Online Learning

Your guide:

Avrim Blum
Carnegie Mellon University

[Machine Learning Summer School 2012]

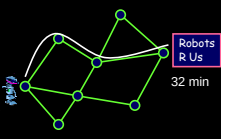
Itinerary

- Stop 1: Minimizing regret and combining advice.
  - Randomized Wtd Majority / Multiplicative Weights alg
  - Connections to game theory
- Stop 2: Extensions
  - Online learning from limited feedback (bandit algs)
  - Algorithms for large action spaces, sleeping experts
- Stop 3: Powerful online LTF algorithms
  - Winnow, Perceptron
- Stop 4: Powerful tools for using these algorithms
  - Kernels and Similarity functions
- Stop 5: Something completely different
  - Distributed machine learning

Stop 1: 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.
  - 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.

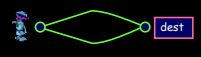
Define **average regret** in T time steps as:
(avg per-day cost of alg) - (avg per-day cost of best fixed row in hindsight).

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

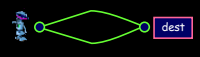
- Note: Not trying to compete with best adaptive strategy – just best fixed path in hindsight.
- No-regret algorithms can do much better than playing minimax optimal, and never much worse.
- Existence of no-regret algs yields immediate proof of minimax thm!

Will define this later

This too

Some intuition & properties of no-regret algs.

- Let's look at a small example:



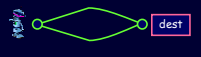
Algorithm \ World – life - fate		
	1	0
	0	1

- 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. (World as adversary)

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 $f \in C$.
  - [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.

					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

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 ½ 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 $\ln(1-x) < -x$)
  - $= \ln(n) - \varepsilon M.$ ($\sum F_t = E[\#\text{ mistakes}]$)
- 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, regret $\rightarrow 0$.

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

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?

Extensions

- 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.

Extensions

- 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)

Connections to 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 gooooooaaaaal!
- Vice-versa for shooter.

2-Player Zero-Sum games

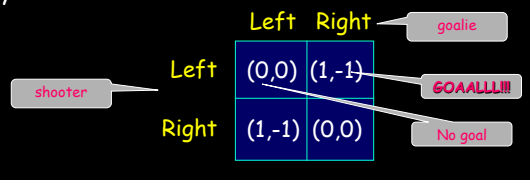
- Two players R and C. Zero-sum means that what's good for one is bad for the other.
- Game defined by matrix with a row for each of R's options and a column for each of C's options. Matrix tells who wins how much.
  - an entry (x,y) means: x = payoff to row player, y = payoff to column player. "Zero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

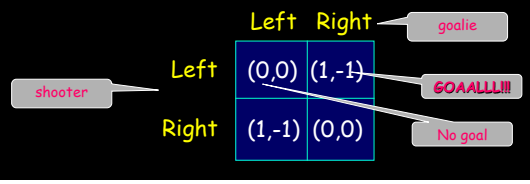
- Minimax optimal strategy is a (randomized) strategy that has the best guarantee on its expected gain, over choices of the opponent. [maximizes the minimum]
- I.e., the thing to play if your opponent knows you well.



Minimax-optimal strategies

- What are the minimax optimal strategies for this game?

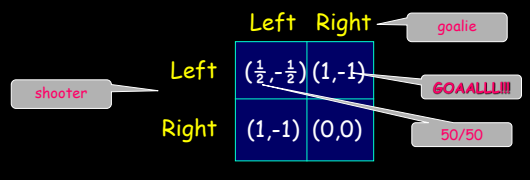
Minimax optimal strategy for both players is 50/50. Gives expected gain of $\frac{1}{2}$ for shooter ($-\frac{1}{2}$ for goalie). Any other is worse.



Minimax-optimal strategies

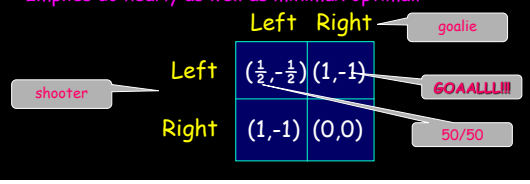
- How about penalty shot with goalie who's weaker on the left?

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-optimal strategies

- Can solve for minimax-optimal strategies using Linear programming
- 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!



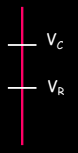
Minimax Theorem (von Neumann 1928)

- 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)

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 $T > 4\log(n)/\delta^2$.

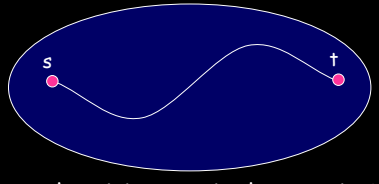
Proof contd

- 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.

Interesting game

"Smuggler vs border guard"

- Graph G, source s, sink t. Smuggler chooses path. Border guard chooses edge to watch.
- If edge is in path, guard wins, else smuggler wins.

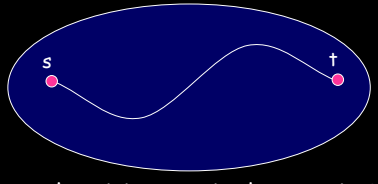


- What are the minimax optimal strategies?

Interesting game

"Smuggler vs border guard"

- Border guard: find min cut, pick random edge in it.
- Smuggler: find max flow, scale to unit flow, induces prob dist on paths.

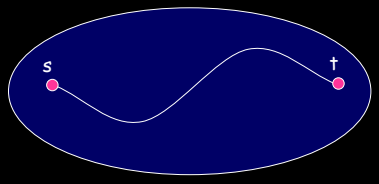


- What are the minimax optimal strategies?

Interesting game

Latest fast approximate max-flow algorithms based on applying RWM to variations on this game.

- Run RWM for border guard (experts = edges)
- Best-response = shortest path or linear system solve.



- What are the minimax optimal strategies?