Lecture 19: Game theory in AI

EMAT31530/April 2018/Raul Santos-Rodriguez

The road so far

- Machine learning
- Search
- Bayesian networks
- MDP/RL
- Game theory

Binary classification:

$x \to \boxed{?} \to y \in \{-1, +1\}$, single action

The road so far

- Machine learning
- Search
- Bayesian networks
- MDP/RL
- Game theory

Binary classification:

$x \rightarrow \boxed{?} \rightarrow y \in \{-1, +1\}$, single action

Search problem:

$x \rightarrow \boxed{?} \rightarrow$ action sequence $(a_1, a_2, a_3, a_4, \dots)$

The road so far

- Machine learning

  Binary classification:

  $x \rightarrow \boxed{?} \rightarrow y \in \{-1, +1\}$, single action

- Search

  Search problem:

  $x \rightarrow \boxed{?} \rightarrow$ action sequence $(a_1, a_2, a_3, a_4, \ldots)$

- Bayesian networks

  Bayesian Networks:

  Model uncertainty

- MDP/RL

- Game theory

The road so far

- Machine learning

  Binary classification:

  $x \rightarrow \boxed{?} \rightarrow y \in \{-1, +1\}$, single action

- Search

  Search problem:

  $x \rightarrow \boxed{?} \rightarrow$ action sequence $(a_1, a_2, a_3, a_4, \ldots)$

- Bayesian networks

  Bayesian Networks:

  Model uncertainty

- MDP/RL

  Decision making in an uncertain world

  Search: state s and action $a \xrightarrow{\textit{deterministic}}$ state s'
  MDPs: state s and action $a \xrightarrow{\textit{randomness}}$?

- Game theory

The road so far

Machine learning	Binary classification: $x \rightarrow \boxed{?} \rightarrow y \in \{-1, +1\}$, single action
Search	Search problem: $x \rightarrow \boxed{?} \rightarrow$ action sequence $(a_1, a_2, a_3, a_4, \ldots)$
Bayesian networks	Bayesian Networks: Model uncertainty
MDP/RL	Decision making in an uncertain world Search: state s and action $a \xrightarrow{\textit{deterministic}}$ state s' MDPs: state s and action $a \xrightarrow{\textit{randomness}}$?
Game theory	Decision making in an uncertain world against other players!

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.

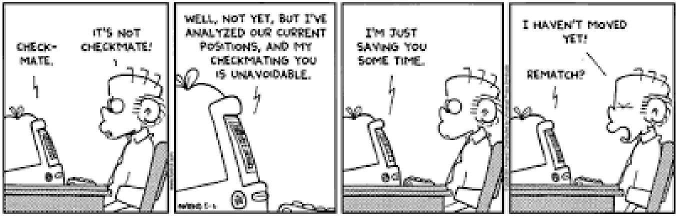
... Python: `http://www.gambit-project.org/`

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

Outline

This lecture presents the main notions of Game Theory in AI. We examine the problems that arise when we try to plan ahead in a world where someone is planning against us. We will study:

- Classical game theory
  - normal form
  - prisoner's dilemma
  - Nash equilibrium
- Game theory in AI
  - minimax
  - alpha-beta pruning


CHECK-MATE.
IT'S NOT CHECKMATE!
WELL, NOT YET, BUT I'VE ANALYZED OUR CURRENT POSITIONS, AND MY CHECKMATING YOU IS UNAVOIDABLE.
I'M JUST SAVING YOU SOME TIME.
I HAVEN'T MOVED YET!
REMATCH?

Why games?

- Problems involving hostile, adversarial, competing players and the uncertainty of interacting with the natural world
- Historical reasons: for centuries humans have used them to exert their intelligence
- Interesting, hard problems which require minimal 'initial structure'
- Clear criteria for success
- Fun!

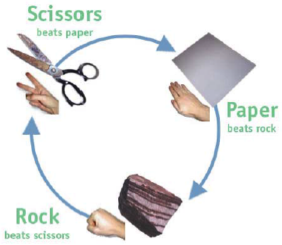
Classical game theory

Game theory (economy)

Game theory includes cooperation, chance, imperfect knowledge, simultaneous moves and tends to represent real-life decision making situations.

http://www.economist.com/node/21527025

Normal form representation



		Player 1		
		🪨	📄	✂️
Player 2	🪨	0,0	1,-1	-1,1
	📄	-1,1	0,0	1,-1
	✂️	1,-1	-1,1	0,0

Payoff matrix

The game is described by its payoff matrix. It contains the **payoffs** or **utilities** for all players and actions. (Column player's utility is listed first)

Types of games

Number of players

2-player

- Chess
- Checkers

n-player / multiplayer

- Monopoly
- Poker

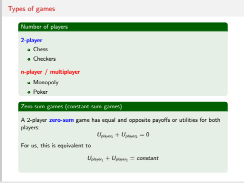
Zero-sum games (constant-sum games)

A 2-player **zero-sum** game has equal and opposite payoffs or utilities for both players:

$$U_{player_1} + U_{player_2} = 0$$

For us, this is equivalent to

$$U_{player_1} + U_{player_2} = constant$$



Types of games

Number of players

2-player
- Chess
- Checkers

n-player / multiplayer
- Monopoly
- Poker

Zero-sum games (constant-sum games)

A 2-player **zero-sum** game has equal and opposite payoffs or utilities for both players:

$$U_{player_1} + U_{player_2} = 0$$

For us, this is equivalent to

$$U_{player_1} + U_{player_2} = constant$$

A zero-sum game is (confusingly) defined as one where the total payoff to all players is the same for every instance of the game. In chess, the outcome is a win, loss, or draw, with values $+1$, 0, or $1/2$. Chess is zero-sum because every game has payoff of either $0+1$, $1+0$ or $1/2+1/2$. 'Constant-sum' would have been a better term, but zero-sum is traditional and makes sense if you imagine each player is charged an entry fee of $1/2$.

Types of games

Information

A perfect information or Markov game is fully observed

- Chess
- Checkers
- Backgammon

An imperfect information game is partially observed

- Scrabble
- Poker

Types of games

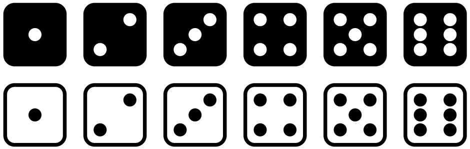
Randomness

Deterministic

- Chess
- Checkers
- Go
- Battleship

Chance

- Backgammon
- Monopoly
- Bridge
- Poker



Prisoner's dilemma

Game description

Two criminals have been arrested and the police visit them separately.

- If one player testifies against the other and the other refuses, the one who testified goes free and the one who refused gets a **10-year** sentence.
- If both players testify against each other, they each get a **5-year** sentence.
- If both refuse to testify, they each get a **1-year** sentence.

	Alice: Testify	Alice: Refuse
Bob: Testify		
Bob: Refuse		

Prisoner's dilemma

Game description

Two criminals have been arrested and the police visit them separately.

- If one player testifies against the other and the other refuses, the one who testified goes free and the one who refused gets a **10-year** sentence.
- If both players testify against each other, they each get a **5-year** sentence.
- If both refuse to testify, they each get a **1-year** sentence.

	Alice: Testify	Alice: Refuse
Bob: Testify	-5,-5	-10,0
Bob: Refuse	0,-10	-1,-1

Prisoner's dilemma

Alice's reasoning:

- Suppose Bob testifies $\rightarrow$ I get 5 years if I testify, I get 10 years if I refuse $\rightarrow$ I should testify
- Suppose Bob refuses $\rightarrow$ I go free if I testify, however I get 1 year if I refuse $\rightarrow$ I should testify

	Alice: Testify	Alice: Refuse
Bob: Testify	-5,-5	-10,0
Bob: Refuse	0,-10	-1,-1

Dominant strategy

A strategy whose outcome is better for the player regardless of the strategy chosen by the other player.

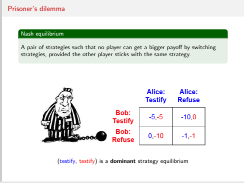
Prisoner's dilemma

Nash equilibrium

A pair of strategies such that no player can get a bigger payoff by switching strategies, provided the other player sticks with the same strategy.

	Alice: Testify	Alice: Refuse
Bob: Testify	-5,-5	-10,0
Bob: Refuse	0,-10	-1,-1

(testify, testify) is a **dominant** strategy equilibrium



Prisoner's dilemma

Nash equilibrium

A pair of strategies such that no player can get a bigger payoff by switching strategies, provided the other player sticks with the same strategy.

	Alice: Testify	**Alice: Refuse**
Bob: Testify	-5,-5	-10,0
Bob: Refuse	0,-10	-1,-1

(testify, testify) is a **dominant** strategy equilibrium

Any game with a finite set of actions has at least one Nash equilibrium. If a player has a dominant strategy, there exists a Nash equilibrium in which the player plays that strategy and the other player plays the best response to that strategy. If both players have strictly dominant strategies, there exists a Nash equilibrium in which they play those strategies.

Prisoner's dilemma

In real life

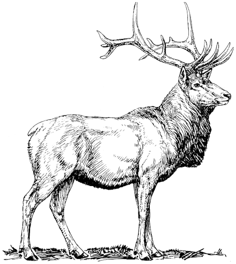
- Price war
- Arms race
- Steroid use
- Pollution control

	Cooperate	Defect
Cooperate	Win – win	Win big – lose big
Defect	Lose big – win big	Lose – lose

"TOM HANKS and MARK RYLANCE RED HOT in STEVEN SPIELBERG'S MAGNIFICENT COLD WAR THRILLER" ★★★★★
"Spielberg's craftsmanship is breathtaking"
"Mark Rylance is extraordinary"
"The tension is nail-biting"
★★★★★
TOM HANKS
BRIDGE OF SPIES
INSPIRED BY TRUE EVENTS
IN CINEMAS NOVEMBER 27

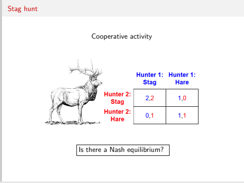
Stag hunt

Cooperative activity



	Hunter 1: Stag	Hunter 1: Hare
Hunter 2: Stag	2,2	1,0
Hunter 2: Hare	0,1	1,1

Is there a Nash equilibrium?



Cooperative activity

	Hunter 1: Stag	Hunter 1: Hare
Hunter 2: Stag	2,2	1,0
Hunter 2: Hare	0,1	1,1

$\boxed{\text{Is there a Nash equilibrium?}}$

Nash equilibrium: (stag, stag) and (hare, hare)

Stag hunt vs Prisoner’s dilemma

Prisoner’ dilemma

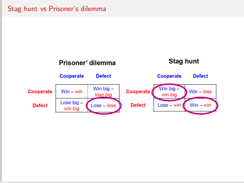
	Cooperate	**Defect**
Cooperate	Win – win	Win big – lose big
Defect	Lose big – win big	Lose – lose

Stag hunt

	Cooperate	**Defect**
Cooperate	Win big – win big	Win – lose
Defect	Lose – win	Win – win

2018-04-22

└─Stag hunt vs Prisoner's dilemma



Stag hunt vs Prisoner's dilemma

Prisoner' dilemma

	Cooperate	Defect
Cooperate	Win – win	Win big – lose big
Defect	Lose big – win big	Lose – lose

Stag hunt

	Cooperate	Defect
Cooperate	Win big – win big	Win – lose
Defect	Lose – win	Win – win

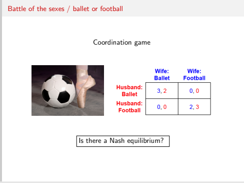
While in Prisoner's dilemma players can gain by defecting unilaterally, in stag hunt players lose by defecting unilaterally.

Battle of the sexes / ballet or football

Coordination game

	Wife: Ballet	Wife: Football
Husband: Ballet	3, 2	0, 0
Husband: Football	0, 0	2, 3

Is there a Nash equilibrium?



Coordination game

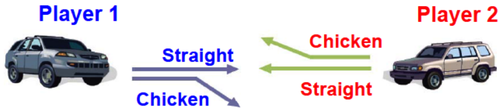
	Wife: Ballet	Wife: Football
Husband: Ballet	3, 2	0, 0
Husband: Football	0, 0	2, 3

Is there a Nash equilibrium?

Nash equilibrium: (ballet, ballet) and (football, football)

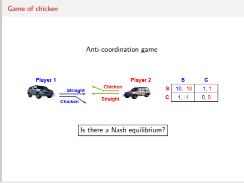
Game of chicken

Anti-coordination game



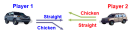
	S	C
S	-10, -10	-1, 1
C	1, -1	0, 0

$\boxed{\text{Is there a Nash equilibrium?}}$



Game of chicken

Anti-coordination game



	S	C
S	-10, -10	-1, 1
C	1, -1	0, 0

Is there a Nash equilibrium?

Nash equilibrium: (Straight, chicken) and (chicken, straight)

It is mutually beneficial for the two players to choose different strategies. This example is a good model of escalated conflict in humans and animals (hawk-dove game).

Something to think about...

The diner's dilemma

A group of people go out to eat and agree to split the bill equally. Each has a choice of ordering a cheap dish or an expensive dish (the utility of the expensive dish is higher than that of the cheap dish, but not enough for you to want to pay the difference).

$\boxed{\text{Is there a Nash equilibrium?}}$

Classical game theory: issues

Is it applicable to real life?

Humans are **not always rational**

Utilities may not always be **known**

Other assumptions made by the game-theoretic model may not hold

Political difficulties may prevent theoretically optimal mechanisms from being implemented

Could it be more applicable to AI than to real life?

Computing equilibria in complicated games is **difficult**

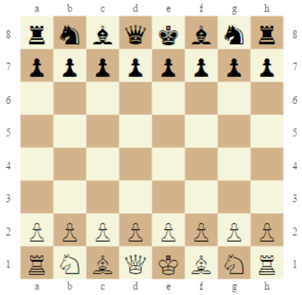
Relationship between Nash equilibrium and rational decision making is subtle

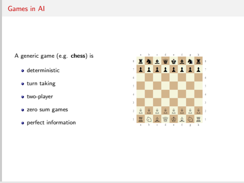
https://www.youtube.com/watch?v=S0qjK3TWZE8

Games in AI

A generic game (e.g. **chess**) is

- deterministic
- turn taking
- two-player
- zero sum games
- perfect information





Games in AI

A generic game (e.g. **chess**) is

- deterministic
- turn taking
- two-player
- zero sum games
- perfect information

In game theory (e.g., economics), any multi-agent environment (either cooperative or competitive) is a game provided that the impact of each agent on the other is significant. However, AI games are a specialised kind - deterministic, turn taking, two-player, zero sum games of perfect information. In our terminology, deterministic, fully observable environments with two players whose actions alternate and the utility values at the end of the game are always equal and opposite (+1 and -1).

A bit of history: chess

1949	Shannon paper - originated the ideas
1951	Turing paper - hand simulation
1958	Bernstein program
1960	Simon-Newell program
1961	Soviet program
1967	MacHack 6 - defeated a good player
1970	NW chess 4.5
1980	Cray Bitz
1990	Belle, Hitech, Deep Thought,
1997	Deep Blue - defeated Garry Kasparov

Setting

Similar to search

Notation

s_0: **Initial state**, initial board position

Actions(s): available **actions**, one per legal move

Result(s,a): **transition model**, which defines the result of a move

Teminal-test(s): true if *s* is a **goal state**, winning board positions

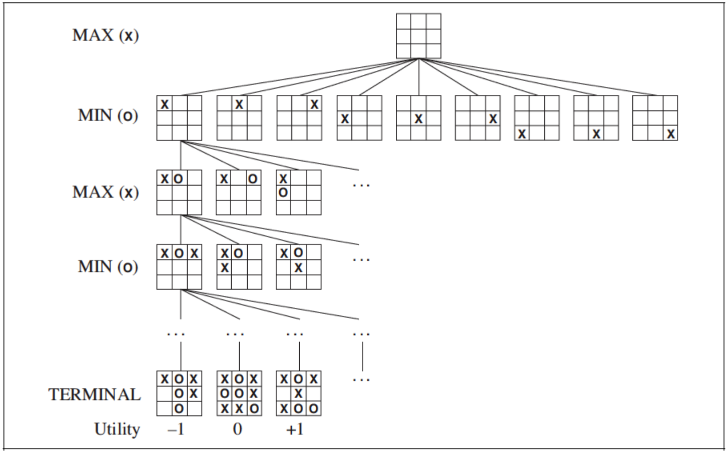
Utility(s,p): **scoring function**, assigns values to states

Game tree: encodes all possible games

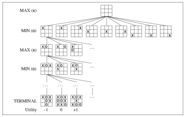
... but

1. Not looking for a path, only the next move to make: hopefully leads to a winning position
2. Our next move depends on what the other player does!

Game tree



Game tree



A (partial) game tree for the game of tic-tac-toe. We have two players, namely, MAX and MIN. The top node is the initial state, and MAX moves first, placing an X in an empty square. We show part of the tree, giving alternating moves by MIN (O) and MAX (X), until we eventually reach terminal states, which can be assigned utilities according to the rules of the game.

In a 2-person game, a solution path is unobtainable because we never know what the other player is going to do at any stage. What we need to work out is the best move. We will introduce next the minimax method, where we use the search process not to find a solution path, but to derive the most accurate evaluation of the possible moves, i.e., an evaluation which takes into account the implications that any given move will have later in the game.

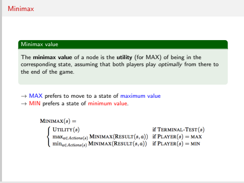
Minimax

Minimax value

The **minimax value** of a node is the **utility** (for MAX) of being in the corresponding state, assuming that both players play *optimally* from there to the end of the game.

→ MAX prefers to move to a state of maximum value
→ MIN prefers a state of minimum value.

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$



Minimax

Minimax value

The **minimax value** of a node is the **utility** (for MAX) of being in the corresponding state, assuming that both players play *optimally* from there to the end of the game.

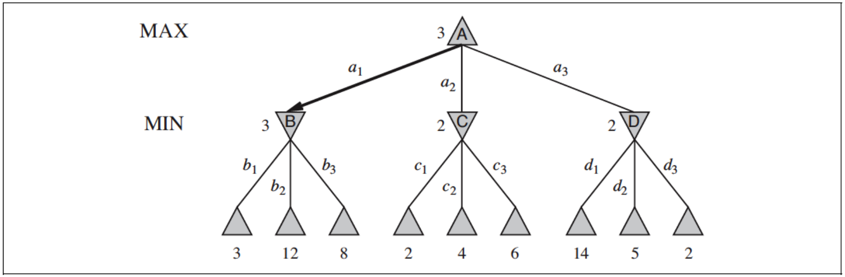
→ MAX prefers to move to a state of maximum value
→ MIN prefers a state of minimum value.

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

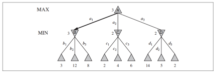
Given a game tree, the optimal strategy can be determined from the minimax value of each node, which we write as MINIMAX(s). The minimax value of a node is the utility (for MAX) of being in the corresponding state, assuming that both players play optimally from there to the end of the game. Obviously, the minimax value of a terminal state is just its utility. Furthermore, given a choice, MAX prefers to move to a state of maximum value, whereas MIN prefers a state of minimum value.
There are three elements to the minimax method.

1. Expand the search tree all the way down to a game conclusion (win, lose or draw). If this is too much search, choose a suitable cutoff.
2. Obtain an evaluation of the relevant terminal state. (e.g., positive for a win, negative for a lose and neutral for a draw). This is known as the static evaluation.
3. Then back-up the evaluations, level by level, working on the basis that when it is the opponent's turn, they will chose a transition which achieves the worst outcome from our point of view, and whenever it is our turn to move, we will choose the best.

Minimax

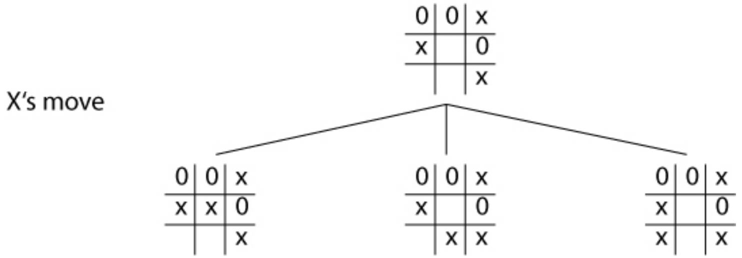


Minimax

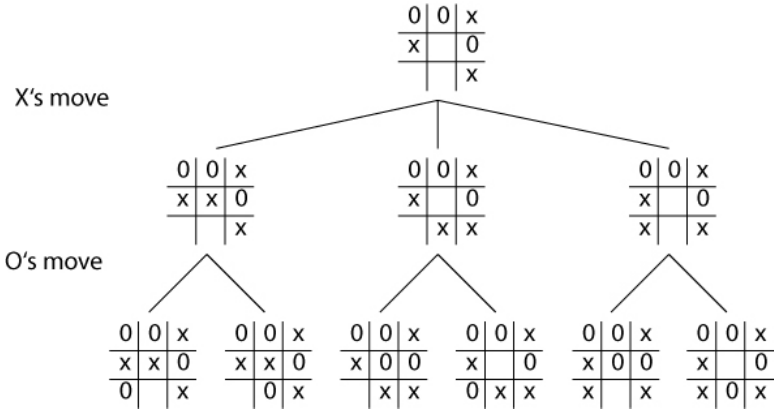


The △ nodes are 'MAX nodes', in which it is MAX's turn to move, and the ∇ nodes are 'MIN nodes'. The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is a_1, because it leads to the state with the highest minimax value, and MIN's best reply is b_1, because it leads to the state with the lowest minimax value.

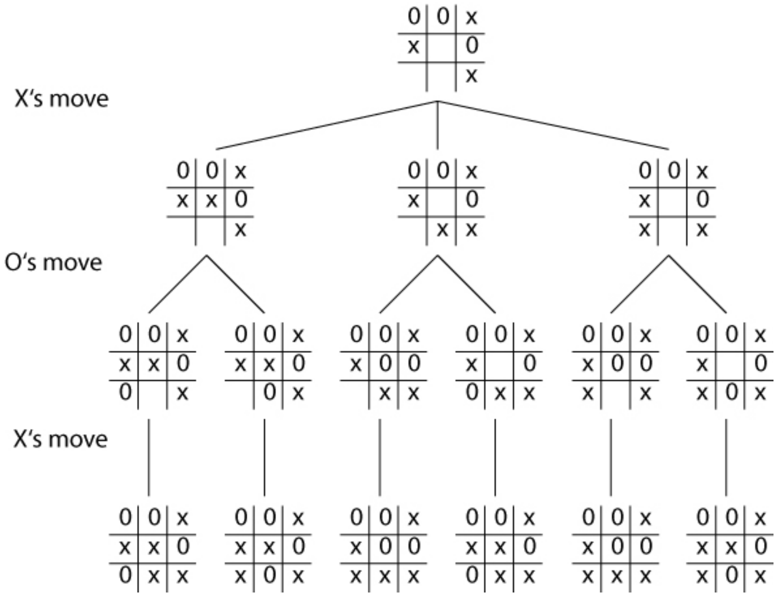
Minimax: tic-tac-toe



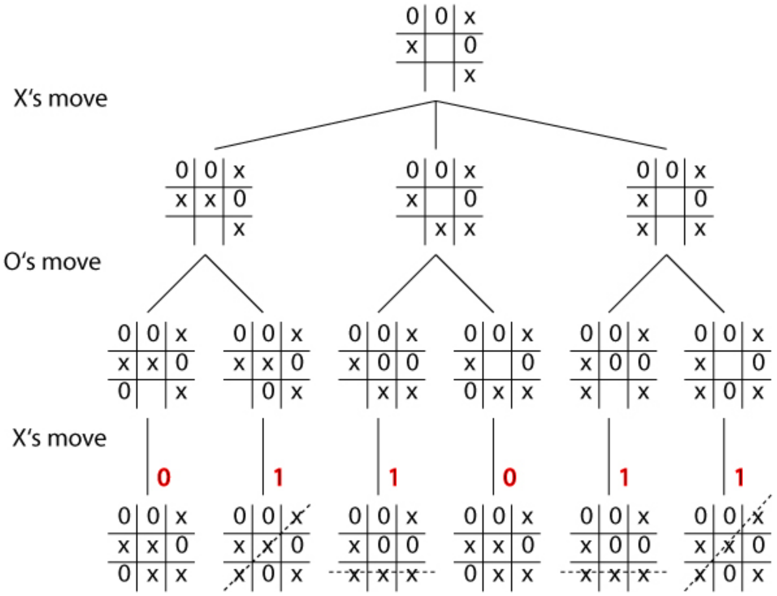
Minimax: tic-tac-toe



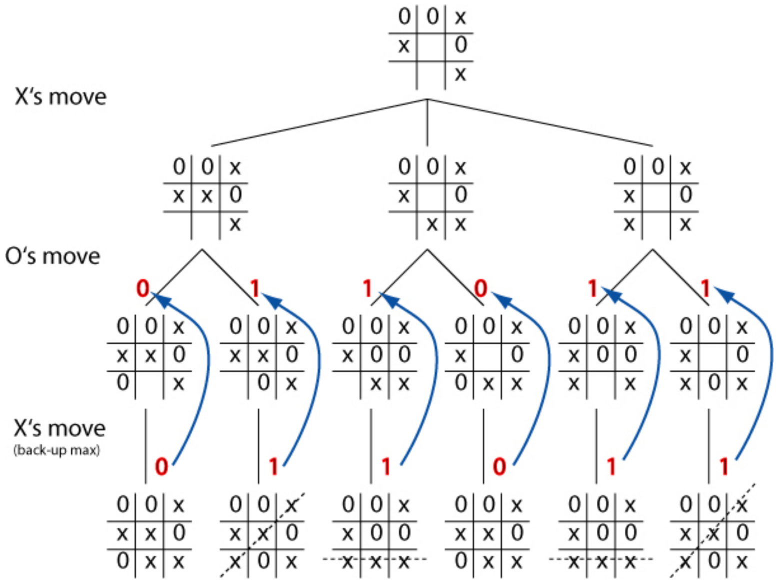
Minimax: tic-tac-toe



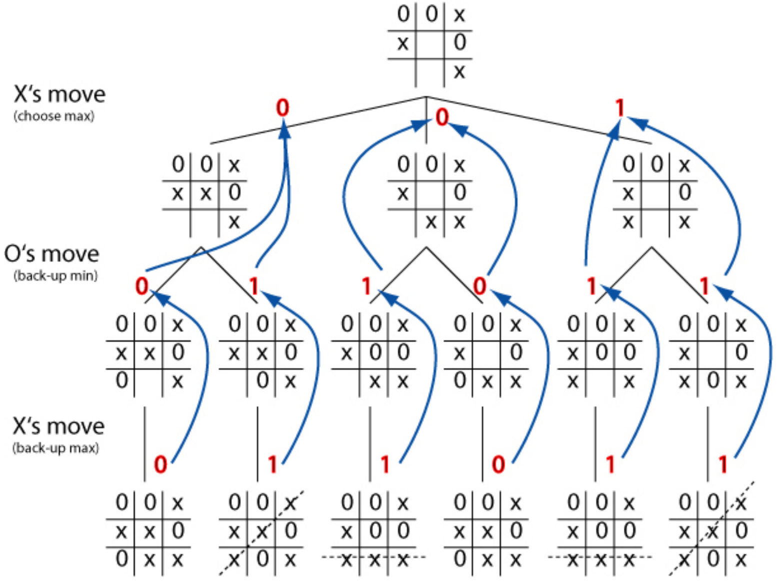
Minimax: tic-tac-toe



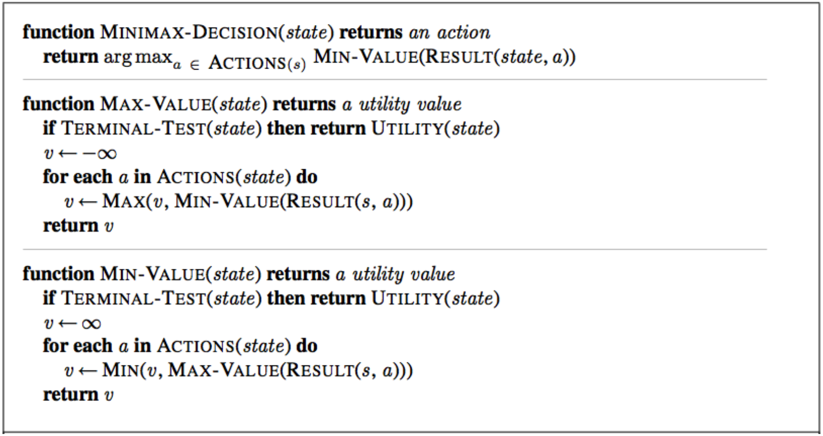
Minimax: tic-tac-toe



Minimax: tic-tac-toe



Minimax: algorithm

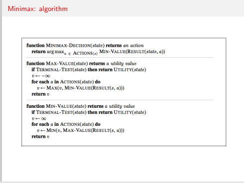


```
function MINIMAX-DECISION(state) returns an action
  return $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  $v \leftarrow -\infty$
  for each a in ACTIONS(state) do
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  $v \leftarrow \infty$
  for each a in ACTIONS(state) do
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

Minimax: algorithm



Minimax: algorithm

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimise utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state.

Minimax: complexity

Similar to depth first search!

Complexity

m: maximum depth of the three

b: number of legal moves

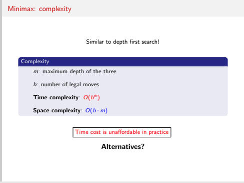
Time complexity: $O(b^m)$

Space complexity: $O(b \cdot m)$

$\boxed{\text{Time cost is unaffordable in practice}}$

Alternatives?

Minimax: complexity


Minimax: complexity

Similar to depth first search!

Complexity

m: maximum depth of the three

b: number of legal moves

Time complexity: $O(b^m)$

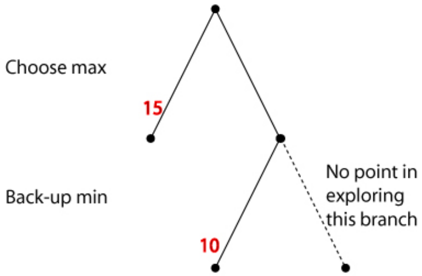
Space complexity: $O(b \cdot m)$

Time cost is unaffordable in practice

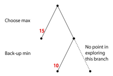
Alternatives?

The minimax algorithm performs a complete depth-first exploration of the game tree. If the maximum depth of the tree is m and there are b legal moves at each point, then the time complexity of the minimax algorithm is $O(b^m)$. The space complexity is $O(b \cdot m)$ for an algorithm that generates all actions at once, or $O(m)$ for an algorithm that generates actions one at a time. For real games, of course, the time cost is totally impractical, but this algorithm serves as the basis for the mathematical analysis of games and for more practical algorithms.When using minimax, situations can arise when search of a particular branch can be safely terminated. Applying an alpha-cutoff means we stop search of a particular branch because we see that we already have a better opportunity elsewhere. Applying a beta-cutoff means we stop search of a particular branch because we see that the opponent already has a better opportunity elsewhere. Applying both forms is alpha-beta pruning.

Alpha-cutoff

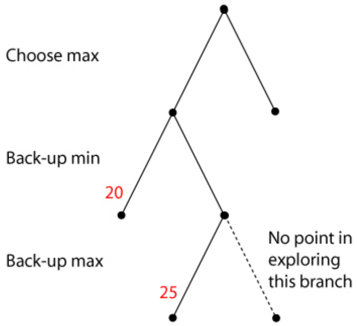


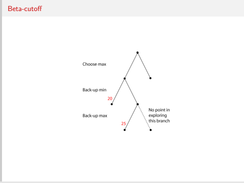
Alpha-cutoff



If, from some state S, the opponent can achieve a state with a lower value for us than one achievable in another branch. we will certainly not move the game to S. We do not need to expand S.

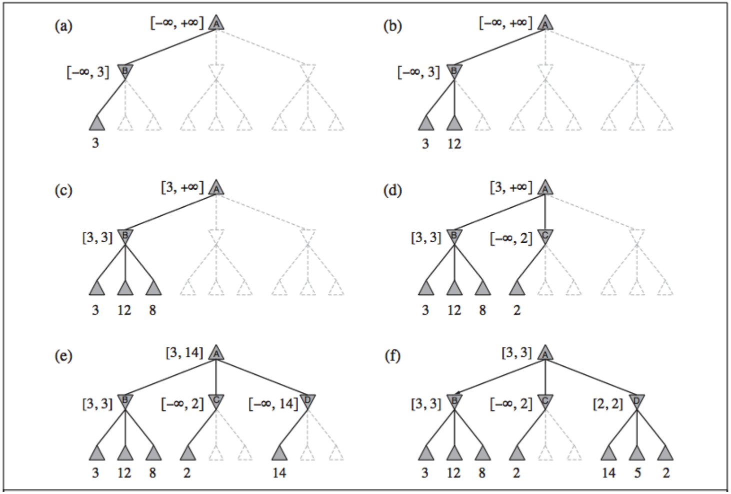
Beta-cutoff

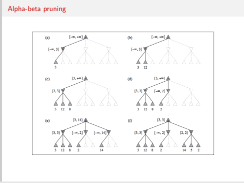

Choose max
Back-up min
20
Back-up max
25
No point in exploring this branch


Beta-cutoff

If, from some state S, we would be able to achieve a state which has a higher value for us than one the opponent can hold us to in another branch, we can assume the opponent will not choose S.

Alpha-beta pruning





Stages in the calculation of the optimal decision for the game tree in Figure 5.2. At each point, we show the range of possible values for each node.
(a) The first leaf below B has the value 3. Hence, B, which is a MIN node, has a value of at most 3.
(b) The second leaf below B has a value of 12; MIN would avoid this move, so the value of B is still at most 3.
(c) The third leaf below B has a value of 8; we have seen all B?s successor states, so the value of B is exactly 3. Now, we can infer that the value of the root is at least 3, because MAX has a choice worth 3 at the root.
(d) The first leaf below C has the value 2. Hence, C,which is a MIN node,has a value of at most 2. But we know that B is worth 3, so MAX would never choose C. Therefore, there is no point in looking at the other successor states of C. This is an example of alpha-beta pruning.
(e) The first leaf below D has the value 14, so D is worth at most 14. This is still higher than MAX's best alternative (i.e., 3), so we need to keep exploring D's successor states. Notice also that we now have bounds on all of the successors of the root, so the root's value is also at most 14.

(f) The second successor of D is worth 5, so again we need to keep exploring. The third successor is worth 2, so now D is worth exactly 2. MAX's decision at the root is to move to B, giving a value of 3.

Summary

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

Next lecture

More advanced games:

- Uncertainty and chance
- Multiplayer games
- Approximate solutions
- AI games in practice