AI Midterm Project Report Group 09

1. Introduce your model, what are advantages of your model.

我們藉由改良TA的normal-ai建立rule-based model，主要藉由找出normal-ai的弱點並給予防守與進攻、改良weight來增加勝率。

基本架構：

以定義weight的高低來決定下一步棋要往哪走，沿用normal-ai基本架構。

1. 首先，設定當我方為黑色，則敵方為白色，反之亦然。
2. 接著，設定weight使我方的勝率高於敵方。
3. 最後，確定下的位置不超出棋盤範圍，根據weight排序，決定下一步位置。

策略優勢：

在比較過原始的MCTS程式後，由於MCTS模擬時間長，且對它掌握度較低，所以選擇以基礎的的model建構。優勢在於效率高，且我們可以針對每次輸掉比賽的缺點改善，然而改善度有限，在後續第三點改善方向會對此作詳述。

(1). 當我方附近有連續兩個敵方，空一格，一個敵方，使我方優先下棋，不讓對手有連三的機會

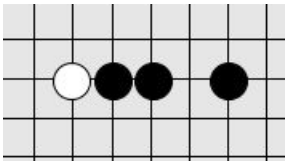

圖1.1「我方附近有連續兩個敵方，空一格，一個敵方」示意圖

(2). 當我方附近有連續三個敵方，不讓對手有連四的機會

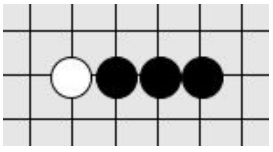

圖1.2「我方附近有連續三個敵方」示意圖

(3). 當我方有連三，加重weight，使我方有連四的機會

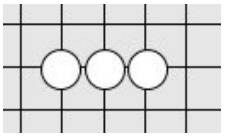

圖1.3「我方有連三」示意圖

(4). 因在測試時發現加入了上述優勢會使得系統遺忘我方連四，因此當我方有連四，加重weight，使我方優先贏得勝利

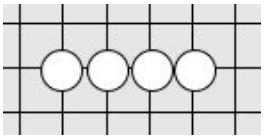

圖1.4「我方有連四」示意圖

2. Analysis why you are defeated, what is the weakness of your model.

我們將模型弱點，分兩部分分析：

(1). 與AI-easy,normal PK

首先在起初設計時，我們是針對每次打輸的問題進行檢討，並去計算權重，探討在rule-based的情況下，如何讓AI下更有侵略性的棋步或防守應該防的防守（ex:製造活三、對方活三優先擋棋）。我們的Solution是，判斷棋盤上的每一個子棋的周圍，給予不同的weight，尤其在於發現幾次打輸的原因分別為：

a. 敵人活三(or活四)而不擋棋：給予最高weight，因為不防守即會落敗。

b. 我方活三(or活四)而不進攻：給予次高weight，如此才有機會獲勝。

基於以上調整，我們成功擊敗easy-ai及normal-ai的程式。

(2). 在round-robin分組循環 PK

然而，這樣AI程式在循環小組PK後，便可以發現諸多問題。起初在初步嘗試MCTS後，卻決定改以rule-based的原因是：

優點：每一步棋的執行效率高、且對於AI的掌握度較高能針對問題進行改善；

缺點：在這樣防守為重的策略下，便可能使總執行時間拉長，且在判斷深度不足的情況，將可能造成敵人雙活三而來不及挽回的局面。

3. How to improve it

透過檢討循環分組PK結果後，我們將改善方法分述如下：

(1). 從**Rule-based角度**改善：

以這方面的角度去想，其實可以發現幾次的落敗都是因為我們在策略調整上太過單調，但真正我們人類思考五子棋的攻防策略往往*不應侷限於「只進攻」或「只防守」*，且在空間上也必須考慮周圍子棋可攻擊的空間，這應是為何此門課程強調利用蒙地卡羅的深度搜尋，才能真正將下一步棋的每一步依依模擬找出有限搜尋時間內的最佳解。

(2). 從**MCTS角度**改善：

MCTS的模擬時間長，且較難掌握。

- **模擬時間問題：**

其實在查詢了AlphaGo的algorithm介紹後，他們有一個重要的技巧在於具有兩個policy network，一個快一個慢。

在TA們在程式中加入了time-out的機制後，我們可以藉由判斷執行時間的機制（ex:規定在6 sec內完成一步棋，可以將4 sec時作為下棋方法的分界），一旦在深度搜索下無法取得較高分的棋路且超過time-out的時間範圍外時，便轉換rule-based機制，避免超時。如此一來，將可以解決模擬時間過長的問題。

- **掌握度問題：**

另外在AlphaGo algorithm上，估計棋盤局勢的走向的關鍵在於value network的配置。原始我們參考的MCTS example code時，並未掌握到active-value function的精髓，也致使在棋路上無法如預期掌握。

下圖3.1，為以不同visit count走訪次數(Variable 'N')的棋路模擬圖。

首先，可以先從最基本機率分佈運算，得到第一步(a)較高的獲勝可能(紅色箭頭a3, Variable 'W')；再來，藉著提高走訪次數(N)去搜尋，計算所有不同Node子點中的獲勝可能機率。

以圖(c)為例，當搜尋N=3時，獲勝可能Q=W/N，可以比起在圖(a)中N=1時，更具參考的精確性。

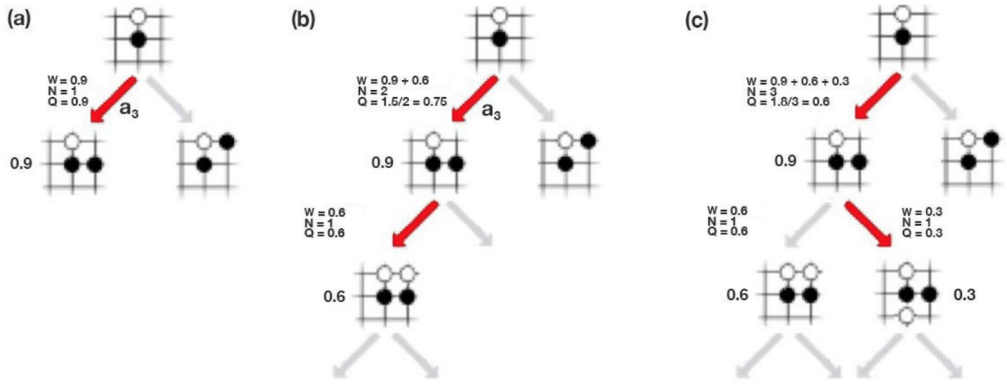

圖3.1 以不同visit count走訪次數(Variable 'N')的棋路模擬圖

因此，透過上述導入value network的觀念，並在最基本的機率分佈運算上以rule based實現，如此應可以提高部分層面上對MCTS的掌握度。

4. What is the work division of each team member

工作內容	工作分配
發想程式演算法	全體組員
撰寫程式	
Debug	
撰寫報告	

5. Reference data

- 蒙地卡羅的coding實作

https://www.cnblogs.com/xmwd/p/python_game_based_on_MCTS_and_UCT_RAVE.html

- Rule-based的coding實作

https://github.com/exeex/gomoku-ai-framework/blob/master/ai/normal_ai.py

- MCTS的觀念

https://medium.com/@quasimik/monte-carlo-tree-search-applied-to-letterpress-34f41c86e238

- MCTS問題改善方法

https://medium.com/@jonathan_hui/monte-carlo-tree-search-mcts-in-alphago-zero-8a403588276a

- AlphaGo algorithm

http://startupbeat.hkej.com/?p=45505

6. Any feedback

此次Project進行方式有三點讓我們受益良多，第一為以「小組形式進行」，讓我們可以藉由多次討論、不斷交換想法，而抵達單一個人無法觸及的深度；其二為「助教循序規劃Project方向」，從Project未進行前的線上程式、演算法教學，到Project由淺入深的題型設計，都使我們逐步累積信心，持續積極地完成它；最後為「分組競賽活動」，參考自己與他組的相對表現後，促使我們回頭審視程式原本沒有料想到的錯誤和效率問題，更思考使自己的程式表現提升的方法，因此，儘管最後競賽表現未到非常出色，依舊學到了很多。