15-859(B) Machine Learning Theory

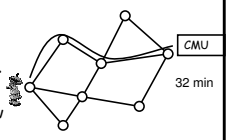
Lecture 04/14/08 & 04/16/08: Online Learning and Game Theory

Avrim Blum

[Slides from MLSS'08, so will be skipping a lot we've already covered]

Consider the following setting…

- Each morning, you need to pick one of N possible routes to drive to work.
- But traffic is different each day.
  - Not clear a priori which will be best.
  - When you get there you find out how long your route took. (And maybe others too or maybe not.)

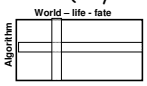


- Is there a strategy for picking routes so that in the long run, whatever the sequence of traffic patterns has been, you've done nearly as well as the best fixed route in hindsight? (In expectation, over internal randomness in the algorithm)
- Yes.

"No-regret" algorithms for repeated decisions

A bit more generally:

- Algorithm has N options. World chooses cost vector. Can view as matrix like this (maybe infinite # cols)



- At each time step, algorithm picks row, life picks column.
  - Alg pays cost for action chosen.
  - Alg gets column as feedback (or just its own cost in the "bandit" model).
  - Need to assume some bound on max cost. Let's say all costs between 0 and 1.

"No-regret" algorithms for repeated decisions

- At each time step, algorithm picks row, life picks column.
  - Alg pays cost for action chosen.
  - Alg gets column as feedback (or just its own cost in the "bandit" model).
  - Need to assume some bound on max cost. Let's say all costs between 0 and 1.

Define **average regret** in T time steps as:
(avg per-day cost of alg) – (avg per-day cost of best fixed row in hindsight).

We want this to go to 0 or better as T gets large.
[called a "no-regret" algorithm]

Some intuition & properties of no-regret algs.

- Let's look at a small example:

Algorithm \ World – life - fate		
	1	0
	0	1

- Note: Not trying to compete with best adaptive strategy – just best fixed path in hindsight.
- No-regret algorithms can do much better than playing minimax optimal, and never much worse.
- Existence of no-regret algs yields immediate proof of minimax thm!

Will define this later

This too

Some intuition & properties of no-regret algs.

- Let's look at a small example:

Algorithm \ World – life - fate		
	1	0
	0	1

- View of world/life/fate: unknown sequence LRLLRLRR…
- Goal: do well (in expectation) no matter what the sequence is.
- Algorithms must be randomized or else it's hopeless.
- Viewing as game: algorithm against the world.

History and development (abridged)

- [Hannan'57, Blackwell'56]: Alg. with regret $O((N/T)^{1/2})$.
  - Re-phrasing, need only $T = O(N/\varepsilon^2)$ steps to get time-average regret down to ε. (will call this quantity T_ε)
  - Optimal dependence on T (or ε). Game-theorists viewed #rows N as constant, not so important as T, so pretty much done.

Why optimal in T?

World – life - fate

Algorithm		
	1	0
	0	1

- Say world flips fair coin each day.
- Any alg, in T days, has expected cost T/2.
- But E[min(# heads,#tails)] = T/2 – $O(T^{1/2})$.
- So, per-day gap is $O(1/T^{1/2})$.

History and development (abridged)

- [Hannan'57, Blackwell'56]: Alg. with regret $O((N/T)^{1/2})$.
  - Re-phrasing, need only $T = O(N/\varepsilon^2)$ steps to get time-average regret down to ε. (will call this quantity T_ε)
  - Optimal dependence on T (or ε). Game-theorists viewed #rows N as constant, not so important as T, so pretty much done.
- Learning-theory 80s-90s: "combining expert advice". Imagine large class C of N prediction rules.
  - Perform (nearly) as well as best $f \in C$.
  - [LittlestoneWarmuth'89]: Weighted-majority algorithm
    - E[cost] $\leq$ OPT$(1+\varepsilon)$ + (log N)/ε.
    - Regret $O((\log N)/T)^{1/2}$. $T_\varepsilon = O((\log N)/\varepsilon^2)$.
  - Optimal as fn of N too, plus lots of work on exact constants, 2nd order terms, etc. [CFHHSW93]…
- Extensions to bandit model (adds extra factor of N).

To think about this, let's look at the problem of "combining expert advice".

[Skipping WM, RWM alg and analysis]

Summarizing

- E[# mistakes] $\leq (1+\varepsilon)m + \varepsilon^{-1}\log(N)$.
- If set $\varepsilon=(\log(N)/m)^{1/2}$ to balance the two terms out (or use guess-and-double), get bound of E[mistakes]$\leq m+2(m\cdot\log N)^{1/2}$
- Since $m \leq T$, this is at most $m + 2(T\log N)^{1/2}$.
- So, regret $\to 0$.

What if we have N options, not N predictors?

- We're not combining N experts, we're choosing one. Can we still do it?
- Nice feature of RWM: can still apply.
  - Choose expert i with probability $p_i = w_i/W$.
  - Still the same algorithm!
  - Can apply to choosing N options, so long as costs are {0,1}.
  - What about costs in [0,1]?

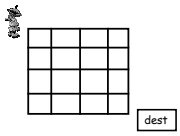
What if we have N options, not N predictors?

What about costs in [0,1]?

- If expert i has cost c_i, do: $w_i \leftarrow w_i(1-c_i\varepsilon)$.
- Our expected cost = $\sum_i c_i w_i/W$.
- Amount of weight removed = $\varepsilon \sum_i w_i c_i$.
- So, fraction removed = $\varepsilon \cdot$ (our cost).
- Rest of proof continues as before…
- So, now we can drive to work! (assuming full feedback)

Efficient implicit implementation for large N…

- Bounds have only log dependence on # choices N.
- So, conceivably can do well when N is exponential in natural problem size, if only could implement efficiently.
- E.g., case of paths…



- nxn grid has N = (2n choose n) possible paths.
- Recent years: series of results giving efficient implementation/alternatives in various settings, plus extensions to bandit model.

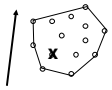
Efficient implicit implementation for large N…

- Recent years: series of results giving efficient implementation/alternatives in various settings:
  - [HelmboldSchapire97]: best pruning of given DT.
  - [BChawlaKalai02]: list-update problem.
  - [TakimotoWarmuth02]: online shortest path in DAGs.
  - [KalaiVempala03]: elegant setting generalizing all above
    - Online linear programming
  - [Zinkevich03]: elegant setting generalizing all above
    - Online convex programming
  - [AwerbuchKleinberg04][McMahanB04]:[KV]→bandit model
  - [Kleinberg,FlaxmanKalaiMcMahan05]: [Z03] → bandit model
  - [DaniHayes06]: improve bandit convergence rate
  - More…

[Kalai-Vempala'03] and [Zinkevich'03] settings

[KV] setting:

- Implicit set S of feasible points in R^m. (E.g., m=#edges, S={indicator vectors 011010010 for possible paths})
- Assume have oracle for offline problem: given vector c, find $x \in S$ to minimize $c \cdot x$. (E.g., shortest path algorithm)
- Use to solve online problem: on day t, must pick $x_t \in S$ before c_t is given.
- $(c_1 \cdot x_1 + \ldots + c_T \cdot x_T)/T \rightarrow \min_{x \in S} x \cdot (c_1 + \ldots + c_T)/T$.

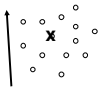


[Z] setting:

- Assume S is convex.
- Allow c(x) to be a convex function over S.
- Assume given any y not in S, can algorithmically find nearest $x \in S$.

Kalai-Vempala algorithm

- Recall setup: Set S of feasible points in R^m, of bounded diameter.
- For t = 1 to T: Alg picks $x_t \in S$, adversary picks cost vector c_t, Alg pays $x_t \cdot c_t$. Goal: compete with $x \in S$ that minimizes $x \cdot (c_1 + c_2 + \ldots + c_T)$.
- Assume have oracle for offline problem: given c, find best $x \in S$. Use to solve online.



- Algorithm is very simple:
  - Just pick $x_t \in S$ that minimizes $x \cdot (c_0 + c_1 + \ldots + c_{t-1})$,
  - where c_0 is picked from appropriate distribution.
    (in fact, closely related to Hannan's original alg.)
- Form of bounds:
  - $T_\varepsilon = O(\text{diam}(S) \cdot L_1 \text{ bound on c's} \cdot \log(m)/\varepsilon^2)$.
  - For online shortest path, $T_\varepsilon = O(nm \cdot \log(n)/\varepsilon^2)$.

Analysis sketch [KV]

Two algorithms walk into a bar…

- Alg A picks x_t minimizing $x_t \cdot c^{t-1}$, where $c^{t-1} = c_1 + \ldots + c_{t-1}$.
- Alg B picks x_t minimizing $x_t \cdot c^t$, where $c^t = c_1 + \ldots + c_t$.
  (B has fairy godparents who add c_t into history)

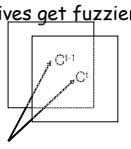
Step 1: prove B is at least as good as OPT:

$$\sum_t (\text{B's } x_t) \cdot c_t \leq \min_{x \in S} x \cdot (c_1 + \ldots + c_T)$$

Uses cute telescoping argument.

Now, A & B start drinking and their objectives get fuzzier…

Step 2: at appropriate point (width of distribution for c_0), prove A & B are similar and yet B has not been hurt too much.



Bandit setting

- What if alg is only told cost $x_t \cdot c_t$ and not c_t itself.
  - E.g., you only find out cost of your own path, not all edges in network.
- Can you still perform comparably to the best path in hindsight? (which you don't even know!)
- Ans: yes, though bounds are worse. Basic idea is fairly straightforward:
  - All we need is an estimate of $c^{t-1} = c_1 + \ldots + c_{t-1}$.
  - So, pick basis B and occasionally sample a random $x \in B$.
  - Use dot-products with basis vectors to reconstruct estimate of c^{t-1}. (Helps for B to be as orthogonal as possible)
  - Even if world is adaptive (knows what you know), still can't bias your estimate too much if you do it right.

A natural generalization

- A natural generalization of our regret goal is: what if we also want that on rainy days, we do nearly as well as the best route for rainy days.
- And on Mondays, do nearly as well as best route for Mondays.
- More generally, have N "rules" (on Monday, use path P). Goal: simultaneously, for each rule i, guarantee to do nearly as well as it *on the time steps in which it fires.*
- For all i, want $E[cost_i(alg)] \leq (1+\varepsilon)cost_i(i) + O(\varepsilon^{-1}\log N)$.
  ($cost_i(X)$ = cost of X on time steps where rule i fires.)
- Can we get this?

A natural generalization

- This generalization is esp natural in machine learning for combining multiple if-then rules.
- E.g., document classification. Rule: "if <word-X> appears then predict <Y>". E.g., if has football then classify as sports.
- So, if 90% of documents with football *are* about sports, we should have error $\leq$ 11% on them.

"Specialists" or "sleeping experts" problem.

Studied theoretically in [B95][FSSW97][BM05]; experimentally [CS'96,CS'99].

- Assume we have N rules, explicitly given.
- For all i, want $E[cost_i(alg)] \leq (1+\varepsilon)cost_i(i) + O(\varepsilon^{-1}\log N)$.
  ($cost_i(X)$ = cost of X on time steps where rule i fires.)

A simple algorithm and analysis (all on one slide)

- Start with all rules at weight 1.
- At each time step, of the rules i that fire, select one with probability $p_i \propto w_i$.
- Update weights:
  - If didn't fire, leave weight alone.
  - If did fire, raise or lower depending on performance compared to weighted average:
    - $r_i = [\sum_j p_j\, cost(j)]/(1+\varepsilon) - cost(i)$
    - $w_i \leftarrow w_i(1+\varepsilon)^{r_i}$
  - So, if rule i does exactly as well as weighted average, its weight drops a little. Weight increases if does better than weighted average by more than a $(1+\varepsilon)$ factor. This ensures sum of weights doesn't increase.
- Final $w_i = (1+\varepsilon)^{E[cost_i(alg)]/(1+\varepsilon)-cost_i(i)}$. So, exponent $\leq \varepsilon^{-1}\log N$.
- So, $E[cost_i(alg)] \leq (1+\varepsilon)cost_i(i) + O(\varepsilon^{-1}\log N)$.

Can combine with [KV],[Z] too:

- Back to driving, say we are given N "conditions" to pay attention to (is it raining?, is it a Monday?, …).
- Each day satisfies some and not others. Want simultaneously for each condition (incl default) to do nearly as well as best path for those days.
- To solve, create N rules: "if day satisfies condition i, then use output of KV_i", where KV_i is an instantiation of KV algorithm you run on just the days satisfying that condition.

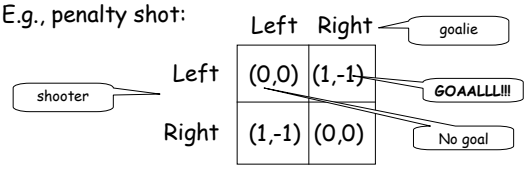
Next Topic: Game Theory

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.

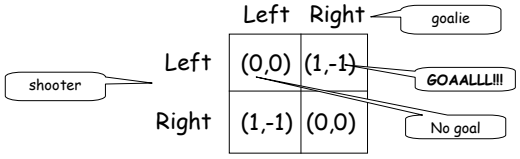
2-Player Zero-Sum games

- 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:



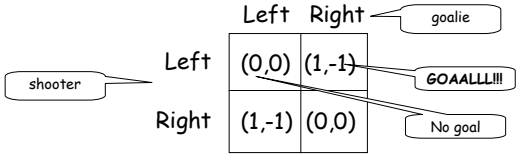
Game Theory terminolgy

- 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).



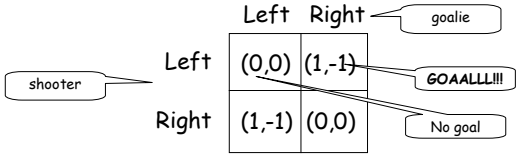
Minimax-optimal strategies

- 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

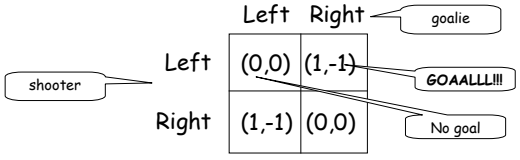
- Can solve for minimax-optimal strategies using Linear programming
- I.e., the thing to play if your opponent knows you well.



Minimax-optimal strategies

- What are the minimax optimal strategies for this game?

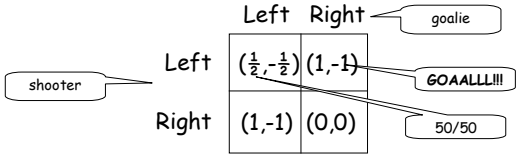
Minimax optimal strategy for both players is 50/50. Gives expected gain of $\frac{1}{2}$ for shooter ($-\frac{1}{2}$ for goalie). Any other is worse.



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.



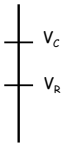
Minimax Theorem (von Neumann 1928)

- 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)

Nice proof of minimax thm

- 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 the Row player 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 [-1,0]. Say $V_R = V_C - \delta$.



Proof contd

- Now, consider playing randomized weighted-majority alg as Row, against Col who plays optimally against Row's distrib.
- In T steps,
  - Alg gets $\geq (1-\varepsilon/2)$[best row in hindsight] – $\log(N)/\varepsilon$
  - BRiH $\geq T \cdot V_C$ [Best against opponent's empirical distribution]
  - Alg $\leq T \cdot V_R$ [Each time, opponent knows your randomized strategy]
  - Gap is δT. Contradicts assumption if use $\varepsilon=\delta$, once $T > 2\log(N)/\varepsilon^2$.

Can use notion of minimax optimality to explain bluffing in poker

Simplified Poker (Kuhn 1950)

- Two players A and B.
- Deck of 3 cards: 1,2,3.
- Players ante $1.
- Each player gets one card.
- A goes first. Can bet $1 or pass.
  - If A bets, B can call or fold.
  - If A passes, B can bet $1 or pass.
    - If B bets, A can call or fold.
- High card wins (if no folding). Max pot $2.

- Two players A and B. 3 cards: 1,2,3.
- Players ante $1. Each player gets one card.
- A goes first. Can bet $1 or pass.
  - If A bets, B can call or fold.
  - If A passes, B can bet $1 or pass.
    - If B bets, A can call or fold.

Writing as a Matrix Game

- For a given card, A can decide to
  - Pass but fold if B bets. [PassFold]
  - Pass but call if B bets. [PassCall]
  - Bet. [Bet]
- Similar set of choices for B.

Can look at all strategies as a big matrix…

	[FP,FP,CB]	[FP,CP,CB]	[FB,FP,CB]	[FB,CP,CB]
[PF,PF,PC]	0	0	-1/6	-1/6
[PF,PF,B]	0	1/6	-1/3	-1/6
[PF,PC,PC]	-1/6	0	0	1/6
[PF,PC,B]	-1/6	-1/6	1/6	1/6
[B,PF,PC]	-1/6	0	0	1/6
[B,PF,B]	1/6	-1/3	0	-1/2
[B,PC,PC]	1/6	-1/6	-1/6	-1/2
[B,PC,B]	0	-1/2	1/3	-1/6
	0	-1/3	1/6	-1/6

And the minimax optimal strategies are…

- A:
  - If hold 1, then 5/6 PassFold and 1/6 Bet.
  - If hold 2, then ½ PassFold and ½ PassCall.
  - If hold 3, then ½ PassCall and ½ Bet.

  Has both bluffing and underbidding…
- B:
  - If hold 1, then 2/3 FoldPass and 1/3 FoldBet.
  - If hold 2, then 2/3 FoldPass and 1/3 CallPass.
  - If hold 3, then CallBet

  Minimax value of game is –1/18 to A.

Now, to General-Sum games!

General-Sum Games

- Zero-sum games are good formalism for worst-case 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?":

you \ person walking towards you	Left	Right
Left	(1,1)	(-1,-1)
Right	(-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?":

	Spartans	Atonement
Spartans	(8,2)	(0,0)
Atonement	(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.

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., "which movie to go to":

	Spartans	Atonement
Spartans	(8,2)	(0,0)
Atonement	(0,0)	(2,8)

NE are: both S, both A, or (80/20,20/80)

Uses

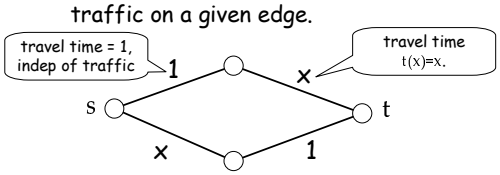
- Economists use games and equilibria as models of interaction.
- E.g., pollution / prisoner's dilemma:
  - (imagine pollution controls cost $4 but improve everyone's environment by $3)

	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

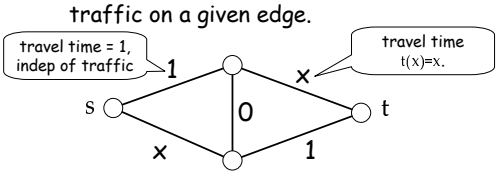
- Braess paradox:
  - Road network, traffic going from s to t.
  - travel time as function of fraction x of traffic on a given edge.



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 × 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 $f:S \to S$ 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...

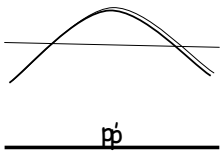
- f(p,q) = (p',q') such that:
  - q' maximizes [(expected gain wrt p) - $||q-q'||^2$]
  - p' maximizes [(expected gain wrt q) - $||p-p'||^2$]

p p'

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!

One more interesting game

"Ultimatum game":

- Two players "Splitter" and "Chooser"
- 3rd party puts $10 on table.
- Splitter gets to decide how to split between himself and Chooser.
- Chooser can accept or reject.
- If reject, money is burned.

One more interesting game

"Ultimatum game": E.g., with $4

Splitter: how much to offer chooser

Chooser: how much to accept

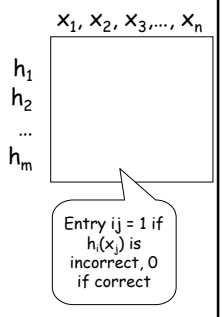
	1	2	3
1	(1,3)	(2,2)	(3,1)
2	(0,0)	(2,2)	(3,1)
3	(0,0)	(0,0)	(3,1)

Boosting & game theory

- Suppose I have an algorithm A that for any distribution (weighting fn) over a dataset S can produce a rule $h \in H$ that gets < 40% error.
- Adaboost gives a way to use such an A to get error $\rightarrow$ 0 at a good rate, using weighted votes of rules produced.
- How can we see that this is even possible?

Boosting & game theory

- Assume for all D over cols, exists a row with cost < 0.4.
- Minimax implies must exist a weighting over rows s.t. for every x_i, the vote is at least 60/40 in the right way.
- So, weighted vote has L_1 margin at least 0.2.
- (Of course, AdaBoost gives you a way to get at it with only access via A. But this at least implies existence…)



Stop 3: What happens if everyone is adapting their behavior?

What if everyone started using no-regret algs?

- What if changing cost function is due to other players in the system optimizing for themselves?
- No-regret can be viewed as a nice definition of reasonable self-interested behavior. So, what happens to overall system if everyone uses one?
- In zero-sum games, empirical frequencies quickly approaches minimax optimal.
  - If your empirical distribution of play didn't, then opponent would be able to (and have to) take advantage, giving you < V.

What if everyone started using no-regret algs?

- What if changing cost function is due to other players in the system optimizing for themselves?
- No-regret can be viewed as a nice definition of reasonable self-interested behavior. So, what happens to overall system if everyone uses one?
- In zero-sum games, empirical frequencies quickly approaches minimax optimal.
- In general-sum games, does behavior quickly (or at all) approach a Nash equilibrium? (after all, a Nash Eq is exactly a set of distributions that are no-regret wrt each other).
- Well, unfortunately, no.

A bad example for general-sum games

- Augmented Shapley game from [Z04]: "RPSF"
  - First 3 rows/cols are Shapley game (rock / paper / scissors but if both do same action then both lose).
  - 4th action "play foosball" has slight negative if other player is still doing r/p/s but positive if other player does 4th action too.
  - NR algs will cycle among first 3 and have no regret, but do worse than only Nash Equilibrium of both playing foosball.
- We didn't really expect this to work given how hard NE can be to find…

What *can* we say?

- If algorithms minimize "internal" or "swap" regret, then empirical distribution of play approaches *correlated* equilibrium.
  - Foster & Vohra, Hart & Mas-Colell,…
  - Though doesn't imply play is stabilizing.
- In some natural cases, like routing in Wardrop model, can show daily traffic actually approaches Nash.

More general forms of regret

1. "best expert" or "external" regret:
   - Given n strategies. Compete with best of them in hindsight.
2. "sleeping expert" or "regret with time-intervals":
   - Given n strategies, k properties. Let S_i be set of days satisfying property i (might overlap). Want to simultaneously achieve low regret over each S_i.
3. "internal" or "swap" regret: like (2), except that S_i = set of days in which we chose strategy i.

Internal/swap-regret

- E.g., each day we pick one stock to buy shares in.
  - Don't want to have regret of the form "every time I bought IBM, I should have bought Microsoft instead".
- Formally, regret is wrt optimal function $f:\{1,\ldots,N\}\rightarrow\{1,\ldots,N\}$ such that every time you played action j, it plays f(j).
- Motivation: connection to correlated equilibria.

Internal/swap-regret

"Correlated equilibrium"

- Distribution over entries in matrix, such that if a trusted party chooses one at random and tells you your part, you have no incentive to deviate.
- E.g., Shapley game.

	R	P	S
R	-1,-1	-1,1	1,-1
P	1,-1	-1,-1	-1,1
S	-1,1	1,-1	-1,-1

Internal/swap-regret

- If all parties run a low internal/swap regret algorithm, then empirical distribution of play is an apx correlated equilibrium.
  - Correlator chooses random time $t \in \{1,2,\ldots,T\}$. Tells each player to play the action j they played in time t (but does not reveal value of t).
  - Expected incentive to deviate: $\sum_j \Pr(j)(\text{Regret}|j)$ = swap-regret of algorithm
  - So, this says that correlated equilibria are a natural thing to see in multi-agent systems where individuals are optimizing for themselves

Internal/swap-regret, contd

Algorithms for achieving low regret of this form:

- Foster & Vohra, Hart & Mas-Colell, Fudenberg & Levine.
- Can also convert any "best expert" algorithm into one achieving low swap regret.
- Unfortunately, time to achieve low regret is linear in n rather than log(n)….

Internal/swap-regret, contd

Can convert any "best expert" algorithm A into one achieving low swap regret. Idea:

- Instantiate one copy A_i responsible for expected regret over times we play i.
- Each time step, if we play $p=(p_1,\ldots,p_n)$ and get cost vector $c=(c_1,\ldots,c_n)$, then A_i gets cost-vector $p_i c$.
- If each A_i proposed to play q_i, so all together we have matrix Q, then define p = pQ.
- Allows us to view p_i as prob we chose action i or prob we chose algorithm A_i.