際上的下一個state的value，得到TD error後，乘上我們的learning rate去做調整。可以看到我們將alpha\_final \* td\_error丟進adjust\_value這個function。

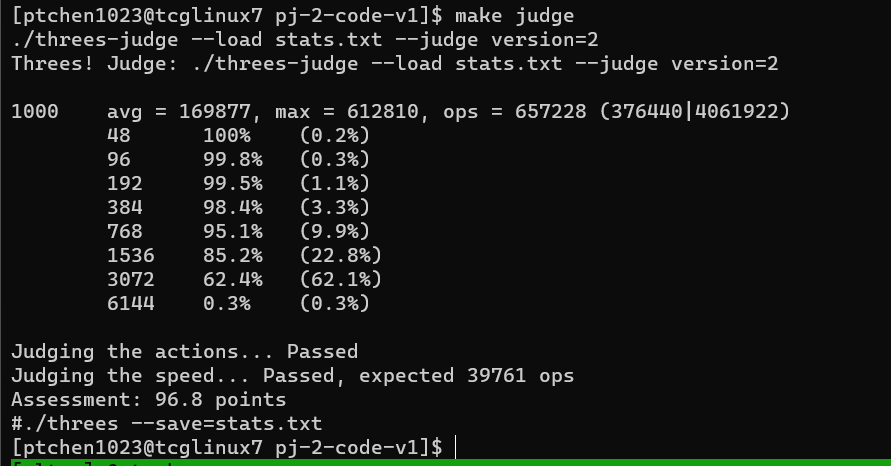


在adjust function裡面，我們一樣用hash\_function得到index後，在weight table裡面加上我們剛剛算出的alpha\_final \* td\_error，就完成調整weight table這個步驟了。

1. Result

這是4x6 tuple network的情況下在前300萬場alpha=0.1，再train 150萬場alpha=0.01，再train 100萬場alpha=0.0001的成績。

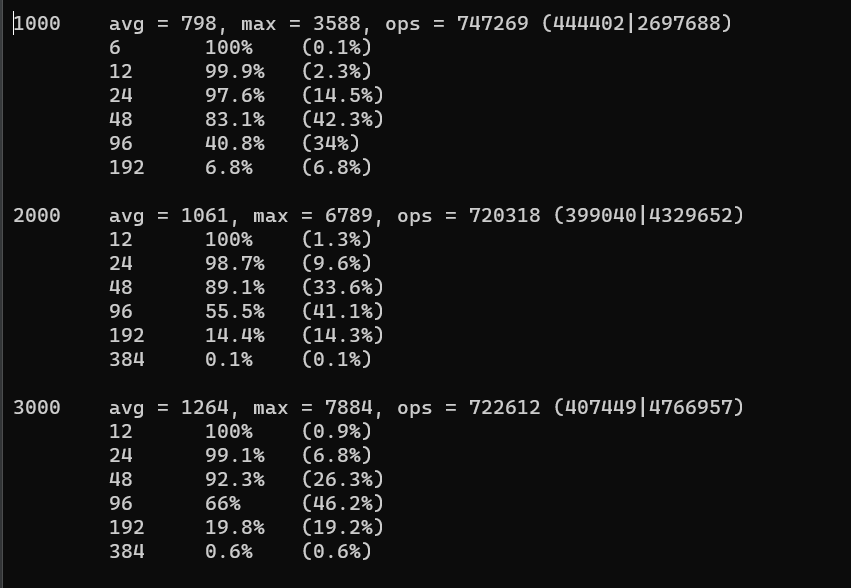
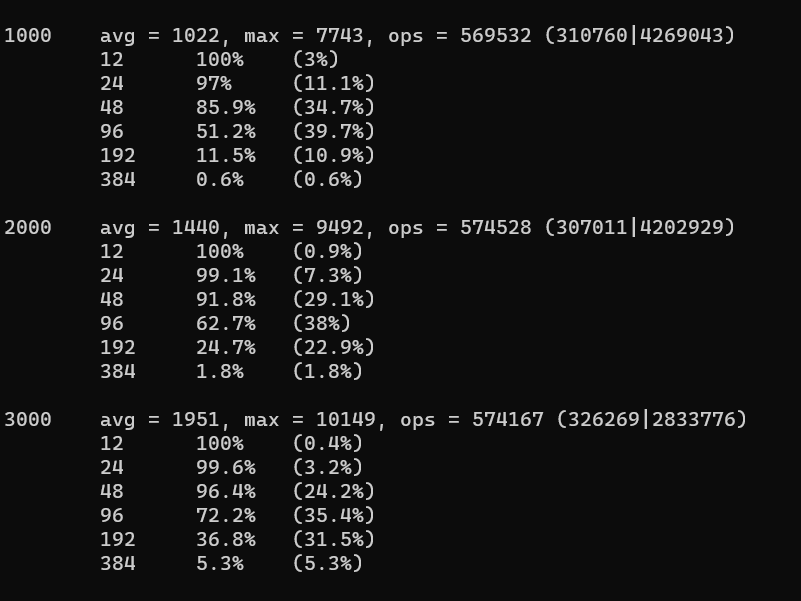
(這邊提到的alpha值要再除以32才是真正使用的learning rate)



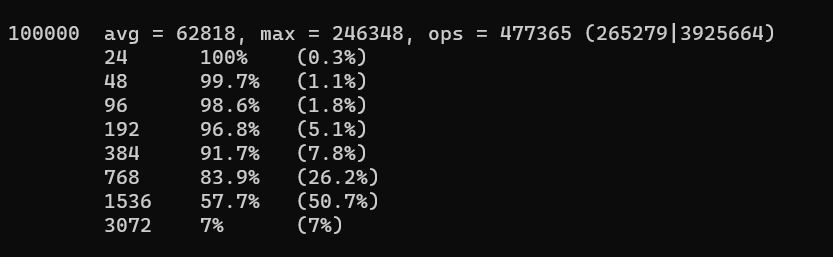
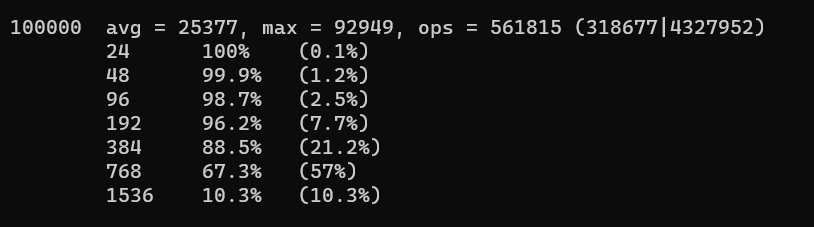
1. Experiment

#Comparison between different N tuple network

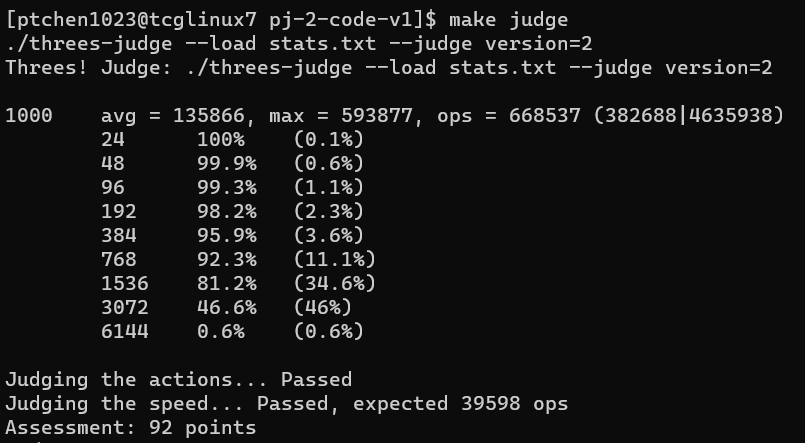
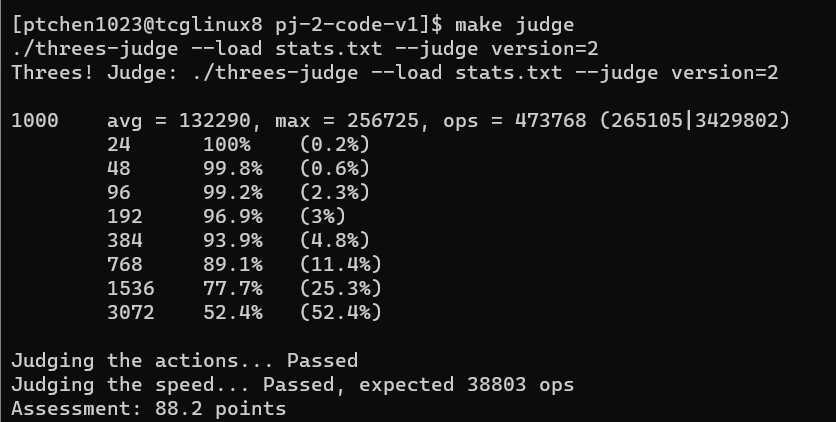
透過簡單的實驗，我們可以看到在相同的環境下，前面幾千次的training過程中，8x8x6(左圖)的表現會比8x4x6(右圖)好很多，但是就會耗掉比較久的時間做相同場數的訓練。



場次到了十萬場，差距更明顯了。



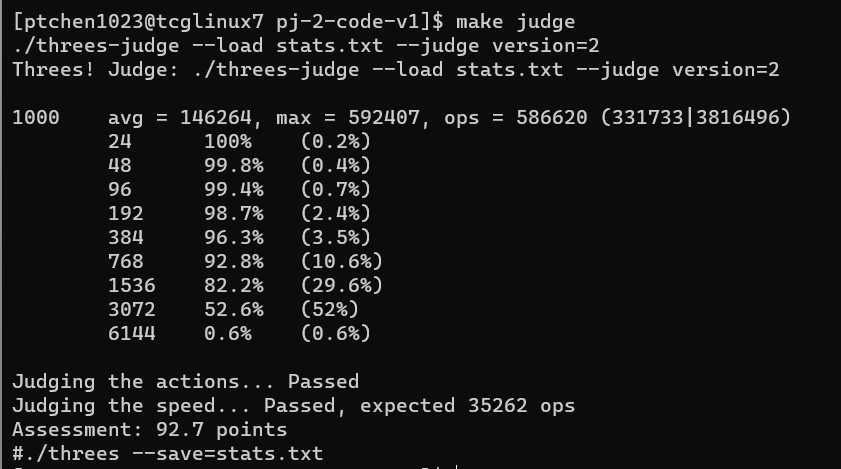
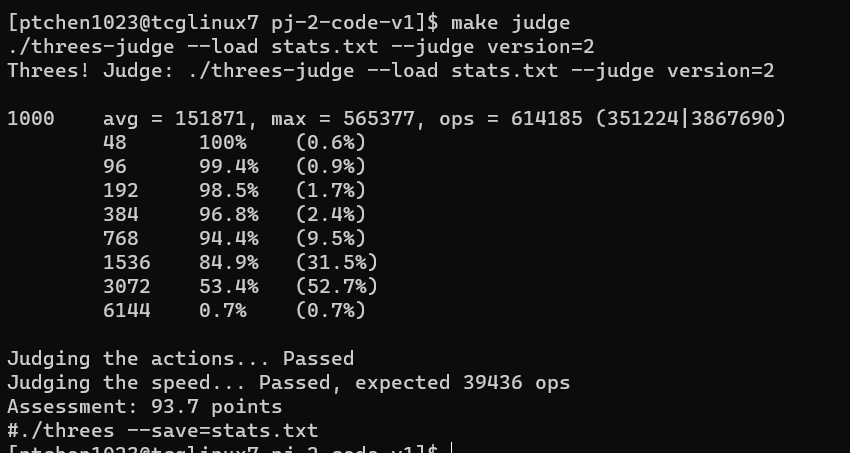
這時就出現了一個有趣的事情了，沒想到在經過50萬場後，8x8x6的達到384的比例竟然比8x4x6的少，看來她也是比較早開始效能下降的，必須提早降低他的alpha。



#comparison between different learning rate(alpha)

首先，這邊提到的alpha都要再除以32，但我是在程式內部除32的，所以在指令上是實際上的learning rate的32倍。

以8x4x6 tuple network為例，先是經過相同的alpha = 0.1訓練500萬場，再經過一個調低alpha = 0.01(左圖)，另一個維持0.1(右圖)，可以看到訓練一段時間後調低alpha對於performance是有一定的幫助的。



除了8x8x6有一樣的效果外，我也發現太快降低alpha也會對效能有反效果，要等一定時間的訓練後再調低會較合適。

1. Conclusion

這次的作業我覺得助教跟教授設計得很好，透過自己設計n tuple network加上基礎的TD(0) afterstate learning，讓我們對上課所學的東西有更深刻的體悟，不然教授上課講得我其實都一知半解。我覺得要拿到90分不會太難，但不知道為甚麼我東調西調都train不到滿分100，可能是要經過長時間的實驗去動態調整learning rate，或是用更厲害的TD learning algorithm吧！總而言之，仍舊是個很有趣的project，所謂的做中學，收益良多。