Untitled

Azat ALEKSANYAN

2/9/2020

Contents

Méthodologie

Afin de répondre au mieux à notre problématique nous avons fait le choix d'utiliser plusieurs algorithms différentes pour analyser nos données.

kNN /k-ppv/

L'algorithme des kNN consiste a trouver est les k observations x_i les plus proches de l'observation x' a classifier. Ensuite, il faut definir $f(x')$ en fonction des reponses y_i des kNN. Pour la regression c'est la valeur moyenne et pour la classification - la vote majoritaire. Les questions ouvertes pour cette algorithme est la choix du **critere de distance** et de la **valeur de k**.

- kNN pour la regression:

$$\tilde{f}(x) = moyenne(y_i | i \in N_k(x))$$

$$\Rightarrow \text{approxime directement la fonction de regression } E[Y|X]$$

Nature de l'approximation

1. espérance → moyenne empirique
2. valeur ponctuelle → voisinage (dans le conditionnement)

- kNN pout la classification:

$$\tilde{f}(x) = majorite(y_i | i \in N_k(x)) = arg \max_{l=1,\dots,K} \tilde{P}_l = \frac{1}{k} \sum_{i \in N_k(x))} 1(y_i = l)$$

$$\Rightarrow \text{approxime directement le classifieur de Bayes:}$$

$$arg \max_{l=1,\dots,K} P(Y = C_l | X = x)$$

Nature de l'approximation

1. probabilité → proportion empirique
2. valeur ponctuelle → voisinage (dans le conditionnement)

CART - Classification & Regression Trees

De base, les algorithmes d'arbre de décision ne sont rien d'autre que des instructions if-else qui peuvent être utilisées pour prédire un résultat basé sur des données. La méthodologie CART ou Arbres de classification et de régression fait référence à ces deux types d'arbres de décision.

- arbres de classification

Un arbre de classification est un algorithme dans lequel la variable cible est fixe ou catégorielle. L'algorithme est ensuite utilisé pour identifier la «classe» dans laquelle une variable cible se situerait le plus probablement. Un arbre de classification divise l'ensemble de données en fonction de l'homogénéité des données. Disons, par exemple, qu'il y a deux variables; revenu et âge; qui déterminent si un consommateur achètera ou non un type particulier de téléphone.

Si les données de formation montrent que 95% des personnes de plus de 30 ans ont acheté le téléphone, les données y sont divisées et l'âge devient un nœud supérieur dans l'arbre. Cette division rend les données «pures à 95%». Des mesures d'impuretés comme l'entropie ou l'indice de Gini sont utilisées pour quantifier l'homogénéité des données en ce qui concerne les arbres de classification.

- arbres de régression

Un arbre de régression fait référence à un algorithme dans lequel se trouve la variable cible et l'algorithme utilisé pour prédire sa valeur. Dans un arbre de régression, un modèle de régression est ajusté à la variable cible en utilisant chacune des variables indépendantes. Après cela, les données sont divisées en plusieurs points pour chaque variable indépendante.

À chacun de ces points, l'erreur entre les valeurs prédites et les valeurs réelles est mise au carré pour obtenir une «somme des erreurs au carré» (SEC). Le SEC est comparée entre les variables et la variable ou le point qui a le SEC le plus bas est choisi comme point de partage. Ce processus se poursuit récursivement.

rfRanger ???

Random Forest

À la base, la méthode des forêts aléatoires est basée sur le bagging /aggrégation par bootstrap/, il est très performant sur de nombreux problèmes et facile à paramétrer. Principe d'algorithmes et de découper l'espace d'entrée en région, estimer et prédire une valeur par région. Pour la régression on choisit une valeur moyenne dans la région, et pour la classification on choisit la classe majoritaire. Les avantages de cette méthode sont l'interprétabilité du modèle et le mécanisme de prédiction proche du processus humain.

XGBoost

La bibliothèque XGBoost implémente l'algorithme d'arbre de décision de renforcement du gradient. Le boosting est une technique d'ensemble où de nouveaux modèles sont ajoutés pour corriger les erreurs commises par les modèles existants. Les modèles sont ajoutés séquentiellement jusqu'à ce qu'aucune autre amélioration ne puisse être apportée.

L'amplification du gradient est une approche où de nouveaux modèles sont créés qui prédisent les résidus ou les erreurs des modèles précédents, puis additionnés pour faire la prédiction finale. Il est appelé boosting de gradient car il utilise un algorithme de descente de gradient pour minimiser la perte lors de l'ajout de nouveaux modèles.

Cette approche prend en charge à la fois les problèmes de modélisation prédictive de régression et de classification.

glmNetL2 - Ridge Regression

Régression linéaire avec perte quadratique et pénalité L2 :

$$(w^*, b^*) = \underset{w\in\mathbb{R}^l, b\in\mathbb{R}}{argmin} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda ||w||_2^2$$

$$= \underset{w\in\mathbb{R}^l, b\in\mathbb{R}}{argmin} \sum_{i=1}^{n} (y_i - b - \sum_{j=1}^{p} w_j x_{ij} + \lambda \sum_{j=1}^{p} w_j^2$$

la régression ridge impose donc une contrainte sur les coefficients (w). Le terme de pénalité (λ) régularise les coefficients de telle sorte que si les coefficients prennent de grandes valeurs, la fonction d'optimisation est pénalisée. Ainsi, la régression ridge réduit les coefficients et contribue à réduire la complexité du modèle et la multi-colinéarité.

Pour le résoudre il faut centrer les variables $\rightarrow \tilde{x}_{ij} = x_{ij} - \bar{x}$

1. on estime alors l'intercept b par $b^* = \hat{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
2. on obtient w avec le meme probleme sans intercept
3. solution : $w^* = (X^T X + \lambda I)^{-1} X^T y$ *ou* $X[i,j] = \tilde{x}_{ij}$

glmNetL1 - Lasso Regression

Régression linéaire avec perte quadratique et pénalité L1 :

$$(w^*, b^*) = \underset{w\in\mathbb{R}^{'}, b\in\mathbb{R}}{argmin} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda ||w||_1$$

$$= \underset{w\in\mathbb{R}^{'}, b\in\mathbb{R}}{argmin} \sum_{i=1}^{n} (y_i - b - \sum_{j=1}^{p} w_j x_{ij} + \lambda \sum_{j=1}^{p} |w_j|$$

Si on regarde l'equation en detail, in peut voir que la seule différence est qu'au lieu de prendre le carré des coefficients, les grandeurs sont prises en compte. Ce type de régularisation (L1) peut conduire à des coefficients nuls, c'est-à-dire que certaines caractéristiques sont complètement négligées pour l'évaluation des résultats. Ainsi, la régression Lasso aide non seulement à réduire le sur-ajustement, mais elle peut également nous aider à sélectionner les variables.

Du coup, par rapport à pénalité ridge :

- même effet de régularisation : shrinkage des coefficients
- mais conduit à des coefficients exactement = 0

⇒ solution parcimonieuse (sparse) : sélection de variables

AdaBoost Classification Trees ????

AdaBoost M1

AdaBoost (boosting adaptatif) est un algorithme d'apprentissage assembliste qui peut être utilisé pour la classification ou la régression. Bien qu'AdaBoost résiste mieux au surajustement qu'un grand nombre d'algorithmes de machine learning, il est parfois sensible aux données bruitées et aux valeurs aberrantes.

AdaBoost est désigné comme adaptatif car il utilise de nombreuses itérations pour générer un apprenant composite fort. AdaBoost crée l'apprenant fort (un classifieur bien corrélé au classifieur correct) en ajoutant de manière itérative des apprenants faibles (un classifieur légèrement corrélé au classifieur correct). Lors de chaque session d'apprentissage, un nouvel apprenant faible est ajouté à l'ensemble et un vecteurpondération est ajusté afin de mettre l'accent sur les exemples ayant été classés de manière incorrecte lors des sessions précédentes. Par conséquent, le classifieur obtenu est doté d'une meilleure précision queclassifieurs des apprenants faibles.

AdaBoost.M1 represente les algorithmes originaux de classification binaire.

Optimisation du modèle - Validation Croisée

L'optimisation du modèle signifie l'optimisation hyperparamétrique, c'est-à-dire il faut trouver le bon niveau de complexité du modèle, qui est en général difficile de choisir a priori. Pour l'optimisation on utilise la methode de la validation croissee.

Au préalable, on définit un jeu de test pour évaluer les performances du modèle. Ensuite, on découpe led'apprentissage en K parties - les folds.

pour $k = 1$ *a* K, il faut :

- mettre de côté la $k - ième$ fold
- apprendre le modèle sur les $K - 1$ folds restantes
- appliquer le modèle sur les données de la $k - ième$ fold

A la fin, on évalue les performances du modèle.

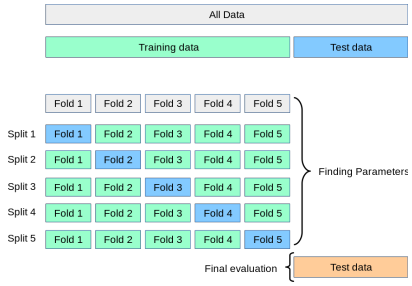

Figure 1: Validation Croisée