CSCI4230 Computational Learning Theory — Spring 2021

*Lecturer: Siu On Chan* — *Based on Rocco Servedio's notes*

# Notes 7: PAC model

## 1. Probably Approximately Correct

Valiant'84 "*Theory of the Learnable*"; Turing Award'14

Average case performance wrt a fixed instance distribution

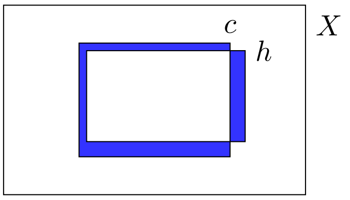
Assume instances $x \in X$ are drawn from a distribution $\mathcal{D}$ (unknown and arbitrary)

(Training phase) Given independent samples $(x, c(x))$, all labelled by an unknown concept $c \in \mathcal{C}$

**Goal:** Output hypothesis $h \subseteq X$ s.t. $\operatorname{err}_{\mathcal{D}}(h, c) := \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq c(x)]$ is small

Equivalently $\operatorname{err}_{\mathcal{D}}(h, c) = \mathbb{P}_{x\sim\mathcal{D}}[x \in h \triangle c]$

Recall $h \triangle c := (h \setminus c) \cup (c \setminus h)$ (symmetric difference)




error region $= h \triangle c$

Want small error region under $\mathcal{D}$

---

$\operatorname{err}_{\mathcal{D}}(h, c) > 0$ unavoidable: some $x \sim \mathcal{D}$ falls inside the error region

Error cannot always be small: if unlucky, training samples may be useless

**New goal:** With high probability over training samples and internal randomness (*probably*), output hypothesis $h \subseteq X$ with small error (*approximately correct*)

---

$\mathrm{EX}(c, \mathcal{D})$ = distribution of labelled samples $(x, c(x))$ when $x$ is drawn from $\mathcal{D}$

Algorithm $A$ **PAC learns** $\mathcal{C}$ if

- for any concept $c \in \mathcal{C}$
- for any distribution $\mathcal{D}$ over $X$
- for any **confidence** parameter $\delta > 0$ and **accuracy** parameter $\varepsilon > 0$
- when $A$ takes $m$ samples from $\mathrm{EX}(c, \mathcal{D})$
- with probability $\geqslant 1 - \delta$ over the samples and $A$'s randomness
- output hypothesis $h \subseteq X$ such that $\operatorname{err}_{\mathcal{D}}(h, c) \leqslant \varepsilon$

$A$ is **efficient** if runs in $\operatorname{poly}(1/\delta, 1/\varepsilon)$ time (plus two more conditions below)

- $\operatorname{poly}(1/\delta, 1/\varepsilon)$ means at most polynomial in $1/\delta$ and $1/\varepsilon$ (e.g. at most $\varepsilon^{-2}\delta^{-1}$)
- or $\operatorname{poly}(n, 1/\delta, 1/\varepsilon)$ time if $X = \{0, 1\}^n$ or $\mathbb{R}^n$

Run time always $\geqslant m$ (just to read the samples)

---

Algorithm $A$ only knows $\mathcal{C}, \delta, \varepsilon$

$A$ doesn't know $\mathcal{D}$ (distribution independent learning)

$A$ works under **any** $\mathcal{D}$ (strong assumption!), but error is also evaluated under $\mathcal{D}$

---

## 2. PAC learning rectangles

$X$ = the plane $= \mathbb{R}^2$ $\mathcal{C}$ = axis-aligned rectangles $= \{R(x_1, y_1, x_2, y_2) \mid x_1, y_1, x_2, y_2 \in \mathbb{R}\}$

where $R(x_1, y_1, x_2, y_2) = \{(x, y) \in \mathbb{R}^2 \mid x_1 \leqslant x \leqslant x_2 \text{ and } y_1 \leqslant y \leqslant y_2\}$

$\mathcal{D}$ = fixed distribution over $\mathbb{R}^2$ (unknown)

**Algorithm**

> Hypothesis $h$ = smallest rectangle containing all positive samples ($\emptyset$ if no positive samples)

**Claim 1.** *Given any $c \in \mathcal{C}$, if $m \geqslant (4/\varepsilon)\ln(4/\delta)$, with probability $\geqslant 1 - \delta$, the Algorithm outputs hypothesis $h$ with $\operatorname{err}_{\mathcal{D}}(h, c) \leqslant \varepsilon$.*

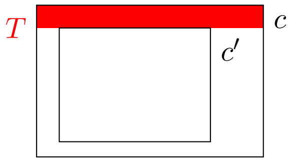
*Proof.* $h \subseteq c$ always

Want to show $h \triangle c = c \setminus h$ small under $\mathcal{D}$

**Case 1:** $c$ has probability mass at least $\varepsilon/4$ under $\mathcal{D}$

Can decompose $c \setminus h$ as union of four strips: top, left, bottom, right

Top strip $T$ = rectangle sharing top & left & right sides with $c$, has probability mass $\varepsilon/4$ under $\mathcal{D}$




Left, bottom, right strips defined analogously
$c'$ = $c$ with top, left, bottom, right strips removed
**Claim:** $c' \subseteq h$ with probability $\geqslant 1 - \delta$
Reason: if each strip contains a sample, then $c' \subseteq h$
    top strip has no sample with probability $(1 - \varepsilon/4)^m$
    same for other strips, union bound:

$$\mathbb{P}[\text{some strip has no sample}] \leqslant 4(1 - \varepsilon/4)^m \leqslant 4(e^{-\varepsilon/4})^m \leqslant \delta$$

$c' \subseteq h$ implies $\mathrm{err}_{\mathcal{D}}(h, c) \leqslant \varepsilon$
    because each strip has probability mass $\varepsilon/4$ under $\mathcal{D}$
**Case 2:** $c$ has probability mass less than $\varepsilon/4$ under $\mathcal{D}$
    Then $c \setminus h$ must have probability mass less than $\varepsilon$ □

---

## 3. Hypothesis size

some concepts $c(x)$ have a natural **size**    (e.g. #bits needed to describe $c$)
e.g. $\mathcal{C}$ = DNF formulae over $X = \{0,1\}^n$
every boolean function $f : X \to \{0,1\}$ can be represented as a DNF
    some as a 2-term DNF    (e.g. $f(x) = (\overline{x}_1 \wedge \overline{x}_2 \wedge x_6) \vee (x_9 \wedge \overline{x}_4 \wedge x_2)$)
    some requires $\geqslant 2^{\sqrt{n}}$ terms
$\mathrm{size}(f)$ = size of the smallest representation of $f$ in $\mathcal{C}$
    e.g. when $\mathcal{C} = \{\mathrm{DNF}\}$, sometimes $\mathrm{size}(f)$ may be #terms
Redefinition:    PAC learning Algorithm $A$ is **efficient** if runs in time $\mathrm{poly}(1/\delta, 1/\varepsilon, \mathrm{size}(c))$
    or $\mathrm{poly}(n, 1/\delta, 1/\varepsilon, \mathrm{size}(c))$ if $X = \{0,1\}^n$ or $\mathbb{R}^n$
    $c$ = target concept
    in particular, $A$ cannot output $h$ with large $\mathrm{size}(h)$
    Algorithm knows $\mathcal{C}, \delta, \varepsilon, \mathrm{size}(c)$
Some $\mathcal{C}$ may not have interesting size measure; size can be ignored
    e.g. monotone conjunctions have size $\leqslant n$

---

## 4. Efficient hypothesis

Often PAC learning Algorithm $A$ outputs hypothesis $h : X \to \{0,1\}$ that is itself a **program**
Not useful if $h$ too slow
If $X = \{0,1\}^n$ or $\mathbb{R}^n$, hypothesis $h$ is **polynomially evaluatable** if $h$ runs in $\mathrm{poly}(n)$ time
PAC learning Algorithm $A$ is **efficient** if it additionally outputs polynomially evaluatable hypothesis
e.g. inefficient $A$:
    stores all training samples in $h$
    then $h$ exhaustively searches for smallest DNF consistent with all training samples