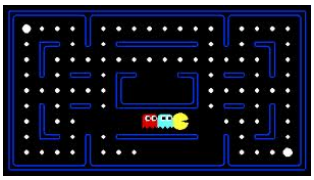
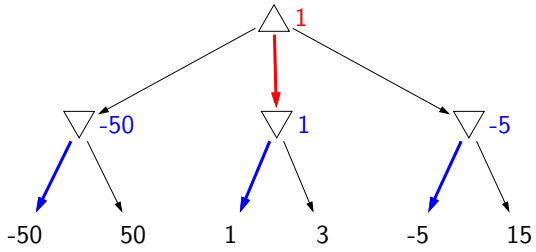
Lecture 12.1: Games III



Review: minimax in turn-based games

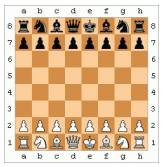
agent (max) versus opponent (min)

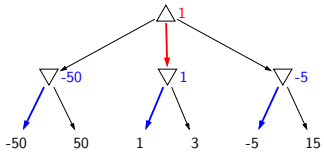


- Recall the minimax principle for turn-based zero-sum games. Since only one player moves at a time, we can apply simple recurrences to compute the minimax value of a game.

A new type of game

Turn-based games:





Simultaneous games:

?

- Game trees were our primary tool to model turn-based games. However, in simultaneous games, there is no ordering on the player's moves, so we need to develop new tools to model these games. Later, we will see that game trees will still be valuable in understanding simultaneous games.

Roadmap

Simultaneous games

Non-zero-sum games

Summary

(Modified) Two-finger Morra

Example: Modified two-finger Morra

Players A and B each show 1 or 2 fingers.
If both show 1, B gives A 1 dollar.
If both show 2, B gives A 4 dollars.
If A shows 2 and B shows 1, A gives B 2 dollars.
If A shows 1 and B shows 2, A gives B 3 dollars.

[play with a partner]

Payoff matrix

Definition: single-move simultaneous game

Players $= \{\text{A}, \text{B}\}$
Actions: possible actions
$V(a, b)$: **A's utility** if A chooses action a, B chooses b
(let V be **payoff matrix**)

Example: Modified two-finger Morra payoff matrix

B \ A	1 finger	2 fingers
1 finger	1	-2
2 fingers	-3	4

- In this lecture, we will consider only single move games. There are two players, A and B who both select from one of the available actions. The value or utility of the game is captured by a payoff matrix V whose dimensionality is |Actions| $\times$ |Actions|. We will be analyzing everything from A's perspective, so entry $V(a, b)$ is the utility that A gets if they choose action a and player B chooses b.

Strategies (policies)

Definition: pure strategy

A pure strategy is a single action:
$a \in$ Actions

Definition: mixed strategy

A mixed strategy is a probability distribution
$0 \leq \pi(a) \leq 1$ for $a \in$ Actions

Example: modified two-finger Morra strategies

Always 1: $\pi = [1, 0]$
Always 2: $\pi = [0, 1]$
Uniformly random: $\pi = [\frac{1}{2}, \frac{1}{2}]$

- Each player has a **strategy** (or a policy). A pure strategy (deterministic policy) is just a single action. Note that there's no notion of state since we are only considering single-move games.
- More generally, we will consider **mixed strategies** (randomized policy), which is a probability distribution over actions. We will represent a mixed strategy π by the vector of probabilities.

Game evaluation

Definition: game evaluation

The **value** of the game if player A follows π_{A} and player B follows π_{B} is

$$V(\pi_{\text{A}}, \pi_{\text{B}}) = \sum_{a,b} \pi_{\text{A}}(a)\pi_B(b)V(a, b)$$

Example: modified two-finger Morra

Player A always chooses 1: $\pi_{\text{A}} = [1, 0]$
Player B picks randomly: $\pi_{\text{B}} = [\frac{1}{2}, \frac{1}{2}]$
Value: $\boxed{-1}$

[whiteboard: matrix]

- Given a game (payoff matrix) and the strategies for the two players, we can define the value of the game.
- For pure strategies, the value of the game by definition is just reading out the appropriate entry from the payoff matrix.
- For mixed strategies, the value of the game (that is, the expected utility for player A) is gotten by summing over the possible actions that the players choose: $V(\pi_A, \pi_B) = \sum_{a \in \text{Actions}} \sum_{b \in \text{Actions}} \pi_A(a)\pi_B(b)V(a,b)$. We can also write this expression concisely using matrix-vector multiplications: $\pi_A^T V \pi_B$.

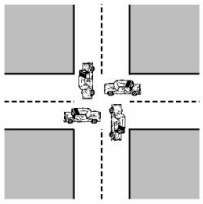
How to optimize?

Game value:

$$V(\pi_A, \pi_B)$$

Challenge: player A wants to maximize, player B wants to minimize...

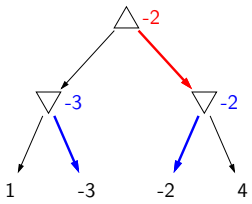
simultaneously



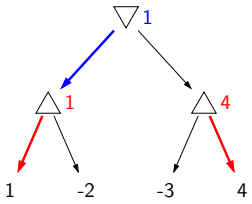
- Having established the values of fixed policies, let's try to optimize the policies themselves. Here, we run into a predicament: player A wants to maximize V but player B wants to minimize V **simultaneously**.
- Unlike turn-based games, we can't just consider one at a time. But let's consider the turn-based variant anyway to see where it leads us.

Pure strategies: who goes first?

Player A goes first:



Player B goes first:



Proposition: going second is no worse

$$\max_a \min_b V(a,b) \leq \min_b \max_a V(a,b)$$

- Let us first consider pure strategies, where each player just chooses one action. The game can be modeled by using the standard minimax game trees that we're used to.
- The main point is that if player A goes first, they get -2, but if they go second, they get 1. In general, it's at least as good to go second, and often it is strictly better. This is intuitive, because seeing what the first player does gives more information.

Mixed strategies

Example: modified two-finger Morra

Player A reveals: $\pi_A = [\frac{1}{2}, \frac{1}{2}]$

Value $V(\pi_A, \pi_B) = \pi_B(1)(-\frac{1}{2}) + \pi_B(2)(+\frac{1}{2})$

Optimal strategy for player B is $\pi_B = [1, 0]$ (**pure!**)

Proposition: second player can play pure strategy

For any fixed mixed strategy π_A:

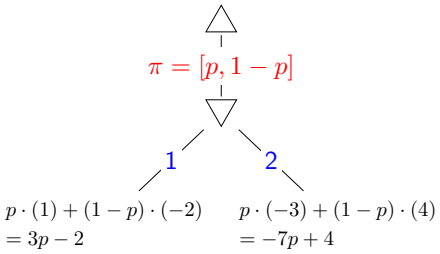
$$\min_{\pi_B} V(\pi_A, \pi_B)$$

can be attained by a pure strategy.

- Now let us consider mixed strategies. First, let's be clear on what playing a mixed strategy means. If player A chooses a mixed strategy, they reveal to player B the full probability distribution over actions, but importantly not a particular action (because that would be the same as choosing a pure strategy).
- As a warmup, suppose that player A reveals $\pi_A = [\frac{1}{2}, \frac{1}{2}]$. If we plug this strategy into the definition for the value of the game, we will find that the value is a convex combination between $\frac{1}{2}(1) + \frac{1}{2}(-2) = -\frac{1}{2}$ and $\frac{1}{2}(-3) + \frac{1}{2}(4) = \frac{1}{2}$. The value of π_B that minimizes this value is $[1, 0]$. The important part is that this is a **pure strategy**.
- It turns out that no matter what the payoff matrix V is, as soon as π_A is fixed, then the optimal choice for π_B is a pure strategy. Ths is useful because it will allow us to analyze games with mixed strategies more easily.

Mixed strategies

Player A first reveals their mixed strategy



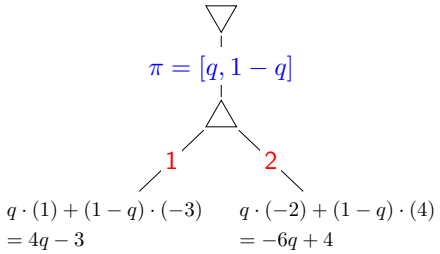
Minimax value of game:

$$\max_{0 \le p \le 1} \min\{3p - 2, -7p + 4\} = \boxed{-\frac{1}{5}} \text{ (with } p = \tfrac{3}{5})$$

- Now let us try to draw the minimax game tree where the player A first chooses a mixed strategy, and then player B chooses a pure strategy.
- There are an uncountably infinite number of mixed strategies for player A, but we can summarize all of these actions by writing a single action template $\pi = [p, 1-p]$.
- Given player A's action, we can compute the value if player B either chooses 1 or 2. For example, if player B chooses 1, then the value of the game is $3p - 2$ (with probability p, player A chooses 1 and the value is 1; with probability $1 - p$ the value is -2). If player B chooses action 2, then the value of the game is $-7p + 4$.
- The value of the min node is $F(p) = \min\{3p - 2, -7p + 4\}$. The value of the max node (and thus the minimax value of the game) is $\max_{0 \le 1 \le p} F(p)$.
- What is the best strategy for player A then? We just have to find the p that maximizes $F(p)$, which is the minimum over two linear functions of p. If we plot this function, we will see that the maximum of $F(p)$ is attained when $3p - 2 = -7p + 4$, which is when $p = \frac{3}{5}$. Plugging that value of p back in yields $F(p) = -\frac{1}{5}$, the minimax value of the game if player A goes first and is allowed to choose a mixed strategy.
- Note that if player A decides on $p = \frac{3}{5}$, it doesn't matter whether player B chooses 1 or 2; the payoff will be the same: $-\frac{1}{5}$. This also means that whatever mixed strategy (over 1 and 2) player B plays, the payoff would also be $-\frac{1}{5}$.

Mixed strategies

Player B first reveals his/her mixed strategy



Minimax value of game:

$$\min_{q \in [0,1]} \max\{4q - 3, -6q + 4\} = \boxed{-\frac{1}{5}} \text{ (with } q = \tfrac{7}{10})$$

- Now let us consider the case where player B chooses a mixed strategy $\pi = [q, 1-q]$ first. If we perform the analogous calculations, we'll find that we get that the minimax value of the game is exactly the same $(-\frac{1}{5})$!
- Recall that for pure strategies, there was a gap between going first and going second, but here, we see that for mixed strategies, there is no such gap, at least in this example.
- Here, we have been computed minimax values in the conceptually same manner as we were doing it for turn-based games. The only difference is that our actions are mixed strategies (represented by a probability distribution) rather than discrete choices. We therefore introduce a variable (e.g., p) to represent the actual distribution, and any game value that we compute below that variable is a function of p rather than a specific number.

General theorem

Theorem: minimax theorem [von Neumann, 1928]

For every simultaneous two-player zero-sum game with a finite number of actions:

$$\max_{\pi_A} \min_{\pi_B} V(\pi_A, \pi_B) = \min_{\pi_B} \max_{\pi_A} V(\pi_A, \pi_B),$$

where π_A, π_B range over **mixed strategies**.

Upshot: revealing your optimal mixed strategy doesn't hurt you!

Proof: linear programming duality

Algorithm: compute policies using linear programming

- It turns out that having no gap is not a coincidence, and is actually one of the most celebrated mathematical results: the von Neumann minimax theorem. The theorem states that for any simultaneous two-player zero-sum game with a finite set of actions (like the ones we've been considering), we can just swap the min and the max: it doesn't matter which player reveals his/her strategy first, as long as their strategy is optimal. This is significant because we were stressing out about how to analyze the game when two players play simultaneously, but now we find that both orderings of the players yield the same answer. It is important to remember that this statement is true only for mixed strategies, not for pure strategies.
- This theorem can be proved using linear programming duality, and policies can be computed also using linear programming. The sketch of the idea is as follows: recall that the optimal strategy for the second player is always deterministic, which means that the $\max_{\pi_A} \min_{\pi_B} \cdots$ turns into $\max_{\pi_A} \min_b \cdots$. The min is now over n actions, and can be rewritten as n linear constraints, yielding a linear program.
- As an aside, recall that we also had a minimax result for turn-based games, where the max and the min were over agent and opponent policies, which map states to actions. In that case, optimal policies were always deterministic because at each state, there is only one player choosing.

Summary

- Challenge: deal with simultaneous min/max moves
- Pure strategies: going second is better
- Mixed strategies: doesn't matter (von Neumann's minimax theorem)

Roadmap

Simultaneous games

Non-zero-sum games

Summary

Utility functions

Competitive games: minimax (linear programming)

Collaborative games: pure maximization (plain search)

Real life: ?

- So far, we have focused on competitive games, where the utility of one player is the exact opposite of the utility of the other. The minimax principle is the appropriate tool for modeling these scenarios.
- On the other extreme, we have collaborative games, where the two players have the same utility function. This case is less interesting, because we are just doing pure maximization (e.g., finding the largest element in the payoff matrix or performing search).
- In many practical real life scenarios, games are somewhere in between pure competition and pure collaboration. This is where things get interesting...

Prisoner's dilemma

Example: Prisoner's dilemma

Prosecutor asks A and B individually if each will testify against the other.
If both testify, then both are sentenced to 5 years in jail.
If both refuse, then both are sentenced to 1 year in jail.
If only one testifies, then that person gets out for free; the other gets a 10-year sentence.

[play with a partner]

Question

What was the outcome?

- player A testified, player B testified
- player A refused, player B testified
- player A testified, player B refused
- player A refused, player B refused

Prisoner's dilemma

Example: payoff matrix

B \ A	testify	refuse
testify	$A = -5, B = -5$	$A = -10, B = 0$
refuse	$A = 0, B = -10$	$A = -1, B = -1$

Definition: payoff matrix

Let $V_p(\pi_A, \pi_B)$ be the utility for player p.

- In the prisoner's dilemma, the players get both penalized only a little bit if they both refuse to testify, but if one of them defects, then the other will get penalized a huge amount. So in practice, what tends to happen is that both will testify and both get sentenced to 5 years, which is clearly worse than if they both had cooperated.

Nash equilibrium

Can't apply von Neumann's minimax theorem (not zero-sum), but get something weaker:

Definition: Nash equilibrium

A **Nash equilibrium** is (π_A^*, π_B^*) such that no player has an incentive to change his/her strategy:

$V_A(\pi_A^*, \pi_B^*) \geq V_A(\pi_A, \pi_B^*)$ for all π_A

$V_B(\pi_A^*, \pi_B^*) \geq V_B(\pi_A^*, \pi_B)$ for all π_B

Theorem: Nash's existence theorem [1950]

In any finite-player game with finite number of actions, there exists **at least one** Nash equilibrium.

- Since we no longer have a zero-sum game, we cannot apply the minimax theorem, but we can still get a weaker result.
- A Nash equilibrium is kind of a state point, where no player has an incentive to change his/her policy unilaterally. Another major result in game theory is Nash's existence theorem, which states that any game with a finite number of players (importantly, not necessarily zero-sum) has at least one Nash equilibrium (a stable point). It turns out that finding one is hard, but we can be sure that one exists.

Examples of Nash equilibria

Example: Modified two-finger Morra

Nash equilibrium: A plays $\pi = [\frac{3}{5}, \frac{2}{5}]$, B plays $\pi = [\frac{7}{10}, \frac{3}{10}]$.

Example: Collaborative modified two-finger Morra

Two Nash equilibria:

- A and B both play 1 (value is 1).
- A and B both play 2 (value is 4).

Example: Prisoner's dilemma

Nash equilibrium: A and B both testify.

- Here are three examples of Nash equilibria. The minimax strategies for zero-sum are also equilibria (and they are global optima).
- For purely collaborative games, the equilibria are simply the entries of the payoff matrix for which no other entry in the row or column are larger. There are often multiple local optima here.
- In the Prisoner's dilemma, the Nash equilibrium is when both players testify. This is of course not the highest possible reward, but it is stable in the sense that neither player would want to change his/her strategy. If both players had refused, then one of the players could testify to improve his/her payoff (from -1 to 0).

Summary so far

Simultaneous zero-sum games:

- von Neumann's minimax theorem
- Multiple minimax strategies, single game value

Simultaneous non-zero-sum games:

- Nash's existence theorem
- Multiple Nash equilibria, multiple game values

Huge literature in game theory / economics

- For simultaneous zero-sum games, all minimax strategies have the same game value (and thus it makes sense to talk about the value of a game). For non-zero-sum games, different Nash equilibria could have different game values (for example, consider the collaborative version of two-finger Morra).

Roadmap

Simultaneous games

Non-zero-sum games

Summary

State-of-the-art: chess

1997: IBM's Deep Blue defeated world champion Gary Kasparov

Fast computers:

- Alpha-beta search over 30 billion positions, depth 14
- Singular extensions up to depth 20

Domain knowledge:

- Evaluation function: 8000 features
- 4000 "opening book" moves, all endgames with 5 pieces
- 700,000 grandmaster games
- Null move heuristic: opponent gets to move twice

State-of-the-art: checkers

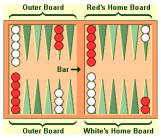
1990: Jonathan Schaeffer's **Chinook** defeated human champion; ran on standard PC

Closure:

- 2007: Checkers solved in the minimax sense (outcome is draw), but doesn't mean you can't win
- Alpha-beta search + 39 trillion endgame positions

Backgammon and Go

Alpha-beta search isn't enough...



Challenge: large branching factor

- Backgammon: randomness from dice (can't prune!)
- Go: large board size (361 positions)

Solution: learning

- For games such as checkers and chess with a manageable branching factor, one can rely heavily on minimax search along with alpha-beta pruning and a lot of computation power. A good amount of domain knowledge can be employed as to attain or surpass human-level performance.
- However, games such as Backgammon and Go require more due to the large branching factor. Backgammon does not intrinsically have a larger branching factor, but much of this branching is due to the randomness from the dice, which cannot be pruned (it doesn't make sense to talk about the most promising dice move).
- As a result, programs for these games have relied a lot on TD learning to produce good evaluation functions without searching the entire space.

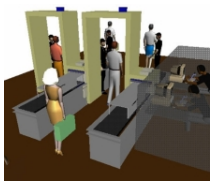
AlphaGo

- Supervised learning: on human games
- Reinforcement learning: on self-play games
- Evaluation function: convolutional neural network (value network)
- Policy: convolutional neural network (policy network)
- Monte Carlo Tree Search: search / lookahead

- The most recent visible advance in game playing was March 2016, when Google DeepMind's AlphaGo program defeated Le Sedol, one of the best professional Go players 4-1.
- AlphaGo took the best ideas from game playing and machine learning. DeepMind executed these ideas well with lots of computational resources, but these ideas should already be familiar to you.
- The learning algorithm consisted of two phases: a supervised learning phase, where a policy was trained on games played by humans (30 million positions) from the KGS Go server; and a reinforcement learning phase, where the algorithm played itself in attempt to improve, similar to what we say with Backgammon.
- The model consists of two pieces: a value network, which is used to evaluate board positions (the evaluation function); and a policy network, which predicts which move to make from any given board position (the policy). Both are based on convolutional neural networks, which we'll discuss later in the class.
- Finally, the policy network is not used directly to select a move, but rather to guide the search over possible moves in an algorithm similar to Monte Carlo Tree Search.

Other games

Security games: allocate limited resources to protect a valuable target. Used by TSA security, Coast Guard, protect wildlife against poachers, etc.

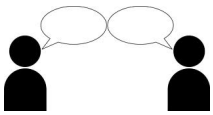


- The techniques that we've developed for game playing go far beyond recreational uses. Whenever there are multiple parties involved with conflicting interests, game theory can be employed to model the situation.
- For example, in a security game a defender needs to protect a valuable target from a malicious attacker. Game theory can be used to model these scenarios and devise optimal (randomized) strategies. Some of these techniques are used by TSA security at airports, to schedule patrol routes by the Coast Guard, and even to protect wildlife from poachers.

Other games

Resource allocation: users share a resource (e.g., network bandwidth); selfish interests leads to volunteer's dilemma

Language: people have speaking and listening strategies, mostly collaborative, applied to dialog systems



- For example, in resource allocation, we might have n people wanting to access some Internet resource. If all of them access the resource, then all of them suffer because of congestion. Suppose that if $n-1$ connect, then those people can access the resource and are happy, but the one person left out suffers. Who should volunteer to step out (this is the volunteer's dilemma)?
- Another interesting application is modeling communication. There are two players, the speaker and the listener, and the speaker's actions are to choose what words to use to convey a message. Usually, it's a collaborative game where utility is high when communication is successful and efficient. These game-theoretic techniques have been applied to building dialog systems.

Summary

- Main challenge: not just one objective
- Minimax principle: guard against adversary in turn-based games
- Simultaneous non-zero-sum games: mixed strategies, Nash equilibria
- Strategy: **search game tree + learned evaluation function**

- Games are an extraordinary rich topic of study, and we have only seen the tip of the iceberg. Beyond simultaneous non-zero-sum games, which are already complex, there are also games involving partial information (e.g., poker).
- But even if we just focus on two-player zero-sum games, things are quite interesting. To build a good game-playing agent involves integrating the two main thrusts of AI: search and learning, which are really symbiotic. We can't possibly search an exponentially large number of possible futures, which means we fall back to an evaluation function. But in order to learn an evaluation function, we need to search over enough possible futures to build an accurate model of the likely outcome of the game.