

Anomaly detection using K-means

Author: Tommaso Puccetti

Università degli Studi di Firenze

21/12/2018

Clustering: basics

Clustering analysis *is a collection of statistical methods that can be used to assign units to groups, where group members share some characteristic.*

- **data reduction**: reduce data that are homogeneous (similar)
- **find natural clusters** and describe their unknown properties
- **find** useful and suitable **groupings**
- **find** unusual data objects (i.e. outlier detection).

The aim is to find a way of **grouping statistical units** (or variables) in a data set in such a way that: **units are similar within groups and dissimilar among groups.**

Clustering algorithm

The essentials steps for a clustering algorithm are the following:

- 1 Define a **metric** to evaluate distances between units;
- 2 Define an **algorithm** to form groups.

A **grouping** algorithm can be:

- **Hierarchical**
- **Non Hierarchical**

Non-hierarchical clustering: K-means

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. The aim is to find a partition of the dataset into a set of K clusters that optimizes the chosen partitioning criterion.

- 1 choose K points at random as cluster centers (centroids);
- 2 assign each unit to its closest centroid, using an opportune distance measure (usually Euclidean or Manhattan);
- 3 calculate the centroid of each cluster and use it as the new cluster center;
- 4 go back to step 2, stop when cluster centers do not change any more.

K-means: implementation (1)

- First of all the algorithm choose K random centroid.

```
z = self.rng.permutation(len(X))[:self.k]  
X = np.array(X, dtype= "float64")  
centers = X[z]
```

K-means: implementation (2)

- The second step of the fit() method is to **calculate the Euclidean distance** between all the dataset points in respect to the K centroids. The distances are used to assign all the points to the nearest cluster, using K labels.

$$d(x, y) = \sqrt{\sum_{i=0}^n (x_i - y_i)^2}$$

```
for i in range(self.k):
    distances[:,i] = np.linalg.norm(X - centers[i],
                                    axis=1)

    labels= np.argmin(distances, axis = 1)
```

K-means: implementation (3)

- For each cluster the algorithm **calculates the new centroid** as the mean of the points assigned to them:

```
for i in range(self.k):  
    if(i in labels):  
        new_centers = X[labels == i].mean(0)  
        center_temp.append(new_centers)  
    else:  
        center_temp.append(self.farthest(X,  
                                         distances, labels))  
new_centers = np.array(center_temp,  
                        dtype= "float64")
```

K-means: implementation (4)

Repeats from point 2 to point 3 until the new centroids are equal to the centroids from the previous step.

```
if np.all(centers == new_centers):  
    break  
  
    centers = new_centers
```

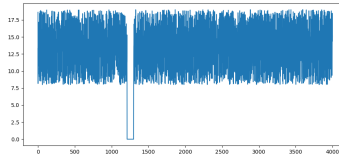
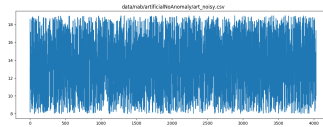

K-means: example

The ***kmeans_sk.py*** file contains an example of usage for the algorithm. The artificial data presents 4 distinct points aggregates. Using the ***fit()*** method with $K = 4$ the algorithm, as we expected, identifies the 4 aggregates of points as 4 clusters.

INSERIRE IMMAGINI ESEMPIO AFFIANCATE

Anomaly detection using K-means

K-means algorithm can be used to spot anomalies given the normal behaviour.



Anomaly detection using K-means: implementation (1)

The idea is the following:

- split the waveform into n segments of length l : every segment is considered a statistical unit;
- apply the K-means algorithm over the l -dimension points obtained (segments). The results are K cluster's *centroids*;
- a centroid is a segments itself that can be used to reconstruct the original waveform: for every segments the nearest centroid is used to approximate the current;

Anomaly detection using K-means: implementation (2)

- once obtained a reconstructed waveform using only centroids, we can subtract it from the original: the result is also a waveform that represents the noise signal between the two;
- the last step is to use a threshold value to detect where the error between the original and the reconstructed waveforms is anomalous in respect to the noise obtained.

Segmentation

INSERIRE IMMAGINE FINESTRE

Segmentation

The segmentation is done using the *segmentation()* function implemented in the K-means class.

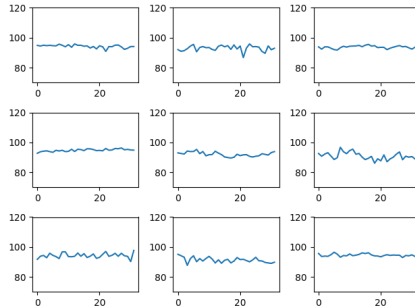
```
segment_len = 32
slide_len = 16
km = skmeans.K_means(30)
segments = km.segmentation(train_array, segment_len, slide_len)
```

The output would be a Numpy's array containing the segments obtained.

Calculate centroids

```
centr = km.fit(segments)
learn_utils.plot_waves(centr, step=3)
```

The output would be a Numpy's array containing K centroids of the dataset. The picture below shows some of the normal waveform obtained.



Reconstruction (1)

Once obtained the centroids we can use them to reconstruct the original waveform. First of all we have to apply again the segmentation process, this time over the entire dataset.

```
segments = km.segmentation(dataset_array, segment_len, slide_len)  
lab = km.predict(centr, segments)
```


Reconstruction (2)

Then the ***predict()*** function is used to find, for all the segments obtained, the nearest centroid. The output would be an array which store in the n-th position a label referring to the nearest centroid for the n-th segment.

```
Predicted Labels
[12 27 3 14 23 22 27 16 12 26 12 19 4 5 7 6 6 6 28 10 9 1 10 4
29 16 6 21 20 15 13 9 24 7 12 2 17 17 11 1 23 9 10 9 11 11 10 6
6 8 0 8 9 9 1 11 0 23 0 21 26 27 6 8 24 23 25 25 10 23 25 8
9 10 25 1 11 10 5 23 24 0 10 8 9 10 23 23 0 23 23 18 10 10 7 23
.....
20 21 26 21 26 26 21 21 26 21 21 26 20 27 6 7 27 12 27 14 6
21 21 15 21 13 13 27 6 10 0 23 9 0 12 21 12 12 20 20 27 12 27 12
27 21 21 20 23 23 9 0 9 10 10 8 9]
```

Reconstruction (3)

Finally the reconstruction algorithm is used to obtain the whole waveform.

```
noise = detection.reconstruction(dataset_array, segments, lab,
                                centr, slide_len, segment_len)
```

The noise obtained as output is plotted along with the original and the reconstructed waveform.

