Algoritmos - Asier Llano - AG2

July 16, 2023

Actividad Guiada 2

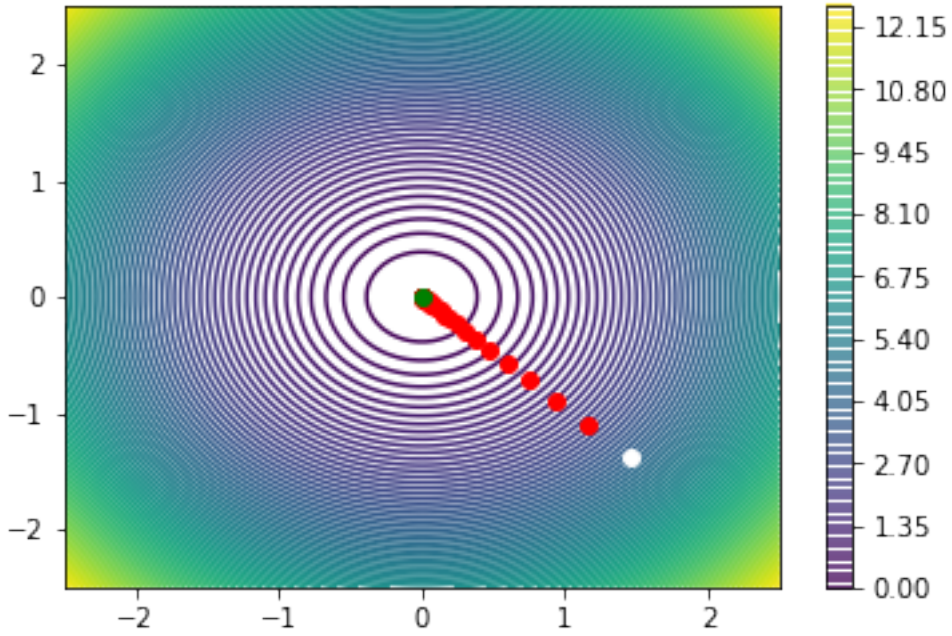
Asier Llano

Link repostorio github: [https://github.com/asierllano/03MIAR—Algoritmos-de-Optimizacion—2023]

1 Descenso de gradiente

[ ]:
```
import math
import matplotlib.pyplot as plt
import numpy as np
import random

# Función y su gradiente
f = lambda X: numpy.array(X[0]**2+X[1]**2)
df = lambda X: numpy.array([2*X[0], 2*X[1]])

# Prepara los datos para dibujar mapa de niveles Z
resolucion = 100
rango=2.5
X=np.linspace(-rango,rango,resolucion)
Y=np.linspace(-rango,rango,resolucion)
Z=np.zeros((resolucion,resolucion))
for ix,x in enumerate(X):
    for iy,y in enumerate(Y):
        Z[iy,ix] = f([x,y])

# Pinta el mapa de niveles de Z
plt.contour(X,Y,Z,resolucion)
plt.colorbar()

# Generamos un punto aleatorio
P=numpy.array([random.uniform(-2,2), random.uniform(-2,2)])
plt.plot(P[0],P[1],"o",c="white")

# Tasa de aprendizaje
```

```
TA=.1

# Iteraciones
for _ in range(500):
    grad = df(P)
    P -= TA*grad
    plt.plot(P[0],P[1],"o",c="red")

# Pintamos la solución
plt.plot(P[0],P[1],"o",c="green")
plt.show()
print(f"Solucion: {P}, {f(P)}")
```

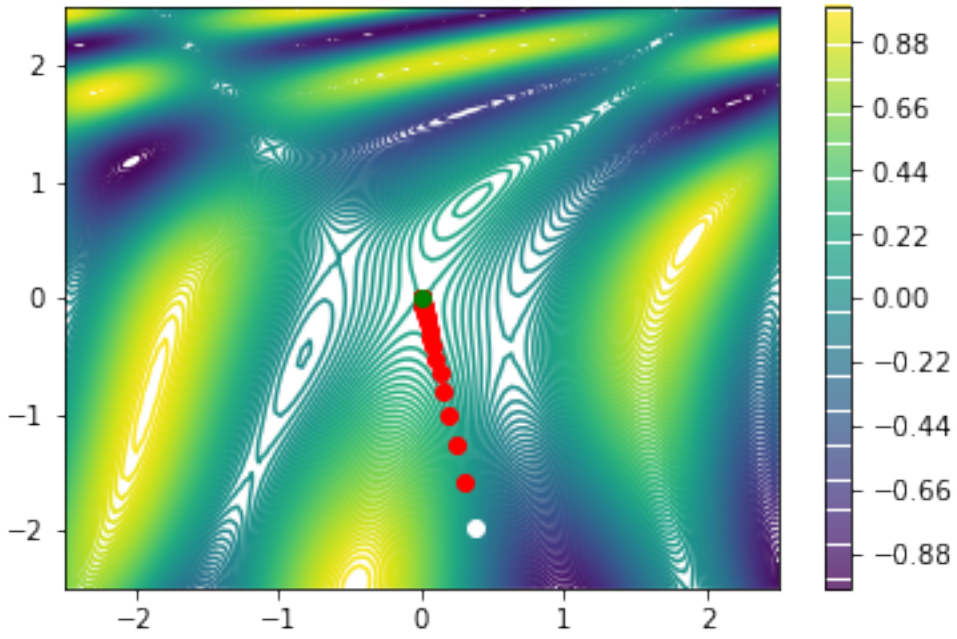



```
Solucion: [ 5.10388316e-49 -4.85779367e-49], 4.964778262532491e-97
```

2 Descenso de gradiente de segunda función tal y como se explica en las transparencias

[ ]:
```
# Función y su gradiente de forma aproximada
f = lambda X: np.sin(1/2*X[0]**2 - 1/4*X[1]**2 + 3)*np.cos(2*X[0] + 1 - np.
  ↪e**X[1])
def dF(PUNTO):
    h=0.01
    T=np.copy(PUNTO)
```

```
    grad = np.zeros(2)
    for it, th in enumerate(PUNTO):
        T[it] = T[it] + h
        grad[it] = (f(T) - f(PUNTO))/h
    return grad

# Prepara los datos para dibujar mapa de niveles Z
resolucion = 100
rango=2.5
X=np.linspace(-rango,rango,resolucion)
Y=np.linspace(-rango,rango,resolucion)
Z=np.zeros((resolucion,resolucion))
for ix,x in enumerate(X):
    for iy,y in enumerate(Y):
        Z[iy,ix] = f([x,y])

# Pinta el mapa de niveles de Z
plt.contour(X,Y,Z,resolucion)
plt.colorbar()

# Generamos un punto aleatorio
Porig=numpy.array([random.uniform(-2,2), random.uniform(-2,2)])
P=Porig.copy()
plt.plot(P[0],P[1],"o",c="white")

# Tasa de aprendizaje
TA=.1

# Iteraciones
for _ in range(500):
    grad = df(P)
    P -= TA*grad
    plt.plot(P[0],P[1],"o",c="red")

# Pintamos la solución
plt.plot(P[0],P[1],"o",c="green")
plt.show()

# Muestra la mejor solucion
print(f"Solucion: {Pbest}, {f(Pbest)}")
```




```
Solucion: [-6.87267151e-49 -6.42373062e-49], 0.1411200080598672
```