Machine Learning

Indah Agustien Siradjuddin

K-Means Clustering ¶

Semester Gasal 2019-2020

Definisi

1. Unsupervised Learning
2. Tujuan : mengelompokkan sejumlah *unlabeled data* berdasarkan kesamaan antara data tersebut

Berikut contoh *unlabeled data* (tanpa ada target)

dapat dikelompokkan menjadi dua buah grup/kelompok

atau dapat dikelompokkan menjadi empat buah grup/kelompok

Implementasi

1. Pengolompokan user dari data social media
2. Sistem Rekomendasi
3.
4.

Algoritma

1. Center cluster sebanyak jumlah kelompok k dibangkitkan secara acak.
2. For each observation:
3. Untuk setiap data pelatihan:
    - Hitung jarak antara data dengan titik tengah seluruh *cluster*, dengan menggunakan perhitungan jarak *euclidien distance*
    - Data dimasukkan ke dalam kelompok (*cluster*) jika data memiliki jarak terdekat dengan *cluster center*
4. Titik tengah cluster diperbaharui ketika terdapat penambahan data baru (hitung rata-rata)
5. Lakukan langkah 2 dan langkah 3 sampai tidak ada data lagi yang akan dikelompokkan

K-Means Clustering dengan Scikit

In [1]:

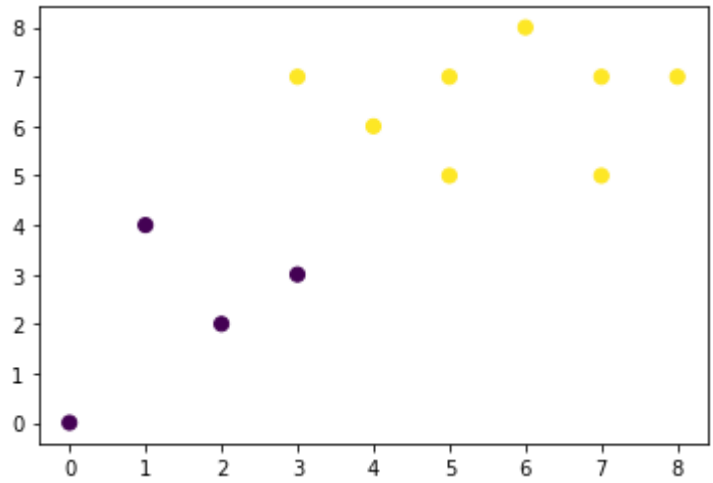
```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

data=np.array([[7,5],[5,7],[7,7],[3,3],[4,6],[1,4],[0,0],[2,2],[8,7],[6,8],[5,5],[3,7
]])

est = KMeans(2)  # 2 clusters

est.fit(data)
y_kmeans = est.predict(data)

plt.scatter(data[:, 0], data[:, 1], c=y_kmeans,s=50);
```



In [2]:

```
print(est.cluster_centers_)
print(y_kmeans)
```

```
[[1.5   2.25 ]
 [5.625 6.5  ]]
[1 1 1 0 1 0 0 0 1 1 1 1]
```

Code - K-Means Clustering

In [3]:

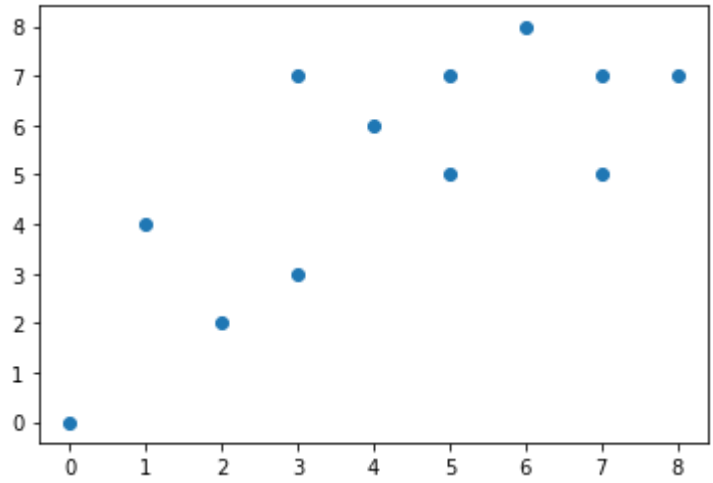
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:

```
data=np.array([[7,5],[5,7],[7,7],[3,3],[4,6],[1,4],[0,0],[2,2],[8,7],[6,8],[5,5],[3,7
]])
centroid=np.array([[4,6],[5,5]],dtype=float)
numOfData=len(data)
plt.scatter(data[:, 0], data[:, 1])
```

Out[4]:

```
<matplotlib.collections.PathCollection at 0x205104a9860>
```



In [5]:

```
import kmeans
```

In [6]:

```
data=np.array([[7,5],[5,7],[7,7],[3,3],[4,6],[1,4],[0,0],[2,2],[8,7],[6,8],[5,5],[3,7
]])
centroid=np.array([[4,6],[5,5]],dtype=float)
dist=np.zeros((numOfData,2))
cluster=np.zeros(numOfData,dtype=int)
c=[[],[]]
centroid=kmeans.trainingKMeans(4,data,centroid)
```

In [7]:

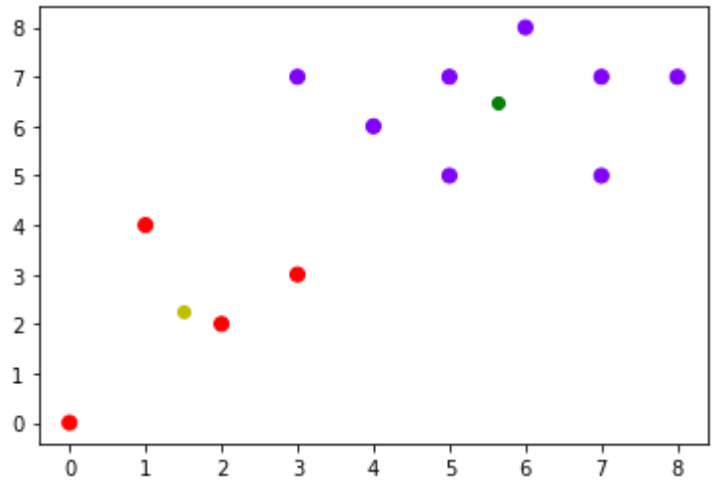
```
print(centroid)
```

```
[[5.625 6.5  ]
 [1.5   2.25 ]]
```

In [8]:

```
#testing
cluster=np.zeros(numOfData,dtype=int)
data=np.array([[7,5],[5,7],[7,7],[3,3],[4,6],[1,4],[0,0],[2,2],[8,7],[6,8],[5,5],[3,7
]])
for i in range (numOfData):
    dist[0,0]=np.linalg.norm(data[i]-centroid[0,:])
    dist[0,1]=np.linalg.norm(data[i]-centroid[1,:])
    cluster[i]=np.argmin(dist[0])
print(cluster)
plt.scatter(data[:, 0], data[:, 1], c=cluster,s=50,cmap='rainbow');
plt.plot(centroid[0,0],centroid[0,1],'go')
plt.plot(centroid[1,0],centroid[1,1],'yo')
plt.show()
```

```
[0 0 0 1 0 1 1 1 0 0 0 0]
```



Dataset:
clustering

Proses Pengelompokkan dengan epoch=3

epoch - 1
clustering
epoch - 2

clustering
epoch - 3

clustering

Tantangan

Pemilihan *center cluster* pada inisialisasi sangatlah penting untuk data pelatihan. Jika *center cluster* tidak dipilih secara tepat, maka akan menimbulkan *local optimum*.

clustering

Perhitungan Akurasi

Algoritma *K-means Clustering* adalah algoritma *unsupervised learning*, oleh karena itu *data training* adalah data pelatihan yang tidak memiliki target label, sehingga tidak dapat dibuat *confusion matrix*. Perhitungan akurasi dihitung dengan menggunakan **silhoutte coefficient**, dimana kinerja *clustering* dihitung berdasarkan *compactness* and *separation* dari sebuah *cluster*.

$$s = \frac{b - a}{max(a, b)}$$

dimana
a= rata-rata jarak antara data dengan data lain di dalam *cluster* yang sama
b= rata-rata jarak antara data dengan data lain dari *cluster* yang terdekat

Silhoutte Coefficient dengan Scikit

In [9]:

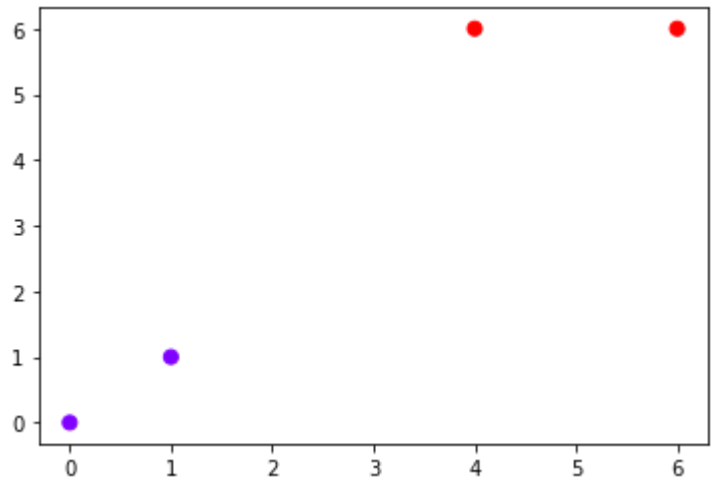
```
from sklearn.cluster import KMeans
from sklearn import metrics


data=np.array([[6,6],[4,6],[1,1],[0,0]])
centroid=np.array([[4,6],[0,0]],dtype=float)

est = KMeans(2)  # 2 clusters
est.fit(data)
y_kmeans = est.predict(data)
plt.scatter(data[:, 0], data[:, 1], c=y_kmeans,s=50,  cmap='rainbow');
print('score=', metrics.silhouette_score(data, est.labels_,metric='euclidean'))

print(y_kmeans)
```

```
score= 0.7591876061783829
[1 1 0 0]
```



Code : Silhoutte Coefficient

In [10]:

```
# assume that data is clustered
a=np.linalg.norm(data[0,:]-data[1,:])
b1=np.linalg.norm(data[0,:]-data[2,:])
b2=np.linalg.norm(data[0,:]-data[3,:])
b=(b1+b2)/2
print(a,b1,b2,b)

coef=(b-a)/np.maximum(a,b)
print(coef)
```

```
2.0 7.0710678118654755 8.48528137423857 7.7781745930520225
0.74287026138671
```

In [ ]:

```
```