

TP2 - Statistique Non Paramétrique

Estimation non paramétrique d'une densité

Le rapport est à rendre avant le mercredi prochain (13 Avril).

Exercice 1 Estimateur histogramme

- (a) Construire l'estimateur histogramme usuel pour une densité supportée sur $[0,2]$, en complétant le code ci-dessous.

```
> # construire l'estimateur histogramme comme une fonction de N, de l'échantillon et de x
> # où N est le nombre de classes / intervalles / bins
> # et x est un point dans [0,2]
> estimateur_hist <- fonction(x, N, ech){
+   n <- length(ech)
+   # découper [0, 2] en N classes de même taille, la fenêtre h ainsi = 1/N
+   breakpoints <- seq(0, 2, length.out = N + 1)
+   h = 2/N # la fenêtre
+   # calculer la valeur de l'estimateur histogramme en point x: <<f_chap_x>>
+   for (i in 1:N){
+     # réperer la bonne classe <<i>> ou se trouve x
+     if(x >= breakpoints[i] & x < breakpoints[i+1]){
+       # calculer la fréquence du nb de obs qui tombent dans la classe <<i>>
+
+
+
+       f_chap_x = # à compléter
+
+
+
+       break
+     }
+   }
+   return(f_chap_x)
+ }
```

- (b) Utiliser le code ci-dessous pour simuler un échantillon i.i.d de taille $n = 200$ de la loi de mélange

des deux gaussiens tronquées sur le même intervalle compact $[0, 2]$:

$$\frac{1}{2}\mathcal{N}(0.5, 0.04, 0, 2) + \frac{0.5}{2}\mathcal{N}(1.5, 0.04, 0, 2),$$

et appliquer l'estimateur crée avec $N = 10$ puis 200. On donne également le code pour tracer les histogrammes (densités estimées).

https://en.wikipedia.org/wiki/Truncated_normal_distribution

https://fr.wikipedia.org/wiki/Loi_de_mélange

```
> #install.packages(truncnorm)
> library(truncnorm)
> set.seed(1)
> ech_mgt <- c(rtruncnorm(100,a = 0,b = 2,mean=0.5,sd = 0.2),
+             rtruncnorm(100,a = 0,b = 2,mean=1.5,sd = 0.2))
> x_list <- seq(0,1.999,length.out = 2000)
> estimation_hist <- sapply(x_list, function(x){
+                               return(estimateur_hist(x,20,ech_mgt))
+                               })
+                               )
> plot(x_list,estimation_hist, type="s")
```

Une grille à 200 classes est trop fine pour obtenir une bonne estimation.

- (c) Retrouver ces résultats avec la fonction de R `hist()`.

```
> hist(ech_mgt, breaks =, freq =)
```

Exercice 2 Estimateur à noyau

- (a) Construire un estimateur à noyau avec le noyau Gaussien en complétant le code ci-dessous.

```
> #construire l'estimateur à noyau Gaussien comme une fonction de h, de l'échantillon et de n
> #NB: la fenêtre h n'est pas le même hyper-paramètre qu'avant, qui au lieu contrôle les
> estimateur_noyau_Gaussien <- fonction(x, h, ech){
+   n <- length(ech)
+   # calculer la valeur de l'estimateur à noyau en point x: <<f_chap_x>>
+   int_K <- (ech - x)/h
+
+
+   f_chap_x <- # à compléter
+
+
+   return(f_chap_x)
+ }
```

- (b) Appliquer l'estimateur à noyau Gaussien à l'échantillon de l'exercice, avec $h = 0.02, 0.05, 0.9, 0.2, 0.5$, et tracer les densités estimées (superposer l'histogramme à $N = 20$). Que pensez vous de l'impact de la valeur de h sur l'estimateur? Quelle est selon vous la bonne valeur de h ?

(c) Comparer avec les densités estimées par la fonction de R `density()`.

```
> install.packages("kdensity")
> estimation_density <- density(ech_mgt, kernel = , bw = , n = 2000, from = 0, to = 2)
> plot(estimation_density,col="black")
```

Exercice 3 Sélection de la fenêtre h : la validation croisée (Leave One Out)

On va maintenant chercher une approche qui permet de choisir le hyperparamètre h en autonomie, et puis la programmer.

Le théorie dans le Section 4.3.6 nous implique une telle approche, qui propose le minimiseur du critère EQMI empirique, $\hat{J}_n(h; X_1, \dots, X_n)$, comme la valeur optimale de h . On reprend les notations du cours (Voir Chap. 3 page 11). Notez qu'étant donnée un échantillon X_1, \dots, X_n , les estimateurs U_n et V_n sont des fonctions de h , qu'il faudrait donc rigoureusement écrire $U_n(h; X_1, \dots, X_n)$ et $V_n(h; X_1, \dots, X_n)$.

Dans la suite, on va créer la fonction $\hat{J}_n(h; X_1, \dots, X_n)$ en 2 étapes. La question (a), (b) vous guide respectivement de créer $U_n(h; X_1, \dots, X_n)$ et $V_n(h; X_1, \dots, X_n)$. Finalement, dans la question c), on minimise la $\hat{J}_n(h; X_1, \dots, X_n)$ créé pour l'échantillon précédent.

(a) Créer une fonction de R qui prend la fenêtre h et l'échantillon X_1, \dots, X_n comme les rentrées et sort $U_n(h; X_1, \dots, X_n)$, en complétant le code ci-dessous.

NB : On pourra utiliser les formules suivantes :

$$\hat{f}_{h,-i}(X_i) = \frac{n}{n-1} \left[\hat{f}_h(X_i) - \frac{K(0)}{nh} \right] \quad (0.1)$$

$$\hat{U}_n(h; X_1, \dots, X_n) = \frac{1}{n-1} \sum_{i=1}^n \hat{f}_h(X_i) - \frac{K(0)}{(n-1)h} \quad (0.2)$$

```
> U_n <- function(h, ech){
+   n <- length(ech)
+   # 1ère terme
+
+   sum_ <- # à compléter
+
+   sum_ <- sum_/(n-1)
+   # 2ème terme
+
+   KO <- # à compléter
+
+   KO <- KO/(n-1)/h
+   return(sum_ + KO)
+ }
```

(b) Le code ci-dessous crée une fonction permet de calculer $V_n(h; X_1, \dots, X_n)$.

NB : En pratique, il faut utiliser la methode numerique pour approcher le calcul comme l'intégral. Dans R, la fonction qui calcule l'intégral d'une fonction univariable est `integrate`.

```

> V_n <- function(h, ech){
+   n <- length(ech)
+   # Fonction wrapper:
+   # convertir la rentrée et la sortie de la estimateur_noyau_Gassien(x_,h,ech) en vecteur
+   # fixer les 2eme 3eme arguments
+   fsq <- function(x){
+     return(sapply(x, function(x_) return(estimateur_noyau_Gassien(x_,h,ech)^2)))
+   }
+   integral_fsq <- integrate(fsq, lower=0, upper =2)$value
+   return(integral_fsq)
+ }

```

- (c) Construire $\hat{J}_n(h; X_1, \dots, X_n)$, et tracer la courbe en fonction de h, avec l'échantillon donnée celui de les exercices précédants, pour repérer son minimum.

```

> J_n <- function(h, ech){
+   return(## à compléter
+     )
+ }
> h_list <- seq(0.01, 2, 0.01)
> val_J_n <- ## à compléter
+ plot(h_list, val_J_n, type = "s")
>
> # h optimal
>
> ## à compléter
>

```