2. More Intro, Ch. 1, example & terminology

Friday, January 7, 2022 5:19 PM

A simple, delicious example

Is this papaya ripe? 

Our data for a given papaya are its color and firmness

These are its features

Write $x = \begin{bmatrix} \text{color} \\ \text{firmness} \end{bmatrix} \in \mathbb{R}^2$, $x \in X \subseteq \mathbb{R}^2$ (X: domain)

⚠ Conceptually, usually better to think of x as the whole papaya, and only later worry about what features to use to represent it.

Ex. "x is an email" ... but later pick out features like # of capital letters.

This choice of features may limit our accuracy (ie. does a papaya's taste really only depend on color and firmness?)

Choosing features encodes an inductive bias.

Our goal is to generalize (from one, or a few, papayas, to all papayas)

i.e. inductive reasoning / inference

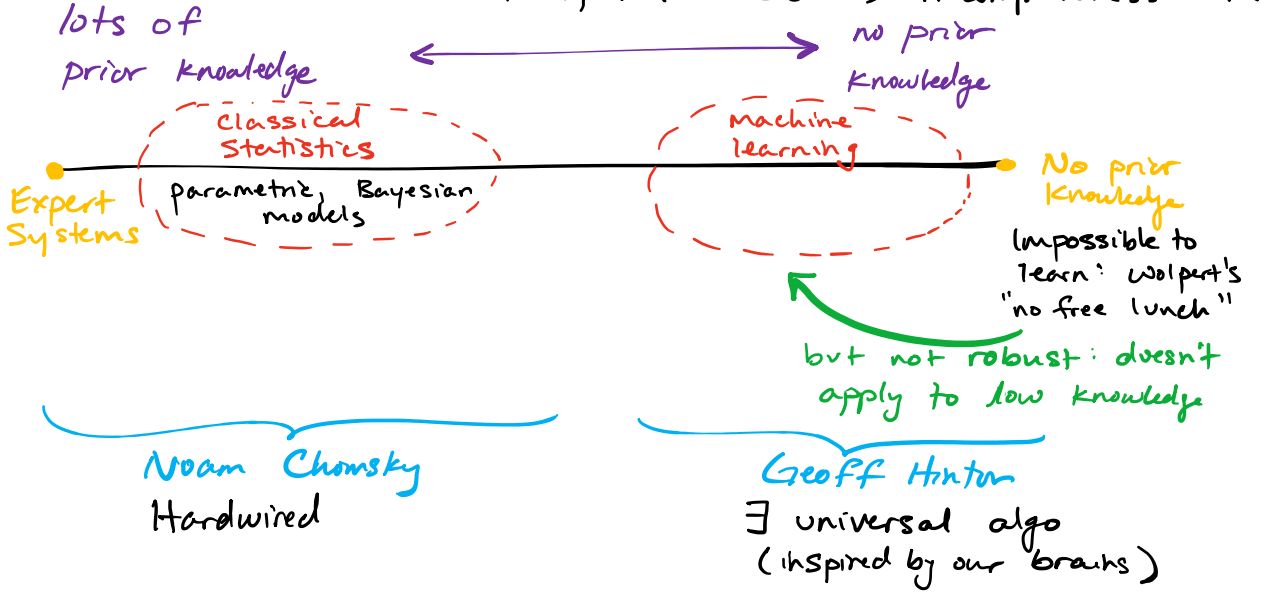
We may have prior knowledge (color, firmness are probably useful, while weight, density, size are not helpful for predicting taste)

... but prior knowledge is inductive bias

(i.e., what if we're wrong!)

Do you need prior knowledge at all? YES

It's a fact, like Gödel's incompleteness thm



Basically, we have to make assumptions
(and realize this is adding possible bias)

Back to our example (and terminology)

- Domain X, and $x \in X$ is an instance (a single papaya, or its features)
  $X \subseteq \mathbb{R}^d$ means d features
- Labels $y \in \mathcal{Y}$ Regression: $\mathcal{Y} = \mathbb{R}$
  Binary classification: $\mathcal{Y} = \{0,1\}$ or $\{-1,1\}$
- Training data $S = ((x_i, y_i))_{i=1}^{m}$
  Often called training "set" but technically ① order might matter ② duplicates allowed
  we'll try to be consistent with "m" = # training data
- Learner (you!) output is a

function / prediction rule / hypothesis / classifier

$h : X \to Y$

Sometimes we use "A" to mean an algorithm that, given training data, outputs such a hypothesis:

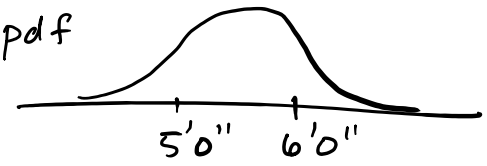
$h = A(S)$

- **Data model**

The big, not always realistic, assumption of statistical learning is that there is a probability distribution over papayas (or whatever) that describes how likely we are to observe them.

Ex: x = person's height
y = how high they can jump

Given a new person, we treat x as a random variable



This is an idealization!

Call this distribution D ← my script D, $D(\ )$ is a probability

Unlike Bayesians, we do not assume we know which distribution D is!

For labels y, for now we assume these are a deterministic function of the data x,

(i.e., $y_i = f(x_i)$ ← "oracle" (unknown to us))

Our dataset $S = (\underbrace{(x_i, y_i)}_{z_i})_{i=1}^m$, so our assumption is (x_i) is iid, $x_i \sim D$

aka $S \sim D^m$ ← note the notation

- Measure of success

"Risk", "True Risk", "True Error", "Generalization Error" (in Mohri)

("Realizable" case)

SAVE!

$$L_{D,f}(h) = \mathop{\mathbb{E}}_{x \sim D} \ell(h, (x, \underbrace{f(x)}_{y}))$$

Note: we will slightly generalize this later when $y \neq f(x)$

↑ Mohri uses "R" for Risk

↑ loss function (up to us to pick a good one)

Special Case: binary classification

in this case, unless otherwise mentioned, we always use the "obvious" loss, the 0-1 loss

$$\ell_{0-1}(h, (x,y)) = \mathbb{1}_{h(x)=y}(x) := \begin{cases} 0 & h(x)=y \\ 1 & h(x)\neq y \end{cases}$$

and then $\mathbb{E}\,\ell_{0-1}(\cdot,\cdot)$ simplifies

$$L_{D,f}^{0-1}(h) = \mathop{\mathbb{P}}_{x \sim D}(h(x) \neq f(x)) = D(\{x : h(x) \neq f(x)\})$$

= probability we misclassify

Summary so far

- We know X, Y (sets), and metrics ℓ
- We observe $S = (z_i)_{i=1}^m$ $\quad z_i = (x_i, y_i)$
- We assume $x_i \sim D$ iid, $\quad y_i = f(x_i)$

- We don't know D nor f
- Goal: design an algorithm A which outputs a classifier h which has low risk.

Algorithms

We'll introduce the succesful (theory and practice) "Empirical Risk Minimization" (ERM) framework

Def: The Empirical Risk is

$$\hat{L}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i) \qquad z_i := (x_i, y_i)$$

aka Empirical Loss

Mohri writes $\hat{R}$

SS writes L_S

I write $\hat{L}_S$

$$\left(= \frac{1}{m} \left| \{ i \in [m] : h(x_i) \neq y_i \} \right| \right) \qquad [m] := \{1, 2, \ldots, m\}$$

for the special case of 0-1 loss

(*) If you give me a fixed classifier h, then I draw $S \sim D^m$, then $\hat{L}_S(h)$ is a very good proxy for $L_D(h)$. [we'll use this later for validation].

So, it might seem reasonable to try a naive ERM

$$A(S) = \operatorname{argmin}_h \hat{L}_S(h)$$

notation:

$4 = \min_x x^2 + 4$

$0 = \operatorname{argmin}_x x^2 + 4$

$17 \in \operatorname{argmin}_x 0 \cdot x + 3$

... but 2 issues:

(1) what set is h in? The set of all functions from X to Y. Unless X and Y are discrete, this set isn't something we can use on a computer

(2) related to above, there are likely too many h and we're going to **overfit**

in detail:

$$h_{\text{memorize},S}(x) := \begin{cases} y_i & \text{if } x = x_i \text{ for some } (x_i, y_i) \in S \\ 0 & \text{(or whatever) else} \end{cases}$$

so $\hat{L}_S(h_{\text{mem}}) = 0$

But (let's assume $|X| = \infty$ and D is absolutely continuous and binary classif. w/ 0-1 loss)

$$L_D(h_{\text{mem}}) = \mathbb{P}(h(x) \neq f(x))$$

since $h(x) = f(x)$ else

$$= D(\{h(x) \neq f(x)\}, x \text{ not in } S)$$

this depends on f, so it could be 0 (lucky), or $1/2$ (typical) or 1 (choose f adversarily)

(note: $D(x \text{ in } S) = 0$ since $|S| < \infty$)

Revisit (*)

what happened? If $h = \text{ERM}(S)$ then h is dependent on S and $\hat{L}_S(h)$ is not a good proxy for $L_D(h)$.

No law of large numbers...