Transformasi Data
Pembangkitan Fitur dan Pengurangan Dimensi

3.1 PENDAHULUAN

Dalam bab ini, akan diterangkan teknik transformasi linear dan nonlinear yang digunakan untuk menghasilkan satu set fitur dari serangkaian pengukuran atau dari satu set dari fitur awal yang dihasilkan. Tujuannya adalah untuk mendapatkan fitur baru yang menyandikan informasi klasifikasi dalam cara yang lebih kompak dibandingkan dengan fitur asli. Hal ini berarti pengurangan jumlah fitur yang dibutuhkan bagi sebuah tugas klasifikasi yang diberikan yang dikenal juga sebagai pengurangan dimensi karena dimensi ruang fitur baru sekarang berkurang. Tujuannya adalah untuk mencapai pengurangan dimensi dalam pengertian yang optimal sehingga hilangnya informasi yang pada umumnya tidak dapat dihindari setelah mengurangi jumlah asli fitur dapat dicapai sekecil mungkin.

3.2 ANALISIS KOMPONEN UTAMA (*Principle Component Analisys* = PCA)

Analisis komponen utama (*Principle Component Analisys* = PCA) adalah salah satu teknik yang paling populer untuk pengurangan dimensi. Dimulai dari set asli dari sampel l (fitur), yang membentuk unsur-unsur vektor $x \in R^l$, dengan tujuan untuk menerapkan transformasi linear dalam mendapatkan satu set sampel baru:

$$y = A^T x$$

sehingga komponen *y* berkorelasi. Dalam tahap kedua, salah satu memilih yang paling signifikan dari komponen ini. Langkah-langkahnya dapat dirangkum sebagai berikut:

1. Perkirakan kovarians matriks S. Biasanya nilai rata-rata diasumsikan nol, E [x] = 0. dalam kasus ini, kovarians dan matriks autokorelasi bertepatan, $R \equiv E[xx^T] = S$. Jika hal ini tidak terjadi, maka kurangi mean-nya. Ingat bahwa, fitur vektor N yang diberikan, $x \in R^l$, i = 1, 2,. . . , *N*, autokorelasi estimasi matriks-nya diberikan oleh:

$$R \approx \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T \qquad (3.1)$$

2. Lakukan eigendecomposition *S* dan hitung *eigen velue* / *eigenvector*, $\lambda i,\ a_i \in R^l$, $i = 0, 2, \ldots, l - 1$.
3. Aturlah *eigenvalue* dalam urutan, $\lambda 0 \geq \lambda_1 \geq \cdots \geq \lambda_{l-1}$.
4. Pilih *eigenvalue* terbesar (m). Biasanya m dipilih sehingga kesenjangan antara λ_{m-1} dan λ_m besar.
   eigenvalue $\lambda_0, \lambda_1, \ldots, \lambda_{m-1}$ yang dikenal sebagai komponen utama m.
5. Gunakan masing-masing (kolom) *eigenvector* ai, i = 0, 1, 2,. . , *m* - 1 untuk membentuk matriks transformasi
   $A = [\ a0 \quad a1 \quad a2 \quad \cdots \quad a_m{-}1\ ]$
6. Ubah setiap dimensi-*l* vektor x di dalam ruang original ke dimensi-m vektor y melalui transformasi $y = A^T x$. Dengan kata lain, elemen ke-i dari y(i) adalah proyeksi x pada a_i, ($y(i) = a_i^T x$).

Seperti yang ditunjukkan dalam [Theo '09, bagian 6.3], varians total unsur dari x, $\sum_{i=0}^{l-1} E[x^2(i)]$ (untuk mean nol), adalah sama dengan jumlah *eigenvalue* $\sum_{i=0}^{l-1} \lambda_i$. Setelah transformasi, varians dari elemen ke-i, $E[y^2(i)]$, $i = 0, 2, \ldots, l - 1$, sama dengan λ_1. Jadi, pemilihan elemen yang sesuai untuk *eigenvalue* terbesar m dengan mempertahankan varians maksimum.
Untuk menghitung komponen utama, jenis

[*eigenval, eigenvec, explain, Y, mean_vec*] = *pca_ fun*(*X, m*)

dimana

X adalah matriks $N \times l$ dengan kolom-kolom yang merupakan vektor data, *m* adalah jumlah komponen utama yang paling signifikan yang diambil ke akun,
eigenval adalah dimensi-*m* vektor kolom yang berisi *eigenvalue* terbesar m dari kovarians matriks X berurutan, *eigenvec* adalah matriks $l \times$ dimensi-*m*, dalam kolom yang berisi *eigenvector* yang sesuai ke *eigenvalue* terbesar m matriks kovarians dari X,
explain adalah dimensi-*l* vektor kolom yang dengan elemen ke-*i* adalah persentase jumlah varians yang ditahan bersama (dalam terminologi MATLAB dijelaskan oleh) komponen utama ke-*i*,
Y adalah matriks $m \times N$ berisi proyeksi titik data X ke ruang yang direntang oleh vektor m dari *eigenvec*,
mean_vec adalah vektor mean dari vektor-vektor kolom dari X.

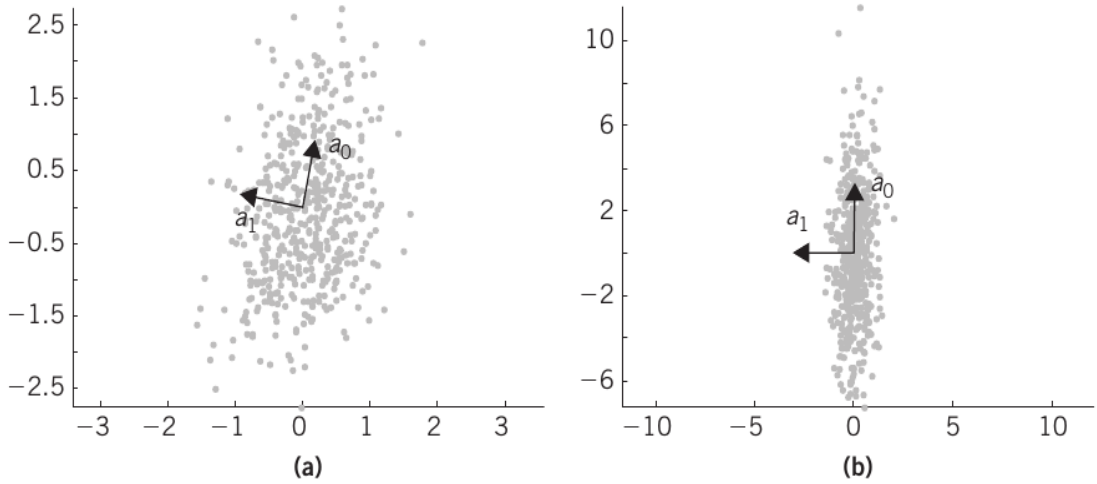
Contoh 3.2.1

1. Bangkitkan sebuah set X_1 dari $N = 500$ vektor dimensi-2 dari sebuah distribusi Gaussian dengan mean nol dan matrikx kovarians

$$S_1 = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 1.0 \end{bmatrix}$$

Lakukan PCA pada X_1, yaitu dengan menghitung dua *eigenvalue* / *eigenvector* dari estimasi $\hat{S}_1$ dari S_1 yang diperoleh dengan menggunakan vektor X_1. Dengan mempertimbangkan bahwa *eigenvalue* ke-i "menjelaskan" varians sepanjang arah *eigenvector* ke-i dari S_1 , hitung persentase dari jumlah varians yang dijelaskan oleh masing-masing dari dua komponen dengan rasio $\frac{\lambda_i}{\lambda_0+\lambda_1}$, $i = 0,1$. Plot X_1 sebagaimana *eigenvector* dari $\hat{S}_1$. Berikan komentar pada hasilnya.

2. Bangkitkan sebuah set data X_2 dengan matriks kovarians $S_2 = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 9.0 \end{bmatrix}$. Ulangi seperti langkah sebelumnya.



Gambar 3.1
Titik-titik data X_1 (a) dan X_2 (b) pada contoh 3.2.1 dengan *eigenvector* $\hat{S}_1$ dan $\hat{S}_2$

Solusi. Gunakan langkah-langkah berikut ini:
Langkah 1. membangkitkan set data X_1 tuliskan:

```
randn('seed',0)
%For reproducubility of the results
S1=[.3 .2; .2 1];
[l,l]=size(S1);
mv=zeros(1,l);
N=500;
m=2;
X1=mvnrnd(mv,S1,N)';
```

untuk mengaplikasikan PCA pada X_1 bersama dengan *eigenvector* $\hat{S}_1$, tuliskan (lihat Gambar 3.1 (a))

```
figure(1), hold on
figure(1), plot(X1(1,:),X1(2,:),'r.')
figure(1), axis equal
figure(1), line([0; eigenvec(1,1)],[0; eigenvec(2,1)])
figure(1), line([0; eigenvec(1,2)],[0; eigenvec(2,2)])
```

Persentase dari keseluruhan varians dijelaskan oleh komponen pertama dan kedua sebesar 78,98% dan 21,02%. Ini berarti bahwa jika kita memproyeksikan titik-titik X_1 sepanjang arah pokok *eigenvector* (yang sesuai dengan *eigenvalue* terbesar S_1), kita mendapatkan 78,98% dari total varians X_1; 21,02% dari varians total terkait dengan komponen utama yang kedua akan "hilang."

Langkah 2. Untuk menghasilkan data set X_2, ulangi kode sebelumnya, di mana sekarang X_1 dan S_1 diganti oleh X_2 dan S_2. Dalam kasus ini, persentase dari keseluruhan varians dijelaskan oleh komponen yang pertama dan kedua sebesar 96,74% dan 3,26%. Ini berarti bahwa jika kita memproyeksikan titik X_2 sepanjang arah *eigenvector* yang sesuai dengan *eigenvalue* terbesar dari $\hat{S}_1$, akan didapat hampir semua varians dari X_2 di ruang 2-dimensi (lihat Gambar 3.1 (b)). Jelaskan ini menggunakan penalaran fisik.

Tujuan dari contoh berikutnya adalah untuk menunjukkan bahwa memproyeksikan dalam ruang dimensi yang lebih rendah, sehingga untuk mempertahankan sebagian besar varians, tidak selalu menjamin bahwa informasi klasifikasi terkait dipertahankan.

Contoh 3.2.2

1. a. Bangkitkan satu set data X_1 yang terdiri dari 400 vektor dimensi-2 yang berasal dari dua kelas. 200 batang yang pertama berasal dari kelas pertama yang dimodelkan dengan distribusi Gaussian dengan mean $m_1 = [-8, 8]^T$; batang sisanya dari kelas kedua, dimodelkan dengan distribusi Gaussian dengan mean $m_2 = [8, 8]^T$. Kedua distribusi berbagi matriks kovarians

$$S = \begin{bmatrix} 0.3 & 1.5 \\ 1.5 & 9.0 \end{bmatrix}$$

   b. Lakukan PCA pada X_1 dan hitung persentase dari jumlah total varians yang dijelaskan oleh masing-masing komponen.
   c. Proyeksikan vektor X_1 sepanjang arah dari komponen utama yang pertama dan plot set data X_1 dan proyeksinya untuk komponen utama pertama. Berikan komentar pada hasilnya.

2. Ulangi pada data set X_2, yang dihasilkan sebagai X_1 tapi dengan $m_1 = [-1, 0]^T$ dan $m_2 = [1, 0]^T$.
3. Bandingkan hasil yang diperoleh dan tarik kesimpulannya.

Solusi. Ambil langkah-langkah berikut:

Langkah 1 (a). Untuk menghasilkan kumpulan data X_1 dan vektor y_1, yang dengan koordinasi ke-i berisi label kelas dari vektor ke-i dari X_1, tuliskan

```
randn('seed',0)
%For reproducibility of the results
S=[.3 1.5; 1.5 9];
[l,l]=size(S);
mv=[-8 8; 8 8]';
N=200;
X1=[mvnrnd(mv(:,1),S,N); mvnrnd(mv(:,2),S,N)]';
y1=[ones(1,N), 2*ones(1,N)];
```

Langkah 1 (b). Untuk menghitung *eigenvalue* / *eigenvector* dan persentase varians yang diperlukan dalam langkah ini, ketik

```
m=2;
eigenval,eigenvec,explained,Y,mean_vec]=pca_fun(X1,m);
```

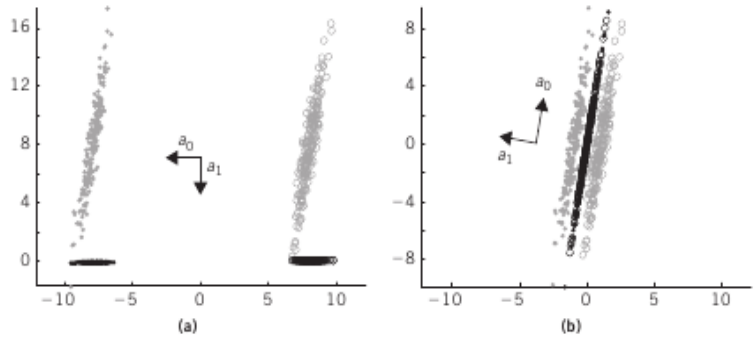
Langkah 1 (c). Proyeksi dari titik data X_1 sepanjang arah dari komponen utama pertama yang terkandung pada baris pertama Y, dikembalikan oleh fungsi *pca_ fun*. Untuk plot data vektor X_1 sebagaimana proyeksinya, ketikkan

```
%Plot of X1
figure(1), hold on
figure(1),
   plot(X(1,y==1),X(2,y==1),'r.',X(1,y==2),X(2,y==2),'bo')
%Computation of the projections of X1
w=eigenvec(:,1);
t1=w'*X(:,y==1);
t2=w'*X(:,y==2);
X_proj1=[t1;t1].*((w/(w'*w))*ones(1,length(t1)));
X_proj2=[t2;t2].*((w/(w'*w))*ones(1,length(t2)));
%Plot of the projections
figure(1), plot(X_proj1(1,:),X_proj1(2,:),'k.',...
X_proj2(1,:),X_proj2(2,:),'ko')
figure(1), axis equal
%Plot of the eigenvectors
figure(1), line([0; eigenvec(1,1)], [0; eigenvec(2,1)])
figure(1), line([0; eigenvec(1,2)], [0; eigenvec(2,2)])
```

Langkah 2 (a). Untuk menghasilkan X_2, kode untuk X_1 dijalankan lagi, dengan $m_1 = [-1, 0]^T$ dan $m_2 = [1, 0]^T$.

Langkah 2 (b) - (c). Kode dalam tahap (b) - (c) langkah 1 dieksekusi untuk X_2. Persentase jumlah total varians yang dijelaskan oleh dua komponen utama adalah 90,19% dan 9,81%.

Langkah 3. Dalam dua kasus sebelumnya, sekitar 90% dari total varians dari set data disimpan setelah memproyeksikan sepanjang komponen utama pertama. Namun, tidak ada jaminan bahwa kelas diskriminasi dipertahankan dalam arah ini. Memang, dalam kasus X_1 data dalam dua kelas,
setelah proyeksi sepanjang *eigenvector* utama pertama, tetap dipisahkan. Namun, ini bukan kasus untuk kumpulan data X_2 (lihat Gambar 3.2).



Gambar 3.2
Titik data (abu-abu) dari X_1 (a) dan X_2 (b) yang digunakan dalam Contoh 3.2.2, (normalisasi)

eigenvector dari S_1 dan S_2. Titik data dari dua kelas X_1, setelah proyeksi sepanjang *eigenvector* utama yang pertama (hitam), tetap dipisahkan. Hal ini tidak terjadi untuk X_2.

Latihan 3.2.1
Ambil langkah-langkah berikut:

1. Bangkitkan sebuah kumpulan data X_1 yang terdiri dari 400 vektor 3-dimensi yang berasal dari dua kelas. Bagian setengah pertama yang berasal dari kelas pertama, yang dimodelkan dengan distribusi Gaussian dengan mean $m_1 = [-6, 6, 6]^T$; sisanya dari kelas kedua, dimodelkan dengan distribusi Gaussian dengan mean $m_2 = [6, 6, 6]^T$. Kedua distribusi berbagi matriks kovarians

$$S = \begin{bmatrix} 0.3 & 1.0 & 1.0 \\ 1.0 & 9.0 & 1.0 \\ 1.0 & 1.0 & 9.0 \end{bmatrix}$$

   Lakukan PCA pada X_1 dan hitung persentase dari jumlah total varians yang dijelaskan oleh masing-masing komponen utama. Proyeksikan vektor X_1 pada ruang yang direntang oleh dua komponen utama yang pertama Y_1 dan Y_2. Plot data dalam subruang X_{11} - X_{12}, X_{11} -X_{13}, X_{12} -X_{13}, Y_1 -Y_2, Y_1 -Y_3, Y_2 -Y_3 (enam gambar MATLAB secara total).
2. Bangkitkan sebuah set data X_2 seperti pada langkah 1, sekarang dengan $m_1 = [-2, 0, 0]^T$ dan $m_2 = [2, 0, 0]^T$. Ulangi proses tersebut seperti dijelaskan pada langkah 1.
3. Bandingkan hasil yang diperoleh dari masing-masing kumpulan data dan tarik kesimpulan.

3.3 METODE DEKOMPOSISI NILAI TUNGGAL

Diberikan sebuah matriks (X) $l \times N$, terdapat matriks persegi unitary U dan V dari dimensi $l \times l$ dan $N \times N$, sehingga

$$X = U \begin{bmatrix} \Lambda^{\frac{1}{2}} & O \\ O & 0 \end{bmatrix} V^T$$

dimana Λ adalah matriks persegi $r \times r$, dengan $r \leq \min\{l, N\}$ (r adalah sama dengan peringkat dari X). Karena matriks U dan V adalah kesatuan (*unitary*), vektor kolomnya didefinisikan, ortonormal dan $UU^T = I$ dan $VV^T = I$. Matriks $\Lambda^{1/2}$ diberikan oleh

$$\Lambda^{\frac{1}{2}} = \begin{bmatrix} \sqrt{\lambda_0} & & & \\ & \sqrt{\lambda_1} & & \\ & & \ddots & \\ & & & \sqrt{\lambda_{r-1}} \end{bmatrix}$$

mana λi, i = 0, 2,. . . , r – 1, adalah *eigenvalue r nonzero* dari XX^T, yang sama dengan *eigenvalue* dari X^TX [Theo 09, Bagian 6.4], dan dikenal sebagai nilai-nilai singular dari X. Demikian pula dapat ditulis

$$X = \sum_{i=0}^{r-1} \sqrt{\lambda_i} u_i v_i^T \qquad (3.2)$$

dimana *ui, vi, i* = 0, 1,. . . , *r* – 1, adalah *eigenvector* yang sesuai XX^T dan X^TX.

Selain itu, *ui* dan *vi, i* = 0,. . . , *r* – 1 adalah vektor-vektor kolom pertama r dari *U* dan *V*. Sisa vektor-vektor kolom dari *U* dan *V* sama dengan *eigenvalue* nol.

Jika digunakan $m \leq r$ dalam penjumlahan persamaan. (3.2), yaitu,

$$\hat{X} = \sum_{i=0}^{m-1} \sqrt{\lambda_i} u_i v_i^T \tag{3.3}$$

maka $\hat{X}$ adalah pendekatan terbaik (dalam arti Frobenius) dari *X* rangking *m* [, Theo 09 Bagian 6.4].

Untuk menghitung Dekomposisi Nilai Singular (Singular Value Decomposition = SVD) dari matriks *X*, ketikkan

$$[U, s, V, Y] = svd\_fun\ (X, m)$$

dimana

X adalah matriks $l \times N$ yang kolom-kolomnya berisi vektor data,
m adalah jumlah nilai-nilai singular terbesar yang akan dihitung,
U adalah suatu matriks $l \times l$ *eigenvector* yang memuat XX^T dalam urutan,
s adalah vektor *r*-dimensi yang mengandung nilai-nilai singular dalam urutan,
V adalah matriks $N \times N$ yang berisi *eigenvector* dari X^TX dalam urutan,
Y adalah matriks $m \times N$ berisi proyeksi titik data *X* pada ruang yang direntang oleh *m* yang berisi *eigenvector* yang terkandung dalam *U*.

Lebih lanjut tentang SVD dapat ditemukan di [Theo 09, Bagian 6.4].

Latihan 3.3.1

1. Gunakan set data X_1 dari Latihan 3.2.1. Lakukan dekomposisi nilai singular menggunakan *svd_fun*. Lalu proyeksikan vektor X_1 pada ruang yang direntang oleh *m* yang berisi *eigenvector* yang terkandung dalam *U* (yang sesuai dengan Y_1 dan Y_2). Akhirnya, plot data dalam ruang X_{11} - X_{12}, X_{11} -X_{13}, X_{12} -X_{13}, Y_1 -Y_2, Y_1 -Y_3, Y_2 -Y_3 (enam gambar MATLAB secara total).
2. Ulangi untuk kumpulan data X_2 dari Latihan 3.2.1 dan bandingkan hasilnya. Perhatikan bahwa hasil yang diperoleh untuk kasus SVD mirip dengan yang diperoleh pada Latihan 3.2.1 untuk kasus PCA (mengapa?).

Contoh 3.3.1. Hasilkan sebuah set data dari $N = 100$ vektor dimensi $l = 2000$. Vektor-vektor batang dari distribusi Gaussian dengan sebuah rata-rata (mean) sama dengan vektor nol *l*-dimensi dan suatu diagonal kovarians matriks S yang memiliki semua elemen *nonzero* yang sama dengan 0,1 kecuali *S* (1, 1) dan *S* (2, 2), yang sama dengan 10.000. Terapkan PCA dan SVD pada set data

sebelumnya dan tarik kesimpulan Anda.

***Solusi*.** Untuk menghasilkan matriks *X*, yang berisi vektor dari himpunan data, ketikkan

```
N=100;
l=2000;
mv=zeros(1,l);
S=0.1*eye(l);
S(1,1)=10000;
S(2,2)=10000;
randn('seed',0)
X=mvnrnd(mv,S,N)';
```

Perhatikan bahwa data menunjukkan penyebaran yang signifikan sepanjang dua sumbu pertama, yaitu, di sepanjang vektor

$$e_1 = [1, \overbrace{0, \ldots, 0}^{l-1}]^T \text{ and } e_2 = [0, 1, \overbrace{0, \ldots, 0}^{l-2}]^T.$$

Untuk menjalankan PCA dan SVD pada *X* dan mengukur waktu eksekusi dari setiap metode, ketikkan

```
%PCA
t0=clock;
m=5;
[eigenval,eigenvec,explain,Y]=pca_fun(X,m);
time1=etime(clock,t0)
'____'
%SVD
t0=clock;
m=min(N,l);
[U,S,V,Y]=svd_fun(X,m);
time2=etime(clock,t0)
```

Dari hasil yang diperoleh sebelumnya, ada dua kesimpulan yang dapat ditarik.
Pertama, kedua metode mengidentifikasi (sekitar) e_1 dan e_2 sebagai petunjuk paling signifikan. untuk memverifikasikannya, bandingkan dua kolom pertama dari eigenvec yang dihasilkan oleh PCA dengan dua kolom pertama dari *U*, yaitu *U* (:, 1: 2), yang dihasilkan oleh SVD, dengan mengetikkan

```
[eigenvec (:, 1:2) U (:, 1:2)] '
```

Kedua, asalkan memori komputer yang cukup tersedia, PCA akan mengambil perintah besarnya lebih banyak waktu dari SVD (jika memori tidak cukup tersedia, PCA tidak akan berjalan sama sekali). Perbedaan kinerja dari dua metode terletak pada kenyataan bahwa dalam PCA dilakukan *eigendecomposition* pada kovarians matriks $l \times l$ sedangkan pada SVD dilakukan *eigendecomposition* pada $N \times NX^TX$ dan kemudian dengan transformasi sederhana, menghitung *eigenvector* dari XX^T (yang dapat dipandang sebagai skala pendekatan dari matriks autokorelasi). Selain itu, harus ditekankan bahwa secara umum untuk kasus-kasus dimana $N < l$,

estimasi yang diperoleh dari matriks autokorelasi adalah tidak bagus.
Contoh kasus dimana $N < l$, muncul dalam aplikasi pengolahan citra, di *web mining*, dalam analisis *microarray*, dan sejenisnya.

3.4 FISHER'S LINEAR DISCRIMINANT ANALYSIS

Pada PCA, pengurangan dimensi dilakukan dalam modus tanpa pengawasan. Vektor fitur yang diproyeksikan pada subruang yang direntang oleh *eigenvector* dominan dari matriks kovarians (autokorelasi). Pada bagian ini, perhitungan subruang di mana satu proyek, dalam rangka untuk mengurangi dimensi berlangsung dalam mode diawasi. Subruang ini juga ditentukan melalui solusi dari sebuah masalah *eigendecomposition*, namun matriks yang sesuai adalah berbeda.

Dalam kasus kelas 2, tujuannya adalah untuk mencari satu arah (w) sehingga proyeksi masing-masing *y* dari *l*-dimensi vektor fitur $x \in R^l$ memaksimalkan rasio diskriminan Fisher. Rasio diskriminan Fisher dari sebuah fitur skalar *y* dalam tugas klasifikasi 2-kelas didefinisikan :

$$FDR = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

dimana μ_1, μ_2 adalah nilai rata-rata dari *y*, dan σ_1^2, σ_2^2 adalah varians dari y dalam dua kelas.

Dengan kata lain, setelah proyeksi pada w bertujuan untuk nilai rata-rata titik data dalam dua kelas akan terpisah sejauh mungkin dan untuk varians untuk menjadi sekecil mungkin. Ternyata bahwa w diberikan oleh *eigenvector* maksimum dari hasil matriks [Theo 09, Bagian 5,8], dimana untuk dua kelas *equiprobable* dikenal sebagai matriks *within-class scatter* (pencar dalam kelas), dengan S_1, S_2 menjadi matriks kovarian. S_b dikenal sebagai matriks *between-class scatter (*pencar antara kelas), didefinisikan oleh

$$S_b = \frac{1}{2}(m_1 - m_0)(m_1 - m_0)^T + \frac{1}{2}(m_2 - m_0)(m_2 - m_0)^T$$

dimana m_0 adalah mean keseluruhan data x dalam ruang R^l asli dan m_1, m_2 adalah nilai rata-rata dalam dua kelas [Theo 09, Bagian 5.6.3]. Hal ini dapat ditunjukkan bahwa dalam hal ini kasus khusus kelas 2, langkah *eigenanalysis* dapat dilewati dan solusi langsung diberikan oleh

$$w = S_w^{-1}(m_1 - m_2)$$

Dalam kasus kelas-c, tujuannya adalah untuk menemukan arah $m \leq c$ - 1 (subruang m-dimensi) yang disebut sebagai kriteria J_3, didefinisikan sebagai

$$J_3 = trace[S_w^{-1} S_b]$$

dimaksimalkan. Pada persamaan sebelumnya

$$S_w = \sum_{i=1}^{c} P_i S_i,\ S_b = \sum_{i=1}^{c} P_i (m_i - m_0)(m_i - m_0)^T$$

dan P_i 's melambangkan kelas sebuah probabilitas apriori. Ini merupakan generalisasi dari kriteria kriteria FDR dalam kasus yang multiclass dengan probabilitas apriori yang berbeda. Arah *m* diberikan oleh *eigenvector* dominan *m* dari hasil matriks $S_w^{-1}S_b$. Ini harus menunjukkan bahwa ranking dari matriks S_b adalah *c* - 1 pada bagian terbesar (meskipun diberikan sebagai jumlah matriks *c*, hanya *c* - 1 dari istilah ini yang independen [Theo 09, Bagian 5.6.3]. Ini adalah alasan dimana *m* atas dibatasi oleh *c* - 1; hanya *c* - 1 *eigenvalue* terbesar (paling banyak) adalah *nonzero*. Pada beberapa kasus, hal ini mungkin merupakan sebuah kelemahan karena jumlah maksimum fitur bahwa metode ini dapat menghasilkan dibatasi oleh jumlah kelas [Theo 09, Bagian 5.8].

Contoh 3.4.1

1. Terapkan linear discriminant analisys (LDA) pada kumpulan data X_2 yang dihasilkan dari bagian kedua dari Contoh 3.2.2.
2. Bandingkan hasil yang diperoleh dengan yang diperoleh dari analisis PCA.

Solusi. Gunakan langkah-langkah berikut:

Langkah 1. Untuk memperkirakan vektor rata-rata setiap kelas menggunakan sampel yang tersedia, ketikkan

```
mv_est(:,1)=mean(X2(:,y2==1)')';
mv_est(:,2)=mean(X2(:,y2==2)')';
```

Untuk menghitung matriks-pencar dalam S_w, gunakan fungsi scatter_mat, yang menghitung di dalam kelas (S_w), antara kelas (S_b), dan kelas campuran (S_m) [Theo 09, Bagian 5.6.3] untuk masalah klasifikasi c-kelas berdasarkan satu set vektor data. Fungsi ini disebut dengan

```
[Sw,Sb,Sm]=scatter_mat(X2,y2);
```

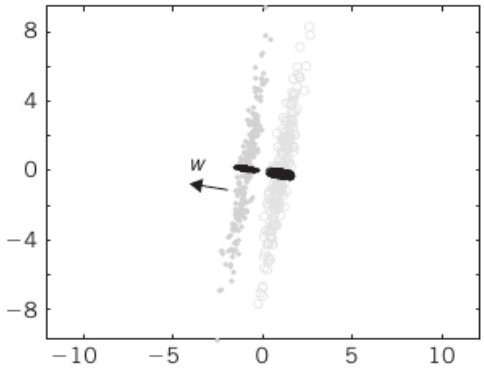
Karena dua kelas *equiprobable*, arah *w* sepanjang yang rasio diskriminan Fisher dimaksimal-kan dan dihitung sebagai $w = S_w^{-1}[m_1 - m_2]$. Dalam hal ini MATLAB ditulis sebagai berikut

```
w=inv(Sw)*(mv_est(:,1)-mv_est(:,2))
```

Akhirnya, proyeksi dari vektor data X_2 pada arah *w* sebagaimana plot hasil dilakukan melalui pernyataan berikut (lihat Gambar 3.3)

```
%Plot of the data set
figure(1), plot(X(1,y==1),X(2,y==1),'r.',...
X(1,y==2),X(2,y==2),'bo')
figure(1), axis equal
%Computation of the projections
t1=w'*X(:,y==1);
t2=w'*X(:,y==2);
X_proj1=[t1;t1].*((w/(w'*w))*ones(1,length(t1)));
X_proj2=[t2;t2].*((w/(w'*w))*ones(1,length(t2)));
%Plot of the projections
```

```
figure(1), hold on
figure(1), plot(X_proj(1,y==1),X_proj(2,y==1),'y.',...
X_proj(1,y==2),X_proj(2,y==2),'co')
```



Gambar 3.3
Titik-titik set data X_2 (abu-abu) dan proyeksinya (hitam) sepanjang arah *w*, dari Contoh 3.4.1

Langkah 2. Bandingkan hasil yang dalam MATLAB gambar 1, yang dihasilkan oleh eksekusi dari kode sebelumnya, dengan hasil yang sesuai diperoleh dengan analisis PCA. Hal ini telah diamati bahwa kelas-kelas tetap terpisah ketika vektor X_2 diproyeksikan sepanjang arah w dimana hasilnya dari analisis diskriminan Fisher. Sebaliknya, kelas yang sangat tumpang tindih ketika diproyeksikan sepanjang arah utama yang disediakan oleh PCA.

Contoh 3.4.2
1a. Bangkitkan sebuah set data 900 vektor data 3-dimensi yang berasal dari dua kelas – 100 vektor yang pertama dari distribusi Gaussian *zero-mean* dengan matriks kovarians

$$S_1 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

Sisanya dikelompokkan dalam 8 kelompok 100 vektor. Setiap kelompok berasal dari sebuah distribusi Gaussian. Semua distribusi ini berbagi matriks kovarians

$$S_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

dengan nilai mean-nya adalah

- $m_1^2 = [a, 0, 0]^T$
- $m_2^2 = [a/2, a/2, 0]^T$
- $m_3^2 = [0, a, 0]^T$
- $m_4^2 = [-a/2, a/2, 0]^T$
- $m_5^2 = [-a, 0, 0]^T$
- $m_6^2 = [-a/2, -a/2, 0]^T$

- $m_7^2 = [0, -a, 0]^T$
- $m_8^2 = [a/2, -a/2, 0]^T$

dimana a = 6 (m_i^2 menunjukkan rata-rata dari distribusi Gaussian ke-i dari kelas kedua). Ambil langkah-langkah berikut:

1b. Atur set data 3-dimensi dan lihat dari sudut yang berbeda untuk mendapatkan gambaran tentang bagaimana data tersebar dalam ruang 3-dimensi (menggunakan utilitas Rotate-3D MATLAB).

1c. Lakukan analisis diskriminan Fisher pada set data sebelumnya. Proyeksikan data pada subruang yang direntang oleh *eigenvector* yang sesuai dengan *nonzero eigenvalue* dari hasil matriks $S_w^{-1} S_b$. Berikan komentar pada hasilnya.

2. Ulangi langkah 1 untuk masalah 3-kelas dimana data yang dihasilkan seperti pada langkah 1, dengan pengecualian bahwa kelompok terakhir dari 100 vektor, yang berasal dari distribusi Gaussian dengan mean m_8^2 diberi label kelas 3.

Solusi. Ikuti langkah-langkah berikut ini:

Langkah 1(a). Untuk membangkitkan matriks 3 × 900-dimensi yang kolom-kolomnya adalah vektor-vektor data, ketikkan

```
%Initialization of random number generator
randn('seed',10)
%Definition of the parameters
S1=[.5 0 0; 0 .5 0; 0 0 .01];
S2=[1 0 0; 0 1 0; 0 0 .01];
a=6;
mv=[0 0 0; a 0 0; a/2 a/2 0; 0 a 0; -a/2 a/2 0;...
-a 0 0; -a/2 -a/2 0; 0 -a 0; a/2 -a/2 0]';
N=100;
% Generation of the data set
X=[mvnrnd(mv(:,1),S1,N)];
for i=2:9
X=[X; mvnrnd(mv(:,i),S2,N)];
end
X=X';
c=2; %No of classes
y=[ones(1,N) 2*ones(1,8*N)]; %Class label vector
```

Langkah 1 (b). Untuk mengatur (plot) kumpulan data *X* dalam ruang 3-dimensi, ketikkan

```
figure(1), plot3(X(1,y==1),X(2,y==1),X(3,y==1),'r.',...
X(1,y==2),X(2,y==2),X(3,y==2),'b.')
figure(1), axis equal
```

Dengan tombol Putar-3D dari MATLAB gambar 1, dapat dilihat set data dari sudut yang berbeda. Sangat mudah untuk melihat bahwa variasi data sepanjang arah ketiga adalah sangat kecil (karena kecil nilai S_1 (3, 3) dan S_2 (3, 3)). Kumpulan data dalam ruang 3-dimensi dapat dianggap sebagai terletak pada bidang x - y dengan variasi yang sangat kecil sepanjang sumbu z. Dengan demikian, proyeksi dari kumpulan data pada bidang x - y mempertahankan pemisahan kelas, tapi ini tidak terjadi dengan proyeksi pada bidang x - z dan y - z. Selain itu, amati bahwa tidak ada satu arah (ruang 1-dimensi) *w* yang mempertahankan pemisahan kelas setelah memproyeksikan *X* di atasnya.

Langkah 1 (c). Untuk melakukan analisis diskriminan Fisher, pertama hitung matriks pencar S_w dan S_b, kemudian lakukan *eigendecomposition* pada matriks $S_w^{-1}S_b$; terakhir, proyeksikan data pada subruang yang direntang oleh *eigenvector* dari $S_w^{-1}S_b$ yang sesuai dengan *nonzero eigenvalue*. Berikut ini kode MATLAB yang dapat digunakan:

```
% Scatter matrix computation
[Sw,Sb,Sm]=scatter_mat(X,y);
% Eigendecomposition of Swˆ (-1)*Sb
[V,D]=eig(inv(Sw)*Sb);
% Sorting the eigenvalues in descending order
% and rearranging accordingly the eigenvectors
s=diag(D);
[s,ind]=sort(s,1,'descend');
V=V(:,ind);
% Selecting in A the eigenvectors corresponding
% to non-zero eigenvalues
A=V(:,1:c-1);
% Project the data set on the space spanned by
% the column vectors of A
Y=A'*X;
```

Di sini digunakan kode untuk kasus multikelas dengan $c = 2$. Karena jumlah kelas adalah sama dengan 2, hanya satu *eigenvalue* dari $S_w^{-1}S_b$ adalah *nonzero* (0,000234). Dengan demikian, analisis diskriminan Fisher memberi satu arah (ruang 1-dimensi) sepanjang data yang akan diproyeksikan. Untuk mengatur (plot) proyeksi *X* pada subruang yang direntang oleh *eigenvector* dari $S_w^{-1}S_b$ yang sesuai dengan *nonzero eigenvalue*, ketikkan

```
figure(2), plot(Y(y==1),0,'ro',Y(y==2),0,'b.')
figure(2), axis equal
```

Amati bahwa proyeksi data dari dua kelas secara bersamaan. Jadi, dalam hal ini analisis diskriminan Fisher tidak dapat memberikan suatu subruang yang lebih kecil dimana diskriminasi kelas dipertahankan. Hal ini terjadi karena jumlah kelas adalah sama dengan 2 sehingga mengurangi dimensi dari subruang ini menjadi 1tidak mencukupi untuk masalah saat ini.

Langkah 2 (a). Untuk menghasilkan matriks 3 × 900-dimensi yang kolom-kolomnya adalah vektor-vektor data, ulangi kode yang diberikan pada langkah 1 (a), gantikan dua baris terakhir dengan

```
% Definition of the number of classes
c=3;
% Definition of the class label of each vector
y=[ones(1,N) 2*ones(1,7*N) 3*ones(1,N)];
```

Langkah 2 (b). Untuk plot kumpulan data *X* dalam ruang 3-dimensi, ketikkan

```
figure(1),
plot3(X(1,y==1),X(2,y==1),X(3,y==1),'r.',X(1,y==2),...
X(2,y==2),X(3,y==2),'b.',X(1,y==3),X(2,y==3),X(3,y==3),'g.')
figure(1), axis equal
```

Langkah 2 (c). Mengadopsi kode MATLAB dari langkah 1 (c) untuk set data saat ini. Dalam hal ini, karena ada tiga kelas, dimungkinkan memiliki paling banyak dua *nonzero eigenvalue* dari $S_w^{-1}S_b$. Dengan demikian *nonzero eigenvalue* sekarang berubah menjadi 0.222145 dan 0,000104. Untuk plot proyeksi *X* pada ruang yang direntang oleh *eigenvector* dari $S_w^{-1}S_b$ yang sesuai ke *nonzero eigenvalue* (ruang 2-dimensi), ketikkan:

```
figure(3), plot(Y(1,y==1),Y(2,y==1),'ro',...
Y(1,y==2),Y(2,y==2),'b.',Y(1,y==3),Y(2,y==3),'gx')
figure(3), axis equal
```

Dalam hal ini, amati bahwa proyeksi dalam subruang 2-dimensi mempertahankan pemisahan antara kelas pada tingkat yang memuaskan.

Akhirnya, perlu diingat bahwa ada set data di mana pengurangan dimensi dari proyeksi dalam setiap ruang bagian dari ruang yang asli dapat menyebabkan kerugian besar pada diskriminasi kelas. Pada beberapa kasus, teknik nonlinier mungkin lebih berguna.

3.5 KERNEL PCA

Tiga metode yang telah digunakan sejauh ini untuk pengurangan dimensi adalah linear. Sebuah subruang dimensi rendah pertama dibangun sebagai rentang arah dominan *m* dalam R^l asli, ruang $l > m$.

Pilihan arah dominan tergantung pada metode yang digunakan. Dalam tahap kedua, semua vektor minat R^l adalah (secara linear) diproyeksikan dalam subruang berdimensi rendah. Seperti teknik-teknik yang sesuai, kapanpun data dalam R^l terletak (sekitar) di atas suatu *manifold* linier (misalnya, *hyperplane*). Namun pada banyak kasus, data didistribusikan sekitar *manifold* dimensi lebih rendah, yang tidak linier (misalnya, sekitar lingkaran atau bola dalam ruang 3-dimensi). Gambar 3.4 (a, b) menunjukkan dua kasus di mana data dalam ruang 2-dimensi terletak (kira-kira) di atas *manifold* linier dan nonlinier. Kedua *manifold* adalah 1-dimensi dengan garis lurus dan

keliling lingkaran dapat diparameterkan dalam parameter tunggal.

Kernel PCA adalah salah satu teknik untuk pengurangan dimensi ketika data terletak (sekitar) di atas *manifold* nonlinier. Menurut metode ini, data pertama kali dipetakan ke ruang dimensi tinggi melalui pemetaan nonlinier:

$$x \in \mathcal{R}^l \mapsto \phi(x) \in H$$

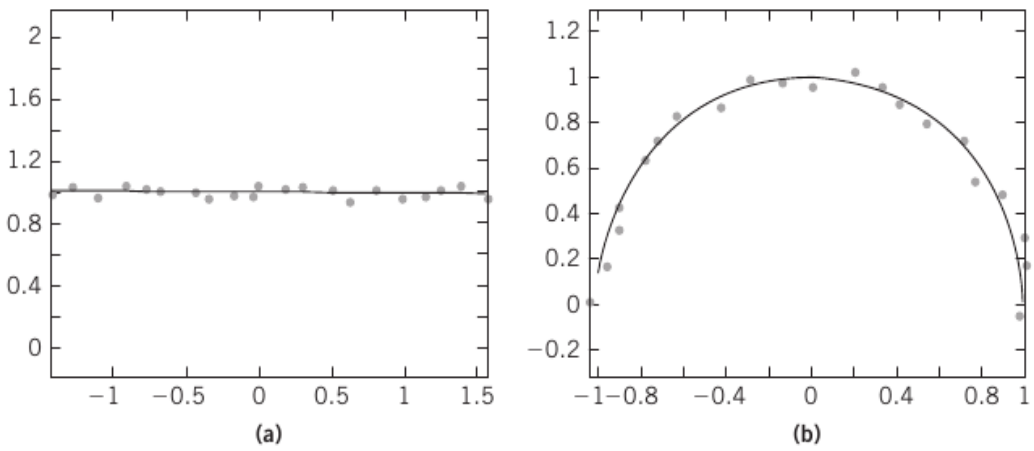
PCA ini kemudian diterrapkan di ruang baru *H*, dipilih menjadi RKHS. *Inner product* dapat dinyatakan dalam bentuk trik kernel, seperti dibahas dalam Bagian 2.5. Meskipun PCA dilakukan di RKHS ruang H, dikarenakan non-linear dari pemetaan fungsi $\varphi(\cdot)$, metode ini setara dengan fungsi nonlinier di ruang asli. Selain itu, karena setiap operasi dapat dinyatakan dalam *inner product* , pengetahuan eksplisit $\varphi(\cdot)$ tidak diperlukan. Hal ini diperlukan untuk mengadopsi fungsi kernel yang mendefinisikan *inner product*. Keterangan lebih lanjut diberikan dalam [, Theo 09 Bagian 6.7.1].

Untuk menggunakan kernel PCA, ketik

[*s, V, Y*] = *kernel_PCA*(*X, m, choice, para*)

dimana

X adalah matriks *l* × *N* yang kolom-kolomnya berisi vektor data,
m adalah jumlah komponen utama (signifikan) yang akan diperhatikan,
choice adalah jenis fungsi kernel yang akan digunakan ('*pol*' untuk polinomial, '*exp*' untuk eksponensial),
para adalah vektor 2-dimensi yang berisi parameter untuk fungsi kernel, untuk polinomial adalah $(x^T y + para(1))^{para(2)}$ dan untuk eksponensial itu adalah
$\exp(-(x-y)^T(x-y)/2para(1)^2)$,
s adalah vektor N-dimensi yang mengandung *eigenvalue* dihitung setelah menerapkan PCA kernel,
V adalah matriks *N* × *N* yang kolom-kolomnya adalah *eigenvector* yang bersesuaian dengan komponen utama dari matriks Gram, *K*, yang terlibat dalam kernel PCA [Theo 09, Bagian 6.7.1],
Y adalah matriks *m* × *N* dimensi yang berisi proyeksi dari vektor data *X* pada subruang yang direntang oleh komponen utama *m*.



GAMBAR 3.4
(a) titik-titik data 2-dimensi yang terletak (di sekitar) pada garis (berjenis linier).
(b) titik-titik data 2-dimensi yang terletak (di sekitar) pada setengah lingkaran (*manifold* nonlinier).

Contoh 3.5.1. Contoh ini menggambarkan alasan di balik kernel PCA. Namun, karena kernel PCA menyiratkan, pertama, pemetaan ke ruang dimensi yang lebih tinggi, visualisasi hasil umumnya tidak mungkin. Oleh karena itu, kita akan "menipu" sedikit dan menggunakan fungsi pemetaan $\varphi(\cdot)$ yang tidak sesuai ke kernel fungsi $k(\cdot, \cdot)$. (Selain itu, pemetaan ke RKHS diperlukan hanya untuk komputasi *tractability* yang diperlukan untuk menghitung *inner product* dengan efisien). Namun fungsi ini memungkinkan transformasi ruang 2-dimensi, di mana titik-titik data terletak di sekitar manifold nonlinier, menjadi ruang 2-dimensi lain, di mana titik data yang dipetakan di sekitar manifold linier.
Perhatikan kumpulan data X yang terdiri dari 21 titik-titik 2-dimensi yang berbentuk $xi = (xi(1), xi(2)) = (\cos \theta_i + s_i, \sin \theta_i + s'_i)$, dimana $\theta_i = (i - 1) * (\pi/20)$, i = 1,. . . , 21 dan s_i, s'_i adalah nomor acak yang berasal dari distribusi seragam pada [-0,1, 0,1] (lihat Gambar 3.5 (a)). Titik-titik terletak di sekitar setengah lingkaran yang dimodelkan oleh $x^2(1) + x^2(2) = 1$, yang berpusat di titik asal dan positif sepanjang sumbu $x2$.

Fungsi pemetaan $\varphi(\cdot)$ didefinisikan sebagai

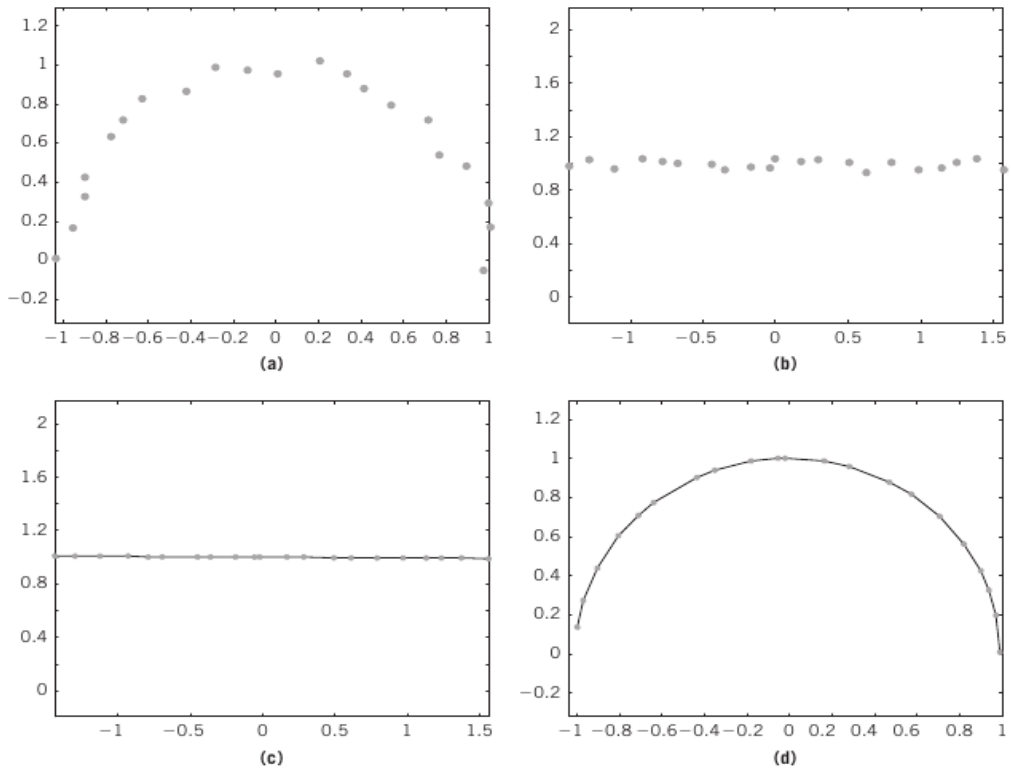
$$\phi\left(\begin{bmatrix} x(1) \\ x(2) \end{bmatrix}\right) = \begin{bmatrix} \tan^{-1}\left(\frac{x(2)}{x(1)}\right) \\ \sqrt{x^2(1) + x^2(2)} \end{bmatrix}$$

Dengan menerapkan $\varphi(\cdot)$ pada data set X, kita mendapatkan set Y = {yi = φ (xi), i = 1,. . . , 21}, yang diilustrasikan pada Gambar 3.5 (b). Perhatikan bahwa titik-titik Y yang terletak di sekitar *manifold* linier (garis lurus) di dalam domain yang terubah. Kemudian diterapkan linier PCA pada Y dan hanya mempertahankan komponen utama pertama, karena hampir semua varians total dari kumpulan data (99,87%) dipertahankan sepanjang arah ini.[1] Biarkan $Z = \{z_i = [\, z_i(1), z_i(2)]^T,$ i = 1,. . . , 21} adalah himpunan yang berisi proyeksi s yi 'pada komponen utama pertama dalam membentuk ruang (lihat Gambar 3.5 (c)). Pemetaan kembali zi 's untuk ruang asli melalui fungsi invers φ (·), yang diberikan oleh

$$\phi^{-1}\left(\begin{bmatrix} z(1) \\ z(2) \end{bmatrix}\right) = \begin{bmatrix} z(2)\cos z(1) \\ z(2)\sin z(1) \end{bmatrix} \equiv \begin{bmatrix} x'(1) \\ x'(2) \end{bmatrix}$$

titik-titik (x' (1), x' (2)) diperoleh dari yang terletak pada setengah lingkaran yang didefinisikan oleh x^2 (1) + x^2 (2) = 1, dengan x (2)> 0 (lihat Gambar 3.5 (d)).

[1] Linear PCA memerlukan pengurangan rata-rata dari vektor data (yang dilakukan dalam fungsi pca_fun). Dalam kasus ini, vektor ini sama dengan $[0, 1]^T$. Setelah PCA, vektor ini ditambahkan ke setiap proyeksi titik sepanjang arah komponen prinsip pertama.



GAMBAR 3.5
Contoh 3.5.1: (a) kumpulan data dalam ruang asli (sekitar setengah lingkaran). (b) kumpulan data dalam ruang yang diubah (sekitar garis lurus). (c) arah yang sesuai dengan komponen utama pertama dan gambar (proyeksi) dari titik-titik di atasnya dalam ruang yang diubah. (d) Gambar dari titik-titik dalam ruang asli.

- *Remark*
  Amati bahwa pemetaan nonlinier mengubah manifold asli menjadi linier. Jadi, penerapan (linier) PCA pada domain terubah adalah sepenuhnya dibenarkan. Tentu saja, umumnya kasus seharusnya tidak berharap untuk menjadi begitu "beruntung" - yaitu untuk memiliki data terubah terletak di manifold linear.

Pada contoh berikutnya, kernel PCA dalam konteks tugas klasifikasi. Lebih khusus, tujuannya adalah untuk menunjukkan potensi kernel PCA untuk mengubah masalah klasifikasi nonlinier, dalam ruang (asli) *l*-dimensi, menjadi linear dalam ruang dimensi *m* (< *l*). Jika ini tercapai, masalah klasifikasi asli dapat diselesaikan dalam ruang terubah menggunakan pengklasifikasi linear.

Contoh 3.5.2

1. Bangkitkan dua set data X dan X_{test}, masing-masing berisi 200 vektor 3-dimensi. Pada setiap vektor yang pertama $N_1 = 100$ berasal dari kelas 1, yang dimodelkan dengan distribusi seragam dalam $[-0{,}5, 0{,}5]^3$, sedangkan sisanya, $N_2 = 100$, berasal dari kelas -1 dan terletak di sekitar *sphere* dengan jari-jari r = 2 dan berpusat pada titik asal. Titik-titik N_2 untuk setiap kumpulan data yang dihasilkan sebagai berikut.
   Secara acak memilih sepasang angka, *x* (1) dan *x* (2), yang berasal dari distribusi seragam dalam rentang [-2, 2], dan periksa apakah $x^2(1) + x^2(2)$ kurang dari r^2. Jika hal ini tidak terjadi, pilih pasangan yang berbeda. Jika tidak, hasilkan dua poin dari *sphere* sebagai ($x(1), x(2), \sqrt{r^2 - x^2(1) - x^2(2)} + \varepsilon_1$) dan ($x(1), x(2), -\sqrt{r^2 - x^2(1) - x^2(2)} + \varepsilon_2$), dimana ε_1 dan ε_2 adalah nomor acak yang berasal dari distribusi seragam pada interval [-0,1, 0,1]. Ulangi prosedur ini $N_2/2$ kali untuk menghasilkan titik N_2. Selain itu bangkitkan juga vektor *y* dan y_{test} yang mengandung label kelas dari titik *X* dan X_{test}. Kemudian plot set datanya.

2. Lakukan kernel PCA pada *X* menggunakan fungsi kernel eksponensial dengan $\sigma = 1$ dan pertahankan hanya dua yang pertama dari komponen utama yang paling signifikan. Proyeksi titik data dari *X* ke subruang yang direntang oleh dua komponen utama dan biarkan *Y* sebagai set proyeksi ini (plot *Y*).

3. Desain sebuah pengklasifikasi *least square* (LS) berdasarkan *Y*.

4. Evaluasi kinerja dari classifier sebelumnya berdasarkan X_{test} sebagai berikut: Untuk setiap vektor pada $x \in X_{test}$, menentukan proyeksi ke ruang yang direntang oleh dua komponen utama yang paling signifikan, dihitung terlebih dulu, dan klasifikasikan menggunakan *classifier* LS yang dihasilkan pada langkah 3. Tetapkan *x* ke kelas di mana proyeksi telah ditetapkan. Plot proyeksi titik X_{test} ke subruang yang direntang oleh dua komponen utama bersama dengan garis lurus yang menggunakan *classifier*.

5. Ulangi langkah 2 sampai 4 dengan $\sigma = 0{,}6$.

Solusi. Gunakan langkah-langkah berikut ini:

Langkah 1. Untuk menghasilkan titik-titik *X* milik kelas 1, ketikkan

```
rand('seed',0)
noise_level=0.1;
n_points=[100 100];
%Points per class
```

```
l=3;
X=rand(l,n_points(1))- (0.5*ones(l,1))*ones(1,n_points(1));
```

Untuk menghasilkan titik-titik *X* yang milik kelas -1, ketik

```
r=2; %Radius of the sphere
for i=1:n_points(2)/2
e=1;
while(e==1)
temp=(2*r)*rand(1,l-1)-r;
if(r^ 2-sum(temp.^ 2)>0)
e=0;
end
end
t=sqrt(r^ 2-sum(temp.^ 2))+noise_level*(rand-0.5);
qw=[temp t; temp -t]';
X=[X qw];
end
```

Set data X_{test} yang dihasilkan mirip (menggunakan nilai 100 sebagai bakal untuk fungsi *rand*). Untuk menentukan label kelas dari vektor data, ketikkan

```
[l,N]=size(X);
y=[ones(1,n_points(1)) -ones(1,n_points(2))];
y_test=[ones(1,n_points(1)) -ones(1,n_points(2))];
```

untuk plot set data *X*, ketik (gunakan tombol putar 3D untuk mengamati set data dari sudut yang berbeda).

```
figure(1), plot3(X(1,y==1),X(2,y==1),X(3,y==1),'r.',...
X(1,y==-1),X(2,y==-1),X(3,y==-1),'b+')
figure(1), axis equal
```

X_{test} diplot sama. Jelas, dua kelas nonlinear dipisahkan.

Langkah 2. Untuk melakukan kernel PCA dengan kernel eksponensial dan $\sigma = 1$, ketikkan

```
[s,V,Y]=kernel_PCA(X,2,'exp',[1 0]);
```

Perhatikan bahwa *Y* berisi di dalam kolomnya berupa gambar dari titik *X* pada ruang yang direntang oleh dua komponen utama yang pertama, sementara *V* berisi masing-masing komponen utama.
Untuk plot *Y*, ketikkan

```
figure(2), plot(Y(1,y==1),Y(2,y==1),'r.',Y(1,y==-1),Y(2,y==-
1),'b+')
```

Langkah 3. Untuk desain *classifier* LS berdasarkan *Y*, ketikkan

```
w=SSErr([Y; ones(1,sum(n_points))],y,0);
```

Perhatikan bahwa setiap vektor kolom dari Y telah ditambah dengan 1. W dihasilkan [33,8001, 2,4356,
-0,8935].

Langkah 4. Ketik berikut ini untuk menghasilkan set Y_{test}, yang berisi dalam kolomnya proyeksi dari vektor X_{test} ke ruang yang direntang oleh komponen utama:

```
[l,N_test]=size(X_test);
Y_test=[];
for i=1:N_test
[temp]=im_point(X_test(:,i),X,V,2,'exp',[1 0]);
Y_test=[Y_test temp];
end
```

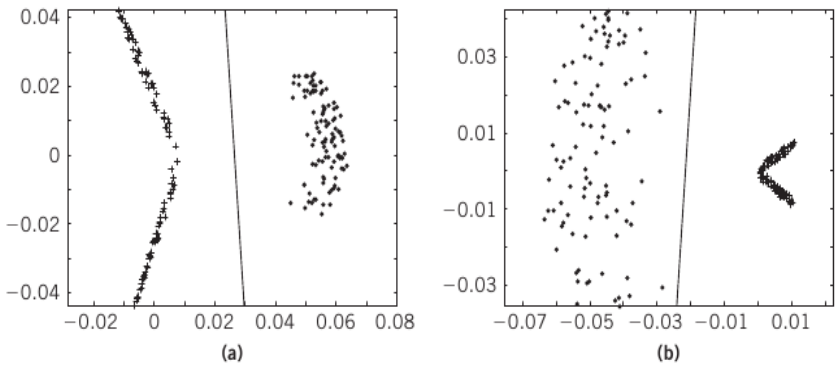
Untuk mengklasifikasikan vektor X_{test} (Y_{test}) dan menghitung kesalahan klasifikasi, ketikkan

```
y_out=2*(w'*[Y_test; ones(1,sum(n_points))]>0)-1;
class_err=sum(y_out.*y_test<0)/sum(n_points);
```

Gambar 3.6 (a) menunjukkan himpunan Y_{test} bersama-sama dengan garis yang sesuai dengan pengklasifikasi linear. Hal ini dihasilkan dengan mengetikkan

```
figure(6), plot(Y_test(1,y==1),Y_test(2,y==1),'r.',...
Y_test(1,y==-1),Y_test(2,y==-1),'b+')
figure(6), axis equal
% Ploting the linear classifier (works only if w(1)~=0)
y_lin=[min(Y_test(2,:)') max(Y_test(2,:)')];
x_lin=[(-w(3)-w(2)*y_lin(1))/w(1) (-w(3)-
w(2)*y_lin(2))/w(1)];
figure(6), hold on
figure(6), line(x_lin,y_lin)
```

Langkah 5. Untuk langkah ini, ulangi kode yang diberikan pada langkah 2 sampai 4. Sekarang dalam panggilan fungsi kernel PCA (langkah 2), [1, 0] digantikan oleh [0,6, 0] (lihat juga Gambar 3.6 (b)).



GAMBAR 3.6
Set Y_{test} dihasilkan pada langkah 4 dan pengklasifikasi linear ditentukan pada langkah 3 dari Contoh 3.5.2 untuk $\sigma = 1$ (a) dan $\sigma = 0{,}6$ (b).

Dari langkah-langkah yang telah dilakukan dapat ditarik tiga kesimpulan:

- Pertama, kernel PCA dapat melakukan pemetaan dalam ruang dimensi rendah di mana kelas yang terlibat dapat dipisahkan secara linear, meskipun hal ini tidak terjadi di ruang asli. Hal ini tidak dapat dicapai dengan PCA linier.

- Kedua, pilihan parameter fungsi kernel sangat penting. Untuk memverifikasinya, coba langkah 5 dengan $\sigma = 3$. Dalam hal ini, susunan kelas dalam ruang terubah terlihat sangat mirip dengan
pengaturan dalam ruang asli.

- Ketiga, untuk masalah ini, dua kelas tetap linear terpisah bahkan di ruang 1-dimensi sebagaimana didefinisikan oleh komponen utama yang pertama.

Contoh 3.5.3

1. Bangkitkan dua set data X_1 dan X_2 sebagai berikut:

    - X_1 terdiri dari 300 titik 2-dimensi. Pertama $N_1 = 100$ titik berasal dari kelas 1 yang dimodelkan dengan distribusi seragam di wilayah [-2,5, -1,5] × [-0,5, 0,5]; berikutnya $N_2 = 100$ titik yang berasal dari kelas 2 yang dimodelkan dengan distribusi seragam di daerah [0,5, 1,5] × [-2.5, -1.5]; Dan terakhir $N_3 = 100$ titik yang berasal dari kelas 3 dan terletak pada lingkaran C dengan jari-jari $r = 4$ dan berpusat di titik asal.
    Lebih khusus, titik N_3 terakhir yang dihasilkan sebagai berikut:

Titik-titik bentuk $x_i = -r \cdot \frac{2r}{N_2/2 - 1} i$, $i = 0,\ldots, N_2/2 - 1$ pada sumbu x yang dipilih (yang terletak pada interval [-2, 2]). untuk setiap x_i jumlah $v_i^1 = \sqrt{r^2 - x_i^2} + \eta_i^1$ dan $v_i^2 = -\sqrt{r^2 - x_i^2} + \eta_i^2$ dihitung, dimana η_i^1, η_i^2 berasal dari distribusi seragam pada interval $[-0{,}1, 0{,}1]^T$. Titik-titik (x_i, v_i^1) dan (x_i, v_i^2), $i = 0,\ldots, N_2 - 1$, terletak di sekitar C.

- X_2 terdiri dari 300 titik 2-dimensi. Titik $N_1 = 100$, $N_2 = 100$, dan $N_3 = 100$, masing-masing berasal dari kelas 1, 2, dan 3. Mereka terletak di sekitar lingkaran berpusat di titik asal dan masing-masing memiliki jari-jari $r_1 = 2$, $r_2 = 4$, dan $r_3 = 6$, (titik dati masing-masing kelas dihasilkan dengan mengadopsi prosedur yang digunakan untuk menghasilkan poin N_3 terakhir dari set data sebelumnya, X_1).

Bangkitkan vektor y_1 dan y_2, yang berisi label kelas dari titik-titik data dari set X_1 dan X_2. Plot X_1 dan X_2.

2. Gunakan kernel PCA pada X_1, menggunakan fungsi kernel eksponensial dengan $\sigma = 1$ dan pertahankan hanya dua yang pertama dari komponen utama. Tentukan dan plot set Y_1, yang berisi proyeksi titik-titik data dari X_1 ke subruang yang direntang oleh dua komponen utama. Ulangi langkah-langkah untuk X_2 dan tarik kesimpulannya.

Solusi. Gunakan langkah-langkah berikut:

Langkah 1. Untuk menghasilkan kumpulan data X_1, ketikkan

```
rand('seed',0)
noise_level=0.1;
%%%
n_points=[100 100 100];
l=2;
% Production of the 1st class
X1=rand(l,n_points(1))- [2.5 0.5]'*ones(1,n_points(1));
% Production of the 2nd class X1=[X1
rand(l,n_points(2))- [-0.5 2.5]'*ones(1,n_points(2))];
% Production of the 3rd class
c1=[0 0];
r1=4;
b1=c1(1)-r1;
b2=c1(1)+r1;
step=(b2-b1)/(n_points(2)/2-1);
for t=b1:step:b2
temp=[t c1(2)+sqrt(r1^ 2-(t-c1(1))^ 2)+noise_level*(rand-
0.5);...
t c1(2)-sqrt(r1^ 2-(t-c1(1))^ 2)+noise_level*(rand-0.5)]';
X1=[X1 temp];
end
```

Untuk menghasilkan vektor dari jenis label y_1, ketikkan

```
y1=[ones(1,n_points(1)) 2*ones(1,n_points(2))
3*ones(1,n_points(3))];
```

Plot set data X_1 , ketikkan

```
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'r.',...
X1(1,y1==2),X1(2,y1==2),'bx',...
X1(1,y1==3),X1(2,y1==3),'go')
```

Langkah 2. Untuk menggunakan kernel PCA pada X_1 dengan fungsi kernel exponensial (σ = 1), ketikkan

```
m=2;
[s,V,Y1]=kernel_PCA(X1,m,'exp',[1 0]);
To plot Y1 , type
figure(2), plot(Y1(1,y1==1),Y1(2,y1==1),'r.',...
Y1(1,y1==2),Y1(2,y1==2),'bx',...
Y1(1,y1==3),Y1(2,y1==3),'go')
```

Gunakan cara yang sama untuk X_2.
Pada Y_1, kelas-kelas terpisah linear, tapi ini tidak terjadi di Y_2. Artinya, kernel PCA tidak perlu mengubah masalah klasifikasi nonlinier menjadi linier.

Latihan 3.5.1
Dalam latihan ini, akan dilihat bagaimana ruang asli ditransformasi menggunakan kernel PCA. Untuk memastikan visualisasi hasilnya, dilihat lagi pada permasalahan 2-dimensi yang akan dipetakan ke ruang 2-dimensi yang direntang oleh dua yang pertama komponen utama yang dihasilkan dari kernel PCA. Untuk lebih jelasnya, gunakan langkah-langkah berikut ini:

Langkah 1. Bangkitkan kumpulan data *X*, yang berisi 200 vektor 2-dimensi. Vektor N_1 = 100 berasal dari kelas 1, yang dimodelkan dengan distribusi seragam dalam [-0,5, 0,5] 2; vektor N_2 = 100 berasal dari kelas -1 dan terletak di sekitar lingkaran dengan jari-jari 1 dan berpusat di titik asal. Titik N_2 dihasilkan sebagai titik terakhir N_3 dari set data X_1 dalam Contoh 3.5.3.
Langkah 2. Ulangi langkah 1 dari Contoh 3.5.2, dengan σ = 0,4.
Langkah 3. Ulangi langkah 2 dari Contoh 3.5.2.
Langkah 4. Perhatikan titik-titik *x* berbentuk (-2 + *i* * 0,1, -2 + *j* * 0,1), *i, j* = 0, ... , 40 (yang membentuk grid persegi panjang pada daerah $[-2,2]^2$) dan citranya di ruang yang direntang oleh dua komponen utama, yang ditentukan pada langkah 2. Klasifikasikan citra masing-masing titik *x* menggunakan *classifier* LS yang dirancang pada langkah 3 dan menghasilkan dua gambar. Yang pertama sesuai dengan ruang terubah dan sebuah titik citra diplot dengan suatu hijau o jika itu diklasifikasikan ke kelas pertama (+1) dan dengan suatu cyan x jika diklasifikasikan ke dalam kelas kedua (-1). Gambar kedua sesuai dengan ruang asli dan titik x yang sesuai digambar sama. Amati bagaimana transformasi nonlinear tersirat membuat deformasi ruang.
Langkah 5. Ulangi langkah 2 sampai 4, dengan σ = 1.

Petunjuk
Untuk menghasilkan *X*, sebagaimana pada Contoh 3.5.3. Untuk menentukan label kelas dari vektor data dan untuk plot *X*, gunakan seperti pada Contoh 3.5.2. Untuk melakukan kernel PCA dan menentukan *classifier* linier, gunakan juga seperti pada Contoh 3.5.2.

Untuk melakukan langkah 4, gunakan fungsi MATLAB *plot_orig_trans_kPCA* dengan mengetikkan:

```
m=2;
choice='exp';
para=[0.4 0];
reg_spec=[-2 0.1 2; -2 0.1 2];
fig_or=5;
fig_tr=6;
plot_orig_trans_kPCA(X,V,m,choice,para,w,reg_spec,fig_or,fig_
tr)
```

Hasilnya ditunjukkan pada Gambar 3.7. Amati bagaimana titik-titik di dalam lingkaran (dilambangkan dengan o) dalam ruang aslinya diperluas di ruang terubah. Juga perhatikan perbedaan antara ruang terubah untuk $\sigma = 0{,}4$ dan $\sigma = 1$ (Gambar 3.7 (a, c)).

3,6 *LAPLACIAN EIGENMAP*

Metode *Laplacian eigenmap* milik keluarga yang disebut sebagai metode berbasis grafik untuk reduksi dimensi. Idenya adalah untuk membangun suatu grafik, G (V, E), dimana node yang sesuai dengan titik data x_i, $i = 1, 2, \ldots, N$.. Jika dua titik yang dekat, node yang sesuai dihubungkan melalui sebuah tepi (edge), yang tertimbang sama. Dua titik dekat adalah lebih tinggi nilai bobot dari masing-masing tepi.

Kedekatan ditentukan melalui nilai ambang batas. Sebagai contoh, jika jarak *squared Euclidean* antara dua titik, $||x_i - x_j||^2$, adalah kurang dari ambang yang ditetapkan pengguna, misalnya e, yang masing-masing node terhubung dan dikatakan sebagai tetangga. Bobot yang sesuai dapat didefinisikan dalam berbagai cara. Sebuah pilihan umum untuk bobot W (i, j) antara node i dan j adalah, untuk beberapa variabel yang ditentukan pengguna σ^2,

$$W(i,j) = \begin{cases} \exp\left(-\frac{||x_i - x_j||^2}{\sigma^2}\right), & \text{jika } ||x_i - x_j||^2 < e \\ 0 & \text{selainnya} \end{cases}$$

Dengan demikian, grafik mengkodekan informasi lokal di dalam ruang dimensi tinggi R^l. Hal ini lebih lanjut diasumsikan bahwa data $x_i \in R^l$ teletak di atas suatu manifold halus dimensi m. Nilai dari m diasumsikan diketahui. Sebagai contoh, data asli dapat berada dalam ruang 3-dimensi R^3 namun terletak di sekitar *sphere*. Selanjutnya adalah sebuah manifold halus 2-dimensi karena dua parameter cukup untuk menggambarkan permukaan bola (*sphere*).
Tujuan dari metode ini adalah untuk mendapatkan representasi m-dimensi dari data sehingga informasi lingkungan lokal dalam manifold dipertahankan secara optimal. Dengan cara ini, struktur geometris lokal dari manifold tercermin dalam solusi yang diperoleh.

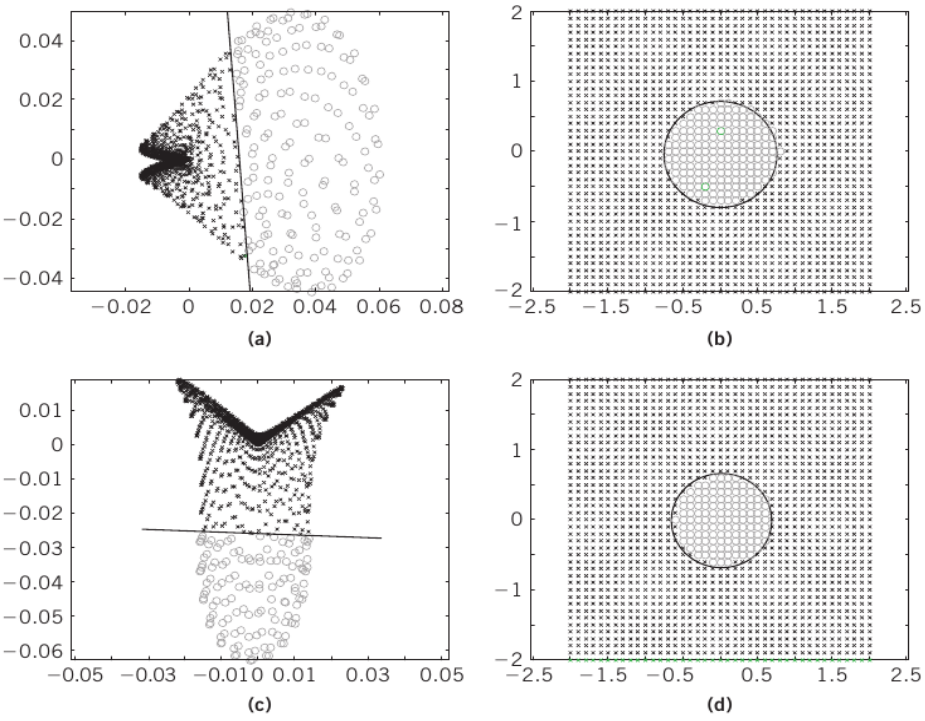
Laplacian eigenmap ternyata setara dengan permasalahan *eigenvalue* / *eigenvector* dari apa yang disebut dengan matriks *Laplacian*. Matriks ini mengkodekan informasi lokal seperti yang dijelaskan oleh bobot tepi dalam grafik [Theo 09, Bagian 6.7.2].

Untuk menggunakan *Laplacian eigenmap*, ketik

```
y = lapl_eig(X, e, sigma2, m)
```

dimana

e adalah nilai ambang batas,
sigma2 adalah parameter σ^2,
y adalah matriks $m \times N$ yang kolom ke-i mendefinisikan proyeksi dari vektor data ke-i ke subruang *m*-dimensi.



GAMBAR 3.7
(a) pengklasifikasi linear dalam ruang yang direntang oleh dua komponen utama pertama yang dihasilkan dari kernel PCA, menggunakan fungsi kernel eksponensial dengan $\sigma = 0{,}4$, dari Latihan 3.5.1. (b) Setara dari pengklasifikasi linear dalam ruang asli. (c) pengklasifikasi linear dalam ruang yang direntang oleh dua komponen utama pertama menggunakan kernel fungsi eksponensial dengan $\sigma = 1$, dari Latihan 3.5.1. (d) Setara dari pengklasifikasi linear dalam ruang asli. Amati pengaruh nilai σ.

Contoh 3.6.1. Hasilkan sebuah spiral Archimedes 3-dimensi sebagai satu pak dari 11 spiral Archimedes identik 2-dimensi, satu di atas yang lain. Sebuah spiral 2-dimensi digambarkan dalam koordinat polar oleh persamaan $r = a\theta$, di mana *a* adalah parameter yang ditetapkan pengguna. Dalam kasus ini, titik-titik spiral 3-dimensi dihasilkan sebagai berikut: Untuk θ, ambil nilai dari θ_{init} hingga θ_{fin} dengan langkah θ_{step} dan hitung

$r = a\theta$
$x = r \cos \theta$
$y = r \sin \theta$

Ke-11 titik-titik berbentuk (x, y, z), di mana z = -1, -0,8, -0,6,. . . , 0,8, 1, adalah titik-titik spiral. Gunakan a = 0,1, θ_{init} = 0,5, θ_{fin} = 2,05 * π, θ_{step} = 0,2.
Plot spiral 3-dimensi sehingga semua titik spiral 2-dimensi yang sama diplot menggunakan simbol yang sama dan semua kelompok dari 11 titik-titik berbentuk (x, y, z), di mana x dan y *tetap* dan z mengambil nilai -1, -0,8, -0,6,. . . , 0,8, 1, yang diplot menggunakan warna yang sama.

1. Lakukan *Laplacian eigenmap* pada titik-titik set data sebelumnya untuk dimensi *manifold* m = 2. Plot hasilnya.
2. Lakukan linier PCA pada set data yang sama, menggunakan dua komponen pertama utama. Plot hasilnya.
3. Bandingkan hasil yang diperoleh pada langkah 1 dan 2.

Solusi. Untuk menghasilkan dan plot spiral 3-dimensi, panggil fungsi *spiral_3D* dengan mengetikkan

```
a=0.1;
init_theta=0.5;
fin_theta=2.05*pi;
step_theta=0.2;
plot_req=1; % Request for plot the spiral
fig_id=1;
% Number id of the figure
% Producing the 3D spiral
[X,color_tot,patt_id]=spiral_3D(a,init_theta,...
fin_theta,step_theta,plot_req,fig_id);
[l,N]=size(X);
```

Gunakan Putar 3D untuk melihat spiral dari sudut pandang yang berbeda.
Lakukan hal berikut:

Langkah 1. Untuk melakukan *Laplacian eigenmap*, panggilan *lapl_eig* fungsi dengan mengetikkan

```
e=0.35;
sigma2=sqrt(0.5);
m=2;
y=lapl_eig(X,e,sigma2,m);
```

Untuk plot hasil, ketikkan

```
figure(2), hold on
for i=1:N
figure(2),
plot(y(1,i),y(2,i),patt_id(i),'Color',color_tot(:,i)')
end
```

Langkah 2. Untuk menggunakan linear PCA, panggil fungsi *pca_fun* dengan mengetikkan:

```
[eigenval,eigenvec,explain,Y]=pca_fun(X,m);
```

Untuk plot hasil, ketikkan

```
figure(3), hold on
for i=1:N
figure(3),
plot(Y(1,i),Y(2,i),patt_id(i),'Color',color_tot(:,i)')
end
```

Langkah 3. Amati Gambar 1 MATLAB pada layar komputer, perhatikan bagaimana warna setiap spiral 2-dimensi yang bervariasi dari merah ke kuning ke hijau ke *cyan* ke biru. Pada pemetaan dihasilkan oleh metode *Laplacian eigenmap* (MATLAB gambar 2), dua komentar berikutnya adalah dalam: setiap spiral 2-dimensi "ditarik" sehingga titik-titik tersebut ditempatkan pada segmen garis (arah horizontal); pada setiap arah vertikal pengganti warna yang diamati dalam MATLAB gambar 1 sama seperti yang diamati dalam MATLAB gambar 2. Jadi untuk pilihan parameter yang diberikan untuk *Laplacian eigenmap*, metode berhasil "membentang" spiral 3-dimensi.
Linier PCA (Matlab Gambar 3) juga berhasil pada "peregangan" setiap spiral 2-dimensi sehingga titik-titik ditempatkan pada segmen garis (arah horisontal). Namun, pengganti warna pada arah vertikal tidak sama dengan yang diamati di MATLAB gambar 1 (hijau, kuning, merah, *cyan*, biru). Hal ini menunjukkan bahwa setelah proyeksi yang dihasilkan oleh PCA, titik-titik yang jauh satu dengan yang lain di ruang 3-dimensi yang terletak dekat dalam representasi 2-dimensi (lihat, misalnya, titik merah dan *cyan* dalam ruang 3-dimensi asli dan dalam ruang 2-dimensi tereduksi).
Akhirnya, hasil yang diperoleh oleh *Laplacian eigenmap* sensitif terhadap pilihan parameter. Jika, misalnya, dilakukan percobaan sebelumnya untuk $e = 0{,}5$ dan *sigma2* = 1, *Laplacian eigenmap* gagal untuk sepenuhnya "membentang" spiral (dalam hal ini, merah dekat dengan hijau). Namun hal ini, meskipun tidak sempurna, masih lebih baik daripada yang dihasilkan oleh linier PCA. Secara umum, eksperimen tambahan diperlukan sebelum memilih nilai-nilai yang tepat untuk parameter yang terlibat dalam metode *Laplacian eigenmap*.

Latihan 3.6.1

1. Hasilkan "silinder dipotong" dari jari-jari $r = 0{,}5$ dalam ruang 3-dimensi sebagai paket dari 11 "lingkaran potong" identik satu di atas yang lain, seperti pada Contoh 3.6.1. Untuk lingkaran ke-j, dengan pusat $c_j = [c_{j\,1}, c_{j\,2}]^T$, titik-titik berikut ini dipilih:

   - $\left(x_i, c_{j2} \pm \sqrt{r^2 - (x_i - c_{j1})^2}\right)$ untuk x_i mulai dari $c_{j\,1} - r$ ke $c_{j\,1} + r$, dengan $(2r) / (N - 1)$.

   - $\left(x_i, c_{j2} \pm \sqrt{r^2 - (x_i - c_{j1})^2}\right)$ untuk x_i mulai dari $c_{j\,1} + r$ ke $(c_{j\,1} - r) / 4$, dengan langkah yang dipilih sebelumnya, dimana N adalah jumlah titik dimana x_i adalah sampel dalam rentang $[c_{j\,1} - r, c_{j\,1} + r]$.

Plot silinder terpotong sehingga semua titik lingkaran potong 2-dimensi yang sama diplot menggunakan simbol yang sama, dan semua kelompok dari 11 titik-titik berbentuk (x, y, z), di mana *x* dan *y* tetap dan *z* mengambil nilai-nilai -1, -0,8, -0,6, ... , 0.8,1 diplot menggunakan warna yang sama.

2. Lakukan *Laplacian eigenmap* pada titik-titik set data sebelumnya untuk dimensi *manifold* *m*= 2. Plot hasilnya.
3. Lakukan linier PCA pada data yang sama diatur menggunakan dua komponen utama pertama dan plot hasilnya.
4. Bandingkan hasil yang diperoleh dalam langkah 2 dan 3.

Petunjuk

1. Untuk menghasilkan dan plot potong silinder 3-dimensi, memanggil fungsi cut_cylinder_3D dengan mengetikkan

```
r=0.5;
center=[0 0];
N=25;
plot_req=1; %Request for plot the cylinder
fig_id=1;
%Number id of the figure
% Producing the 3D cut cylinder
[X,color_tot,patt_id]=cut_cylinder_3D(r,center,N,plot_req,...
fig_id);
[l,N]=size(X);
```

Gunakan 3D Rotate untuk mengamati silinder terpotong dari sudut pandang yang berbeda.

2. Terapkan langkah 1 dari Contoh 3.6.1, menggunakan parameter yang sama untuk metode *Laplacian eigenmap*.
3. Terapkan langkah 2 dari Contoh 3.6.1.
4. Perhatikan bahwa, sekali lagi, *Laplacian eigenmap* memberikan hasil yang lebih baik dibandingkan dengan hasil dari linier PCA. Namun, hal ini tidak terjadi ketika silinder terpotong memiliki radius sama dengan 5. (Mengapa? Bandingkan ketinggian silinder dengan jari-jarinya.)