LAPORAN TUGAS 2.1

MACHINE LEARNING

K-*Means Clustering*

Disusun Oleh:
Aditya Alif Nugraha
1301154183
IF 39-01

PRODI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2018

Analisa Masalah

Pada tugas 2.0 ini, mahasiswa diuji kemampuannya untuk menganalisa, mendesain, dan mengimplementasi salah satu algoritma *Unsupervised Learning* yaitu *K-Means Clustering.* Metode tersebut adalah salah satu algoritma *clustering*, yaitu mengelompokkan data yang tidak memiliki label/kelas.

Pada tugas ini, diberikan 2 jenis data yaitu Train Set dan Test Set. Kedua data tersebut tidak memiliki label. Mahasiswa diminta mengkelompokkan data tersebut menggunakan algoritma K-Means Clustering. Hasil kluster yang baik dapat dilihat dari visualisasi data yang telah dikelompokkan. Tetapi, harus dicari terlebih dahulu jumlah K atau jumlah centroid agar data-data tersebut memiliki jumlah kelas yang benar. Akan digunakan ***elbow method*** untuk mencari jumlah K pada tugas ini.

Desain

Algoritma K-Means

Berikut adalah langkah-langkah mengcluster data dengan K-Means Clustering:

1. Bangkitkan K buah titik centroid secara acak.
2. Bentuk K buah cluster dengan mengelompokkan tiap data ke centroid terdekatnya.
3. Hitung kembali titik centroid dari rata-rata koordinat data yang telah dicluster pada langkah 2.
4. Ulangi langkah 2 dan 3 hingga titik centroid tidak berubah.

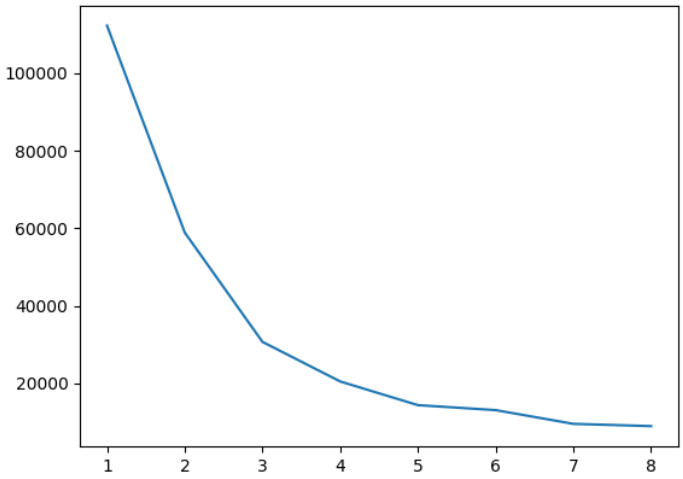
Hasil training dari K-Means yaitu anggota kluster dan titik centroid yang telah diupdate. Anggota kluster yang dihasilkan akan digunakan untuk menghitung ***Sum Squared Error* (SSE).** Sedangkan, titik centroid akan dijadikan pusat dari kluster yang akan dibentuk pada memprediksi test set.

Menentukan Jumlah K (Centroid)

Salah satu parameter terpenting dalam K-Means Clustering yaitu jumlah K. Hal tersebut menjadi penting karena jumlah K akan menentukan jumlah kelompok/kelas pada data tersebut. Salah satu cara menentukan Jumlah K yaitu dengan ***Elbow Method*.**

Elbow Method adalah metode interpretasi dan validasi konsistensi dalam *cluster analysis* yang dirancang untuk membantu menemukan jumlah kluster yang sesuai dalam dataset.

Dalam K-Means Clustering, digunakan ***Sum Squared Error* (SSE)** untuk membuat plot yang nantinya akan dianalisa dengan Elbow Method. Berikut adalah hasil plotting SSE dengan range jumlah K=1 hingga 8:



Dari gambar plot diatas, dapat dilihat bahwa K=5 dan K=6 terlihat stabil nilai SSEnya. **Hal tersebut menjadikan alasan untuk tugas ini akan digunakan K=5 untuk jumlah K/banyaknya centroid.**

Implementasi

Main Function

```
from fun import load_data, visualize_data, write_to_file

import KMeans as kmeans

if __name__ == '__main__':
    # Train model K-Means dan visualisasi hasil prediksi train set
    X_train = load_data("TrainsetTugas2.txt")
    centroids, clusters = kmeans.train(5, X_train, 300)
    y_train = kmeans.predict(X_train, centroids)
    visualize_data(X_train, y_train, centroids)
    print(kmeans.getSumSquaredError(centroids, clusters))

    # Prediksi class test set dan menuliskan ke file hasil.csv
    X_test = load_data("TestsetTugas2.txt")
    y_test = kmeans.predict(X_test, centroids)
    write_to_file("hasil_running_nanang.csv", y_test)
    visualize_data(X_test, y_test, centroids)
```

K-Means Core Code

```
import random
import numpy as np


def init_centroids(k, X):
    """Membangkitkan nilai acak untuk menentukan titik awal centroid."""
    centroids = []
    for _ in range(k):
        np.random.seed(100)
        centroids.append([random.random()*max(X[:, 0]),
                          random.random()*max(X[:, 1])])
    return np.array(centroids)
```

```
def train(k, X, iteration):
    """Mencari titik centroid terbaru yang dapat mengcluster data dengan
baik."""
    centroids = init_centroids(k, X)
    old_centroids = np.zeros_like(centroids)

    for _ in range(iteration):
        clusters = {i: [] for i in range(k)}  # Mengosongkan anggota
cluster.

        # Membuat kluster dengan mengelompokkan data sesuai centroid
terdekatnya.
        for row in X:
            distances = np.linalg.norm(row-centroids, axis=1)
            clusters[np.argmin(distances)].append(row)

        # Digunakan untuk menghitung kondisi konvergen
        old_centroids[:, :] = centroids[:, :]

        # Mengupdate titik centroid dengan rata-rata anggota kluster
centroid tersebut.
        for i in range(len(centroids)):
            centroids[i] = np.mean(clusters[i], axis=0)

        # Kondisi berhenti saat tidak ada perubahaan titik di semua
centroid
        if np.linalg.norm(centroids-old_centroids, axis=None) == 0:
            break
    return np.array(centroids), clusters


def predict(X, centroids):
    """Memprediksi class dari data dengan input cluster yang telah
ditrain."""
    return [np.argmin(np.linalg.norm(row-centroids, axis=1)) for row in X]


def getSumSquaredError(centroids, clusters):
    """Mendapatkan nilai Sum Squared Error."""
    sse = 0
    for centroid, members in clusters.items():
        sse += np.sum((centroids[centroid]-members) ** 2)
    return sse
```

Additional Function

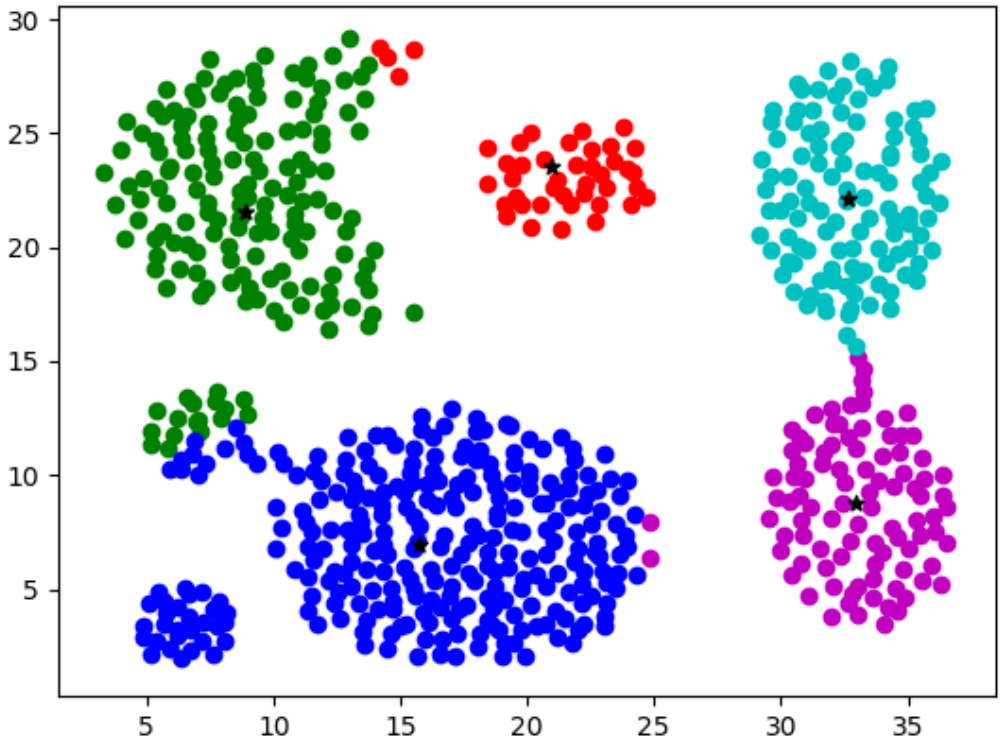
```
import numpy as np

import matplotlib.pyplot as plt
import csv

def load_data(filename):
    """Membuka file data dan merubah menjadi array numpy."""
    file = open(filename, "r")
    lines = file.readlines()

    data = []
    for line in lines:
        data.append(line.split("\n")[0].split("\t"))

    return np.array(data, dtype=float)


def visualize_data(X, y=None, centroids=[]):
    """Memvisualisasikan data dan/atau centroid."""
    if centroids == []:
        plt.scatter(X[:, 0], X[:, 1])
    else:
        colors = ["b", "g", "r", "c", "m", "y", "k"]
        for i in range(len(y)):
            plt.scatter(X[i, 0], X[i, 1], c=colors[y[i]])
        plt.scatter(centroids[:, 0], centroids[:, 1], marker="*", c="k")
    plt.show()


def write_to_file(filename, result):
    with open(filename, "w") as csv_file:
        writer = csv.writer(csv_file, delimiter=',')
        for res in result:
            writer.writerow(str(res+1))
    csv_file.close()
```

Elbow Plotter

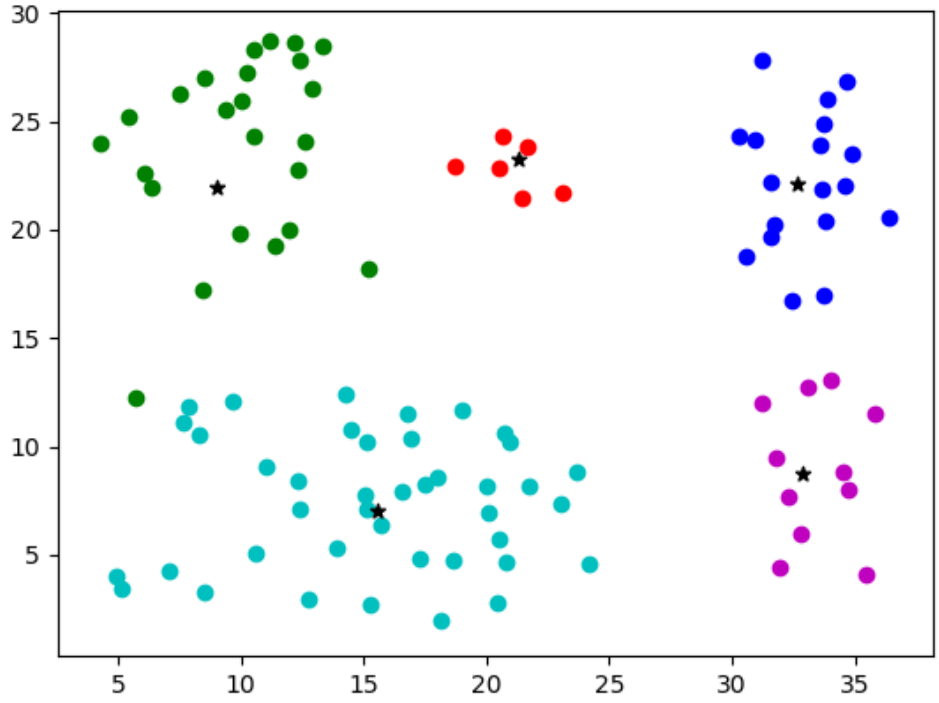
```
import KMeans as kmeans

import matplotlib.pyplot as plt
from fun import load_data

X_train = load_data("TrainsetTugas2.txt")

err = []
idx = []
for i in range(1, 9):
    print(i)
    idx.append(i)
    centroids, clusters = kmeans.train(i, X_train, 300)
    err.append(kmeans.getSumSquaredError(centroids, clusters))

print(err)
plt.plot(idx, err)
plt.show()
```

Hasil

Hasil Visualisasi Train Set



Hasil Visualisasi Test Set



NB: Untuk hasil kluster test set dapat dilihat lebih jelas pada file hasil.csv