Distances et clustering

Introduction

Distance (Wikipedia): En mathématiques, une distance est une application qui formalise l'idée intuitive de distance, c'est-à-dire la longueur qui sépare deux points.

Sans faire de maths, on connaît de nombreuses manières de calculer des distances:

- Distance entre 2 villes (nombre de km via une route terrestre, navale ou aérienne)
- Distance à vol d'oiseau.
- Distance sur un clavier (nombre de touches séparant deux lettres)
- Distance sur un arbre généalogique (nombre de générations, principe du frère, du cousin, ...)
- Distance entre amis (ami proche, connaissance, collègue...)

Notion de groupes/clusters

Grâce aux distances, on forme intuitivement des groupes ou des **clusters**:

- Des clusters d'amis
- Des clusters de vêtements
- Des clusters de pays
- Des clusters de mots
- Des clusters d'animaux...

Oui, mais: sur quelle base? Selon ce qu'on regarde et ce qui nous semble important, les clusters ne sont pas les mêmes.

Clustering

Clustering : à partir d'un jeu de données, faire du clustering revient à séparer les données en clusters de telle manière qu'au sein d'un cluster, les données se ressemblent (distance faible) et qu'entre les clusters, les données ne se ressemblent pas (distance élevée).

Exemple:

Proposons des manières de clusteriser les personnes de la classe.

Clustering

Clustering : à partir d'un jeu de données, faire du clustering revient à séparer les données en clusters de telle manière qu'au sein d'un cluster, les données se ressemblent (distance faible) et qu'entre les clusters, les données ne se ressemblent pas (distance élevée).

Exemple:

Proposons des manières de clusteriser les personnes de la classe.

- Par sexe?
- Par couleur de cheveux?
- Par âge?
- Par couleur de cheveux et d'yeux?
- Tout ceci en même temps?

Conclusions anticipées

Intuitivement on remarque que cela dépend au moins :

- du nombre de clusters que l'on veut
- des attributs (variables) que l'on regarde
- de la manière de calculer la distance

Il n'y a donc pas de clustering unique.

Objectifs du clustering

Un algorithme de **clustering** est une méthode qui sépare les points en groupes en minimisant la distance intra-groupes et à maximisant la distance inter-groupes.

De nombreuses méthodes existent : on ne peut pas dire avec certitude que l'une est meilleure que l'autre. Cela dépend des données, de la distance, etc.

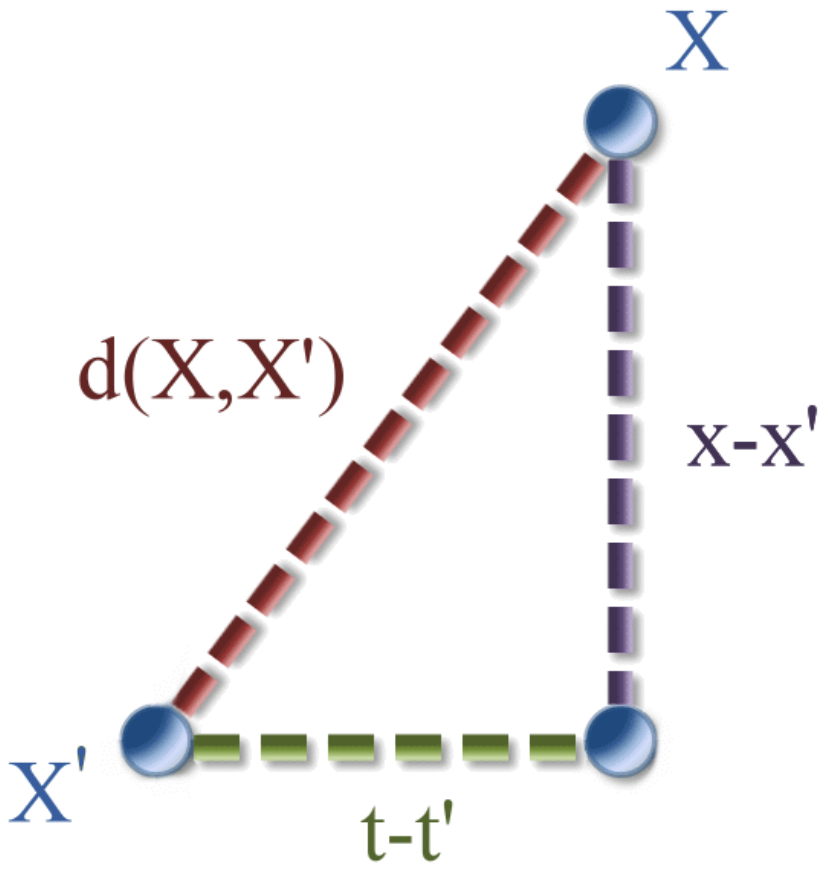
Les projets concernés

Presque tous !

- Distance entre ebooks
- Distance entre équipes de foot
- Distance entre photos
- Distance entre amis
- Distance entre pays
- Distance entre appartements
- Distance entre tweets
- Distance entre produits
- Distance entre parcours de course
- ...

Outline

Distance euclidienne



Distance euclidienne : le théorème de Pythagore, le plus court chemin entre deux vecteurs.

$$d(X,X')^2 = (x-x')^2+(t-t')^2$$

Source : naturelovesmath.blogspot.fr

Application aux textes

- Document 1: "voici un premier texte"
- Document 2: "voici un second texte"
- Document 3: "ce document contient ce texte".

	voici	*un*	*premier*	*texte*	*second*	*ce*	*document*	*contient*
d_1	0; 1	0; 1	0; 275	0	0	0	0	0
d_2	0; 1	0; 1	0	0	0; 275	0	0	0
d_3	0	0	0	0	0	0; 44	0; 22	0; 22

- $D(d_1, d_2) = \sqrt{0^2 + 0^2 + 0.275^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2} = 0.275$
- $D(d_1, d_3) = \sqrt{0.1^2 + 0.1^2 + 0.275^2 + 0^2 + 0^2 + (-0.44)^2 + (-0.22)^2 + (-0.22)^2} = 0.62$
- $D(d_2, d_3) = \sqrt{0.1^2 + 0.1^2 + 0^2 + 0^2 + 0.275^2 + (-0.44)^2 + (-0.22)^2 + (-0.22)^2} = 0.62$

Les premier et second textes sont proches entre eux et loin du troisième.

D'autres distances

Distance de Tchebychev, dite distance l_∞:

$$D(x; y) = \max_i |x_i - y_i|$$

i.e. la plus grande des distances entre deux éléments. Dans notre exemple:

	voici	*un*	*premier*	*texte*	*second*	*ce*	*document*	*contient*
d_1	0; 1	0; 1	0; 275	0	0	0	0	0
d_2	0; 1	0; 1	0	0	0; 275	0	0	0
d_3	0	0	0	0	0	0; 44	0; 22	0; 22

- $D(d_1, d_2) = |0.275 - 0| = 0.275$
- $D(d_1, d_3) = |0.44 - 0| = 0.44$
- $D(d_2, d_3) = |0.44 - 0| = 0.44$

Les premier et second textes sont toujours plus proches entre eux que du troisième, mais la différence est moindre.

D'autres distances

Distance de Manhattan, dite distance l_1 :

$$D(x; y) = \sum_i |x_i - y_i|$$

i.e. la somme des valeurs absolues des distances entre éléments. Dans notre exemple:

	voici	*un*	*premier*	*texte*	*second*	*ce*	*document*	*contient*
d_1	0; 1	0; 1	0; 275	0	0	0	0	0
d_2	0; 1	0; 1	0	0	0; 275	0	0	0
d_3	0	0	0	0	0	0; 44	0; 22	0; 22

- $D(d_1, d_2) = |0| + |0| + |0.275| + |0| + |0| + |0| + |0| + |0| = 0.275$
- $D(d_1, d_3) = |0.1| + |0.1| + |0.275| + |0| + |0| + | - 0.44| + | - 0.22| + | - 0.22| = 1.365$
- $D(d_2, d_3) = |0.1| + |0.1| + |0| + |0| + |0.275| + | - 0.44| + |0.22| + |0.22| = 1.365$

Les premier et second textes sont toujours plus proches entre eux que du troisième, mais la différence est plus grande.

D'autres distances

Distance de Hamming:

$$D(x; y) = \sum_i (x_i \neq y_i)$$

où $(A) = 1$ si A est vraie et 0 sinon. i.e. la somme des fois où les valeurs sont différentes Dans notre exemple:

	voici	*un*	*premier*	*texte*	*second*	*ce*	*document*	*contient*
d_1	0; 1	0; 1	0; 275	0	0	0	0	0
d_2	0; 1	0; 1	0	0	0; 275	0	0	0
d_3	0	0	0	0	0	0; 44	0; 22	0; 22

- $D(d_1, d_2) = 0 + 0 + 1 + 0 + 1 + 0 + 0 + 0 = 2$
- $D(d_1, d_3) = 1 + 1 + 1 + 0 + 0 + 1 + 1 + 1 = 6$
- $D(d_2, d_3) = 1 + 1 + 0 + 0 + 1 + 1 + 1 + 1 = 6$

Distance plus grossière. Plus appropriée pour comparer deux mots du même nombre de lettres (ou deux phrases du même nombre de mots).

Données de différents types

Exemple: profils Facebook

Anne-Claire Haury


```
{
  "id": "610174245",
  "name": "Anne-Claire Haury",
  "first_name": "Anne-Claire",
  "last_name": "Haury",
  "link": "https://www.facebook.com/anneclaire.haury",
  "birthday": "09/06/1984",
  "hometown": {
    "id": "110774245616525",
    "name": "Paris, France"
  },
  "location": {
    "id": "110774245616525",
    "name": "Paris, France"
  },
  "gender": "female",
  "timezone": 1,
  "locale": "en_US",
  "verified": true,
  "updated_time": "2013-12-27T21:59:45+0000",
  "username": "anneclaire.haury"
}
```


Paris Hilton


```
{
  "about": "\"Good Time\" Feat. Lil Wayne on iTunes now! \nDownload here: smarturl.it/ParisGoodTimeiT",
  "bio": "Live life to the fullest. You only live once.",
  "birthday": "02/17/1981",
  "category": "Entertainer",
  "is_published": true,
  "location": {
    "street": "",
    "city": "Beverly Hills",
    "state": "CA",
    "country": "United States",
    "zip": "90210"
  },
  "talking_about_count": 31359,
  "username": "parishilton",
  "website": "www.parishilton.com www.twitter.com/parishilton www.youtube.com/parishilton www.myspace.com/parishilton ",
  "were_here_count": 0,
  "id": "113208373525",
  "name": "Paris Hilton",
  "link": "https://www.facebook.com/parishilton",
  "likes": 3512787,
  "cover": {
    "cover_id": "10151700399393526",
    "source": "https://fbcdn-sphotos-h-a.akamaihd.net/hphotos-ak-frc3/t1/s720x720/1382396_10151700399393526_2043950597_n.jpg",
    "offset_y": 0,
    "offset_x": 0
  }
}
```


Objectif : calculer la distance entre Paris Hilton et moi...

Brainstorming

Ce problème n'a pas de réponse toute faite, il est difficile. Que nous dit le bon sens?

- La localisation géographique est importante.
- L'âge est important.
- Le nombre d'amis en commun est important.
- Le sexe n'est pas important.
- Les likes en commun sont importants.
- ...

Principe

Puisque les données sont de types différents, il va falloir les regarder séparément. On établit donc une distance entre:

- Les localisations (facile avec les coordonnées géographiques)
- Les likes : une fonction qui dépend des likes en commun.
- Les amis : une fonction qui dépend du nombre d'amis en commun.
- etc.

Puis on les agrège:
Distance totale = distance géo + distance amis + distance likes + ...

On peut mettre des poids:
Distance totale = $_1\times$ **distance géo +** $_2\times$ **distance amis +** $_3\times$ **distance likes + ...**

Parti pris

Il y a donc un parti à prendre sur :

- ce qu'on regarde
- quelle(s) distance(s) on choisit
- comment on les agrège

Tous les choix sont ok, il faut être capable de les justifier.

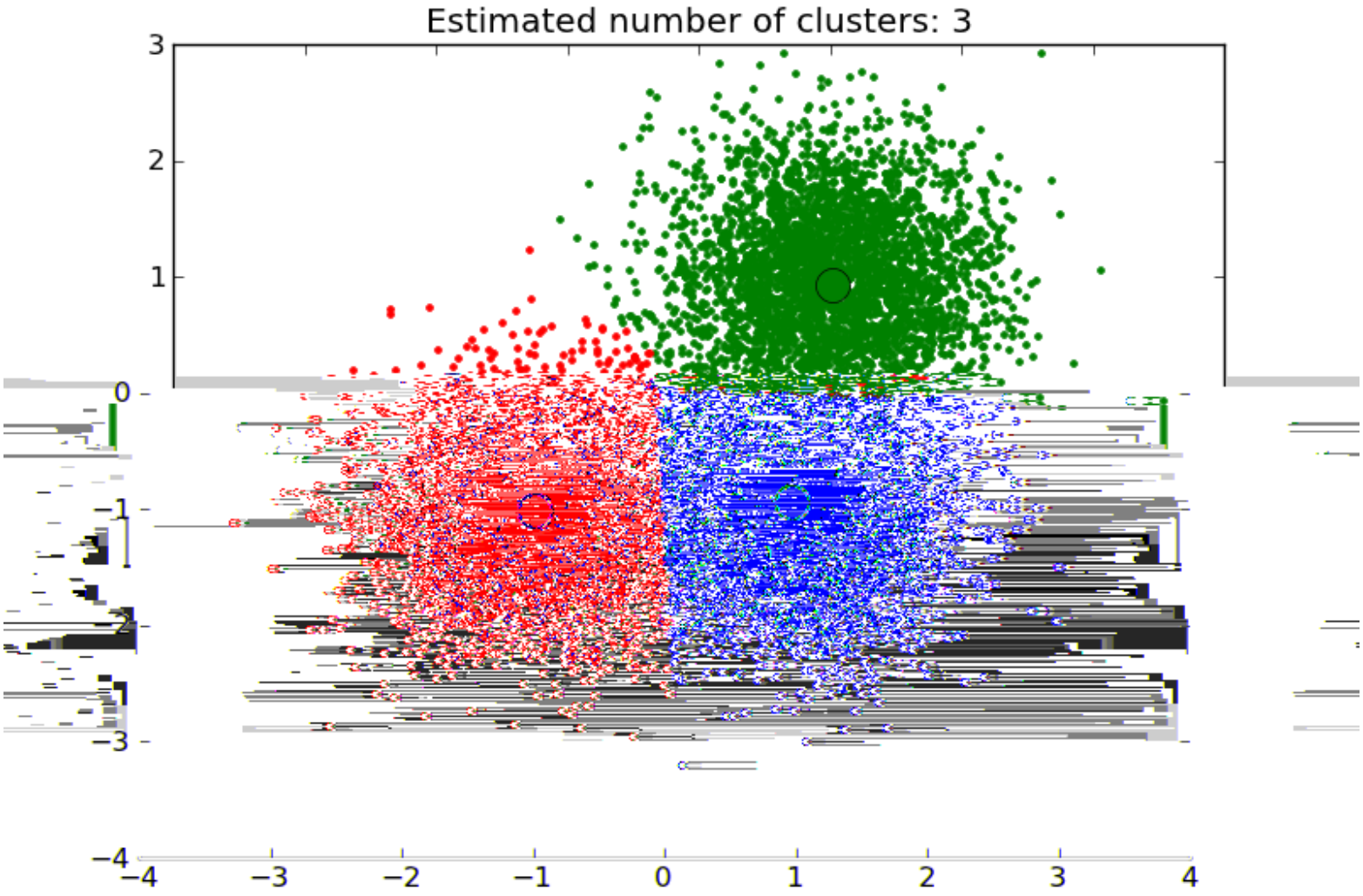
La distance est un outil

En soi, la distance n'est qu'un outil pour arriver à une fin, par exemple clustering ou recommandation.

- Clustering : la distance permet de grouper les données proches.
- Recommandation : la distance permet de proposer à des personnes proches des objets proches.

Outline

Principe



Source : scikit-learn.org

Méthodes

Toutes les méthodes reposent sur des distances.
Nous verrons ici:

- Une méthode probabiliste : K-Means
- Une méthode algorithmique : CAH (classification ascendante hiérarchique)

Outline

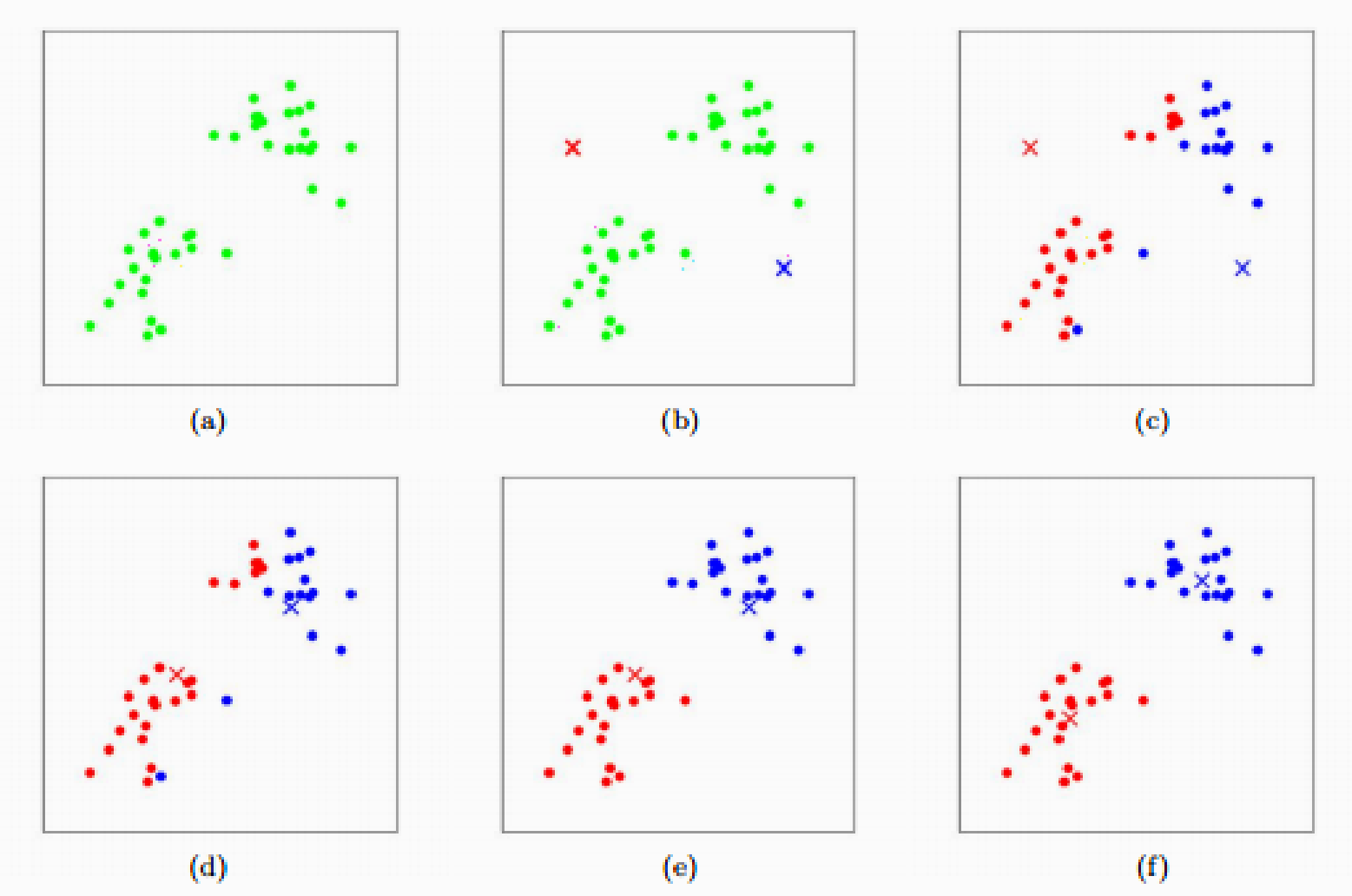
Principe

Cet algorithme **itératif** prend en entrée les données et le nombre de clusters.

Algorithm 1 K-Means

1: **Initialisation** : $m_1; m_2; :::m_K$: K points choisis au hasard parmi tous les points
2: **while** Les points changent de cluster **do**
3: **for** Chaque point **do**
4: Trouver le cluster dont le centre m_j est le plus proche du point;
5: Assigner le point au cluster en question;
6: **end for**
7: Recalculer la moyenne m_j de chaque cluster;
8: **end while**

Etapes



Source : http://bikramhanjra.blogspot.fr/

Calcul du centre le plus proche

- Cet algorithme part du principe que les clusters suivent des **lois normales** (cf cours précédent).
- Il utilise donc la **distance euclidienne** (distance associée à cette loi).
- L'étape d'**assignation** d'un point à un cluster consiste donc à calculer, pour chaque point x_i de taille k et pour chaque centre m_j:

$$d(x_i; m_j) = \sqrt{\sum_k (x_{i,k} - m_{j,k})^2}$$

- On cherche alors le cluster j^* **le plus proche**:

$$j^* = \arg\min_j d(x_i; m_j)$$

- Le point i fait désormais partie du cluster j^*.

Mise à jour des moyennes:

Une fois que les points ont tous été assignés à un cluster, **on recalcule** pour tous les j **la moyenne** m_j du cluster C_j:

$$m_j^{(t+1)} = \frac{1}{n_j^{(t)}} \sum_{i:x_i \in C_j^{(t)}} x_i$$

Arrêt

On arrête lorsque plus aucun point ne change de cluster.

Evaluer la performance

Comment décider si un clustering est meilleur qu'un autre ?
On mesure la distance totale qui est aussi la **variance totale**:

$$L = \frac{1}{n} \sum_i (x_i - m_{c(x_i)})^2$$

où $c(x_i)$ est le cluster final du point x_i.
Plus L est petit, plus le clustering est bon.

Initialisation

- Cet algorithme est très **sensible à l'initialisation** : deux initialisations différentes peuvent produire deux clusterings différents.
- Une solution consiste à le lancer **plusieurs fois**, on obtient donc potentiellement différents clusters. On choisit le **meilleur clustering** en termes de distance (moyenne des distances de chaque point à son centre).
- Il existe de nombreuses méthodes d'initialisation dont nous ne parlerons pas.

Nombre idéal de clusters K

- Dans la plupart des cas, on ne le connaît pas.
- Une solution consiste à essayer différentes valeur pour K. On obtient donc différents clusterings. On choisit le meilleur: celui qui minimise la moyenne des distances entre chaque point et son centre.

Conclusion sur le K-Means

Avantages :

- Un algorithme **intuitif**.
- Un algorithme **itératif**
- Un algorithme **simple à implémenter**
- Un algorithme **rapide** la plupart du temps

Inconvénients:

- **Hypothèse forte** sur la forme des clusters (loi normale). Ne donnera pas toujours des résultats attendus.
- Ne fonctionne qu'avec la **distance euclidienne**.

En python :

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters = 5)
model.fit(X)
```

Outline

Principe

- Au départ, on a n points : n clusters.
- On regroupe les deux points se ressemblant le plus.
- On continue en regroupant à chaque itération les deux clusters se ressemblant le plus.
- On arrête lorsqu'on a le nombre voulu de clusters.

Distance entre deux clusters (1)

Il existe différentes manières de calculer la distance entre deux clusters C_1 et C_2:

- **Distance minimale**: on calcule toutes les distances entre tous les points des deux classes. On prend **la plus petite**:

$$D(C_1; C_2) = \min_{x \in C_1, y \in C_2} d(x; y)$$

- **Distance maximale**: on calcule toutes les distances entre tous les points des deux classes. On prend **la plus grande**:

$$D(C_1; C_2) = \max_{x \in C_1, y \in C_2} d(x; y)$$

Distance entre deux clusters (2)

Il existe différentes manières de calculer la distance entre deux clusters C_1 et C_2:

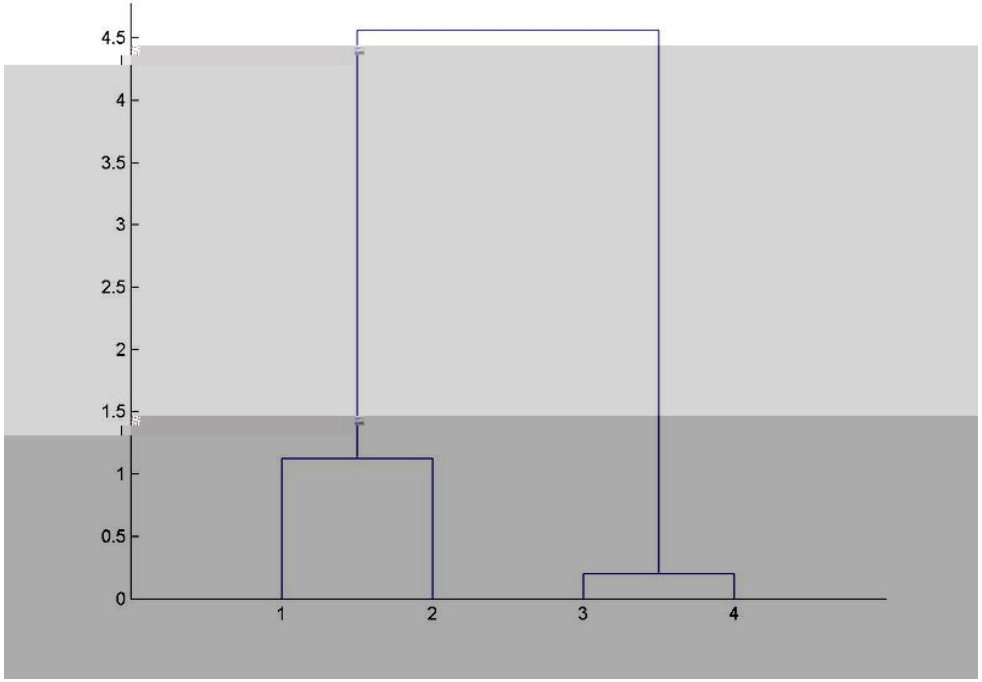
- **Distance moyenne**: on calcule toutes les distances entre tous les points des deux classes. On prend **la moyenne**:

$$D(C_1; C_2) = \frac{1}{n_1 \times n_2} \sum_{x \in C_1, y \in C_2} d(x; y)$$

- **Distance de Ward**: on pondère la distance entre les **centres**:

$$D(C_1; C_2) = \frac{n_1 \times n_2}{n_1 + n_2} d(m_1; m_2)$$

Résultat



Source : Wikipedia

Conclusion sur la CAH

Avantages:

- Intuitif
- Simple
- Fonctionne avec toutes les distances
- On voit l'évolution des clusters

Inconvénients:

- On doit choisir le nombre final de clusters
- On doit choisir le type de distance entre deux clusters.
- On peut donc obtenir des résultats très différents.

En Python:

```
from sklearn.cluster import Ward
model = ward(n_clusters = 5)
model.fit
model.labels_
```

Conclusion

- Différents algorithmes = différentes manières de voir des groupes
- Beaucoup de choix à faire : le résultat ne dépend pas que de l'ordinateur!
- Facile à implémenter => satisfaction immédiate!