Online Learning and Game Theory

+

On Learning with Similarity Functions

Your Guide: Avrim Blum

Carnegie Mellon University

[Machine Learning Summer School 2008]

Plan for the tour:

- Stop 1: Online learning, minimizing regret, and combining expert advice.
- Stop 2: Game theory, minimax optimality, and Nash equilibria.
- Stop 3: Correlated equilibria, internal regret, routing games, and connections between 1 and 2.
- Stop 4: (something completely different) Learning and clustering with similarity functions.

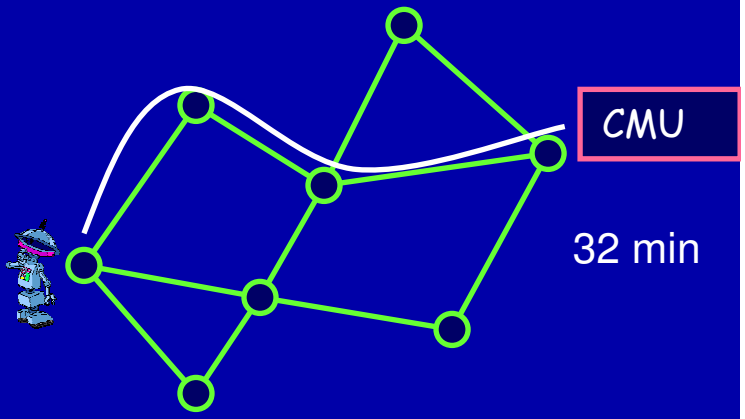
Some books/references:

- *Algorithmic Game Theory*, Nisan, Roughgarden, Tardos, Vazirani (eds), Cambridge Univ Press, 2007. [Chapter 4 "Learning, Regret Minimization, & Equilibria" is on my webpage]
- *Prediction, Learning, & Games*, Cesa-Bianchi & Lugosi, Cambridge Univ Press, 2006.
- My course notes: www.machinelearning.com
  - Stop 1: Online learning, minimizing regret, and combining expert advice.
  - Stop 2: Game theory, minimax optimality, and Nash equilibria.
  - Stop 3: Correlated equilibria, internal regret, routing games, and connections between 1 and 2.
  - Stop 4: (something completely different) Learning and clustering with similarity functions.

Stop 1: Online learning, minimizing regret, and combining expert advice

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.)

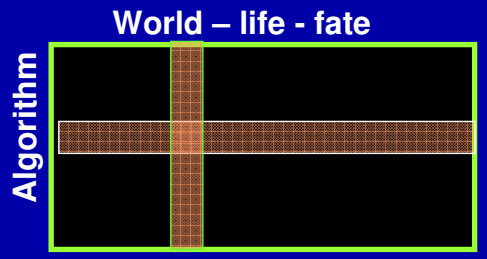


- Is there a strategy for picking routes so that in the long run, whatever the sequence of traffic patterns has been, you've done nearly as well as the best fixed route in hindsight? (In expectation, over internal randomness in the algorithm)
- Yes.

"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

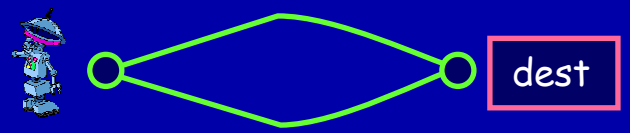
- At each time step, algorithm picks row, life picks column.
  Define **average regret** in T time steps as:
  - Alg pays cost for action chosen.
  - (avg per-day cost of alg) - (avg per-day cost of best fixed row in hindsight).
  - Alg gets column as feedback (or just its own cost in the "bandit" model).

  We want this to go to 0 or better as T gets large.
  - Need to assume some bound on max cost. Let's say all costs between 0 and 1.

  [called a "no-regret" algorithm]

Some intuition & properties of no-regret algs.

- Let's look at a small example:



World – life - fate

Algorithm		
	1	0
	0	1

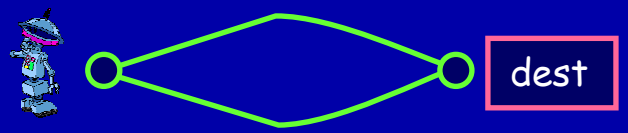
- Note: Not trying to compete with best adaptive strategy – just best fixed path in hindsight.
- No-regret algorithms can do much better than playing minimax optimal, and never much worse.
- Existence of no-regret algs yields immediate proof of minimax thm!

Will define this later

This too

Some intuition & properties of no-regret algs.

- Let's look at a small example:



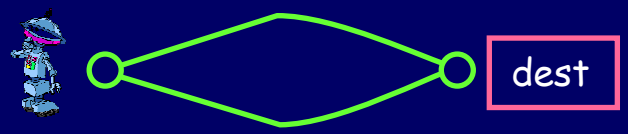
Algorithm \ World – life - fate	
1	0
0	1

- 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?



Algorithm \ World – life - fate		
	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 f2C.
  - [LittlestoneWarmuth'89]: Weighted-majority algorithm
    - $E[\text{cost}] \cdot OPT(1+\varepsilon) + (\log N)/\varepsilon$.
    - 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".

Using "expert" advice

Say we want to predict the stock market.

- We solicit N "experts" for their advice. (Will the market go up or down?)
- We then want to use their advice somehow to make our prediction. E.g.,

Expt 1	Expt 2	Expt 3	neighbor's dog	truth
down	up	up	up	up
down	up	up	down	down
…	…	…	…	…

Can we do nearly as well as best in hindsight?

["expert" ´ someone with an opinion. Not necessarily someone who knows anything.]

Simpler question

- We have N "experts".
- One of these is perfect (never makes a mistake). We just don't know which one.
- Can we find a strategy that makes no more than lg(N) mistakes?

Answer: sure. Just take majority vote over all experts that have been correct so far.

- Each mistake cuts # available by factor of 2.
- Note: this means ok for N to be very large.

"halving algorithm"

Using "expert" advice

But what if none is perfect? Can we do nearly as well as the best one in hindsight?

Strategy #1:

- Iterated halving algorithm. Same as before, but once we've crossed off all the experts, restart from the beginning.
- Makes at most $\lg(N)[OPT+1]$ mistakes, where OPT is #mistakes of the best expert in hindsight.

Seems wasteful. Constantly forgetting what we've "learned". Can we do better?

Weighted Majority Algorithm

Intuition: Making a mistake doesn't completely disqualify an expert. So, instead of crossing off, just lower its weight.

Weighted Majority Alg:

- Start with all experts having weight 1.
- Predict based on weighted majority vote.
- Penalize mistakes by cutting weight in half.

					prediction	correct
weights	1	1	1	1		
predictions	Y	Y	Y	N	Y	Y
weights	1	1	1	.5		
predictions	Y	N	N	Y	N	Y
weights	1	.5	.5	.5		

Analysis: do nearly as well as best expert in hindsight

- M = # mistakes we've made so far.
- m = # mistakes best expert has made so far.
- W = total weight (starts at N).
- After each mistake, W drops by at least 25%.
  So, after M mistakes, W is at most $N(3/4)^M$.
- Weight of best expert is $(1/2)^m$. So,

$$(1/2)^m \leq N(3/4)^M$$
$$(4/3)^M \leq N2^m$$
$$M \leq 2.4(m + \lg N)$$

constant ratio

Randomized Weighted Majority

2.4(m + lg N) not so good if the best expert makes a mistake 20% of the time. Can we do better? Yes.

- Instead of taking majority vote, use weights as probabilities. (e.g., if 70% on up, 30% on down, then pick 70:30) Idea: smooth out the worst case.
- Also, generalize $\frac{1}{2}$ to 1- ε.

Solves to: $M \leq \frac{-m \ln(1-\varepsilon) + \ln(N)}{\varepsilon} \approx (1+\varepsilon/2)m + \frac{1}{\varepsilon}\ln(N)$

M = expected #mistakes

$M \leq 1.39m + 2\ln N \quad \leftarrow \varepsilon = 1/2$

$M \leq 1.15m + 4\ln N \quad \leftarrow \varepsilon = 1/4$

$M \leq 1.07m + 8\ln N \quad \leftarrow \varepsilon = 1/8$

unlike most worst-case bounds, numbers are pretty good.

Analysis

- Say at time t we have fraction F_t of weight on experts that made mistake.
- So, we have probability F_t of making a mistake, and we remove an εF_t fraction of the total weight.
  - $W_{final} = N(1-\varepsilon F_1)(1 - \varepsilon F_2)\ldots$
  - $\ln(W_{final}) = \ln(N) + \sum_t [\ln(1 - \varepsilon F_t)] \cdot \ln(N) - \varepsilon \sum_t F_t$

    (using $\ln(1-x) < -x$)

    $= \ln(N) - \varepsilon M.$ ($\sum F_t$ = E[# mistakes])
- If best expert makes m mistakes, then $\ln(W_{final}) > \ln((1-\varepsilon)^m)$.
- Now solve: $\ln(N) - \varepsilon M > m \ln(1-\varepsilon)$.

$$M \leq \frac{-m \ln(1-\varepsilon) + \ln(N)}{\varepsilon} \approx (1+\varepsilon/2)m + \frac{1}{\varepsilon}\log(N)$$

Summarizing

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

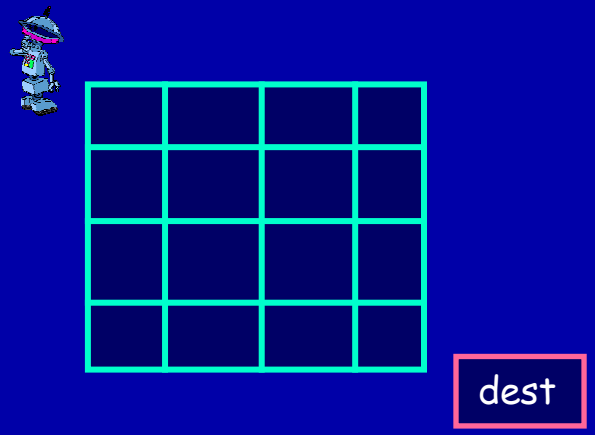
What if we have N options, not N predictors?

What about costs in [0,1]?

- If expert i has cost c_i, do: w_i Ã $w_i(1-c_i\varepsilon)$.
- Our expected cost = $\sum_i c_i w_i / W$.
- Amount of weight removed = $\varepsilon \sum_i w_i c_i$.
- So, fraction removed = ε ¢ (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.

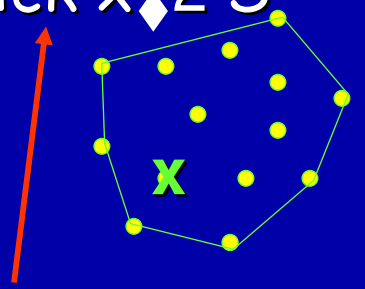
Efficient implicit implementation for large N…

- 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 2 S to minimize c¢x. (E.g., shortest path algorithm)
- Use to solve online problem: on day t, must pick x_t 2 S before c_t is given.
- $(c_1¢x_1+...+c_T¢x_T)/T$! $\min_{x2S} x¢(c_1+...+c_T)/T$.



[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 2 S.

Problems that can be modeled:

Online shortest paths.

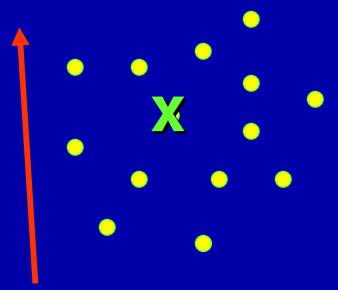
Web-search-results ordering problem:

- For a given query (say, "Ancient Greece"), output an ordering of search results based on what people have clicked on before, where cost = f(depth of item clicked) (S = set of permutation matrices)

Some inventory problems, adaptive pricing problems

Kalai-Vempala algorithm

- Recall setup: Set S of feasible points in R^m, of bounded diameter.
- For ♦ = 1 to T: Alg picks $x_♦$ 2 S, adversary picks cost vector $c_♦$, Alg pays $x_♦$ ¢ $c_♦$. Goal: compete with x 2 S that minimizes ☒ ¢ $(c_1 + c_2 + \ldots + c_T)$.
- Assume have oracle for offline problem: given c, find best x 2 S. Use to solve online.



- Algorithm is very simple:
  - Just pick $x_♦$ 2 S that minimizes $x¢(c_0 + c_1 + \ldots + c_{♦-1})$,
  - where c_0 is picked from appropriate distribution.

  (in fact, closely related to Hannan's original alg.)
- Form of bounds:
  - $T_\varepsilon = O(\text{diam}(S)\ ¢\ L_1$ bound on c's ¢ $\log(m)/\varepsilon^2)$.
  - For online shortest path, $T_\varepsilon = O(nm¢\log(n)/\varepsilon^2)$.

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)

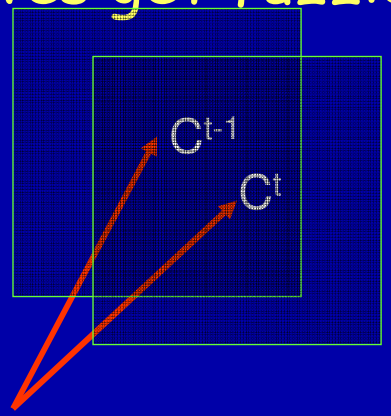
Step 1: prove B is at least as good as OPT:

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

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 ☒♦¢♍♦ and not ♍♦ 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 x2B.
  - 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 $E[cost_i(alg)] \cdot (1+\varepsilon)cost_i(i) + O(\varepsilon^{-1}\log N)$.
  ($cost_i(X)$ = 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 · 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 $E[cost_i(alg)] \cdot (1+\varepsilon)cost_i(i) + O(\varepsilon^{-1}\log N)$.

($cost_i(X)$ = 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 / w_i.
- Update weights:
  - If didn't fire, leave weight alone.
  - If did fire, raise or lower depending on performance compared to weighted average:
    - $r_i = [\sum_j p_j \text{ cost}(j)]/(1+\varepsilon) - \text{cost}(i)$
    - w_i Ã $w_i(1+\varepsilon)^{r_i}$
  - 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 $w_i = (1+\varepsilon)^{E[\text{cost}_i(\text{alg})]/(1+\varepsilon)-\text{cost}_i(i)}$. So, exponent · $\varepsilon^{-1}\log N$.
- So, $E[\text{cost}_i(\text{alg})] \cdot (1+\varepsilon)\text{cost}_i(i) + O(\varepsilon^{-1}\log N)$.

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.

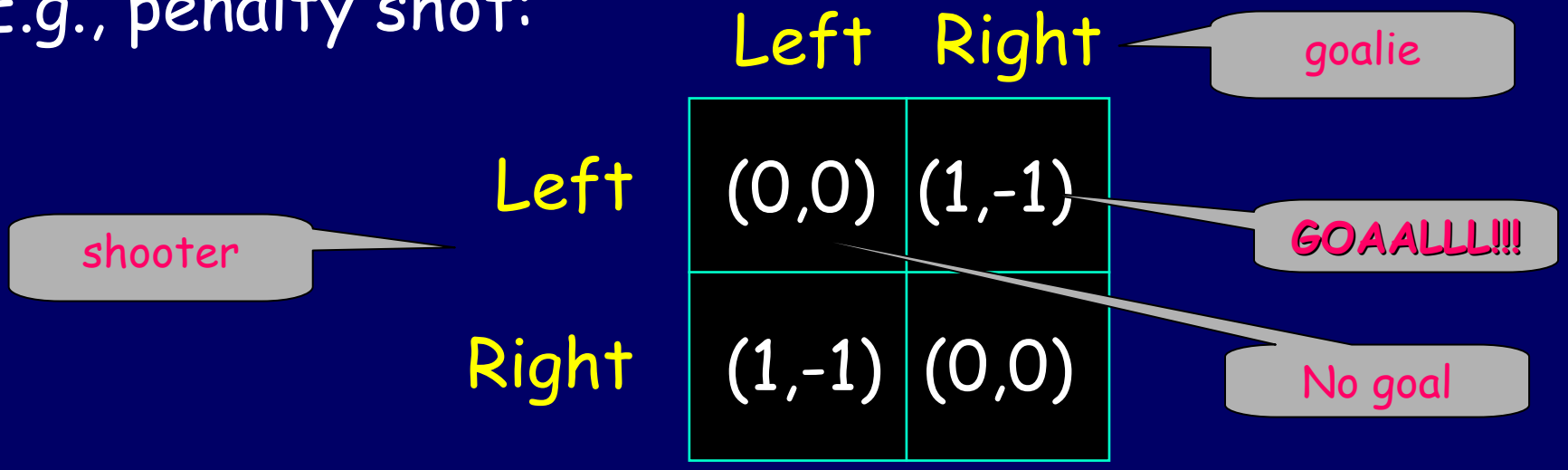
Stop 2: Game Theory

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 goooooaaaaaall!
- Vice-versa for shooter.

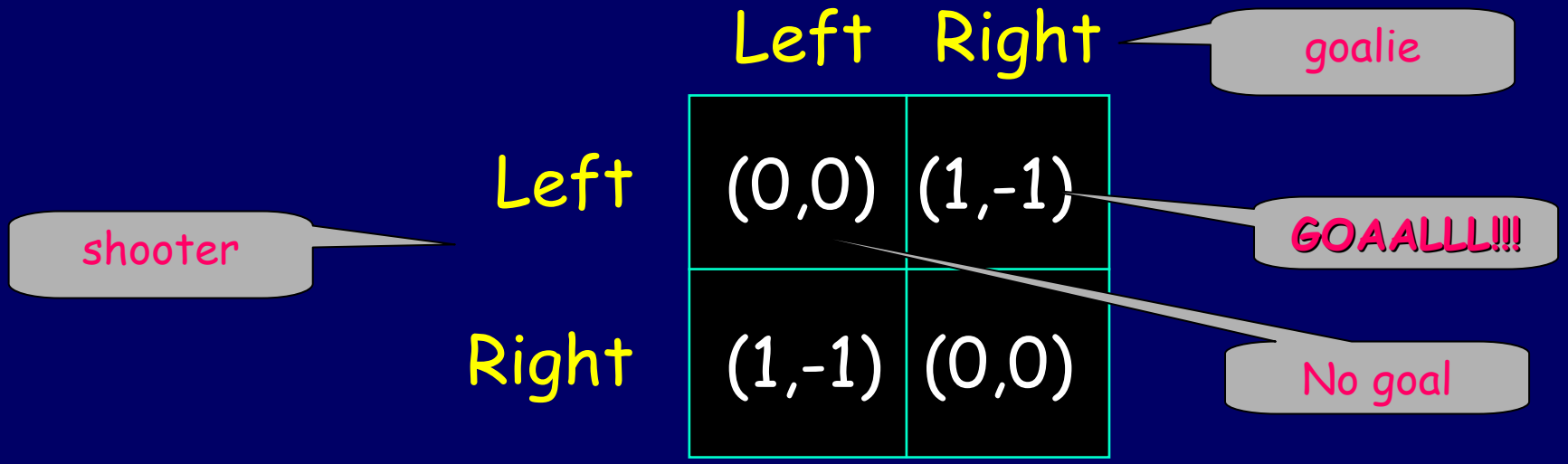
2-Player Zero-Sum games

- 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:



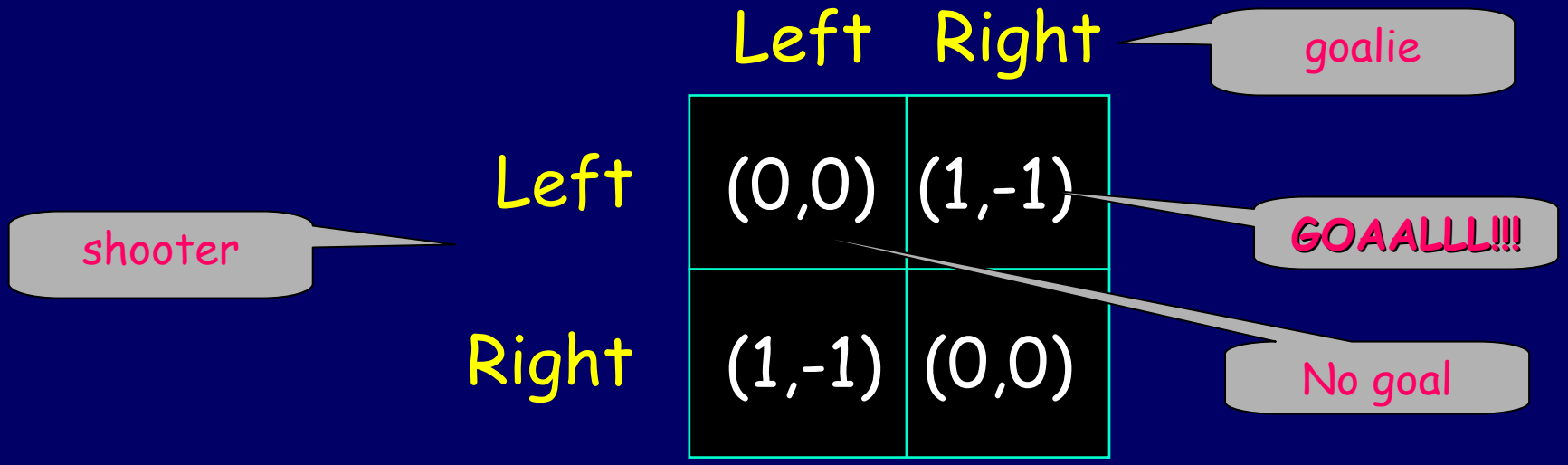
Game Theory terminolgy

- 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

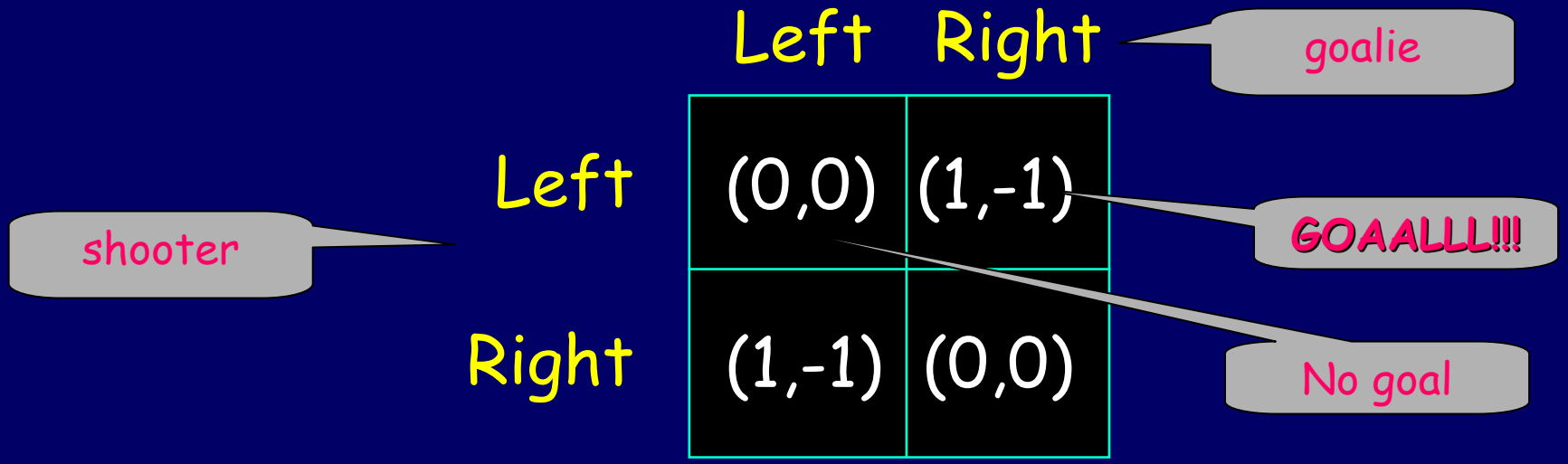
- Minimax optimal strategy is a (randomized) strategy that has the best guarantee on its expected gain, over choices of the opponent. [maximizes the minimum]
- I.e., the thing to play if your opponent knows you well.



Minimax-optimal strategies

- Can solve for minimax-optimal strategies using Linear programming

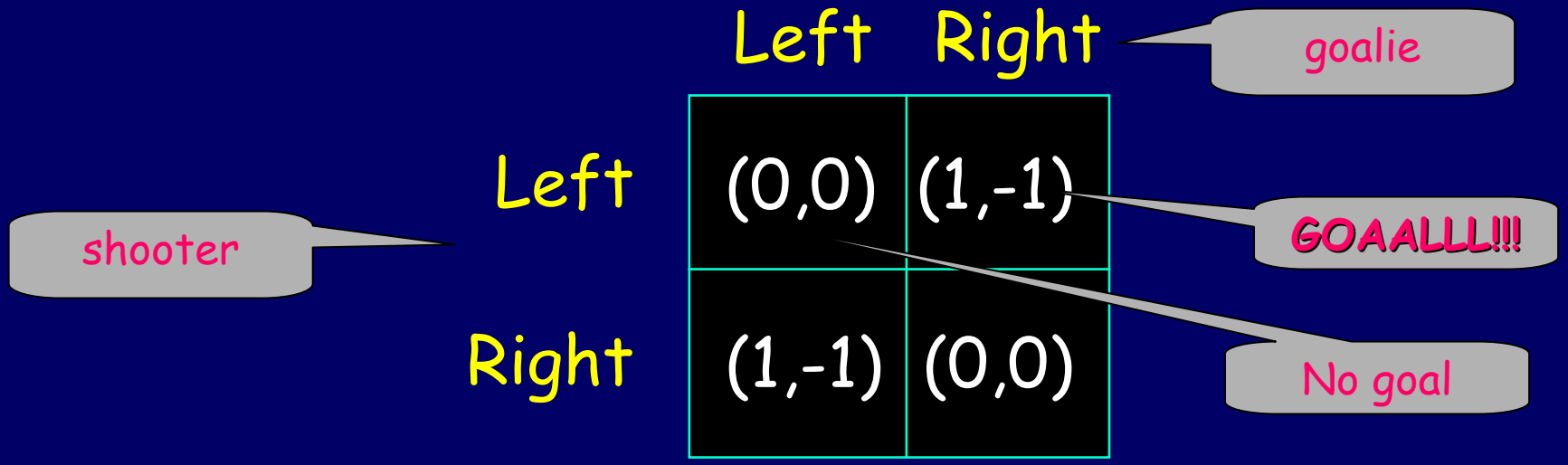
- I.e., the thing to play if your opponent knows you well.



Minimax-optimal strategies

- What are the minimax optimal strategies for this game?

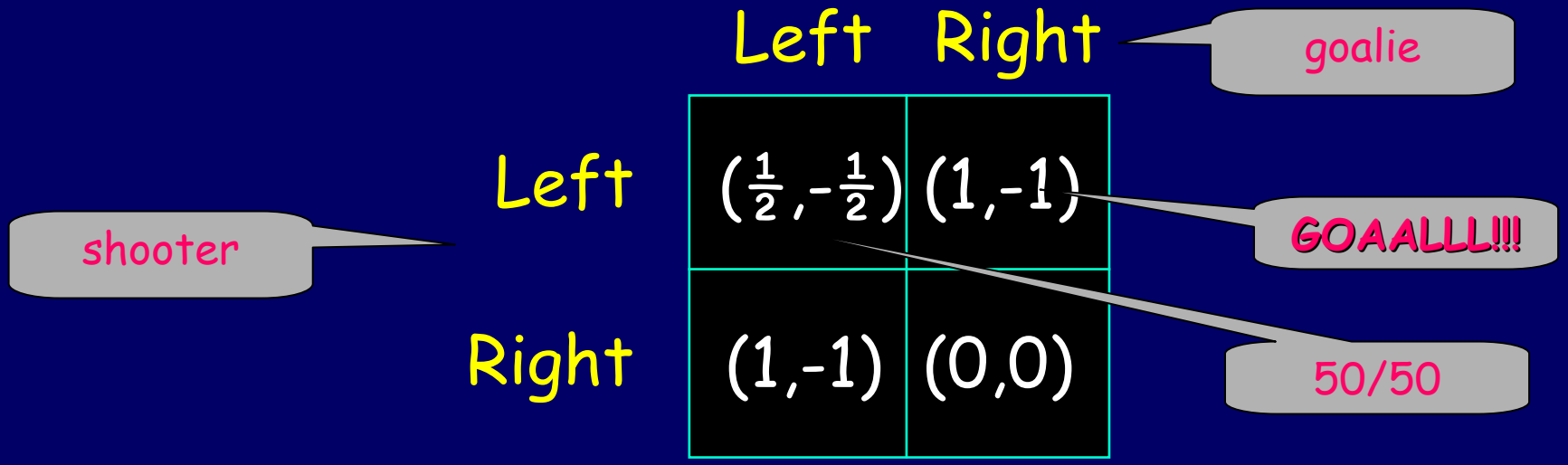
Minimax optimal strategy for both players is 50/50. Gives expected gain of $\frac{1}{2}$ for shooter ($-\frac{1}{2}$ for goalie). Any other is worse.



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.



Shall we play a game...?

I put either a quarter or nickel in my hand. You guess. If you guess right, you get the coin. Else you get nothing.

All right!

Summary of game

Value to guesser

guess:	hide N	hide Q
N	5	0
Q	0	25

Should guesser always guess Q? 50/50?

What is minimax optimal strategy?

Summary of game

Value to guesser

guess:	hide N	hide Q
N	5	0
Q	0	25

If guesser always guesses Q, then hider will hide N. Value to guesser = 0.

If guesser does 50/50, hider will still hide N. E[value to guesser] = $\frac{1}{2}(5) + \frac{1}{2}(0) = 2.5$

Summary of game

Value to guesser

guess:	hide N	hide Q
N	5	0
Q	0	25

If guesser guesses 5/6 N, 1/6 Q, then:

- if hider hides N, E[value] = (5/6)*5 ~ 4.2
- if hider hides Q, E[value] = 25/6 also.

Summary of game

Value to guesser

		hide N	hide Q
guess:	N	5	0
	Q	0	25

What about hider?

Minimax optimal strategy: 5/6 N, 1/6 Q. Guarantees expected loss at most 25/6, no matter what the guesser does.

Interesting. The hider has a (randomized) strategy *he* can reveal with expected loss · 4.2 against any opponent, and the guesser has a strategy *she* can reveal with expected gain ¸ 4.2 against any opponent.

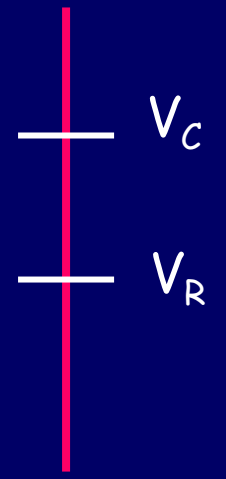
Minimax Theorem (von Neumann 1928)

- 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 ¸ $(1-\varepsilon/2)$[best row in hindsight] – $\log(N)/\varepsilon$
  - BRiH ¸ T¢V_C [Best against opponent's empirical distribution]
  - Alg · T¢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

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.

Recent work

- [Gilpin & Sandholm] solved for minimax optimal in micro version of 2-player Texas hold'em (Rhode Island hold'em).

  [52 card deck. Ante of $5. Get one card face down. Round of betting at $10. Flop card. Round of betting at $20. Turn card. Round of betting at $20. Showdown with 3-card hands.]

  - Use various methods to reduce to LP with about 1m rows/cols. Solving: 1 week using 25 gig RAM.
- [McMahan & Gordon] show online learning techniques can get minimax 2[-0.28,-0.46] in 2 hrs.

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

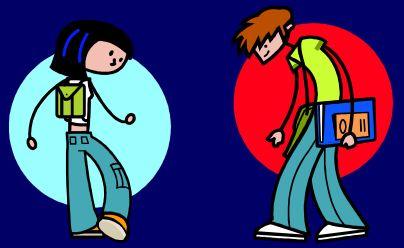
General-sum games

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

person walking towards you

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

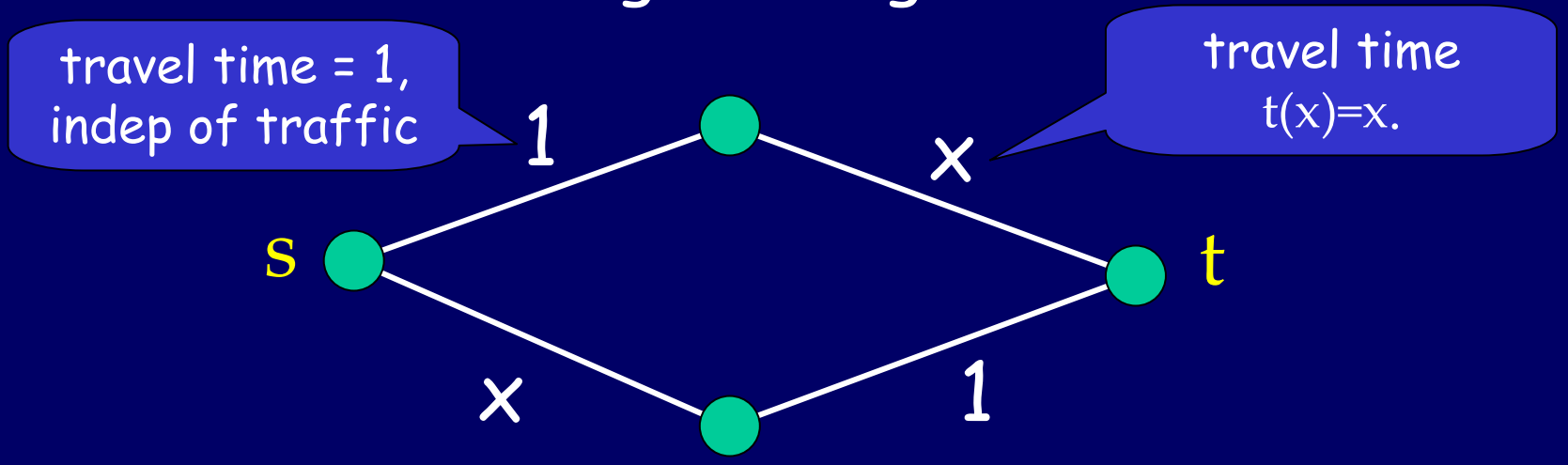
- Economists use games and equilibria as models of interaction.
- E.g., pollution / prisoner's dilemma:
  - (imagine pollution controls cost $4 but improve everyone's environment by $3)

	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

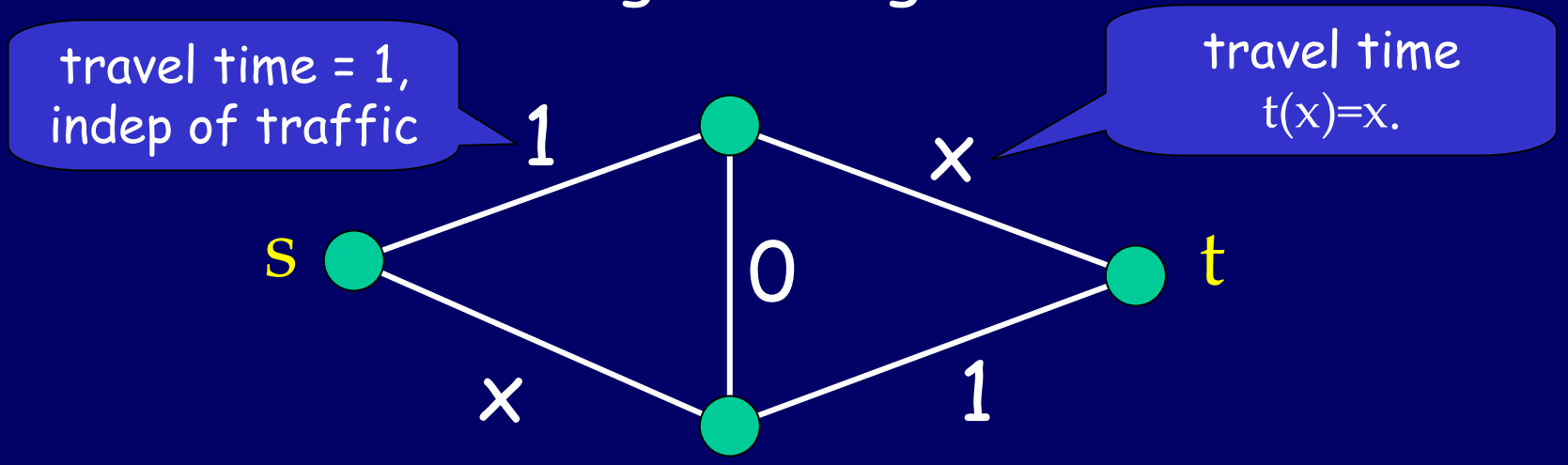
- Braess paradox:
  - Road network, traffic going from s to t.
  - travel time as function of fraction x of traffic on a given edge.



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 ! S be a continuous function.
  - Then there must exist x 2 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 = $simplex_n$ £ $simplex_n$

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)

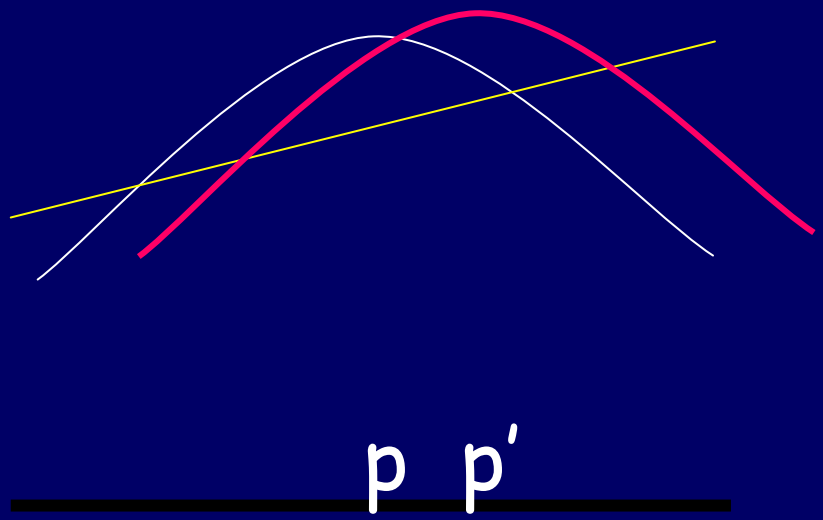
Try #1

- 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…

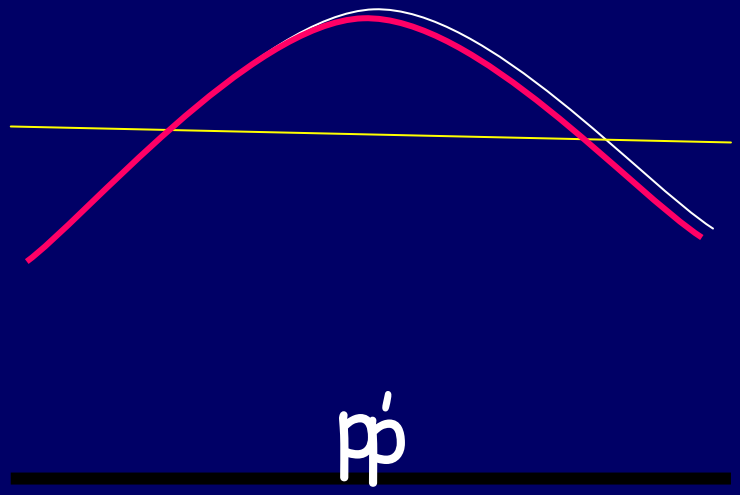
- f(p,q) = (p′,q′) such that:
  - q′ maximizes [(expected gain wrt p) - $||q-q'||^2$]
  - p′ maximizes [(expected gain wrt q) - $||p-p'||^2$]



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

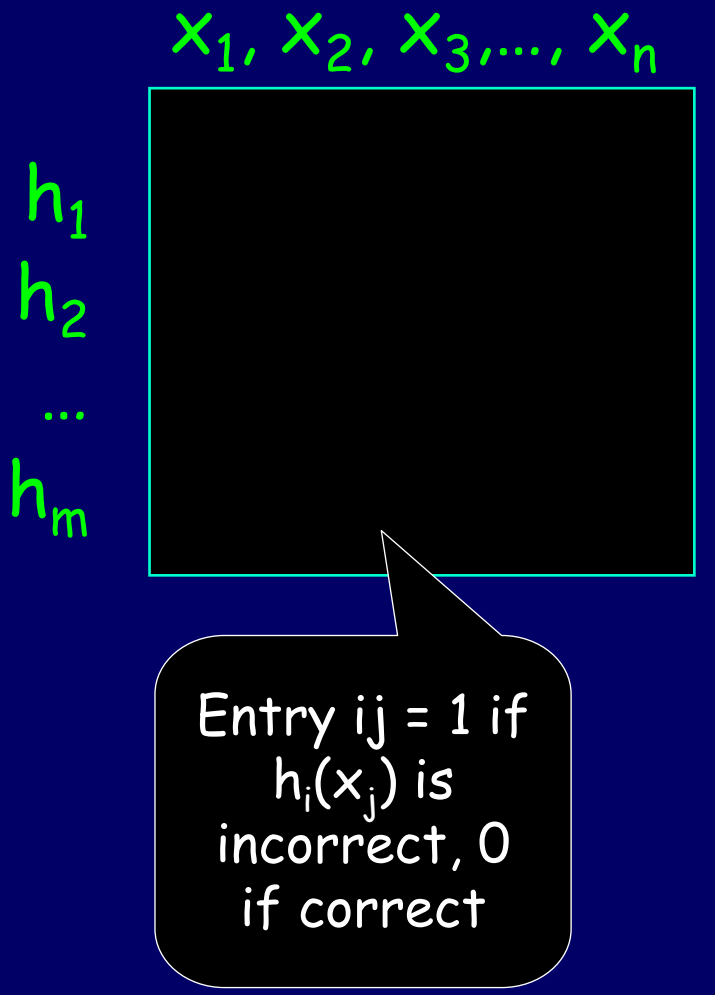
	1	2	3
1	(1,3)	(2,2)	(3,1)
2	(0,0)	(2,2)	(3,1)
3	(0,0)	(0,0)	(3,1)

Boosting & game theory

- Suppose I have an algorithm A that for any distribution (weighting fn) over a dataset S can produce a rule h2H that gets < 40% error.
- Adaboost gives a way to use such an A to get error ! 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,…,N}!{1,…,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.

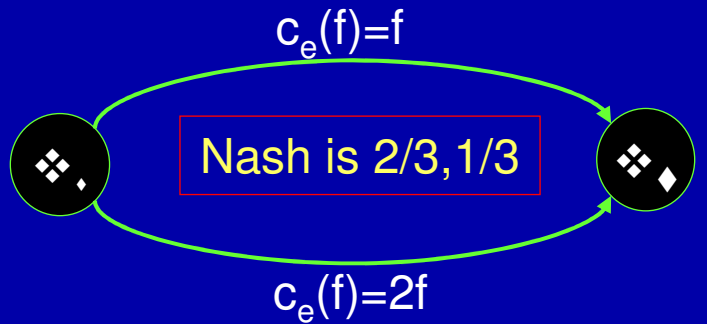
What *can* we say?

- 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.

Consider Wardrop/Roughgarden-Tardos traffic model

- Given a graph G. Each edge e has non-decreasing cost function $c_e(f_e)$ that tells latency of that edge as a function of the amount of traffic using it.
- Say 1 unit of traffic (infinitesimal users) wants to travel from v_s to v_t. E.g., simple case:



- Nash equilibrium is flow f^* such that all paths with positive flow have the same cost, and no path is cheaper.
  - $\text{Cost}(f) = \sum_e c_e(f_e)f_e$ = cost of average user under f.
  - $\text{Cost}_f(P) = \sum_{e \in P} c_e(f_e)$ = cost of using path P given f.
  - So, at Nash, $\text{Cost}(f^*) = \min_P \text{Cost}_{f^*}(P)$.
- What happens if people use no-regret algorithms?

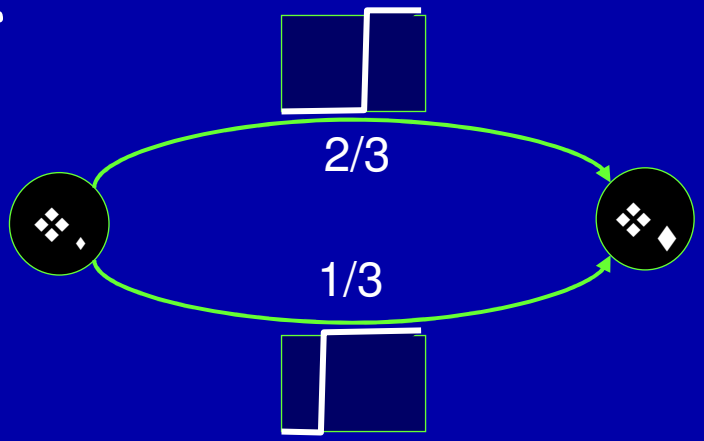
Consider Wardrop/Roughgarden-Tardos traffic model

- These are "potential games" so Nash Equilibria are not that hard to find.
- In fact, a number of distributed procedures are known that will approach Nash at a good rate.

Analyzing result of no-regret algorithms ¼ are Nash Equilibria the inevitable result of users intelligently behaving in their own interest?

Global behavior of NR algs

- Interesting case to consider:



  - NR alg: cost approaches $\frac{1}{2}$ per day, which is cost of best fixed path in hindsight.
  - But none of the individual days is an ε-Nash flow (a flow where only a small fraction of traffic has significant incentive to switch). Same for time-average flow f^{avg}.

Global behavior of NR algs [B-EvenDar-Ligett]

Assuming edge latency functions have bounded slope, can show that traffic flow approaches equilibrium in that for $\varepsilon \to 0$ we have:

- For $1-\varepsilon$ fraction of days, a $1-\varepsilon$ fraction of people have at most ε incentive to deviate. (traveling on a path at most ε worse than optimal given the traffic flow).
- And you really do need all three caveats. E.g., if users are running bandit algorithms, then on any given day a small fraction will be performing "exploration" steps.

Global behavior of NR algs [B-EvenDar-Ligett]

Argument outline:

1. For any edge e, time-avg cost · flow-avg cost. So,
$$avg_t[c_e(f^\blacklozenge)] \cdot avg_\blacklozenge[c_e(f^\blacklozenge) \text{¢} f^\blacklozenge] / f_e^{avg}$$

Global behavior of NR algs [B-EvenDar-Ligett]

Argument outline:

1. For any edge e, time-avg cost · flow-avg cost. So,
   $$f_e^{avg} \cent avg_t[c_e(f^\blacklozenge)] \cdot avg_\blacklozenge[c_e(f^\blacklozenge) \cent f^\blacklozenge]$$
2. Summing over all edges, and applying the regret bound:
   $avg_\blacklozenge[Cost_f\blacklozenge(f^{avg})] \cdot avg_\blacklozenge[Cost(f^\blacklozenge)] \cdot \varepsilon + \min_p avg_\blacklozenge[Cost_f\blacklozenge(P)]$, which in turn is $\cdot \varepsilon +$ $avg_\blacklozenge[Cost_f\blacklozenge(f^{avg})]$.
3. This means that actually, for each edge, the time-avg cost must be pretty close to the flow-avg cost, which (by the assumption of bounded slope) means the costs can't vary too much over time.
4. Once you show costs are stabilizing, then easy to show that low regret) Nash conditions approximately satisfied.

Global behavior of NR algs [B-EvenDar-Ligett]

Argument outline:

1. For any edge e, time-avg cost · flow-avg cost. So,
$$f_e^{avg} \cent avg_t[c_e(f^\blacklozenge)] \cdot avg_\blacklozenge[c_e(f^\blacklozenge) \cent f^\blacklozenge]$$
2. But can show the sum over all edges of this gap is a lower bound on the average regret of the players.
3. This means that actually, for each edge, the time-avg cost must be pretty close to the flow-avg cost, which (by the assumption of bounded slope) means the costs can't vary too much over time.
4. Once you show costs are stabilizing, then easy to show that low regret) Nash conditions approximately satisfied.

Last stop: On a Theory of Similarity Functions for Learning and Clustering

[Includes work joint with Nina Balcan, Nati Srebro, and Santosh Vempala]

Last stop: On a Theory of Similarity Functions for Learning and Clustering

[Includes work joint with Nina Balcan, Nati Srebro, and Santosh Vempala]

2-minute version

- Suppose we are given a set of images   , and want to learn a rule to distinguish men from women. Problem: pixel representation not so good.
- A powerful technique for such settings is to use a **kernel**: a special kind of pairwise functi
- In pr od measure of simila of implicit mappings.

Caveat: speaker knows next to nothing about computer vision.

Q: Can ory that just views K as a me ally, make it easier to desig nore general too.

2-minute version

- Suppose we are given a set of images   , and want to learn a rule to distinguish men from women. Problem: pixel representation not so good.
- A powerful technique for such settings is to use a **kernel**: a special kind of pairwise function **K(**  ,  **)**.
- In practice, choose K to be good measure of similarity, but theory in terms of implicit mappings.

Q: What if we only have unlabeled data (i.e., clustering)? Can we develop a theory of properties that are sufficient to be able to cluster well?

2-minute version

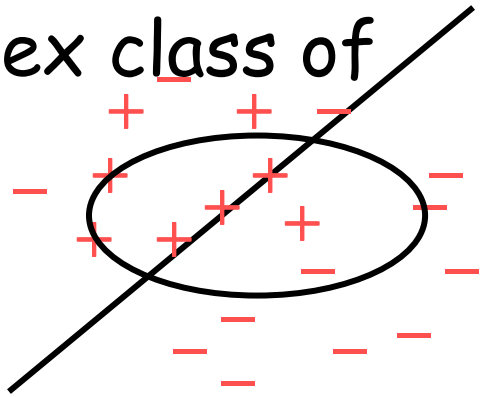
- Suppose we are given a set of images   , and want to learn a rule to distinguish men from women. Problem: pixel representation not so good.
- A powerful technique for such settings is to use a **kernel**: a special kind of pairwise function **K(**  ,  **)**.
- In practice, choose K to be good measure of similarity, but theory in terms of implicit mappings.

Develop a kind of PAC model for clustering.

Part 1: On similarity functions for learning

Kernel functions and Learning

- Back to our generic classification problem. E.g., given a set of images   , labeled by gender, learn a rule to distinguish men from women. [Goal: do well on new data]
- Problem: our best algorithms learn linear separators, but might not be good for data in its natural representation.
  - Old approach: use a more complex class of functions.
  - New approach: use a kernel.



What's a kernel?

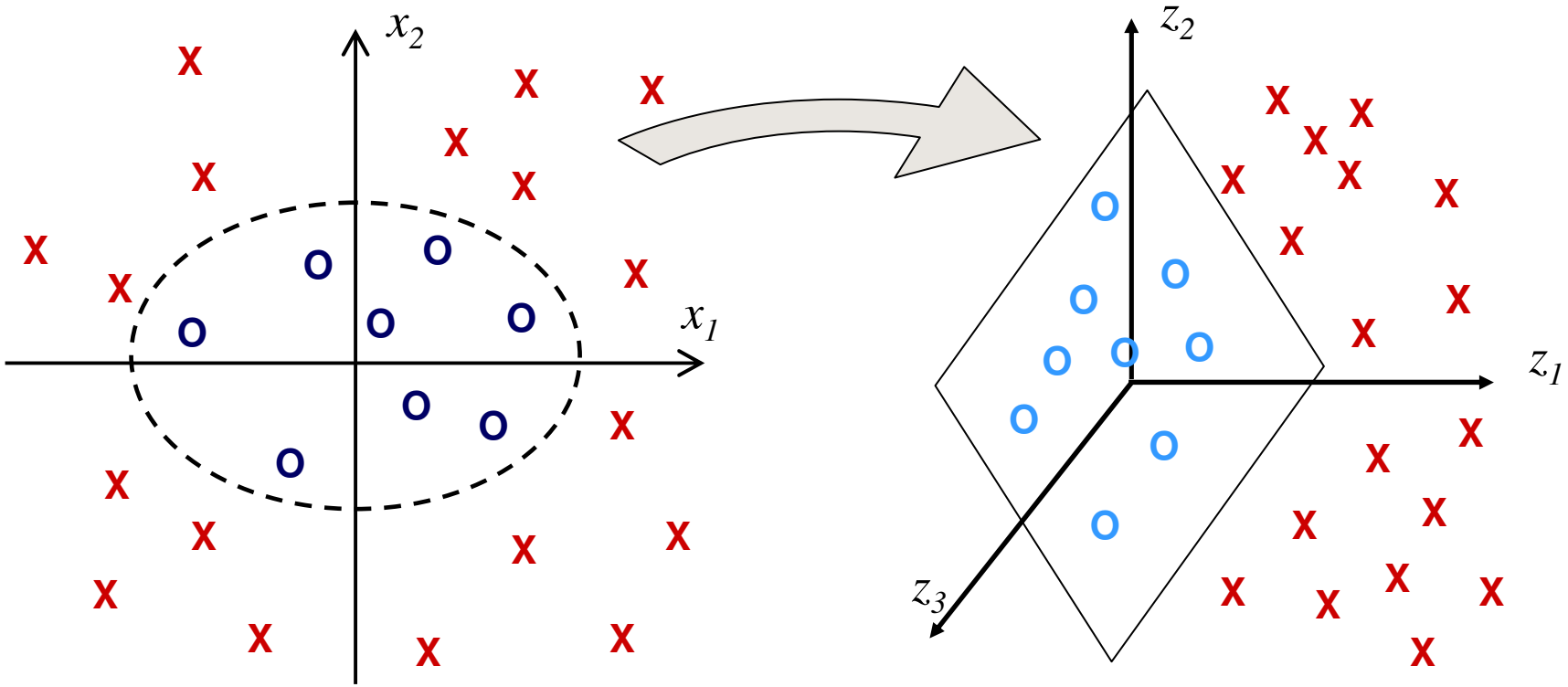
- A kernel K is a legal def of dot-product: fn s.t. there exists an implicit mapping Φ_K such that K( , )=Φ_K()¢Φ_K().
- E.g., K(x,y) = (x ¢ y + 1)^d.
  - Φ_K:(n-diml space) ! (n^d-diml space).
- Point is: many learning algs can be written so only interact with data via dot-products.
  - If replace x¢y with K(x,y), it acts implicitly as if data was in higher-dimensional Φ-space.

Kernel should be pos. semi-definite (PSD)

Example

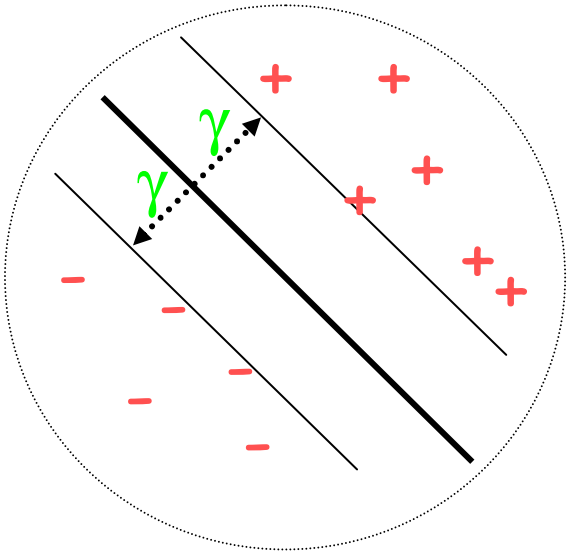
- E.g., for the case of n=2, d=2, the kernel $K(x,y) = (1 + x¢y)^d$ corresponds to the mapping:

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$



Moreover, generalize well if good margin

- If data is lin. separable by margin γ in Φ-space, then need sample size only $\tilde{O}(1/\gamma^2)$ to get confidence in generalization.



Assume $|\Phi(x)| \cdot 1$.

- Kernels found to be useful in practice for dealing with many, many different kinds of data.

Moreover, generalize well if good margin

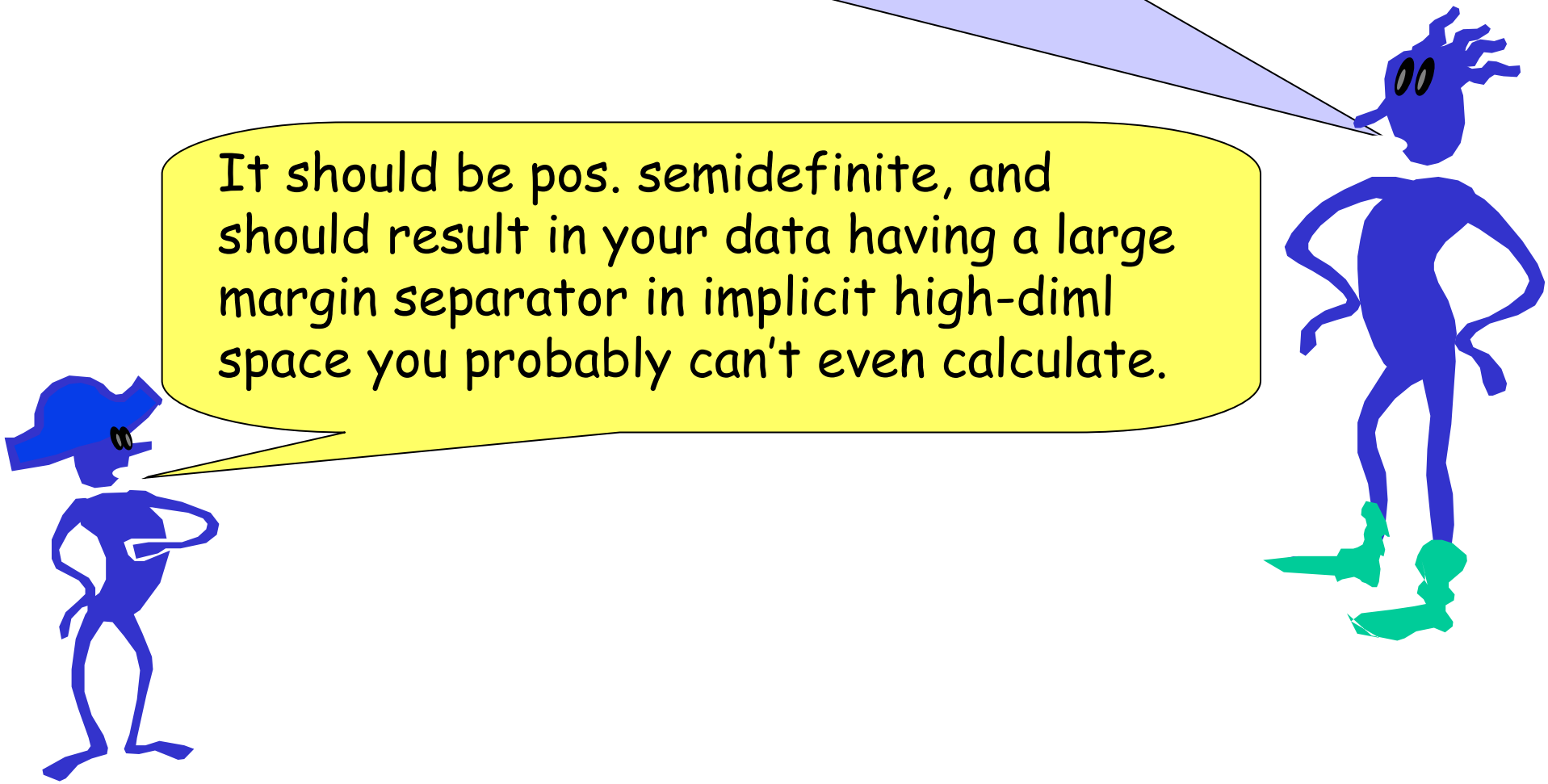
…but there's something a little funny:

- On the one hand, operationally a kernel is just a similarity measure: $K(x,y) \in [-1,1]$, with some extra reqts.

- But Theory talks about margins in implicit high-dimensional Φ-space. $K(x,y) = \Phi(x) \cdot \Phi(y)$.

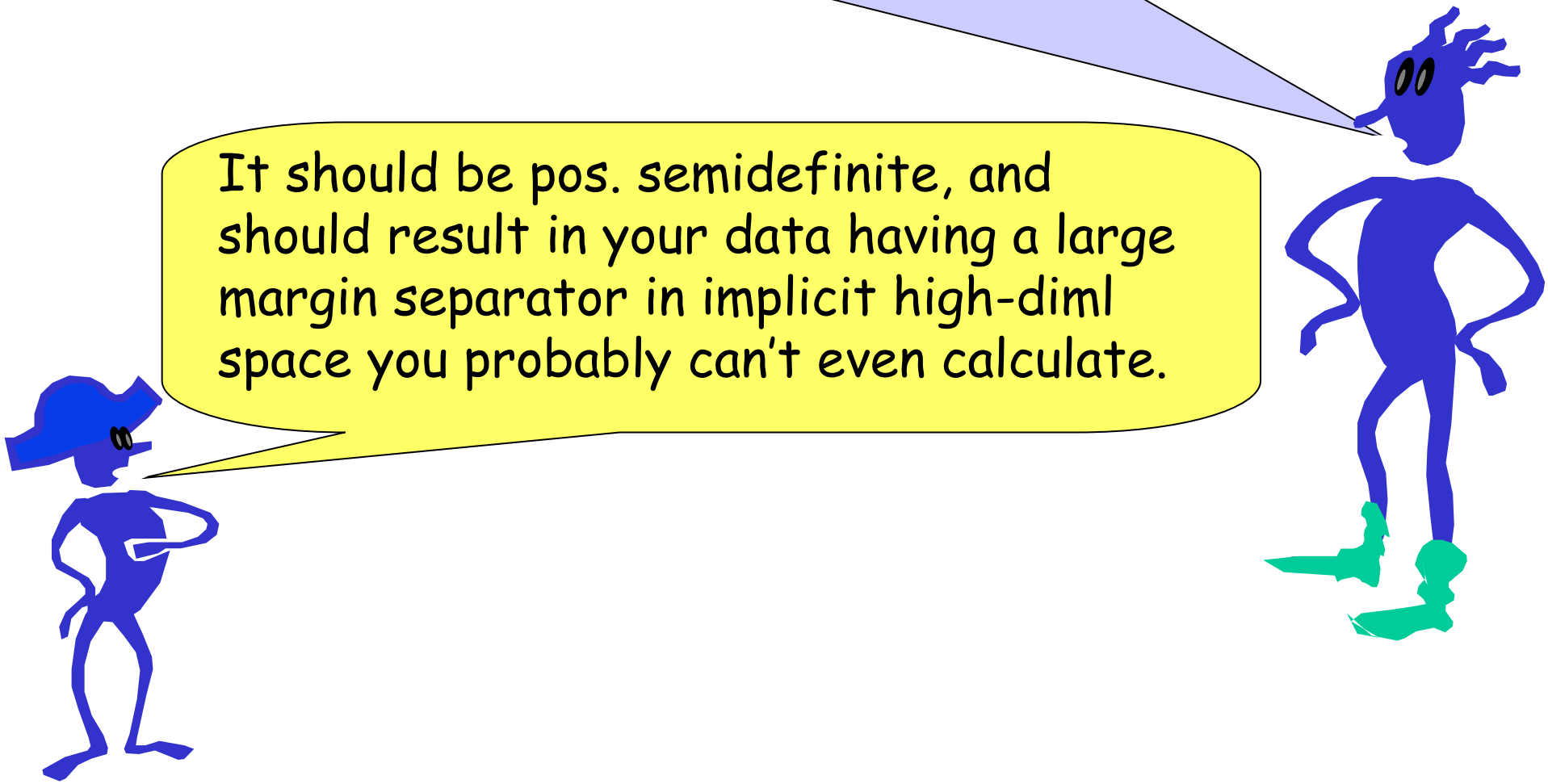
I want to use ML to classify protein structures and I'm trying to decide on a similarity fn to use. Any help?

It should be pos. semidefinite, and should result in your data having a large margin separator in implicit high-diml space you probably can't even calculate.



Umm… thanks, I guess.

It should be pos. semidefinite, and should result in your data having a large margin separator in implicit high-diml space you probably can't even calculate.



Moreover, generalize well if good margin

…but there's something a little funny:

- On the one hand, operationally a kernel is just a similarity function: $K(x,y) \in [-1,1]$, with some extra reqts.

- But Theory talks about margins in implicit high-dimensional Φ-space. $K(x,y) = \Phi(x) \cdot \Phi(y)$.
  - Can we bridge the gap?
  - Standard theory has a something-for-nothing feel to it. "All the power of the high-dim'l implicit space without having to pay for it". More prosaic explanation?

Question: do we need the notion of an implicit space to understand what makes a kernel helpful for learning?

Goal: notion of "good similarity function" for a learning problem that…

1. Talks in terms of more intuitive properties (no implicit high-diml spaces, no requirement of positive-semidefiniteness, etc)
   - E.g., natural properties of weighted graph induced by K.
2. If K satisfies these properties for our given problem, then has implications to learning

3. Is broad: includes usual notion of "good kernel" (one that induces a large margin separator in Φ-space).

Defn satisfying (1) and (2):

- Say have a learning problem P (distribution D over examples labeled by unknown target f).
- Sim fn $K:(x,y)![-1,1]$ is (ε,γ)-good for P if at least a $1-\varepsilon$ fraction of examples x satisfy:

$$E_{y\sim D}[K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

"most x are on average more similar to points y of their own type than to points y of the other type"

Defn satisfying (1) and (2):

- Say have a learning problem P (distribution D over examples labeled by unknown target f).
- Sim fn K:(x,y)![-1,1] is (ε,γ)-good for P if at least a $1-\varepsilon$ fraction of examples x satisfy:

$$E_{y\sim D}[K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

- Note: it's possible to satisfy this and not even be a valid kernel.
- E.g., K(x,y) = 0.2 within each class, uniform random in {-1,1} between classes.

Defn satisfying (1) and (2):

- Say have a learning problem P (distribution D over examples labeled by unknown target f).
- Sim fn K:(x,y)![-1,1] is (ε,γ)-good for P if at least a $1-\varepsilon$ fraction of examples x satisfy:

$$E_{y\sim D}[K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

How can we use it?

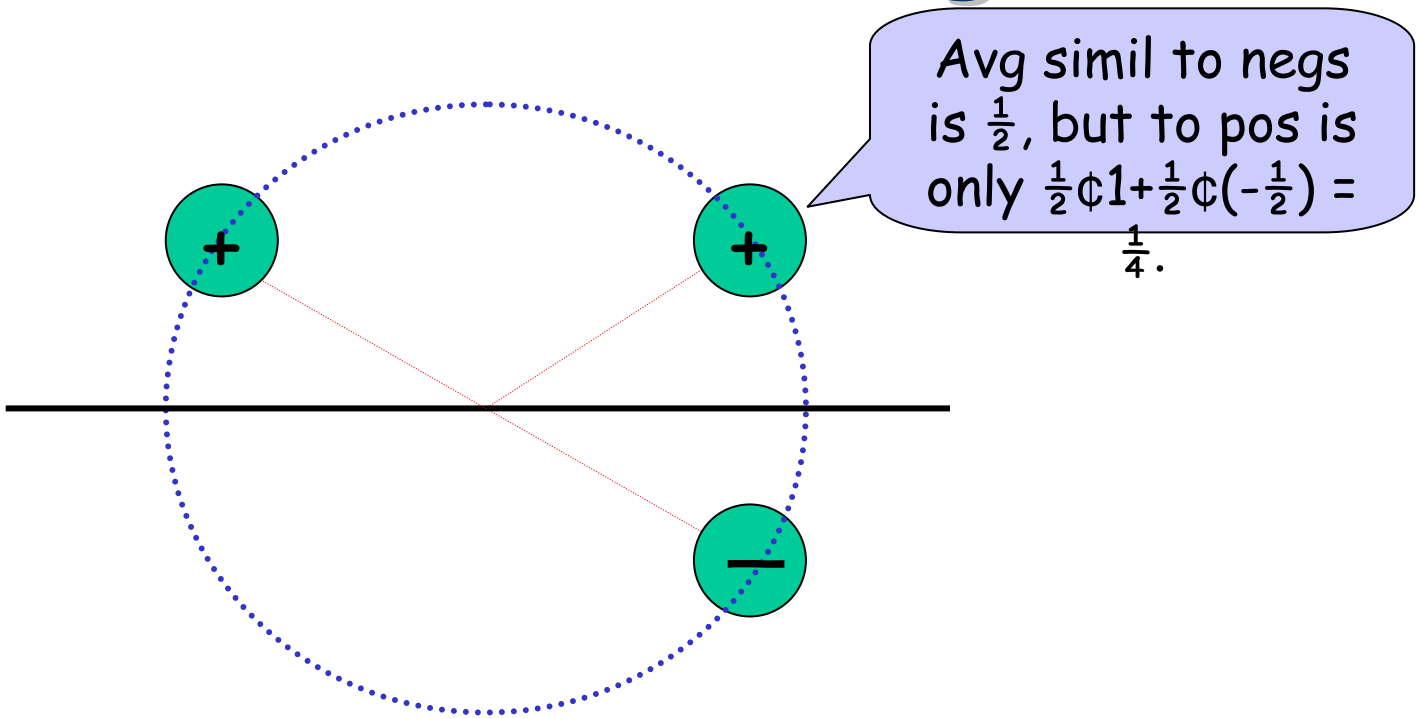
How to use it

At least a $1-\varepsilon$ prob mass of x satisfy:

$$E_{y\sim D}[K(x,y)|\ell(y)=\ell(x)] \geq E_{y\sim D}[K(x,y)|\ell(y)\neq\ell(x)]+\gamma$$

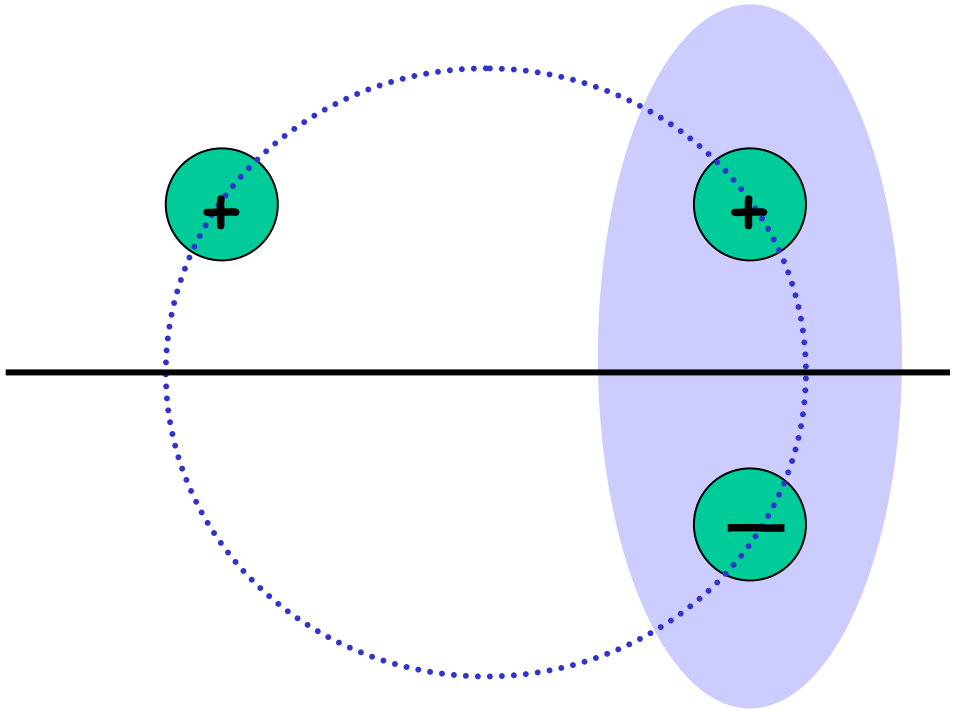
- Draw sets S^+, S^- of positive/negative points
- Classify x based on which gives better score.
  - Hoeffding: if $|S^+|, |S^-| = O((1/\gamma^2)\ln 1/\delta^2)$, for any given "good x", prob of error over draw of S^+, S^- at most δ^2.
  - So, at most δ chance our draw is bad on more than δ fraction of "good x".
- With prob $\geq 1-\delta$, error rate $\leq \varepsilon + \delta$.

But not broad enough



- K(x,y)=x¢y has good separator but doesn't satisfy defn. (half of positives are more similar to negs that to typical pos)

But not broad enough



- Idea: would work if we didn't pick y's from top-left.
- Broaden to say: OK if 9 large region R s.t. most x are on average more similar to y2R of same label than to y2R of other label. (even if don't know R in advance)

Broader defn…

- Ask that exists a set R of "reasonable" y (allow probabilistic) s.t. almost all x satisfy

$$E_y[K(x,y)|\ell(y)=\ell(x), R(y)] \geq E_y[K(x,y)|\ell(y)\neq\ell(x), R(y)]+\gamma$$

- And at least ε probability mass of reasonable positives/negatives.
- **Claim 1**: this is a legitimate way to think about good kernels:
  - If γ-good kernel then $(\varepsilon,\gamma^2,\varepsilon)$-good here.
  - If γ-good here and PSD then γ-good kernel

Broader defn…

- Ask that exists a set R of "reasonable" y (allow probabilistic) s.t. almost all x satisfy

$$E_y[K(x,y)|\ell(y)=\ell(x), R(y)] \geq E_y[K(x,y)|\ell(y)\neq\ell(x), R(y)]+\gamma$$

- And at least ε probability mass of reasonable positives/negatives.
- **Claim 2:** even if not PSD, can still use for learning.
  - So, don't need to have implicit-space interpretation to be useful for learning.
  - But, maybe not with SVMs directly…

How to use such a sim fn?

- Ask that exists a set R of "reasonable" y (allow probabilistic) s.t. almost all x satisfy

$$E_y[K(x,y)|\ell(y)=\ell(x), R(y)] \geq E_y[K(x,y)|\ell(y)\neq\ell(x), R(y)]+\gamma$$

  - Draw $S = \{y_1,\ldots,y_n\}$, $n \approx 1/(\gamma^2\varepsilon)$. could be unlabeled
  - View as "landmarks", use to map new data:

$$F(x) = [K(x,y_1), \ldots, K(x,y_n)].$$

  - Whp, exists separator of good L_1 margin in this space: $w=[0,0,1/n_+,1/n_+,0,0,0,-1/n_-,0,0]$
  - So, take new set of examples, project to this space, and run good linear sep. alg.

And furthermore

Now, defn is broad enough to include all large margin kernels (some loss in parameters):

- γ-good margin) apx $(\varepsilon,\gamma^2,\varepsilon)$-good here.

But now, we don't need to think about implicit spaces or require kernel to even have the implicit space interpretation.

If PSD, can also show reverse too:

- γ-good here & PSD) γ-good margin.

And furthermore

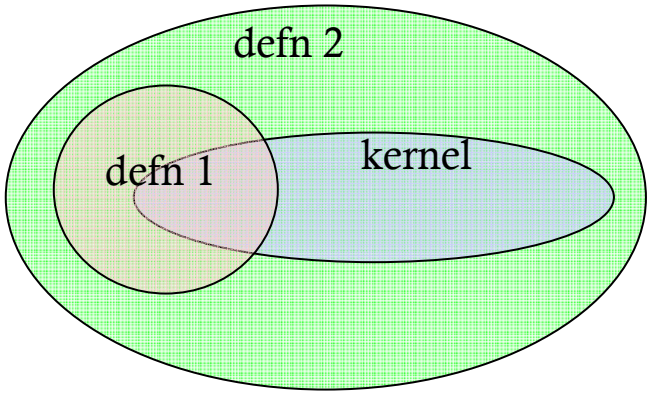
In fact, can even show a separation.

- Consider a class C of n pairwise uncorrelated functions over n examples (unif distrib).
- Can show that for any kernel K, expected margin for random f in C would be $O(1/n^{1/2})$.
- But, can define a similarity function with $\gamma=1$, $P(R)=1/n$. $[K(x_i,x_j)=f_j(x_i)f_j(x_j)]$

technically, easier using slight variant on def: $E_y[K(x,y)\ell(x)\ell(y) \mid R(y)]_3 \gamma$

Summary: part 1

- Can develop sufficient conditions for a similarity fn to be useful for learning that don't require notion of implicit spaces.
- Property includes usual notion of "good kernels" modulo the loss in some parameters.
  - Can apply to similarity fns that aren't positive-semidefinite (or even symmetric).

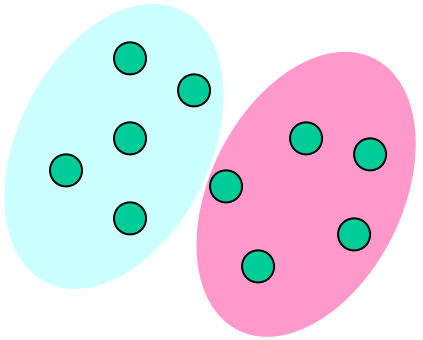


Summary: part 1

- Potentially other interesting sufficient conditions too. E.g., [WangYangFeng07] motivated by boosting.
- Ideally, these more intuitive conditions can help guide the design of similarity fns for a given application.

Part 2: Can we use this angle to help think about clustering?

Can we use this angle to help think about clustering?



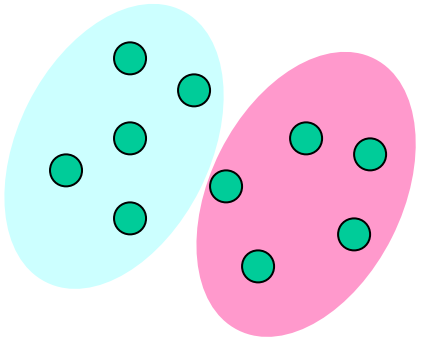
Consider the following setting:

- Given data set S of n objects. [documents, web pages]
- There is some (unknown) "ground truth" clustering. Each x has true label $\ell(x)$ in $\{1,\ldots,t\}$. [topic]
- Goal: produce hypothesis h of low error up to isomorphism of label names.

Problem: only have unlabeled data!

But, we are given a pairwise similarity fn K.

What conditions on a similarity function would be enough to allow one to cluster well?



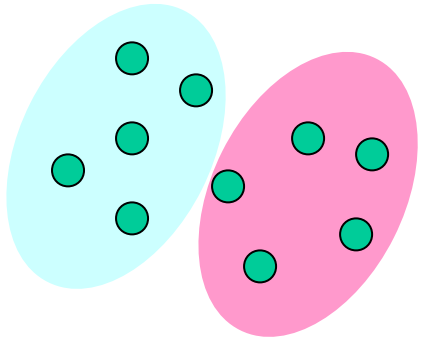
Consider the following setting:

- Given data set S of n objects. [documents, web pages]
- There is some (unknown) "ground truth" clustering. Each x has true label $\ell(x)$ in $\{1,\ldots,t\}$. [topic]
- Goal: produce hypothesis h of low error up to isomorphism of label names.

Problem: only have unlabeled data!

But, we are given a pairwise similarity fn K.

What conditions on a similarity function would be enough to allow one to cluster well?

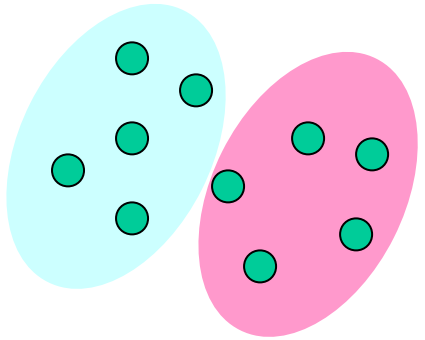


Will lead to something like a PAC model for clustering.

Contrast to approx algs approach: view weighted graph induced by K as ground truth; try to optimize various objectives.

Here, we view target as ground truth. Ask: how should K be related to let us get at it?

What conditions on a similarity function would be enough to allow one to cluster well?



Will lead to something like a PAC model for clustering.

E.g., say you want alg to cluster docs the way *you* would. How closely related does K have to be to what's in your head? Or, given a property you think K has, what algs does that suggest?

What conditions on a similarity function would be enough to allow one to cluster well?

Here is a condition that trivially works:

Suppose K has property that:

- $K(x,y) > 0$ for all x,y such that $\ell(x) = \ell(y)$.
- $K(x,y) < 0$ for all x,y such that $\ell(x) \neq \ell(y)$.

If we have such a K, then clustering is easy.

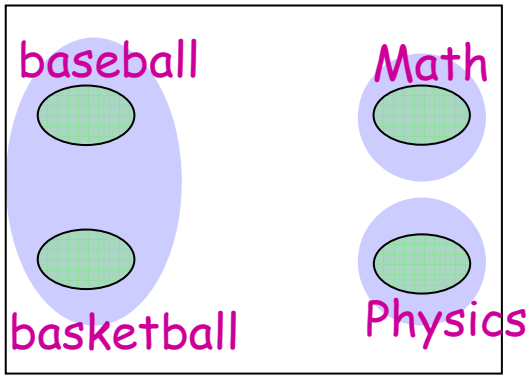
Now, let's try to make this condition a little weaker….

What conditions on a similarity function would be enough to allow one to cluster well?

Suppose K has property that all x are more similar to points y in their own cluster than to any y′ in other clusters.

- Still a very strong condition.

Problem: the same K can satisfy for two very different clusterings of the same data!

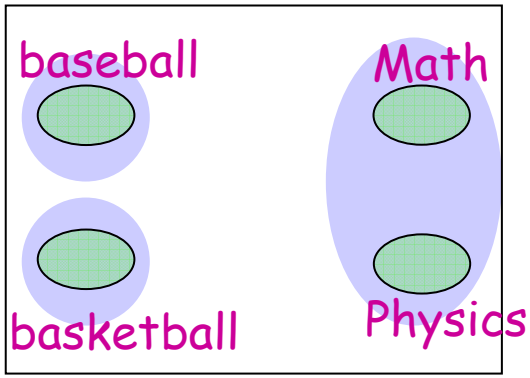


What conditions on a similarity function would be enough to allow one to cluster well?

Suppose K has property that all x are more similar to points y in their own cluster than to any y′ in other clusters.

- Still a very strong condition.

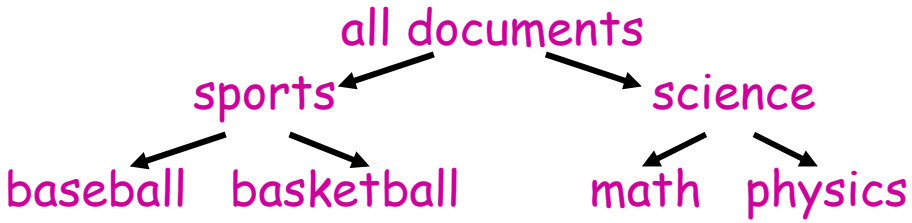
Problem: the same K can satisfy for two very different clusterings of the same data!

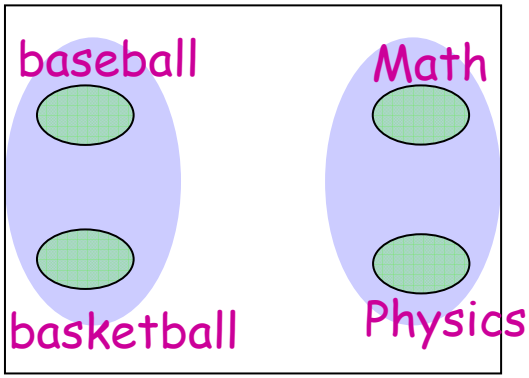


Unlike learning, you can't even test your hypotheses!

Let's weaken our goals a bit…

- OK to produce a hierarchical clustering (tree) such that correct answer is apx some pruning of it.
  - E.g., in case from last slide:





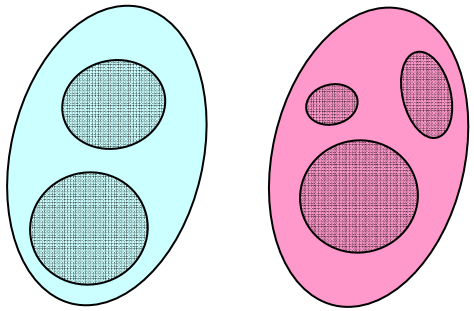
- We'll let it be the user's job to decide how specific they intended to be.

Then you can start getting somewhere….

1. "all x more similar to all y in their own cluster than to any y' from any other cluster"

is sufficient to get hierarchical clustering such that target is some pruning of tree. (Kruskal's / single-linkage works)

- Just find most similar pair (x,y) that are in different clusters and merge their clusters together.



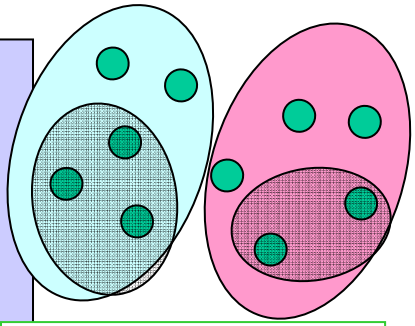
Then you can start getting somewhere….

1. "all x more similar to all y in their own cluster than to any y′ from any other cluster"

is sufficient to get hierarchical clustering such that target is some pruning of tree. (Kruskal's / single-linkage works)

2. Weaker condition: ground truth is "stable":

For all clusters C, C′, for all A½C, A′½C′: A and A′ are not both more similar to each other than to rest of their own clusters.



K(x,y) is attraction between x and y

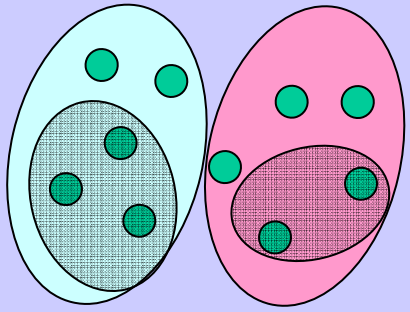
Weaker reqt: correct ans is stable

Assume for all C, C', all $A \subset C$, $A' \subset C'$, we have

$$K(A, C-A) > K(A, A'),$$

$\text{Avg}_{x \in A,\, y \in C-A}[K(x,y)]$

and say K is symmetric.



Algorithm: average single-linkage

- Like Kruskal, but at each step merge pair of clusters whose average similarity is highest.

Analysis: (all clusters made are laminar wrt target)

It's getting late, let's skip the proof

Example analysis for simpler version

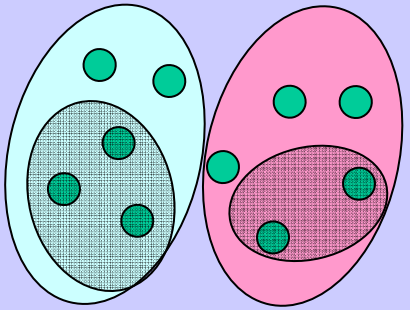
Assume for all C, C', all $A \subset C$, $A' \subset C'$, we have

$K(A, C-A) > K(A, A')$,

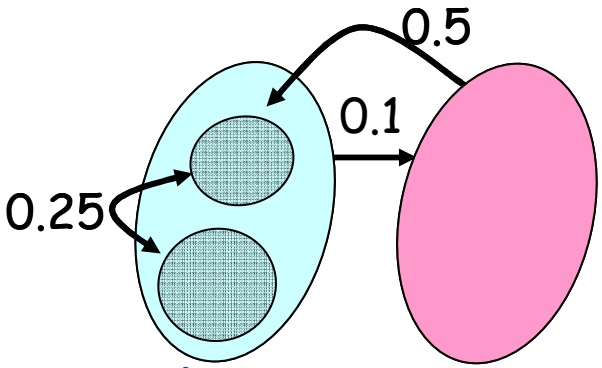
[Think of K as "attraction"]

$\text{Avg}_{x \in A, y \in C-A}[K(x,y)]$

~~and say K is symmetric.~~



Algorithm breaks down if K is not symmetric:



Instead, run "Boruvka-inspired" algorithm:

- Each current cluster C_i points to $\text{argmax}_{C_j} K(C_i, C_j)$
- Merge directed cycles. (not all components)

Properties Summary

Property	**Model, Algorithm**	**Clustering Complexity**
Strict Separation	Hierarchical, Linkage based	$\Theta(2^t)$
Stability, all subsets. (Weak, Strong, etc)	Hierarchical, Linkage based	$\Theta(2^t)$
Average Attraction (Weighted)	List, Sampling based & NN	$t^{O(t/\gamma^2)}$
Stability of large subsets (SLS)	Hierarchical, complex algorithm (running time $t^{O(t/\gamma^2)}$)	$\Theta(2^t)$
ν–strict separation	Hierarchical	$\Theta(2^t)$
$(2,\varepsilon)$ k-median	special case of ν–strict separation, $\nu=3\varepsilon$	

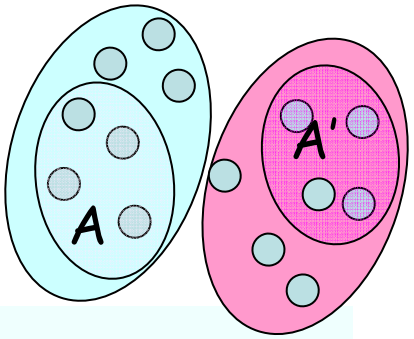
Properties Summary

Property	**Model, Algorithm**	**Clustering Complexity**
Strict Separation	Hierarchical, Linkage based	$\Theta(2^t)$
Stability, all subsets. (Weak, Strong, etc)	Hierarchical, Linkage based	$\Theta(2^t)$
Average Attraction (Weighted)	List, Sampling based & NN	$t^{O(t/\gamma^2)}$
Stability of large subsets (SLS)	Hierarchical, complex algorithm (running time $t^{O(t/\gamma^2)}$)	$\Theta(2^t)$
ν–strict separation	Hierarchical	$\Theta(2^t)$
$(2,\varepsilon)$ k-median	special case of ν–strict separation, $\nu=3\varepsilon$	

Stability of Large Subsets

For all C, C', all $A \subset C$, $A' \subset C'$, $|A|+|A'| \geq sn$

$K(A, C-A) > K(A, A')+\gamma$



Algorithm

1) Generate list L of candidate clusters; average attraction alg.

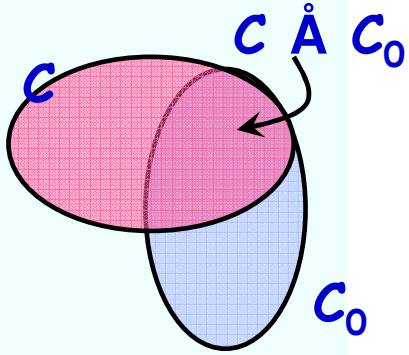
   Ensure that any ground-truth cluster is f-close to one in L.

2) For every pair C, C_0 in L that are

   $|C \setminus C_0| \geq gn$, $|C_0 \setminus C| \geq gn$, $|C_0 \cap C| \geq gn$

   If $K(C \cap C_0, C \setminus C_0) \geq K(C \cap C_0, C_0 \setminus C)$, throw out C

   Else throw out C_0



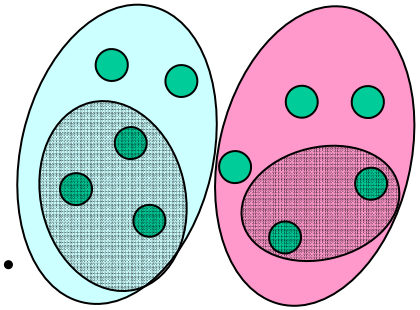
3) Clean and hook up the surviving clusters into a tree

More general conditions

What if only require stability for large sets?

(Assume all true clusters are large.)

- E.g, take example satisfying stability for all sets but add noise.



- Might cause bottom-up algorithms to fail.

Instead, can pick some points at random, guess their labels, and use to cluster the rest. Produces big list of candidates.

Then 2nd testing step to hook up clusters into a tree. Running time not great though. (exponential in # topics)

Like a PAC model for clustering

- PAC learning model: basic object of study is the concept class (a set of functions). Look at which are learnable and by what algs.
- In our case, basic object of study is a property: a relation between target and similarity function. Want to know which allow clustering and by what algs.

Other properties

- Can also relate to implicit assumptions made by approx algorithms for standard objectives like k-median, k-means.
  - E.g., if you assume that any apx k-median solution must be close to the target, this implies that most points satisfy simple ordering condition.

Conclusions

What properties of a similarity function are sufficient for it to be useful for **clustering**?

- View as unlabeled-data multiclass learning prob. (Target fn as ground truth rather than graph)
- To get interesting theory, need to relax what we mean by "useful".
- Can view as a kind of PAC model for clustering.
- A lot of interesting directions to explore.

Conclusions

- Natural properties (relations between sim fn and target) that motivate spectral methods?
- Efficient algorithms for other properties? E.g., "stability of large subsets"
- Other notions of "useful".
  - Produce a small DAG instead of a tree?
  - Others based on different kinds of feedback?
- A lot of interesting directions to explore.

Some books/references:

- *Algorithmic Game Theory*, Nisan, Roughgarden, Tardos, Vazirani (eds), Cambridge Univ Press, 2007. [Chapter 4 "Learning, Regret Minimization, & Equilibria" is on my webpage]
- *Prediction, Learning, & Games*, Cesa-Bianchi & Lugosi, Cambridge Univ Press, 2006.
- My course notes: www.machinelearning.com
  - Stop 1: Online learning, minimizing regret, and combining expert advice.
  - Stop 2: Game theory, minimax optimality, and Nash equilibria.
  - Stop 3: Correlated equilibria, internal regret, routing games, and connections between 1 and 2.
  - Stop 4: (something completely different) Learning and clustering with similarity functions.