EPSI l'École d'ingénierie informatique

Big Data & Traitement par l'IA
II

Jean-Claude Houbart

houbart.jc@gmail.com

EPSI l'École d'ingénierie informatique

Réduction de dimensionnalité
PCA, tSNE

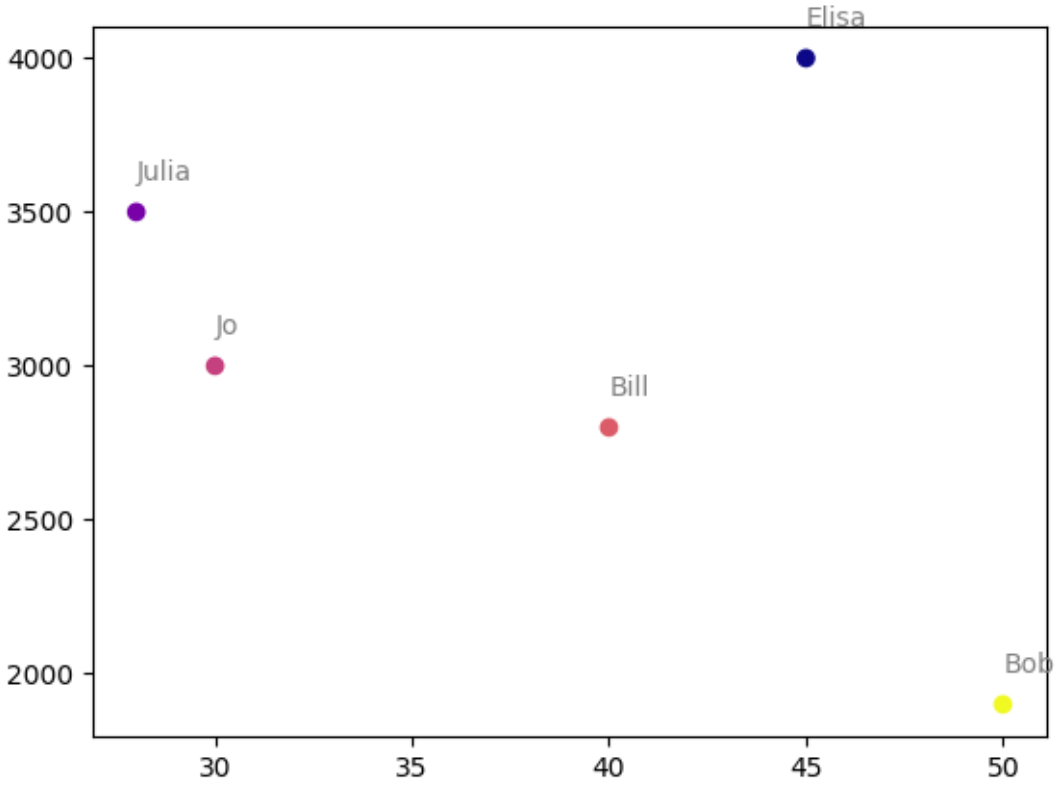
Clustering avec KMeans

Jean-Claude Houbart

houbart.jc@gmail.com

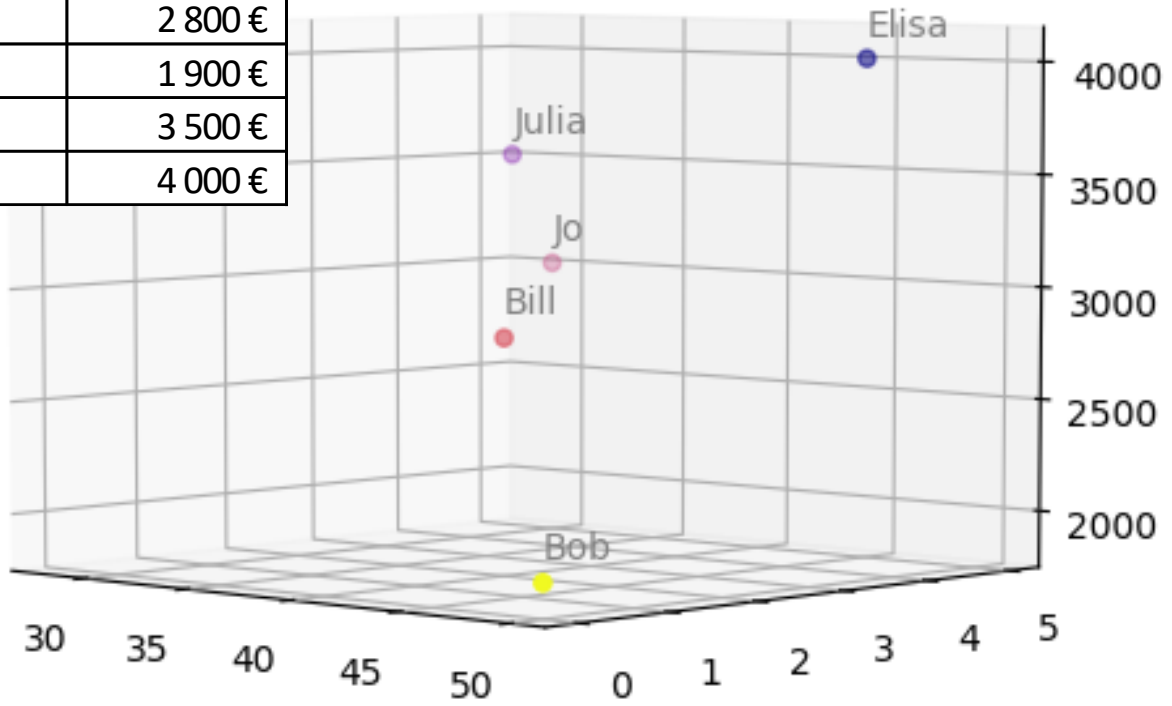
Affichage des données 2D

Nom	Age	Salaire
Jo	30	3 000 €
Bill	40	2 800 €
Bob	50	1 900 €
Julia	28	3 500 €
Elisa	45	4 000 €



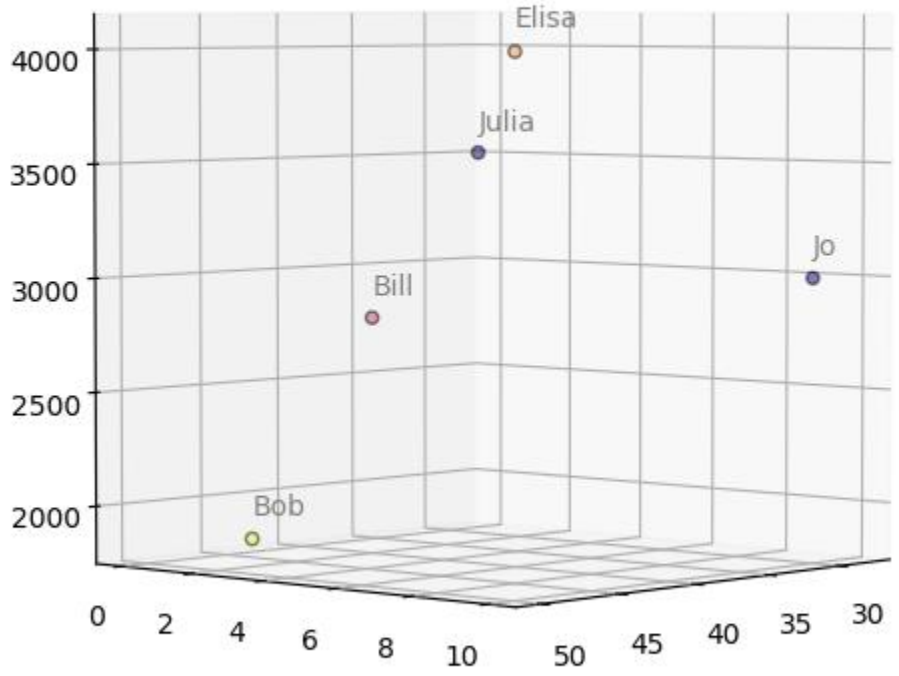
Affichage des données 3D

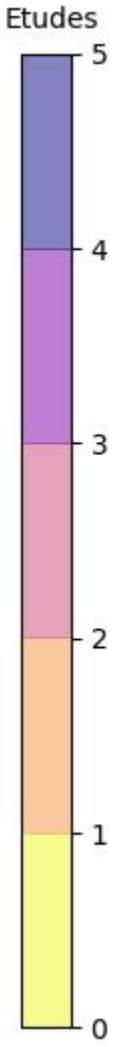
Nom	Age	Années études	Salaire
Jo	30	4	3 000 €
Bill	40	2	2 800 €
Bob	50	0	1 900 €
Julia	28	5	3 500 €
Elisa	45	1	4 000 €



Gagner une dimension en utilisant un code couleur

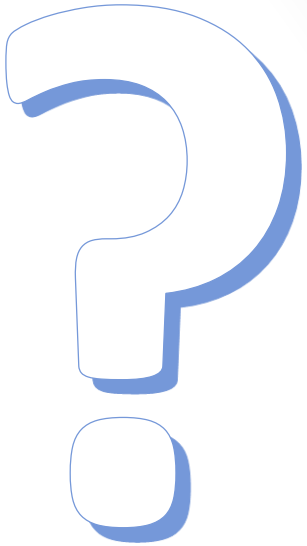
Nom	Age	Années études	Personnes gérées	Salaire
Jo	30	4	10	3 000 €
Bill	40	2	2	2 800 €
Bob	50	0	3	1 900 €
Julia	28	5	0	3 500 €
Elisa	45	1	10	4 000 €





Et Après ?

Nom	Age	Années études	Personnes gérées	Ancienneté	Salaire
Jo	30	4	10	2	3 000 €
Bill	40	2	2	5	2 800 €
Bob	50	0	3	8	1 900 €
Julia	28	5	0	4	3 500 €
Elisa	45	1	10	10	4 000 €

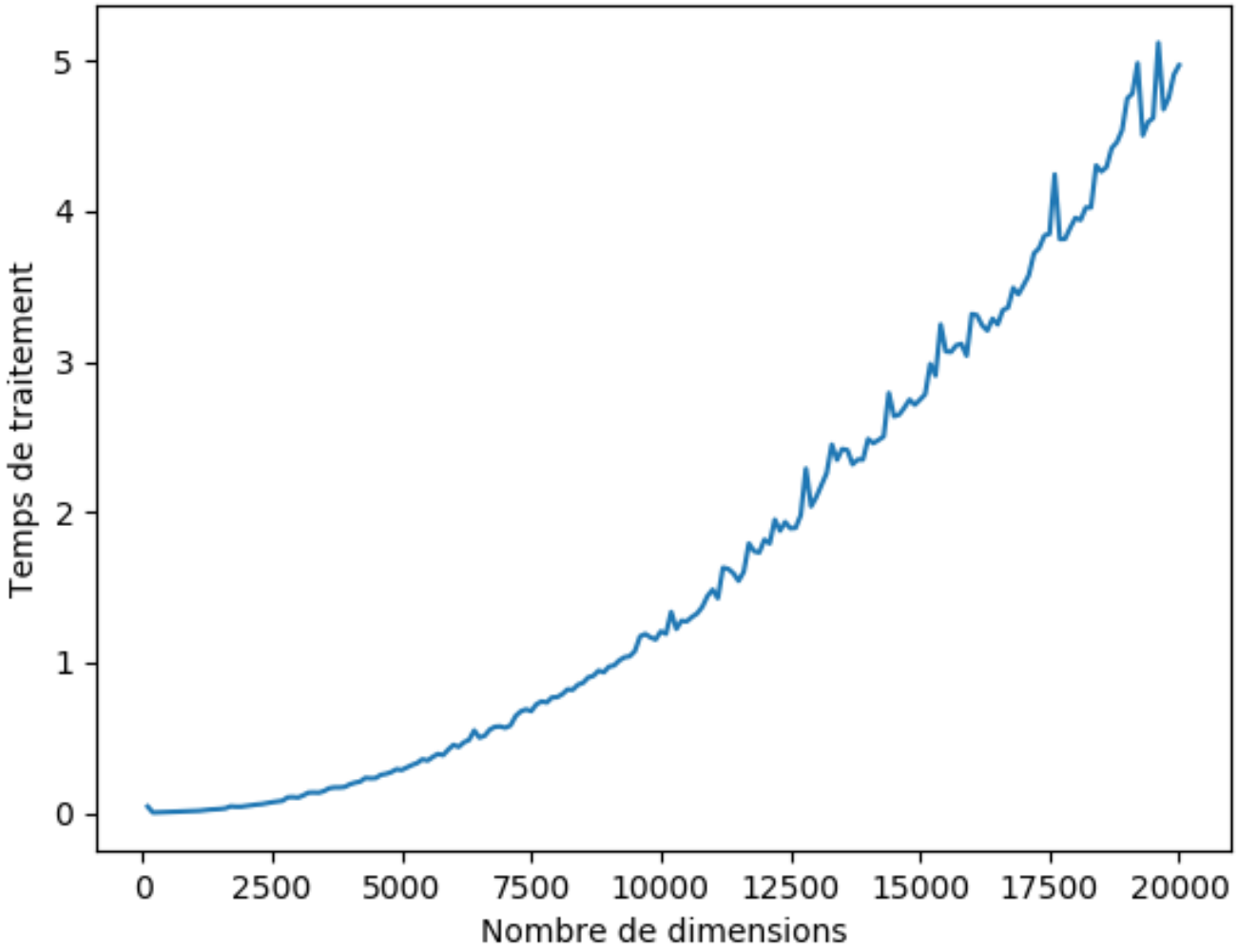


Solution : Construire des nouvelles variables qui permettent de réduire le nombre de dimensions tout en perdant un minimum d'information.

Avantage : On peut visualiser en 2D ou 3D les points proches, on peut aussi réduire le nombre de dimensions, et donc le temps de calcul, pour l'apprentissage.

Inconvénient : Les variables n'ont plus vraiment de sens.

La « Malédiction des dimensions »



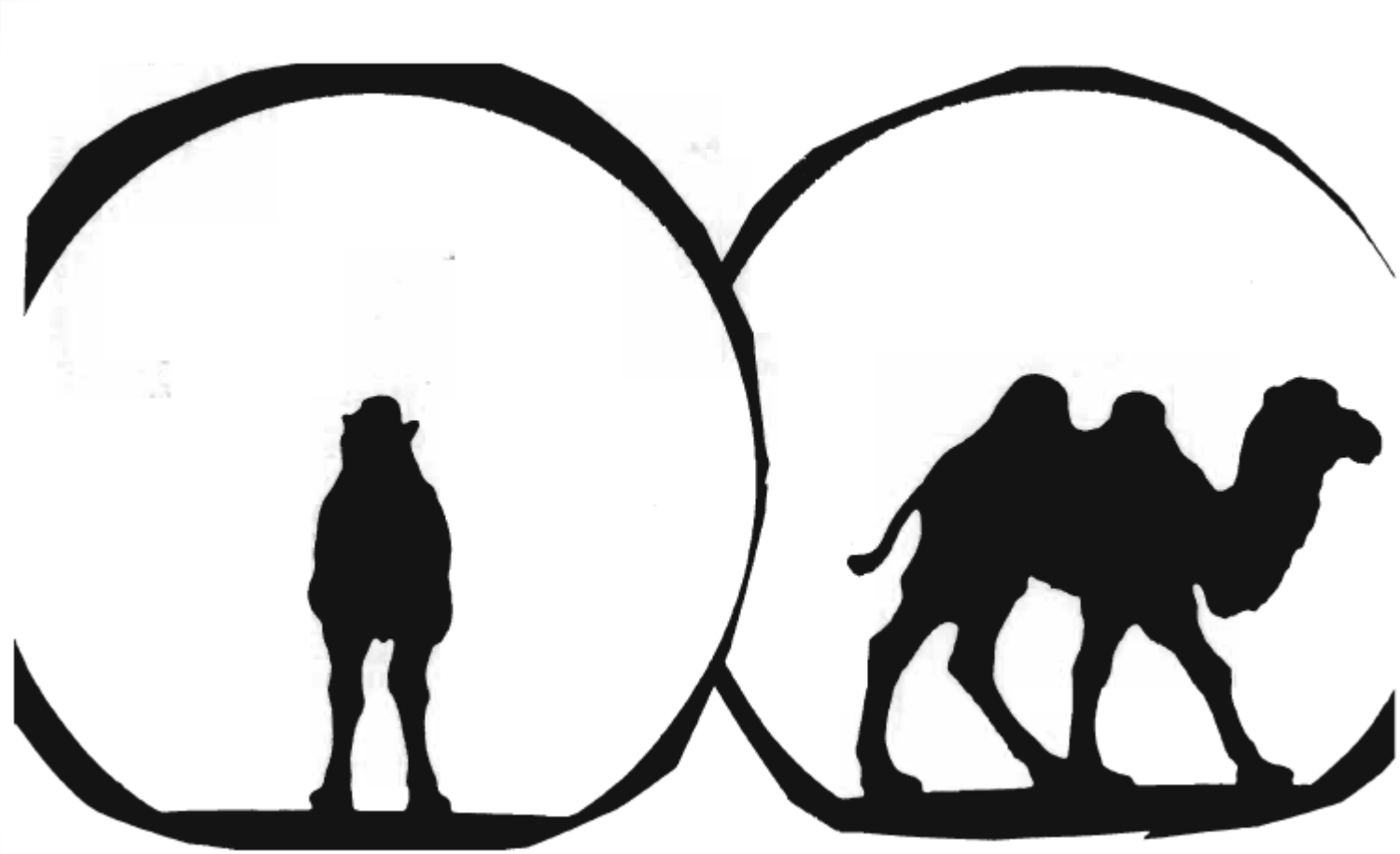
EPSI l'École d'ingénierie informatique

Analyse en composantes principales (ou PCA)

Jean-Claude Houbart

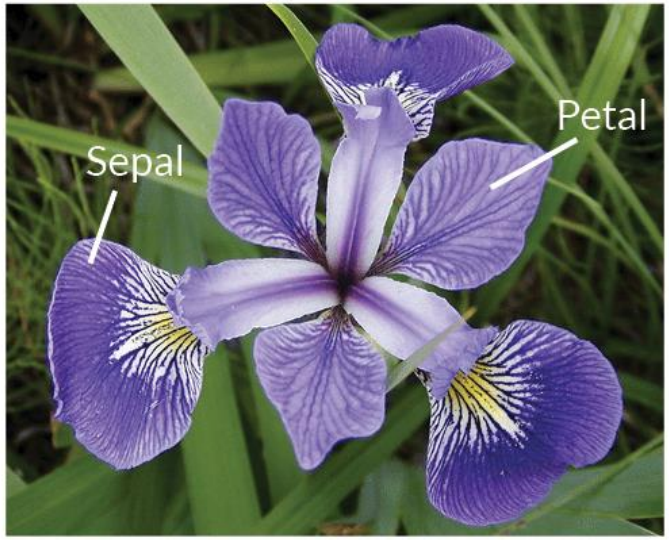
houbart.jc@gmail.com

Trouver la projection qui explique le mieux



Créé en 1901

Le Dataset Iris

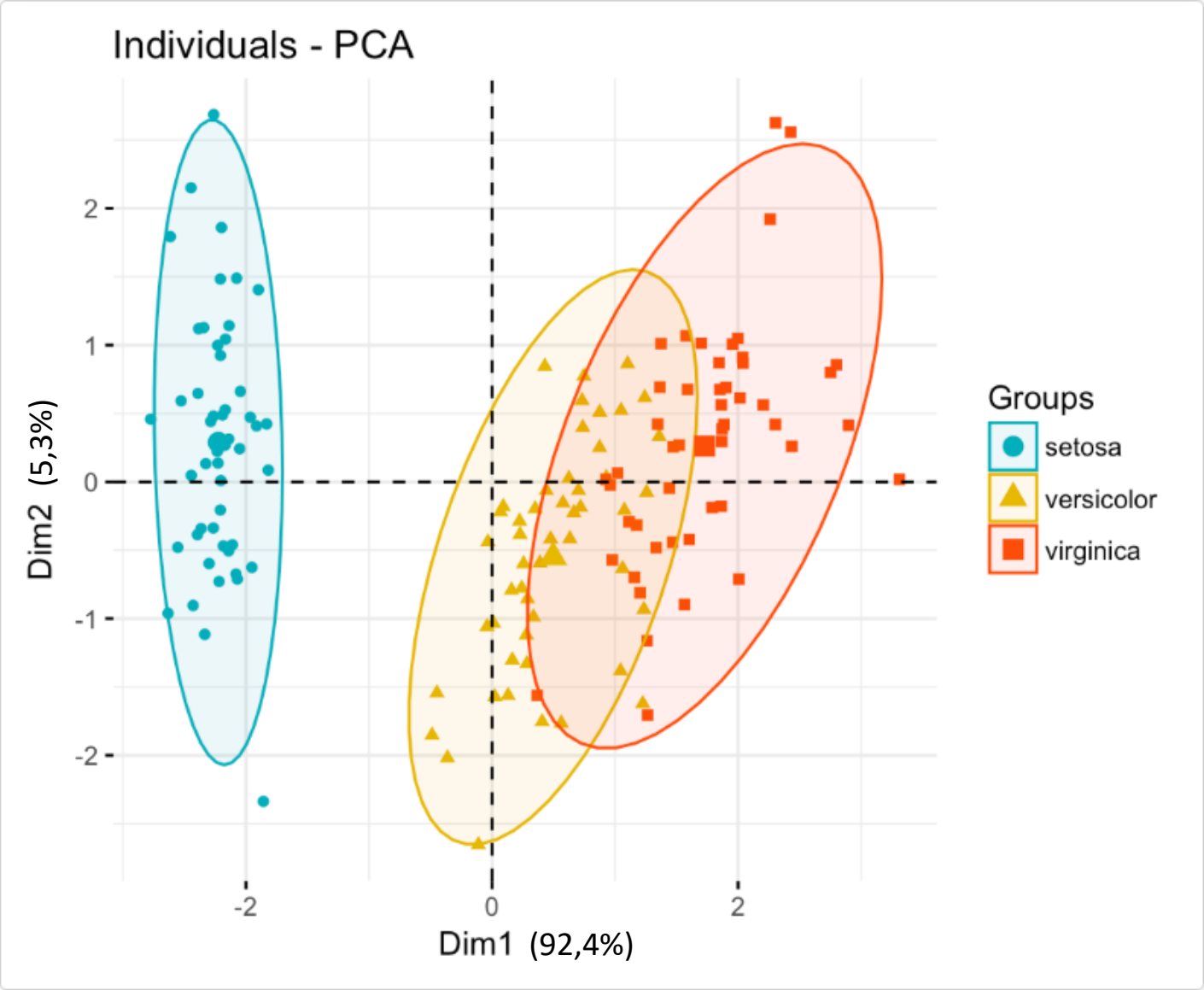


Iris Versicolor **Iris Setosa** **Iris Virginica**

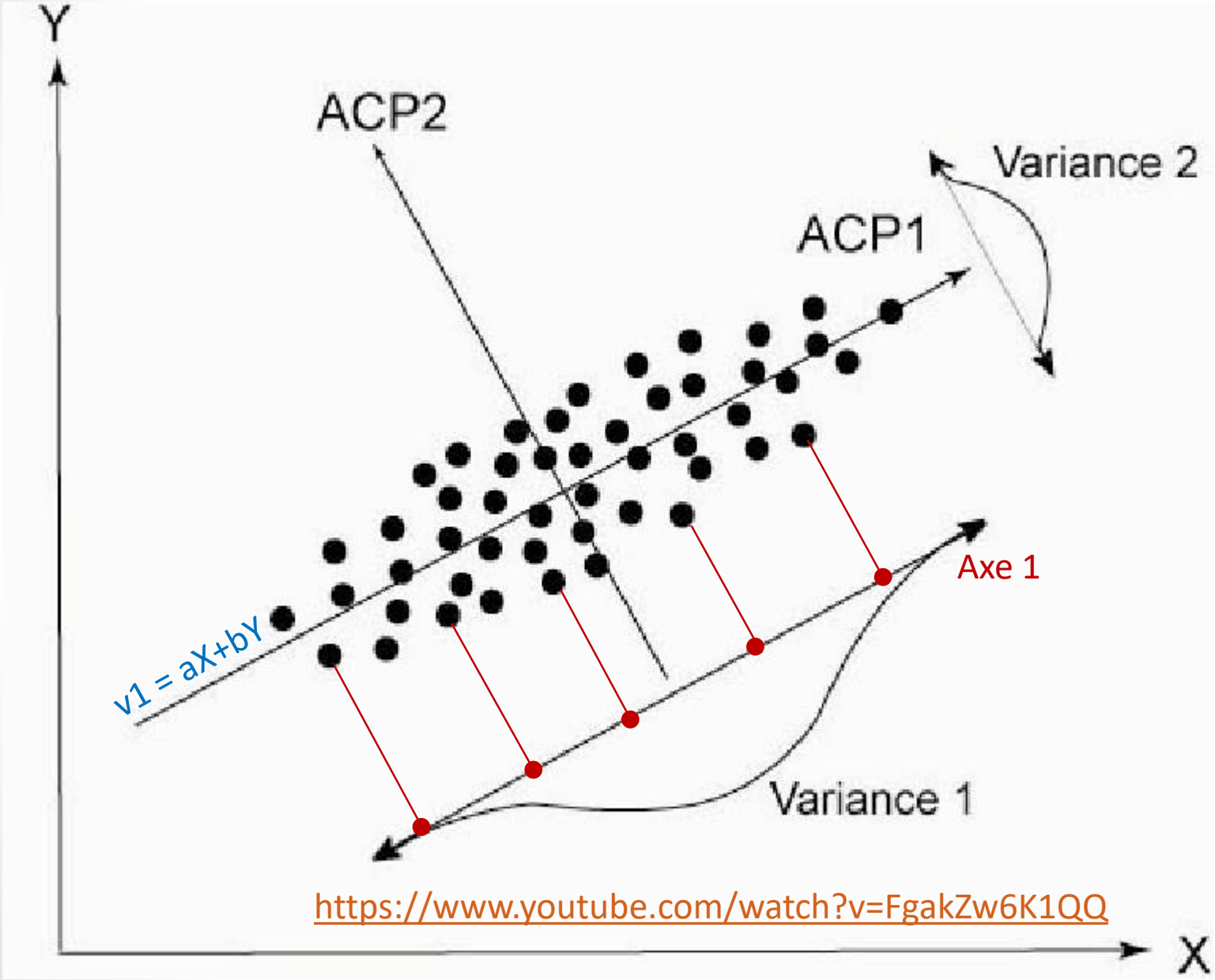
	sepal length	sepal width	petal length	petal width	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

- 50 exemples pour chaque type d'Iris.
- Disponible dans Scikit_Learn en utilisant la fonction load_iris() du package **sklearn.datasets**.

4D vers 2D


Individuals - PCA
Dim2 (5,3%)
Dim1 (92,4%)
2
1
0
-1
-2
-2
0
2
Groups
setosa
versicolor
virginica

Maximiser la variance



https://www.youtube.com/watch?v=FgakZw6K1QQ

EPSI l'École d'ingénierie informatique

T-SNE
t-distributed stochastic neighbor embedding

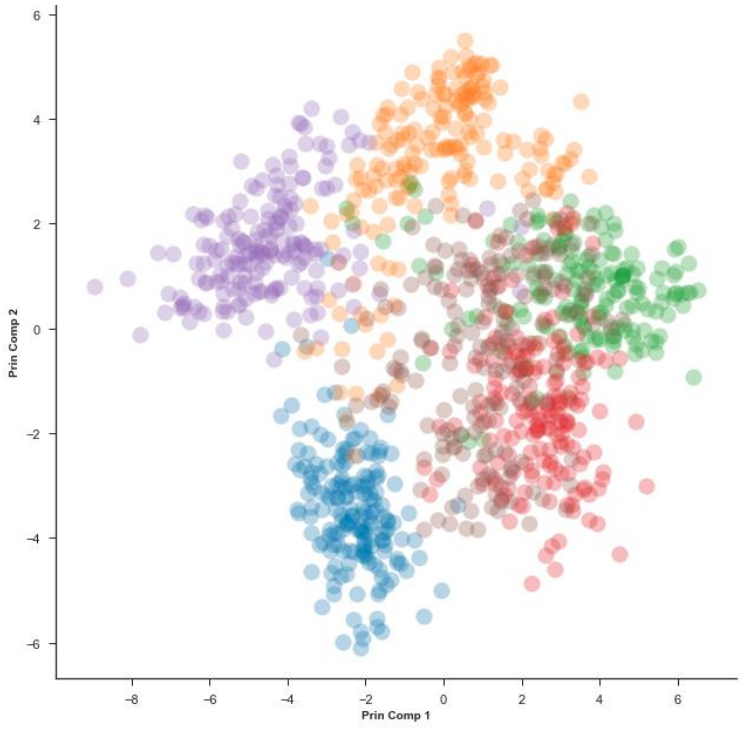
Jean-Claude Houbart

houbart.jc@gmail.com

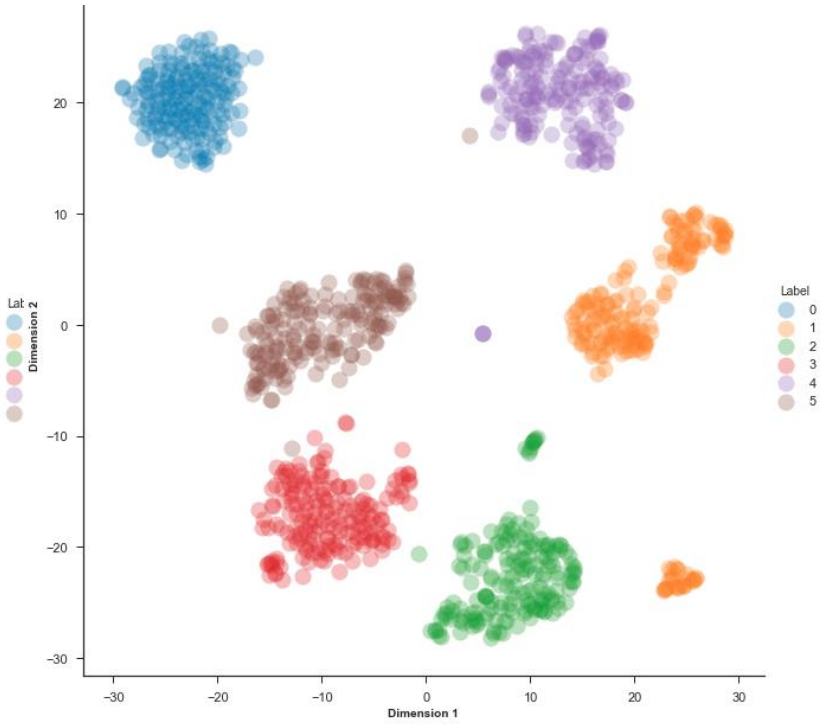
T-SNE

- Une méthode non-linéaire imaginée en 2008.
- Représentation d'un ensemble de points d'un espace à grande dimension dans un espace de deux ou trois dimensions.

PCA

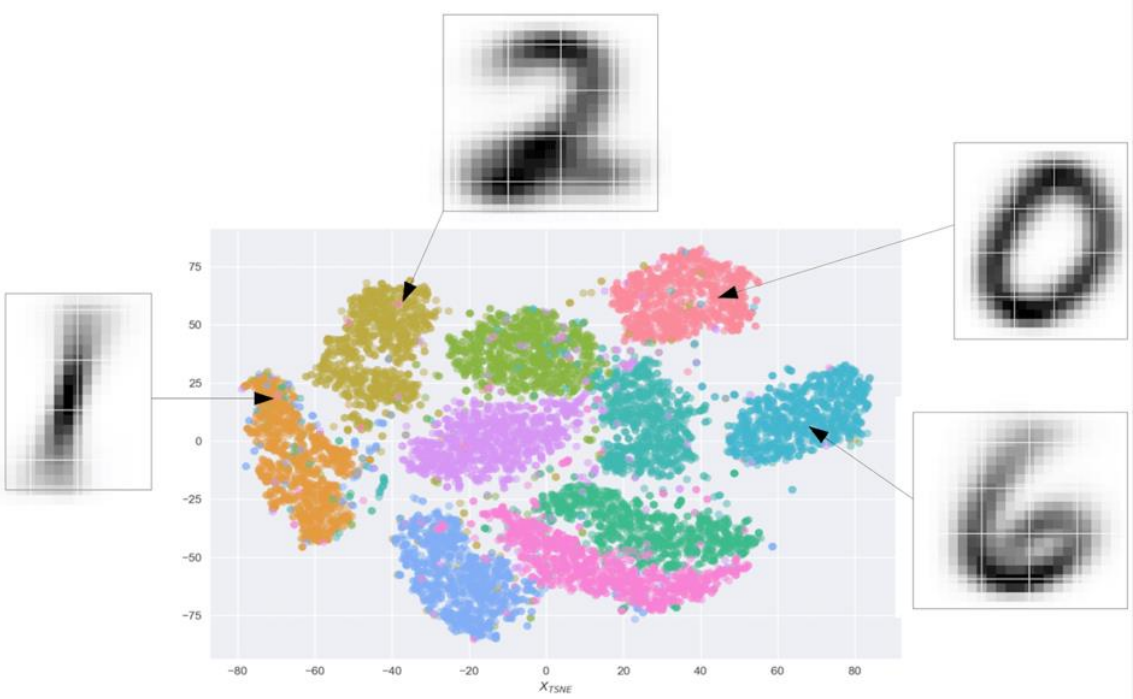


T-SNE



T-SNE

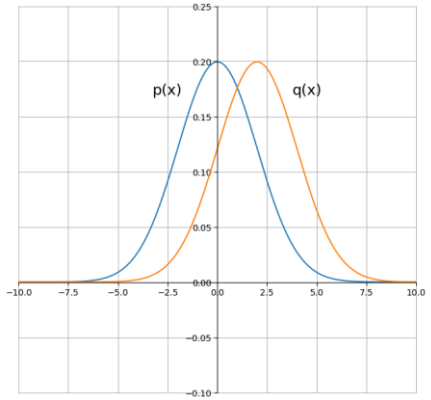
- Deux points qui sont proches (éloignés) dans l'espace d'origine devront être proches (éloignés) dans l'espace de faible dimension.
- Forte Complexité algorithmique $o(n^2)$

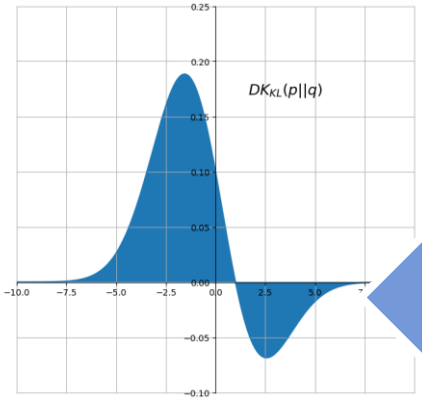


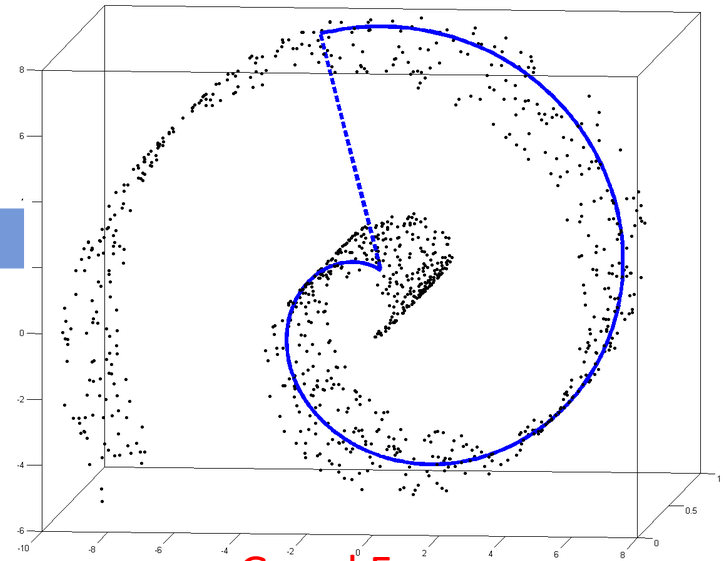
T-SNE on MNIST handwritten digits dataset

Respect statistique des distances

Pour chaque couple de points, la distance est transformé en probabilité (similaire à la fonction softmax) ? Un variance est introduite la perplexité, qui va correspondre à la chute de la probabilité avec la distance.

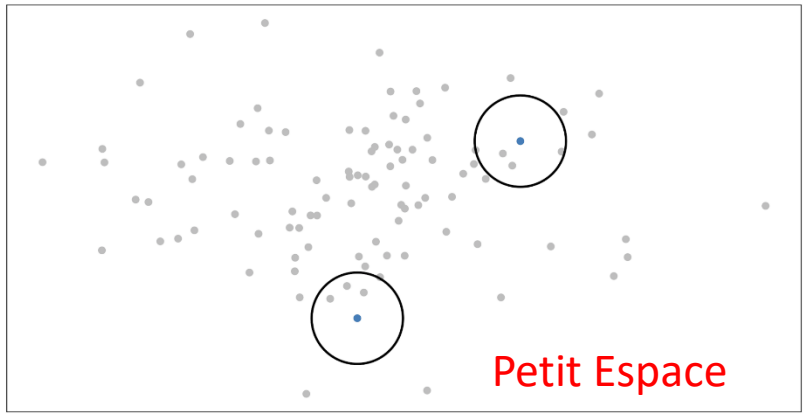






Grand Espace

Minimiser la divergence de Kullback-Leibler entre les deux distribution de probabilité.



Petit Espace

En utilisant la descente de gradient, avec un Learning Rate epsilon , on va faire converger les distributions des points du petit espace vers celle du grand espace.

https://distill.pub/2016/misread-tsne/

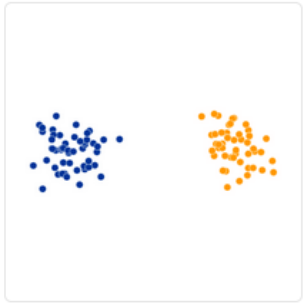
Perplexité dans t-sne

Pour chaque point quelle est la probabilité pour un autre point d'être un voisin ?
On utilise une loi normale dont l'écart type basé sur le nombre moyen de voisins perplexité

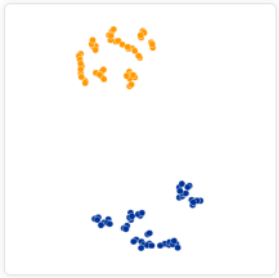
=> Une perplexité trop faible surestime les variations locales.

=> Une perplexité trop importante donne un résultat aléatoire (surtout si supérieur au nombre de points).

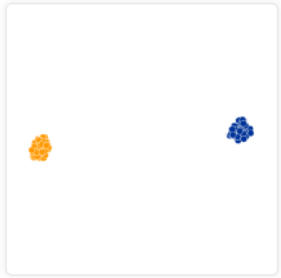
La valeur préconisée est entre 5 et 50.



Original

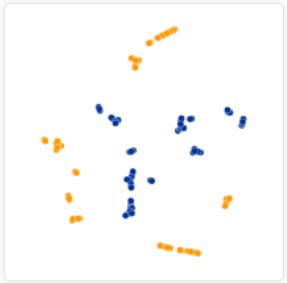


Perplexity: 5
Step: 5,000



Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000



Perplexity: 2
Step: 5,000

Perplexity: 100
Step: 5,000

EPSI l'École d'ingénierie informatique

Clustering

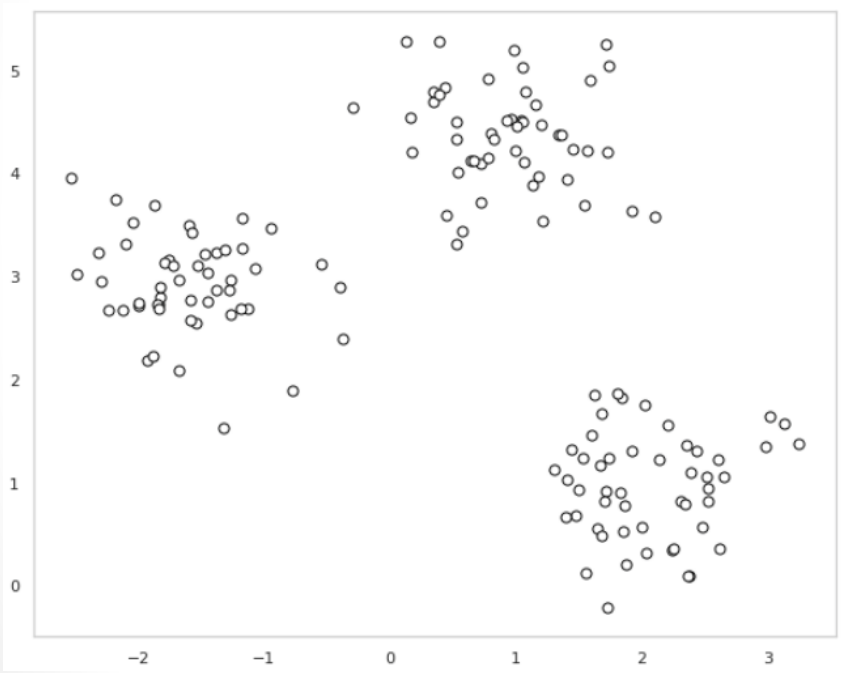
Jean-Claude Houbart

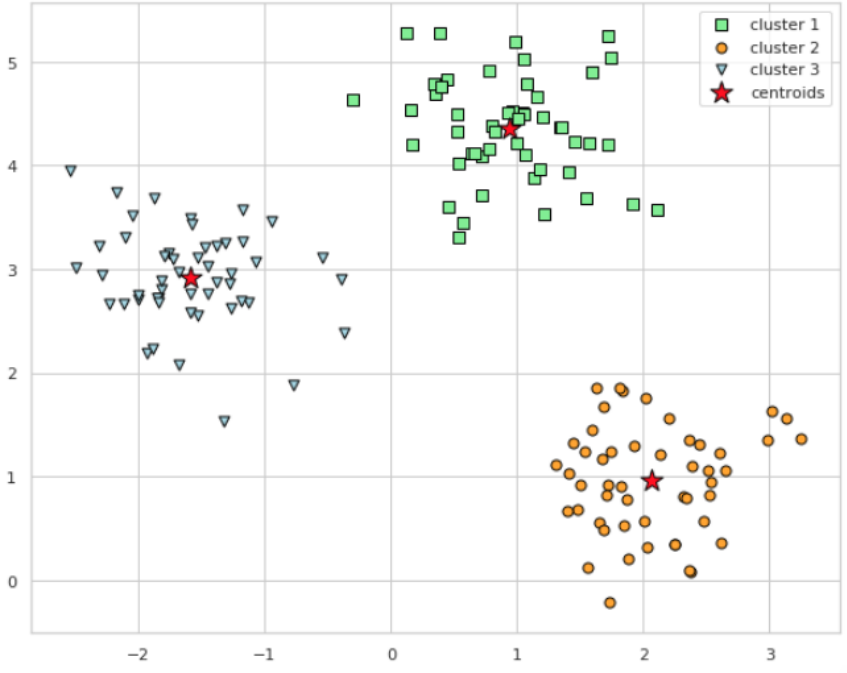
houbart.jc@gmail.com

Clustering

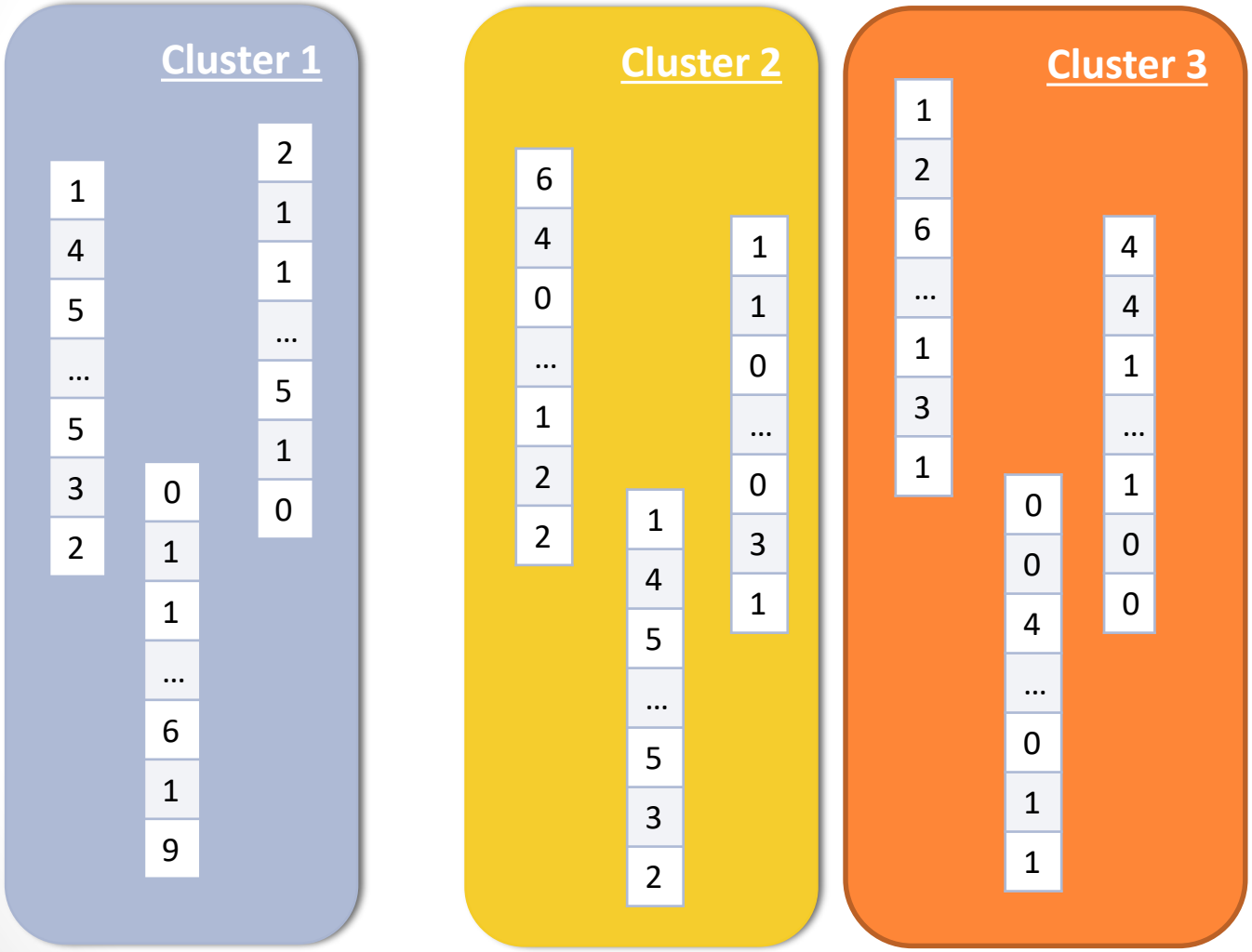
Regroupement les données (en N dimensions) en un nombre quelconque de clusters.

Peut souvent précéder une réduction de dimension.





Regroupement de vecteurs



KMeans

L'algorithme le plus populaire pour faire du clustering est le KMeans.

KMeans est un apprentissage non supervisé.

Pour faire K clusters, on prend K points (centroïds) au hasard et on itère sur 2 étapes :

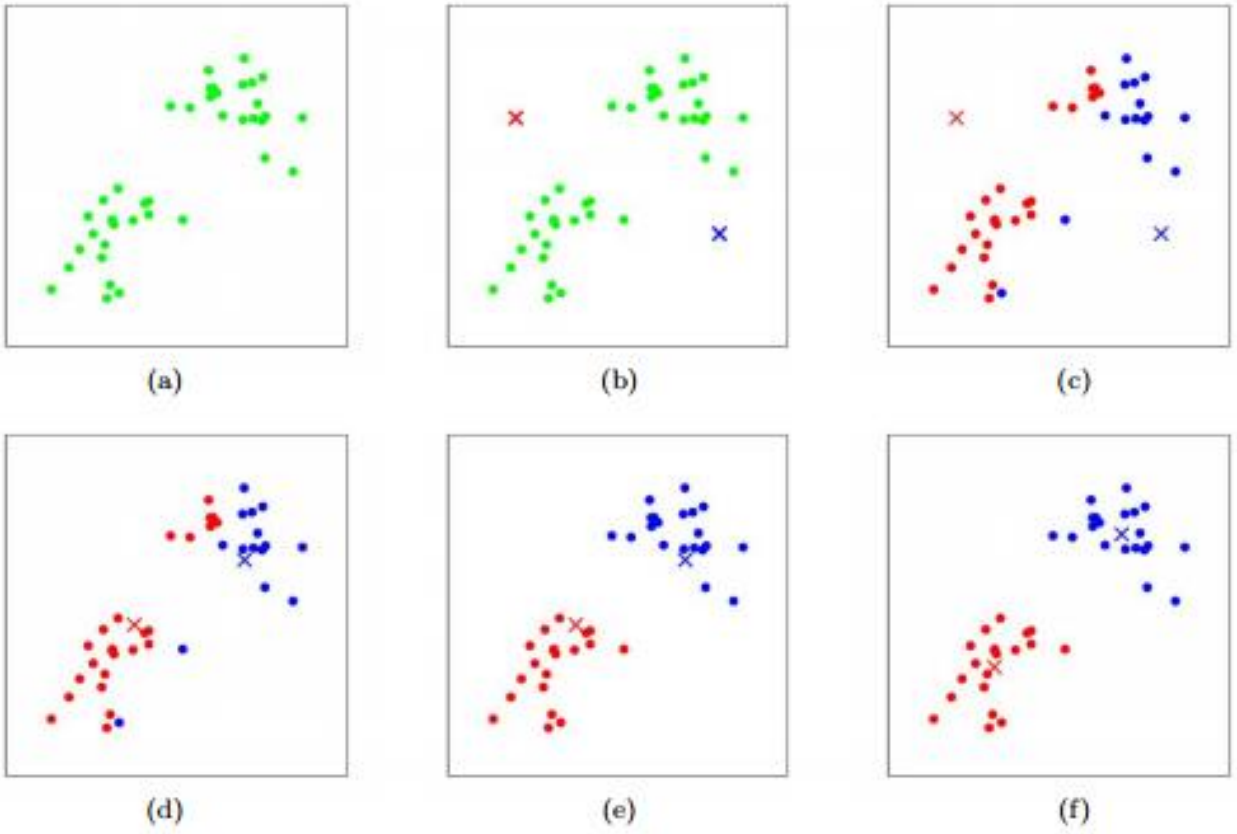
1. On associe les points les plus proches pour chaque centroïd
2. On déplace les centroïds au centre des points qui lui étaient associés.

... et on continue jusqu'à ce que les centroïdes ne bougent plus.

Si à un moment, un centroïd se retrouve sans point, alors soit on l'élimine (le + courant) soit on le déplace aléatoirement.

Kmeans/Illustration

2 clusters, 2 itérations



EPSI l'École d'ingénierie informatique

Codage avec SKLearn

Jean-Claude Houbart

houbart.jc@gmail.com

sklearn.cluster.KMeans

class sklearn.cluster.KMeans(*n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='auto'*)

fit(self, X[, y, sample_weight])	Compute k-means clustering.
fit_predict(self, X[, y, sample_weight])	Compute cluster centers and predict cluster index for each sample.
fit_transform(self, X[, y, sample_weight])	Compute clustering and transform X to cluster-distance space.
get_params(self[, deep])	Get parameters for this estimator.
predict(self, X[, sample_weight])	Predict the closest cluster each sample in X belongs to.
score(self, X[, y, sample_weight])	Opposite of the value of X on the K-means objective.
set_params(self, **params)	Set the parameters of this estimator.
transform(self, X)	Transform X to a cluster-distance space.

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html?highlight=kmean#sklearn.cluster.KMeans

sklearn.decomposition.PCA

class sklearn.decomposition.PCA(n_components=None, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', random_state=None)

fit(self, X[, y])	Fit the model with X.
fit_transform(self, X[, y])	Fit the model with X and apply the dimensionality reduction on X.
get_covariance(self)	Compute data covariance with the generative model.
get_params(self[, deep])	Get parameters for this estimator.
get_precision(self)	Compute data precision matrix with the generative model.
inverse_transform(self, X)	Transform data back to its original space.
score(self, X[, y])	Return the average log-likelihood of all samples.
score_samples(self, X)	Return the log-likelihood of each sample.
set_params(self, **params)	Set the parameters of this estimator.
transform(self, X)	Apply dimensionality reduction to X.

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html?highlight=pca#sklearn.decomposition.PCA

sklearn.manifold.TSNE

class sklearn.manifold.TSNE(*n_components=2, *, perplexity=30.0, early_exaggeration=12.0, learning_rate='warn', n_iter=1000, n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean', init='warn', verbose=0, random_state=None, method='barnes_hut', angle=0.5, n_jobs=None, square_distances='legacy'*)

fit(self, X[, y])	Fit X into an embedded space.
fit_transform(self, X[, y])	Fit X into an embedded space and return that transformed output.
get_params(self[, deep])	Get parameters for this estimator.
set_params(self, **params)	Set the parameters of this estimator.

https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html?highlight=t%20sne#sklearn.manifold.TSNE