Laporan Tugas

Tugas 2 Machine Learning dengan Menggunakan Metode Self Organizing Map (SOM)

Mata Kuliah

Machine Learning

Disusun Oleh :

M. Naufal Syawali. A

1301164488

S1 Teknik Informatika

Telkom University

Bandung

2019

1. K-Means

A. Kelebihan K-Means :

1. Algoritma K-Means salah satu algoritma yang umum dan sering digunakan maka dari itu akan ada banyak Referensinya.
2. Karena Algoritma K-Means banyak referensinya maka dari itu K-Means mudah untuk dipelari dan relative lebih cepat untuk dimengerti.
3. Dapat dijelaskan dengan cara non-statistik karena prinsip cukup mudah

B. Kekurangan K-Means :

1. Jika titik data sangat banyak misalkan 1.000.000 titik data maka perlu waktu yang lama untuk menentukan titik yang maksimal.
2. Adanya penggunaan nilai k yang random, jika random maka tidak ada jaminan untuk menentukan kumpulan cluster yang optimal.
3. Jika nilai K di inisialisasi dengan cara random maka pengelompokkan data yang didapatkan bisa berbeda-beda.

C. K-Means Clustering contoh kasus dan penjelasannya

Algoritma dari K-Means Sendiri

1. Inisialisasi, tentukan nilai K sebagai jumlah cluster.
2. Pilih K data dari data set sebagai centroid
3. Alokasikan semua data ke centroid terdekat dengan menghitung masing – masing jaraknya.
4. Hitung kembali centroid C berdasarkan data cluster masing – masing.

Sebagai contoh misalkan ada sebuah toko pizza yang ingin menjual produknya ke berbagai penjuru kota, maka toko pizza tersebut perlu membuat sebuah cabang dengan masing – masing koordinat tertentu, anggaplah titik koordinat seperti pada table :

Toko Ke-i (Data Ke-i)	Koordinat X	Koordinat Y
1	1	1
2	4	1
3	6	1
4	1	2
5	2	3
6	5	3
7	2	5
8	3	5
9	2	6
10	3	8

Berdasarkan data set tersebut, dilakukan proses pengelompokan menjadi 3 cluster (k = 3), berdasarkan k = 3, maka ditentukan titik centroid sebanyak k berdasarkan titik – titik koordinat tertentu. Jika misalkan data ke-2, 4, 6 dipilih sebagai centroid awalnya pengukuran jarak pada setiap data terhadap titik centroid dilakukan dengan perhitungan jarak Euclidean.

Rumus mencari jarak Euclidean

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

Anggaplah p = x dan q = y

Maka hasil dari centroid 3 data tersebut adalah, pada lokasi berikut inilah toko pizza tersebut perlu membuka cabang.

Centroid	X	Y
1	4	1
2	1	2
3	5	3

2. Agglomerative Hierarchical Clustering
   Agglomerative Hierarchical Clustering merupakan proses pengelompokkan dengan 2 atau lebih objek yang mempunyai kesamaan paling dekat, kemudian harus diteruskan ke objek lainnya yang terdekat, sehingga menjadi sebuah kesatuan yang membentuk hirarki. Algoritma ini memiliki beberapa metode seperti, Single Linkage, Complete Linkage, dan Average Linkage, tiga metode tersebut merupakan metode umum dipakai, adapun Kekurangan dan Kelebihannya sebagai berikut

   - Kekurangan : Sering terdapat kesalahan data outer, perbedaan ukuran jarak yang digunakan, variable yang tidak relevan
   - Kelebihan : Menghemat waktu dan mempercepat pengolahan data, yang akan membentuk hirarki jadi pengguna dapat mengetahui sistem pembelajaran dengan lebih mudah

   Contoh Diketahui :

Node	X	Y
P1	8	4
P2	9	7
P3	5	8
P4	4	6
P5	8	7
P6	5	2
P7	3	9
P8	1	8

Jawab :

a. Hitung jarak setiap node terhadap node lainnya menggunakan rumus Euclidean dan petakan pada table

   Rumus Euclidian Distance :

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

   Hasil yang didapat

	P1	P2	P3	P4	P5	P6	P7	P8
P1	0	3,16	5	4,47	3	3,61	7,07	8,06
P2	3,16	0	4,12	5,09	1	6,4	6,32	8,06
P3	5	4,12	0	2,24	3,16	6	2,24	4
P4	4,47	5,09	2,24	0	4,12	4,12	3,16	3,61
P5	3	1	3,16	4,12	0	5,83	5,39	7,07
P6	3,61	6,4	6	4,12	5,83	0	7,28	7,21
P7	7,07	6,32	2,24	3,16	5,39	7,28	0	2,24
P8	8,06	8,06	4	3,61	7,07	7,21	2,24	0

b. Dari data table diatas, cari 2 nilai minimum(paling kecil) pada setiap baris node. Maka didapatkan node p2 dan p5. Node-Node tersebut akan menajdi satu cluster.
c. Setelah didapatkan 2 nilai minimum, maka digunakan salah satu metode Hierarchical Clustering seperti, Single Linkage, Complete Linkage, Average Linkage, DLL. Pada Contoh ini menggunakan metode Single Linkage. Single Linkage adalah metode mencari jarak node terkecil.

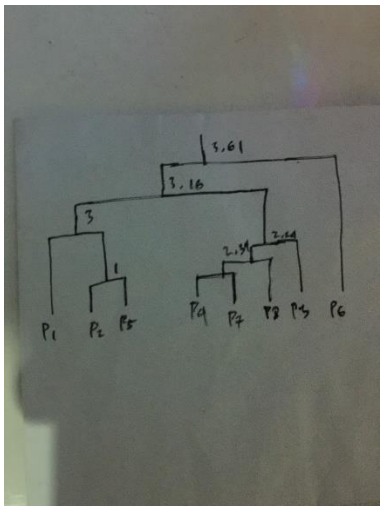
	P1	P2,P5	P3	P4,P7	P6	P8
P1	0	3	5	4,47	3,61	8,06
P2,P5	3	0	3,16	4,12	5,83	7,07
P3	5	3,16	0	2,24	6	4
P4,P7	4,47	4,12	2,24	0	4,12	2,24
P6	3,61	5,83	6	4,12	0	7,21
P8	8,06	7,07	4	2,24	7,21	0

	P1	P2,P5	P3	P4,P7,P8	P6
P1	0	3	5	4,47	3,61
P2,P5	3	0	3,16	4,12	5,83
P3	5	3,16	0	2,24	6
P4,P7,P8	4,47	4,12	2,24	0	4,12
P6	3,61	5,83	6	4,12	0

d. Lakukan perulangan dari step a-c sampai hanya tersisa 1 cluster akhir
   - Dan hasil Cluster akhir dalam bentuk table sebagai berikut

Cluster	Node
A	P2, P5
B	P4, P7
C	B, P8
D	P3. C
E	P1, A
F	(P1,P2,P5,P3,P4,P7,P8), P6

   - Dari Tabel tersebut maka dibentuk lah Graphical Hierarchical Clustering



3. Self Organizing Map
   Analisis

   i. Terdapat 600 Objek yang setiap objeknya terdiri dari 2 atribut
   ii. Cara Penyelesaiannya :

      1) Import file “Tugas_2_ML_Genap_2018_2019_Dataset_Tanpa_Label.csv”
      2) Initialisasi learning rate, sigma, tn (Perubahan learning rate), dan tSigma(perubahan sigma).
      3) Initialisasi Neuron yang berisi 2 atribut dengan nilai random antar 0-1.
      4) Buat fungsi untuk mencari BMU (Best Matching Unit) dengan menghitung jarak/Density antara setiap neuron terhadap data yang sedang di akses.

```
#function buat cari BMU, cari nilai BMU yang paling kecil
def searchBMU(x1,x2):
    density = [] #Cari Jarak Weight dengan Data yang lagi di akses
    for itemN in Nrn :
        density.append(math.sqrt((x1-itemN[0])**2+(x2-itemN[1])**2))
    min = density[0]
    j = 0
    nIdx = j
    for i in density:
        if min > i:
            min = i
            nIdx = j
        j = j + 1
    return nIdx
```

5) Ketika telah menemukan BMU, Buatlah function untuk mencari Topological Neighborhood dari BMU, dengan kode:

```
def THood(we, ob):
    S = []
    T = []
    i = 0
    #S & T must exist like Neightborhood function
    for itemN in Nrn:
        S.append(math.sqrt((itemN[0]-Nrn[we][0])**2+(itemN[1]-Nrn[we][1])**2))
        T.append(math.exp((-(S[i]**2))/2*(ob**2)))
        i = i + 1
    return S, T
```

6) Selanjutnya buatlah function untuk menyesuaikan weight dari neuron yang berdekatan dengan BMU, serta carilah perubahan weight dari setiap neuronnya, dengan kode :

```
def adjust_weight(x1,x2,t,n):
    Y = []
    deltaW = [] #see the different with deltaW
    i = 0
    for itemN in Nrn:
        Y.append([(x1-itemN[0]), (x2-itemN[1])])
        deltaW.append([n*Y[i][0]*t[i], n*Y[i][1]*t[i]])
        i = i + 1
    return deltaW
```