Online Learning, Regret Minimization, Minimax Optimality, and Correlated Equilibrium

Avrim Blum

High level

Last time we discussed notion of Nash equilibrium.

- Static concept: set of prob. Distributions (p,q,…) such that nobody has any incentive to deviate.
- But doesn't talk about how system would get there. Troubling that even finding one can be hard in large games.

What if agents adapt (learn) in ways that are well-motivated in terms of their own rewards? What can we say about the system then?

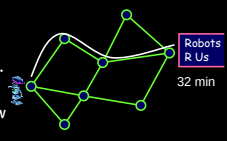
High level

Today:

- Online learning guarantees that are achievable when acting in a changing and unpredictable environment.
- What happens when two players in a zero-sum game both use such strategies?
  - Approach minimax optimality.
  - Gives alternative proof of minimax theorem.
- What happens in a general-sum game?
  - Approaches (the set of) correlated equilibria.

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.

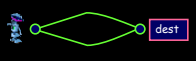
Define **average regret** in T time steps as:
(avg per-day cost of alg) – (avg per-day cost of best fixed row in hindsight).

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:



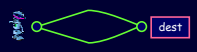
Algorithm \ World – life - fate	
1	0
0	1

- Note: Not trying to compete with best adaptive strategy – just best fixed row (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 see in a bit)

History and development (abridged)

- [Hannan'57, Blackwell'56]: Alg with avg 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 – $\Theta(\sqrt{T})$.
- So, expected avg per-day gap is $\Theta(1/\sqrt{T})$.

History and development (abridged)

- [Hannan'57, Blackwell'56]: Alg with avg 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".

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
…	…	…	…	…

Basic question: Is there a strategy that allows us to do nearly as well as best of these 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"

What if no expert is perfect?

One idea: just run above protocol until all experts are crossed off, then repeat.

Makes at most log(n) mistakes per mistake of the best expert (plus initial log(n)).

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.

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

So, if m is small, then M is pretty small too.

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.
  - $W_{final} = n(1-\varepsilon F_1)(1 - \varepsilon F_2)...$
  - $\ln(W_{final}) = \ln(n) + \sum_t [\ln(1 - \varepsilon F_t)] \leq \ln(n) - \varepsilon \sum_t F_t$ (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] $\leq (1+\varepsilon)m + \varepsilon^{-1}\log(n)$.
- If set $\varepsilon = (\log(n)/m)^{1/2}$ to balance the two terms out (or use guess-and-double), get bound of
  $E[\text{mistakes}] \leq m + 2(m \cdot \log n)^{1/2}$
- Since $m \leq T$, this is at most $m + 2(T\log n)^{1/2}$.
- So, avg regret = $2(T\log n)^{1/2}/T \rightarrow 0$.

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

What can we use this for?

- Can use to combine multiple algorithms to do nearly as well as best in hindsight.
- But what about cases like choosing paths to work, where "experts" are different actions, not different predictions?

Game-theoretic version

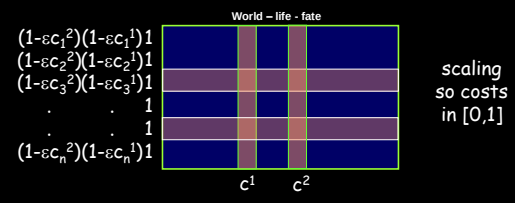
- What if experts are actions? (paths in a network, rows in a matrix game,…)
- At each time t, each has a loss (cost) in {0,1}.
- Can still run the algorithm
  - Rather than viewing as "pick a prediction with prob proportional to its weight" ,
  - View as "pick an expert with probability proportional to its weight"
  - Choose expert i with probability $p_i = w_i/\sum_i w_i$.
- Same analysis applies.

Game-theoretic version

- What if experts are actions? (paths in a network, rows in a matrix game,…)
- What if losses (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)

Illustration



Guarantee: $E[\text{cost}] \leq OPT + 2(OPT \cdot \log n)^{1/2}$

Since $OPT \leq T$, this is at most $OPT + 2(T \log n)^{1/2}$.

So, regret/time step $\leq 2(T \log n)^{1/2}/T \rightarrow 0$.

Connections to Minimax Optimality

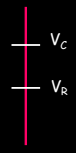
Minimax-optimal strategies

- Can solve for minimax-optimal strategies using Linear programming
- Claim: no-regret strategies will do nearly as well or better against any sequence of opponent plays.
  - Do nearly as well as best fixed choice in hindsight.
  - Implies do nearly as well as best distrib in hindsight
  - Implies do nearly as well as minimax optimal!

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

Proof of minimax thm using RWM

- 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 $\geq$ [best row in hindsight] – $2(T\log n)^{1/2}$
  - 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 once $\delta T > 2(T\log n)^{1/2}$, or $T > 4\log(n)/\delta^2$.

Proof contd

- Now, consider playing randomized weighted-majority alg as Row, against Col who plays optimally against Row's distrib.
- Note that our procedure gives a fast way to compute apx minimax-optimal strategies, if we can simulate Col (best-response) quickly.

What if two RWMs play each other?

- Can anyone see the argument that their time-average strategies must be approaching minimax optimality?

Internal/Swap Regret and Correlated Equilibria

What if all players minimize regret?

- 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. So *if the distributions stabilize*, they must converge to a Nash equil.
- Well, unfortunately, no.

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.

What are internal/swap regret and correlated equilibria?

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, swap regret is wrt optimal function $f:\{1,\dots,n\}\to\{1,\dots,n\}$ such that every time you played action j, it plays f(j).

Weird… why care?

"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

In general-sum games, if all players have low swap-regret, then empirical distribution of play is apx correlated equilibrium.

Connection

- If all parties run a low swap regret algorithm, then empirical distribution of play is an apx correlated equilibrium.
  - Correlator chooses random time $t \in \{1,2,\dots,T\}$. 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 suggests correlated equilibria may be natural things to see in multi-agent systems where individuals are optimizing for themselves

Correlated vs Coarse-correlated Eq

In both cases: a distribution over entries in the matrix. Think of a third party choosing from this distr and telling you your part as "advice".

"Correlated equilibrium"

- You have no incentive to deviate, even after seeing what the advice is.

"Coarse-Correlated equilibrium"

- If only choice is to see and follow, or not to see at all, would prefer the former.

Low external-regret $\Rightarrow$ apx coarse correlated equilib.

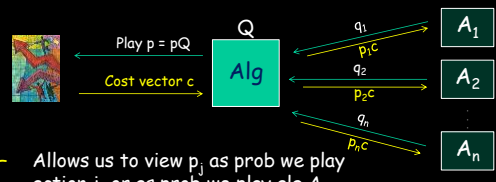
Internal/swap-regret, contd

Algorithms for achieving low regret of this form:

- Foster & Vohra, Hart & Mas-Colell, Fudenberg & Levine.
- Will present method of [BM05] showing how to convert any "best expert" algorithm into one achieving low swap regret.
- Unfortunately, #steps to achieve low swap regret is O(n log n) rather than O(log n).

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

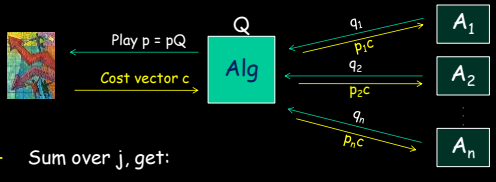
- Instantiate one copy A_j responsible for expected regret over times we play j.



- Allows us to view p_j as prob we play action j, or as prob we play alg A_j.
- Give A_j feedback of $p_j c$.
- A_j guarantees $\sum_t (p_j^t c^t) \cdot q_j^t \le \min_i \sum_t p_j^t c_i^t$ + [regret term]
- Write as: $\sum_t p_j^t (q_j^t \cdot c^t) \le \min_i \sum_t p_j^t c_i^t$ + [regret term]

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

- Instantiate one copy A_j responsible for expected regret over times we play j.



- Sum over j, get:

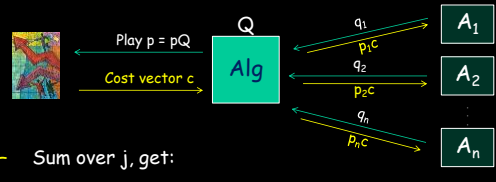
$$\sum_t p^t Q^t c^t \le \sum_j \min_i \sum_t p_j^t c_i^t + n[\text{regret term}]$$

Our total cost | For each j, can move our prob to its own i=f(j)

- Write as: $\sum_t p_j^t (q_j^t \cdot c^t) \le \min_i \sum_t p_j^t c_i^t$ + [regret term]

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

- Instantiate one copy A_j responsible for expected regret over times we play j.



- Sum over j, get:

$$\sum_t p^t Q^t c^t \le \sum_j \min_i \sum_t p_j^t c_i^t + n[\text{regret term}]$$

Our total cost | For each j, can move our prob to its own i=f(j)

- Get swap-regret at most n times orig external regret.