Politecnico di Torino

1859

Data Science and Machine Learning for Engineering Applications

Scikit-learn Clustering

DataBase and Data Mining Group

Salvatore Greco
Andrea Pasini
Flavio Giobergia
Elena Baralis
Tania Cerquitelli

- Scikit-learn
  - Machine learning library built on **NumPy**, **SciPy** and **Matplotlib**
- What Scikit-learn can do
  - **Unsupervised** learning
    - Clustering
  - **Supervised** learning
    - Regression, classification
  - Data **preprocessing**
    - Feature extraction, feature selection, dimensionality reduction

- What Scikit-learn **cannot** do
  - Distributed computation on multiple computers
    - Only multi-core optimization
  - Deep learning
    - Use Torch, Keras and Tensorflow instead

- Scikit learn models work with structured data
  - Data must be in the form of **2D Numpy arrays**
    - Rows represent the **samples**
    - Columns represent the **attributes (or features)**
  - This table is called **features matrix**

shape = (3, 3)

	Price	Quantity	Liters
Sample 1	1.0	5	1.5
Sample 2	1.4	10	0.3
Sample 3	5.0	8	1

- Features can be
  - **Real** values
  - **Integer** values to represent categorical data
- If you have strings in your data, you first have to convert them to integers (**preprocessing**)

Input data

1.0	**January**	1.5
1.4	**February**	0.3
5.0	**March**	1

→

Features matrix

1.0	**0**	1.5
1.4	**1**	0.3
5.0	**2**	1

- Also **missing values** must be solved before applying any model
  - With imputation or by removing rows

Input data

1.0	0.5	1.5
1.4	**NaN**	0.3
5.0	0.5	1

→

Features matrix

1.0	0.5	1.5
1.4	**0.5**	0.3
5.0	0.5	1

Input data

1.0	0.5	1.5
1.4	**NaN**	0.3
5.0	0.5	1

→

Features matrix

1.0	0.5	1.5
5.0	0.5	1

- For **unsupervised** learning you only need the features matrix
- For **supervised** learning you also need a **target** array to train the model
  - It is typically one-dimensional, with length n_samples
    - May be 2-dimensional for multi-output models

Features matrix
shape = (n_samples, n_features)

1.0	5	1.5
1.4	10	0.3
5.0	8	1

Target array
shape = (n_samples,)

A
A
B

- The target array can contain
  - Integer values, each corresponding to a class label

Target labels		Target array
Dog	→	0
Dog		0
Cat		1

  - Real values for regression

Target array
0.4
1.8
-6.9

- Scikit-learn estimator API
  - All models are represented with Python classes
  - Their classes include
    - The values of the **hyperparameters** used to configure the model
    - The values of the **parameters** learned after training
      - By convention these attributes end with an underscore
    - The **methods** to train the model and make inference
  - Scikit-learn models are provided with sensible **defaults** for the hyperparameters

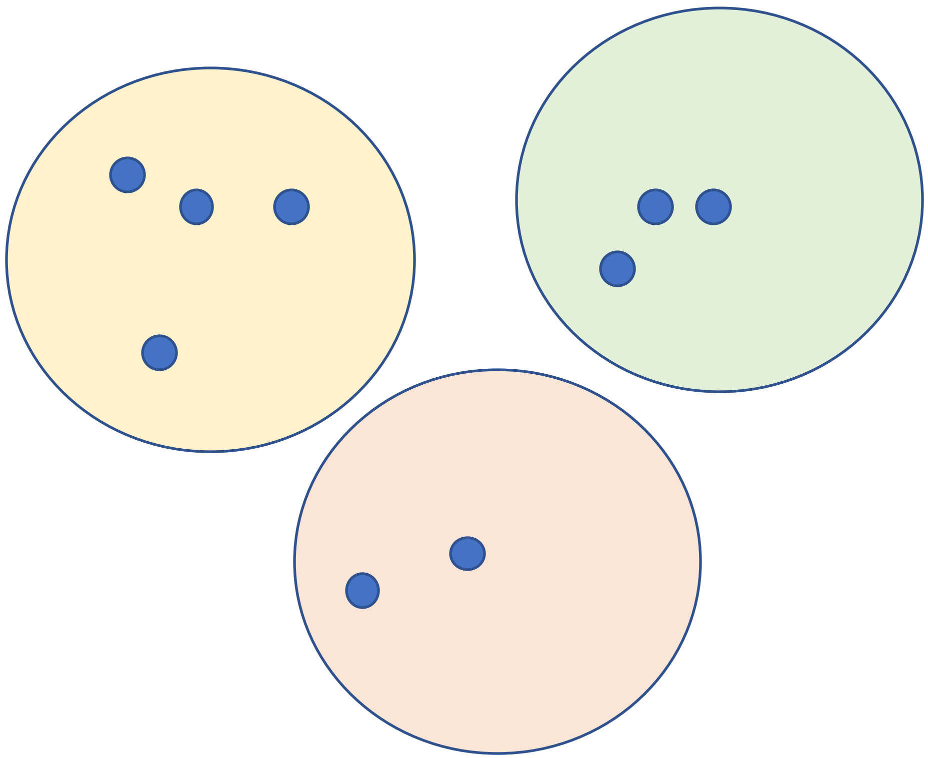
- Scikit learn models follow a simple, shared **pattern**

1. **Import** the model that you need to use
2. **Build** the model, setting its hyperparameters
3. **Train** model parameters on your data
   - Using the fit() method
4. **Use** the model to make predictions
   - Using the predict()/transform() methods

- Sometimes fit and predict/transform are implemented within the same class method

- **fit():** learn model parameters from input data
  - E.g. train a classifier
- **predict():** apply model parameters to make predictions on data
  - E.g. predict class labels
- **transform()**: transform data into a different representation
  - E.g. normalize test data
- **fit_predict():** fit model and make predictions
  - E.g. apply clustering to data
- **fit_transform():** fit model and transform data
  - E.g. apply PCA to transform data

- **Unsupervised** technique that analyzes the data distribution to generate N partitions
  - Unsupervised = it only requires a features matrix



- Import a model

```
from sklearn.cluster import KMeans
```

- Build model object

```
km = KMeans(n_clusters = 5)
```

  - The hyperparameter **n_clusters** specifies the number of centroids (= number of clusters)
    - Default is 8 (buy may change across different library versions)

- Apply clustering to input data (Numpy)

```
In [1]: y_pred = km.fit_predict(X)
```

```
Out[1]: [3, 1, 1, 1, 2, 2, 0]
```

- This operation assigns data to their respective cluster
  - X is the 2D NumPy array with input features (**features matrix**)
  - y_pred is a 1D array with cluster labels

1.0	5	1.5
1.4	10	0.3
…	…	…

→

3
1
1

- Apply clustering to input data (Pandas)

```
In [1]: y_pred = km.fit_predict(df)
```

```
Out[1]: [3, 1, 1, 1, 2, 2, 0]
```

- This operation assigns data to their respective cluster
  - df is the 2D DataFrame with input features (**features matrix**)
  - y_pred is a 1D array with cluster labels

	c1	c2	c3
Sample 1	1.0	5	1.5
Sample 2	1.4	10	0.3
Sample 3	5.0	8	1

→

3
1
1

Clustering

- Example: DBSCAN

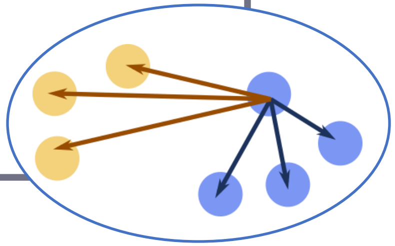
```
from sklearn.cluster import DBSCAN
cl_alg = DBSCAN(eps=3, min_samples=2)
```

- Example: Hierarchical clustering, n_clusters=5, average linkage

```
from sklearn.cluster import AgglomerativeClustering
cl_alg = AgglomerativeClustering(5, linkage='average')
```

- Assessing clustering results
  - **Internal** metrics: use only the information of the features matrix
    - E.g. Silhouette, SSE

```
from sklearn.metrics import silhouette_score, silhouette_samples
silh_avg = silhouette_score(X, clusters)
silh_i = silhouette_samples(X, clusters)
```



  - **Silhouette** is a number in the range [−1, 1]
  - Higher values mean higher cluster quality
    - Clusters are well separated and cohesive
  - Expensive computation! $O(n^2)$

a(i): Average distance to points in the same cluster.

b(i): Smallest average distance to another cluster.

Formula:

$$s(i) = \frac{b(i) - a(i)}{max(b(i), a(i))}$$

Average s(i):

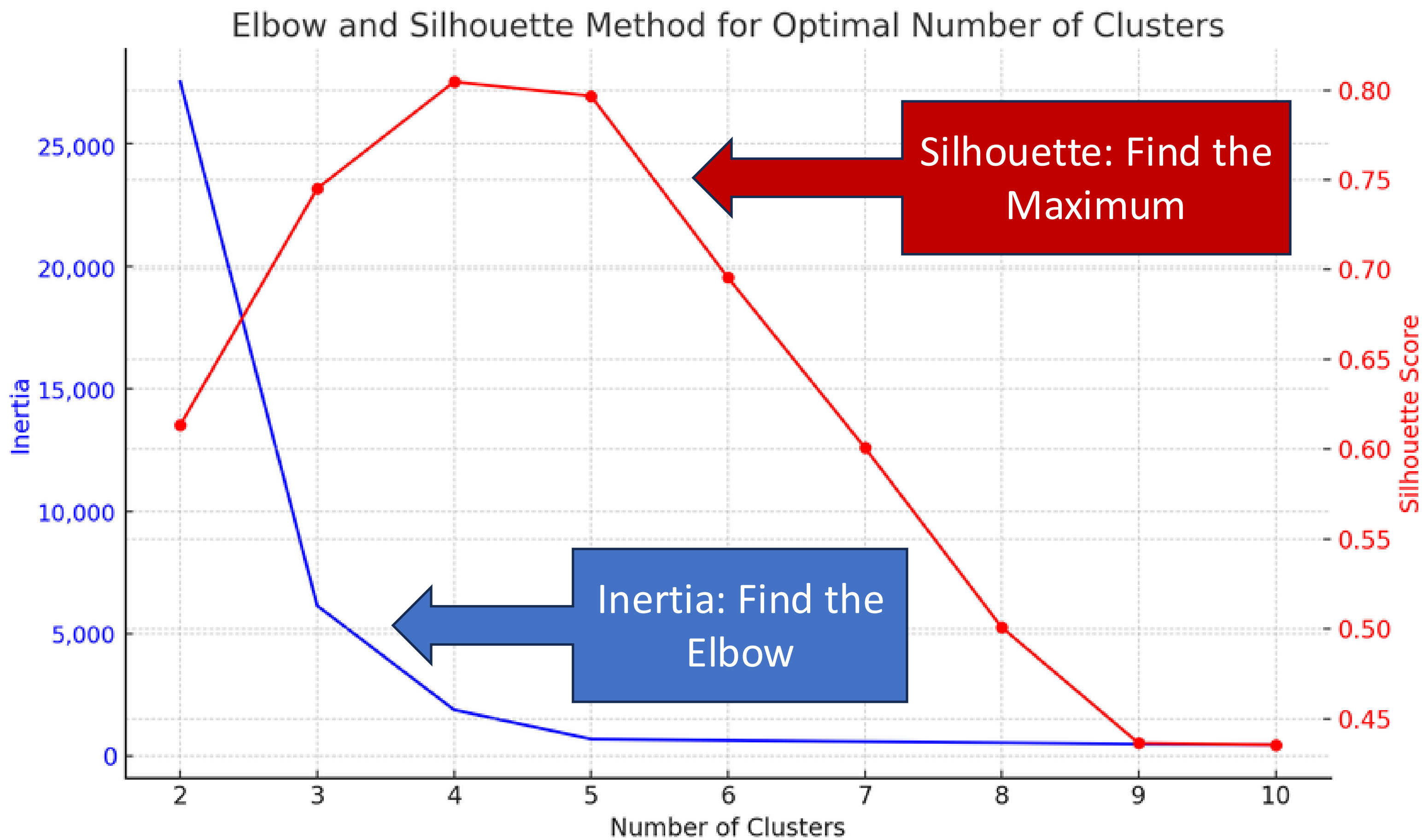
- Per cluster ⟶ cohesion.
- Whole dataset ⟶ clustering quality (close to 1).

- **SSE (Sum of Squared Errors):**
  Sum of the squared distances between each point $\boldsymbol{x_i}$ and its cluster center $\boldsymbol{C(x_i)}$.
- Also called ***Inertia***
- **Formula:**

$$SSE = \sum_{i}^{N} ||x_i - C(x_i)||^2$$

- **Meaning:**
  - Lower **SSE** ⟶ points are closer to their cluster center.
  - Used to evaluate clustering compactness.

Find the optimal K



- Assessing clustering results
  - **External** metrics: compare a clustering result with some ground-truth labels
    - E.g. Adjusted Rand Score, Fowlkes-Mallows index

```
from sklearn.metrics import adjusted_rand_score
ars = adjusted_rand_score(c_truth, c_pred)
```

ground truth

  - The ARS score ranges* in [0, 1]
    - ~ 0: randomly assigned clusters
    - 1: perfect agreement
    - [!] Values < 0 may occur if cluster assignments are worse than random
  - It is close to 1 when data in the predicted clusters is grouped in a similar way compared with ground truth

- Adjusted Rand Score (ARS)
  - Does not check for equality of target and predictions
  - It checks whether data are **clustered in the same way**

  - Example:
    - c_truth = [1, 1, 2, 2, 2, 1]
    - c_pred = [2, 2, 1, 1, 1, 2]
    - ARS(c_truth, c_pred) is 1