**AIMA Final Project Report**

**骰子吹牛**

  R00943104 電子三 許家豪

R01942048 電信二 唐偉軒

R01942049 電信二 柯品伃

1. **研究動機與目的**

骰子吹牛除了因骰子的隨機性包含不完全的資訊，玩家的個性亦會顯現於叫牌的內容與抓牌的時機。不需語意分析與影像處理，便有許多角度能切入進行 learning，在此研究中我們將探討兩個面向：何種個性勝率較高，與當所有玩家的個性固定時，要如何learn對手的個性。

**二、骰子吹牛規則介紹**

1. 人數：至少兩人

2. 道具：每人五顆骰子，各一個骰盅

3. 玩法：

(1) 每個玩家搖動骰盅後，在不讓他人看到的情況下看自己的點數

(2) 由起始玩家開始叫牌，接著每回合玩家都可以選擇「叫牌」或「抓牌」，若有人抓牌則遊戲結束

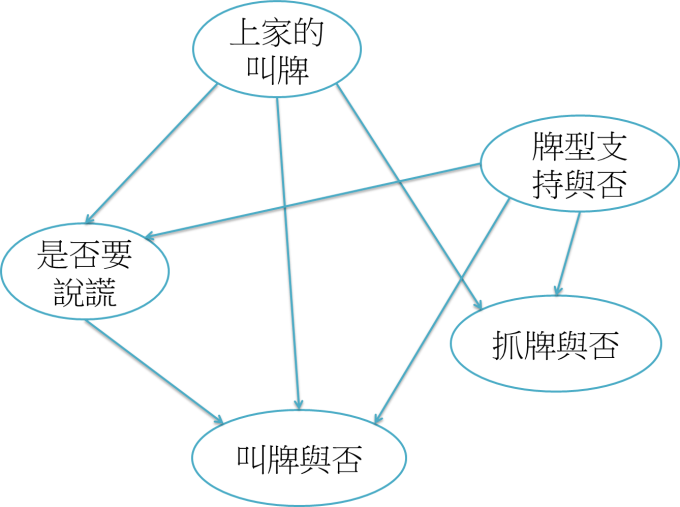
(3) 叫牌：叫牌「X個Y」，代表其玩家預測所有玩家的骰子總數至少有X個點數為Y的骰子。每回合的叫牌需比上家的叫牌個數更大，或個數相同但點數更大

(4) 抓牌：若認為上家的喊牌不成立則「抓牌」，此時所有玩家打開骰盅，若場面上不足上家喊的X個Y，則抓牌的一方勝，若成立，則抓牌方輸

(5) 骰子上的「一點」可代表任何點數，但若曾有玩家喊的點數是一，骰子上的「一點」便不再能代表其他點數

**三、過去文獻**

文獻[師大]的遊戲規則與我們的相似，但限定玩家為兩人，且骰子上的「一點」不能當作任意點的點數。實際去計算每種牌型的組合個數[tactic]，有一半以上的機率，對手的手牌中某個點數有兩顆以上，因此若相信對手是誠實喊牌，則當作他有兩顆該點數的骰子，若不相信，則限定該點數最多只有一個，以此方式便能依對手的喊牌，逐漸推論對手底牌為何。除上述經驗法則外，[師大]亦提出Bayesian network的架構〈圖一〉計算條件機率。首先考慮上一家叫牌的個數，並判斷自己的牌型是否支持 (若對手的叫牌個數只比自己該點數的骰子個數多三個以下就算支持)，依照自己牌型是否支持可以考慮要抓牌或叫牌。其中叫牌可以分為誠實叫牌或說謊叫牌，若叫的個數只比自己所有的骰子個數多一顆則算誠實喊牌。



圖一、骰子吹牛的Bayesian network [師大]

最後的實驗成果分別設定不同固定個性的電腦對手，在不加入Bayesian network的狀況下勝率約 50~70%，加入 Bayesian network的架構後勝率能增加5~10%；但若與真人對戰勝率約 40~50%，加入 Bayesian network的架構卻不一定會使勝率提高，當真人知道程式會根據原先行為做策略調整時，真人便會不斷變換策略，此時Bayesian network的存在反而會使結果變差。

**四、研究方法**

我們選擇嘗試與過去文獻不同的策略，不使用Bayesian network的方式，將遊戲由以下架構描述：

1. 不完全資訊

每個人無法看到他人的骰子，只能依照自己的骰子及其他對手的叫牌獲取資訊。我們的agent會計算機率表 **()**，記錄在遊戲當下，各種骰面情形出現的機率，並且根據所有其他玩家喊的骰面點數與個數來作更新。

1. 機率表：

遊戲一開始骰完骰子後，初始機率表的算法為，將自己所骰到的骰子點數與個數當作已知，剩下的對手骰子當作未知，算出所有骰面成立的機率。舉例來說，若共有三個玩家，自己的骰子狀況為2個一點，3個兩點，則未知骰子總數為10個。

骰面成立的機率詳細算法如下：

，其中為自己點的個數。

註：「一點」可當作任何點數，故當時，點的個數=實際點的個數+實際一點的個數

，

其中若Y1則 (骰到Y點或1點都可當作Y點); 反之則。

1. 更新機率表：

當有對手喊出「X個Y」時，若選擇相信他，則將此資訊和自己的骰子狀況當作已知，算出機率表 **()**。舉例來說，若共有三個玩家，自己的骰子狀況為2個一點，3個兩點，當對手喊出「7個兩點」時，已知資訊為2個一點，5個兩點，且未知骰子總數為8個。將此與原本的機率表 做linear combination，即可得到更新後的機率表：

，

其中為此對手的可信度，將於下方「2. 個性」時詳細說明。

1. 個性

遊戲進行時，需要判斷是否要抓上家，若不抓則需依據對上家的信任度更新自己的機率表，並依照機率表決定該叫什麼牌。

1. 玩家本身：
2. 抓牌的機率底線 ()：若上家叫的牌面成立的機率小於catchThreshold就抓他。當代表永遠不抓，當 越高則代表越傾向於抓別人。
3. 叫牌的風險程度 (riskRate)：

對手機率若低於這個數就抓對方，若不抓就喊

玩家本身：

riskRate = 0.1 #risk rate = 0:會挑所有能喊的骰子裡面，機率最高的；risk rate = 1，代表挑裡面機率最小的那個

catchThreshold = 0.1 #

1.     機率表 (Probability Table)：記錄在遊戲當下，各種骰面情形出現的機率，並且根據現在所有玩家所喊的個數＆骰面點數來作更新

2.     抓人的機率判斷(catchThreshold)：用在機率表上判斷是否應該抓人，超過就抓

3.     喊聲的機率判斷(yellThreshold)：用在機率表上判斷，超過就喊聲

4. 個別玩家喊聲狀況(agentYell)：針對個別玩家記錄他們每一次的喊聲狀況，用以做最後跟實際狀況的比較，以更新每一個玩家的可信度

敵對玩家：

1.    遊戲現況(prevYell)：遊戲到上一個玩家為止所喊的個數＆骰面

2.    每個玩家的可信度(Credibility)：根據每次玩完的結果去對每次玩家做一次可信度的更新

玩家本身：

1.

敵對玩家：

1.    遊戲現況(prevYell)：遊戲到上一個玩家為止所喊的個數＆骰面

2.    每個玩家的可信度(Credibility)：根據每次玩完的結果去對每次玩家做一次可信度的更新

過去文獻中沒有

1. 不完全資訊：

每個人無法看到他人的骰子，只能依照自己的骰子及其他玩家的叫牌獲取資訊。

計算各種牌面的機率。

2. 不同玩家有不同個性：

骰子吹牛包含不完全的資訊，而且需要判斷其他玩家說的真偽，不同玩家有不同的個性，

五、研究方法

(1) For multiple players more than 2, you might need to implant some “modes or personalities” into agents.

(2) I’m not familiar with this game (so correct me if I’m wrong), but isn’t that a simple probability problem? I meant, you should be able to calculate the probability of “X個Y”, and then simply catch when that prob is less than 0.5?

(3) Based on your descriptions, there’s only one winner and one loser? But it seems (from the wiki) usually you need to bet, and try to maximize your profit in that case?

參考資料：

[1] 黃信翰，“吹牛骰子之人工智慧研究”，國立臺灣師範大學資訊工程研究所碩士論文，2009。

[2] G. H. Freeman, “The Tactics of Liar Dice,” Applied Statistics, Vol. 38, No. 3, pp.

507-516, 1989.

攻擊強 or 弱？ 用risk\_rate決定

3小時前 · 讚

唐偉軒 傾向抓牌 or 叫牌？用catch\_threshold決定

3小時前 · 讚

唐偉軒 信賴 or 猜忌？用credibility\_list決定

3小時前 · 讚

唐偉軒 不可預測 or 乖乖玩？用yell\_one\_prob決定

四、Learning：

就如名稱「吹牛」，因為在遊戲結束前無從得知對手的底牌，對手的叫牌過

程是唯一的資訊，於是在叫牌中吹牛、提供錯誤的資訊來誤導對手也是合理的策

略。即使是相同的牌型，也會因為玩家的心態、個性不同，影響到中途的發展，

導致不同的結果。

**GameState:**

玩家本身：

1.     機率表 (Probability Table)：記錄在遊戲當下，各種骰面情形出現的機率，並且根據現在所有玩家所喊的個數＆骰面點數來作更新

2.     抓人的機率判斷(catchThreshold)：用在機率表上判斷是否應該抓人，超過就抓

3.     喊聲的機率判斷(yellThreshold)：用在機率表上判斷，超過就喊聲

4. 個別玩家喊聲狀況(agentYell)：針對個別玩家記錄他們每一次的喊聲狀況，用以做最後跟實際狀況的比較，以更新每一個玩家的可信度

敵對玩家：

1.    遊戲現況(prevYell)：遊戲到上一個玩家為止所喊的個數＆骰面

2.    每個玩家的可信度(Credibility)：根據每次玩完的結果去對每次玩家做一次可信度的更新

**Implementation:**

我們probability table的initial state算法是，將自己所骰到的骰子數字與數量當作determined，剩下的骰子則假設每種點數出現的機率各是1/6，並算出所有可能的機率。舉例來說，假設有3個玩家，每個玩家有5個骰子，則我們共有15個骰子。假設我們骰到3個2、2個1，則已知場面上有3個2、2個1，剩下的10個骰子再算出各有1~10個1~6的機率。另外因為骰子吹牛的特殊規定：當沒有人喊過「X個1點」時，1點可代表任何點數，所以其他的骰子數字數量需要加上1點的數量。

當每次有玩家喊「X個Y」(yell) 時，我們在更新probability table時，會乘上該玩家的credibility，其概念與課堂上學到的Weighted Majority Algorithm 相似，對於不同的玩家(hypothesis) 我們給予不同的weighting，藉此來差異化不同個性的玩家。

上一家喊完後，我們需要判斷我們要不要抓上一家。利用 probability table 我們可以知道我們猜測的綜觀骰子數量機率，如果上一家喊的 yell 機率小於我們訂的 catchThreshold 我們就抓他。輪到自己要喊 yell 時，我們可以選比自己的 yellThreshold 大的最小機率的 yell 喊。yellThreshold 代表的是一種 safety 的概念，當自己覺得這種 pattern 出現的機率大於我們的 yellThreshold，即代表喊此 yell 的風險是我們可以接受的。喊越接近此 yellThreshold 的 yell 能讓下一玩家承受越大的風險。舉例來說，當我們能接受喊「5個2」的風險時，就不要只喊「4個2」，要盡量讓下一家的難喊又不敢抓自己。

當遊戲進行到後面，可能會出現上一家喊的pattern機率大於我們 catchThreshold，但又沒有任何pattern比我們訂的yellThreshold大，就需要比較catchRisk和yellRisk，如果抓人風險較大則選擇繼續喊下一組個數與骰面點數；若被抓風險較大則去抓上一個玩家。

**進階的功能(接下來打算完成的部分)：**

如果之前的功能和實驗的相當成功順利, 我們將把之前的一些限制所拿掉,

1.

在之前的版本,我們假設每個玩家都知道彼此間的個性, 所以會給定他們相信其他玩家的機率, 對應到實際現實世界的情況, 可以看成和一群老朋友一起玩骰子吹牛, 因為知道彼此間的個性,所以在玩遊戲的過程中,可以知道每個人的可信度,但如果和一群不認識的人玩骰子吹牛時,就要經過學習的過程,才能知道各個玩家的特性,所以我們之後希望能讓電腦學會彼此間的個性, 目前的想法是, 一開始初始化電腦對每個玩家有相同的信任程度, 經過一個回合結束後, 每個玩家會亮出自己所有的骰子,這時每個人就可以根據這時的牌面狀況和當時所喊的情形去反推當時這個人在喊牌時是否過於誇大或是保守,借此來學習場面上玩家的特性。

2.

第二個能做的事情是,之前的版本,每個玩家要不要冒險的行為是由給定參數所決定的,但實際上真人在玩遊戲的時候,我們會根據目前所處於的狀況來調整策略,所以進階版本還能做的是,除了先前提過可以去學習對他人的信任度之外,自己在玩的過程中可以適度的調整參數,比如說電腦玩家可以根據自己的輸贏結果來改變下次玩的時候的參數。