

# INF8175 - Intelligence artificielle

*Méthodes et algorithmes*

## Module 8: Apprentissage non-supervisé



POLYTECHNIQUE  
MONTRÉAL

Quentin Cappart

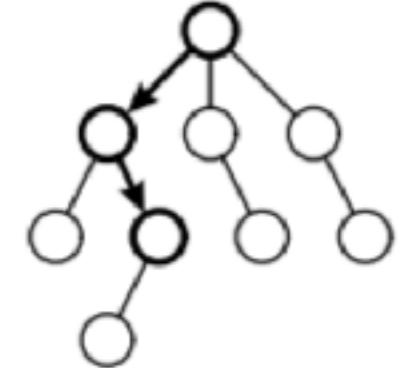
# Contenu du cours

## Raisonnement par recherche (essais-erreurs avec de l'intuition)

**Module 1:** Stratégies de recherche

**Module 2:** Recherche en présence d'adversaires

**Module 3:** Recherche locale



## Raisonnement logique

**Module 4:** Programmation par contraintes

**Module 5:** Agents logiques



## Raisonnement par apprentissage

**Module 6:** Apprentissage supervisé

**Module 7:** Réseaux de neurones et apprentissage profond

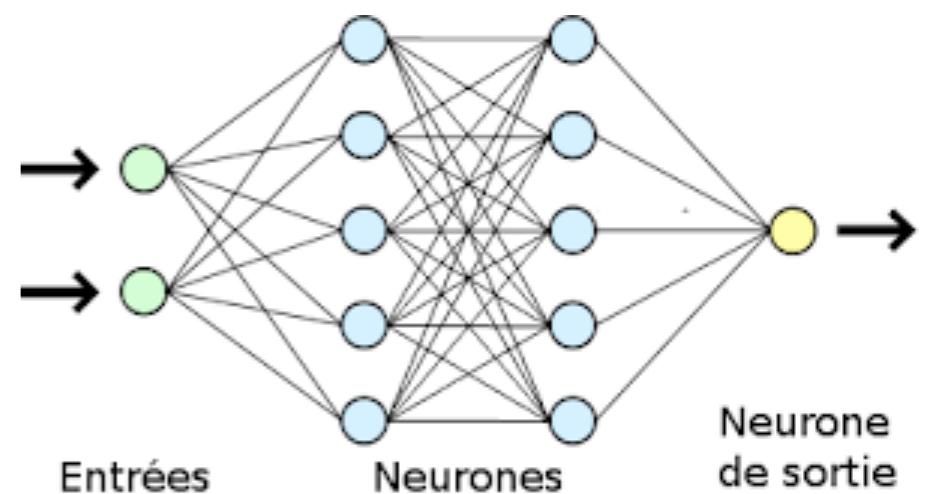
**Module 8:** Apprentissage non-supervisé

**Module 9:** Apprentissage par renforcement



## Considérations pratiques et sociétales

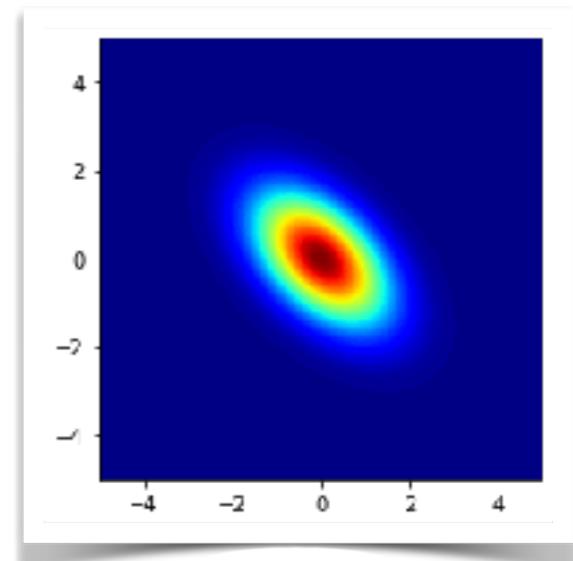
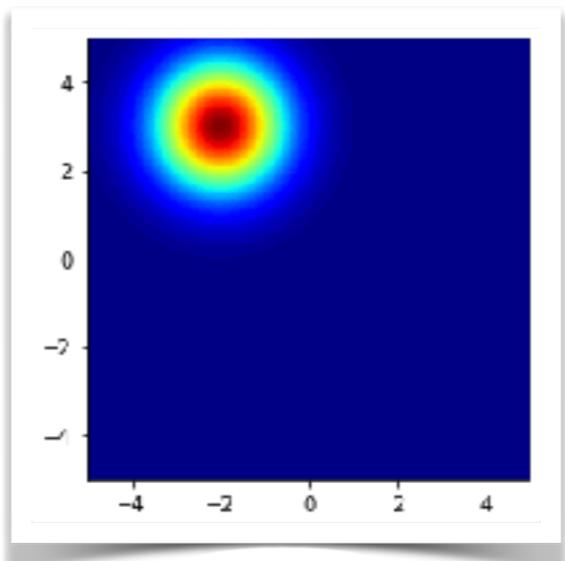
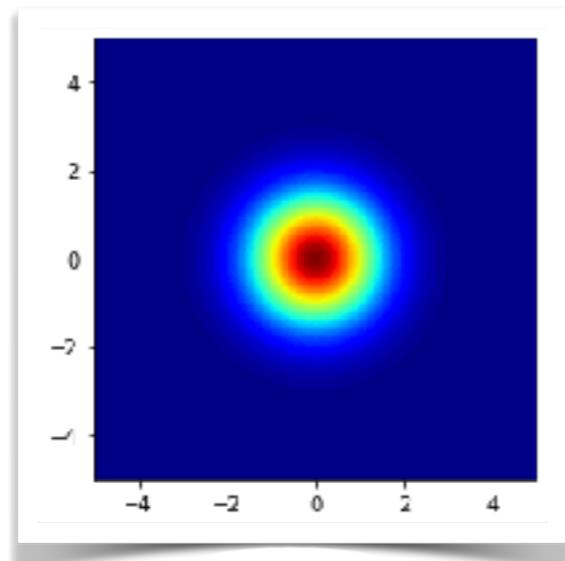
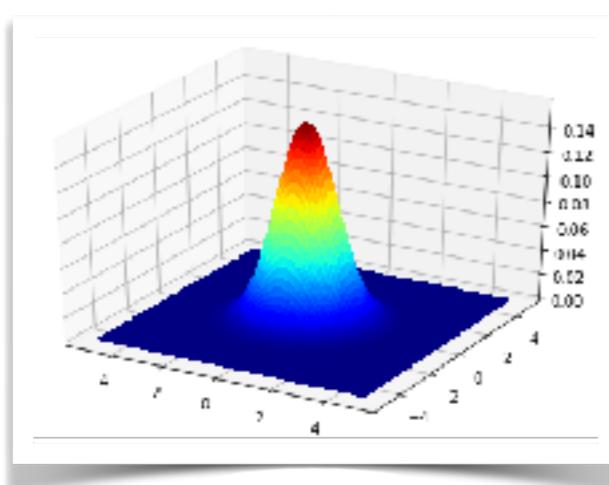
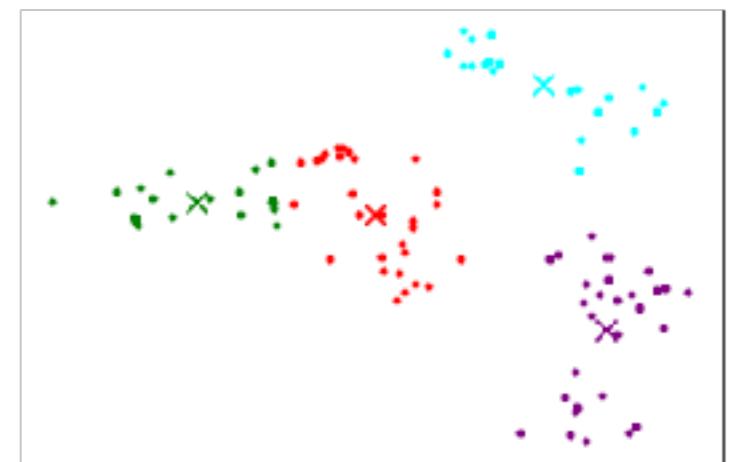
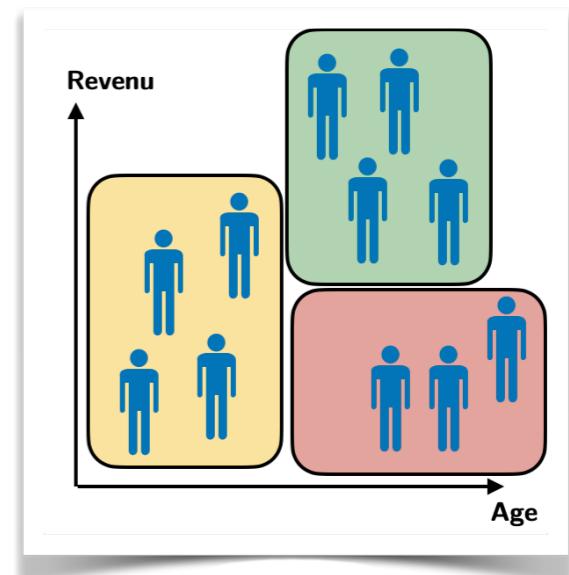
**Module 10:** Utilisation en industrie, éthique, et philosophie



# Table des matières

## Apprentissage non-supervisé

1. Définition et applications de l'apprentissage non-supervisé
2. Formalisation du problème de regroupement (*clustering*)
3. Algorithme *K-means*
4. Formalisation du problème de détection d'anomalies
5. Notion de statistique: maximum de vraisemblance
6. Modélisation par des distributions normales univariées
7. Modélisation par une distribution normale multivariée



# Apprentissage non-supervisé

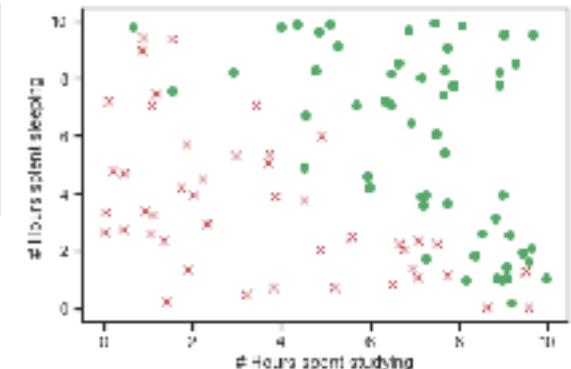
## Apprentissage supervisé

**Principe:** apprentissage qui se fait à l'aide de données dont on connaît la vraie valeur



$$D : \left\{ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)}) \right\}$$

**Vraie valeur (ground truth):** Pour chaque donnée  $x^{(i)}$ , on connaît le label  $y^{(i)}$



Ce label est indispensable pour entraîner un modèle (minimiser la fonction de coût)



Que se passe t-il si on a pas accès à ces labels ?

On n'a plus aucun moyen d'entraîner le modèle...



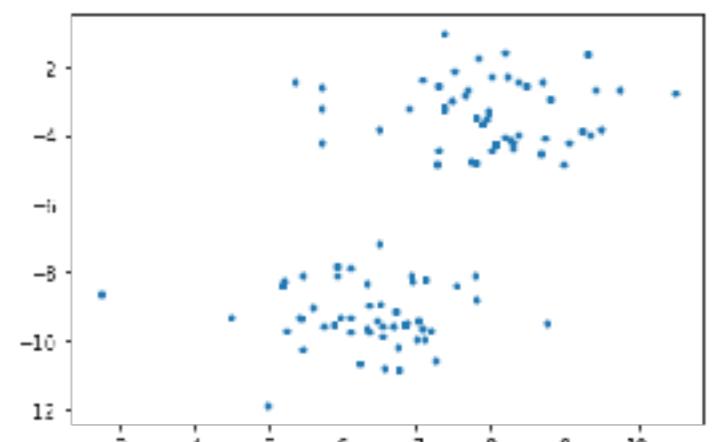
## Apprentissage non-supervisé

Type d'apprentissage qui se fait sûr base de données **non-labellisées**



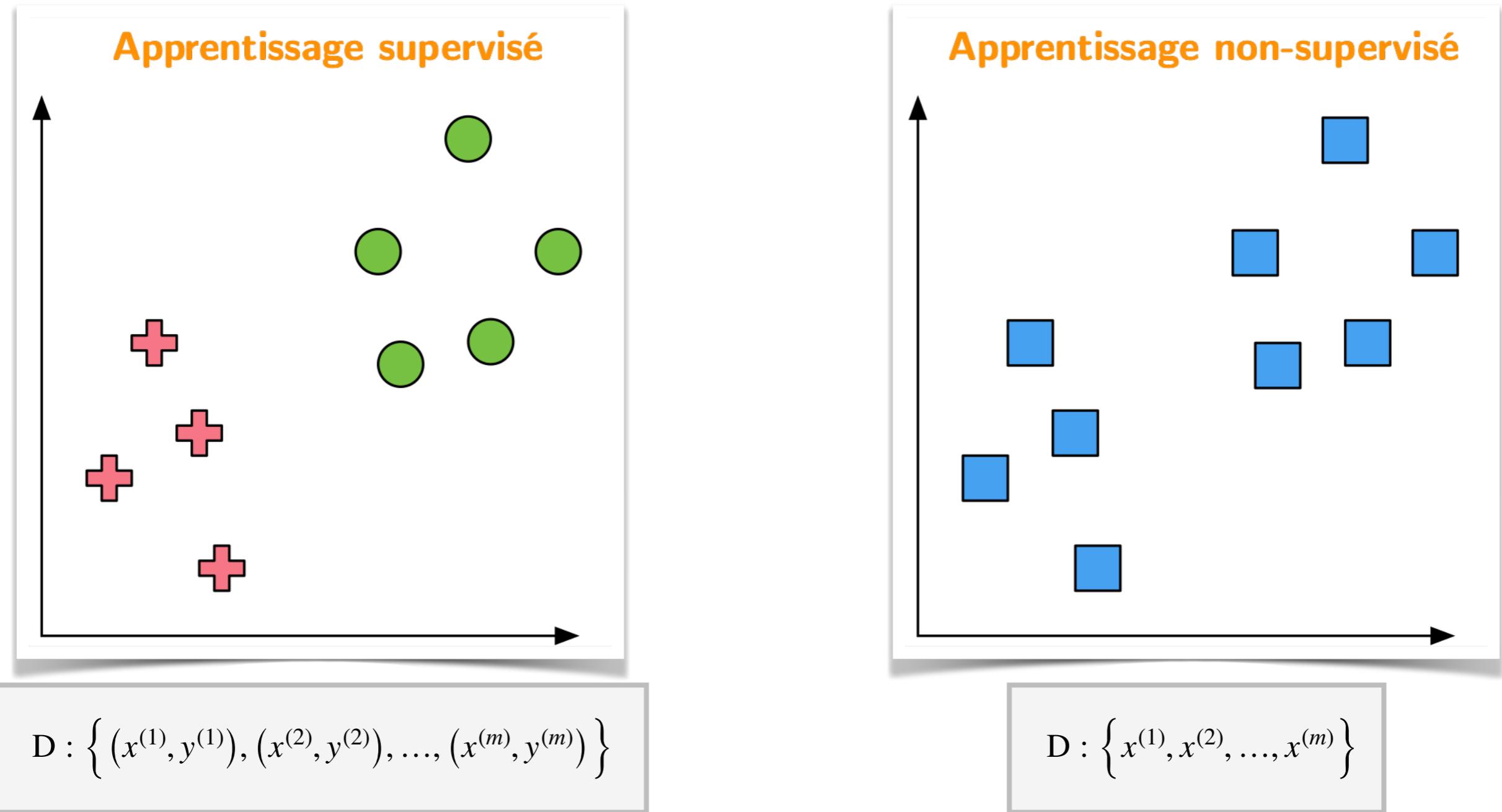
$$D : \left\{ x^{(1)}, x^{(2)}, \dots, x^{(m)} \right\}$$

**Note:** on a quand même besoin de données, mais sans la présence d'un label



**Intuition:** la tâche est plus compliquée car on a moins d'information à disposition

# Différences avec l'apprentissage supervisé



**Apprentissage supervisé:** apprentissage d'une fonction de séparation

**Apprentissage non-supervisé:** apprentissage d'une fonction de regroupement ou d'exclusion

**Point commun:** cela revient à la minimisation d'une fonction coût (ou la maximisation d'une précision)

# Application: segmentation du marché

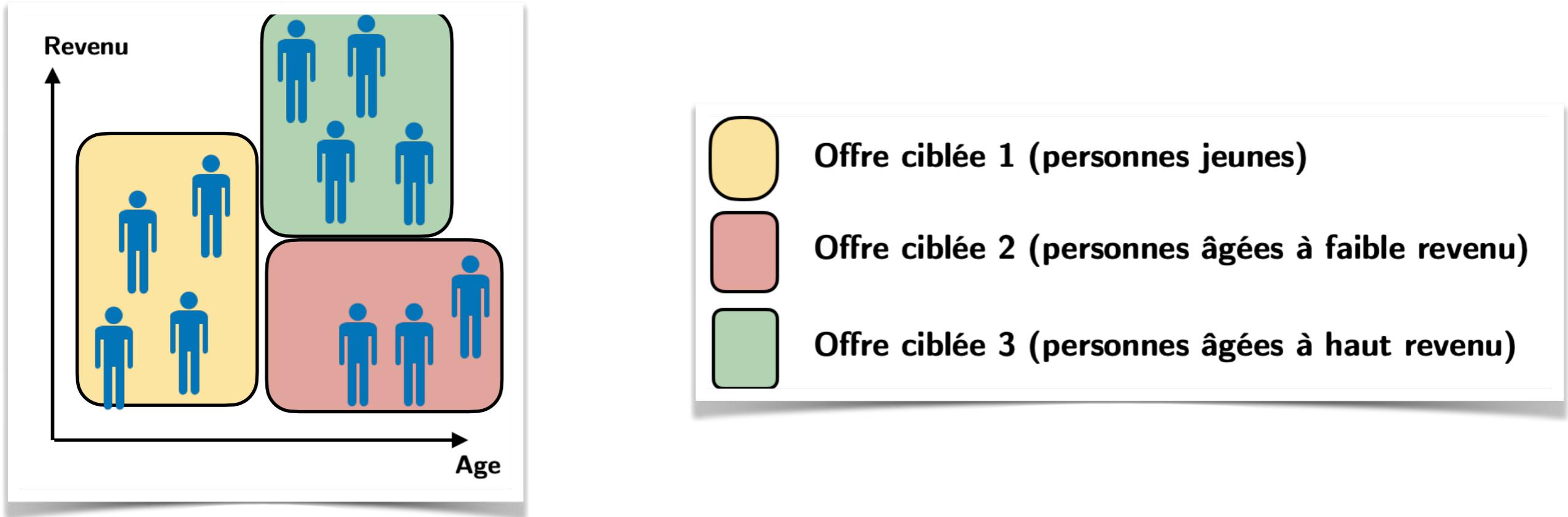
## Segmentation du marché

Opération effectuée en marketing stratégique pour diviser la demande en plusieurs groupes cohérents

**Etape 1:** établir les caractéristiques des différents clients (critères géographiques, civils, individuels, etc.)

**Etape 2:** regrouper les clients similaires en un groupe cohérent (*cluster*)

**Etape 3:** établir des offres ou des stratégies ciblées pour chacun des groupes



**Objectif:** permet de définir des offres ciblées à chaque groupe

**Applications:** problème présent dans énormément de contextes impliquant une clientèle à toucher

# Application: cybersécurité

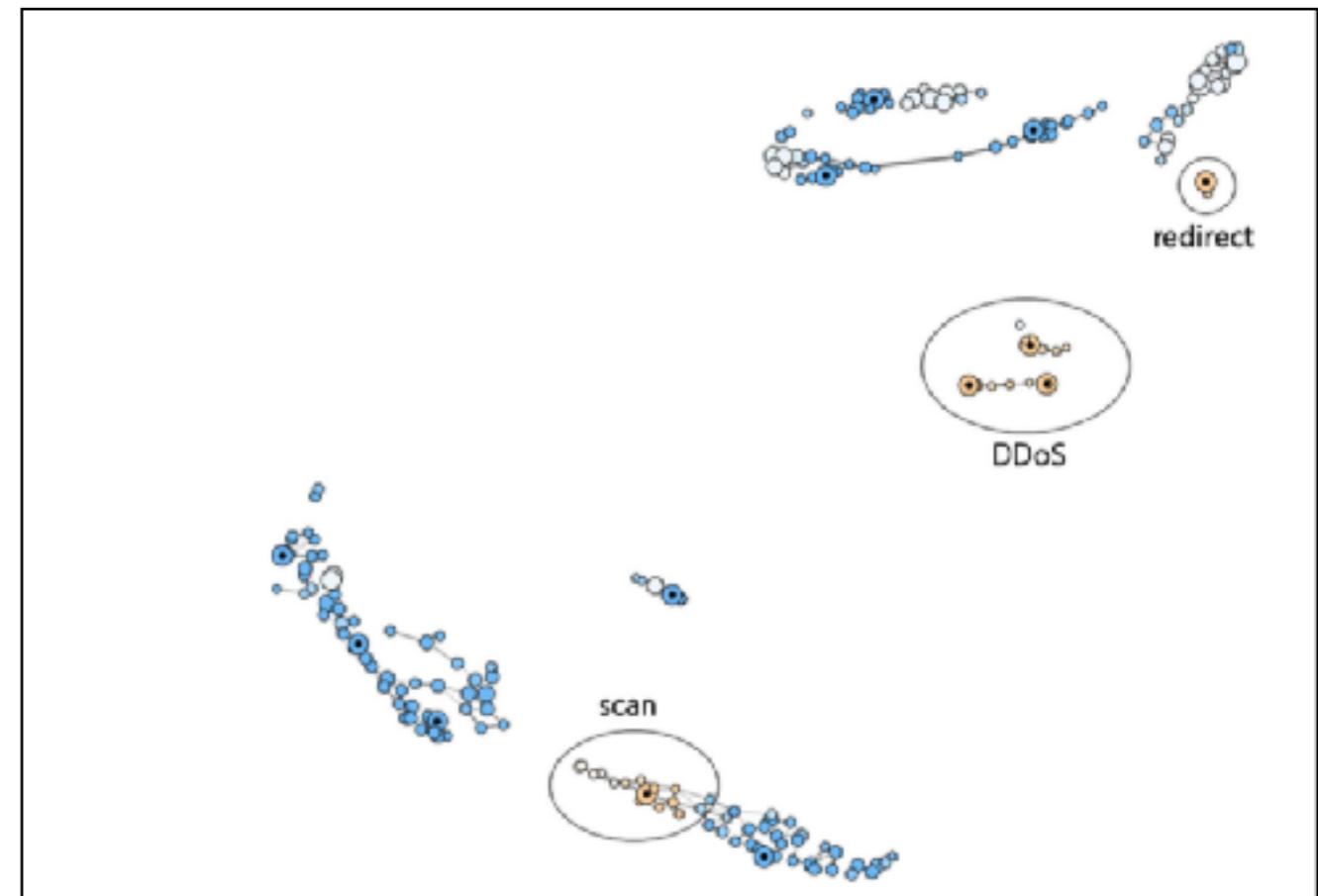
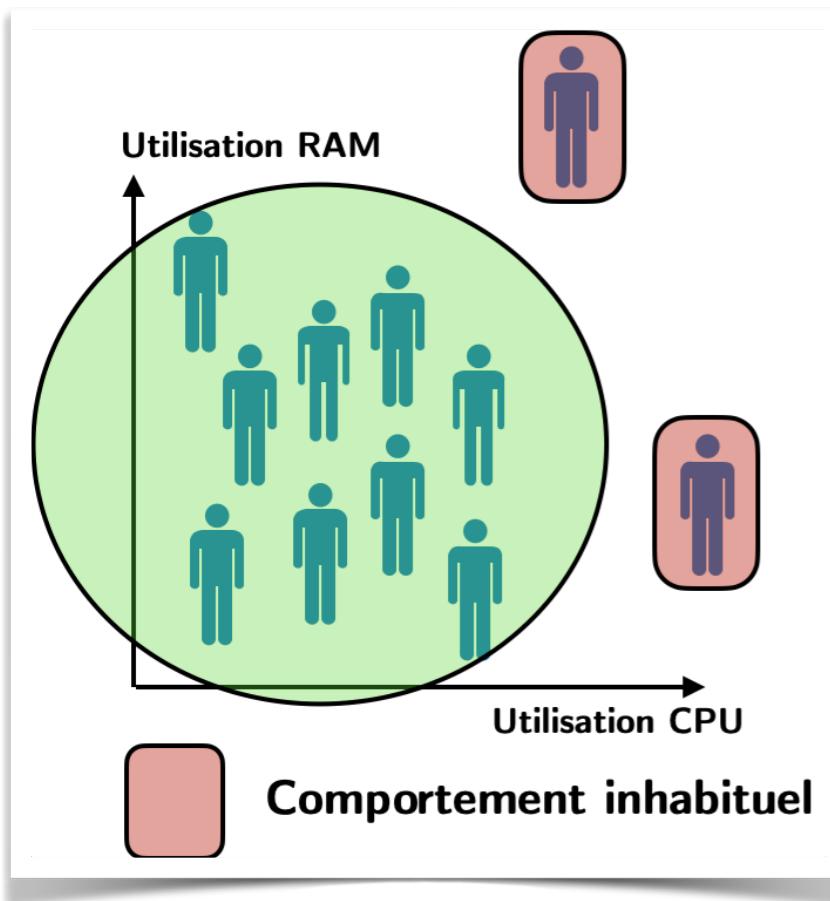
## Détection de comportements frauduleux

Opération effectuée en cybersécurité pour surveiller le comportement d'utilisateurs dans un réseau

**Etape 1:** établir les caractéristiques des différents utilisateurs (ressources utilisées, position actuelle, etc.)

**Etape 2:** établir un profil d'utilisation normale en fonction des utilisateurs présents

**Etape 3:** détecter les comportements inhabituels



**Objectif:** permet de contrôler automatiquement les comportements suspects dans un réseau

# Application: systèmes de recommandation

## Systèmes de recommandation

Opération consistant à recommander un ensemble d'entités à un utilisateur

Etape 1: établir les caractéristiques des utilisateurs (préférences, achats précédents, etc.)

Etape 2: déceler les entités qu'un utilisateur serait enclins à vouloir

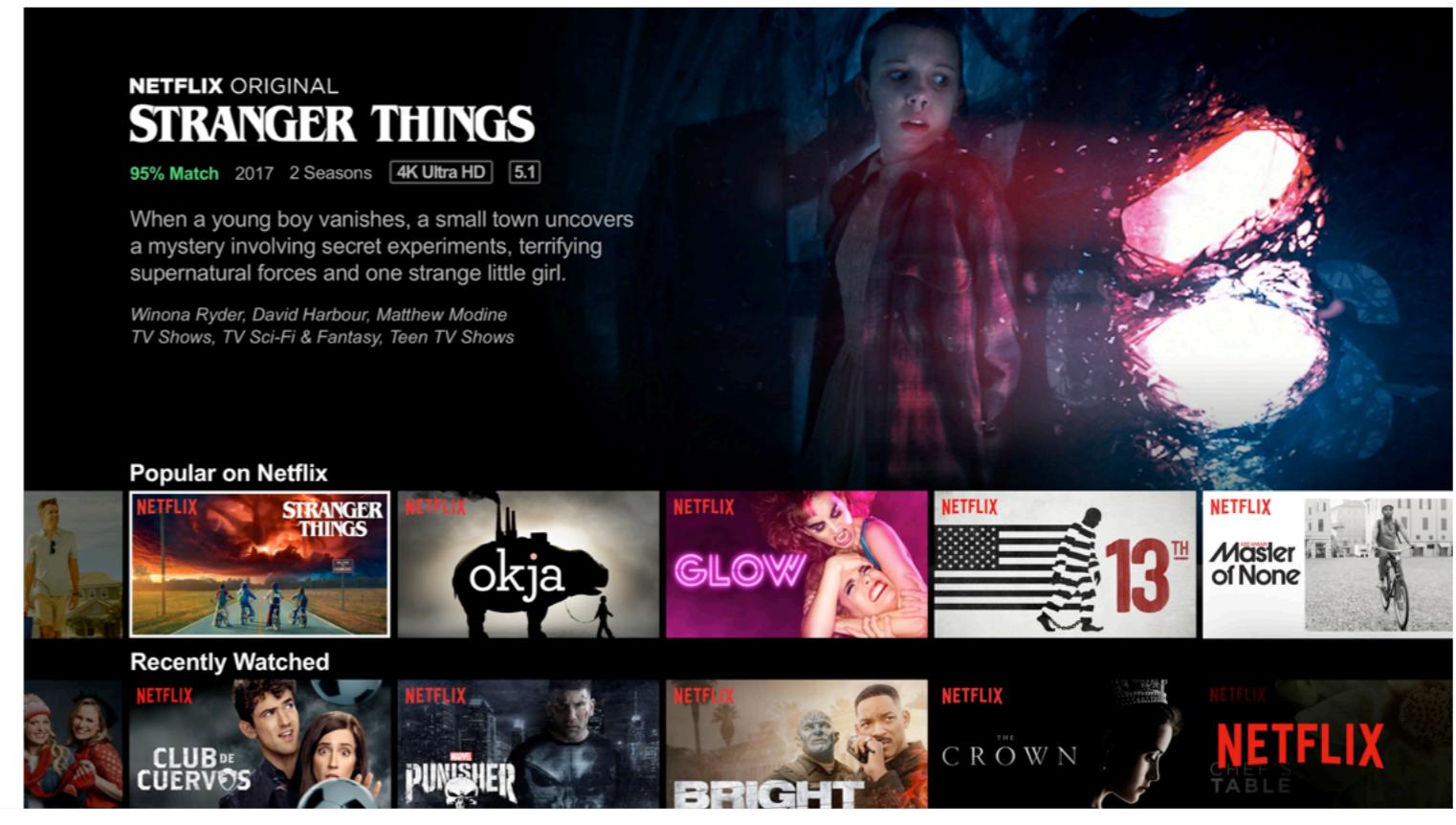
### Ajout dans le panier



### Recommendations



### Plateforme de vidéos à la demande (streaming)



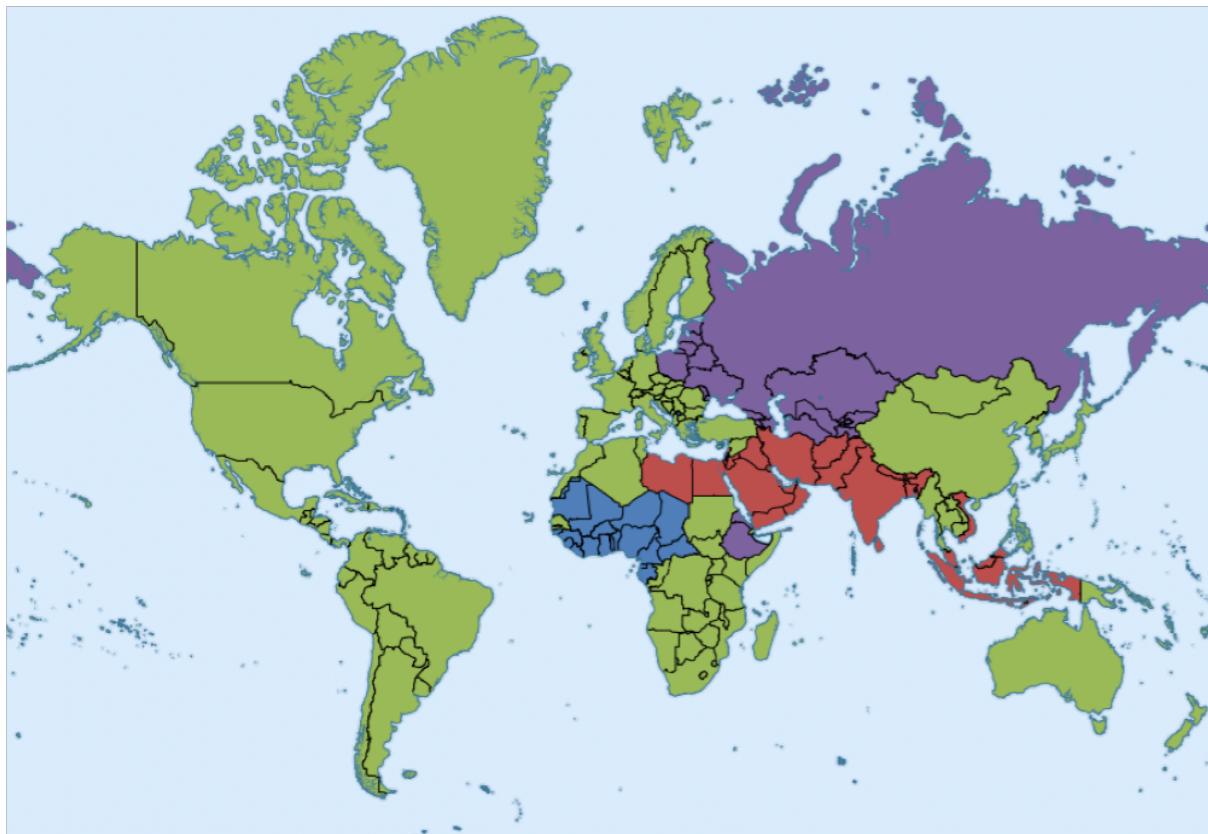
Applications: énormément d'utilisations dans le commerce en ligne

# Application: visualisation de données

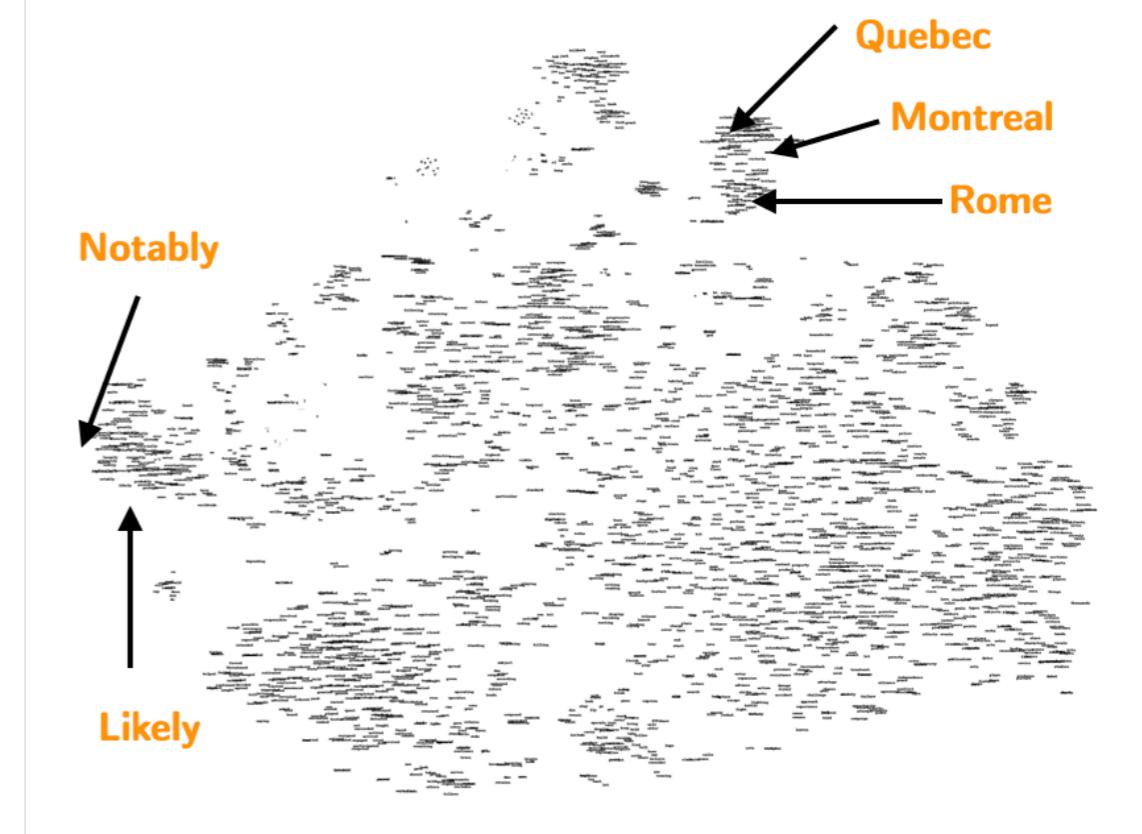
## Visualisation de données

**Objectif:** offrir une représentation visuelle des données à des fins d'analyse par un humain

### Regroupement des pays en fonction de leurs migrations



### Réduction de dimensions



Version lisible: <https://lvdmaaten.github.io/tsne/> (Words in 2D)

Mise en évidence de migrations importantes inattendues

Cluster des pays ex-URSS avec l'Ethiopie

Réduction de données de 100 dimensions à 2

Les mots semblables sont projetés à proximité

The World Migration Network: rankings, groups and gravity models [Cappart and Thonet, 2015]

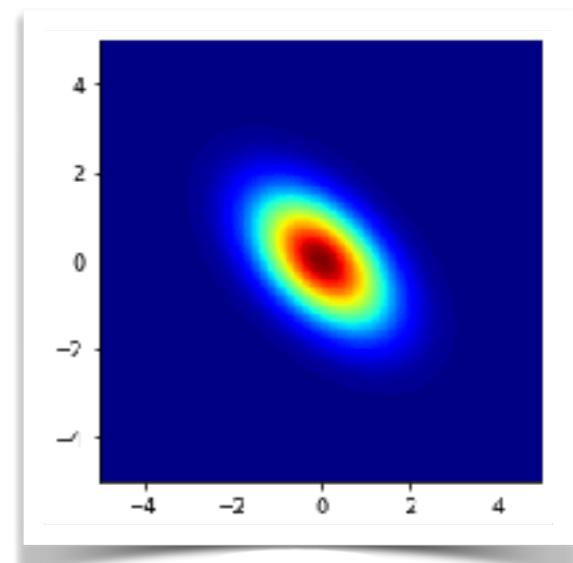
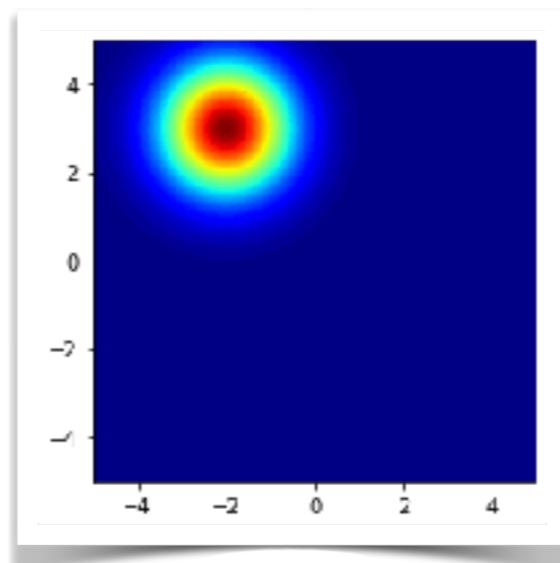
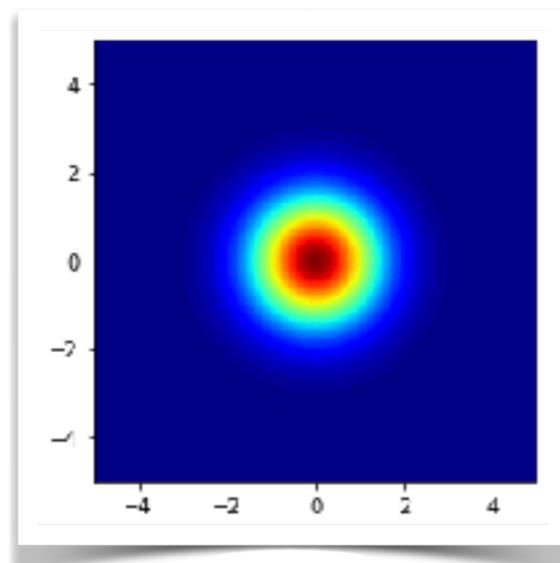
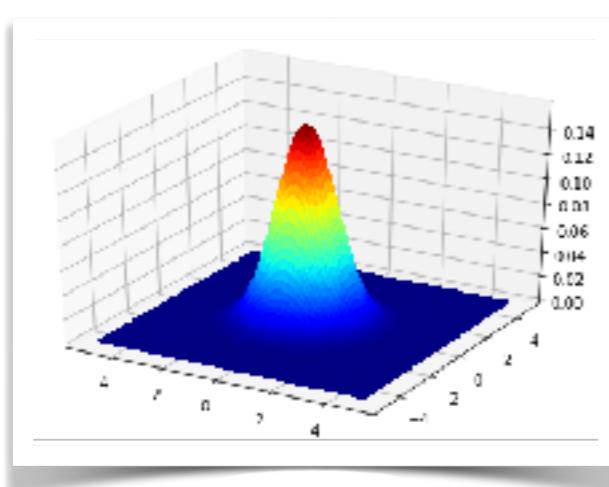
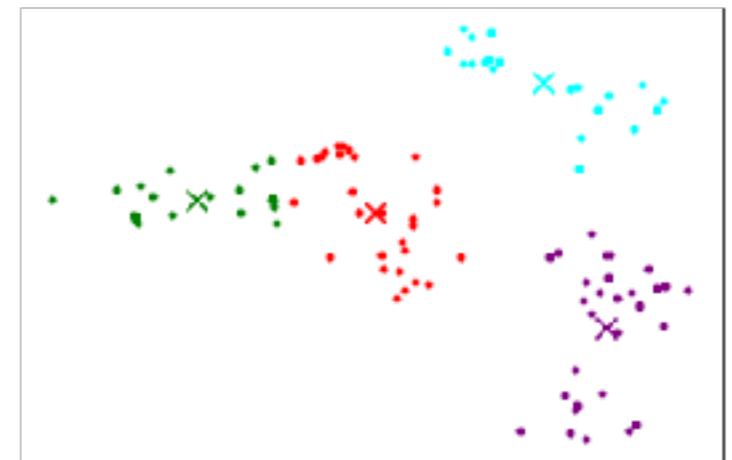
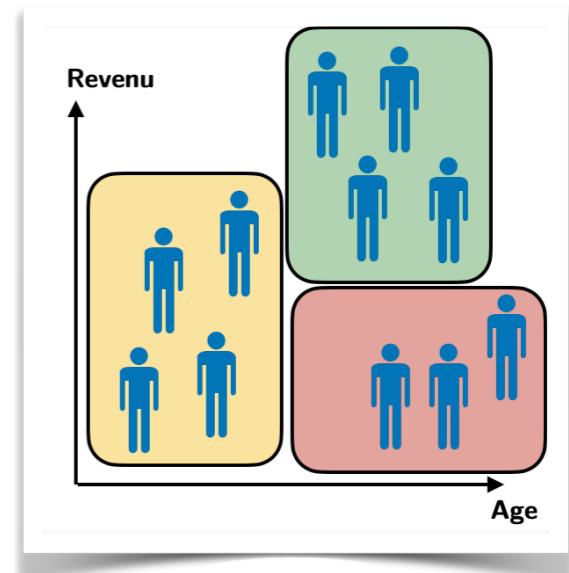
Visualizing Data using t-SNE [van der Maaten and Hinton, 2008]

Quentin Cappart

# Table des matières

## Apprentissage non-supervisé

- ✓ 1. Définition et applications de l'apprentissage non-supervisé
- 2. Formalisation du problème de regroupement (*clustering*)
- 3. Algorithme *K-means*
- 4. Formalisation du problème de détection d'anomalies
- 5. Notion de statistique: maximum de vraisemblance
- 6. Modélisation par des distributions normales univariées
- 7. Modélisation par une distribution normale multivariée



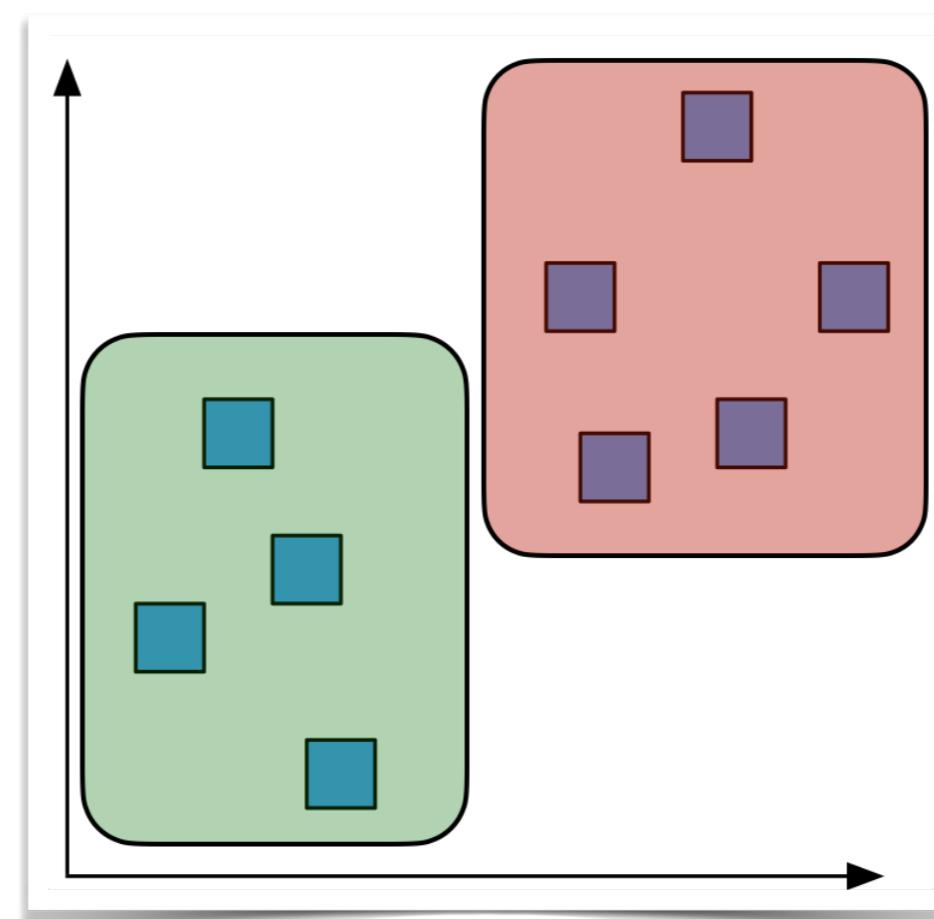
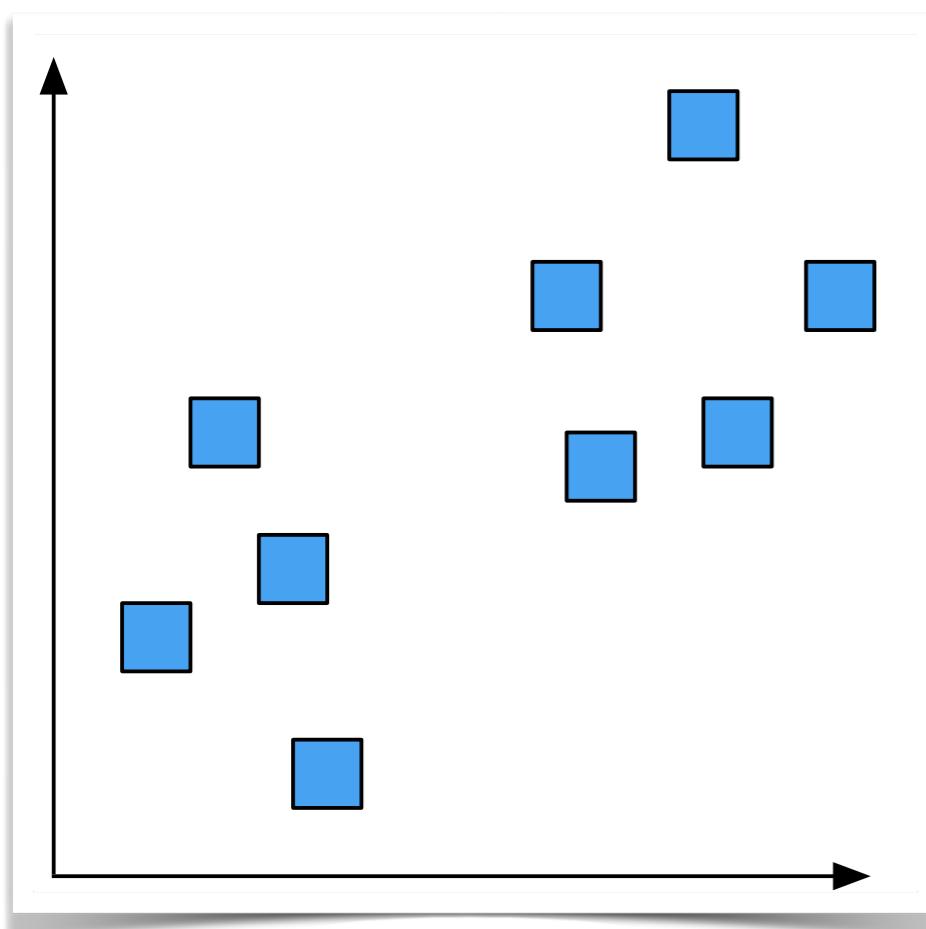
# Problème du regroupement (clustering)



## Problème du regroupement (*clustering*)

Problème consistant à regrouper les données similaires en groupes (clusters) cohérents

Note: il s'agit d'une des grandes familles d'application de l'apprentissage non-supervisé



**Construction du cluster:** besoin de définir un algorithme dédié

**Evaluation d'un clustering:** besoin de définir une fonction de coût

# Algorithme *K-means*

## Algorithme *K-means*

**Principe:** définir le centre de  $k$  groupes, et associer les données au groupe ayant le centre le plus proche

**Définition des centres:** de manière itérative, en 3 étapes successives à partir d'une valeur initiale

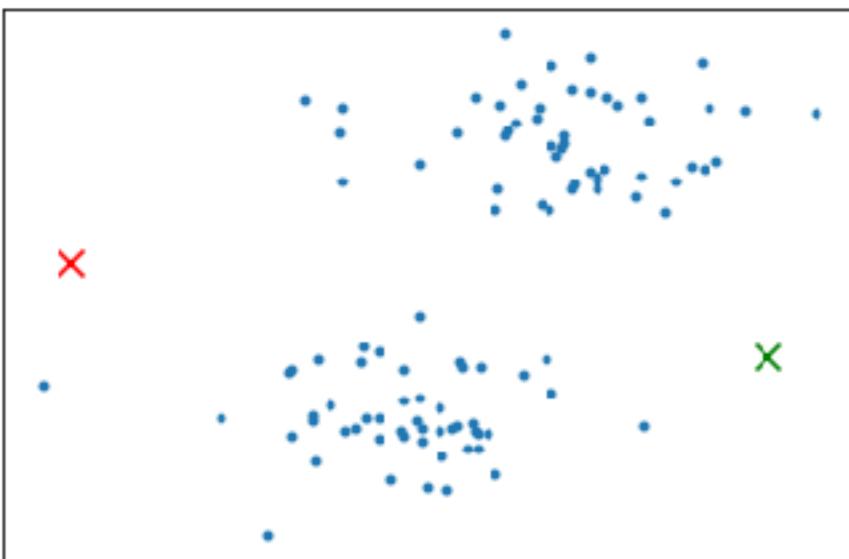
**Initialisation:** choisir  $k$  valeurs aléatoirement, correspondant au centre de  $k$  clusters

**Etape 1:** regrouper les données au centre (*centroïdes*) du cluster le plus proche

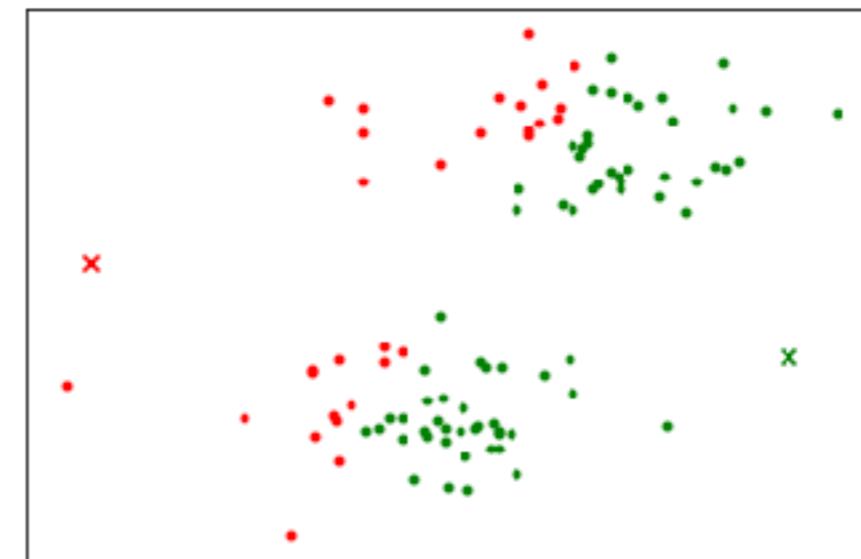
**Etape 2:** mettre à jour les centres des clusters, en fonction des données qui y ont été ajoutées/retirées

**Etape 3:** répéter les étapes (1) à (3) jusqu'à convergence

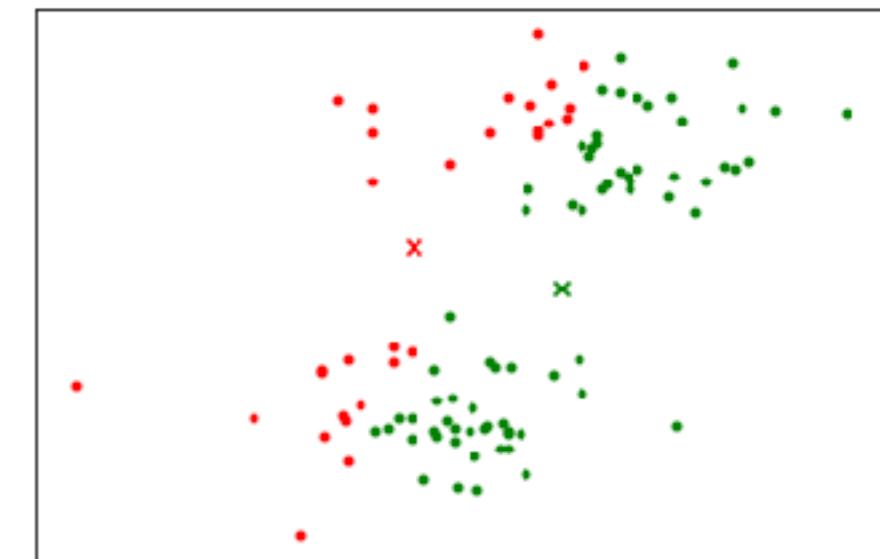
**Convergence:** l'itération n'a pas changé les centres des clusters



Etape d'initialisation



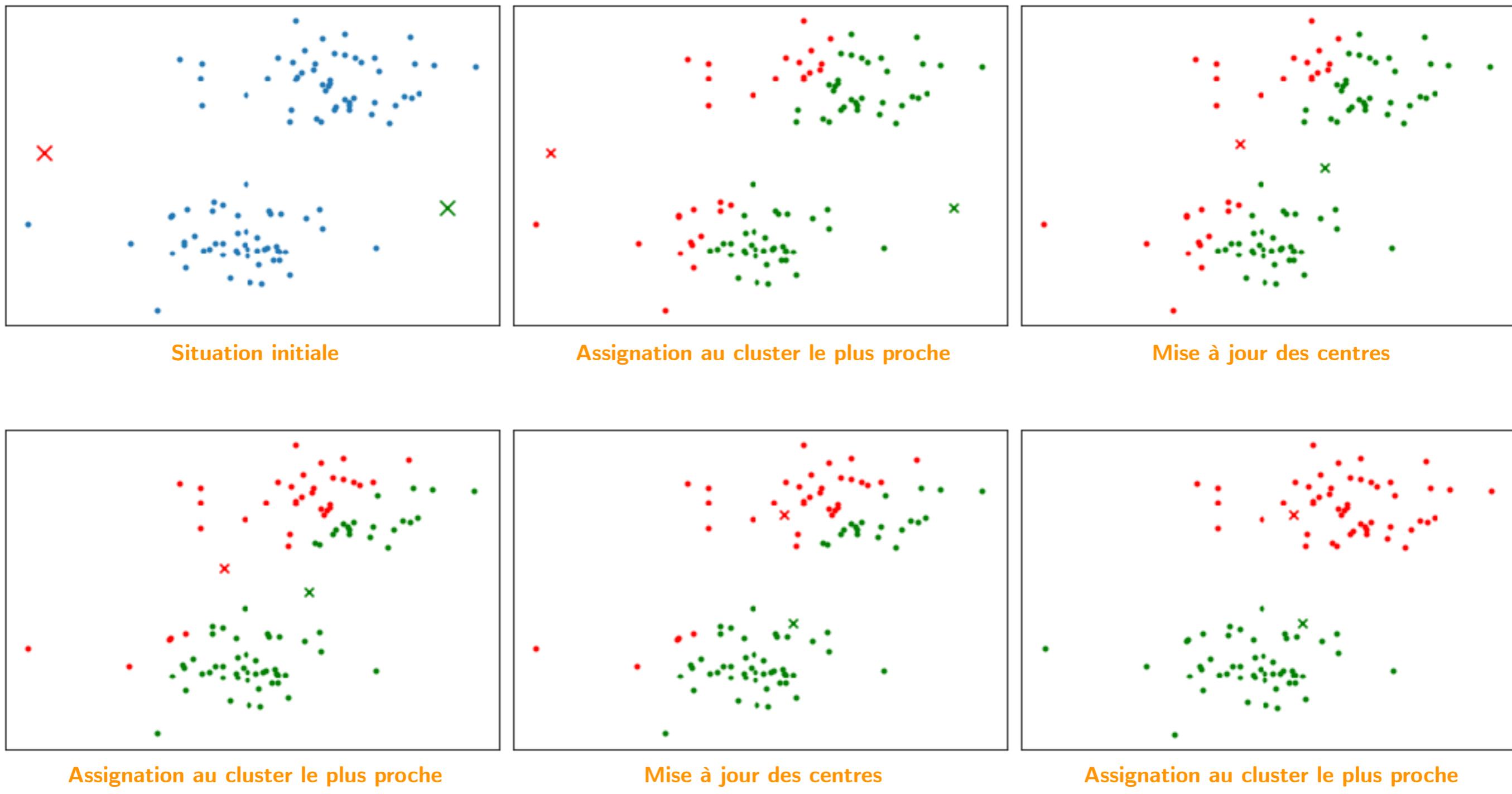
Etape 1: assignation au cluster le plus proche



Etape 2: mise à jour des centres

**Stratégie de regroupement:** les nouvelles données sont assignées au cluster ayant le centre le plus proche

# Algorithme $K$ -means - visualisation



Le processus est répété jusqu'à ce que les centres ne bougent plus (convergence)

# Algorithme K-means - Formalisation

## Notations

$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  : les données d'entraînement

$k$  : le nombre de clusters

$x^{(i)} \in \mathbb{R}^n$  : dimension d'une donnée (nombre de caractéristiques)

$\mu_j$  : le centre du cluster  $j : \forall j \in \{1, \dots, k\}$

$c^{(i)}$  : indice du cluster où la donnée  $i$  a été assignée

$\mu_j \in \mathbb{R}^n$  : dimension du centre du cluster

$\mu_{c^{(i)}}$  : centre du cluster où la donnée  $i$  a été assignée

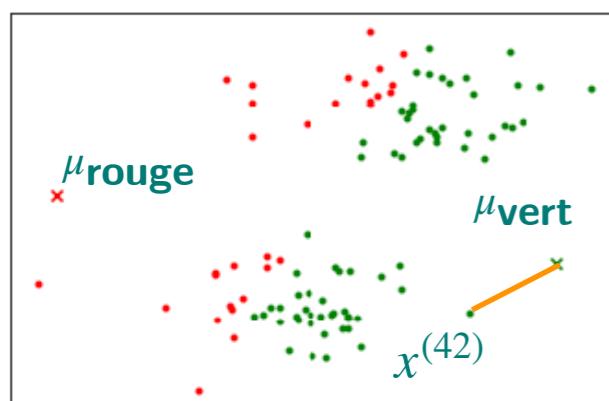
**Objectif:** minimiser la distance moyenne entre les données et le centre du cluster associé

 **Fonction de coût d'un regroupement**

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_j) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|_2$$

avec  $\|x^{(i)} - \mu_{c^{(i)}}\|_2 = \sqrt{(x_1^{(i)} - \mu_{1,c^{(i)}})^2 + \dots + (x_n^{(i)} - \mu_{n,c^{(i)}})^2}$  (pour  $n$  caractéristiques)

**Intuition:** cela correspond à la moyenne des distances euclidiennes entre chaque donnée et le centre associé



**Visualisation pour des données ayant deux caractéristiques numériques (réelles)**

$k = 2$  (cluster rouge et vert)

$c^{(42)} = \text{vert}$

$$\|x^{(i)} - \mu_{c^{(i)}}\|_2 = \|x^{(42)} - \mu_{\text{vert}}\|_2$$

# Algorithme *K-means*

KMeans( $D, k$ ) :

$\mu_j = \text{initializeRandomly}() \quad \forall j \in \{1, \dots, k\}$

repeat until convergence :

$\forall i \in \{1, \dots, m\}$  :

$$c^{(i)} = \operatorname{argmin}_j \|x^{(i)} - \mu_j\|_2$$

$\forall j \in \{1, \dots, k\}$  :

$$P = \left\{ x^{(i)} \in D \mid c^{(i)} = j \right\}$$

$\mu_j = \text{computeMeanValue}(P)$

return  $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k$

On donne en entrée le jeu de données et le nombre de clusters souhaités

On place les centres initiaux aléatoirement

**Etape 1:** on assigne les données au cluster le plus proche

On ne considère que les données associées au cluster actuel dans l'itération

**Etape 2:** on calcule le nouveau centre du cluster, étant donné les points liés

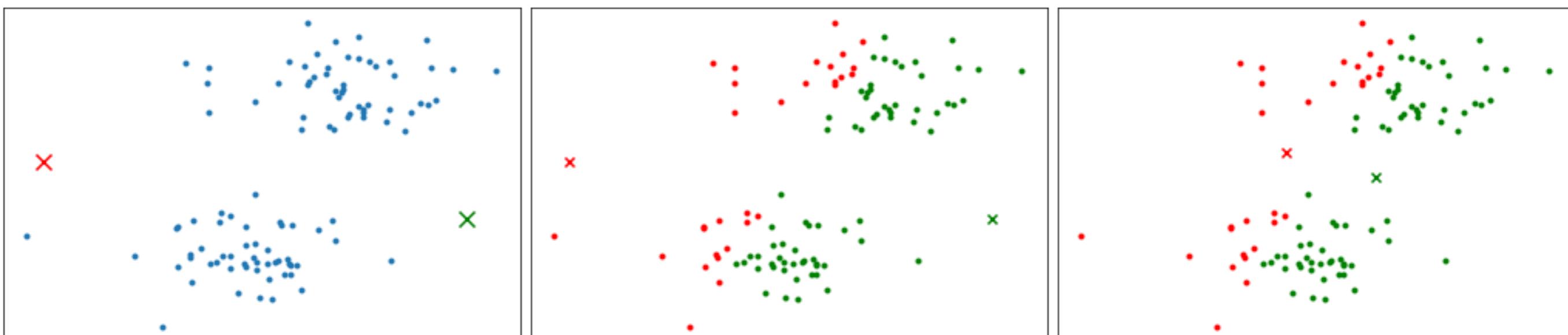
On retourne les assignations de chaque donnée, et les centres des clusters

**Principe:** chaque itération va rapprocher les centres d'un groupe de données et ainsi réduire le coût actuel

Situation initiale

Assignation au cluster le plus proche

Mise à jour des centroïdes



# Analyse de l'algorithme

KMeans( $D, k$ ) :

$\mu_j = \text{initializeRandomly}() \quad \forall j \in \{1, \dots, k\}$

repeat until convergence :

$\forall i \in \{1, \dots, m\}$  :

$$c^{(i)} = \underset{j}{\operatorname{argmin}} \|x^{(i)} - \mu_j\|_2$$

$\forall j \in \{1, \dots, k\}$  :

$$P = \left\{ x^{(i)} \in D \mid c^{(i)} = j \right\}$$

$\mu_j = \text{computeMeanValue}(P)$

return  $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k$

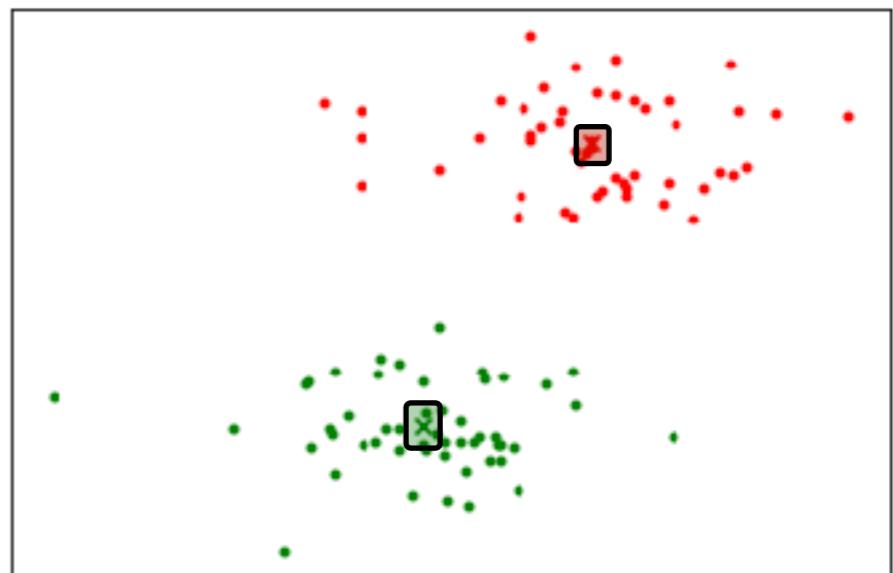


Est-ce que cet algorithme va converger ?

Bonne nouvelle: chaque étape rapproche les centres des données

Conséquence: chaque étape réduit ainsi le coût  $J$

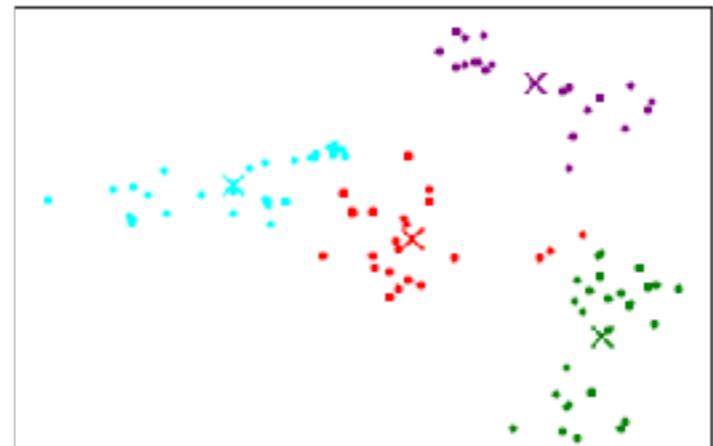
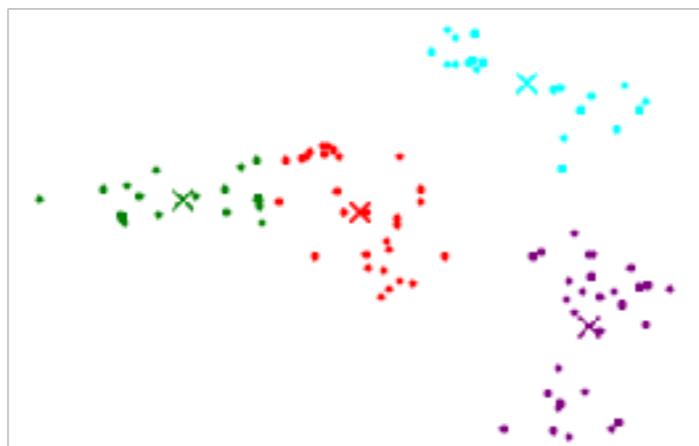
Convergence: assurée car le coût est borné (jamais négatif)



Est-ce que la solution obtenue est optimale ?

Mauvaise nouvelle: on risque de tomber dans un minimum local

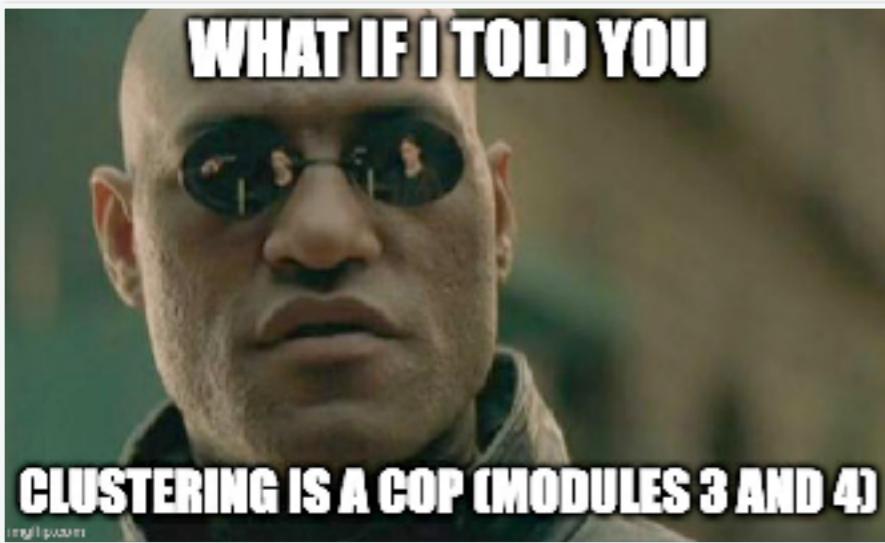
Augmentation du risque: lorsqu'on augmente le nombre de clusters ou le nombre de caractéristiques



# Amélioration de l'algorithme



Avez-vous une idée pour pallier cette difficulté ?



**Cadre général:** minimiser cette fonction de coût est NP-difficile

**Module 3:** déterminer les meilleurs clusters est un COP

**Modules 3 et 4:** méthodes pour résoudre ce genre de problèmes

**Programmation par contraintes:** difficile à mettre en oeuvre

**Difficulté:** le problème implique énormément de variables



**Algorithme *K-means*:** algorithme de recherche locale

**Initialisation:** points de départ de la recherche

**Etapes 1 et 2:** mouvements locaux pour minimiser le coût

**Stratégie actuelle:** de type *Hill climbing* sans diversification

**Difficulté:** forte chance de rester coincé dans un minimum local

**Bonne nouvelle:** on a vu plusieurs méthodes pour mitiger ce problème

**Exemples:** redémarrages, initialisation intelligente, ou utilisation de métahéuristiques

# K-means avec des redémarrages

## Méthode de redémarrages

**Observation:** la solution où l'algorithme converge dépend des positions initiales des clusters

**Amélioration simple:** redémarrer plusieurs fois l'algorithme et prendre la meilleure solution trouvée

```
KMeansWithRestart( $D, k, \Gamma$ ) :  
     $s^* = \perp$   
    for  $i \in 1$  to  $\Gamma$  :  
         $s = \text{KMeans}(D, k)$   
        if  $J(s) < J(s^*)$  :  
             $s^* = s$   
    return  $s^*$ 
```

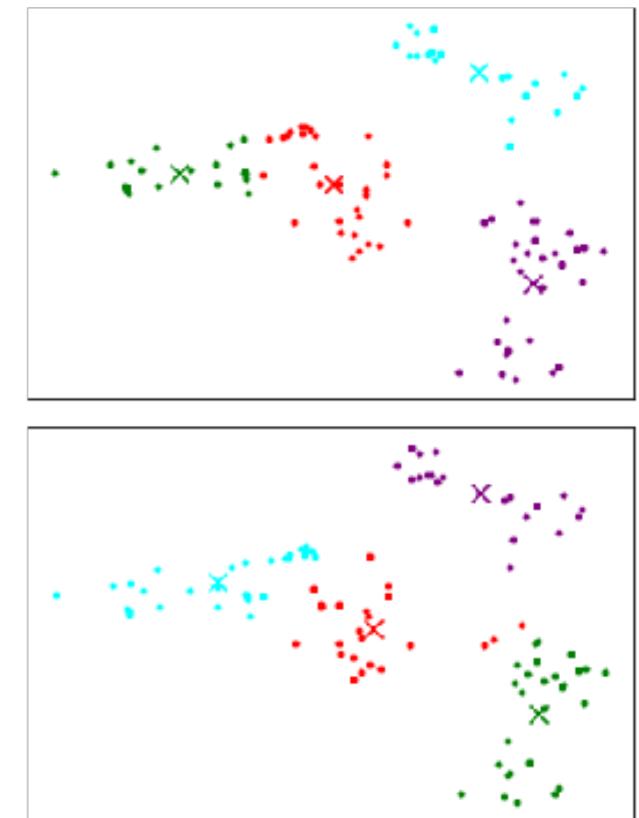
**KMeans( $D, k$ ) :**

$$\mu_j = \text{initializeRandomly}() \quad \forall j \in \{1, \dots, k\}$$

repeat until convergence :

$$\forall i \in \{1, \dots, m\} :$$
$$c^{(i)} = \underset{j}{\operatorname{argmin}} \|x^{(i)} - \mu_j\|_2$$
$$\forall j \in \{1, \dots, k\} :$$
$$P = \left\{ x^{(i)} \in D \mid c^{(i)} = j \right\}$$
$$\mu_j = \text{computeMeanValue}(P)$$

return  $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k$



**Avantage:** très facile à mettre en oeuvre

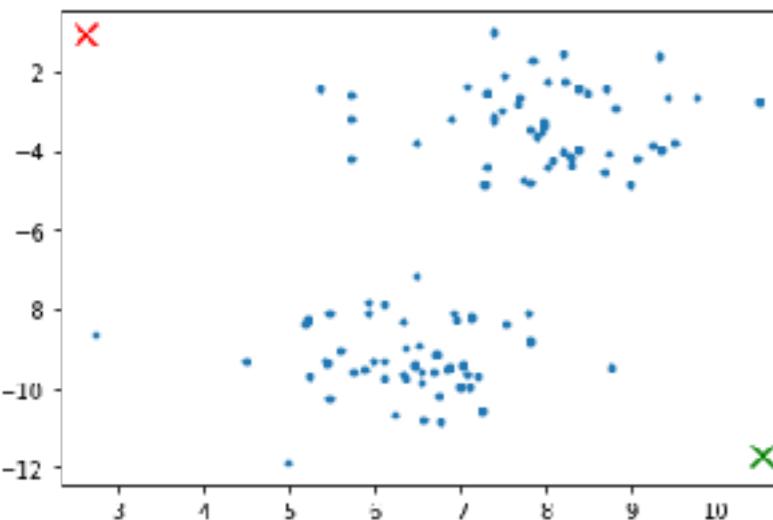
**Note:** montre les connexions entre les différentes modes de l'intelligence artificielle

**Autres améliorations:** utilisation de métaheuristiques (algorithmes génétiques)

# Algorithme *K-means*: initialisation des centres des clusters



Comment initialiser aléatoirement les centres ?

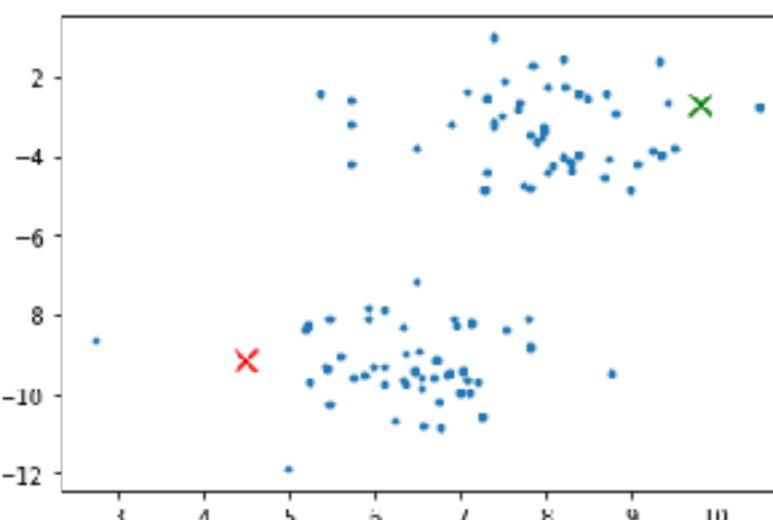


**Option 1:** Initialisation totalement aléatoire

**Principe:** les valeurs sont initialisées aléatoirement dans une zone délimitée

**Difficulté:** les centres peuvent se retrouver loin des données

**Conclusion:** convergence plus lente et méthode peu utilisée



**Option 2:** initialisation aléatoire sur des données

**Principe:** les valeurs initiales sont prises aléatoirement sur des données

**Avantage:** méthode simple avec de bons résultats en pratique

**Difficulté:** certains placements sont moins pertinents que d'autres

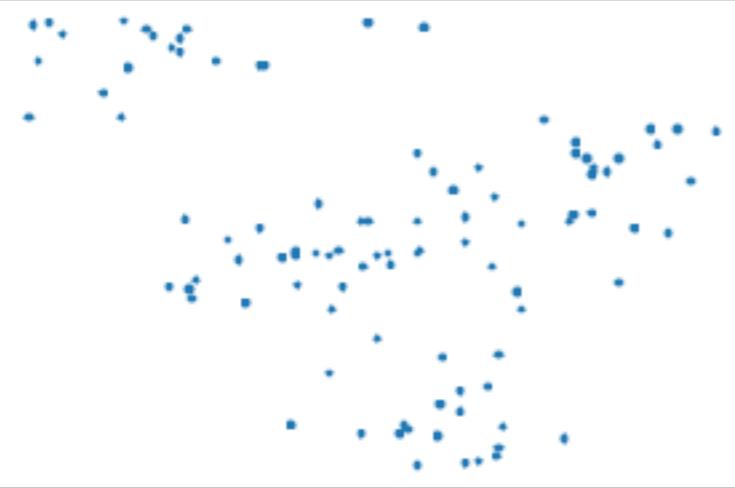
**Exemple:** 2 centres initialisés très proches l'un de l'autre

**Option 3:** initialisation aléatoire contrôlée sur les données

**Principe:** choisir des centres initialement bien distants des autres

**Algorithme *k-means++*:** algorithme implémentant ce principe

# Choix du nombre de clusters

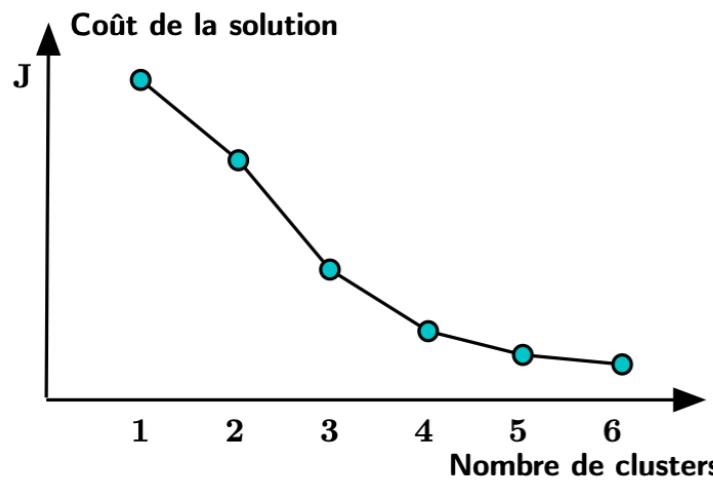


? Combien de clusters doit-on définir pour ces données ?

**Difficulté de la tâche:** choix pas toujours clair, et qui peut être subjectif

**Alternatives:** plusieurs critères de sélection sont possibles

**Option 1:** décision itérative



**Observation:** augmenter le nombre de cluster va réduire le coût

**Principe:** augmenter ce nombre jusqu'à ce que la réduction soit minime

**Exemple de gauche:** prendre 4 ou 5 cluster semble approprié

**Option 2:** décision en fonction de l'objectif subséquent

**Principe:** le choix est fait sur base de l'objectif de la tâche de clustering

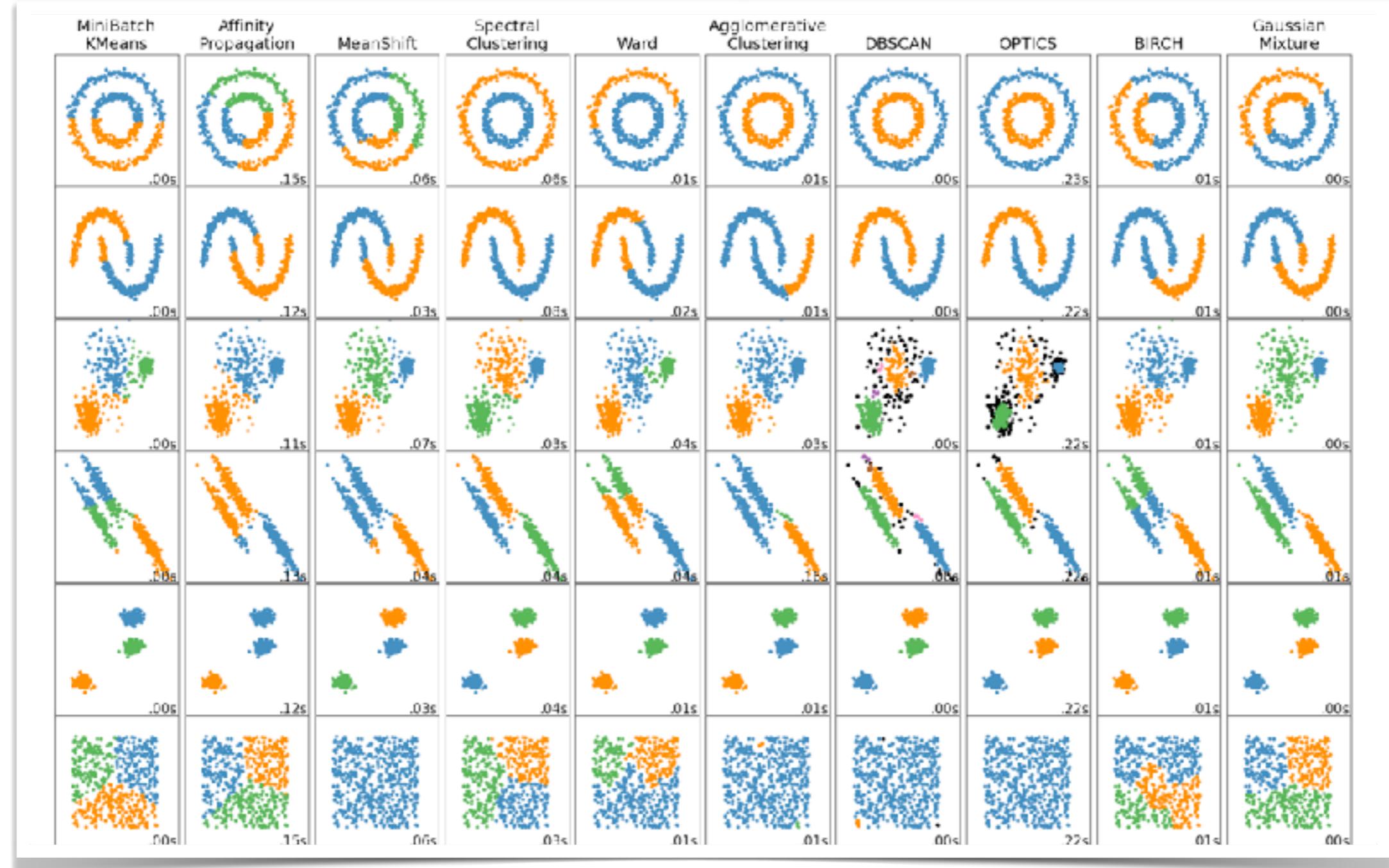
**Exemple:** on souhaite que le marché soit divisé en exactement 3 segments

**Option 3:** utilisation d'un autre algorithme de regroupement

**Affinity propagation:** le choix de ce nombre est intégré dans l'algorithme



# Tâche de regroupement: autres alternatives



Il existe énormément de variantes et d'autres algorithmes pour réaliser une tâche de regroupement

Algorithme *K-modes*: dédié aux données avec des catégories en caractéristiques

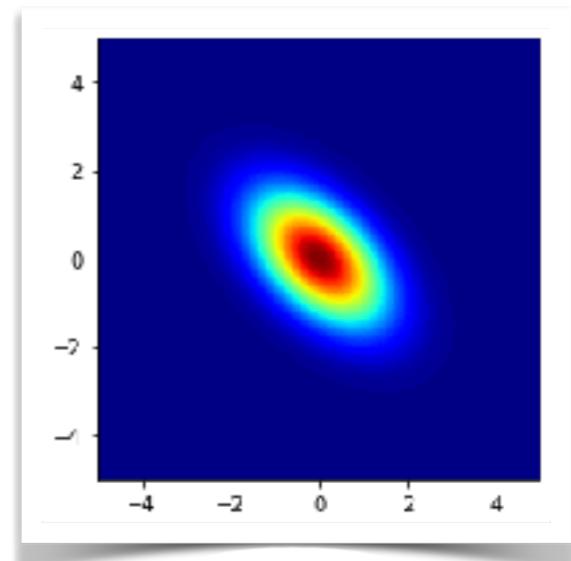
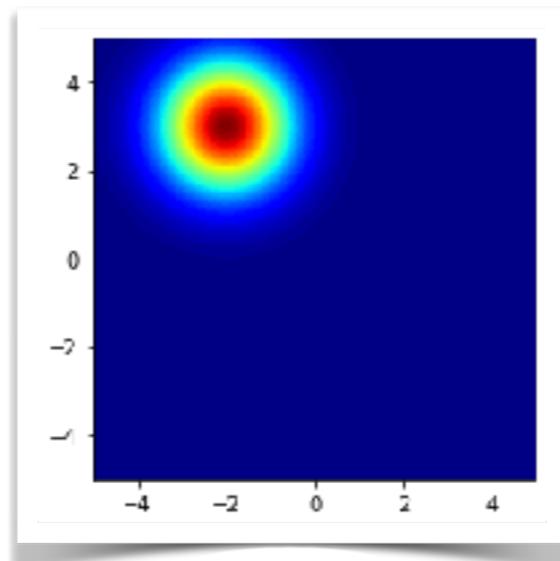
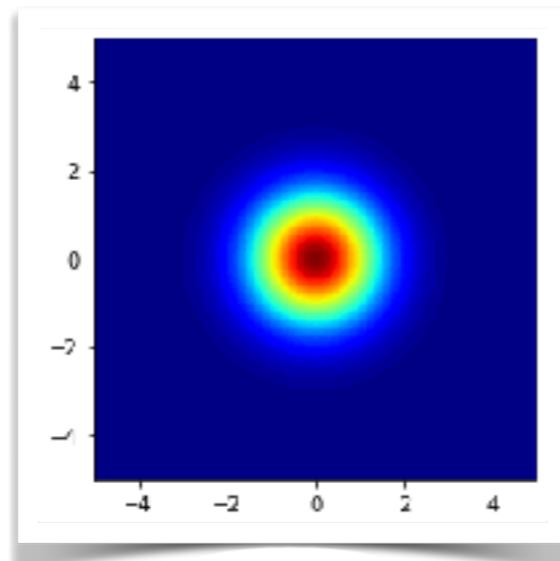
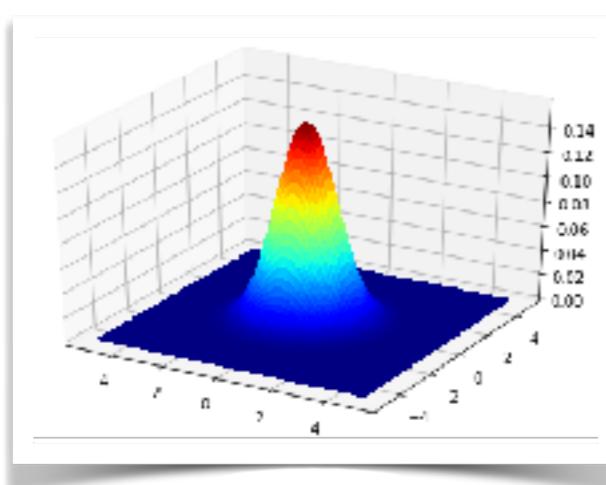
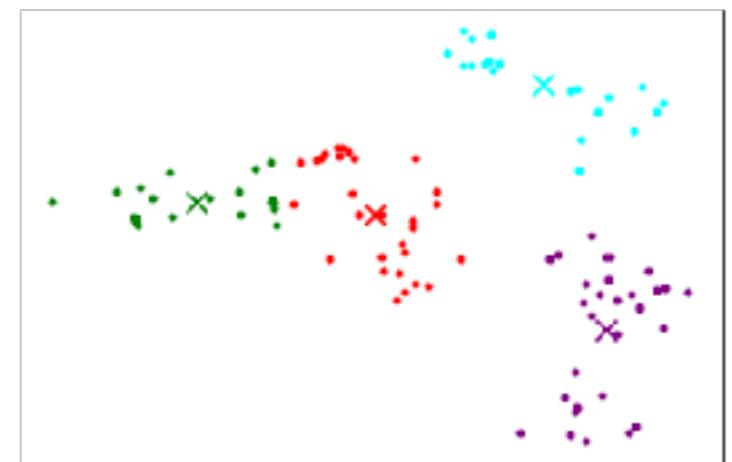
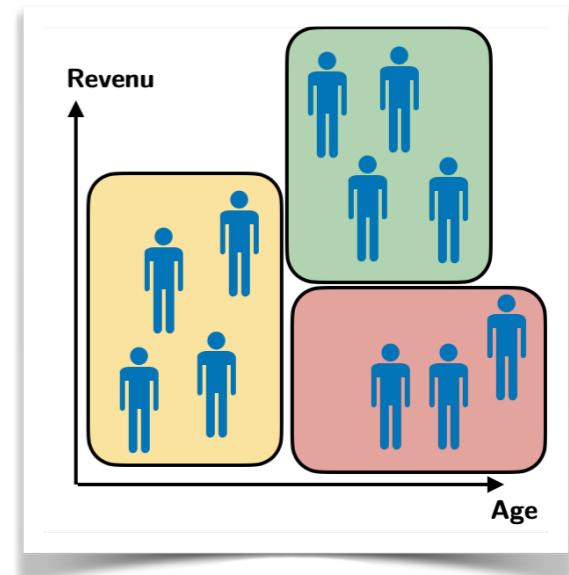
Algorithme *K-prototypes*: dédié aux données avec caractéristiques quelconques



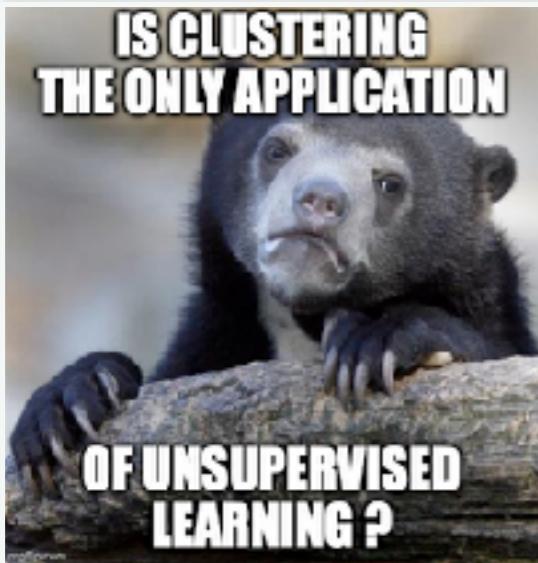
# Table des matières

## Apprentissage non-supervisé

- ✓ 1. Définition et applications de l'apprentissage non-supervisé
- ✓ 2. Formalisation du problème de regroupement (*clustering*)
- ✓ 3. Algorithme *K-means*
- 4. Formalisation du problème de détection d'anomalies
- 5. Notion de statistique: maximum de vraisemblance
- 6. Modélisation par des distributions normales univariées
- 7. Modélisation par une distribution normale multivariée



# Détection d'anomalies



 Détection d'anomalies (*anomaly detection*)

Problème consistant à repérer les données hors-normes par rapport à ensemble de données dans la norme

**Objectif:** prédire la probabilité qu'une nouvelle donnée soit habituelle

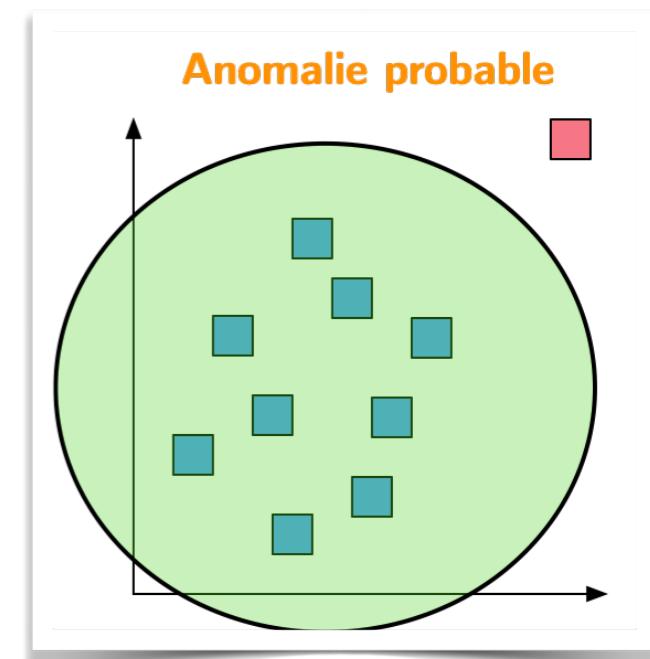
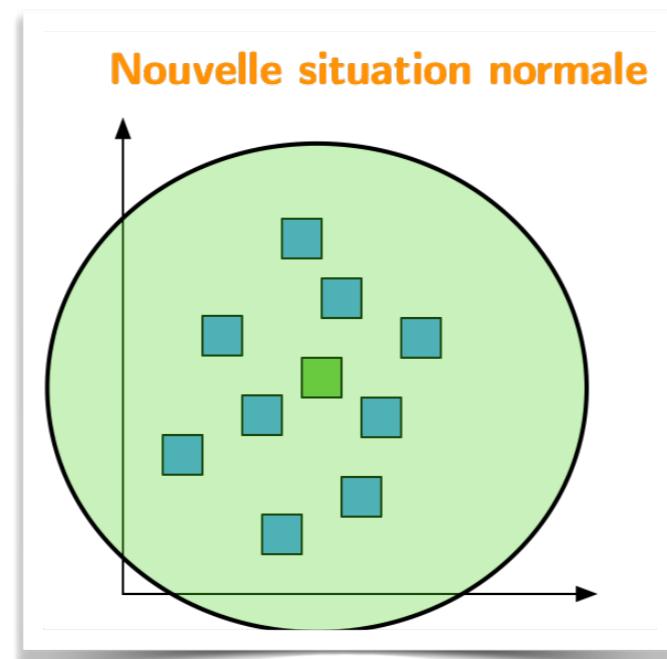
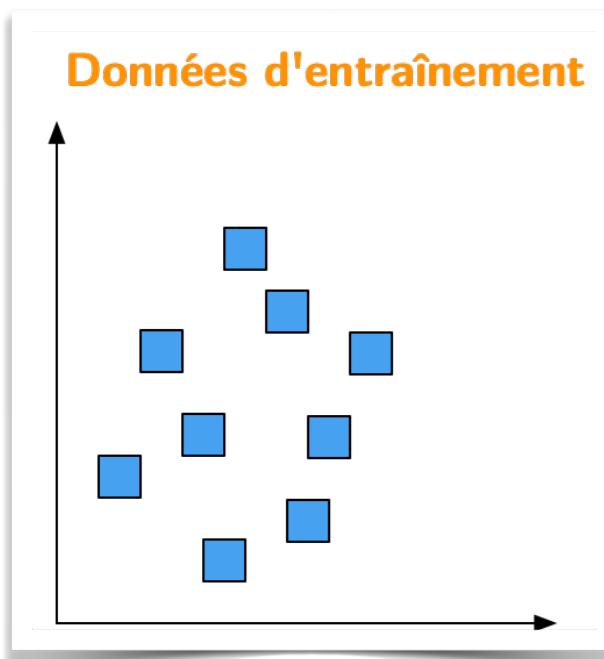
**Décision:** si la probabilité est inférieure à un seuil, alors c'est **une anomalie**

## Principe de la détection d'anomalies

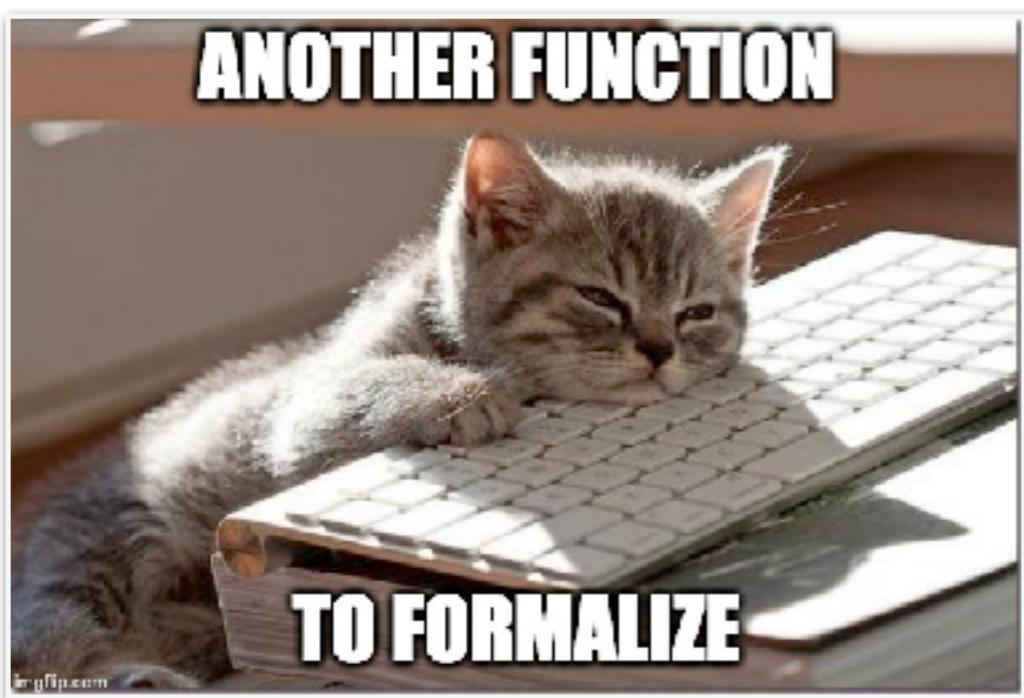
**Etape 1:** collecter un ensemble de données d'utilisation normale d'un système

**Etape 2:** construire un modèle reflétant une utilisation normale du système, sur base de ces données

**Etape 3:** utiliser ce modèle afin de prédire si une nouvelle utilisation est normale ou suspecte

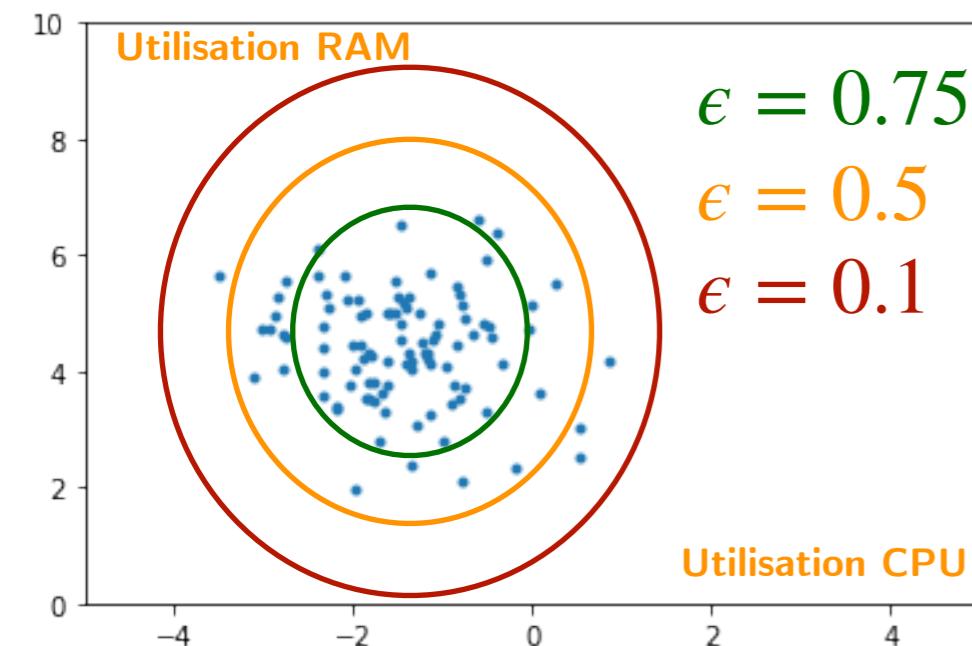


# Détection d'anomalies - Formalisation



$\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  : les données d'entraînement  
 $x^{(i)} \in \mathbb{R}^n$  : dimension de chaque donnée (nombre de caractéristiques)  
 $x^{(\text{test})}$  : donnée de test  
 $\hat{y} = f(x)$  :  $\mathbb{P}(x \text{ est une utilisation normale du système})$   
 $\epsilon \in [0,1]$  : seuil de décision pour la considération d'une anomalie  
 $\hat{y} \geq \epsilon$  : la donnée  $x$  est dans la norme  
 $\hat{y} < \epsilon$  : la donnée  $x$  est une anomalie

L'objectif est d'apprendre une fonction de prédiction



Exemple: ensemble de données à deux caractéristiques

En dessous du seuil: considéré comme une anomalie

Le gestionnaire pourra ensuite analyser cette utilisation en détail

?

Comment créer une fonction pour cette tâche ?

Cela va passer par la définition d'une hypothèse sur la fonction

Hypothèse que l'on va faire: chaque caractéristique est régie par une loi normale qui lui est propre

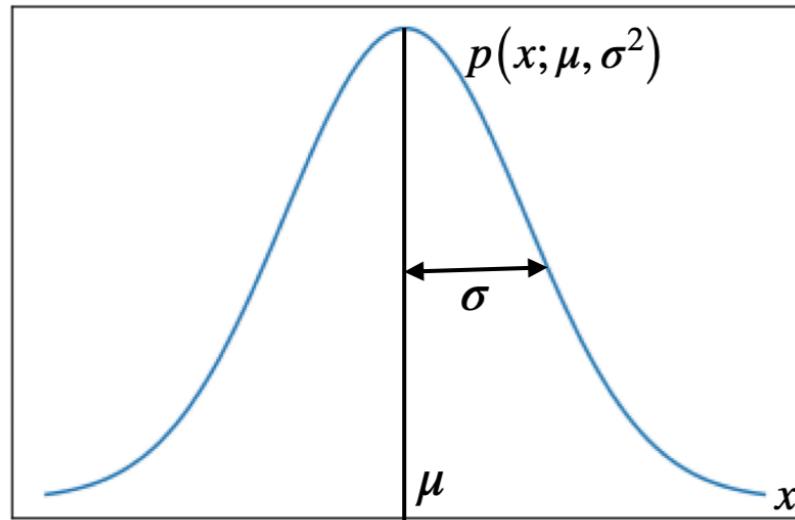
# Distribution normale (gausienne)

**Loi normale:**  $p(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

$\mu$  : espérance (paramètre de la loi)

$\sigma^2$  : variance (paramètre de la loi)

$x \sim \mathcal{N}(\mu, \sigma^2)$  :  $x$  prend une valeur tirée d'une distribution normale de paramètres  $\mu, \sigma^2$



**Hypothèse:** chaque caractéristique est tirée d'une loi normale spécifique

$$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

$$x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

...

$$x_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$$

**Conséquence:** chaque caractéristique a ses propres paramètres, relatifs à une loi normale

**Idée:** les paramètres des ces lois normales sont appris à l'aide des données disponibles

**?** Combien de paramètres doit-on déterminer pour des données à  $n$  caractéristiques ?

**Loi normale:** comporte 2 paramètres (espérance, et variance)

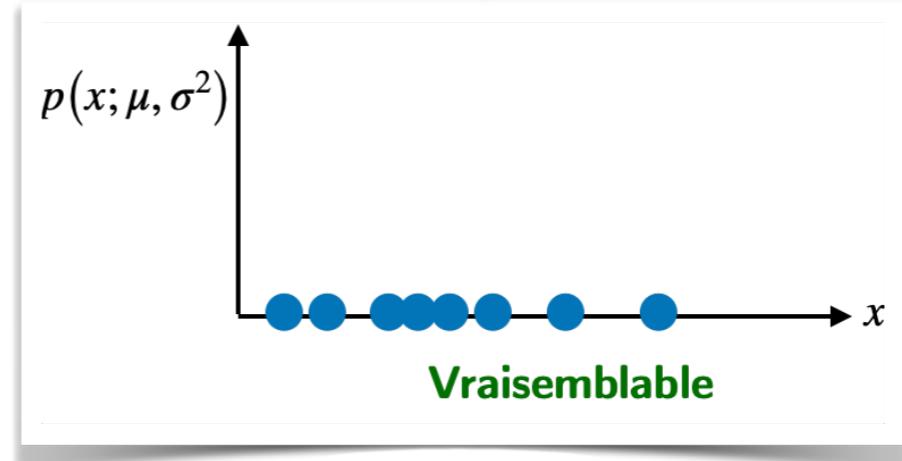
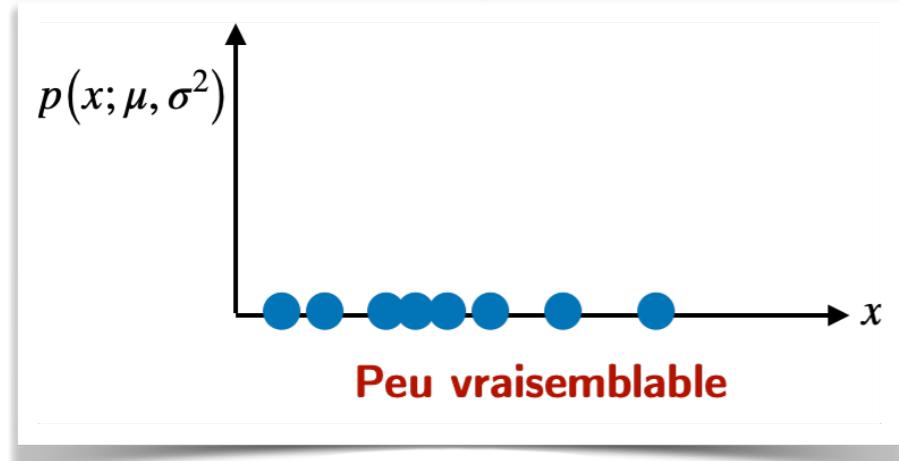
**Total:**  $2n$  paramètres car on a  $n$  lois normales dans notre modèle

# Fonction de vraisemblance



Comment déterminer ces paramètres ?

**Objectif:** obtenir la loi normale qui est la plus plausible (ou vraisemblable) par rapport à nos données



**Besoin:** formaliser mathématiquement cette notion de vraisemblance



**Fonction de vraisemblance (*likelihood function*)**

Mesure de la vraisemblance d'une distribution statistiques pour les données considérées

$$L(\mu, \sigma^2 | x^{(1)}, \dots, x^{(m)}) = p(x^{(1)} | \mu, \sigma^2) \times \dots \times p(x^{(m)} | \mu, \sigma^2)$$

**Note:** cette fonction donne un score d'à quel point le modèle (la loi normale) *fitte* les données

**Principe du calcul:** on prend le produit des probabilités de chaque donnée étant donné la loi

**Condition:** suppose que les données sont indépendantes, et identiquement distribuées

**Attention:** confusion courante entre la notion de **probabilité** et de **vraisemblance**

**Probabilité:** plausibilité d'obtenir un évènement aléatoire selon un certain modèle

**Vraisemblance:** plausibilité d'un modèle, étant donné l'observation de réalisations d'une variable aléatoire



# Estimation du maximum de vraisemblance



**Maximum de vraisemblance (maximum likelihood estimation - MLE)**

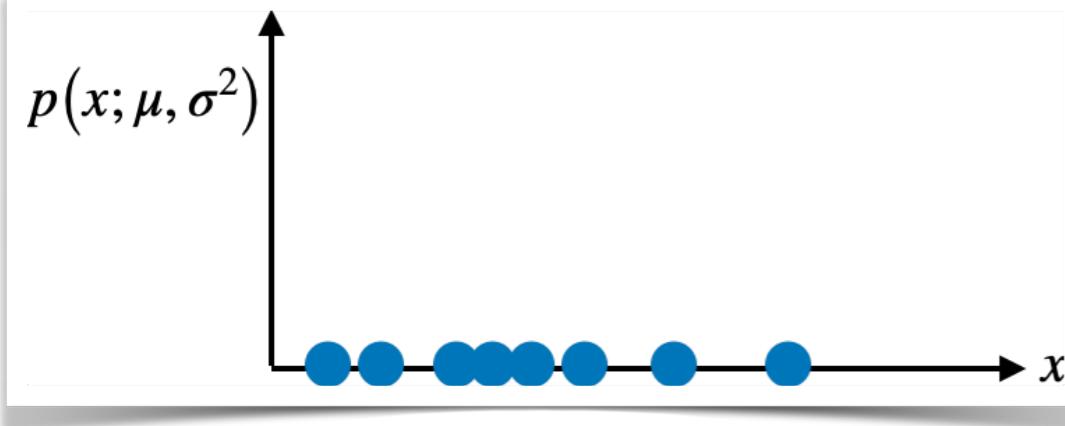
Problème consistant à trouver les paramètres d'une loi normale afin de maximiser la fonction de vraisemblance par rapport aux données observées

$$\max_{\mu, \sigma^2} L(\mu, \sigma^2 | x^{(1)}, \dots, x^{(m)})$$

**Intuition:** on souhaite construire la loi normale la plus plausible par rapport aux données

**Objectif:** trouver les valeurs des deux paramètres maximisant la fonction de vraisemblance

**Note:** cette définition peut également être étendue à des modèles qui ne sont pas des lois normales



?

Quelles sont ces valeurs ?

Bonne nouvelle: il s'agit d'un résultat statistique connu

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

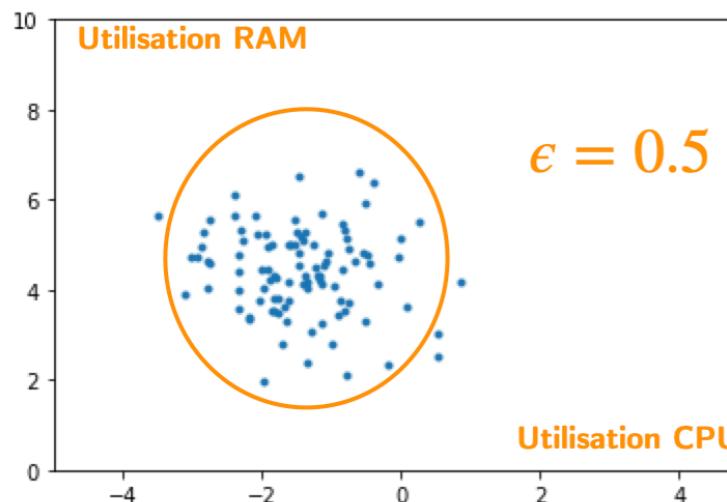
$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)^2$$

**Résultat:** ces paramètres donnent la loi normale la plus vraisemblable pour nos données

**Attention:** notez bien qu'on a une loi normale par caractéristique des données

**Note:** la définition des paramètres peut être beaucoup plus difficile avec d'autres modèles

# Estimations des paramètres pour la détection d'anomalies



**Hypothèse 1:** chaque caractéristique vient d'une loi normale spécifique

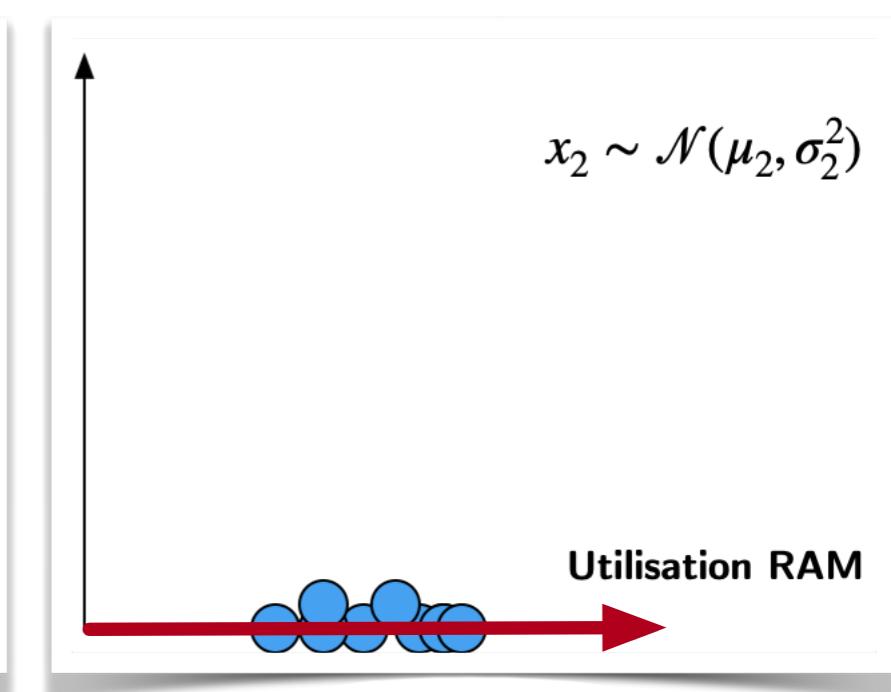
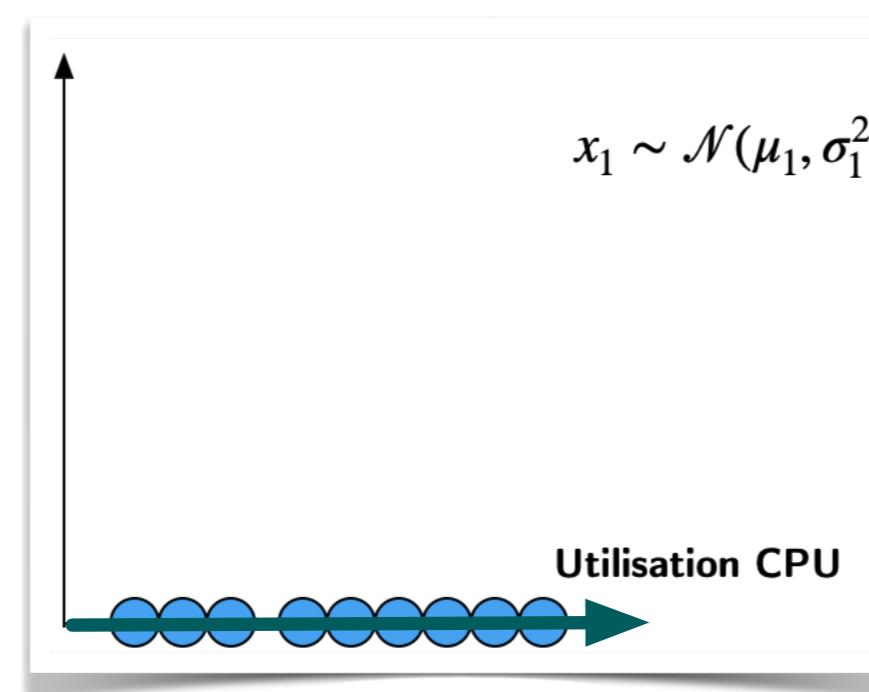
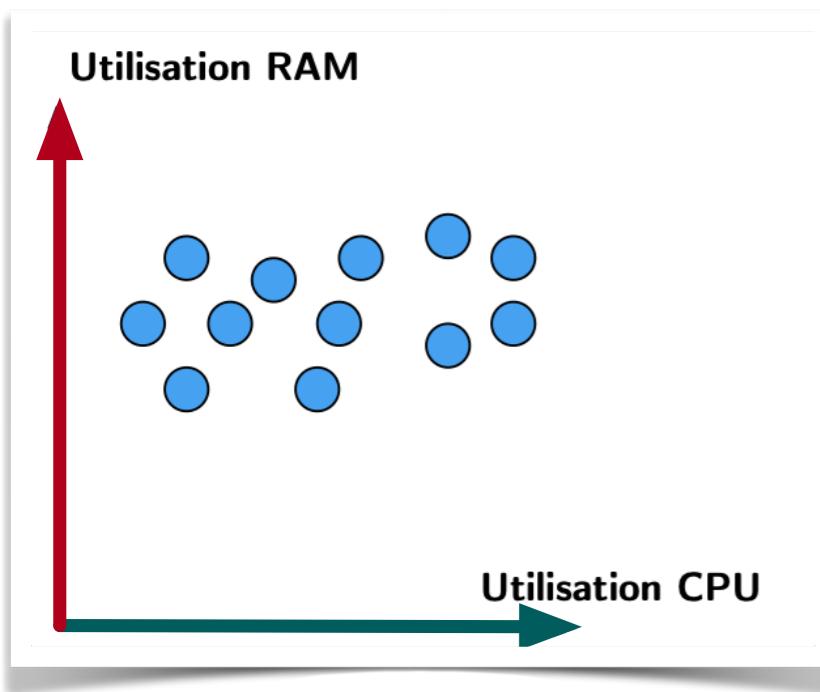
$$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

...

$$x_n \sim \mathcal{N}(\mu_n, \sigma_n^2)$$

**Hypothèse 2:** les caractéristiques sont considérées indépendantes des autres

**Objectif:** trouver les meilleurs paramètres relatifs à la distribution normale de chaque paramètre



**Modèle final:** probabilité jointe des distributions de chaque caractéristique (leur multiplication)

$$\hat{y} = p(x_1 | \mu_1, \sigma_1^2) \times \dots \times p(x_n | \mu_n, \sigma_n^2) = \prod_{j=1}^n p(x_j | \mu_j, \sigma_j^2)$$

Suppose l'indépendance entre chaque caractéristique

# Algorithme de détection d'anomalies

## Phase d'entraînement

**Objectif:** déterminer les paramètres des lois normales

AnomalyDetectionTraining( $D$ ) :

for  $j \in \{1, \dots, n\}$  :

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

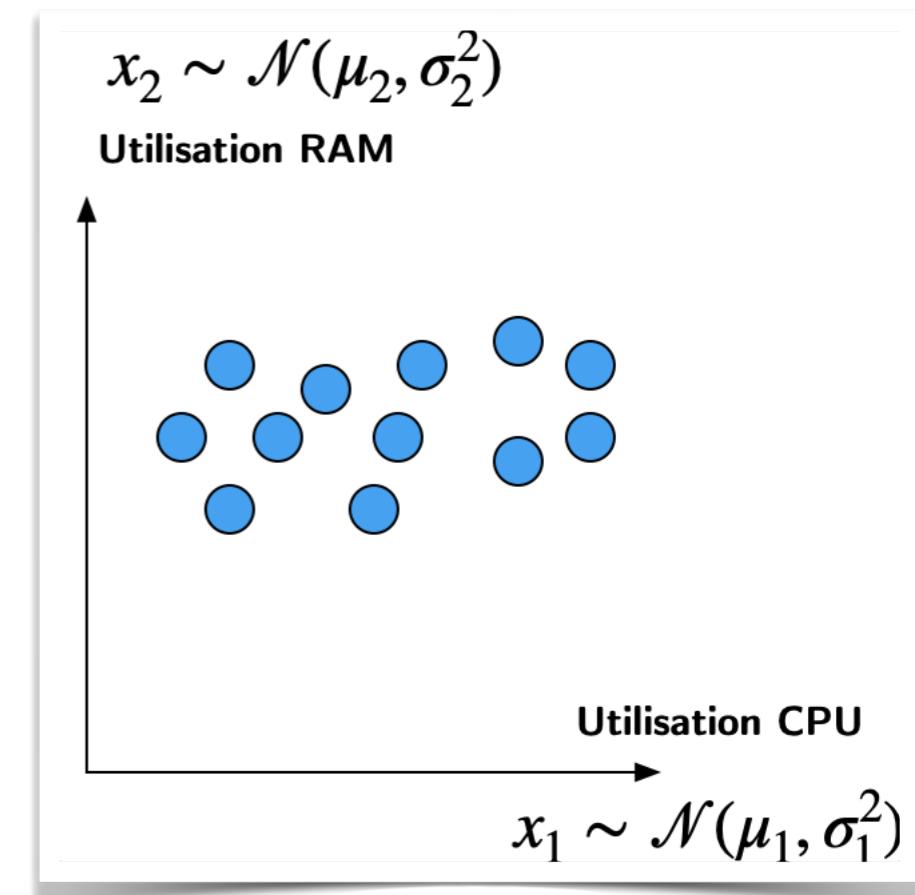
return  $\mu_1, \sigma_1^2, \dots, \mu_n, \sigma_n^2$

On utilise les données d'entraînement

Pour chaque caractéristique...

...On calcule les deux paramètres

On retourne tous les paramètres (c-à-d, le modèle)



## Phase d'évaluation

**Objectif:** calculer la probabilité qu'une nouvelle donnée soit régulière

AnomalyDetection( $x^{(\text{test})}, \epsilon, \mu_1, \sigma_1^2, \dots, \mu_n, \sigma_n^2$ ) :

$$\hat{y} = \prod_{j=1}^n p(x_j^{(\text{test})} | \mu_j, \sigma_j^2)$$

$$= \prod_{j=1}^n \frac{1}{\sigma_j \sqrt{2\pi}} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

if  $\hat{y} < \epsilon$  : return anomaly for  $x^{(\text{test})}$

else : return regular data for  $x^{(\text{test})}$

On utilise les lois normales déterminées précédemment

On calcule la probabilité jointe que la nouvelle donnée soit régulière

Notre modèle est le produit des lois normales de chaque caractéristique

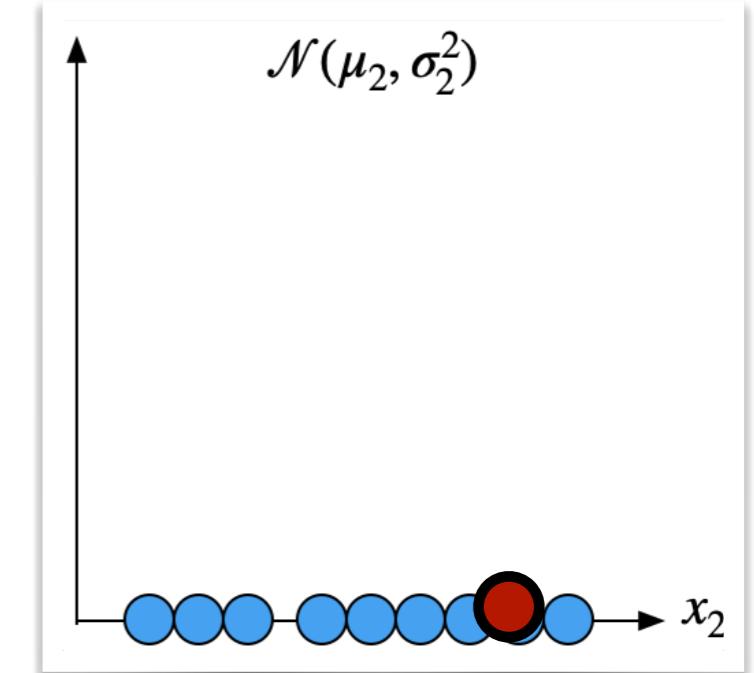
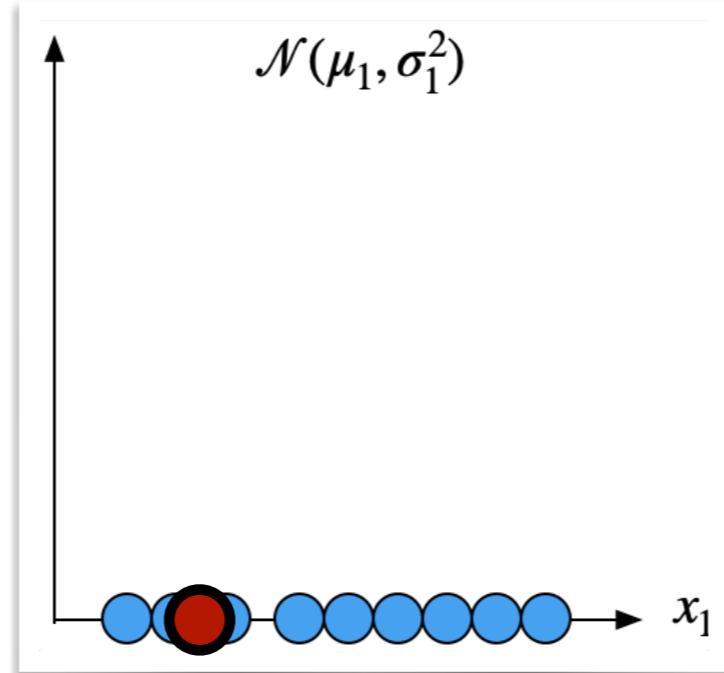
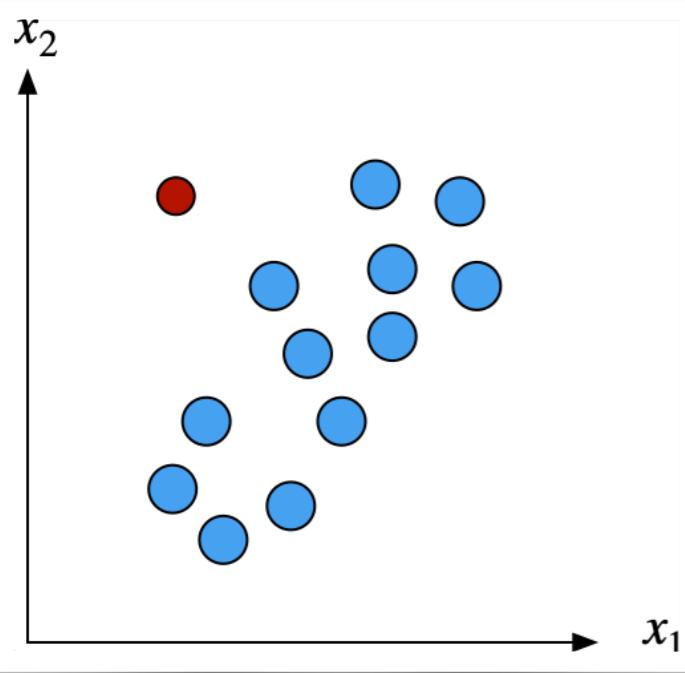
Si elle est en dessous du seuil, on la considère comme une anomalie

# Limitations de notre modèle



Quelles sont les limitations de notre modèle ?

Exemple: considérons les données suivantes



Est-ce que la donnée rouge est une anomalie ou non ?

Visuellement: il semble que oui car elle est fort différente de nos données d'entraînement

Dans notre modèle: chaque caractéristique prise séparément possède une valeur probable

Difficulté: cette anomalie ne sera pas probablement pas détectée par notre modèle

Soucis: on ne tient pas compte de la corrélation entre nos caractéristiques

Limitation: l'indépendance des caractéristiques est une hypothèse trop restrictive pour ces situations

Nature de l'anomalie: c'est d'avoir une valeur basse pour  $x_1$  et une haute pour  $x_2$  simultanément

# Distribution normale multivariée

**Modèle actuel:**  $\hat{y} = p(x_1 | \mu_1, \sigma_1^2) \times \dots \times p(x_n | \mu_n, \sigma_n^2) = \prod_{j=1}^n p(x_j | \mu_j, \sigma_j^2)$

$\mu \in \mathbb{R}$  : espérance

$\sigma^2 \in \mathbb{R}$  : variance

Total :  $2n$  paramètres

**Principe clef:** la probabilité d'occurrence pour chaque caractéristique est régit par sa propre loi normale



Comment peut-on tenir compte des corrélations entre les caractéristiques ?

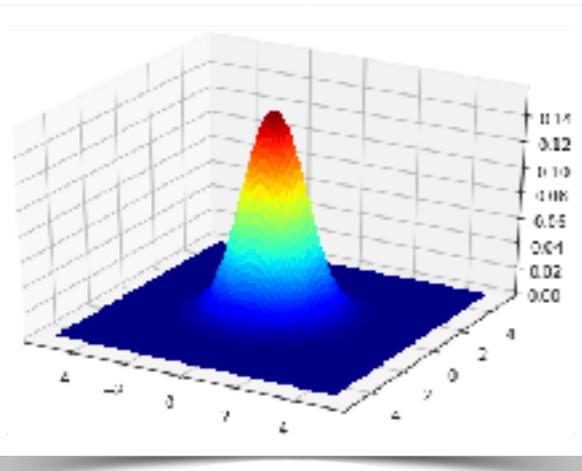


Modèle basé sur une distribution normale multivariée

$$\hat{y} = p(x | \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} \times \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

$\mu \in \mathbb{R}^n$  : vecteur des espérances (une valeur pour chaque caractéristique)

$\Sigma \in \mathbb{R}^{n \times n}$  : matrice variance-covariance (variance/covariance entre chaque paire de caractéristiques)



**Principe:** il s'agit d'une généralisation de la loi normale à plusieurs dimensions

**Différence fondamentale:** permet d'exprimer la corrélation entre les caractéristiques

**Apprentissage:** déterminer les paramètres du modèle

**Intuition:** on aura un modèle plus expressif que le précédent

# Distribution normale multivariée - visualisation

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

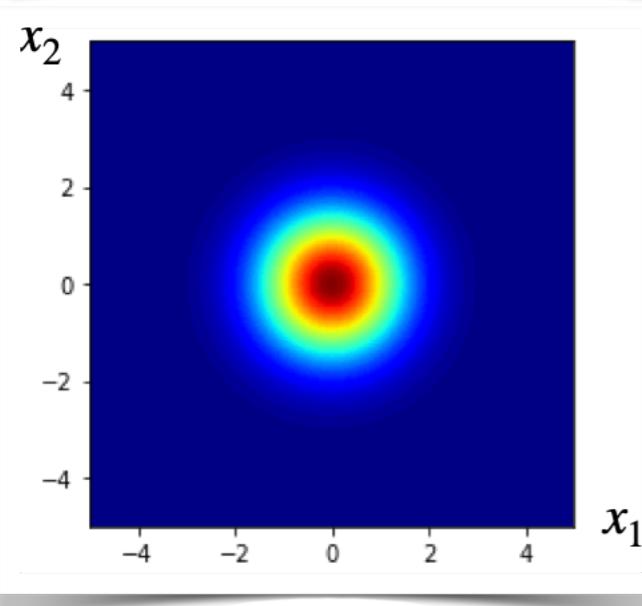
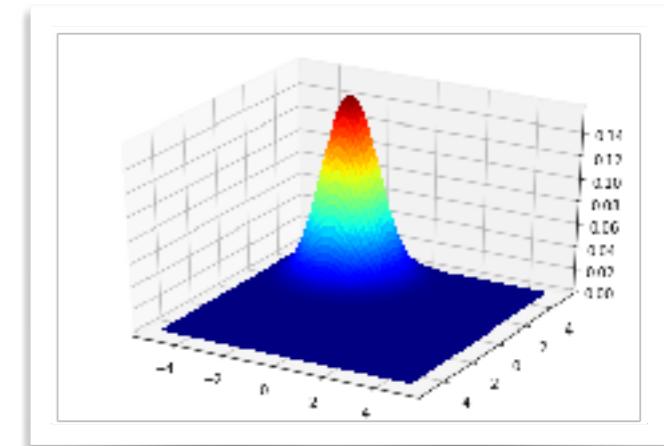
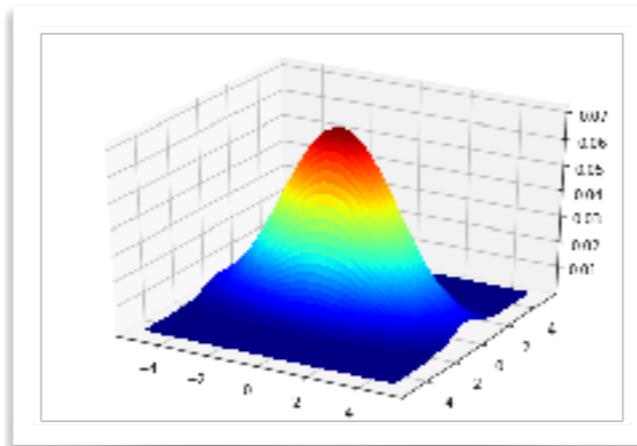
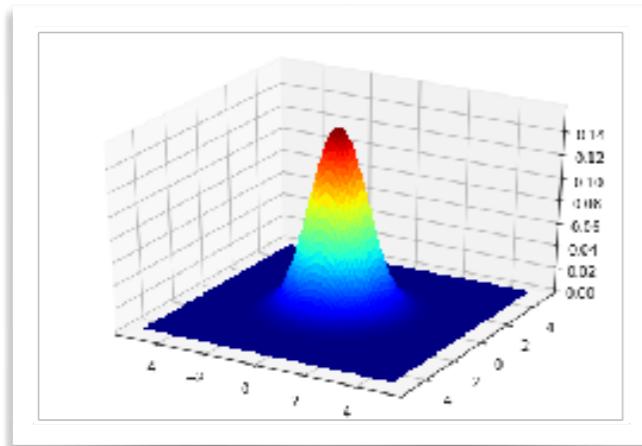
$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix}$$

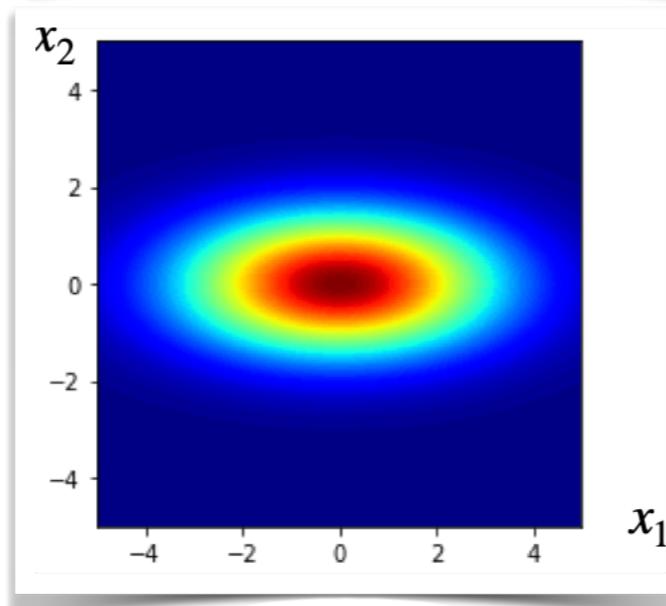
$$\mu = \begin{pmatrix} -2 \\ 3 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



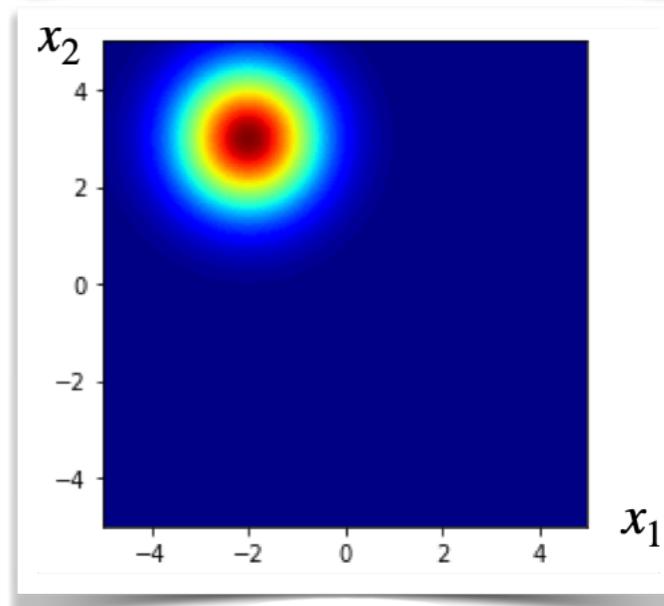
Espérance centrée en (0,0)

Même variance pour chaque caractéristique



Espérance centrée en (0,0)

Les caractéristiques  $x_1$  sont plus dispersées



Espérance centrée en (-2,3)

Même variance

**Diagonale de la matrice: correspond à la variance de chaque caractéristique**

Ces distributions pouvaient déjà être modélisée avec notre premier modèle (covariances nulles)

# Distribution normale multivariée - visualisation

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

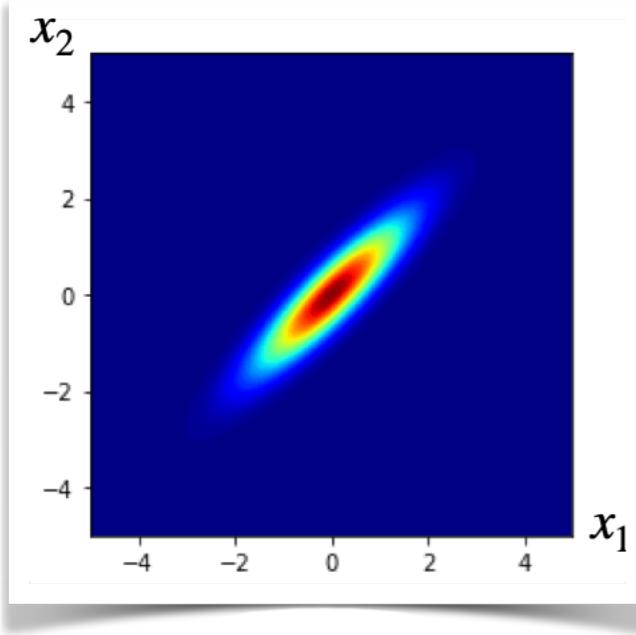
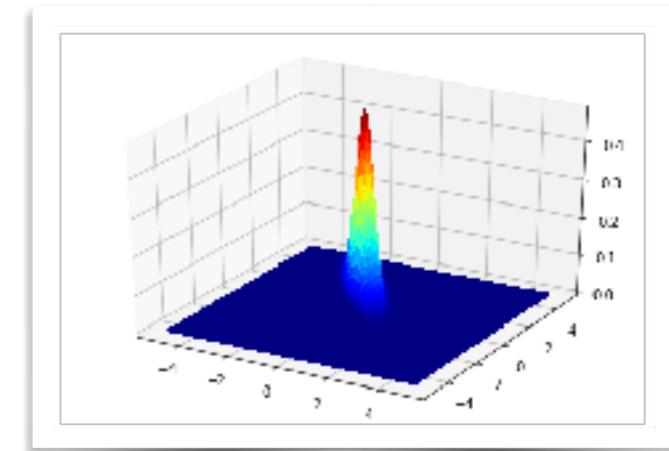
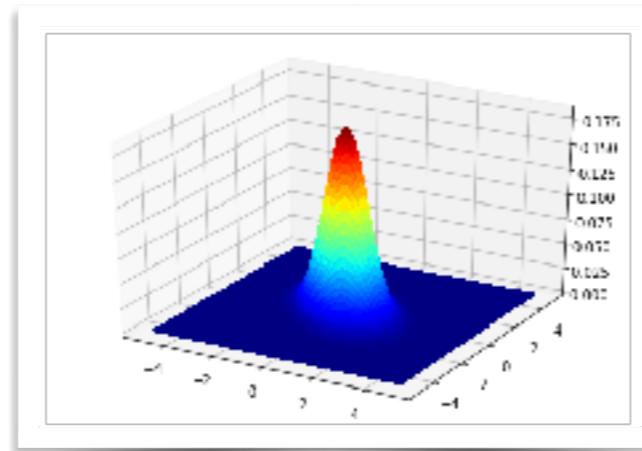
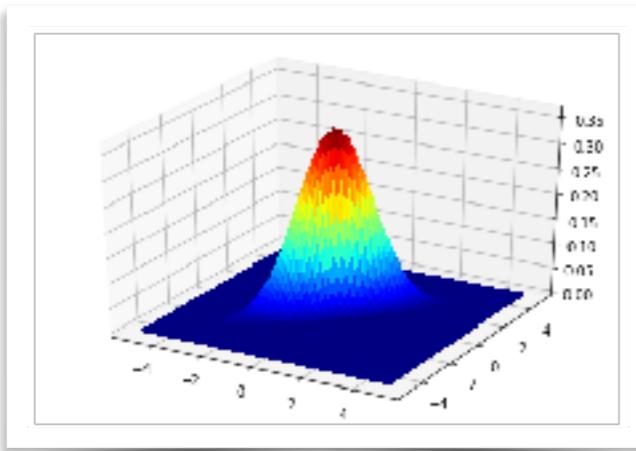
$$\Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}$$

$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$$

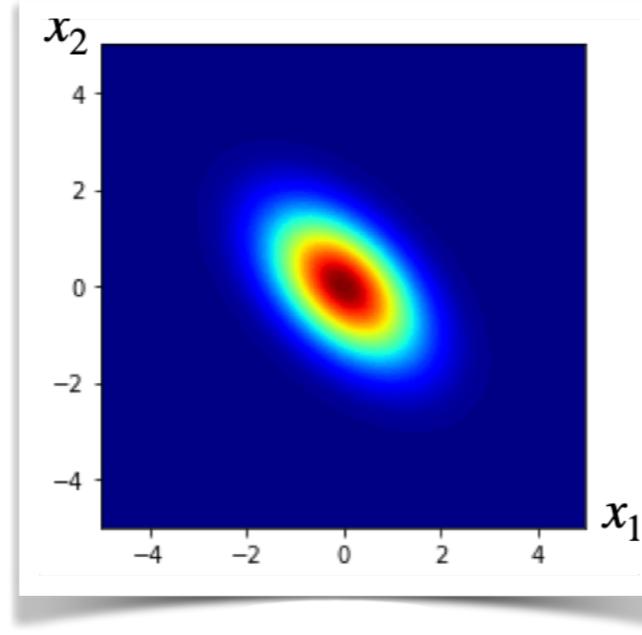
$$\mu = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 0.6 & -0.7 \\ -0.7 & 1 \end{pmatrix}$$



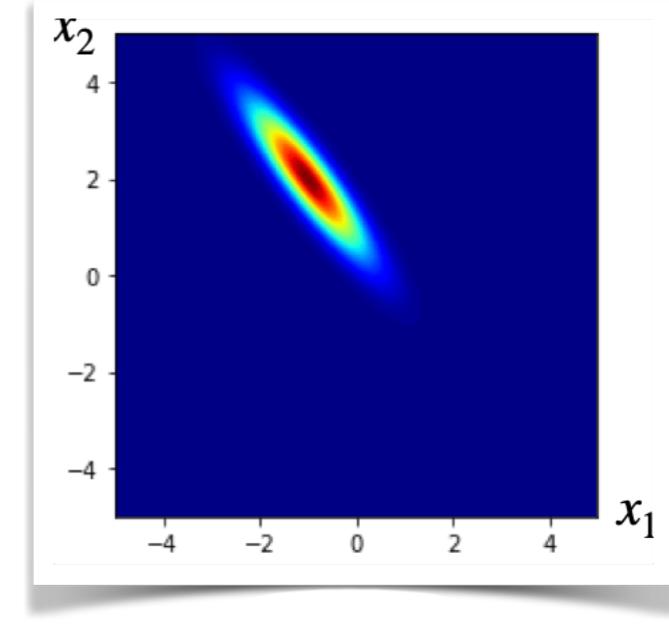
**Covariance positive**

Les deux caractéristiques ont tendance à augmenter ensemble



**Covariance négative**

Une caractéristique a tendance à diminuer lorsque l'autre augmente



**Situation quelconque**

**Expressivité du modèle:** ces situations ne pouvaient pas être modélisées avec notre premier modèle

**Généralité:** s'adapte à n'importe quel nombre de dimensions

# Détection d'anomalies - second modèle



**Objectif:** trouver la meilleure loi normale multivariée

**Principe:** prendre les paramètres maximisant la fonction de vraisemblance

$$\text{Maximum de vraisemblance: } \max_{\mu, \Sigma} L(\mu, \Sigma | x^{(1)}, \dots, x^{(m)})$$

**Bonne nouvelle:** il s'agit encore d'un résultat statistique connu

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

Notez bien que chaque donnée est un vecteur de dimension  $n \times 1$



Combien de paramètres doit-on déterminer pour  $n$  caractéristiques ?

**Exemple:** considérons des données à deux caractéristiques

**Vecteur des moyennes:**  $\mu = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$

Vecteur de  $n$  paramètres

**Matrice variance/covariance:**  $\Sigma = \begin{pmatrix} 0.6 & -0.7 \\ -0.7 & 1 \end{pmatrix}$  Matrice symétrique de taille  $n \times n$

Valeurs de la diagonale

Valeurs de la partie supérieure

$$n + \frac{(n^2 - n)}{2} \text{ paramètres}$$

**Total:**  $n + n + \frac{(n^2 - n)}{2}$  paramètres à déterminer (contre  $2n$  dans notre modèle précédent)

**Attention:** ne pas oublier de considérer que la matrice variance/covariance est symétrique

# Algorithme de détection d'anomalies - second modèle

## Phase d'entraînement

**Objectif:** déterminer les paramètres de la loi normale multivariée

AnomalyDetectionTrainingMulti( $D$ ) :

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

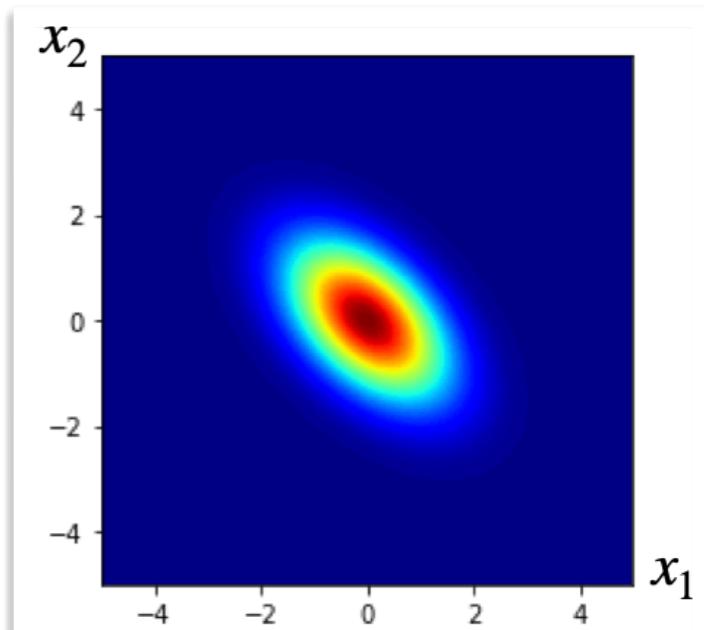
return  $\mu, \Sigma$

On utilise les données d'entraînement

Définition du vecteur des espérances

Définition de la matrice variance/covariance

On retourne tous les paramètres (c-à-d, le modèle)



## Phase d'évaluation

**Objectif:** calculer la probabilité qu'une nouvelle donnée soit régulière

AnomalyDetectionMulti( $x^{(\text{test})}, \epsilon, \mu, \Sigma$ ) :

$$\begin{aligned}\hat{y} &= p(x^{(\text{test})} | \mu, \Sigma) \\ &= \frac{1}{(2\pi)^{n/2} \times \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)\end{aligned}$$

if  $\hat{y} < \epsilon$  : return anomaly for  $x^{(\text{test})}$

else : return regular data for  $x^{(\text{test})}$

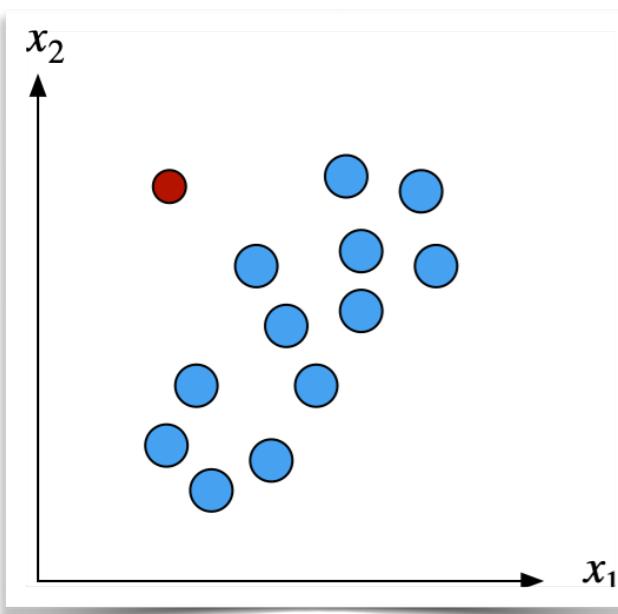
On utilise la loi normale multivariée définie

La probabilité est définie selon cette loi normale

En dessous du seuil: on considère la donnée comme une anomalie

# Comparaison entre les deux modèles

## Distribution normale univariée



? Quelles sont les forces et faiblesses des deux modèles ?

$$\begin{aligned}\hat{y} &= \prod_{j=1}^n p(x_j | \mu_j, \sigma_j^2) \\ &= \prod_{j=1}^n \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(x - \mu_j)^2}{2\sigma_j^2}\right)\end{aligned}$$

$\mu \in \mathbb{R}$  : espérance

$\sigma^2 \in \mathbb{R}$  : variance

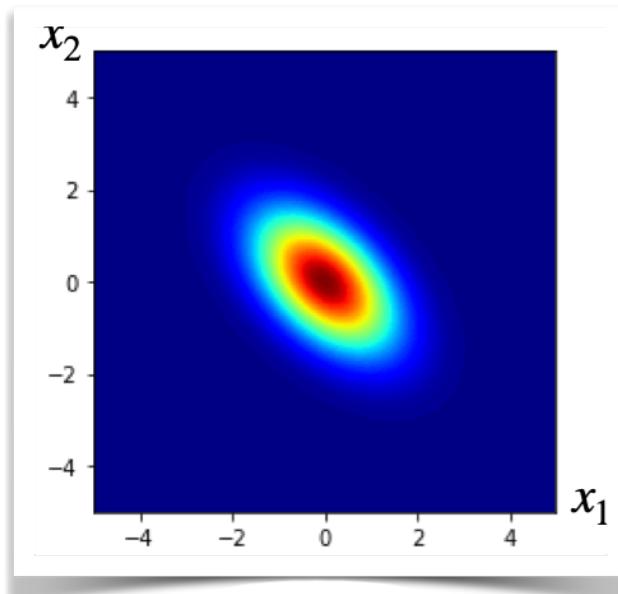
#paramètres :  $2n$

**Avantage:** croissance linéaire des paramètres avec le nombre de caractéristiques

**Avantage:** très peu coûteux à calculer

**Inconvénient:** manque d'expressivité pour considérer les corrélations

## Distribution normale multivariée



$$\begin{aligned}\hat{y} &= p(x | \mu, \Sigma) \\ &= \frac{1}{(2\pi)^{n/2} \times \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)\end{aligned}$$

$\mu \in \mathbb{R}^n$  : vecteur des espérances

$\Sigma \in \mathbb{R}^{n \times n}$  : matrice covariance

#paramètres :  $2n + \frac{(n^2 - n)}{2}$

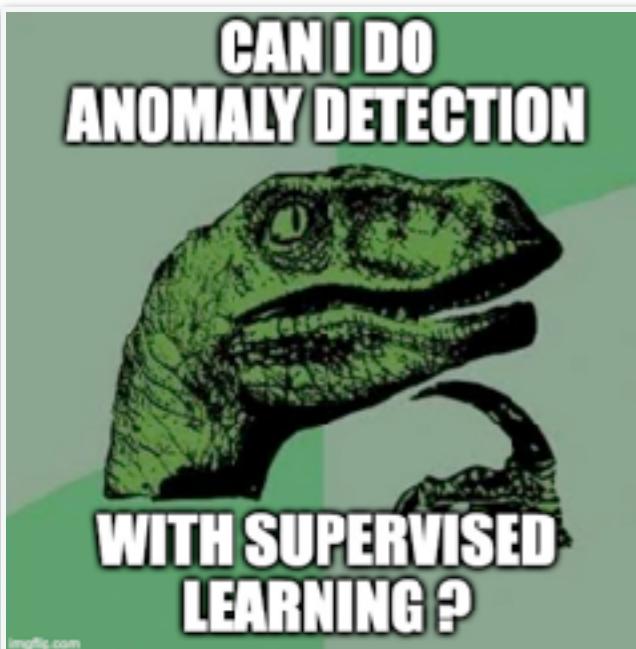
**Avantage:** le modèle assez expressif pour capturer les corrélations

**Inconvénient:** la croissance des paramètres est maintenant quadratique

**Inconvénient:** plus coûteux à calculer (inversion d'une matrice)

**Difficulté:** requiert un grand nombre de données (au moins plus que  $n$ ) pour avoir une matrice inversible

# Différences avec l'apprentissage supervisé



**Idée:** collecter un grand nombre de données concernant la tâche à étudier

$$D : \left\{ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)}) \right\}$$

$(x^{(i)}, y^{(i)})$  avec  $y^{(i)} = 0 \rightarrow$  donnée régulière

$(x^{(i)}, y^{(i)})$  avec  $y^{(i)} = 1 \rightarrow$  anomalie

?

Que pensez-vous de cette idée ?



**Apprentissage supervisé:** suppose un grand nombre d'exemples de chaque classe

**Anomalie:** difficile à caractériser (peu semblable à tout ce qu'on a pu voir)

**Difficulté:** on a rarement un nombre suffisant de données pour les anomalies

Même si on peut avoir beaucoup de données concernant les cas sains

**Apprentissage non-supervisé:** pas sensible à cette difficulté

## Passage progressif à de l'apprentissage supervisé



**Au fil du temps:** on peut avoir assez de données pour caractériser une anomalie

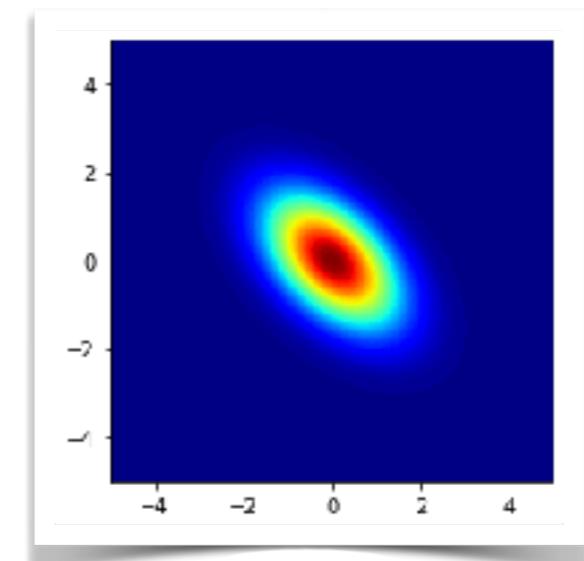
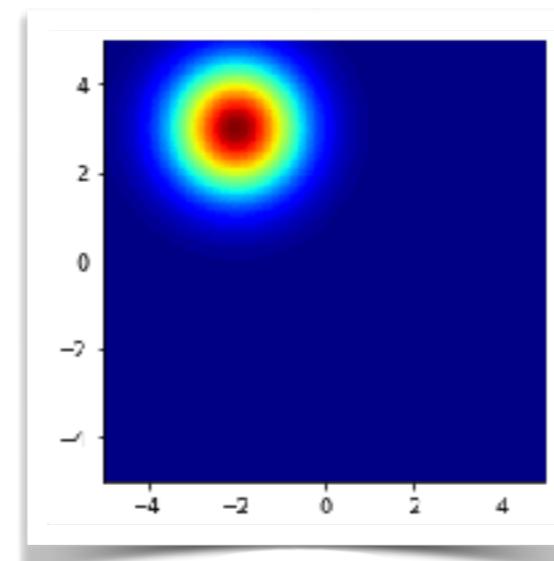
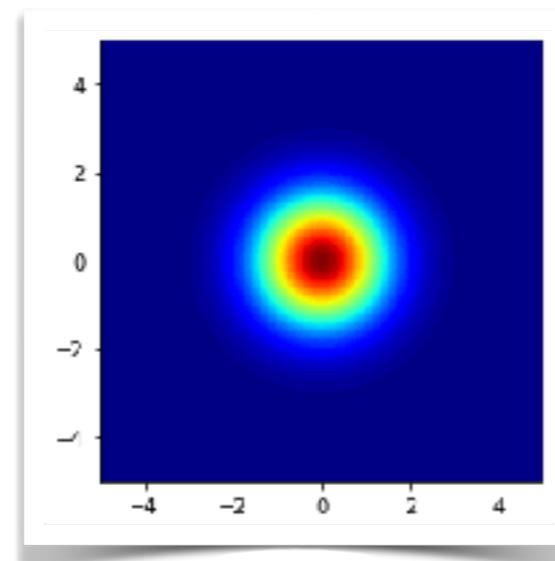
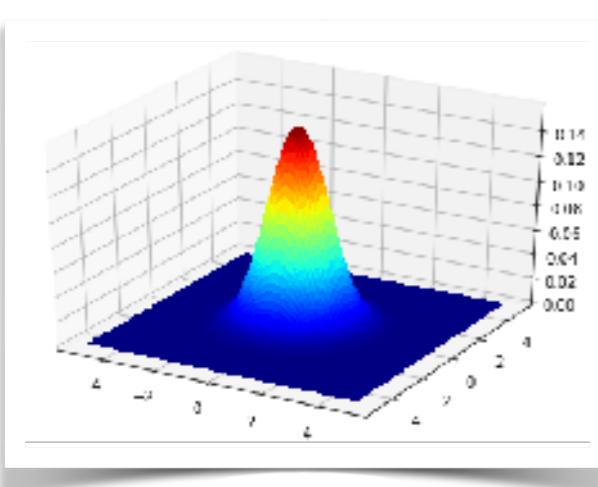
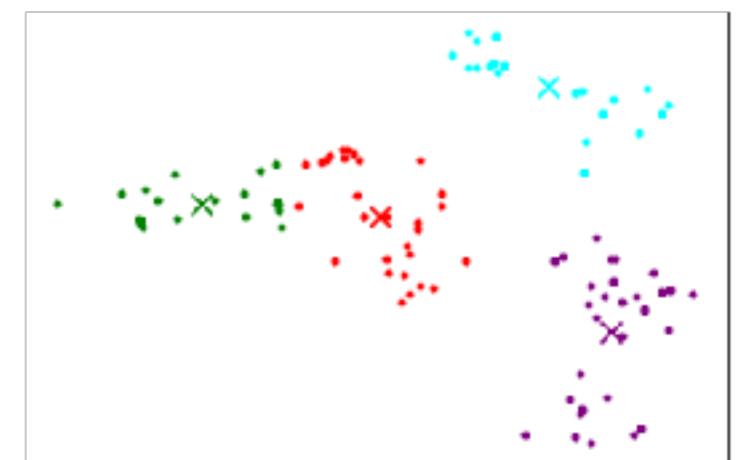
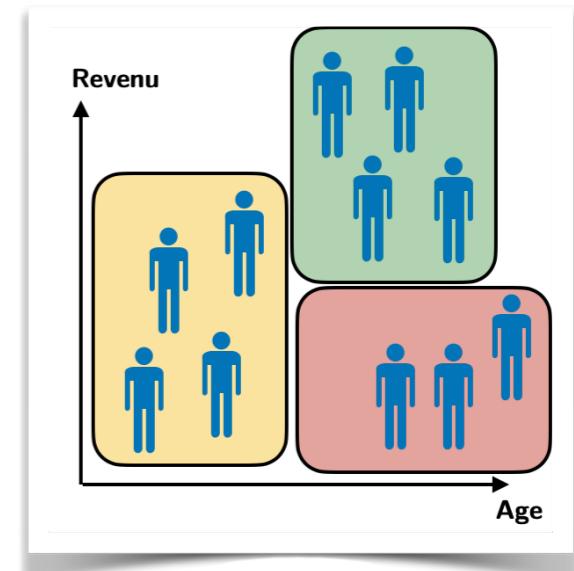
**Conséquence:** il peut être intéressant de passer à une méthode supervisée

**Exemple:** détection de spams (qui sont devenus très fréquents)

# Table des matières

## Apprentissage non-supervisé

-  1. Définition et applications de l'apprentissage non-supervisé
-  2. Formalisation du problème de regroupement (*clustering*)
-  3. Algorithme *K-means*
-  4. Formalisation du problème de détection d'anomalies
-  5. Notion de statistique: maximum de vraisemblance
-  6. Modélisation par des distributions normales univariées
-  7. Modélisation par une distribution normale multivariée



# Synthèse des notions vues

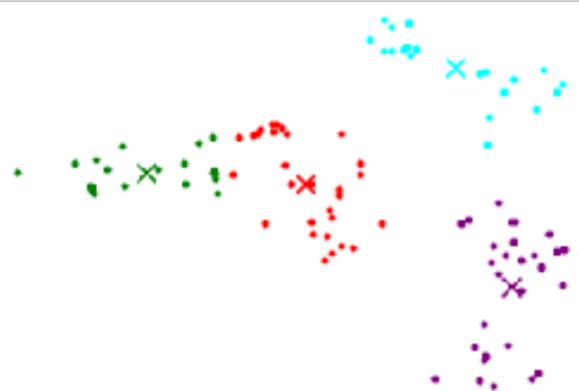
## Apprentissage non-supervisé

$$D : \left\{ x^{(1)}, x^{(2)}, \dots, x^{(m)} \right\}$$

**Définition:** apprentissage qui se fait sur base de données non-labellisées

**Principe:** exploitation de la similarité entre les données pour l'apprentissage

## Problème de regroupement



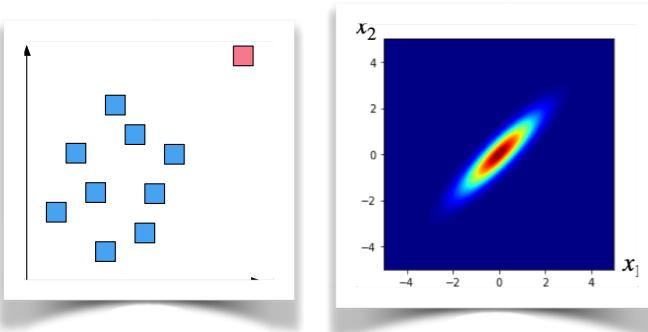
**Objectif:** regrouper les données similaires en groupes (clusters) cohérents

**Méthode de résolution:** algorithme *K-means*

**Principe:** recherche locale pour minimiser une fonction de coût

**Améliorations possibles:** redémarrages, choix des positions initiales, etc.

## Problème de détection d'anomalies



**Objectif:** prédire la probabilité qu'une nouvelle donnée soit régulière

**Principe:** maximiser une fonction de vraisemblance

**Modélisation:** produit de lois normales univariées ou une loi normale multivariée

## Cours données à Polytechnique



**INF8111:** Fouille de données (Daniel Aloise)

**LOG6308:** Systèmes de recommandation (Michel Desmarais)

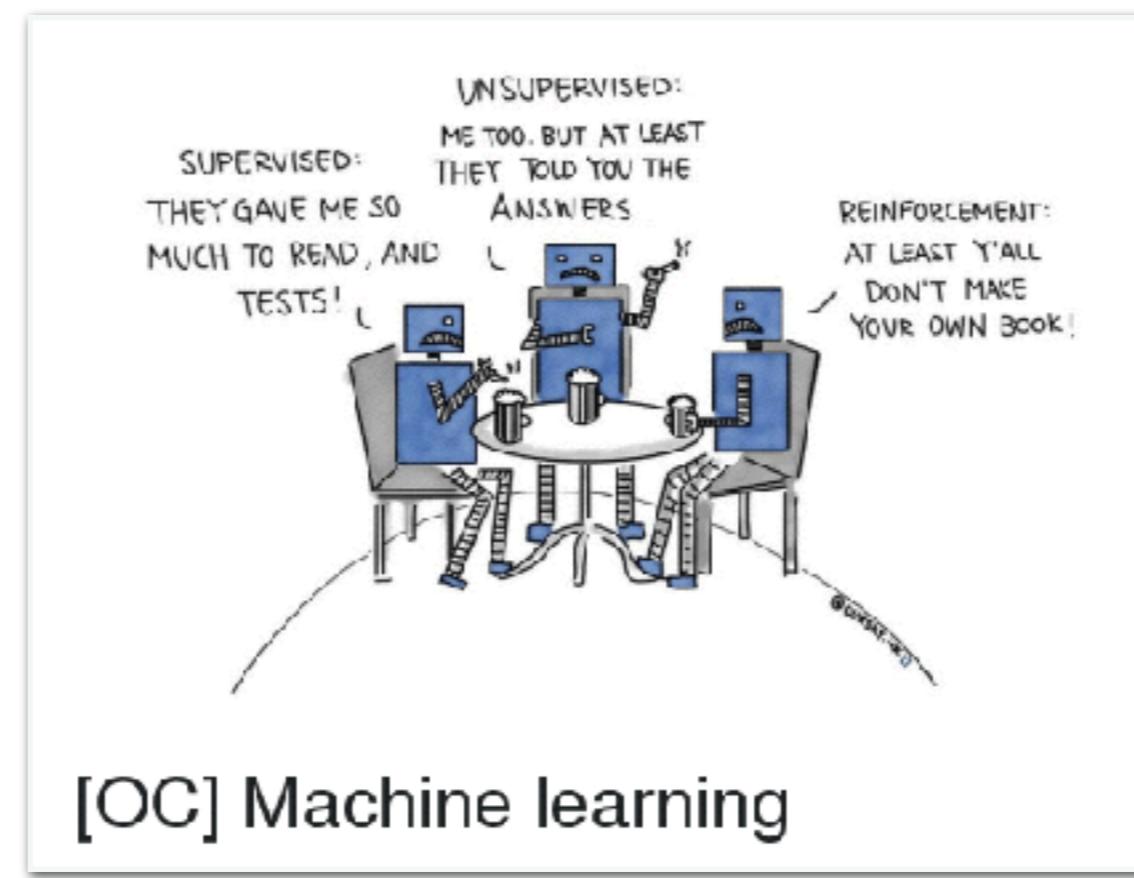
# Exemples de questions d'examen

## Théorie

1. Donner deux applications et exemple d'application de l'apprentissage non-supervisé
2. Expliquer les garanties de convergence de l'algorithme k-means
3. Expliquer les avantages/inconvénients des deux modèles pour la détection d'anomalie

## Pratique

1. Effectuer plusieurs itérations de l'algorithme k-means sur un jeu de donnée
2. Savoir détecter si une donnée est une anomalie à l'aide des modèles vus





DALLE: *Detecting an anomaly, by Picasso*