Examen Optimisation

Clément Malonda

28 juin 2022

Présentation du problème

Optimiser la fonction suivante :

$$f(\omega) = 418,9829 \times D - \sum_{i=1}^{D} \omega_i \times \sin(\sqrt{|\omega_i|})$$

où D est la dimension du problème. Dans notre cas, nous allons utiliser $D = 1$, $D = 2$ et $D = 10$, avec respectivement

- $\omega = 20$,
- $\omega = [-20, 20]$
- et $\omega = [20, 20, -20, 20, -20, 20, 20, 20, 20, 20]$.

Pour l'ensemble du sujet, j'ai fait le choix d'utiliser l'algorithme de descente de gradient avec momentum.

1 Implémentation de la fonction et de son gradient

```python
def f(w):
    res = 418.9829 * w.shape[0]
    sum = 0
    for i in range(w.shape[0]):
        sum = sum + w[i] * np.sin(np.sqrt(np.abs(w[i])))
    return res - sum
```

```python
def grad(w):
    res = 0
    for i in range(w.shape[0]):
        res = res + (w[i]**2 + np.cos(np.sqrt(np.abs(w[i])))) / (2*np.abs(w[i])**(3/2)) + np.sin(np.sqrt(np.abs(w[i])))
    return res
```

Problème en dimension 1

```
w0 = np.array([20])
lr = 0.7
res_1 = gd2_momentum(w0, grad, lr, max_iter=33)
```

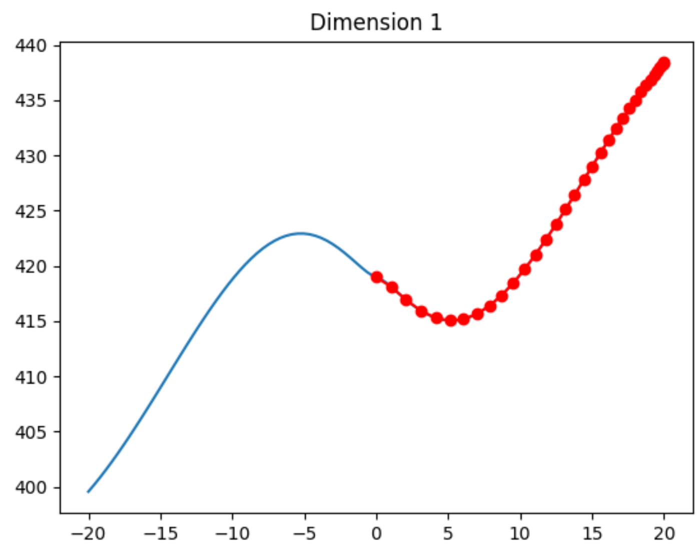

FIGURE 1 – Solution pour un problème de dimension 1

Dans le cas où $D = 1$ la solution arrive en 33 itération avec un learning rate fixé à $0,7$. La figure 1 montre le profil de la fonction f en bleu et les étapes de la solutions en rouge. La solution finale est $w = -0.04144376$.

Problème en dimension 2

```
w0 = np.array([-20, 20])
lr = 0.6
res_2 = gd2_momentum(w0, grad, lr, max_iter=22)
```

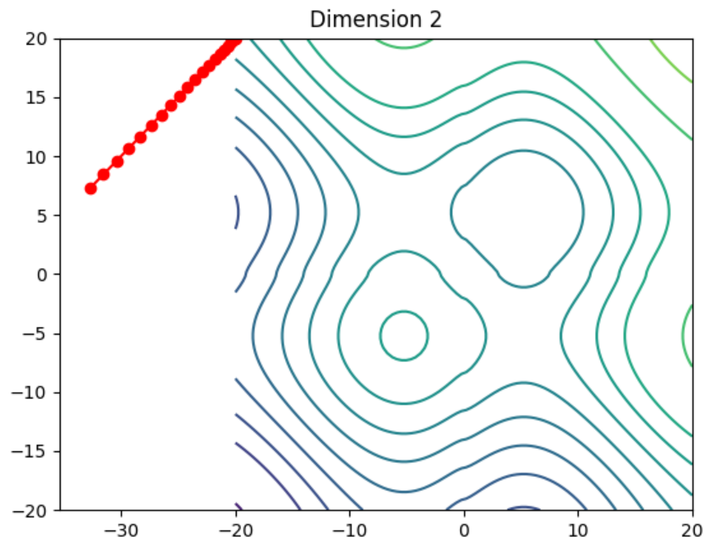

FIGURE 2 – Solution pour un problème de dimension 2

Dans le cas où $D = 2$ la solution semble diverger avec learning rate fixé à $0,7$ et 22 itérations. La figure 2 montre le profil de la fonction f sous forme de courbe de niveau et les étapes de la solutions en rouge. La solution finale est $w = [-39.1737414, 0.8262586]$.

Nous pouvons constater que la solution diverge, la valeur ω_2 semble tendre vers 0 mais ω_2 non.

Dans le cas d'un point de départ en $\omega = [-20, 20]$ alors on converge en 23 itération avec le même learning rate (cf. Figure 3).

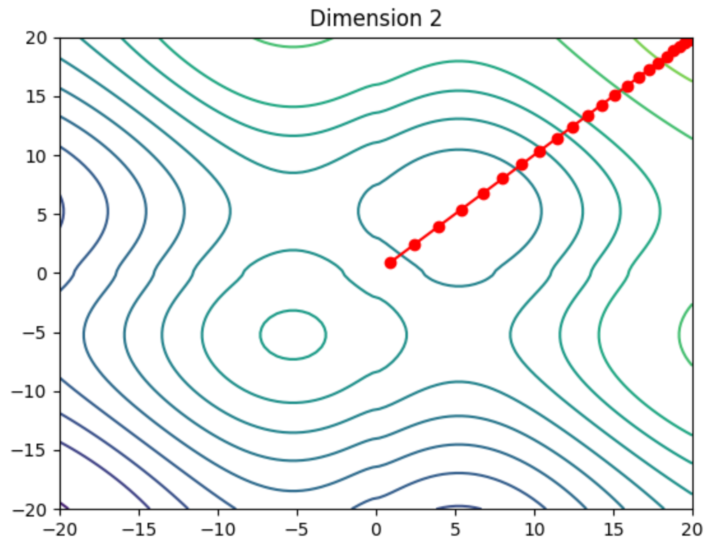

FIGURE 3 – Autre test pour le problème de dimension 2

Problème en dimension 10

```
w0 = np.array([20, 20, -20, 20, -20, 20, 20, 20, 20, 20])
lr = 0.1
res_2 = gd2_momentum(w0, grad, lr, max_iter=10)
```

w = [16.52988674, 16.52988674, -23.47011326, 16.52988674 -23.47011326 , 16.52988674, 16.52988674, 16.52988674, 16.52988674, 16.52988674]

Fonction de gradient avec momentum

```python
def gd2_momentum(x, grad, alpha, beta=0.9, max_iter=10):
    xs = np.zeros((1 + max_iter, x.shape[0]))
    xs[0, :] = x
    v = 0
    for i in range(max_iter):
        v = beta*v + (1-beta)*grad(x)
        vc = v/(1+beta**(i+1))
        x = x - alpha * vc
        xs[i+1, :] = x
    return xs
```