

ALGORITHME DE DESCENTE DE GRADIENT

La méthode de descente de gradient constitue l'**outil fondamental** sur lequel repose les réseaux de neurones.

1 Rappels de résultats mathématiques

Dans le cours de calcul différentiel, on a montré une condition nécessaire pour être un minimum : il faut que la dérivée (ou le gradient en dimension supérieure) s'annule en ce point.

Proposition 1.1 (*Condition nécessaire d'existence pour une fonction d'une variable*)

Soit $f : \mathbb{R} \rightarrow \mathbb{R}$ une fonction que l'on suppose de classe $\mathcal{C}^1(\mathbb{R})$.

$$(f \text{ admet un minimum local en } x_0) \Rightarrow (x_0 \text{ est un point critique : } f'(x_0) = 0)$$

Proposition 1.2 (*Condition nécessaire d'existence pour une fonction de plusieurs variables*)

Pour $D > 1$, soit $f : \mathbb{R}^D \rightarrow \mathbb{R}$ une fonction que l'on suppose de classe $\mathcal{C}^1(\mathbb{R}^D)$.

$$(f \text{ admet un minimum local en } x_0) \Rightarrow (x_0 \text{ est un point critique : } \nabla f(x_0) = 0)$$

Définition 1.3 (*Problème d'optimisation*)

On appelle **problème d'optimisation** d'une fonction f la recherche du **point x^* réalisant son minimum**, dont on suppose l'existence. Le point x^* vérifie :

$$x^* = \underset{x \in \mathbb{R}^D}{\operatorname{argmin}} f(x)$$

Remarque 1

Il faut bien noter que :

- ✓ x^* est l'argument minimum, le point où f réalise son minimum.
- ✓ $f(x^*)$ est le minimum atteint par f .



Point Méthode : Recherche du minimum par recherche d'un point critique



L'algorithme de descente de gradient va donc chercher à obtenir le minimum de f en cherchant un point critique x vérifiant $f'(x) = 0$.
Il n'y a cependant aucune garantie que ce point x soit l'argument minimal x^* car la réciproque dans les Prop 1.1 et Prop 1.2 est **fausse en général**.

2 Descente de gradient pour le minimum d'une fonction

Dans cette section, nous allons présenter l'algorithme de descente de gradient qui permet d'obtenir un point critique x tel que $f'(x) = 0$, candidat à réaliser le minimum de f .

2.1 Fonction réelle

Soit $f : \mathbb{R} \rightarrow \mathbb{R}$ supposée de classe \mathcal{C}^∞ sur \mathbb{R} .

Pour $x \in \mathbb{R}$ et $h \in \mathbb{R}$, on a le développement limité en f :

$$f(x+h) = f(x) + hf'(x) + o(h)$$

En remplaçant h par $\alpha f'(x)$, ce développement limité devient :

$$f(x + \alpha f'(x)) = f(x) + \alpha(f'(x))^2 + o(\alpha)$$

- ✓ Si $f'(x) = 0$, alors x est le point critique recherché par l'algorithme, donc on ne cherche pas à modifier x .
- ✓ Si $f'(x) \neq 0$, alors, en prenant α suffisamment petit pour que le o n'influe pas sur le résultat, on a, pour $\alpha > 0$:

$$f(x + \alpha f'(x)) = f(x) + \underbrace{\alpha(f'(x))^2}_{>0} + o(\alpha) > 0$$

Ainsi, en ajoutant $\alpha f'(x)$ à x , on a **augmenté la valeur de f** .

Au contraire, en prenant $\alpha = -\beta < 0$ (donc $\beta > 0$) :

$$f(x - \beta f'(x)) = f(x) - \underbrace{\beta(f'(x))^2}_{>0} + o(\alpha) < 0$$

Ainsi, en soustrayant $\beta f'(x)$ à x , on a **diminué la valeur de f** . On s'est donc **rapproché du minimum de f** .



Point Méthode : Algorithme de descente de gradient pour une fonction réelle

L'algorithme de descente de gradient pour une fonction $f : \mathbb{R} \rightarrow \mathbb{R}$ est :

1. Choisir un point $x_0 \in \mathbb{R}^D$. (Typiquement aléatoirement ou une valeur simple).
2. Calcul itératif de la suite (x_n) :

$$\forall n \geq 0, x_{n+1} = x_n - lr \cdot f'(x_n)$$

3. Critère d'arrêt : un nombre d'itération choisi ou une valeur cible pour la condition d'arrêt atteinte.

Ce qui peut se résumer schématiquement par :

$$x_0 \rightarrow f'(x_0) \rightarrow x_1 \rightarrow f'(x_1) \rightarrow \cdots \rightarrow x_n \rightarrow f'(x_n) \rightarrow x_{n+1} \rightarrow \cdots$$

2.2 Fonction de $D \geq 2$ variables

Pour une fonction $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, il est possible d'écrire les mêmes calculs que dans la partie précédente en remplaçant les $f'(x)$ par des $\nabla f(x, y)$:

$$f(x + h, y + k) = f(x, y) + \begin{pmatrix} h \\ k \end{pmatrix} \cdot \nabla f(x, y) + o(\|(h, k)\|)$$



Point Méthode : Algorithme de descente de gradient pour une fonction de plusieurs variables

L'algorithme de descente de gradient pour une fonction $f : \mathbb{R}^D \rightarrow \mathbb{R}$ est :

1. Choisir un point $x_0 \in \mathbb{R}^D$. (Typiquement aléatoirement ou une valeur simple).
2. Calcul itératif de la suite (x_n) , pour un pas de descente $lr \neq 0$ fixé :

$$\forall n \geq 0, x_{n+1} = x_n - lr \cdot \nabla f(x_n)$$

3. Critère d'arrêt : un nombre d'itération choisi ou une valeur cible pour la condition d'arrêt atteinte.

Ce qui peut se résumer schématiquement par :

$$x_0 \rightarrow \nabla f(x_0) \rightarrow x_1 \rightarrow \nabla f(x_1) \rightarrow \cdots \rightarrow x_n \rightarrow \nabla f(x_n) \rightarrow x_{n+1} \rightarrow \cdots$$

2.3 Condition d'arrêt

Dans le cas où l'on connaît le minimum, il suffit de tester la distance du point x_n à la valeur finale x^* .

Proposition 2.1 (*Condition d'arrêt pour un minimum connu*)

Soit $f : \mathbb{R}^D \rightarrow \mathbb{R}$ une fonction $\mathcal{C}^1(\mathbb{R})$ admettant un minimum en x^* .
Une condition d'arrêt valide est, pour une tolérance $\varepsilon > 0$ fixée :

$$|x_n - x^*| \leq \varepsilon$$

Dans le cas général, on ne connaît pas x^* puisque c'est la valeur qu'on cherche à déterminer. Comme la descente de gradient cherche à diminuer f en cherchant un point critique, nous pouvons prendre comme critère d'arrêt que la dérivée (ou le gradient) est proche de 0

Proposition 2.2 (*Condition d'arrêt pour un minimum inconnu*)

Soit $f : \mathbb{R}^D \rightarrow \mathbb{R}$ une fonction $\mathcal{C}^1(\mathbb{R})$ admettant un minimum en une valeur inconnue x^* . En utilisant la Prop 1.2, on sait qu'en ce point :

$$\nabla f(x^*) = 0$$

On choisit alors pour critère d'arrêt, pour une tolérance $\varepsilon > 0$ fixée :

$$|\nabla f(x_n)| < \varepsilon$$

Remarque 2 (*Nombre maximal d'itérations*)

Il est indispensable de mettre un nombre maximum d'itérations pour que notre algorithme termine.

1. La fonction f considérée n'a peut être pas de point critique.
2. La convergence peut être arbitrairement longue sans étude a priori de la fonction f .

3 Descente de gradient pour l'optimisation de paramètres

Dans cette partie, nous supposons avoir N points de **données** qui sont issus d'une même expérience caractérisée par des **paramètres**. Étant donnée une **fonction d'erreur**, on cherche à déterminer, par descente de gradient, les paramètres optimaux.

Définition 3.1 (*Formalisation du problème*)

1. Des données formées de N points de dimension D stockés dans une matrice $X \in \mathcal{M}_{(N,D)}(\mathbb{R})$. Chaque point correspond à une ligne et chaque colonne correspond à une dimension.
2. Des paramètres $(a_{sol}, b_{sol}, c_{sol}, \dots)$ qui caractérisent les données, typiquement leur position dans l'espace \mathbb{R}^D . Dans la suite de cette définition, on notera (a, b, c) les paramètres mais il peut y en avoir un nombre quelconque.
3. Une fonction d'erreur $E(X, (a, b, c))$ vérifiant les propriétés suivantes :

$$\begin{cases} \forall (a, b, c) \in \mathbb{R}^3, E(X, (a, b, c)) \geq 0 \\ E(X, (a_{sol}, b_{sol}, c_{sol})) = \min_{(a,b,c)} E(X, (a, b, c)) = 0 \end{cases}$$

L'**objectif** est de déterminer les paramètres du problème $(a_{sol}, b_{sol}, c_{sol})$ en optimisant la fonction E et en utilisant les données X .

Exemple 1 : Recherche de la droite optimale

On considère que N points de plans sont répartis sur une droite $D_{a_{sol}, b_{sol}}$ définie par l'équation :

$$y = a_{sol}x + b_{sol}$$

Les points sont stockés dans la matrice X à N lignes (chaque point correspond à une ligne) et 2 colonnes (les abscisses x_i dans la 1ère colonne, les ordonnées y_i dans la 2ème).

$$X = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_N & y_N \end{pmatrix} \in \mathcal{M}_{N,2}(\mathbb{R})$$

L'objectif est donc de déterminer le couple de paramètres (a_{sol}, b_{sol}) déterminant la droite $D_{a_{sol}, b_{sol}}$ à partir des données X .

La descente de gradient va donc porter sur les paramètres (a, b) et la fonction que nous allons optimiser est la fonction d'erreur ℓ^2 , notée E :

$$E(a, b, X) = \sum_{i=1}^N (y_i - (ax_i + b))^2$$

Cette fonction compare :

- ✓ La valeur estimée de l'ordonnée du point d'abscisse x_i par le modèle de paramètre (a, b) : $ax_i + b$.
- ✓ La valeur réelle de l'ordonnée du point : y_i .

Ainsi, le minimum de cette fonction E est atteint pour $(a, b) = (a_{sol}, b_{sol})$, c'est-à-dire les valeurs solution du modèle. La descente de gradient va donc créer une suite $(a_n, b_n)_n$ de paramètres en partant de paramètres aléatoires (par exemple $(a_0, b_0) = (1, 1)$) et convergeant vers (a_{sol}, b_{sol}) .

Remarque 3

1. La descente de gradient porte donc sur les **paramètres** et non sur les **données**.
2. Il s'agit d'un problème d'**apprentissage supervisé** car on **suppose a priori** le comportement du modèle. Cette supposition se matérialise à travers les paramètres (dans l'exemple, le modèle est supposé être une droite $y = ax + b$). Cela s'oppose à un problème d'apprentissage non supervisé où l'algorithme s'exécute sans **supposition a priori** sur le modèle.



Point Méthode : Algorithme de descente de gradient pour l'optimisation de paramètres

Pour un problème portant sur les données X , les paramètres (a, b) et une fonction d'erreur $E(X, (a, b))$, la descente de gradient pour optimiser (a, b) est :

1. Choix de (a_0, b_0) . (Typiquement aléatoirement ou une valeur simple).
2. Calcul itératif des suites (a_n, b_n) et (E_n) :

$$\forall n \geq 0 \begin{cases} \nabla E_n = \nabla E(X, (a_n, b_n)) \\ E_n = E(X, (a_n, b_n)) \\ \begin{pmatrix} a_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} a_n \\ b_n \end{pmatrix} - lr \cdot \nabla E_n \end{cases}$$

3. Critère d'arrêt : un nombre d'itération choisi ou une valeur cible pour l'erreur E .
4. Affichage de l'évolution de E_n pour évaluer les performances de l'algorithme.

Cela repose donc sur l'alternance :

- ✓ Obtention de $E_n = E(X, (a_n, b_n))$ et $\nabla E_n = \nabla E(X, (a_n, b_n))$ en fonction de (a_n, b_n) .
- ✓ Obtention de (a_{n+1}, b_{n+1}) en fonction de $E_n = E(X, (a_n, b_n))$

Ce qui peut se résumer schématiquement par :

$$(a_0, b_0) \rightarrow \nabla E_0 \rightarrow (a_1, b_1) \rightarrow \nabla E_1 \rightarrow \dots \rightarrow (a_n, b_n) \rightarrow \nabla E_n \rightarrow (a_{n+1}, b_{n+1}) \rightarrow \dots$$

Descente de gradient classique et stochastique

Dans toute cette section, on notera :

- ✓ X la matrice des données de taille $(N \times D)$.
- ✓ (a, b) les paramètres du problème.
- ✓ E la fonction d'erreur.

Définition 3.2 (Fonction d'erreur totale et fonctions d'erreurs partielles)

Dans tous les problèmes considérés, on utilise l'intégralité des données à disposition dans X . Cela signifie que la fonction E dépend de **toutes les lignes** X_i de X . On peut alors découper la fonction E comme suit :

$$E(X, (a, b)) = \sum_{i=1}^N E_i(X_i, (a, b))$$

- ✓ On appelle $E(X, (a, b))$ la **fonction d'erreur totale**. Elle mesure la qualité des paramètres (a, b) par rapport à l'ensemble des données.
- ✓ On appelle $E_i(X, (a, b))$ les **fonctions d'erreurs partielles**. Chaque E_i mesure la qualité des paramètres (a, b) par rapport à la donnée X_i .

Définition 3.3 (Époque)

On appelle **époque** d'entraînement l'ensemble des opérations faites pour **traiter les données** X **une fois intégralement**.

Cela signifie qu'au bout d'une époque, chaque point de données a influé une fois et une seule sur les directions de descente dans l'algorithme.

Définition 3.4 (Descente de gradient classique)

On appelle **descente de gradient classique** l'optimisation des paramètres par rapport à la fonction E prise dans sa globalité. Une étape de descente est donc faite dans la direction :

$$-\nabla E(X, (a_n, b_n))$$

Pour la **méthode classique**, une étape de descente correspond à une époque :

$$\begin{pmatrix} a_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} a_n \\ b_n \end{pmatrix} - \text{lr} \cdot \nabla E(X, (a_n, b_n))$$

Définition 3.5 (Descente de gradient stochastique)

On appelle **descente de gradient stochastique** l'optimisation des paramètres par rapport aux fonctions E_i prise les unes après les autres. Une étape de descente est donc faite dans une direction :

$$-\nabla E_i(X, (a_n, b_n))$$

Pour la **méthode stochastique**, une étape de descente correspond à un pas par rapport à une donnée. Une époque correspond donc à N étapes de descente (une pour chaque donnée) :

$$\begin{pmatrix} a_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} a_n \\ b_n \end{pmatrix} - \text{lr} \cdot \nabla E_i(X, (a_n, b_n))$$