15-859(A) Machine Learning Theory

Avrim Blum

01/20/04

Plan for today:
- MB model recap.
- problem of "combining expert advice"
- Weighted-majority alg and applications

Mistake-bound model recap

- View learning as a sequence of trials.
- In each trial, algorithm is given *x*, asked to predict *f(x)*, and then is told correct value.
- Make no assumptions about how examples are chosen.
- Goal is to minimize 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 $f \in C$.

Simple example: learning an OR fn

- Suppose features are boolean: $X = \{0,1\}^n$.
- Target is an OR function, like $x_3 \vee x_9 \vee x_{12}$, with no noise.
- 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: {vars in h} contains {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: learning an OR fn

- 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 − ?
...

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.
- More generally, for any (prefix-free) description language, can make at most 1 mistake per bit to describe target fn.
  - give each h a weight of $(\frac{1}{2})^{\text{size}(h)}$
  - Total sum of weights ≤ 1.
  - Take weighted vote. Each mistake cuts total weight left by at least a factor of 2.

Is halving alg optimal?

- Not necessarily (see hwk).
- 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.

Optimal strategy

- What is the optimal strategy for the algorithm?
- Given x, we "just" calculate $OPT(C_-(x))$ and $OPT(C_+(x))$. Throw out the one that's worse.
- Equivalently: can define $OPT(C)$ as:
  - If $|C|=1$ then $OPT(C) = 0$. Else,
  - $OPT(C) = 1 + \max_x[\min[OPT(C_-(x)), OPT(C_+(x))]]$

Next topic

- What if there's no perfect function?
- Think of as n "experts" giving advice to you. Want to do nearly as well as best of them in hindsight.
  - Can view each "expert" as a different $h \in C$.
  - Or, think of the special case of C={single variable functions}. Goal is efficient alg that does nearly as well as best single variable.

These are called "regret bounds".

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

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 ≤ 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 log(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	1		
predictions	Y	Y	Y	N	Y	Y
weights	1	1	1	.5		
predictions	Y	N	N	Y	N	Y
weights	1	.5	.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 ½ to 1- ε.

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

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

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

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

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

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

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

Summarizing

- At most $(1+\varepsilon)$ times worse than best expert in hindsight, with additive $\varepsilon^{-1}\log(n)$.
- If have prior, can replace additive term with $\varepsilon^{-1}\log(1/p_i)$. [$\varepsilon^{-1}$ x number of bits]
- Often written in terms of additive loss. If running T time steps, set epsilon to get additive loss $(2T \log n)^{1/2}$

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.
  - E.g., repeated game-playing.
  - E.g., online shortest path problem

  [OK if losses in [0,1]. Replace F_t with $P_t \cdot L_t$ and penalize expert i by $(1-\varepsilon)^{loss(i)}$]
- Extensions:
  - "bandit" problem.
  - efficient algs for some cases with many experts.
  - Sleeping experts / "specialists" setting.

A nice application

- Play repeated game to do nearly as well as best strategy in hindsight.
- (This will be at least as good as minimax optimal).
- Gives a proof of minimax theorem.

2-player zero-sum games

E.g., Rock-Paper-Scissors.

Payoff to row player:

	R	P	S
R	0	−1	1
P	1	0	−1
S	−1	1	0

- Minimax optimal strategy: (randomized) strategy with best worst-case guarantee.

  What is minimax optimal for RPS?
- What about the game below:

Payoff to row player:

	N	D
N	−5	5
D	10	−10

Optimal strategy for row player?

Column player?

The min-max theorem

	N	D
N	−5	5
D	10	−10

- Suppose that for any (randomized) strategy of your opponent, there exists a deterministic counter-strategy for you that guarantees you an expected gain $\geq V$.

  Then, there exists a randomized strategy for you such that for any counter-strategy of the opponent, you get an expected gain $\geq V$.

- Equivalently:

$$\max_{S_{row}} \min_{S_{col}} E[\text{payoff}] = \min_{S_{col}} \max_{S_{row}} E[\text{payoff}]$$

  I.e., suppose that for all S_{col} there exists S_{row} such that expected gain is $\geq V$. Then there exists a fixed S_{row} such that for all S_{col} the expected gain is $\geq V$ too.

  (strategy ≡ randomized strategy)

Using RWM for online play

- rows are "experts". Pick row j with prob w_j/W.
- To keep with terminology, let's talk in terms of gains (doesn't really matter):
  - scale matrix entries to range [0,1].
  - reward expert of gain g by multiplying by $(1+\varepsilon)^g$
  - For any sequence of games, our expected gain $\geq OPT(1 - \varepsilon/2) - (\ln n)/\varepsilon$, where OPT is best fixed strategy in hindsight (which is at least as good as minimax optimal).
- We've actually just proven the Min-Max theorem!

Why?

- What would it mean for min-max to be false?
  - If we know opponents randomized strategy, we can get expected gain $\geq V$, but if we have to choose our randomized strategy first, then opponent can force us to get $\leq V - \delta$.
- This contradicts our bound if we use $\varepsilon = \delta$. Our gain per game is approaching $OPT(1-\varepsilon/2)$, where $OPT \geq V$.
- In other words: if there was a gap (V versus $V - \delta$), then for *any* randomized strategy we chose, an opponent knowing our strategy could force us to get no more than $V - \delta$ on average per play.
- But, we are doing better.