Université Lumière Lyon 2

Librairie MIMOSA

Mixed Input Multinomial Optimization for Statistical Analysis

Régression logistique multinomiale sur données mixtes, par descente de gradient

Par
Linh Nhi Le Dinh
Antoine Oruezabala
Béranger Thomas

Rapport présenté dans le cadre du
Master 2 SISE - Promotion 2024/2025

Table des matières

Résumé

Ce rapport détaille l'implémentation de la régression logistique multinomiale pour des données mixtes, en langage R, et en utilisant la descente de gradient. Nous couvrons tout d'abord quelques fondements théoriques, puis l'approche pratique et enfin discutons des résultats obtenus. En fin d'ouvrage, un lexique donne une définition des termes et expressions souvent usité en régression logistique.

Ce rapport a été rédigé en novembre 2024 dans le cadre d'un projet du master 2 SISE.

Chapitre 1

Théorie

1.1 La régression logistique multinomiale

1.1.1 Principes Généraux

La régression logistique vise à prédire la probabilité qu'un événement binaire se produise (oui/non, 1/0) en fonction de variables explicatives. Elle utilise une fonction logistique (ou sigmoïde) pour transformer la combinaison linéaire des variables explicatives en une probabilité comprise entre 0 et 1.

La régression logistique **multinomiale** est une généralisation de la régression logistique binaire. Elle permet d'estimer la probabilité d'appartenance à chacune des catégories en fonction de variables explicatives quantitatives et/ou qualitatives. Contrairement à la régression logistique binaire qui modélise une seule probabilité p, la régression logistique multinomiale estime simultanément les probabilités d'appartenance à toutes les catégories $(p_1, p_2, ..., p_K)$, avec la contrainte que leur somme soit égale à 1.

Le modèle utilise une transformation logistique qui garantit que les probabilités estimées restent comprises entre 0 et 1. Une catégorie est choisie comme référence, et le modèle estime les logarithmes des rapports de probabilités (log-odds) entre chaque catégorie et cette référence.

Nous nous placerons dans le cas où cette régression est nominale, c'est-à-dire que l'on ne suppose aucun ordre particulier entre les modalités de la variable à prédire.

1.1.2 Modélisation mathématique

Pour construire un modèle de régression logistique, l'équation de régression linéaire est utilisée comme point de départ[2, 1] :

$$\hat{y} = a + b_1 x_1 + \ldots + b_k x_k \tag{1.1}$$

Avec :
- $\hat{y}$ la variable cible
- a l'intercept,
- x_k la variable prédictive k,
- b_k le coefficient rattaché à cette variable,
- K le nombre de variables prédictives.

Toutefois, une telle équation produirait des coefficients $\in \mathbb{R}$, et l'objectif de la régression logistique est d'estimer la probabilité d'occurrence et non la valeur de la variable elle-même. Cette équation doit donc être transformée.

C'est le rôle de la fonction fonction softmax. Cette dernière, pour toute valeur comprise entre $-\inf et + \inf$ renvoie une valeur comprise entre 0 et 1.

Si on applique la fonction softmax à l'équation de régression linéaire, on obtient :

$$\log\left(\frac{P(Y=j|X)}{P(Y=J|X)}\right) = \beta_{0j} + \beta_{1j}X_1 + \beta_{2j}X_2 + ... + \beta_{kj}X_k \tag{1.2}$$

1.1.3 Fonction de Coût

La fonction de coût utilisée pour la régression logistique multinomiale est la cross-entropie catégorielle. Elle évalue la divergence entre les probabilités prédites par le modèle et les probabilités réelles des classes. Elle est définie comme suit :

$$J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\log\left(P(Y=k \mid X_i)\right)$$

où :

- N est le nombre total d'observations. - K est le nombre de classes. - $y_{i,k}$ est une variable indicatrice qui vaut 1 si l'observation i appartient à la classe k, et 0 sinon. - $P(Y = k \mid X_i)$ est la probabilité prédite que l'observation i appartienne à la classe k.

Minimiser cette fonction de coût permet d'ajuster les paramètres β du modèle pour améliorer la précision des prédictions. La descente de gradient est une méthode d'optimisation utilisée pour trouver les valeurs optimales de β qui minimisent $J(\beta)$.

1.1.4 Hypothèses du Modèle

Dans le cadre de la régression logistique multinomiale, plusieurs hypothèses fondamentales sont posées pour assurer la validité et la robustesse du modèle :

- **Indépendance des observations** : Chaque observation est supposée indépendante des autres. Cette hypothèse garantit que l'information apportée par chaque donnée est unique et non redondante.

— **Relation linéaire entre les variables explicatives et le logit** : Le modèle suppose une relation linéaire entre les variables explicatives X et le logarithme des probabilités conditionnelles (logits). Cela signifie que l'effet de chaque variable explicative est additif sur le logit.
— **Absence de multicolinéarité parfaite** : Les variables explicatives ne doivent pas être des combinaisons linéaires exactes les unes des autres. La présence de multicolinéarité peut entraîner des estimations instables des coefficients du modèle.
— **Catégories mutuellement exclusives et exhaustives** : Les classes de la variable dépendante doivent être clairement définies sans chevauchement, et couvrir toutes les possibilités.

La fonction *SoftMax* est essentielle dans ce modèle car elle transforme les scores linéaires en probabilités. Elle garantit que les sorties du modèle sont comprises entre 0 et 1 et que la somme des probabilités pour toutes les classes est égale à 1 pour chaque observation. La fonction *SoftMax* est définie par :

$$P(Y = k \mid X) = \frac{\exp(X\beta_k)}{\sum_{j=1}^{K} \exp(X\beta_j)}$$

où K est le nombre total de classes. Cette transformation est basée sur l'hypothèse que les logits peuvent être exponentiés et normalisés pour obtenir des probabilités, ce qui est cohérent avec le principe de maximum d'entropie.

L'utilisation de la fonction *SoftMax* implique également que les classes sont mutuellement exclusives, ce qui signifie qu'une observation ne peut appartenir qu'à une seule classe. Cela correspond à la nature des problèmes de classification multinomiale où le but est de prédire une seule catégorie parmi plusieurs possibles.

1.2 L'algorithme de descente de gradient

La descente de gradient est un algorithme doptimisation itératif, qui permet de trouver le minimum d'une fonction.

L'idée est la suivante :
— on part d'un point choisi aléatoirement,
— depuis ce point nous cherchons le gradient (la pente),
— et nous descendons d'un pas dans la direction opposé au gradient,
— nous recommençons le processus à partir du nouveau point atteint.

Les critères pour s'arrêter peuvent être :
— le nombre d'itérations ; on définit un maximum au début de l'algorithme, que l'on ne dépasse pas,
— la différence de "niveau" entre deux points ; si elle passe en dessous d'un seuil très faible, l'approximation est jugée suffisamment bonne,
— une combinaison de ces deux critères, qui a l'avantage de s'arrêter le plus tôt possible.

Vocabulaire :
— la taille du pas est dénommé taux d'apprentissage (en anglais learning rate),
— la différence minimale souhaitée entre deux points consécutifs est la tolérance,

—
—
Mathématiquement, la pente est définie par le gradient. La taille du pas est dénommé taux d'apprentissage.

La pente, forte au départ forte, doit aller en décroissant au fur et à mesure que l'on se rapproche du point le plus bas (ou d'un minimum local). On peut, pour limiter le temps de calcul, définir une différence minimale entre deux points consécutifs en-dessous de laquelle il n'est pas utile d'entamer une nouvelle itération. Cette différence minimale s'appelle la tolérance.

Il existe trois types d'algorithmes d'apprentissage par descente de gradient : la descente de gradient simple (sur tous les points), la descente de gradient stochastique et la descente de gradient par mini-lots.

1.2.1 Modélisation

1.2.2 Test

Chapitre 2

Implémentation Pratique en R

2.1 Structure du programme

L'architecture du programme est la suivante :

```
projet_racine/
├── datasets/
├── docs/
│   ├── Rapport.pdf
│   └── Rapport.tex
├── utils/
│   ├── LogisticRegression.r
│   ├── preprocessing.r
│   └── utils.r
└── main.r
```

Les fichiers sont décrits ci-après, avec du pseudo-code pour certaines fonctions.

2.1.1 main.r

Le fichier `main.r` contient le programme principal, qui instancie l'objet `LogisticRegression`, qui contient les fonctions suivantes :

- `fit` : pour modéliser
- `transform` : pour
- `probas`

2.1.2 utils.r

Le fichier `utils.r` contient une fonction *check_type*, qui vérifie si l'objet passé en premier argument est du même type que le type passé en second argument.

Si ce n'est pas le cas, le programme s'arrête.

2.1.3 preprocessing.r

Le fichier `preprocessing.r` contient plusieurs fonctions de prétraitement :

- imputer_numeriques : réalise une imputation par la moyenne des valeurs manquantes dans le jeu de données passé en argument.
- imputer_categorielles : réalise une imputation par le mode des valeurs manquantes dans le jeu de données passé en argument.
- split_train_test : réalise une séparation du jeu de données passé en argument

2.1.4 LogisticRegression.r

Le fichier `LogisticRegression.r` contient toutes les fonctions nécessaires à la réalisation de la régression logistique :

Fonction « sigmoid »

Applique la fonction sigmoïde à la valeur passée en argument et renvoie le résultat.

Fonction « softmax »

Applique la fonction softmax à la valeur passée en argument et renvoie le résultat.

Fonction « cout »

Fonction de coût pour la régression logistique, avec cross-entropy :

```
cout <- function(X, y, beta) {
    n <- nrow(X)
    scores <- X %*% beta
    probs <- exp(scores) / rowSums(exp(scores))
    cost <- -mean(sum(y * log(probs)))
    return(cost)
    }
```

Fonction « hypothese »

Fonction « descente_gradient »

Applique une descente de gradient pour la régression logistique :

```
descente_gradient <- function(X, y, beta) {
 n <- nrow(X)
 scores <- X %*% beta
 probs <- exp(scores) / rowSums(exp(scores))
 gradient <- -t(X) %*% (y - probs) / n
 return(gradient)
}
```

2.1.5 Répertoire docs

Il contient :
- ce rapport au format pdf : `Rapport.pdf`,
- et au format Latex : `Rapport.tex`

2.2 Vérification des données

Afin de s'assurer de l'intégrité des données, nous procédons à plusieurs vérifications lors de l'instanciation de l'objet *LogisticRegression*.

2.2.1 Format des objets passés en argument

Les deux vérifications suivantes sont réalisées à l'aide de la fonction *check_type* du fichier *utils.r* :

1. les données explicatives X doivent être dans un data.frame
2. la cible y doit être un vecteur ou un character

2.2.2 Format des données

1. Les données de la cible y doivent être de type catégorielle (character ou factor).

2.2.3 Cohérence des données

1. La cible y doit être présente dans les données X,
2. il faut au moins deux modalités pour la variable cible y.

2.3 Pré-traitement des données

2.3.1 Variable cible y

Si le format des données de la variable cible ne correspondaient pas aux pré-requis (variable catégorielle et avec au moins deux modalités), le programme l'aura signalé lors de l'instanciation de l'objet *LogisticRegression*.

Il n'y a pas d'autre contrainte à respecter pour la variable cible y, mais une opération de suppression des lignes sans la variable cible est mené ici. Le but étant bien évidemment de ne pas charger à modéliser sur des données manquantes pour a cible, ce qui n'aurait pas de sens puisque nous sommes en apprentissage supervisé.

2.3.2 Variables explicatives X

Les variables explicatives peuvent être mixtes, numériques et/ou catégorielles. Dans les deux cas, des pré-traitements sont nécessaires.

Variables numériques

Afin de pouvoir comparer des données aux unités quelques fois différentes, on centre et réduit les données numériques. Les coefficients seront alors normalisés ce qui permettra leur comparaison directe.

Variables catégorielles

Chaque variable catégorielle subit deux étapes pour les préparer à la descente de gradient :

1. un *codage disjonctif* (one-hot encoding) pour produire des valeurs numériques,
2. la suppression d'une des modalités.

Cette dernière étape permet d'éviter la colinéarité, c'est-à-dire la redondance d'information entre variables (une variable désignant ici une des colonnes produites par le one-hot encoding ; c'est-à-dire une des modalités de la variable encodée).

Chapitre 3

Éléments introductifs

3.1 La régression logistique

Quelle soit binaire (la variable cible possède deux modalités) ou multinomiale (la variable cible possède plus de deux modalités), le principe de la régression logistique est de maximiser la vraisemblance, ou pour le dire autrement, de minimiser la déviance.

Quel outil pour mesurer ce paramètre ? La log-vraisemblance.

3.1.1 Notations

Soit une population Ω de n individus, définie par J variables explicatives notées $\{X_1, \ldots, X_J\}$, et une variable cible Y possédant K valeurs :

Ω	Cible	X_1	$\ldots$	X_J
1	Y_1			
$\ldots$	$\ldots$			
ω	Y_ω	$X_1(\omega)$	$\ldots$	$X_J(\omega)$
$\ldots$	$\ldots$			
n	Y_K			

Dans le cas binaire, la probabilité d'un individu ω d'être positif à priori se note p par commodité, pour simplifier $p(\omega)$, lui-même une notation simplifiée de $P[Y(\omega) = +]$.

Toujours dans le cas binaire, la probabilité d'un individu ω d'être positif à posteriori, c'est-à-dire la probabilité que l'on modélisera en apprentissage supervisée, se note π par commodité, pour simplifier $\pi(\omega)$, lui-même une notation simplifiée de $P[Y(\omega) = +/X(\omega)]$.

La fonction LOGIT pour cet individu ω est :

$$\text{LOGIT}(\omega) = \ln\left[\frac{\pi(\omega)}{1-\pi(\omega)}\right] = a_0 + a_1 X_1(\omega) + \cdots + a_J X_J(\omega) \tag{3.1}$$

Soit en écriture matricielle :

$$\ln\left[\frac{\pi(\omega)}{1-\pi(\omega)}\right] = X(\omega) \times a \tag{3.2}$$

Avec $a_0, \ldots, a_J$ les paramètres que l'on souhaite estimer.

3.1.2 Algorithmes

Un algorithme utilisé pour optimiser la log-vraisemblance est celui de Newton-Raphson. Il s'agit d'une approche itérative, qui utilise la dérivée de la fonction considérée pour approcher une solution.

Il a toutefois pour défaut d'utiliser la matrice hessienne, qui peut être coûteuse en mémoire et temps de calcul.

Nous expliquerons et utiliserons un autre algorithme ici, la descente de gradient.

Il s'agit aussi d'un algorithme itératif, dont l'objectif est de trouver le minimum d'une fonction $f(x)$ en partant d'un point arbitraire x_0. On se « déplace » pour cela dans la direction opposée au gradient, c'est-à-dire que l'on « descend » le long de la pente.

Mathématiquement, on part d'une valeur arbitraire x_0, puis pour trouver x_1 on utilise la formule suivante :

$$x_{n+1} = x_n - \alpha \nabla f(x_n) \tag{3.3}$$

Avec :
- x_n la position actuelle,
- α le taux d'apprentissage,
- $\nabla f(x_n)$ le gradient de la fonction au point x_n

Les étapes sont donc les suivantes :
- choisir un point x_0 arbitraire pour commencer les calculs,
- calculer le gradient à ce point,
- mettre à jour la position en se déplaçant dans la direction opposée,
- vérifier que l'on a atteint un critère d'arrêt :
  - nombre d'itérations maximum atteint,
  - et/ou précision souhaitée (différence entre deux itérations) atteinte,

3.1.3 Champs d'application

La régression logistique n'émet pas d'hypothèse directement sur les distributions des probabilités $P(X/Y = +)$ et $P(X/Y = -)$, mais uniquement sur leur rapport.

Elle permet donc, en théorie, une application plus large, et est nommé semi-paramétrique, pour la différencier des modèles qui supposent une loi donnée sur la distribution des probabilités.

Chapitre 4

Lexique

CENTRAGE ET RÉDUCTION ■ L'opération de centrage et réduction transforme les données pour avoir une moyenne de 0 et un écart-type de 1 :

$$x_{cr} = \frac{x_i - \bar{x}}{\sigma} \tag{4.1}$$

Centrer et réduire les données ne change pas les distributions, seulement l'échelle des données. Le centrage déplace le centre de la distribution au point zéro de l'axe des abscisses, et la réduction modifie l'échelle pour que l'écart-type soit égal à 1.

Il rend les variables indépendantes de leur unité de mesure initiale et permet donc de comparer des variables mesurées dans des unités ou sur des échelles différentes.

FONCTION SIGMOÏDE ■ La fonction sigmoïde ou logistique s'applique aux nombres réels et est utilisée pour transformer des valeurs réelles en probabilités. Elle produit une courbe en forme de S, avec des valeurs comprises entre 0 et 1.

Elle est l'inverse de la fonction LOGIT. Sa formule est :

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

Où :
- $\sigma(x)$ représente la fonction sigmoïde
- e est la base du logarithme naturel (constante d'Euler)
- x est la variable d'entrée

Les deux fonctions logistique et LOGIT étant inverses l'une de l'autre, cela signifie que :
- $\text{logit}(\sigma(x)) = x$
- $\sigma(\text{logit}(p)) = p$

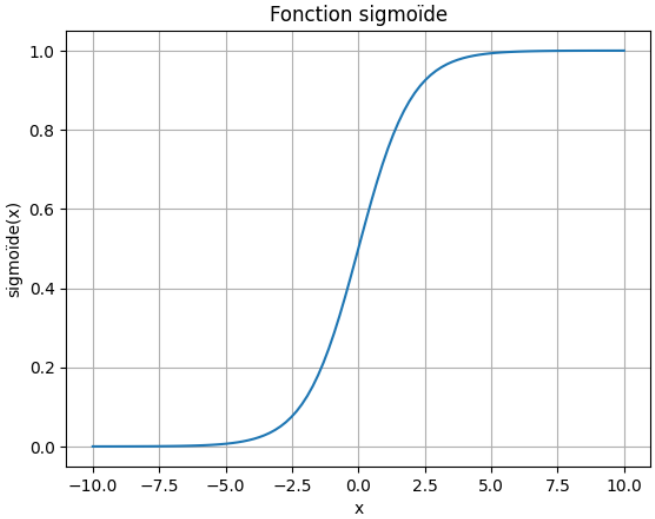

FIGURE 4.1 – La fonction sigmoïde

FONCTION SOFTMAX ■ La fonction softmax est une fonction d'activation qui transforme un vecteur de nombres réels en une distribution de probabilités.

MODALITÉ DE RÉFÉRENCE ■ Modalité de référence par rapport à laquelle le LOGIT est exprimé.

TAUX D'APPRENTISSAGE ■ Ou learning Rate : il fixe la ń grandeur ż du pas de chaque itération de la descente de gradient.

DESCENTE DE GRADIENT ■ La descente de gradient est un algorithme doptimisation couramment utilisé pour entraîner les modèles de machine learning et les réseaux neuronaux. Ce type dalgorithme entraîne les modèles de machine learning par réduction des erreurs entre les résultats prédits et les résultats réels. ■ Le point de départ n'est qu'un point arbitraire qui nous permet d'évaluer les performances. À partir de ce point de départ, nous allons trouver la dérivée (ou la pente) et, à partir de là, nous pourrons utiliser une ligne tangente pour observer l'inclinaison de la pente. La pente renseigne sur les mises à jour des paramètres, c'est-à-dire les poids et les biais. La pente au point de départ est plus forte, mais au fur et à mesure que de nouveaux paramètres sont générés, elle devrait progressivement diminuer jusqu'à atteindre le point le plus bas de la courbe, dénommé point de convergence. ■ Comme pour trouver la ligne de meilleur ajustement dans la régression linéaire, l'objectif de la descente de gradient est de minimiser la fonction de coût, ou l'erreur entre y prédit et y réel. Pour ce faire, deux points de données sont nécessaires : une orientation et un taux d'apprentissage. Ces facteurs déterminent les calculs de dérivée partielle des itérations futures, ce qui lui permet d'atteindre progressivement le minimum local ou global (c'est-à-dire le point de convergence).. La fonction de perte dans la descente de gradient agit spécifiquement comme un baromètre, évaluant sa précision à chaque itération des mises à jour de paramètres. Jusqu'à ce que la fonction soit proche ou égale à zéro, le modèle continue à ajuster ses paramètres pour obtenir l'erreur la plus faible possible. ■ Il existe trois types d'algorithmes d'apprentissage par descente de gradient : la des-

cente de gradient par lots, la descente de gradient stochastique et la descente de gradient par mini-lots.

Table des figures

Bibliographie

[1] Ricco RAKOTOMALALA. *Pratique de la régression logistique*. 2017.
[2] DATAtab TEAM. *Régression logistique*. URL : https://datatab.fr/tutorial/logistic-regression. (accessed : 28.11.2024).