

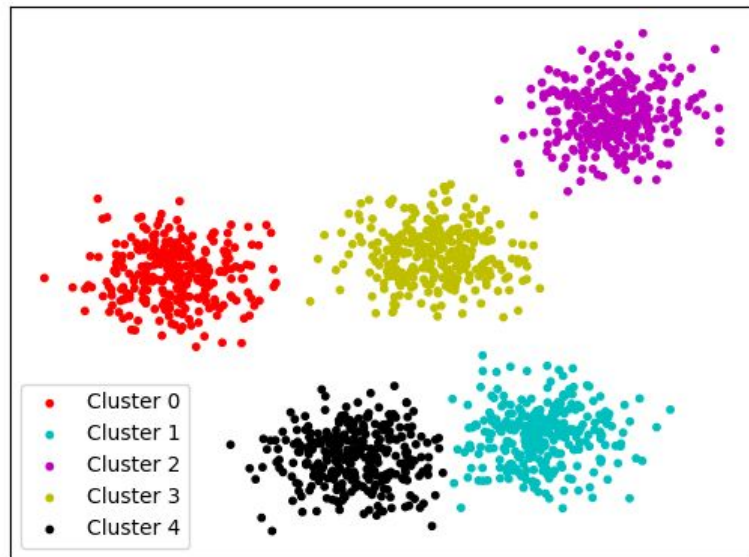
Clustering & Réduction de la dimensionnalité

Pereg Hergoualc'h - Baptiste Le Goff

Le clustering

Le **clustering** est une méthode d'apprentissage automatique qui consiste à regrouper des points de données par similarité ou par distance.

C'est une méthode d'apprentissage **non supervisée** et une technique populaire d'analyse statistique des données. Pour un ensemble donné de points, vous pouvez utiliser des algorithmes de classification pour classer ces points de données individuels dans des groupes spécifiques.



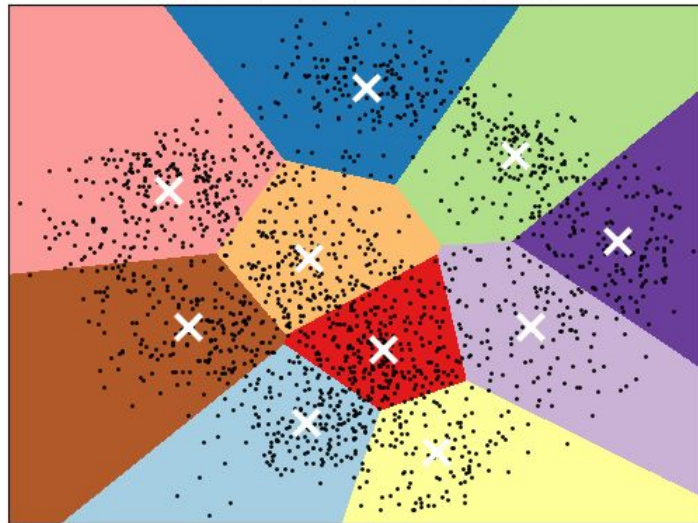
Algorithme de Kmeans

Le but est de regrouper les données en essayant de séparer les échantillons en n groupes d'égale variance, minimisant un critère connu sous le nom d'**inertie** ou **somme des carrés intra-cluster**.

Il nécessite que le nombre de clusters soit spécifié. Il s'adapte bien à un grand nombre d'échantillons et a été utilisé dans une large gamme de domaines d'application dans de nombreux domaines différents.

$$\sum_{i=0}^n \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross



Réduction de la dimensionnalité

La **réduction de la dimensionnalité** est un processus étudié en mathématiques et en informatique, qui consiste à prendre des données dans un espace de grande dimension, et à les remplacer par des données dans un espace de plus petite dimension. Pour que l'opération soit utile il faut que les données en sortie représentent bien les données d'entrée.

Pour réduire la dimension, on peut agir sur deux leviers :

1. Supprimer des dimensions (ou descripteurs)
2. Combiner les variables afin d'obtenir un plus petit nombre de nouvelles variables moins redondantes



PCA et t-SNE

- PCA

Pour de nombreuses applications d'apprentissage automatique, il est utile de pouvoir visualiser les données. Visualiser des données en 2 ou 3 dimensions n'est pas si difficile. Par exemple, le jeu de données Iris est en 4 dimensions. Nous pouvons donc utiliser PCA pour réduire ces données de 4 dimensions à 2 ou 3 dimensions afin de pouvoir tracer et mieux comprendre les données.



```
from sklearn.decomposition import PCA
```

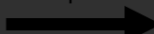
```
pca = PCA(n_components=2)
```

```
principalComponents = pca.fit_transform(x)
```

```
principalDf = pd.DataFrame(data = principalComponents,  
                           columns = ['principal component 1', 'principal component 2'])
```

	sepal length	sepal width	petal length	petal width
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

PCA
(2 components)



	principal component 1	principal component 2
0	-2.264542	0.505704
1	-2.086426	-0.655405
2	-2.367950	-0.318477
3	-2.304197	-0.575368
4	-2.388777	0.674767

- t-SNE

C'est un outil de visualisation de données de grande dimension. Il convertit les similitudes entre les points de données en probabilités conjointes et tente de minimiser la divergence de Kullback-Leibler entre les probabilités conjointes de l'inclusion de faible dimension et les données de haute dimension. t-SNE a une fonction de coût qui n'est pas convexe, c'est-à-dire qu'avec différentes initialisations, nous pouvons obtenir des résultats différents.

Il est fortement recommandé d'utiliser une autre méthode de réduction de dimensionnalité (par exemple PCA pour les données denses) pour réduire le nombre de dimensions à un montant raisonnable (par exemple 50) si le nombre de features est très élevé. Cela supprimera certains bruits et accélèrera le calcul des distances par paires entre les échantillons



```
from sklearn.manifold import TSNE
```

```
tsne_em = TSNE(n_components=2, perplexity=30.0, n_iter=1000, verbose=1).fit_transform(df)
```

