Fonctions de coût et Machine Learning

Franck Jeannot - Janvier 2018 - Q489 - v1.1

Résumé : En ***Machine Learning***, on parle de fonction-objectif, de fonction de coût et fonction de perte. On fait ici une revue des usages avec un focus sur la **fonction des moindres carrés**.

Keywords/Mots-clés : *Cost function ; Loss function ; Machine Learning ; Méthode des moindres carrés ; supervised learning problem ; Squared error function ; Mean squared error ; Objective function*

1 Introduction

Les termes de **fonction de coût**, **fonction de perte** et de **fonction objectif** sont très utilisés dans de multiples domaines et sont tous étroitement liés. Dans cet article on se concentre plus sur les usages adaptés au machine learning.

2 Fonction de coût

Méthode des moindres carrés

On considère un ensemble de test avec **M** comme le nombre d'occurrences et une **fonction de coût** J avec θ_0, θ_1 comme **paramètres** et **une fonction hypothèse** de format $h_\theta(x) = \theta_0 + \theta_1 x$. On considère aussi dans un contexte de **machine learning**, $x^{(i)}$ représentant les "variables d'entrée" et $y^{(i)}$ les "sorties" de la fonction. On peut alors considérer :

$$J(\theta_0, \theta_1) = \frac{1}{2M} \sum_{i=1}^{M} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{1}$$

On nomme aussi cette approche la "Méthode des moindre carrés" [1]. On note X l'espace des valeurs d'entrée et Y l'espace des valeurs de sorties avec $X = Y = \mathbb{R}$.
Pour décrire un problème de *Machine Learning* **supervisé**, de manière formelle, on considère que, étant donné un jeu de test, pour "apprendre", la fonction $h : X \to Y$ de telle manière que $h(x)$ permet une bonne prédiction des valeurs y. Pour des raisons historiques dans le domaine du Machine Learning, on appelle *hypothèse* la fonction h (même si ce terme n'est pas parfaitement adapté). Dans le cas d'une progression linéraire, la fonction de coût va permettre de déterminer la meilleure approximation linéaire des données disponibles.

1. https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

La fonction hypothèse s'écrit : $h_\theta(x) = \theta_0 + \theta_1 x$ avec θ_i appelés **paramètres**. Il s'agira de choisir θ_0, θ_1 de telle manière que $h_\theta(x)$ est proche de y dans nos jeux de données (x, y).

En synthèse cela revient à **Minimiser** avec θ_0, θ_1 et donc de **minimiser la différence des carrés** entre la sortie de l'hypothèse et la valeur réelle $(h_\theta(x) - y)^2$.

On a donc respectivement :

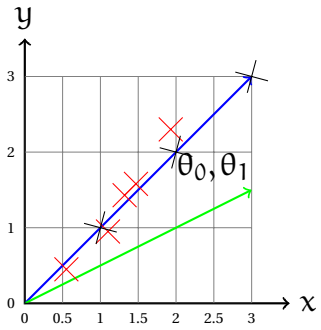
$$\textbf{Minimiser } \theta_0, \theta_1 \quad \frac{1}{2M} \sum_{i=1}^{M} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{2}$$

$$h_\theta(x^{(i)}) = \theta_0 + \theta_i x^{(i)} \tag{3}$$

$$J(\theta_0, \theta_1) = \frac{1}{2M} \sum_{i=1}^{M} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{4}$$

2.1 Progression linéaire : exemple et simplifications

On considère un jeu de données simplifié (croix noires) : $(1,1)(2,2)(3,3)$ et les représentations graphiques $\theta_1 = 1$ (ligne bleue) et $\theta_1 = 0.5$ (ligne verte) :



2.2 Hypothèse 1 : $h_\theta(x) = \theta_0 + \theta_1 x$

— Paramètres : θ_0, θ_1
— Fonction de coût : $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$
— Objectif : minimiser θ_0, θ_1 $J(\theta_0, \theta_1)$

2.3 Hypothèse 2 : $h_\theta(x) = \theta_1 x$

— Paramètres : θ_1
— Fonction de coût : $J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{M} (h_\theta(x^{(i)}) - y^{(i)})^2$
— Objectif : minimiser θ_1 $J(\theta_1)$

Pour $\theta_1 = 1$, $J(\theta_1) = \frac{1}{2m}(0^2 + 0^2 + 0^2)$ so $J(1) = 0$

Pour $\theta_1 = 0.5$, et le jeu de données $(1,1)(2,2)(3,3)$, on a :

$$\begin{aligned}
J(0.5) &= \frac{1}{2m}\sum_{i=1}^{m}(h_1(x^{(i)}) - y^{(i)})^2 \\
&= \frac{1}{2*3}\sum_{i=1}^{3}(h_1(x^{(i)}) - y^{(i)})^2 \\
&= \frac{1}{6}[(h_1(x^{(1)}) - y^{(1)})^2 + (h_1(x^{(2)}) - y^{(2)})^2 + (h_1(x^{(3)}) - y^{(3)})^2)] \\
&= \frac{1}{6}[(0.5-1)^2 + (2-1)^2 + (1.5-3)^2)] \\
&= \frac{1}{6}[(-0.5)^2 + (1)^2 + (-1.5)^2)] \\
&= \frac{1}{6}[0.25 + 1 + 2.25] \\
&= \frac{3.5}{6} \\
&\approx 0.583
\end{aligned} \tag{5}$$

3 Revue des termes

D'un point de vue sémantique les termes *fonction de perte* et *fonction de coût* sont plus ou moins synonymes et il n'existe pas de convention parfaite de distinction. La ***fonction de perte*** (*loss function*[2]) est généralement une fonction définie sur un point de données, une prédiction et une étiquette, et mesure la **pénalité**.[3] Des exemples sont :[4]

— *square loss* $l(f(x_i|\theta), y_i) = (f(x_i|\theta) - y_i)^2$, utilisé en **régression linéaire**[5],

— *hinge loss*[6] $l(f(x_i|\theta), y_i) = \max(0, 1 - f(x_i|\theta)y_i)$ utilisé en SVM (Support Vector Machine[7])

Une **fonction de coût** est souvent un terme plus général. Il peut par exemple être composé de sommes de *fonctions de pertes* et des régularisations. Par exemple :

— **Méthode des moindres carrés** (*Mean Squared Error*) $MSE(\theta) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i|\theta) - y_i)^2$

— *SVM cost function* $SVM(\theta) = \|\theta\|^2 + C\sum_{i=1}^{N}\xi_i$ (prononcer xi pour ξ)(comporte des contraintes additionnelles : ξ, C, training set...etc)

2. https://en.wikipedia.org/wiki/Loss_function
3. http://www.statsoft.fr/concepts-statistiques/glossaire/f/fonction-perte.html
4. https://en.wikipedia.org/wiki/Loss_functions_for_classification
5. https://web.stanford.edu/class/cs221/lectures/learning1.pdf
6. https://en.wikipedia.org/wiki/Hinge_loss
7. https://en.wikipedia.org/wiki/Support_vector_machine