ADLxMLDS hw3 報告

b03902083 陳力

Basic Performance

PG model

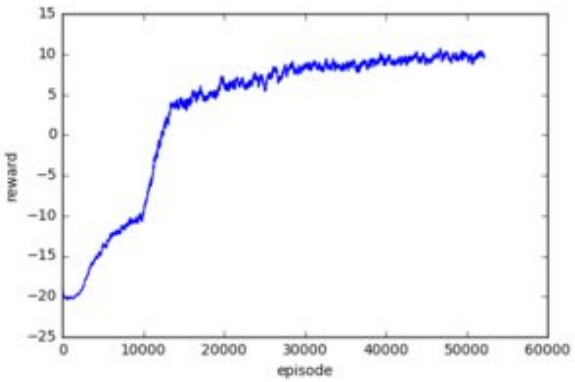
我的 policy network 輸入是處理成灰階的畫面變化 ($s_{t+1} - s_t$, 大小 80 x 80)，經過一層 dense 與 ReLU 後成為 256 個 hidden unit，再過一層 dense 與 Softmax 變成 6 種操作的機率分佈。
使用 Adam 做優化，learning rate = 1e-3，每 10 個 episode 去做網路的更新。Discounted reward 去算 loss 前會先 normalize 成 mean=0, std=1。

DQN model

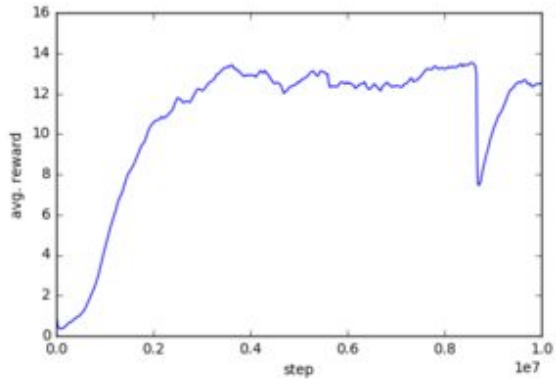
我的 DQN network 輸入是預設的 84 x 84 x 4 的 frame stack，然後就跟 TA 提供的 model 一樣，過 3 層 CNN (filter=32, 64, 64，kernel=8,4,3，stride=4,2,1，不做pooling，過 ReLU)，然後過一層 Dense (512 hidden unit)。
Replay memory 大小是 10000，每 4 個 step 取 32 個狀態為一個 batch 做 network update，前 10000 個 step 不做任何 update。Exploration rate 則是在 1000000 steps 內從 1 線性遞減至 0.05。而 target model 則是每 1000 個 steps 複製一次。

Figures

PG learning curve

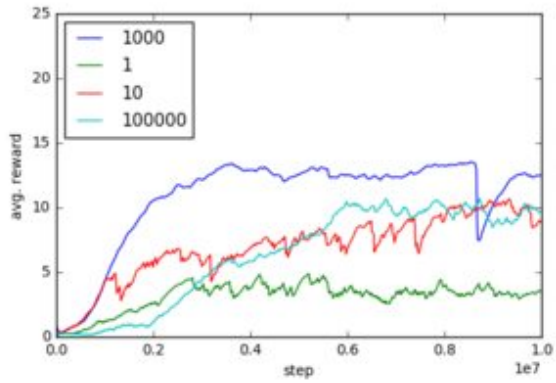


DQN learning curve



Experimenting with DQN

在這部分我比較了 target model 的更新頻率，我實驗了 4 種頻率分別是每 1, 10, 1000, 100000 個 step 更新一次（將 正在訓練的 model 複製成 target model）。
對這個參數有興趣的原因是這個頻率的大或小直觀來看都好像很有道理，更新頻率小對 Q 函數有比較準確的目標值，而更新頻率大則是降低訓練時的波動性。



實驗結果發現更新頻率過大過小都沒對效果有好的影響，

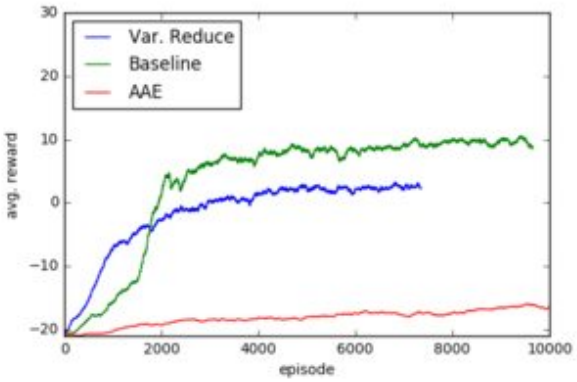
Bonus

PG bonus

PG 的加分部分我實作了 1. Variance reduction 2. Advanced Advantage Estimation。

Variance reduction 所做的事情是把每次 sample 出來的 trajectory 的 discounted reward sequence $r_1, r_2, \dots, r_T$ 減去一個 baseline b，目的在減少每次更新 network 的梯度的 variance。

Advanced Advantage Estimation (AAE) 則引入了 value function 與 advantage 的概念，每次更新的梯度變成 $\nabla_\theta \sum_t (\log \pi_\theta(a_t|s_t) \times (R(t) - V(s_t)) + (R(t) - V(s_t))^2)$，其中 $R(t)$ 是第 t step 往後的 discounted reward、$V(s_t)$ 是對狀態 s_t 的 value 估計。

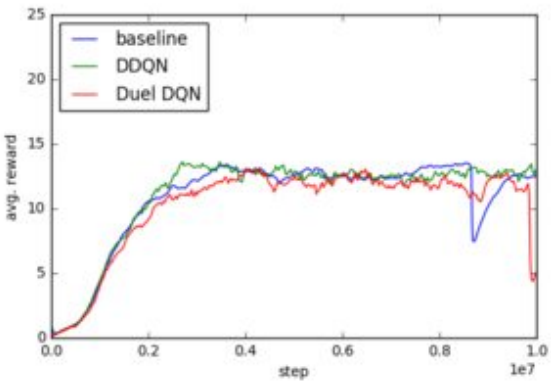


實驗顯示，使用 AAE 非常難訓練，而使用 variance reduction 雖然一開始訓練得比較好，但收斂的分數卻比不用還低一大截。

DQN bonus

DQN 的加分部分我實作了 1. Double DQN 2. Duel DQN。

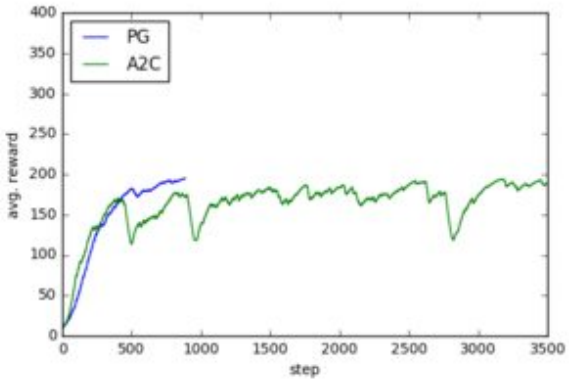
Double DQN 把訓練 $Q(s_{cur}, a_{cur}; \theta)$ 的目標值修改成為
$r + \gamma \times Q(s_{next}, \arg\max_{a'} Q(s_{next}, a'; \theta); \theta^{-})$，希望藉此修改 $Q(,)$ 值高估的問題。
Duel DQN 則是引入了 value function 與 advantage 的概念，重新定義
$Q(s, a; \theta) = V(s; \theta) + A(s, a; \theta) - \frac{1}{|a|} \sum_{a'} A(s, a'; \theta)$。
以下是實驗結果：



實驗顯示此二法相較原始 DQN 並無明顯優勢。

Other RL methods

第 3 個加分項目我實作了 A2C 方法在 CartPole 這個遊戲上。A2C 與傳統的 Policy gradient 差別在於 1. 引入 state value function，以之為基準計算一個 action 的 advantage，並用它去調整 $\pi_\theta(a|s)$；2.有著更小的更新週期，大概每 20 個 step 就會做一次 network 的更新。在先人們的實驗裡，這個方法在 atari game 上有很好的進步。以下是訓練過程中每個 episode 得到的 reward 圖。



由實驗可知在 cartpole 上 A2C 比起傳統的 policy gradient，訓練時的不穩定性多得多，雖然一開始學得比較快，但後來就時而吃鱉。

Reference

Deep Reinforcement Learning with Double Q-learning
(https://arxiv.org/abs/1509.06461)

Asynchronous Methods for Deep Reinforcement Learning (https://arxiv.org/pdf/1602.01783.pdf)

Dueling Network Architectures for Deep Reinforcement Learning
(https://arxiv.org/pdf/1511.06581.pdf)