10-806 Foundations of Machine Learning and Data Science

Lecturer: Avrim Blum 10/12/15

Lecture 10: Online learning I
Mistake-bound model:
- Basic results
- Connection to PAC/distributional learning
- Halving alg

Combining "expert advice":
- (Randomized) Weighted Majority algorithm

PAC model

- Data arrives from some distribution D, labeled by some target c^*.
- We see $S = (x_1, y_1), (x_2, y_2), \dots (x_m, y_m)$ where $x_i \sim D$, and $y_i = c^*(x_i)$.
- Goal: produce h with low true error $err_D(h)$.

Online learning

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

Idea: mistake bounds & regret bounds.

(Mistake bounds: $c^* \in C$, Regret bounds: general case)

Mistake-bound 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 sequence of x's.
- Goal is to bound total number of mistakes.

Alg A learns class C with mistake bound M if A makes ≤ M mistakes on any sequence of examples consistent with some f ∈ C.

Mistake-bound model

Alg A learns class C with mistake bound M if A makes ≤ M mistakes on any sequence of examples consistent with some f ∈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…
- Want mistake bound poly(n, s), where n is size of example and s is size of smallest consistent f ∈ C.
- 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 $h(x) = x_1 \vee x_2 \vee \dots \vee x_n$
  - Invariant: {vars in h} ⊇ {vars in f }
  - Mistake on negative: throw out vars in h set to 1 in x. Maintains invariant and 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 by a deterministic algo with mistake-bound M, then also learnable by a conservative alg with mistake bound M.

Why?

- Take generic alg A. Create new conservative A' by running A, but rewinding state if no mistake made.
- Still ≤ M mistakes because algo 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 ≥ (1/ε)ln(M/δ) examples.
- If h_1 is bad, Pr(fooled by h_1) ≤ δ/M.
- If h_2 is bad, Pr(fooled by h_2) ≤ δ/M.
- …
- Pr(fooled ever) ≤ δ.

Uses at most $\frac{M}{\epsilon}\ln\left(\frac{M}{\delta}\right)$ examples total.

MB learnable ⇒ PAC learnable

Fancier method gets $O\left(\frac{1}{\epsilon}\left[M+\ln\left(\frac{1}{\delta}\right)\right]\right)$.

- Run conservative A for $O\left(\frac{1}{\epsilon}\left[M+\ln\left(\frac{1}{\delta}\right)\right]\right)$ examples. Argue that whp at least one of hyps produced has error ≤ ε/2.
- Test the M hyps produced on $O\left(\frac{1}{\epsilon}\left[\ln\left(\frac{M}{\delta}\right)\right]\right)$ new examples and take the best.
- Nice correctness proof using Chernoff bounds, but will skip here.

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 gets too hot)
- In PAC model, could just pick any $h \in C$ with $err_S(h) = 0$. 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 ∈ C.
- Each mistake guarantees to reduce version space (set of h ∈ C consistent with data so far) by at least a factor of 2.
- Makes at most lg(|C|) mistakes.

Is halving alg optimal?

- Halving algorithm: predict using larger set (h in version space that predict + versus h in version space that predict -).
- Optimal algorithm: predict using the set with larger mistake bound.
- In some cases, these can differ by a bit.

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.]

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 $\lg(n)[\mathrm{OPT}+1]$ 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, multiply by 1-ε rather than by ½.

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

M = expected #mistakes

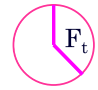
$M \le 1.39m + 2\ln n \quad \leftarrow \varepsilon = 1/2$

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

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

unlike most worst-case bounds, numbers are pretty good.

Analysis



- Say at time t we have fraction F_t of weight on experts that make mistake.
- So, we have probability F_t of making a mistake, and we remove an εF_t fraction of the total weight.
  - $W_{final} = n(1-\varepsilon F_1)(1 - \varepsilon F_2)\ldots$
  - $\ln(W_{final}) = \ln(n) + \sum_t [\ln(1 - \varepsilon F_t)] \le \ln(n) - \varepsilon \sum_t F_t$ (using ln(1-x) < -x)

    $= \ln(n) - \varepsilon M.$ ($\sum F_t = E[\#\text{ mistakes}] = M$)
- If best expert makes m mistakes, then $\ln(W_{final}) > \ln((1-\varepsilon)^m)$.
- Now solve: $\ln(n) - \varepsilon M > m \ln(1-\varepsilon)$.

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

Summarizing

- $M \le (1+\epsilon)OPT + \frac{\log(n)}{\epsilon}$, where OPT is the loss of best expert in hindsight.
- If run for $T \ge \log(n)$ steps, and set $\epsilon = \sqrt{\frac{\log(n)}{T}}$, and use the fact that $OPT \le T$, we get:

$$M \le OPT + \sqrt{T\log(n)} + \sqrt{T\log(n)}$$

- Dividing both sides by T to get avg loss per round:

$$\frac{M}{T} \le \frac{OPT}{T} + 2\sqrt{\frac{\log(n)}{T}}$$

Regret term 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 $\vec{p_t} \cdot \vec{c_t} = F_t$.
- Same analysis applies.

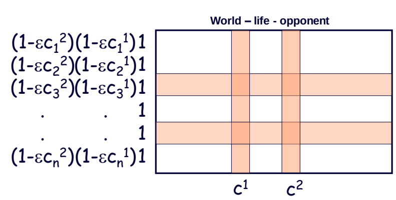
  Do nearly as well as best action in hindsight!

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 fixed row in hindsight