Hendro Susanto

1301160284

IF-4012

Tugas 2

1. Kelebihan dan Kekurangan *k-means Clustering*

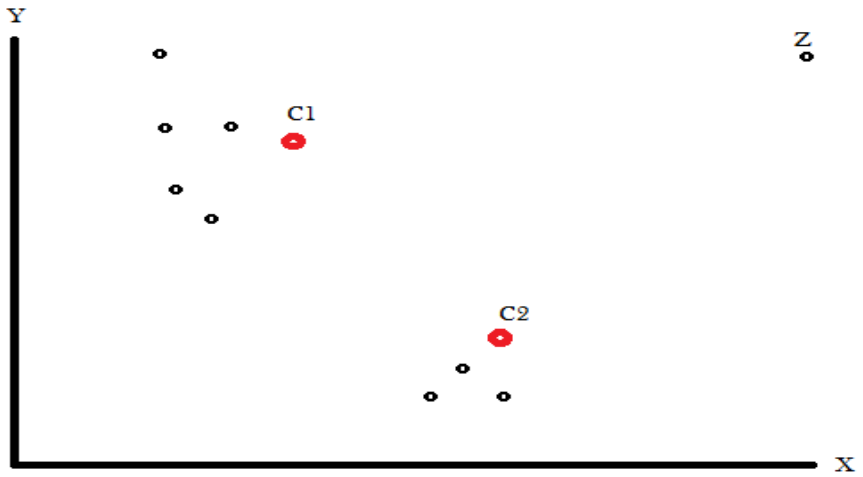
   Kelebihan :

   - Selalu mampu melakukan klasterisasi sampai posisi centroid stabil.
   - Menggunakan kalkulasi yang sederhana sehingga mudah dijelaskan.
   - Waktu yang dibutuhkan relatif cepat karena tidak melakukan kalkulasi yang rumit.
   - Mudah dilakukan karena setiap iterasi melakukan langkah-langkah yang sama dari awal sampai akhir.

   Kekurangan :

   - Sangat bergantung pada centroid awal karena dibangkitkan secara random, sehingga apabila posisi centroid awal didapatkan kurang baik maka klasterisasi yang didapatkan menjadi tidak optimal
   - Dalam kasus data berdimensi banyak maka akan sulit dilakukan karena setiap data akan dihitung jaraknya ke centroid
   - Menjadi tidak optimal ketika ada data yang terisolasi, karena akan mempengaruhi nilai rata-rata jarak ke centroid secara signifikan.

   Contoh :

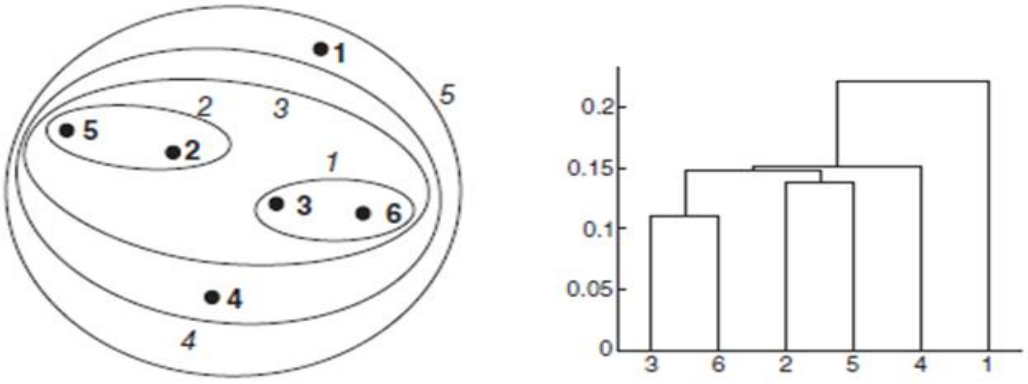


Dalam contoh kasus diatas proses klasterisasi dilakukan dengan menghitung jarak centroid C1 dan centroid C2 ke setiap data. Posisi C1 dan C2 yang dibangkitkan secara random dapat dikatakan cukup baik karena tidak jauh dari kumpulan data.Setiap iterasi akan dipudate posisi centroid berdasarkan nilai rata-rata jarak data ke centoidnya pada setiap klaster. Permasalahan muncul ketika data Z terlihat terisolir dari kumpulan data lainnya, sehingga ini akan mempengaruhi nilai rata-rata dari klasternya karena memiliki jarak yang sangat jauh dari centroidnya.

2. Konsep Dasar *Agglomerative Hierarchical Clustering*

   Agglomerative Hierarchical Clustering adalah sebuah metode pengelompokkan data dengan konsep dimulai dari 1 buah klaster yang berisi 1 objek data lalu memilih klaster lainnya yang paling dekat untuk digabungkan menjadi 1 klaster. Proses ini dilakukan sampai dihasilkan 1 klaster akhir. Proses penggabungannya dapat dilakukan menggunakan beberapa pendekatan antara lain *single link, multi link, group average,* dan *mean.* Metode ini juga sering disebut *bottom-up* karena dilakukan dari bawah ke atas. Untuk mengetahui kedekatan dari 2 buah klaster biasanya digunakan sebuah matriks data yang berisi jarak antar data.lalu menggabungkan 2 klaster yang memiliki kemiripan berdasarkan metode pendekatan yang telah ditentukan.

   Contoh :



   Pada contoh diatas proses klasterisasi dimulai dengan menggabungkan objek 3 dengan 6 menjadi 1 klaster karena memiliki jarak paling dekat yaitu 0,1. Langkah selanjutnya menggabungkan objek 5 dan 2 yang memiliki jarak 0,14. Lalu menggabungkan klaster {3,6} dengan klaster {2,5} yang memiliki jarak 0,15. Lalu menggabungkan klaster {2,3,5,6} dengan objek 4 yang memiliki jarak 0,151. Langkah terakhir menggabungkan objek 1 ke klaster sehingga menjadi 1 klaster saja.

Laporan model klasterisasi metode *Self Organizing Map* (SOM)

1. Analisis Masalah

   Terdapat 600 data set yang berisi 2 atribut setiap atributnya memiliki nilai. Data-data tersebut belum memiliki label untuk mendefinisikan klasternya. Akan ditentukan label klaster dari setiap data set menggunakan metode *Self Organizing Map* (SOM).

2. Strategi Penyelesaian Masalah

   Strategi yang dilakukan untuk memberi label klaster adalah dengan menerapkan metode *Self Organizing Map* (SOM) pada program dengan menggunakan bahasa Java. Langkah-langkah nya sebagai berikut :

- Menetukan nilai parameter learning rate=0,1 dan topological neighbourhood =2

```
double Q=2; double n=0.1;
```

- Membangkitkan nilai-nilai neuron. Jumlah neuron yang dipilih adalah 11 karena setelah dilakukan beberapa kali percobaan 11 neuron memiliki nilai SSE yang paling kecil sehingga dapat disimpulkan yang **paling optimum**.

```
double[][] neuron = new double [11][2];

for (int i = 0; i < neuron.length; i++) {
    neuron[i][0]= (Math.random()*20);
    neuron[i][1]= (Math.random()*20);
}
```

- Melakukan perulangan sebanyak 100 kali

```
while (j<100){
```

- Melakukan perulangan sebanyak data objek (600)

```
   for (int i = 0; i < objek.length; i++) {
```

- Menghitung jarak setiap neuron ke objek yang dipilih, lalu dipilih neuron pemenang

```
int a=neuronterdekat(objek[i],neuron);

static double euclidan(double[] objek, double[] neuron){
    return Math.sqrt(
                Math.pow( ((objek[0])-neuron[0]),2)+
                Math.pow( ((objek[1])-neuron[1]),2));
}
```

```
static int neuronterdekat(double[] objek, double[][] neuron){
    int a=0;
    double b=1000;
    for (int i = 0; i < neuron.length; i++) {
       if(b>euclidan(objek,neuron[i])){
          b=euclidan(objek,neuron[i]);
          a=i;
       }
    }
    return a;
}
```

- Mengupdate nilai neuron pemenang dan neuron tetangganya (misal neuron pemenang adalah neuron ke 5 maka yang di update adalah neuron 4,5,6. Jika neuron pemenang adalah neuron ke 11 yang diupdate adalah neuron 10 dan 11)

```
static double deltaW(double nt,double[] neuronP, double[] neuron, double Q, double[] x, int idx){
   return nt*Tx(neuronP, neuron, Q)*(x[idx]-neuron[idx]);
}
```

```
static double Tx(double[] neuronP, double[] neuron, double Q){
    return Math.exp(-(euclidan(neuronP,neuron))/(2*(Math.pow(Q, 2))));
}
```

- Keluar dari perulangan for lalu mengupdate nilai learning rate dan topological neighbourhood

```
n = n*Math.exp(-j/2);
Q = Q*Math.exp(-j/2);
if (n<0.000001){
    n = 0.1; Q = 2;
}
```

- Lakukan sampai perulangan while berakhir (100 kali)

```
j++;
```

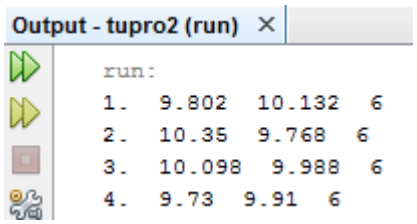
- Menghitung nilai SSE. Nilai SSE adalah jumlah jarak setiap objek ke neuron terdekatnya

```
for (int i = 0; i < objek.length; i++) {
    int a = neuronterdekat(objek[i],neuron);
    sse=sse+euclidan(objek[i],neuron[a]);
}
```

- Menampilkan data-data objek dan neuron terdekatnya sebagai label dan juga nilai SSE nya.

```
for (int i = 0; i < objek.length; i++) {
   System.out.println((i+1)+".  "+objek[i][0]+"  "+objek[i][1]+"  "+neuronterdekat(objek[i],neuron));
}System.out.println("Nilai SSE : "+sse);
```

Output :



...

```
159.  9.024  11.846  5
160.  10.296  11.61  5
161.  7.87  10.838  3
162.  8.164  10.534  3
163.  8.214  10.62  6
164.  8.166  10.698  6
165.  8.05  10.746  3
166.  7.978  11.13  6
```

...

```
597.  14.32  4.59  10
598.  13.636  5.218  10
599.  14.41  4.656  10
600.  14.02  5.614  10
Nilai SSE : 825.6146428182782
BUILD SUCCESSFUL (total time: 1 second)
```