

15-451 Algorithms, Spring 2007

Lectures 27-29

Author: Avrim Blum
Instructor: Manuel Blum

April 24, 2007

Game Theory 15-451 12/05/06
 - Zero-sum games
 - General-sum games

shall we play a game?

Game Theory and Computer Science

Plan for Today

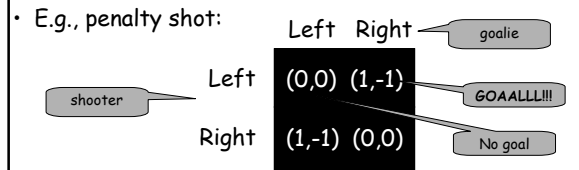
- 2-Player Zero-Sum Games (matrix games)
 - Minimax optimal strategies
 - Minimax theorem test material
and proof not test material
- General-Sum Games (bimatrix games)
 - notion of Nash Equilibrium
- Proof of existence of Nash Equilibria
 - using Brouwer's fixed-point theorem

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 gooooooaaaaaall!
- 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 $y = -x$.



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.



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.
- In class on Linear Programming, we saw how to solve for this using LP.
 - polynomial time in size of matrix if use poly-time LP alg.

Minimax-optimal strategies

- E.g., penalty shot:

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

Minimax optimal strategy for both players is 50/50. Gives expected gain of $\frac{1}{2}$ for shooter ($-\frac{1}{2}$ for goalie). Any other is worse.

Minimax-optimal strategies

- E.g., penalty shot with goalie who's weaker on the left.

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

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 $2/3$.

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 5×5 but thought was false for larger games)

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

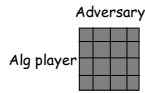
- Gives a useful way of thinking about guarantees on algorithms for a given problem.
- Think of rows as different algorithms, columns as different possible inputs. E.g., sorting
- $M(i,j)$ = cost of algorithm i on input j .
- Algorithm design goal: good strategy for row player. Lower bound: good strategy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

Of course matrix may be HUGE. But helpful conceptually.

Matrix games and Algs

- What is a deterministic alg with a good worst-case guarantee? Adversary
Alg player 
 - 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 $M(i,j)$ is bad.
- How to give lower bound for randomized algs?
 - Give randomized strategy for adversary that is bad for all i . Must also be bad for all distributions over i .

E.g., hashing



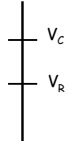
- Rows are different hash functions.
- Cols are different sets of n items to hash.
- $M(i,j)$ = #collisions incurred by alg i on set j .

We saw:

- For any row, can reverse-engineer a bad column.
- Universal hashing is a randomized strategy for row player that has good behavior for every column.
 - For any set of inputs, if you randomly construct hash function in this way, you won't get many collisions in expectation.

Nice proof of minimax thm (sketch)

- Suppose for contradiction it was false.
- This means some game G has $V_C > V_R$:
 - If Column player commits first, there exists a row that gets at least V_C .
 - But if Row player has to commit first, the Column player can make him get only V_R .
- Scale matrix so payoffs to row are in $[0,1]$. Say $V_R = V_C - \delta$.



Proof sketch, contd

- Consider exponential weighting alg from Nov16 lecture as Row, against opponent who always plays best response to Row's distrib.
- In T steps,
 - Alg gets $\geq (1-\epsilon/2)OPT - \log(n)/\epsilon$ [use $\epsilon=\delta$]
 - $OPT \geq T \cdot V_C$ [Best against opponent's empirical distribution]
 - Alg $\leq T \cdot V_R$ [Each time, opponent knows your randomized strategy]
 - Gap is δT . Contradicts assumption once $\delta T > (\epsilon/2)OPT + \log(n)/\epsilon$.

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 road to drive on?":

		Left	Right	
	you	(1,1)	(-1,-1)	person driving towards you
		(-1,-1)	(1,1)	

General-sum games

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

		Borat	Happy-feet
	Borat	(8,2)	(0,0)
	Happy-feet	(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 road to drive on":

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

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

Nash Equilibrium

- A Nash Equilibrium is a stable pair of strategies (could be randomized).
- Stable means that neither player has incentive to deviate.
- E.g., "which movie to go to":

	Borat	Happy-feet
Borat	(8,2)	(0,0)
Happy-feet	(0,0)	(2,8)

NE are: both B, both HF, or (80/20,20/80)

Uses

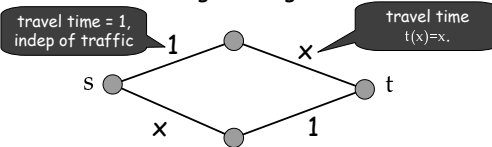
- Economists use games and equilibria as models of interaction.
- E.g., pollution / prisoner's dilemma:
 - (imagine pollution controls cost \$4 but improve everyone's environment by \$3)

	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.

NE can do strange things

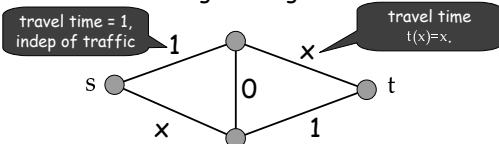
- Braess paradox:
 - Road network, traffic going from s to t.
 - travel time as function of fraction x of traffic on a given edge.



Fine. NE is 50/50. Travel time = 1.5

NE can do strange things

- Braess paradox:
 - Road network, traffic going from s to t.
 - travel time as function of fraction x of traffic on a given edge.



Add new superhighway. NE: everyone uses zig-zag path. Travel time = 2.

Existence of NE

- Nash (1950) proved: any general-sum game must have at least one such equilibrium.
 - Might require randomized strategies (called "mixed strategies")
- 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 $n \times n$ matrix.
 - Use (p_1, \dots, p_n) to denote mixed strategy for row player, and (q_1, \dots, 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 \mathbb{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:
 - $S = \{(p,q): p,q \text{ are legal probability distributions on } 1, \dots, n\}$. I.e., $S = \text{simplex}_n \times \text{simplex}_n$

Proof (cont)

- $S = \{(p,q): p,q \text{ are mixed strategies}\}$.
- 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)

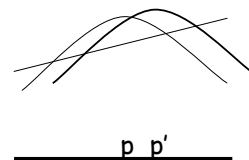
Try #1

- 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...

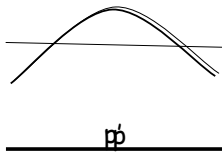
- $f(p,q) = (p',q')$ such that:
 - q' maximizes [(expected gain wrt p) - $\|q-q'\|^2$]
 - p' maximizes [(expected gain wrt q) - $\|p-p'\|^2$]



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, that's it!

Machine Learning: Learning finite state environments



Avrim Blum
15-451 lecture 12/07/06

Machine Learning

A big topic in Computer Science. We'd like programs that learn with experience.

- Because it's hard to program up complicated things by hand.
- Want software that personalizes itself to users needs.
- Because it's a necessary part of anything that is going to be really intelligent.

What ML can do

- Learn to steer a car.  Pomerleau NHAA
- Learn to read handwriting, recognize speech, detect faces.  Schneiderman Kanade
- Learn to play backgammon (best in world).
- Identify patterns in databases.

Generally, program structure developed by hand. Learning used to set (lots of) parameters. ML as programmer's assistant.

More conceptually...

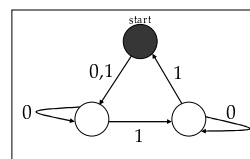
- Can we use CS perspective to help us understand what learning is?
 - Think about learning as a computational task just like multiplying?
 - How does a baby learn to figure out its environment? To figure out the effect of its actions?
- Lots of parts to all this. Today: one problem that captures some small piece of it.

Imagine...

- Say we are a baby trying to figure out the effects our actions have on our environment...
- Sometimes actions have effects we can notice right away, sometimes effects are more long-term.

A model: learning a finite state environment

- Let's model the world as a DFA. We perform actions, we get observations.
- Our actions can also change the state of the world. # states is finite.

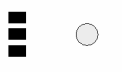


← Actions 0 and 1.
Observations white or purple.

Learning a DFA

Another way to put it:

- We have a box with buttons and lights.



- Can press the buttons, observe the lights.
 - $lights = f(current\ state)$
 - $next\ state = g(button, prev\ state)$
- **Goal: learn predictive model of device.**

Learning DFAs



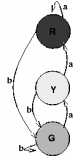
This seems really hard. Can't tell for sure when world state has changed.

Let's look at an easier problem first: state = observation.



An example w/o hidden state

2 actions: a, b.



Generic algorithm for lights=state:

- Build a model.
- While not done, find an unexplored edge and take it.

Now, let's try the harder problem!

Some examples

Example #1 (3 states)

Example #2 (3 states)

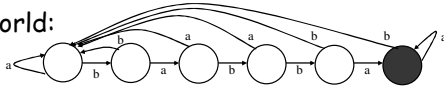
Can we design a procedure to do this in general?

One problem: what if we always see the same thing? How do we know there isn't something else out there?

Our model:



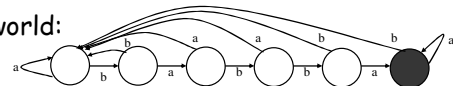
Real world:



Called "combination-lock automaton"

Can we design a procedure to do this in general?

Real world:



Called "combination-lock automaton"

This is a serious problem. It means we can't hope to efficiently come up with an exact model of the world from just our own experimentation.

How to get around this?

- Assume we can propose model and get counterexample.
- Alternatively, goal is to be predictive. Any time we make a mistake, we think and perform experiments.
- Goal is not to have to do this too many times. For our algorithm, total # mistakes will be at most # states.

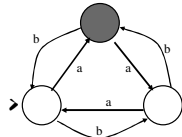
Today: a really cool algorithm by Dana Angluin

(with extensions by R.Rivest & R.Schapire)

- To simplify things, let's assume we have a RESET button.
- If time, we'll see how to get rid of that.

The problem (recap)

- We have a DFA:

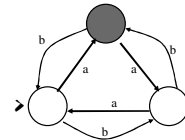


- $observation = f(current\ state)$
- $next\ state = g(button, prev\ state)$
- Can feed in sequence of actions, get observations. Then resets to start.
- Can also propose/field-test model. Get counterexample.

Key Idea

Key idea is to represent the DFA using a state/experiment table.

		experiments	
		λ	a
states	λ	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	a	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	b	<input type="checkbox"/>	<input type="checkbox"/>
transitions	aa	<input type="checkbox"/>	<input type="checkbox"/>
	ab	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	ba	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	bb	<input checked="" type="checkbox"/>	<input type="checkbox"/>



Either $aa=b$ or else aa is a totally new state and we need another expt to distinguish.

Key Idea

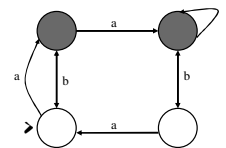
Key idea is to represent the DFA using a state/experiment table.

		experiments	
		λ	a
states	λ	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	a	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	b	<input type="checkbox"/>	<input type="checkbox"/>
transitions	aa	<input type="checkbox"/>	<input type="checkbox"/>
	ab	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	ba	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	bb	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Guarantee will be: either model is correct, or else the world has $> n$ states. In that case, need way of using counterexs to add new state to model.

The algorithm

We'll do it by example...



Algorithm (formally)

Begin with $S = \{\lambda\}, E = \{\lambda\}$.

1. Fill in transitions to make a hypothesis FSM.
2. While exists $s \in SA$ such that no $s' \in S$ has $row(s') = row(s)$, add s into S , and go to 1.
3. Query for counterexample z .
4. Consider all splits of z into (p_i, s_i) , and replace p_i with its predicted equivalent $\alpha_i \in S$.
5. Find $\alpha_i r_i$ and $\alpha_{i+1} r_{i+1}$ that produce different observations.
6. Add r_{i+1} as a new experiment into E . go to 1.

Summary / Related problems

- All states look distinct: easy.
- Not all look distinct:
 - can do with counterex.
- All distinct but probabilistic transitions?
 - Markov Decision Process(MDP) / Reinforcement Learning.
 - Usual goal: maximize discounted reward (like probabilistic shortest path). DP-based algs.
- Not all distinct & probabilistic transitions?
 - POMDP. hard.

Today: Fair Cake Cutting
Envy-Free Cake Cutting.

DEFINITION: A PROTOCOL is a set of instructions for n participants. A protocol is like an algorithm EXCEPT that the participants may choose to not follow the protocol. Certain guarantees are made to those who follow the protocol. No guarantees are made to those who don't.

DEFINITION: A CAKE CUTTING PROTOCOL is a protocol for n participants to divide a cake among themselves.

ASSUMPTIONS:

- * each person has a personal value system for assigning a real positive value to each piece of cake.
 - * each person's value system gives value 1 to the entire cake.
 - * value is a measure: fix a person; if a cake is divided into a finite number of pieces, then for that person, each piece has positive value, and the sum of the values of all the pieces of cake equals 1.
 - * for any real number x , $0 < x < 1$, a person can divide any piece of cake having value v , say, into 2 pieces having values xv and $(1-x)v$ respectively.
- Each person has a steady hand and can cut a cake exactly as he chooses to cut it.
- * cake is not necessarily uniform: it may have chocolate, vanilla, frosting, and fruit. Some parts of the cake (eg chocolate) may be more valuable to one person than another.
 - * each person follows protocol except possibly in the matter of declaring honestly the size or value they assign each piece. For example, if the protocol calls for Alice to cut the cake into 3 equal pieces, she will indeed divide it into 3 pieces, not 2 or 4. However, the pieces may not have equal value to her (in which case she is not following protocol).

DEFINITION: A cake cutting protocol is FAIR if each participant who follows protocol is guaranteed to receive at least $1/n$ of the cake according to his own personal value system.

DEFINITION: A cake cutting protocol is ENVY-FREE if each participant who follows protocol is guaranteed to receive at least as much cake by his measure as any other participant.

For example, the standard 2-person protocol is envy-free:

Alice: cut cake into two equal pieces
Bob: choose the piece of greater value
If Alice follows protocol, she will divide the cake into two pieces of equal value to her. Suppose it turns out that the two pieces are worth $1/3$ and $2/3$ to Bob.
If Bob follows protocol, he will get the piece worth $2/3$ by his measure. If he does not follow protocol, he may or may not get the bigger piece (by his measure).

Notice that ENVY-FREE \Rightarrow FAIR (why?) but not the other way around (why not?) .

An Envy-free Cake Cutting protocol for $n=3$
by Selfridge and Conway

STAGE 0.

- 0.1. Alice is to cut the cake into three equal pieces [by her measure].
- 0.2. Bob is to trim the largest piece to create a 2-way tie for largest [by his measure], and set the trimmings aside.

Note that no one has yet gotten a piece of cake.
THIS ENDS STAGE 0.

STAGE 1.

Trimmings aside, there are now 3 "pieces," which together we call cake1, and the "trimmings," which we call cake2.

Cake1 is distributed in reverse order (Charlie, Bob, Alice) as follows:

- 1.1 Charlie picks [what he considers to be] the largest piece.
- 1.2. Bob takes the trimmed piece if Charlie has not taken it; otherwise Bob takes [what he considers to be] the largest of the two pieces.
- 1.3. Alice takes the last remaining (untrimmed!) piece of cake1.
THIS ENDS STAGE 1.

As you may have noticed, this protocol has one gal, Alice, and two guys, Bob and Charlie.
This helps to clarify the next and final stage 2:
Call the guy who took the trimmed piece T,
and the other guy NT.

STAGE 2.

Cake2 (the trimmings) is distributed as follows:

- 2.1. NT divides cake2 into [what he thinks are] thirds.
- 2.2. T picks first, Alice picks second, NT gets the last of it.
THIS ENDS STAGE 2.
THIS ENDS THE PROTOCOL.

THEOREM: The above protocol is envy-free.

PROOF: By the end of stage 0, the protocol has generated 2 cakes: cake1 (in 3 pieces) and cake2 (trimmings). By the end of stage 1, as we shall see, cake1 has been distributed among the participants in an envy-free fashion. By the end of stage 2, as we shall see, cake2 has been divided into 3 pieces and distributed also in envy-free fashion. Thus the entire cake is distributed among the 3 participants in envy-free fashion*.

* Here we use the fact that if participant P_i values his piece of cake1 at least as much as he values any other person's piece of cake1, and if he values his piece of cake2 at least as much as he values any other person's piece of cake2, then P_i values his final 2 pieces of cake at least as much as he values any other participant's two pieces of cake.

It remains to prove that the protocol is envy-free to each of the participants, Alice, Bob, and Charlie. Proofs for the latter 2 are divided into 2 cases, depending on whether Bob or Charlie is T. In each case, we show that the protocol is envy-free immediately after stage 1, then extend this to show that it is envy-free at the end of stage 2.

ENVY_FREE to ALICE: Alice divides the cake into 3 equal pieces. By the end of stage 1, she gets one of the untrimmed pieces. So at that point she will not envy either of the 2 guys, one of which got a trimmed piece, the other an untrimmed piece. In fact, A will not envy T, the guy who got the trimmed piece, even if that guy gets ALL the trimmings. Thus it is okay that T gets first pick at a so-called "third" of the trimmings. So Alice will not envy T. After that, she will get at least as much of the remaining trimming as NT. Thus Alice will not envy NT.

ENVY-FREE to BOB: At the end of stage 1, Bob has one of the two largest pieces, so at that point he envies neither Alice nor Charlie. We now consider 2 cases:

CASE BOB = T: At the end of stage 1, Bob has the trimmed piece, which by his measure is at least as large as each of the other two pieces of cake. In stage 2, he is given first pick at one of three pieces of trimming, so he will not envy whoever gets second or third choice.

CASE BOB = NT: At the end of stage 1, Bob does not envy Alice or Charlie. At the start of stage 2, Bob divides the trimmings into 3 equal parts, so he will not envy either of the other two even though he gets last pick on the trimmings.

ENVY-FREE to CHARLIE: In stage 1, Charlie gets first choice of the 3 pieces. So at the end of stage 1, he does not envy either Alice or Bob. We now consider 2 cases:

CASE CHARLIE = T: Charlie gets first choice on the 3 pieces of trimming, so he will not envy the other 2 after they get their (smaller) pieces of trimming.

CASE CHARLIE = NT: Charlie decided that the untrimmed piece is the largest of the 3 pieces, and took that piece, so he does not envy the other two at the end of stage 1. As NT, he divides the trimmings into 3 equal pieces, so he will not envy the other two even though he gets last pick at the trimmings.

THIS ENDS THE PROOF.