Minimisation par Gradient descent

Dr F. Moreau

03/07/2024

1 Introduction

Cette technique de minimisation a pour objectif de déterminer la position d'un minimum local d'un fonction. Elle est largement la plus utilisée dans le contexte du « Machine Learning ». Elle fait partie des méthodes qui utilisent les dérivées premières de la fonction à minimiser. Elle est relativement simple à mettre en oeuvre dans sa version de base et s'adapte facilement à des fonctions de plusieurs millions de variables.

2 Minimisation de fonctions d'une variable

Dans cette section, nous cherchons à minimiser une fonction d'une seule variable $f : \mathbb{R} \to \mathbb{R}$, appelée la fonction objectif.

Si la dérivée de cette fonction est positive en un point, la fonction est croissante en ce point et il faut donc se déplacer à gauche pour se rapprocher du minimum local. Si la dérivée est négative, la fonction est décroissante et il faut se déplacer à droite. Pour faire un petit pas vers le minimum local, on utilise alors la relation de récurrence

$$x_{n+1} = x_n - \alpha f'(x_n) \tag{1}$$

Où α est une constante positive. Cette constante est appelée « learning rate » dans le contexte du machine learning. Si α est choisit trop grand, x_{n+1} peut se retrouver au-delà du minimum local. La relation de récurrence va alors zigzager de part et d'autre du minimum, se brancher sur un autre minimum local ou même diverger (voir figure 1). Si *alpha* est trop petit, la relation de récurrence va converger vers le minimum local mais très lentement.

On peut se poser la question du α optimum. Si on se situe déjà à proximité d'un minimum local, cet α optimal peut être défini. Considérons l'approximation quadratique

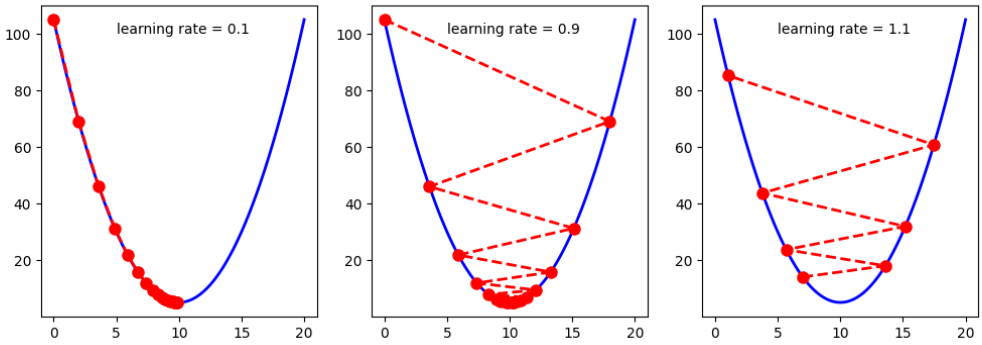

FIGURE 1 – Comparaison de la convergence pour différents learning rates pour la fonction $f(x) = (x-10)^2 + 5$. Pour le panneau de droite, même en partant d'une abscisse proche du minimum, l'itération diverge.

de la fonction objectif autour du minimum x_m :

$$f(x) \cong f(x_m) + \frac{1}{2} f''(x_m)(x - x_m)^2 \tag{2}$$

La dérivée de f est alors approchée par

$$f'(x) \cong f''(x_m)(x - x_m)$$

Introduisons cela dans la relation de récurrence 1, nous obtenons

$$x_{n+1} \cong x_n - \alpha f''(x_m)(x_n - x_m)$$

Si on veut que $x_{n+1} \cong x_m$, il faut utiliser

$$\alpha_{\text{opt}} = \frac{1}{f''(x_m)}$$

Evidemment, la dérivée seconde au minimum est normalement inconnue et il n'est pas simple de savoir si on se trouve proche d'un minimum. Pour cet algorithme, nous n'essayons pas d'évaluer cet α optimal. Nous tentons un α assez important pour débuter. Nous vérifions si le nouveau point a une ordonnée plus faible que le précédent. Si ce n'est pas le cas, on n'utilise pas ce nouveau point, on refait l'itération avec un α deux fois plus faible. Le α est divisé par jusqu'à ce que le nouveau point soit plus bas que le précédent. Voici le code pour une iteration de gradient descent :

```
def grad_iter(x, p, fct, fct_der):
    """ One iteration of the gradient descent algorithm
     'p' is the learning rate. Return the new position est the learning rate
         used. """
    fx = fct(x)
    while True:
        x_try = x - p * fct_der(x)
        if fct(x_try) <= fx:
            break
        else:
            p /= 2.

    return x_try, p
```

Reprenons l'approximation quadratique 2 à proximité du minimum local. La dérivée $f'(x) \approx f''(x_m)(x - x_m)$ où x_m est la position exacte du minimum local. Si on introduit ceci dans la relation de récurrence 1, on obtient

$$x_{n+1} - x_n = (x_m - x_n)\alpha f''(x_m)$$

On cherche à ce que $|x_m - x_n|$ soit inférieur à un certain seuil TOL. Si la dérivée seconde est inconnue et que l'on peut supposer qu'elle n'est pas très éloignée de l'unité, on peut fixer la condition d'arrêt à $|x_{n+1} - x_n| < \alpha TOL$. Le code qui gère la convergence est alors

```
def grad_descent(x0, pas_grad, fonc, fonc_der):
    # gradient-descent in 1D with the initial learning rate 'pas_grad'.
    x = x0
    x_new, pas_grad = grad_iter(x, pas_grad, fonc, fonc_der)
    nit = 1
    while abs(x_new - x) > TOL*pas_grad:
        x = x_new
        x_new, pas_grad = grad_iter(x, pas_grad, fonc, fonc_der)
        nit += 1

    return x_new, fonc(x_new), nit
```

Si on suspecte la dérivée seconde à l'origine d'être très différente de l'unité, on peut l'estimer par différences finies à partir des dernières valeurs de l'itération. Bien entendu, la position du minimum étant inconnue nous estimons la dérivée seconde en x_n

$$
\begin{aligned}
f''(x_n) &\cong \frac{f'_{n+1/2} - f'_{n-1/2}}{x_{n+1/2} - x_{n-1/2}} \\
&= \frac{2(f'_{n+1/2} - f'_{n-1/2})}{x_{n+1} - x_{n-1}} \\
&\cong \frac{2}{x_{n+1} - x_{n-1}} \left(\frac{f_{n+1} - f_n}{x_{n+1} - x_n} - \frac{f_n - f_{n-1}}{x_n - x_{n-1}} \right) \\
&= \frac{2\left(h_{n+1} f_{n-1} + h_n f_{n+1} - (h_n + h_{n+1}) f_n\right)}{h_n h_{n+1}(h_n + h_{n+1})}
\end{aligned}
$$

où $f_i = f(x_i)$n $f'_{i+1/2}$ est la dérivée évaluée entre les abscisses x_i et x_{i+1} et $h_i = x_i - x_{i-1}$. Cette évaluation alourdit énormément l'algorithme. Il est préférable de n'évaluer la dérivée seconde qu'une fois toutes les 10 itérations par exemple. De plus, la dérivée seconde étant elle-même entachée d'une erreur au moins égale à $3\,10^{-8}$, il fait doubler la valeur de TOL dans la condition d'arrêt.

3 Minimisation de fonctions de plusieurs variables

La technique est très simple à généraliser aux fonctions $f : \mathbb{R}^n \to \mathbb{R}$. Nous notons $\vec{r}$ la position dans l'espace $\mathbb{R}^n$ des variables de f. Le gradient de cette fonction, $\vec{\nabla} f$, est orienté dans la direction de plus grande croissance de f. A chaque pas, il suffit donc de se déplacer dans le sens opposé au gradient

$$
\vec{r}_{n+1} = \vec{r}_n - \alpha \vec{\nabla}_{\vec{r}_n} f \tag{3}
$$

Ici, plus encore qu'à une dimension, il est illusoire de vouloir utiliser les dérivées secondes pour estimer le pas optimal ou le condition d'arrêt. L'algorithme est habituellement stoppé après une nombre d'itérations déterminé à l'avance.

Pour une fonction $z = f(x, y)$ de $\mathbb{R}^2 \to \mathbb{R}$, l'équation vectorielle 3 donne les deux équations scalaires

$$
\begin{aligned}
x_{n+1} &= x_n - \alpha \frac{\partial f(x_n, y_n)}{\partial x} \\
y_{n+1} &= y_n - \alpha \frac{\partial f(x_n, y_n)}{\partial y}
\end{aligned}
$$

Sur la figure 2, on voit la fonction $f(x, y) = cos(x) + sin(2y)$, le chemin parcouru par les premièrres itérations de gradient descent et les gradients à chaque position. La position initiale est $(1, 0.5)$ et le learning rate est $\alpha = 0.2$.

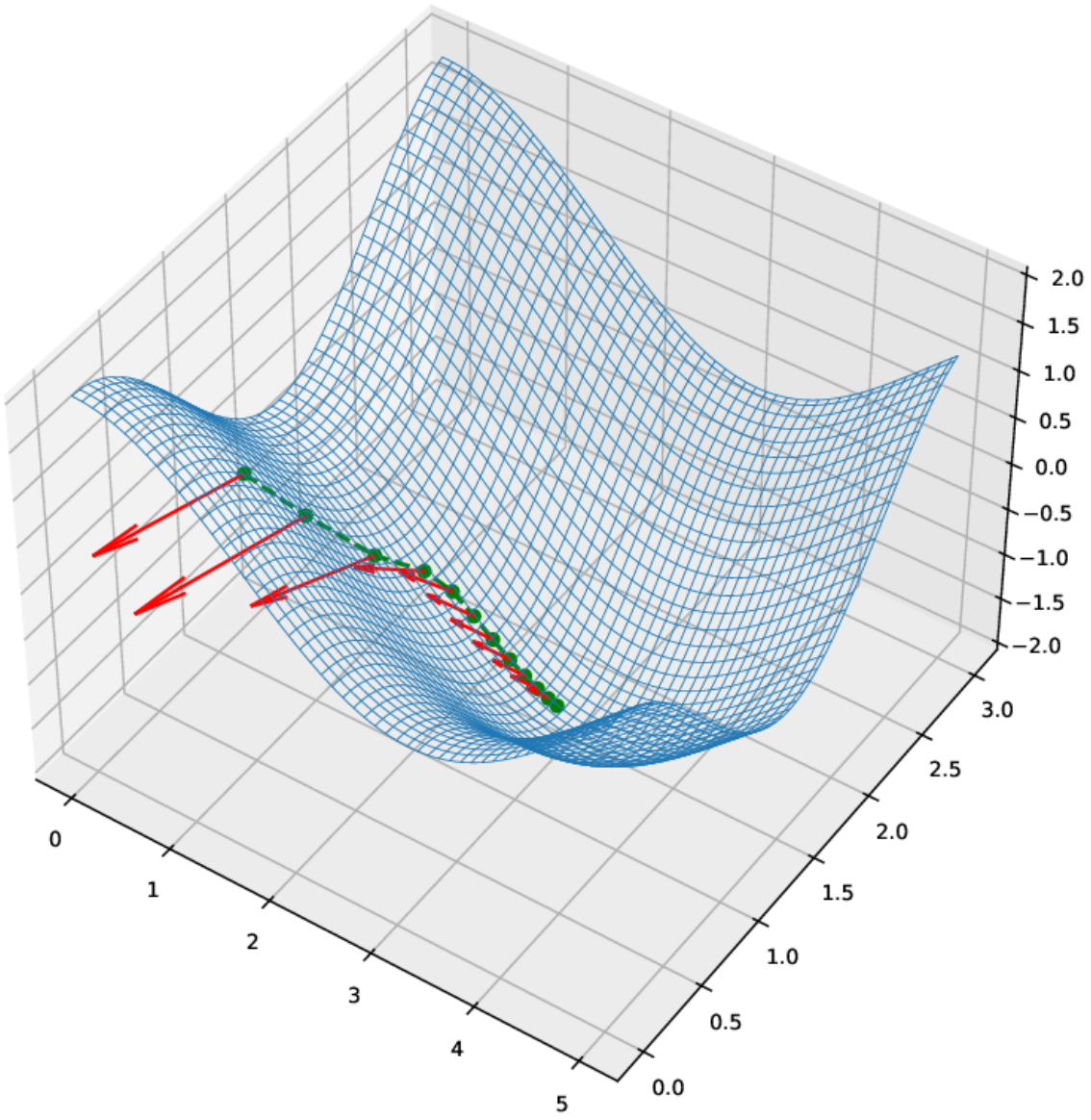

FIGURE 2 – Chemin de convergence vers un minimum local. A chaque point de l'itération le gradient est dessiné en rouge.