學號：R08946006　　系級：資料科學學程碩一　姓名：　周逸平

*ps1：此處 discount PG 以及 non discount PG 訓練寫於 tutorial.py
*ps2：此處 a2c 訓練寫於 A2C.py
*ps3：此處所有圖都利用 make_plot.py 做出，需要 repo 中所有 npy 檔案

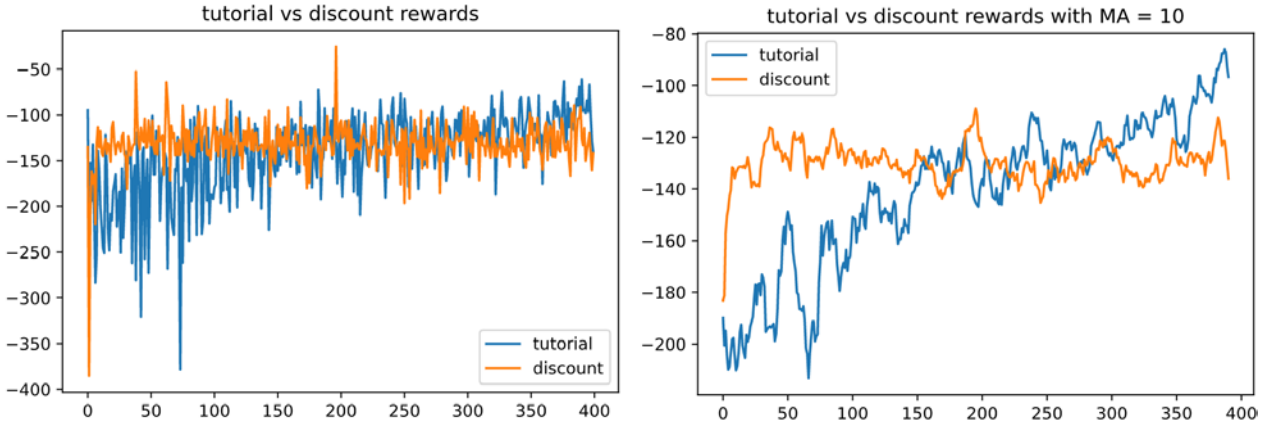
1. (20%) Policy Gradient 方法

    a. 請閱讀及跑過範例程式，並試著改進 reward 計算的方式。

    b. 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？

使用的是 discount reward 方法，但並無減去 baseline。

基本上就是實現老師的 tip2，將 reward array 反轉後一筆一筆疊加後 insert 進新 reward array 的首項，疊加前必須先乘上一個 discount factor



以下有使用 discount 方法的 PG 稱作"discount"，未使用的稱作 PG，由於左方作圖過於雜亂，故我使用了 MA=10 的方法讓曲線 smooth 一點後做右圖。

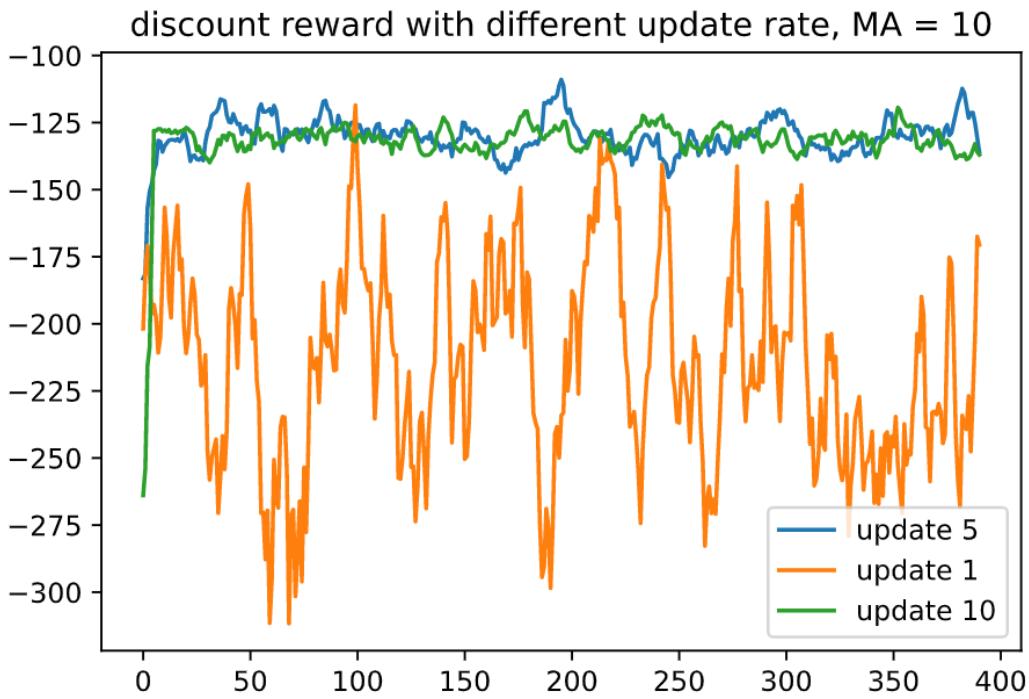
可以看到有使用 discount 方法的 PG 明顯較沒使用的可以更迅速的抵達收斂區域，相較之下 PG 花了約 200 個 episode 才爬到跟 discount 一樣的高度，但 discount 在後面幾個 episode 卻無法像 PG 一樣成績穩定成長，猜測可能是 discount 方法由於增加了 state 間 reward 的相依性，故若是"某幾個"起始 state 做的不好，相對應的會影響後續的 state 之 action。

2. (30%) 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。

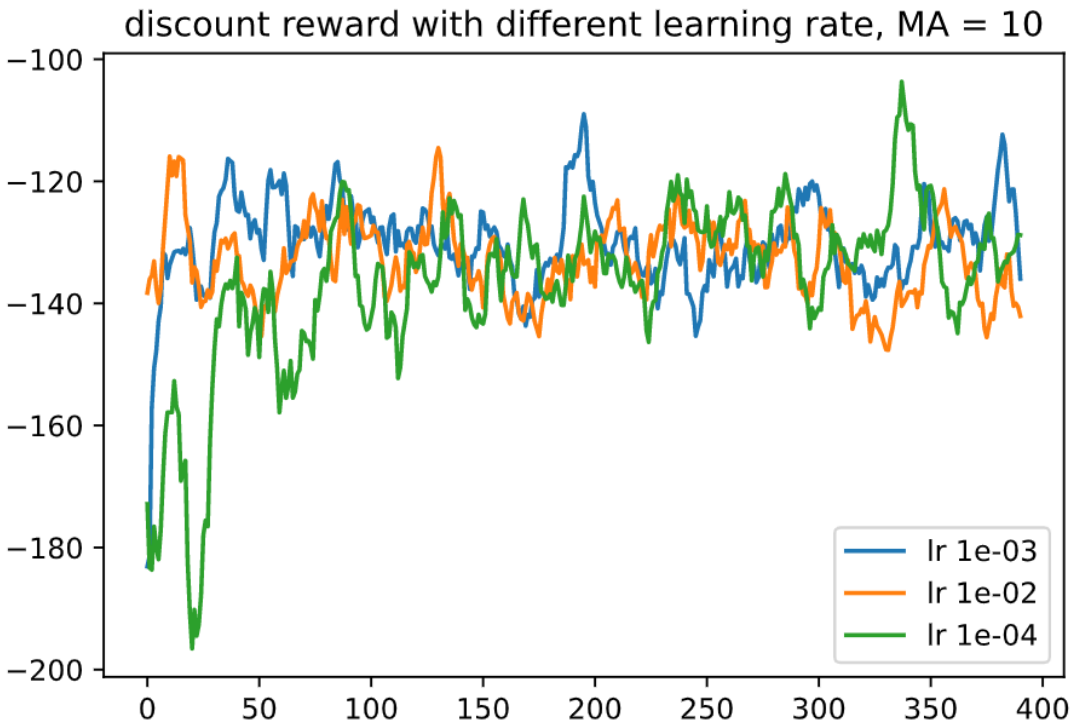
此處以 discount 方法分別修改了以下三項超參數

1. episode per batch : 1, 5, 10
2. learning rate: 1e-02, 1e-03, 1e-04
3. num_batch : 200, 400, 600

==========



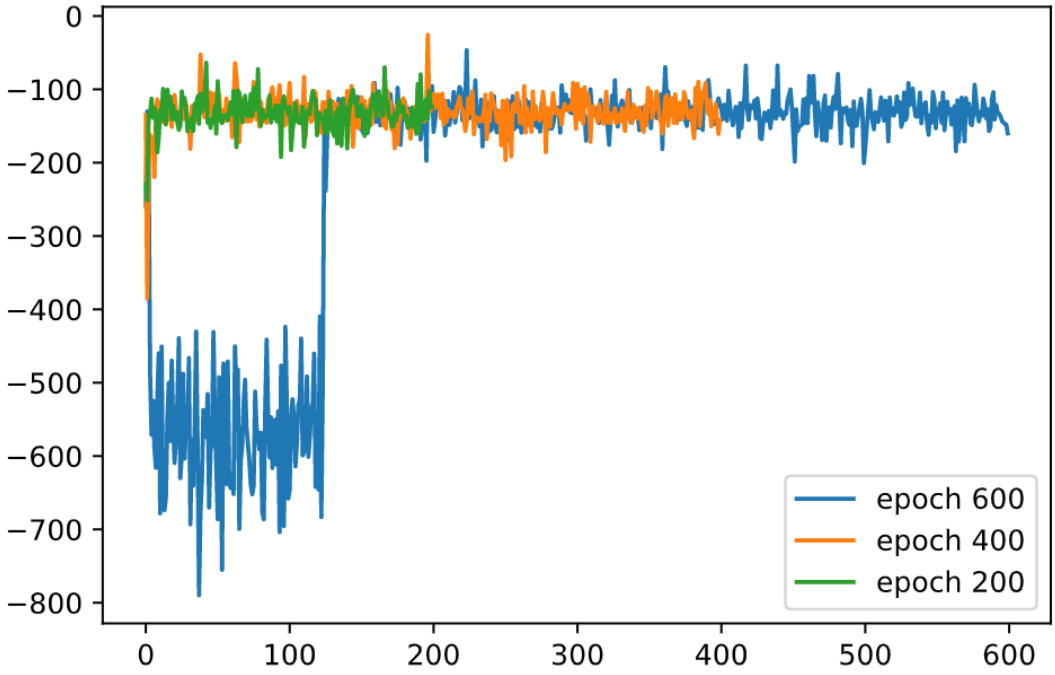
上圖為固定 num_epoch = 400, learning_rate = 1e-03 的情況下更動 episode_per_batch 的圖，可以看到將此數值調高，PG 在一次的 update 中就可以參照更多 episode 的過程，所以如果只設定 1，會導致訓練過於浮動，相較於 update 5 跟 10，越多的 episode per batch 可以提供一個更 smooth 的 distribution 讓 PG 學習，但由於 10 所耗費的時間是 5 的兩倍，而從圖上可以看到效果並無太大差異，故此處認為使用 5 是比較好的選擇。



此圖為固定 num_batch = 400, episode_per_batch = 5 下更動 learning rate 的圖。

可以看到三種不同的 learning rate 其實都可以讓 PG 趨於穩定，但可以注意到 1e-04 的情況下，需要花約 100 次更新才可以達到跟 1e-02, 1e-03 差不多的區域，而後續的訓練似乎也沒有帶來相較 1e-02, -03 來說更穩定的學習曲線。

但 1e-02 的學習自一開始就幾乎沒有波動，相較於 1e-03 還帶來了數次超越 1e-02 的 reward，所以此處認定 1e-03 可以帶來相對 1e-04 更有效率的訓練，且對 1e-02 更有效的區域探索。



此圖為固定 learning_rate = 1e-03, episode_per_batch = 5 下更動 learning rate 的圖。

可以看到每個數量的 epoch 都可以讓訓練 reward 穩定落在單一區間內，惟 epoch 600 在起頭時摔到了很低的區塊，又在約 150epoch 時重新拉回與 200, 400 一樣的區間。

從這樣的表現我推論 epoch 數量並不會影響訓練 ”效率”，但由於 RL 本身就帶有一定的隨機成分存在，越多的 epoch 可以讓 model “不小心”踏入更高 reward 區域的可能性更高，雖然此處選擇 600 並沒有讓 model 成功達成比 200, 400 更高的 reward 區間，但有可能再往後訓練這是有可能發生的。所以 epoch 數量取決於使用者想讓 model 嘗試多久時間而定。

(20%) Actor-Critic 方法

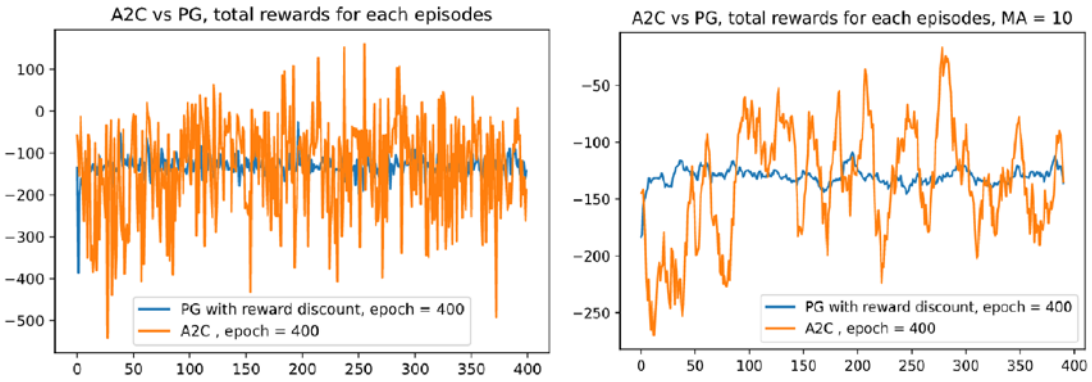
c. 請同學們從 REINFORCE with baseline、Q Actor-Critic、A2C 等眾多方法中擇一實作。

d. 請說明你的實做與前者（Policy Gradient）的差異。

2. (30%) 具體比較（數據、作圖）以上幾種方法有何差異，也請說明其各自的優缺點為何。

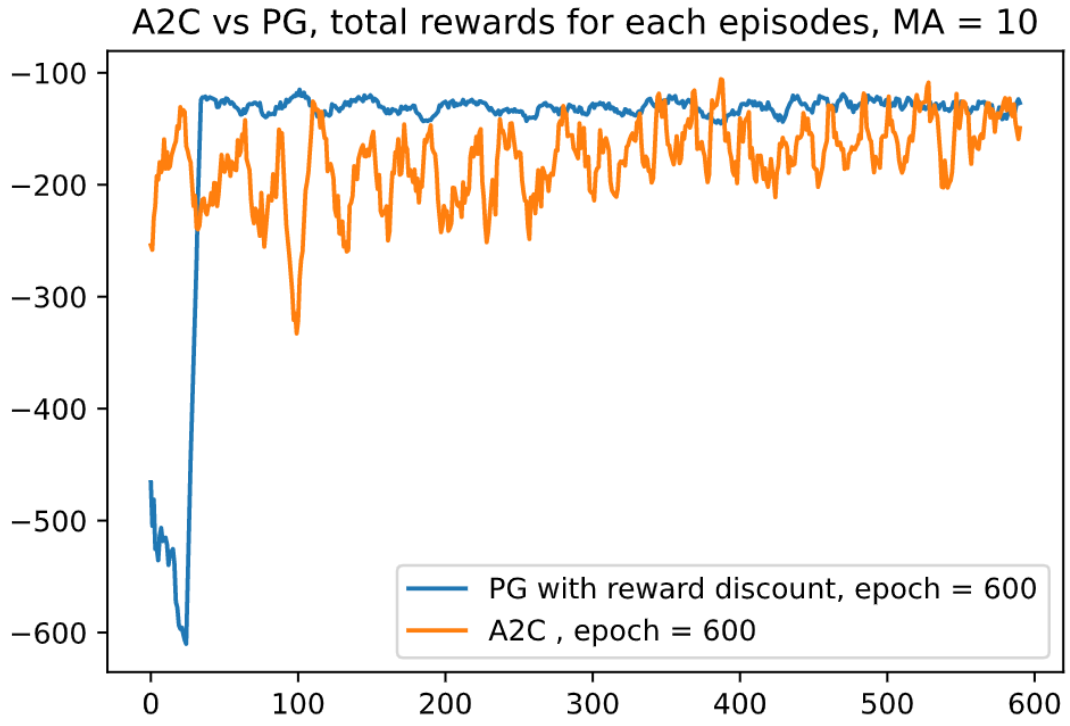
此處我選擇使用 A2C 作為實作
根據李老師在影片教學中所述，baseline 方法可以有效避免因 unbalance sampling 造成的 action 機率誤估，而該如何設定這一 baseline 變成了一個重要的問題。A2C 提出了一個合理的方法來預測此一 baseline，透過將 model 拆做兩個部分 actor 以及 critic，讓 actor 負責做出動作，讓 critic 來預測該局的 reward，稱做 advantage，並讓 actor 最後的 reward 減去 advantage 來達到此一效果。
而由於我用於與 A2C 比較的 PG 是有使用 discount 的，所以我寫了有 discount 效果的 A2C，以及由於 tutorial 的 PG 是採計 N 個 episode 後一齊 update 的，所以我讓 A2C 具有可以採計 N 步後一齊訓練的機制，設定如下；

	A2C	PG with discount
optimizer	SGD	SGD
Learning rate	1e-03	1e-03
Num batches	400	400
Episode per batch	-	5
Steps per update	200	-
Discount factor (gamma)	0.99	0.99

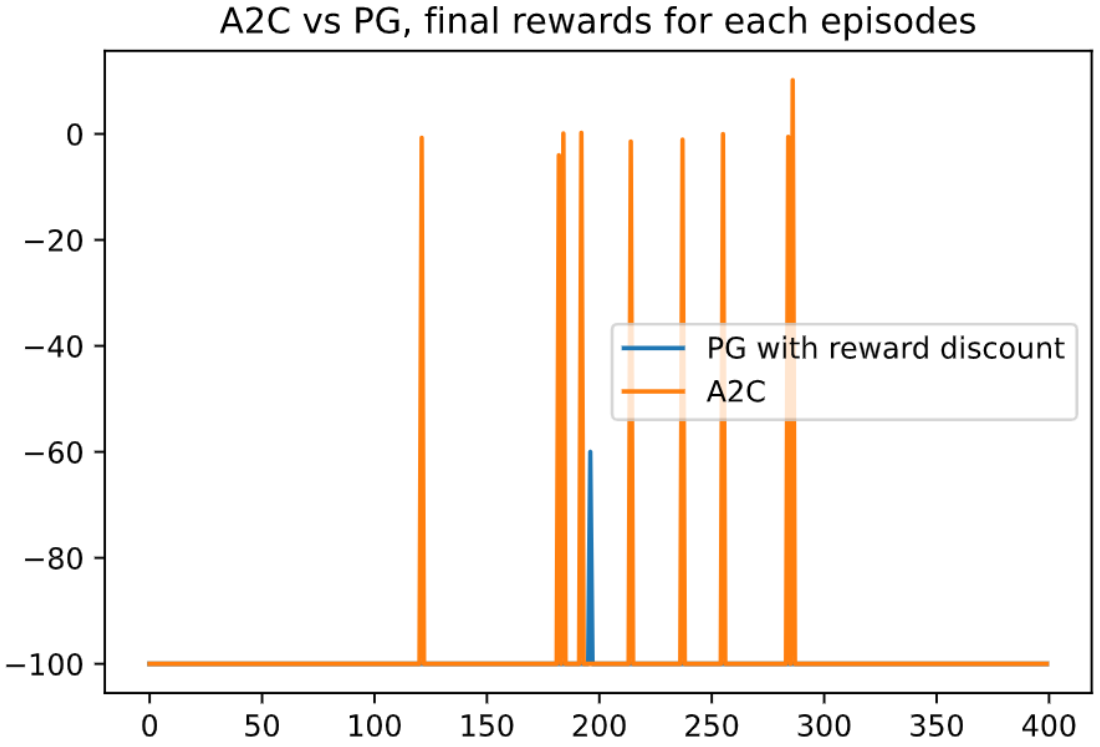


右圖為左圖做 MA = 100 後的圖，二者皆為 avg_reward。
可以看到使用 A2C 可以讓 model 更加激進的去探索空間，從右方的 ma 圖可以看出 A2C 與 discount PG 其實有著差不多哦 mean，但是 A2C 的 variance 明顯高出許多，此處我猜是由於 critic 每一次所 propose 的 advantage 其實是不固定，且會

受到 actor model 訓練影響的，可以看到在 episode300 之後的 variance 相較之前的浮動開始緩慢縮減，故可以期待越之後的訓練其 variance 會越趨穩定，同時帶有相對 discount PG 更好的平均表現。



而此圖為實際嘗試 600 epochs 的結果，可以看到 A2C 的確有如我上述之隨著訓練 epoch 數量增加逐漸減小 vaiance，且其中不乏超越 PG 的 reward。
既然如此，那是否在 final reward 上 A2C 可以有著更亮眼的表現?



其答案是肯定的，透過上圖可以發現 A2C 有著相對 PG 更多突出的 reward，且幾乎都比 PG 高出一倍，故可以知道 A2C 雖然有著較為不穩定的訓練過程，但是他可以帶來的訓練潛力是高於基礎 PG 算法的，而這個不穩定我認為是因為同時有兩個 model 需要訓練，而且對象 env 是帶有浮動隨機性值的，這個問題透過增加訓練 epochs 數量應該可以得到良好的解決。