R AVANCÉ: 2021-2022
masedki.github.io

TP2: Optimisation de code sous R

En statistique, l'estimation de la densité d'une variable aléatoire peut se faire par une méthode à noyaux. Soit $(X_1, X_2, \ldots, X_n)$ un n échantillon iid issu d'une distribution ayant pour densité f. On souhaite obtenir un estimateur de f à partir de l'échantillon. Pour cela, on utilise l'estimateur de Parzen-Rosenblatt défini par

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K((x - X_i)/h),$$

où K est un noyau (*i.e.,* une fonction non-négative qui s'intègre à 1) et $h > 0$ est un paramètre de lissage appelé fenêtre. Dans cet exercice, on s'intéresse au cas d'un noyau gaussien, on a donc

$$K(u) = \phi(u) = \frac{1}{\sqrt{2\pi}} \exp^{-\frac{1}{2}u^2}.$$

La fenêtre du noyau est un paramètre qui influence fortement les résultats. Son choix est donc important. Une façon de choisir la fenêtre est de chercher à minimiser l'erreur quadratique moyenne définie par

$$R(\hat{f}_h, f) = \mathbb{E}_f[L(\hat{f}_h, f)] \text{ with } L(\hat{f}_h, f) = \int (\hat{f}_h(x) - f(x))^2 dx.$$

Cependant, la fonction de risque R est-elle même inconnue. Ainsi, on peut utiliser une méthode de validation croisée *leave-one-out* pour estimer cette fonction de risque. On peut vérifier que

$$L(\hat{f}_h, f) = J(\hat{f}_h, f) + \int f^2(x)dx \text{ avec } J(\hat{f}_h, f) = \int \hat{f}_h^2(x)dx - 2\int \hat{f}_h(x)f(x)dx.$$

Dans le cas du noyau Gaussien, on a

$$\int \hat{f}_n^2(x)dx = c + \frac{1}{n^2 h\sqrt{2}} \sum_{i=1}^{n}\sum_{j=1}^{n} \phi\left(\frac{X_i - X_j}{h\sqrt{2}}\right)$$

où c est une constante qui ne dépend ni de h ni de l'échantillon. L'estimateur de la fonction de risque obtenu par validation croisée est défini (à une constante près) par

$$CV(h) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}_{h\sqrt{2}}(X_i) - \frac{2}{n} \sum_{i=1}^{n} \hat{f}_{h,(-i)}(X_i),$$

où $\hat{f}_{h,(-i)}$ est l'estimateur de la densité obtenu en ôtant l'observation i.

Ainsi, la fenêtre sélectionnée $\hat{h}$ est

$$\hat{h} = \arg\min_{h>0} CV(h).$$

Le but de ce TP est de coder de manière efficace une procédure de sélection de fenêtre pour une estimation non-paramétrique de la densité. Pour cela, on va créer un projet R dans R studio, en suivant le chemin suivant:

`File` > `NewProject` > `Empty Project`

1. On cherche à programmer de façon efficace une façon d'évaluer $\hat{f}_h(x)$ pour un ensemble de points connus lorsque $\hat{f}_h(x)$ est estimé sur échantillon. On définiera differentes versions de la fonction d'estimation de densité. Toutes ces fonctions prendront 3 arguments: les points où la fonction sera évaluée `xpts_num` , l'échantillon permettant de calculer l'estimateur `x_num` et une taille de fenetre `h_num` qui par défaut vaudra $h = n^{-1/5}$, n étant la taille de l'échantillon. Chaque fonction retournera un vecteur contenant la densité estimée pour tout point dans `xpts_num`. Il est important de séparer les scripts où vous définissez vos fonctions des scripts où vous effectuez des tests ou du profilage.
   (a) Implémenter la fonction `ksmooth1` en utilisant une double boucle (une sur les valeurs de `x_num` et une sur les valeurs de `xpts_num`).
   (b) Implémenter la fonction `ksmooth2` qui remplace l'appel à la double boucle précédente par l'appel à la fonction `outer` (cette fonction est utilisée pour calculer la différence entre chaque point de `x_num` et chaque point de `xpts_num`).
   (c) Comparer les deux méthodes lorsque l'échantillon est composé de 10^4 réalisations de $\mathcal{N}(0,1)$ et que les points d'évaluation de l'estimateur sont 17 points uniformément répartis entre -4 et 4.
   (d) Appeler la fonction `ksmooth2` pour un échantillon composé de 10^7 réalisations de $\mathcal{N}(0,1)$ et avec des points d'évaluation qui sont 10^7 points uniformément répartis entre -4 et 4.
   (e) Implémenter la fonction `ksmooth3` qui utilise une seule boucle et un calcul vectoriel.
   (f) Implémenter la fonction `ksmooth4` qui utilise un appel à la fonction `sapply` et un calcul vectoriel.
   (g) Comparer les quatre méthodes lorsque l'échantillon est composé de 10^4 réalisations de $\mathcal{N}(0,1)$ et que les points d'évaluation de la fonction sont 17 points uniformément répartis entre -4 et 4.
2. On cherche à illustrer l'effet de la fenêtre h sur l'estimateur de la densité. Générer 100 observations suivants une loi de Student à 2 degrés de libertés (votre expérience doit être reproductible). Sur le même graphique, tracer la vraie densité, et ses estimateurs de densité obtenus avec $h = 0.05,\ 0.337$ et 2.
3. On souhaite pouvoir faire la sélection de la fenêtre par validation croisée. Pour cela, implémenter la fonction `basic.cv` (voir fin de l'énoncé). Cette fonction a deux arguments: l'échantillon `x_num` et la fenêtre considérée `h_num`. Cette fonction retourne la valeur de $CV(h)$.
4. Appliquer cette fonction à l'échantillon simulé à la question 2 et considérant les 3 fenêtres proposées. Quelles fenêtre est sélectionnée?
5. Définir la fonction `basic.mydensity` de la façon la plus basique possible. Cette fonction prend 3 arguments un échantillon (`x_num`), les points où la densité sera évaluée pour la fenêtre sélectionnée (`xpts_num`) et un ensemble de fenêtres (`hvals_num`). Elle retourne les résultats associés à la fenêtre minimisant CV(h): valeurs de la fenêtre et de CV(h), et densité estimée aux points `xpts_num`.
6. On souhaite optimiser le code. Pour cela, faire le profilage de code pour identifier les parties les plus lentes de la fonction `basic.mydensity` puis optimiser la fonction `basic.mydensity`. Vous pouvez comparer vos résultats sur des échantillons de taille $n = 25$, $n = 50$ and $n = 10^3$.
7. On souhaite effectuer une expérience numérique pour étudier le lien entre la taille de l'échantillon et la fenêtre sélectionnée. Ainsi, on met en place une simulation où les données sont issues d'une loi de Student à 2 degrés de liberté. Pour différentes tailles d'échantillon (n = 25, 50, 100, 200, 400, 800), générer 10 réplications et estimer votre fenêtre par validation croisée. Votre expérience doit être reproductible et la simulation doit être efficace computationnellement. Effectuer la simulation et tracer la valeur de la fenêtre sélectionnée en fonction de n et en fonction de $n^{-1/5}$.
8. On souhaite pouvoir diffuser notre fonction. Pour cela, à l'aide des commandes `stop` et `warnings`, vérifier les arguments de la fonction passés par un utilisateur et proposer des valeurs par défaut pour certains arguments.

```r
basic.cv <- function(x_num, h_num){
  # Calcul de l'integrale de \hat{f}^2_n
  res.hatfsquare <- 0
  for (i in 1:length(x_num))
    res.hatfsquare <- res.hatfsquare + ksmooth3(x_num, x_num[i], h_num * sqrt(2))

  res.hatfsquare <- res.hatfsquare / length(x_num)

  # Calcul de \hat_f_{h,(-i)}
  res.hatfiremove <- NULL
  for (i in 1:length(x_num))
    res.hatfiremove <- c(res.hatfiremove,
                         2 * ksmooth3(x_num[-i], x_num[i], h_num) / length(x_num))

  # Calcul de CV
  CV <- res.hatfsquare - sum(res.hatfiremove)
  return(CV)
}
```