TP 3 : Fonctions discriminantes

Ce TP a pour but de générer des échantillons appartenant à des distributions gaussiennes différentes afin de pouvoir calculer le taux d'erreur de classification en ayant recourt aux fonctions discriminantes.

1. Génération des échantillons

Génération de 2 distributions normales de dimension *D=3* :

La densité normale univariée continue est donnée par :

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \quad (1)$$

où μ la moyenne est donnée par :

$$\mu = \varepsilon[x] = \int_{-\infty}^{\infty} xp(x)dx = \sum_{x\in D} xP(x) \quad (2)$$

et la variance σ^2 est donnée par :

$$\sigma^2 = E\left[(x-\mu)^2\right] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx = \sum_{x\in D} (x-\mu)^2 P(x) \quad (3)$$

Dans le cas de la loi normale multidimensionnelle, la distribution multivariée à *d* dimensions des nuages de points conditionnels s'écrit :

$$p(x) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} exp\left[-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right] \quad (4)$$

où:

- x est un vecteur colonne de d composantes,
- μ est le vecteur moyenne de d composantes,

- Σ est la matrice de covariance $d \times d$, $|\Sigma|$ est son déterminant et Σ^{-1} est son inverse,
- et le vecteur de la moyenne μ devient :

$$\mu = E[x] = \left[E\left[x_1\right] \ \dots \ E\left[x_d\right]\right] = \left[\mu_1 \ \dots \ \mu_d\right] = \int_{-\infty}^{\infty} xp(x)dx = \sum_x xP(x) \qquad (5)$$

Questions :

1. Générez, à l'aide d'un programme **progPrincipal.m**, des échantillons appartenant au deux distributions normales différentes, dans l'espace de dimension 3.
2. Essayer d'estimer les différents paramètres de chaque distribution (moyenne et variance).
3. Dans la même figure, illustrer les échantillons appartenant au deux distributions gaussiennes, dans l'espace de dimension 3.

1. Fonctions discriminantes

Les fonctions discriminantes peuvent être facilement évaluées, si les densités sont des gaussiennes multivariées. Dans ce cas, l'équation de la fonction discriminante peut être écrite sous la forme suivante :

$$g_k(x) = -\frac{1}{2}\left(x - \mu_k\right)^T \Sigma_k^{-1}\left(x - \mu_k\right) - \frac{D}{2} ln 2\pi - \frac{1}{2} ln\left|\Sigma_k\right| + lnP(w_k) \qquad (6)$$

Trois cas peuvent se présenter:

- $\Sigma_k = \sigma^2 I$
- $\Sigma_k = \Sigma$
- $\Sigma =$ arbitraire

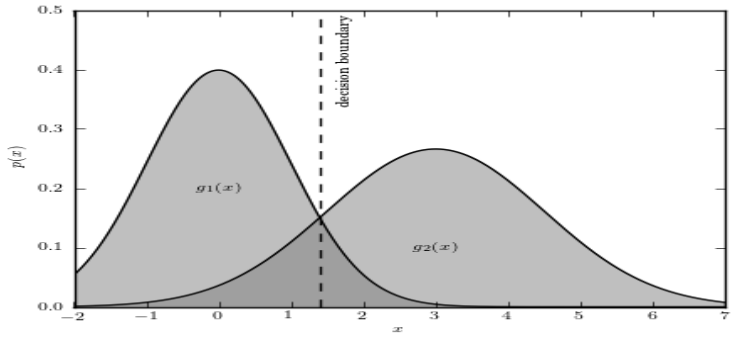
Question :

Essayer de créer des fonctions Matlab permettant de calculer le taux d'erreur pour chacun des trois cas vus précédemment.

Annexe

Partie 1 :

Illustration d'une fonction discriminante entre deux distributions gaussiennes univariées.



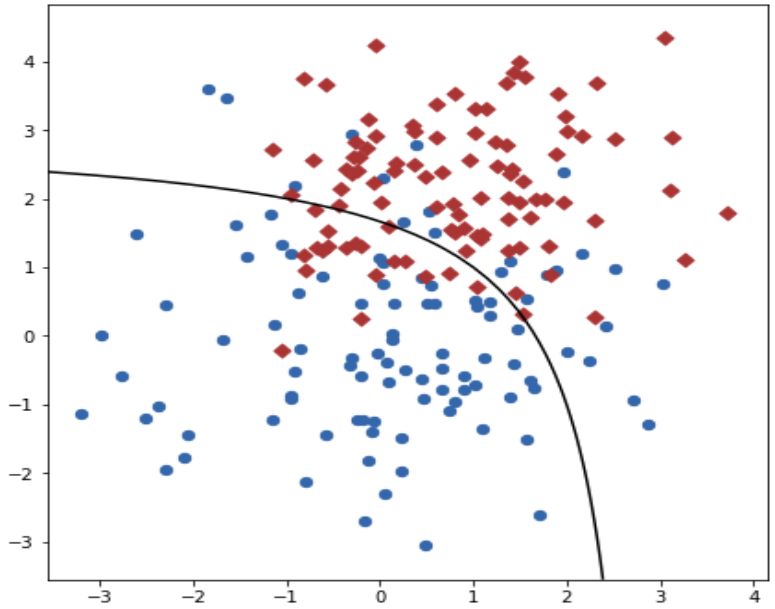
Code source python:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm
#------------------------------------------------------------
# Compute the two PDFs/Gaussians
x = np.linspace(-3, 7, 1000)
pdf1 = norm(0, 1).pdf(x)
pdf2 = norm(3, 1.5).pdf(x)
x_bound = x[np.where(pdf1 < pdf2)][0]
#------------------------------------------------------------
# Plot the pdfs and decision boundary
fig = plt.figure(figsize=(5, 3.75))
ax = fig.add_subplot(111)
ax.plot(x, pdf1, '-k', lw=1)
ax.fill_between(x, pdf1, color='gray', alpha=0.5)
ax.plot(x, pdf2, '-k', lw=1)
ax.fill_between(x, pdf2, color='gray', alpha=0.5)
# plot decision boundary
ax.plot([x_bound, x_bound], [0, 0.5], '--k')
ax.text(x_bound + 0.2, 0.49, "decision boundary", ha='left', va='top', rotation=90)
ax.text(0, 0.2, '$g_1(x)$', ha='center', va='center')

ax.text(3, 0.1, '$g_2(x)$', ha='center', va='center')
ax.set_xlim(-2, 7)
ax.set_ylim(0, 0.5)
ax.set_xlabel('$x$')
ax.set_ylabel('$p(x)$')
plt.show()
```

Partie 2 :

Illustration d'une fonction discriminante entre deux distributions gaussiennes bivariées.



Code source python:

```
mu_vec1 = np.array([0,0])
cov_mat1 = np.array([[2,0],[0,2]])
x1_samples = np.random.multivariate_normal(mu_vec1, cov_mat1, 100)
mu_vec1 = mu_vec1.reshape(1,2).T # to 1-col vector
mu_vec2 = np.array([1,2])
cov_mat2 = np.array([[1,0],[0,1]])
x2_samples = np.random.multivariate_normal(mu_vec2, cov_mat2, 100)
mu_vec2 = mu_vec2.reshape(1,2).T
def decision_boundary(x_vec, mu_vec1, mu_vec2):
    g1 = (x_vec-mu_vec1).T.dot((x_vec-mu_vec1))
    g2 = 2*( (x_vec-mu_vec2).T.dot((x_vec-mu_vec2)) )
    return g1 - g2
f, ax = plt.subplots(figsize=(7, 7))
c1, c2 = "#3366AA", "#AA3333"
ax.scatter(*x1_samples.T, c=c1, s=40)
ax.scatter(*x2_samples.T, c=c2, marker="D", s=40)
x_vec = np.linspace(*ax.get_xlim())
ax.contour(x_vec, x_vec,
           decision_boundary(x_vec, mu_vec1, mu_vec2),
           levels=[0], cmap="Greys_r")
```