tags: ML

b07902040 資工二 吳承軒

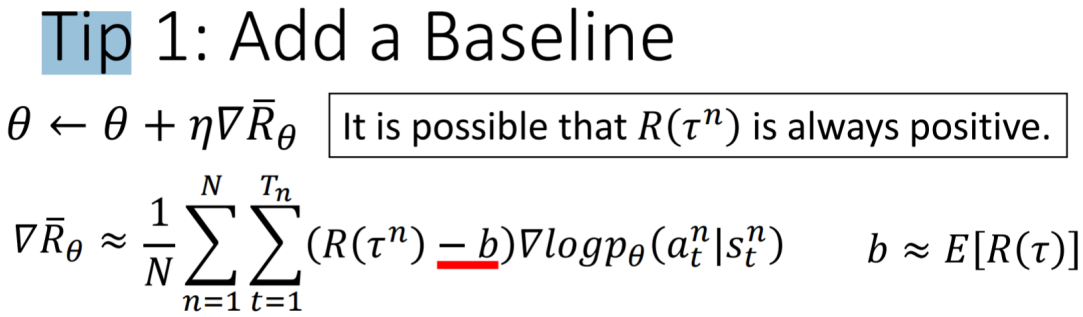
ML HW15 report

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。

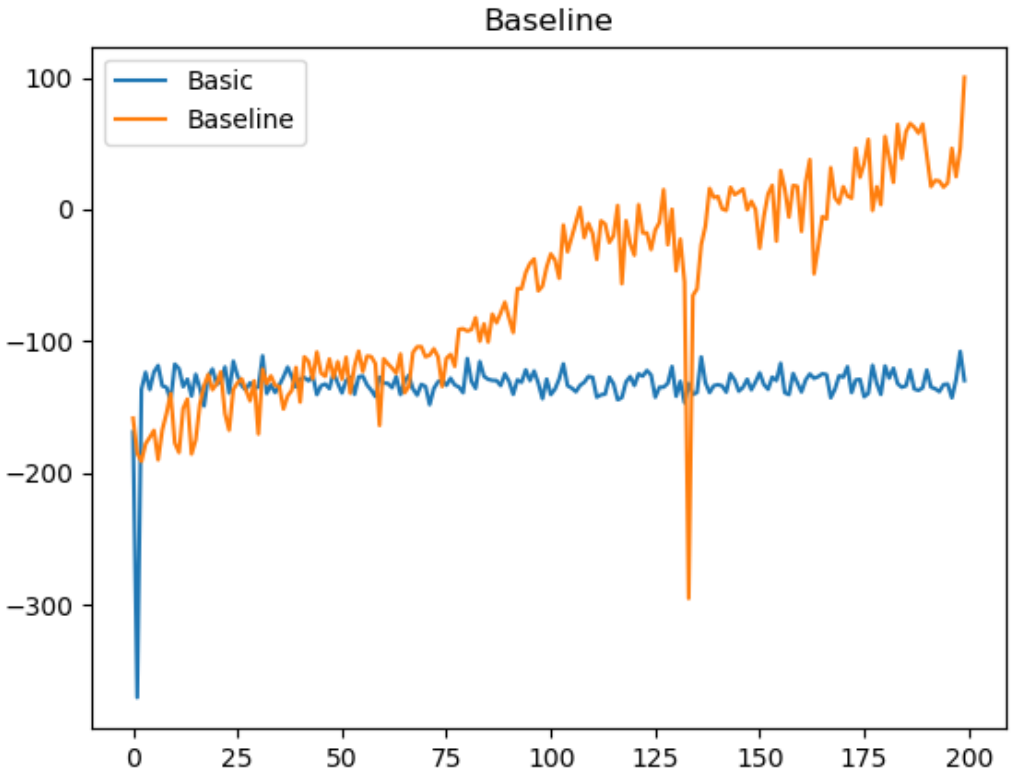
b.請說明你如何改進 reward 的算法，而不同的算法又如何影響訓練結果？

使用教授投影片中tip 1的作法



直接將rewards都減掉一個定值，設定條件為:

```
Learning rate=0.02
Episode_per_batch = 30
Network:
    self.fc1 = nn.Linear(8, 128)
    self.fc2 = nn.Linear(128, 128)
    self.fc3 = nn.Linear(128, 4)
```



可以發現應用baseline的reward雖然前期表現較差，波動也較大，但basic卡在-100多就上不去了，baseline最終表現要比basic好上不少。

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。

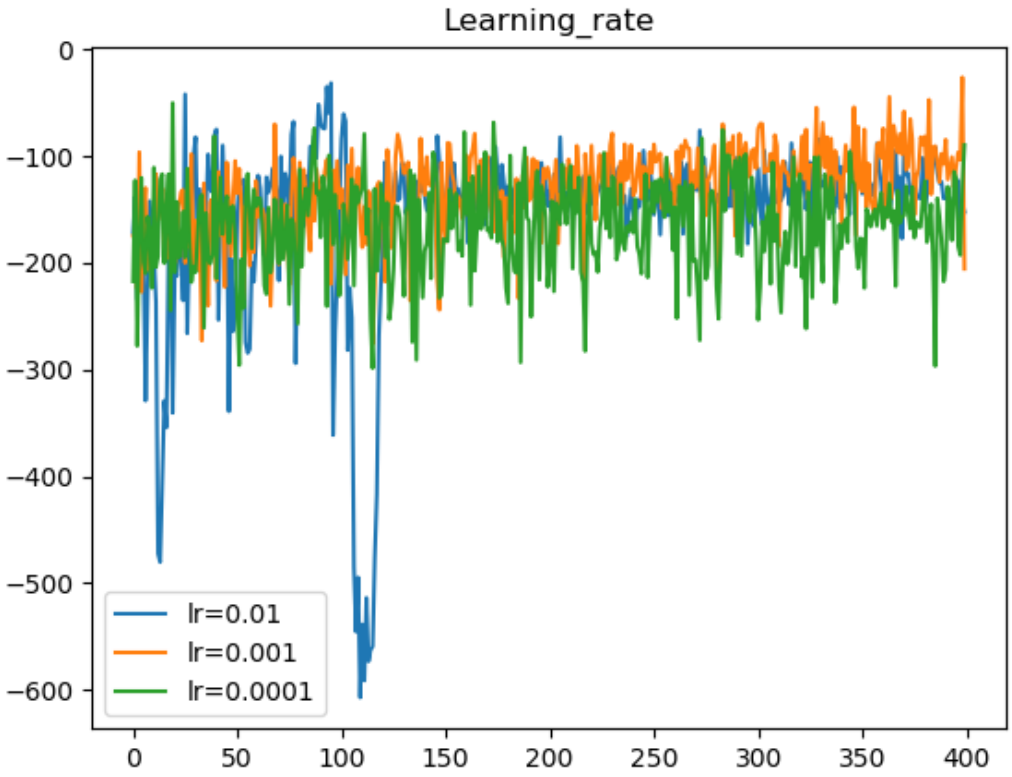
在未使用discount rewards的條件下，修改超參數: Learning rate、Episode_per_batch、神經網路，初始條件為:

```
Learning rate=0.001
Episode_per_batch = 5
Network:
    self.fc1 = nn.Linear(8, 16)
    self.fc2 = nn.Linear(16, 16)
    self.fc3 = nn.Linear(16, 4)
```

每種測試只更改其中一個超參數。

Learning rate

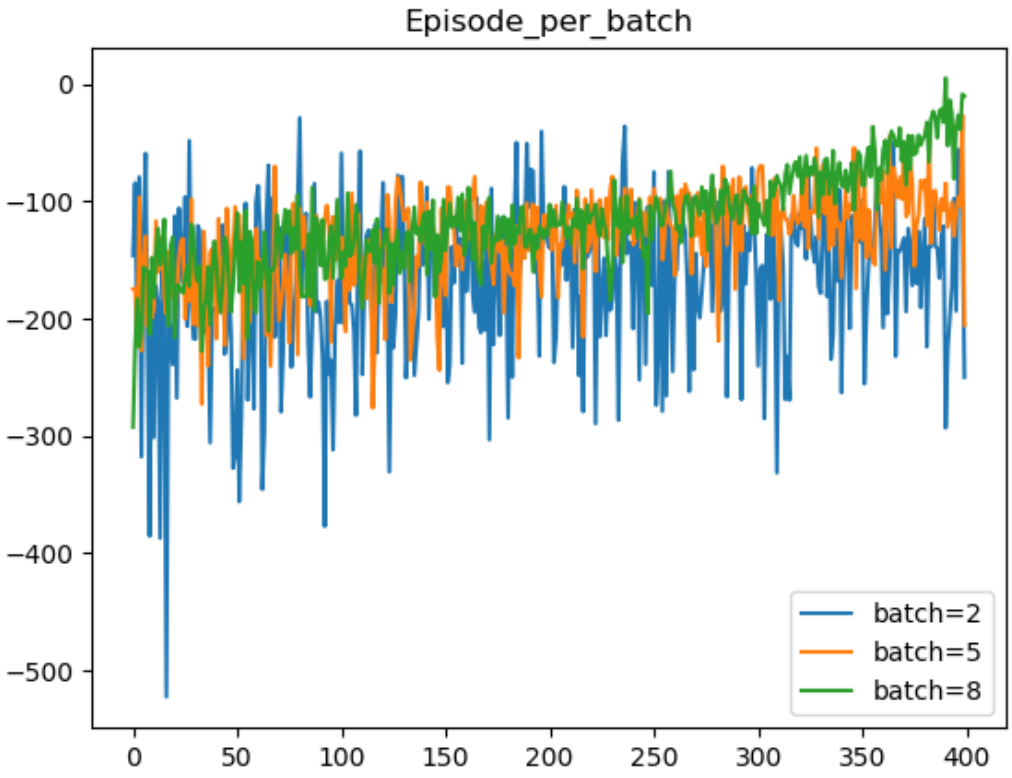
比較Learning rate為0.01, 0.001, 0.0001:



可以發現Learning rate對於結果的影響並不明顯，但在過程中，較大的lr=0.01明顯在前期波動較大。

Episode_per_batch

比較Episode_per_batch為2, 5, 8:

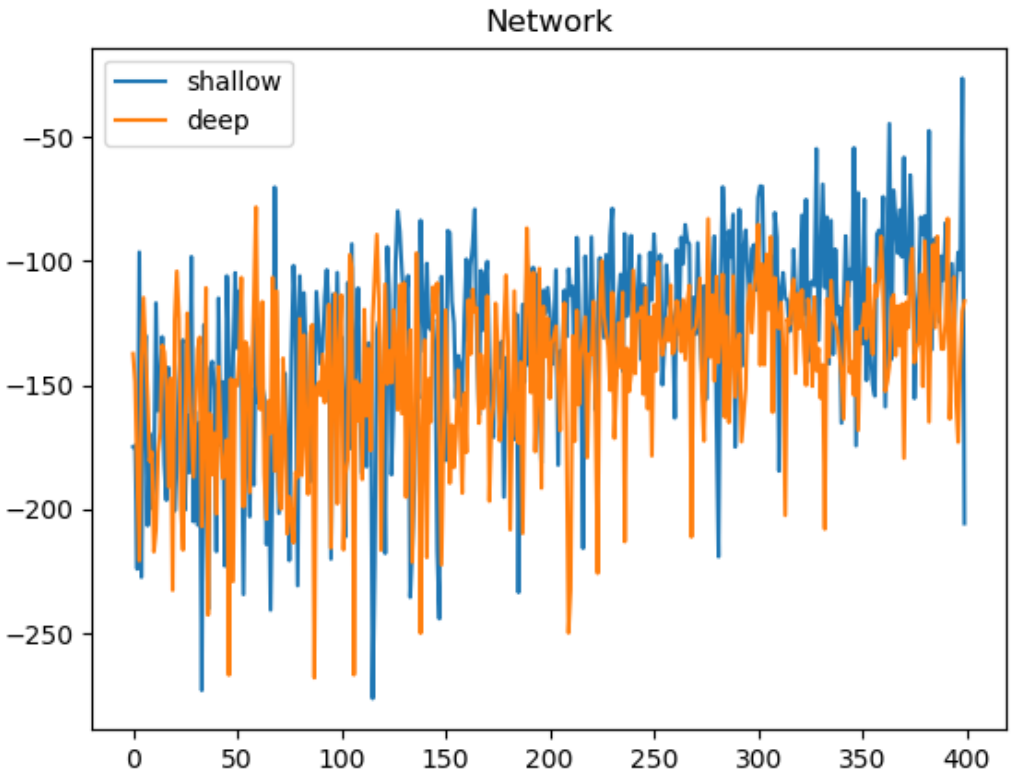


可以發現batch越小時，訓練的過程越不平穩，batch越大時越平穩，且在batch越大時，得到的結果越好。這是因為每次sample的episode越多，取到極端值的機率就越小，並且取平均後，得到的會更接近真正的機率分布；而sample越多次，所需時間也越久，所以train較久的結果較好是合理的。

神經網路

比較原本和較深的神經網路:

```
self.fc1 = nn.Linear(8, 16)
self.fc2 = nn.Linear(16, 16)
self.fc3 = nn.Linear(16, 32)
self.fc4 = nn.Linear(32, 32)
self.fc5 = nn.Linear(32, 4)
```



可以發現較深的神經網路並沒有得到更好的結果。

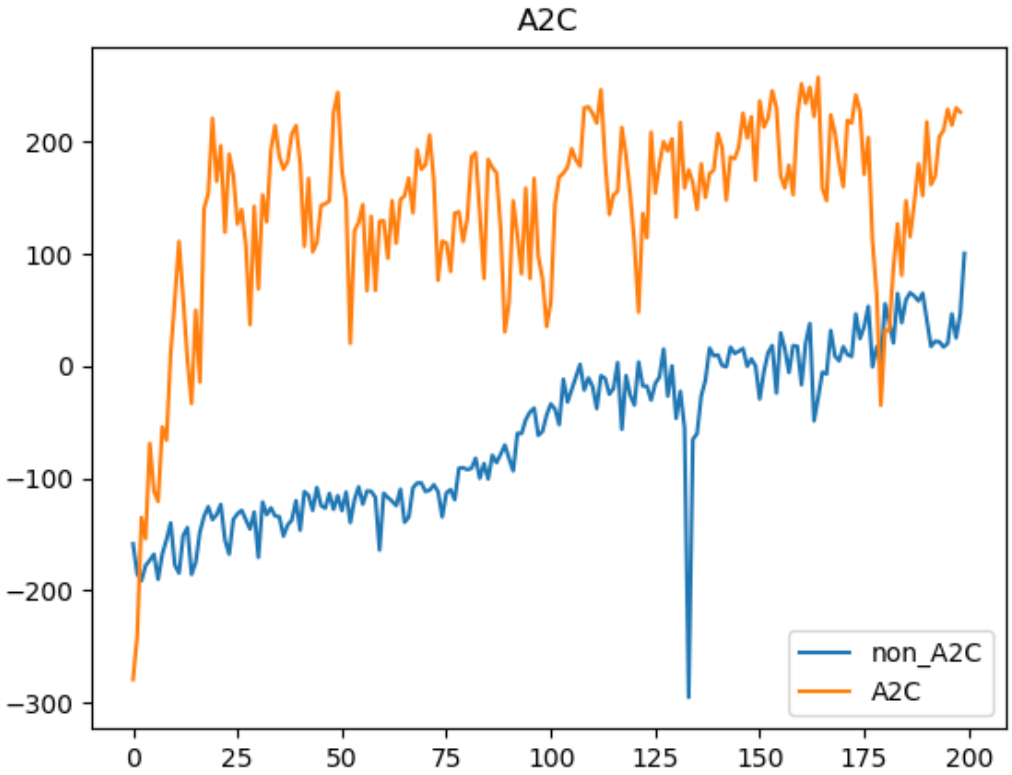
3.(20%) Actor-Critic 方法:

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

實作A2C(Advantage Actor-Critic)

b.請說明你的實做與前者（Policy Gradient）的差異。

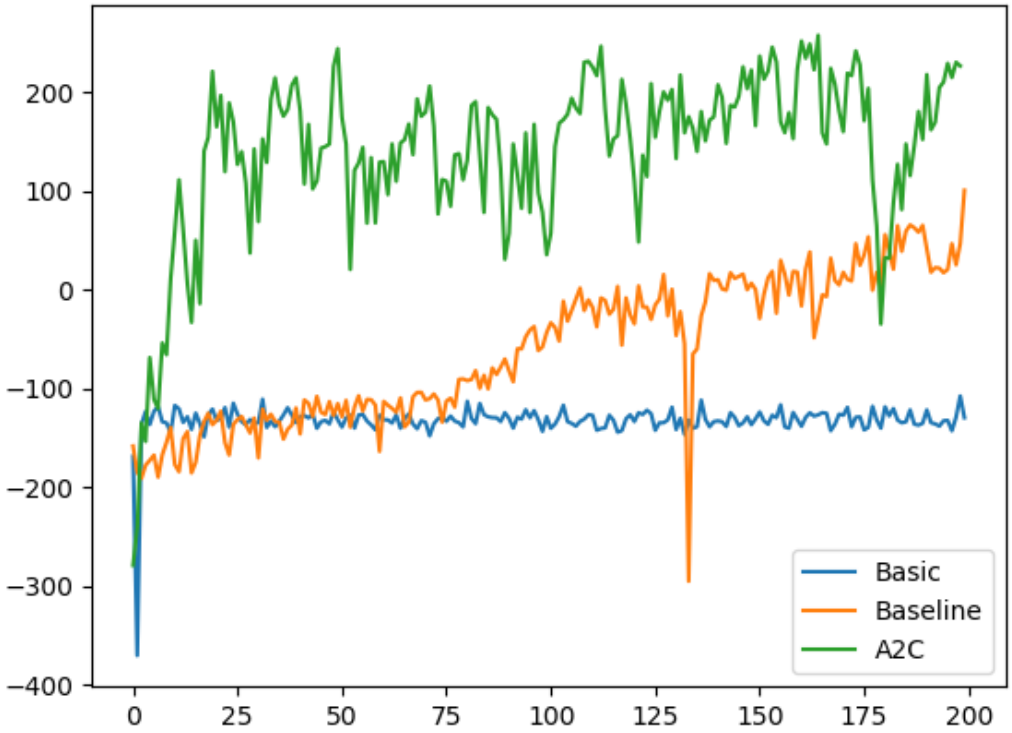
這邊用來比較的non_A2C有運用baseline，其餘設定都和第一題相同，結果如下:



可以發現使用A2C，在很短的時間內reward就達到了200，之後便一直震盪，整體平均約在170左右，較固定的baseline有長足的進步。

4. (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。

大部分的比較及作圖都放在上面了，這裡比較一下Basic(原始的policy gradient)、Baseline和A2C的表現:



可以發現A2C因為其baseline是由環境來訓練Critic所產生的，相較Baseline而言學習速度快上許多，且效果也較好。而Basic的則是在-100左右就train不上去了，說明了添加baseline來避免sample和更新誤差的重要性。

reference

https://github.com/nikhilbarhate99/Actor-Critic-PyTorch