Game Theory 15-451 09/13/12

- Zero-sum games
- General-sum games

Game Theory

(and its connections to Algorithm Analysis and Computer Science)

Plan for Today

- 2-Player Zero-Sum Games (matrix games)
  - Minimax optimal strategies
  - Connection to randomized algorithms
  - Minimax theorem and proof
- General-Sum Games (bimatrix games)
  - notion of Nash Equilibrium
- Proof of existence of Nash Equilibria
  - using Brouwer's fixed-point theorem

2-player zero-sum games (aka matrix games)

Consider the following scenario…

- Shooter has a penalty shot. Can choose to shoot left or shoot right.
- Goalie can choose to dive left or dive right.
- If goalie guesses correctly, (s)he saves the day. If not, it's a goooooaaaaall!
- Vice-versa for shooter.

2-Player Zero-Sum games

- Two players R and C. Zero-sum means that what's good for one is bad for the other.
- Game defined by matrix with a row for each of R's options and a column for each of C's options. Matrix tells who wins how much.
  - an entry (x,y) means: x = payoff to row player, y = payoff to column player. "Zero sum" means that y = -x.
- E.g., penalty shot:

shooter \ goalie	Left	Right
Left	(0,0)	(1,-1)
Right	(1,-1)	(0,0)

goalie

shooter

Goooaaaal!

No goal

Game Theory terminolgy

- Rows and columns are called pure strategies.
- Randomized algs called mixed strategies.
- "Zero sum" means that game is purely competitive. (x,y) satisfies x+y=0. (Game doesn't have to be fair).

shooter \ goalie	Left	Right
Left	(0,0)	(1,-1)
Right	(1,-1)	(0,0)

goalie

shooter

GOAALLL!!!

No goal

Game Theory terminolgy

- Often describe in terms of 2 matrices R and C, where for zero-sum games we have C = -R.

(I am putting them into a single matrix where each entry is a pair, because it is easier visually).

goalie

	Left	Right
Left	(0,0)	(1,-1)
Right	(1,-1)	(0,0)

shooter

GOAALLL!!!

No goal

Minimax-optimal strategies

- Minimax optimal strategy is a (randomized) strategy that has the best guarantee on its expected gain, over choices of the opponent. [maximizes the minimum]
- I.e., the thing to play if your opponent knows you well.

goalie

	Left	Right
Left	(0,0)	(1,-1)
Right	(1,-1)	(0,0)

shooter

GOAALLL!!!

No goal

Minimax-optimal strategies

- What are the minimax optimal strategies for this game?

Minimax optimal strategy for both players is 50/50. Gives expected gain of ½ for shooter (-½ for goalie). Any other is worse.

goalie

	Left	Right
Left	(0,0)	(1,-1)
Right	(1,-1)	(0,0)

shooter

GOAALLL!!!

No goal

Minimax-optimal strategies

- How about penalty shot with goalie who's weaker on the left?

Minimax optimal for shooter is (2/3,1/3).
Guarantees expected gain at least 2/3.
Minimax optimal for goalie is also (2/3,1/3).
Guarantees expected loss at most 2/3.

goalie

	Left	Right
Left	($\frac{1}{2}$,-$\frac{1}{2}$)	(1,-1)
Right	(1,-1)	(0,0)

shooter

GOAALLL!!!

50/50

Minimax-optimal strategies

- How about if shooter is less accurate on the left too?

Minimax optimal for shooter is (4/5,1/5).
Guarantees expected gain at least 3/5.
Minimax optimal for goalie is (3/5,2/5).
Guarantees expected loss at most 3/5.

goalie

	Left	Right
Left	($\frac{1}{2}$,-$\frac{1}{2}$)	($\frac{3}{4}$,-$\frac{3}{4}$)
Right	(1,-1)	(0,0)

shooter

Minimax Theorem (von Neumann 1928)

- Every 2-player zero-sum game has a unique value V.
- Minimax optimal strategy for R guarantees R's expected gain at least V.
- Minimax optimal strategy for C guarantees C's expected loss at most V.

Counterintuitive: Means it doesn't hurt to publish your strategy if both players are optimal. (Borel had proved for symmetric 5x5 but thought was false for larger games)

We will see one proof in a bit…

Matrix games and Algorithms

- Gives a useful way of thinking about guarantees on algorithms for a given problem.
- Think of rows as different algorithms, columns as different possible inputs. (E.g., sorting)
- M(i,j) = cost of algorithm i on input j.
- Algorithm design goal: good strategy for row player. Lower bound: good strategy for adversary.

One way to think of upper-bounds/lower-bounds: on value of this game

Matrix games and Algorithms

- Gives a useful way of thinking about guarantees on algorithms for a given problem.
- Think of rows as different algorithms, columns as different possible inputs. (E.g., sorting)
- M(i,j) = cost of algorithm i on input j.
- Algorithm design goal: good strategy for row player. Lower bound: good strategy for adversary.

Of course matrix may be HUGE. But helpful conceptually.

Matrix games and Algs

Adversary

Alg player

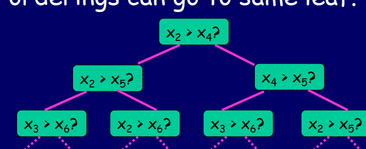
- What is a deterministic alg with a good worst-case guarantee?
  - A row that does well against all columns.
- What is a lower bound for deterministic algorithms?
  - Showing that for each row i there exists a column j such that the cost M(i,j) is high.
- How to give lower bound for randomized algs?
  - Give randomized strategy (ideally minimax optimal) for adversary that is bad for all i. Must also be bad for all distributions over i. Sometimes called Yao's principle.

Lower bounds for randomized sorting

Adversary

Alg player

- Adversary strategy: uniform random permutation of {1,2,…,n}
- Any deterministic algorithm can be viewed as a decision tree with n! leaves. No two input orderings can go to same leaf.



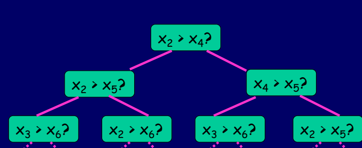
How deep is random leaf in tree with n! leaves?

Lower bounds for randomized sorting

Adversary

Alg player

- Q: How many leaves at depth < lg(n!)-10?
- A: At most 1+2+4+…+n!/1024 < n!/512.
- So, over 99% of leaves at depth $\geq$ lg(n!)-10, so average depth is $\Omega(\lg(n!)) = \Omega(n \log n)$.



How deep is random leaf in tree with n! leaves?

E.g., hashing

Adversary

Alg player

- Rows are different hash functions.
- Cols are different sets of n items to hash.
- M(i,j) = #collisions incurred by alg i on set j.

We will see:

- For any row, can reverse-engineer a bad column (if universe of keys is large enough).
- Universal hashing: a randomized strategy for row player that has good behavior for every column.
  - For any sequence of operations, if you randomly construct hash function in this way, you won't get many collisions in expectation.

Minimax-optimal strategies

- In small games we can solve by considering a few cases (equality or dominant strategy).
- Later, we will see how to solve for minimax optimal in NxN games using Linear Programming.
  - poly time in size of matrix if use poly-time LP alg.
- Of course for probs like sorting, N is huge…

	Left	Right
Left	$(\frac{1}{2},-\frac{1}{2})$	(1,-1)
Right	(1,-1)	(0,0)

General-Sum Games

- Zero-sum games are good formalism for design/analysis of algorithms.
- General-sum games are good models for systems with many participants whose behavior affects each other's interests
  - E.g., routing on the internet
  - E.g., online auctions

General-sum games

- In general-sum games, can get win-win and lose-lose situations.
- E.g., "what side of ~~sidewalk to walk on~~ street to drive on?":

	Left	Right
Left	(1,1)	(-1,-1)
Right	(-1,-1)	(1,1)

you

person walking towards you

General-sum games

- In general-sum games, can get win-win and lose-lose situations.
- E.g., "which movie should we go to?":

	Muppets	Twilight
Muppets	(8,2)	(0,0)
Twilight	(0,0)	(2,8)

No longer a unique "value" to the game.

Nash Equilibrium

- A Nash Equilibrium is a stable pair of strategies (could be randomized).
- Stable means that neither player has incentive to deviate on their own.
- E.g., "what side of sidewalk to walk on":

	Left	Right
Left	(1,1)	(-1,-1)
Right	(-1,-1)	(1,1)

NE are: both left, both right, or both 50/50.

Uses

- Economists use games and equilibria as models of interaction.
- E.g., pollution / prisoner's dilemma:
  - (imagine pollution controls cost \$4 but improve everyone's environment by \$3)

	don't pollute	pollute
don't pollute	(2,2)	(-1,3)
pollute	(3,-1)	(0,0)

Need to add extra incentives to get good overall behavior.

Existence of NE

- Nash (1950) proved: any general-sum game must have at least one such equilibrium.
  - Might require using randomization as in minmax.
- This also yields minimax thm as a corollary.
  - Pick some NE and let V = value to row player in that equilibrium.
  - Since it's a NE, neither player can do better even knowing the (randomized) strategy their opponent is playing.
  - So, they're each playing minimax optimal.

Existence of NE

- Proof will be non-constructive.
- Unlike case of zero-sum games, we **do not know any** polynomial-time algorithm for finding Nash Equilibria in n $\times$ n general-sum games. [known to be "PPAD-hard"]
- Notation:
  - Assume an nxn matrix.
  - Use $(p_1,\ldots,p_n)$ to denote mixed strategy for row player, and $(q_1,\ldots,q_n)$ to denote mixed strategy for column player.

Proof

- We'll start with Brouwer's fixed point theorem.
  - Let S be a compact convex region in R^n and let $f:S \rightarrow S$ be a continuous function.
  - Then there must exist $x \in S$ such that $f(x)=x$.
  - x is called a "fixed point" of f.
- Simple case: S is the interval [0,1].
- We will care about:
  - S = {(p,q): p,q are legal probability distributions on 1,...,n}. I.e., S = $\text{simplex}_n \times \text{simplex}_n$

Proof (cont)

- S = {(p,q): p,q are mixed strategies}.
- Want to define f(p,q) = (p',q') such that:
  - f is continuous. This means that changing p or q a little bit shouldn't cause p' or q' to change a lot.
  - Any fixed point of f is a Nash Equilibrium.
- Then Brouwer will imply existence of NE.

Try #1

- What about f(p,q) = (p',q') where p' is best response to q, and q' is best response to p?
- Problem: not necessarily well-defined:
  - E.g., penalty shot: if p = (0.5,0.5) then q' could be anything.

	Left	Right
Left	(0,0)	(1,-1)
Right	(1,-1)	(0,0)

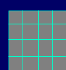
Try #1

- What about f(p,q) = (p',q') where p' is best response to q, and q' is best response to p?
- Problem: also not continuous:
  - E.g., if p = (0.51, 0.49) then q' = (1,0). If p = (0.49,0.51) then q' = (0,1).

	Left	Right
Left	(0,0)	(1,-1)
Right	(1,-1)	(0,0)

Instead we will use…

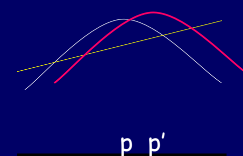
- f(p,q) = (p',q') such that:
  - q' maximizes [(expected gain wrt p) - $||q-q'||^2$]
  - p' maximizes [(expected gain wrt q) - $||p-p'||^2$]



Fixing q, any given row i has some expected gain a_i.
So, payoff for some p' is $a_1p'_1 + a_2p'_2 + \ldots + a_np'_n$.
Key point: this is a linear function of p', to which we then add a quadratic penalty.

Instead we will use…

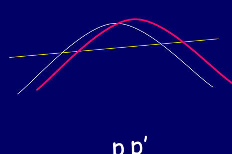
- f(p,q) = (p',q') such that:
  - q' maximizes [(expected gain wrt p) - $||q-q'||^2$]
  - p' maximizes [(expected gain wrt q) - $||p-p'||^2$]



Note: quadratic + linear = quadratic.

Instead we will use…

- f(p,q) = (p',q') such that:
  - q' maximizes [(expected gain wrt p) - $||q-q'||^2$]
  - p' maximizes [(expected gain wrt q) - $||p-p'||^2$]



Note: quadratic + linear = quadratic.

Instead we will use…

- f(p,q) = (p',q') such that:
  - q' maximizes [(expected gain wrt p) - $||q-q'||^2$]
  - p' maximizes [(expected gain wrt q) - $||p-p'||^2$]
- f is well-defined and continuous since quadratic has unique maximum and small change to p,q only moves this a little.
- Also fixed point = NE. (even if tiny incentive to move, will move little bit)
- So, apply Brower and that's it!