Machine Learning

Tugas 2

Agung Nursatria Banyuwiguna

1301150073

IF-39-03

Program Studi S1 Teknik Informatika – Fakultas Informatika

Universitas Telkom

Jalan Telekomunikasi Terusan Buah Batu, Bandung

Indonesia

1. **Analisis Masalah**

   Diketahui terdapat 688 titik yang terpencar dalam grafik/plot 2 dimensi dengan tanpa label. Dibutuhkan suatu pengelompokkan titik - titik tersebut secara optimal dengan menggunakan metode k-means. Untuk mendapatkan pengelompokkan yang optimal, dibutuhkan centroid dengan jumlah tepat untuk menjadi titik tengah dari kelompok yang ada. Jumlah tepat centroid ditentukan berdasarkan SSE yang merupakan jarak dari anggota ke centroid. Digunakan metode elbow yang akan menyatakan jumlah centroid optimal dimana k turun drastis terakhir. Disini penulis menyatakan turun drastis jika penurunan SSE setidaknya lebih dari 30% penurunan pertama.

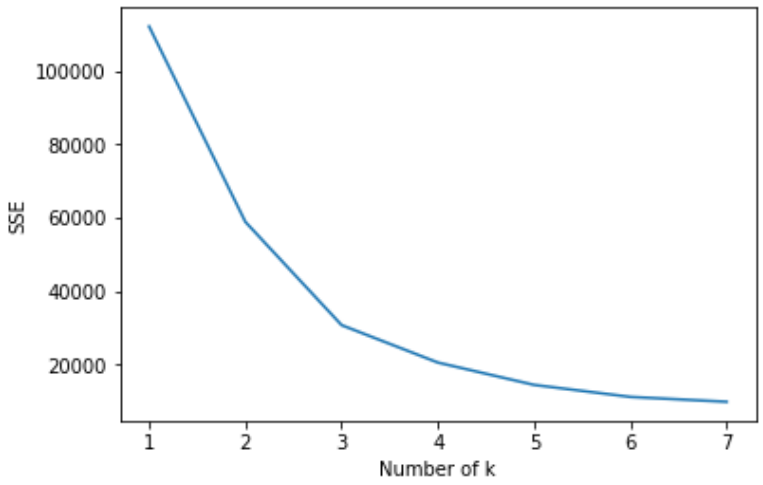
2. **Desain System**

   Berikut langkah untuk membangun sistem k-means:

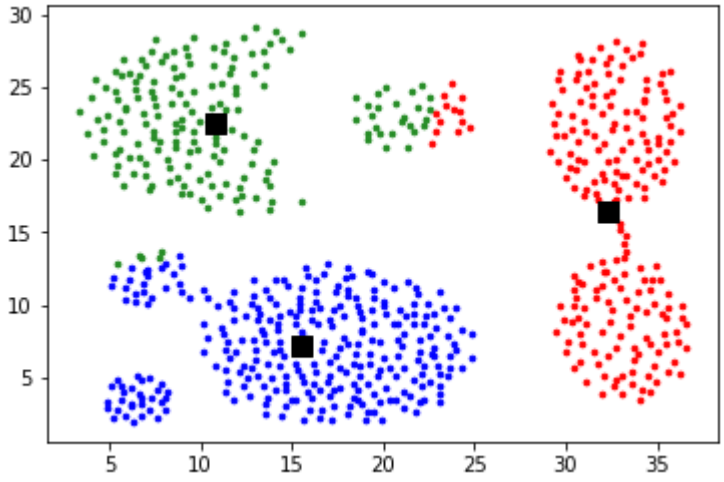
   1. Load data dari file
   2. Cari K optimal dengan method elbow
   3. Inisialisasi centroid sejumlah K
   4. Lakukan clustering menentukan kelompok dari titik-titik yang ada berdasarkan jarak ke centroid
   5. Inisialisasi centroid baru
   6. ulangi langkah 4 dan 5 hingga centroid lama dan centroid baru ditempat yang sama
   7. Keluarkan hasil
   8. Simpan hasil clustering [opsional]

3. **Evaluasi Hasil Running**

   Dengan data training:



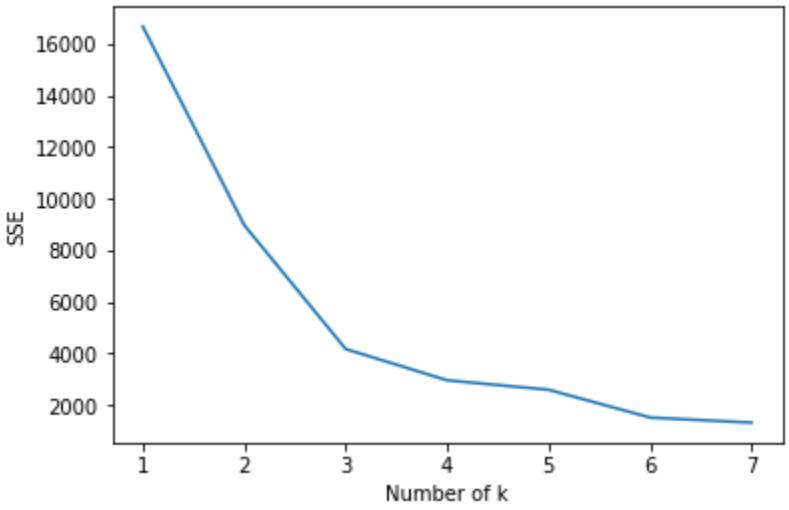
Berdasarkan method elbow pada grafik diatas, ditentukan k optimal yaitu 3. Dikarenakan setelah 3, penurunan tidak drastis lagi.



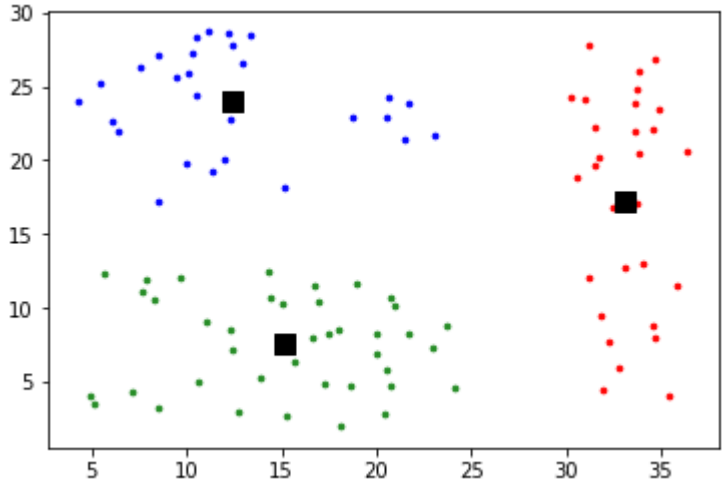
Dengan k=3, didapat hasil clustering seperti gambar diatas dengan perbandingan masing – masing titik:

```
cluster
total   : 688 titik
0       : 216 titik
1       : 294 titik
2       : 178 titik
```

Dengan data testing:



Berdasarkan method elbow pada grafik diatas, ditentukan k optimal yaitu 3. Dikarenakan setelah 3, penurunan tidak drastis lagi.



Dengan k=3, didapat hasil clustering seperti gambar diatas dengan perbandingan masing – masing titik:

```
cluster
total   : 100 titik
0       : 42 titik
1       : 28 titik
2       : 30 titik
```

4. **Screenshot program**

   Main.py

```python
import numpy as np
from matplotlib import pyplot as plt
from elbow import optimal_k
from kmeans import initCentroid,clustering

# Load data dan jadikan array
data = np.loadtxt('TrainsetTugas2.txt')
x_data = data[:,0]
y_data = data[:,1]

k= optimal_k(data)

# koordinat X,Y centroid data random
centroid = initCentroid(data,k)

# clustering
clusters = clustering(data,centroid,k)

# Buat Plot
colors = ['blue', 'forestgreen', 'red', 'darkgoldenrod', 'purple', 'cyan']
fig, ax = plt.subplots()
print('cluster')
print('total\t:', len(data),'titik')
for i in range(k):
        titik = np.array([data[j] for j in range(len(data)) if clusters[j] == i])
        print(i,'\t:',len(titik),'titik')
        ax.scatter(titik[:, 0], titik[:, 1], s=7, c=colors[i])
ax.scatter(centroid[:, 0], centroid[:, 1], marker='s', s=100, c='black')

# simpan result ke txt
result = []
result = np.column_stack((data, clusters))
np.savetxt('result_trains.txt', result, newline='\r\n', fmt="%s\t%s\t%i")
```

Test.py

```python
import numpy as np
from matplotlib import pyplot as plt
from elbow import optimal_k
from kmeans import initCentroid,clustering

# Load data dan jadikan array
data = np.loadtxt('TestsetTugas2.txt')
x_data = data[:,0]
y_data = data[:,1]

k= optimal_k(data)

# koordinat X,Y centroid data random
centroid = initCentroid(data,k)

# clustering
clusters = clustering(data,centroid,k)

# Buat Plot
colors = ['blue', 'forestgreen', 'red', 'darkgoldenrod', 'purple', 'cyan']
fig, ax = plt.subplots()
print('cluster')
print('total\t:', len(data),'titik')
for i in range(k):
        titik = np.array([data[j] for j in range(len(data)) if clusters[j] == i])
        print(i,'\t:',len(titik),'titik')
        ax.scatter(titik[:, 0], titik[:, 1], s=7, c=colors[i])
ax.scatter(centroid[:, 0], centroid[:, 1], marker='s', s=100, c='black')

# simpan result ke txt
result = []
result = np.column_stack((data, clusters))
np.savetxt('result_test.txt', result, newline='\r\n', fmt="%s\t%s\t%i")
```

Euclidean.py

```python
import numpy as np

# Euclidean distance untuk banyak data centroid
def euclid(data, centroid, ax=1):
    return np.linalg.norm(data - centroid, axis=ax)
```

Kmeans.py

```python
import numpy as np
from copy import deepcopy
from euclidean import euclid

def initCentroid(data,k):
    x_centroid = np.random.randint(0, np.max(data), size=k)
    y_centroid = np.random.randint(0, np.max(data), size=k)
    centroid = np.array(list(zip(x_centroid, y_centroid)), dtype=np.float32)
    return centroid

def clustering(data, centroid, k):
    old_centroid = np.zeros(centroid.shape)
    centers = np.zeros(len(data))
    sameLocation = euclid(centroid, old_centroid, None)
    while sameLocation != 0:
        for i in range(len(data)):
            distances = euclid(data[i], centroid)
            center = np.argmin(distances)
            centers[i] = center
        old_centroid = deepcopy(centroid)
        for i in range(k):
            titik = [data[j] for j in range(len(data)) if centers[j] == i]
            centroid[i] = np.mean(titik, axis=0)
        sameLocation = euclid(centroid, old_centroid, None)
    return centers
```

Elbow.py

```python
from kmeans import clustering,initCentroid
from matplotlib import pyplot as plt

def optimal_k(data):
    # mencari daftar sse
    sse = []
    for k in range(1,8):
        centroid = initCentroid(data,k)
        clusters = clustering(data,centroid,k)
        sumtoCluster = 0
        for i in range(k):
            for j in range(len(data)):
                if clusters[j] == i:
                    # Euclidean Distance untuk sumtoCluster
                    sumtoCluster += (abs((data[j,0]-centroid[i][0])**2) +
                                     abs((data[j,1]-centroid[i][1])**2))
        sse.append([k,sumtoCluster])

    # Plot grafik kluster
    sseKey = [sse[j][0] for j in range(len(sse))]
    sseValue = [sse[j][1] for j in range(len(sse))]
    plt.figure()
    plt.plot(sseKey, sseValue)
    plt.xlabel("Number of k")
    plt.ylabel("SSE")
    plt.show()

    # mencari k optimal dengan patokan jika nilai yang berkurang lebih sam
    k_optimal = 2
    first_drop_sse = (sse[0][1]-sse[1][1])
    for k in range(1,5):
        if (sse[k][1]-sse[k+1][1] >= (first_drop_sse*0.3)):
            k_optimal = sse[k+1][0]
    return k_optimal
```