TP R sur les SVM

Emmanuel Rachelson and Matthieu Vignes

22 février 2013, SupAero - ISAE

Pour les deux premiers exercices, des fichiers utiles sont disponibles là `http://carlit.toulouse.inra.fr/wikiz/images/e/e5/Exos1+2B2.rar`

1 SVM linéaires

Un premier exemple sur données artificielles

Le but ici est de mieux appréhender le principe de fonctionnement des SVM.

Il est rappelé que le SVM linéaire dans $\mathbb{R}^p$ s'écrit comme un problème d'optimisation: (1) $\min_{\omega\ in\mathbb{R}^p} C\sum_{i=1}^{n} L(y_i, \omega.x_i + b) + \parallel \omega \parallel^2$ avec une solution de la forme $\omega = \sum_{i=1}^{n} \alpha_i x_i$, où $\alpha_i \neq 0$ si et seulement si x_i est un vecteur du support.

1. Rappeler brièvement les objectifs et le principe des SVM. Vous pourrez vous appuyer sur des considérations géométriques.
2. Charger les données `datalin` (ou les générer avec le code fourni) et la librarie `kernlab`.
3. Entraîner et visualiser le SVM linéaire pour plusieurs valeurs de la constante C. Aide R:

```
svp<-ksvm(xtrain,ytrain,type="C-svc",kernel='vanilladot',C=??,scaled=c())
plotsvm(svp,xtrain)
```

4. Interpréter les deux termes de la fonctionnelle qui définit (1).
5. Expliquer et vérifier expérimentalement l'effet de C sur la position de la séparatrive et sur le nombre de vecteurs support. Que sont les lignes pointillées ?
6. Que se passerait-il si les données étaient "plus mélangées", comme par exemple générées par 2 distributions gaussiennes de centre plus proche et/ou de variance plus grande ?

7. Visualiser les points du jeu de données test avec le classifieur et calculer les prédictions sur les points test (aide R: `predict(svp,xtest)`). Comment se calcule la prédiction sur un point inconnu x étant donné ω^*, solution de (1) ? Et étant donnée la solution en α ? Certains algorithmes optimiseront en ω, d'autres en α...

Application aux données SPAM

1. Charger les données SPAM `dataC` avec leurs labels.
2. Entraîner un SVM avec plusieurs valeurs de C.
3. Tracer l'erreur de classification sur les données d'entraînement et de test en fonction de C.
4. Expliquer le comportement des deux erreurs précédemment tracées.
5. Si on a un problème de classification pour lequel on utilise certains points en entraînement. Comment utiliser ces points pour estimer une constante "optimale" (minimisant l'erreur sur des données inconnues) ? Puis comment utiliser ces points pour estimer l'erreur qui sera faite par le classifieur sur des données inconnues ?

2 SVM à noyaux

Le but de cet exercice sera de mieux appréhender le concept de noyau et son utilisation dans un algorithme d'apprentissage.

On rappelle qu'un noyau à base radiale gaussien est défini pour tout $(x, y) \in (\mathbb{R}^p)^2$ par : $K(x, y) = \exp\left(-\frac{\|x-y\|}{2\sigma^2}\right)$.

De manière générale, l'estimateur donné par le SVM à noyaux a la forme: $f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) + b$.

1. Rappeler brièvement la définition d'un noyau défini positif. Dans quel cas peut-on s'en servir pour implicitement plonger les données dans un espace de descripteurs ? Comment ? Quel en est l'intérêt ?
2. Charger et regarder les données `datamix` (ou les générer avec le code fourni). Penser à charger la librarie `kernlab` si ce n'est déjà fait.
3. Entraîner et visualiser un SVM linéaire sur les données. Des observations ? Aide R:

   ```
   svp <- ksvm(xtrain,ytrain,type="C-svc",kernel='rbf',kpar=list(sigma=1),C=1)
   ```

4. Utiliser maintenant un noyau à base radiale gaussien. Qu'observe-t-on ?
5. Expliquer comment est construite la séparatrice en termes de noyau et (de façon équivalente) d'espace des descripteurs.

6. Décrire l'effet de C et σ sur la séparatrice. Sur les vecteurs du support.

3 Application à un problème réel: retour sur les données OZONE

On va ici travailler sur des données "ozone" (différentes de celles vues dans le TP sur les tests). On va chercher à prédire un pic d'ozone en fonction de conditions atmosphériques/météorologiques par SVM.

Si on veut changer, on pourra utiliser la librairie `e1071`.

On rappelle que le principe fondateur repose sur une recherche *parcimonieuse* du nombre de support dans l'estimation. Les degrés de liberté qui vont sont laissés peuvent influencer les résultats, il faut donc regarder leur impact sur la qualité des prévisions: (i) choix du paramètre de régularisation ou pondération d'ajustement, (ii) choix du noyau et (ii-bis) choix d'un ou plusieurs paramètre(s) associé(s) au noyau: largeur du noyau gaussien, degré d'un noyau polynomial ... On est presque dans la planification expérimentale !

Exemple élémentaire guidé

Les données de ce paragraphe (uniquement) sont les données `iris` pour illustrer ce qui se passe. Trois variétés (*stosa, versicolor, virginica*) doivent être *discriminées* en fonction de quatre mesures: longueur, largeur des sépales et des pétales. Il y a 50 fleurs par variété.

```
library(e1071)
# declaration des donnees
data(iris)
# le modele est calcule avec les valeurs par defaut des parametres
# (noyau gaussien, penalisation a 1, gamma=0.25)
model = svm(Species ~ ., data = iris)
print(model)
summary(model)
# prevision de l'echantillon d'apprentissage
pred = predict(model, iris[,1:4])
# matrice de confusion pour l'echantillon 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 supports dans les zones frontières (plus difficile à discriminer donc). Ces observations participent fortement à la définition des marges et sont les contraintes "actives" du problème d'opti.

On pourrait être tenté de régler les paramètres automatiqueement avec la fonction `tune()` mais l'interpr{etation n'en est pas évidente:

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

Régression sur la concentration d'ozone

On l'a déjà vu, les résultats dépendent (fortement) du choix des paramètres. On se limitera en premier lieu au noyau gaussien. La fonction `tune.svm()` permet de tester plusiuers situations en estimant la qualité de prévision par validation croisée sur une grille. Le temps d'éxécution peut alors être long: l'algorithme de résolution des SVM croît assez sensiblement avec la taille des observations. Peu avec le nombre de variables. C'est une remarque à prendre en compte dans l'adéquation d'une méthode pour analyser des données.

Les données ont été extraites et mises en forme par le service concerné de MétéoFrance; elles contiennent les variables suivantes :

- `JOUR`: le type de jour ; férié (1) ou pas (0),
- `O3obs`: la concentration d'ozone effectivement observée le lendemain Ã 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-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 ou Plan de Cuques,
- `VentMOD`: force du vent et
- `VentANG`: orientation du vent

1. Charger les données `ozone.dat` (dispos là `http://carlit.toulouse.inra.fr/wikiz/images/0/02/Ozone.dat.rar`). Passer la variable `JOUR` en facteur. Faire une transformation carrée pour la variable `RMH2O` et une transformation log pour les variables `NO2` et `NO` (pourquoi ?). Retirer les variables initiales.

2. Séparer les échantillons en un échantillon d'apprentissage `datappr` (sur lequel le modèle sera entraîné) et un échantillon de test `datestr` (sur lequel on mesurera les écarts entre prévisions et valeurs réllement observées). Les tailles respectives de ces deux jeux seront de 80 et 20%.

3. Les SVM, développée initialement pour traiter le cas d'une variable binaire (discrimination) ont été étendues aux problèmes de régression. On peut estimer et optimiser le coefficient de pénalisation *via*:

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

4. Par défaut, la pénalisation (`cost`) vaut 1. Noter la pénalisation optimale pour le noyau considéré (gaussien). Réestimer le modèle "optimal" avant de tracer le graphe des résidus. Observer qye plusieurs éxécutions ne condisent pas forcément au même résultat. Est-il évident d'"optimiser" ce paramètre ?

5. Observer un effet "couloir" sur les résidus. C'est une conséquence de la fonction d'erreur robuste utilisée pour l'estimation des SVM.

Discrimination

On va se coup-ci étudier une variable binaire (présence d'un pic d'ozone ou non). On est dans le cadre de la *discrimination*, plus celui de la régression. On créra à partir de la variable `O3obs` une variable seuil binaire selon que le taux dépasse $150\,\mu g.m^{-3}$ ou non et on la nommera `DepSeuil`. On refera des échantillons d'apprentissage `datappq` et test `datestq` (attention à ne pas inclure la variable `O3obs` ici !) avec les mêmes individus que dans le cas de la régression.

```
# 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 directement. Et pour mesurer la qualité de l'ajustement ?

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

En toute exhaustivité, il aurait fallu omptimiser la valeur du paramètre γ du noyau gaussien ici aussi. Voire choisir un noyau différent...

Prévision de l'échantillon test

Avec les "meilleurs" combinaisons des paramètres obtenus dans les parties précédentes, estimer les erreurs (quadratique ou taux de mauvais classement par exemple selon le cas: régression ou discrimination) sur l'échantillon test.

Noter, comparer et commenter ces taux d'erreur.

Comparaison des courbes ROC (optionnel)

Avec la librairie `ROCR`, comparer les courbes ROC obtenues par un modèle logistique avec interaction (détaillé ci-dessous) et

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

Puis faire quelquechose de similaire avec (1) les prédictions continues du modèle de régression et (2) les prédictions du modèle de discrimination.

Remarquer que 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 "probabilité" pour les valeurs de prévisions divisées par 300:

```
rocsvmr=pred.svmr/300
predsvmr=prediction(rocsvmr,datestq$DepSeuil)
perfsvmr=performance(predsvmr,"tpr","fpr")
```

Superposer les courbes à la précédente:

```
plot(perfsvmr,col=2,add=TRUE)
plot(perfsvmq,col=3,add=TRUE)
```

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