TTIC 31290 - Machine Learning for Algorithm Design (Fall 2025)

Avrim Blum and Dravyansh Sharma

Lecturer: Avrim Blum

Lecture 12: Online learning

Mistake-bound model:

- Basic results, relation to PAC, halving algorithm
- Connections to information theory

Combining "expert advice":

- (Randomized) Weighted Majority algorithm
- Regret-bounds, connections to game theory

Online learning

- What if we don't want to make assumption that data is coming from some fixed distribution? Or any assumptions at all?
- Can no longer talk about past performance predicting future results.
- Can we hope to say anything interesting??

Idea: mistake bounds & regret bounds.

Mistake-bound learning model

- View learning as a sequence of stages.
- In each stage, algorithm is given x, asked to predict $f(x)$, and then is told correct value.
- Make no assumptions about order of examples.
- Goal is to bound total number of mistakes.

Alg A learns class C with mistake bound M if A makes $\leq M$ mistakes on any sequence of examples consistent with some $f \in C$.

Mistake-bound learning model

Alg A learns class C with mistake bound M if A makes $\leq$ M mistakes on any sequence of examples consistent with some f $\in$C.

- Note: can no longer talk about "how much data do I need to converge?" Maybe see same examples over again and learn nothing new. But that's OK if don't make mistakes either…
- Try to bound in terms of size of examples n and complexity of target s.
- C is **learnable** in MB model if exists alg with mistake bound and running time per stage poly(n,s).

Simple example: disjunctions

- Suppose features are boolean: $X = \{0,1\}^n$.
- Target is an OR function, like $x_3 \vee x_9 \vee x_{12}$.
- Can we find an on-line strategy that makes at most n mistakes?
- Sure.
  - Start with $h(x) = x_1 \vee x_2 \vee \ldots \vee x_n$
  - Invariant: {features in h} $\supseteq$ {features in f}
  - Mistake on negative: discard features in h set to 1 in x. Maintains invariant & decreases $|h|$ by 1.
  - No mistakes on positives. So at most n mistakes total.

Simple example: disjunctions

- Algorithm makes at most n mistakes.
- No deterministic alg can do better:

1 0 0 0 0 0 0 + or – ?

0 1 0 0 0 0 0 + or – ?

0 0 1 0 0 0 0 + or – ?

0 0 0 1 0 0 0 + or – ?

…

MB model properties

An alg *A* is "conservative" if it only changes its state when it makes a mistake.

Claim: if *C* is learnable with mistake-bound *M*, then it is learnable by a conservative alg.

Why? (Assume learning alg is deterministic)

- Take generic alg *A*. Create new conservative *A'* by running *A*, but rewinding state if no mistake is made.
- Still $\leq M$ mistakes because *A* still sees a legal sequence of examples.

MB learnable ⇒ PAC learnable

Say alg *A* learns *C* with mistake-bound M.

Transformation 1:

- Run (conservative) *A* until it produces a hyp h that survives $\geq (1/\varepsilon)\ln(M/\delta)$ examples.
- Pr(fooled by any given h) $\leq \delta/M$.
- Pr(fooled ever) $\leq \delta$.

  Uses at most $(M/\varepsilon)\ln(M/\delta)$ examples total.

- Fancier method gets $O(\varepsilon^{-1}[M + \ln(1/\delta)])$

One more example…

- Say we view each example as an integer between 0 and 2^n-1.
- $C = \{[0,a] : a < 2^n\}$. (device fails if it gets too hot)
- In PAC model we could just pick any consistent hypothesis. Does this work in MB model?
- What would work?

What can we do with unbounded computation time?

- "Halving algorithm": take majority vote over all consistent $h \in C$. Makes at most $\lg(|C|)$ mistakes.
- What if we had a "prior" p over fns in C?
  - Weight the vote according to p. Make at most $\lg(1/p_f)$ mistakes, where f is target fn.
- What if f was really chosen according to p?
  - Expected number of mistakes $\leq \sum_h [p_h \lg(1/p_h)]$
    = entropy of distribution p.

What can we do with unbounded computation time?

- "Halving algorithm": take majority vote over all consistent $h \in C$. Makes at most $\lg(|C|)$ mistakes.
- What if C has functions of different sizes?
- For any (prefix-free) representation, can make at most 1 mistake per bit of target.
  - Give each f a weight of $1/2^{size(f)}$.
  - Total sum of weights is at most 1. (Can you see why?).
  - So, make at most $size(f)$ mistakes.

Is halving alg optimal?

- Not necessarily
- Can think of MB model as 2-player game between alg and adversary.
  - Adversary picks x to split C into $C_-(x)$ and $C_+(x)$. [fns that label x as – or + respectively]
  - Alg gets to pick one to throw out.
  - Game ends when all fns left are equivalent.
  - Adversary wants to make game last as long as possible.
- OPT(C) = MB when both play optimally.

Is halving alg optimal?

- Halving algorithm: throw out larger set.
- Optimal algorithm: throw out set with larger mistake bound.

What if there is no perfect function?

Think of as $h \in C$ as "experts" giving advice to you. Want to do nearly as well as best of them in hindsight.

These are called "regret bounds".

- Show that our algorithm does nearly as well as best predictor in some class.

We'll look at a strategy whose running time is $O(|C|)$. So, only computationally efficient when C is small.

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

[In i.i.d. setting, could just sample a while and then pick the best on your sample]

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.

Weights:	1	1	1	1				
Predictions:	U	U	U	D	We predict:	U	Truth:	D
Weights:	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1				

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- ε.

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

M = expected #mistakes

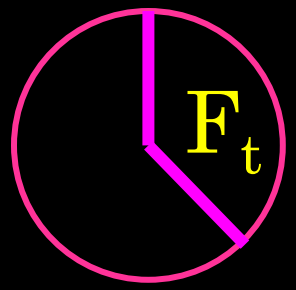
$M \leq 1.39m + 2\ln n \quad \leftarrow \varepsilon = 1/2$

$M \leq 1.15m + 4\ln n \quad \leftarrow \varepsilon = 1/4$

$M \leq 1.07m + 8\ln n \quad \leftarrow \varepsilon = 1/8$

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.
  - $W_{final} = n(1-\varepsilon F_1)(1 - \varepsilon F_2)\ldots$
  - $\ln(W_{final}) = \ln(n) + \sum_t [\ln(1 - \varepsilon F_t)] \leq \ln(n) - \varepsilon \sum_t F_t$

    (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] $\leq (1+\varepsilon)$OPT + ε^{-1}log(n)
  = OPT + (εOPT + ε^{-1}log(n))

Assuming here that $OPT \geq \log(n)$

- If set ε=(log(n)/OPT)$^{1/2}$ to balance the two terms out (or use guess-and-double), get bound of
  $M \leq OPT+2(OPT \cdot \log n)^{1/2} \leq OPT+2(T \log n)^{1/2}$

- 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→ ∞ = "no-regret" algorithm].

Extensions

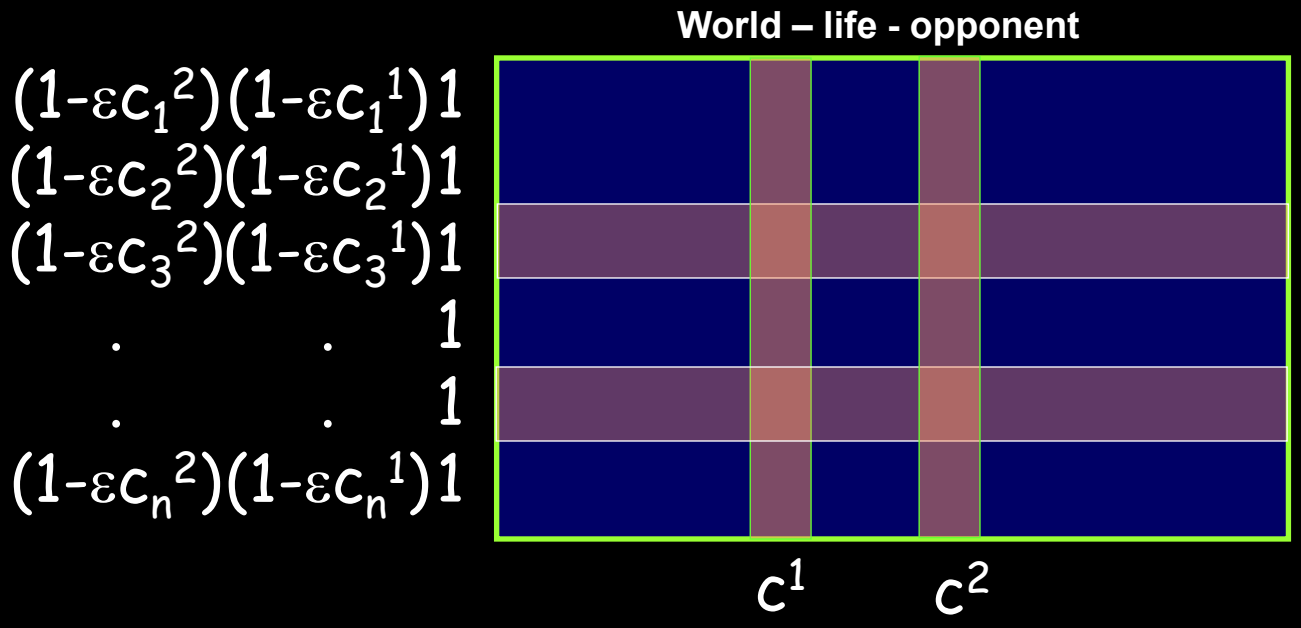
- What if experts are actions? (rows in a matrix game, ways to drive to work,…)
- At each time t, each has a loss (cost) in {0,1}.
- Can still run the algorithm
  - Rather than viewing as "pick a prediction with prob proportional to its weight" ,
  - View as "pick an expert with probability proportional to its weight"
  - Alg pays expected cost $\overrightarrow{p_t} \cdot \overrightarrow{c_t} = F_t$.
- Same analysis applies.

  Do nearly as well as best action in hindsight!

Extensions

- What if losses (costs) in [0,1]?
- Just modify alg update rule: $w_i \leftarrow w_i(1 - \epsilon c_i)$.
- Fraction of wt removed from system is:
  $(\sum_i w_i \epsilon c_i)/(\sum_j w_j) = \epsilon \sum_i p_i c_i = \epsilon[our\ expected\ cost]$
- Analysis very similar to case of {0,1}.

RWM (multiplicative weights alg)



scaling so costs in [0,1]

Guarantee: do nearly as well as best fixed row in hindsight

The algo as stated requires seeing entire cost vector at each time step, but can extend to the bandit case where at each step you only see the cost of the row you chose (but $\log n$ becomes $O(n)$).

Some References

- N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm." Machine learning, vol 2, 1988.
- N. Littlestone and M.K. Warmuth. "The weighted majority algorithm." Information and computation, vol 108.2, 1994.
- N. Cesa-Bianchi, Y. Freund, D. Haussler, D.P. Helmbold, R.E. Schapire, and M.K. Warmuth. "How to use expert advice." Journal of the ACM, vol 44, no. 3, 1997.
- Y. Freund and R.E. Schapire. "Adaptive game playing using multiplicative weights." Games and Economic Behavior 29, no. 1-2, 1999.