ml-04-ex2-dbscan_nocode

November 3, 2020

Claudio Sartori

Elaboration from the example given in Sebastian Raschka, 2015

https://github.com/rasbt/python-machine-learning-book

1 Machine Learning - Lab

1.1 Working with Unlabeled Data – Clustering Analysis

1.1.1 Use DBSCAN

1.1.2 Overview

In this example we will use an *artificial* data set
1. load the data
2. check the shape and plot the content
3. observe the plot and decide which are the most interesting columns, to use in the plots of the clusters
- make a 2d plot of the two most promising columns
4. initialize and `fit_predict` an estimator for `DBSCAN`, using the default parameters, then print the results
- print the estimator to check the parameter values
- the labels are the unique values of the predicted values
- print if there is noise
- if there is noise the first cluster label will be `-1`
- print the number of clusters (noise excluded)
- the other clusters are labeled starting from `0`
- for each cluster (noise excluded) compute the **centroid**
- plot the data with the centroids and the colors representing clusters
- use the `plot_clusters` function provided
5. find the best parameters using `ParameterGrid`
- prepare a dictionary with the parameters lists
- generate the list of the parameter combinations with `ParameterGrid`
- for each combination of parameters
- initialize the DBSCAN estimator
- `fit_predict`
- extract the labels and the number of clusters excluding the *noise*
- compute the silhouette score and the number of unclustered objects (noise)
- filter and print the parameters and the results

- print if the silhouette score is above a threshold and the percentage of unclustered is below a threshold

6. observe visually the most promising combination of parameters

- fit and predict the estimator
- plot the clusters
- plot the silhouette scores for each sample using the function `plot_silhouette`

[1]:
```
from IPython.display import Image
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score, silhouette_samples
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import ParameterGrid
from sklearn.preprocessing import MinMaxScaler

%matplotlib inline

rnd_state = 42 # This variable will be used in all the procedure calls allowing␣
 ↪a random_state parameter
               # in this way the running can be perfectly reproduced
               # just change this value for a different experiment

# the .py files with the functions provided must be in the same directory of␣
 ↪the .ipynb file
from plot_clusters import plot_clusters      # python script provided separately
from plot_silhouette import plot_silhouette  # python script provided separately
```

[2]:
```
help(plot_clusters)
```

```
Help on function plot_clusters in module plot_clusters:

plot_clusters(X, y, dim, points, labels_prefix='cluster',
points_name='centroids', colors=<matplotlib.colors.ListedColormap object at
0x114d78250>, points_color=(0.09019607843137255, 0.7450980392156863,
0.8117647058823529, 1.0))
    Plot a two dimensional projection of an array of labelled points
    X:      array with at least two columns
    y:      vector of labels, length as number of rows in X
    dim:    the two columns to project, inside range of X columns, e.g. (0,1)
    points: additional points to plot as 'stars'
    labels_prefix: prefix to the labels for the legend ['cluster']
    points_name:   legend name for the additional points ['centroids']
    colors: a color map
    points_color: the color for the points
```

[3]:
```
help(plot_silhouette)
```

```
Help on function plot_silhouette in module plot_silhouette:

plot_silhouette(silhouette_vals, y, colors=<matplotlib.colors.ListedColormap
object at 0x114d78250>)
    Plotting silhouette scores for the individual samples of a labelled data
set.
    The scores will be grouped according to labels and sorted in descending
order.
    The bars are proportional to the score and the color is determined by the
label.

    silhouette_vals: the silhouette values of the samples
    y:               the labels of the samples
```

1.1.3 1. Load the data

[4]:
```
# data_file = 'ex1_4dim_data.csv'
# data_file = 'ex1_4dim_mod_data.csv'
# data_file = 'ex1_data.csv'
data_file = 'ex1_4d_moon.csv'
delimiter = ','

# to fill
```

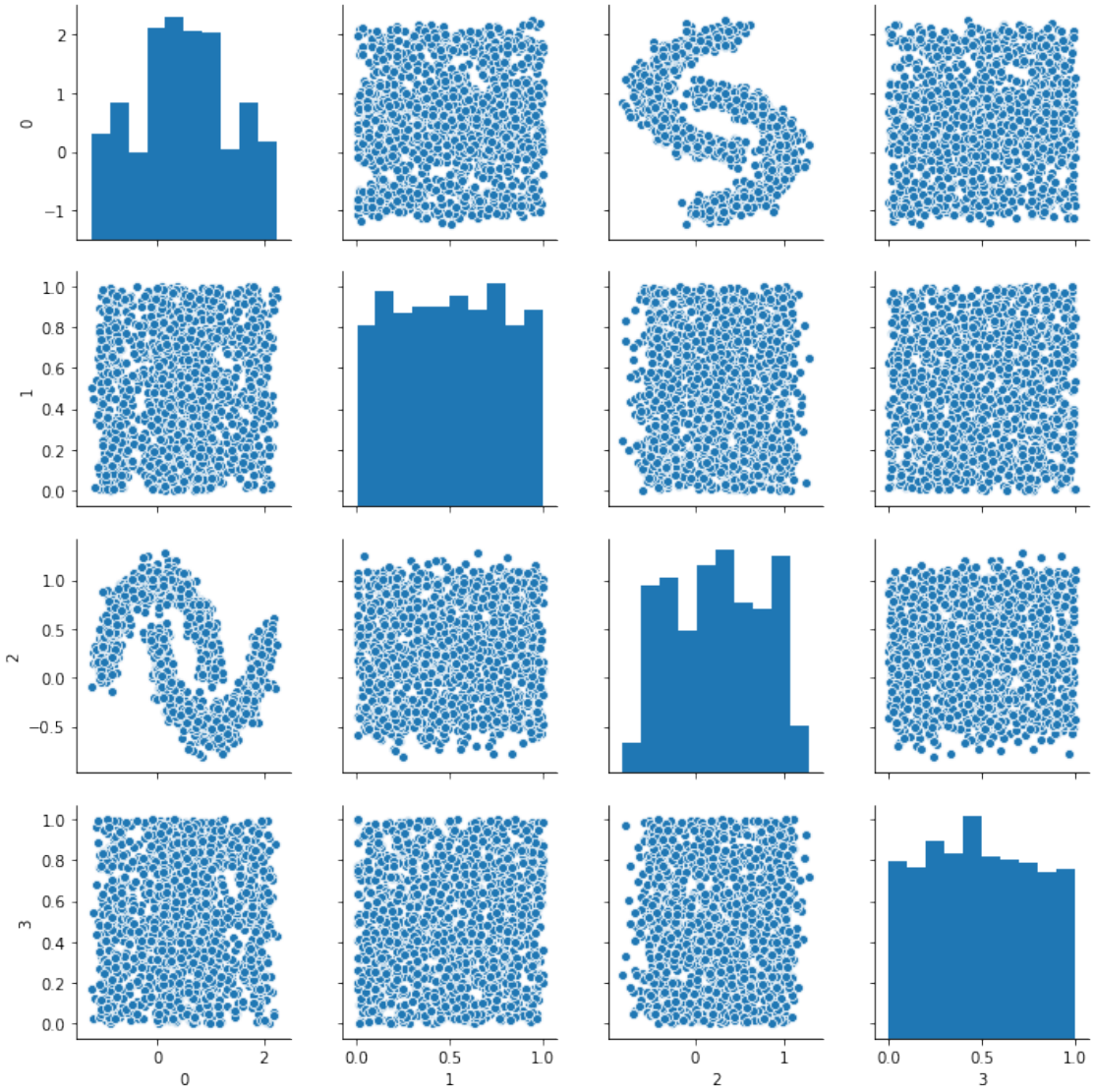
1.1.4 2. Inspect

[5]:
```
# to fill
```

[5]:
```
(1500, 4)
```

[6]:
```
# to fill
```

[6]:
```
<seaborn.axisgrid.PairGrid at 0x1a1a130350>
```

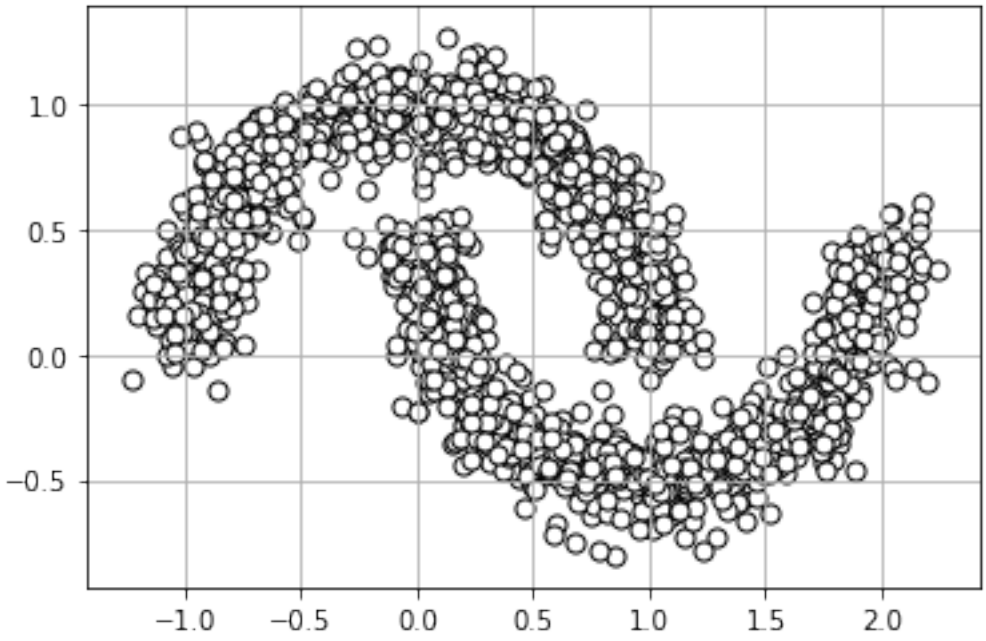


1.1.5 3. Observing the pairplots

In this simple example you can easily see which are the two most interesting columns.

All the plots will focus on those columns

[7]:
```
# to fill
```



1.1.6 4. Initialize, fit_predict and plot the clusters

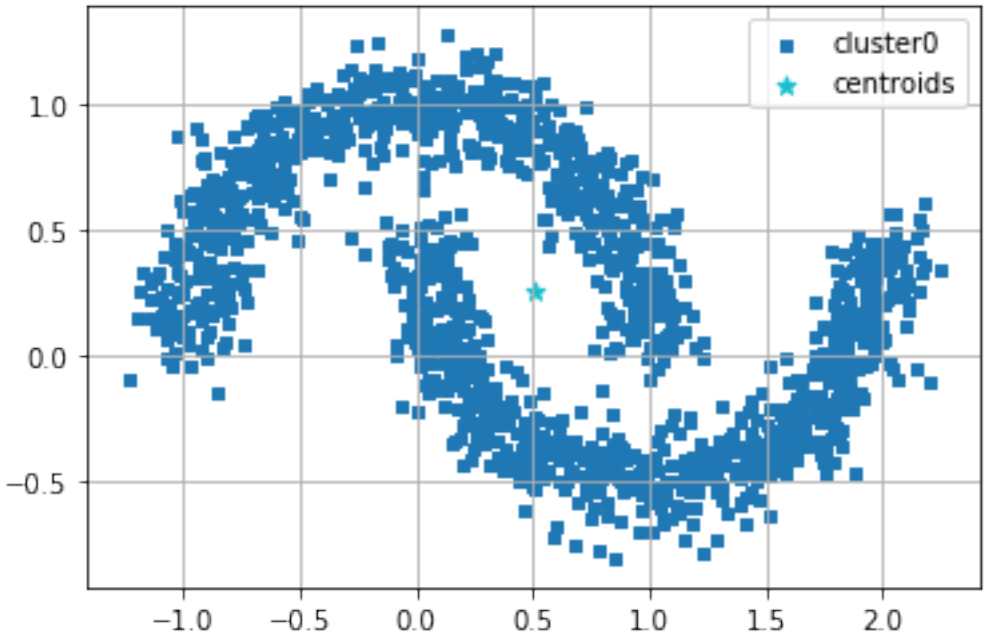
[9]:
```
# to fill
```

```
DBSCAN(algorithm='auto', eps=0.5, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=5, n_jobs=None, p=None)
```

[10]:
```
# to fill
```

```
There is/are 1 cluster(s)
```

[11]:
```
# to fill
```



1.1.7 5. Find the best parameters using `ParameterGrid`

[12]:
```
param_grid = {'eps': list(np.arange(0.05, 1, 0.05)), 'min_samples':␣
↪list(range(1,10,1))}
params = list(ParameterGrid(param_grid))
sil_thr = 0  # visualize results only for combinations with silhouette above␣
↪the threshold
unc_thr = 33 # visualize results only for combinations with unclustered% below␣
↪the threshold
```

[13]:
```
# to fill
```

eps	min_samples	n_clusters	silhouette	unclust%
0.05	1	1490	0.01	0.00%
0.10	1	1297	0.09	0.00%
0.15	1	698	0.07	0.00%
0.15	2	253	0.25	29.67%
0.25	2	2	0.14	1.27%
0.25	7	2	0.28	3.27%
0.25	8	2	0.28	5.80%
0.25	9	2	0.29	8.47%

1.1.8 6. Observe

- Observe visually the most promising combination of parameters.

- Plot the clusters with the centers
- Plot the silhouette indexs for all the clustered samples

[14]:
```
# to fill
```

[16]:
```
# to fill
```

```
There are 2 clusters
```

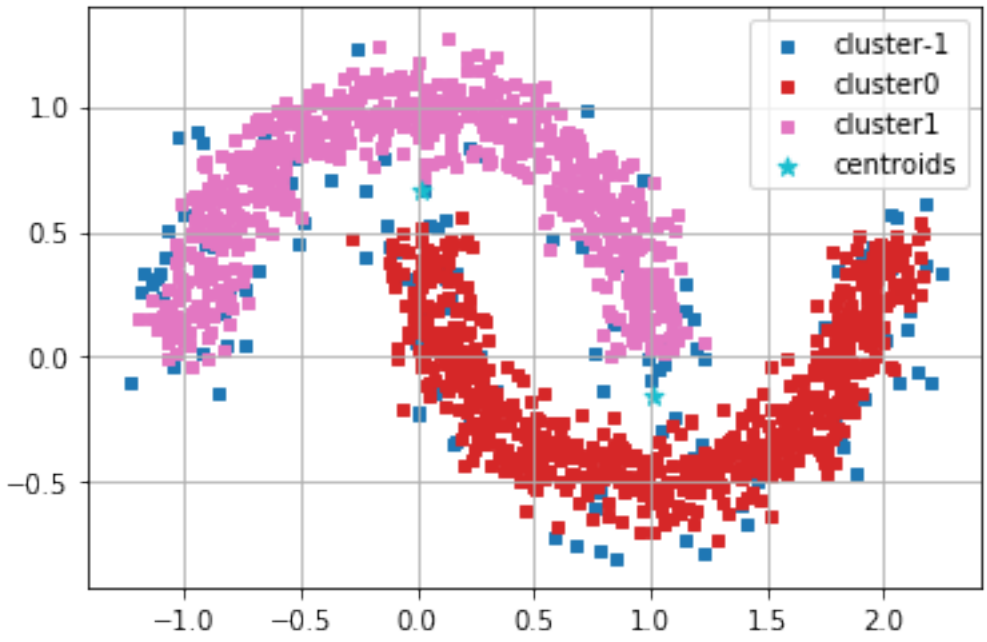
[17]:
```
# to fill
```

```
The cluster labels are [0 1]
```

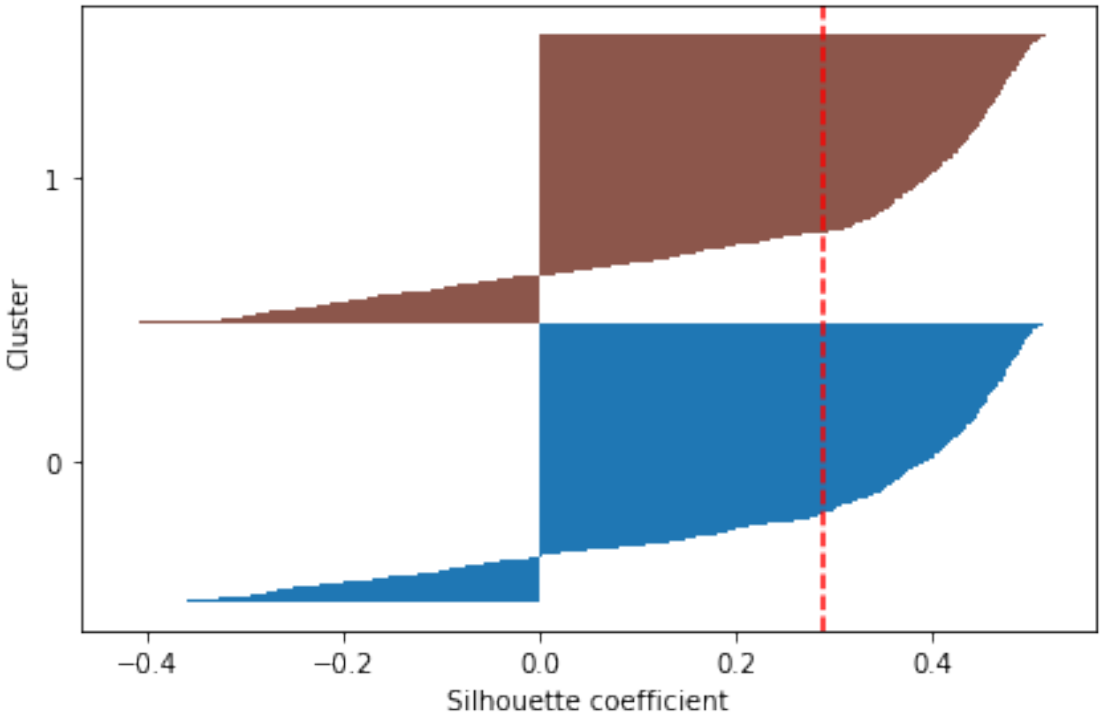
[18]:
```
# to fill
```

[18]:
```
array([[ 1.01020828,  0.5258109 , -0.15197393,  0.49321332],
       [ 0.01924021,  0.48326285,  0.66582183,  0.4831518 ]])
```

[19]:
```
# to fill
```



[20]:
```
# to fill
```


Cluster
1
0
−0.4
−0.2
0.0
0.2
0.4
Silhouette coefficient