Introduction to Machine Learning

Alessandro Rudi, Umut Simsekli
Notes by Antoine Groudiev

8th February 2024

Contents

Introduction

1 An overview of Machine Learning

1.1 What is ML?

Considering a problem, such as image classification: given an input image of a dog or a cat, the program is asked to determine whether the image is a dog or a cat. Conventional programming would hardcode the solution to this problem. But this process takes time and is not easily generalisable. Instead, an ML model is trained on a dataset to produce a program to solve the problem.

Many successfull applications of Machine Learning are:

- Face recognition
- Spam filtering
- Speech recognition
- Self-driving systems; pedestrian detection

1.2 Topics in Machine Learning

1.2.1 Supervised Learning

Example (Classification)**.** *Features* $x \in \mathbb{R}^d$*, labels* $y \in \{1, \ldots, k\}$

Definition (Regression)**.** Features $x \in \mathbb{R}^d$, labels $y \in \mathbb{R}$. To tackle such problem, we look for a parametrized function $f_\theta(x_i) \simeq y_i$ for some f_θ in a function space

$$\mathcal{F} = \{f_\theta : \theta \in \Theta\}$$

Our goal is therefore to find the best function in $\mathcal{F}$ such that f "fits" the training data. For example, we can say that f "fits" the training data when

$$\frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

is "small". Such a function is not interesting in general, like for classification.

Definition (Loss function)**.** Assums that the features are in $\mathcal{X}$ and the labels are in $\mathcal{Y}$. We introduce the more general *loss function* notion:

$$l : \mathcal{Y}^2 \to \mathbb{R}_+$$

For a regression task, we can use $l(\hat{y}, y) = (\hat{y} - y)^2$. For a classification task, $l(\hat{y}, y) = \mathbb{1}_{\hat{y}=y}$.

Therefore, for a regression problem, we might choose:

$$f^\star = \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} l(f(x_i), y_i)$$

In the parametric case, when $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$, we might minimize with respect to θ:

$$\theta^\star = \operatorname*{argmin}_{\theta \in \Theta} \frac{1}{n}\sum_{i=1}^{n} l(f(x_i), y_i)$$

1.2.2 Probabilistic approach

Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ be the feature space. Let D be a distribution on $\mathcal{Z}$; we make the assumption that the training data is iid from D:

$$(x_i, y_i) \sim D$$

and the same thing hold for the test data:

$$(\tilde{x_i}, \tilde{y_i}) \sim D$$

According to the Strong Law of Large Numbers, the test loss converges almost surely:

$$\lim_{n \to \infty} \frac{1}{n}\sum_{i=1}^{n} l(f_\theta(\tilde{x_i}), \tilde{y_i}) = \mathbb{E}_{(x,y)\sim D}[l(f_\theta(x), y)] =: R(\theta) = R(f_\theta)$$

where $R(\theta)$ is the *population risk.*

Definition (Risk minimization)**.**

1.2.3 Unsupervised Learning

Example (Clustering)**.**

Example (Dimensionnality reduction)**.** *We are given features $x \in \mathbb{R}^d$ and labels $y \in \{0, 1\}$ which form a "training" dataset $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. We assume that $d >> 1$; our goal is to find $d' << d$ such that $(x_1, y_1, \ldots,)$*