k-means

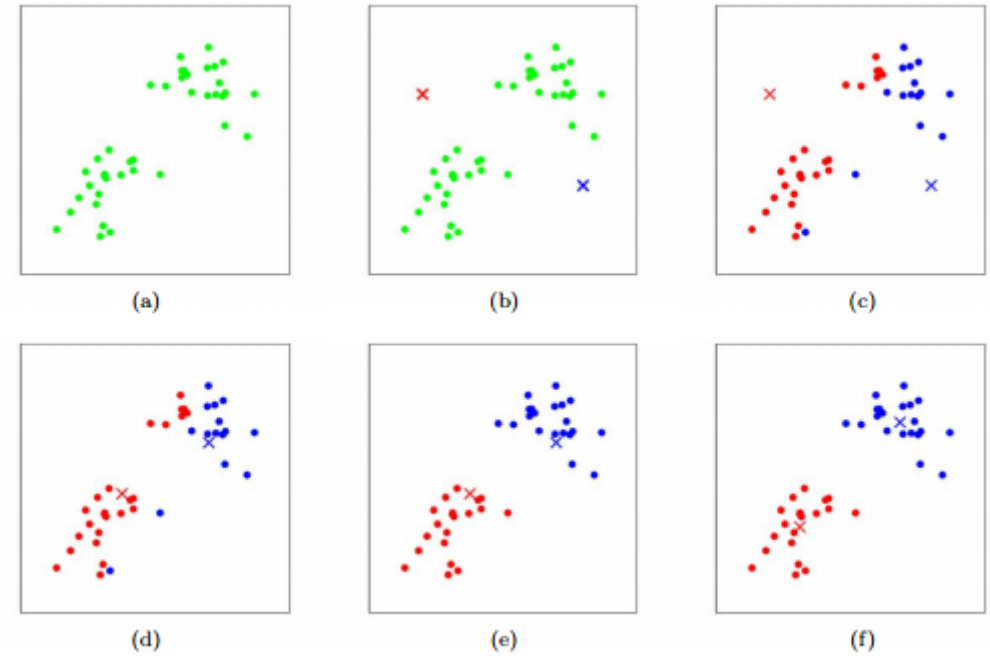
Theory

K-means is an unsupervised machine learning technique used to identify clusters of data objects in a dataset. A point is considered to be in a particular cluster if it closer to that cluster's centroid than any other centroid.

K-means requires only a few steps. The first step is to randomly select k centroids, where k is equal to the number of clusters you choose. Centroids are data points representing the center of a cluster.

The second steps called expectation-maximization. The expectation step assigns each data point to its nearest centroid. Then, the maximization step computes the mean of all the points for each cluster and sets the new centroid. Here's the k-means algorithm looks like:

The quality of the cluster assignments is determined by computing the sum of the squared error (SSE) after the centroids converge, or match the previous iteration's assignment. The SSE is defined as the sum of the squared Euclidean distances of each point to its closest centroid. Since this is a measure of error, the objective of k-means is to try to minimize this value.
K-means can be described as follows:



1. Specify the number k of clusters to assign.
2. Randomly initialize k centroids.
3. Repect
    - Assign each point to its closet centroid.
    - Compute the new centroid(mean) of each cluster.
4. Until The centroid position do not change

Implement

Import Packages And Dataset

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
# from sklearn import datasets
from sklearn.datasets import load_iris
```

```
iris = load_iris()
# Take the last two dimensions in the feature
X = iris.data[:,2:]
# drawer
plt.scatter(X[:, 0], X[:, 1], c = "blue", marker='+', label='label')
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.legend(loc=2)
plt.show()
```

Start process

```
estimator = KMeans(n_clusters=3)
estimator.fit(X)
label_pred = estimator.labels_
```

Visualization

```
x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
plt.scatter(x0[:, 0], x0[:, 1], c = "red", marker='o', label='label0')
plt.scatter(x1[:, 0], x1[:, 1], c = "green", marker='*', label='label1')
plt.scatter(x2[:, 0], x2[:, 1], c = "blue", marker='+', label='label2')
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.legend(loc=2)
plt.show()
```

Practice

Please imitate it to perform on the following dataset.The result show be visualized like above

https://archive.ics.uci.edu/ml/datasets/seeds