パターン認識
Pattern Recognition

佐藤真一
Shin'ichi Satoh

国立情報学研究所
National Institute of Informatics

Apr 18, 2023

Course Information

- Course web page: http://research.nii.ac.jp/~satoh/utpr/
- Credits will be given based on final report (mandatory) and assignments (3 out of 7 are mandatory)
- Attendance record will NOT be taken
- If you fail to submit minimum 3 assignments and final report, you will not obtain credits.

Schedule (subject to change)

4/11 Orientation, Bayes decision theory, probability distribution

4/18 Random variable, random vector, normal distributions

4/25 Parametric density estimation, discriminant function

5/2 Nonparametric density estimation, Parzen windows, k-nearest neighbor estimate

5/9 k-nearest neighbor classification, classification error estimation

5/16 Bayes error estimation, classification error estimation, cross-validation, bootstrap

5/23 Linear classifier, perceptron, MSE classifier, Widrow-Hoff rule

5/30 neural network, deep learning

6/6 all about SVM

6/13 Orthogonal expansions, Eigenvalue decomposition

6/20 no class

6/27 Clustering, dendrogram, aggromerative clustering, k-means

7/4 Graphs, normalized cut, spectral clustering, Laplacian Eigenmaps

7/11 extra (if needed)

Roadmap

- We learned that we can perform classification based on Bayes Theorem if we know probability distributions.
- Today we learn how to estimate parameters which describe probability distributions such as normal distributions.
- Especially parameter estimation based on samples, and statistical properties of estimated parameters will be visited.
- We also consider linear transformation of the space.
- It can be used to select statistically principal features and to reduce correlations among features.

Random Vectors and Distributions

Input to pattern recognition system to be a *random vector*:

$$\mathbf{X} = [\mathbf{x}_1\, \mathbf{x}_2\, \cdots\, \mathbf{x}_n]^T.$$

Distribution function:

$$P(x_1, \ldots, x_n) = Pr\{\mathbf{x}_1 \leq x_1\ \ldots, \mathbf{x}_n \leq x_n\}$$

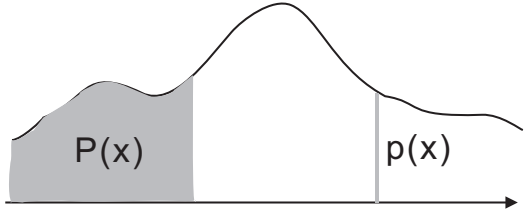
$$P(X) = Pr\{\mathbf{X} \leq X\}$$

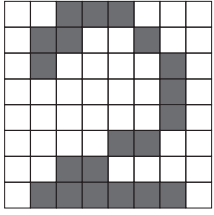
Density Function:

$$p(X) = \lim_{\substack{\Delta x_1 \to 0 \\ \vdots \\ \Delta x_n \to 0}} \frac{Pr\{x_1 < \mathbf{x}_1 \leq x_1 + \Delta x_1, \ldots, x_n < \mathbf{x}_n \leq x_n + \Delta x_1\}}{\Delta x_1 \cdots \Delta x_n}$$

$$= \frac{\partial^n P(X)}{\partial x_1 \ldots \partial x_n}$$

Random Vectors and Distributions





$$p(X) = \frac{\partial^n P(X)}{\partial x_1 \dots \partial x_n}$$

$$P(X) = \int_{-\infty}^{X} p(Y)dY = \int_{-\infty}^{x_1} \dots \int_{-\infty}^{x_n} p(y_1, \dots, y_n)dy_1 \dots dy_n$$

Example 1 Coin toss, **x** takes H (head) or T (tail) where $P(H) = P(T) = \frac{1}{2}$.

Example 2 Uniform distribution **x**

$$p(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

Example 3 Normal distribution **x**

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

Conditional density, prior probability, and posterior probability

Conditional density

$$p(X|\omega_i) = p_i(X)$$

Prior probability

$$P(\omega_i) = P_i$$

(unconditional) Density

$$p(X) = \sum_i p_i(X)P_i$$

Posterior probability

$$P(\omega_i|X) = \frac{p_i(X)P_i}{p(X)}$$

Parameters of Distributions

A random vector $\mathbf{X}$ is fully characterized by its distribution, however, its function is hard to represent. Instead distribution is characterized by parameters.

Expectation

$$M = E\{\mathbf{X}\} = \int X p(X) dX$$

$$m_i = \int x_i p(X) dX = \int_{-\infty}^{\infty} x_i p(x_i) dx_i$$

$$p(x_i) = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} p(X) dx_1 \cdots dx_{i-1} dx_{i+1} \cdots dx_n$$

(marginal density)

Conditional expectation

$$M_i = E\{\mathbf{X}|\omega_i\} = \int X p_i(X) dX$$

Covariance matrix:

$$
\begin{aligned}
\Sigma &= E\{(\mathbf{X} - M)(\mathbf{X} - M)^T\} \\
&= E\left\{ \begin{bmatrix} \mathbf{x}_1 - m_1 \\ \vdots \\ \mathbf{x}_n - m_n \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - m_1 & \dots & \mathbf{x}_n - m_n \end{bmatrix} \right\} \\
&= E\left\{ \begin{bmatrix} (\mathbf{x}_1 - m_1)(\mathbf{x}_1 - m_1) & \dots & (\mathbf{x}_1 - m_1)(\mathbf{x}_n - m_n) \\ \vdots & & \vdots \\ (\mathbf{x}_n - m_n)(\mathbf{x}_1 - m_1) & \dots & (\mathbf{x}_n - m_n)(\mathbf{x}_n - m_n) \end{bmatrix} \right\}
\end{aligned}
$$

Covariance matrix:

$$\Sigma = \begin{bmatrix} E\{(\mathbf{x}_1 - m_1)(\mathbf{x}_1 - m_1)\} & \dots & E\{(\mathbf{x}_1 - m_1)(\mathbf{x}_n - m_n)\} \\ \vdots & & \vdots \\ E\{(\mathbf{x}_n - m_n)(\mathbf{x}_1 - m_1)\} & \dots & E\{(\mathbf{x}_n - m_n)(\mathbf{x}_n - m_n)\} \end{bmatrix}$$

$$= \begin{bmatrix} c_{11} & \dots & c_{1n} \\ \vdots & & \vdots \\ c_{n1} & \dots & c_{nn} \end{bmatrix}$$

Properties:

- Covariance matrix is composed of variances and covariances.
- Covariance matrix is symmetric.

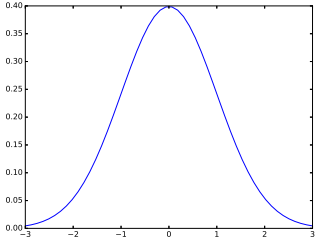
Normal Distributions

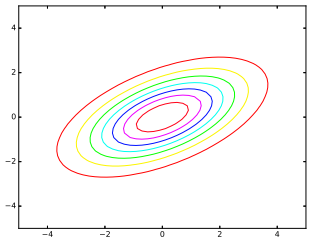
$$N_x(M,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(X-M)^T\Sigma^{-1}(X-M))$$

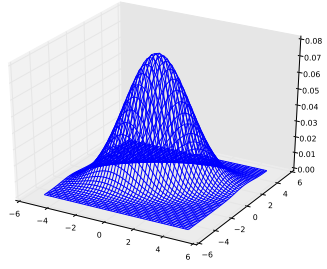
Characteristics:

- M and Σ are *sufficient statistics* for the normal distribution.
- If $\mathbf{x}_i$ are mutually uncorrelated they are also independent.
  - $\mathbf{x} \sim N_x(0,\sigma)$ and $\mathbf{y} = \mathbf{x}^2$ are uncorrelated but dependent (proof?).
  - $\mathbf{X}$ and $\mathbf{Y}$ are uncorrelated $\rightleftharpoons E\{\mathbf{XY}\} = E\{\mathbf{X}\}E\{\mathbf{Y}\}$
  - $\mathbf{X}$ and $\mathbf{Y}$ are independent $\rightleftharpoons P(\mathbf{X}=x,\mathbf{Y}=y) = P(\mathbf{X}=x)P(\mathbf{Y}=y)$
- Normal marginal densities are normal.
- Physical justification due to central limit theorem.
  - Average of independent and identically distributed (i.i.d.) random samples converges to normal.

Normal Distributions (normdist.py)







Mahalanobis Distance

$$
\begin{aligned}
d^2(X) &= (X - M)^T \Sigma^{-1} (X - M) \\
&= tr\{\Sigma^{-1}(X - M)(X - M)^T\} \\
&= \sum_i \sum_j h_{ij}(x_i - m_i)(x_j - m_j)
\end{aligned}
$$

where h_{ij} is the i, j component of Σ^{-1}.
Normal distribution is an exponential function of the Mahalanobis distance.

$$
N_x(M, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(X - M)^T \Sigma^{-1} (X - M))
$$

Estimation of Parameters

Let $\mathbf{y}$ be a function of $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$$

with the expected value m_y and variance σ_y^2:

$$m_y = E\{\mathbf{y}\} \quad \text{and} \quad \sigma_y = Var\{\mathbf{y}\}.$$

(components of) M and Σ are special case of $\mathbf{y}$.
Especially, when $\mathbf{y} = \mathbf{x}_1^{i_1} \cdots \mathbf{x}_n^{i_n}$, m_y is called $(i_1 + \cdots + i_n)$th order moment.

Moments of the estimates

Sample estimate:

$$\hat{m_y} = \frac{1}{N} \sum_{k=1}^{N} y_k.$$

Sample estimate $\hat{m_y}$ is *unbiased*:

$$\begin{aligned} E\{\hat{\mathbf{m}}_y\} &= \frac{1}{N} \sum_{k=1}^{N} E\{\mathbf{y}_k\} \\ &= \frac{1}{N} \sum_{k=1}^{N} m_y = m_y. \end{aligned}$$

Variance:

$$\begin{aligned} Var\{\hat{\mathbf{m}}_y\} &= E\{(\hat{\mathbf{m}}_y - m_y)^2\} \\ &= \frac{1}{N^2}\sum_{k=1}^{N}\sum_{\ell=1}^{N} E\{(\mathbf{y}_k - m_y)(\mathbf{y}_\ell - m_y)\} \\ &= \frac{1}{N^2}\sum_{k=1}^{N} E\{(\mathbf{y}_k - m_y)^2\} = \frac{1}{N}\sigma_y^2. \end{aligned}$$

Since $Var\{\hat{\mathbf{m}}_y\}$ goes to zero along with N to infinity, sample estimates are *consistent*.

Central Moments

Central moments such as variances and covariance matrices are more complicated.

$$\mathbf{y} = (\mathbf{x}_i - m_i)(\mathbf{x}_j - m_j)$$

since m_i and m_j are unknown, should be replaced by sample estimates:

$$\mathbf{y} = (\mathbf{x}_i - \hat{\mathbf{m}}_i)(\mathbf{x}_j - \hat{\mathbf{m}}_j)$$

Sample Covariance Matrix

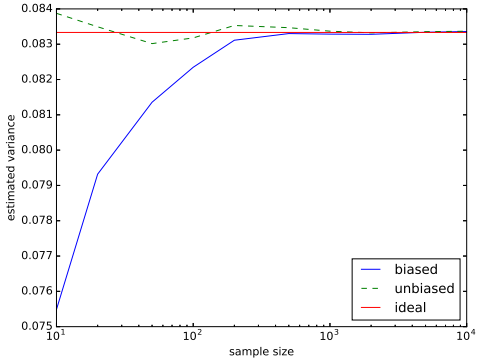
$$
\begin{aligned}
\hat{\Sigma} &= \frac{1}{N}\sum_{k=1}^{N}(\mathbf{X}_k - \hat{\mathbf{M}})(\mathbf{X}_k - \hat{\mathbf{M}})^T \\
&= \frac{1}{N}\sum_{k=1}^{N}\{(\mathbf{X}_k - M) - (\hat{\mathbf{M}} - M)\}\{(\mathbf{X}_k - M) - (\hat{\mathbf{M}} - M)\}^T \\
&= \frac{1}{N}\sum_{k=1}^{N}(\mathbf{X}_k - M)(\mathbf{X}_k - M)^T - (\hat{\mathbf{M}} - M)(\hat{\mathbf{M}} - M)^T \\
E\{\hat{\Sigma}\} &= \Sigma - E\{(\hat{\mathbf{M}} - M)(\hat{\mathbf{M}} - M)^T\} \\
&= \Sigma - \frac{1}{N}\Sigma = \frac{N-1}{N}\Sigma
\end{aligned}
$$

thus biased. Unbiased sample covariance matrix is then:

$$\hat{\Sigma} = \frac{1}{N-1} \sum_{k=1}^{N} (\mathbf{X}_k - \hat{\mathbf{M}})(\mathbf{X}_k - \hat{\mathbf{M}})^T.$$

Exercise

biasvar.py (biasvar.m, biasvar.sci)
Compute variances at each sample size of unit random values with biased and unbiased methods and plot the means of N (=1000) epochs.



Linear Transformation

When a vector $\mathbf{X}$ is transformed linearly by A to another vector $\mathbf{Y}$,

$$\mathbf{Y} = A^T \mathbf{X}$$

Then expected vector and covariance matrix of Y:

$$\begin{aligned} M_Y &= E\{\mathbf{Y}\} = A^T E\{\mathbf{X}\} = A^T M_X \\ \Sigma_Y &= E\{(\mathbf{Y} - M_Y)(\mathbf{Y} - M_Y)^T\} \\ &= A^T E\{(\mathbf{X} - M_X)(\mathbf{X} - M_X)^T\} A \\ &= A^T \Sigma_X A. \end{aligned}$$

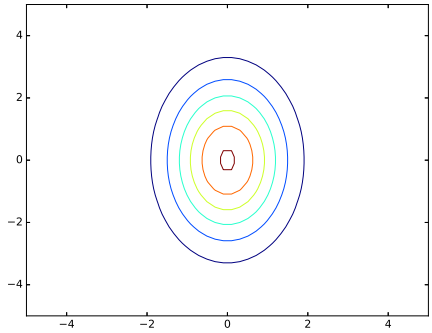
Python sample (linearnorm.py)

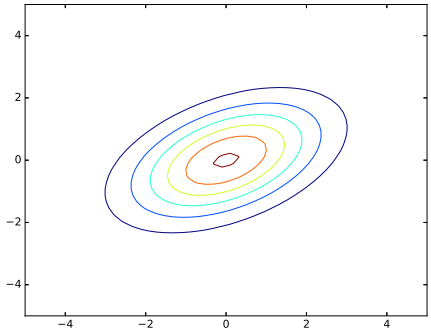
```
import numpy as np
import matplotlib.pyplot as plt

cov = np.matrix([[1, 0], [0, 3]])
icov = np.linalg.inv(cov)
r = np.matrix([[np.cos(np.pi / 3), -np.sin(np.pi / 3)],
               [np.sin(np.pi / 3), np.cos(np.pi / 3)]])
xx, yy = np.meshgrid(np.linspace(-5, 5), np.linspace(-5, 5))
plt.figure()
plt.axis('equal')
p = 1. / (2. * np.pi * np.sqrt(np.linalg.det(cov))) \
    * np.exp(-1. / 2. * \
             (icov[0, 0] * xx * xx \
              + (icov[0, 1] + icov[1, 0]) * xx * yy \
              + icov[1, 1] * yy * yy))
plt.contour(xx, yy, p)
```

Python sample (linearnorm.py)

```
# plt.savefig('linearnorm-1.eps')
plt.figure()
plt.axis('equal')
cov2 = r.T.dot(cov).dot(r)
icov2 = np.linalg.inv(cov2)
pp = 1. / (2. * np.pi * np.sqrt(np.linalg.det(cov2))) \
    * np.exp(-1. / 2. \
             * (icov2[0, 0] * xx * xx \
                + (icov2[0, 1] + icov2[1, 0]) * xx * yy \
                + icov2[1, 1] * yy * yy))
plt.contour(xx, yy, pp)
# plt.savefig('linearnorm-2.eps')
plt.show()
```

Python sample (linearnorm.py)





Matlab sample

```
cov=[1 0; 0 3];
r=[cos(pi/3) -sin(pi/3); sin(pi/3) cos(pi/3)];
[xx,yy]=meshgrid(-5:0.5:5,-5:0.5:5);
xy=[xx(:) yy(:)];
figure
p=1/(2*pi*sqrt(det(cov))) * ...
    exp(-1/2*diag(xy*inv(cov)*xy'));
contour(xx,yy,reshape(p,size(xx)),10);
figure
p=1/(2*pi*sqrt(det(cov))) * ...
    exp(-1/2*diag(xy*inv(r'*cov*r)*xy'));
contour(xx,yy,reshape(p,size(xx)),10);
```

Invariance of Mahalanobis Distance

$$
\begin{aligned}
d_Y^2(Y) &= (Y - M_Y)^T \Sigma_Y^{-1} (Y - M_Y) \\
&= (X - M_X)^T A A^{-1} \Sigma_X^{-1} (A^T)^{-1} A^T (X - M_X) \\
&= (X - M_X) \Sigma_X^{-1} (X - M_X) \\
&= d_X^2(X).
\end{aligned}
$$

Orthonormal Transformation

Let's shift the coordinate system to be zero mean:

$$Z = X - M$$

Then

$$d_Z^2(Z) = Z^T \Sigma^{-1} Z.$$

Let's find a vector Z which maximizes $d_Z^2(Z)$ subject to $Z^T Z = 1$ (unit vector).

$$\frac{\partial}{\partial Z}\{Z^T \Sigma^{-1} Z - \mu(Z^T Z - 1)\} = 2\Sigma^{-1} Z - 2\mu Z = 0. \text{ (derive!)}$$

Lagrange multipliers method

To solve optimization problem below:

$$\text{maximize } f(x) \text{ subject to } g(x) = 0$$

with Lagrangian with Lagrange multiplier λ:

$$\Lambda(x, \lambda) = f(x) + \lambda g(x)$$

the original problem can be solved by

$$\frac{\partial}{\partial x}\Lambda = \frac{\partial}{\partial \lambda}\Lambda = 0$$

Orthonormal Transformation

$\Sigma^{-1}Z = \mu Z$ or $\Sigma Z = \lambda Z$ ($\lambda = \frac{1}{\mu}$) can be solved as Eigenvalue problem.
The characteristic equation:

$$|\Sigma - \lambda I| = 0.$$

Any value of λ satisfies the equation is *eigenvalue* and corresponding Z is called *eigenvector*.
For a symmetric $n \times n$ matrix Σ we have n real eigenvalues $\lambda_1, \ldots, \lambda_n$ and n real eigenvectors $\phi_1, \ldots, \phi_n$.
Properties: the eigenvectors corresponding to different eigenvalues are orthogonal

$$\phi_i^T \phi_j = 0 \ i \neq j$$

Eigenvector matrix $\Phi = [\phi_1 \ \cdots \ \phi_n]$
Eigenvalue matrix $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$

Orthonormal Transformation

Apparently

$$\Sigma\Phi = \Phi\Lambda$$
$$\Phi^T\Phi = I$$

Orthonormal Transformation

Let's use Φ as the transformation matrix

$$Y = \Phi^T X$$

Then

$$\Sigma_Y = \Phi^T \Sigma_X \Phi = \Lambda.$$

Properties:

- Y represents X in the new coordinate system spanned by $\phi_1, \ldots, \phi_n$.
- A covariance matrix is diagonalized. This means the corresponding random variables are uncorrelated in general and independent for normal distributions.
- Since eigenvectors maximize $d_Z^2(Z)$, the transformation selects principal components of the distribution.
- The transformation is called *orthonormal transformation*.

Whitening Transformation

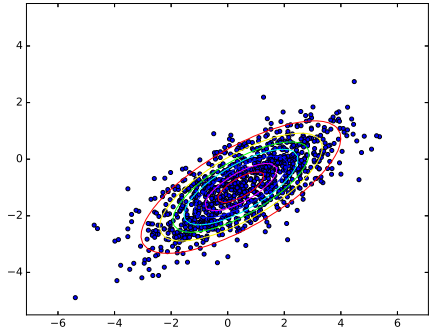
After applying the orthonormal transformation, we can add another transformation $\Lambda^{-\frac{1}{2}}$ that will make the covariance matrix to be identity matrix:

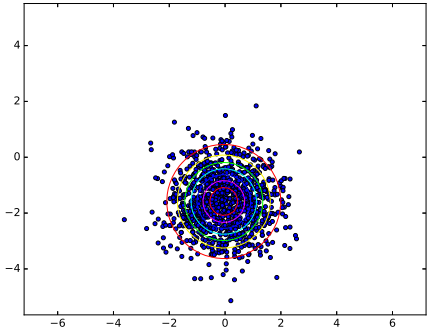
$$Y = \Lambda^{-\frac{1}{2}}\Phi^T X = (\Phi\Lambda^{-\frac{1}{2}})^T X$$
$$\Sigma_Y = \Lambda^{-\frac{1}{2}}\Phi^T\Sigma_X\Phi\Lambda^{-\frac{1}{2}} = \Lambda^{-\frac{1}{2}}\Lambda\Lambda^{-\frac{1}{2}} = I.$$

This transformation $\Phi\Lambda^{-\frac{1}{2}}$ is called the whitening transformation or the whitening process.
Properties:

- Whitening transformations are not orthonormal.
- After a whitening transformation, the covariance matrix is invariant under any orthonormal transformation
$$\Psi^T I \Psi = \Psi^T\Psi = I.$$

Whitening Transformation (whitening.py)





Simultaneous Diagonalization

We can diagonalize two symmetric matrices Σ_1 and Σ_2 simultaneously by a linear transformation.

(1) Whiten Σ_1 by

$$Y = \Theta^{-\frac{1}{2}} \Phi^T X$$

where Θ and Φ are the eigenvalue and eigenvector matrices of Σ_1 as

$$\Sigma_1 \Phi = \Phi \Theta \text{ and } \Phi^T \Phi = I.$$

Then Σ_1 and Σ_2 are transformed to

$$\Theta^{-\frac{1}{2}} \Phi^T \Sigma_1 \Phi \Theta^{-\frac{1}{2}} = I$$
$$\Theta^{-\frac{1}{2}} \Phi^T \Sigma_2 \Phi \Theta^{-\frac{1}{2}} = K.$$

In general, K is not diagonal matrix.

(2) We apply the orthonormal transformation to diagonalize K.

$$Z = \Psi^T Y$$

where Ψ and Λ are the eigenvector and eigenvalue matrices of K as

$$K\Psi = \Psi\Lambda \text{ and } \Psi^T\Psi = I.$$

Then by using Ψ

$$\Psi^T I \Psi = \Psi^T \Psi = I$$
$$\Psi^T K \Psi = \Lambda$$

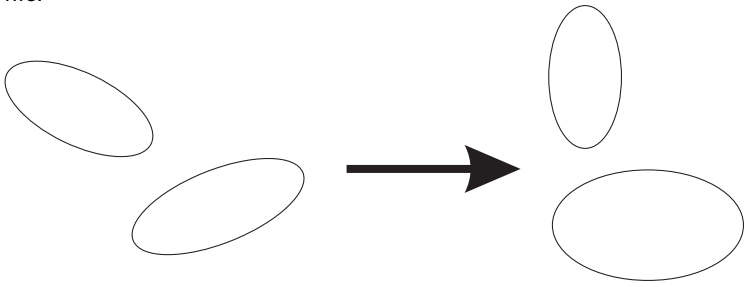
Thus both matrices are diagonalized.

Assignment

- Programming project and non-programming project will be imposed.
- You are expected to solve either programming project OR non-programming project.
- Programming project is recommended.
- Of course you are most welcomed to solve both.

Programming project

1. Two data files (data.mat and data2.mat) are given: each contains data matrix itself (2D 2000 points) yielding two different normal distributions, and sufficient statistics of the normal distributions.
   Implement simultaneous diagonalization using parameters and apply it to the data.
   Show scatter plot of points for two cases: before and after diagonalization, for each data file.



2. (Complicated, OPTIONAL) Apply the simultaneous diagonalization to the parameters of the normal distributions and overlay the contour maps.

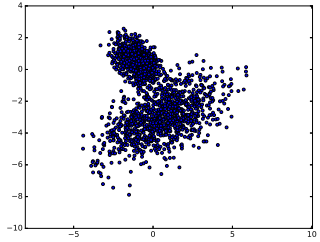
Programming project

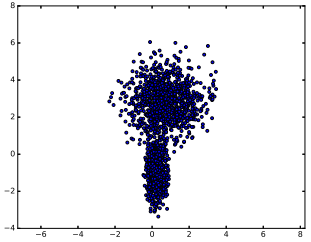
You can load data as follows:

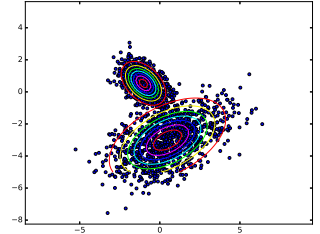
```
from scipy.io import loadmat
data = loadmat('data.mat')
x = data['x']  # data matrix
cov1 = data['cov1']  # covariance matrix 1
cov2 = data['cov2']  # covariance matrix 2
m1 = data['m1']  # mean vector 1
m2 = data['m2']  # mean vector 2
```

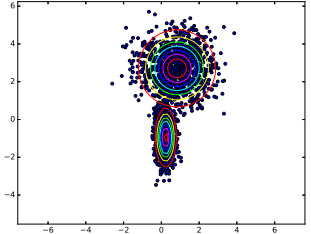
The files can be downloaded from the course web page.
The files can be obtained from the shared folder on Google drive in you use Google Colaboratory. Please refer to the colab link in the course web page for the method. You can also put downloaded files in your Google drive to refer from Google Colaboratory.

Programming project









Non-programming project

(1) Assume that $(\mathbf{x}_1, \mathbf{x}_2, \cdots)$ yield multivariate normal distribution. Show that $\mathbf{x}_i$ and $\mathbf{x}_j (i \neq j)$ independent if they are uncorrelated.

(2) Show that $\mathbf{x} \sim N_x(0, \sigma)$ and $\mathbf{y} = \mathbf{x}^2$ are uncorrelated but not independent.

How to submit assignment

- Submit your report as a PDF file.
- As for a report for the programming project, prepare a PDF file including your code, brief explanation of your code, example of the output, and explanation of the output. Don't just send your code!
- By default the due date will be two weeks after the issued date (for instace the due of today's assignment would be May 2).
- Upload your assignment via ITC-LMS. Don't send your assignment via email!