

# Introduction to Machine Learning (67577)

## Recitation 7 Classification

Second Semester, 2021

### Contents

<b>1</b>	<b>Probabilistic Approach for Estimation and Classification</b>	<b>2</b>
1.1	The Basics of Generative Models . . . . .	2
1.2	Maximum Likelihood . . . . .	4
<b>2</b>	<b>Logistic regression</b>	<b>5</b>
2.1	The Model . . . . .	5
2.2	Making Predictions . . . . .	5
<b>3</b>	<b>Classification Trees</b>	<b>6</b>
3.1	CART Heuristic . . . . .	7
3.2	Time Complexity . . . . .	7

## 1 Probabilistic Approach for Estimation and Classification

### 1.1 The Basics of Generative Models

Whenever we begin working on a dataset, we first try to understand how does the data "behave". We do that so as to come up with appropriate ways to model the phenomenon we are investigating/learning. We think about what assumptions (hopefully) hold for it, and by that what learning algorithms can be used.

Up until now, we thought of the data in the following manner: We somehow obtained samples from a domain set  $X \subset \mathcal{X}$  and labels/responses  $Y \subset \mathcal{Y}$ . We defined the connection between them as  $Y = f(X)$  for some function  $f$ , and sometimes also added some expression of noise  $\varepsilon$  with different statistical properties (for example Gaussian noise in a linear regression model).


Now we will make a different assumption. Suppose that the data comes from an underlying distribution  $\mathcal{D}$  which is a joint distribution over both the domain space  $\mathcal{X}$  and the responses  $\mathcal{Y}$ . Let us assume that when we obtain a labeled dataset the items are drawn independently and identically distributed (iid) over  $\mathcal{D}$ . Namely:

$$(x_1, y_1), \dots, (x_m, y_m) \stackrel{\text{iid}}{\sim} \mathcal{D} \text{ over } \mathcal{X} \times \mathcal{Y}$$

- One way to think of this resembles how we approached the data in the previous lessons. Suppose we have some data  $x_1, \dots, x_m$ . We consider it as if constant. Then we analyse it to obtain the responses/labels. That is, by analysing  $x_i$  (with perhaps assuming additional noise) we try to learn the conditional distribution  $y_i|x_i$ .
- A different approach is driven by the assumption of an underlying joint distribution over  $x_i, y_i$  (that is:  $\mathcal{D}$  over  $\mathcal{X} \times \mathcal{Y}$ ). To understand it consider the following process to generate the data: Let there be two points  $\mu_0, \mu_1 \in \mathbb{R}^d$  each representing a different label: 0 or 1 respectively. Then, for any  $i \in [n]$  we flip a coin with probability  $p$  for label 1. Let  $y_i$  be zero or one depending on the coin's outcome. Then, given  $y_i$  we sample  $x_i$  from a Gaussian of the corresponding class. Namely:

$$\begin{aligned} \forall i \in [n] \quad & y_i \stackrel{\text{ind.}}{\sim} \text{Ber}(p) \\ & x_i|y_i \sim \mathcal{N}(\mu(y_i), \Sigma(y_i)) \end{aligned}$$

where  $\Sigma_0, \Sigma_1$  are some covariance matrices of the multivariate Gaussians.

 For a bit more explanations (not too in depth and technical), check this Cross-Validation question: [Generative vs. discriminative](#)

■ **Example 1.1** Let  $\mu_0 = (0, 5)^\top$ ,  $\mu_1 = (5, 5)^\top$  and let  $p = 0.4$ . Suppose we draw 1000 points where  $\forall i \in [n] \quad y_i \sim \text{Ber}(p)$  and that we got 400 points assigned to class 1.

Then sampling the points in  $\mathbb{R}^2$  from the respective Gaussians yielded the following graph:

$$x_i|y_i = 0 \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}\right), \quad x_i|y_i = 1 \sim \mathcal{N}\left(\begin{bmatrix} 5 \\ 5 \end{bmatrix}, \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}\right)$$



■

By assuming there exists a joint distribution  $\mathcal{D}$  over  $\mathcal{X} \times \mathcal{Y}$  we gain a lot. Now, to evaluate our model we are no longer restricted to some specific test set that we are given, but instead we can draw our own test sets and ask what is the expectation of the risk:

$$L_{\mathcal{D}}(h) = \mathbb{E}_{(x,y) \sim \mathcal{D}} [h(x) \neq y]$$

So, suppose we **knew**  $\mathcal{D}$ , then solving a classification problem according to the ERM principle would simply be done by assigning the label with the highest conditional probability. That is:

**Definition 1.1** Given any probability distribution  $\mathcal{D}$  over  $\mathcal{X} \times \mathcal{Y}$ , the **Bayes Optimal Classifier**  $f_{\mathcal{D}}$  is defined by:

$$f_{\mathcal{D}} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \mathbb{P}_{\mathcal{D}}(Y = y | X = x)$$

**Claim 1.1** Let  $h : \mathcal{X} \rightarrow \mathcal{Y}$  be a hypothesis, then  $L_{\mathcal{D}}(f_{\mathcal{D}}) \leq L_{\mathcal{D}}(h)$



The Bayes Optimal classifier is a kind of an **Oracle classifier**. This means that we will never be able to actually have it but it helps us define bounds on how good we could do. In addition, as you will see in the exercise, by assuming further different assumptions we can estimate some parts of the distribution from our data, and approximate the Bayes Optimal Classifier under those assumptions.

## 1.2 Maximum Likelihood

In reality we do not know  $\mathcal{D}$ , and only have a window to it through our dataset. Therefore, we need to devise methods that will come as close as we can to  $f_{\mathcal{D}}$  without knowing it completely. One of the critical choices we need to make is that of a hypothesis class (a.k.a model). This is where our prior knowledge about the problem comes in. Intuitively, if we have some educated guess about properties of  $\mathcal{D}$ , we can try to embody them in our choice of  $\mathcal{H}$ .

Once we have decided on a hypothesis class, of hypotheses derived from  $\mathcal{D}$ , perhaps we could use our training set to estimate which hypothesis is the **most likely**.

**Definition 1.2** Let  $\mathcal{F}(\Theta)$  be some distribution function with parameters  $\Theta$ , and  $x_1, \dots, x_m \stackrel{iid}{\sim} \mathcal{F}(\Theta)$ . We define the **likelihood** as

$$L(\Theta) = L(\Theta|x_1, \dots, x_m) := \mathbb{P}_{\mathcal{F}}(x_1, \dots, x_m|\Theta)$$

■ **Example 1.2** Let  $x_1 = 1, x_2 = 2, x_3 = 3$  be 3 samples drawn from a Normal distribution with an unknown expectation  $\mu$  and variance  $\sigma^2 = 1$ . Calculate the likelihood of the following:

- $\mu = 1$ :

$$\begin{aligned} L(\mu = 1|X_1, X_2, X_3, \sigma^2) &= \prod_i \mathbb{P}(X_i = x_i|\mu = 1, \sigma^2 = 1) = \prod_i \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \\ &= \prod_i \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_i - 1)^2}{2}\right) = \frac{1}{(\sqrt{2\pi})^3} \exp\left(-\frac{1}{2} \sum (x_i - 1)^2\right) \\ &= \frac{1}{(\sqrt{2\pi})^3} \exp\left(-\frac{5}{2}\right) \approx 0.005211 \end{aligned}$$

- $\mu = 2$ :

$$\begin{aligned} L(\mu = 2|X_1, X_2, X_3, \sigma^2) &= \prod_i \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_i - 2)^2}{2}\right) = \frac{1}{(\sqrt{2\pi})^3} \exp\left(-\frac{1}{2} \sum (x_i - 2)^2\right) \\ &= \frac{1}{(\sqrt{2\pi})^3} \exp\left(-\frac{2}{2}\right) \approx 0.023358 \end{aligned}$$

So for the given data it is **more likely** that it is drawn from  $(2, 1)$ . ■

**Definition 1.3 — Maximum Likelihood Estimator.** Let  $X = x_1, \dots, x_m \stackrel{iid}{\sim} \mathcal{F}(\Theta)$  be a sample set, then the **maximum likelihood estimator** is defined as:

$$\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}} L(\theta|x_1, \dots, x_m)$$

Intuitively, the MLE retrieves the parameter  $\theta$  that maximizes the likelihood of the data. In other words, the parameter that given the data, is most likely to be the parameter used to draw data from  $\mathcal{F}$ .

■ **Example 1.3** Going back to our first example, suppose we obtained our dataset of  $S = \{(x_i, y_i)\}_{i=1}^m$ . Let us try and estimate the different parameters of the model:  $p$ , the mean vectors and the covariance matrices.

- Applying the Maximum Likelihood principle we begin with estimating the coin probability  $\hat{p}$ :

$$\begin{aligned}
 \hat{p}_{MLE} &= \operatorname{argmax}_{p'} L(p' | y_1, \dots, y_n) \\
 &= \operatorname{argmax}_{p'} \mathbb{P}(y_1, \dots, y_n | p') \\
 &\stackrel{iid}{=} \operatorname{argmax}_{p'} \prod_{i: y_i=1} \mathbb{P}(y_i = 1 | p') \cdot \prod_{i: y_i=0} \mathbb{P}(y_i = 0 | p') \\
 &= \operatorname{argmax}_{p'} p^{n_1} \cdot (1 - p)^{n_0}
 \end{aligned}$$

where  $n_0$  the number of samples with label 0 and  $n_1$  the number of samples with label 1. By taking the derivative of the above and equating to zero we get that:

$$\hat{p}_{MLE} = \frac{1}{n} \sum_{i=1}^n \mathbb{1}[y_i = 1]$$

- Try to think how to estimate the  $\widehat{\mu}_0, \widehat{\mu}_1, \widehat{\Sigma}_0, \widehat{\Sigma}_1$  parameters. You can either think about a MLE approach or use estimators you have already seen in our first lecture!

■

## 2 Logistic regression

### 2.1 The Model

Recall the setup of a linear regression problem. Let  $X \subseteq \mathbb{R}^{d \times n}$  and  $Y \in \mathbb{R}^n$ . Now suppose that instead of having a continuous response vector we are actually facing a classification problem where (without loss of generality)  $Y \in \{0, 1\}$ . In the case of logistic regression, rather than modeling  $Y$  directly, we model the *probability* that  $Y$  belongs to a particular category. That is, we are interested in modeling  $p(X) = \mathbb{P}(Y = 1 | X)$ .

So instead of modeling  $Y = X^\top w$ , we want to find a function that outputs values in  $[0, 1]$ . In logistic regression we use the *logistic function*:

$$p(X) = \frac{e^{X^\top w}}{1 + e^{X^\top w}}$$

### 2.2 Making Predictions

Suppose we knew what were our coefficients  $\hat{w}$ . Then, by plugging them in we could easily obtain the probability of that sample being classified 1:

$$\mathbb{P}(y_i = 1 | x_i) = p(x_i) = \frac{e^{x_i^\top \hat{w}}}{1 + e^{x_i^\top \hat{w}}}$$

Next, for some cut-off threshold  $t$  we can obtain a classification rule as follows:

$$y_i = \mathbb{1}[p(x_i) \geq t] \quad t \in [0, 1]$$

Try and think: how does changing  $t$  influence our predictions? What kinds of misclassification errors will we have?

### 3 Classification Trees

Classification trees is another class of classification algorithms seen in class. In short we described the:

- Hypothesis class  $\mathcal{H}_{CT}^k$  as the piece-wise-constant functions induced by axis-aligned rectangles, of depth at most  $k$ .
- Loss function as counting misclassification errors (accuracy): Let  $B_1, \dots, B_N$  be a partitioning of  $\mathbb{R}^d$ :  $\cup B_i = \mathbb{R}^d$ . Then:

$$P_y^S(B) = \frac{1}{n_S(B)} \sum_{x_i \in B} \mathbb{1}[y_i = y]$$

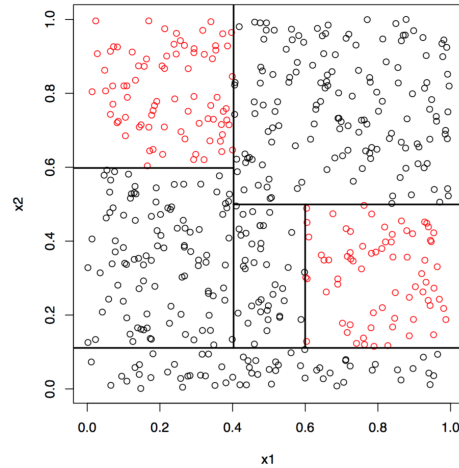
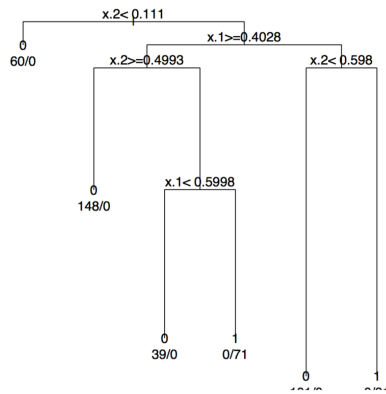
and over the entire space:

$$L(B_1, \dots, B_N) = \sum_{j=1}^N P_1^S(B_j)$$

- Label that minimizes the empirical risk (maximizes the success) given some box  $B$  as:

$$\hat{y}_S(B) = \operatorname{argmax}_{y \in \{0,1\}} P_y^S(B)$$

Then, using the ERM principle, we wanted to find the tree that minimizes the loss over our training data  $S$ . We reasoned the introduction of the hyper-parameter  $k$  to control the depth of the tree and noticed that by doing so each  $\mathcal{H}_{CT}^k$  is in fact a family of hypothesis classes, each consisting of all the trees of depth  $k$  over our features and possible partitions of  $\mathbb{R}^d$ .



As such, for some selection of  $k$ , solving for ERM is to find:

$$h^* = \operatorname{argmin}_{h \in \mathcal{H}_{CT}^k} L_S(h)$$

but this introduced a new problem: the number of possible trees is exponential in the number of features and maximal depth. In fact, it has been proven to be an NP-Complete problem. That is, there is no deterministic polynomial algorithm that can solve this optimization problem.

### 3.1 CART Heuristic

This called the need to define some heuristics: partial methods that do not guarantee finding the optimal solution, but nevertheless achieve a sufficiently good solution. The **Classification and Regression Trees** (CART) solves the problem (approximately) in two steps: **growing** a tree and then **pruning** it.

Let  $S = \{(x_i, y_i)\}_{i=1}^n$  be our training data where  $x_i \in \mathbb{R}^d$  and  $y_i \in \{0, 1\}$ . Let  $k$  be the maximal tree depth and  $n_{\min} \geq 1$  a minimal number of samples in some subset of  $\mathbb{R}^d$ .

Now, the CART growing step is as follows:

- Initialize  $B^{(1)} = \mathbb{R}^d$
- Over some subset  $B^{(j)}$ , for every  $i \in [d]$  define  $g_i(t)$  as follows:
  - Denote  $B_{+,t}^{(j)} = \{x | x_i > t\}$  and  $B_{-,t}^{(j)} = \{x | x_i \leq t\}$
  - Let  $\hat{y}_S(B_{\pm,t})$  be the class assignment minimizing the empirical mis-classification, and  $P_{\hat{y}_S(B_{\pm,t})}^S$  be the optimal empirical mis-classification risk.
  - Denote  $g_i(t) = P_{\hat{y}_S(B_{-,t}^{(j)})}^S(B_{-,t}^{(j)}) + P_{\hat{y}_S(B_{+,t}^{(j)})}^S(B_{+,t}^{(j)})$
- Define  $t_i$  the **best** value to split  $B^{(j)}$  by to be:  $t_i = \underset{t \in \mathbb{R}}{\operatorname{argmax}} g_i(t)$ . Then let  $i_* = \underset{t \in [d]}{\operatorname{argmax}} g_i(t_i)$  be the **best** coordinate to split  $B^{(j)}$  by.
- Split  $B^{(j)}$  as described into  $B^{(2j)}$  and  $B^{(2j+1)}$  and repeat the procedure over each part until reached maximal depth or that the number of items in the partition is  $\leq n_{\min}$ .

### 3.2 Time Complexity

It is left to convince ourselves that this algorithm is indeed polynomial in the input:

- For every split we perform we scan  $d$  features and over our  $n$  samples find the best  $t \in \mathbb{R}$ . Notice that we do not need to actually search over all  $\mathbb{R}$  but only over our  $\leq n$  values of each given feature. As such, each split has a time complexity of  $\mathcal{O}(dn)$ .
- Then, to grow a tree of depth  $k$  we need  $\mathcal{O}(dn \cdot 2^k)$  steps. **However**, as we do not allow empty partitions  $2^k$  is bounded above by having  $n$  partitions (one for every data point). Therefore our time complexity is  $\mathcal{O}(dn^2)$ .