

15-859 (B) Machine Learning Theory
 15-859 (B) Machine Learning Theory

15-859 (B) Machine Learning Theory

Avrim Blum
 —02/01/06— 12/06/06

Plan for today:

- problem of "combining expert advice"
- Weighted-majority algorithm
- Generalizations: time-intervals & "sleeping experts"
- Regret-bounds and connections to game-theory (minimax optimality and correlated equilibria)

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" \equiv someone with an opinion. Not necessarily someone who knows anything.]

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?

Or game-theoretic version: experts are different strategies. In each time step you pick one and get its payoff. (Will get back to this later...)

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?)

Simpler question:

- What if we are told that some expert is perfect?
- Can we guarantee to make at most $\log(n)$ mistakes?

Easy: halving/majority-vote algorithm. Can we do nearly as well as best in hindsight?

Using "expert" advice

If one expert is perfect, can get $\leq \lg(n)$ mistakes with halving alg.

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
predictions	Y	Y	N
weights	1	1	.5
predictions	Y	N	Y
weights	1	.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,

$$\begin{aligned} (1/2)^m &\leq n(3/4)^M \\ (4/3)^M &\leq n2^m \\ M &\leq 2.4(m + \lg n) \end{aligned}$$

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 $1 - \epsilon$.

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

$M = \text{expected \#mistakes}$

- $M \leq 1.39m + 2 \ln n \quad \leftarrow \epsilon = 1/2$
- $M \leq 1.15m + 4 \ln n \quad \leftarrow \epsilon = 1/4$
- $M \leq 1.07m + 8 \ln n \quad \leftarrow \epsilon = 1/8$

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.
 - $W_{\text{final}} = n(1 - \epsilon F_1)(1 - \epsilon F_2) \dots$
 - $\ln(W_{\text{final}}) = \ln(n) + \sum_t [\ln(1 - \epsilon F_t)] \leq \ln(n) - \epsilon \sum_t F_t$
(using $\ln(1-x) < -x$)
($\sum F_t = E[\text{\# mistakes}]$)
- If best expert makes m mistakes, then $\ln(W_{\text{final}}) > \ln((1 - \epsilon)^m)$.
- Now solve: $\ln(n) - \epsilon M > m \ln(1 - \epsilon)$.

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

Summarizing

- At most $(1 + \epsilon)$ times worse than best expert in hindsight, with additive $\epsilon^{-1} \log(n)$.
- Often written in terms of additive loss. If running T time steps, set epsilon to get bound $E[\text{Alg cost}] \leq \text{OPT} + (2T \log n)^{1/2}$
- Define average regret in T time steps as:
(avg per-day cost of alg) - (avg per-day cost of best fixed expert in hindsight).
 Goes to 0 or better as $T \rightarrow \infty$ [= "no-regret" algorithm].

What can we use this for?

- Can use to combine multiple algorithms to do nearly as well as best in hindsight.
- Can apply RWM in situations where experts are making choices that cannot be combined.
 - Choose expert i with probability $p_i = w_i / \sum_i w_i$.
 - E.g., repeated game-playing, repeated strategy-choosing. (Alg generalizes to case where in each time step, each expert gets a cost in $[0,1]$)

Repeated play of matrix game

- Let's use a no-regret alg.
- Time-average performance guaranteed to approach minimax value V of game (or better, if life isn't adversarial).
- In fact, existence of no-regret alg yields proof of minimax thm....



A natural generalization

- A natural generalization of this setting: say we have a list of n prediction rules, but not all rules fire on any given example.
- E.g., document classification. Rule: "if <word-X> appears then predict <Y>". E.g., if has football then classify as sports.
- E.g., path-planning: "on snowy days, use this route".
- Natural 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 $E[\text{cost}_i(\text{alg})] \leq (1+\epsilon)\text{cost}_i(i) + O(\epsilon^{-1} \log n)$.
- So, if 80% of documents with football *are* about sports, we should have error $\leq 21\%$ on them.
 - "Specialists" or "sleeping experts" problem.

A natural generalization

Generalized version of randomized WM:

- Initialize all rules to have 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:
 - $R_i = [\sum_j p_j \text{cost}(j)] / (1+\epsilon) - \text{cost}(i)$
 - $w_i \leftarrow w_i (1+\epsilon)^{R_i}$
- 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 $(1+\epsilon)$ factor.
- Can then prove that total sum of weights never goes up.

Why does this work?

- Update weights:
 - If didn't fire, leave weight alone.
 - If did fire, raise or lower depending on performance compared to weighted average:
 - $R_i = [\sum_j p_j \text{cost}(j)] / (1+\epsilon) - \text{cost}(i)$
 - $w_i \leftarrow w_i (1+\epsilon)^{R_i}$
- Can then prove that total sum of weights never goes up.
- One way to look at weights:
 - $w_i = (1+\epsilon)^{E[\text{cost}_i(\text{alg})] / (1+\epsilon) - \text{cost}_i(i)}$
 - I.e., we are explicitly giving large weights to rules for which we have large regret.
 - Since sum of weights $\leq n$, exponent must be $\leq \log_{1+\epsilon} n$
- This implies our bound: $E[\text{cost}_i(\text{alg})] \leq (1+\epsilon)\text{cost}_i(i) + O(\epsilon^{-1} \log n)$.

Why does this work?

- Update weights:
 - If didn't fire, leave weight alone.
 - If did fire, raise or lower depending on performance compared to weighted average:
 - $R_i = [\sum_j p_j \text{cost}(j)] / (1+\epsilon) - \text{cost}(i)$
 - $w_i \leftarrow w_i (1+\epsilon)^{R_i}$
- Can then prove that total sum of weights never goes up.

Say $i \in I$ if i fires at time t . $W^t = \sum_{i \in I} w_i^t$.

$$\begin{aligned} \sum_{i \in I} w_i^{t+1} &= \sum_{i \in I} w_i^t (1 + \epsilon)^{[\sum_j p_j \text{cost}^t(j)] / (1 + \epsilon) - \text{cost}^t(i)} \\ &\leq \sum_{i \in I} w_i^t (1 + \frac{\epsilon}{1 + \epsilon} [\sum_j p_j \text{cost}^t(j)]) (1 - \frac{\epsilon}{1 + \epsilon} \text{cost}^t(i)) \\ &\leq W^t + \frac{\epsilon}{1 + \epsilon} W^t \sum_j p_j \text{cost}^t(j) - \frac{\epsilon}{1 + \epsilon} \sum_{i \in I} w_i^t \text{cost}^t(i) \\ &= W^t \quad (\text{since } w_i^t = W^t p_i^t \text{ for } i \in I) \end{aligned}$$

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".
- Real motivation: connection to correlated equilibria.
 - 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.

Internal/swap-regret, contd

- If all parties run a low internal/swap regret algorithm, then empirical distribution of play is an apx correlated equilibrium.
 - Correlator chooses random time $t \in \{1, 2, \dots, T\}$. 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 gives a nice distributed way to get apx correlated equilibria in multiplayer games.

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 ϵ regret is linear in n rather than $\log(n)$