La Cross Validation

Marko ARSIC / Rindra LUTZ

15/11/2020

Introduction

Le monde d'aujourd'hui est un monde connecté. Cette connectivité apporte son lot de d'informations diverses : ce sont les données, ou data en anglais.

Pour traiter ces données, de nouveaux métiers ont vu le jour. Ces nouveaux métiers font appel à des traitements spécifiques dans le domaine des données comme par exemple la data analyse, la data science ou encore le datamining.

Le Datamining regroupe des méthodes scientifiques destinées à l'exploration et l'analyse de données, à partir de grands volumes d'informations/de données, dans le but de créer de la valeur, comprendre des phénomènes, comprendre notre monde, et en particulier afin d'aider à prendre des décisions, anticiper des événements et agir.

Dans le Datamining, il existe principalement deux grandes familles de méthodes. Ce sont les méthodes descriptives et les méthodes prédictives.

Méthodes descriptives

Une méthode descriptive (dite non supervisée) correspond à la recherche de structure, de relation, de corrélation.

- Permet de mettre en évidence des informations non visibles simplement
- Permet de résumer, synthétiser les données
- Sans variable ou phénomène à expliquer a priori

Ci-dessous, un tableau des méthodes descriptives.

Analyse factorielle ***Projection dans un espace de dimension inférieure***	Analyse en composantes principales (variables continues), Analyse factorielle des correspondances (2 var. qualitatives), Analyse des correspondances multiples (+ 2 var. qualitatives)
Analyse typologique ***Regroupement en classes homogènes***	Méthodes hiérarchiques : classification ascendante/descendante hiérarchique Méthodes de partitionnement : centres mobiles, k-means, nuées dynamiques Classification neuronale (cartes de Kohonen : analyse typologique + réduction de dimensions)
Modèles combinatoires	Classification relationnelle (variables qualitatives) : peu répandu
Modèles à base de règles	Règles d'associations : détection de liens

Méthodes prédictives

Une méthode prédictive (dite supervisée) correspond à la modélisation et la prédiction d'un phénomène donné.

- Permet de définir un pattern (un modèle/une relation) pour expliquer un événement
- Permet d'extrapoler la cible
- Avec une variable/un événement à expliquer

Ci-dessous, un tableau des méthodes préditives.

Régression (linéaire, logistique, modèle linéaire général GLM, PLS)	
Arbres de décision (ou encore d'induction ou de segmentation)	Modèle à base de règles, sur variable à expliquer continue ou qualitative
Réseaux de neurones	Réseaux à apprentissage supervisé (perceptron multicouche, réseau de neurones profond)
Analyse discriminante	
SVM – Support Vecteur Machine	

Cross-Validation

Traitant le sujet de la cross-validation, qui est une régression logistique, nous nous concentrerons sur les méthodes prédictives, et donc plus particulièrement sur les méthodes de régression.

La régression logistique est une technique prédictive. Elle vise à construire un modèle permettant de prédire/expliquer les valeurs prises par une variable cible qualitative (le plus souvent binaire, on parle alors de régression logistique binaire, et si elle possède plus de 2 modalités, on parle de régression logistique polytomique) à partir d'un ensemble de variables explicatives quantitatives ou qualitatives (un codage est nécessaire dans ce cas).

Une fois que le modèle a été établi grâce à différents outils statistiques, il est alors nécessaire de valider sa fiabilité.

La cross validation pour mesurer la fiabilité du modèle

Lors de toute modélisation, il est nécessaire de définir :

• Une population d'apprentissage (Train) : pour entrainer le modèle
• Une population de test (Test) : pour mesurer, tester la performance et robustesse du modèle

Or, une véritable vérification via la cross validation demanderait de travailler ici avec 3 types d'échantillons :

• Train
• Valid
• Test

Cet échantillon complémentaire (Valid) permet par exemple de tester plusieurs modèles (en faisant varier les paramètres du modèle ou les variables) : on essaie plusieurs modèles sur le Train et on identifie le plus performant sur le Valid. On teste enfin sur l'échantillon de Test (totalement vierge) le pouvoir de généralisation/sur-apprentissage sur des données toutes fraîches.

Il est possible de sophistiquer la structure des échantillons de Train/Validation au travers de quelques méthodes de validation croisée. En effet, les résultats dépendent de la manière dont ont été construits les 3 sous-ensembles Train/Valid/Test.

Les 3 principales méthodes de validation croisée sont :
• LOOV (leave-one-out cross-validation) (= LKOV avec k=1) • LKOV (leave-k-out cross-validation) • k-fold cross-validation

LOOV (leave-one-out cross-validation)	KLOV (leave-k-out cross-validation)	k-fold cross-validation
Sortie d'1 observation i de l'ensemble de données (à l'exception des données de test qui est intouché) et calcul du modèle sur les m-1 données restantes. On prédit i et on calcule l'erreur de prévision. On répète ce processus pour toutes les valeurs de i = 1 à m. Les m erreurs de prévision peuvent alors être utilisées pour évaluer la performance du modèle en validation croisée	Cette méthode fonctionne selon le même principe que LOOV, sauf que l'on sort non pas une, mais k observations à prédire à chaque étape. Le processus est répété de façon à avoir réalisé tous les découpages possibles en données de modélisation/de prévision.	Les observations sont aléatoirement divisées en k sous-échantillons de tailles égales, dont un est utilisé pour la prévision et les k-1 restants pour l'estimation du modèle. Contrairement à la KLOV, le processus n'est répété que k fois. C'est une méthode non exhaustive.

La cross validation permet aussi de déterminer des paramètres du modèle : on met en compétition k « sous-modèles » dont on mesure la performance pour déterminer le paramètre testé dont la performance du modèle est la meilleure.

Parlons rapidement du problème du sur-apprentissage, problème largement présent en modélisation.

Le sur apprentissage : problématique majeure en modélisation

Un modèle trop complexe, intégrant trop d'inputs et « épousant » trop les données d'apprentissage amènera donc une très bonne performance sur l'échantillon d'apprentissage (par construction), mais aura trop appris, notamment les bruits ou cas aberrants.
Il sera alors moins performant sur des données qui n'ont pas servi à la construction du modèle, c'est-à-dire sur les données sur lesquelles on souhaite faire la prédiction.
L'enjeu est donc de trouver le bon niveau de sophistication pour obtenir un bon niveau de performance sur l'échantillon d'apprentissage et sur l'échantillon de test.

Il n'y a pas sur-apprentissage lorsque la performance du modèle en Test est légèrement plus faible que celle en Train. Un écart trop grand est signe de **sur-apprentissage**.

Exemple pratique

Téléchargeons les packages tidyverse pour une manipulation et une visualisation des données plus faciles, ainsi que caret pour calculer facilement les méthodes de validation croisée. Nous pouvons ensuite les appeler comme suit :

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse
1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

Nous allons ici utiliser la table swiss, présente sous R

```
# Téléchargement des données
data("swiss")
# Inspecter les données
sample_n(swiss, 3)
```

```
##         Fertility Agriculture Examination Education Catholic
Infant.Mortality
## Conthey      75.5        85.9           3         2    99.71
15.1
## Broye        83.8        70.2          16         7    92.85
23.6
## Glane        92.4        67.8          14         8    97.16
24.9
```

Après avoir construit un modèle, nous souhaitons déterminer la précision de ce modèle sur la prédiction du résultat de nouvelles observations non utilisées pour construire le modèle. En d'autres termes, nous voulons estimer l'erreur de prédiction.

Pour ce faire, la stratégie de base consiste donc à :

- Construire le modèle sur un ensemble de données d'entraînement
- Appliquer le modèle sur un nouvel ensemble de données de test pour faire des prévisions
- Calculer les erreurs de prédiction

Il est ensuite temps de passer à l'étape suivante : la validation croisée.

Généralement, on utilise la validation croisée répétée k fois pour estimer le taux d'erreur de prédiction. Elle peut être utilisée dans les paramètres de régression et de classification. Il est toutefois à noter que d'autres méthodes sont applicables, dont les noms ont été mentionnés plus haut.

Une autre alternative à la validation croisée que nous n'avons pas évoqué est la méthode de rééchantillonnage bootstrap, qui consiste à sélectionner de manière répétée et aléatoire un échantillon de n observations à partir de l'ensemble de données original, et à évaluer la performance du modèle sur chaque copie.

La méthode de validation croisée k-fold évalue la performance du modèle sur différents sous-ensembles de données de formation et calcule ensuite le taux moyen d'erreur de prédiction. L'algorithme est le suivant :

1- Diviser aléatoirement l'ensemble de données en k sous-ensembles (ou k fois) (par exemple 5 sous-ensembles)
2- Réserver un sous-ensemble et former le modèle sur tous les autres sous-ensembles
3- Tester le modèle sur le sous-ensemble réservé et enregistrer l'erreur de prédiction
4- Répétez ce processus jusqu'à ce que chacun des k sous-ensembles ait servi d'ensemble de test.
5- Calculez la moyenne des k erreurs enregistrées. C'est ce qu'on appelle l'erreur de validation croisée qui sert de mesure de performance pour le modèle.

La validation croisée ou Cross validation (CV) par K fois est une méthode robuste pour estimer la précision d'un modèle.

L'avantage le plus évident de la CV k-fois par rapport à la LOOCV est le calcul. Un avantage moins évident, mais potentiellement plus important de la CV k-fold est qu'elle donne souvent des estimations plus précises du taux d'erreur des tests que la LOOCV.

La question typique est la suivante : comment choisir la bonne valeur de k ?

Une valeur inférieure de K est plus biaisée et donc indésirable. En revanche, une valeur plus élevée de K est moins biaisée, mais peut souffrir d'une grande variabilité. Il n'est pas difficile de voir qu'une valeur plus faible de k (disons k = 2) nous conduit toujours vers une approche de validation d'ensemble, alors qu'une valeur plus élevée de k (disons k = nombre de points de données) nous conduit à une approche de LOOCV.

Dans la pratique, on effectue généralement une validation croisée k fois plus élevée en utilisant k = 5 ou k = 10, car il a été démontré empiriquement que ces valeurs donnent des estimations du taux d'erreur de test qui ne souffrent ni d'un biais trop élevé ni d'une variance très élevée.

L'exemple suivant utilise une validation croisée décuplée pour estimer l'erreur de prédiction.

```
# Définition de l'échantillon d'entraînement
set.seed(123)
train.control <- trainControl(method = "cv", number = 10)
# Entraîner le modèle
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
# Résultats résumés
print(model)
```

```
## Linear Regression
##
## 47 samples
##  5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 42, 42, 42, 42, 42, 44, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.424916  0.6922072  6.31218
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Le processus de division des données en k-fold peut être répété un certain nombre de fois, c'est ce qu'on appelle la validation croisée répétée en k-fold.

L'erreur finale du modèle est considérée comme l'erreur moyenne du nombre de répétitions.

L'exemple suivant utilise une validation croisée décuplée avec 3 répétitions :

```
# Définiiton de l'échantillon d'entraînement
set.seed(123)
train.control <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 3)
# Entraîner le modèle
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
# Résultats résumés
print(model)
```

```
## Linear Regression
##
## 47 samples
##  5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 42, 42, 42, 42, 42, 44, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.357186  0.6992415  6.15871
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

A noter : lorsque l'on compare deux modèles, celui qui produit l'échantillon d'essai le plus faible RMSE est le modèle préféré.

La RMSE et la MAE sont mesurées à la même échelle que la variable de résultat. En divisant la RMSE par la valeur moyenne de la variable de résultat, on obtient le taux d'erreur de prédiction, qui doit être aussi faible que possible.