More on games (Ch. 5.4-5.7)

IT'S A CHRISTMAS TREE WITH A HEAP OF PRESENTS UNDERNEATH!

... WE'RE NOT INVITING YOU HOME NEXT YEAR.
Random games

If there are too many possibilities for all the chance outcomes to “average them all”, you can sample.

This means you can search the chance-tree and just randomly select outcomes (based on probabilities) for each chance node.

If you have a large number of samples, this should converge to the average.
How to find which actions are “good”? 

The “Upper Confidence Bound applied to Trees” UCT is commonly used: 

$$\max_{n \in \text{children}} \left( \frac{\text{win}(n)}{\text{times}(n)} + \sqrt{\frac{2 \ln \text{times}(\text{parent}(n))}{\text{times}(n)}} \right)$$

This ensures a trade off between checking branches you haven't explored much and exploring hopeful branches

( [https://www.youtube.com/watch?v=Fbs4lnGLS8M](https://www.youtube.com/watch?v=Fbs4lnGLS8M) )
MCTS
MCTS

\[
\frac{\text{win}(n)}{\text{times}(n)} + \sqrt{\frac{2 \ln \text{times}(\text{parent}(n))}{\text{times}(n)}}
\]

\[
= \frac{0}{0} + \sqrt{\frac{2 \ln 0}{0}}
\]

\[
= \infty
\]
MCTS

\[
\frac{\text{win}(n)}{\text{times}(n)} + \sqrt{\frac{2 \ln \text{times}(\text{parent}(n))}{\text{times}(n)}}
\]

= \frac{0}{0} + \sqrt{\frac{2 \ln 0}{0}}

= \infty
MCTS

Pick max on depth 1 (I'll pick left-most)
MCTS

(random playout)

lose
MCTS

update (all the way to root)
(random playout)
update UCB values (all nodes)
select max UCB on depth 1 & rollout

MCTS

win
update statistics

MCTS

0 0/1 ∞ 1/1 ∞ 0/0

win
update UCB vals

MCTS

1.1 0/1 2.1 1/1 ∞ 0/0
MCTS

select max UCB on depth 1 & rollout

1/2

1/1 0/1 2.1

∞ 0/0

win
MCTS

update statistics

update statistics

1.1 0/1 2.1 1/1 ∞ 1/1

win
update UCB vals

MCTS
select max UCB on depth 1

max on depth 1 a tie, can pick either
MCTS

select max UCB on depth 2

also a tie on depth 2, can pick either (I go left)
MCTS

update statistics

1.4 0/1 2.5 2/2 2.5 1/1

∞ 1/1 ∞ 0/0

win
update UCB vals

MCTS

3/4

1.7 0/1 2.1 2/2 2.7 1/1

2.2 1/1 ∞ 0/0
MCTS

Pros:
(1) The “random playouts” are essentially generating a mid-state evaluation for you
(2) Has shown to work well on wide & deep trees, can also combine distributed comp.

Cons:
(1) Does not work well if the state does not “build up” well
(2) Often does not work on 1-player games
MCTS in games

AlphaGo/Zero has been in the news recently, and is also based on neural networks.

AlphaGo uses Monte-Carlo tree search guided by the neural network to prune useless parts.

Often limiting Monte-Carlo in a static way reduces the effectiveness, much like mid-state evaluations can limit algorithm effectiveness.
MCTS in games

Basically, AlphaGo uses a neural network to “prune” parts for a Monte-carlo search.
DIFFICULTY OF VARIOUS GAMES FOR COMPUTERS

EASY

SOLVED for all possible positions

SOLVED for starting positions

Computers can play perfectly

Computers can beat top humans

Computers still lose to top humans
  (but focused R&D could change this)

Computers may never outplay humans

HARD

- Tic-Tac-Toe
- Nim
- Ghost (1989)
- Connect Four (1995)
- Gomoku
- Checkers (2007)
- Scrabble
- Counterstrike
- Reversi
- Beer Pong (UWAC robot)
- February 10, 1996: First win by computer against top human
- November 21, 2005: Last win by human against top computer
- Chess
- Jeopardy!
- Starcraft
- Poker
- Go
- Arimaa
- Mao
- Snakes and Ladders
- Seven Minutes in Heaven
- Calvinball
Game theory

Typically game theory uses a payoff matrix to represent the value of actions.

The first value is the reward for the left player, right for top (positive is good for both).
Dominance & equilibrium

Here is the famous “prisoner's dilemma”

Each player chooses one action without knowing the other's and the game is only played once.

<table>
<thead>
<tr>
<th>PRISONER 1</th>
<th>Confess</th>
<th>Lie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confess</td>
<td>-8, -8</td>
<td>0, -10</td>
</tr>
<tr>
<td>Lie</td>
<td>-10, 0</td>
<td>-1, -1</td>
</tr>
</tbody>
</table>

- If both players confess, they each get -8 years.
- If one player confesses and the other lies, the confessor gets 0 years and the liar gets -10 years.
- If both players lie, they each get -1 year.

- If one player is not guilty and the other is guilty, the innocent player gets 2 years, and the guilty player gets 5 years.
- If both players are not guilty, they each get 5 years.
- If both players are guilty, the innocent player gets 1 year, and the guilty player gets 3 years.
Dominance & equilibrium

What option would you pick?
Why?

<table>
<thead>
<tr>
<th></th>
<th>PRISONER 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confess</td>
<td>Confess</td>
</tr>
<tr>
<td>Confess</td>
<td>-8, -8</td>
</tr>
<tr>
<td>Lie</td>
<td>-10, -10</td>
</tr>
</tbody>
</table>

[Decision Matrix Diagram]
What would a rational agent pick?

If prisoner 2 confesses, we are in the first column... -8 if we confess, or -10 if we lie
--> Thus we should confess

If prisoner 2 lies, we are in the second column, 0 if we confess, -1 if we lie
--> We should confess
Dominance & equilibrium

It turns out regardless of the other player's action, it is in our personal interest to confess.

This is the Nash equilibrium, as any deviation of strategy (i.e. lying) can result in a lower score (i.e. if opponent confesses).

The Nash equilibrium looks at the worst case and is greedy.

<table>
<thead>
<tr>
<th>PRISONER 2</th>
<th>Confess</th>
<th>Lie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confess</td>
<td>-8, -8</td>
<td>0, -10</td>
</tr>
<tr>
<td>Lie</td>
<td>-10, 0</td>
<td>-1, -1</td>
</tr>
</tbody>
</table>
Dominance & equilibrium

Formally, a Nash equilibrium is when the combined strategies of all players give no incentive for any single player to change.

In other words, if any single person decides to change strategies, they cannot improve.
Dominance & equilibrium

Alternatively, a **Pareto optimum** is a state where no other state can result in a gain or tie for all players (excluding all ties)

If the PD game, [-8, -8] is a Nash equilibrium, but is not a Pareto optimum (as [-1, -1] better for both players)

However [-10,0] is also a Pareto optimum...
Every game has at least one Nash equilibrium and Pareto optimum, however...

- Nash equilibrium might not be the best outcome for all players (like PD game, assumes no cooperation)

- A Pareto optimum might not be stable (in PD the [-10,0] is unstable as player 1 wants to switch off “lie” and to “confess” if they play again or know strategy)
Find the Nash and Pareto for the following: (about lecturing in a certain csci class)

| Teacher | Student
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>prepare well</td>
<td>pay attention</td>
</tr>
<tr>
<td>slack off</td>
<td>sleep</td>
</tr>
<tr>
<td></td>
<td>1, -5</td>
</tr>
</tbody>
</table>
How do we formally find a Nash equilibrium?

If it is zero-sum game, can use minimax as neither player wants to switch for Nash (our PD example was not zero sum)

Let's play a simple number game: two players write down either 1 or 0 then show each other. If the sum is odd, player one wins. Otherwise, player 2 wins (on even sum)
Find best strategy

This gives the following payoffs:

<table>
<thead>
<tr>
<th></th>
<th>Pick 0</th>
<th>Pick 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick 0</td>
<td>-1, 1</td>
<td>1, -1</td>
</tr>
<tr>
<td>Pick 1</td>
<td>1, -1</td>
<td>-1, 1</td>
</tr>
</tbody>
</table>

(player 1's value first, then player 2's value)

We will run minimax on this tree twice:
1. Once with player 1 knowing player 2's move (i.e. choosing after them)
2. Once with player 2 knowing player 1's move
Find best strategy

Player 1 to go first (max):

If player 1 goes first, it will always lose
Find best strategy

Player 2 to go first (min):

If player 2 goes first, it will always lose
Find best strategy

This is not useful, and only really tells us that the best strategy is between -1 and 1 (which is fairly obvious)

This minimax strategy can only find pure strategies (i.e. you should play a single move 100% of the time)

To find a mixed strategy, we need to turn to linear programming
Find best strategy

A **pure strategy** is one where a player always picks the same strategy (deterministic)

A **mixed strategy** is when a player chooses actions probabilistically from a fixed probability distribution (i.e. the percent of time they pick an action is fixed)

If one strategy is better or equal to all others across all responses, it is a **dominant strategy**
Find best strategy

The definition of a Nash equilibrium is when no one has an incentive to change the combined strategy between all players.

So we will only consider our opponent's rewards (and not consider our own).

This is a bit weird since we are not considering our own rewards at all, which is why the Nash equilibrium is sometimes criticized.
Find best strategy

First we parameterize this and make the tree stochastic:

Player 1 will choose action “0” with probability $p$, and action “1” with $(1-p)$

If player 2 always picks 0, so the payoff for $p_2$: 
$$(1)p + (-1)(1-p)$$

If player 2 always picks 1, so the payoff for $p_2$: 
$$(-1)p + (1)(1-p)$$
Find best strategy

Plot these two lines:
\[ U = (1)p + (-1)(1-p) \]
\[ U = (-1)p + (1)(1-p) \]

As we maximize, the opponent gets to pick which line to play.

Thus we choose the intersection.
Thus we find that our best strategy is to play 0 half the time and 1 the other half.

The result is we win as much as we lose on average, and the overall game result is 0.

Player 2 can find their strategy in this method as well, and will get the same 50/50 strategy (this is not always the case that both players play the same for Nash).
Find best strategy

We have two actions, so one parameter (p) and thus we look for the intersections of lines.

If we had 3 actions (rock-paper-scissors), we would have 2 parameters and look for the intersection of 3 planes (2D).

This can generalize to any number of actions (but not a lot of fun).
Find best strategy

How does this compare on PD?

<table>
<thead>
<tr>
<th></th>
<th>Confess</th>
<th>Lie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confess</td>
<td>-8, -8</td>
<td>0, -10</td>
</tr>
<tr>
<td>Lie</td>
<td>-10, 0</td>
<td>-1, -1</td>
</tr>
</tbody>
</table>

Player 1: \( p = \) prob confess...

P2 Confesses: \(-8*p + 0*(1-p)\)
P2 Lies: \(-10*p + (-1)*(1-p)\)

Cross at negative \( p \), but red line is better (confess)