

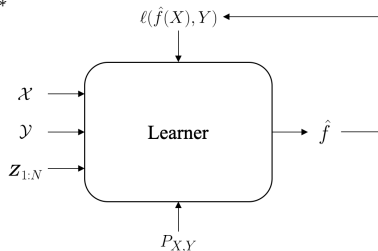
Topic 1: Learning

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 $f^* : \mathcal{X} \rightarrow \mathcal{Y}$ is the true labeling function.
- ▶ **Training Data:** A finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$, denoted as $\mathbf{z}_{1:N} = (z_1, \dots, z_N) = \left((x_1, y_1), \dots, (x_N, y_N) \right)^1$
- ▶ **Learner's Output:** A prediction rule $f : \mathcal{X} \rightarrow \mathcal{Y}$ (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 $\mathcal{A}(\mathbf{z}_{1:N})$.
- ▶ **Data Generation Model:** A probability distribution $P_{X,Y}$ derived from a family of probability distributions \mathcal{P} , from which the training data $\mathbf{z}_{1:N}$ is generated.
- ▶ **Measure of Success:** Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$ (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.

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



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

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: $\mathbf{x}_{1:N} = (x_1, \dots, x_N)$, 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

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

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

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

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

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

$$\mathbf{Z}_1 = (X_1, Y_1) = \left(X_1, \mathbb{1}_{\{X_1 \in C^*\}}\right), \dots, \mathbf{Z}_N = (X_N, Y_N) = \left(X_N, \mathbb{1}_{\{X_N \in C^*\}}\right).$$

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

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

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

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

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

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 $\Rightarrow L(C^*, \hat{C}_N) \rightarrow 0$ as $N \rightarrow \infty$.

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

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

- ▶ 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

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

- ▶ We say that \mathcal{A} is PAC, if it is PAC to accuracy ϵ for every $\epsilon > 0$.
- ▶ The concept class \mathcal{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 \mathcal{C} is PAC-learnable, if there exists an algorithm that is PAC.

^aD. 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 $N \geq N^*(\epsilon, \delta)$, $C \in \mathcal{C}$ and $P \in \mathcal{P}$, we have

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

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 \mathcal{P} denote the set of all probability distributions on \mathcal{X}
- ▶ Let \mathcal{C} denote the collection of all axis-parallel rectangles in \mathcal{X} , i.e., C is in \mathcal{C} if it takes the form

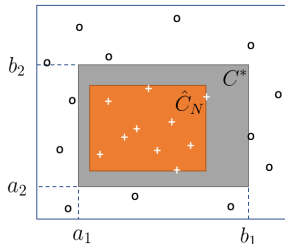
$$C = [a_1, b_1] \times [a_2, b_2] = \left\{ (x_1, x_2) \in [0, 1]^2 \mid a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2 \right\},$$

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

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



Connections to Computer Vision:

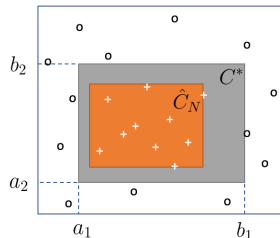
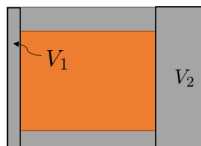
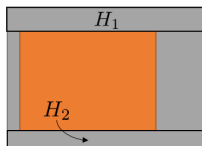
This problem is tangentially similar to estimating bounding boxes in images. The number of samples in \mathcal{X} is similar to the resolution of an image.

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

Theorem 1

The above tightest rectangle algorithm \mathcal{A} is PAC, and therefore, the class \mathcal{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

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

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 $f : \mathcal{X} \rightarrow [0, 1]$.
- ▶ A learning algorithm for $(\mathcal{X}, \mathcal{P}, \mathcal{F})$ is a sequence $\mathcal{A} = \{A_n\}_{n=1}^\infty$ of mappings $A_N : (\mathcal{X} \times \{0, 1\})^N \rightarrow \mathcal{F}$.

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

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

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

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

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

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

Definition 2: PAC for Function Learning

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

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

- 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.