École Centrale Paris
Statistique Avancée

Dalalyan Arnak
Année 2010-2011

Estimation de densité - TP

1 Introduction

Le but de cette séance est d'appliquer les méthodes non-paramétriques vues en cours aux données simulées mais également aux données réelles. Pour cela, on utilisera le logiciel R. Toutes les fonctions que nous allons utiliser ont été préalablement programmées. Pour s'en servir,

- téléchargez le fichier nommé `TP-Stat2010.R`
  `http://certis.enpc.fr/~dalalyan/Links/TP-Stat2010.R`
- Sauvegardez-le dans votre répertoire personnel.
- Exécutez-le en choisissant dans le menu `Fichier > Sourcer du code R` le fichier que vous venez de télécharger.

Il est possible de travailler directement dans la fenêtre principale de `R`, nommée `R Console`, mais cela n'est pas très pratique. Il vaut mieux ouvrir une fenêtre d'editeur en allant dans le menu `Fichier > Nouveau script`. Une fois la fenêtre d'editeur ouverte, vous pouvez sauvegarder le fichier qui est encore vide mais sera bientôt rempli en utilisant le menu `Fichier > sauver sous`. Choisissez votre répertoire personnel et sauvegardez le fichier sous le nom `TP_stat_NP.R`.

Par la suite, vous pouvez copier-coller les commandes à partir de ce document dans la fenêtre d'editeur. Pour les exécuter, il suffira de les selectionner (à l'aide des touches `Ctrl A` ou à l'aide de la souris) et de les lancer à l'aide des touches `Ctrl R`.

On commencera par étudier le comportement empirique des estimateurs de densité vus en cours (histogramme et estimateur à noyau) sur des données simulées selon des lois de probabilité usuelles : loi uniforme, loi gaussienne et loi de Student. Ensuite, on appliquera ces méthodes aux données réelles de la vitesse de galaxies.

2 Estimation par histogramme à nombre de classe variable

Pour visualiser les histogrammes basés sur un même échantillon mais utilisant un nombre de classes variable, nous allons :

1. donner des valeurs aux paramètres `a, b, mu, sigma` et `d`. Par exemple
   ```
   a = 0
   b = 5
   mu = 2
   sigma = 4
   d = 6
   ```
2. générer un vecteur aléatoire de taille $n = 4000$ de coordonnées i.i.d. :
   ```
   x = runif(n,a,b)        # uniformes sur [a,b]
   x = rnorm(n,mu,sigma)   # gaussiennes N(mu,sigma^2)
   x = rt(n,d)             # loi de Student à d degrés de liberté
   ```

3. une fois que le vecteur `x` est généré, tracer l'histogramme à m (n'ouliez pas de donner une valeur à m !) classes de l'échantillon formé par les coordonnées de x en utilisant la commande :

   ```
   histogram(x,m,add=FALSE)
   ```

   Exercice 1. En utilisant la fonction `histogram(x,m=100,add=FALSE)`, deviner la densité de l'échantillon généré par la fonction `x=rexp(1000,t)` lorsque $t = 1$. Que se passe-t-il lorsqu'on fait varier le paramètre t ?

4. afin de pouvoir visualiser dans une même fenêtre plusieurs histogrammes correspondant à des valeurs différentes de m (nombre de classes), on peut utiliser les commandes

   ```
   op = par(mfcol=c(2,2),pty="m",omi=c(0,0,0,0))
   histogram(x,5,add=F,main="histogramme à 5 classes")
   histogram(x,20,add=F,main="histogramme à 20 classes")
   histogram(x,80,add=F,main="histogramme à 80 classes")
   histogram(x,200,add=F,main="histogramme à 200 classes")
   par(op)
   ```

5. pour que la comparaison visuelle soit plus facile, on peut forcer R à utiliser la même échelle pour les ordonnées des 4 graphiques :

   ```
   n=1000
   x=rnorm(n,0,1)
   h_top=0.7;
   op = par(mfcol=c(2,2),pty="m",omi=c(0,0,0,0))
   histogram(x,5,add=F,main="histogramme à 5 classes",ylim=c(0,h_top))
   histogram(x,25,add=F,main="histogramme à 25 classes",ylim=c(0,h_top))
   histogram(x,200,add=F,main="histogramme à 200 classes",ylim=c(0,h_top))
   histogram(x,600,add=F,main="histogramme à 600 classes",ylim=c(0,h_top))
   par(op)
   ```

   Question : Lequel des quatres histogrammes obtenus
   (a) approche le mieux la densité de la loi normale ?
   (b) a une erreur d'approximation trop grande ? (oversmoothing)
   (c) a une erreur stochastique trop grande ? (undersmoothing)
   Pour mieux répondre à ces questions, on pourra superposer la courbe de la loi gaussienne aux histogrammes affichés avec les commandes ci-dessus. Cela peut se faire, par exemple, de la manière suivante :

   ```
   op = par(mfcol=c(2,2),pty="m",omi=c(0,0,0,0),lwd=2)
   histogram(x,5,add=F,main="histogramme à 5 classes",ylim=c(0,h_top))
   curve(dnorm(x),add=T,col="red")
   histogram(x,30,add=F,main="histogramme à 50 classes",ylim=c(0,h_top))
   curve(dnorm(x),add=T,col="red")
   histogram(x,200,add=F,main="histogramme à 200 classes",ylim=c(0,h_top))
   curve(dnorm(x),add=T,col="red")
   histogram(x,600,add=F,main="histogramme à 600 classes",ylim=c(0,h_top))
   curve(dnorm(x),add=T,col="red")
   par(op)
   ```

3 Choix du nombre de classes par validation croisée

Les expériences réalisées montrent clairement que le choix du nombre de classes m influence considérablement la qualité de l'estimation.
La fonction `CV_hist` fournit la valeur du nombre des classes qui minimise le critère de validation croisée $\widehat{J}(m)$ (cf. le cours). Pour utiliser cette fonction, il suffit de lui donner comme argument un vecteur de nombres réels (l'échantillon) :

```
n = 1000
x = rnorm(n,0,1)
mCV = CV_hist(x)
mCV
```

Il faut attendre quelques secondes pour avoir le résultat. On remarquera qu'outre le point de minimum de $\widehat{J}(m)$, la fonction `CV_hist` fournit les graphes de la fonction $m \mapsto \widehat{J}(m)$ et de l'histogramme basé sur $\widehat{m} = \arg\min_m \widehat{J}(m)$.
Si l'on veut avoir le graphique de l'histogramme basé sur le h choisie par la validation croisée sans avoir la courbe de la fonction J, il suffit de taper :

```
histogram(x,mCV,add=F)
curve(dnorm,type="l",col="darkred",add=TRUE,lwd=2)
```

Exercice 2. Quel résultat obtient-on lorsque la méthode de validation croisée est appliquée à un échantillon de taille $n = 400$ [1] de loi uniforme sur $[0, 1]$? Expliquer intuitivement le résultat obtenu.

4 Estimateur à noyau

On étudie maintenant l'estimateur à noyau basé sur l'échantillon `x`$= (X_1, \ldots, X_n)$. La fonction `KernelEst` calcule l'estimateur $\hat{f}_h(x)$ pour 500 valeurs de x. Pour tester cette fonction, exécutez les commandes suivantes :

```
x=rnorm(1000)
ff=KernelEst(x,0.6,"Tri",ylim=c(0,0.6))
curve(dnorm,col="blue",lwd=2,add=T)
legend(-10,0.55,c("kernel estimator","true density"),
   col=c("darkred","blue"),bg="cyan",lty=1,lwd=2)
```

La syntaxe générale de cette fonction est `f=KernelEst(x,h,Noyau,...)` où
– `x` est l'échantillon,
– `h` est la fenêtre,
– `Noyau` est une chaîne de caractère qui (pour le moment) doit être choisie dans la liste
  ▷ `'Rect'` pour le noyau rectangulaire,
  ▷ `'Tri'` pour le noyau triangulaire,
  ▷ `'EP'` pour le noyau d'Epanechnikov,
  ▷ `'Gaus'` pour le noyau gaussien,
  ▷ `'sinc'` pour le noyau sinc.

1. Ce nombre peut être revu à la baisse si les calculs prennent beaucoup de temps.

– les trois points indique qu'on peut passer des paramètres graphiques à la fonction `KernelEst` ; ces paramètres seront utilisés pour tracer la courbe de l'estimateur à noyau. Exemples de tels arguments sont `main` pour le titre du graphe, `xlab` pour le text décrivant les abscisses, `ylim` pour les limites des ordonnées, ...

Exercice 3. Tester cette fonction avec l'option `'sinc'`. Pouvez-vous proposer une façon simple d'améliorer l'estimateur à noyau basé sur le noyau sinc ?

4.1 Impact du choix de noyau

On cherche à se convaincre que le choix du noyau n'a pas d'impact très significatif sur la qualité d'estimation, dans le sens où si la fenêtre est bien choisie, les différents noyaux précités produisent des estimateurs de qualités comparables.
Exécuter les commandes

```
n=400
h=0.8
x=rnorm(n,0,1)
op = par(mfcol=c(2,2),pty="m",omi=c(0,0,0,0))
f=KernelEst(x,h,"Rect",ylim=c(-0.05,0.4),main="Rectangulaire");
curve(dnorm,col="blue",lwd=2,add=T)
legend(3,0.4,c("kernel estimator","true density"),
   col=c("darkred","blue"),bg="cyan",lty=1,lwd=2)
f=KernelEst(x,h,"Tri",ylim=c(-0.05,0.4),main="Triangulaire");
curve(dnorm,col="blue",lwd=2,add=T)
legend(3,0.4,c("kernel estimator","true density"),
   col=c("darkred","blue"),bg="cyan",lty=1,lwd=2)
f=KernelEst(x,h,"EP",ylim=c(-0.05,0.4),main="Epanechnikov");
curve(dnorm,col="blue",lwd=2,add=T)
legend(3,0.4,c("kernel estimator","true density"),
   col=c("darkred","blue"),bg="cyan",lty=1,lwd=2)
f=KernelEst(x,h,"Gaus",ylim=c(-0.05,0.4),main="Gaussien");
curve(dnorm,col="blue",lwd=2,add=T)
legend(3,0.4,c("kernel estimator","true density"),
   col=c("darkred","blue"),bg="cyan",lty=1,lwd=2)
par(op)
```

Commenter le résultat obtenu.

4.2 Impact du choix de la fenêtre

Vérifions maintenant que si l'on choisit la fenêtre indépendamment de n et de l'échantillon, alors le résultat peut être catastrophique.

▷ Pour se convaincre que la fenêtre doit dépendre de l'échantillon, considérons le même exemple que dans la partie 1.3.1 en remplaçant la loi gaussienne centrée réduite par la loi $\mathcal{N}(0, 100)$:

```
n=400
x=rnorm(n,0,10)
h=0.8
```

```
par(bg="cornsilk")
op = par(mfcol=c(2,2),pty="m",omi=c(0,0,0,0))
f=KernelEst(x,h,"Rect",main="Rectangulaire",ylim=c(0,0.05))
curve(dnorm(x,0,10),col="blue",lwd=2,add=T)
f=KernelEst(x,h,"Tri",main="Triangulaire",ylim=c(0,0.05))
curve(dnorm(x,0,10),col="blue",lwd=2,add=T)
f=KernelEst(x,h,"EP",main="Epanechnikov",ylim=c(0,0.05))
curve(dnorm(x,0,10),col="blue",lwd=2,add=T)
f=KernelEst(x,h,"Gaus",main="Gaussien",ylim=c(0,0.05))
curve(dnorm(x,0,10),col="blue",lwd=2,add=T)
par(op)
```

Question : Doit-on augmenter h pour améliorer la qualité de l'estimation ou le diminuer ? Trouver une fenêtre (de façon expérimentale) qui conduit vers des résultats similaires à ceux obtenus dans le cas d'une loi $\mathcal{N}(0,1)$. Expliquer ce résultat.

▷ Pour se convaincre que la fenêtre doit dépendre également de la taille de l'échantillon, exécuter (plusieurs fois) les commandes

```
n=20
x=rnorm(n,0,1)
h=0.8
par(bg="cornsilk")
op = par(mfcol=c(2,2),pty="m",omi=c(0,0,0,0))
f=KernelEst(x,h,"Rect",main="Rectangulaire",ylim=c(0,0.5))
curve(dnorm(x,0,1),col="blue",lwd=2,add=T)
f=KernelEst(x,h,"Tri",main="Triangulaire",ylim=c(0,0.5))
curve(dnorm(x,0,1),col="blue",lwd=2,add=T)
f=KernelEst(x,h,"EP",main="Epanechnikov",ylim=c(0,0.5))
curve(dnorm(x,0,1),col="blue",lwd=2,add=T)
f=KernelEst(x,h,"Gaus",main="gaussien",ylim=c(0,0.5))
curve(dnorm(x,0,1),col="blue",lwd=2,add=T)
par(op)
```

5 Choix de la fenêtre par validation croisée

A partir de maintenant, on travaillera toujours avec le noyau gaussien. L'avantage principal de travailler avec le noyau gaussien consiste dans le fait que le critère de validation croisée est explicitement calculable dans ce cas. On cherche à choisir la fenêtre h de façon presque "optimale".
La fonction `CV_kern` a une syntaxe très similaire à celle de la fonction `CV_hist` vue ci-dessus.

Exercice 4. Pour tester la fonction `CV_kern`, déterminer la fenêtre optimale pour estimer la densité d'un échantillon de taille 400 de loi $\mathcal{N}(0,1)$.

Exercice 5. Est-ce que la validation croisée est invariante par changement d'échelle ? C'est-à-dire, si $X_1, \ldots, X_n$ est un échantillon pour lequel $\widehat{h}$ est la fenêtre minimisant le critère de validation croisée, est-ce que $a\widehat{h}$ est le minimiseur du critère de validation croisée construit à partir de l'échantillon $aX_1, \ldots, aX_n$?
Faites des expériences pour trouver la réponse à cette question.

Exercice 6. Vérifier (expérimentalement) que la validation croisée est également invariante par translation. En déduire qu'elle est invariante par transformations affines.

6 Données de galaxies

Les données qu'on cherche à étudier maintenant contient les vitesses de mouvement (en km/seconde) de 82 galaxies.

▷ Pour charger ces données, tapez les commandes

```
library(MASS)
```

▷ Pour lire la description de ces données et tracer le boxplot :

```
help(galaxies)
boxplot(galaxies)
```

▷ Trouver $\widehat{m}$ et $\widehat{h}$, le nombre de classes et la fenêtre qui minimisent le critère de validation croisée.

▷ Tracer côte à côte l'histogramme et l'estimateur à noyau optimaux.

```
op=par(mfcol=c(1,2),pty="m",omi=c(0,0,0,0))
histogram(galaxies,m,F,xlim=c(5000,40000))
f=KernelEst(galaxies,h,"Gaus",xlim=c(5000,40000))
par(op)
```