ADL hw5

B02902034 資工四 邱筱晴

1. Explain why DQN algorithm need these functions and how you implement them:
(1) Experience replay
建一個出現過的歷史事件 dataset。隨機從這個 pool 裡取以前出現過的 sample pair(oldstate, action, reward, state, terminal?) 來 update loss function。比起一次只看一個 observation，取歷史樣本一起 train 可以避免 data correlation 讓 model 容易被一連串出現的 state 影響，使得 training 結果 converge 得比較快。
Implement:

```
# Record experience in replay memory and train
if isTraining and oldState is not None:
    clippedReward = min(1, max(-1, reward))
    replayMemory.addSample(replay.Sample(oldState, action, clippedReward, state, isTerminal))

    if environment.getStepNumber() > args.observation_steps and environment.getEpisodeStepNumber() % 4 == 0:
        batch = replayMemory.drawBatch(32)
        dqn.train(batch, environment.getStepNumber())
```

pool size 預設為 1000000。每個 iteration，將 sample pair 加入 replayMemory。每四步從 dataset 裡 random sample 32 個 experience 來 train dqn.

(2) Target network
固定 Q network 的參數，避免因為同時調整 estimator 跟 Q network 導致 training 的方向跳來跳去，難以收斂。
Implement:

```
# build the variable copy ops
self.update_target = []
trainable_variables = tf.trainable_variables()
all_variables = tf.all_variables()
for i in range(0, len(trainable_variables)):
                self.update_target.append(all_variables[len(trainable_variables) + i].assign(trainable_variables[i]))
```

預設每 10000 步 update 一次 Q network。 呼叫 self.sess.run(self.update_target) 來用現在 Q network 裡的值取代 target network 裡的 variables。

(3) epsilon greedy
用一個微小的機率去嘗試選擇一些不同的 action，而非當前估算最高分的 action。如此一來可以 explore 其他可能也不錯的 action。
Implement:
epsilon = max(.1, 1.0 - 0.9 * environment.getStepNumber() / 1e6)

(4) clip reward
設定 reward 在一個區間內避免 reward 的值太大或太小。
Implement:
clippedReward = min(1, max(-1, reward))

2. If a game can perform two actions at the same time, how will you solve this problem using DQN algorithm? (there's no absolute answer)
把不同 action 的組合看作一個新的 action，產生新的 (C n 取 2) 個 actions。用新的 action set 去 train。