TD Machine Learning

Régression Linéaire

Problématique général : Dataset constitué d'un ensemble de variables $x_i, i \in \{1, ..., N\}$ et d'un ensemble de cibles $y_i, i \in \{1, ..., M\}$.

Le dataframe est donc de la forme :

$$\begin{array}{|c|c|c|c|c|c|}
 & x^1 & x^2 & \dots & x^N & y \\
\hline
1 & x_1^1 & x_1^2 & \dots & x_1^N & y_1 \\
\hline
2 & x_2^1 & x_2^2 & \dots & x_2^N & y_2 \\
\hline
 & \vdots & \vdots & \dots & \vdots & \vdots \\
\hline
 & \vdots & \vdots & \dots & \vdots & \vdots \\
\hline
\text{M} & x_M^1 & x_M^2 & \dots & x_M^N & y_M \\
\hline
\end{array} \tag{1}$$

On veut "mapper" les valeurs des variables x^i aux cibles y_j.

Trouver une fonction F qui effectue l'association avec le moins d'erreur possible.

$$\forall j = 1, ..., M, y_j = F\left(x_j^1, ..., x_j^N\right) \tag{2}$$

L'erreur mesure de l'écart entre valeurs trouvées par F et les cibles y_j.

En machine learning on utilise l'erreur qudratique moyenne.

$$E = \sum_{i=1}^{N} \frac{1}{2 \times M} \sqrt{\sum_{j=1}^{M} \left(F\left(x_j^1, ..., x_j^N\right) - y_j\right)^2} \tag{3}$$

Ce qui revient à minimiser

$$\sum_{i=1}^{N} \frac{1}{2 \times M} \sum_{j=1}^{M} \left(F\left(x_j^1, ..., x_j^N\right) - y_j\right)^2 \tag{4}$$

Exemple applicatif : 1 seule variable avec 5 caractéristiques,

	x	y
1	4	1
2	7	3
3	8	3
4	10	6
5	12	7

(5)

1. Sur le plan, grâce à un repère orthogonal, placer les points de coordonnées (x_i, y_i).

   En régression linéaire, le modèle f est une droite, l'objectif est donc de trouver la droite d'équation $y = ax+b$ qui minimise les erreurs entre les points du plan et les points de cette droite (les paramètres inconnues sont a et b).

   Autrement dit, les abscisses x_i étant fixés, on cherche à minimiser les quantités $(ax_i + b) - y_i$, où les y_i sont données par le Dataframe.

2. Visualiser graphiquement ces erreurs sur le plan.

3. On pose $X = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_5 & 1 \end{pmatrix}$ et $\theta = \begin{pmatrix} a \\ b \end{pmatrix}$

   quel est le vecteur F tel que :

   $$F(X) = X\theta$$

   Rappel $f(x_i) = y_i$ pour notre dataframe et $f(x) = ax + b$ est notre modèle (régression linéaire).

   Commenter sur l'utilité de Numpy dans le cas d'un très grand dataset.

4. On considère l'exemple de notre dataset, donc $M = 1$ (une seule variable) et $N = 5$ (5 caractéristiques), montrer que minimiser (3) revient à minimier la quantité :

   $$E(\theta) = \frac{1}{2 \times 5} \sum_{j=1}^{5} \left(F\left(x_i\right) - y_j\right)^2 \tag{6}$$

   En considérant comme ci-dessus $F(X) = X\theta$, trouver une écriture matricielle de $E(\theta)$

5. Descente du gradient : La méthode de la descente du gradient consiste à trouver la meilleure droite permettant de minimiser l'erreur $E(\theta)$. Elle revient donc à trouver les coefficients a et b de la droite optimale $y = ax + b$ minimisant ces erreurs.

   On appelle gradient de $E(\theta)$ le vecteur des dérivées partielles par rapport à a et b, soit :

   $$\nabla E(\theta) = (\frac{\partial E}{\partial a}, \frac{\partial E}{\partial b})^T$$

   Calculer le gradient de l'erreur $E(\theta)$ lorsque le modèle est linéaire $F(x) = ax + b$.
   Il faut trouver $\left(\frac{1}{n}\sum_i \left(x(ax+b-y)\right), \frac{1}{n}\sum_i \left(x(ax+b-y)\right)\right)^T$.

6. La méthode de la descente de gradient consiste à trouver itérativement les paramètre (a, b) minimisant la fonction perte $E(\theta)$ (Loss). Cette méthode effectue l'algorithme suivant :

   - On initialise par $\theta^0 = (a(0), b(0))^T$
   - Pour t=0,1,..., jusqu'à convergence (erreur suffisamment petite)
     - On calcule
       $\theta^{t+1} = (a(t+1), b(t+1))^T$ de la façon suivante:
       $\theta^{t+1} = \theta^t - \eta \nabla E(\theta)$

   On initialise avec $\theta^0 = (a(0), b(0))^T = \left((\frac{3}{4}), -2\right)^T$,

   Tracer la droite $y = a(0)x + b(0)$.

   calculer $\theta^{10} = (a(10), b(10))^T$ et $E(\theta^{10})$.

7. Tracer la droite $y = a(10)x + b(10)$.

8. Conclure.

   Vous venez de réaliser une régression linéaire à la main. Numpy et Pandas permettent de faire le calcul matriciel à votre place. Sickit-Learn permet d'appler directement avec :

   from sklearn.linear_ model import LinearRegression