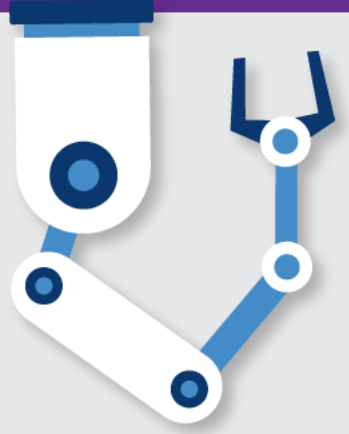
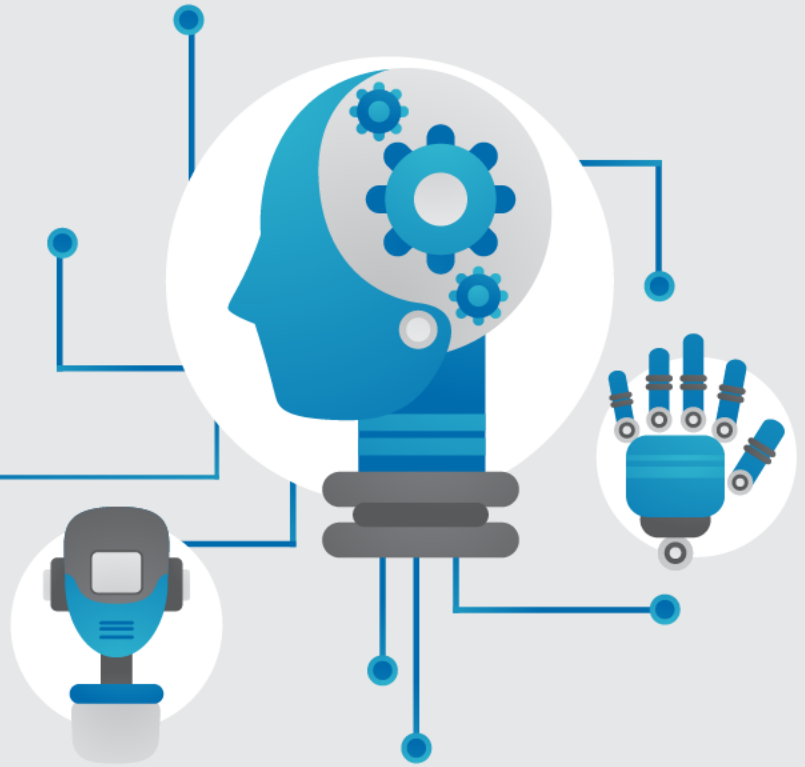
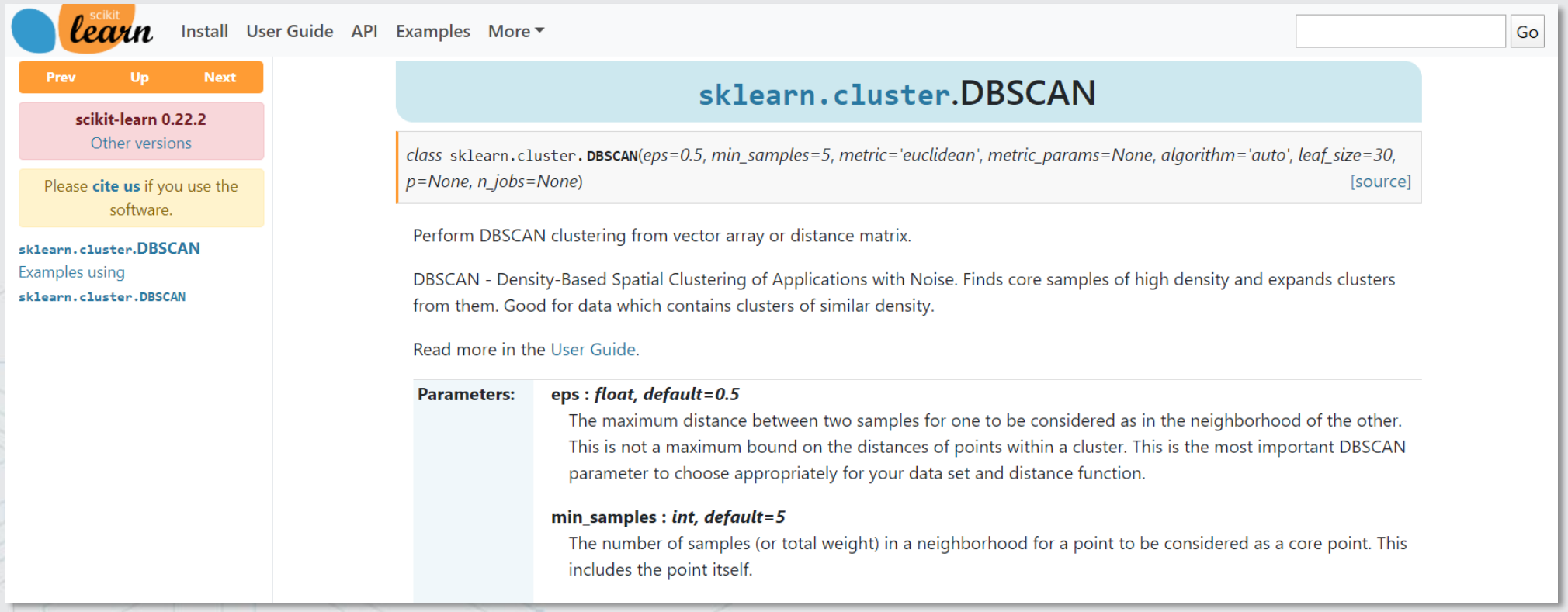


DBSCAN 實作



DBSCAN 實作

- sklearn.cluster.DBSCAN是DBSCAN演算法的實作



scikit learn Install User Guide API Examples More Go

Prev Up Next

scikit-learn 0.22.2
Other versions

Please cite us if you use the software.

sklearn.cluster.DBSCAN
Examples using
sklearn.cluster.DBSCAN

sklearn.cluster.DBSCAN

class sklearn.cluster.**DBSCAN**(*eps=0.5, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*) [source]

Perform DBSCAN clustering from vector array or distance matrix.

DBSCAN - Density-Based Spatial Clustering of Applications with Noise. Finds core samples of high density and expands clusters from them. Good for data which contains clusters of similar density.

Read more in the User Guide.

Parameters:

eps : *float, default=0.5*
The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.

min_samples : *int, default=5*
The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.

DBSCAN 參數說明

- class sklearn.cluster.DBSCAN(eps=0.5, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None)

- 常用參數
  - eps
  - min_samples
  - metric
  - metric_params
  - algorithm
  - leaf_size
  - p

DBSCAN 參數說明

- **eps：float, default=0.5**
  - 偵測距離，也就是兩個樣本點可以視為鄰居的最大距離
- **min_samples：int, default=5**
  - 決定一個樣本點能否為核心點的最少鄰居數
- **metric：string, or callable, default='euclidean'**
  - 距離度量函數，包含euclidean, manhattan和minkowsky等
- **metric_params：dict, default=None**
  - 距離度量函數的參數

DBSCAN 參數說明

algorithm：{‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’}, default=’auto’

- 用於尋找鄰居點的演算法

leaf_size：int, default=30

- 為使用KD-tree或Ball-tree時，停止建子樹的葉子節點中樣本數量的閾值

p：float, default=None

- 距離度量函數的參數
- 只用於Minkowsky距離中p值的選擇，p=1為Manhattan距離，p=2為Euclidean距離

DBSCAN 範例


```
from sklearn.cluster import DBSCAN
import numpy as np
X = np.array([[1, 2], [2, 2], [2, 3], [8, 7], [8, 8], [25, 80]])
clustering = DBSCAN(eps=3, min_samples=2).fit(X)
print(clustering.labels_)
# -1代表離群點
```

```
In [27]: print(clustering.labels_)
[ 0  0  0  1  1 -1]
```

資料分群：Iris


```
from sklearn.datasets import load_iris
data = load_iris()
x=data['data']
y=data['target']


from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.4, min_samples=4).fit(x)
for i in range(3):
    print('cluster'+str(i)+': ', dbscan.labels_[y==i], end='\n\n')
```

資料分群：Iris

- 分群結果

```
cluster0:  [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0 -1  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0
  0  0]

cluster1:  [ 1  1  1  1  1  1  1  3  1  1  3  1 -1  1 -1  1  1  1 -1  1  2  1  2  1
  1  1  1  1  1  1  1  1  1  2  1 -1  1 -1  1  1  1  1  1  3  1  1  1  1
  3  1]

cluster2:  [-1  2  2  2  2 -1 -1 -1 -1 -1  2  2  2  2 -1  2  2 -1 -1 -1  2  2 -1  2
  2 -1  2  2  2 -1 -1 -1  2  2 -1 -1  2  2  2  2  2  2  2  2  2  2  2  2
  2  2]
```

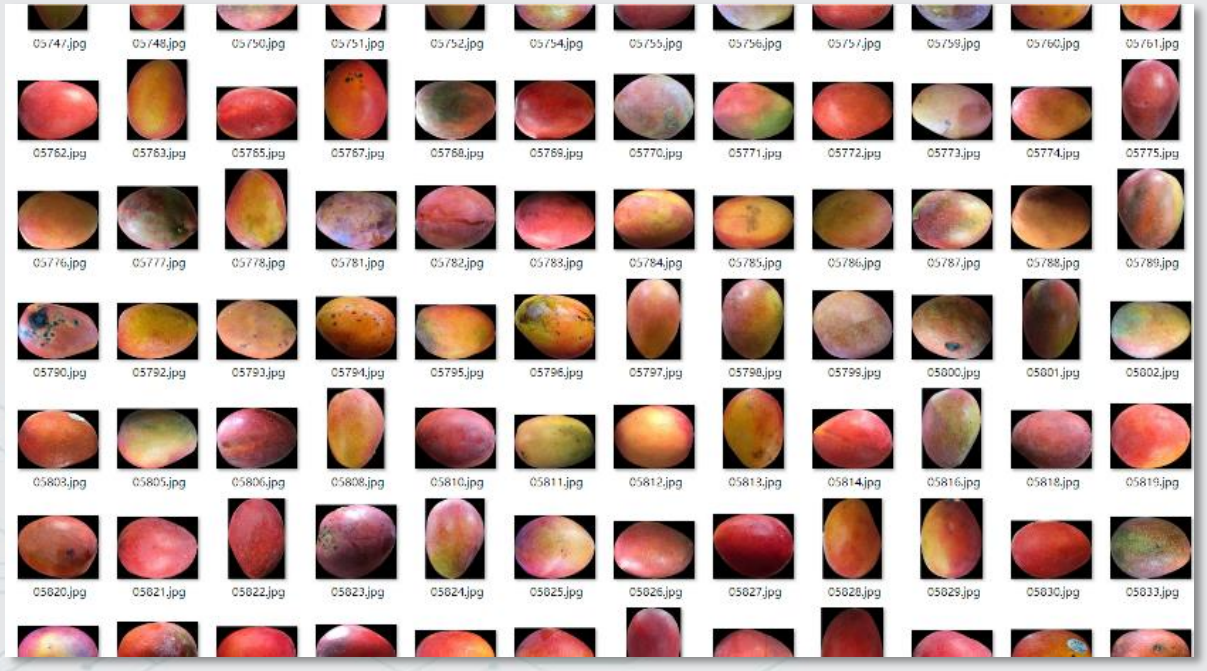
離群影像：色彩、亮度特徵


```
import cv2, os
files=os.listdir('image')
x=[]
for file in files:
    img = cv2.imread('image/'+file)
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    hist = cv2.calcHist(
            [lab],
            [0, 1, 2],
            None,
            [16,16,16],
            [0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, None).ravel()
    x.append(hist)
```

離群影像：色彩、亮度特徵


```
from sklearn.cluster import DBSCAN
import numpy as np
import shutil
dbscan = DBSCAN(eps=0.6, min_samples=30).fit(x)
for i in np.where(dbscan.labels_==-1)[0]:
  shutil.copyfile('image/'+files[i], 'output/'+files[i])
```

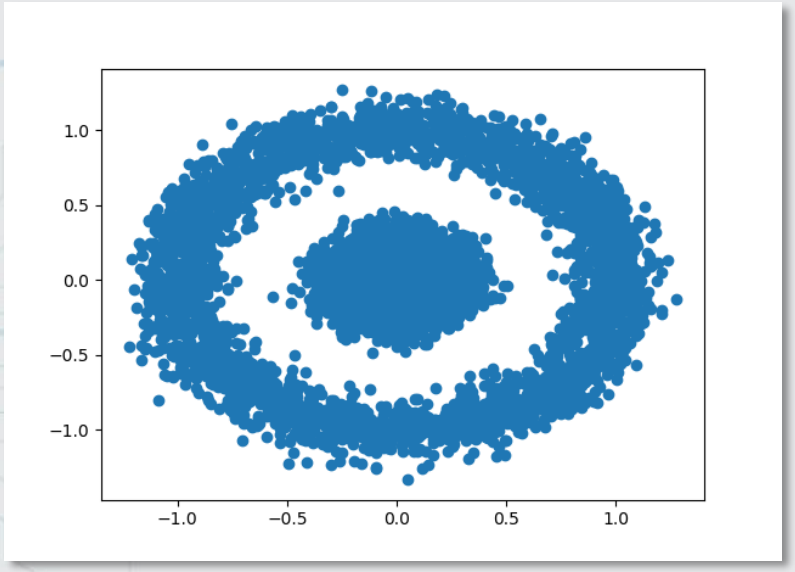
離群影像：色彩、亮度特徵


05747.jpg
05748.jpg
05750.jpg
05751.jpg
05752.jpg
05754.jpg
05755.jpg
05756.jpg
05757.jpg
05759.jpg
05760.jpg
05761.jpg
05762.jpg
05763.jpg
05765.jpg
05767.jpg
05768.jpg
05769.jpg
05770.jpg
05771.jpg
05772.jpg
05773.jpg
05774.jpg
05775.jpg
05776.jpg
05777.jpg
05778.jpg
05781.jpg
05782.jpg
05783.jpg
05784.jpg
05785.jpg
05786.jpg
05787.jpg
05788.jpg
05789.jpg
05790.jpg
05792.jpg
05793.jpg
05794.jpg
05795.jpg
05796.jpg
05797.jpg
05798.jpg
05799.jpg
05800.jpg
05801.jpg
05802.jpg
05803.jpg
05805.jpg
05806.jpg
05808.jpg
05810.jpg
05811.jpg
05812.jpg
05813.jpg
05814.jpg
05816.jpg
05818.jpg
05819.jpg
05820.jpg
05821.jpg
05822.jpg
05823.jpg
05824.jpg
05825.jpg
05826.jpg
05827.jpg
05828.jpg
05829.jpg
05830.jpg
05833.jpg

K-means vs. DBSCAN

- 產生範例

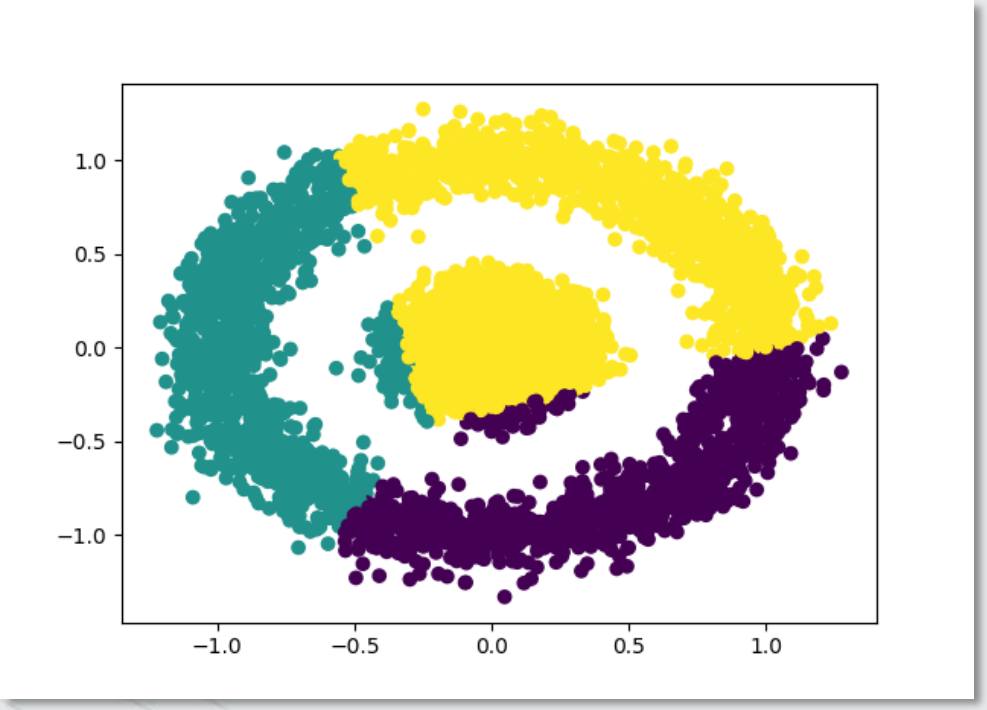
```
import matplotlib.pyplot as plt
from sklearn import datasets

X, y =datasets.make_circles(n_samples=6000, factor=0.2 ,noise =0.1 )

plt.scatter(X[:, 0], X[:, 1], marker= 'o' )
plt.show()
```



K-means vs. DBSCAN

- K-means

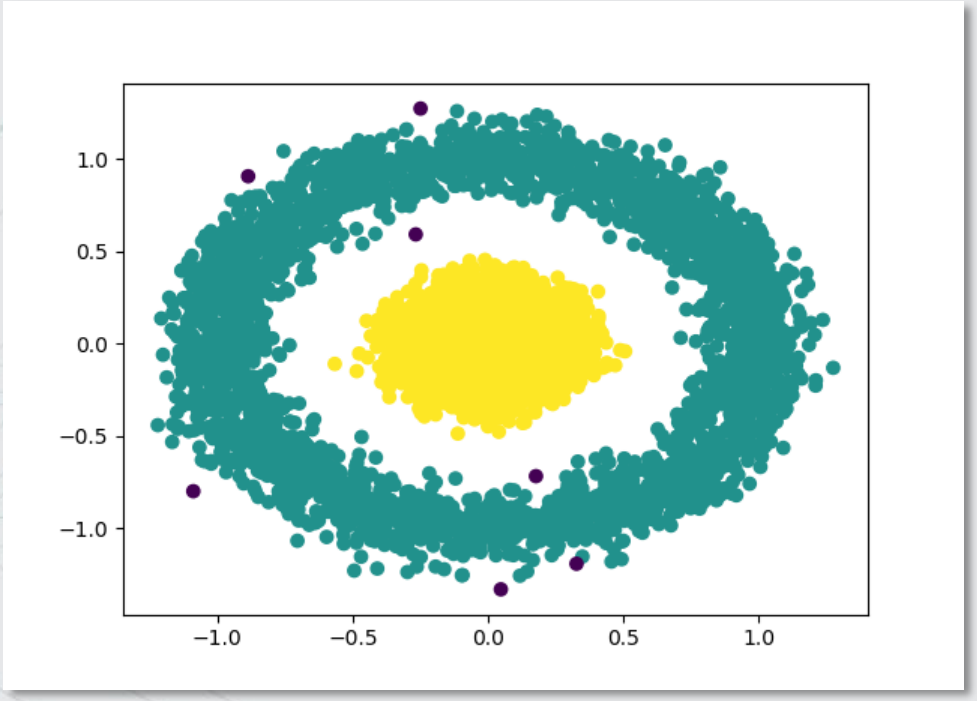
```
from sklearn.cluster import KMeans
y_pred = KMeans(n_clusters=3, random_state=1).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c= y_pred)
plt.show()
```



K-means vs. DBSCAN

- DBSCAN, eps = 0.1

```
from sklearn.cluster import DBSCAN
y_pred = DBSCAN(eps=0.1).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c= y_pred)
plt.show()
```



K-means vs. DBSCAN

- DBSCAN, eps = 0.3

```
from sklearn.cluster import DBSCAN
y_pred = DBSCAN(eps=0.3).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c= y_pred)
plt.show()
```