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.

**Algorithm A learns class \( C \) with mistake bound \( M \) if A makes \( \leq M \) mistakes on any sequence of examples consistent with some \( f \in C \).**

Recap: open problems

**Can one efficiently PAC-learn...**
- \( C=\{ \text{fns with only } O(\log n) \text{ relevant variables}\} \)? (or even \( O(\log \log n) \) or \( o(1) \)
  relevant variables)? This is a special case of DTs, DNFs.
- Monotone DNF over uniform D?
- Weak agnostic learning of monomials.

Recap from last time

- Last time: PAC model and Occam’s razor.
  - If data set has \( m \geq (1/e)[s \ln(2) + \ln(1/\delta)] \)
    examples, then whp any consistent hypothesis with \( \text{size}(h) \leq s \) has \( \text{err}(h) < \varepsilon \).
  - Equivalently, suffices to have \( s \leq (cm \ln(1/\delta))/\ln(2) \)
  - “Compression = learning”
- [KV] book has esp. good coverage of this and related topics.
- Occam bounds ⇒ any class is learnable 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.**
**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 \( h(x) = x_1 \lor x_2 \lor ... \lor x_n \).
  - Invariant: \( \{\text{vars in } h\} \supseteq \{\text{vars in } f\} \).
  - 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.

**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 \( \geq (1/\epsilon) \ln (M/\delta) \) examples.
- \( \Pr(\text{fooled by any given } h) \leq \delta/M \).
- \( \Pr(\text{fooled ever}) \leq \delta \).
  - Uses at most \( (M/\epsilon) \ln (M/\delta) \) examples total.
- Fancier method gets \( O(\epsilon^{-1} [M + \ln (1/\delta)]) \).

**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 \( C \) has functions of different sizes?
  - For any (prefix-free) representation, can make at most 1 mistake per bit of target.
    - give each \( h \) a weight of \( (1/2)^{\text{size}(h)} \)
    - Total sum of weights \( \leq 1 \)
    - Take weighted vote. Each mistake removes at least \( 1/2 \) of total weight left.
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 \( p \).

Is halving alg optimal?

• Not necessarily (see hwk TBA).
• Can think of MB model as 2-player game between alg and adversary.
  - Adversary picks \( x \) to split \( C \) into \( C(x) \) and \( C, \). \([\text{fns that label } x \text{ as - or + respectively}]\)
  - Alg gets to pick one to throw out.
  - Game ends when all fns left are equivalent.
• 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 \( p \).
• OPT(C) = MB when both play optimally.

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.,

<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>down</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>down</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 \( \cdot \lg(n) \) 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 \( \lg(n) \cdot \text{OPT+1} \) mistakes, where \( \text{OPT} \) 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.

<table>
<thead>
<tr>
<th>Weights</th>
<th>Predictions</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>1</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>1</td>
<td>U</td>
<td>D</td>
</tr>
</tbody>
</table>

We predict: U

<table>
<thead>
<tr>
<th>Weights</th>
<th>Predictions</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>½</td>
<td>U</td>
<td>D</td>
</tr>
</tbody>
</table>

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\%.
- Weight of best expert is \( (1/2)^m \).

\[
\frac{(1/2)^m}{n(3/4)^M} \leq \left(\frac{4}{3}\right)^M \leq n^{2^m} \leq 2.4(m + \log n)
\]

**Randomized Weighted Majority**

2.4\( (m + \log n) \) 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 \( 1 - \varepsilon \).

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

Summarizing

- \( E[\# \) mistakes\] \( \leq (1+\varepsilon)\)OPT + \( \varepsilon^{-1}\)log\( n \)
- \( M \leq \frac{1.39m + 2 \ln n}{\varepsilon} \approx \frac{1}{2} \)
- \( M \leq \frac{1.15m + 4 \ln n}{\varepsilon} \approx \frac{5}{4} \)
- \( M \leq \frac{1.07m + 8 \ln n}{\varepsilon} \approx \frac{1}{\varepsilon} \)

**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 \( \frac{1}{2}p_i \cdot c_i = F_i \).
  - 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: $w_i \leftarrow w_i (1 - \varepsilon c_i)$.
- Fraction of wt removed from system is: $\left(\sum_i w_i (c_i) \right) / \left(\sum_i w_i \right) = \varepsilon \sum_i p_i c_i = \varepsilon \text{[our expected cost]}
- Analysis very similar to case of (0,1).

Connections to minimax optimality

If play RWM against a best-response oracle, $\bar{p}$ will approach minimax optimality (most $\bar{p}$ will be close).

(If if didn't, wouldn't be getting promised guarantee)

A natural generalization

- A natural generalization of our regret goal (thinking of driving) is: what if we also want that on rainy days, we do nearly as well as the best route for rainy days.
- And on Mondays, do nearly as well as best route for Mondays.
- More generally, have N "rules" (on Monday, use path P). 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 $E[cost(alg)] \leq (1+\varepsilon) cost(i) + O(\varepsilon^{-2} \log N)$.
  - $cost(X) = \text{cost of } X \text{ on time steps where rule } i \text{ fires.}$
- Can we get this?

A natural generalization

- This generalization is esp natural in machine learning for combining multiple if-then rules.
- E.g., document classification. Rule: "if <word-X> appears then predict <Y>". E.g., if has football then classify as sports.
- So, if 90% of documents with football are about sports, we should have error - 11% on them.
  - "Specialists" or "sleeping experts" problem.

Assume we have N rules.

- For all i, want $E[cost(alg)] \leq (1+\varepsilon) cost(i) + O(\varepsilon^{-2} \log N)$.
  - $cost(X) = \text{cost of } X \text{ on time steps where rule } i \text{ fires.}$
A simple algorithm and analysis (all on one slide)

- Start with all rules at weight 1.
- At each time step, of the rules $i$ that fire, select one with probability $p_i / w_i$.
- Update weights:
  - If didn’t fire, leave weight alone.
  - If did fire, raise or lower depending on performance compared to weighted average:
    - $r_i = \left[ \frac{\sum j p_j \text{cost}(j)}{1+e} - \text{cost}(i) \right]$.
    - $w_i \leftarrow w_i (1+e)^{-r_i}$.
  - 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. This ensures sum of weights doesn’t increase.
- Final $w_i = (1+\epsilon)^{E[\text{cost(alg)}]/(1+\epsilon) - \text{cost}(i)}$. So, exponent $\cdot e^{-1}\log N$.
- So, $E[\text{cost(alg)}] \cdot (1+\epsilon)\text{cost}(i) \cdot O(e^{-1}\log N)$.

Application: adapting to change

- What if we want to adapt to change - do nearly as well as best recent expert?
- For each expert, instantiate copy who wakes up on day $t$ for each $0 \leq t \leq T-1$.
- Our cost in previous $t$ days is at most $(1+\epsilon)(\text{best expert in last } t \text{ days}) \cdot O(e^{-1}\log(NT))$.
- (not best possible bound since extra $\log(T)$ but not bad).