TP ozone : SVM ou machine à vecteurs supports

Résumé

Prévision du pic d'ozone par machine à vecteurs supports ou séparateurs à vaste marge.

1 Introduction

Cette section propose d'aborder une nouvelle famille d'algorithmes : les SVM ou (*Support Vector Machines* traduit par Séparateurs à Vaste Marge ou machine à vecteurs support) dont le principe fondateur est d'intégrer l'optimisation de la complexité d'un modèle à son estimation ou plus exactement une partie de cette complexité ; cela concerne le nombre de vecteurs supports. Une bibliothèque de R, réalisée par Chang, Chih-Chung et Lin Chih-Jen, est destinée à cette approche ; elle est intégrée au package `e1071`.

Au delà du principe fondateur de recherche de parcimonie (nombre de supports) incorporé à l'estimation, il n'en reste pas moins que cette approche laisse, en pratique, un certain nombre de choix et réglages à l'utilisateur. Il est donc important d'en tester l'influence sur la qualité des résultats.

1. choix du paramètre de régularisation ou pondération d'ajustement,
2. choix du noyau,
3. le cas échéant, choix du paramètre associé au noyau : largeur d'un noyau gaussien, degré d'un noyau polynomial...

Notons la même remarque qu'avec les techniques précédentes sur l'intérêt à mettre en œuvre une approche "plan d'expérience" voire même "surface de réponse" afin d'optimiser le choix des paramètres.

2 Exemple élémentaire

Les données sont celles, triviales, des iris dits de Fisher afin d'illustrer la méthode en explicitant graphiquement l'emplacement des vecteurs supports. Trois variétés (stosa, versicolor, viginica) d'iris doivent être discriminées en fonction de 4 mesures (longueur et largeur des sépales et des pétales) réalisées sur les fleurs. Il y a 50 fleurs par variétés.

```
library(e1071)
# déclaration des données
data(iris)
# le modèle est calculé avec les valeurs
# par défaut des paramètres
# (noyau gaussien, pénalisation à 1, gamma=0,25)
     model = svm(Species ~ ., data = iris)
     print(model)
     summary(model)
# prévision de l'échantillon d'apprentissage
     pred = predict(model, iris[,1:4])
# Matrice de confusion pour l'échantillon
# d'apprentissage
     table(pred, iris$Species)
# Visualisation des classes (couleurs)
# et des vecteurs supports ("+")
     plot(cmdscale(dist(iris[,-5])),
     col = as.integer(iris[,5]),
     pch = c("o","+")[1:150 %in% model$index + 1])
```

Noter la densité des vecteurs supports dans les zones plus difficiles à discriminer. Ce sont les observations qui participent à la définition des marges ("+") correspndant donc aux contraintes "actives" du problème d'optimisation.

Le réglage des paramètres peut être optimisé automatiquement avec la fonction `tune` mais le graphique n'est pas toujours simple à interpréter.

```
obj = tune.svm(Species~., data = iris,
   gamma = 2^(-7:0), cost = 2^(-2:3))
summary(obj)
plot(obj)
```

3 Régression sur la concentration d'ozone

Malgré les assurances théoriques concernant ce type d'algorithme, les résultats dépendant fortement du choix des paramètres. Nous nous limiterons d'abord au noyau gaussien (choix par défaut) ; la fonction `tune.svm` permet de tester facilement plusieurs situations en estimant la qualité de prévision par validation croisée sur une grille. Le temps d'exécution est un peu long... en effet, contrairement à beaucoup d'algorithmes de modélisation, la complexité de l'algorithme de résolution des SVM croît très sensiblement avec le nombre d'observations mais moins avec le nombre de variables. Cette remarque est importante quant à l'adéquation à trouver entre données et méthode.

Bien qu'initialement développés dans le cas d'une variable binaire, les SVM ont été étendus aux problèmes de régression. L'estimation et l'optimisation du coefficient de pénalisation sont obtenues par les commandes suivantes.

```
svm.reg=svm(O3obs~.,data=datappr)
plot(tune.svm(O3obs~.,data=datappr,
   cost=c(1, 1.5,2,2.5,3,3.5)))
```

Par défaut la pénalisation (cost) vaut 1. Noter la pénalisation optimale pour le noyau considéré (gaussien). Ré-estimer le modèle supposé optimal avant de tracer le graphe des résidus. Comme précédemment, observer que plusieurs exécutions conduisent à des résultats différents et donc que l'optimisaiton de ce critère est pour le moins délicate.

```
svm.reg=svm(O3obs~.,data=datappr,cost=2.5)
# calcul et graphe des résidus
fit.svmr=predict(svm.reg,data=datappr)
res.svmr=fit.svmr-datappr[,"O3obs"]
plot.res(fit.svmr,res.svmr)
```

Observer l'effet "couloir" sur les résidus. Ceci est une conséquence de la fonction d'erreur robuste utilisée pour l'estimation des svm.

4 Discrimination

```
#optimisation
plot(tune.svm(DepSeuil~.,data=datappq,
  cost=c(1,1.25,1.5,1.75,2)))
# apprentissage
svm.dis=svm(DepSeuil~.,data=datappq,cost=1.25)
```

Calculer l'erreur apparente ou par re-substitution mesurant la qualité de l'ajustement.

```
#matrice de confusion
table(svm.dis$fitted,datappq$DepSeuil)
```

Remarque : il faudrait aussi optimiser la valeur du paramètre gamma du noyau gaussien ainsi que le choix du noyau...

5 Prévision de l'échantillon test

Avec les "meilleures" combinaisons de paramètres précédentes, estimer les erreurs sur l'échantillon test.

```
svm.reg=svm(O3obs~.,data=datappr,cost=2.5)
svm.dis=svm(DepSeuil~.,data=datappq,cost=1.25)
pred.svmr=predict(svm.reg,newdata=datestr)
pred.svmq=predict(svm.dis,newdata=datestq)
# Erreur quadratique moyenne de prévision
sum((pred.svmr-datestr[,"O3obs"])^2)/nrow(datestr)
# Matrice de confusion pour la prévision
# du dépassement de seuil (régression)
table(pred.svmr>150,datestr[,"O3obs"]>150)
# Même chose pour la discrimination
table(pred.svmq,datestq[,"DepSeuil"])
```

Noter, comparer les taux d'erreur.

6 Comparaison des courbes ROC

```
library(ROCR)
rocsvmr=pred.svmr/300
predsvmr=prediction(rocsvmr,datestq$DepSeuil)
perfsvmr=performance(predsvmr,"tpr","fpr")
# re-estimer le modèle pour obtenir des
```

```
# probabilités  de classe plutôt que des classes
svm.dis=svm(DepSeuil~.,data=datappq,cost=1.25,
  probability=TRUE)
pred.svmq=predict(svm.dis,newdata=datestq,
  probability=TRUE)
rocsvmq=attributes(pred.svmq)$probabilities[,2]
predsvmq=prediction(rocsvmq,datestq$DepSeuil)
perfsvmq=performance(predsvmq,"tpr","fpr")
# tracer les courbes ROC en les superposant
# pour mieux comparer
plot(perflogit,col=1)
plot(perfsvmr,col=2,add=TRUE)
plot(perfsvmq,col=3,add=TRUE)
```

Une méthode de prévision d'occurence de dépassement du pic de pollution est-elle globalement meilleure ?