Algoritma K-*Means*

Kuliah : 12 Oktober 2023

Pengklasteran (*Clustering*)

- pengelompokkan sejumlah data atau objek ke dalam klaster (group) sehingga dalam setiap klaster akan berisi data yang semirip mungkin
- Algoritmanya termasuk dalam kategori **unsupervised learning**
- Data pada teknik pengklasteran tidak diketahui keluarannya (outputnya atau labelnya) digunakan fungsi kriteria: jumlah dari kesalahan kuadrat (**sum of squared-error, SSE**) yang dapat mengukur kualitas klastering yang dibuat

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} d(p, m_i)^2$$

p Є Ci = tiap data poin pada cluster i, mi = centroid dari cluster i, *d* = jarak/ distances/ variance terdekat pada masing-masing cluster i. mengoptimalkan nilai fungsi kriteria tersebut

- Nilai SSE tergantung pada jumlah klaster dan bagaimana data dikelompokkan ke dalam klaster-klaster. Semakin kecil nilai SSE semakin bagus hasil klastering yang dibuat

Metode *k-Means*

- Termasuk **partitioning clustering**
- objek-objek dikelompokkan ke dalam *k* kelompok atau klaster
- Untuk melakukan klastering ini, nilai *k* harus ditentukan terlebih dahulu
- Kluster-kluster tersebut mempunyai suatu nilai tengah / nilai pusat yang disebut dengan centroid
- menggunakan ukuran ketidakmiripan untuk mengelompokkan objek.
- Ketidakmiripan diterjemahkan dalam **konsep jarak (distance (*d*))**
- Jika jarak dua objek atau data titik cukup dekat, maka dua objek itu mirip. Semakin dekat berarti semakin tinggi kemiripannya
- Tujuan dari *k-Means* : meminimalisir total dari jarak elemen-elemen antar kluster (jarak antara suatu elemen dalam sebuah kluster dengan nilai centroid kluster tersebut)

Algoritma *k-Means*

1. Pilih jumlah klaster *k* yang diinginkan
2. Inisialisasi *k* pusat klaster (centroid) secara random/ acak
3. Tempatkan setiap data atau objek ke klaster terdekat. Kedekatan dua objek ditentukan berdasar jarak. Jarak yang dipakai pada algoritma *k-Means* adalah *Euclidean distance (d)*.

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

x = *x*1, *x*2, . . . , *xn*, dan **y** = *y*1, *y*2, . . . , *yn* merupakan banyaknya n atribut(kolom) antara 2 record.

4. Hitung kembali pusat klaster dengan keanggotaan klaster yang sekarang. Pusat klaster adalah rata-rata (mean) dari semua data atau objek dalam klaster tertentu.

Algoritma *k-Means* (Lanjutan)

Misal: untuk masing-masing klaster terdapat n poin-poin data (a1,b1,c1), (a2,b2,c2), (a3,b3,c3),. . . , (an,bn,cn), dimana a,b,c merupakan jumlah atribut (dimensi dari data), centroid dari poin-poin data tersebut adalah nilai mean/ titik tengahnya yaitu

$$m_k = \left(\sum a_i/n, \sum b_i/n, \sum c_i/n\right)$$

Sebagai contoh, poin-poin data (1,1,1), (1,2,1), (1,3,1), dan (2,1,1) memiliki centroid yaitu

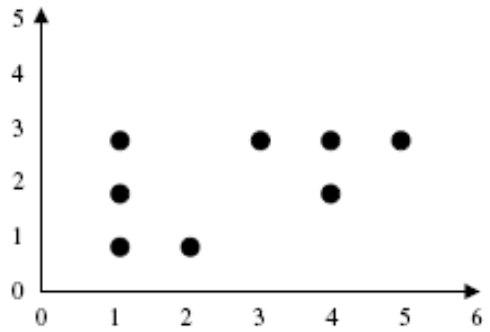
$$m_k = \left(\frac{1+1+1+2}{4}, \frac{1+2+3+1}{4}, \frac{1+1+1+1}{4}\right) = (1.25, 1.75, 1.00)$$

- Tugaskan lagi setiap objek dengan memakai pusat klaster yang baru. Jika pusat klaster sudah tidak berubah lagi, maka proses pengklasteran selesai. Atau, kembali lagi ke langkah nomor **3** sampai pusat klaster tidak berubah lagi/ stabil atau tidak ada penurunan yang signifikan dari nilai SSE (S*um of Squared Errors)*

Contoh Algoritma *k-Means*

Tabel 1 Data point

Instances	X	Y
A	1	3
B	3	3
C	4	3
D	5	3
E	1	2
F	4	2
G	1	1
H	2	1



Gambar 1 tampilan data awal

1. Tentukan jumlah klaster *k*=2
2. Tentukan centroid awal secara acak misal dari data disamping m1 =(1,1), m2=(2,1)
3. Tempatkan tiap objek ke klaster terdekat berdasarkan nilai centroid yang paling dekat selisihnya(jaraknya). Pada tabel 2.Didapatkan hasil: anggota cluster1 = {A,E,G}, cluster2={B,C,D,F,H}. Nilai SSE yaitu :

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} d(p, m_i)^2$$

=

$$2^2 + 2{,}24^2 + 2{,}83^2 + 3{,}61^2 + 1^2 + 2{,}24^2 + 0^2 + 0^2 = 36$$

Contoh Algoritma *k-Means(Lanjutan)*

Tabel 2

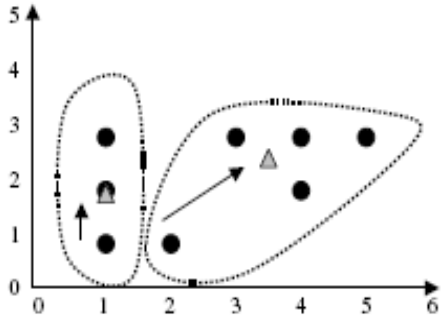
Point	Distance from m_1	Distance from m_2	Cluster Membership
a	2.00	2.24	C_1
b	2.83	2.24	C_2
c	3.61	2.83	C_2
d	4.47	3.61	C_2
e	1.00	1.41	C_1
f	3.16	2.24	C_2
g	0.00	1.00	C_1
h	1.00	0.00	C_2

4. Menghitung nilai centroid yang baru :

$$m_1 = \left[(1+1+1)/3, (3+2+1)/3\right] = (1,2)$$

$$m_2 = \left[(3+4+5+4+2)/5, (3+3+3+2+1)/5\right] = (3,6;2,4)$$

5. Tugaskan lagi setiap objek dengan memakai pusat klaster yang baru. Pada tabel 3. Nilai SSE yang baru :



Gambar Clusters dan centroid *setelah tahap pertama*.

$$\text{SSE} = \sum_{i=1}^{k} \sum_{p \in C_i} d(p,m_i)^2 = 1^2 + 0.85^2 + 0.72^2 + 1.52^2 + 0^2 + 0.57^2 + 1^2$$

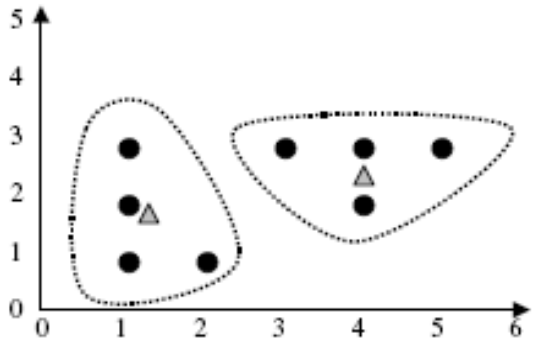
$$+1.41^2 = 7.88$$

Contoh Algoritma *k-Means(Lanjutan)*

Tabel 3

Point	Distance from m_1	Distance from m_2	Cluster Membership
a	1.00	2.67	C_1
b	2.24	0.85	C_2
c	3.16	0.72	C_2
d	4.12	1.52	C_2
e	0.00	2.63	C_1
f	3.00	0.57	C_2
g	1.00	2.95	C_1
h	1.41	2.13	C_2

- Terdapat perubahan anggota cluster yaitu cluster1={A,E,G,H}, cluster2={B,C,D,F}, maka cari lagi nilai centroid yang baru yaitu : m1=(1,25;1,75) dan m2=(4;2,75)
- Tugaskan lagi setiap objek dengan memakai pusat klaster yang baru. Pada tabel 4. Nilai SSE yang baru :



$$\text{SSE} = \sum_{i=1}^{k} \sum_{p \in C_i} d(p,m_i)^2 = 1.27^2 + 1.03^2 + 0.25^2 + 1.03^2 + 0.35^2 + 0.75^2$$
$$+0.79^2 + 1.06^2 = 6.25$$

Gambar Clusters dan centroid *setelah tahap kedua*.

Contoh Algoritma *k-Means(Lanjutan)*

Tabel 4

Point	Distance from m_1	Distance from m_2	Cluster Membership
a	1.27	3.01	C_1
b	2.15	1.03	C_2
c	3.02	0.25	C_2
d	3.95	1.03	C_2
e	0.35	3.09	C_1
f	2.76	0.75	C_2
g	0.79	3.47	C_1
h	1.06	2.66	C_2

- Dapat dilihat pada tabel 4.Tidak ada perubahan anggota lagi pada masing-masing cluster
- Hasil akhir yaitu :

  cluster1={A,E,G,H}, dan cluster2={B,C,D,F} dengan nilai SSE = 6,25 dan jumlah iterasi 3

Algoritma K-Modes

Algoritma K-Modes

- K-Modes dikembangkan oleh Leonard Kaufman dan Peter J. Rousseeuw sekitar tahun 1987, diperkenalkan pertama kali oleh Huang pada tahun 1997.
- Algoritma K-Meodes sama seperti algoritma K-Means yaitu salah satu teknik dalam data mining untuk mengelompokkan (clustering) data kedalam beberapa kelompok berdasarkan jarak, kriteria, kondisi, atau karakteristik. (Buulolo, 2020)
- Perbedaan antara keduanya adalah algoritma K-Means yang terbatas pada penggunaan data **bertipe numerik** karena pengelompokannya dengan menghitung rata-rata (mean) dari suatu data dengan data yang lain.

- Sedangkan adanya algoritma K-Modes bertujuan untuk **memperbaiki keterbatasan** pada algoritma K-Means dengan cara dimodifikasi dari perhitungan rata-rata (mean) menjadi nilai yang sering muncul (modus).
- Algotirma K-Modes merupakan pengembangan dari algoritma K-Means untuk mengelompokkan data kategorik (nominal dan ordinal) dan memiliki kelebihan untuk mengatasi kelemahan algoritma K-Means yang sensitif terhadap data noise atau data outlier.
- Jollyta, Deni, dkk dalam Furqon, dkk., 2015 mengemukakan kelebihan lainnya yaitu hasil proses clustering tidak bergantung pada urutan masuk dataset.

Tahapan Algoritma K-Modes

- Algoritma K-Modes memiliki beberapa tahapan sebagai berikut :
  1. Menyiapkan sampel dari dataset yang dipilih.
  2. Menentukan jumlah klaster yang akan dibentuk pada r data. Banyaknya klaster tidak boleh lebih dari banyaknya data (k<r) nilai k = 2, 3, 4, 5, …, 8 atau yang lainnya.
  3. Menentukan centroid (pusat klaster) **secara acak** dari masing-masing klaster.
  4. Menghitung jarak tiap data (objek) terhadap semua centroid.

5. Mengelompokkan objek berdasarkan jarak terdekat ke centroid.
6. Memperbarui centroid masing-masing klaster berdasarkan **modus** dari setiap variable anggota klaster yang terbentuk.
7. Menghitung ulang jarak setiap data (objek) terhadap centroid baru menggunakan ukuran ketidaksamaan sederhana seperti langkah ke-3.
8. Apabila masih ada objek yang berpindah klaster, maka ulangi langkah ke-6 sampai ke-8 hingga tidak ada objek yang berpindah klaster.

Algoritma K-Medoids

Algoritma k-medoids

- K-Medoids atau Partitioning Around Method (PAM) adalah metode cluster non hirarki yang merupakan varian dari metode K-Means.
- K-Medoids hadir untuk mengatasi kelemahan K-Means yang **sensitif terhadap outlier** karena suatu objek dengan suatu nilai yang besar mungkin secara substansial menyimpang dari distribusi data (Jiawei & Kamber, 2006).

- Hal ini didasarkan pada penggunaan medoids bukan dari pengamatan mean yang dimiliki oleh setiap cluster, dengan tujuan mengurangi sensitivitas dari partisi sehubungan dengan nilai ekstrim yang ada dalam dataset (Vercellis, 2009)
- K-Medoids menggunakan metode pengelompokan partisi untuk mengelompokan sekumpulan n objek menjadi sejumlah k cluster.
- Algoritma ini menggunakan objek pada kumpulan objek yang mewakili sebuah cluster.
- Objek yang mewakili sebuah cluster disebut dengan medoids.

- Medoid merupakan objek yang letaknya terpusat di dalam suatu cluster sehingga robust terhadap outlier.
- Cluster dibangun dengan menghitung kedekatan yang dimiliki antara medoids dengan objek non medoids (Setyawati, 2017)
- Sama halnya dengan K-Means, metode K-Medoid lebih menguntungkan diterapkan pada data yang sangat besar.

Algoritma k-medoids

1. Tentukan k (jumlah cluster) yang diinginkan
2. Pilih **secara acak medoid** awal sebanyak k dari n data
3. Hitung jarak masing-masing obyek ke medoid sementara, kemudian tandai jarak terdekat obyek ke medoid dan hitung totalnya.
4. Lakukan iterasi medoid
5. Hitung total simpangan (S),

   Jika a adalah jumlah jarak terdekat antara obyek ke medoid awal, dan b adalah jumlah jarak terdekat antara obyek ke medoid baru, maka total simpangan adalah S = b — a

   Jika S < 0, maka tukar obyek dengan data untuk membentuk sekumpulan k baru sebagai medoid
6. Ulangi langkah 3 sampai 5 dan hentikan jika sudah tidak terjadi perubahan anggota medoid

Fuzzy Clustering

Ukuran Fuzzy

- Teori himpunan fuzzy akan memberikan jawaban terhadap suatu masalah yang mengandung ketidakpastian
- Yang menjadi pertanyaan :
  - Seberapa besar kekaburan suatu himpunan fuzzy?
  - Seberapa sama antara 2 himpunan fuzzy?
- Pertanyaan pertama akan melahirkan konsep ukuran fuzzy (Fuzzy Measurement) dan pertanyaan kedua akan melahirkan konsep ukuran persamaan (Similiarity Measurement)

Ukuran Fuzzy

- Menunjukkan derajat kekaburan dari himpunan fuzzy
- Ukuran kekaburan ditulis f:P(X) → R
- Dalam mengukur nilai kekaburan fungsi f harus mengikuti hal-hal berikut :
  1. f(A)=0, jika dan hanya jika A crisp
  2. Jika A < B maka f(A) < f(B), jika A < B berarti B lebih kabur atau A lebih tajam dari B
  3. F(x) akan mencapai max ⇔ A benar-benar kabur secara maksimum (nilai fuzzy maks=0.5).

Indeks Kekaburan

- Indeks kekaburan adalah jarak antara suatu himpunan fuzzy A dengan Himpunan crisp terdekat.
- Himpunan crisp C terdekat dari himpunan fuzzy A dinotasikan
    - Jika $\mu_A[x] \leq 0.5$ maka $\mu_C[x] = 0$
    - Jika $\mu_A[x] \geq 0.5$ maka $\mu_C[x] = 1$
- Ada 3 kelas yang paling sering digunakan dalam mencari indeks kekaburan, yaitu:

- Hamming distance.

$$f(A) = \sum |\mu_A[x] - \mu_C[x]| \text{ atau}$$

$$f(A) = \sum \min[\mu_A[x], 1 - \mu_A[x]]$$

- Euclidean distance.

$$f(A) = \{\sum [\mu_A[x] - \mu_C[x]]^2\}^{1/2}$$

- Minkowski distance.

$$f(A) = \{\sum [\mu?_A[x] - \mu_C[x]]^w\}^{1/w}$$

dengan $w \in [1,\sim]$.

Fuzzy Entrophy

- Fuzzy entropy didefinisikan dengan fungsi:

$$f(A) = -\sum\{\mu A[x]\log \mu A[x]+[1-\mu A[x]]\log[1-\mu A[x]]\}$$

Fuzzy C-Means (FCM)

- Fuzzy clustering adalah salah satu teknik untuk menentukan cluster optimal dalam suatu ruang vektor yang didasarkan pada bentuk normal Euclidian untuk jarak antar vektor.
- Fuzzy C-Means (FCM) adalah suatu teknik pengclusteran data yang mana keberadaan tiap-tiap titik data dalam suatu cluster ditentukan oleh derajat keanggotaan.
- Output dari FCM bukan merupakan *fuzzy inference system*, namun merupakan deretan pusat cluster dan beberapa derajat keanggotaan untuk tiap-tiap titik data. Informasi ini dapat digunakan untuk membangun suatu *fuzzy inference system*.
- Teknik ini pertama kali diperkenalkan oleh Jim Bezdek pada tahun 1981.

- Konsep dasar FCM, pertama kali adalah menentukan pusat cluster, yang akan menandai lokasi rata-rata untuk tiap-tiap cluster.
- Pada kondisi awal, pusat cluster ini masih belum akurat. Tiap-tiap titik data memiliki derajat keanggotaan untuk tiap-tiap cluster.
- Dengan cara memperbaiki pusat cluster dan derajat keanggotaan tiap-tiap titik data secara berulang, maka akan dapat dilihat bahwa pusat cluster akan bergerak menuju lokasi yang tepat.
- Perulangan ini didasarkan pada minimisasi fungsi obyektif yang menggambarkan jarak dari titik data yang diberikan ke pusat cluster yang terbobot oleh derajat keanggotaan titik data tersebut.

Algoritma FCM

1. Input data yang akan dicluster X, berupa matriks berukuran n x m (n = jumlah sampel data, m = atribut setiap data). X_{ij} = data sampel ke-i (i=1,2,...,n), atribut ke-j (j=1,2,...,m).
2. Tentukan:
    - Jumlah cluster = c;
    - Pangkat = w;
    - Maksimum iterasi = MaxIter;
    - Error terkecil yang diharapkan = ξ.
    - Fungsi obyektif awal = $P_0 = 0$;
    - Iterasi awal = t = 1;

3. Bangkitkan bilangan random μ_{ik}, i=1,2,...,n; k=1,2,...,c; sebagai elemen-elemen matriks partisi awal U.

Hitung jumlah setiap kolom:

$$Qj = \sum_{k=1}^{c} \mu_{ik}$$

dengan j=1,2,...,n.

Hitung:

$$\mu_{ik} = \frac{\mu_{ik}}{Q_i}$$

4. Hitung pusat cluster ke-k: V_{kj}, dengan k=1,2,...,c; dan j=1,2,...,m.

$$V_{kj} = \frac{\sum_{i=1}^{n} \left((\mu_{ik})^{w} * X_{ij} \right)}{\sum_{i=1}^{n} (\mu_{ik})^{w}}$$

5. Hitung fungsi obyektif pada iterasi ke-t, P_t:

$$P_t = \sum_{i=1}^{n} \sum_{k=1}^{c} \left(\left[\sum_{j=1}^{m} \left(X_{ij} - V_{kj} \right)^2 \right] (\mu_{ik})^{w} \right)$$

6. Hitung perubahan matriks partisi:

$$\mu_{ik} = \frac{\left[\sum_{j=1}^{m}\left(X_{ij} - V_{kj}\right)^2\right]^{\frac{-1}{w-1}}}{\sum_{k=1}^{c}\left[\sum_{j=1}^{m}\left(X_{ij} - V_{kj}\right)^2\right]^{\frac{-1}{w-1}}}$$

dengan: i = 1,2,...,n; dan k = 1,2,...,c.

7. Cek kondisi berhenti:

- Jika: ($|P_t - P_{t-1}| < \xi$) atau (t > MaxIter) maka berhenti;
- Jika tidak: t = t+1, ulangi langkah ke-4.

Fuzzy Subtractive Clustering

Fuzzy Subtractive Clustering

- Fuzzy C-Means (FCM) adalah algoritma pengclusteran yang terawasi, sebab pada FCM kita perlu tahu terlebih dahulu jumlah cluster yang akan dibentuk.
- Apabila jumlah cluster yang akan dibentuk belum diketahui sebelumnya, maka kita harus menggunakan algoritma yang tidak terawasi.
- *Subtractive clustering* didasarkan atas ukuran densitas (potensi) titik-titik data dalam suatu ruang (variabel).

- Konsep dasar dari *subtractive clustering* adalah menentukan daerah-daerah dalam suatu variabel yang memiliki densitas tertinggi terhadap titik-titik di sekitarnya.
- Titik dengan jumlah tetangga terbanyak akan dipilih sebagai pusat cluster. Titik yang sudah terpilih sebagai pusat cluster ini kemudian akan dikurangi densitasnya.
- Kemudian algoritma akan memilih titik lain yang memiliki tetangga terbanyak untuk dijadikan pusat cluster yang lain.
- Hal ini akan dilakukan berulang-ulang hingga semua titik diuji.

- Apabila terdapat N buah data: X_1, X_2, ..., X_N dan dengan menganggap bahwa data-data tersebut sudah dalam keadaan normal, maka densitas titik X_k dapat dihitung sebagai:

$$D_k = \sum_{j=1}^{N} \exp\left(- \frac{\left\| X_k - X_j \right\|}{(r/2)^2} \right)$$

  - dengan $||X_k - Xj||$ adalah jarak antara X_k dengan X_j, dan r adalah kontanta positif yang kemudian akan dikenal dengan nama *influence range* atau jari-jari (r).
  - Jari-jari, berupa vektor yang akan menentukan seberapa besar pengaruh pusat *cluster* pada tiap-tiap variabel.
- Dengan demikian, suatu titik data akan memiliki densitas yang besar jika dia memiliki banyak tetangga dekat.

- Setelah menghitung densitas tiap-tiap titik, maka titik dengan densitas tertinggi akan dipilih sebagai pusat cluster.
- Misalkan X_{c1} adalah titik yang terpilih sebagai pusat cluster, sedangkan D_{c1} adalah ukuran densitasnya. Selanjutnya densitas dari titik-titik di sekitarnya akan dikurangi menjadi:

$$D_k^{'} = D_k - D_{C1} * \exp\left(- \frac{\left\| X_k - X_{C1} \right\|}{(r_b / 2)^2} \right)$$

  - dengan r_b adalah konstanta positif.
- Titik-titik yang berada dekat dengan pusat cluster u_{c1} akan mengalami pengurangan densitas besar-besaran. Titik-titik tersebut akan sangat sulit untuk menjadi pusat cluster berikutnya.
- Nilai r_b menunjukkan suatu lingkungan yang mengakibatkan titik-titik berkurang ukuran densitasnya. Biasanya r_b bernilai lebih besar dibanding dengan r, r_b = q*r (biasanya squash_factor (q) = 1,5).

- Setelah densitas tiap-tiap titik diperbaiki, maka selanjutnya akan dicari pusat cluster yang kedua yaitu X_{c2}. Sesudah X_{c2} didapat, ukuran densitas setiap titik data akan diperbaiki kembali, demikian seterusnya.
- Pada implementasinya, bisa digunakan 2 pecahan sebagai faktor pembanding, yaitu Accept ratio dan Reject ratio.
- Baik accept ratio maupun reject ratio keduanya merupakan suatu bilangan pecahan yang bernilai 0 sampai 1.
- Accept ratio merupakan batas bawah dimana suatu titik data yang menjadi kandidat (calon) pusat cluster diperbolehkan untuk menjadi pusat cluster.
- Sedangkan reject ratio merupakan batas atas dimana suatu titik data yang menjadi kandidat (calon) pusat cluster tidak diperbolehkan untuk menjadi pusat cluster.

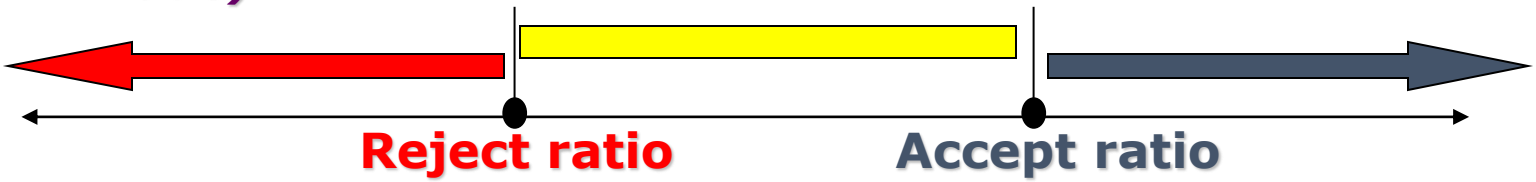
- Pada suatu iterasi, apabila telah ditemukan suatu titik data dengan potensi tertinggi (misal: X_k dengan potensi D_k), kemudian akan dilanjutkan dengan mencari Rasio potensi titik data tersebut dengan potensi tertinggi suatu titik data pada awal iterasi (misal: X_h dengan potensi D_h).
- Hasil bagi antara D_k dengan D_h ini kemudian disebut dengan Rasio (Rasio = D_k/D_h).

- Ada 3 kondisi yang bisa terjadi dalam suatu iterasi:
  - Apabila Rasio > Accept ratio, maka titik data tersebut diterima sebagai pusat cluster baru.
  - Apabila Reject Ratio < Rasio $\leq$ Accept ratio maka titik data tersebut baru akan diterima sebagai pusat cluster baru hanya jika titik data tersebut terletak pada jarak yang cukup jauh dengan pusat cluster yang lainnya (hasil penjumlahan antara Rasio dan jarak terdekat titik data tersebut dengan suatu pusat cluster lainnya yang telah ada $\geq$ 1). Apabila hasil penjumlahan antara Rasio dan jarak terdekat titik data tersebut dengan pusat cluster lainnya yang telah ada < 1, maka selain titik data tersebut tidak akan diterima sebagai pusat cluster, dia sudah tidak akan dipertimbangkan lagi untuk menjadi pusat cluster baru (potensinya diset sama dengan nol).
  - Apabila Rasio $\leq$ Reject ratio, maka sudah tidak ada lagi titik data yang akan dipertimbangkan untuk menjadi kandidat pusat cluster, iterasi dihentikan.

Data tidak diterima sebagai pusat cluster
(Rasio ≤ Reject ratio)

Data diterima sebagai pusat cluster tapi dengan syarat.
(Reject ratio < Rasio ≤ Accept ratio)

Data diterima sebagai pusat cluster
(Rasio > Accept ratio)



Perbedaan FCM & Subtractive

- Pada metode FCM pusat cluster bisa jadi bukan merupakan salah satu dari data yang dicluster.
- Pada metode subtractive clustering, suatu pusat cluster pasti merupakan salah satu data yang ikut dicluster, yaitu data dimana derajat keanggotaannya pada cluster tersebut sama dengan 1.
- Penjumlahan semua derajat keanggotaan pada FCM selalu bernilai sama dengan 1.
- Pada metode subtractive clustering, penjumlahan semua derajat keanggotaannya belum tentu (bahkan jarang) bernilai sama dengan 1.

Algoritma Subtractive Clustering

1. Input data yang akan dicluster: X_{ij}, dengan i=1,2,...,n; dan j=1,2,...,m.
2. Tetapkan nilai:
   - r_j (jari-jari setiap atribut data); j=1,2,...m
   - q (squash factor);
   - Accept_ratio;
   - Reject_ratio;
   - XMin (minimum data diperbolehkan);
   - XMax (maksimum data diperbolehkan);

3. Normalisasi

$$X_{ij} = \frac{X_{ij} - XMin_j}{XMax_j - XMin_j}, \quad i = 1,2,...,n;\ j = 1,2,...,m$$

4. Tentukan potensi awal tiap-tiap titik data

- i=1
- Kerjakan hingga i=n,
  - $T_j = X_{ij}$; j=1,2,…,m
  - Hitung:

$$Dist_{kj} = \left(\frac{T_j - X_{kj}}{r} \right) \quad j = 1,2,...,m;\ k = 1,2,...,n$$

- Hitung Potensi awal:
  - Jika m = 1, maka

$$D_i = \sum_{k=1}^{n} e^{-4\left(Dist_{k1}^2\right)}$$

  - Jika m > 1, maka

$$D_i = \sum_{k=1}^{n} e^{-4\left(\sum_{j=1}^{m} Dist_{kj}^2\right)}$$

- i=i+1

5. Cari titik dengan potensi tertinggi

- M = max[D_i | i=1,2,…,n];
- h = i, sedemikian hingga D_i = M;

6. Tentukan pusat *cluster* dan kurangi potensinya terhadap titik-titik di sekitarnya.

- Center = []
- $V_j = X_{hj}$; j=1,2,…,m;
- C = 0 (jumlah *cluster*);
- Kondisi=1;
- Z=M;
- Kerjakan jika (Kondisi≠0) & (Z≠0):
  - Kondisi=0 (sudah tidak ada calon pusat baru lagi);
  - Rasio = Z/M.
  - Jika Rasio > accept_ratio, maka Kondisi=1; (ada calon pusat baru)

- Jika tidak,
  - Jika Rasio > reject_ratio, (calon baru akan diterima sebagai pusat jika keberadaannya akan memberikan keseimbangan terhadap data-data yang letaknya cukup jauh dengan pusat cluster yang telah ada), maka kerjakan
    - Md = -1;
    - Kerjakan untuk i=1 sampai i=C:
      - Hitung:

$$G_{ij} = \frac{V_j - Center_{ij}}{r}$$

j=1,2,…,m

- Hitung:

$$Sd_i = \sum_{j=1}^{m} (G_{ij})^2$$

- Jika (Md < 0) atau (Sd < Md), maka Md = Sd;

- Smd = √Md;
- Jika (Rasio + Smd) ≥ 1, maka Kondisi = 1; (Data diterima sebagai pusat *cluster*)
- Jika (Rasio + Smd) < 1, maka Kondisi = 2; (Data tidak akan dipertimbangkan kembali sebagai pusat *cluster*).

- Jika Kondisi=1 (Calon pusat baru diterima sebagai pusat baru), kerjakan:
  - C = C+1;
  - CC = V;
  - Kurangi potensi dari titik-titik di dekat pusat *cluster*:
    - Hitung:

$$S_{ij} = \frac{V_j - X_{ij}}{r_j * q}; \; j = 1,2,...,m; \; i = 1,2,...,n.$$

    - Hitung:

$$Dc_i = M * e^{-4\left[\sum_{j=1}^{m}(S_{ij})^2\right]}; \; i = 1,2,...,n$$

    - D = D – D_c;
    - Jika $D_i \leq 0$, maka D_i = 0; i=1,2,...,n.
    - Z = max[D_i|i=1,2,…,n];
    - Pilih h = i, sedemikian hingga D_i = Z;

- Jika Kondisi=2 (Calon pusat baru tidak diterima sebagai pusat baru), maka
  - $D_h = 0$;
  - $Z = \max[D_i | i=1,2,\ldots,n]$;
  - Pilih h = i, sedemikian hingga $D_i = Z$;

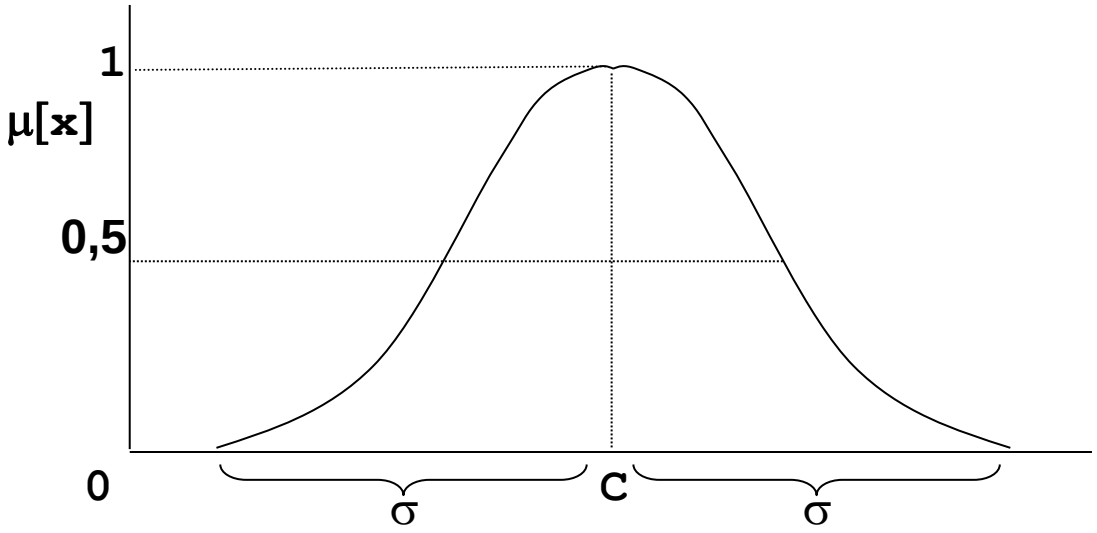
7. Kembalikan pusat *cluster* dari bentuk ternormalisasi ke bentuk semula.

- $Center_{ij} = Center_{ij} * (XMax_j - XMin_j) + XMin_j$;

8. Hitung nilai sigma *cluster*.

- $\sigma_j = r_j * (XMax_j - XMin_j) / \sqrt{8}$.

- Hasil dari algoritma *Subtractive Clustering* ini berupa matriks pusat *cluster* (C) dan sigma (σ) akan digunakan untuk menentukan nilai parameter fungsi keanggotaan Gauss:



- Dengan kurva Gauss tersebut, maka derajat keanggotaan suatu titik data X_i pada cluster ke-k, adalah:

$$\mu_{ki} = e^{-\sum_{j=1}^{m} \frac{\left(X_{ij} - C_{kj}\right)^2}{2\sigma_j^2}}$$

Algoritma DBSCAN

- Konsep utama dari algoritma DBSCAN adalah mengelompokkan data dengan tetangganya dalam satu/beberapa cluster, dimana dalam satu cluster harus terdapat jumlah anggota yang memenuhi nilai dari minimum points (minPts). Selain itu, jarak antara titik utama (core point) dengan anggota dalam satu cluster tidak boleh lebih dari nilai epsilonnya. Titik utama (core point) akan dipilih secara acak pada iterasi pertama.
- Dikarenakan jangkauan nilai epsilon yang konstan pada setiap clusternya, algoritma DBSCAN ini memiliki kekurangan, yaitu kesulitan untuk menjangkau titik dengan kepadatan yang berbeda-

DBSCAN

- Algoritma **Density-based Spatial Clustering of Application with Noise** (DBSCAN) merupakan metode clustering yang berbasis kepadatan (density-based) dari posisi amatan data dengan prinsip mengelompokkan data yang relatif berdekatan.
- DBSCAN sering diterapkan pada data yang banyak mengandung noise, hal ini dikarenakan DBSCAN tidak akan memasukkan data yang dianggap noise kedalam cluster manapun.

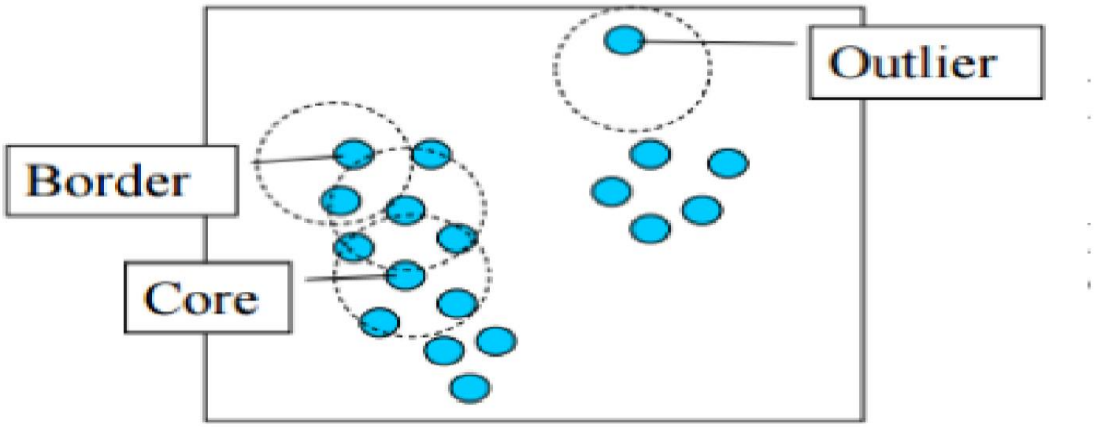
- Density Clustering adalah metode clustering yang memanfaatkan tingkat kerapatan object untuk dilakukan pengelompokan.
- Salah satu algoritma density clustering yang paling terkenal adalah DBSCAN (Density-based Spatial Clustering of Application with Noise

- DBSCAN memerlukan dua parameter input sebelum melakukan proses clustering yaitu **epsilon** (eps) dan **minimum points** (minPts).
- Epsilon merupakan jarak maksimal antara dua data dalam satu cluster yang diizinkan, dan minimum points adalah banyaknya data minimal dalam jarak epsilon agar terbentuk suatu cluster.
- Metode jarak yang digunakan dalam DBSCAN adalah jarak Euclidian.

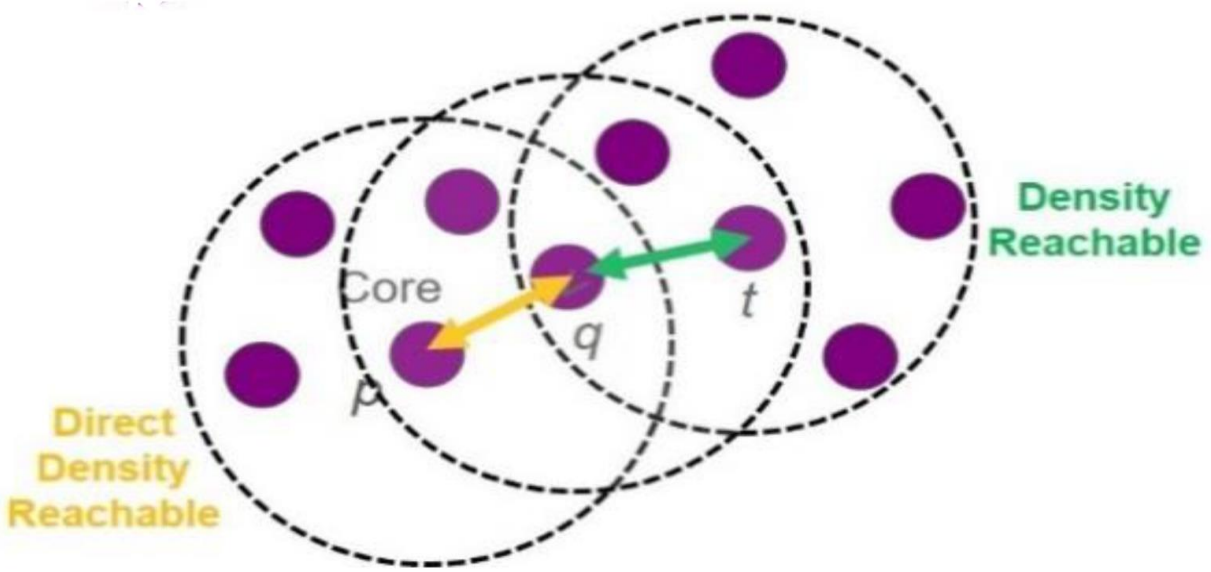
- Konsep utama dari algoritma DBSCAN adalah mengelompokkan data dengan tetangganya dalam satu/beberapa cluster, dimana dalam satu cluster harus terdapat jumlah anggota yang memenuhi nilai dari minimum points (minPts).
- Selain itu, jarak antara titik utama (core point) dengan anggota dalam satu cluster tidak boleh lebih dari nilai epsilonnya.
- Titik utama (core point) akan dipilih secara acak pada iterasi pertama.

- Dikarenakan jangkauan nilai epsilon yang konstan pada setiap clusternya, algoritma DBSCAN ini memiliki kekurangan, yaitu kesulitan untuk menjangkau titik dengan kepadatan yang berbeda-beda (varying density).
- Padahal sangat memungkinkan titik yang tidak dapat terjangkau tersebut termasuk dalam kategori meaningful cluster.

- Selain epsilon dan MinPts ada beberapa terminologi lain dalam metode DBSCAN yaitu :
  - directly density-reachable : Observasi q berhubungan langsung dengan p, jika p adalah core point dan q merupakan tetangga dari p dalam jangkauan epsilon.
  - density-reachable : Observasi q dan x dalam satu cluster namun x bukan tetangga dari q dalam jangkauan epsilon.
  - core point : Core point merupakan observasi yang memiliki jumlah tetangga lebih dari sama dengan dari MinPts pada jangkauan Eps.
  - border point : Border point memiliki tetangga lebih sedikit dari Minpts namun ia merupakan tetangga dari core point.
  - outlier/noise point : Observasi yang bukan border points atau core points.



$\varepsilon = 1$ unit, MinPts =5



Cara Kerja DBSCAN

- Dalam proses pembuatan cluster menggunakan DBSCAN sebuah data akan dikelompokkan dengan tetangganya. Sepasang amatan dikatakan bertetangga apabila jarak antara dua amatan tersebut kurang dari sama dengan nilai epsilon. Secara sederhana cara kerja DBSCAN adalah sebagai berikut :

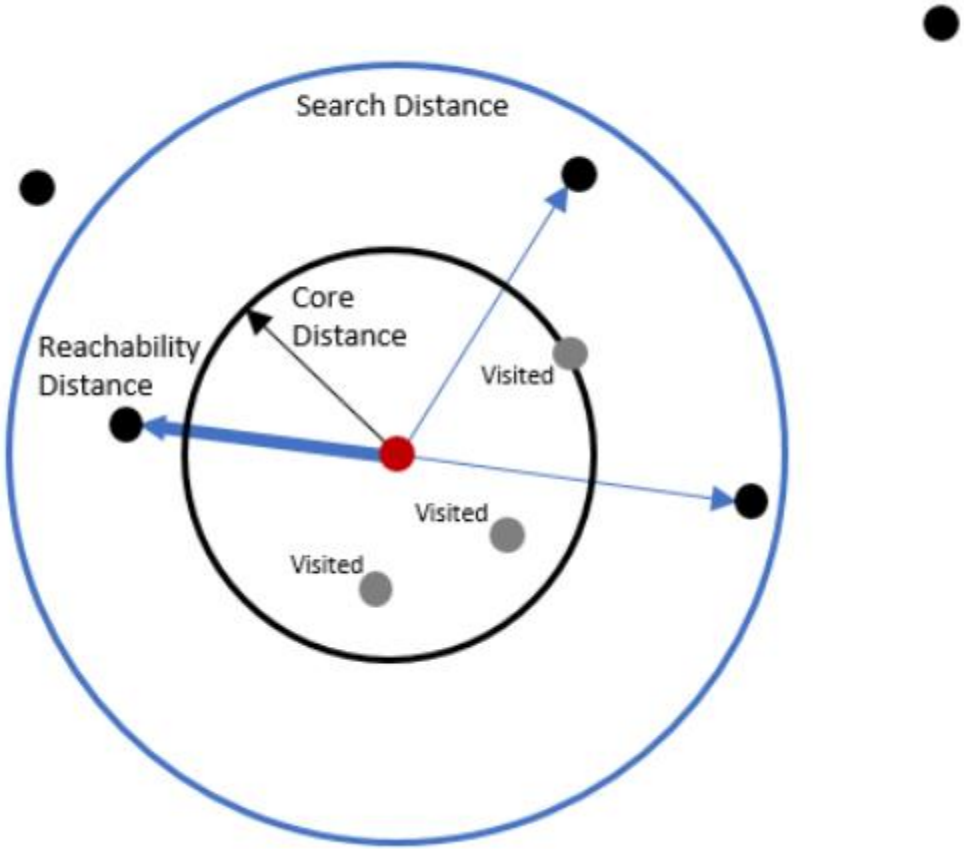
1. Tentukan nilai minPts dan epsilon (eps) yang akan digunakan.
2. Pilih data awal “p” secara acak.
3. Hitung jarak antara data “p” terhadap semua data menggunakan Euclidian distance.
4. Ambil semua amatan yang density-reachable dengan amatan “p”
5. Jika amatan yang memenuhi nilai epsilon lebih dari jumlah minimal amatan dalam satu gerombol maka amatan “p” dikategorikan sebagai core points dan gerombol terbentuk.
6. Jika amatan “p” adalah border points dan tidak ada amatan yang density-reachable dengan amatan “p”, maka lanjutkan pada amatan lainnya.
7. Ulangi langkah 3 sampai 6 hingga semua amatan diproses.

Algoritma OPTICS

- Secara umum, cara kerja algoritma OPTICS sama dengan cara kerja algoritma DBSCAN.
- Parameter yang dimilikipun sama yaitu parameter epsilon (eps) dan parameter minimum points (minPts). Namun, pada algoritma OPTICS ini terdapat dua terminologi baru yang sebelumnya tidak ada pada algoritma DBSCAN, yaitu core distance dan reachability distance.

- Berikut d adalah pengertian dari kedua terminologi tersebut :
  - Core distance : Jarak/radius minimum dari core point p (yang di identifikasikan sebagai core point) dengan anggota neighborhood dalam satu cluster.
  - Reachability distance : Jarak minimum titik q dari core point p di luar dari bagian core objek (memenuhi minPts).

- Bagaimana MinPts Dan Epsilon Mengambil Peran Dalam Pembentukan Cluster?
  - Sama halnya dengan DBSCAN, pada OPTICS kita dapat memanfaatkan parameter minPts dan epsilon sebagai unsur pembentukan cluster.
  - Akan tetapi jika pada DBSCAN epsilon dianggap sebagai jarak konstan untuk membentuk sebuah cluster, pada OPTICS nantinya kita akan mempertimbangkan jarak antara titik-titik yang berdekatan untuk membuat reachability distance
  - Selanjutnya digunakan untuk memisahkan cluster dengan kepadatan yang bervariasi dari noise.


Search Distance
Core
Distance
Reachability
Distance
Visited
Visited
Visited

- Parameter minPts pada OPTICS digunakan untuk menentukan jumlah minimum fitur/titik yang diperlukan untuk mempertimbangkan pengelompokan titik-titik pada sebuah cluster.
  - Misalnya jika kita memiliki sejumlah cluster berbeda dengan ukuran mulai dari 10 titik hingga 50 titik dan kita menetapkan nilai minPts sebanyak 20, maka untuk semua cluster yang memiliki anggota kurang dari 20 titik akan dianggap sebagai noise karena tidak membentuk pengelompokan yang cukup besar untuk dianggap sebagai cluster atau titik-titik tersebut akan digabungkan dengan cluster terdekat untuk memenuhi jumlah minimum titik yang diperlukan.

- Pada OPTICS, epsilon diperlakukan sebagai jarak maksimum yang akan dibandingkan dengan jarak inti (core distance).
- OPTICS menggunakan konsep jarak jangkauan maksimum, yaitu jarak dari titik ke tetangga terdekatnya yang belum pernah dikunjungi oleh pencarian (titik-titik pada boundary 2).
- OPTICS akan mencari semua jarak tetangga dalam jangkauan epsilon yang ditentukan, kemudian membandingkan masing-masing titik tersebut dengan core distance.

- Jika ada jarak yang lebih kecil dari jarak inti(semua titik yang masuk pada boundary 1), maka titik tersebut menetapkan core distance sebagai reachability distance-nya (CD=RD).
- Jika semua jarak lebih besar dari jarak inti(semua titik yang ada di boundary 2), maka jarak terkecil ditetapkan sebagai reachability distancenya. Jarak ini kemudian digunakan untuk membuat reachable plot, yang merupakan plot terurut dari reachability distance yang digunakan untuk mendeteksi cluster.

Tugas Pendalaman Materi

- Buatklan contoh kasus sederhana untuk masing-masing algoritma clustering, dan dilengkapi dengan implementasinya dengan Python.
- Tugas kelompok (anggota maksimal 3 mhs)
- Dikumpul dalam bentuk slide, dan Laporan dalam format docx.
- Dikumpulkan di spada, maksimal tgl 19 oktober 2023