{EPITECH.}
PARIS GRADUATE SCHOOL OF DIGITAL INNOVATION

Vizualisation

Part 4. Reliability

B9 - Visualisation of Massive Data

M-ALG-103

Reliability

Dissimiliraty

Convergence and the law of large numbers

Dissimilarity

- Let us consider a **supervised learning** setup
- In order to evaluate the quality of our model, we need to mesure the **discrepancy** between predictions and observations, noted $d(\tilde{f}(x), y)$ for one given prediction $\tilde{f}(x)$ and one given observation y.
- And we want to minimize the aggregated discrepancy, summed over all the training samples :

$$\sum_{i=1}^{n} d(\tilde{f}(x_i), y_i) \tag{1}$$

Dissimilarity

- Let us consider a **supervised learning** setup
- In order to evaluate the quality of our model, we need to mesure the **discrepancy** between prediction and observation, noted $d(\tilde{f}(x), y)$.
- And we want to minimize the aggregated discrepancy, summed over all the training samples :

$$\sum_{i=1}^{n} d(\tilde{f}(x_i), y_i) \tag{2}$$

- But what is $d(\tilde{f}(x), y)$?

Examples of distances

$x = (x_1, ..., x_p)$ and $y = (y_1, ..., y_p)$ are p-dimensional **vectors**.

Examples of distances

$x = (x_1, ..., x_p)$ and $y = (y_1, ..., y_p)$ are p-dimensional **vectors**.

- $L_2 : ||x - y||_2 = \sqrt{\sum_{k=1}^{p}(x_k - y_k)^2}$ (Euclidian distance, 2-norm distance)

Examples of distances

$x = (x_1, ..., x_p)$ and $y = (y_1, ..., y_p)$ are p-dimensional **vectors**.

- $L_2 : ||x - y||_2 = \sqrt{\sum_{k=1}^{p} (x_k - y_k)^2}$ (Euclidian distance, 2-norm distance)
- $L_1 : ||x - y||_1 = \sum_{k=1}^{p} |x_k - y_k|$ (Manhattan distance, 1-norm distance)

Examples of distances

$x = (x_1, ..., x_p)$ and $y = (y_1, ..., y_p)$ are p-dimensional **vectors**.

- $L_2 : ||x - y||_2 = \sqrt{\sum_{k=1}^{p} (x_k - y_k)^2}$ (Euclidian distance, 2-norm distance)
- $L_1 : ||x - y||_1 = \sum_{k=1}^{p} |x_k - y_k|$ (Manhattan distance, 1-norm distance)
- weighted $L_1 : \sum_{k=1}^{p} w_k |x_k - y_k|$

Examples of distances

$x = (x_1, ..., x_p)$ and $y = (y_1, ..., y_p)$ are p-dimensional **vectors**.

- L2 : $||x - y||_2 = \sqrt{\sum_{k=1}^{p}(x_k - y_k)^2}$ (Euclidian distance, 2-norm distance)
- L1 : $||x - y||_1 = \sum_{k=1}^{p} |x_k - y_k|$ (Manhattan distance, 1-norm distance)
- weighted L_1 : $\sum_{k=1}^{p} w_k|x_k - y_k|$
- L_∞ : $\max(x_1, \ldots, x_n)$ (infinity norm distance)

Hamming distance

- $\#\{x_i \neq y_i\}$ (Hamming distance)

Hamming distance and edit distance

- $\#\{x_i \neq y_i\}$ (Hamming distance)
- linked to **edit distance** : used to quantify how dissimilar two strings are by counting the number of operations needed to transform one into the other (several variants exist)

General definition of a distance

A **distance** on a set E is an application $d : E \times E \to \mathbb{R}_+$ that must :

General definition of a distance

A **distance** on a set E is an application $d : E \times E \to \mathbb{R}_+$ that must :

- be **symetric** : $\forall x, y, d(x, y) = d(y, x)$

General definition of a distance

A **distance** on a set E is an application $d : E \times E \to \mathbb{R}_+$ that must :

- be **symetric** : $\forall x, y, d(x, y) = d(y, x)$
- **separate the values** : $\forall x, y, d(x, y) = 0 \Leftrightarrow x = y$

General definition of a distance

A **distance** on a set E is an application $d : E \times E \rightarrow \mathbb{R}_+$ that must :

- be **symetric** : $\forall x, y, d(x, y) = d(y, x)$
- **separate the values** : $\forall x, y, d(x, y) = 0 \Leftrightarrow x = y$
- respect the **triangular inequality**
  $\forall x, y, z, d(x, y) \leq d(x, z) + d(y, z)$

General definition of a distance

We could verify that :

- L2 is a distance
- Hamming is a distance

Examples of usage of L2 in Machine Learning:

- kmeans
- k-nearest neighbors

Examples when L2 is used

When is L2 used in machine learning ?

- kmeans
- k-nearest neighbors
- agglomerative clustering

k-nearest neighbors

In a **supervised learning** context :

- Given training samples that are **labeled**
- in a **classification** context : for a new input x, find the k closest neighbors of x, and choose a class for x with the majorities of the classes of the training samples
- in an **regression** context : for a new input x, find the k closest neighbors of x, and return the average of the outputs of the training samples
- to find the nearest neighbors, we can use the L2 distance.

k-nearest neighbors

- Manhattan (L1) can also be used for kNN.

Back to L2

- The euclidean distance can also be used to compute **cost functions** in neural netwoks
- $$C = \frac{(y - \tilde{f}(x))^2}{2} \tag{3}$$

Examples when Hamming is used

- compare strings
- actually we can also use it for k-nearest neighbors

Cross entropy

- The **cross-entropy** is another cost function that can be used in neural networks doing classification.
- A neural network is just a kind of function, here we can just think of it as a function $\tilde{f}(x)$ of an input x.

Cross entropy

- The **cross-entropy** is another cost function that can be used in neural networks doing classification.
- A neural network is just a kind of function, here we can just think of it as a function $\tilde{f}(x) \in \mathbb{R}$ of an input x.
- Say we have two possible classes : $y = 0$ or $y = 1$, and a large number of inputs x_i, each labeled with a class y_i.
- Instead of using the **quadratic cost** $C = \frac{(y-\tilde{f}(x))^2}{2}$, we can use the **cross entropy**.

Cross entropy

- Say we have two possible classes : $y = 0$ or $y = 1$, and a large number of inputs x_i, each labeled with a class $y_i \in \{0, 1\}$. $x = (x_1, ...x_n)$, $y = (y_1, ...y_n)$
- **quadratic cost**

$$C(\tilde{f}(x), y) = \sum_{i=0}^{n} \frac{(y_i - \tilde{f}(x_i))^2}{2} \tag{4}$$

- **cross entropy**

$$C'(\tilde{f}(x), y) = -\frac{1}{n} \sum_{i=0}^{n} [y_i \log \tilde{f}(x_i) + (1 - y_i) \log(1 - \tilde{f}(x_i))] \tag{5}$$

Cross entropy

- Is it a cost ?

$$C'(\tilde{f}(x), y) = -\frac{1}{n}\sum_{i=0}^{n}[y_i \log \tilde{f}(x_i) + (1-y_i)\log(1-\tilde{f}(x_i))] \quad (6)$$

Cross entropy

- Is it a cost ? $x_i \in \mathbb{R}^p$, $\tilde{f}(x_i) \in [0,1]$ and $y_i \in \{0,1\}$

$$C'(\tilde{f}(x), y) = -\frac{1}{n}\sum_{i=0}^{n}[y_i \log \tilde{f}(x_i) + (1-y_i)\log(1-\tilde{f}(x_i))] \quad (7)$$

-
  - **positivity** : $C'(\tilde{f}(x), y) \geq 0$

Cross entropy

- Is it a cost ? $x_i \in \mathbb{R}^n$, $\tilde{f}(x) \in [0,1]$ and $y_i \in \{0,1\}$

$$C'(x,y) = -\frac{1}{n}\sum_{i=0}^{n}[y_i \log \tilde{f}(x_i) + (1-y_i)\log(1-\tilde{f}(x_i))] \quad (8)$$

-
  - **positivity** : $C'(\tilde{f}(x), y) \geq 0$
  - **symmetry** : $C'(\tilde{f}(x), y) = C'(y, \tilde{f}(x))$
  - **separation** : $C'(\tilde{f}(x), y) = 0 \Leftrightarrow \tilde{f}(x) = y$

Cross entropy

- The biggest asset of cross entropy is that it prevents from learning slowdown in neural networks
- With a quadratic cost, it is possible that the magnitude of the gradient is very small (slowdown)
- The cross entropy can help fix this issue

Cross entropy

- The biggest asset of cross entropy is that it prevents from learning slowdown in neural networks
- With a quadratic cost, it is possible that the magnitude of the gradient is very small (slowdown)
- The cross entropy can help fix this issue
- If you are interested in this it is necessary to understand **backpropagation**, and **gradient descent**.
- interesting ressource : `http://neuralnetworksanddeeplearning.com/chap3.html`

Kullbach-Leibler Divergence

- What if you want measure the discrepancy between **distributions**, instead of **vectors** ?

Expected value (esprance)

- For a discrete random variable X that takes the values x_i with probability p_i:
$$E(X) = \sum_{i=1}^{n} p_i x_i \tag{9}$$
- For a continuous random variable X with density p(x):
$$E(X) = \int x p(x) dx \tag{10}$$

Expected value (esprance)

Exercice 1 : Computing an expected value

- For a discrete random variable X that takes the values x_i with probability p_i:

$$E(X) = \sum_{i=1}^{n} p_i x_i \tag{11}$$

- For a continuous random variable X with density p(x):

$$E(X) = \int xp(x)dx \tag{12}$$

Compute the expected value of the dice game.

Variance

$$var(X) = E\left(\left(X - E(X)\right)^2\right) \tag{13}$$

Variance and Covariance

$$var(X) = E\left((X - E(X))^2\right) \tag{14}$$

$$cov(X, Y) = E\left((X - E(X))(Y - E(Y))\right) \tag{15}$$

Kullbach-Leibler Divergence

- What if you want measure the discrepancy between **distributions**, instead of **vectors** ?
- For instance, you want to fit a distribution of your choice **model** to **empirical data**.
- The data points $(x_1, .., x_n)$ are described by an empirical distribution.

Kullbach-Leibler Divergence

- The data points $(x_1, .., x_n)$ are described by an empirical distribution.
- The **Kullbach-Leibler divergence** is a tool to compare distributions.
- It is not a distance (it is not symmetric).

Kullbach-Leibler Divergence

- $$\mathcal{D}[p||q] = \mathbb{E}_{\sim p}[\log(\frac{p}{q})] \tag{16}$$
- For discrete variables
$$\mathcal{D}[p||q] = \sum_i p(i) \log \frac{p(i)}{q(i)} \tag{17}$$
- for continuous variables
$$\mathcal{D}[p||q] = \int_X p(x) \log \frac{p(x)}{q(x)} dx \tag{18}$$

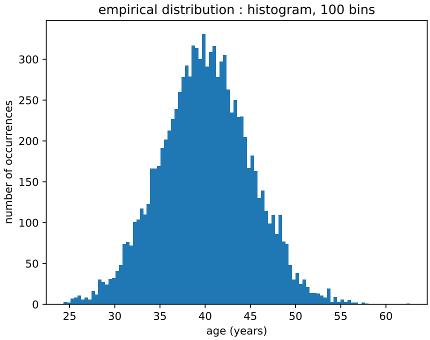
Exercice 2 : **Fitting a distribution**

- **cd kl_divergence**
- A two dimensional dataset is contained in **empirical_distribution.csv**. It represents the **age distribution** of some groupe of people. We want to study this age distribution.
- load it in **fit_empirical.py**. We will use the functions provided in the file in order to find the best model, meaning here the model M, such that $KL(M||\tilde{P})$ is smallest, with $\tilde{P}$ the empirical distribution of the data.

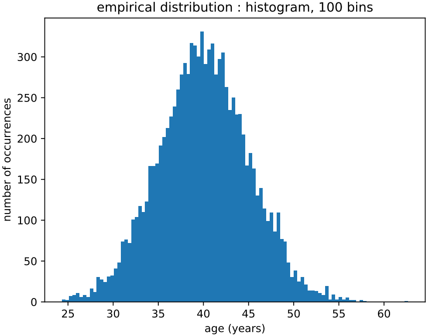
Exercice 2 : **Fitting a distribution**

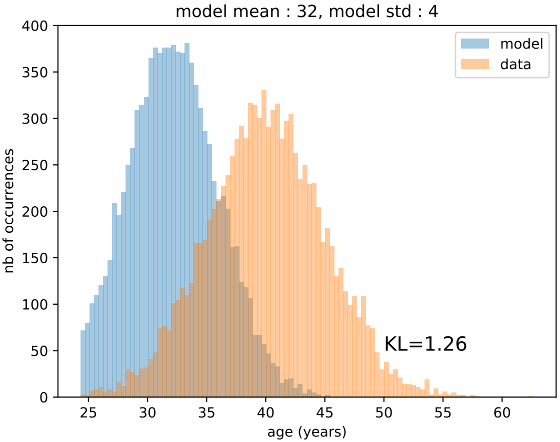
- **First step :** choice of the model
- Plot the histogram of the data : what model seems to be relevant ?

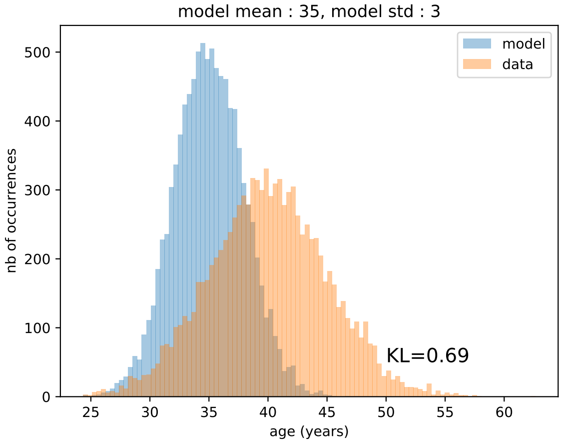
- **First step :** choice of the model
- Plot the histogram of the data : what model seems to be relevant ?



- We will use **normal laws**. We want to fint the normal law that is **the closest to the empirical data**
- We measure the proximity between the model and the empirical data with the KL divergence.




model mean : 32, model std : 4
model
data
KL=1.26
nb of occurrences
age (years)
0
50
100
150
200
250
300
350
400
25
30
35
40
45
50
55
60


model mean : 35, model std : 3
model
data
KL=0.69
nb of occurrences
age (years)
0
100
200
300
400
500
25
30
35
40
45
50
55
60

Prediction error

- Given a distance $d(\tilde{f}(x), y)$ between a prediction and an actual label, we want to measure the **statistical error** of our model
- This would mean the expected value :

$$\mathbb{E}(d(\tilde{f}(x), y)) \tag{19}$$

Prediction error

- Given a distance $d(\tilde{f}(x), y)$ between a prediction and an actual label, we want to measure the **statistical error** of our model
- This would mean the expected value :

$$\mathbb{E}(d(\tilde{f}(x), y)) \tag{20}$$

- Can we compute this ?

Prediction error

- Given a distance $d(\tilde{f}(x), y)$ between a prediction and an actual label, we want to measure the **statistical error** of our model
- This would mean the expected value :

$$\mathbb{E}(d(\tilde{f}(x), y)) \tag{21}$$

- Can we compute this ? We can **not** compute it, because we do not know the laws of probabilities of x and y.

Prediction error

We must here understand the difference between :

- The expected value coming from a certain process with some law probability $\mathbb{E}$: we can compute it when we have access to the actual probabilities of the process :

$$\mathbb{E}(d(\tilde{f}(x), y)) \tag{22}$$

Prediction error

We must here understand the difference between :

- The expected value coming from a certain process with some law probability $\mathbb{E}$: we can compute it when we have access to the actual probabilities of the process :

$$\mathbb{E}(d(\tilde{f}(x), y)) \tag{23}$$

- The empirical mean that we observe from our datapoints $(x_1, ..., x_n)$:

$$\frac{1}{n}\sum_{i=0}^{n} d(\tilde{f}(x_i), y_i) \tag{24}$$

Prediction error

We must here understand the difference between :

- The expected value coming from a certain process with some law probability $\mathbb{E}$: we can compute it when we have access to the actual probabilities of the process :

$$\mathbb{E}(d(\tilde{f}(x), y)) \tag{25}$$

- The empirical mean that we observe from our datapoints $(x_1, ..., x_n)$:

$$\frac{1}{n}\sum_{i=0}^{n} d(\tilde{f}(x_i), y_i) \tag{26}$$

- We have direct access to **only one of these :** which one ?

Prediction error

We must here understand the difference between :

- The expected value coming from a certain process with some law probability $\mathbb{E}$: we can compute it when we have access to the actual probabilities of the process :

$$\mathbb{E}(d(\tilde{f}(x), y)) \tag{27}$$

- The empirical mean that we observe from our datapoints $(x_1, ..., x_n)$:

$$\frac{1}{n}\sum_{i=0}^{n} d(\tilde{f}(x_i), y_i) \tag{28}$$

- We have direct access to **only one of these :** the empirical mean

Convergence

- **However**, under some restrictions, the empirical mean of the error **converges** towards the model error.
- If the samples are **independent and identically distributed**, and $d(\tilde{f}(x_i), y_i)$ is integrable.
- This means that there must be no bias in the samples.

The end

Questions ?