Master Informatique EID2

Traitement numérique des données

TP5 Descente de gradient

Mohamed Ben Saad
Elias ABDELLI

```python
import argparse
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn.datasets as datasets
import sys
import scipy
```

```python
########################
# Descente de gradient #
########################

#------------------------------------------------------------------------------------------#
## 1. Calculez l'expression analytique de la fonction E(x) = (x − 1)(x − 2)(x − 3)(x − 5) ##
#------------------------------------------------------------------------------------------#

def E(x):
    return (x-1)*(x-2)*(x-3)*(x-5)
```

```python
#-----------#
## Dérivée ##
#-----------#

def dE(x):
    return 4*np.power(x,3) - 33*np.power(x,2) + 82*x -61

```

```python
#--------------------------------------------------------------------#
## 2. Implémentez l'algorithme DG sous Python pour la fonction E(x) ##
#--------------------------------------------------------------------#

def DG(x0,n,e,t):
    x =[x0]
    res= np.array([0,E(x[0])])
    exmin = E(x[0])
    i = 1
    x.append(x[0] - n*dE(x[0]))
    while i < t and abs(x[i] - x[i-1]) > e :
        x.append(abs(x[i] - n*dE(x[i])))
        if exmin > E(x[i]):
            exmin = E(x[i])
        pred = [i,E(x[i])]
        res = np.vstack((res,pred))
        i = i + 1
    print ('Max_iteration =',i)
    print ('Min = ',exmin)
    return res
```

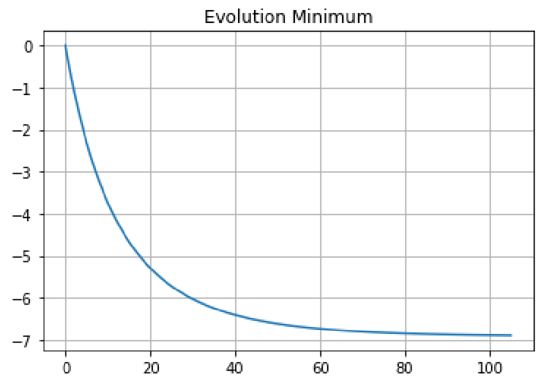
```python
#-------------------------------------------------------------------------------#
##   3. testez l'algorithme implémenté en utilisant des exemples d'exécution   ##
## 4. Affichez le minimum trouvé, ainsi que E(xmin) et le nombre d'itérations ##
#-------------------------------------------------------------------------------#

print('(a) x0 = 5 et η = 0.001')
a = DG(5,0.001,0.001,1000)

print('\n(b) x0 = 5 et η = 0.01')
b = DG(5,0.01,0.001,1000)

print('\n(c) x0 = 5 et η = 0.1')
c = DG(5,0.1,0.001,1000)

print('\n(d) x0 = 5 et η = 0.17')
d = DG(5,0.17,0.001,1000)
```

```

#print('\n(e) x0 = 5 et η = 1')
#e = DG(5,1.0,0.001,1000)

print('\n(f) x0 = 0 et η = 0.001')
f = DG(0,0.001,0.001,1000)

```

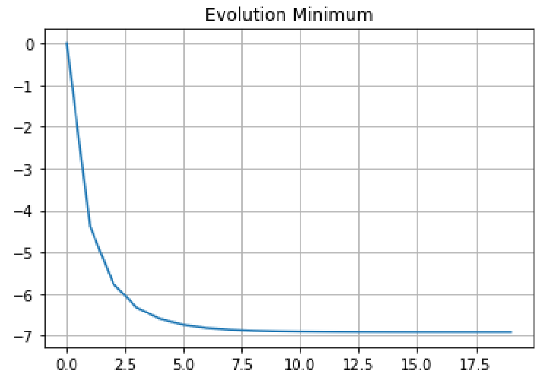
```
(a) x0 = 5 et η = 0.001
Max_iteration = 106
Min =  -6.89196485736797

(b) x0 = 5 et η = 0.01
Max_iteration = 20
Min =  -6.913929319624811

(c) x0 = 5 et η = 0.1
Max_iteration = 1000
Min =  -6.630728959124845

(d) x0 = 5 et η = 0.17
Max_iteration = 1000
Min =  -6.538588969216836

(f) x0 = 0 et η = 0.001
Max_iteration = 151
Min =  -1.349463100975843
```

```
#----------------------------------------------------------------------------------------------#
## 5. Visualisez l'évolution des minimums de la fonction E(x) trouvés au cours des itérations ##
#----------------------------------------------------------------------------------------------#

def afficherDG(res):
    plt.plot(res[:,0],res[:,1])
    plt.title("Evolution Minimum")
    plt.grid()
    plt.show()

print(a[:10])
afficherDG(a)

print(b[:10])
afficherDG(b)
```

```
[[ 0.          0.        ]
 [ 1.         -0.56114808]
 [ 2.         -1.06646858]
 [ 3.         -1.52269704]
 [ 4.         -1.93560652]
 [ 5.         -2.31016646]
 [ 6.         -2.65067188]
 [ 7.         -2.96084912]
 [ 8.         -3.24394284]
 [ 9.         -3.5027879 ]]
```

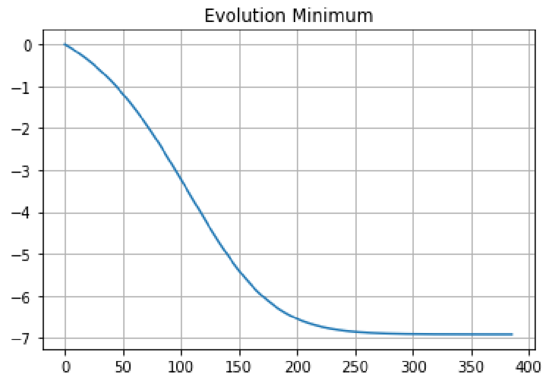


```
[[ 0.          0.        ]
 [ 1.         -4.38349824]
 [ 2.         -5.75937754]
 [ 3.         -6.32923246]
 [ 4.         -6.59920625]
 [ 5.         -6.73772514]
 [ 6.         -6.81259422]
 [ 7.         -6.85454135]
 [ 8.         -6.87865559]
 [ 9.         -6.89278239]]
```




```python
#-------------------------------------------------------------------#
## 6. Testez votre algorithme avec d’autres valeurs de e et nombremax ##
#-------------------------------------------------------------------#
g = DG(3,0.001,0.0001,500)
afficherDG(g)
```

```
Max_iteration = 386
Min =  -6.913865530471172
```

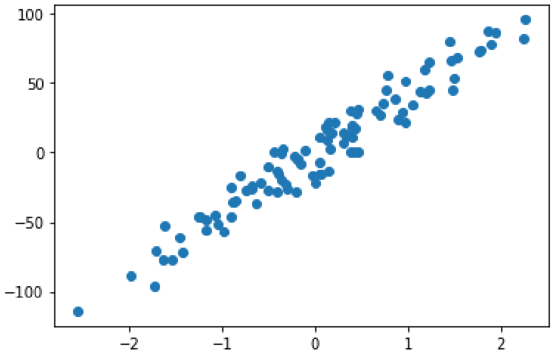



```python
#----------------------------------------------------------------------#
## 1. Calculez les dérivées partielles de la fonction E(a, b) selon a et b ##
#----------------------------------------------------------------------#

def modele(X, O):
    return X.dot(O)

def cout(X, y, O):
    m = len(y)
    return 1/(2*m) * np.sum((modele(X, O) - y)**2)

def grad(X, y, O):
    m = len(y)
    return 1/m * X.T.dot(modele(X, O) - y)

```

```python
#----------------------------------#
## 2. Implémentez l'algorithme DG ##
#----------------------------------#

def DG_R(X, y, n, it):
    c = np.zeros(it)
    np.random.seed(0)
    O = np.random.randn(2,1)
    for i in range(0, it):
        O = O - n * grad(X, y, O)
        c[i] = cout(X, y, O)
    p = modele(X, O)
    print("\tDescente pour n =",n,"et nombreMax=",it)
    plt.figure()
    plt.scatter(x, y)
    plt.plot(x, p, c='r')
    return O, c
```

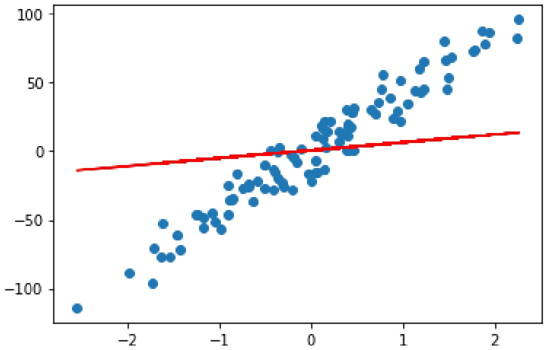
```python
#---------------------------------------------------#
## 3. Importez la fonction datasets.make_regression ##
#---------------------------------------------------#
np.random.seed(0)
x, y = datasets.make_regression(n_samples=100, n_features=1, noise=10)
y = y.reshape(y.shape[0], 1)
X = np.hstack((x, np.ones(x.shape)))
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7f546ab6f898>
```

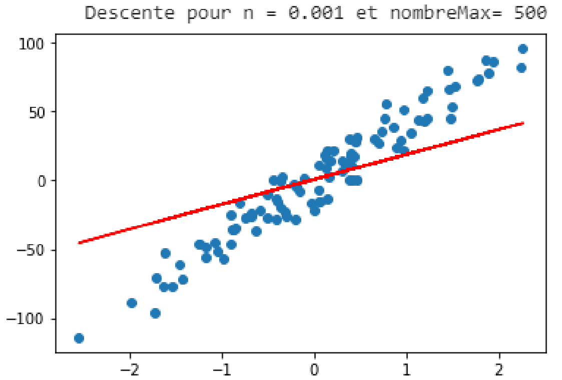



```python
#----------------------------------------#
## 4. Affichez les coefficients trouvés ##
#----------------------------------------#
a,a1 = DG_R(X,y,0.001,100)
```

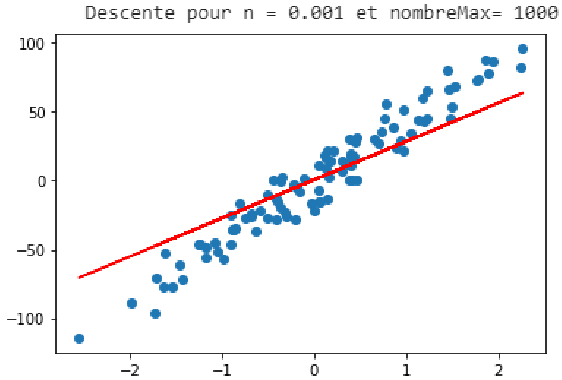
```
	Descente pour n = 0.001 et nombreMax= 100
```



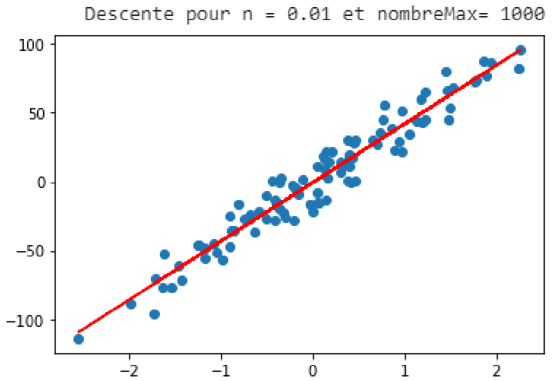
```
1 b = DG_R(X,y,0.001,500)
```



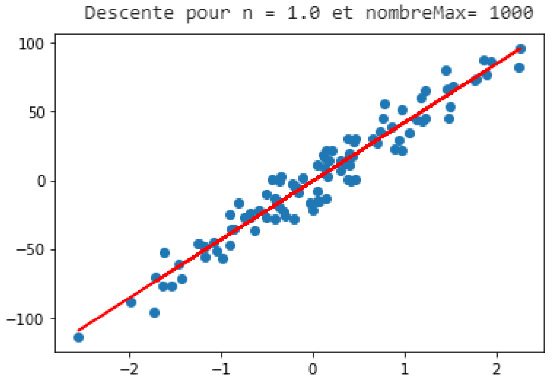

```
1 c = DG_R(X,y,0.001,1000)
```




```
1 d = DG_R(X,y,0.01,1000)
```




```
1 e,e1 =DG_R(X,y,1.0,1000)
```



```
#-----------------------------------------------------#
## 5. Importez la fonction stats.linregress de scipy ##
#-----------------------------------------------------#

#slope, intercept, r_value, p_value, std_err = scipy.stats.linregress(e, e1)

```