姓名：黃展皇

學號：110621013

1. (10%) Policy Gradient 方法

請閱讀及跑過範例程式，並試著改進 reward 計算的方式。

請說明你如何改進 reward 的算法，而不同的算法又如何影響訓練結果？

在reward計算的部分主要改動的是這一條code：



在原先的策略中是考慮完一整局投資(episode)後，其得到的total reward就是每個action的reward。而在改進reward算法上老師投影片提到兩種tip，分別是Baseline以及Suitable Credit (gamma)，**兩者皆有嘗試並測試，且還有兩者的結合測試**，另外**Baseline也有針對不同的數值進行測試(0.1、長期趨勢線0.1634、1.0)**，自己想的方法也有兩個，分別為每個action只看當下reward的**短視近利法(single)**，以及**波段reward(****band reward)算法**，即：一段時間(波段)的利益率最佳算法應該是(賣出價格-買入價格)/持有時間，將action為1時的reward設為利益率(或是利益率\*10，僅測試用)，action為0時的reward設為0。

以上reward算法又分別對於['Close','Open']**兩個feature**s以及['Shares','Amount','Open','High','Low','Close','Change','Turnover']**八個feature**s做檢驗，執行test.py可以得到以下：

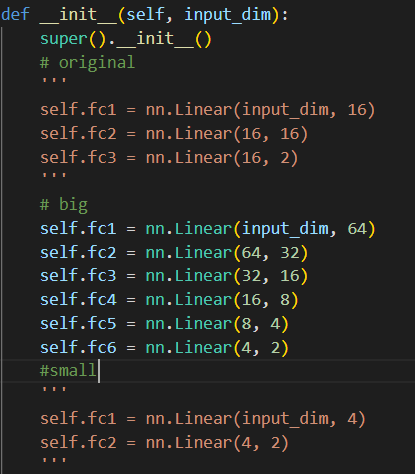
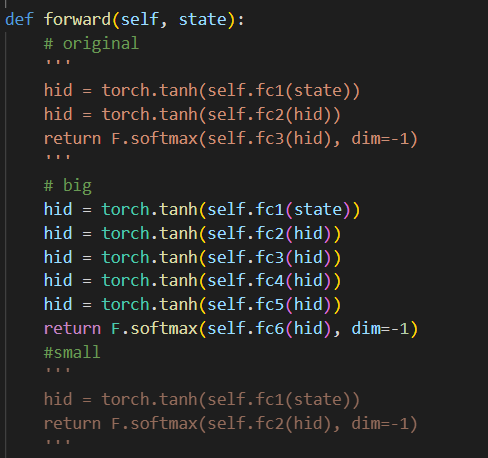
* 原total reward as all reward算法：
  + 只用2 feature："total\_reward": 18.4, "total\_profit": 1.1875138529091713，接近全持有僅3筆賣出
  + 用全8 feature："total\_reward": 29.4, "total\_profit": 1.0338217658470048，接近全持有僅17筆賣出
* Baseline算法：
  + 用全8 feature，Baseline=0.1634："total\_reward": 29.4, "total\_profit": 1.0338217658470048
* Suitable Credit (gamma)算法：
  + 只用兩feature，gamma=0.9：output\_gamma\_reward："total\_reward": 18.0, "total\_profit": 0.28983855519801993，頻繁交易
  + 用全8 feature，gamma=0.9："total\_reward": 9.5, "total\_profit": 0.19616716656007824，頻繁交易
* Baseline + Suitable Credit (gamma)算法：
  + 用全8 feature，gamma=0.9，Baseline=0.1："total\_reward": 12.3, "total\_profit": 0.20515171645320274，頻繁交易
  + 用全8 feature，gamma=0.9，Baseline=0.1634："total\_reward": 9.1, "total\_profit": 0.1380516136024575，頻繁交易
  + 用全8 feature，gamma=0.9，Baseline=1.0："total\_reward": 2.2, "total\_profit": 0.13012830750074414，頻繁交易
* Single算法：
  + 只用兩feature：output\_gamma\_reward："total\_reward": 1.3, "total\_profit": 0.9542408566646335，接近全賣出4持有
  + 用全8 feature："total\_reward": 2.2, "total\_profit": 0.16138687348747074，頻繁交易
* band reward算法：
  + 用全8 feature，一般波段reward："total\_reward": 26.9, "total\_profit": 0.5468846400183093，偏向持有(僅數十筆賣出)
  + 用全8 feature，波段reward\*10："total\_reward":29.3, "total\_profit": 0.7808031971608166，偏向持有(僅數十筆賣出)

由以上可得知：

1. total\_reward最高者為**原total reward as all reward算法的8 features** 以及Baseline算法的8 features (29.4)，次高為band reward算法的8 features 波段reward\*10 (29.3)。
2. total\_profit大於1.0者有**原total reward as all reward算法的只用2 feature (1.1875)** 以及 8 features (1.0338)，以及Baseline算法的 (1.0338)。
3. 原total reward as all reward算法傾向持續持有，加入多feature可增加賣出頻率並大幅增加total\_reward，但小幅減少total\_profit。
4. Baseline算法在Baseline設為長期趨勢線的0.1634之下跟原total reward as all reward算法一模一樣。
5. Suitable Credit (gamma)算法以及Baseline + Suitable Credit (gamma)算法會有非常頻繁的交易，並且表現不佳。
6. Single算法在2 feature情況傾向不持有，有不錯的total\_profit；8 feature情況下頻繁交易，效率變差很多。
7. band reward算法表現近似於原total reward as all reward算法，且在買賣頻率上在資料區間有數十筆波段買賣，算是在實務操作上相對合理的算法。
8. (15%) 試著修改與比較至少三項超參數（神經網路大小、一個 batch 中的回合數等），並說明你觀察到什麼。

超參數選擇更改神經網路大小、一個 batch 中的回合數以及frame\_bound的選取區間。

\* 神經網路大小部分將原先的神經網路設計分別淺化/加深並定義為small/big NN，分別對1.中表現較佳的三種模型(model、model\_all\_feature、model\_band\_10times)進行不同NN深度的修改：

* model\_smallNN："total\_reward": 30.7, "total\_profit": 1.2829499785446996，僅1筆賣出
* model\_bigNN："total\_reward": 26.6, "total\_profit": 0.9544167816572734，22筆賣出
* model\_all\_feature\_smallNN："total\_reward": 16.5, "total\_profit": 0.26126994406135545，傾向多持有，頻繁交易
* model\_all\_feature\_bigNN："total\_reward": 29.7, "total\_profit": 1.2121626206786262，僅6筆賣出
* model\_band\_10times\_smallNN："total\_reward": 29.7, "total\_profit": 0.598246618639677，傾向多持有
* model\_band\_10times\_bigNN："total\_reward": 11.5, "total\_profit": 0.5324529337750625，傾向多持有

由以上可得知：

* + 對於原total reward as all reward算法 2 feature而言，原先"total\_reward": 18.4, "total\_profit": 1.1875138529091713，total\_reward無論是NN變大/變小皆有提升，total\_profit則是在小一點的NN有較好的表現。
  + 對於原total reward as all reward算法 8 feature而言，原先"total\_reward": 29.4, "total\_profit": 1.0338217658470048，total\_reward在小NN時降低很多，大NN時有提升，total\_profit表現類似，證明**越多feature需要越深NN**。
  + 對於band reward算法 8 feature而言，原先"total\_reward": 18.4, "total\_profit": 1.1875138529091713，total\_reward無論是變大/變小皆有提升，total\_profit則是在小一點的NN有較好的表現。

\* 一個 batch 中的回合數部分選擇改變EPISODE\_PER\_BATCH的數量，改成1次(變少)與10次(變多)，分別對1.中表現較佳的三種模型(model、model\_all\_feature、model\_band\_10times)進行修改：



* model\_EPISODEPERBATCH\_1："total\_reward": -5.8, "total\_profit": 0.14681082002860824，無傾向
* model\_EPISODEPERBATCH\_10："total\_reward": 17.7, "total\_profit": 1.1442317050596507，僅5筆賣出
* model\_all\_feature\_EPISODEPERBATCH\_1："total\_reward": 17.7, "total\_profit": 0.2558831259190814，無傾向
* model\_all\_feature\_EPISODEPERBATCH\_10："total\_reward": 19.0, "total\_profit": 0.441220550476819，傾向持有
* model\_band\_10times\_EPISODEPERBATCH\_1："total\_reward": 23.4, "total\_profit": 0.5406089627959313，傾向持有
* model\_band\_10times\_EPISODEPERBATCH\_10："total\_reward": 23.7, "total\_profit": 0.7560904654410261，傾向持有

由以上可得知：

* + 對於原total reward as all reward算法 2 feature而言，原先"total\_reward": 18.4, "total\_profit": 1.1875138529091713，total\_reward在EPISODEPERBATCH\_1甚至是負的，相當不穩定，total\_profit則是差很多；EPISODEPERBATCH\_10則與原先的EPISODEPERBATCH\_5差不多(total\_reward微降)。
  + 對於原total reward as all reward算法 8 feature而言，原先"total\_reward": 29.4, "total\_profit": 1.0338217658470048，total\_reward和total\_profit在EPISODEPERBATCH\_1或EPISODEPERBATCH\_10表現都不佳。
  + 對於band reward算法 8 feature而言，原先"total\_reward": 18.4, "total\_profit": 1.1875138529091713，total\_reward和total\_profit在EPISODEPERBATCH\_1或EPISODEPERBATCH\_10表現都更好。

\* frame\_bound選取區間部分改變MyStocksEnv的frame\_bound，從(1000, 1500)改成(1500, 2000)(換選取範圍), (1000, 1100) (縮小選取範圍)，分別對1.中表現較佳的三種模型(model、model\_all\_feature、model\_band\_10times)進行修改：





* model\_change："total\_reward": 26.6, "total\_profit": 0.9072735422869351，22筆賣出
* model\_minify："total\_reward": 27.1, "total\_profit": 0.8905167668958032，26筆賣出
* model\_all\_feature\_change："total\_reward": 1.6, "total\_profit": 0.2491160007658047，傾向持有
* model\_all\_feature\_minify："total\_reward": 20.9, "total\_profit": 0.5909762568405135，傾向持有
* model\_band\_10times\_change："total\_reward": 23.9, "total\_profit": 0.5360988541428936，傾向持有
* model\_band\_10times\_minify："total\_reward": 20.8, "total\_profit": 0.5221147988667466，傾向持有

由以上可得知：

* + 對於原total reward as all reward算法 2 feature而言，原先"total\_reward": 18.4, "total\_profit": 1.1875138529091713，無論是change 還是minify的情況，total\_reward有顯著的提升但total\_profit有所下降。
  + 對於原total reward as all reward算法 8 feature而言，原先"total\_reward": 29.4, "total\_profit": 1.0338217658470048，無論是change 還是minify的情況，total\_reward和total\_profit表現都不佳，尤其以change更為嚴重。
  + 對於band reward算法 8 feature而言，原先"total\_reward": 18.4, "total\_profit": 1.1875138529091713，無論是change 還是minify的情況，total\_reward有小幅的更好，但total\_profit從1.0以上降到0.5左右，表現顯著下降。

1. (15%) 請同學們從 Q Learning、Actor-Critic、PPO、 DDPG、TD3 等眾多 RL 方法中擇一實作，並說明你的實作細節。

在這邊採用stable\_baselines3的演算法套件(A2C, DQN, PPO)，並參考針對gym\_anytrading的範例訓練，其中feature依然分為['Close', 'Open'] 2 feature以及['Shares','Amount','Open','High','Low','Close','Change','Turnover'] 8 feature分別嘗試，my\_process\_data以及MyStocksEnv跟PolicyGradientAgent訓練時一樣，而訓練僅用到三行：

model = DQN('MlpPolicy', env, verbose=1) #A2C, DQN, PPO

model.learn(total\_timesteps=100000)

model.save("DQN\_MlpPolicy\_100000")

其中模型可能是(A2C, DQN, PPO)，而Policy採用'MlpPolicy'，total\_timesteps皆為100000。結果如下：

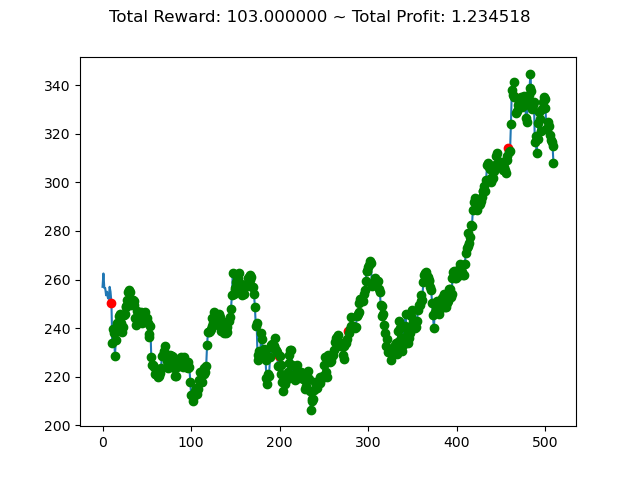
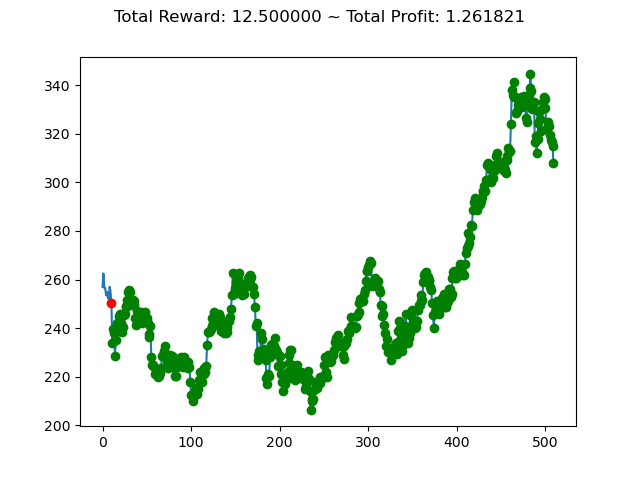
* A2C\_MlpPolicy\_100000："total\_reward": 7.0, "total\_profit": 0.1503470382827645，持有/賣出均衡
* A2C\_MlpPolicy\_100000\_all\_feature："total\_reward": -20.5, "total\_profit": 0.13058173352993843，持有/賣出均衡
* DQN\_MlpPolicy\_100000："total\_reward": 32.0, "total\_profit": 0.3364423991078638，傾向賣出，123筆持有
* DQN\_MlpPolicy\_100000\_all\_feature："total\_reward": 73.5, "total\_profit": 0.41885424962328466，傾向賣出，107筆持有
* PPO\_MlpPolicy\_100000："total\_reward": 34.0, "total\_profit": 0.3965528057191327，傾向持有，82筆賣出
* PPO\_MlpPolicy\_100000\_all\_feature："total\_reward": 3.0, "total\_profit": 0.13459356189174448，持有/賣出均衡

由以上可得知：

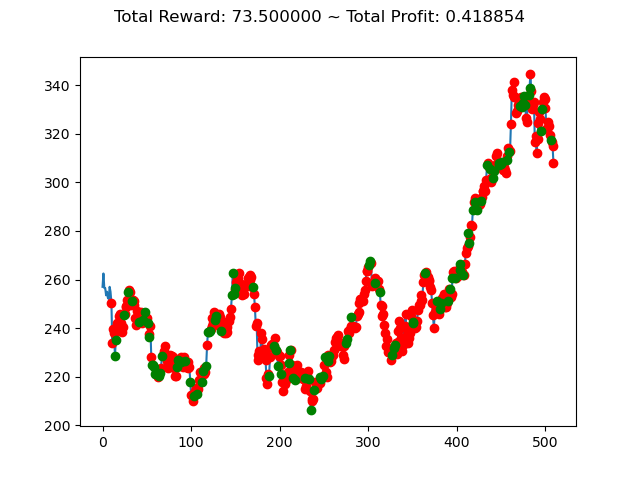
* + 對於A2C以及PPO的\_all\_feature(8 feature)情況，total\_reward以及total\_profit相較於2 feature顯著變差，甚至會有負的total\_reward，因此這兩者接下來只需討論2 feature情況，不過DQN中total\_reward有顯著上升。
  + total\_reward以及total\_profit最高者為DQN 8 feature(73.5, 0.4189)，其次PPO 2 feature也有(34.0, 0.3966)。
  + 各種模型產生的交易傾向也不盡相同，例如A2C會有頻繁交易的情況，DQN傾向賣出，PPO則在頻繁交易的情況下傾向持有。

1. (10%) 請具體比較（數據、作圖等）你實作的方法與 Policy Gradient 方法有何差異，並說明其各自的優缺點為何。

最佳的Policy Gradient做法：最高total\_reward，同時也是最高total\_profit：2 feature用小NN(30.7, 1.2829)、8 feature用大NN(29.7, 1.2122)，下兩圖為測試資料區間的買賣情況，左側為2 feature用小NN(1筆賣出)，右側為8 feature用大NN(6筆賣出)：



而套件方法中total\_reward以及total\_profit最高者為DQN 8 feature(73.5, 0.4189)，下圖為測試資料區間的買賣情況(傾向賣出，107筆持有)：



比較Policy Gradient做法以及套件方法可以知道，對於total\_reward來說套件方法73.5>30.7，顯然對於2000~2500的frame\_bound而言套件方法較好；而對於total\_profit來說套件方法0.4189<1.2829，顯然對於2000~2500的frame\_bound而言Policy Gradient方法較好。

這直接讓我們陷入兩難，是應該以total\_reward還是total\_profit做為參考依據呢？我們發現total\_profit本身是有收取交易的手續費的，也就是說在同樣的reward之下，更頻繁的交易會導致真實利潤profit較低。因此如果你是一位長期投資者(以年為單位)，可用Policy Gradient做法避免多次操作；反之若是波段餐作者(以天/禮拜/月為單位)，則可用套件方法得到更為頻繁的交易操作建議。