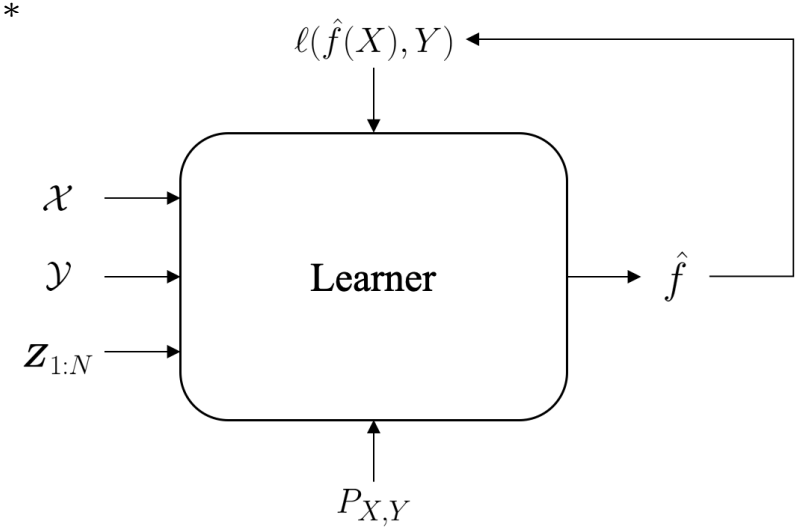
Topic 1: Learning

Learning: A Formal Introduction

- **Domain Set:** An arbitrary set $\mathcal{X}$ which we may wish to label.
- **Label Set:** Set of possible labels $\mathcal{Y} = f^*(\mathcal{X})$, where $f^* : \mathcal{X} \to \mathcal{Y}$ is the true labeling function.
- **Training Data:** A finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$, denoted as $\boldsymbol{z}_{1:N} = (z_1, \cdots, z_N) = \Big((x_1, y_1), \cdots, (x_N, y_N)\Big)$[1]
- **Learner's Output:** A prediction rule $f : \mathcal{X} \to \mathcal{Y}$ (also called a predictor, hypothesis, classifier) that predicts the labels of a new domain point from a predictor class $\mathcal{F}$. Technically, f is implemented using an algorithm $\mathcal{A}(\boldsymbol{z}_{1:N})$.
- **Data Generation Model:** A probability distribution $P_{X,Y}$ derived from a family of probability distributions $\mathcal{P}$, from which the training data $\boldsymbol{z}_{1:N}$ is generated.
- **Measure of Success:** Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ (also called risk, generalized error, true error) that quantifies how bad the prediction rule is, when compared to the true labeling function f^*

Goal: Choose the best predictor, i.e.

$$\hat{f} = \underset{f \in \mathcal{F}}{\text{minimize}}\ \mathbb{E}_{P_{X,Y}}\left[\ell(f(X), Y)\right]$$



[1] Sometimes, training data does not come with labels. In such a case, training data is $\boldsymbol{x}_{1:N} = (x_1, \cdots, x_N)$.

Example: Bias Estimation in Coin Tossing

- Goal: Given a (biased) coin with unknown probability θ of turning heads, determine θ as accurately as possible.
- Outcomes of N coin tosses: $\boldsymbol{x}_{1:N} = (x_1, \cdots, x_N)$, where $x_i = \begin{cases} 1, & \text{if Heads,} \\ 0, & \text{otherwise.} \end{cases}$
- Given θ, N and $\hat{\theta}_N(\cdot)$, we can partition $\{0,1\}^N$, the set of all N coin tosses, as

$$\text{Good Data:} \quad G_{N,\epsilon} \triangleq \left\{ \boldsymbol{x}_{1:N} \in \{0,1\}^N \;\middle|\; \left\| \hat{\theta}_N(\boldsymbol{x}_{1:N}) - \theta \right\| \leq \epsilon \right\}$$

$$\text{Bad Data:} \quad B_{N,\epsilon} \triangleq \left\{ \boldsymbol{x}_{1:N} \in \{0,1\}^N \;\middle|\; \left\| \hat{\theta}_N(\boldsymbol{x}_{1:N}) - \theta \right\| > \epsilon \right\}$$

Claim 1

For any true value θ of the coin bias, given any $\epsilon, \delta > 0$, it suffices to collect $N \geq \frac{1}{2\epsilon^2} \log\left(\frac{2}{\delta}\right)$ samples to guarantee

$$\mathbb{P}_\theta\left(G_{N,\epsilon}\right) = \mathbb{P}_\theta\left(\left\| \hat{\theta}_N(\boldsymbol{x}_{1:N}) - \theta \right\| \leq \epsilon\right) > 1 - \delta.$$

Example: Bias Estimation in Coin Tossing (cont...)

Proof of Claim 1:

Main Essence of Learning

Our main wish is to learn something about a phenomenon of interest, via observing random samples of a quantity that is relevant to the phenomenon.

Two basic questions to ask:

- **Statistical Learning:** How many samples are needed to achieve a given accuracy (ϵ) with a given confidence (δ)?
- **Computational Learning:** How efficient is the learning algorithm?

Typical learning frameworks:

- Estimation (e.g. coin tossing)
- Prediction (e.g. classification)
- Clustering
- Representation (Feature) Learning
- Density Estimation...

All the frameworks can be broadly generalized into two learning problems:

- Concept Learning (Binary Outcomes)
- Function Learning (Generalized Outcomes)

Generalization 1: Concept Learning

- *Concept class:* A class $\mathscr{C}$ of subsets of $\mathcal{X}$
- *Unknown target concept:* $C^* \in \mathscr{C}$ picked by Nature
- Binary Label: $Y_i = \mathbb{1}_{\{X_i \in C^*\}}$
- The N feature-label pairs form the training set

$$Z_1 = (X_1, Y_1) = \left(X_1, \mathbb{1}_{\{X_1 \in C^*\}}\right), \cdots, Z_N = (X_N, Y_N) = \left(X_N, \mathbb{1}_{\{X_n \in C^*\}}\right).$$

The objective is to approximate target concept C^* as accurately as possible.

Examples: Classification

Problem 1: Concept Learning

- A concept learning problem is a triple $(\mathcal{X}, \mathcal{P}, \mathscr{C})$, where $\mathcal{X}$ is the feature space, $\mathcal{P}$ is a family of probability distributions on $\mathcal{X}$, and $\mathscr{C}$ is a concept class.
- A learning algorithm for $(\mathcal{X}, \mathcal{P}, \mathscr{C})$ is a sequence $\mathcal{A} = \{A_n\}_{n=1}^{\infty}$ of mappings $A_N : (\mathcal{X} \times \{0,1\})^N \to \mathscr{C}$.

Given a training set $\boldsymbol{Z}_{1:N} = (Z_1, \cdots, Z_N) \in \mathcal{Z}^N$ and a learning algorithm $\mathcal{A}$, the approximation to C^* is

$$\hat{C}_N = A_N(\boldsymbol{Z}_{1:N}) = A_N(Z_1, \cdots, Z_N) = A_N\left(\left(X_1, \mathbb{1}_{\{X_1 \in C^*\}}\right), \cdots, \left(X_N, \mathbb{1}_{\{X_N \in C^*\}}\right)\right).$$

Generalization 1: Concept Learning (cont...)

Two types of errors: (i) $X \in C^* \cap \hat{C}_N^c$, (ii) $X \in (C^*)^c \cap \hat{C}_N$.

Combining the two, misclassification happens when X lies in the symmetric difference

$$C^* \Delta \hat{C}_N = \left(C^* \cap \hat{C}_N^c\right) \cup \left((C^*)^c \cap \hat{C}_N\right).$$

Performance measure of $\mathcal{A}$: $L(C^*, \hat{C}_N) = \mathbb{P}(C^* \Delta \hat{C}_N) = \mathbb{P}(X \in C^* \Delta \hat{C}_N)$.

Good Algorithm $\Rightarrow L(C^*, \hat{C}_N) \to \infty$ as $N \to \infty$.

- Let $X \sim P$, and $(X, \mathbb{1}_{X_N \in C}) \sim P_C$ for any $C \in \mathscr{C}$.
- Since $\hat{C}_N$ is a random element in $\mathscr{C}$, the above convergence can only be achieved in a stochastic sense.
- Define "worst case" size of set of "bad" samples as

$$\phi_{\mathcal{A}}(N, \epsilon, P) = \sup_{C \in \mathscr{C}} P_C^N \left(L\Big(C, A_N(\boldsymbol{Z}_{1:N})\Big) > \epsilon \right)$$

- Since we do not know P, consider the worst case distribution as shown below:

$$\Phi_{\mathcal{A}}(N, \epsilon, \mathcal{P}) = \sup_{P \in \mathcal{P}} \phi_{\mathcal{A}}(N, \epsilon, P).$$

Generalization 1: Concept Learning (cont...)

Definition 1: PAC for Concept Learning

A learning algorithm $\mathcal{A} = \{A_N\}$ is probably approximately correct[a] (or PAC) to accuracy ϵ if

$$\lim_{N \to \infty} \Phi_{\mathcal{A}}(N, \epsilon, \mathcal{P}) = 0.$$

- We say that $\mathcal{A}$ is PAC, if it is PAC to accuracy ϵ for every $\epsilon > 0$.
- The concept class $\mathscr{C}$ is called PAC-learnable to accuracy ϵ w.r.t. P, if there exists an algorithm that is PAC to accuracy ϵ.
- Finally, we say that $\mathscr{C}$ is PAC-learnable, if there exists an algorithm that is PAC.

[a] D. Angluin. Queries and concept learning. Machine Learning, 2:319–342, 1988.

Equivalently, a learning algorithm $\mathcal{A} = \{A_N\}$ is PAC, if for any $\epsilon > 0$ and $\delta > 0$, there exists some $N^*(\epsilon, \delta) \in \mathbb{N}$ such that, for all $N \geq N^*(\epsilon, \delta)$, $C \in \mathscr{C}$ and $P \in \mathscr{P}$, we have

$$P_C^N \left(L\Big(C, A_N(\boldsymbol{Z}_{1:N})\Big) > \epsilon \right) \leq \delta.$$

Note: $N^*(\epsilon, \delta)$ is called the ***sample complexity*** of the learning algorithm $\mathcal{A}$.

Example: Axis-Parallel Rectangles

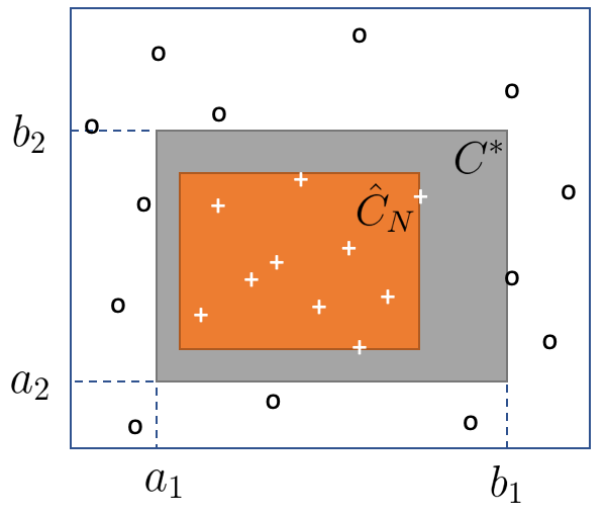
- Let $\mathcal{X} = [0,1]^2$ and $\mathscr{P}$ denote the set of all probability distributions on $\mathcal{X}$
- Let $\mathscr{C}$ denote the collection of all axis-parallel rectangles in $\mathcal{X}$, i.e., C is in $\mathscr{C}$ if it takes the form

$$C = [a_1, b_1] \times [a_2, b_2] = \left\{ (x_1, x_2) \in [0,1]^2 \;\middle|\; a_1 \leq x_1 \leq b_1,\ a_2 \leq x_2 \leq b_2 \right\},$$

for some $0 \leq a_1 \leq b_1 \leq 1$ and $0 \leq a_2 \leq b_2 \leq 1$.

Learning Algorithm:
Consider an intuitive algorithm $\mathcal{A} = \{A_N\}_{N=1}^{\infty}$ where, for each N, we have

$$\begin{aligned} \hat{C}_N &= A_N(\boldsymbol{Z}_{1:N}) \\ &= \text{smallest rectangle } C \in \mathscr{C} \text{ that contains all positive samples in } \boldsymbol{Z}_{1:N}. \end{aligned}$$



Connections to Computer Vision:

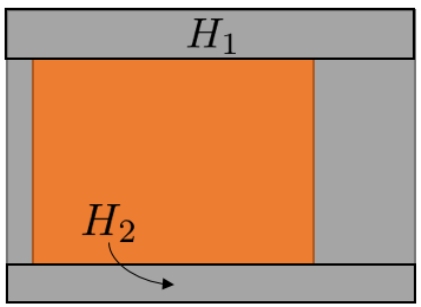
This problem is tangentially similar to estimating bounding boxes in images. The number of samples in $\mathcal{X}$ is similar to the resolution of an image.

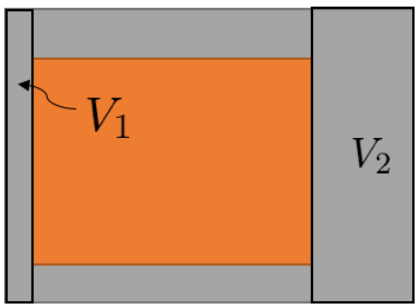
Example: Axis-Parallel Rectangles (cont...)

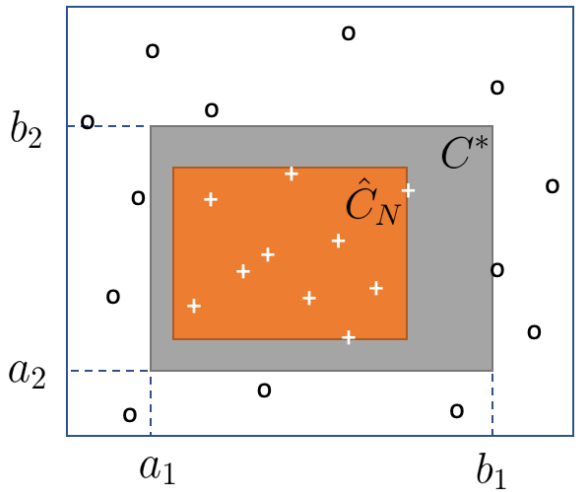
Theorem 1

The above tightest rectangle algorithm $\mathcal{A}$ is PAC, and therefore, the class $\mathscr{C}$ is PAC-learnable, since

$$\Phi_{\mathcal{A}}(N, \epsilon, \mathcal{P}) \leq 4\left(1 - \frac{\epsilon}{4}\right)^N .$$







Proof of Theorem 1:

Example: Axis-Parallel Rectangles (cont...)

Generalization 2: Function Learning

- *Function class:* A class $\mathcal{F}$ defined on $\mathcal{X}$
- *Target function:* $f^* \in \mathcal{F}$ picked by nature
- Real-valued output: $Y_i = f^*(X_i)$
- The N input-output pairs

$$Z_1 = (X_1, Y_1) = (X_1, f^*(X_1)), \cdots, Z_N = (X_N, Y_N) = (X_N, f^*(X_n)).$$

The objective is to approximate target function f^* as accurately as possible.

Examples: Estimation

Problem 2: Function Learning

- A function learning problem is a triple $(\mathcal{X}, \mathcal{P}, \mathcal{F})$, where $\mathcal{X}$ is the feature space, $\mathcal{P}$ is a family of probability distributions on $\mathcal{X}$, and $\mathcal{F}$ is a class of functions $f : \mathcal{X} \to [0, 1]$.
- A learning algorithm for $(\mathcal{X}, \mathcal{P}, \mathcal{F})$ is a sequence $\mathcal{A} = \{A_n\}_{n=1}^{\infty}$ of mappings $A_N : (\mathcal{X} \times \{0, 1\})^N \to \mathcal{F}$.

Given a training set $\boldsymbol{Z}_{1:N} = (Z_1, \cdots, Z_N) \in \mathcal{Z}^N$ and a learning algorithm $\mathcal{A}$, the approximation of f^* is

$$\hat{f}_N = A_N(\boldsymbol{Z}_{1:N}) = A_N\left(\left(X_1, \mathbb{1}_{\{X_1 \in C^*\}}\right), \cdots, \left(X_N, \mathbb{1}_{\{X_N \in C^*\}}\right)\right).$$

Generalization 2: Function Learning (cont...)

Performance of $\mathcal{A}$: $L_P(\hat{f}_N, f^*) = \mathbb{E}_P\left[\left|\hat{f}_N - f^*\right|^2\right] = \int_{\mathcal{X}} |\hat{f}_N(x) - f^*(x)|^2 P(dx)$

Remark: Concept learning is a special case of function learning.

- Define "worst case" size of set of "bad" samples as

$$\phi_{\mathcal{A}}(N, \epsilon, P) = \sup_{f \in \mathcal{F}} P_f^N \left(L\Big(A_N(\boldsymbol{Z}_{1:N}), f\Big) > \epsilon \right)$$

- Since we do not know P, consider the worst case distribution as shown below:

$$\Phi_{\mathcal{A}}(N, \epsilon, \mathcal{P}) = \sup_{P \in \mathcal{P}} \gamma_{\mathcal{A}}(N, \epsilon, P).$$

Definition 2: PAC for Function Learning

A learning algorithm $\mathcal{A} = \{A_N\}$ is probably approximately correct (or PAC) to accuracy ϵ if

$$\lim_{N \to \infty} \Phi_{\mathcal{A}}(N, \epsilon, \mathcal{P}) = 0.$$

- The function class $\mathcal{F}$ is called PAC-learnable to accuracy ϵ w.r.t. P, if there exists an algorithm that is PAC to accuracy ϵ.
- Finally, $\mathcal{F}$ is called PAC-learnable, if there exists an algorithm that is PAC.

Limitations of Model-Based Approaches

- We assume $C^* \in \mathscr{C}$ (or equivalently, $f^* \in \mathcal{F}$) $\Rightarrow$ Fit data regarding a well-studied phenomenon to some ***a priori*** **known hypothesis class**
- Labels $y = \mathbf{1}_{x \in C^*}$ (or equivalently, $y = f^*(x)$) are assumed to be **noiseless**.

Such limitations will lead us naturally towards a new framework called ***model-agnostic learning*** (also called model-free learning).

The main goal is to find the best possible hypothesis (concept/function) within a chosen hypothesis class $\mathcal{F}$.

Problem 3: Model-Agnostic Learning

A model-agnostic learning problem is a tuple $(\mathcal{X}, \mathcal{Y}, \mathcal{P}, \mathcal{F})$, where

- Sets: $\mathcal{X}$ (input feature space), $\mathcal{Y}$ (label space) and $\mathcal{U}$[a] (hypothesis space)
- A class $\mathcal{P}$ of probability distributions on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.
- A class $\mathcal{F}$ of functions $f : \mathcal{X} \to \mathcal{U}$.

A learning algorithm for $(\mathcal{X}, \mathcal{Y}, \mathcal{U}, \mathcal{P}, \mathcal{F}, \ell)$ is a sequence of mappings $\mathcal{A} = \{A_N\}_{N=1}^{\infty}$, where $A_N : \mathcal{Z}^N \to \mathcal{F}$.

[a] $\mathcal{U} \neq \mathcal{Y}$, since the true-hypothesis labels are corrupted by noise.

PAC Learnability for Model-Agnostic Learning

- Given a learning algorithm $\mathcal{A} = \{A_N\}_{N=1}^{\infty}$ with $A_N : \mathcal{Z}^N \to \mathcal{F}$, if $\hat{f}_N = A_N(\boldsymbol{Z}_{1:N})$, then the performance can be measured as

$$L_P(\hat{f}_N) = \mathbb{E}_P\left[\ell(Y, \hat{f}_N(X))\right] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, \hat{f}_(x))P(dx, dy)$$

- **Minimum risk:** $L_P^*(\mathcal{F}) = \inf_{f\in\mathcal{F}} L_P(f)$ for an induced function class $\mathcal{F}$.

  i.e., given any algorithm $\mathcal{A}$, we have $0 \leq L_P^*(\mathcal{F}) \leq L_P(\hat{f}_N) \leq 1$.
- Given any $\epsilon > 0$, let the worst case probability of getting a bad sample be

$$\Phi_{\mathcal{A}}(N, \epsilon) = \sup_{P\in\mathcal{P}} P^N\left(L_P(\hat{f}_N) > L_P^*(\mathcal{F}) + \epsilon\right)$$

Definition 3: PAC for Model-Agnostic Learning

A learning algorithm $\mathcal{A} = \{A_N\}$ for a problem $(\mathcal{X}, \mathcal{Y}, \mathcal{U}, \mathcal{P}, \mathcal{F}, \ell)$ is PAC to accuracy ϵ if

$$\lim_{N\to\infty} \Phi_{\mathcal{A}}(N, \epsilon) = 0.$$

- An algorithm that is PAC to accuracy ϵ for every $\epsilon > 0$ is said to be PAC.
- Finally, a learning problem $(\mathcal{X}, \mathcal{Y}, \mathcal{U}, \mathcal{P}, \mathcal{F}, \ell)$ is *model-agnostically learnable* if there exists an algorithm for it, which is PAC.

Empirical Risk Minimization and McDiarmid's Inequality

But, we do not always know the input distribution $P \in \mathcal{P}$.

- $L_P(f)$ is unknown. Can we replace this with some surrogate?
- **ERM Algorithm:** $\hat{f}_N = \arg\min_{f \in \mathcal{F}} L_N(f) \triangleq \frac{1}{N} \sum_{i=1}^{N} \ell\big(Y_i, f(X_i)\big)$

Definition 4: Bounded Differences Property

A function $f : X_1 \times \cdots \times X_N \to \mathbb{R}$ satisfies $(c_1, ..., c_N)$-bounded differences property, if for every $i = 1, \cdots, N$ and every $(x_1, \cdots, x_N), (x'_1, ..., x'_N) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$ that differ only in the i-th coordinate (i.e, $x_j = x'_j$, for all $j \neq i$), we have $\left| f\left(x_1, ..., x_N\right) - f\left(x'_1, ..., x'_N\right) \right| \leq c_i$.

Theorem 2: McDiarmid's Inequality

Let $X_1, \cdots, X_N$ be independent random variables, where $X_i \in \mathcal{X}_i$. Let $f : X_1 \times \cdots \times X_N \to \mathbb{R}$ be any function with the $(c_1, ..., c_N)$-bounded differences property. Then, for any $t > 0$, we have

$$\mathbb{P}\left[f\left(x_1, ..., x_N\right) - \mathbb{E}\Big(f\left(x_1, ..., x_N\right) \Big) \geq t \right] \leq \exp\left[-2t \left(\sum_{i=1}^{N} c_i^2 \right)^{-1} \right].$$

McDiarmid's Inequality for ERM

Corollary 1: McDiarmid's Inequality for ERM

For any $\ell : \mathcal{Z} \to [0, 1]$, we have

$$\mathbb{P}\left(\left|L_N(f) - L_P(f)\right| < \epsilon\right) \geq 1 - 2e^{-2N\epsilon^2}.$$

Proof:

Mismatched Minimization Lemma

Suppose we wish to minimize a function G defined on some domain U, but has access only to its approximation $\hat{G}$, then...

Lemma 1: Mismatched Minimization Lemma

Suppose that $\hat{G}$ is an ϵ-uniform approximation of G for some $\epsilon > 0$ (i.e. $|G(u) - \hat{G}(u)| \leq \epsilon$ for all $u \in U$), then

- For any $u' \in U$, we have $G(u') \leq \hat{G}(u') + \epsilon$.
- Suppose u^* is a minimizer of $\hat{G}$ (i.e. $\hat{G}(u^*) \leq \hat{G}(u)$ for all $u \in U$), then $G(u^*) \leq \inf_{u \in U} G(u) + 2\epsilon$.

Proof: