Clustering

Le clustering est une méthode non supervisée qui vise à répartir les données selon leur similarité dans des clusters. L'inconnu de ce type de problèmes est le nombre de clusters.

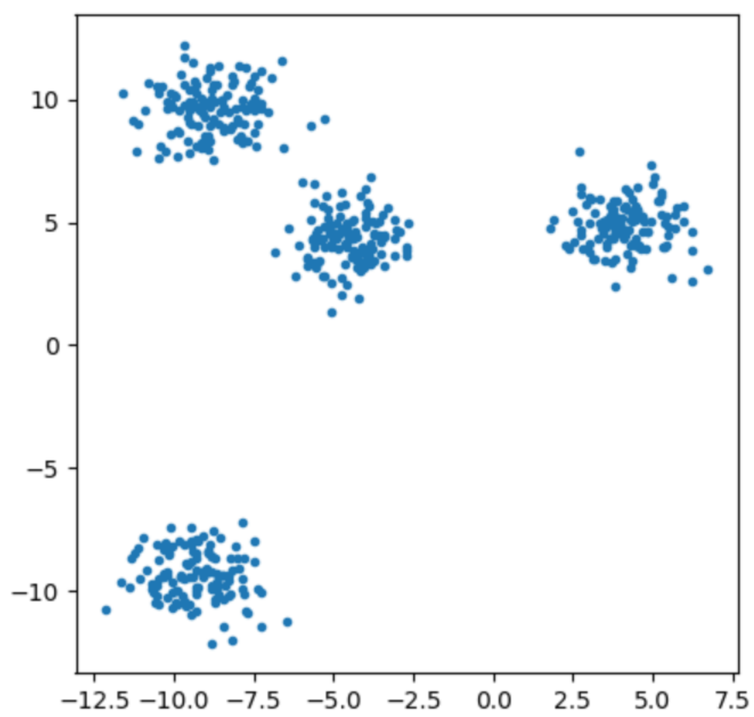
Principe

On commence par générer un nuage de points artificiel.

```
from sklearn.datasets import make_blobs
X, Y = make_blobs(n_samples=500, n_features=2, centers=4)
```

On représente ces données.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 1, figsize=(5, 5))
ax.plot(X[:, 0], X[:, 1], '.')
```



On utilise un algorithme très utilisé : KMeans.

```
from sklearn.cluster import KMeans
km = KMeans()
km.fit(X)
```

L'optimisation du modèle produit autant de points que de clusters.

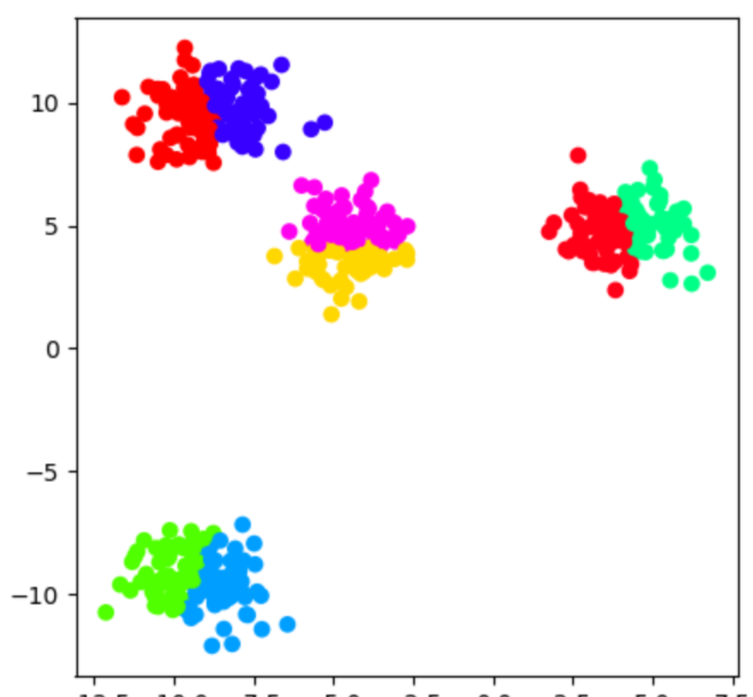
```
print(km.cluster_centers_)
```

Out:

```
[[-4.62432952  3.50843322]
 [-9.92513665 -8.91021625]
 [ 3.43244388  4.71641361]
 [-7.85519744  9.80587987]
 [ 4.94563993  5.04322209]
 [-9.65910309  9.42679338]
 [-4.43175903  5.04537728]
 [-8.56127946 -9.91964001]]
```

On dessine le résultat en choisissant une couleur différente pour chaque cluster.

```
cmap = plt.cm.get_cmap("hsv", km.cluster_centers_.shape[0])
fig, ax = plt.subplots(1, 1, figsize=(5, 5))
colors = [cmap(i) for i in km.fit_predict(X)]
ax.scatter(X[:, 0], X[:, 1], c=colors)
```



−12.5 −10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5

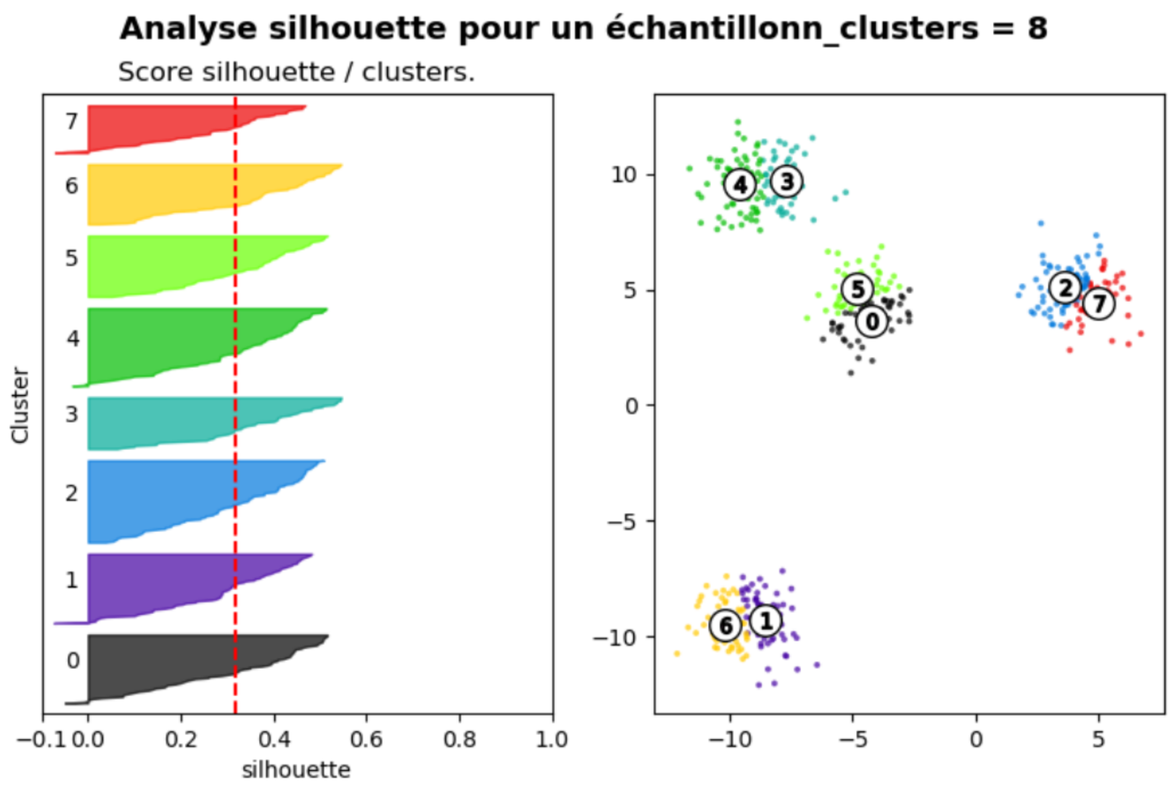
Autre graphe et métrique silhouette

Inspiré de Selecting the number of clusters with silhouette analysis on KMeans clustering. Il s'agit de représenter la dispersion au sein de chaque cluster. Sont-ils concentrés autour d'un point ou plutôt regroupés parce que loin de tout ? On commence par calculer le score silhouette puis à prendre un échantillon aléatoire sous peine d'avoir un graphique surchargé.

```
from sklearn.metrics import silhouette_samples, silhouette_score
cluster_labels = km.fit_predict(X)
silhouette_avg = silhouette_score(X, cluster_labels)
sample_silhouette_values = silhouette_samples(X, cluster_labels)
centers = km.cluster_centers_
n_clusters = centers.shape[0]
```

On dessine.

```
import matplotlib.cm as cm
import numpy

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.set_size_inches(9, 5)

ax1.set_xlim([-0.1, 1])
ax1.set_ylim([0, len(X) + (km.n_clusters + 1) * 10])

y_lower = 10
for i in range(n_clusters):
    ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]
    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cm.spectral(float(i) / n_clusters)
    ax1.fill_betweenx(numpy.arange(y_lower, y_upper),
                      0, ith_cluster_silhouette_values,
                      facecolor=color, edgecolor=color, alpha=0.7)

    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
    y_lower = y_upper + 10

ax1.set_title("Score silhouette / clusters.")
ax1.set_xlabel("silhouette")
ax1.set_ylabel("Cluster")

ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

ax1.set_yticks([])
ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

colors = cm.spectral(cluster_labels.astype(float) / n_clusters)
ax2.scatter(X[:, 0], X[:, 1], marker='.', s=30,
            lw=0, alpha=0.7, c=colors, edgecolor='k')
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=200, edgecolor='k')

for i, c in enumerate(centers):
    ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                s=50, edgecolor='k')

plt.suptitle(("Analyse silhouette pour un échantillon"
              "n_clusters = %d" % n_clusters),
             fontsize=14, fontweight='bold')
```



Total running time of the script: (0 minutes 1.547 seconds)

Download Python source code: plot_clustering.py

Download Jupyter notebook: plot_clustering.ipynb

Gallery generated by Sphinx-Gallery

Source

Back to top

Mis à jour le 2018-03-07.