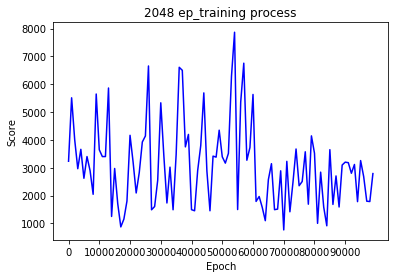
**Lab7 - 2048 TD Learning**

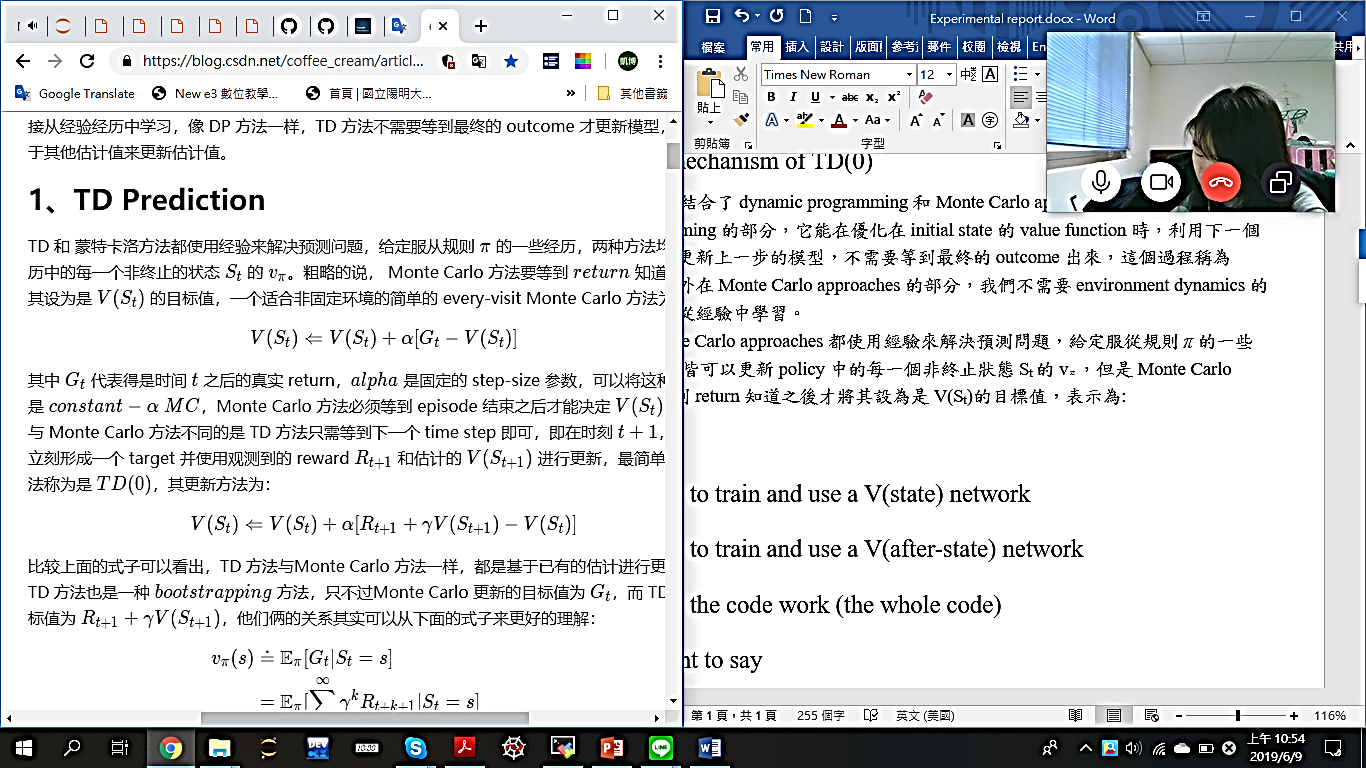
1. A plot shows episode scores of at least 100,000 training episodes



1. Explain the mechanism of TD(0)

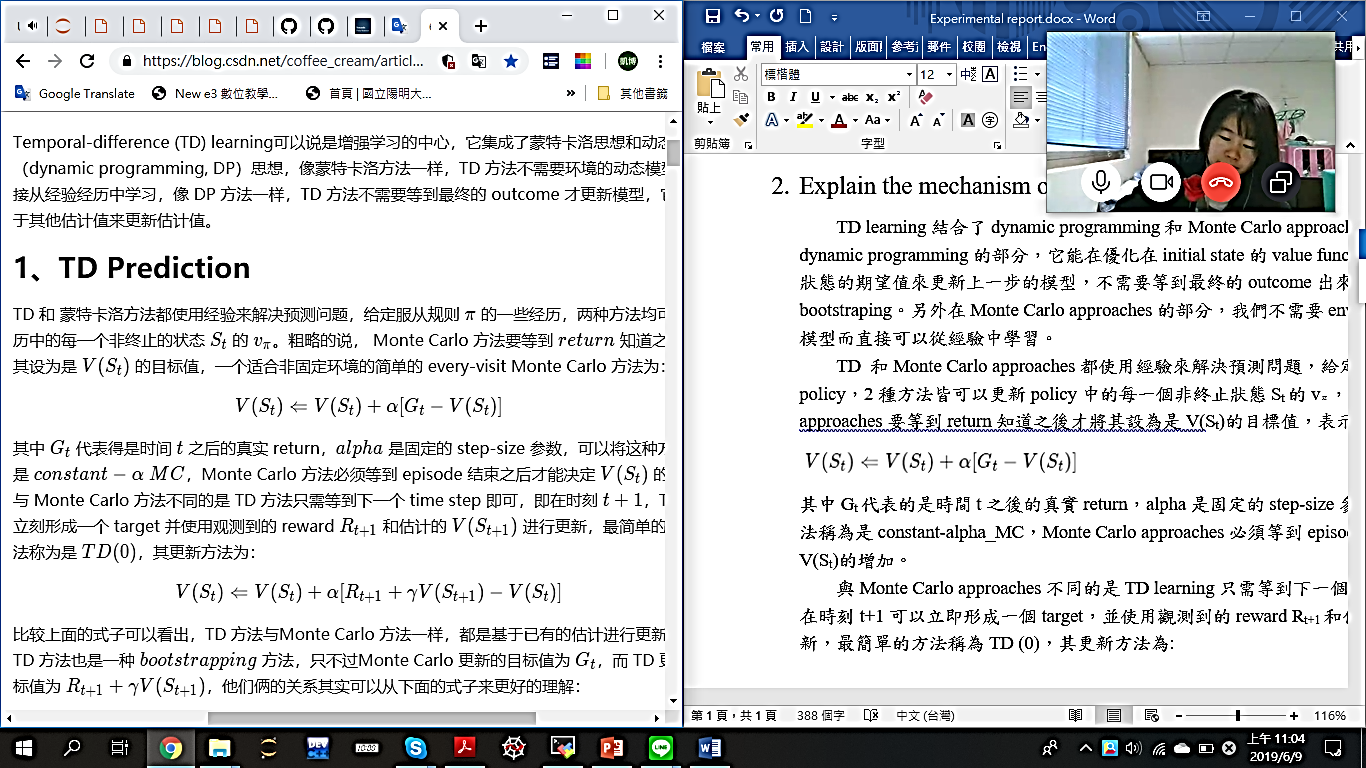
TD learning結合了dynamic programming和Monte Carlo approaches的優點，首先是dynamic programming的部分，它能在優化在initial state的value function時，利用下一個狀態的期望值來更新上一步的模型，不需要等到最終的outcome出來，這個過程稱為bootstraping。另外在Monte Carlo approaches的部分，我們不需要environment dynamics的模型而直接可以從經驗中學習。

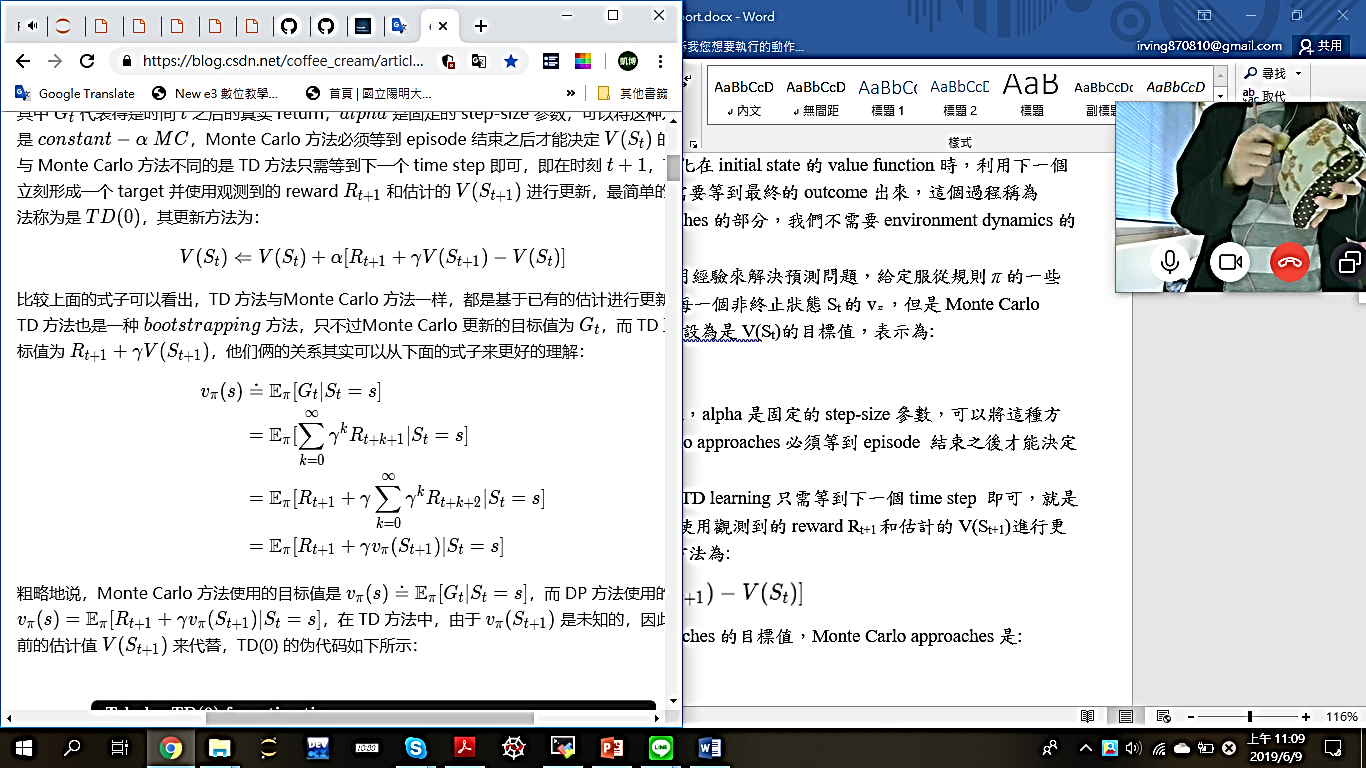
TD 和Monte Carlo approaches都使用經驗來解決預測問題，給定服從規則π的一些policy，2種方法皆可以更新policy中的每一個非終止狀態St的vπ，但是Monte Carlo approaches要等到return知道之後才將其設為是V(St)的目標值，表示為:



其中Gt代表的是時間t之後的真實return，alpha是固定的step-size參數，可以將這種方法稱為是constant-alpha\_MC，Monte Carlo approaches必須等到episode 結束之後才能決定V(St)的增加。

與Monte Carlo approaches不同的是TD learning只需等到下一個time step 即可，就是在時刻t+1可以立即形成一個target，並使用觀測到的reward Rt+1和估計的V(St+1)進行更新，最簡單的方法稱為TD (0)，其更新方法為:



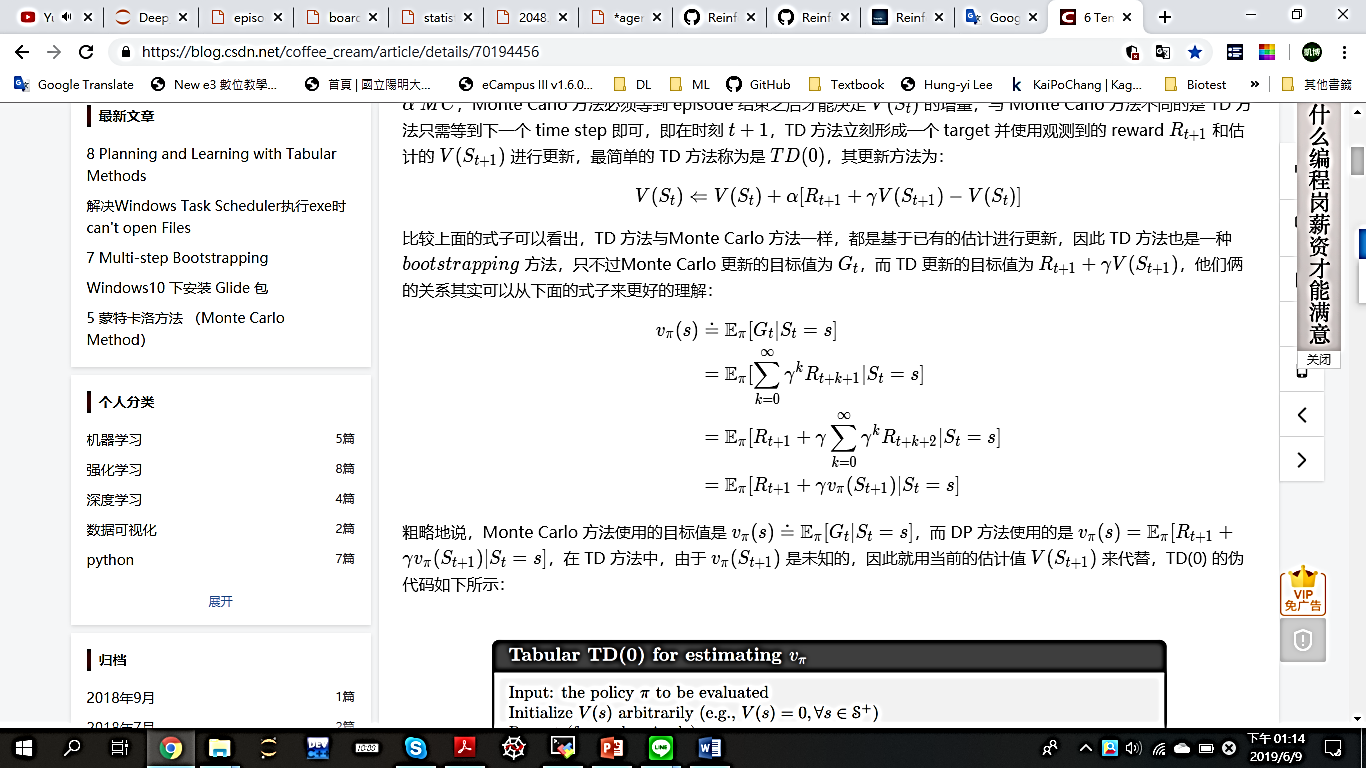
若比較Monte Carlo approaches、dynamic programming、TD learning的目標值，

Monte Carlo approaches:

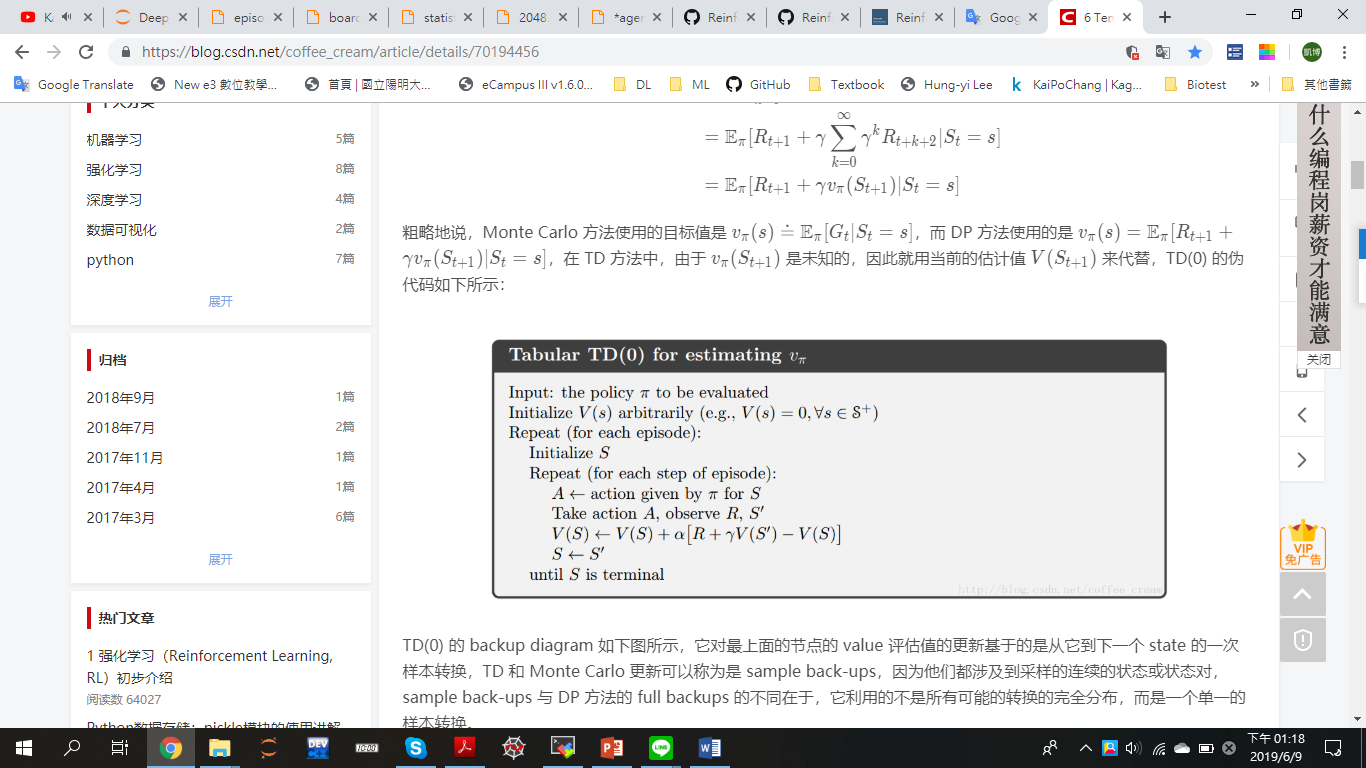
​Dynamic programming:

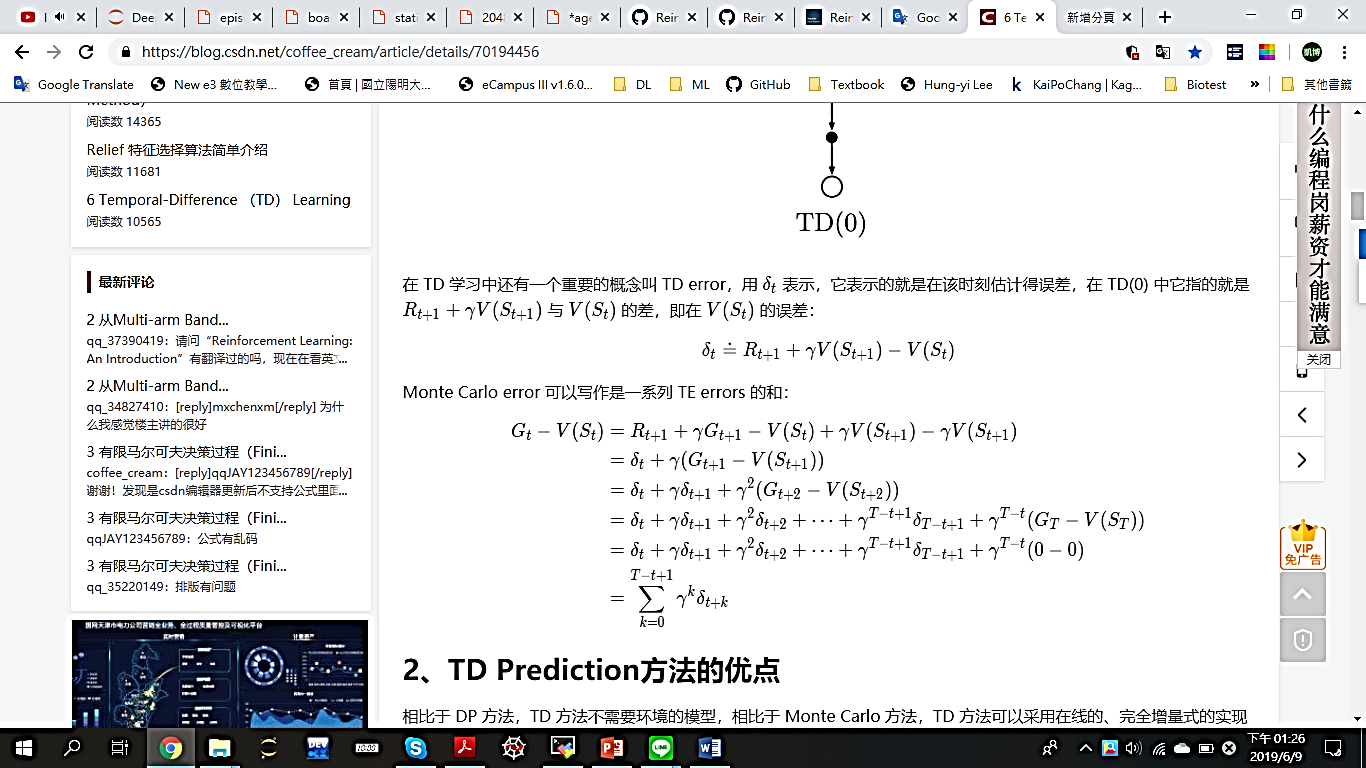
TD learning因為未知，因此利用當前的估計值來代替，

其中Monte Carlo approaches和TD (或DP)更新的更新可由下圖來看:

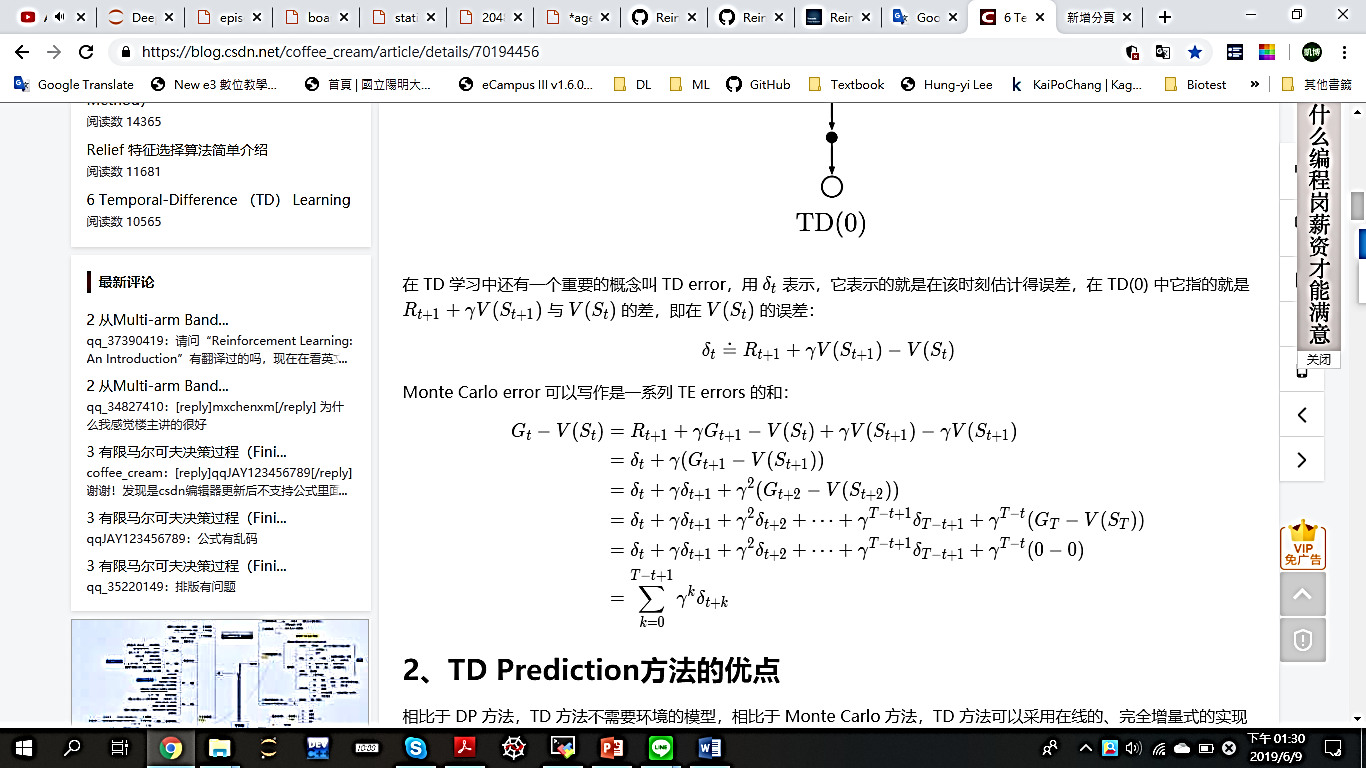






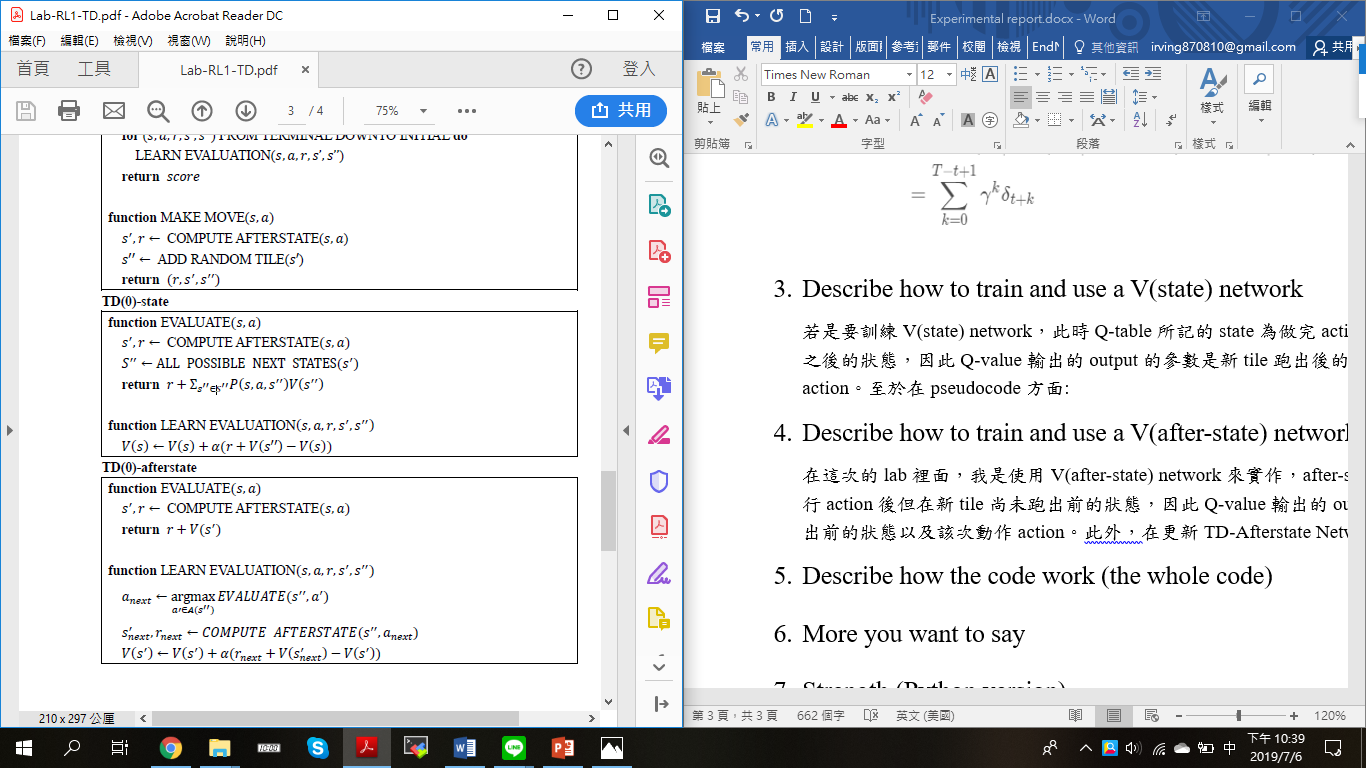
在TD學習中還有一個重要的概念叫 TD error，以δt表示，表示的就是在該時刻的估計誤差，就是在V(St)的誤差:

另外，Monte Carlo error 可以寫作是一系列 TE errors 的和：



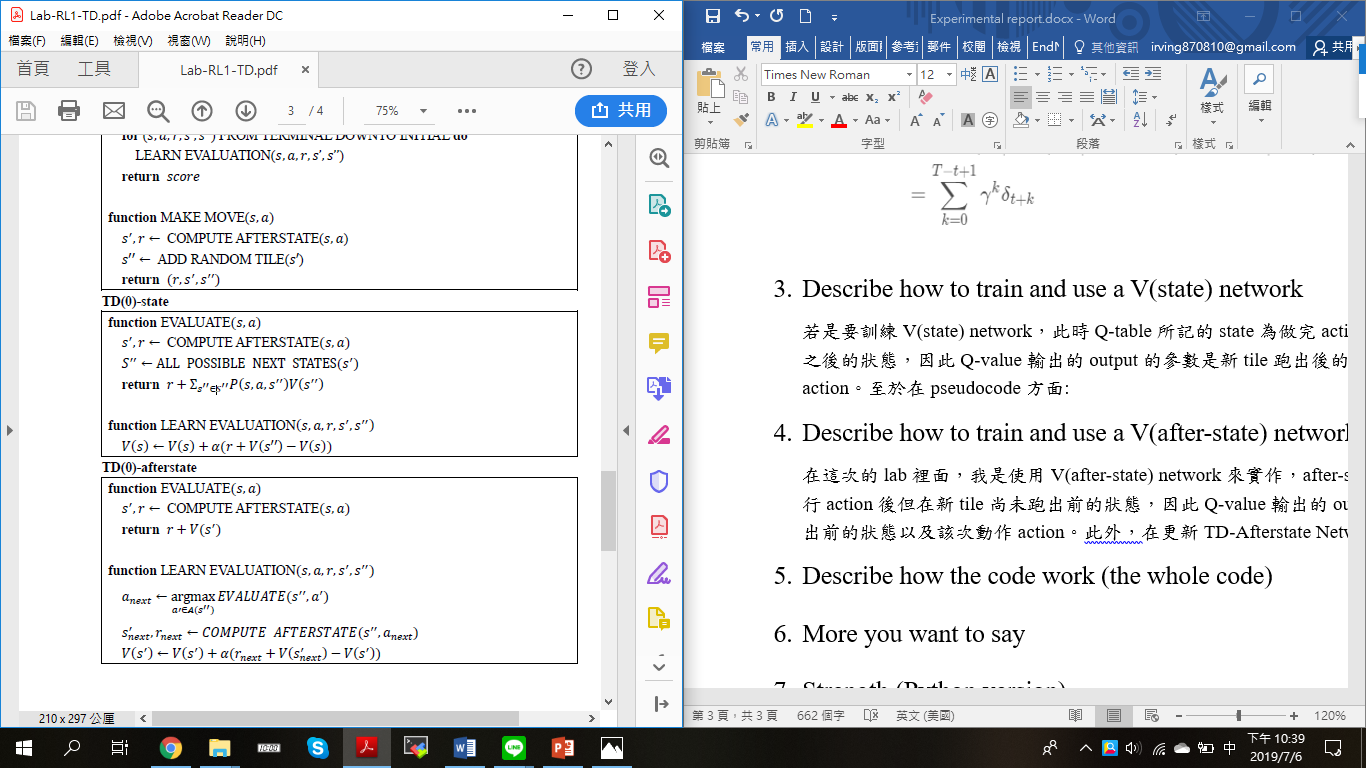
1. Describe how to train and use a V(state) network

若是要訓練V(state) network，此時Q-table所記的state為做完action且新tile 隨機跑出來之後的狀態，因此Q-value輸出的output的參數是新tile跑出後的狀態以及該次動作action。至於在pseudocode方面:



V(state) network一開始要先計算出after-state及其reward，還要再藉由after-state去推算每一種有下一個可能的state的發生機率是多少。第一個evaluation function，除了要return下一個state的value function，因為中間隔一層不確定性的after-state，所以在下一個state的value function前還要在乘上該state的發生機率，最後更新Q值就是很正常的依照已經推倒過後的TD(0)公式去做更新。

1. Describe how to train and use a V(after-state) network

在這次的lab裡面，我是使用V(after-state) network來實作，after-state就是指board在執行action後但在新tile尚未跑出前的狀態，因此Q-value輸出的output的參數是新tile跑出前的狀態以及該次動作action。此外，在更新TD-Afterstate Network

V(after-state) network比V(state) network要簡單上許多，因為少了計算機率矩陣的關係，V(after-state) network的更新主要是先計算出after-state及其reward。第一個evaluation function，只要return下一個after-state的value function即可，最後更新Q值就是去用當前after-state和action下Q-table的最大值，去和前一次after-state的value function以及reward去做在之前已經推倒過後的TD(0)公式去做更新。

1. Describe how the code work (the whole code)

在這整個程式碼中，主要分成兩部分，第一個部分是建構環境，第二個是做運算，建構環境的部分不用gym函式庫的原因是之前助教說可能會有運算太久的問題，所以提供給我們他們實驗室自己手刻的2048環境，在這個lab主要是以Q-Learning去做學習，但是我發現我並沒有做得很好，並且因為Q-table是單純地將每個狀態都記住，不像DQN有generalization的特性，所以計算到一半也發生memory爆炸的問題，所以我最後就改用agent只往reward較大的這個方式去執行action，步驟就是先有一個處於initial state的board，之後在每個board的after-state狀態之下，計算出agent往哪個方向移動，會有較大的reward，就往該方向執行action，最後terminate的時候是當無論哪個方向的reward皆為-1，也就是死盤的時候，就可以重新reset一個全新的board了。

1. More you want to say

在這次的lab中，TD learning的理論相對於之前的深度學習網絡其實並不難理解，但是困難的點是在於reinforcement在網路上的資源幾乎都是搭配Neuron Network去做運用，此外，網路上在玩小遊戲的reinforcement learning algorithm主要都是搭配gym函式庫，而這次2048主要卻是以手刻為主，以上是我覺得這次的lab稍難的原因。

1. Strength (Python version)

