Introduction to Machine Learning
TTIC 31020

Prof. Nati Srebro

Lecture 2:

A non-probabilistic start: Online Learning
No Free Lunch vs Universal Learning
The Statistical Learning Model

Online Learning Process

- At each time $t = 1,2,\ldots$
  - We receive an instance $x_t \in \mathcal{X}$
  - We predict a label $\hat{y}_t = h_t(x_t)$
  - We see the correct label y_t of x_t
  - We update the predictor h_{t+1} based on (x_t, y_t)

- Learning rule: mapping $A: (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{Y}^{\mathcal{X}}$
  - $h_t = A\big((x_1, y_1), (x_2, y_2), \ldots, (x_{t-1}, y_{t-1})\big)$

- Goal: make few mistakes $\hat{y}_t \neq y_t$

- Is this possible?

  $\mathcal{X}$ ={items in basket}, $\mathcal{Y}$ ={Ian , Oliver}

No Free Lunch: Online Version

- For **any finite $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{n}$ elements**, and **any learning rule $\boldsymbol{A}$**, there exists a mapping $f(x)$ and a sequence $\left\{\left(x_t, y_t = f(x_t)\right)\right\}_t$ on which A **makes at least $\boldsymbol{n}$ mistakes**
  - $x_1, \ldots, x_n$ all different
  - Define f inductively as:
    $f(x_t) = -A\Big(\big(x_1, f(x_1)\big), \big(x_2, f(x_2)\big), \ldots, \big(x_{t-1}, f(x_{t-1})\big)\Big), t = 1..n$

- For any **infinite $\boldsymbol{\mathcal{X}}$**, and any learning rule A, there exists a mapping $f(x)$ and a sequence $\left\{\left(x_t, y_t = f(x_t)\right)\right\}_t$ on which A **makes a mistake on every round**

- If $\mathcal{X}$ is small, we can limit ourselves to $|\mathcal{X}|$ mistakes by memorizing $f(x_t)$, but “memorizing” doesn’t quite feel like “learning”….

Prior Knowledge

- Assume $y_t = f(x_t)$ for some $f \in \mathcal{H}$
- $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ is a **"hypothesis class"**
  - Learner knows $\mathcal{H}$, but of course doesn't know f
- $\mathcal{H}$ represents our "Prior Knowledge" or "Expert Knowledge"
- We say the sequence $\{(x_t, y_t)\}_t$ is **realizable by $\mathcal{H}$** if $\exists_{f\in\mathcal{H}}\ \forall_t\ y_t = f(x_t)$

E.g.:

$$\mathcal{H} = all\ predictors\ based\ on\ single\ word\ occurence$$

$$\mathcal{H} = \{h_i \mid dictionary\ word\ i\}, \quad h_i(x) = [[\text{word } i \text{ appears in } x]]$$

$$[[condition]] = \begin{cases} +1, & condition \text{ is } true \\ -1, & \text{otherwise} \end{cases}$$

- What if this assumption is wrong??

  ➔ Later…

Learning Finite Hypothesis Classes

- How can we learn problems realizable by a finite hypothesis class?

- The learning rule **CONSISTENT**:
  - use $h \in \mathcal{H}$ consistent with examples so far
  - $\mathbf{CONSISTENT}_{\mathcal{H}}(S) = \left(some\ h \in \mathcal{H}\ s.t.\ \forall_{(x,y)\in S}(h(x) = y)\right)$

  (strictly speaking: not a specific function—we will refer to any rule returning a consistent h as "**CONSISTENT**")

use h that has been correct for all examples seen, but we use the information from one.

- Iterative implementation:
  - Initialize $V_1 = \mathcal{H}$
  - For $t = 1,2, \ldots$
    - Choose some $h_t \in V_t$ (and predict $\hat{y}_t = h_t(x_t)$)
    - Based on (x_t, y_t), update $V_{t+1} = \{h \in V_t | h(x_t) = y_t\}$

eventually discard all hypotheses that are ever incorrect.

- Theorem:
  If $\{(x_t, y_t)\}_t$ is realizable by $\mathcal{H}$, $\mathbf{CONSISTENT}_{\mathcal{H}}$ will make $< |\mathcal{H}|$ mistakes

- Proof:
  If $h_t(x_t) \neq y_t$, h_t is removed from V_t, hence $|V_{t+1}| \leq |V_t| - 1$. Since true f always remains in V_t, $|V_t| \geq 1$.
  Hence, #mistakes $\leq |V_1| - 1$.

We only kick out one hypothesis at a time.

Majority/Halving

- The $\mathbf{MAJORITY}_{\mathcal{H}}$ learning rule (aka the HALVING learning rule):
  - Initialize $V_1 = \mathcal{H}$
  - For $t = 1,2,\ldots$
    - Use h_t, where $h_t(x) = \text{MAJORITY}(\ h(x) : h \in V_t\ )$

      i.e. $h_t = A\big((x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\big) = \text{MAJORITY}(\ h(x) : h \in \mathcal{H}, \forall_{i=1\ldots t-1} h(x_i) = y_i)$

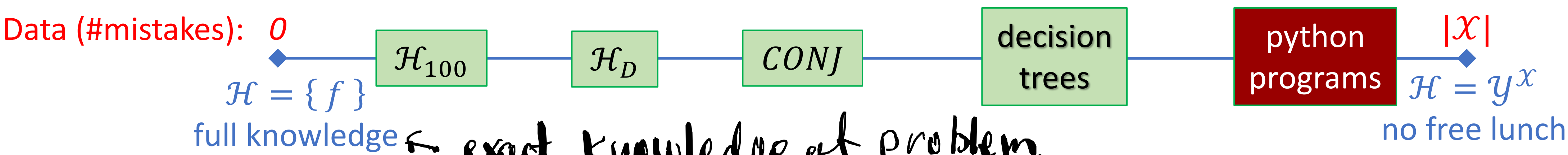
      ➔ predict $\hat{y}_t = \text{MAJORITY}(\ h(x_t) : h \in V_t\ ))$
    - Based on (x_t, y_t), update $V_{t+1} = \{h \in V_t | h(x_t) = y_t\}$

Here we use information from many learning rules.

- Theorem:
  If $\{(x_t, y_t)\}_t$ is realizable by $\mathcal{H}$, $\mathbf{MAJORITY}_{\mathcal{H}}$ will make $< \mathbf{log}_2\, |\mathcal{H}|$ mistakes
- Proof:
  If $h_t(x_t) \neq y_t$, then at least half of the functions $h \in V_t$ are wrong and will be removed, hence $|V_{t+1}| \leq |V_t|/2$.
  Since true f always remains in V_t, $|V_t| \geq 1$. Hence, #mistakes $\leq \log_2 |V_1|$.

The Complexity of $\mathcal{H}$

- $\log_2 |\mathcal{H}|$ measures the "complexity" of the hypothesis class
  - More complex ➔ more mistakes ➔ more data until we learn
  - More specific "expert knowledge" ➔ smaller $\mathcal{H}$ ➔ less mistakes, learn quicker



← exact knowledge of problem.

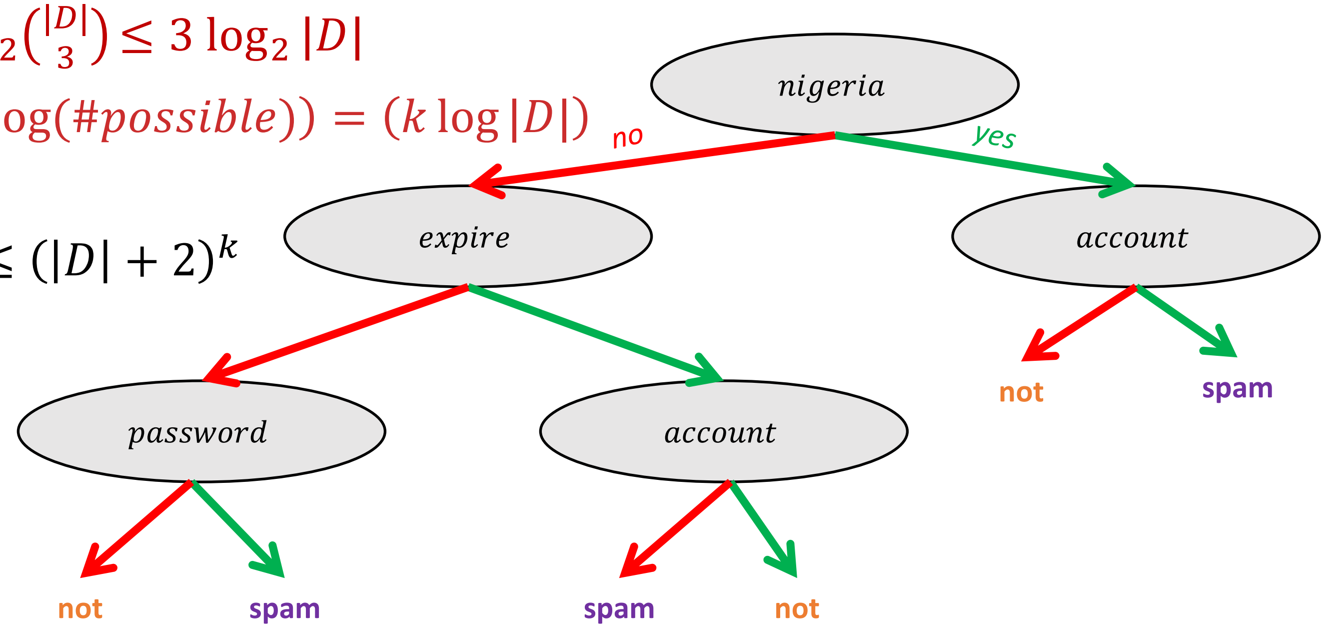
$\mathcal{H}_{100} = \{ [[\text{word } i \text{ in } x]] \mid i \in 100 \text{ special words} \}$ $\quad \log_2 |100|$

$\mathcal{H}_D = \{ [[\text{word } i \text{ in } x]] \mid i \in \text{dictionary } D \}$ $\quad \log_2 |D|$

$CONJ3 = \{ [[\text{word } i_1, i_2 \text{ OR } i_3 \text{ in } x]] \mid i_1, i_2, i_3 \in D \}$ $\quad \log_2 \binom{|D|}{3} \leq 3 \log_2 |D|$

Decision trees with k nodes over predicates $\{ [[\text{word } i \text{ in } x]] \mid i \in D \}$ $\quad O(\log(\#possible)) = (k \log |D|)$

#possible decision trees $\leq (|D| + 2)^k$



→ we sps all words are possible

→ expert tells us "actually spam happens w/ only these 100 words".

→ Actually, just 3 words happening at the same time.

→ It's obvious we need to be more restrictive.

Look at decision trees.

- Most popular Hypothesis class

The Complexity of $\mathcal{H}$

- $\log_2 |\mathcal{H}|$ measures the "complexity" of the hypothesis class
  - More complex ➔ more mistakes ➔ more data until we learn
  - More specific "expert knowledge" ➔ smaller $\mathcal{H}$ ➔ less mistakes, learn quicker

Data (#mistakes): *0* — $\mathcal{H}_{100}$ — $\mathcal{H}_D$ — $CONJ$ — decision trees — python programs — $|\mathcal{X}|$

$\mathcal{H} = \{ f \}$
full knowledge

$\mathcal{H} = \mathcal{Y}^{\mathcal{X}}$
no free lunch

$\mathcal{H}_{100} = \{ [[\text{word } i \text{ in } x]] \mid i \in 100 \ special\ words\}$ $\qquad \log_2 |100|$

$\mathcal{H}_D = \{ [[\text{word } i \text{ in } x]] \mid i \in dictionary\ D\}$ $\qquad \log_2 |D|$

$CONJ3 = \{ [[\text{word } i_1, i_2 \text{ OR } i_3 \text{ in } x]] \mid i_1, i_2, i_3 \in D\}$ $\qquad \log_2 \binom{|D|}{3} \leq 3 \log_2 |D|$

Decision trees with *k* nodes
over predicates $\{ [[\text{word } i \text{ in } x]] \mid i \in D\}$ $\qquad O(\log(\#possible)) = (k \log |D|)$

Python programs with n lines that take x as input $\qquad \leq \log_2 128^{(80\, n)} = 560n = O(n)$

↳ Say 80 char per line, any 128 ASCII char are valid.

Why not use $\mathcal{H} = \{$ short programs $\}$?

- Learn SPAM detectable by 100-line program with $\leq \log_2 128^{100\cdot 80} = 56{,}000$ mistakes (that's nothing!)
- Running MAJORITY requires checking, at each step, and for each program, whether it returns the right answer. ***That's not even computable!*** → Can't tell if a program even terminates!
- Even for classes where predictors are easily computable, such as decision trees:
  - #mistakes (≈data needed to learn) $\leq \log_2 |\mathcal{H}|$
  - But runtime scales linearly with $|\mathcal{H}|$ (need to check all $h \in \mathcal{H}$) a lot of things in $\mathcal{H}$ that take a while.
  - E.g. for decision trees of size k over features D: #mistakes $\leq O(\log(\#trees) = O(k \log|D|)$
    Runtime = $O\big(\#trees \cdot (checktree)\big) = \big(|D|^k \cdot \#points \cdot k\big)$
- We want hypothesis classes that:
  - Capture lots of interesting stuff with low complexity (e.g. low cardinality)
  - Are computationally efficiently learnable

But we don't care about $\mathcal{H}_{python}$, since we want to replace programmers, i.e. we only care about solving solvable problems.

Interim Summary

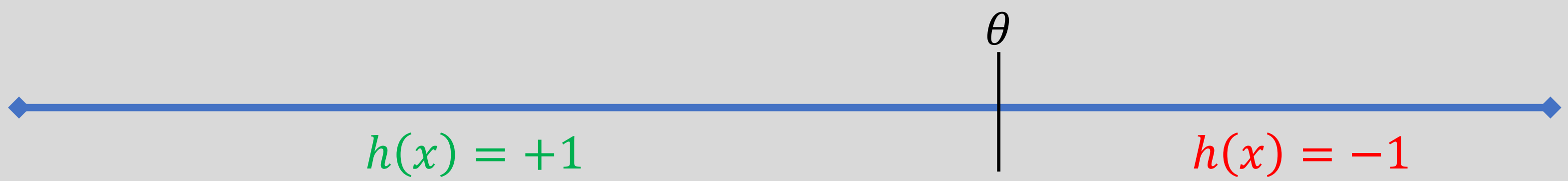
- $\log_2 |\mathcal{H}|$ measures complexity, gives bounds on number of mistakes ($\approx$ data required for learning), at least in realizable case
- Lots of interesting "universal" classes with small log-cardinality
- … but runtime is exponential (or worse)

- Issues we still need to worry about:
  - Computational efficiency
  - Errors (non-realizability)

Bonus slides—not required

Initial Segments: Can we bound the number of mistakes?

$\mathcal{H} = \{ [[x \le \theta]] \mid \theta \in \mathbb{R} \}$ $\quad x \in [0,1]$ $\quad$ e.g. $x = \frac{\#\text{CAPS in } x}{\text{total } \#\text{chars}}$



- **Try it out!**
- Can you think of a rule that will limit the number of mistakes?
- Or play the adversary: no matter what the learning rule predicts, you can always force many mistakes?

Bonus slides—not required

Initial Segments: Can we bound the number of mistakes?

$\mathcal{H} = \{ [[\, x \leq \theta \,]] \mid \theta \in \mathbb{R} \}$ $\quad x \in [0,1]$ $\quad$ e.g. $x = \frac{\#\text{CAPS in } x}{\text{total } \#\text{chars}}$

- **Theorem**: For any learning rule A, there exists a sequence realized by $\mathcal{H}$, on which A makes a mistake on every round
- Proof:
  - $x_1 = 0.5$
  - $y_t = -\hat{y}_t$
  - $x_{t+1} = x_t + y_t 2^{-(t+1)}$
  - Realized by $\theta = 0.5 + \sum_t y_t 2^{-(t+1)}$

Bonus slides—not required

So we really can't learn initial segments (which are linear predictors, so we can't learn linear predictors?)

- Answer 1:
  - Counterexample based on extremely high resolution
  - If we discretize $\theta \in \left\{0, \frac{1}{r}, \frac{2}{r}, \frac{3}{r}, \ldots, 1\right\}$, $\log_2|\mathcal{H}| = \log_2(r+1)$
  - More generally, for linear predictors over $\mathcal{H}_{\text{linear}} = \left\{\ h_w : x \mapsto sign(\langle w, x\rangle)\ \middle|\ w \in \mathbb{R}^d\ \right\}$:
$$\log|\mathcal{H}_{\text{linear}}| = O(d\ \log r) = O\big(d \cdot (\text{\#bits per number})\big)$$
  - But runtime of MAJORITY would still be $O\left(r^d\right)$…
  - Using [Online Ellipsoid](): Can ensure $O(d^2 \log(rd))$ mistakes in time $poly(d)$
    keep track of outer ellipsoid containing all consistent predictors, i.e.
$$V_{t+1} = \text{smallest}-\text{enclosing}-\text{ellipsoid}(\{w \in V_t | sign(\langle w, x_t\rangle) = y_t\})$$
  - Also: randomized poly-time methods can approximate MAJORITY and further reduce mistake bound

- Answer 2:
  - Counterexample based on very specific sequence, in very specific order
  - What happens if examples (x_t, y_t) come in a random order?

From Adversarial Online to Statistical

- What if data not *exactly* realized by $\mathcal{H}$? How do we deal with errors?
- Want to avoid non-learnability due to very specific, adversarial, order of examples

See optional expansion material on course website, or challenge problem in HW2

- Also, want to depart from online model where we always receive the correct label after each prediction.
- Instead:
  1. Learn from labeled training data
  2. Ship your predictor
  3. Get tested on how well the predictor you shipped does on future data

The Statistical Learning Model

should never be uniform

- **Unknown source distribution $\mathcal{D}$ over (x, y)**
  - Describes "reality". What we want to classify, and what should it be classified as.
  - E.g. joint distribution over ( , b)
  - Can think of $\mathcal{D}$ as: distribution over x and $y|x = f(x)$
    - Distribution over images we expect to see (we don't expect to see uniformly distributed images: ), and what character each image represents

it i write "b" "what characters would this be?

  - Or, as: distribution over y and over $x|y$
    - Distribution over characters ('e' more likely then '&'), and for each character, over possible images of that character

- **Goal: find predictor h with small expected error:** (also called *generalization error*, *risk* or *true error*)

$$L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$$

or

$$L_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[loss(h(x); y)]$$

$$loss(\hat{y}; y) = \begin{cases} 0, \hat{y} = y \\ 1, \hat{y} \neq y \end{cases}$$

did i make a mistake or not?

- **Based on a sample $S = ((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m))$** of m training points $(x_t, y_t) \sim \text{i.i.d.}\, \mathcal{D}$

(we can also write: $S \sim \mathcal{D}^m$)

The Statistical Learning Model

- Unknown source distribution $\mathcal{D}$ over (x, y)
- Goal: find predictor h with small expected error:

$$L_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[loss(h(x); y)] = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$$

- Based on sample $S = ((x_1, y_1), (x_2, y_2), \dots, (x_m, y_m))$ of m training points

$$(x_t, y_t) \sim \text{i.i.d.}\, \mathcal{D} \qquad (\text{i.e. } S \sim \mathcal{D}^m)$$

- **Statistical (batch) learning:**
  1. Receive training set $S \sim \mathcal{D}^m$
  2. Learn $h = A(S)$ using learning rule $A: (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{Y}^{\mathcal{X}}$ → what Learning Rule?
  3. Use h on future examples drawn form $\mathcal{D}$, suffering expected error $L_{\mathcal{D}}(h)$

- **Main assumption:**
  - i.i.d. samples
  - Samples drawn from distribution $\mathcal{D}$ we will later use the predictor on

→ Rarely satisfied:

• e.g. I'm not going to ask 10K ppl to write 1 char
I'll ask 100 ppl to write 100 chars.

• e.g. time dependent data - 30 fps self-driving car over 1min
we don't have 30·60 sec i.i.d. samples.

→ Also Rare:

• Train car based on pics in Chicago vs driving in Delhi

Expected vs Empirical Error

- What we care about is the **expected error**

$$L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[loss(h(x);y)]$$

- Why not just minimize it directly?

$$h^* = \arg\min L_{\mathcal{D}}(h)$$

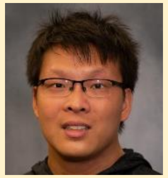
we don't know D!

The "Bayes Optimal" predictor

- Instead, a given sample S we can calculate the **empirical (training) error**

$$L_S(h) = \frac{1}{m}\sum_{t=1}^{m} loss(h(x_t);y_t)$$

Details in tutorial



- Is it a good estimate for the expected error?

For any h, with probability at least $1-\delta$: $|L_{\mathcal{D}}(h) - L_S(h)| \le \sqrt{\frac{\log 2/\delta}{2m}} \le 0.02$

$\delta = 0.001$

$m = 10000$

comes from binomial tail bound.

When $0 \le loss \le 1$, e.g. 0/1 error

Empirical Risk Minimization

$$L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[loss(h(x); y)] \qquad L_S(h) = \frac{1}{m}\sum_{t=1}^{m} loss(h(x_t); y_t)$$

$$loss(\hat{y}; y) = \begin{cases} 0, & y = \hat{y} \\ 1, & y \neq \hat{y} \end{cases}$$

$$ERM(S) = \hat{h} = \arg\min_h L_S(h)$$

what does $\hat{h}$ look like?

- Solution: memorize

$$\hat{h}(x) = \begin{cases} y_t, & x = x_t \\ 0, & \text{otherwise} \end{cases}$$

Empirical Risk Minimization

$L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[loss(h(x);y)]$ $\quad$ $L_S(h) = \frac{1}{m}\sum_{t=1}^{m} loss(h(x_t);y_t)$

$$loss(\hat{y};y) = \begin{cases} 0, & y = \hat{y} \\ 1, & y \neq \hat{y} \end{cases}$$

We don't want to memorize learn over hypothesis in the class.

$$ERM_{\mathcal{H}}(S) = \hat{h} = \arg\min_{h\in\mathcal{H}} L_S(h)$$

Example:
- $\mathcal{H} = \{\text{decision trees of depth } \leq 5\}$
- $ERM_{\mathcal{H}}(S)$ - find decision tree $\hat{h}$ of depth ≤ 5 that's best on the training data S, i.e. with minimum training error (smallest number of mistakes on S)

Example:
- $\mathcal{H} = \mathcal{Y}^{\mathcal{X}}$ (all functions $h: \mathcal{X} \to \mathcal{Y}$)
- $ERM_{\mathcal{H}}(S)$ - memorize

$$\hat{h}(x) = \begin{cases} y_t, & x = x_t \\ 0, & \text{otherwise} \end{cases}$$

This is ERM before.

- We said that for any h, with probability at least $1-\delta$: $|L_{\mathcal{D}}(h) - L_S(h)| \leq \sqrt{\frac{\log 2/\delta}{2m}}$

$L_S(\hat{h}) = 0$, but is $L_{\mathcal{D}}(\hat{h}) \leq \sqrt{\frac{\log 2/\delta}{2m}} \leq 0.02$ (with $m = 10{,}000$) ???

No, if $\mathcal{H}$ is too large.

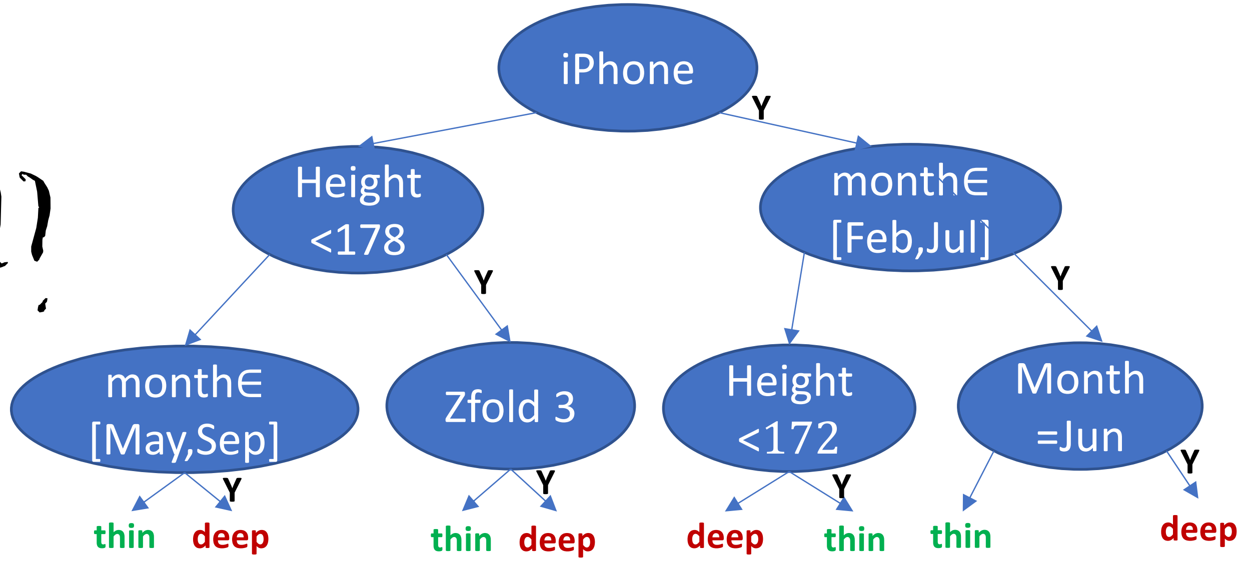
Overfitting

- $\mathcal{X}$ = students in UChicago
- Predict $y =$ prefer deep dish or thin crust, $\mathcal{D}$ is joint distribution on (student,pizza-pref)
- $\mathcal{H}$ = Decision tree of depth five (63 nodes) over: birth month in some range, phone, threshold on height, parity of floor you live on
- S (training set) = students in the class

We can probably find some crazy tree that works for the students in the class, i.e. with $L_S(\hat{h}) = 0$ but will it generalize?? Will it get small $L_{\mathcal{D}}(\hat{h})$?

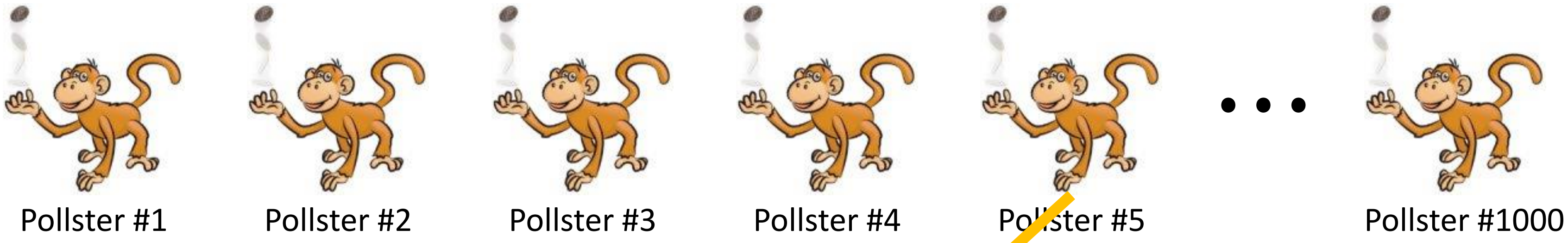
no.

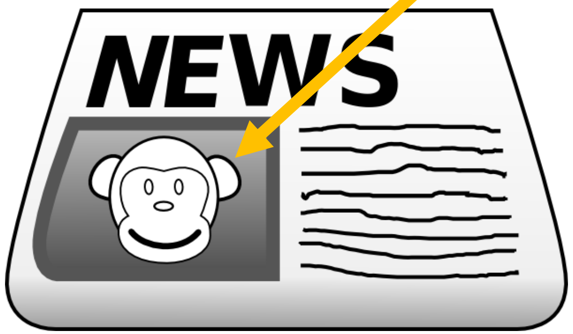
Q: what is being violated?



Predicting Election Outcome in Eight Races

P(all correct) $= 2^{-8} = {}^{1}/_{256} = 0.004$





at least one is

P(all correct) $\approx 1 - e^{-\frac{1000}{256}} = 0.98$

- For any particular h, chosen **before we see the sample**, we can ensure that with high probability $L_S(h)$ is close to $L_{\mathcal{D}}(h)$:

$$\forall_h \ \mathbb{P}_S(|L_S(h) - L_{\mathcal{D}}(h)| \leq t) \leq 1 - 2e^{-2mt^2}$$

- But there is some tiny probability $|L_S(h) - L_{\mathcal{D}}(h)|$ is large…
  With many h to consider, the probability adds up,
  and our search might really prefer that one lucky h with $L_S(h) \ll L_{\mathcal{D}}(h)$

In fact we are biased towards finding the h that screws everything up.

- We want to ensure that with high probability, **all** empirical errors are close to their expectations:

$$\mathbb{P}_S(\forall_h \ |L_S(h) - L_{\mathcal{D}}(h)| \leq t) \leq \cdots$$

This is much harder to bound.

Bounds presented are for bounded loss, $0 \leq loss \leq 1$, e.g. 0/1 error
Results can be extended to unbounded loss, but beyond scope of course.

- For any particular h, chosen **before we see the sample**, we can ensure that with high probability $L_S(h)$ is close to $L_\mathcal{D}(h)$:

$$\forall_h \ \mathbb{P}_S(|L_S(h) - L_\mathcal{D}(h)| \le t) \le 1 - 2e^{-2mt^2}$$

- We want to ensure that with high probability, **all** empirical errors are close to their expectations

Easiest way to bound is union bound.

$$\mathbb{P}_S(\exists_{h\in\mathcal{H}} \ |L_S(h) - L_\mathcal{D}(h)| \ge \epsilon) \quad \le \quad \sum_{h\in\mathcal{H}} \mathbb{P}_S(|L_S(h) - L_\mathcal{D}(h)| \ge \epsilon) \quad \le \quad |\mathcal{H}| \cdot 2e^{-\epsilon^2/m}$$

➔ For any hypothesis class $\mathcal{H}$ and any $\mathcal{D}$, $\mathbb{P}_{S\sim\mathcal{D}^m}\left[\forall_{h\in\mathcal{H}}, |L_S(h) - L_\mathcal{D}(h)| \le \sqrt{\frac{\log|\mathcal{H}| + \log 2/\delta}{2m}}\right] \ge 1 - \delta$

- Another way to view this: $\mathbb{P}_S\left[|L_S(h) - L_\mathcal{D}(h)| \ge \sqrt{\frac{\log 2/\delta_h}{2m}}\right] \le \boldsymbol{\delta_h} \stackrel{\text{def}}{=} \frac{\delta}{|\mathcal{H}|}$

and then $\log 2/\delta_h = \log 2|\mathcal{H}|/\delta = \boldsymbol{\log|\mathcal{H}|} + \log 2/\delta$

want to choose $\delta_\mathcal{H}$ s.t. adding it up will still give us a good bound.

Bounds presented are for bounded loss, $0 \le loss \le 1$, e.g. 0/1 error
Results can be extended to unbounded loss, but beyond scope of course.

with prob $\geq 1-\delta$

$\hat{h} = ERM_{\mathcal{H}}(S) = \arg\min_{h \in \mathcal{H}} L_S(h)$

- Theorem: For any $\mathcal{H}$ and any $\mathcal{D}$, $\forall^{\delta}_{S \sim \mathcal{D}^m}$,

$$L_{\mathcal{D}}(\hat{h}) \leq L_S(\hat{h}) + \sqrt{\frac{\log|\mathcal{H}| + \log {}^2/_{\delta}}{2m}}$$

➔ For any hypothesis class $\mathcal{H}$ and any $\mathcal{D}$, $\mathbb{P}_{S \sim \mathcal{D}^m}\left[\forall_{h \in \mathcal{H}}, |L_S(h) - L_{\mathcal{D}}(h)| \leq \sqrt{\frac{\log|\mathcal{H}| + \log {}^2/_{\delta}}{2m}}\right] \geq 1-\delta$

Bounds presented are for bounded loss, $0 \leq loss \leq 1$, e.g. 0/1 error
Results can be extended to unbounded loss, but beyond scope of course.

with prob $\geq 1-\delta$

$\hat{h} = ERM_{\mathcal{H}}(S) = \arg\min_{h\in\mathcal{H}} L_S(h)$

- Theorem: For any $\mathcal{H}$ and any $\mathcal{D}$, $\forall^{\delta}_{S\sim\mathcal{D}^m}$,

$$L_{\mathcal{D}}(\hat{h}) \leq L_S(\hat{h}) + \sqrt{\frac{\log|\mathcal{H}| + \log {}^{2}/_{\delta}}{2m}}$$

- Without ANY assumptions about the source distribution (i.e. about reality), if we find a predictor h with low $L_S(h)$, we can promise (with high probability) that it will perform well on future examples!

Should we fire the programmer? Yes! Why? idk.

- Instead, use independent test set S' (e.g. split available examples into a training set S and test set S').

$$L_{\mathcal{D}}(A(S)) \leq L_{S'}(A(S)) + \sqrt{\frac{\log 1/\delta}{2|S'|}}$$

Random, but depends only on S, independent of S'

- Even better: tighter numerical confidence intervals using inverse CDF of Binomial or its Gaussian approx

- **Crucial: $h = A(S)$ should be fixed before peeking at S'!**

Disclaimer: all bounds presented are for bounded loss, $0 \leq loss \leq 1$, e.g. 0/1 error
Results can be extended to unbounded loss (eg squared loss), but beyond scope of course.

- Theorem: For any $\mathcal{H}$ and any $\mathcal{D}$, $\forall^{\delta}_{S\sim\mathcal{D}^m}$,

with prob $\geq 1-\delta$

$\hat{h} = ERM_{\mathcal{H}}(S) = \arg\min_{h\in\mathcal{H}} L_S(h)$

$$L_{\mathcal{D}}(\hat{h}) \leq L_S(\hat{h}) + \sqrt{\frac{\log|\mathcal{H}| + \log {}^{2}/_{\delta}}{2m}}$$

Post-Hoc Guarantee

- Theorem: For any $\mathcal{H}$ and any $\mathcal{D}$, $\forall^{\delta}_{S\sim\mathcal{D}^m}$,

$$L_{\mathcal{D}}(\hat{h}) \leq \inf_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + 2\sqrt{\frac{\log|\mathcal{H}| + \log {}^{2}/_{\delta}}{2m}}$$

A-priori Guarantee

Proof: if indeed $\forall_{h\in\mathcal{H}}, |L_{\mathcal{D}}(h) - L_S(h)| \leq \sqrt{\cdots}$, then:

$$L_{\mathcal{D}}(\hat{h}) \leq L_S(\hat{h}) + \sqrt{\cdots} \leq L_S(h^*) + \sqrt{\cdots} \leq L_{\mathcal{D}}(h^*) + \sqrt{\cdots} + \sqrt{\cdots}$$

$h^* = \arg\min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$

Disclaimer: all bounds presented are for bounded loss, $0 \leq loss \leq 1$, e.g. 0/1 error

Results can be extended to unbounded loss (eg squared loss), but beyond scope of course.

with prob $\geq 1-\delta$

$$\hat{h} = ERM_{\mathcal{H}}(S) = \arg\min_{h \in \mathcal{H}} L_S(h)$$

- Theorem: For any $\mathcal{H}$ and any $\mathcal{D}$, $\forall^{\delta}_{S \sim \mathcal{D}^m}$,

$$L_{\mathcal{D}}(\hat{h}) \leq L_S(\hat{h}) + \sqrt{\frac{\log|\mathcal{H}| + \log {}^{2}/_{\delta}}{2m}}$$

Post-Hoc Guarantee

- Theorem: For any $\mathcal{H}$ and any $\mathcal{D}$, $\forall^{\delta}_{S \sim \mathcal{D}^m}$,

$$L_{\mathcal{D}}(\hat{h}) \leq \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + 2\sqrt{\frac{\log|\mathcal{H}| + \log {}^{2}/_{\delta}}{2m}}$$

A-priori Guarantee

- Conclusion: For any $\delta, \epsilon > 0$, using

$$m = 2\frac{\log|\mathcal{H}| + \log {}^{2}/_{\delta}}{\epsilon^2}$$

samples is enough to ensure $L_{\mathcal{D}}(\hat{h}) \leq L_{\mathcal{D}}(h^*) + \epsilon$ w.p. $\geq 1-\delta$

Sample Complexity Bound

Disclaimer: all bounds presented are for bounded loss, $0 \leq loss \leq 1$, e.g. 0/1 error
Results can be extended to unbounded loss (eg squared loss), but beyond scope of course.

Complexity of Learning

$$\hat{h} = \arg\min_{h \in \mathcal{H}} L_S(h)$$

$$L_{\mathcal{D}}(\hat{h}) \le \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + 2\sqrt{\frac{\log|\mathcal{H}| + \log {}^{2}/_{\delta}}{2m}}$$

$$m = O\left({}^{\log|\mathcal{H}|}/_{\epsilon^2}\right)$$

$\mathcal{H} = \{ f \}$ full knowledge — $\mathcal{H}_{100}$ — $\mathcal{H}_D$ — $CONJ$ — Decision Trees — python programs — $\mathcal{H} = \mathcal{Y}^{\mathcal{X}}$ no free lunch

Lecture 2: Summary

- Basic Concepts: domain $\mathcal{X}$, label set $\mathcal{Y}$, predictor h, hypothesis class $\mathcal{H}$
- Online Learning Model
    - No Free Lunch
    - $\log_2|\mathcal{H}|$ mistake bound
- Complexity Control; Specific prior knowledge vs #mistakes
- Importance of computational issues
- Why statistical?
    - Deal with errors
    - Train-then-ship
- Statistical Learning Model: source distribution $\mathcal{D}$, training set S, exp. error $L_{\mathcal{D}}$
- ERM as a template learning rule
- Union bound ➔ again $\log_2|\mathcal{H}|$ controls complexity