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.
Announcements

Midterm will be on “gradescope” (will get an email from them... signup optional)

Writing 2 posted

Writing 1 regrades – until 10/25
If we are playing a “game of chance”, we can add chance nodes to the search tree.

Instead of either player picking max/min, it takes the expected value of its children.

This expected value is then passed up to the parent node which can choose to min/max this chance (or not).
Here is a simple slot machine example:

Random games

\[ V(\text{chance}) = -1 \cdot \frac{999}{1,000} + 100 \cdot \frac{1}{1,000} = -0.899 \]
Random games

You might need to modify your mid-state evaluation if you add chance nodes

Minimax just cares about the largest/smallest, but expected value is an implicit average:

R is better

L is better

\[ \begin{array}{ccc}
0.9 & 0.1 & 0.9 \\
0.1 & 0.4 & 0.2 \\
0.2 & 0.4 & 0.2 \\
\end{array} \]

\[ \begin{array}{ccc}
0.9 & 0.1 & 0.9 \\
0.1 & 0.4 & 0.2 \\
0.2 & 0.4 & 0.2 \\
\end{array} \]
Random games

Some partially observable games (i.e. card games) can be searched with chance nodes.

As there is a high degree of chance, often it is better to just assume full observability (i.e. you know the order of cards in the deck).

Then find which actions perform best over all possible chance outcomes (i.e. all possible deck orderings).
Random games

For example in blackjack, you can see what cards have been played and a few of the current cards in play.

You then compute all possible decks that could lead to the cards in play (and used cards).

Then find the value of all actions (hit or stand) averaged over all decks (assumed equal chance of possible decks happening).
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.
MCTS

How to find which actions are “good”?

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

\[
\max\left( \frac{\text{win}(n)}{\text{times}(n)} + \sqrt{\frac{2 \ln \text{TotalTimes}}{\text{times}(n)}} \right)
\]

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

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

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

\[
= \frac{0}{0} + \sqrt{\frac{2\ln0}{0}}
\]

\[
= \infty
\]
MCTS

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

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

\[
= \infty
\]
MCTS

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

(random playout)

lose
MCTS

update (all the way to root)
(random playout)

lose
MCTS

update UCB values (all nodes)
MCTS

select max UCB & rollout

win
update statistics

MCTS

win
update UCB vals

MCTS

1.1 0/1 2.1 1/1 ∞ 0/0
select max UCB & rollout

MCTS

lose
MCTS

update statistics

1/3

1.1 0/1 2.1 1/1 ∞ 0/1

lose
update UCB vals

1/3

1.4 0/1 2.5 1/1 1.4 0/1
select max UCB

MCTS
MCTS

rollout

1/3

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

∞ 0/0 0/0

win
MCTS

update statistics

win
MCTS

update UCB vals

2/4

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

2.2 1/1 ∞ 0/0
DIFFICULTY OF VARIOUS GAMES FOR COMPUTERS

EASY

SOLVED
COMPUTERS CAN
PLAY PERFECTLY

SOLVED
FOR ALL POSSIBLE
POSITIONS

SOLVED
FOR STARTING
POSITIONS

INIM

GHOST (1989)

CONNECT FOUR (1995)

GOMOKU

CHECKERS (2007)

SCRABBLE

COUNTERSTRIKE

REVERS

BEER PONG (UIUC 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

ARIMAA

GO

MAO

SNAKES AND LADDERS

SEVEN MINUTES
IN HEAVEN

CALVINBALL

COMPUTERS CAN BEAT TOP HUMANS

COMPUTERS STILL
LOSE TO TOP HUMANS
(BUT FOCUSED REK
COULD CHANGE THIS)

COMPUTERS MAY NEVER
OUTPLAY HUMANS

HARD