ALGORITMA CLUSTERING

Mukhamad Subkhan, 06874-TE
Jurusan Teknik Elektro FT UGM,
Yogyakarta

7.5. OPTIMASI FUNGSI NOMINAL ALGORITMA CLUSTERING

Pada bagian ini, masing-masing cluster, C_j ,dalam clustering parameter ini oleh vektor parameter θ_j. Tujuannya adalah untuk mengidentifikasi nilai-nilai dari parameter vektor, yang mencirikan struktur clustering X dalam arti yang optimal. Hal ini dilakukan melalui definisi yang sesuai fungsi optimalisasi.

7.5.1. Hard Clustering Algorithms

Dalam algoritma dari kategori ini, diasumsikan bahwa setiap data vektor tergolong eksklusif untuk satu cluster.

Algoritma k-Means, atau Isodata

Algoritma clustering ini paling banyak dikenal, dan dasar pemikirannya yang sangat sederhana. Dalam hal ini, parameter vektor θ_j (disebut juga cluster representative atau representative yang sederhana) sesuai dengan titik di ruang dimensi 1, di mana kumpulan data vektor X aktif. k-means mengasumsikan bahwa jumlah cluster yang mendasari dikenal X, m. Tujuannya adalah untuk memindahkan titik θ_j, j = 1,. . . , m, ke daerah yang tersusun rapat di titik X (cluster).

Algoritma k-means adalah jenis iteratif. Hal ini dimulai dengan beberapa perkiraan awal θ_1 (0),. . . , $\theta\,\theta_m$ (0), untuk vektor-vektor parameter θ_1,. . . , θ_m. Pada t setiap iterasi,

- vektor x_i yang terletak dekat dengan setiap θ_j (t -1) dan kemudian di identifikasi
- nilai yang baru (diperbaharui) θ_j, θ_j (t), dihitung sebagai rata-rata dari vektor data yang lebih dekat dengan θ_j (t -1).

Algoritma berakhir ketika tidak ada perubahan yang terjadi di antara dua iterasi θ_j berturut-turut. Untuk menjalankan algoritma k-means, ketik

[*theta,bel, J*] = *k_means(X, theta_ini)*

Dimana
X adalah N × l matriks yang kolom-kolomnya berisi vektor data,

theta_ini merupakan matriks l × m yang kolom-kolomnya adalah perkiraan awal θ_j (jumlah cluster, m, secara implisit ditentukan oleh ukuran theta_ini),

theta adalah matriks ukuran yang sama seperti theta_ini, yang berisi perkiraan akhir untuk θ_j.

bel adalah vektor N-dimensi yang berisi elemen ke i dengan label cluster untuk vektor data yang ke i, J adalah nilai dari fungsi nominal diberikan dalam Persamaan. (7.1) untuk clustering yang dihasilkan.

Keterangan

- k-means cocok untuk cluster terurai yang tersusun rapat.
- k-means algoritma adalah algoritma iteratif cepat karena (a) dalam prakteknya hanya memerlukan beberapa iterasi untuk menuju ke satu titik dan (b) perhitungan dibutuhkan pada setiap iterasi tidak rumit. Jadi, itu bertindak sebagai kandidat untuk pengolahan set data yang besar.
- Hal ini dapat ditunjukkan bahwa algoritma k-means meminimalkan fungsi nominal.

$$J(\theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} ||x_i - \theta_j||^2 \qquad (7.1)$$

Dimana $\theta = [\theta_1^T, \ldots, \theta_m^T]^T$, $\|\cdot\|$ Berada pada jarak Euclidean, dan u_{ij} = 1 jika x_i terletak paling dekat dengan θ_j; sebaliknya 0. Dengan kata lain, k-means meminimalkan jumlah jarak kuadrat Euclidean dari setiap vektor data dari parameter vektor terdekatnya. Ketika data vektor dari cluster X bentuk m yang tersusun rapat (dengan tidak ada perbedaan signifikan dalam ukuran), diharapkan bahwa J diminimalkan ketika setiap θ_i ditempatkan (sekitar) di tengah setiap cluster, asalkan m diketahui. Ini tidak selalu terjadi ketika (a) vektor data tidak membentuk cluster-cluster yang tersusun rapat, atau (b) ukuran mereka berbeda secara signifikan, atau (C) jumlah cluster, m, belum diperkirakan dengan benar.

k-means tidak menjamin konvergensi global yang minimum dari J (θ, U) (yang berharap sesuai dengan clustering terbaik). Dengan kata lain, itu mengembalikan cluster-cluster sesuai dengan minimal dari daerah J (θ, U). Akibatnya, inisialisasi yang berbeda dari algoritma ini dapat mengakibatkan clustering akhir yang berbeda. Perawatan harus diambil dalam inisialisasi θ_i (lihat petunjuk praktis berikut ini). Jika nilai awal, katakanlah, m_1, dari θ_j terletak jauh dari wilayah di mana vektor data yang berada,mungkin tidak diperbarui. Sesuai dengan konsekuensi itu, algoritma k-means akan diproses jika nilainya hanya m-$m_1\theta_j$.

- Estimasi akurat jumlah cluster (representative) sangat penting untuk algoritma, karena perkiraan buruknya akan mencegah dari penguraian struktur clustering X. Lebih khusus lagi, jika jumlah besar dari representative yang digunakan, kemungkinan bahwa setidaknya satu "fisik" cluster akan dibagi menjadi dua atau lebih. Di sisi lain, jika sejumlah kecil dari representative digunakan, dua atau lebih cluster fisik kemungkinan akan diwakili oleh representative tunggal, yang pada umumnya akan terletak di wilayah yang tersebar dimana-mana (sehubungan dengan jumlah titik data) antara cluster.
- Algoritma ini sensitif terhadap adanya outlier (yaitu, titik yang terletak jauh dari hampir semua vektor data dalam X) dan "noise" vektor data. titik tersebut adalah hasil dari suatu gangguan proses yang tidak terkait dengan struktur clustering X. Karena baik outlier dan titik gangguan biasanya seharusnya ditugaskan untuk cluster, mereka mempengaruhi rata-rata masing-masing representative.
- k-means cocok untuk nilai real data dan, pada prinsipnya, tidak boleh digunakan dengan nilai data yang diskrit.

Petunjuk Praktis

- Dengan asumsi bahwa m adalah tetap, dan meningkatkan kesempatan untuk mendapatkan clustering yang dapat diandalkan, kita dapat menjalankan k-means beberapa kali, setiap kali menggunakan nilai awal yang berbeda untuk representative, dan pilih clustering terbaik (sesuai dengan J). Tiga metode sederhana untuk memilih nilai awal untuk θ_j adalah (a) inisialisasi acak, (b) secara acak pemilihan vektor m data dari X sebagai perkiraan awal dari θ_j, dan (c) pemanfaatan output clustering algoritma (misalnya, sekuensial) sebagai masukan sederhana.
- Dua cara sederhana untuk memperkirakan m adalah
  - dijelaskan penggunaan metodologi untuk algoritma BSAS
  - Untuk setiap nilai m, dipilih sesuai dengan rentangnya [mmin, Mmax], menjalankan algoritma k-means sejumlah nrun (setiap kali menggunakan nilai awal yang berbeda) dan menentukan clustering (menghasilkan nrun) yang meminimalkan fungsi nominal J.

Membiarkan J_m menjadi nilai dari J untuk clustering terakhir. Plot J_m dibandingkan m dan mencari perubahan daerah yang signifikan (itu timbul sebagai "knee" yang signifikan). Jika seperti knee terjadi, posisinya menunjukkan jumlah cluster yang diinginkan. Jika tidak, itu merupakan indikasi bahwa tidak ada struktur clustering (cluster yang berisi tersusun rapat) dalam kumpulan data.

- Dua cara sederhana untuk menangani outlier adalah (a) untuk menentukan titik yang terletak "luas" dari jarak di sebagian besar data dalam vektor X dan membuangnya, atau (b) untuk menjalankan k-means dan sangat mengidentifikasi cluster dengan beberapa elemen. Alternatifnya adalah dengan menggunakan algoritma yang kurang sensitif terhadap outlier (ini adalah kasus yang diselesaikan dengan algoritma PAM, yang akan dibahas lebih lanjut).

Contoh 7.5.1.

Menghasilkan dan memplot satu set data, X_3, yang terdiri dari N = 400 titik 2-dimensi. Titik-titik ini membentuk empat kelompok yang berukuran sama. Setiap kelompok berisi vektor yang merupakan suku dari distribusi gaussian dengan cara $m_1 = [0, 0]^T$, $m_2 = [10, 0]$, $m_3 = [0, 9]$, dan $M_4 = [9, 8]^T$, berturut-turut, dan masing-masing matriks kovarians

$$S_1 = I, \quad S_2 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1.5 \end{bmatrix}, \quad S_3 = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1.1 \end{bmatrix}, \quad S_4 = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.5 \end{bmatrix}$$

Di mana *I* menunjukkan matriks identitas 2 × 2. Kemudian lakukan hal berikut:

1. Terapkan algoritma k-means pada X_3 untuk m = 4. Menggunakan fungsi *rand* built-inMATLAB, menginisialisasi parameter vektor θ_j itu. Bandingkan perkiraan akhir dari nilai-nilai θ_j dengan rata-rata dari Gaussian, m_j. Plot parameter vektor θ_j dan titik X_3. Gunakan warna yang berbeda untuk vektor dari cluster yang berbeda.
2. Ulangi langkah 1 untuk m = 3.
3. Ulangi langkah 1 untuk m = 5.
4. Ulangi langkah 1, sekarang dengan nilai θ_j sebagai berikut: θ_1 (0) = [-2,0, -2,0]^T, θ_2 (0) = [-2,1, -2,1]^T, θ_3 (0) = [-2,0, -2,2] ^{T}, θ_4 (0) = [-2,1, -2,2] ^{T}.
5. Ulangi langkah 1, sekarang dengan θ_1, θ_2, dan θ_3 diinisialisasi secara acak seperti sebelumnya dan θ_4 (0) ditetapkan sama dengan [20, 20] ^{T}.
6. Berikan Komentar pada hasilnya.

Solusi. Untuk menghasilkan dan memplot X_3, kerjakan seperti pada Contoh 7.4.2, tapi dengan cara Gaussian yang berbeda. menunjukkan bahwa plot X_3 berisi empat cluster yang tersusun rapat secara jelas dipisahkan.

Lanjutkan sebagai berikut:

Langkah 1. Untuk menerapkan algoritma k-means untuk m = 4 dan inisialisasi acak dari θ_j ketik:

```
m=4;
[l,N]=size(X3);
rand('seed',0)
theta_ini=rand(l,m);
```

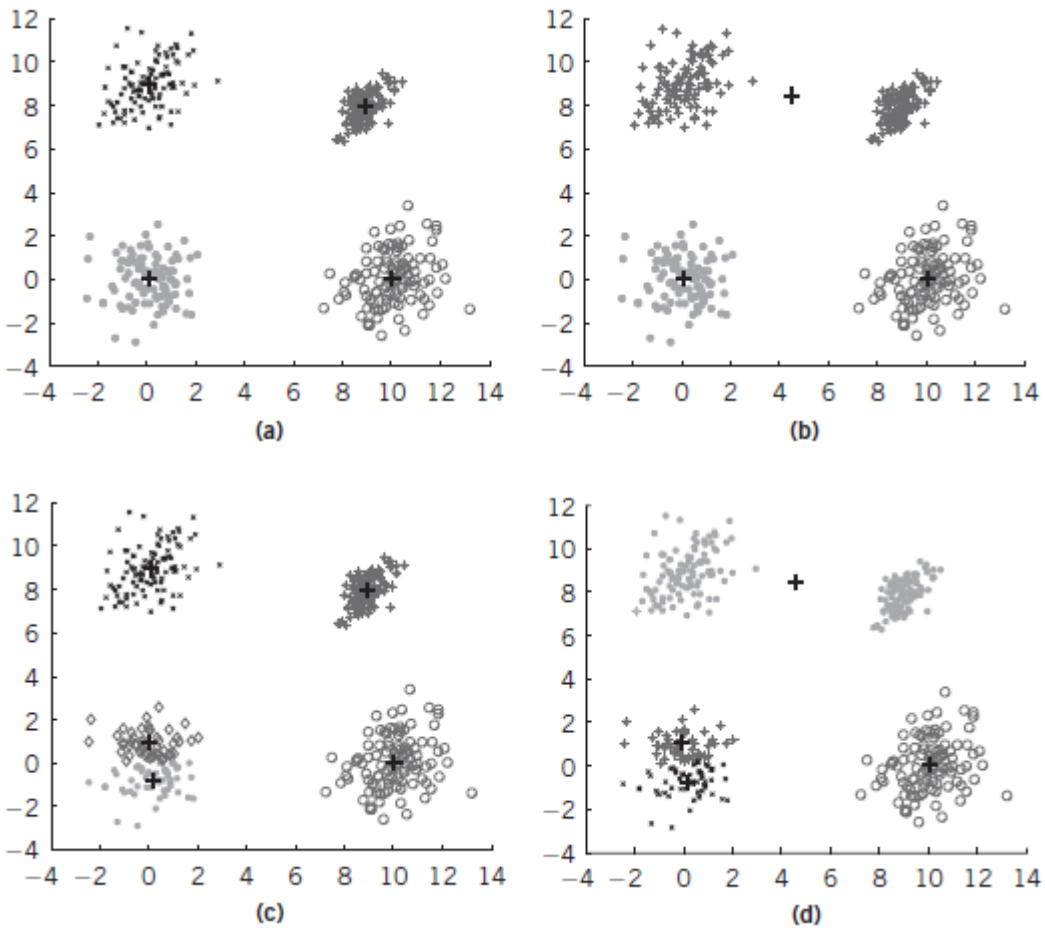
```
[theta,bel,J]=k_means(X3,theta_ini);
```

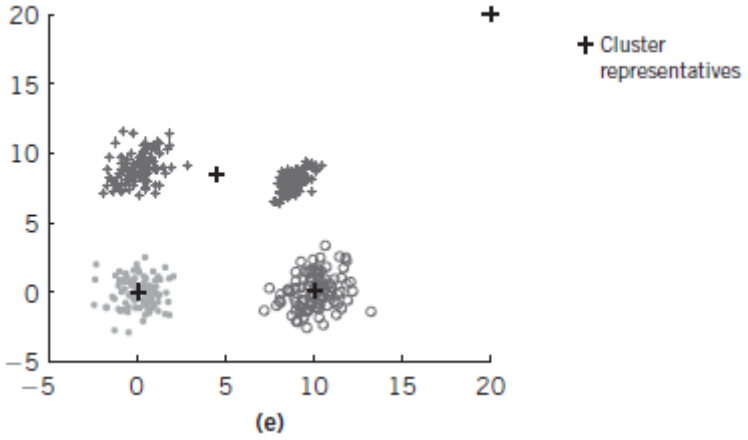
Untuk plot X_3, menggunakan warna yang berbeda untuk titk dari cluster yang berbeda, dan (Gambar 7.4 (a)) θ_j, ketik:

```
figure(1), hold on
figure(1), plot(X3(1,bel==1),X3(2,bel==1),'r.',...
X3(1,bel==2),X3(2,bel==2),'g*',X3(1,bel==3),X3(2,bel==3),'bo',...
X3(1,bel==4),X3(2,bel==4),'cx',X3(1,bel==5),X3(2,bel==5),'md',...
X3(1,bel==6),X3(2,bel==6),'yp',X3(1,bel==7),X3(2,bel==7),'ks')
figure(1), plot(theta(1,:),theta(2,:),'k+')
figure(1), axis equal
```

Langkah 2. kerjakan seperti pada langkah 1 untuk m = 3 (Gambar 7.4 (b)).

Langkah 3. kerjakan seperti pada langkah 1 untuk m = 5 (Gambar 7.4 (c)).





Hasil Clustering yang diperoleh oleh algoritma k-means dalam Contoh 7.5.1. titik dari cluster yang berbeda ditunjukkan oleh simbol dan / atau bayangan yang berbeda.

Langkah 4. Bekerja seperti pada langkah 1, untuk inisialisasi θ_j (lihat Gambar 7.4 (d)), ketik

theta_ini=[-2 -2; -2.1 -2.1; -2 -2.2; -2.1 -2.2]';

Langkah 5. Bekerja seperti pada langkah 1, untuk inisialisasi θ_j (lihat Gambar 7.4 (e)), ketik

theta_ini=rand(l,m);

theta_ini(:,m)=[20 20]';

Langkah 6. Dari hasil yang diperoleh pada langkah 1, amati bahwa k-means telah bisa mengidentifikasi cluster yang mendasari X_3. Selain itu, diperkirakan θ_j ialah di dalam persesuaian erat dengan m_j. Dalam semua kasus, bagaimanapun, k-means gagal untuk mengidentifikasi struktur clustering X_3. Secara khusus, dalam langkah 2 dan 3 itu membebankan struktur cluster-cluster pada X_3 dengan masing-masing tiga dan lima cluster, meskipun jumlah sebenarnya dari cluster yang mendasari adalah empat. Pada langkah 4, inisialisasi yang buruk mengarah θ_j untuk clustering yang berkualitas rendah. Akhirnya, pada langkah 5, kita memiliki inisialisasi yang buruk, di mana sekarang vektor parameter (θ_4) diinisialisasi berada jauh dari daerah tempat vektor-vektor X_3. Akibatnya, tidak pernah diperbarui dan hasil dari k-means jika m = 3.

Petunjuk

Perhatikan bahwa, meskipun X_2 tidak mengandung cluster, aplikasi k-means membebankan struktur clustering di atasnya, seperti yang terjadi dalam Contoh 7.5.1.

Tabel 7.1 Nilai Sarana Gaussian dan Nilai θ_j untuk Data Set dalam Contoh 7.5.1 dan 7.5.2

	Means (m_j's)	θ_j's (Example 7.5.1)	θ_j's (Example 7.5.2)
$j=1$	0	0.073	0.308
	0	0.026	0.282
$j=2$	9	8.955	8.778
	8	7.949	7.878
$j=3$	10	10.035	9.700
	0	0.058	0.214
$j=4$	0	0.075	0.290
	9	8.954	8.875

Contoh 7.5.2.

Menghasilkan dan memplot satu set data X_4, yang terdiri dari N = 500 titik 2-dimensi.Yang 400 pertama dihasilkan seperti pada Contoh 7.5.1, sedangkan 100 sisanya dihasilkan dari distribusi seragam di wilayah tersebut [-2, 12] × [-2, 12].

1. Terapkan algoritma k-means di X_4 untuk m = 4. Menginisialisasi θ_j seperti pada langkah 1 Contoh 7.5.1.
2. Bandingkan perkiraan yang diperoleh untuk θ_j dengan yang diperoleh pada langkah 1 contoh 7.5.1

Solusi. Untuk menghasilkan 400 titik pertama X_4, seperti pada Contoh 7.5.1. Untuk menghasilkan 100 yang tersisa, ketik

```
noise=rand(2,100)*14-2;
X4=[X4 noise];
```

Plot kumpulan data, ketik

```
figure(1), plot(X4(1,:),X4(2,:),'.b')
figure(1), axis equal
```

Jelas, titik data dari bentuk X_4 empat cluster, seperti yang terjadi dengan data X_3 yang ditetapkan dalam Contoh 7.5.1. Namun, sekarang mereka berada di adanya gangguan.

Langkah 1. Untuk menerapkan algoritma k-means untuk m = 4, bekerja seperti pada langkah 1 dari Contoh 7.5.1.

Langkah 2. Tabel 7.1 pada halaman sebelumnya menunjukkan nilai-nilai Gaussian upaya sebagai perkiraan θ_j yang diperoleh di sini dan di langkah 1 dari Contoh 7.5.1. Jelas, adanya gangguan degradasi memperkirakan kualitas θ_j yang diperoleh.

Contoh 7.5.3

1. Menghasilkan data set X_5 yang terdiri dari 515 titik data 2-dimensi. 500 suku pertama dari distribusi normal dengan rata-rata = m_1 $[0, 0]^T$; 15 suku yang tersisa dari distribusi normal dengan rata-rata $m_2 = [5, 5]^T$. Matriks kovarians dari distribusi adalah $S_1 = 1.5\ I$ dan $S_2 = I$, masing-masing, dimana *I* adalah matriks identitas yang ber ordo 2 x 2.
2. Terapkan algoritma kmeans pada X_5 untuk m = 2 dan mengambil kesimpulan.
   Solusi. Ambil langkah-langkah berikut:
   Langkah 1. Untuk menghasilkan kumpulan data X_5, ketik

```
randn('seed',0)
m=[0 0; 5 5];
S(:,:,1)=1.5*eye(2);
S(:,:,2)=eye(2);
n_points=[500 15];
```

```
X5=[];
for i=1:2
  X5=[X5; mvnrnd(m(i,:),S(:,:,i),n_points(i))];
end
X5=X5';
```

Plot set data, ketik

```
figure(1), plot(X5(1,:),X5(2,:),'.b')
figure(1), axis equal
```

Set data terdiri dari dua kelompok terpisah dengan ukuran yang baik secara signifikan tidak sama.

Langkah 2. Untuk menerapkan algoritma k-means dan plot hasilnya, bekerja seperti pada langkah 1 dalam Contoh 7.5.1. Dari hasil yang diperoleh, kita dapat melihat bahwa kegagalan algoritma untuk mengidentifikasi dua cluster telah berhasil. khususnya, itu menghentikan dengan dua cluster yang pertama dari yang (menyatakan kira-kira) satu setengah dari cluster yang sebenarnya "besar" yang mendasari X_5, yang kedua berisi titik yang tersisa dari cluster yang sebenarnya "besar" serta titik cluster yang sebenarnya "kecil".

Contoh 7.5.4

1. Menghasilkan dan memplot data set X_6 yang terdiri dari beragam bentuk cluster tidak tumpang tindih dalam ruang 2-dimensi. Cluster pertama terdiri dari 600 titik terletak di sekitar lingkaran berpusat di (0, 0) dan memiliki jari-jari sama dengan 6. Cluster kedua terdiri dari 200 titik terletak di sekitar elips berpusat di (0, 0) dan memiliki parameter a = 3 dan b = 1. Cluster ketiga terdiri dari 200 titik terletak di sekitar ruas garis dengan titik akhir (8, -7) dan (8, 7). Cluster keempat terdiri dari 100 titik terletak di sekitar setengah lingkaran berpusat di (13, 0) dan memiliki jari-jari sama dengan 3 dan koordinat y yang semuanya negatif.
2. Terapkan algoritma k-means set data X_6 dan plot hasil clustering. mengambil kesimpulan.

Solusi. Ambil langkah-langkah berikut:
Langkah 1. Untuk menghasilkan cluster pertama dari titik set data X_6, ketik

```
rand('seed',0)
n_points=[600 200 200 100]; %Points per cluster
noise=.5;
X6=[];
%Construction of the 1st cluster (circle, center (0,0), R=6)
R=6;
mini=-R;
maxi=R;
```

```
    step=(maxi-mini)/(fix(n_points(1)/2)-1);
    for x=mini:step:maxi
    y1=sqrt(R^2-x^2)+noise*(rand-.5);
    y2=-sqrt(R^2-x^2)+noise*(rand-.5);
    X6=[X6; x y1; x y2];
end
```

Untuk menghasilkan jenis cluster kedua, ketik

```
    %Construction of the 2nd cluster (ellipse, centered at (0,0), a=3,b=1))
    a=3;
    b=1;
    mini=-a;
    maxi=a;
    step=(maxi-mini)/(fix(n_points(2)/2)-1);
    for x=mini:step:maxi
    y1=b*sqrt(1-x^2/a^2)+noise*(rand-.5);
    y2=-b*sqrt(1-x^2/a^2)+noise*(rand-.5);
    X6=[X6; x y1; x y2];
end
```

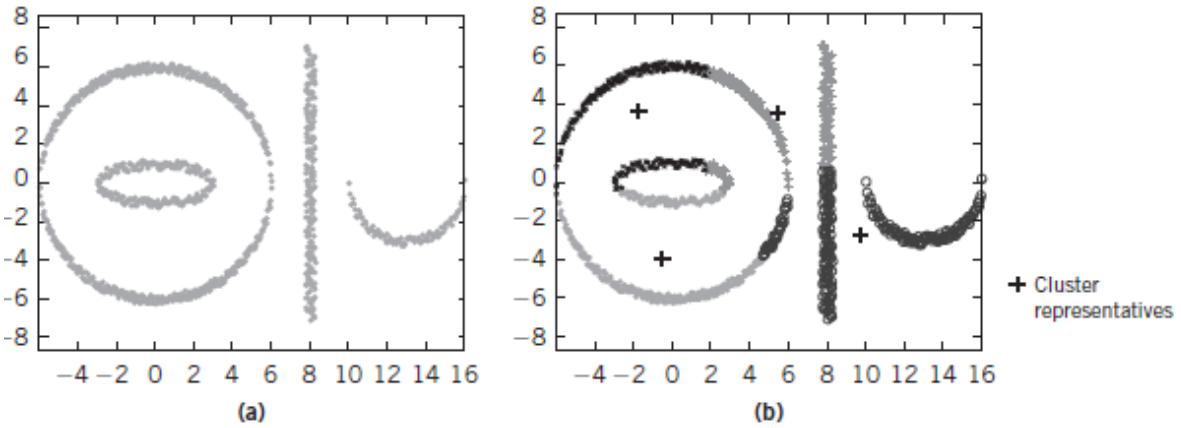
Untuk menghasilkan jenis cluster ketiga, ketik

```
    % Construction of the 3rd cluster (line segment, endpoints (8,-7), (8,7))
    mini=-7;
    maxi=7;
    step=(maxi-mini)/(n_points(3)-1);
    x_coord=8;
    for y=mini:step:maxi
    X6=[X6; x_coord+noise*(rand-.5) y+noise*(rand-.5)];
end
```

Terakhir, untuk menghasilkan jenis cluster keempat, ketik

```
    %Construction of the 4th cluster (semicircle, center (13,0), R=3;, y<0)
    R=3;
    x_center=13;
    mini=x_center-R;
    maxi=x_center+R;
    step=(maxi-mini)/(n_points(4)-1);
    for x=mini:step:maxi
```

```
y=-sqrt(R^2-(x-x_center)^2)+noise*(rand-.5);
X6=[X6; x y];
end
X6=X6';
```



Gambar 7.5

(a) data yang dihasilkan diatur dalam Contoh 7.5.4. (b) hasil Clustering diperoleh dengan k-means. Simbol yang berbeda dan / atau abu-abu menunjukkan titik dari cluster yang berbeda.

Plot set data (lihat Gambar 7.5 (a)), ketik

```
figure(5), plot(X6(1,:),X6(2,:),'.b')
figure(5), axis equal
```

Langkah 2. Terapkan k-means pada X_6 dan plot hasilnya, menjalankan seperti pada langkah 1 dari Contoh 7.5.1 (lihat Gambar 7.5 (b)). Hal ini jelas bahwa, pada prinsipnya, k-means tidak dapat menangani kasus-kasus dimana cluster yang tidak tersusun rapat mendasari set data. Jika ada indikasi bahwa seperti mendasari clustering dalam set data, algoritma clustering lainnya harus dimanfaatkan, sebagaimana akan dibahas nanti.

Contoh berikutnya menunjukkan bagaimana k-means dapat memperkirakan jumlah cluster, m, dan, berdasarkan itu, memperkirakan clustering yang paling cocok dengan data. Tentu saja, diasumsikan bahwa hanya cluster yang tersusun rapat yang terlibat.

Contoh 7.5.5

1. Pertimbangkan (dan plot) kumpulan data X_3 yang dihasilkan dalam Contoh 7.5.1. Untuk setiap nilai (integer) m dalam rentang [mmin, Mmax], algoritma k-means menjalankan beberapa kali nruns dan dari nruns yang diproduksi clustering tetap satu dengan nilai minimum, Jm,dari J. Plot Jm vs m. Jika grafik yang dihasilkan merupakan sebuah nilai "knee," yang signifikan menunjukkan posisi jumlah cluster yang mungkin mengidikasi X_3. Jika tidak, kita memiliki indikasi bahwa X_3 memiliki adanya kemungkinan struktur clustering. Gunakan mmin = 2, Mmax = 10, dan nruns = 10.
2. Ulangi langkah 1 untuk data set X_4, X_1, dan X_2, yang membandingkan dalam Contoh 7.5.2 dan 7.4.2 dan 7.5.1 masing-masing.
3. Mengambil kesimpulan.

Solusi. Ambil langkah-langkah berikut:

Langkah 1. Untuk menghasilkan set data X_3, upaya seperti pada Contoh 7.5.1. Untuk melakukan prosedur yang diuraikan sebelumnya, ketik

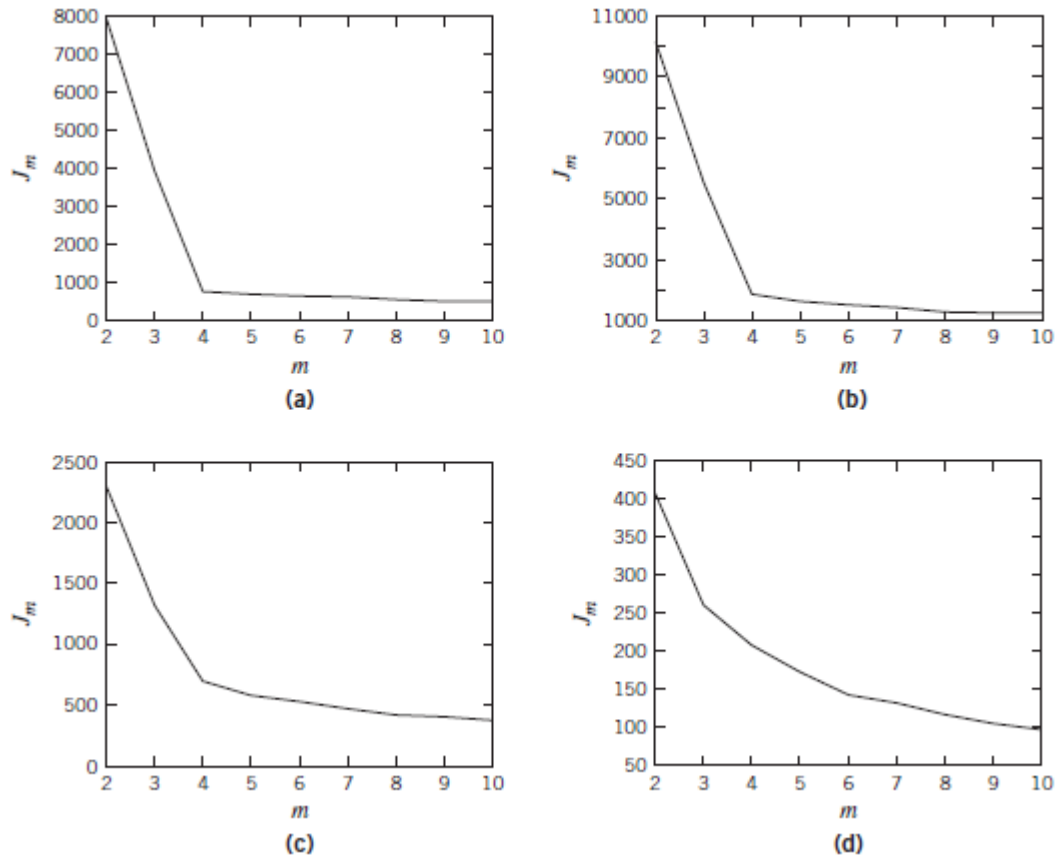
```
[l,N]=size(X3);
nruns=10;
m_min=2;
m_max=10;
J_m=[];
for m=m_min:m_max
        J_temp_min=inf;
        for t=1:nruns
                rand('seed',100*t)
                theta_ini=rand(l,m);
                [theta,bel,J]=k_means(X3,theta_ini);
                if(J_temp_min>J)
                        J_temp_min=J;
                end
        end
        J_m=[J_m J_temp_min];
end
m=m_min:m_max;
figure(1), plot(m,J_m)
```

Langkah 2. Ulangi langkah 1 untuk masing-masing tiga kasus.

Langkah 3. Plot dari J_m dibandingkan m untuk setiap kasus yang ditunjukkan pada Gambar 7.6. Dari gambar tersebut, maka untuk set data X_3 terdapat knee yang tajam pada m = 4. Hal ini merupakan indikasi bahwa jumlah cluster yang mendasari X_3 adalah empat. Sebuah knee yang tajam serupa diperoleh pada m = 4 untuk X_4, yang merupakan versi noise dari X_3. Dalam plot untuk X_1, di mana cluster yang tidak begitu jelas dipisahkan, knee yang kurang tajam ditemui pada m = 4. Dalam semua kasus ini, metodologi menyediakan indikasi dari jumlah cluster yang mendasari set data. Namun, dalam plot X_2, tidak ada knee yang signifikan yang dihadapi, yang merupakan kenyataan bahwa tidak ada indikasi struktur clustering yang ada.

Partitioning Around Medoids Algorithm

Algoritma Partisi Around Medoids (PAM) menyerupai algoritma k-means. Perbedaan utama adalah bahwa representative cluster menjadi titik set data X yang terbatas. Sebagai contoh, seperti:



Plot Jm dibandingkan m untuk data set pada Contoh 7.5.5: (a) X_3 (knee tajam), (b) X_4 (knee tajam), (c) X_1 (knee kurang tajam), dan (d) X_2 (knee yang tidak signifikan).

Kendala yang dapat diterapkan bila vektor dari X memiliki unsur-unsur yang diambil dari nilai sekumpulan data diskrit, yaitu, dari subset dari himpunan bilangan bulat [Theo 09, Bagian 11.2.2]. Sekali lagi, jumlah cluster (tersusun rapat) yang mendasari set data diasumsikan telah diketahui. Set Θ dari vektor di dalam X yang menggambarkan struktur clustering yang terbaik (juga dikenal sebagai medoids) ditentukan melalui minimalisasi fungsi nominal J(Θ). Hal ini didefinisikan sebagai penjumlahan, atas semua vektor data,

jarak antara setiap vektor data dan medoid terdekatnya.

Algoritma iteratif. Ini dimulai dengan menetapkan m secara acak dipilih vektor dari X didalam Θ dan kemudian menghitung nilai dari J, J (Θ). Pada setiap iterasi, semua set $\Theta_{ij} = (\Theta-\{x_i\}) \cup \{x_j\}$, $x_i \in \Theta$ dan $x_j \in X - \Theta$ dipertimbangkan. Dengan kata lain, Θ_{ij} akan menghasilkan Jika x_i dihapus dari Θ dan x_j dimasukkan. Untuk setiap Θ_{ij}, Nilai dari fungsi niminal J, $J(\Theta_{ij})$, dan yang dihitung (katakanlah Θ_{qr}) Yang $J(\Theta qr)$ Adalah minimum yang dipilih. Jika $J(\Theta qr) < J(\Theta)$, Kemudian Θ_{qr} menggantikan Θ dan prosedur ini diulang. Jika tidak, algoritmanya berhenti. Cluster ke i, C_i, dibentuk oleh algoritma ini diidentifikasi oleh vektor-vektor yang terletak lebih dekat medoid ke i (yang akhirnya menghasilkan(Θ) dibandingkan dengan medoids lainnya.

Untuk menerapkan algoritma PAM pada set data, X, ketik

[*bel,cost,w,a,cost*] = *k_medoids(X,m, sed)*

Dimana

X adalah matriks l x N yang kolom-kolomnya berisi data vektor.

m adalah jumlah cluster,

sed merupakan skalar integer yang digunakan sebagai sumber untuk fungsi rand built-inMATLAB,
bel adalah vektor N-dimensi yang berisi label elemen ke i dari cluster dimana data ke-i
vektor diberikan setelah konvergensi dari algoritma,

nominal adalah nilai dari J (Θ) yang sesuai dengan clustering (akhir) yang dihasilkan oleh algoritma,

w adalah sebuah matrix $l \times m$ dengan kolom-kolom cluster representative (medoids) yang diperoleh setelah konvergensi dari algoritma,

a adalah vektor m-dimensi yang berisi indeks dari vektor data yang digunakan sebagai medoids.

Keterangan

- Seperti k-means, PAM menentukan struktur clustering pada set data X, meskipun vektor data yang di X tidak menunjukkan struktur clustering.
- Algoritma ini cocok untuk bernilai real serta data yang bernilai diskrit.
- PAM kurang sensitif terhadap adanya gangguan dibandingkan dengan k-means.
- PAM cocok untuk set data yang kecil. Namun, tidak efisien untuk set data besar karena nominal komputasi per iterasi adalah sangat signifikan [Theo 09, Bagian 14.5.2]. Untuk mengatasi masalah ini, algoritma lain dengan filosofi yang sama, seperti Clara dan CLARANS, komputasinya yang kurang intensif, telah dikembangkan. Namun, mereka tidak dapat menjamin konvergensi ke lokal minimum dari fungsi nominal J (?) [Theo 09, Bagian 14.5.2].

Contoh 7.5.6

1. Menghasilkan dan memplot set data X_7, yang terdiri dari N = 216 vektor 2-dimensi. 100 Suku pertama dari distribusi Gaussian dengan mean $m_1 = [0, 0]^T$; 100 suku berikutnya dari distribusi Gaussian dengan mean $m_2 = [13, 13]^T$. Dua kelompok lainnya dari delapan titik suku distribusi Gaussian dengan masing-masing mean $m_3 = [0, -40]^T$ dan $m_4 = [-30, -30]^T$. Matriks kovarians untuk semua Gaussians sama dengan matriks identitas ber ordo 2×2. Jelas, dua terakhir titik cluster dapat dianggap outlier.
2. Terapkan k-means dan algoritma PAM di X_7, untuk m = 2. Plot clustering dalam setiap kasus dan beri komentar hasilnya.

Solusi. Ambil langkah-langkah berikut:

Langkah 1. Untuk menghasilkan dan plot X_7, ketik

```
randn('seed',0)
m=[0 0; 13 13; 0 -40; -30 -30]'; %means
[l,n_cl]=size(m);
S=eye(2); %covariance matrix
n_points=[100 100 8 8]; %points per distribution
X7=[];
for i=1:n_cl
    X7=[X7; mvnrnd(m(:,i)',S,n_points(i))];
```

```
end
X7=X7';
figure(1), plot(X7(1,:),X7(2,:),'.b')
```

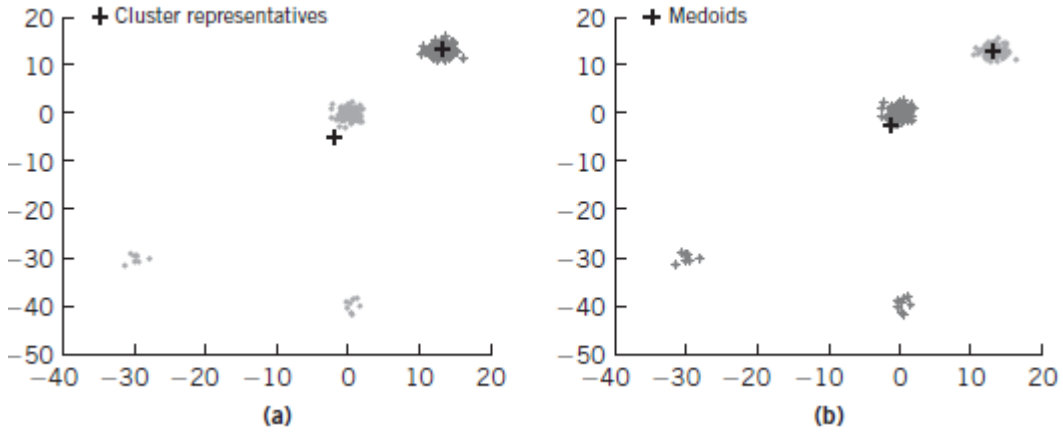
Langkah 2. Untuk menerapkan algoritma k-means di X_7, ketik

```
m=2;
[l,N]=size(X7);
rand('seed',0)
theta_ini=rand(l,m);
[l,m]=size(theta_ini);
[theta,bel,J]=k_means(X7,theta_ini);
```

Plot vektor data, dengan menggunakan warna yang berbeda untuk ttik yang berasal dari cluster yang berbeda, bekerja seperti pada langkah 1 dari Contoh 7.5.1 (lihat Gambar 7.7 (a)).

Untuk menerapkan algoritma PAM di X_7, ketik

```
[l,N]=size(X7);
m=2; %Number of clusters
sed=0; %Seed for the rand function
[bel,cost,w,a]=k_medoids(X7,m,sed)
```



Gambar 7.7

Clusterings dihasilkan oleh (a) k-means dan (b) PAM bila diterapkan pada set data X_7, dibandingkan pada Contoh 7.5.6.

Plot hasil clustering, bekerja seperti pada langkah 1 dari Contoh 7.5.1 (lihat Gambar 7.7 (b)). Representative dari cluster pertama (C1) dihitung oleh k-means θ_1 = [-1,974, -4,789] ^{T}; medoid yang sesuai dihitung dengan PAM yaitu [-1,3414, -2,6695] ^{T}. Perkiraan kedua adalah jauh lebih dekat [0, 0]^T dari pada yang pertama (berarti aktual dari volume utama dari vektor data yang di cluster C_1). Ini terlihat dalam Gambar 7.7 (b), yang menunjukkan bahwa medoid yang sesuai dengan cluster pertama,C_1, volume data utama tetap dekat pada C_1. Hal ini karena medoids dibatasi pada X_7. Dengan kata lain, medoid dari C_1 tidak diperbolehkan untuk dipindahkan di "wilayah kosong" antara 100 titik pertama dan kedua cluster outlier. Pembatasan ini tidak berlaku untuk k-means (lihat Gambar 7.7 (a)). Akibatnya, dalam hal ini, representative dari C_1 dipengaruhi oleh

outlier dan dengan demikian bukanlah representasi yang baik dari volume utama dari vektor data C_1.

Latihan 7.5.2
Contoh 7.5.1 Ulangi untuk algoritma PAM.

Latihan 7.5.3
Contoh 7.5.5 Ulangi untuk algoritma PAM.

Generalized Mixture Decomposition Algorithmic Scheme

Algoritma ini (GMDAS) bergantung pada kerangka probabilistik. Sekali lagi, jumlah cluster m (yang tersusun rapat), diasumsikan yang diketahui. Secara khusus, probability density function (pdf) p (x) yang menggambarkan set data X dimodelkan oleh campuran yang diberatkan dari distribusi Gaussian m, p (x | j), j = 1,. . . , m, masing-masing terkait dengan sebuah cluster:

$$p(x) = \sum_{j=1}^{m} P_j p(x|j)$$

dimana p (x | j) model cluster j. Setiap p (x | j) ditentukan oleh rata-rata m_j dan kovarians matriks S_j.

Tujuan GMDAS adalah untuk menyesuaikan parameter m_i dan S_j dari masing-masing p (x | j), serta parameter campuran P_j (dikenal juga sebagai probabilitas apriori). Untuk mencapai hal ini, ia bekerja secara iteratif. Beberapa inisialisasi awal m_i (0), S_i (0), P_i (0) yang diadopsi untuk m_j, S_j, P_j, masing-masing, j = 1,. . . , m. Kemudian, pada setiap iterasi, update algoritma, dalam urutan

- sebuah probabilitas posteriori, P (j | xi) yang berasal dari distribusi x_i bahwa model cluster C_j, j = 1,. . . , M, i = 1,. . , N.,
- rata-rata m_j,
- kovarians matriks S_j,
- probabilitas apriori P_j.

GMDAS berakhir ketika tidak ada perubahan yang signifikan dalam nilai-nilai dari parameter m_j, S_j, dan P_j, j = 1,. . . , m, ditemui antara dua iterasi yang berurutan [Theo 09, Bagian 14.2].

Perhatikan bahwa GMDAS tidak menentukan secara eksplisit clustering berdasarkan X. Sebaliknya, ini memberikan (di samping perkiraan parameter dari distribusi Gaussian) sebuah probabilitas posteriori P (j | xi), j = 1,. . . , m, i = 1,. . . , N. Namun, jika clustering spesifik diperlukan, kita dapat mendefinisikan C_q sebagai cluster yang berisi semua x_i untuk P (q | xi) adalah maksimum antara semua P (j | xi) 's, j = 1,. . . , m.

Untuk menerapkan GMDAS pada set data X, ketik

[*ap,cp,mv,mc, iter,diffvec*] = *GMDAS(X,mv_ini,mc_ini, e,maxiter, sed)*

Dimana

X adalah matriks l x N yang berisi vektor data dalam kolom-kolomnya,
mv_ini adalah matriks l × m kolom yang berisi inisialisasi awal rata-rata dari distribusi,

mc_ini adalah matriks l × l × m yang 2-dimensi l × l "irisan" adalah inisialisasi awal dari distribusi matriks kovarians,

e adalah melibatkan ambang dalam kondisi mengakhiri dari algoritma,[4]

maxiter adalah jumlah maksimum iterasi untuk menjalankan algoritma yang diperbolehkan,
sed adalah sumber yang digunakan untuk inisialisasi fungsi rand built-inMATLAB,

ap adalah vektor m-dimensi yang berisi perkiraan akhir dari suatu probabilitas apriori,

cp adalah matriks N × m dengan elemen (i, j) yang merupakan probabilitas bahwa vektor i berasal dari distribusi model cluster j,

mv dan mc berisi perkiraan akhir dari rata-rata dan matriks kovarians, masing-masing, dan berbagi struktur dengan masing-masing *mv_ini* dan *mc_ini*,

iter adalah jumlah iterasi yang dilakukan oleh algoritma,

diffvec adalah vektor dengan koordinat ke t yang berisi perbedaan absolut dari jumlah elemen dari mv, mc, dan apriori probabilitas antara iterasi ke t dan ke (t -1).

Keterangan

- seperti k-means dan algoritma PAM, GMDAS menentukan struktur clustering pada X, bahkan jika struktur seperti ini tidak dibenarkan.
- sebuah algoritma yang meminimalkan sesuai fungsi yang didefinisikan dan itu menjamin daerah konvergensi minimum.
- algoritma sensitif terhadap outlier, karena persyaratan bahwa $\sum_{j=1}^{m} P(j \mid x_i) = 1$ untuk semua x_i.
- GMDAS menuntut komputasi karena, pada setiap iterasi, itu memerlukan inversi matriks kovarians dari m. Dua cara untuk mengatasi masalah ini adalah (a) untuk menganggap bahwa matriks kovarians dari semua distribusi adalah sama dan / atau (b) untuk mengasumsikan bahwa setiap matriks kovarians adalah diagonal.

Latihan 7.5.4
Contoh 7.5.1 Ulangi menggunakan algoritma GMDAS.

Petunjuk
Untuk mendapatkan hard clustering berbasis pada sebuah probabilitas posteriori pada matriks N × m, cp, ketik

```
[qw,bel]=max(cp');
```

Dimana bel berisi label cluster dari vektor data.

Algoritma berakhir ketika jumlah perbedaan absolut dari mv, mc, dan apriori probabilitas antara dua iterasi yang berurutan lebih kecil dari e.

Latihan 7.5.5
Contoh 7.5.5 Ulangi menggunakan algoritma GMDAS.

Petunjuk
Lihat petunjuk dalam latihan sebelumnya untuk mendapatkan hard clustering dari sebuah probabilitas aposteriori.

7.5.2 Algoritma Clustering Nonhard

Berbeda dengan algoritma sebelumnya, algoritma dalam kategori ini mengasumsikan bahwa setiap data vektor mungkin memiliki (atau mungkin tidak kompatibel dengan) lebih dari satu cluster sampai jumlah tertentu.

Fuzzy c-Means Algoritma

Dalam algoritma c-means fuzzy (FCM) setiap cluster (yang tersusun rapat) direpresentasikan oleh sebuah parameter vektor θ_j, j = 1,. . . , m. Juga, diasumsikan bahwa vektor x_i dari set data X tidak selalu eksklusif memiliki sebuah cluster C_j. Sebaliknya, mungkin dimiliki secara bersamaan untuk lebih dari satu cluster hingga beberapa derajat. variabel uij mengkuantifikasi "keanggotaan kelas" dari x_i dalam cluster C_j, dan diperlukan bahwa $u_{ij} \in [0, 1]$ dan $\sum_{j=1}^{m} u_{ij} = 1$ untuk semua x_i. Sekali lagi, jumlah cluster, m, diasumsikan yang diketahui.

Tujuan dari FCM adalah untuk memindahkan setiap m l-dimensi yang didapatkan parameter vektor (representative) θ_j, j = 1,. . , m, menuju daerah di ruang data yang padat di titik data. Akhirnya, algoritma melibatkan tambahan parameter q (> 1) tersebut disebut fuzzifier.

FCM adalah salah satu algoritma yang paling populer. Hal ini sangat iterative, dimulai dengan beberapa perkiraan awal, θ_1 (0),. . . , θ_m (0), untuk θ_1,. . . , θ_m, masing-masing, dan pada setiap t iterasi:

- Nilai keanggotaan, u_{ij} (t -1), dari data vektor x_i dalam cluster C_j, i = 1,. . . , N, j = 1,. . , m., Dihitung, dengan mempertimbangkan jarak (Euclidean kuadrat) x_i dari semua θ_j, j = 1,. . ., m.
- Representative θ_j diperbarui sebagai sarana yang diberatkan dari semua vektor data (setiap data vektor xi dengan bobot $u_{ij}^{q}(t-1))$.

Algoritma berakhir ketika perbedaan dalam nilai-nilai θ_j antara dua iterasi yang berurutan cukup kecil. Ia mengembalikan nilai dari parameter vektor (representative) θ_j dan u_{ij} itu, i = 1,. . . , N, j = 1,. . . , m. Jika hard clustering diperlukan, kita dapat mendefinisikan C_j sebagai cluster yang berisi semua x_i untuk $u_{ij} > u_{ik}$, k ≠ j.

Untuk menerapkan algoritma FCM, ketik

[*theta,U, obj_ fun*] = *fuzzy_c_means(X,m,q)*

Dimana

vektor *X* berisi data dalam kolom-kolomnya,
m adalah jumlah cluster,
q adalah fuzzifier,
theta yaitu berisi representative cluster dalam kolom-kolomnya,
U adalah matriks N × m yang berisis di dalam jajaran ke i dari keanggotaan kelas x_i dalam cluster m,

obj_fun adalah vektor koordinat ke t adalah nilai dari fungsi nominal, J, untuk clustering dihasilkan pada iterasi ke t.

Keterangan

- Seperti semua algoritma fungsi optimasi nominal disajikan sebelumnya, FCM mentukan struktur clustering pada X, bahkan jika hal ini tidak dibenarkan secara fisik.
- FCM berasal dari minimalisasi fungsi nominal.

$$J(\theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^{q} ||x_i - \theta_j||^2$$

Dimana $\theta = [\theta_1^T, \ldots, \theta_m^T]^T$ pada batasan pokok $u_{ij} \in [0, 1]$ dan $\sum_{j=1}^{m} u_{ij} = 1$ Artinya, J(θ, U) adalah pembobot jumlah jarak semua x_i dari semua θ_j.

- Keterlibatan q sangat penting dalam clustering fuzzy. Nilai-nilai khas dari q dalam rentang [1,5, 3] [Theo 09, Bagian 14.3].
- Algoritma ini sensitif dengan adanya outlier karena syaratnya bahwa m_j = 1 uij = 1 untuk semua x_i.
- algoritma fuzzy clustering lainnya dimana hipercurve dari tingkat kedua atau hyperplanes digunakan sebagai representative juga telah diusulkan. Hal ini berguna terutama dalam aplikasi pengolahan citra [Theo 09, Bagian 14.3.2].

Latihan 7.5.6
Ulangi Contoh 7.5.1, menggunakan FCM dengan q = 2.

Latihan 7.5.7
Ulangi Contoh 7.5.3, menggunakan FCM dengan q = 2.

Latihan 7.5.8
Ulangi Contoh 7.5.5, menggunakan FCM dengan q = 2.

Latihan berikutnya menunjukkan pengaruh parameter q fuzzifier dalam clustering yang dihasilkan.

Latihan 7.5.9

Terapkan FCM pada kumpulan data X3 yang dihasilkan dalam Contoh 7.5.1 untuk q = 2,q = 10, dan q = 25. Tentukan dan plot tiga clusterings keras yang sesuai, seperti yang dibahas sebelumnya. Bandingkan parameter u_{ij} dan θ_j untuk tiga kasus dan tarik kesimpulan.

Petunjuk

Untuk nilai-nilai rendah dari q (misalnya, q = 2), masing-masing data vektor ternyata secara eksklusif memiliki hampir cluster tunggal [Theo 09, Bagian 14.3]. Artinya, untuk setiap x_i, hanya u_{ij} tunggal memiliki nilai yang sangat tinggi (di atas 90%) di antara $u_i1,\ldots, u_{im}$. Namun, seperti menaikkan q, u_{ij} untuk setiap data vektor x_i cenderung menjadi sama dengan 1/m = 0,25. Terutama dalam kasus di mana q = 25, ini mengarah pada clustering yang tidak sesuai dengan struktur pengelompokan sejati yang mendasari X3.

Contoh berikutnya menunjukkan efek dari outlier pada kinerja FCM tersebut.

Contoh 7.5.7. Terapkan algoritma FCM pada set data yang dihasilkan X_7 dalam Contoh 7.5.6. Menghasilkan hard clustering, seperti yang dibahas sebelumnya, dan plot hasilnya. Komentar pada grade keanggotaan dari titik data dalam dua kelompok yang diperoleh. Bandingkan representative yang dihasilkan dengan yang diperoleh dari aplikasi k-means dan PAM di X_7.

Solusi. Untuk menerapkan algoritma FCM pada X_7, ketik

```
[theta,U,obj_fun] = fuzzy_c_means(X7,m,q)
```

Untuk mendapatkan hard clustering menggunakan U, ketik

```
[qw,bel]=max(U');
```

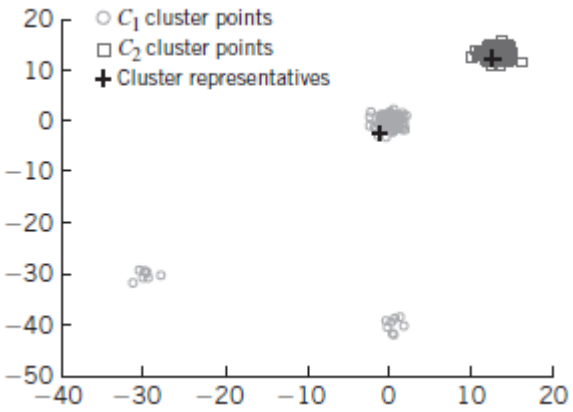
dimana bel berisi label cluster dari vektor data.

Plot hasil clustering, menggunakan simbol yang berbeda dan corak untuk vektor yang dimiliki cluster yang berbeda, seperti pada langkah 1 dari Contoh 7.5.1 (lihat Gambar 7.8).

Observasi keanggotaan kelas mengungkapkan bahwa

- Untuk 100 titik pertama, kelas keanggotaan dalam cluster C_1 adalah signifikan lebih tinggi (> 89,4%) dibandingkan pada cluster C_2 (<10,6%) (lihat Gambar 7.8).
- Untuk 100 titik berikutnya, kelas keanggotaan dalam cluster C_2 adalah signifikan lebih tinggi (> 97,2%) dibandingkan pada cluster C_1 (<2,8%).
- Selama 16 titik terakhir (outlier), kelas keanggotaan dalam cluster C_1 dan C_2 adalah signifikan (> 66,62% untuk C1 dan> 30,10% untuk C2), sehingga efeknya pada perhitungan θ_1 dan θ_2 keduanya tidak dapat diabaikan.

Membandingkan hasil yang ditunjukkan pada Gambar 7.8 dengan yang pada Gambar 7.7, kita amati bahwa perkiraan θ_2 (representative dari cluster kanan atas) lebih baik untuk k-means dan PAM dari pada FCM (ini karena outlier tidak berpengaruh pada estimasi θ_2 pada k-means dan PAM, yang tidak terjadi di FCM), dan bahwa perkiraan θ_1 (representative dari cluster lain) yang lebih baik dalam PAM dan FCM dari pada k-means, dalam arti bahwa di PAM dan FCM θ_1 tetap dekat dengan volume utama dari kumpulan data. Clustering berlaku oleh FCM pada set data X_7 yang ada di dalam Contoh 7.5.7. Tiga kelompok kiri bawah titik dari cluster C_1, kelompok titik kanan atas merupakan kluster C_2.



Gambar 7.8 Clustering diperoleh oleh FCM pada data diatur dalam Contoh 7.5.7 X_7 Tiga kelompok kiri bawah titik dari cluster C1;. Kelompok kanan atas titik merupakan kluster C2.

Hal ini terjadi karena pada FCM outlier berkontribusi untuk estimasi dari θ_1 oleh (setidaknya) 30%, sementara di k-means mereka berkontribusi dengan 100% (karena dalam kasus hard clustering vektor memiliki eksklusif (100%) ke cluster tunggal) .

Possibilistic c-Means Algoritma

Algoritma ini (dikenal sebagai PCM) juga sesuai untuk mengungkap cluster yang tersusun rapat. Kerangka di sini adalah sama dengan yang digunakan dalam FCM: Setiap data vektor x_i dikaitkan dengan cluster C_j melalui skalar u_{ij}. Namun, kendalanya bahwa semua $u_{ij.}$ diberikan jumlah x_i sampai dengan 1 dibuang (hanya diperlukan bahwa terletak pada interval [0, 1]). Sebagai akibatnya, nilai u_{ij} (untuk x_i diberikan) ini tidak berhubungan lagi dan tidak dapat diartikan sebagai "nilai keanggotaan" dari vektor x_i dalam cluster C_j, karena istilah ini menyiratkan bahwa penjumlahan yang u_{ij} untuk setiap x_i yang seharusnya konstan. Sebaliknya, u_{ij} ditafsirkan sebagai

"derajat kesesuaian" antara x_i dan C_j. Derajat kesesuaian antara x_i dan C_j adalah independen antara x_i dan cluster yang tersisa.

Seperti FCM, parameter q (> 1) menyangkut dalam PCM. Namun itu tidak berlaku sebagai fuzzifier seperti pada kasus di FCM. Juga, berbeda dengan FCM, PCM kurang sensitif dalam mengetahui jumlah cluster yang tepat. Sebaliknya, nilai lebih dari m dapat digunakan (lihat juga pernyataan yang diberikan di bawah). Satu set dari parameter η_j, j = 1,. . , m., Masing-masing sesuai dengan cluster, juga diperlukan (bebas menyatakan, parameter estimasi dari "ukuran" cluster [Theo 09, Bagian 14,4]). Seperti k-means dan FCM, tujuan PCM adalah untuk memindahkan ke ruang θ_j yang rapat di daerah titik data.

PCM itu iteratif. Dimulai dengan beberapa perkiraan awal, θ_1 (0),. . . , θ_m (0), untuk θ_1,. . . , θ_m, masing-masing, dan pada setiap iterasi,

- Derajat kesesuaian", u_{ij} (t -1), dari vektor data x_i ke cluster C_j, i = 1,. . . , N, j = 1,. . , m., Dihitung, dengan mempertimbangkan jarak (Euclidean kuadrat) dari parameter x_i dari θ_j dan η_j.
- Representative, θ_j, diperbarui, seperti di FCM, sebagai upaya pembobotan dari semua vektor data (setiap data vektor x_i dibobot dengan $u_{ij}^{\tilde{q}}(t-1))$.

Algoritma berakhir ketika perbedaan dalam nilai θ_j antara dua iterasi yang berurutan cukup kecil. ini mengembalikan nilai dari parameter vektor (representative) θ_j dan "koefisien kompatibilitas", u_{ij}, i = 1,. . . , N, j = 1,. . , m.

Untuk menerapkan PCM pada set data X, ketik

[*U*, *theta*] = *possibi(X,m, eta,q, sed, init_ proc, e_thres)*

Dimana
vektor *X* berisi data dalam kolom-kolomnya,
m adalah jumlah cluster,
eta adalah array m-dimensi ke-j yang parameter koordinat η_j untuk cluster C_j , q adalah "q" dari parameter algoritma,
sed adalah sebuah skalar integer yang digunakan sebagai sumber untuk fungsi rand built-inMATLAB,

init_ proc adalah bilangan bulat yang mengambil nilai 1, 2, atau 3, dengan 1 yang sesuai dengan prosedur inisialisasi *rand_init*, yang memilih secara acak vektor *m* dari yang terkecil *hyper-rectangular* yang berisi semua vektor dari X dan sisi-sisinya sejajar dengan sumbu; 2 sesuai dengan *rand_data_init*, yang kemudian memilih m secara acak diantara vektor N dari X, dan 3 sesuai dengan *distant_init*, yang memilih vektor m dari X yang "paling jauh" dari satu sama lain. (Prosedur yang terakhir mencapai, secara umum, inisialisasi yang lebih baik pada nominal perhitungan meningkat),

e_thres adalah ambang batas yang terlibat dalam kondisi akhir dari algoritma,

U adalah matriks N × m dengan elemen (i, j) yang menunjukkan "derajat kesesuaian" dari data vektor ke-i dengan cluster ke-j (setelah konvergensi dari algoritma),

theta adalah matriks l × m, setiap kolom sesuai dengan cluster yang representative (setelah konvergensi dari algoritma).

Keterangan

- Berbeda dengan algoritma yang sebelumnya dibahas dalam bagian ini, PCM tidak memaksakan struktur clustering berdasarkan X. Ini berarti bahwa ketika bilangan representative menggunakan lebih tinggi dari bilangan "sejati" dari beberapa cluster, beberapa konvergensi setelah θ_j akan (hampir) bertepatan, dan jika algoritma dimulai dari inisialisasi titik yang tepat, maka berharap semua cluster (bidang rapat) akan diwakili oleh satu θ_j sementara beberapa dari mereka dapat representasi oleh dua atau lebih θ_j yang (hampir) sama. Di sisi lain, ketika jumlah representative, m, kurang dari jumlah sebenarnya dari cluster, katakanlah k, kemudian setelah konvergensi algoritma akan berpotensi mengulangi m yang keluar dari cluster k. konsekuensinya, dimana kasus representative yang jarang terletak di daerah antara cluster, yang tidak ditemui.
- Seperti algoritma sebelumnya, hasil minimalisasi PCM yang sesuai dari fungsi nominal yang didefinisikan. telah diusulkan juga alternatif skema PCM [Theo 09, Bagian 14,4].
- PCM sensitif terhadap nilai-nilai θ_j awal dan perkiraan η_j. Salah satu cara untuk memperkirakan nilai η_j, dengan asumsi bahwa X tidak mengandung banyak outlier, yaitu dengan menjalankan algoritma FCM dan, setelah, estimasi konvergensi setiap η_j sebagai rata-rata (pembobot) dari perbedaan-perbedaan antara x_i dan θ_j, terakhir dihitung dengan FCM. Kemudian, perkiraan θ_j yang dihasilkan oleh FCM yang dapat digunakan untuk menginisialisasi PCM [Theo 09, Bagian 14.4].

Latihan 7.5.10

1. Terapkan algoritma PCM pada kumpulan data X_3 yang dihasilkan dalam Contoh 7.5.1 untuk m, = 4 m = 6, dan m = 3. Gunakan q = 2 dan η_j = 4, j = 1,. . . , m, dan menginisialisasi θ_j dengan menggunakan vektor-vektor m dari X yang "paling jauh" dari satu sama lain (gunakan prosedur distant_init). Bandingkan nilai estimasi dengan θ_j yang benar dan komentari hasilnya.
2. Ulangi langkah 1, menggunakan fungsi rand_init dan *rand_data_init* MATLAB untuk inisialisasi θ_j untuk m = 4.
3. Tarik kesimpulan.

Perhatikan bahwa dalam kasus di mana jumlah sebenarnya dari cluster adalah terlalu besar (dalam kasus untuk m = 6), PCM berhasil memperkirakan empat θ_j yang sesuai dengan (benar) empat cluster yang mendasari dalam X_3 (bertepatan dari beberapa perkiraan). Dalam kasus sebaliknya, PCM berhasil memperkirakan tiga dari empat θ_j yang sejatinya. akhirnya, inisialisasi yang kurang dari PCM dapat menyebabkan hasil clustering yang kurang.

Latihan 7.5.11

Terapkan PCM pada set data X_5 dihasilkan dalam Contoh 7.5.3 untuk m = 2, q = 2. Gunakan prosedur distant_init untuk inisialisasi θ_j dan set η_j = 4, j = 1,. . . , m.

Petunjuk

Perhatikan bahwa PCM gagal untuk mengidentifikasi kedua cluster yang kecil.