K-Means Clusterring: TEORI DAN PRAKTIK

Isram Rasal, S.T., MMSI., M.Sc

Machine Learning Course

17/02/2019

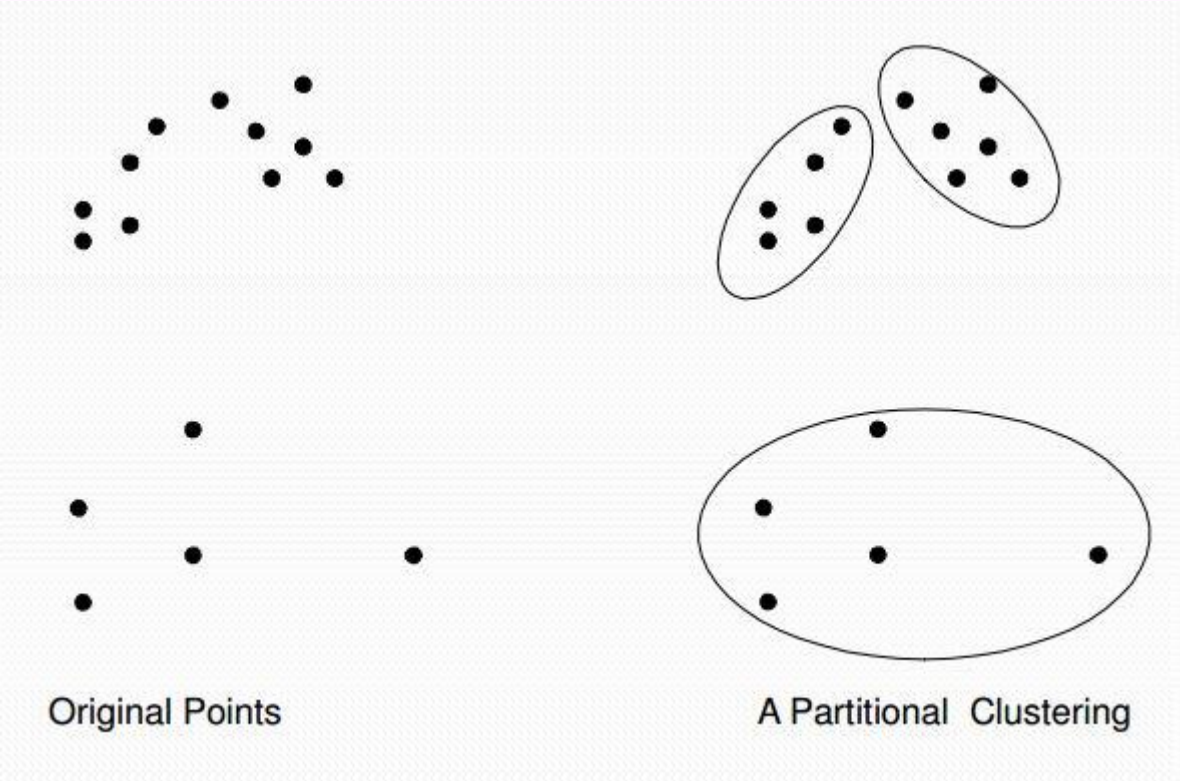
Sejarah K-Means

- Istilah "k-means" pertama kali digunakan oleh James MacQueen pada tahun 1967, meskipun idenya berasal dari Hugo Steinhaus pada tahun 1957.
- Algoritma standar pertama kali diusulkan oleh Stuart **Lloyd** pada tahun 1957 sebagai teknik modulasi kode-pulsa, meskipun tidak dipublikasikan.
- Pada tahun 1965, E. W. **Forgy** pada dasarnya menerbitkan metode yang sama, itulah sebabnya kadang-kadang k-means disebut juga sebagai **Lloyd-Forgy**.

K-Means Clustering

- K-Means Clustering adalah suatu metode penganalisaan data atau metode Data Mining yang melakukan proses pemodelan tanpa supervisi (*unsupervised*) dan merupakan salah satu metode yang melakukan pengelompokan data dengan sistem partisi.
- Terdapat dua jenis data clustering yang sering dipergunakan dalam proses pengelompokan data yaitu **Hierarchical** dan **Non-Hierarchical**, dan **K-Means** merupakan salah satu metode data clustering non-hierarchical atau **Partitional Clustering**.

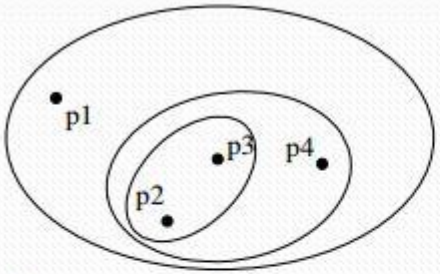
K-Means Clustering



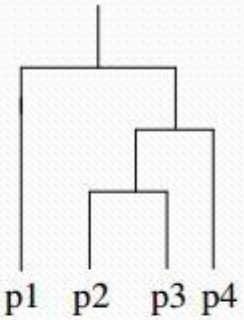
K-Means Clustering

- Metode **K-Means Clustering** berusaha mengelompokkan data yang ada ke dalam beberapa kelompok, dimana data dalam satu kelompok mempunyai karakteristik yang sama satu sama lainnya dan mempunyai karakteristik yang berbeda dengan data yang ada di dalam kelompok yang lain.

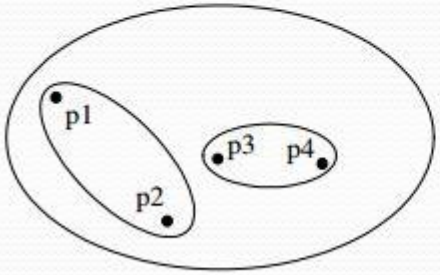
K-Means Clustering



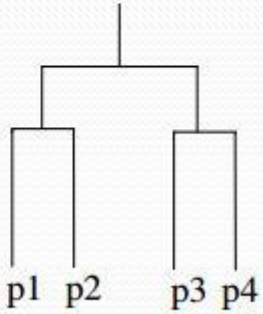
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering

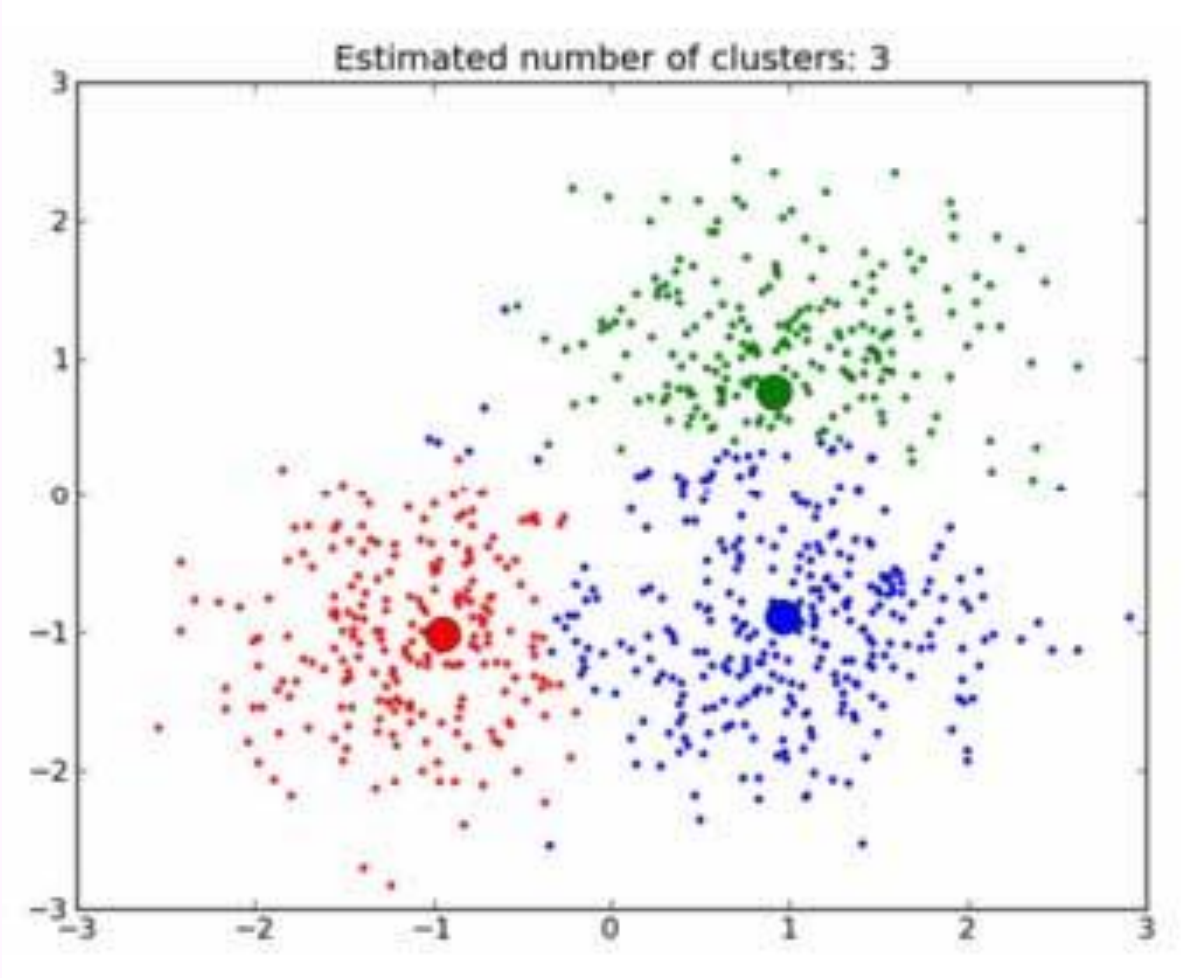


Non-traditional Dendrogram

K-Means Clustering

- Dengan kata lain, metode **K-Means Clustering** bertujuan untuk meminimalisasikan **objective function** yang diset dalam proses clustering dengan cara **meminimalkan variasi antar data yang ada di dalam suatu cluster dan memaksimalkan variasi dengan data yang ada di cluster lainnya**

K-Means Clustering



Algoritma

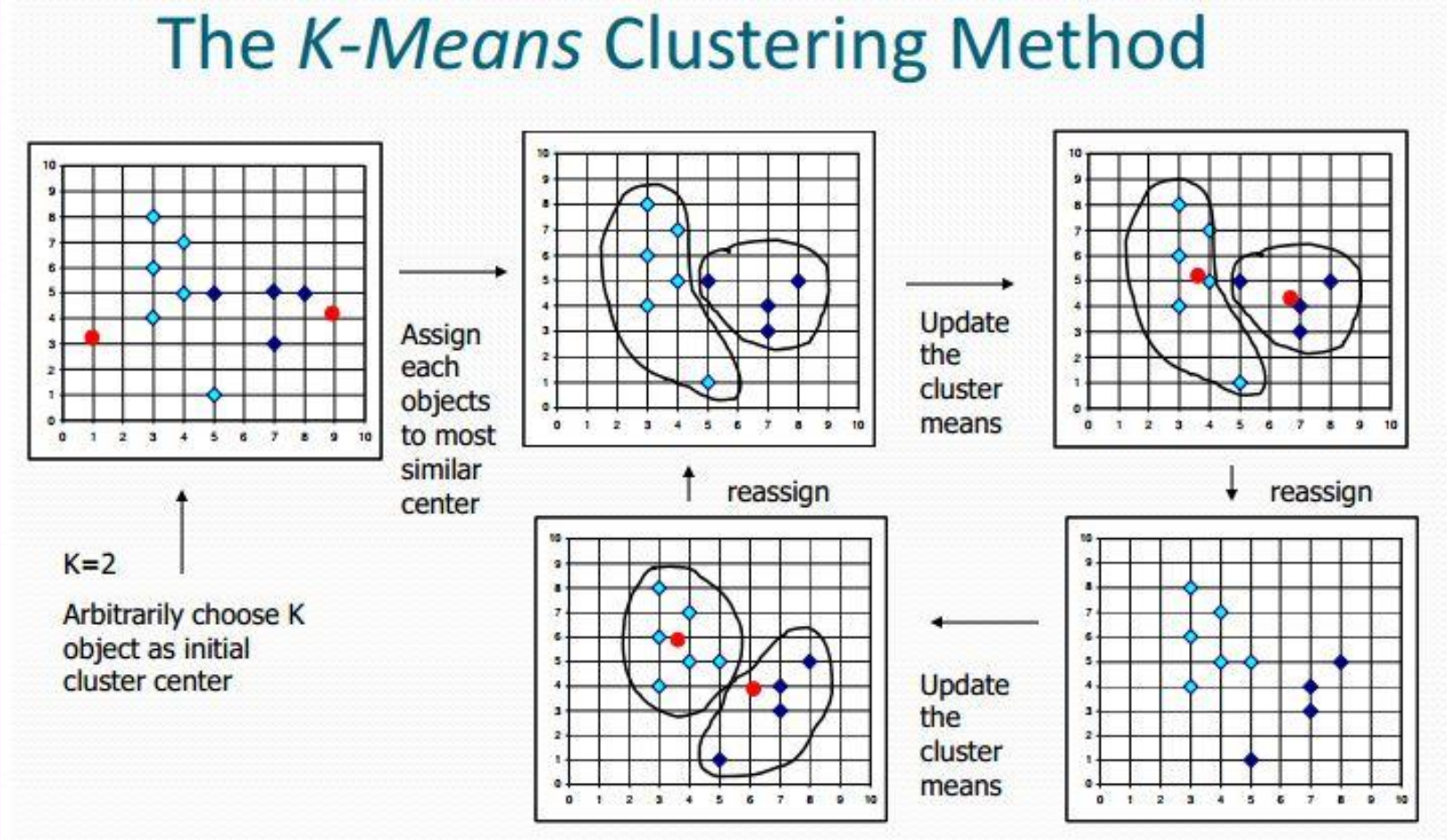
- Data *clustering* menggunakan metode **K-Means Clustering** ini secara umum dilakukan dengan algoritma dasar sebagai berikut:

1. Tentukan jumlah cluster
2. Alokasikan data ke dalam cluster secara random
3. Hitung centroid/rata-rata dari data yang ada di masing-masing cluster
4. Alokasikan masing-masing data ke centroid/rata-rata terdekat
5. Kembali ke Step 3, apabila masih ada data yang berpindah cluster atau apabila perubahan nilai centroid, ada yang di atas nilai threshold yang ditentukan atau apabila perubahan nilai pada objective function yang digunakan di atas nilai threshold yang ditentukan

Inisialisasi Titik Pusat (centroid)

- Inisialisasi centroid dapat dilakukan dengan beberapa cara, contohnya 3 cara berikut:
  - Dipilih secara dinamik: Metode ini tepat digunakan jika data baru ditambahkan secara cepat dan banyak. Untuk menyederhanakan persoalan, inisial cluster dipilih dari beberapa data baru, misal jika data dikelompokkan menjadi 3 clusters, maka inisial cluster berarti 3 item pertama dari data.
  - Dipilih secara random: Paling banyak digunakan, dimana inisial cluster dipilih secara random dengan range data antara nilai terendah sampai nilai tertinggi..
  - Memilih dari batasan nilai tinggi dan rendah: tergantung pada tipe datanya, nilai data tertinggi dan terendah dipilih sebagai inisial cluster.

K-Means Clustering



Kelebihan dan Kekurangan

- Kelebihan:
  - Menggunakan prinsip yang sederhana, dapat dijelaskan dalam non-statistik
  - Waktu yang dibutuhkan untuk menjalankannya relatif cepat
  - Sangat fleksibel, dapat dengan mudah diadaptasi.
  - Sangat umum digunakan
- Kekurangan:
  - Karena menggunakan k buah acak, tidak di jamin untuk menemukan kumpulan cluster yang optimal
  - dapat terjadinya *curse of dimensionality*, apabila jarak antara cluster yang satu dengan yang lain memiliki banyak dimensi.
  - Tidak optimal digunakan untuk data yang jumlahnya terlalu banyak sampai bermiliyar.

Contoh K-Means, step by step

- Berikut ini akan diilustrasikan metode K-Means untuk mengelompokkan data berikut:

	A	B	C
1	Subject	A	B
2	1	1.0	1.0
3	2	1.5	2.0
4	3	3.0	4.0
5	4	5.0	7.0
6	5	3.5	5.0
7	6	4.5	5.0
8	7	3.5	4.5

Inisialisasi centroid

- Data ini dikelompokkan ke dalam dua cluster. Misal A & B adalah data yang memiliki jarak yang paling jauh (euclidean distance), maka kita dapatkan data 1 dan data 4 sebagai inisialisasi centroid

	A	B	C
1	Subject	A	B
2	1	1.0	1.0
3	2	1.5	2.0
4	3	3.0	4.0
5	4	5.0	7.0
6	5	3.5	5.0
7	6	4.5	5.0
8	7	3.5	4.5

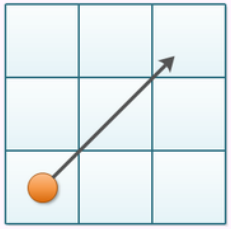
	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Menghitung jarak dari data yang lain, ke centroid

- Data yang lain dihitung jaraknya ke dua centroid tadi.

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Euclidean Distance



$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

			iterasi1	
Subject	A	B	Cluster1	Cluster2
1	1.0	1.0	0	7.211103
2	1.5	2.0	1.118034	6.103278
3	3.0	4.0	3.605551	3.605551
4	5.0	7.0	7.211103	0
5	3.5	5.0	4.716991	2.5
6	4.5	5.0	5.315073	2.061553
7	3.5	4.5	4.301163	2.915476
centroid1	1.0	1.0		
centroid2	5.0	7.0		

- Contoh perhitungan:

1.118034=sqrt((1.5-**1.0**)^2+(2.0-**1.0**)^2) ; 6.103278=sqrt((1.5-**5.0**)^2+(2.0-**7.0**)^2)

Membandingkan jarak ke centroid dari tiap cluster

- Dibandingkan jarak ke centroid dari tiap cluster, dicari yang terpendek untuk menjadi anggota dari cluster tersebut.

			iterasi1	
Subject	A	B	Cluster1	Cluster2
1	1.0	1.0	0	7.211103
2	1.5	2.0	1.118034	6.103278
3	3.0	4.0	3.605551	3.605551
4	5.0	7.0	7.211103	0
5	3.5	5.0	4.716991	2.5
6	4.5	5.0	5.315073	2.061553
7	3.5	4.5	4.301163	2.915476
centroid1	1.0	1.0		
centroid2	5.0	7.0		

- Contoh: perhatikan data-2 (1.5,2.0), karena 1.118034 (jarak dengan cluster1) < 6.103278 (jarak dengan cluster 2) maka data-2 termasuk dalam cluster1

Mencari titik centroid yang baru

- Untuk mencari titik centroid yang baru, kita perhatikan, keanggotaan dari tiap cluster.
- Cluster1 terdiri dari data: ((1,1),(1.5,2),(3,4)) maka titik centroid dari cluster1 ini ada di :

– X = (1+1.5+3)/3 =1.83333

– Y = (1+2+4)/3= 2.33333 • Sehingga kita dapatkan titik centroid yang baru sebagai berikut:

centroid1	1.833333	2.333333
centroid2	4.125	5.375

			iterasi1	
Subject	A	B	Cluster1	Cluster2
1	1.0	1.0	0	7.211103
2	1.5	2.0	1.118034	6.103278
3	3.0	4.0	3.605551	3.605551
4	5.0	7.0	7.211103	0
5	3.5	5.0	4.716991	2.5
6	4.5	5.0	5.315073	2.061553
7	3.5	4.5	4.301163	2.915476
centroid1	1.0	1.0		
centroid2	5.0	7.0		

Mencari titik centroid yang baru

- Untuk mencari titik centroid yang baru, kita perhatikan, keanggotaan dari tiap cluster.
- Cluster1 terdiri dari data: ((1,1),(1.5,2),(3,4)) maka titik centroid dari cluster1 ini ada di :

– X = (1+1.5+3)/3 =1.83333

– Y = (1+2+4)/3= 2.33333 • Sehingga kita dapatkan titik centroid yang baru sebagai berikut:

centroid1	1.833333	2.333333
centroid2	4.125	5.375

			iterasi1	
Subject	A	B	Cluster1	Cluster2
1	1.0	1.0	0	7.211103
2	1.5	2.0	1.118034	6.103278
3	3.0	4.0	3.605551	3.605551
4	5.0	7.0	7.211103	0
5	3.5	5.0	4.716991	2.5
6	4.5	5.0	5.315073	2.061553
7	3.5	4.5	4.301163	2.915476
centroid1	1.0	1.0		
centroid2	5.0	7.0		

- Untuk memulainya kita akan menerapkan dan menerapkan K-means dataset 2 dimensi sederhana untuk mengetahui tentang cara kerja K-Means
- Algoritma akan:
  - Inisialisasi secara acak centroid awal untuk setiap kluster
  - berulang kali menetapkan titik data ke kluster terdekat
  - hitung ulang centroid dari setiap cluster

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from scipy.io import loadmat
%matplotlib inline
```

In [2]:
```
# Assign every training example x^(i) to its closest centroid, given the current centroid positions
def find_closest_centroids(X, centroids):
    m = X.shape[0]   # number of training examples
    k = centroids.shape[0]
    idx = np.zeros(m)   # idx is the index of the centroid that is clostest to x^(i)

    for i in range(m):
        min_dist = 1000000
        for j in range(k):
            dist = np.sum((X[i,:] - centroids[j,:]) ** 2)
            if dist < min_dist:
                min_dist = dist
                idx[i] = j

    return idx
```

In [3]:
```
data = loadmat('./data/ex7data2.mat')
#import scipy.io as sio
#da = sio.loadmat('./data/ex7data2.mat')

X = data['X']
initial_centroids = initial_centroids = np.array([[3, 3], [6, 2], [8, 5]])

idx = find_closest_centroids(X, initial_centroids)
idx[0:3]
```

Out[3]:
```
array([0., 2., 1.])
```

In [4]:
```
def compute_centroids(X, idx, k):
    m, n = X.shape
    centroids = np.zeros((k, n))

    for i in range(k):
        indices = np.where(idx == i)
        centroids[i,:] = (np.sum(X[indices,:], axis=1) / len(indices[0])).ravel()

    return centroids
```

In [5]:
```
compute_centroids(X, idx, 3)
```

Out[5]:
```
array([[2.42830111, 3.15792418],
       [5.81350331, 2.63365645],
       [7.11938687, 3.6166844 ]])
```

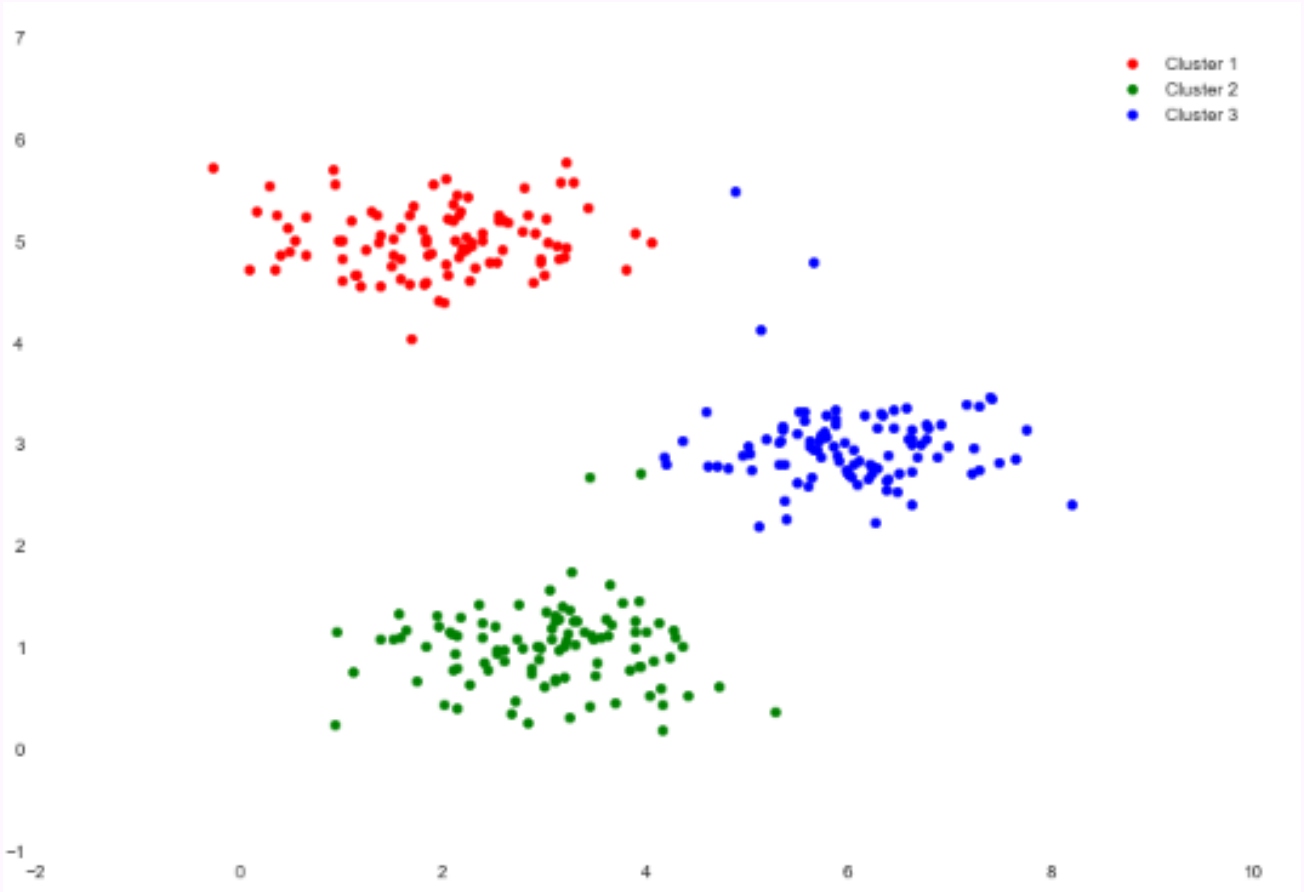
- Full algorithm:

In [6]:
```
def run_k_means(X, initial_centroids, max_iters):
    m, n = X.shape
    k = initial_centroids.shape[0]
    idx = np.zeros(m)
    centroids = initial_centroids

    for i in range(max_iters):
        idx = find_closest_centroids(X, centroids)
        centroids = compute_centroids(X, idx, k)

    return idx, centroids
```

In [7]:
```
idx, centroids = run_k_means(X, initial_centroids, 10)
```

In [8]:
```
cluster1 = X[np.where(idx == 0)[0],:]
cluster2 = X[np.where(idx == 1)[0],:]
cluster3 = X[np.where(idx == 2)[0],:]

fig, ax = plt.subplots(figsize=(12,8))
ax.scatter(cluster1[:,0], cluster1[:,1], s=30, color='r', label='Cluster 1')
ax.scatter(cluster2[:,0], cluster2[:,1], s=30, color='g', label='Cluster 2')
ax.scatter(cluster3[:,0], cluster3[:,1], s=30, color='b', label='Cluster 3')
ax.legend()
```

- result:



Referensi

- https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html
- https://id.wikipedia.org/wiki/K-means
- http://www.bigendiandata.com/2017-04-18-Jupyter_Customer360/
- https://informatikalogi.com/algoritma-k-means-clustering/