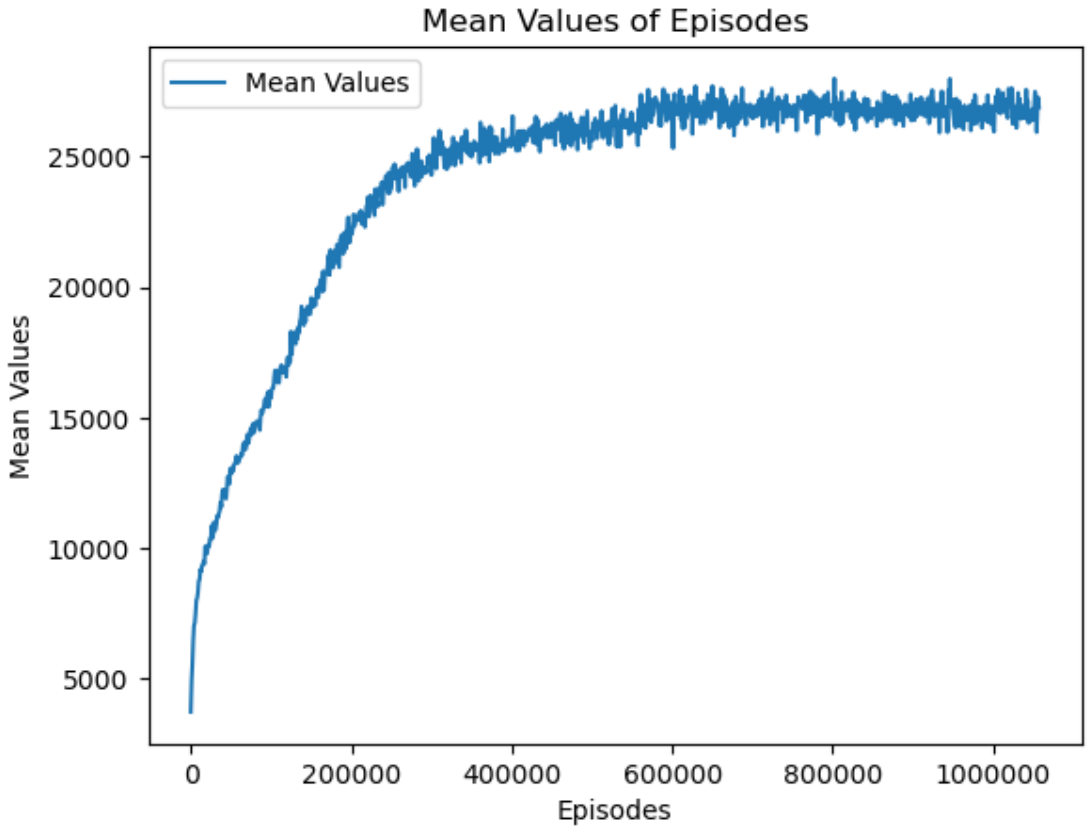
# Lab1: Temporal Difference Learning

學生 ： 陳澤昕
學號 ： 311356003

Report :

Plot shows scores (mean) with 1000K




Demo :

```
[Running] cd "/home/tzeshinchen/data/RF_learning/HW1/" && g++ 2048_demo.cpp -o 2048_demo && "/home/tzeshinchen/data/RF_learning/HW1/"2048_demo
TDL2048-Demo
alpha = 0
lambda = 0.5
total = 1000
seed = 3321882045
6-tuple pattern 012345, size = 16777216 (64MB)
6-tuple pattern 456789, size = 16777216 (64MB)
6-tuple pattern 012456, size = 16777216 (64MB)
6-tuple pattern 45689a, size = 16777216 (64MB)
6-tuple pattern 012345 is loaded from /home/tzeshinchen/data/RF_learning/HW1/output/2048_lamba_to_0/800000_weight.bin
6-tuple pattern 456789 is loaded from /home/tzeshinchen/data/RF_learning/HW1/output/2048_lamba_to_0/800000_weight.bin
6-tuple pattern 012456 is loaded from /home/tzeshinchen/data/RF_learning/HW1/output/2048_lamba_to_0/800000_weight.bin
6-tuple pattern 45689a is loaded from /home/tzeshinchen/data/RF_learning/HW1/output/2048_lamba_to_0/800000_weight.bin
1000    mean = 25221.9  max = 54564
    64  100%    (0.1%)
    128 99.9%   (0.2%)
    256 99.7%   (1.1%)
    512 98.6%   (2.9%)
    1024    95.7%   (16.5%)
    2048    79.2%   (79%)
    4096    0.2%    (0.2%)
```

Bonus :

a. Describe the implementation and the usage of $n$-tuple network. (5%)

雖然將每一個盤面都存儲並賦予估計值是理想的方法，但在實際操作中，記憶體限制（約為 $12^{16}$）無法負擔，因此產生了 N-tuple 網絡的解決方法。

N-tuple network 的方法是選擇一部分盤面作為"特徵"，並在估計值計算中僅針對這一小部分特徵進行操作。這種方法稱為 N-tuple 網絡，例如 6-tuple 網絡表示將棋盤分為 6 個特徵區域，並在棋盤上的每個特徵區域進行估值更新。此外，當存在 10 個這樣的特徵區域時，應考慮四個旋轉方向 x 兩種鏡像反射，總共八種 isomorphism。因此，必須對每一種 isomorphism 都進行一次特徵區域的選擇和估計值計算。最終，將這八個估計值總和，形成了該盤面的最終估計值，進一步去決定下一步的行動。

調整$n$-tuple network 時，需要將 feature_error = error / feature_size 再來 isomo_error = feature_error/isomorphism_size，isomo_error * error 才是一個 isomorphism 盤面需要更新的值。

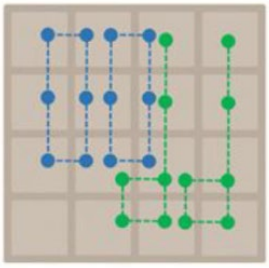
b. Explain the mechanism of TD(0). (5%)

"TD" 是強化學習（Reinforcement Learning）中的一個方法，通常用於值函數估計和預測任務。

公式：$V(s) \leftarrow V(s) + \alpha(Rt+1 + \lambda V(s') - V(s))$

在這個公式中，"TD target" 被定義為 $Rt+1 + \lambda V(s')$。其中，$Rt+1$ 代表執行該動作後獲得的分數，$V(s')$ 代表執行該動作後的狀態估計值。將 TD target 減去原始的狀態估計值，即 TD target - V(s)，就得到了誤差（error）。透過乘以一個學習速率（learning rate）的方法，來控制每次更新當前狀態的估計值的幅度。最後，將這個誤差加回到當前狀態的原始估計值 V(s) 上，完成一次更新過程。

c. Describe your implementation in detail including action selection and TD-backup diagram. (10%)



- Learning Rate（學習率）：一開始設定為 0.1，當均值（mean）有五次來回震盪時，將學習率調整為 mean*0.8，直到下降至 0.0026 不再改變。
- $n$-tuple Network（$n$元組網絡）：使用了 8 個 6-tuple 網絡，特徵如左圖所示。
- Random Rate（隨機率）：一開始設定為 0.1，會隨著訓練次數逐漸下降。這個隨機率決定了模型在初始階段有 10%的機會進行隨機探索，然後隨著訓練次數增加，這個隨機率逐漸下降至 0。

- Action Selection（Action Selection）：由於本次更新是針對 V(state)，所以在選擇行動時需要考慮下一個方塊生成的位置。因此，算法首先測試當前盤面的上、下、左、右四種行動，並估計下一個狀態中方塊出現的位置以及其機率（2: 0.9，4: 0.1）。然後，將方塊添加到當前盤面，計算估計當前盤面可能獲得的分數，再乘以機率。將所有行動的估計值相加，最終選擇具有最大估計值的行動作為下一個行動。
- TD-Backup Diagram（TD 後向備份圖）**：一開始，所有盤面的初始值設定為 0，TD 目標（TD target）設定為 0，因為遊戲結束。因此，最後一個盤面的學習目標為 0。計算誤差（error）時，將 TD 目標減去估計值（estimate(V(st)）），然後更新這個盤面的估計值。接下來，計算前一步的 TD 目標，即 TD target_pre = pre_reward + update($V(s_t)$，error * alpha)，然後向前更新 $V(s_{t-1})$。重複這個過程，直到遊戲更新到初始值。