Banning and queueing in tournament Hearthstone

svangen#2391 (EU)

Abstract

We propose a game theoretic approach to *banning* and *queueing* decisions in the most common type of tournament Hearthstone: conquest with a ban. Optimal strategies can be calculated as Nash equilibrium of appropriately constructed zero-sum games, using pairwise deck win rates as input. We investigate a number of realistic numerical examples, concluding that the queueing decision is almost irrelevant (the cost of deviating to a "naive" random strategy is very low), but banning the right deck is very important.

Introduction

In tournament Hearthstone, players bring a set of prepared decks, their *lineup*. The most common tournament setting is best-of-5 conquest with a ban: players bring four decks each, one of which is banned by the opponent in the beginning of each match. The ban is the first strategic decision of the match. The players then play game after game using their remaining three decks, and the first player to get a win with every deck is the winner. The order of the wins does not matter, so in every game the players first have to make a choice: which deck shall I play with? (Or *queue*, in the jargon.)

This note studies these two types of decisions from a game theoretic point of view: which deck to ban, and which deck to queue? As input we take the matrix of pairwise win percentages for the decks, which are assumed to be known. This is not completely unrealistic: in competitive tournaments, players usually bring decks corresponding to a reasonably small set of *archetypes*, for which the pairwise win rates are sort-of well known: the Vicious Syndicate (http://www.vicioussyndicate.com/) collects data from ranked play and continuously publishes win rates of different decks, see Table 1 for the seven most popular decks as of 13 Feb 2017. These statistics are of course not the full story, since players will "tech" their decks to target certain matchups, thereby increasing certain win rates at the price of decreasing others. Nevertheless, it is not unreasonable to assume that experienced players can estimate a matrix of pairwise win rates given knowledge about their opponent's decks.

We will consider a number of numerical examples below, using the win rates from Table 1. Matlab code to replicate the calculations is given in the Appendix.

	Aggro Shaman	Mid-Jade Shaman	Pirate Warrior	Reno Warlock	Miracle Rogue	Reno Mage	Jade Druid
Aggro Shaman	0.50	0.51	0.61	0.53	0.61	0.46	0.66
Mid-Jade Shaman	0.49	0.50	0.49	0.43	0.50	0.52	0.56
Pirate Warrior	0.39	0.51	0.50	0.54	0.63	0.43	0.60
Reno Warlock	0.47	0.57	0.46	0.50	0.40	0.44	0.40
Miracle Rogue	0.39	0.50	0.37	0.60	0.50	0.52	0.59
Reno Mage	0.54	0.48	0.57	0.56	0.48	0.50	0.32
Jade Druid	0.34	0.44	0.40	0.60	0.41	0.68	0.50

Table 1: Vicious Syndicate pairwise win rates for the seven most popular decks as of 13 Feb 2017.

Zero-sum games and Nash equilibrium

A *zero-sum game* is a simultaneous game for two players, Player A and Player B, where each player has a finite number of possible actions. The game is characterized by a matrix $M = (m_{ij})$, where m_{ij} is Player A's payoff given that the players choose actions i and j, respectively. The payoff for the Player B is $-M_{ij}$ (hence zero-sum). A mixed strategy for Player A is a vector α with $\alpha \geq 0$ and $\sum_i \alpha_i = 1$, with the interpretation that Player A chooses action i with probability α_i. Similarly, let β be denote a mixed strategy for Player B. The expected payoff for Player A is then $\alpha^T M \beta$ (and the expected payoff for Player B is $-\alpha^T M \beta$). A Nash equilibrium is a set of strategies $(\bar{\alpha}, \bar{\beta})$ such that no single player can do better by deviating:

$$\bar{\alpha}^T M \beta \geq \bar{\alpha}^T M \bar{\beta} \text{ and } -\alpha^T M \bar{\beta} \geq -\bar{\alpha}^T M \bar{\beta}, \text{ for all } \alpha, \beta.$$

A Nash equilibrium always exists for each matrix M, and is easy to compute with linear programming.

The queuing problem

We consider the following conquest setting: Player A has m decks, $A_1, \ldots, A_m$, and Player B has n decks $B_1, \ldots, B_n$. The first player to win with all their decks is the winner. We let $P = (p_{ij})$ denote the win probability matrix:

$$p_{ij} = \text{Prob}(A_i \text{ wins against } B_j).$$

We call this matrix the *game*, and would like to talk about the *value* $V(P)$ of this game, where $V(P)$ is Player A's win probability for the whole match, given optimal queuing decisions by both players. This concept is not yet well defined, but there are two simple cases that we can handle directly:

$$\begin{aligned} \text{If } n = 1, \text{ then } V(P) &= \prod_{i=1}^{m} p_{in}, \\ \text{If } m = 1, \text{ then } V(P) &= 1 - \prod_{j=1}^{n} (1 - p_{mj}). \end{aligned}$$

The reason is pretty obvious: if $n = 1$, that means that Player B has only one deck remaining, so to win, Player A needs to beat that deck with all their m remaining decks. Conversely, if $m = 1$, Player A has only one deck remaining, so will have to avoid getting beat by all the opponents decks.

Next, we make the following observation, when $m > 1$ and $n > 1$. After a given queuing decision, A_i and B_j, two things can happen. With probability p_{ij}, player A wins, and the players now face a different game:

$$W_i := \text{submatrix of } P \text{ where the } i\text{th row has been removed.}$$

With probability $1 - p_{ij}$, Player A loses, and the players instead face the game

$$L_j := \text{submatrix of } P \text{ where the } j\text{th row has been removed.}$$

To summarize, conditional on the queuing choices A_i and B_j, we can say that the value of the game is

$$q_{ij} := p_{ij}V(W_i) + (1 - p_{ij})V(L_j),$$

assuming that both $V(W_i)$ and $V(L_j)$ are well defined. $M := Q - 0.5$, where $Q = (q_{ij})$, is the zero-sum game we have been looking for: let $(\bar{\alpha}, \bar{\beta})$ denote the corresponding Nash equilibrium. The expected win rate for the whole match for Player A is

$$V(P) := \bar{\alpha}^T Q \bar{\beta}.$$

Note that this completes the (recursive) definition of $V(P)$ for all P: the matrices for which we need to evaluate V loses either a row or a column in each recursive call, until they reach one of the well defined base cases.

Example 1: m = n = 2

Let (A_1, A_2) = (Pirate Warrior, Reno Mage) and (B_1, B_2) = (Mid-Jade Shaman, Reno Warlock). From Table 1 we see that

$$P = \begin{pmatrix} 0.51 & 0.54 \\ 0.48 & 0.56 \end{pmatrix}, \text{ which gives } Q = \begin{pmatrix} 0.5415 & 0.5291 \\ 0.5291 & 0.5415 \end{pmatrix}.$$

The Nash equilibrium is $\bar{\alpha} = \bar{\beta} = (0.5, 0.5)$. This symmetry is not a coincidence: we can calculate

$$\begin{aligned} q_{11} = q_{22} &= p_{11}p_{21} + p_{11}p_{22} + p_{12}p_{22} - p_{11}p_{12}p_{22} - p_{11}p_{21}p_{22} \\ q_{12} = q_{21} &= p_{11}p_{21} + p_{12}p_{21} + p_{12}p_{22} - p_{12}p_{11}p_{21} - p_{12}p_{21}p_{22}, \end{aligned}$$

for any P. This gives the Nash equilibrium $\bar{\alpha} = \bar{\beta} = (0.5, 0.5)$ for any values of q_{11} and q_{12}. Moreover, it is not costly to deviate: Player A's win probability equals $(0.5\ 0.5)Q\beta = 0.5(q_{11} + q_{12})(\beta_1 + \beta_2) = 0.5(q_{11} + q_{12})$ for any β.

It is still important to not be predictable: if it was somehow known that Player B will queue B_1 first, Player A could gain by queueing A_1, resulting in win rate 0.542 instead of the "fair" win rate $V(P) = 0.535$ corresponding to the Nash equilibrium.

Example 2: m = 2, n = 3

The case $m = 2$, $n = 3$ (or vice versa) is the smallest situation where we get non-symmetric queuing strategies. Let A_1, A_2, B_1, B_2 be the same as in the previous example, but add in B_3 = Jade Druid for Player B. Then

$$P = \begin{pmatrix} 0.51 & 0.54 & 0.60 \\ 0.48 & 0.56 & 0.32 \end{pmatrix}.$$

The Nash equilibrium is quite interesting: $\alpha = (0.49, 0.51)$, $\beta = (0, 0.60, 0.40)$, (rounded to 2 decimal places), i.e. Player B should never queue the Mid-Jade Shaman in this situation.

However, the cost of deviating is small: if Player B queues all decks with equal probability, i.e. $\beta = (1/3, 1/3, 1/3)$, Player A's gain is only 0.025 percentage points, for a total win probability of 0.6879 instead of 0.6876.

Example 3: Is optimal queuing important?

The two previous examples indicate that deviating from the Nash equilibrium queueing strategies might typically not be very costly. To investigate this, we form all possible 3-vs-3 matches with decks listed in Table 1. Then we compute Player A's equilibrium win rates, and the win rates when Player B deviates and queues randomly, i.e. $\beta = (1/3, 1/3, 1/3)$. The gains are very small: the maximal gain is less than 0.1 percentage points, and occurs for the following lineups: (Aggro Shaman, Miracle Rogue, Jade Druid) vs (Aggro Shaman, Miracle Rogue, Reno Mage).

We conclude that the queueing decision is actually not important at all from a practical point of view: one can simply queue each deck with equal probability.

The banning problem

As above, Player A has m decks, $A_1, \ldots, A_m$, and Player B has n decks $B_1, \ldots, B_n$, and the $m \times n$-matrix P encodes the pairwise win rates for the decks. (Since banning happens in the start of the match, we typically have $m = n$ here.) We again consider mixed banning strategies α and β, where Player A bans B_i with probability α_i, and Player B bans A_j with probability β_j. Now, assume that B_i and A_j were banned. Then the players face the game

$$R_{ij} = \text{submatrix of } P \text{ where the } i\text{th row and } j\text{th column has been removed.}$$

But this game has a value, namely $V(R_{ij})$. We collect these in a matrix:

$$S = (s_{ij}), \text{ where } s_{ij} = V(R_{ij}),$$

and compute the Nash equilibrium $(\bar{\alpha}, \bar{\beta})$ for the payoff matrix $S - 0.5$. These are the optimal banning strategies.

Example 4: m = n = 4

Let (A_1, A_2, A_3, A_4) = (Aggro Shaman, Pirate Warrior, Reno Warlock, Miracle Rogue) and (B_1, B_2, B_3, B_4) = (Mid-Jade Shaman, Reno Warlock, Reno Mage, Jade Druid). From Table 1, we see that

$$P = \begin{pmatrix} 0.51 & 0.53 & 0.46 & 0.66 \\ 0.51 & 0.54 & 0.43 & 0.60 \\ 0.57 & 0.50 & 0.44 & 0.40 \\ 0.50 & 0.60 & 0.52 & 0.59 \end{pmatrix},$$

which gives $\bar{\alpha} = (0, 0, 1, 0)$ and $\bar{\beta} = (1, 0, 0, 0)$. That is, Player A always bans Reno Mage, and Player B always bans Aggro Shaman.

The result makes intuitive sense: Player A will ban Reno Mage, since that is an unfavored matchup for all decks except Miracle Rogue, which is almost break-even at 0.52. Similarly, Aggro Shaman is a poor matchup for all Player B's decks except Reno Mage (which Player B understands will get banned).

It is important to ban the right deck. For example, if Player B were to ban Pirate Warrior instead, Player A's win rate goes from 0.56 to 0.57, a gain of a whole percentage point. Even worse, if Player B bans the Reno Warlock, Player A's win rate goes to 0.62, for a gain of more than 5.5 percentage points.

Appendix: Matlab code

Note that the functions call `LemkeHowson.m` to compute the Nash equilibrium, which can be found here: `https://uk.mathworks.com/matlabcentral/fileexchange/44279-lemke-howson-algorithm-for-2-player-games`.

Functions

```
function [V Q alpha beta] = get_V(P)
[m,n] = size(P);
if m == 1
    V = 1 - prod(1 - P);
    alpha = 1;
    beta = ones(1,n) / n;
    Q = [];
elseif n == 1
    V = prod(P);
    alpha = ones(m,1)/m;
    beta = 1;
    Q = [];
else
    Q = NaN(m,n);
    for i = 1:m
        for j = 1:n
            [Pw Pl] = deal(P);
            Pw(i,:) = [];
            Pl(:,j) = [];
            Q(i,j) = P(i,j)*get_V(Pw) + (1-P(i,j))*get_V(Pl);
        end
    end
    M = Q - 0.5;
    x = LemkeHowson(M,1-M);
    alpha = x{1};
    beta  = x{2};
    V = alpha'*Q*beta;
end
```

```
function [V S alpha beta] = get_V_ban(P)
assert(size(P,1) == size(P,2))
n = size(P,1);
S = NaN(n,n);
for i = 1:n
    for j = 1:n
        R = P;
        R(:,i) = []; % ban Bi
        R(j,:) = []; % ban Aj
        S(i,j) = get_V(R);
    end
end
M = S - 0.5;
x = LemkeHowson(M,1-M);
alpha = x{1};
beta  = x{2};
V = alpha'*S*beta;
```

Examples

```
% VS win rates:
deck = {'Aggro Shaman','Mid-Jade Shaman','Pirate Warrior',...
        'Reno Warlock','Miracle Rogue','Reno Mage','Jade Druid'};
PP   = [0.50    0.51    0.61    0.53    0.61    0.46    0.66
        0.49    0.50    0.49    0.43    0.50    0.52    0.56
        0.39    0.51    0.50    0.54    0.63    0.43    0.60
        0.47    0.57    0.46    0.50    0.40    0.44    0.40
        0.39    0.50    0.37    0.60    0.50    0.52    0.59
        0.54    0.48    0.57    0.56    0.48    0.50    0.32
        0.34    0.44    0.40    0.60    0.41    0.68    0.50];

% Example 1:
A = {'Pirate Warrior','Reno Mage'};
B = {'Mid-Jade Shaman','Reno Warlock'};
P = PP(ismember(deck,A),ismember(deck,B));
[V,Q,alpha,beta] = get_V(P);
gain = [1 0]*Q*[1; 0] - V;

% Example 2:
A = {'Pirate Warrior','Reno Mage'};
B = {'Mid-Jade Shaman','Reno Warlock','Jade Druid'};
P = PP(ismember(deck,A),ismember(deck,B));
[V,Q,alpha,beta] = get_V(P);
gain = alpha'*vv*[1/3;1/3;1/3] - V;

% Example 3:
lineups = NaN(50,3); n = 0;
for j1 = 1:N-2
    for j2 = j1+1:N-1
        for j3 = j2+1:N
            js = [j1 j2 j3];
            n = n+1;
            lineups(n,:) = js;
        end
    end
end
lineups = lineups(1:n,:);
gain = zeros(n,n);
for j = 1:n-1
    for k = j+1:n
        ia = lineups(j,:);
        ib = lineups(k,:);
        P = PP(ia,ib);
        [V,Q,alpha,beta] = get_V(P);
        gain(j,k) = alpha'*Q*[1/3;1/3;1/3] - V;
    end
end
[j,k] = find(gain == max(max(gain)));
deck(lineups(j,:))
deck(lineups(k,:))

% Example 4:
A = {'Aggro Shaman','Pirate Warrior','Reno Warlock','Miracle Rogue'};
B = {'Mid-Jade Shaman','Reno Warlock','Reno Mage','Jade Druid'};
P = PP(ismember(deck,A),ismember(deck,B));
[V,S,alpha,beta] = get_V_ban(P);
alpha'*S*[0 1 0 0]'; % ban Warrior
alpha'*S*[0 0 1 0]' - V; % ban Warlock
```