Lecture 2: 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

Recap from last time
- Last time: PAC model and Occam's razor.
  - If data set has $m \geq \left(\frac{1}{e}\right)s \ln(2) + \ln(1/\delta)$ examples, then whp any consistent hypothesis with $\text{size}(h) < s$ has $\text{err}(h) < \varepsilon$.
  - Equivalently, $\text{size}(h) \leq \frac{(um - \ln(1/\delta))/\ln(2)}{s}$ suffices.
  - "compression ⇒ learning"
- Occam bounds ⇒ any class is learnable in PAC model if computation time is no object.

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 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.

Simple example: disjunctions
- Suppose features are boolean: $X = \{0,1\}^n$.
- Target is an OR function, like $x_3 \lor x_9 \lor x_{12}$.
- Can we find an on-line strategy that makes at most $n$ mistakes?
- Sure.
  - Start with $h(x) = x_1 \lor x_2 \lor \ldots \lor x_n$.
  - Invariant: $(\text{features in } h) \supseteq (\text{features in } f)$.
  - 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:
  
  $\begin{array}{c}
  1 & 0 & 0 & 0 & 0 & 0 & 0 & +
  \\
  0 & 1 & 0 & 0 & 0 & 0 & 0 & +
  \\
  0 & 0 & 1 & 0 & 0 & 0 & 0 & +
  \\
  0 & 0 & 0 & 1 & 0 & 0 & 0 & +
  \\
  \vdots
  \end{array}$

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?

- 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 $\Rightarrow$ PAC learnable

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

Transformation 1:

- Run (conservative) $A$ until it produces a hyp $h$ that survives $\geq (1/e)\ln(M/\delta)$ examples.
- $Pr(\text{fooled by any given } h) \leq \delta/M$.
- $Pr(\text{fooled ever}) \leq \delta$.
  - Uses at most $(M/e)\ln(M/\delta)$ examples total.
- Fancier method gets $O(e^{-1}[M + \ln(1/\delta)])$

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 $\leq \sum_p [p_{i}\lg(1/p_{i})] = \text{entropy of distribution } p$.

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.
    - Think of writing random 0s and 1s until hit a legal hypothesis or no longer a prefix of one.
    - $p_f = Pr(\text{reach } f) = 1/2^{\text{size}(f)}$
    - $\lg(1/p_f) = \text{size}(f)$. 
**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).
  - Alg gets to pick one to throw out.
  - Game ends when all functions left are equivalent.
- OPT(C) = MB when both play optimally.

**What if there is no perfect function?**

Think of as h ∈ 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.,

<table>
<thead>
<tr>
<th>Expt 1</th>
<th>Expt 2</th>
<th>Expt 3</th>
<th>neighbor’s dog</th>
<th>truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>down</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Can we do nearly as well as best in hindsight?

[*“expert” = someone with an opinion. Not necessarily someone who knows anything.*]

[Note: would be trivial in PAC (i.i.d.) setting]

**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.

Weights: 1 1 1 1
Predictions: U U U D  We predict: U  Truth: D
Weights: § § § § 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%.
  
After \( M \) mistakes, \( W \) is at most \( n(3/4)^M \).

Weight of best expert is \( (1/2)^m \). So, 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 \( 1/2 \) to \( 1 - \epsilon \).

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

**Summarizing**

- \( E[\# \) mistakes] \( \leq \) \( (1+\epsilon)OPT + \epsilon^{1}(\log(n)) \)
  
Assuming here that \( OPT \geq \log(n) \)

- If set \( \epsilon = (\log(n)/OPT)^{1/2} \) 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:
  
(\text{avg per-day cost of alg}) - (\text{avg per-day cost of best fixed expert in hindsight}).

Goes to 0 or better as \( T \to \infty \) = "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 \( p_t \cdot c_t = F_t \).

- 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:
  
(\text{sum over i of} \( w_i \cdot \epsilon c_i \)) / (\text{sum over i of} \( w_i \)) = \epsilon \sum_i w_i \cdot c_i = \epsilon \text{(our expected cost)}

- Analysis very similar to case of \( \{0,1\} \).
Guarantee: do nearly as well as fixed row in hindsight
Which implies doing nearly as well (or better) than minimax optimal

If play two RWM against each other, then empirical distributions must be near-minimax-optimal.
(Else, one or the other could & would take advantage)