

# DISTANCES ET CLUSTERING

# DISTANCE

Une distance est une fonction qui calcule la.....  
**longueur** qui sépare **deux points**.

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

- Distance entre 2 villes (nb de km par la route).
- Distance à vol d'oiseau.
- Distance sur un clavier (entre 2 lettres)
- Distance sur un arbre généalogique (nombre de générations, principe du frère, cousin, ...)
- Distance entre amis (ami proche, connaissance, collègue...)

# GROUPES / CLUSTERS

On “sait” former intuitivement des **groupes** ou des **clusters**:

- Des clusters d'amis
- Des clusters d'images
- 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'à l'**intérieur** d'un cluster, les données se ressemblent (distance faible) et qu'entre clusters différents, les données ne se ressemblent pas (distance élevée).

**Exemple:** Clusterisons les personnes de la classe:

- Par sexe ?
- Par couleur de cheveux ?
- Par couleur de cheveux et d'yeux, sexe et âge ?

# CONCLUSIONS ANTICIPEES

---

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 **disjoints** en **minimisant la distance au sein des groupes** et en **maximisant la distance entre les 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.

# POUR VOS PROJETS

- 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
- Distance entre films



# DISTANCES

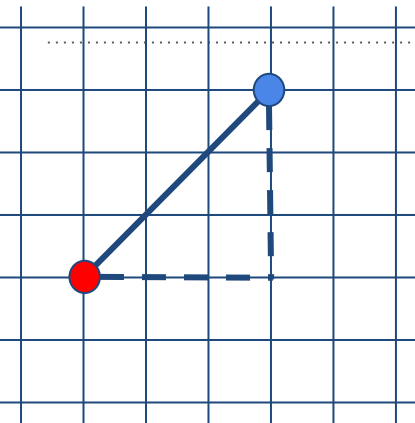
# PROPRIETES DES DISTANCES

---

Pour être *mathématiquement* une distance, une fonction doit vérifier les trois propriétés suivantes:

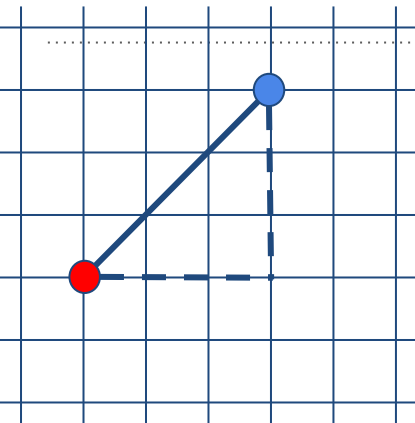
- **Symétrie:**  $d(a, b) = d(b, a)$
- **Inégalité triangulaire:**  $d(a, c) \leq d(a, b) + d(b, c)$
- **Séparation:**  $d(a, b) = 0 \Leftrightarrow a = b$

# DISTANCE EUCLIDIENNE



Calculée grâce au  
théorème de Pythagore,  
le plus court chemin  
entre deux vecteurs.

# DISTANCE EUCLIDIENNE



Calculée grâce au théorème de Pythagore, le plus court chemin entre deux vecteurs.

$$D(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_d - b_d)^2}$$

$$D(A, B) = \sqrt{\sum_{i=1}^d (a_i - b_i)^2}$$

où **A** =  $(a_1, a_2, \dots, a_d)$  et  
**B** =  $(b_1, b_2, \dots, b_d)$

# EXEMPLE AVEC DU TEXTE (TF-IDF)

	voici	un	1er	texte	2nd	ce	doc	a
doc1	0.1	0.1	0.28	0	0	0	0	0
doc2	0.1	0.1	0	0	0.28	0	0	0
doc3	0	0	0	0	0	0.44	0.22	0.22

doc1: voici un 1er texte

doc2: voici un 2nd texte

doc3: ce doc a ce texte

# EXEMPLE AVEC DU TEXTE (TF-IDF)

	voici	un	1er	texte	2nd	ce	doc	a
doc1	0.1	0.1	0.28	0	0	0	0	0
doc2	0.1	0.1	0	0	0.28	0	0	0
doc3	0	0	0	0	0	0.44	0.22	0.22

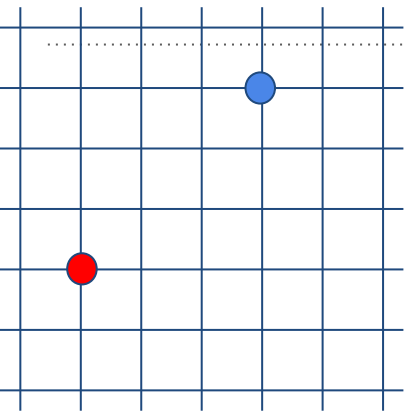
$$D(d_1, d_2) = \sqrt{0^2 + 0^2 + 0.28^2 + 0^2 + 0.28^2 + 0^2 + 0^2 + 0^2} \approx 0.39$$

$$D(d_1, d_3) = \sqrt{0.1^2 + 0.1^2 + 0.28^2 + 0^2 + 0^2 + 0.44^2 + 0.22^2 + 0.22^2} \approx 0.62$$

$$D(d_2, d_3) = \sqrt{0.1^2 + 0.1^2 + 0^2 + 0^2 + 0.28^2 + 0.44^2 + 0.22^2 + 0.22^2} \approx 0.62$$

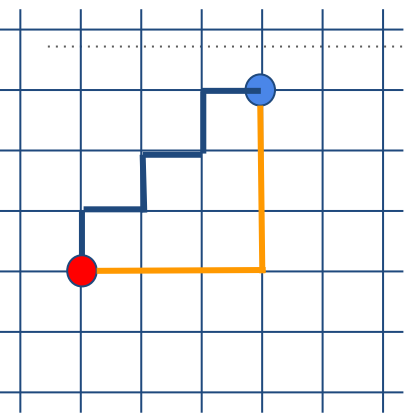
→ Les deux premiers documents sont **proches** entre eux et **loin** du troisième.

# DISTANCE DE MANHATTAN



Vous êtes obligés  
d'utiliser les rues et les  
avenues pour vous  
déplacer dans Manhattan.

# DISTANCE DE MANHATTAN



Vous êtes obligés  
d'utiliser les rues et les  
avenues pour vous  
déplacer dans Manhattan.  
Tous les chemins sont  
équivalents si vous ne  
faites aucun détour.

$$D(A, B) = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_d - b_d|$$

$$D(A, B) = \sum_{i=1}^d |a_i - b_i| \quad \text{où } \mathbf{A} = (a_1, a_2, \dots, a_d) \text{ et } \mathbf{B} = (b_1, b_2, \dots, b_d)$$



# EXEMPLE AVEC DU TEXTE (TF-IDF)

	voici	un	1er	texte	2nd	ce	doc	a
doc1	0.1	0.1	0.28	0	0	0	0	0
doc2	0.1	0.1	0	0	0.28	0	0	0
doc3	0	0	0	0	0	0.44	0.22	0.22

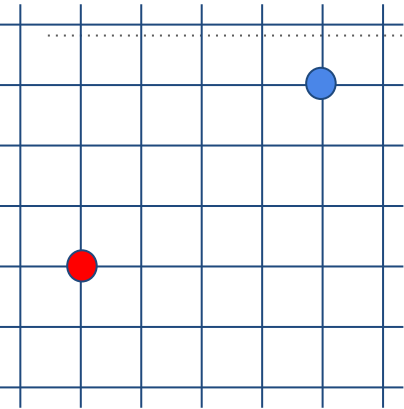
$$D(d_1, d_2) = |0| + |0| + |0.28| + |0| + |0.28| + |0| + |0| + |0| = 0.56$$

$$D(d_1, d_3) = |0.1| + |0.1| + |0.28| + |0| + |0| + |0.44| + |0.22| + |0.22| = 1.36$$

$$D(d_2, d_3) = |0.1| + |0.1| + |0| + |0| + |0.28| + |0.44| + |0.22| + |0.22| = 1.36$$

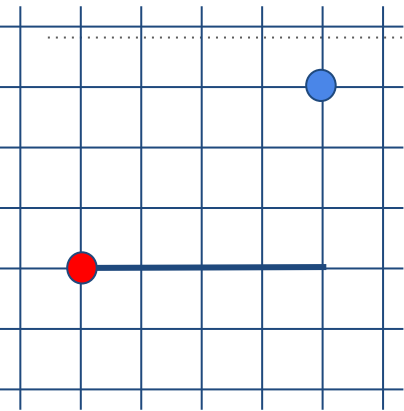
→ Les deux premiers documents sont **toujours proches** entre eux et **loin** du troisième mais la différence est plus accentuée car les valeurs sont  $< 1$ .

# DISTANCE DE TCHEBYCHEV



On prend la pire (plus grande) des différences sur toutes les variables.

# DISTANCE DE TCHEBYCHEV



On prend la pire (plus grande) des différences sur toutes les variables.

$$D(A, B) = \max (|a_1 - b_1|, |a_2 - b_2|, \dots, |a_d - b_d|)$$

$$D(A, B) = \max_{i=1..d} |a_i - b_i| \quad \text{où } \mathbf{A} = (a_1, a_2, \dots, a_d) \text{ et } \mathbf{B} = (b_1, b_2, \dots, b_d)$$

# EXEMPLE AVEC DU TEXTE (TF-IDF)

	voici	un	1er	texte	2nd	ce	doc	a
doc1	0.1	0.1	0.28	0	0	0	0	0
doc2	0.1	0.1	0	0	0.28	0	0	0
doc3	0	0	0	0	0	0.44	0.22	0.22

$$D(d_1, d_2) = |0.28 - 0| = 0.28$$

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

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

→ Cette distance reflète **le poids du mot le plus différent** pour chaque paire de documents.

# DISTANCE DE HAMMING

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

	voici	un	1er	texte	2nd	ce	doc	a
doc1	0.1	0.1	0.28	0	0	0	0	0
doc2	0.1	0.1	0	0	0.28	0	0	0
doc3	0	0	0	0	0	0.44	0.22	0.22

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

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

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

→ Distance plus grossière. Pas appropriée ici.

# DISTANCE ENTRE DES PERSONNES ?

Fabien Viger.....

Paris Hilton.....

```
{
  "id": 610274245,
  "name": "Fabien Viger",
  "birthday": "29/02/1982",
  "hometown": {
    "id": 110718762143124
    "name": "Paris, France"
  },
  "location": {
    "id": 110718762143124
    "name": "Paris, France"
  },
  "gender": "male",
  "timezone": 2,
  "locale": "en_US",
  first_name: "Fabien",
  last_name: "Viger",
}
```

```
{
  "name": "Paris Hilton",
  "about": "Official Facebook Page for Paris",
  "id": "113208373525",
  "birthday": "02/17/1981",
  "location": {
    "city": "Beverly Hills",
    "country": "United States",
    "state": "CA",
    "zip": "90210"
  },
  "website": "www.ParisHilton.com",
  "category": "Public Figure",
  "bio": "Twitter - @ParisHilton
Instagram - @ParisHilton
Snapchat - ParisHilton",
  "talking_about_count": 50900,
  "rating_count": 0
}
```

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

# DISTANCE ENTRE DES PERSONNES ?

Que prendre en compte? Quels sont les attributs qui ont un sens pour ce problème?

- Localisation géographique ?
- Âge ? (et: anniversaire?)
- Nombre d'amis en commun ?
- Sexe ?
- Likes en commun ?
- Prénom ?
- ....

# DISTANCE ENTRE DES PERSONNES ?

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

- Les localisations (facile, avec les lat/lng)
- Les amis : une fonction qui dépend du nombre d'amis en commun.
- Les likes :  $f(\text{num\_likes\_from\_same\_user})$
- etc

Puis on les agrège:

**Distance totale** = distance\_géo + distance\_amis  
+ 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...**

Mais il faut être capable de les **justifier** en fonction de votre application; car une distance est toujours appliquée à un objectif donné.

# LA DISTANCE EST UN OUTIL

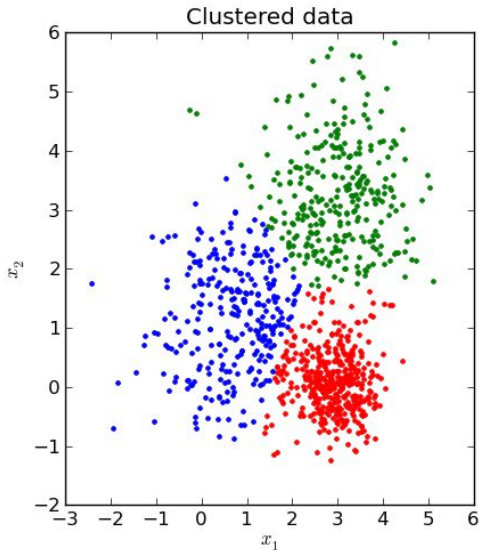
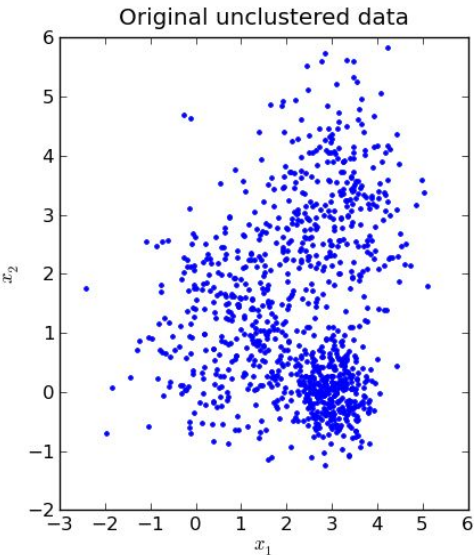
---

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.

# CLUSTERING

# PRINCIPE



# ALGORITHMES

Toutes(\*) les méthodes reposent sur des ..... distances.

Nous verrons ici:

- Une méthode probabiliste : **K-Means**
- Une méthode algorithmique : **CAH**  
(**C**lassification **A**scendante **H**iéarchique)

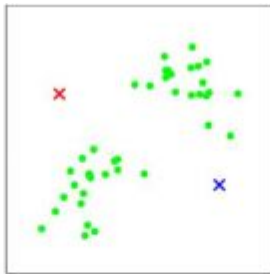
(\*) *presque toutes*

# K-MEANS

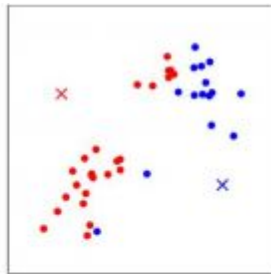
# FONCTIONNEMENT



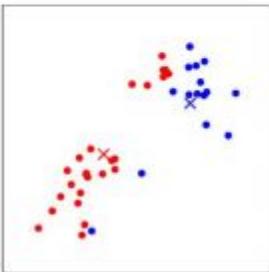
(a)



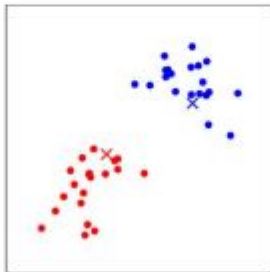
(b)



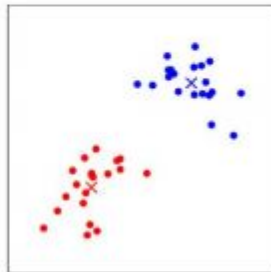
(c)



(d)



(e)



(f)

Source : stanford.edu

# NOTATIONS

- **N** points à clusteriser  $X_1, \dots, X_N$ 
  - En dimension **d** : chaque point = un feature vector de d nombres.  
$$X_1 = (x_{1,1}, x_{1,2}, \dots, x_{1,d})$$
- **K** clusters finaux
  - Chaque cluster  $C_k$  a un centroïde  $M_k$ , qui est aussi un point en dimension d.  
(centroïde = moyenne des points)



# PSEUDO-CODE

Algor **itératif**. **Input:** points  $X$ , nb de clusters  $K$

## K-Means

### 1: Initialisation:

Tirer  $K$  “centres de clusters” choisis au hasard parmi les points  $X$ :  $M_1, M_2, \dots, M_K$

### 2: **While** clusters changé depuis dernière iter:

### 3:   **for each** point $x \in X$ :

4:     Trouver  $j^*$  qui minimise  $\text{distance}(x, M_{j^*})$

5:     Assigner le point au cluster  $j^*$

6:   Re-calculer centroïde  $M_{j^*}$  de chaque

# CENTROÏDE LE PLUS PROCHE

Cet algo part du principe que les clusters suivent des **lois normales**, et utilise donc la **distance euclidienne** (associée naturellement).

L'**assignation** d'un point à un cluster consiste donc à calculer, pour chaque point  $X_i$  de taille  $d$  et pour chaque centre  $M_j$  :

$$D(X_i, M_j) = \sqrt{\sum_{k=1}^d (x_{i,k} - m_{j,k})^2}$$

On cherche alors le cluster  $j^*$  **le plus proche** :  $j^* = \arg \min_j D(X_i, M_j)$

Et enfin on assigne le point  $X_i$  au cluster  $C_{j^*}$

# MISE A JOUR DES MOYENNES

Une fois que les points ont tous été assignés à un cluster, **on recalcule** pour tous les  $j$  **le centroïde**, i.e. la **moyenne**  $M_j$  du cluster  $C_j$  :

$$M_j^{(t+1)} = \frac{1}{N} \sum_{i \in C_j^{(t)}} X_i$$

# ARRÊT


---

On arrête lorsque plus aucun point ne change de cluster entre une itération  $t$  et la suivante  $t+1$ .

# INITIALISATION

- Cet algorithme est très **sensible à** .....  
**l'initialisation** : deux initialisations un peu différentes peuvent produire deux clusterings très différents.
- La solution la plus courante 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).

# NOMBRE DE CLUSTERS K

- Dans la plupart des cas, on ne le connaît pas K
- Une solution consiste à **essayer différentes valeurs** pour K, obtenant différents clusterings
- On peut comparer leur propriétés:
  - (des-)équilibre des tailles de clusters
  - distance de chaque point au centre  
 : baisse naturellement avec K
  - distance de chaque point au cluster (autre que le sien) le plus proche. Ça baisse aussi naturellement avec K; on veut maximiser.
  - ... etc ...

# EVALUATION

Comment décider si un clustering est meilleur.. qu'un autre ?

On peut mesurer la distance totale qui est aussi la **variance totale**:

$$L = \frac{1}{N} \sum_{i=1}^N (X_i - M_{\text{cluster}(i)})^2$$

où  $\text{cluster}(i)$  est la fonction qui à  $i$  associe le cluster  $C_{\square}$  dans lequel on a placé  $X_i$ .

→ Plus  $L$  est petit, mieux c'est.

# EN PYTHON

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

Autres arguments intéressants:

- `n_jobs`: pour paralléliser
- `n_init`: nombre d'initialisations
- `max_iter`: nombre d'itérations max



# CONCLUSION SUR LE K-MEANS

## Avantages :

- **Intuitif, itératif, simple à implémenter**
- **Rapide** la plupart du temps, et parallélisable

## Inconvénients :

- **Hypothèse forte** sur la forme des clusters (loi normale). Ne donnera pas toujours les résultats qu'on attend
- Ne fonctionne qu'avec la **distance euclidienne**
- Oblige l'utilisateur à tatonner pour le **K**
- Ne produit pas des cluster *équilibrés* en taille.

# CLUSTERING HIÉRARCHIQUE

# PRINCIPE

---

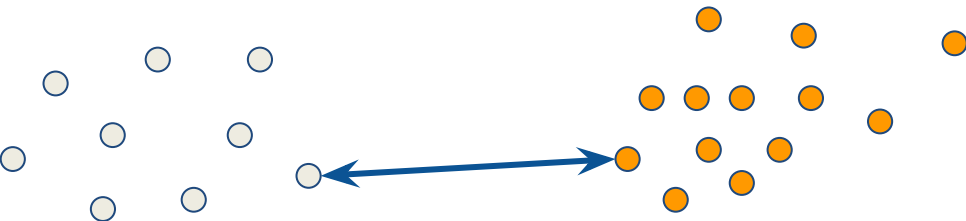
- Au départ, on a  $N$  points, chaque point est un cluster :  $N$  clusters “singletons”.
- À chaque itération on groupe les deux clusters se ressemblant le plus.
- On arrête lorsqu'on a le nombre voulu de clusters, ou lorsqu'un critère d'arrêt est atteint

# DISTANCE ENTRE DEUX CLUSTERS

## Distance minimale / Single Linkage:

On calcule toutes les distances entre tous les points des deux clusters. On prend **la plus petite**:

$$D(C_1, C_2) = \min_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$$

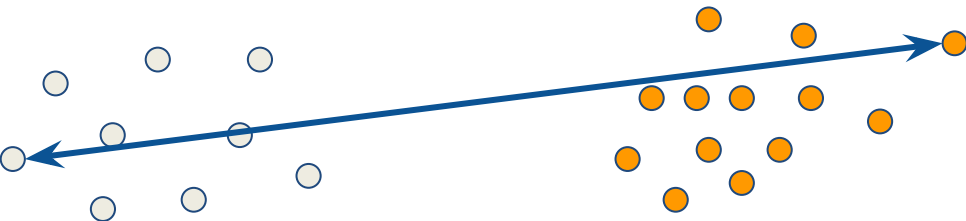


# DISTANCE ENTRE DEUX CLUSTERS

## Distance maximale / Complete Linkage:

On calcule toutes les distances entre tous les points des deux clusters. On prend **la plus grande**:

$$D(C_1, C_2) = \max_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$$

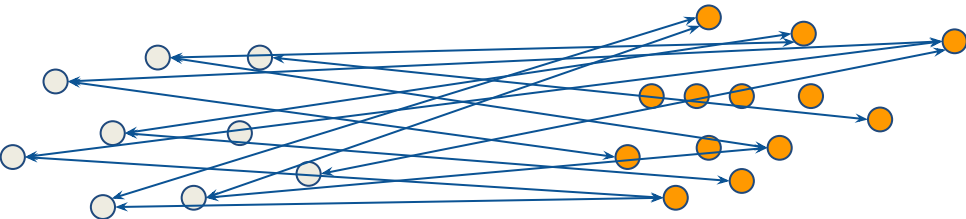


# DISTANCE ENTRE DEUX CLUSTERS

## Distance moyenne / Average Linkage:.....

On calcule toutes les distances entre tous les points des deux clusters. On prend **la moyenne** :

$$D(C_1, C_2) = \frac{1}{n_1 n_2} \sum_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$$

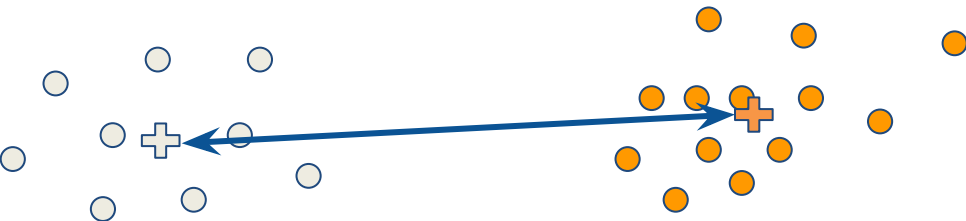


# DISTANCE ENTRE DEUX CLUSTERS

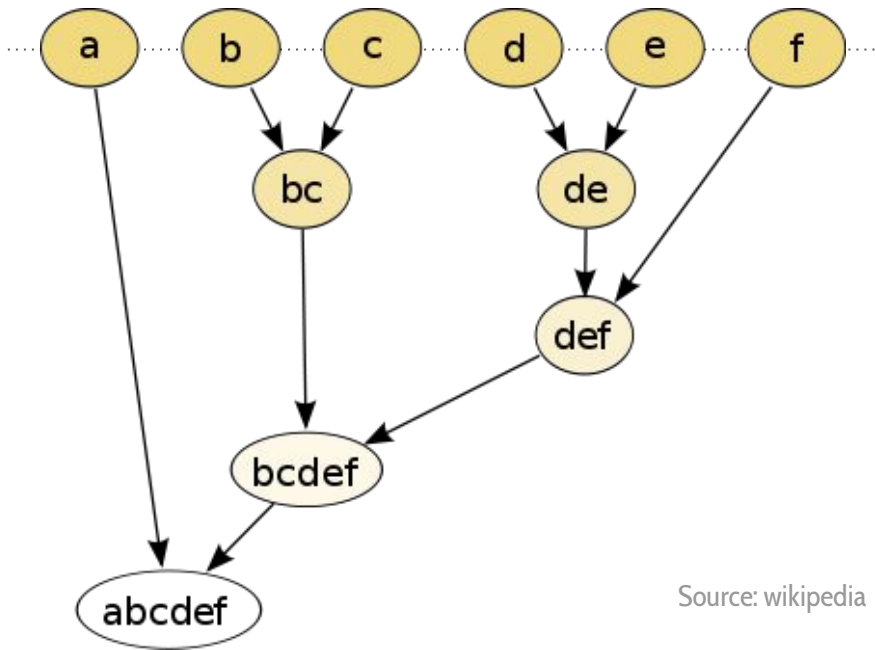
## Distance de Ward / Ward Linkage:

Distance entre les deux centroïdes, pondérée par la taille des clusters (objectif: minimiser l'augmentation de la variance en fusionnant)

$$D(C_1, C_2) = \frac{n_1 n_2}{n_1 + n_2} d(M_1, M_2)$$



# RESULTAT



Source: wikipedia



# PERSONALISATION

À chaque étape, le choix des clusters à fusionner peut prendre en compte d'autres objectifs !

## Exemples:

- Taille max du cluster fusionné  $\leq T_m$   
→ objectif: obtenir des cluster à peu près équilibrés
- Les clusters doivent se toucher  
(clustering géographique)
- etc...

# UTILISATION EN PYTHON

---

```
from sklearn.cluster import
    AgglomerativeClustering
model = AgglomerativeClustering(
    linkage = "complete",
    n_clusters = 10)
model.fit(X)
y = model.labels_
```

# CONCLUSION SUR LA CAH

## Avantages :

- **Simple**
- Fonctionne avec **toutes les distances**.
- Hierarchie: on voit l'évolution des cluster
- Facile à interpréter/analyser
- Très flexible pour intégrer des contraintes supplémentaires

## Inconvénients :

- On doit choisir le nombre final de clusters
- On doit choisir le type de distance.
- On peut donc avoir des résultats très variables

# UN EXEMPLE DE TERRAIN

---

[au tableau]

# CONCLUSION

---

- Différents algorithmes et distances = différentes manières de voir des groupes
- **Beaucoup de choix** à faire qui vont influencer le résultat de manière cruciale.
- Facile à implémenter  $\Rightarrow$  satisfaction immédiate!

# ENCODER DES IMAGES

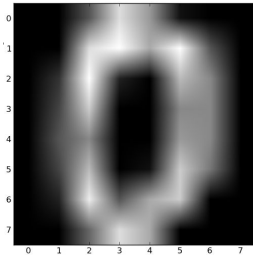
# EXEMPLE

Input : Image N/B, 8 x 8 pixels.....


Pixel  $\rightarrow$  **valeur**  $\in [0..255]$ . 0 = noir,  
255 = blanc, entre: niveaux de gris.

La matrice obtenue est

Transformée en **vecteur**:



```
array([[0, 0, 5, 13, 9, 1, 0, 0],  
       [0, 0, 14, 15, 10, 15, 5, 0],  
       [0, 3, 15, 2, 0, 11, 8, 0],  
       ... ( 5 more rows ) ... ]])
```



```
array([0, 0, 5, 13, 9, 1, 0, 0, 0, 0,  
       14, 15, 10, 15, 5, 0, 0, 3, 15, ...])
```

# DES IMAGES PLUS COMPLIQUEES





# Pré-Traitement(s) : DoG

DoG (Difference of Gaussians) est un outil classique pour repérer les endroits intéressants d'une image (bords, coins). Elle consiste à soustraire une version floutée de l'image à une autre version moins floutée.



sources: wikipedia



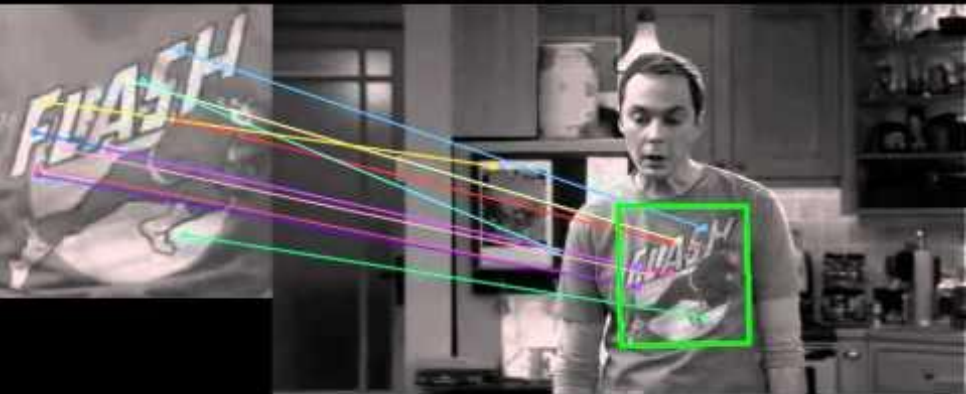
# RESUMER L'INFORMATION 1: SIFT

SIFT (**S**cale **I**nvariant **F**eature **T**ransform, 1999)  
repère des points saillants dans l'image qui vont résumer cette image.



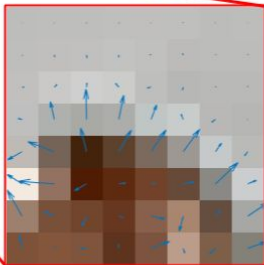
# RESUMER L'INFORMATION 2: SURF

SURF (**S**peeded **U**p **R**obust **F**eatures, 2006)  
s'inspire de SIFT et des ondelettes, en théorie  
du signal.



# RESUMER L'INFORMATION 3: HOG

HOG: s'intéresse aux courbes dans l'image,.....  
détectées par des gradients de couleur.



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

**Gradient Magnitude**

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

**Gradient Direction**

# C'EST DU DEEP LEARNING?

Non! Pas du machine learning (donc pas DL).....

C'est du **traitement du signal**.

Le Deep Learning (DL), en traitement d'images, trouve les features intéressantes "tout seul".

C'est **magique** mais cela a au moins 4 coûts:

1. il va lui falloir **beaucoup** d'images
2. on ne sait pas interpréter ces modèles
3. ils sont très difficiles et longs à tuner
4. complexité en termes de calculs

Mais depuis 2010+: traitement d'images  $\Rightarrow$  DL

# CONCLUSION

Encoder et nettoyer les données consiste entre autres à **choisir ce qui est important** pour la suite.

Pas (forcément) la partie la plus drôle.

Probablement **la partie la plus déterminante**.

En tout cas une partie où l'humain (vs machine) joue un rôle important.