Piscine Machine Learning | Jour 4

Fonction de coût,
dérivées, dérivées partielles
et dérivées en chaîne

Nom de repo: ml_d04

Droits: florian.bacho@epitech.eu, voravo_d, girard_z

Langage : Python

Introduction:

Maintenant que vous avez un réseau de neurone fonctionnel, il faut lui faire apprendre à partir d'un dataset. On peut traduire l'apprentissage comme un processus qui vise à maximiser les bonnes réponses. Or, maximiser les bonnes réponses signifie aussi minimiser les erreurs commises !

Le travail d'apprentissage va donc s'apparenter à un travail d'optimisation où l'on va chercher à optimiser l'erreur de façon à ce qu'elle soit à son minima.

L'erreur du réseau de neurone va être caractérisée par ce que l'on appelle une « fonction de coût » ou « fonction de perte ». On la note généralement : $Loss(y_t, h_w(x_t))$ où x représente les entrées du réseau de neurone, y les sorties attendues et h(x) les sorties calculées par les neurones en fonction de l'entrée.

Il existe plusieurs fonctions de coût possibles. Cependant, nous allons nous intéresser à la fonction cross-entropy définie selon la formule suivante:

$$Loss(x, y, w) = -\sum_{i=0}^{k} (y_i \log(h_{wi}(x)) + (1 - y_i) \log(1 - h_{wi}(x)))$$

où :

- x représente les entrées de l'exemple;

- y représente la sortie attendue de l'exemple;

- w les poids synaptiques du neurone;

- k le nombre de sortie;

- i le $i^{ème}$ neurone de sortie;

- $h_{wi}(x)$ la $i^{ème}$ sortie calculée du réseau de neurones avec les entrées x.

Ex00 :

Fichier: ex00.txt

Soit la sortie attendue: y = 1 et la sortie du neurone: h(x) = 0.2

Calculer le coût du neurone.

> ! La base de log est e

Ex01 :

Fichier: ex01.txt

Soit la sortie attendue: y = [0; 1] et la sortie du réseau de neurones: h(x) = [0.8; 0.4]

Calculer le coût du réseau de neurones.

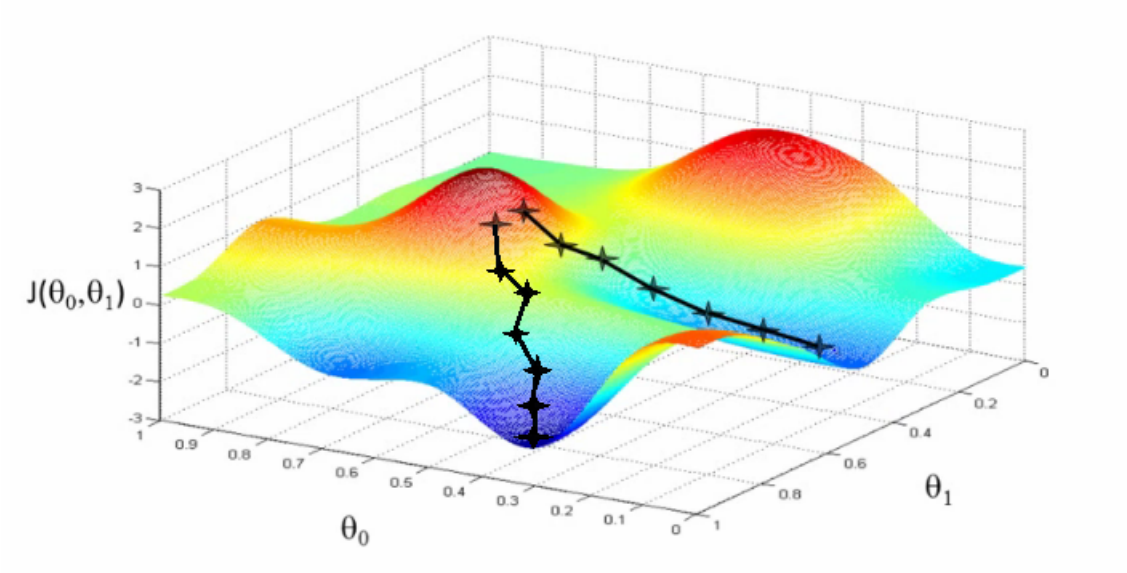
> ! La base de log est e

Ex02 :

Fichier: Neuron.py

Reprenez votre fichier Neuron.py pour y ajouter une fonction *calcLoss* qui prend en paramètres un dataset. Cette fonction retourne la moyenne des coûts du neurone sur le dataset.

Voici les variations du coût en fonction des poids d'un neurone à 1 entrée (1 poids + 1 biais):



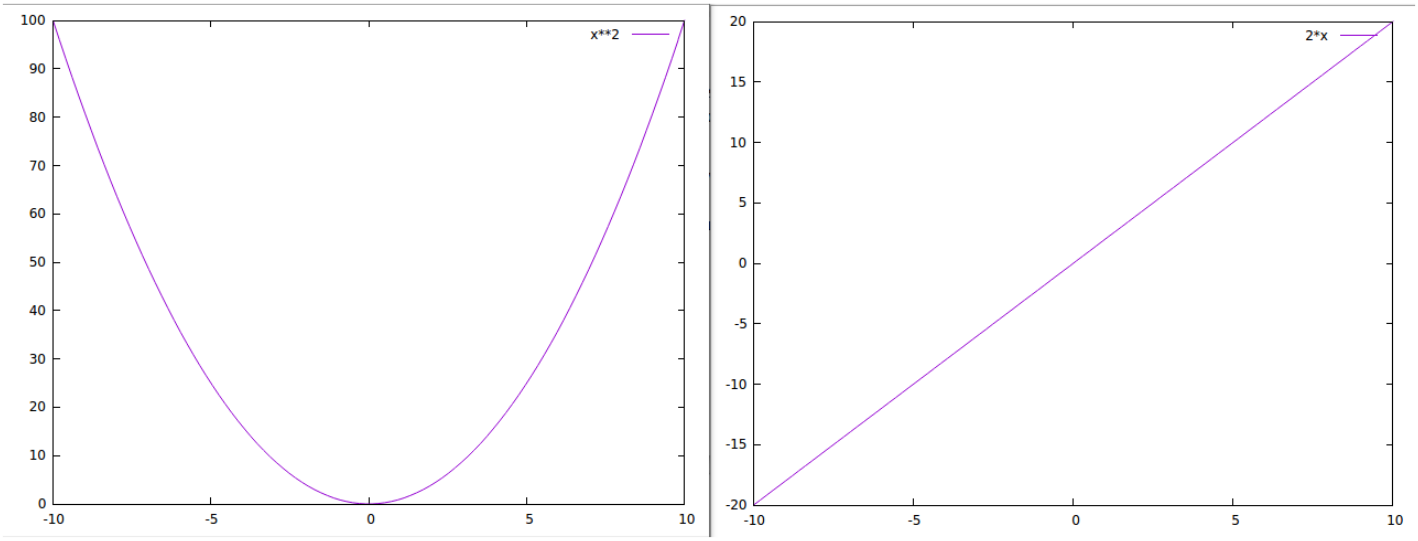
On remarque qu'en fonction des paramètres θ_0 et θ_1 , le coût est plus ou moins élevé. L'objectif de notre apprentissage est de trouver les bons paramètres pour avoir le coût le plus faible (représenté en bleu sur le graphique).

Pour se faire, on va effectuer des petits sauts en direction de la pente (en ajustant les paramètres) afin de descendre l'erreur. On appelle cette méthode « descente de gradient ». Le gradient est un vecteur qui indique la direction dans laquelle déplacer nos paramètres.

Intuition de la dérivée:

La dérivée d'une fonction en un point est le coefficient directeur de la tangente de cette fonction en ce point.

Autrement dit, la dérivée d'une fonction représente la « pente » de cette fonction: comment elle évolue en fonction de ses paramètres. Si la fonction augmente lorsque son paramètre augmente, sa dérivée est positive. À l'inverse, si la fonction diminue lorsque son paramètre augmente, sa dérivée est négative. Si la fonction ne varie pas en fonction de ses paramètres, sa dérivée est 0.



Fonction x^2 (à gauche) et sa dérivée (à droite)

On note f'(x) la dérivée de la fonction f(x).

Ex03 :

Fichier: ex03.txt

Calculer la dérivée de $f(x) = 4x^3$

Ex04 :

Fichier: ex04.txt

Calculer la dérivée de $f(x) = 4 \log(x) – 7x^2$

> ! La base de log est e

Ex05 :

Fichier: ex05.txt

Mais comment peut-on faire pour dériver une fonction ayant plusieurs paramètres (comme nos neurones, qui possèdent plusieurs poids) ?

Exemple: f(x, y) = 3x + 4y

Tout simplement, on va effectuer ce que l'on appelle une « dérivée partielle ». C'est-à-dire que l'on va dériver selon un des paramètres et considérer les autres comme des constantes. Ainsi, l'ensemble des dérivées partielles d'une fonction par rapport à chaque paramètre se nomme « gradient » (vous voyez le lien ?).

Les dérivées partielles de la fonction f(x, y) par rapport à x et y se notent respectivement:

$\frac{\partial f(x,y)}{\partial x}$ et $\frac{\partial f(x,y)}{\partial y}$

Calculer les dérivées partielles $\frac{\partial f(x,y)}{\partial x}$ et $\frac{\partial f(x,y)}{\partial y}$ où $f(x,y)=4x^2+7xy^4-x^{-3}$

Ex06 :

Fichier: ex06.txt

Jusqu'à maintenant nous vous avons demandé de calculer les dérivées de fonctions relativement simples, mais sauriez-vous calculer les dérivées de fonctions plus complexes telles que $f(x)=(4x^3+x^2+5)^3$ ou $f(x)=\log(x^3+4x)$?

Ces dérivées peuvent paraître pénibles à calculer si vous ne connaissez pas le théorème de la dérivation de fonctions composées (également appelé dérivation en chaîne). Une fonction composée est une fonction qui peut s'exprimer comme la composition de 2 fonctions. Par exemple, nous pouvons représenter les fonctions précédentes de la manière suivante :

$(4x^3+x^2+5)^3$ = f(g(x)) où f(x) = x^3 et g(x) = $4x^3 + x^2 + 5$

$\log(x^3+4x)$ = f(g(x)) où f(x) = log(x) et g(x) = $x^3 + 4x$

Théorème :

1) Si F(x) = f(g(x)), F'(x) = f'(g(x)) * g'(x)

2) Si y = f(u) et u = g(x), la dérivée de y est $\frac{dy}{dx}=\frac{dy}{du}\frac{du}{dx}$

Calculer la dérivée de $f(x)=(4x^3+x^2+5)^3$

Calculer la dérivée de $f(x)=\log(x^3+4x)$

Ex07 :

Fichier: ex07.py

Considérons un apprentissage supervisé avec les valeurs suivantes:

w	-2	-1	0	1	2.5	3.5	4	5	6	7
Loss	202.5	122.5	62.5	22.5	0	10.0	22.5	62.5	122.5	202.5
Dérivée de la fonction de coût par rapport à *w*	-45	-35	-25	-15	0	10	15	25	35	45

Afficher la fonction de coût (les valeurs de la rangée Loss) en fonction de w en posant les points ci-dessus sur un repère 2D, puis en les reliant.

Les points devront être rouges si la dérivée est négative, bleus si elle est positive, et verts si elle est nulle.

Interpréter le résultat affiché : quelle est la valeur optimale pour w ? Quelles décisions pouvons nous prendre à partir de la valeur de la dérivée ? (Répondez dans le .py)

Conclusion:

La fonction de coût calcule la différence entre le résultat attendu et le résultat prédit par le réseau de neurones: plus le coût est grand, plus le réseau de neurones s'est « trompé » sur ses prédictions, et plus nous devons modifier nos paramètres (les poids synaptiques). Le but de l'apprentissage est d'optimiser nos poids **par rapport à cette fonction de coût**, afin d'avoir le coût (ou erreur) le plus faible possible.

Il est très utile d'afficher la valeur de la fonction de coût pendant un apprentissage. Cela nous permet par exemple de vérifier que votre implémentation est fonctionnelle: si on constate que le coût ne baisse pas à chaque modification de nos paramètres (itération), il y a un problème quelque part!

Afin savoir comment modifier nos paramètres (les augmenter ou les diminuer), nous devons comprendre comment ils influencent la fonction de coût, et pour cela, nous utilisons **les dérivées.**

Les exercices d'aujourd'hui avaient pour but de vous donner une intuition de ces 2 concepts. N'hésitez pas à aller plus loin pour comprendre le mieux possible les notions abordées: plus elles seront claires, plus les jours suivants seront faciles.