

Algorithms. Matching

Part II. Preference model.

B9 - Algorithms Matching

M-ALG-102

Introduction

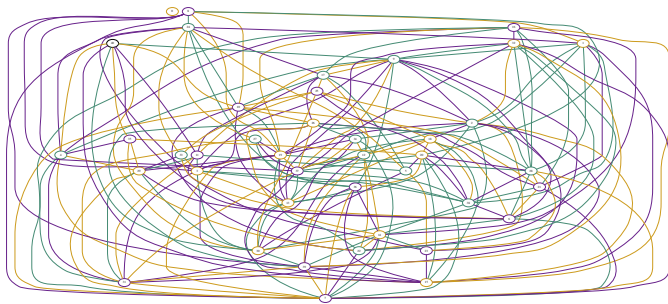


Figure: Graph

Introduction

<https://github.com/nlehir/ALGO2>

We will need matplotlib and pandas.

Day 2

Compatibility graphs

- Simple geometrical data

- Complex data

Probability distributions

- Reminders on probabilities

- Analyzing a distribution

- Optimization and Maximum Likelihood

- Gradients

Multivariate analysis and clustering

- Correlation

- Kmeans clustering

Applying the matching on a concrete example

Additional considerations and conclusions

Compatibility graphs

- ▶ Yesterday we processed graphs describing **relationship between data**
- ▶ If two nodes were related, they were linked by an edge in the graph.

Compatibility graphs

- ▶ Yesterday we processed graphs describing **relationship between data**
- ▶ If two nodes were related, they were linked by an edge in the graph.
- ▶ Today we are interested in building such graphs directly from the data, we call them **compatibility graphs**.

Compatibility graphs

We are interested in building **compatibility graphs**.

Given two nodes in a graph, should there be an edge between them
?

Compatibility graphs

We are interested in building **compatibility graphs**.

Given two nodes in a graph, should there be an edge between them ?

Note : it is not the same problem as the matching problem. In the matching problem, the edges are already defined (however we can then apply a matching algorithm to the graph obtained).

Example applications

- ▶ Social networks management
- ▶ Recommendations

Building compatibility graphs

- ▶ We will build graphs first from simple data
- ▶ Then from more complex data.

Building a graph from simple data

- ▶ We will first build a graph from simple data in the 2D euclidian space.

Euclidian distance and compatibility

Consider the following data :

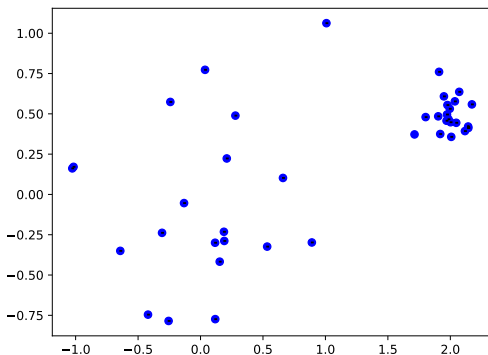


Figure: Data : we would like to define **edge** between some of them

Is this set of edges a good solution ?

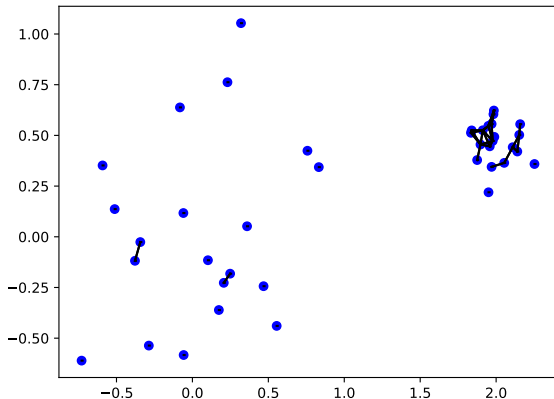


Figure: Some definition of edges

Is this set of edges a good solution ?

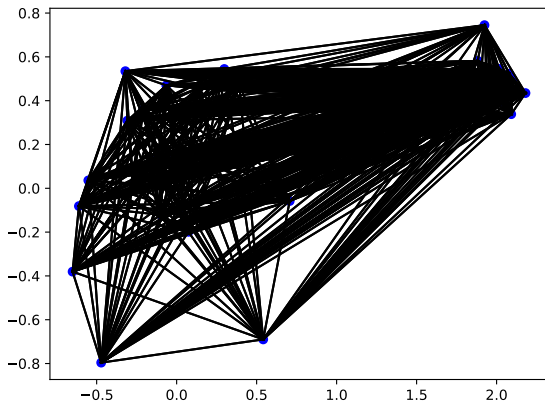
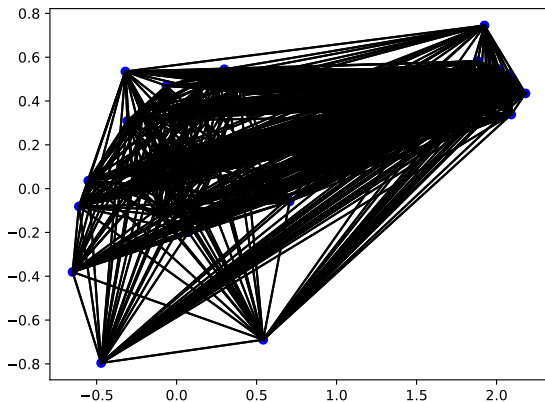


Figure: Some definition of edges

Euclidian distance and compatibility

Here, all we know about the data is their **euclidian distance** :



This one looks ok

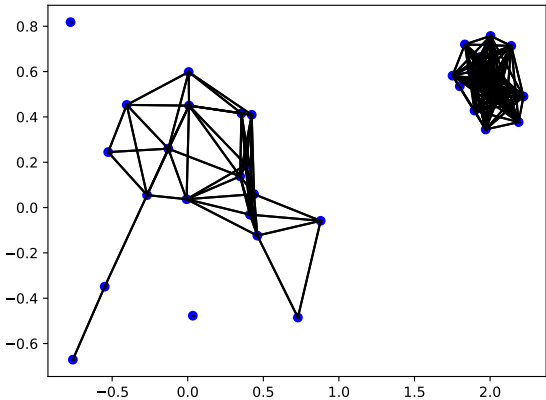


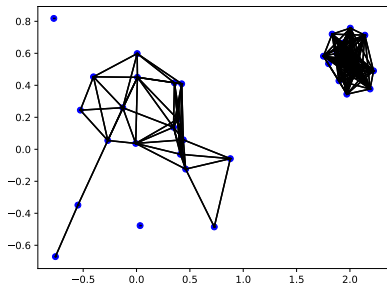
Figure: A proposition of edges

Backboard

- ▶ Euclidian distance and threshold.

Exercise 1

Set the threshold used in **buld_graph.py** to draw relevant edges between the nodes. Feel free to use another dataset !



Exercise 2

- ▶ Assess the impact of changing the distance used. Possible choices :
 - ▶ $L1$ distance (Manhattan)
 - ▶ $|||_{\infty}$ distance (backboard)
- ▶ use **build_graph_other_distance** and edit the distances used at the end of the file.
- ▶ Try several values for the threshold.

General notion of a distance

- ▶ Let us generalize what we experimentally studied.

Examples of distances

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

Examples of distances

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

- ▶ L2 : $\|x - y\|_2^2 = \sum_{k=1}^p (x_k - y_k)^2$ (Euclidian distance)

Examples of distances

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

- ▶ L2 : $\|x - y\|_2^2 = \sum_{k=1}^p (x_k - y_k)^2$ (Euclidian distance)
- ▶ L1 : $\|x - y\|_1 = \sum_{k=1}^p |x_k - y_k|$ (Manhattan distance)

Examples of distances

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

- ▶ L2 : $\|x - y\|_2^2 = \sum_{k=1}^p (x_k - y_k)^2$ (Euclidian distance)
- ▶ L1 : $\|x - y\|_1 = \sum_{k=1}^p |x_k - y_k|$ (Manhattan distance)
- ▶ weighted L1 : $\sum_{k=1}^p w_k |x_k - y_k|$

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 $d : E \times E \rightarrow \mathbb{R}_+$ that must :

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

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** : $\forall x, y, d(x, y) = 0 \Leftrightarrow x = y$

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**
 $\forall x, y, z, d(x, y) \leq d(x, z) + d(y, z)$

Building compatibility graphs for more complex data

- ▶ We will do the same with more complex data:
 - ▶ possibly more dimensions
 - ▶ possibility categorical variables

Random variables

- ▶ A **random variable** is a quantity that can take several values

Random variables

- ▶ A **random variable** is a quantity that can take several values
- ▶ For instance :
 - ▶ the result of a dice throw

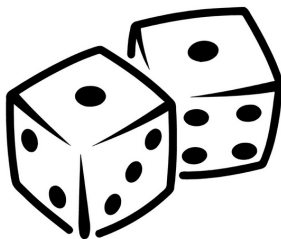


Figure: Dice

Random variables

- ▶ A **random variable** is a quantity that can take several values
- ▶ For instance :
 - ▶ the result of a dice throw
 - ▶ waiting time with RATP



Figure: Some metro station

Random variables

- ▶ A **random variable** is a quantity that can take several values
- ▶ For instance :
 - ▶ the result of a dice throw
 - ▶ waiting time with RATP
 - ▶ weather



Figure: Weather in November

Random variables

- ▶ A **random variable** is a quantity that can take several values
- ▶ For instance :
 - ▶ the result of a dice throw
 - ▶ waiting time with RATP
 - ▶ weather
 - ▶ number of cars taking the periphrique at the same time

Random variables

What are the differences between these random variables ?

Random variables

What are the differences between these random variables ?

- ▶ Some are **continuous**, others **discrete**
- ▶ **continuous** :

Random variables

What are the differences between these random variables ?

- ▶ Some are **continuous**, others **discrete**
- ▶ **continuous** : weather, RATP

Random variables

What are the differences between these random variables ?

- ▶ Some are **continuous**, others **discrete**
- ▶ **continuous** : weather, RATP
- ▶ **discrete** : dice (6 possibilities), number of cars (> 10000), PSG (3)

Probability distributions

- ▶ A random variable is linked to a **probability distribution**.

Probability distributions

- ▶ A random variable is linked to a **probability distribution**.
- ▶ It quantifies the probability of observing one outcome.

Probability distributions

- ▶ A random variable is linked to a **probability distribution**, which is a function P
- ▶ It quantifies the probability of observing one outcome.
- ▶ For a discrete variable : each possible outcome is associated with a number between 0 and 1

Probability distributions

- ▶ For a dice game, the possible outcomes are in the set $\{1, 2, 3, 4, 5, 6\}$
- ▶ For a dice game : $P(1) = ?$ $P(2) = ?$ $P(3) = ?$ $P(4) = ?$
 $P(5) = ?$ $P(6) = ?$

Probability distributions

- ▶ For a dice game, the possible outcomes are in the set $\{1, 2, 3, 4, 5, 6\}$
- ▶ For a dice game : $P(1) = \frac{1}{6}$, $P(2) = \frac{1}{6}$, $P(3) = \frac{1}{6}$, $P(4) = \frac{1}{6}$, $P(5) = \frac{1}{6}$, $P(6) = \frac{1}{6}$
- ▶ This is called a **uniform distribution**

Probability distributions

- ▶ Periphrique :

Probability distributions

- ▶ Periphrique : probably a time-dependent very complicated distribution

Continuous variables

- ▶ How would you model a continuous variable ? Can you assign a number to a waiting time or a weather ?

Continuous variables

- ▶ How would you model a continuous variable ? Can you assign a number to a waiting time or a weather ?
- ▶ One needs to use **probability densities**. Formally, the probability of being between x and $x + dx$ is $p(x)dx$.

Continuous variables

- ▶ How would you model a continuous variable ? Can you assign a number to a waiting time or a weather ?
- ▶ One needs to use **probability densities**. Formally, the probably of being between x and $x + dx$ is $p(x)dx$.
- ▶ Let's see some examples

Uniform discrete

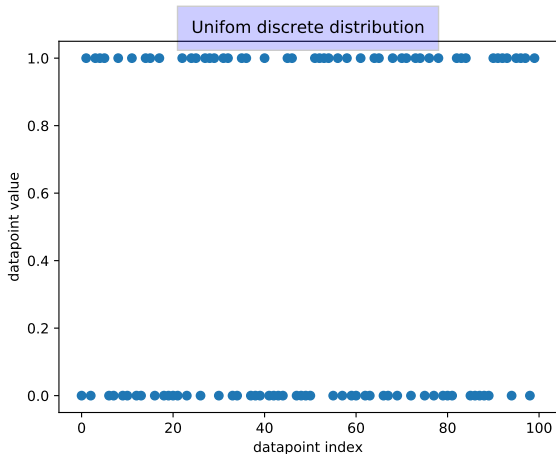


Figure: Uniform discrete distribution

Bernoulli

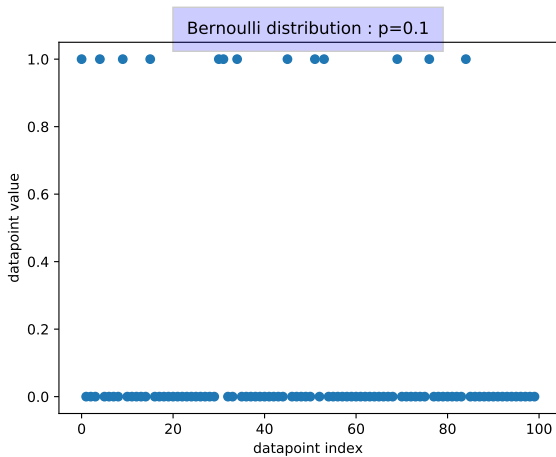


Figure: Bernoulli distribution

Bernoulli

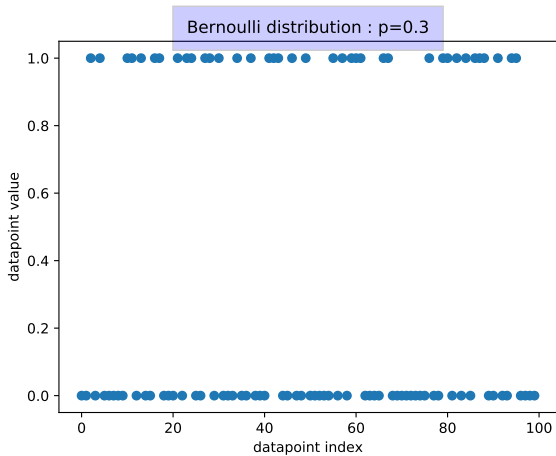


Figure: Bernoulli Distribution

Bernoulli

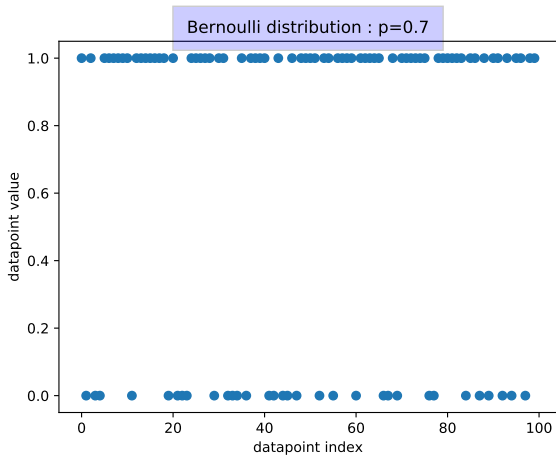


Figure: Bernoulli Distribution

Uniform continuous

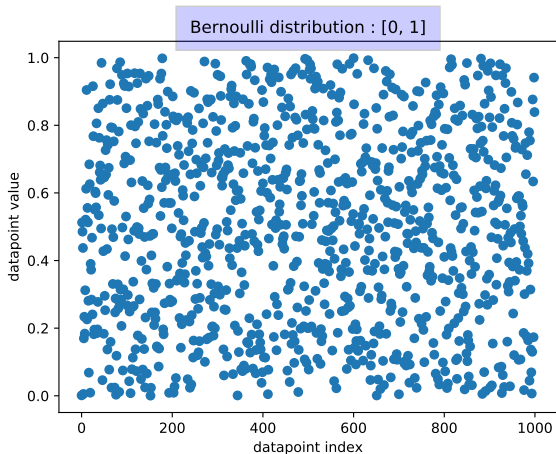


Figure: Uniform continuous distribution

Uniform continuous

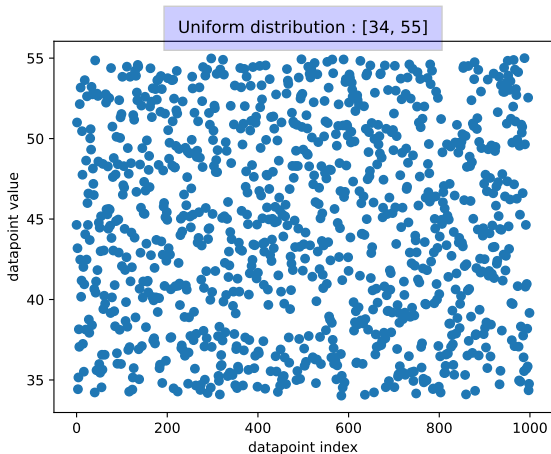


Figure: Uniform continuous distribution

Normal

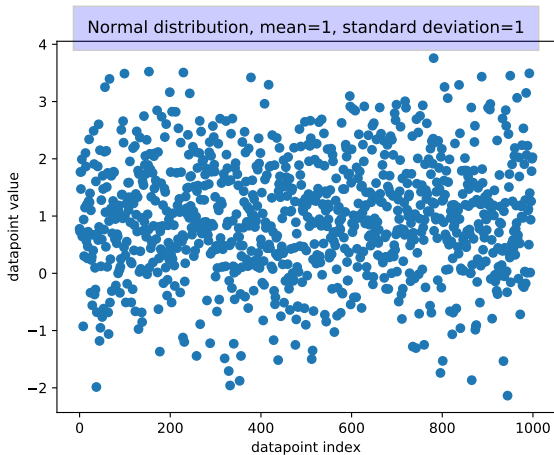


Figure: Normal distribution

Normal

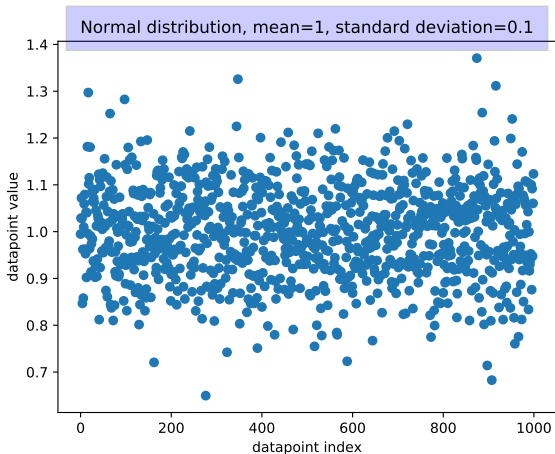


Figure: Normal distribution

Normal

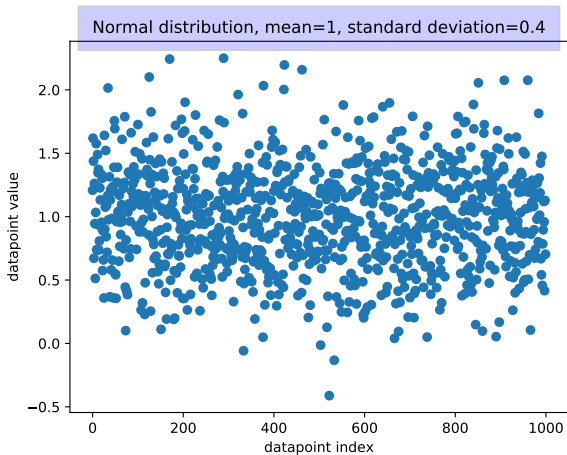


Figure: Normal distribution

White noise

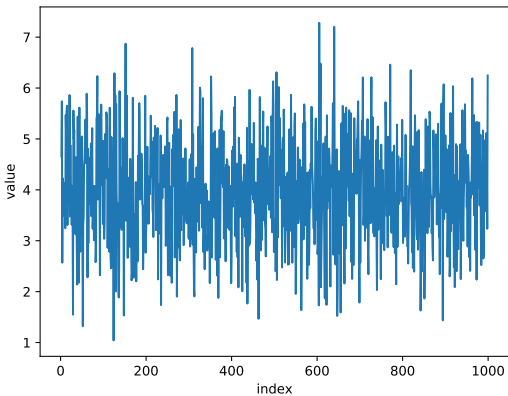


Figure: White noise

Histograms

Is looking at the raw dataset really **informative** ?

Histograms

Is looking at the raw dataset really **informative** ?
It is informative, but often a **histogram** tells more.

Uniform discrete

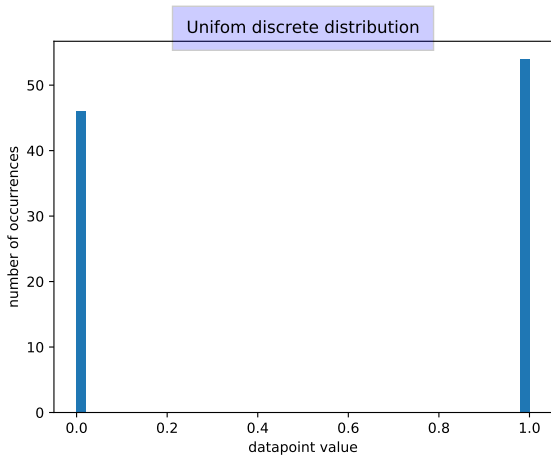


Figure: Histogram 1

Bernoulli

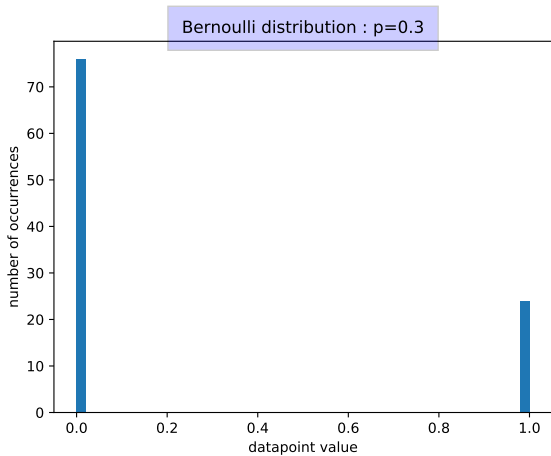


Figure: Histogram 2

Uniform continuous

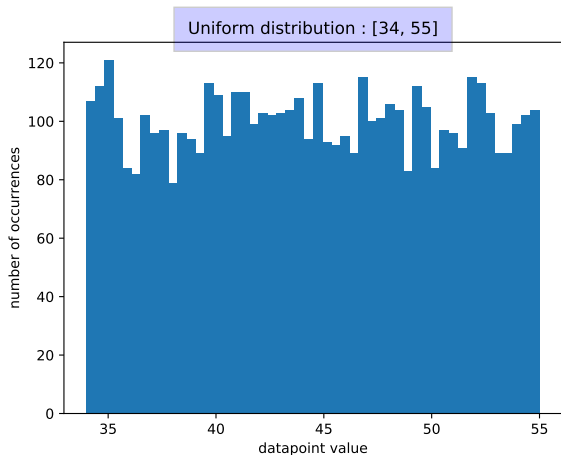


Figure: Histogram 3

Normal

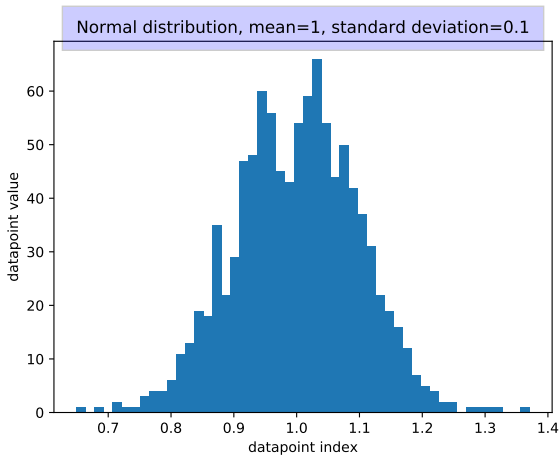


Figure: Histogram 4

Exercise

I put values in the file **mysterious_distro_1.csv**

Exercise

I put values in the file **mysterious_distro_1.csv**

Can you analyze these values in terms of a **distribution** ?

Use **read_myst_1** to analyze the distribution (suggestion : change the number of bins used)

Exercise

When you have guessed the kind of distribution it is, you need to find its **parameters**.

- ▶ its mean
- ▶ its standard deviation

This is called **fitting** a distribution to a dataset : it's a classical machine learning problem.

To do so, uncomment the last section of the script **read_myst_1**

Distribution 1

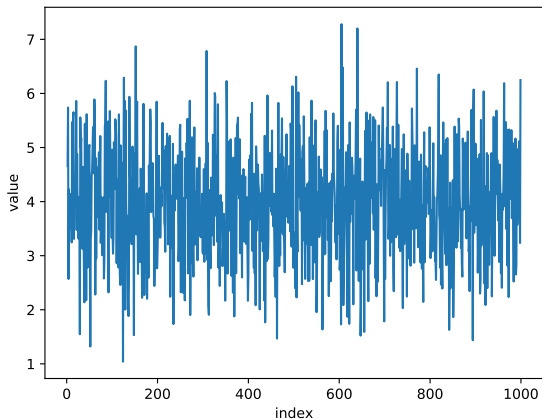


Figure: The data we analyze

histograms

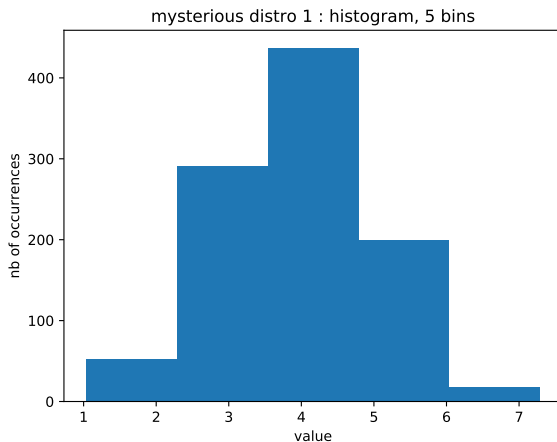


Figure: 5 bins

histograms

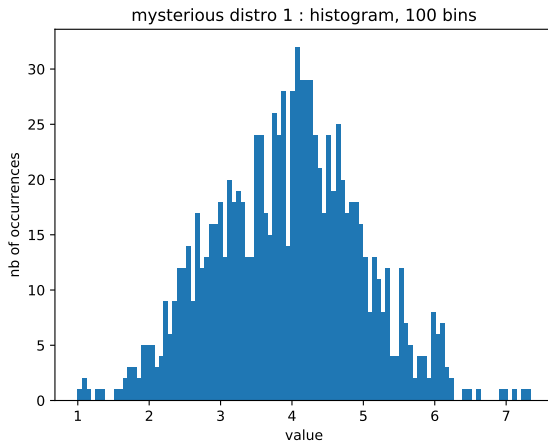


Figure: 100 bins

histograms

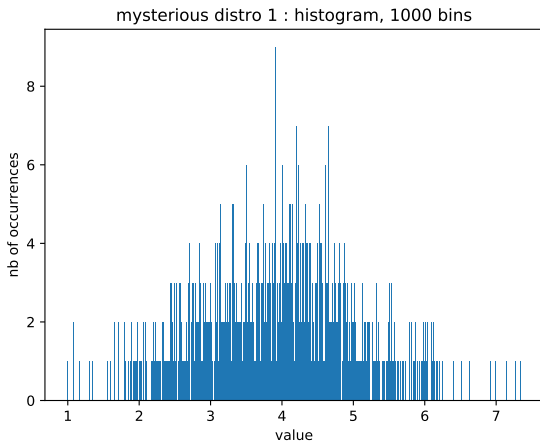


Figure: 1000 bins (too many)

Normal distribution

```
import csv
import numpy as np

file_name = 'mysterious_distro_1.csv'

mean = 4
std_dev = 1
nb_point = 1000

with open('csv_files/' + file_name, 'w') as csvfile:
    filewriter = csv.writer(csvfile, delimiter=',')
    for point in range(1, nb_point):
        random_variable = np.random.normal(loc=mean, scale=std_dev)
        filewriter.writerow([str(point), str(random_variable)])
```

Figure: **create_normal.py** : Creation of the distribution

Second example

Let's try to perform the same analysis on the file **mysterious_distro_2.csv** using **read_myst_1**.

Second example

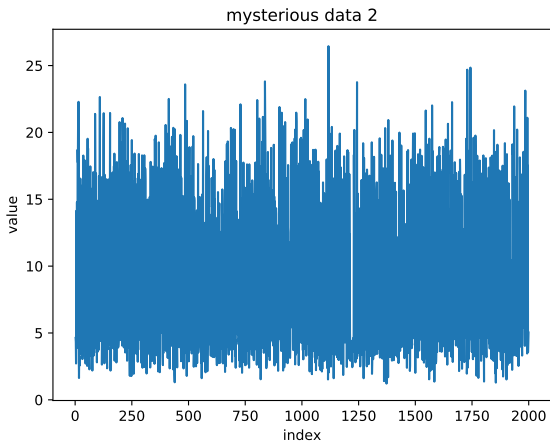


Figure: Second distribution

Multimodal distribution

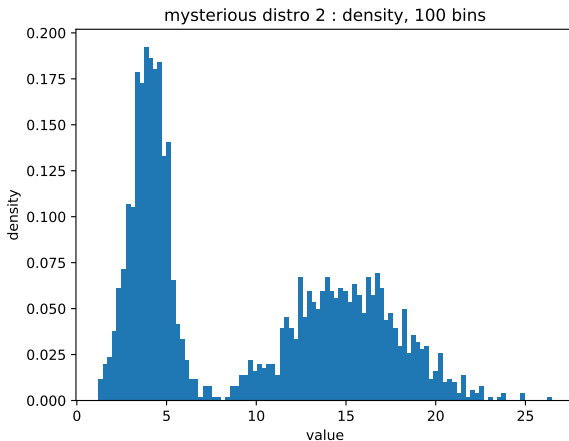


Figure: This distribution has several **modes**

Multimodal distribution

```
mean_1 = 4
std_dev_1 = 1
nb_point_1 = 1000

mean_2 = 15
std_dev_2 = 3
nb_point_2 = 1000

nb_point = nb_point_1 + nb_point_2

with open('csv_files/' + file_name, 'w') as csvfile:
    filewriter = csv.writer(csvfile, delimiter=',')
    for point in range(1, nb_point):
        if random.randint(1, 2) == 1:
            random_variable = np.random.normal(loc=mean_1, scale=std_dev_1)
            filewriter.writerow([str(point), str(random_variable)])
        else:
            random_variable = np.random.normal(loc=mean_2, scale=std_dev_2)
            filewriter.writerow([str(point), str(random_variable)])
```

Figure: `create_bimodal.py` : Generation of multimodal distribution

Fitting

In most cases, it won't be that straightforward to fit a distribution :

Fitting

In most cases, it won't be that straightforward to fit a distribution
:

- ▶ what distribution do we want to use ?
- ▶ even if we know the right shape of the distribution, how to choose the parameters ?

Maximum Likelihood

The **Maximum Likelihood** method is one example method used in Machine Learning.

Say you have a dataset (x_1, \dots, x_n) .

Maximum Likelihood

The **Maximum Likelihood** method is one example method used in Machine Learning.

Say you have a dataset (x_1, \dots, x_n) .

You first need to choose a **model** (which is the distribution) of your dataset, p .

Maximum Likelihood

The **Maximum Likelihood** method is the one used in Machine Learning.

Say you have a dataset (x_1, \dots, x_n) .

You first need to choose a **model** (which is the distribution) of your dataset, p .

Then, you must optimize the **parameters of this model**, noted θ .

Maximum Likelihood

The Likelihood of your model is

$$L(\theta) = \prod_{i=1}^n p(x_i|\theta) \quad (1)$$

Maximum Likelihood

The Likelihood of your model is

$$L(\theta) = \prod_{i=1}^n p(x_i|\theta) \quad (2)$$

This is the function that you want to **maximise**.

Maximum Likelihood

Most of the time it's written this way : "minimise $-\log L(\theta)$ "

Why ?

Maximum Likelihood

Most of the time it's written this way : "minimise $-\log L(\theta)$ "
Because the log **transforms the product into a sum**, which is easier to **derivate**.

Maximum Likelihood

$$-\log L(\theta) = -\sum_{i=1}^n \log(p(x_i|\theta)) \quad (3)$$

Max Likelihood

So how can we minimise $-\log L(\theta)$? In the case of very large datasets, and large numbers of parameters (tens, hundredths, more), most of the time an **analytic solution** is not available. So people use **gradient descent**.

The gradient descent

We want x to **minimise** f . We perform, until some criteria is satisfied :

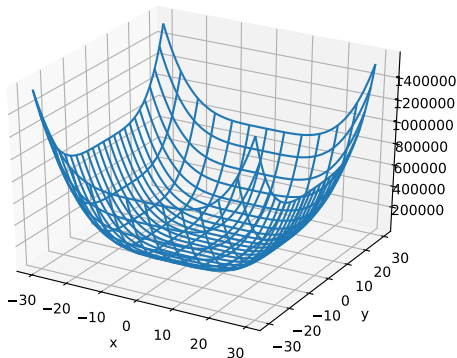
$$x \leftarrow x - \alpha \nabla_f(x) \quad (4)$$

Use the file "gradient_algo.py" and implement the gradient algorithm on a simple example.

I inserted two errors in the code.

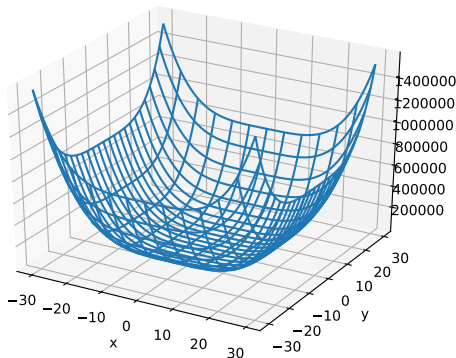
The gradient descent

$$x \leftarrow x - \alpha \nabla_f(x) \quad (5)$$



The gradient descent

Experiment with it, try to change all the parameters and to break it again. Is it stable ?



Multidimensional vectors

We can consider data that live in higher dimensional spaces than 2.

Multidimensional vectors

We can consider data that live in higher dimensional spaces than

2. Examples ?

Multidimensional vectors

We can consider data that live in higher dimensional spaces than

2. Examples ?

- ▶ images
- ▶ sensor that receives **multimodal information**

Correlation

Sometimes the components of a multidimensional vector (x_1, \dots, x_n) are not independent.

Correlation

Sometimes the components of a multidimensional vector (x_1, \dots, x_n) are not independent.

To study this, we can use the **covariance** of the two components, or the **correlation** which is actually clearer.

Example

Look at the data contained in **mysterious_distro_3.csv**

They contain a random variable with 5 dimensions. Some of these dimensions are correlated.

Think for instance to physics : temperature and pressure, etc. If you have measurements of temperature and pressure, the two would probably be **correlated**.

Correlation matrix

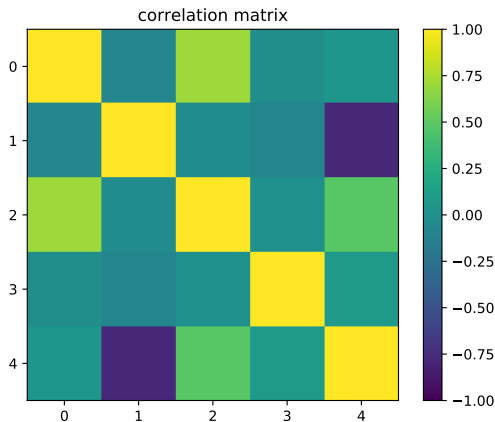


Figure: Correlation matrix for the distribution

Generation of the data

```
mean_1 = 4
std_dev_1 = 1

mean_2 = 15
std_dev_2 = 3

mean_3 = -5
std_dev_3 = 2

mean_noise = 0
noise_std_dev = 1

nb_point = 1000

with open('csv_files/' + file_name, 'w') as csvfile:
    filewriter = csv.writer(csvfile, delimiter=',')
    for point in range(1, nb_point):
        noise = np.random.normal(loc=mean_noise, scale=noise_std_dev)
        random_variable_1 = np.random.normal(loc=mean_1, scale=std_dev_1)
        random_variable_2 = np.random.normal(loc=mean_2, scale=std_dev_2)
        random_variable_3 = random_variable_1 + noise
        random_variable_4 = np.random.normal(loc=mean_3, scale=std_dev_3)
        random_variable_5 = -0.4 * random_variable_2 + noise
        filewriter.writerow([str(point),
                             str(random_variable_1),
                             str(random_variable_2),
                             str(random_variable_3),
                             str(random_variable_4),
                             str(random_variable_5)])
```

Figure: Multidimensional random variable

Kmeans clustering

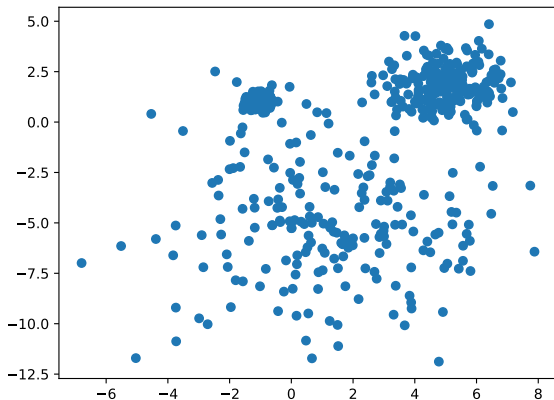


Figure: Data we want to cluster

Kmeans clustering

Modify the **kmeans.py** file so that it performs the kmeans algorithm.

I inserted two errors in the code.

You should obtain something like this:

Kmeans

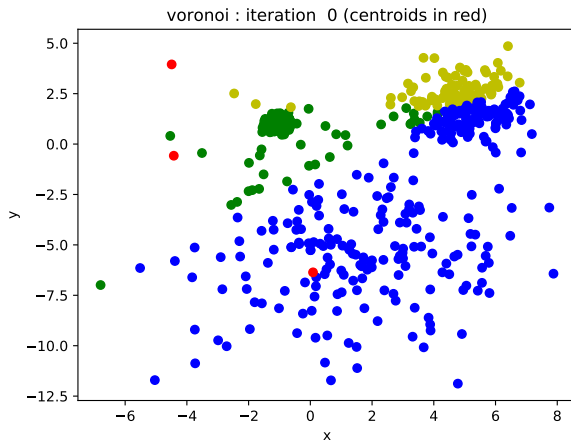


Figure: Voronoi 0th iteration

Kmeans

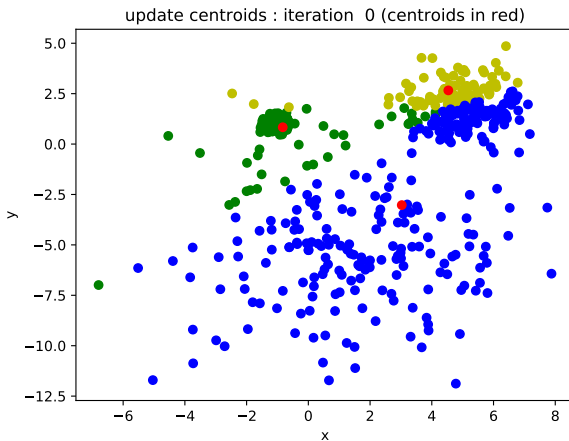


Figure: Centroids 0th iteration

Kmeans

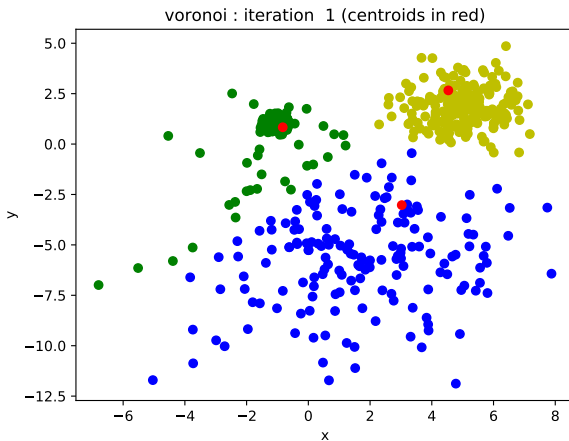


Figure: Voronoi 1st iteration

Kmeans

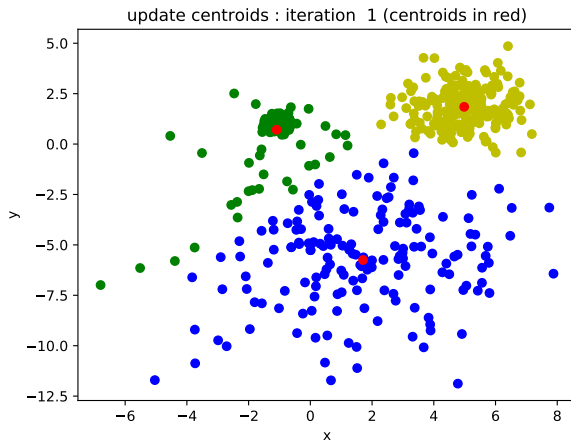


Figure: Centroids 1st iteration

Matching

- ▶ Let's apply what we learnt on the matching to a concrete situation.

Similarities

How do you cluster when you don't have a distance between your data ?

Similarities

How do you cluster when you don't have a distance between your data ?

In some cases, all you have is a **similarity**.

Similarities

How do you cluster when you don't have a distance between your data ?

In some cases, all you have is a **similarity**.

Some clustering methods like **Spectral Clustering** work with similarities.

How do you evaluate the quality of a clustering ?

Also, how do you choose the number of clusters ?

How do you evaluate the quality of a clustering ?

Also, how do you choose the number of clusters ? Example methods :

- ▶ Homogeneity of the clusters
- ▶ Stability of the result when launching the algorithm many times
- ▶ Elbow method
- ▶ Normalized cut
- ▶ Separation of the clusters (the mean distance between pairs of centroids is large)
- ▶ Silhouette coefficient

Example

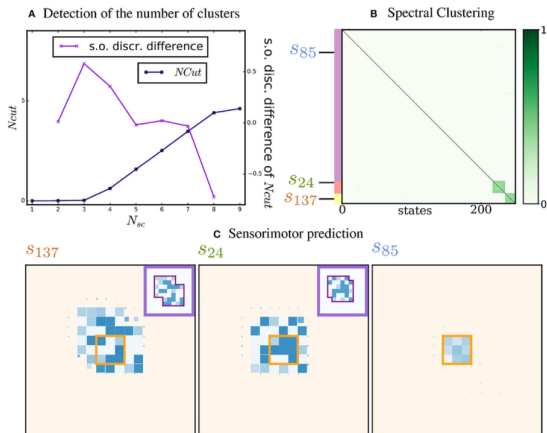


Figure: In a), the elbow method is used to choose the number of clusters.
[Le Hir et al., 2018]

Other interesting notions

- ▶ Agglomerative clustering (CHA : classification Hirarchique Ascendante)
- ▶ Xmeans : improvement of k means
- ▶ If you know more about probabilities or are curious :
 - ▶ Latent variables and variational learning
 - ▶ Auto Encoders
 - ▶ Boltzmann Machines

Conclusion

Different kinds of problems exist :

- ▶ P problems where exact polynomial solutions exist (max matching)
- ▶ For other problems :
 - ▶ exhaustive search works but is too slow
 - ▶ to solve the problem a balance between rapidity and quality must be found.
- ▶ Evaluating the quality of a result is not an easy task.

Questions ?

References



Le Hir, N., Sigaud, O., and Laflaquière, A. (2018). Identification of Invariant Sensorimotor Structures as a Prerequisite for the Discovery of Objects. *Frontiers in Robotics and AI*, 5(June):1–14.

Project

- ▶ Description of the project