REGRESSION LINEAIRE ET CLASSIFICATION

ALIREZA BANAEI

QUESTION 1

Générer un jeu de données simulées selon le modèle de mélange précédent. On prendra comme valeurs numériques : $\pi_+ = 0.5$, $p = 2$, $n = 500$, $\mu_- = (-1, -1)$, $\mu_+ = (1, 1)$, $\Sigma_+ = 3\, Id_p$, $\Sigma_- = 2\, Id_p$. Détailler le pseudocode (l'algorithme) utilisé.

CODE :

GENERATION DES DONNEES :

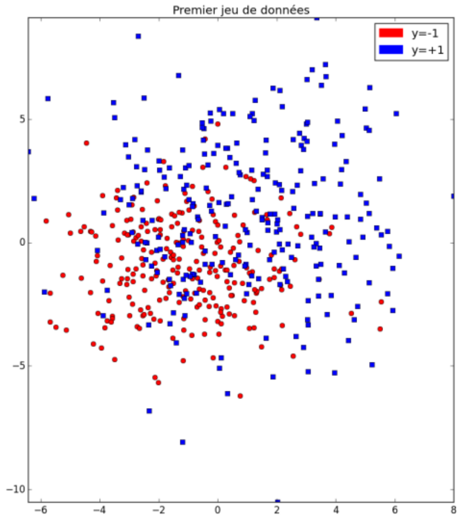
La fonction *generate_data*

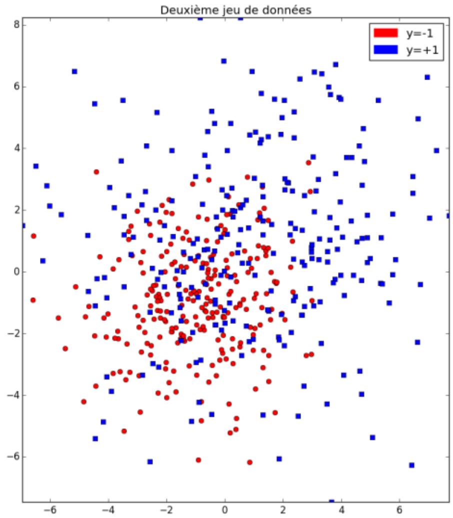
```
def generate_data(n=500,
                  mu_plus=[1,1], mu_moins=[-1, -1],
                  sigma_plus=[3, 3], sigma_moins=[2, 2],
                  p_plus=0.5):

        génération d'un échantillon de taille n (500 par défaut) composé de deux groupes de
        données : premier groupe avec une distribution normale N(mu_1, sigma_1) et le label +1
        et deuxième groupe avec la distribution normale N(mu_2, sigma_2)et le label -1. La
        proportion des individus du groupe 1 dans l'échantillon global est égale à p_plus.

        # génération des deux échantillons en utilisant la fonction rand_gausse du TP et
        l'ajout d'une colonne à la fin (des 1 pour le premier échantillon et des -1 pour le
        deuxième.
        data_plus = np.hstack([rand_gauss(n, mu_plus, sigma_plus), np.ones((n,1))])
        data_moins = np.hstack([rand_gauss(n, mu_moins, sigma_moins), -1*np.ones((n,1))])
        # génération d'un vecteur aléatoire (>=0 et <=1) de taille n
        rands = np.random.rand(n)
        # sélection des données de la première population avec une probabilité p_plus et de la
        deuxième population avec la probabilité 1-p_plus
        data = np.vstack([data_plus[rands<=p_plus,:], data_moins[rands>p_plus,:]])
        # mélange des lignes de data avent son renvoi
        idx = np.arange(n)
        np.random.shuffle(idx)
        return data[idx, :]
```

AFFICHAGE DES JEUX DE DONNÉES

Afficher deux exemples de jeux de données de sorte que le label de chaque point soit visible (en s'inspirant éventuellement de la fonction plot_2d du TP d'introduction à la classification).

Le plot de deux jeux de données de 500 points chacun générés selon l'algorithme ci-dessus (fonction *plot_data*)





QUESTION 2

On suppose que l'échantillon considéré contient n observations notées $(x_1, y_1), \ldots, (x_n, y_n)$, où $x_i \in R^p$ et $y_i \in \{-1, 1\}$ pour $1 \leq i \leq n$. Le nombre d'observations dans la classe « y = +1 » est $\sum_{i=1}^{n} \mathbb{I}\{y_i = +1\} = n_+$. On note $n_- = \sum_{i=1}^{n} \mathbb{I}\{y_i = -1\} = n_- = n - n_+$. En utilisant par exemple la méthode des moments (i.e., on remplace les espérances par leurs contre-parties empiriques), proposer des estimateurs non biaisés $\hat{\pi}_+$, $\hat{\pi}_-$ des paramètres π_+, μ_+, μ_-. Montrer que l'espérance théorique de X est $\mu = \pi_+\mu_+ + \pi_-\mu_-$.

LE CODE :

```
# Génération des données
data = generate_data();
# Séparation des deux groupes
data_plus = data[data[:,2]==1]
data_moins = data[data[:,2]==-1]
# Estimation du P+ en calculant la proportion des Y=+1
p_plus_estime = data_plus.shape[0]/data.shape[0]
print 'Probabilite estimee des Y=-1 : ', p_plus_estime
# Estimation de moyennes
means_plus = data_plus.mean(0)
means_moins = data_moins.mean(0)
means = data.mean(0)
print 'Mu plus estime : ', means_plus[:2]
print 'Mu moins estime : ', means_moins[:2]
print 'Mu globale : ', means[:2]
```

RESULTATS

Affichage des résultats :

```
Probabilite estimee des Y=+1 :  0.516
Mu plus estime :  [ 0.97617648  0.90152217]
Mu moins estime :  [-0.72515626 -1.10556885]
Mu globale :  [ 0.15273143 -0.06990989]
```

Certaines estimations sont assez fausses. En montant la taille de l'échantillon à 1 000 000 on obtient :

```
Probabilite estimee des Y=-1 :  0.499616
Mu plus estime :  [ 1.0020369   1.00769163]
Mu moins estime :  [-1.00434872 -0.99819708]
Mu globale :  [-0.00192636  0.00397701]
```

qui sont plus précises.

ESPERANCE DE X

$$E(X) = \mu = \sum P_i x_i = \sum \pi_+ x_{+i} + \sum \pi_- x_{-i} = \pi_+ \mu_+ + \pi_- \mu_-$$

QUESTION 3

Sur le même principe, donner des estimateur Σ + , Σ− des variances au sein de chaque classe. Ces estimateurs sont-ils biaisés ?

LE CODE :

Utilisation de la fonction var de np. Ces estimations sont en principe biaisées et il faut les multiplier par n/(n-1), mais vue la taille importante de l'échantillon cette correction est pratiquement sans effet.

```
var_plus = data_plus.var(0)
var_moins = data_moins.var(0)
print 'Varience plus estimee (biaisée) : ', var_plus[:2]
print 'Varience plus estimee (non biaisée) : ', (data_plus.shape[0]/(data_plus.shape[0]-1)) * var_plus[:2]
print 'Varience moins estime (biaisée): ', var_moins[:2]
print 'Varience moins estime (non biaisée): ', (data_moins.shape[0]/(data_moins.shape[0]-1)) * var_moins[:2]
```

RESULTATS :

```
Varience plus estimee (biaisée) :  [ 7.81895572  9.02172407]
Varience plus estimee (non biaisée) :  [ 7.84937968  9.05682805]
Varience moins estime (biaisée):  [ 4.46772253  3.6848898 ]
Varience moins estime (non biaisée):  [ 4.48626079  3.7001798 ]
```

Les mêmes résultats pour n = 1 000 000 :

```
Varience plus estimee (biaisée) :  [ 8.99659435  9.00558954]
Varience plus estimee (non biaisée) :  [ 8.99661236  9.00560756]
Varience moins estime (biaisée):  [ 3.98464091  3.9981681 ]
Varience moins estime (non biaisée):  [ 3.98464888  3.99817609]
```

QUESTION 4

On résout un problème de minimisation d'un critère des moindres carrés …

LE CODE :

```
# Utilisation de l'objet LinearRegression du package sklearn et sa méthode fit pour calculer les paramètres du classifieur
regr = linear_model.LinearRegression()
```

```
regr.fit(data[:,:2], data[:,2])
# Extraction de Theta 0 et Theta 1
T1 = regr.coef_
T0 = regr.intercept_
print 'Theta 1 = ', T1
print 'Theta 0 = ', T0
```

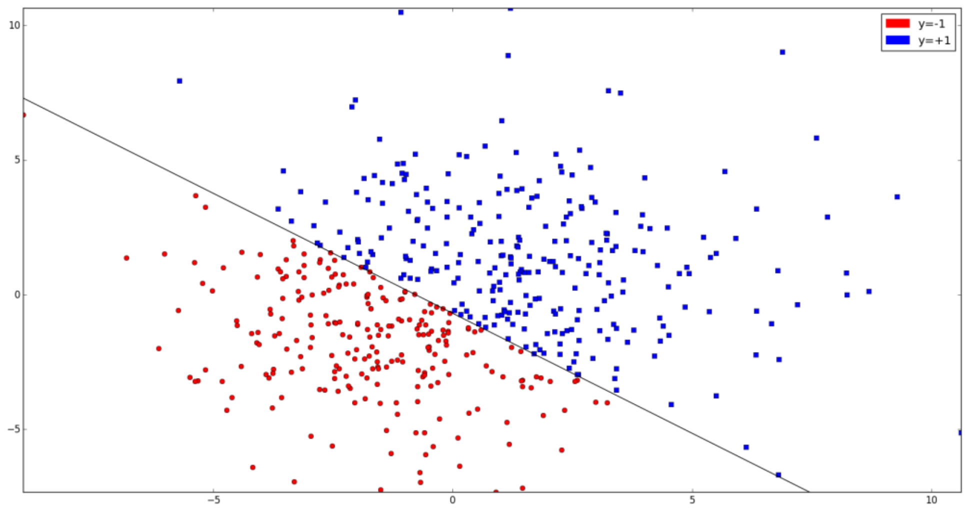
RESULTAT

```
Theta 1 =  [ 0.10070542  0.11299061]
Theta 0 =  0.0787559869317
```

Construire numériquement le classifieur ci-dessus, et l'appliquer aux jeux de données simulées. Détailler le pseudo-code. Afficher le résultat en faisant apparaître les zones du plan correspondant à chaque classe, en même temps que les données d'apprentissage. On pourra par exemple s'inspirer de la fonction frontiere du TP d'introduction à la classification.

```
# Construction de la matrice Theta :
T0 = T0.reshape(1,1)
T1 = T1.reshape(2,1)
Theta = np.vstack([T0, T1])
# Construction de la matrice Z
Z = np.hstack([np.ones((data.shape[0],1)), data[:,:2]])
# Calcul des labels Y estimé en applicant la function signe (np.sign)
Y_estime = np.sign(np.dot(Z,Theta))
# Construction d'une matrice des données avec les labels
data_with_y_estime = np.hstack([data[:,:2], Y_estime])
# Affichage du nuage des points avec la function plot_data
plot_data(data_with_y_estime)
# Affichage de la ligne correspondant à l'équation θ1x1 + θ2x2 + θ0 = 0
ax_1 = np.arange(-10,10,0.1).reshape(200,1)
ax_2 = calcule_y(ax_1, T1, T0).reshape(200,1)
plt.plot(ax_1, ax_2, 'k-')

# La fonction pour calculer y (ix=ici x2)selon l'équation θ1x1 + θ2x2 + θ0 = 0, en fonction de x (ici x1, le vecteur Theta 1 et le scalaire Theta 0
def calcule_y(x, T1, T0):
    return (T0 - T1[1]*x)/T1[0]
```

LE GRAPHE :



QUESTION 4

Écrire la fonction de coût L(θ) en faisant intervenir les matrices Z, θ et Y.

REPONSE

$$l(\theta) = Y - Z\theta$$

Ecrire la condition d'annulation du gradient sous forme d'une équation matricielle faisant intervenir les matrices précédentes.

REPONSE

$$\theta = (Z^T Z)^{-1} Z^T Y$$