Online Learning, Regret Minimization, and Minimax Optimality

01/17/13

Your guide:
Avrim Blum

[Readings: Ch. 4.1-4.3 of AGT book]

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?

Today:
- Fairly strong 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.

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.

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.
- 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 algns.

- Let's look at a small example:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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 $\varepsilon$. (will call this quantity $T_\varepsilon$)
  - Optimal dependence on $T$ (or $N$). Game theorists viewed $T$ as constant, not so important as $T$, so pretty much done.

Why optimal in $T$?

- 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 $O(1/T^{1/2})$.

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.,

<table>
<thead>
<tr>
<th>Expt 1</th>
<th>Expt 2</th>
<th>Expt 3</th>
<th>neighbor’s dog</th>
<th>truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>down</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>down</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>down</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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.

| weights | 1 | 1 | 1 |
| predictions | Y | Y | N | T | Y |

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,

\[
(1/2)^m \leq n(3/4)^M \\
(4/3)^M \leq n2^m
\]

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 \( 1 - \epsilon \).

\[
M \leq \frac{m \ln(1 - \epsilon) + \ln(n)}{\epsilon} \Rightarrow (1 + \epsilon/2)m + \frac{1}{\epsilon} \ln(n)
\]

Solves to:
\[
M \leq 1.35m + 2 \ln n + - \epsilon = 1/2
\]
\[
M \leq 1.15m + 4 \ln n + - \epsilon = 1/4
\]
\[
M \leq 1.07m + 8 \ln n + - \epsilon = 1/8
\]

Analysis

• Say at time \( t \) we have fraction \( F_t \) of weight on experts that made mistake.

• Instead of taking majority vote, use weights as probabilities (e.g. 70% up, 30% down, then pick 70:30). Idea: smooth out the worst case.

• Also, generalize \( \frac{1}{2} \) to \( 1 - \epsilon \).

\[
M \leq \frac{m \ln(1 - \epsilon) + \ln(n)}{\epsilon} \Rightarrow (1 + \epsilon/2)m + \frac{1}{\epsilon} \ln(n)
\]

Solves to:
\[
M \leq 1.35m + 2 \ln n + - \epsilon = 1/2
\]
\[
M \leq 1.15m + 4 \ln n + - \epsilon = 1/4
\]
\[
M \leq 1.07m + 8 \ln n + - \epsilon = 1/8
\]

• If best expert makes \( m \) mistakes, then \( \ln(W_{final}) > \ln((1-\epsilon)^m) \).

• Now solve: \( \ln(n) - \epsilon M > m \ln(1-\epsilon) \).

\[
M \leq \frac{-m \ln(1 - \epsilon) + \ln(n)}{\epsilon} \Rightarrow (1 + \epsilon/2)m + \frac{1}{\epsilon} \log(n)
\]
Summarizing

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

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 $p_i = w_i / \sum w_i$.
- Same analysis applies.

Guarantee:

- $E[\text{cost}] \leq \text{OPT} + 2(\text{OPT} \log n)^{1/2}$
- Since $\text{OPT} \leq T$, this is at most $\text{OPT} + 2(T \log n)^{1/2}$.
- So, regret/time step $\leq \frac{2(T \log n)^{1/2}}{T} \to 0$.

Illustration

Connections to Minimax Optimality

- 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: $w_i \leftarrow w_i (1-c_i)$.
- Our expected cost $= \sum c_i w_i / W$.
- Amount of weight removed $= \varepsilon \sum w_i c_i$.
- So, fraction removed $= \varepsilon \cdot (\text{our cost})$.
- Rest of proof continues as before...

So, now we can drive to work! (assuming full feedback)
**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!

<table>
<thead>
<tr>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>(½,-½)</td>
<td>(1,-1)</td>
</tr>
<tr>
<td>(1,-1)</td>
<td>(0,0)</td>
</tr>
</tbody>
</table>

**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(Tlog n)\(1/2\)
  - BRiH \( \geq T V_C \) [Best against opponent’s empirical distribution]
  - Alg \( \leq T V_R \) [Each time, opponent knows your randomized strategy]
  - Gap is \( \delta T \). Contradicts assumption once \( \delta T > 2(Tlog n)^{1/2} \), or \( T > 4\log(n)/\delta^2 \).

**What if two RWMs play each other?**

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

**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?