COMP 585 **Spring 2024**

Probabilistic Toolkit for Learning and Computing *Instructor: Maryam Aliakbarpour*

March 27th, 2024 *Scriber: Erdong Hu*

Lecture 13

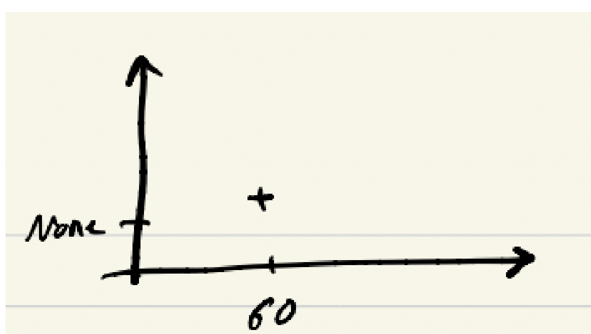
1 PAC Learning

1.1 Introduction

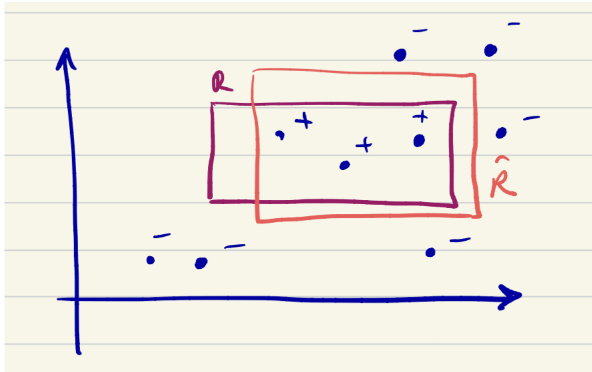
At a high level, machine learning is a way to generate functions for which we do not have easy way to write directly. Statistical element to this problem is determining how many points are needed to determine such a function. One framework is Probably approximately correct learning (PAC-learning).

1.2 Example: Jogging based on precipitation

Suppose you and your friend want to go jogging but your friend is particularly picky about the weather conditions. The weather is determined as a pair of temperature, $T \in [-20, 110]$, and precipitation, from the set $P \in \{\text{None, Mild, Heavy, Snow}\}$. Your friend either goes jogging in a particular weather or not, denoted by $+$ for yes and $-$ for not. We want to learn a rectangle in $T \times P$ that accurately predicts the conditions that your friend would want to jog.



Goal: Learn rectangular range $\hat{R}$ that approximates the true region R^*



Specifically, considering samples $p \sim D$ drawn from distribution D, we want to minimize the error:

$$err(\hat{R}) \coloneqq Pr[\hat{R}(p) \neq R^*(p)]$$

We would like to find an algorithm that gives low $err(\hat{R}) < \epsilon$ with probability $1-\delta$ for given pair (ϵ, δ)

We give this simple algorithm:

1. Given samples $(x_i, y_i) \sim D$, $i \in [m]$
2. Determine $\hat{R}$ as any consistent rectangle with the above data

Let A be the region of mismatched prediction between $\hat{R}$ and R^*. Hence,

$$\begin{aligned}
\alpha &\coloneqq err(\hat{R}) \\
&= Pr[\hat{R}(x) = + \text{ and } R^*(x) = -, \text{ or } \hat{R}(x) = - \text{ and } R^*(x) = +] \\
&= Pr[x \in A]
\end{aligned}$$

A bad outcome algorithm occurs if $\alpha \geq \epsilon$, and we want to bound the probability of this bad outcome occurring for all data points by δ. Since each point is independent and Bernoulli,

$$\begin{aligned}
Pr[\text{bad event}] &= (1-\alpha)^m \\
&\leq (1-\epsilon)^m \\
&\leq e^{-m\epsilon} \leq \delta \\
m &\geq \frac{\log 1/\delta}{\epsilon}
\end{aligned}$$

This gives us a minimum number of samples we need to achieve the desired error bounds.

1.3 Definition

Let X be *instance space.* $c : X \to \{+1, -1\}$ be a concept (hypothesis). Let C be the *concept class*, collection of such functions c. Let c^* denote the *target concept* $c^* \in C$ which labels every $x \in X$ correctly. Let D be the target distribution over X (*unlabeled*) or $X \times \{+1, -1\}$ (*labeled*).

We denote $(x_i, y_i) \sim D, i \in [m]$ the *training set.* Alternatively in the unlabeled formulation, $(x_i, c^*(x_i)) \sim D, i \in [m]$.

If c^* exists, this is known as the *realizable* case, otherwise the *agnostic* case. Denote ϵ as the *error parameter* and δ as the *confidence parameter.*

Definition 1.1. *We say a class C is PAC-learnable if there is an algorithm A such that for all D, ϵ, δ, there is an m as a function of C, ϵ, δ such that with m i.i.d. samples $(x_i, y_i) \sim D$, A has probability $1 - \delta$, A outputs $\hat{c} \in C$ such that*

$$\begin{aligned} err(\hat{c}) &= Pr[\hat{c}(x) \neq y] \\ &\leq \min_{c \in C} err(c) + \epsilon \end{aligned}$$

where $\min_{c \in C} err(c) = 0$ *in the realizable case.*

If we allow $\hat{c} \notin C$, we call this an *improper learnner*, otherwise a *proper learner*

Because classes of functions like polynomials and neural networks can universally approximate functions, $\min_{c \in C} err(c) \to 0$ for these classes.

We consider such an algorithm is efficient if $m = O(poly(1/\epsilon, 1/\delta))$