Lecture 20: Game theory in AI (II)

University of BRISTOL

EMAT31530/April 2018/Raul Santos-Rodriguez

Have a look at ...

... Russell and Norvig (Ch. 5)

... D.A. Blackwell and M.A. Girshick. Theory of Games and Statistical Decisions, Dover books on Mathematics.

... Python: `http://www.gambit-project.org/`

... Matlab: `http://mmiras.webs.uvigo.es/TUGlab/`

Outline

This lecture explores some of the strategies that make Game Theory in AI feasible. We will discuss:

- Evaluation functions
- Chance
- More complex games

Previously...

Games are good **models to understand decision making** in practice: simple rules but relevant conclusions.

- Classical game theory
- Adapting search for game playing
- Minimax method
- Alpha-Beta pruning

[AI Games] *2-player turn-taking zero sum games of perfect information with no randomness*

How to play a game

How to play

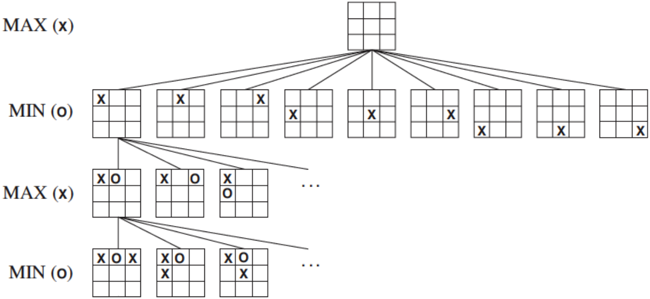
- Consider **all** the legal **moves** you can make
- Compute new position resulting from each move
- **Evaluate** each to determine which is best
- Make that move
- **Wait** for your opponent to move and **repeat**

Key problems are

- Representing the 'board' (**game state**)
- Generating all legal next boards (**game tree**)
- Evaluating a position (**utility**)

Complete game tree → only for simple games

[Idea] generate **partial game tree** for some turns.



How do we evaluate a state without a complete tree?

Evaluation function

[Evaluation function] measures goodness of a game position

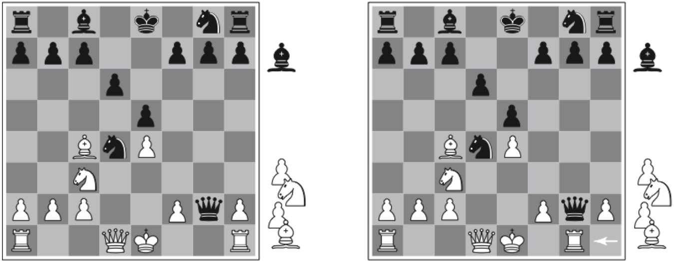
[Heuristic] non-negative estimate of the cost to a goal

[Zero-sum assumption] a single evaluation describes goodness of a state wrt both players

How to interpret an evaluation function

- $f(n) > 0$: position n good for me and bad for you
- $f(n) < 0$: position n bad for me and good for you
- $f(n)$ near 0: position n is a neutral position
- $f(n) = +\infty$: win for me
- $f(n) = -\infty$: win for you

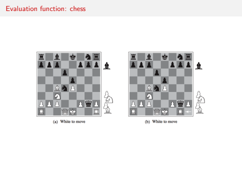
Evaluation function: chess



(a) White to move (b) White to move

2018-04-22

Evaluation function: chess



Two chess positions that differ only in the position of the rook at lower right. In (a), Black has an advantage of a knight and two pawns, which should be enough to win the game. In (b), White will capture the queen, giving it an advantage that should be strong enough to win.

Evaluation function: examples

Alan Turing's function for chess

$$f(n) = \frac{w(n)}{b(n)}$$

where $w(n)$ = sum of the point value of whites pieces and $b(n)$ = sum of black's

Evaluation functions specified as a weighted sum of position features

$$f(n) = w_1 \cdot feat_1(n) + w_2 \cdot feat_2(n) + \ldots + w_n \cdot feat_k(n)$$

Features for chess: piece count, piece placement, squares controlled, etc.

Tic-Tac-Toe

$$f(n) = [\text{\# of 3-lengths open for me}] - [\text{\# of 3-lengths open for you}]$$

where a 3-length is a complete row, column, or diagonal

Evaluation function: examples

Evaluation function: examples

Alan Turing's function for chess

$$f(n) = \frac{w(n)}{b(n)}$$

where $w(n)$ = sum of the point value of whites pieces and $b(n)$ = sum of black's

Evaluation functions specified as a weighted sum of position features

$$f(n) = w_1 \cdot feat_1(n) + w_2 \cdot feat_2(n) + \ldots + w_n \cdot feat_k(n)$$

Features for chess: piece count, piece placement, squares controlled, etc.

Tic-Tac-Toe

$$f(n) = [\text{\# of 3-lengths open for me}] - [\text{\# of 3-lengths open for you}]$$

where a 3-length is a complete row, column, or diagonal

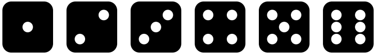
For instance, Deep Blue had >8K features in its evaluation function.

- **Adaptive horizon + iterative deepening**
- **Extended search**: retain $k > 1$ best paths (not just one)
- **Singular extension**: if a move is obviously better than the others in a node at horizon h $\rightarrow$ expand it
- **Repeated states**
- **Null-move search**: assume player forfeits move $\rightarrow$ do a shallow analysis of tree; result must surely be worse than if player had moved

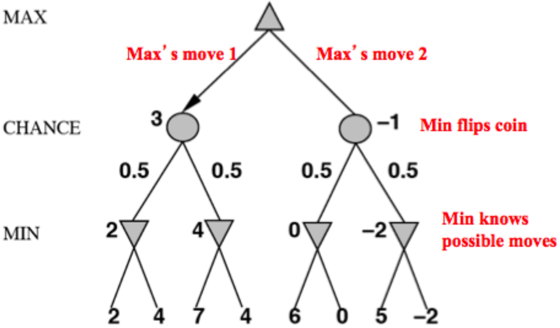
Stochastic games: motivation

When

- Unpredictable external events $\rightarrow$ unforeseen situations
- In games: unpredictability through a random element, e.g., **dice**



Stochastic games: example



Stochastic games: backgamon

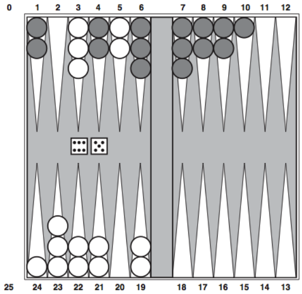


Figure 5.10 A typical backgammon position. The goal of the game is to move all one's pieces off the board. White moves clockwise toward 25, and Black moves counterclockwise toward 0. A piece can move to any position unless multiple opponent pieces are there; if there is one opponent, it is captured and must start over. In the position shown, White has rolled 6–5 and must choose among four legal moves: (5–10,5–11), (5–11,19–24), (5–10,10–16), and (5–11,11–16), where the notation (5–11,11–16) means move one piece from position 5 to 11, and then move a piece from 11 to 16.

Stochastic games: backgamon

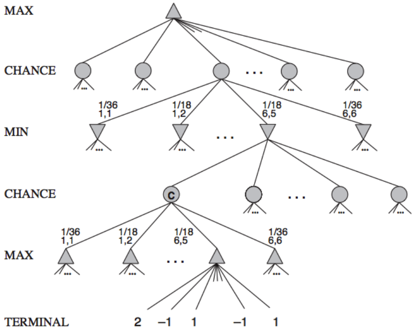


Figure 5.11 Schematic game tree for a backgammon position.

Why can't we use minimax?

- Before a player chooses her move ...

  ... she rolls dice and then knows exactly what they are
  ... and the immediate outcome of each move is also known

- But she does not know what moves her opponent will have available to choose from!

$$\textsc{Minimax}(s) = \begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$

Stochastic games: minimax

Why can't we use minimax?

- Before a player chooses her move ...
  - ... she rolls dice and then knows exactly what they are
  - ... and the immediate outcome of each move is also known
- But she does not know what moves her opponent will have available to choose from!

$$\textsc{Expectiminimax}(s) = \begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_a \textsc{Expectiminimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_a \textsc{Expectiminimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \\ \sum_r P(r)\textsc{Expectiminimax}(\textsc{Result}(s,r)) & \text{if } \textsc{Player}(s) = \textsc{chance} \end{cases}$$

[Example] card games where opponent's initial cards are unknown!

Standard approach

Calculate a probability for each possible deal

Like having one big dice roll at the beginning of the game.

- **Intuition**: compute minimax value of each action in each deal, then choose action with highest expected value over all deals
- **Special case**: if action is optimal for all deals, it's optimal!

Other Issues

Multi-player games, e.g., many card games like Hearts

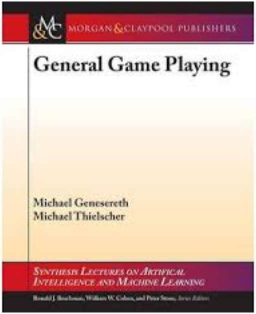
Multi-player games with alliances, like Risk

Stanford Logic Group

Idea don't develop specialised systems to play specific games (e.g., Checkers) very well

Goal design AI programs to be able to play more than one game successfully

Goal work from a description of a novel game



---

https://en.wikipedia.org/wiki/General_game_playing

General Game Playing: Deepmind

$\boxed{\text{Deep learning + Reinforcement learning}}$

Next lecture

No more theory. Congrats!