Implementation of K-Means

Raúl Maximiliano Urrutia Hernández

November 7th, 2019

Algorithm

The algorithm that I used to make the implementation is (taken from: M. Zaki and W. Meira, *Data Mining and Analysis Fundamental Concepts and algorithms*.)

Algorithm 13.1: K-means Algorithm

K-MEANS $(\mathbf{D}, k, \epsilon)$:

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

repeat

  $t \leftarrow t + 1$

  // Cluster Assignment Step

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do**

    $j^* \leftarrow \arg\min_i \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_i^t \right\|^2 \right\}$ // Assign $\mathbf{x}_j$ to closest centroid

    $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$

  // Centroid Update Step

  foreach $i = 1$ *to* k **do**

    $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$

until $\sum_{i=1}^k \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\| \leq \epsilon$

Implementation

First the algorithm randomly initializes the centers:

```
24          #Randomly initialize k centers
25          random_state = np.random.mtrand._rand
26          seeds = random_state.permutation(n_samples)[:self.k]
27          self.centers = X[seeds]
```

In the cluster assignment step, each point is assigned to the closer center:

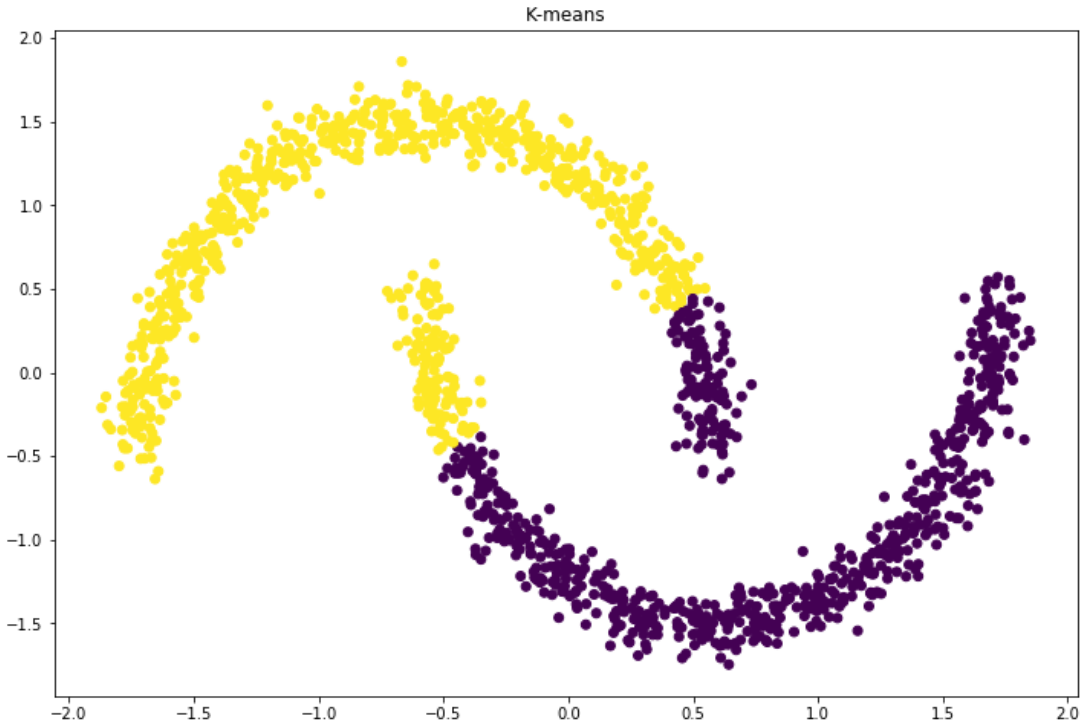
```
36              #Cluster assignment step
37              for entry in X :
38                  distances = [np.linalg.norm(entry - center) ** 2
39                               for center in self.centers]
40                  label = distances.index(min(distances))
41                  groups[label].append(entry)
```

In the center update step, each center is updated to the average or mean of the cluster:

```
46              #Centers update step
47              for l in groups :
48                  self.centers[l] = np.average(groups[l], axis = 0)
```

Testing the implementation

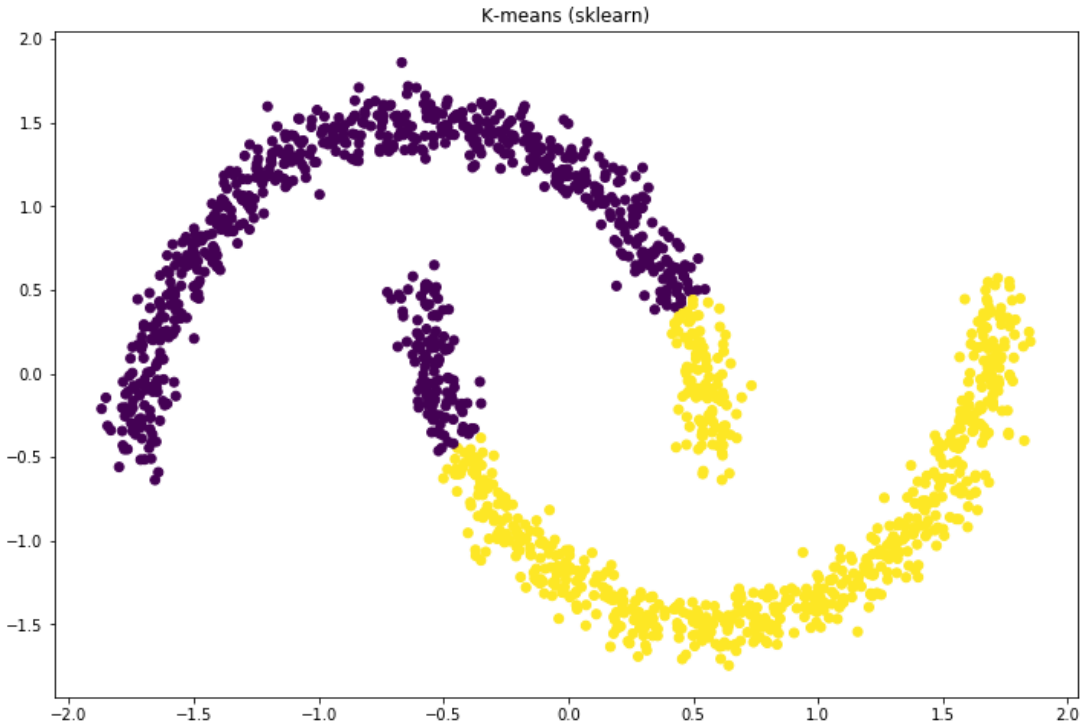
The scatter plot of the clustering of the make_moons dataset made by this implementation goes as follows



The adjusted Rand Index takes a value of 0.48407, and the adjusted Mutual information is 0.38492.

Testing the implementation

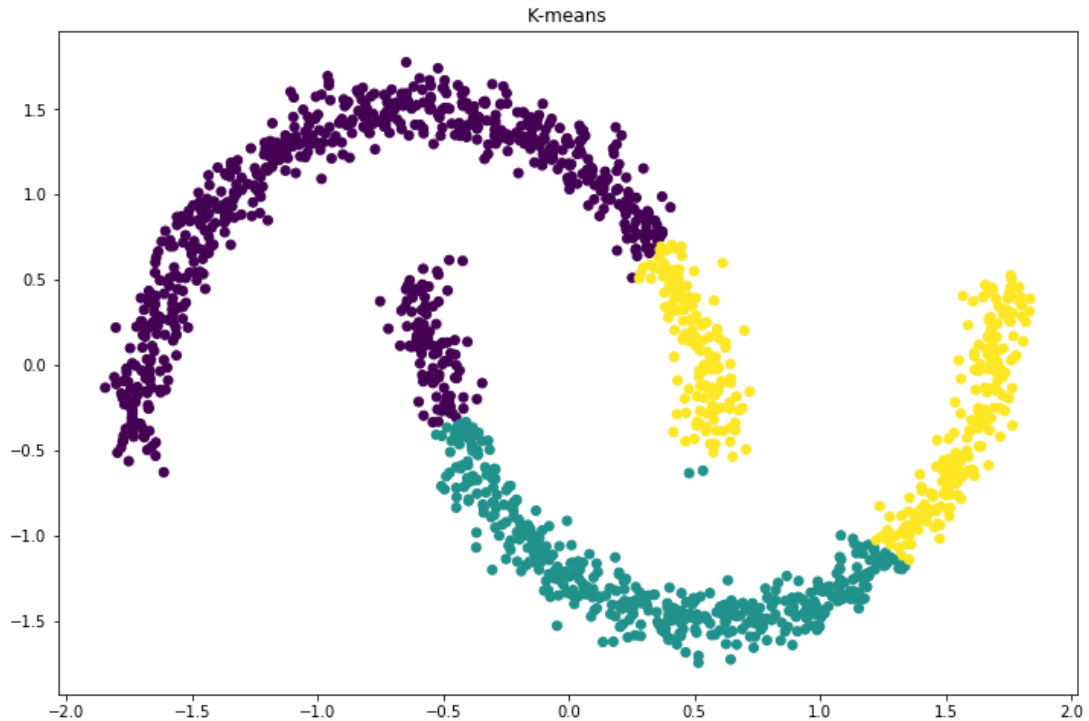
The scatter plot of the clustering made by sklearn kmeans



With adjusted Rand Index of 0.48407 and adjusted Mutual information of 0.38492.
As we can see from the plots and the metrics, both results (from the implementation and from the sklearn kmeans) are the same.

Testing the implementation

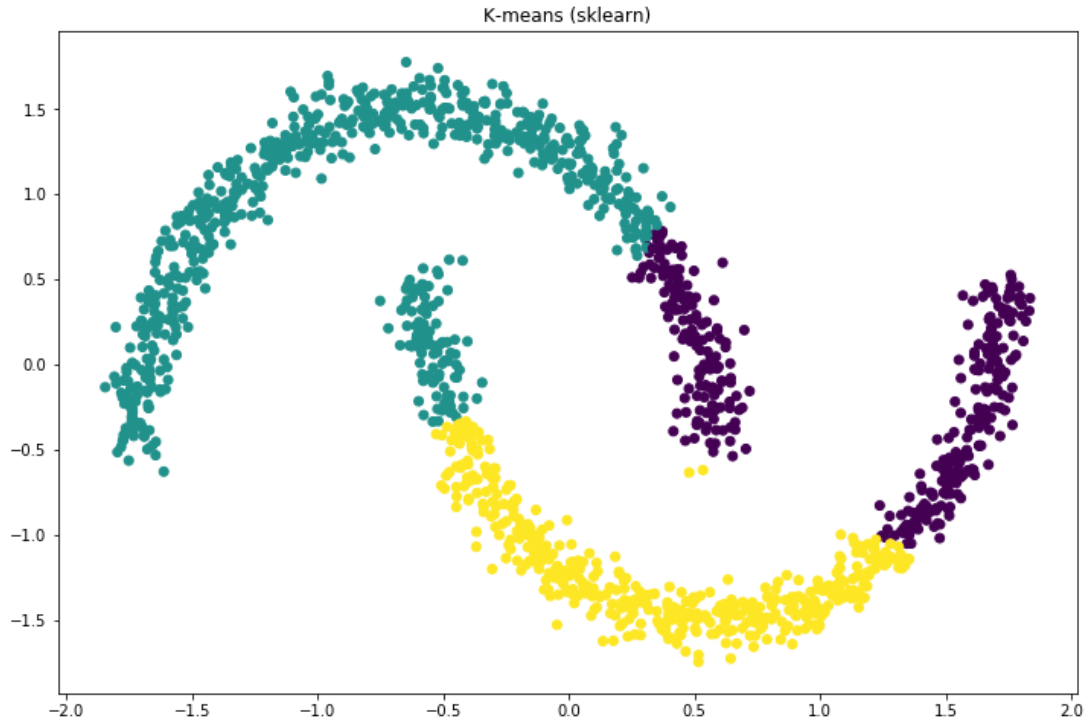
Lets make another test, now with 3 clusters. The scatter plot of the clustering made by the implementation is



This time, the adjusted Rand Index takes a value of 0.39336, and the adjusted Mutual information is 0.31165.

Testing the implementation

The scatter plot of the clustering made by sklearn kmeans is



With adjusted Rand Index of 0.38748 and adjusted Mutual information of 0.30811.
The metrics of both results are almost the same (slightly different due the random nature of k-means).