# 15-859(B) Machine Learning Theory

Bandit Problems and sleeping experts

Avrim Blum

---

## Start with recap

---

## "No-regret" algorithms for repeated decisions

General framework:

- 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 <u>average regret</u> 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]

---

## 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 $\varepsilon$). Game-theorists viewed #rows N as constant, not so important as T, so pretty much done.

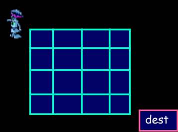
---

## 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 $\varepsilon$). 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
    - $E[\text{cost}] \leq OPT(1+\varepsilon) + (\log N)/\varepsilon$.
    - 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).

## 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…

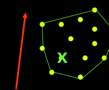



- This is what we discussed last time.

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

[KV] setting:

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




[Z] setting:

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

## Plan for today

- Bandit algorithms
- Sleeping experts
- But first, a quick discussion of [0,1] vs {0,1} costs for RWM algorithm

## [0,1] costs vs {0,1} costs.

We analyzed Randomized Wtd Majority for case that all costs in {0,1} (correct or mistake).

Here is a simple way to extend to [0,1].

- Given cost vector c, view $c_i$ as bias of coin. Flip to create boolean vector c', s.t. $E[c'_i] = c_i$. Feed c' to alg A.

world $\xrightarrow{c}$ \$ $\xrightarrow{c'}$ A (Cost' = cost on c' vectors)

- For any sequence of vectors c', we have:
  - $E_A[\text{cost}'(A)] \leq \min_i \text{cost}'(i) + [\text{regret term}]$
- So, $E_\$[E_A[\text{cost}'(A)]] \leq E_\$[\min_i \text{cost}'(i)] + [\text{regret term}]$
- LHS is $E_A[\text{cost}(A)]$. (since A picks weights before seeing costs)
- RHS $\leq \min_i E_\$[\text{cost}'(i)] + [\text{r.t.}] = \min_i[\text{cost}(i)] + [\text{r.t.}]$

In other words, costs between 0 and 1 just make the problem easier…

## Experts $\rightarrow$ Bandit setting

- In the bandit setting, only get feedback for the action we choose. Still want to compete with best action in hindsight.
- [ACFS02] give algorithm with cumulative regret $O((TN \log N)^{1/2})$. [average regret $O(((N \log N)/T)^{1/2})$.]
- Will do a somewhat weaker version of their analysis (same algorithm but not as tight a bound).
- Talk about it in the context of online pricing…

## Online pricing

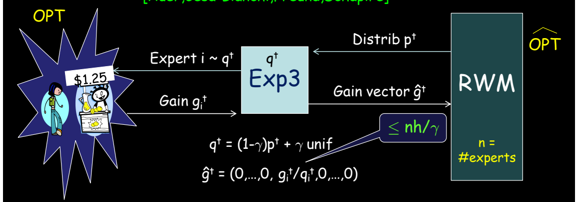
- Say you are selling lemonade (or a cool new software tool, or bottles of water at the world expo).
- Protocol #1: for t=1,2,…T
  - Seller sets price $p^t$
  - Buyer arrives with valuation $v^t$
  - If $v^t \geq p^t$, buyer purchases and pays $p^t$, else doesn't.
  - $v^t$ revealed to algorithm.
  - Repeat.
- Protocol #2: same as protocol #1 but without $v^t$ revealed.
- What can we do now?
- Assume all valuations in [1,h]
- Goal: do nearly as well as best fixed price in hindsight.

Can solve by setting one expert per price level and using RWM!

## Multi-armed bandit problem

Exponential Weights for Exploration and Exploitation (exp³)

[Auer,Cesa-Bianchi,Freund,Schapire]

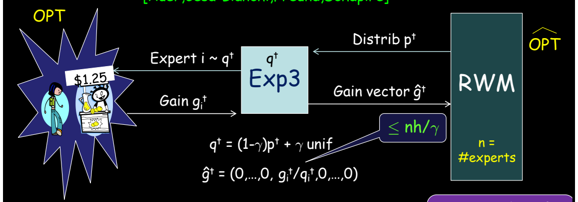



1. RWM believes gain is: $p^t \cdot \hat{g}^t = p_i^t(g_i^t/q_i^t) \equiv g^t_{RWM}$
2. $\sum_t g^t_{RWM} \geq \widehat{OPT}/(1+\epsilon) - O(\epsilon^{-1}\, nh/\gamma \log n)$
3. Actual gain is: $g_i^t = g^t_{RWM}(q_i^t/p_i^t) \geq g^t_{RWM}(1-\gamma)$
4. $E[\widehat{OPT}] \geq OPT$. Because $E[\hat{g}_j^t] = (1-q_j^t)0 + q_j^t(g_j^t/q_j^t) = g_j^t$, so $E[\max_j[\sum_t \hat{g}_j^t]] \geq \max_j[E[\sum_t \hat{g}_j^t]] = OPT$.

## Multi-armed bandit problem

Exponential Weights for Exploration and Exploitation (exp³)

[Auer,Cesa-Bianchi,Freund,Schapire]




Can even reduce $\epsilon^{-2}$ to $\epsilon^{-1}$ with more care in analysis.

Conclusion ($\gamma = \epsilon$):

$E[Exp3] \geq OPT/(1+\epsilon)^2 - O(\epsilon^{-2}\, nh \log(n))$

## A natural generalization

(Going back to full-info setting)

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

## A natural generalization

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

"Specialists" or "sleeping experts" problem.

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

## A simple algorithm and analysis (all on one slide)

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

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

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

## Other uses

- What if we want to adapt to change - do nearly as well as best recent expert?
- Say we know # time steps T in advance (or guess and double). Make T copies of each expert, one who wakes up on day i for each $0 \leq i \leq T-1$.
- Our cost in previous t days is at most $(1+\epsilon)$(best expert in last t days) + $O(\epsilon^{-1} \log(NT))$.
- (not best possible bound since extra log(T) but not bad).