Chapitre 2 : Régression linéaire

M2 IWOCS, Apprentissage Automatique

Partie B : Régression linéaire multivariée et écriture matriciellle de la méthode de descente du gradient

Nous nous intéressons ici à une extension du modèle univarié (à un degré de liberté) à un modèle à plusieurs degrés de liberté (ici, n) qui correspondent à un ensemble de m données d'entrée à n composantes (x_{ij}) et qui forment donc une matrice de dimension (m, n):

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$

Les données sont toujours constituées d'une valeur cible (y_i) et l'ensemble des données est décrit donc par

$$\{(x_{ij}, y_i)\ ; \quad 1 \leq j \leq n \quad \text{et} \quad 1 \leq i \leq m\}$$

Nous considérons donc que dans un modèle multivarié, nous avons n observations plutôt qu'une seule dans le modèle univarié. Chaque observation correspond à une colonne de la matrice X précédente. On note X_i la colonne i de la matrice.

1. Fonction hypothèse

L'hypothèse (ou fonction de régression) prend en entrée une matrice et calcule un vecteur :

$$h_\theta(X) = \theta_0 I_1 + \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_n X_n$$

où l'on note I_1 le vecteur dont toutes les m composantes sont égales à 1.

Cette expression est linéaire par rapport aux observations X_i.

On note x_i le vecteur contenant la donnée i composée de n observations (attention à l'usage des majuscules et minuscules !).

$x_i = i^{eme}$ ligne de la matrice X

$x_i = (x_{i1}\ x_{i2}\ \dots\ x_{in})$

La projection de h_θ sur chacun de ces vecteurs x_i s'écrit donc :

$$h_\theta(x_i) = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \cdots + \theta_n x_{in}$$

Le problème de régression lineaire multivariée consiste ainsi à trouver le vecteur

$$\theta = \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix}$$

tel que $h_\theta(X)$ soit "le plus proche" de y.

2. Fonction de coût

La fonction de coût utilisée correspond au critère des moindres carrés utilisé dans le chapitre sur la régression linéaire univariée :

$$\begin{aligned} J(\theta) &= J(\theta_0, \theta_1, \dots, \theta_n) \\ &= \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2 \end{aligned}$$

3. Minimisation de la fonction de coût par la méthode de gradient

Résoudre le problème de régression linéaire multivarié consiste à chercher $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$ qui rend $J(\theta)$ minimal. Le processus décrit pour la régression linéaire univariée se généralise assez simplement comme cela est décrit ci-après.

A partir de valeurs initiales (arbitraires) des composantes de $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$, une méthode itérative est utilisée.

Description d'une itération

A partir de $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$ on va calculer $\theta^* = (\theta_0^*, \theta_1^*, \ldots, \theta_n^*)$ tel que

$$J(\theta^*) < J(\theta)$$

θ^* se calcule à partir du gradient de la fonction J, noté $\nabla J(\theta)$, grâce à la formule :

$$\theta^* = \theta - \alpha \nabla J(\theta)$$

$$\begin{cases} \theta_0^* & = \theta_0 - \alpha \dfrac{\partial}{\partial \theta_0} J(\theta) \\ & \vdots \\ \theta_j^* & = \theta_j - \alpha \dfrac{\partial}{\partial \theta_j} J(\theta) \\ & \vdots \\ \theta_n^* & = \theta_n - \alpha \dfrac{\partial}{\partial \theta_n} J(\theta) \end{cases}$$

Le calcul des dérivées partielles a été expliqué dans la partie A du chapitre portant sur la régression linéaire univarié. Il n'est pas décrit ici mais sera repris dans la section suivante.

Avant de passer à l'itération suivante, on va recopier le vecteur θ^* dans θ.

Ce schéma itératif est exécuté un nombre fini de fois qui peut être calculé automatiquement lorsque la différence relative entre θ^* et θ est faible, c'est à dire :

$$\frac{\|\theta^* - \theta\|}{\|\theta\|} < \epsilon$$

pour ϵ une "petite" valeur.

4. Etape de normalisation

Dans les données d'entrée représentées par la matrice X, il est possible que des observations aient des ordres de grandeur différents. Par exemple, prenons le problème "classique" d'estimation du prix d'un appartement en fonction de deux observations : le nombre de pièces et sa superficie. Dans un jeu de données, on va supposer que le nombre de pièces varie entre 1 et 9 alors que la superficie varie entre 20 et 200 m2. Du fait de la différence d'ordre de grandeur entre ces 2 observations, l'algorithme de descente du gradient aura du mal à converger.

Pour pallier à cette difficulté, on va "normaliser" chaque observation, c'est à dire chaque vecteur colonne X_i de la matrice X, de sorte que ces différentes observations soient comparables et du même ordre de grandeur.

Plusieurs méthodes de normalisation sont possibles. Nous en présentons deux.

4.1 Normalisation unitaire

On normalise chaque observation X_i pour que ses coefficients soient compris entre 0 et 1.

Soit X_{ij} la j^{eme} composante du vecteur X_i, soit $\max(X_i)$ la plus grande composante du vecteur X_i et $\min(X_i)$ la plus petite, on utilise la formule suivante pour calculer $\overline{X_{ij}}$ la valeur normalisée de X_{ij} :

$$\overline{X_{ij}} = \frac{X_{ij} - \min(X_i)}{\max(X_i) - \min(X_i)}$$

4.2 Normalisation "centrage réduction"

Plutôt que d'imposer un intervalle dans lequel s'inscrivent les coefficients, on va contrôler la dispersion des valeurs en ramenant l'écart-type à 1, et la moyenne à 0.

> **Rappels** : Soit Z un vecteur de valeurs de composantes z_i, pour $1 \leq i \leq n$. On définit la moyenne par :
>
> $$\mu(Z) = \frac{1}{n}\sum_{i=1}^{n} z_i$$
>
> L'écart-type, racine carrée de la variance, permet de mesurer la dispersion des données autour de la moyenne. Il se calcule par :
>
> $$\sigma(Z) = \sqrt{V(Z)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i - \mu(Z))^2}$$

La normalisation "centrage réduction" des données de l'observation X_i se calcule par

$$\overline{X_{ij}} = \frac{X_{ij} - \mu(X_i)}{\sigma(X_i)}$$

Ainsi la moyenne des valeurs normalisées vaut 0 et son écart-type vaut 1.

4.3 Modules de normalisation de la bibliothèque `scikit-learn`

La bibliothèque `scikit-learn` de Python, offre plusieurs méthodes permettant de normaliser les données. Cette normalisation est donc une étape importante pour "préparer" les jeux de données avant leur traitement. Nous donnons ci-après les deux modules de normalisation de `scikit-learn` (parmi d'autres) qui réalisent les 2 méthodes de normalisations décrites précédemment :

- `sklearn.preprocessing.MinMaxScaler` est le module de normalisation qui implémente la formule du paragraphe 4.1
- `sklearn.preprocessing.StandardScaler` est le module de normalisation qui implémente la formule du paragraphe 4.2

Voici un exemple d'utilisation

```
from sklearn.preprocessing import StandardScaler,
MinMaxScaler
scaler = StandardScaler()
# house correspond aux données sur les appartements
new_house = scaler.fit_transform(house)
```

5. Ecriture matricielle de la méthode de descente de gradient

5.1 Formulation matricielle de la fonction hypothèse

En reprenant les notations introduites dans les sections précédentes, la fonction hypothèse s'écrit

$$h_\theta(X) = \theta_0 I_1 + \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_n X_n$$

où l'on note I_1 le vecteur dont toutes les m composantes sont égales à 1.

Pour chaque donnée i dont les valeurs des n observations sont dans le vecteur x_i, l'hypothèse se projette de la manière suivante :

$$h_\theta(x_i) = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \cdots + \theta_n x_{in}$$

A la matrice de données d'entrée X on ajoute en première colonne le vecteur I_1 et la matrice X sera donc maintenant :

$$X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{m1} & \dots & x_{mn} \end{bmatrix}$$

Et la fonction hypothèse se calcule comme le produit matriciel :

$$h_\theta(X) = X\theta$$

où θ correspond au vecteur définit précédemment :

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

En effet

$$X\theta = \begin{bmatrix} 1 & x_{11} & \dots & x_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{m1} & \dots & x_{mn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$X\theta = \begin{bmatrix} \theta_0 + \theta_1 x_{11} + \dots + \theta_n x_{1n} \\ \vdots \\ \theta_0 + \theta_1 x_{i1} + \dots + \theta_n x_{in} \\ \vdots \\ \theta_0 + \theta_1 x_{m1} + \dots + \theta_n x_{mn} \end{bmatrix}$$

On retrouve bien que l'expression du i^{eme} coefficient de ce vecteur est celle de $h_\theta(x_i)$

5.2 Formulation matricielle de la fonction de coût

La fonction de coût est une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$; elle s'écrit

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Soit le vecteur de données cible

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

On pose le vecteur

$$E = h_\theta(X) - Y$$

$$E = \begin{bmatrix} h_\theta(x_1) \\ \vdots \\ h_\theta(x_n) \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$E = \begin{bmatrix} h_\theta(x_1) - y_1 \\ \vdots \\ h_\theta(x_n) - y_n \end{bmatrix}$$

Sachant que le carré scalaire d'un vecteur Z, de composantes z_i s'écrit

$$Z^t Z = \sum_{i=1}^{m} (z_i)^2$$

Le carré scalaire du vecteur E s'écrit alors

$$E^t E = \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

et la fonction de coût s'écrit alors sous forme matricielle

$$J(\theta) = \frac{1}{2m} E^t E$$

5.3 Formulation matricielle de la méthode de descente du gradient

La méthode de descente du gradient consiste à appliquer des itérations successives qui prennent en entrée un vecteur θ pour calculer un nouvelle valeur de ce vecteur θ^* tel que $J(\theta^*) < J(\theta)$.

Cette nouvelle valeur θ^* sert alors d'entrée pour la nouvelle itération. Pour initier le processus, on choisit arbitrairement la valeur initiale de θ.

On peut choisir de s'arrêter après un nombre arbitraire d'itérations ou alors contrôler la convergence du calcul de θ et prendre un critère d'arrêt qui peut s'écrire :

$$\left| \frac{J(\theta^*) - J(\theta)}{J(\theta)} \right| < \epsilon$$

où ϵ est une valeur petite de l'ordre de 10^{-3} ou plus petit.

Le processus itératif dans une régression linéaire multivariée est une généralisation du processus utilisé pour une régression linéaire univariée :

$$\begin{cases} \theta_0^* = \theta_0 - \dfrac{\alpha}{m} \displaystyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i) \\ \theta_1^* = \theta_1 - \dfrac{\alpha}{m} \displaystyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{i1} \\ \vdots \\ \theta_j^* = \theta_j - \dfrac{\alpha}{m} \displaystyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{ij} \\ \vdots \\ \theta_n^* = \theta_n - \dfrac{\alpha}{m} \displaystyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{in} \end{cases}$$

Ce processus s'écrit sous la forme matricielle suivante :

$$\theta^* = \theta - \frac{\alpha}{m} X^t E$$

En effet

$$X^t E = \begin{bmatrix} 1 & \dots & 1 \\ x_{11} & \dots & x_{m1} \\ \vdots & & \\ x_{1n} & \dots & x_{mn} \end{bmatrix} \begin{bmatrix} h_\theta(x_1) - y_1 \\ \vdots \\ h_\theta(x_m) - y_m \end{bmatrix}$$

$$X^t E = \begin{bmatrix} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) \\ \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{i1} \\ \vdots \\ \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{ij} \\ \vdots \\ \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{in} \end{bmatrix}$$

5.4 Implémentation de la formulation matricielle en Python

1. On lit les données correspondant à n observations en entrée et une valeur de sortie

$$\begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \dots & x_{mn} \end{bmatrix} \quad \text{et} \quad \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

2. On construit la matrice X augmentée et le vecteur Y

$$X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1n} \\ \vdots & \vdots & & \vdots \\ 1 & x_{m1} & \dots & x_{mn} \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

1. On définit le fonction hypothèse ayant pour variables, le vecteur θ et la matrice augmentée X et renvoyant le vecteur :

$$h(\theta, X) = X\theta$$

2. On définit la fonction de coût, de variable θ :

$$J(\theta) = \frac{1}{2m} E^t E$$

où $E = h_\theta(X) - Y$

3. On définit une fonction itération de la méthode de descente du gradient, qui prend en entrée θ et qui calcule θ^* par la formule :

$$\theta^* = \theta - \frac{\alpha}{m} X^t E$$

On fixe alors un nombre donné d'itérations en partant d'une valeur arbitraire de θ ou alors on contrôle la convergence de calcul de θ avec le critère décrit précédemment.

In [ ]: