

Modèles Linéaires

TD 5 : Modèles généralisés : Régression logistique Licence 3 MIASHS

Guillaume Metzler


Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2
Laboratoire ERIC UR 3083, Lyon, France

guillaume.metzler@univ-lyon2.fr

Résumé

Nous commençons notre étude des modèles linéaires généralisés avec le modèle de régression logistique. Les détails de la présentation de ce modèle ont été donnés en cours et nous ne reviendrons pas sur la façon dont sont estimés les paramètres du modèle.

Plus précisément, nous allons

- Apprendre à effectuer une régression logistique sous 
- Analyser les résultats de la régression logistique
- Construire un bon modèle de régression logistique
- Évaluer ses performances et sa capacité à généraliser sur de nouvelles données.

1 Généralités sur la régression logistique

On rappelle que notre modèle de régression logistique repose sur l'estimation du *logit* par une relation linéaire des covariables, *i.e.* on modélise le logarithme de l'*odd ratio* par un modèle linéaire

$$\ln \left(\frac{\Pr[Y = 1 \mid X = \mathbf{x}]}{\Pr[Y = 0 \mid X = \mathbf{x}]} \right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p \quad (1)$$

Dans ce type modèle, on ne cherche donc plus à prédire un nombre réel, mais plutôt la probabilité d'appartenance à une classe donnée, dite de *référence*, ici la classe 1. Le modèle de régression logistique est spécifiquement dédiée à des problèmes de **classification binaire**, où l'on cherche à **ranger les individus dans deux groupes distincts**. Ces groupes sont généralement notés 0 et 1, parfois -1 et 1.

En cours, nous avons vu que ce modèle, contrairement au modèle de régression linéaire classique n'admet de pas de solution explicite et nous devons donc avoir recours à des algorithmes d'optimisation pour approcher la solution.

1. Rappeler ce qu'est le modèle *logit*
2. Montrer, à partir de l'Equation (1), que la probabilité d'appartenir à la classe 1 est égale à

$$Pr(Y = 1 | X = \mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p))}$$

Rappeler l'expression de la vraisemblance.


Dans la suite de ce TP, on va travailler sur le jeu de données *glass.csv* que vous pourrez télécharger sur la page du cours.

```
# Renommer le jeu de données
data = read.csv("../data/glass.csv", header = TRUE)
head(data)

##           X1      X2      X3      X4      X5      X6      X7      X8      X9      y
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75 0 0.00 1
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83 0 0.00 1
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78 0 0.00 1
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22 0 0.00 1
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07 0 0.00 1
## 6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07 0 0.26 1
```

4. Evaluer la proportion d'individus dans chaque classe.

2 Un premier modèle de régression

On cherche maintenant à construire un modèle de régression logistique. Sur  nous pouvons apprendre un modèle de régression logistique à l'aide de la commande suivante

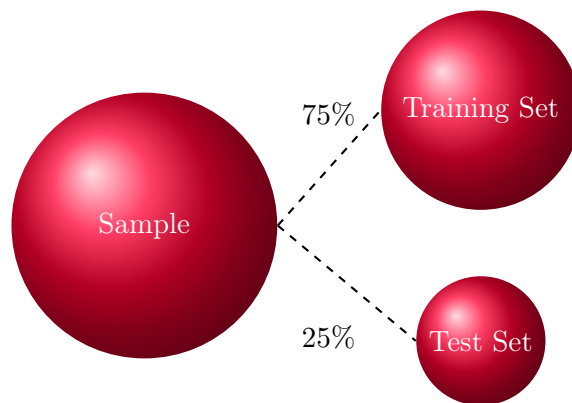


FIGURE 1 – Separating the dataset into train/test with proportion 75/25 %

```
mymodel = glm(y ~., data = data)
```

On commence par regarder quelques éléments de bases de la régression à l'aide de la fonction :

5. Commenter les sorties du modèle de régression en identifiant les variables significatives.
6. Quel critère peut-on utiliser pour évaluer la qualité du modèle et quelle est sa valeur ?

On va maintenant chercher à mettre en place une procédure d'apprentissage en utilisant le fait que nous devons diviser notre jeu de données en deux ensembles : un ensemble d'*entraînement* et un ensemble *test*, comme illustré en Figure 1.

Nous allons donc apprendre les paramètres du modèle sur l'ensemble d'entraînement et évaluer ses performances sur les données tests (non utilisées pour apprendre).

7. Regarder la fonctionnement de la fonction *CreateDataPartition* du package **caret** de **R** pour séparer votre jeu de données avec les proportions indiquées en Figure 1. Créer votre jeu de données *train* et *test*.

```
set.seed(42)
index_train <- createDataPartition(y=data$y, p = 0.75, list=FALSE)
```

8. Apprendre le modèle sur le jeu de données *train*.

Pour évaluer les performances du modèles, nous pouvons chercher à évaluer son accuracy, *Acc*.

$$Acc = \frac{TP + TN}{FP + FN + TP + TN}.$$

1. On peut effectuer les prédictions sur les nouvelles données à l'aide de la fonction *predict*

```
prediction = predict(mymodel, newdata=train, type="response")
```

Les valeurs retournées seront des probabilités. Comment attribuer une classe à l'aide ces probabilités?

2. Déterminer l'accuracy sur le jeu de données d'entraînement et sur le jeu de données tests.
3. Les résultats, en terme d'accuracy, sont-ils stables lorsque l'on bouge le seuil de décision? Faites l'expérience.

3 Un modèle de régression simplifié

Reprenez les questions précédentes, en supprimant les variables non significatives et comparer les deux modèles.

Remarque 1. Pour améliorer la confiance que l'on a en les résultats observés et donc comparer plusieurs modèles, il faudrait normalement répéter les expériences plusieurs fois (10 fois en général) sur différents *splits* train et test, et calculer la valeur moyenne et l'écart-type sur ces 10 expériences.

Remarque 2. Il faudrait normalement *tuner* le seuil qui permet de prendre sa décision, *i.e.* affecter une données dans une classe ou une autre, il faudrait donc effectuer ce que l'on appelle une *cross-validation*.