M2 - Probabilités et statistiques des nouvelles données

Projet 2 : Regression with reject option

Auteurs :
Confiac Hendrick
Wague Yakhoub

Novembre 2021

Préface

Ce mémoire aura pour but de mettre en avant la méthode de régression avec option de rejet. Méthode que l'on peut qualifier d'extension de la classification avec option de rejet. Le but étant de trouver un estimateur optimal du modèle, afin de s'abtenir de toute prédiction dans le cas où il est difficle de prendre une décision. Il s'agira alors de limiter l'erreur d'une décision érronée dans des domaines où par exemple, la santé d'un individu est en jeu. Au cours de cette étude nous définirons dans un premier temps le cadre de régression avec option de rejet en y présentant par la suite et de manière explicite l'estimateur optimale de cette méthode sous certaines conditions. Nous utiliserons ensuite un jeu de données afin de mettre en avant l'intérêt de cette méthode.

Table des matières

Chapitre 1

Régression avec option de rejet :

Introduisons quelques notations préliminaires.
Soit (X, Y) un couple aléatoire prenant ses valeurs dans $\mathbb{R}^d \times \mathbb{R}$: où X correspond aux *features* et Y la prédiction en fonction de X. Soit $\mathbb{P}$ la distribution jointe de (X, Y) et $\mathbb{P}_X$ la distribution marginale de la caractéristique X. Soit $x \in \mathbb{R}^d$, on introduit la fonction de régression $f^* = \mathbb{E}[Y|X = x]$ ainsi que la fonction de variance conditionnel $\sigma^2(x) = \mathbb{E}[(Y - f^*(X))^2|X = x]$. De plus, on note $||.||$ la distance l'euclidienne sur $\mathbb{R}^d$. Enfin, $|.|$ représente la cardinalité lorsqu'il s'agit d'un ensemble fini.

1.1 Prédicteur avec option de rejet :

Définition :

Un prédicteur avec "l'option de rejet", noté Γ_f est un opérateur associé à une fonction mesurable f à valeur dans $\{\emptyset, f(x)\}$; si la prédiction est effective alors il coïncide avec la fonction f (qui est un donc predicteur) sinon il n'y a pas eu de prédiction au point x. Deux caractéristiques seront mises en exergue ; la première est le taux de rejet définit comme suit :

$$r(\Gamma_f) = \mathbb{P}(|\Gamma(X)| = 0)$$

La seconde est l'erreur L_2 en cas de prédiction effective.

$$Err(\Gamma_f) = \mathbb{E}[(Y - f(X))^2 | \Gamma_f(X) = 1|]$$

Le but de la regression avec option de rejet est de construire un estimateur ayant un taux de rejet faible et admettant une erreur dans L_2 la plus petite que possible.

Pour ce faire, nous allons définir le risque suivant :

$$\mathcal{R}_\lambda(\Gamma_f) = \mathbb{E}[(Y - f(X))^2 \mathbb{1}_{\{|\Gamma_f(X)|=1)\}}] + \lambda r(\Gamma_f)$$

où λ est le "compromis" entre l'erreur et le taux de rejet. On dit que λ est le prix à payer pour utiliser l'option de rejet.

Ainsi, trouver un estimateur optimal pour la regrssion avec option de rejet revient donc à minimiser $\mathcal{R}_\lambda$.

Prédicteur optimal

On note $\mathcal{M}$ l'ensemble des fonction mesurables. Notons également, Υ_f l'ensemble de tous les prédicteurs avec option de rejet qui repose sur f. Ainsi le prédicteur optimale de régression avec option de rejet s'écrit :

$$\Gamma_\lambda^* \in \underset{\substack{f \in \mathcal{M} \\ \Gamma_f^* \in \Upsilon_f}}{\operatorname{argmin}} \mathcal{R}_\lambda(\Gamma_f)$$

.

Nous allons chercher à écrire Γ_λ^* de manière explicite, ce qui nous mène à la proposition suivante :

Proposition :

Soient $\lambda \geq 0$ et Γ^*_λ défini comme précédement. On a alors :

1. $$\Gamma^*_\lambda(X) = \begin{cases} \{f^*(X)\} & \text{si } \sigma^2(X) \leq \lambda \\ \emptyset & \text{sinon.} \end{cases}$$

2. Pour tout $\lambda \leq \lambda'$, on a :
$$Err(\Gamma^*_\lambda) \leq Err(\Gamma^*_{\lambda'})$$
$$\text{et } r(\Gamma_\lambda) \geq r(\Gamma_{\lambda'})$$

Deux points sont à soulignés dans cette proposition :

D'une part, le predicteur repose sur le seuille de la variance conditionnelle σ^2

D'autre part, l'erreur et le taux de rejet du predicteur optimale "fonctionnent" de manière opposée. En effet, pour un λ petit, le risque $\mathcal{R}_\lambda$ ainsi que l'erreur $Err()$ sont petits avec un taux de rejet pouvant être arbritrairement grand. Par ailleurs, quelle que soit la valeur de λ, aussi optimale soit-elle, il nous est impossible de contrôler à la fois l'erreur et la partie dépendante de λ dans la fonction de risque $\mathcal{R}_\lambda$.

Nous allons donc étudier le cas sous la contrainte d'un taux de rejet au moins borné, c'est-à-dire pour un certain taux de rejet $\epsilon \in (0, 1)$, $r(\Gamma_f) \leq \epsilon$.

Prédicteur optimal avec taux de rejet borné

Sous la contrainte ci-dessus, on écrit notre estimateur optimal dans ce cadre :

$$\Gamma^*_\epsilon \in argmin\{Err(\Gamma_f); r(\Gamma_f) \leq \epsilon\}$$

que nous allons chercher à réécrire de manière explcite. D'où la proposition suivante :

Proposition :

À supposer que la fonction de répartiton de σ^2 noté F_{σ^2} soit continue ; considérons $\epsilon \in (0, 1)$ et notons $\lambda_\epsilon = F^{-1}_{\sigma^2}(1 - \epsilon)$, avec $F^{-1}_{\sigma^2}(u) = \inf\{t \in \mathbb{R} : F_{\sigma^2}(t) \geq u\}$ pour tout $u \in (0, 1)$. Alors on a

$$\Gamma^*_\epsilon = \Gamma^*_{\lambda_\epsilon}$$

Ainsi on écrit :

$$\Gamma^*_\epsilon(X) = \begin{cases} \{f^*(X)\} & \text{si } F_{\sigma^2}(\sigma^2(X)) \leq 1-\epsilon \\ \emptyset & \text{sinon.} \end{cases}$$

Et puisque $r(\Gamma^*_\epsilon) = \mathbb{P}(|\Gamma^*_\epsilon(X)| = 0) = \mathbb{P}(\sigma^2(X) \geq \lambda_\epsilon) = \mathbb{P}(F_{\sigma^2}(\sigma^2(X)) \geq 1-\epsilon) = \epsilon$ par continuité de F_{σ^2},

On a alors dans ce cadre une fonction de risque s'écrivant :

$$\mathcal{R}_{\lambda_\epsilon}(\Gamma^*_\epsilon) = \mathbb{E}[(Y - f^*(X))^2 \mathbb{1}_{\{|\Gamma^*_\epsilon(X)|=1)\}}] + \lambda_\epsilon \epsilon$$

Chapitre 2

Applications numériques :

Nous allons utiliser l'algorithme des forêts aléatoires pour appliquer notre méthodologie sur un jeu de données. Plus précisément, à partir du Dataset 'airfoil' nous allons séparer les données en 3 parties :

La première correspondant à 50% du Dataset, nous permettras d'entrainer notre modèle afin de pouvoir estimer

$$f^*, \sigma^2$$

La deuxième correspondant à 20% du Dataset, nous permettras d'estimer

$$F_{\sigma^2}(\sigma^2)$$

La troisième correspondant à 30% du Dataset, sera notre "test set" et nous permettras d'estimer le taux de rejet aisi que l'erreur L_2

1 Application numérique avec le RandomForest

Nous allons appliquer la méthodologie sur le data set : Airfoll à l'aide de l'algo de Random Forest

[545]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
```

[546]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import validation_curve
from sklearn.model_selection import GridSearchCV
```

[547]:
```
dataset = pd.read_excel (r'C:\Users\confi\Downloads\airfoil.xls')
```

[548]:
```
dataset.head()
```

[548]:

	Frequency, in Hertzs.	Angle of attack, in degrees.	Chord length, in meters.	Free-stream velocity, in m/s
0	800	0.0	0.3048	71.3
1	1000	0.0	0.3048	71.3
2	1250	0.0	0.3048	71.3
3	1600	0.0	0.3048	71.3
4	2000	0.0	0.3048	71.3

```
   Suction side displacement thickness (m)  \
0                                 0.002663
1                                 0.002663
2                                 0.002663
3                                 0.002663
4                                 0.002663

   Scaled sound pressure level, in decibels
0                                   126.201
1                                   125.201
2                                   125.951
3                                   127.591
4                                   127.461
```

[ ]:

[549]:
```
X = dataset.drop('Scaled sound pressure level, in decibels', axis=1)
```

[550]:
```
X.head()
```

[550]:
```
   Frequency, in Hertzs.  Angle of attack, in degrees.  \
0                    800                           0.0
1                   1000                           0.0
2                   1250                           0.0
3                   1600                           0.0
4                   2000                           0.0

   Chord length, in meters.  Free-stream velocity, in m/s  \
0                    0.3048                          71.3
1                    0.3048                          71.3
2                    0.3048                          71.3
3                    0.3048                          71.3
4                    0.3048                          71.3

   Suction side displacement thickness (m)
0                                 0.002663
1                                 0.002663
2                                 0.002663
3                                 0.002663
4                                 0.002663
```

[ ]:

[551]:
```
Y = dataset['Scaled sound pressure level, in decibels']
```

[552]:
```
Y.head()
```

[552]:
```
0    126.201
1    125.201
2    125.951
3    127.591
4    127.461
Name: Scaled sound pressure level, in decibels, dtype: float64
```

[553]:
```
X.iloc[:,].shape
```

[553]:
```
(1503, 5)
```

[563]:
```
X_train, x_test, Y_train, y_test = train_test_split(X,Y,test_size=0.3)
```

[564]:
```
X_train.shape
```

[564]:
```
(1052, 5)
```

Paramètres

[565]:
```
kn = int(len(X_train)*0.71)   #nombre d'observation pour estimer f et sigma
u = 10**(-10)                 #borne sup de l'intervalle où la perturbation est␣
 ↪tirée
N = 306                       #nombre d'observation pour estimer F_sigma
```

[566]:
```
#séparation labeled et unlabeled

X_train_labeled = X_train.iloc[0:kn]
Y_train_labeled = Y_train[0:kn]
X_train_unlabeled =  X_train.iloc[kn:,]
Y_train_unlabeled =  Y_train[kn:,]
```

[567]:
```
model = RandomForestRegressor(500)
model.fit(X_train_labeled,Y_train_labeled)
model.score(x_test,y_test)
```

[567]:
```
0.9030296877244637
```

[568]:
```
#Entrainement pour évaluer E[Y²|X]  avec RandomForest

Y_train_labeled_sigma = Y_train_labeled**2
y_test_sigma = y_test**2


model.fit(X_train_labeled,Y_train_labeled_sigma)
model.score(x_test,y_test_sigma )
```

[568]:
```
0.897198010061431
```

2 epsilon predictor

[569]:
```
#X_train, x_test, Y_train, y_test = train_test_split(X,Y,test_size=0.3)
def epsilon_predictor (X_train, Y_train , kn, N, x_test, y_test,␣
↪taux_de_rejet,optimale):



        #séparation labeled et unlabeled
    X_train_labeled = X_train.iloc[0:kn]
    Y_train_labeled = Y_train[0:kn]
    X_train_unlabeled =  X_train.iloc[kn:,]
    Y_train_unlabeled =  Y_train[kn:,]

    # entrainement pour estimer f_etoile et sigma
    model = RandomForestRegressor(67)
    model.fit(X_train_labeled,Y_train_labeled)

    #estimation de f_etoile
    f_etoile_labeled = model.predict(x_test) #estimation de E[Y|X] a partir␣
↪des données labeled



    # estimation de sigma_labeled et de F (fonction de répartition)  en␣
↪estimant : E[Y²|X] - (E[Y|X])²
    ␣
↪#############################################################################################

    f_etoile_unlabeled = model.predict(X_train_unlabeled) #estimation de␣
↪E[Y|X] a partir des données unlabeled

    #Entrainement pour estimer E[Y²|X] à partir des données labeled
    Y_train_labeled_sigma = Y_train_labeled**2
    y_test_sigma = y_test**2

    #Entrainement pour estimer E[Y²|X]
    model.fit(X_train_labeled,Y_train_labeled_sigma)

    #ESTIMATION  de E[Y²|X]
    f_etoile_unlabeled_carré = model.predict(X_train_unlabeled)
    f_etoile_labeled_carré  = model.predict(x_test)

    # Finalement...
    sigma_unlabeled = abs(f_etoile_unlabeled_carré - f_etoile_unlabeled**2) ␣
↪      #sigma à partir des données unlabeled
    sigma_test = abs(f_etoile_labeled_carré - f_etoile_labeled**2)          ␣
↪         #sigma à partir des données labeled
```

```
        ## Estimation de F_sigma

        index = list()
        vect_ksy = np.random.uniform(0, u ,N)                    #perturbation
        ksy = np.random.uniform(0,u,len(sigma_test))

        Fonction_repar= Err = np.zeros(len(sigma_test))
        for i in np.arange(len(sigma_test)):

            #F_sigma(sigma(x))
            Fonction_repar[i] = (len(sigma_unlabeled[sigma_unlabeled+np.random.
↪uniform(0, u ,1)  <sigma_test[i] + np.random.uniform(0, u ,1)])/N)

            if Fonction_repar[i]<1-taux_de_rejet:          #taux_de_rejet est␣
↪l'epsilon du poly
                Err[i] = (y_test.iloc[i]- f_etoile_labeled[i])**2


        ## Estimation de l'erreur L2 et du taux de rejet
        Err_chapeau = Err                        #Erreur L2 des prédictions␣
↪effectives

        cardinal_Gamma = len(Fonction_repar[Fonction_repar<1-taux_de_rejet])

        r_chapeau = 1- cardinal_Gamma/451       #taux de rejet (nombre total de␣
↪predictions - nombre de prediction_effective)/nbr pred totale

        return(r_chapeau, Err_chapeau.mean())
```

[570]:
```
r_chapeau,Err_chapeau = epsilon_predictor (X_train, Y_train , kn, N, x_test,␣
↪y_test, taux_de_rejet=0,optimale=500)
```

[572]:
```
epsilon = np.arange(0,10)/10
matrice_r_chapeau =  np.zeros((100,10))
matrice_Err_chapeau = np.zeros((100,10))
```

[573]:
```
for eps in np.arange(10):

    for i in np.arange(100):

        a,b=epsilon_predictor (X_train, Y_train , kn, N, x_test, y_test,␣
↪taux_de_rejet= epsilon[eps],optimale=500)
```

```
        matrice_r_chapeau[i][eps] = a

        matrice_Err_chapeau[i][eps] = b
```

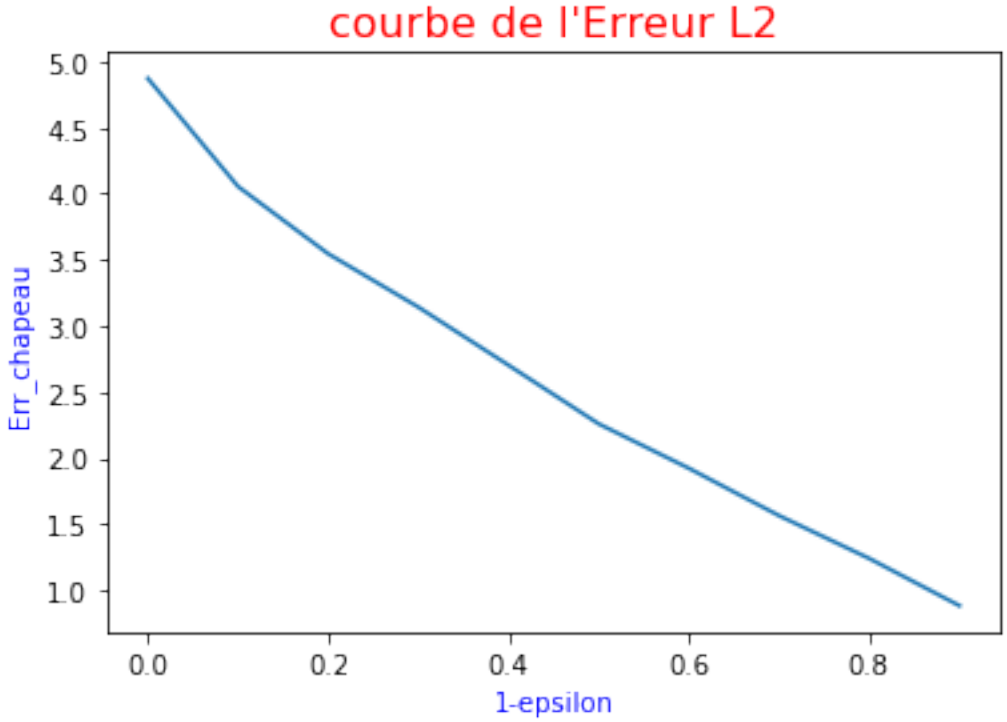
[579]:
```
pd.DataFrame({'1-epsilon':1-epsilon,'1-r_chapeau': 1-matrice_r_chapeau.
↪mean(axis=0),'Err_chapeau':matrice_Err_chapeau.mean(axis=0)})
```

[579]:

	1-epsilon	1-r_chapeau	Err_chapeau
0	1.0	0.440887	4.871731
1	0.9	0.392483	4.054114
2	0.8	0.339268	3.545186
3	0.7	0.287716	3.141667
4	0.6	0.235144	2.702330
5	0.5	0.186741	2.258721
6	0.4	0.132616	1.922738
7	0.3	0.084545	1.564570
8	0.2	0.050000	1.240776
9	0.1	0.017450	0.883624

[575]:
```
plt.plot(epsilon,matrice_Err_chapeau.mean(axis=0))
plt.title("courbe de l'Erreur L2",size=16,c='r')
plt.xlabel("1-epsilon", size = 10,c='b')
plt.ylabel("Err_chapeau", size = 10,c='b')
```

[575]:
```
Text(0, 0.5, 'Err_chapeau')
```



2.1 Remarques:

Nous constatons de par nos résultats que plus epsilon est petit plus l'erreur L2 l'est aussi. Sur la courbe ci-dessus décroît suivant que epsilon devient petit. Par ailleurs nous pouvons affirmer que la prédiction avec option de rejet nous donne une prédiction plus précise. En enffet, sans rejet, c-a-d pour epsilon = 0, l'erreur L2 est 2 fois plus grande que si on avait rejeté à 50% (epsilon=0.5).

2.1.1 CONCLUSION:

Nous avons établi une méthode afin de pouvoir utiser la régression avec option de rejet (avec rejet fixé) en ayant choisi l'algorithme 'RandomForest'. Il en résulte une précision de la prédiction plus importante, avec en contrepartie une abscence de prédiction pour certaines observations. Ceci permettant de réduire le risque d'erreur de ces prédictions lorsqu'il s'agira de prendre des décisions très importantes.

[ ]:

Bibliographie

[1] H.MOHAMED, Cours Statistiques en Grande Dimension , *Université Gustave Eiffel*(2021)

[2] Christophe Denis , Mohamed Hebiri et Ahmed Zaoui, Regression with reject option and application to kNN , *Université Gustave Eiffel*