Lecture 16

learning Boolean functions

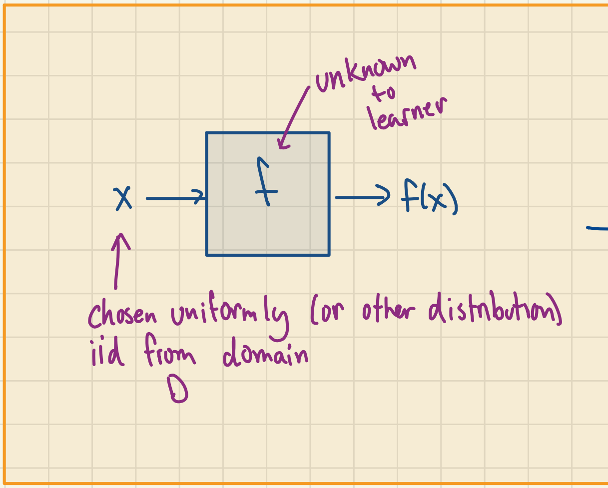
- a model
- an example: conjunctions
- Occam's razor
- Fourier-based learning algorithms

Learning

Learn from random, uniform examples

How to formalize?

lots of other models – will mention some others later



Example oracle $Ex(f)$

$x_1, f(x_1)$
$x_2, f(x_2)$
$\vdots$
$x_m, f(x_m)$

m random, labelled examples

After seeing several examples, learner should output hypothesis h.

what do we hope h satisfies?

- hopefully $h = f$ ← asking too much?
- at least $\underbrace{dist(h,f)}_{\Pr_{x \in D}[h(x) \neq f(x)]} \le \varepsilon$

Valiant's PAC model "probably approximately correct"

what distribution on inputs do we use?

today uniform

in general, match distribution of example oracle

Common terms for same thing:

- $\underset{x \in D}{dist}(h,f)$
- $\underset{x \in D}{error}(h)$ (w.r.t. f is understood from context)
- h is ε-close to f (dist D understood from context)

Note in above:

$x \in D$ can be chosen according to uniform or any other prespecified distribution

Note if f is arbitrary, there is nothing you can do that is "efficient" in terms of sample complexity (e.g. you can't learn a random fctn f without seeing the value of f for most inputs)

However, if you know something about f, there may be hope.

here: what if you know that f is a member of fctn family $\mathcal{C}$?

e.g. $\mathcal{C}$ = linear fctns

k-term DNF

$\vdots$

def uniform distribution learning algorithm for concept class $\mathcal{C}$ is algorithm $\mathcal{A}$ s.t.

- $\mathcal{A}$ given $\varepsilon, \delta > 0$
  access to $Ex(f)$ for $f \in \mathcal{C}$

- $\mathcal{A}$ outputs h st. with prob $\geq 1-\delta$
  $\underbrace{\text{error}(h) \text{ w.r.t. } f \text{ is}}_{h \text{ is } \varepsilon\text{-close to } f} \leq \varepsilon$

Parameters of interest:

- m # samples used by $\mathcal{A}$ "Sample Complexity"
- ε accuracy parameter
- δ confidence parameter
- runtime?   hope for $\text{poly}\left(\log(\text{domainsize}), \frac{1}{\varepsilon}, \frac{1}{\delta}\right)$

- description of h?
    - should it be similar to description of fctns in $\mathcal{C}$? "proper learning"
    - at least should be relatively $\underbrace{\text{compact}}_{O(\log|\mathcal{C}|)}$ + efficient to evaluate

Remarks

- as before, dependence on δ needn't be more than $O(\log(1/\delta))$ why?

- uniform case is special case of PAC-model:
  given $Ex_{\mathcal{D}}(f)$ for unknown $\mathcal{D}$
  output h with small error with respect to same $\mathcal{D}$
  (some $\mathcal{D}$ can be harder than others)

Efficient learning algorithm for conjunctions:

$\mathcal{C}$ = conjunctions over $\{0,1\}^n$

i.e. $f(x) = x_i x_j \bar{x}_k$

note:

- can't hope for 0-error from subexponential # of random examples

  e.g. how to distinguish $f(x) = x_1 x_2 \cdots x_n$ from $f'(x) = 0 \ \forall x$?

Poly time algorithm:

- draw $\text{poly}(\frac{1}{\varepsilon})$ random examples
- estimate $\Pr[f(x)=1]$ to additive error $\pm \frac{\varepsilon}{4}$
- if estimate $< \varepsilon/2$, output "$h(x)=0$" & halt

  else estimate $\geq \varepsilon/2$ so $\Pr[f(x)=1] > \frac{\varepsilon}{4}$

  $\Rightarrow$ see new random positive example every (at most) $O(1/\varepsilon)$ examples

- collect $\underline{?}$ more positive examples
- let $V = \{$vars set same way in each positive example$\}$
- Output $h(x) = \bigwedge_{i \in V} x_i^{b_i}$

  b_i tells us if x_i complemented or not

Behavior of poly time algorithm:

for i in conjunction:

must be set same way in each positive example $\Rightarrow$ in V

for i not in conjunction:

$Pr[i \in V] \leq Pr[i$ set same way in each of k positive examples$]$

$$\leq \frac{1}{2^{k-1}}$$

$Pr[$any i not in conjunction manages to survive$]$

$$\leq \frac{n}{2^{k-1}}$$

$\leq \delta$ if pick $k = \log \frac{n}{\delta}$

So if use $\Omega(\log \frac{n}{\delta})$ positive examples

or $\Omega(\frac{1}{\varepsilon} \log \frac{n}{\delta})$ total examples, will suffice to rule out all $i \notin$ conjunction.

Occam's Razor

"high level claim":

if ignore runtime, then learning is easy

with respect to sample complexity

Brute force algorithm

- draw $M = \frac{1}{\varepsilon}\left(\ln|C| + \ln\frac{1}{\delta}\right)$ uniform examples
- search over all $h \in C$ until find one consistent with all examples. Output it. (choose arbitrarily if ≥ 1 such h works)

Behavior of brute force algorithm

what should behavior be?

- f is a good thing to output ✓
- what is a bad thing to output?

h is "bad" if error(h) w.r.t. $f \geq \varepsilon$

$\Pr[$bad h consistent with examples$]$

$$\leq (1-\varepsilon)^M$$

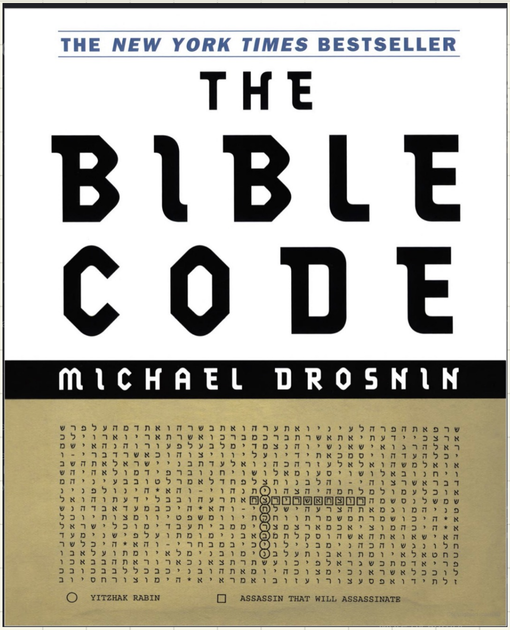
$\Pr[$any bad h consistent with examples$]$

$$\leq |\mathcal{C}| \cdot (1-\varepsilon)^M \qquad \text{union bnd}$$

$$\leq |\mathcal{C}| (1-\varepsilon)^{\frac{1}{\varepsilon}\left(\ln |\mathcal{C}| + \ln \frac{1}{\delta}\right)}$$

$$\leq \delta$$

$\therefore$ unlikely to output any bad h



divine inspiration?

Coincidence?

not enough samples to kill off union bound?

<u>Comments:</u>

- proof didn't use anything special about uniform distribution

  actually works for any dist $\mathcal{D}$ as long as error defined w.r.t. same $\mathcal{D}$ as sample generator

- Once have good h

  1) can predict values of f on new random inputs since
     according to $\mathcal{D}$
     $$\Pr_{x \in \mathcal{D}}[f(x)=h(x)] \geq 1-\delta$$

  2) can compress description of samples

     $(x_1, f(x_1)) (x_2, f(x_2)), \dots (x_m, f(x_m))$ $\quad m(\log|D| + \log|R|)$ range of f

     $\Downarrow$ $\qquad\qquad\qquad\qquad \Downarrow$

     $x_1 \dots x_m$, description of h $\quad m \cdot \log|D| + \log|C|$

So learning, prediction + compression are related

learning $\Rrightarrow$ prediction & compression

formal relations in other direction too

Occam's razor:

Simplest explanation is best