Announcements

1. Calibration HW due Friday (22nd)
2. Scribe note sign-up sheet (add a link in class website)

Part I Statistical learning

- PAC learning (Probably Approximately correct)

Batch (supervised learning) pipeline

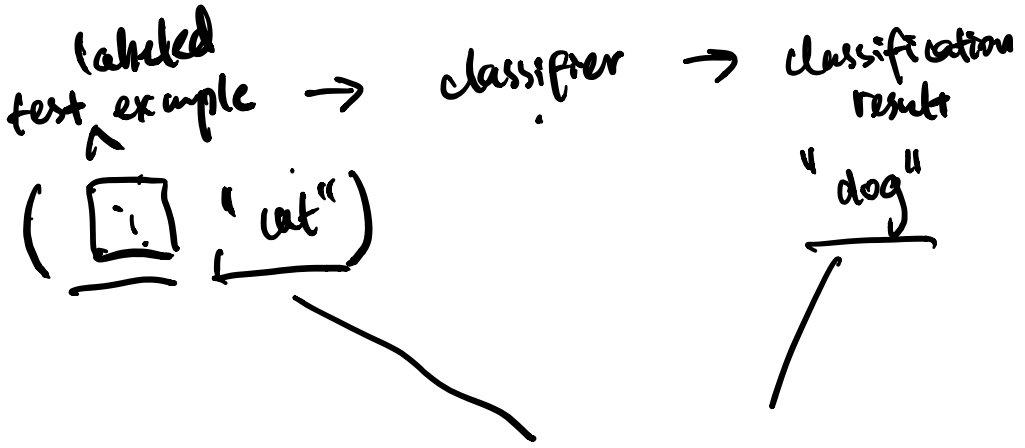
Training stage

training data → learning algorithm → classifier

Test stage

test example → classifier → classification result "dog"

Test stage (another view)



Assumption: training & testing examples come from the same distribution.

Qns:

1. How can we formalize this process mathematically?
2. Can we formally define what a good predictor / good learning algorithm should look like?

PAC Model (Valiant, 1984)

— instance domain $\mathcal{X}$: eg. images. $\mathcal{X} = \mathbb{R}^{64 \times 48 \times 3}$

— label space $\mathcal{Y} = \{-1, +1\}$ $+1$: "cat". -1 : "dog"

— data distribution D. supported on $\mathcal{X} \times \mathcal{Y}$.
(x, y)

— training set S : drawn from D.

$(x_1, y_1) \cdots (x_m, y_m)$ independent & identically distributed (i.i.d.)

$\sim D$

— classifier (hypothesis) $h : \mathcal{X} \to \mathcal{Y}$

 $\to h \to$ "cat"

— Hypothesis class (concept class) : a structured collection of classifiers.

$\mathcal{H} = \{$ all neural nets with ResNet-18 architecture $\}$

— Performance evaluation :

h (classifier)'s generalization error :=

$$\underset{(x,y)\sim D}{P}\left(h(x) \neq y\right) := err(h, D)$$

h's training error:

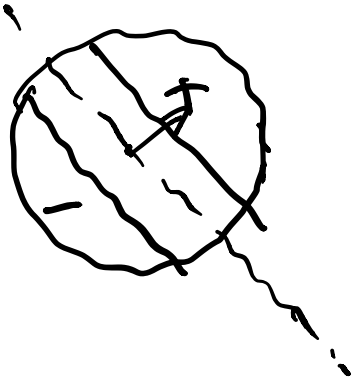
$$\frac{\sum_{i=1}^{m} I(h(x_i) \neq y_i)}{m} := err(h, S)$$

$$= \underset{(x,y)\sim \underset{S}{unif(S)}}{P}\left(h(x) \neq y\right).$$

$$I(A) = \begin{cases} 1 & \text{if } A \text{ is true} \\ 0 & \text{if } \ldots \text{ false} \end{cases}$$

- Realizable (separable): we are given the promise that there exists a classifier $h^* \in H$, such that $err(h^*, D) = 0$.

Agnostic: there may or may not exist a classifier in H w/ zero test error.



Def We call Alg A PAC learns hypothesis class H with sample complexity function: $f: (0,1) \times (0,1) \to \mathbb{N}$, if for any distribution D realizable with respect to H, $\varepsilon, \delta > 0$, when A receives $m \geq f(\varepsilon, \delta)$ $\overset{\text{iid}}{\wedge}$ training examples as input, it outputs a classifier $\hat{h}$, such that with probability $1-\delta$,

$$\text{err}(\hat{h}, D) \leq \varepsilon .$$

Remarks:

1. ε: target error rate

   δ: confidence parameter

2. polynomial running time required in Valiant's original paper, but not here.

3. m: sample complexity function: want it to be as small as possible.

   m here is distribution independent

   It may sometimes be better to consider distribution dependent sample complexity results.

$m(\varepsilon, \delta, D)$

4. If A returns $\hat{h}$ in $\mathcal{H}$; A is called a proper learning alg.

If A doesn't nec. return ~~—~~ ; ~~- —~~

improper le~~——~~ .

Sometimes improper learning is beneficial.

Sample complexity of finite hypothesis classes.

Result: each element can be represented by

(64 × # weights) bits.

- $\mathcal{H}$ is a finite set.
- Realizability

(x_1, y_1) — (x_n, y_n) → Alg → $\hat{h}$

The consistency algorithm:

1. Return $\hat{h}$ in $\mathcal{H}$ such that it agrees w/ all examples in S. i.e. for all i, $\hat{h}(x_i) = y_i$.

Theorem : ~~a~~ finite $\mathcal{H}$. If the consistency alg. is given m iid training examples from D realizable w.r.t. $\mathcal{H}$, then

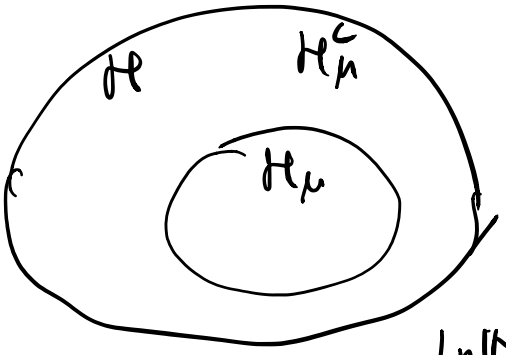
With probability $1-\delta$, its output $\hat{h}$ is such that

$$err(\hat{h}, D) \leq \underbrace{\varepsilon_R}_{\text{realizable}}(m, \delta, \ln|\mathcal{H}|) := \frac{\ln|\mathcal{H}| + \ln\frac{1}{\delta}}{m}.$$

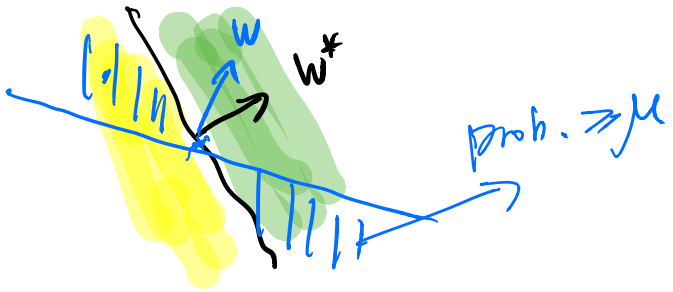
$\leq \varepsilon$

[In other words, it PAC learns $\mathcal{H}$ with sample complexity

$$f(\varepsilon, \delta) = \frac{1}{\varepsilon}\left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right).$$



$$\mathcal{H}_\mu = \left\{ h \in \mathcal{H} : err(h, D) > \mu = \frac{\ln|\mathcal{H}| + \ln\frac{1}{\delta}}{m} \right\}$$



Next class:

Show there exists an event E_μ such that $P(E) \geq 1-\delta$, for all $h \in \mathcal{H}_\mu$, there will be at least one i.

$$h(x_i) \neq y_i.$$