TP ozone : Modèle linéaire gaussien, binomial, courbe ROC

Résumé

Comparaison sur le même jeu de données des qualités de prévision de plusieurs modèles obtenus par analyse de la covariance et régression logistique. Utilisation d'un échantillon test et courbe ROC.

1 Introduction

1.1 Objectif

L'objectif, sur ces données, est d'améliorer la prévision déterministe (`MOCAGE`), calculée par les services de MétéoFrance, de la concentration d'ozone dans certaines stations de prélèvement. Il s'agit d'un problème dit d'*adaptation statistique* d'une prévision locale de modèles à trop grande échelle en s'aidant d'autre variables également prévues par MétéoFrance, mais à plus petite échelle (température, force du vent...). Plus précisément, deux variables peuvent être prévues : soit la concentration quantitative d'ozone, soit le dépassement (qualitatif) d'un certain seuil fixé à $150\mu g$ [1]. Dans chaque cas, deux approches sont considérées : soit prévoir la concentration quantitative puis en déduire l'éventuel dépassement ou bien prévoir directement le dépassement. Dans le premier cas, il s'agit d'abord d'une régression tandis que dans le deuxième il s'agit d'un problème de discrimination ou de régression logistique. Question : quelles sont les meilleures méthodes et stratégies pour prévoir la concentration d'ozone du lendemain d'une part et l'occurence d'un pic de pollution d'autre part.

On se propose de tester différentes méthodes : régression logistique, analyse discriminante, réseau de neurones, arbre de décision, agrégation d'arbres (bagging, boosting, random forest), SVM. L'objectif final, à ne pas perdre de vue, est la comparaison de ces méthodes afin de déterminer la plus efficace pour répondre au problème de prévision. Ceci passe par la mise en place d'un protocole très strict afin de s'assurer d'un minimum d'objectivité pour cette comparaison.

Penser à conserver dans des fichiers les commandes successives utilisées ainsi que les principaux résultats tableaux et graphes.

Toutes les opérations sont réalisées dans R avec l'appui de bibliothèques complémentaires éventuellement à télécharger (`mlbench`, `MASS`, `boot`, `class`, `e1071`, `rpart`, `nnet`, `ipred`, `gbm`, `randomForest`). D'autres logiciels sont bien évidemment utilisables (Splus, SAS, SPSS...). Néanmoins, dans le cas de SAS, cela nécessiterait l'accès au module Enterprise Miner pour certaines méthodes (arbres, neurones) encore peu diffusé car très cher. De plus certaines procédures du module classique SAS/Stat ne comportent pas d'option de sélection automatique de variables quantitatives et qualitatives.

1.2 Rappel : protocole de comparaison

La démarche mise en œuvre enchaîne les étapes suivantes :

1. Après une éventuelle première étape descriptive uni ou multidimensionnelle visant à repérer les incohérences, les variables non significatives ou de distribution exotique, les individus non concernés... et à étudier les structures des données, procéder à un tirage aléatoire d'un échantillon *test* qui ne sera utilisé que lors de la *dernière étape*.
2. Sur la partie restante qui sera découpée en échantillon d'*apprentissage (des paramètres du modèle) et échantillon de* validation (pour estimation sans biais du taux de mauvais classés), optimiser les choix afférents à chacune des méthodes de régression et/ou discrimination :
   - variables et interactions à prendre en compte dans la régression linéaire ou logistique,
   - variables et méthode pour l'analyse discriminante,
   - nombre de nœuds dans l'arbre de régression ou de classification,
   - architecture (nombre de couches, de neurones par couche, fonctions de transferts, nombre de cycles...) du perceptron,
   - algorithme d'agrégation,
   - noyau et complexité des SVMs.

   Remarques :

1. Le seuil légal a été porté à $180\mu g$ mais celui-ci est bien trop rarement dépassé par les observations et serait trop difficile à prévoir par estimation d'un modèle.

  - En cas d'échantillon petit face au nombre des variables il est recommandé d'itérer la procédure de découpage par validation croisée, c'est le cas de ces données, afin de réduire la variance des estimations des erreurs de classement.

3. Comparaison finale des qualités de prévision sur la base du taux de mal classés pour le seul échantillon test qui est resté à l'écart de tout effort ou "acharnement" pour l'optimisation des modèles.

– *Attention, ne pas "tricher" en modifiant le modèle obtenu lors de l'étape précédente afin d'améliorer le résultat sur l'échantillon test !*
– Le critère utilisé dépend du problème : erreur quadratique, taux de mauvais classement, AUC (aire sous la courbe ROC)...

1.3 Prise en charge des données

Les données ont été extraites et mises en forme par le service concerné de MétéoFrance. Elles sont disponibles sur la plateforme moodle dans le fichier `ozone.dat` et décrites par les variables suivantes :

JOUR Le type de jour ; férié (1) ou pas (0) ;

O3obs La concentration d'ozone effectivement observée le lendemain à 17h locales correspondant souvent au maximum de pollution observée ;

MOCAGE Prévision de cette pollution obtenue par un modèle déterministe de mécanique des fluides (équation de Navier et Stockes) ;

TEMPE Température prévue par MétéoFrance pour le lendemain 17h ;

RMH2O Rapport d'humidité ;

NO2 Concentration en dioxyde d'azote ;

NO Concentration en monoxyde d'azote ;

STATION Lieu de l'observation : Aix-en-Provence, Rambouillet, Munchhausen, Cadarache et Plan de Cuques ;

VentMOD Force du vent ;

VentANG Orientation du vent.

```
# Lecture des données
ozone=read.table("ozone.dat",header=T)
# Vérification du contenu
summary(ozone)
# Changement du type de la variable jour
ozone[,"JOUR"]=as.factor(ozone[,"JOUR"])
```

Remarquer le type des variables. Il est nécessaire d'en étudier la distribution.

```
hist(ozone[,"O3obs"]);hist(ozone[,"MOCAGE"])
hist(ozone[,"TEMPE"]);hist(ozone[,"RMH2O"])
hist(ozone[,"NO2"]);hist(ozone[,"NO"])
hist(ozone[,"VentMOD"]);hist(ozone[,"VentANG"])
```

Des transformations sont proposées pour rendre certaines distributions plus symétriques et ainsi plus "gaussiennes".

```
ozone[,"SRMH2O"]=sqrt(ozone[,"RMH2O"])
ozone[,"LNO2"]=log(ozone[,"NO2"])
ozone[,"LNO"]=log(ozone[,"NO"])
```

Vérifier de l'opportunité de ces transformations puis retirer les variables initiales et construire la variable "dépassement de seuil".

```
ozone=ozone[,c(1:4,8:13)]
ozone[,"DepSeuil"]=as.factor(ozone[,"O3obs"]>150)
```

pour obtenir le fichier qui sera effectivement utilisé.

1.4 Extraction des échantillons apprentissage et test

Le programme ci-dessous réalise l'extraction du sous-ensemble des données d'apprentissage et de test. Attention, chaque étudiant ou binôme tire un échantillon différent ; il est donc "normal" de na pas obtenir les mêmes modèles.

Utiliser trois chiffres au hasard, en remplacement de "XXX" ci-dessous, comme initialisation du générateur de nombres aléatoires.

```
set.seed(XXX) # initialisation du générateur
# Extraction des échantillons
test.ratio=.2   # part de l'échantillon test
npop=nrow(ozone) # nombre de lignes dans les données
nvar=ncol(ozone) # nombre de colonnes
# taille de l'échantillon test
```

```
ntest=ceiling(npop*test.ratio)
# indices de l'échantillon test
testi=sample(1:npop,ntest)
# indices de l'échantillon d'apprentissage
appri=setdiff(1:npop,testi)
```

Construction des échantillons pour la régression (prévision de la concentration).

```
# construction de l'échantillon d'apprentissage
datappr=ozone[appri,-11]
# construction de l'échantillon test
datestr=ozone[testi,-11]
summary(datappr) # vérifications
summary(datestr)
```

Construction des échantillons pour la discrimination (dépassement du seuil).

```
# construction de l'échantillon d'apprentissage
datappq=ozone[appri,-2]
# construction de l'échantillon test
datestq=ozone[testi,-2]
summary(datappq) # vérifications
summary(datestq)
```

2 Prévison par modèle gaussien

Le premier modèle à tester est un simple modèle de régression linéaire mais, comme certaines variables sont qualitatives, il s'agit d'une analyse de covariance. D'autre part, on s'intéresse à savoir si des interactions sont à prendre en compte. Le modèle devient alors polynomiale d'ordre 2 ou quadratique.

2.1 Modèle linéaire

Le modèle de régression linéaire simple intégrant des variables qualitatives correspondant est estimé avec la fonction `aov` mieux adaptée à l'analyse de covariance.

```
# estimation du modèle sans interaction
reg.lm=aov(O3obs~.,data=datappr)
summary(reg.lm)
# Extraction des résidus et des valeurs ajustées
res.lm=reg.lm$residuals
fit.lm=reg.lm$fitted.values
# Graphe des résidus
plot(fit.lm,res.lm)
# Définition d'une fonction pour un graphe coloré et
# des échelles fixes sur les axes
plot.res=function(x,y,titre="titre")
{
plot(x,y,col="blue",xlim=c(0,300),ylim=c(-100,100),
ylab="Résidus",xlab="Valeurs predites",main=titre)
# points(x2,y,col="red")
abline(h=0,col="green")
}
plot.res(fit.lm,res.lm,"")
# Graphe des résidus au modèle MOCAGE
plot.res(datappr[,"MOCAGE"],
   datappr[,"MOCAGE"]-datappr[,"O3obs"],"")
```

Contrôler les graphes des résidus, que conclure de ce modèle ? L'étude suivante met en œuvre toutes les interactions d'ordre 2 pour ajuster le modèle :

2.2 Modèle quadratique

Le modèle de régression quadratique est estimé avec la fonction `glm` qui permet une sélection automatique de modèle. La méthode descendante est utilisée mais celle pas-à-pas pourrait également l'être.

```
# Estimation du modèle de toute interaction d'ordre 2
reg.glm=glm(O3obs~(.)^2,data=datappr)
# Recherche du meilleur modèle au sens
# du critère d'Akaïke par méthode descendante
reg.glm.step=step(reg.glm,direction="backward")
anova(reg.glm.step,test="F")
# Extraction des valeurs ajustées et des résidus
fit.glm=reg.glm.step$fitted.values
```

```
res.glm=reg.glm.step$residuals
# Graphe des résidus
plot.res(fit.glm,res.glm,"")
```

On remarque que la présence de certains interactions ou variables sont pertinentes au sens du critère d'Akaïke mais pas significative au sens du test de Fisher. Cette présence dans le modèle peut être plus finement analysée en considérant une estimation de l'erreur par validation croisée. L'idée est de retirer une à une les variables ou interactions les moins significatives et de voir comment se comporte la validation croisée. D'autre part, si la procédure pas-à-pas conduit à un modèle différent, l'estimation de l'erreur par validation croisée permet également d'optimiser le choix.

L'estimation des erreurs par validation croisée est calculée en utilisant une fonction proposée dans la bibliothèque `boot`.

```
library(boot) # chargement de la bibliothèque
# validation croisée  10-plis
# modèle complet
cv.glm(datappr, reg.glm, K=10)$delta[1]
# modèle "Akaïke"
cv.glm(datappr, reg.glm.step, K=10)$delta[1]
```

La première commande suivante permet de récupérer le modèle optimal avant de redéfinir celui-ci en retirant l'interaction la moins significative avant de réestimer l'erreur par validation croisée.

```
#la liste des effets du modèle optimal
reg.glm.step$formula
cv.glm(datappr, glm(O3obs ~ JOUR + MOCAGE + TEMPE +
   STATION + VentMOD + VentANG + LNO2 + LNO + SRMH2O +
   JOUR:VentMOD + JOUR:VentANG + MOCAGE:STATION +
   MOCAGE:VentMOD + MOCAGE:VentANG + MOCAGE:LNO2 +
   MOCAGE:LNO +     MOCAGE:SRMH2O + TEMPE:STATION +
   TEMPE:VentMOD + TEMPE:LNO2 +     TEMPE:LNO +
   TEMPE:SRMH2O + STATION:VentMOD + STATION:LNO2 +
   STATION:LNO + STATION:SRMH2O + VentMOD:LNO2 +
   VentMOD:LNO +     VentMOD:SRMH2O + VentANG:LNO2 +
   VentANG:LNO+LNO:SRMH2O,data=datappr),K=10)$delta[1]
```

2.3 Prévision de l'échantillon test

Retenir le meilleur modèle obtenu pour prédire l'échantillon test et estimer ainsi sans biais une erreur de prévision. Deux erreurs sont estimées ; la première est celle quadratique pour la régression tandis que la deuxième est issue de la matrice de confusion qui croise les dépassements de seuils prédits avec ceux effectivement observés.

```
# Calcul des prévisions
pred.glm=predict(reg.glm.step,newdata=datestr)
# Erreur quadratique moyenne de prévision
sum((pred.glm-datestr[,"O3obs"])^2)/nrow(datestr)
# Matrice de confusion pour la prévision du
# dépassement de seuil
table(pred.glm>150,datestr[,"O3obs"]>150)
```

Noter ces erreurs pour les comparer avec celles obtenues par les autres méthodes. Par exemple avec la prévision de MOCAGE.

```
# matrice de confusion du modèle MOCAGE
table(datestq[,"MOCAGE"]>150,datestq[,"DepSeuil"])
```

3 Prévision par modèle binomial

Plutôt que de prévoir la concentration puis le dépassement, on peut se poser la question de savoir s'il ne serait pas pertinent de prévoir directement la présence ou l'absence d'un dépassement. La variable à modéliser étant binaire, c'est la régression logistique qui va être employée. Comme pour la régression, différentes stratégies de choix de modèle peuvent être utilisées et comparées avant d'estimer l'erreur de prévision sur l'échantillon test.

3.1 Régression logistique sans interaction

```
# estimation du modèle complet
log.lm=glm(DepSeuil~.,data=datappq,family=binomial)
# significativité des paramètres
anova(log.lm,test="Chisq")# Recherche d'un modèle optimal au sens d'Akaïke
```

```
log.lm.step=step(log.lm,direction="backward")
anova(log.lm.step,test="Chisq")
# matrice de confusion de l'échantillon
# d'apprentissage et erreur apparente
table(log.lm.step$fitted.values>0.5,
   datappq[,"DepSeuil"])
```

3.2 Régression logistique avec interaction

```
# estimation du modèle complet
log.qm=glm(DepSeuil~(.)^2,data=datappq,
   family=binomial)
```

Avec autant de paramètres, l'algorithme de régression peut rencontrer quelques soucis si la régression ajuste "trop bien" les données. On propose alors une optimisation du modèle par une méthode pas-à-pas à comparer avec le modèle fournit par l'algorithme descendant.

```
# régression avec le modèle minimum
log.qm=glm(DepSeuil~1,data=datappq,family=binomial)
# algorithme stepwise en précisant le plus grand
# modèle possible
log.qm.step1=step(log.qm,direction="both",
  scope=list(lower=~1,upper=~(JOUR + MOCAGE +
  TEMPE + STATION + VentMOD + VentANG + LNO2 +
  LNO + SRMH2O)^2), family=binomial)
# significativité des paramètres
anova(log.qm.step1,test="Chisq")

# algorithme backward
log.qm=glm(DepSeuil~(.)^2,data=datappq,
   family=binomial)
log.qm.step2=step(log.qm,direction="backward",
   family=binomial)
# significativité des paramètres
anova(log.qm.step2,test="Chisq")
```

Deux modèles sont en concurrence, il s'agit de les comparer en minimisant l'erreur calculée par validation croisée. La même procédure est utilisée que pour le modèle linéaire classique mais une autre fonction de coût adaptée à une variable binaire doit être précisée. Elle est définie ci-dessous puis appliquée au calcul de l'erreur apparente de prévision.

```
cout = function(r, pi=0)
   mean(abs(as.integer(r)-pi)>0.5)
# Application de cette fonction pour
# l'erreur apparente ou d'ajustement :
cout(datappq$DepSeuil,log.qm.step1$fitted.values)
cout(datappq$DepSeuil,log.qm.step2$fitted.values)
```

Estimation de l'erreur de prévision par validation croisée.

```
library(boot)
cv.glm(datappq, log.qm.step1, cout,K=10)$delta[1]
cv.glm(datappq, log.qm.step2, cout,K=10)$delta[1]
```

Retenir le "meilleur" modèle.

3.3 Prévision de l'échantillon test

```
pred.log=predict(log.qm.step1,newdata=datestq)
# Matrice de confusion pour la prévision du
# dépassement de seuil
table(pred.log>0.5,datestq[,"DepSeuil"])
```

Mémoriser les résultats obtenus pour comparer avec les autres méthodes.

3.4 Courbe ROC

```
library(ROCR)
roclogit=predict(log.qm.step1,newdata=datestq,
   type="response")
predlogit=prediction(roclogit,datestq[,"DepSeuil"])
perflogit=performance(predlogit, "tpr","fpr")
plot(perflogit,col=1)
```

Il est également possible de construire une courbe ROC en association de la prévision obtenue à partir d'un modèle gaussien. En effet, la variation du seuil

théorique de dépassement (150) va faire varier les proportions respectives des taux de vrais et faux positifs. Cela revient encore à faire varier le seuil d'une "proba" pour les valeurs de prévisions divisées par 300.

```
rocglm=pred.glm/300
predglm=prediction(rocglm,datestq[,"DepSeuil"])
perfglm=performance(predglm, "tpr","fpr")
plot(perfglm,col=2,add=TRUE)
```

Les résultats obtenus dépendent évidemment en plus de l'échantillonnage initial entre appentissage et test. Dans le cas où les courbes se croisent, cela signifie qu'il n'y a pas de prévision uniformément meilleure de l'occurence de dépassement. Cela dépend de la sensibilité ou de la spécificité retenue pour le modèle. Ceci souligne l'importance de la bonne définition du critère à utiliser pour le choix d'une "meilleure" méthode. Ce choix dépend directement de celui , "politique" ou "économique" de sensibilité et / ou spécificité du modèle retenu. En d'autres termes, quel taux de fausse alerte, avec des imputations économiques évidentes, est supportable au regard des dépassements non détectés et donc de la dégradation sanitaire de la population à risque ?

C'est une fois ce choix arrêté que le statisticien peut opérer une comparaison des méthodes en présence.