École Nationale
Supérieure
de l'Électronique
et de ses Applications

Statistiques Multidimentionnelles et Inférentielles

Responsable du cours : Bastien Faucard
bastien.faucard@ensea.fr

E.N.S.E.A. 2024 – 2025, Semestre 8

Travaux Pratiques

Table des matières

1 TP1 : Pierre-feuille-ciseaux

Problématique

Pierre-feuille-ciseaux est un jeu opposant deux joueurs. À chaque tour, les joueurs annoncent simultanément le coup qu'ils ont choisi parmi les trois possibilités "Pierre", "Feuille" ou "Ciseaux". Les coups gagnants sont représentés dans la figure 1.1. Si les deux joueurs choisissent le même coup, la manche est nulle.

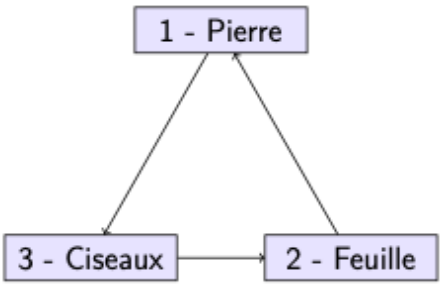

FIGURE 1.1 – Coups gagnants à Pierre-feuille-ciseaux

Une stratégie efficace pour ne pas perdre à ce jeu consiste à choisir ses coups de manière indépendante, et uniformément dans l'ensemble $\{\text{Pierre}, \text{Feuille}, \text{Ciseaux}\}$. Cependant, les humains éprouvent d'énormes difficultés à produire un véritable hasard : ils sont influencés par diverses considérations psychologiques [1]. Un ordinateur peut ainsi battre un humain, s'il arrive à anticiper les action s de son adversaire. Pour cela, il connaît l'historique (l'échantillon) des coups réalisés par son adversaire. À partir de cet échantillon, l'ordinateur peut inférer la loi de génération des coups de son adversaire humain. Il s'agit donc d'un problème d'estimation. L'objectif de ce TP est de construire une intelligence artificielle basée sur ces principes.

Modèle du comportement du joueur humain par chaîne de Markov

Nous allons dans un premier temps développer un modèle extrêmement rudimentaire. Nous allons respectivement utiliser les entiers $0, 1$ et 2 pour représenter les coups "Pierre", "Feuille" et "Ciseaux". On suppose que les choix de l'humain sont des réalisations d'un processus aléatoire $(X_k)_{k\geq 1}$ où

1. Par exemple, les hommes ont plus tendance à choisir le coup "Pierre" (en anglais *rock*, associé à une idée de virilité : "Rock is for rookies").

1. les X_k sont des variables aléatoires à valeurs dans $\{0,1,2\}$,
2. X_1 est distribuée selon la loi uniforme sur $\{0,1,2\}$,
3. pour tout $a_1,\dots,a_{n+1}\in\{0,1,2\}$, on a

$$P(X_{n+1}=a_{n+1}|X_1=a_1,\dots,X_n=a_n)=P(X_{n+1}=a_{n+1}|X_n=a_n),$$

4. pour tout $i,j\in\{0,1,2\}$, on a

$$P(X_{n+1}=j|X_n=i)=p_{i,j}$$

où les $p_{i,j}$ sont les coefficients d'une matrice

$$\mathscr{P}=\begin{pmatrix}p_{0,0}&p_{0,1}&p_{0,2}\\p_{1,0}&p_{1,1}&p_{1,2}\\p_{2,0}&p_{2,1}&p_{2,2}\end{pmatrix}$$

Un processus qui vérifie les conditions (3) et (4) est appelé *chaîne de Markov*. La condition (3) signifie que lorsque l'on cherche à prédire le coup futur X_{n+1}, l'information donnée par la connaissance du passé (les réalisation de $X_1,\dots,X_n$) n'est pas entièrement utilisée : seul l'état présent (la réalisation de X_n) permet de prédire l'état futur. La matrice $\mathscr{P}$ est appelée la *matrice de transition* du processus, $p_{i,j}$ représente la probabilité de choisir le coup j sachant que l'on avait précédemment choisi le coup i. Remarquons que les coefficients $p_{i,j}$ sont les paramètres inconnus de notre modèle.

1.1 Préparation

1.1.1 Estimation des paramètres du modèle

1. Donner une relation entre les colonnes de la matrice $\mathscr{P}$. En déduire que la loi du processus $(X_k)_{k\geq 1}$ est entièrement déterminée seulement par les six paramètres inconnus $p_{i,j}$, pour $i\in\{0,1,2\}$ et $j\in\{0,1\}$.
2. Montrer que pour tout $a_1,\dots,a_n\in\{0,1,2\}$,

$$P(X_1=a_1,\dots,X_n=a_n)=\frac{1}{3}\prod_{k=1}^{n-1}p_{a_k,a_{k+1}},$$

Indication : on pourra effectuer une récurrence sur $n\geq 1$.

3. On suppose que l'historique des coups passés est $(a_1,\dots,a_n)$. Pour $i,j\in\{0,1,2\}$, on note $n_{i,j}$ le nombre de transitions du coup i vers le coup j :

$$n_{i,j}=\mathrm{Card}\{k\in\{1,\dots,n-1\},(a_k,a_{k+1})=(i,j)\}$$

Montrer que la vraissemblance de $(X_1,\dots,X_n)$ est

$$\mathcal{L}_{X_1,\dots,X_n}(a_1,\dots,a_n,\mathscr{P})=\frac{1}{3}\prod_{i=0}^{2}\Big(p_{i,0}^{n_{i,0}}p_{i,1}^{n_{i,1}}(1-p_{i,0}-p_{i,1})^{n_{i,2}}\Big).$$

4. On appelle $(p'_{0,0},p'_{0,1},p'_{1,0},p'_{1,1},p'_{2,0},p'_{2,1})$ l'estimation du maximum de vraissemblance de $(p_{0,0},p_{0,1},p_{1,0},p_{1,1},p_{2,0},p_{2,1})$. En considérant la Log-vraissembance, montrer que pour tout $i\in\{0,1,2\}$ et $j\in\{0,1\}$

$$\frac{n_{i,j}}{p'_{i,j}}=\frac{n_{i,2}}{1-p'_{i,0}-p'_{i,1}}.$$

5. En déduire que pour tout $i,j\in\{0,1,2\}$, on a

$$p'_{i,j}=\frac{n_{i,j}}{n_{i,0}+n_{i,1}+n_{i,2}}.$$

1.1.2 Prise de décision

Grâce à la question 5, nous avons estimé les probabilités de transition à partir des n derniers coups du joueur humain. Notons i le dernier coup joué par l'humain (au tour n). L'ordinateur doit maintenant jouer au tour $n+1$ une valeur $k \in \{0,1,2\}$ que nous cherchons à déterminer. Il pourra gagner (relativement à son adversaire) 1, 0 ou -1 points.
Pour $k \in \{0,1,2\}$, on note $g(k)$ le gain relatif moyen réalisé par l'ordinateur.

6. Montrer que

$$\begin{cases} g(0) &= p_{i,2} - p_{i,1} \\ g(1) &= p_{i,0} - p_{i,2} \\ g(2) &= p_{i,1} - p_{i,0} \end{cases}$$

7. De manière idéale, l'ordinateur souhaiterait jouer au coup $n+1$ la valeur k en laquelle g admet un maximum. En prenant en compte le fait que les paramètres $p_{i,j}$ sont inconnus, proposer une méthode de prise de décision.

1.2 Travail en séance

1.2.1 Simulation d'une chaîne de Markov

Dans cette partie, nous allons générer un tirage aléatoire d'une chaîne de Markov et observer l'influence des paramètres $p_{i,j}$ de la matrice $\mathscr{P}$.

1. On pose

$$\mathscr{P} = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.3 & 0.4 & 0.3 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

   Dans le notebook récupéré sur Moodle, créer la matrice de transition $\mathbf{P}$ définie ci-dessus
2. Construire une fonction **randomPFS**(a,b), qui prend deux nombres $a, b \in [0,1]$ et qui renvoie 0 avec une probabilité de a, 1 avec une probabilité de b et 2 avec une probabilité de $1-a-b$. Expliquer votre algorithme.
3. Créer la fonction **sampleMarkov**$(\mathbf{P}, n)$ qui prend pour entrées une matrice de transition (ici $\mathscr{P}$), un entier $n > 1$ et retourne une liste simulant la chaîne de Markov de la manière suivante.

   Pour générer une chaîne de Markov, nous avons besoin d'un état initial. Celui-ci va être choisi aléatoirement, de manière uniforme, dans l'ensemble $\{0,1,2\}$. Ceci peut être fait avec "**np.random.choice**(**np.array**([0,

   Étant donné $\mathbf{A}(0)$, le résultat de notre premier tirage, on doit choisir, pour notre second tirage, un nombre tel que la probabilité d'obtenir $i \in \{0,1,2\}$ soit $p_{\mathbf{A}(0),i}$ et ainsi de suite.
   Pour tester cette fonction, générer une chaîne de Markov de taille $n = 100$.
4. Observer la liste obtenue. Expliquer ce que vous observez par l'analyse des coefficients de la matrice de transition $\mathscr{P}$.

1.2.2 Implémentation du jeu

Dans cette partie, nous allons en premier lieu programmer une interface pour pouvoir jouer à Pierre-Feuille-Ciseaux contre l'ordinateur qui jouera aléatoirement de manière uniforme. Puis nous allons programmer l'intelligence artificielle de l'ordinateur en utilisant la stratégie que vous avez étudiée dans la préparation.

7. Cette question fait référence au code préalablement écrit sur le notebook disponible sur Moodle.
   (A) Analyser le code du notebook et le compléter pour que la stratégie de l'ordinateur consiste à jouer aléatoirement de manière uniforme.

(B) Expliquer ce que représente le graphique affiché.

8. Nous allons maintenant implémenter l'intelligence artificielle. Commencer par copier la cellule de code (pour garder une trace de la question précédente).
   (A) Créer la fonction **Jouer2**(a) qui est un copier-coller de **Jouer**(a). De même, recopiez les parties **Initialisation** et **Interface Homme-Machine**.
   (B) Initialiser à zéro une matrice de taille 3×3 notée **Mem** dans la partie **Initialisation**.
   (C) Compléter la partie **Sauvegarde du coup de l'humain** de sorte que le coefficient **Mem**$[i, j]$ contienne le nombre de coups successifs (i, j) joués par l'humain à partir du troisième tour.
   (D) Vérifier que le code fonction toujours ! Vous pouvez afficher l'évolution dynamique de la matrice "Mem" en ajoutant un "label" :
   label3=Label(formulaire)
   label3.grid(row=5,column=1)
   avant la fonction **afficherPierre** puis ajouter dans chacune des trois fonctions d'affichage de l'interface la ligne :
   label3.config(text=Mem).
9. Il s'agit maintenant d'implémenter l'intelligence artificielle.
   (A) De même qu'à la question précédente : faire un copier-coller des cellules précédentes en créant la fonction **Jouer3**(a).
   (B) Dans la partie **Prise de décision de l'ordinateur**, implémenter l'intelligence artificielle par la méthode développée dans les questions préparatoires.
   Indication : vous pouvez utiliser la question 6 et le fait que $p_{i,2} - p_{i,1}$ est proportionnel à $n_{i,2} - n_{i,1}$ (c'est-à-dire que vous pouvez vous contenter d'utiliser la matrice **Mem** sans avoir à créer de matrice de transition $\mathscr{P}$).
   (E) Commenter les résultats obtenus.

1.2.3 Tests statistiques

Dans cette dernière partie, nous allons vérifier, à l'aide d'un test statistique, qu'une suite de nombres générés par un être humain "au hasard" n'a pas de caractère aléatoire parfait.

10. Créer une liste **listeHasard** que vous remplirez **vous-mêmes** de 100 entiers égaux à 1 ou -1. Vous essaierez de choisir chaque composante de manière indépendante et uniformément sur $\{-1, 1\}$.
    Créer une liste **listeMachine** composée de 100 entiers égaux à 1 ou -1 choisis de manière indépendante et uniforme sur $\{-1, 1\}$ en utilisant la fonction **np.random.choice**.

    Soit $(X_1, \ldots, X_{100})$ un vecteur aléatoire, dont les composantes X_i sont mutuellement indépendantes et uniformément distribuées sur $\{-1, 1\}$. On définit la fonction d'autocorrélation I de ce vecteur
    $$I(p) = \sum_{k=1}^{100-p} X_k X_{k+p}.$$
11. Dans cette question, on fait un peu de théorie...
    (A) Calculer la valeur de $I(0)$.
    (B) Quelle information donne $I(p)$, pour $p > 0$?
    (C) Que vaut $I(p)$ pour $p > 0$ lorsque les X_k sont des variables aléatoires mutuellement indépendantes identiquement distribuées de loi uniforme sur $\{-1, 1\}$?
12. Créer une fonction **autocorr**(X, p) qui prend en arguments une liste X de 100 entiers, un entier $p \in \{0, \ldots, 99\}$ et qui renvoie l'autocorrélation de la liste X en p.
13. Afficher l'autocorrélation de **listeHasard** et **listeMachine** sur un plot en fonction de p et commenter.
14. On cherche maintenant à déterminer empiriquement la loi de l'autocorrélation $I(1) = \sum_{k=1}^{99} X_k X_{k+1}$. Créer un échantillon **ech** de 5000 réalisations de la variables aléatoire $I(1)$.
15. Visualiser la loi empirique de l'échantillon **ech** par exemple à l'aide d'un histogramme. Calculer la moyenne et la variance empirique de **ech** [2]. En déduire une loi approximative de $I(1)$.

2. Utiliser **np.mean** et **np.var**.

16. Dans cette question, on va construire un test d'hypothèses permettant de déterminer si la liste **listeHasard** de la question 10 a un caractère aléatoire parfait (composantes choisies indépendamment et uniformément sur $\{-1, 1\}$). Pour cela
    - énoncer mes hypothèses nulle et alternative,
    - utiliser la statistique $I(1)$,
    - indiquer la valeur critique du test bilatérale pour un risque de première espèce $\alpha = 5\%$ et conclure.

    Reprendre ce test avec la liste **listeMachine**. Commenter.

1.2.4 Pierre-Feuille-Ciseaux-Lézard-Spock

Il existe une version complétée de Pierre-Feuille-Ciseaux à 5 états, appelée Pierre-Feuille-Ciseaux-Lézard-Spock. Dans cette version, les ciseaux coupent le papier, le papier recouvre la pierre, la pierre écrase le lézard, le lézard empoisonne Spock, Spock casse les ciseaux, les ciseaux décapitent le lézard, le lézard mange le papier, le papier réfute Spock, Spock vaporise la pierre et la pierre écrase les ciseaux (représentés figure 1.2).

FIGURE 1.2 – Coups gagnants au jeu Pierre-Feuille-Ciseaux-Lézard-Spock

17. En vous inspirant des méthodes employées dans ce TP, créer une intelligence artificielle permettant de battre un joueur humain à Pierre-Feuille-Ciseaux-Lézard-Spock.

2

TP2 : Tests d'hypothèses

2.1 Préparation

On considère un échantillon $(X_1, \dots, X_n)$ constitué de n variables aléatoires mutuellement indépendantes et identiquement distribuées selon une loi inconnue. Cet échantillon représente les résultats obtenus à un contrôle de synthèse de Statistiques ! ! Et on souhaite en réaliser une étude statistique.

1. Rechercher les définitions du coefficient d'asymétrie (aussi appelé *skewness*) et du kurtosis d'une loi de probabilité et expliquer ce qu'ils permettent de mesurer. Montrer que le coefficient d'asymétrie d'une loi normale est nul.
2. Nous souhaitons savoir si l'échantillon $(X_1, \dots, X_n)$ est en adéquation avec une loi normale $\mathcal{N}(\mu, \sigma^2)$, et nous réaliserons pour cela un test de χ^2.
   (A) Formuler les hypothèses nulle H_0 et alternative H_1 du test de χ^2 d'adéquation à la loi $\mathcal{N}(\mu, \sigma^2)$.
   (B) Donner les estimations du maximum de vraissemblance μ' et $(\sigma')^2$ de μ et σ^2 en fonction de la réalisation $(x_1, \dots, x_n)$ de l'échantillon $(X_1, \dots, X_n)$, sous l'hypothèse H_0.
   (C) Soient a et b deux réels avec $a < b$. Exprimer l'effectif théorique de la classe des notes appartenant à $[a, b]$ en fonction de n, de la fonction de répartition F de la loi $\mathcal{N}(\mu', (\sigma')^2)$, de a et de b.
   (D) À quelle condition sur les effectifs des classes peut-on procéder au test de χ^2 sans regroupement de classes ?
   (E) Quelle est l'influence de la valeur de α, le risque de première espèce, sur le seuil de la zone de rejet du test ?

2.2 Travail en séance

2.2.1 Adéquation à une loi normale

Soit **data** l'ensemble des notes obtenues aux contrôles de synthèse de Statistiques de l'année 2022, réalisation de l'échantillon $(X_1, \dots, X_n)$. L'objectif de cette séance est de faire une étude statistique de ces résultats.

1. Donner une estimation de la moyenne **muEstim** et de la variance **varEstim**.
2. Les valeurs obtenues sont-elles celles attendues du calcul de la question $2B$ de la préparation ? Justifier.
3. Créer une fonction **skewnessEmp**(X) et une fonction **kurtosisEmp**(X) permettant de calculer respectivement les estimateurs empiriques du coefficient d'asymétrie et du kurtosis pour une liste de données X en entrée. Générer une estimationn **skewnessEstim** et une estimation **kurtosisEstim** de ces deux coefficients pour la liste **data**. Comparer les valeurs théoriques du kurtosis et coefficient d'asymétrie à ces estimations. Vous semble-t-il plausible que l'échantillon **data** soit issu d'une loi normale ?

4. Représenter la liste **data** via un histogramme contenant 20 classes.
5. Superposer la fonction de densité d'une loi normale de moyenne **muEstim** et de variance **varEstim**. Commenter.
6. Créer une fonction **fR**(x) qui renvoie l'image de x par la fonction de répartition d'une loi normale de moyenne **muEstim** et de variance **varEstim**. On pourra utiliser le calcul intégrale du module **quad** de **scipy.integrate**.
7. Représenter la fonction de répartition **fR** sur un intervalle de votre choix et justifier ce choix.

Nous souhaitons à présent tester l'adéquation de l'échantillon **data** avec une loi normale, à l'aide d'un test de χ^2 et nous devons au préalable répartir les notes dans des classes. On choisira 20 classes $]-\infty, 1[, [1, 2[, \ldots, [18, 19[, [19, +\infty[$.

8. Créer les listes **effObs** et **effTh** qui contiennent respectivement les effectifs observés et théoriques. Vous pouvez, pour ce faire, créer une fonction **effeectif**$(X, nb_classes)$ qui compte le nombre de valeurs d'une liste X dans chaque classes. Vous utiliserez la préparation (question $2C$) pour la liste des effectifs théoriques.
9. Créer une fonction **regroupeGauche**(liste, k) qui prend en entrée une liste et un entier positif k et renvoie la liste obtenue à partir de **liste** en regroupant et en sommant les k premiers termes ; On vérifiera que

$$\textbf{regroupeGauche}([2, 4, 3, 6, 3, 7], 4)$$

renvoie $[15, 3, 7]$.

10. Faire de même à droite en créant la fonction **regroupeDroite**(liste, k).
11. À l'aide des fonctions précédentes, effectuer un regroupement de classes de sorte à ce que tous les effectifs soient plus grands que 5 (le nombre de classes à regrouper sera déterminer "à vue de nez !"). On appellera **effThBis** et **effObsBis** les nouveaux effectifs théoriques et observés.
12. Calculer la distance du χ^2 entre les effectifs théoriques et observés de la question précédente.
13. Conclure quant à l'adéquation de l'échantillon **data** à une loi normale pour un risque de première espèce $\alpha = 5\%$. On précisera le nombre de degrés de liberté du problème et on déterminera la valeur seuil du test unilatéral gauche du χ^2 sur la table du polycopié de cours.

2.2.2 Prise en compte de l'asymétrie

L'histogramme de la question 4.A devrait sembler légèrement "étalé vers la droite". Dans cette partie, nous allons tester l'adéquation de **data** à une version modifiée de la loi normale, appelée loi normale asymétrique (*skew normal distribution*). Cette loi est décrite par trois paramètres (au lieu de deux pour la loi normale) : le paramètre de position $m \in \mathbb{R}$, le paramètre d'échelle $s > 0$ et le paramètre de forme $a \in \mathbb{R}$.

14. Compiler le code permettant de définir la densité d'une loi normale asymétrique de paramètres m, s, a : c'est la fonction **skewnorm**(x, m, s, a). Quelle valeur de a faut-il choisir pour que **skewnorm**(x, m, s, a) soit une loi normale ? Il n'est pas demandé de faire des calculs ici mais plutôt de comparer les paramètres des lois concernées ou leurs densités. Afin d'observer l'effet de l'asymétrie dans un cas particulier, superposer les densités de probabilité de la loi normale centrée réduite et de la loi normale asymétrique de paramètres $0, 1$ et 4.
15. Créer une fonction **logvraissemblance**(m, s, a) qui donne la Log-vraissemblance de l'échantillon **data** pour le modèle de la loi normale asymétrique.
16. Déterminer les estimations du maximum de vraissemblance **mEstim**, **sEstim** et **aEstim** des paramètres m, s, a (compiler le code donné, il peut être un peu long).
17. Superposer l'histogramme de la question 4 avec le graphe de la densité de probabilité de la loi normale asymétrique de paramètres **mEstim**, **sEstim** et **aEstim**. Commenter.
18. Effectuer le test du χ^2 d'adéquation de l'échantillon **data** à une loi normale asymétrique et conclure.

2.2.3 Indépendance entre deux caractères

Dans cette dernière partie, nous allons tester l'indépendance (supposée) entre le sexe d'un individu et ses résultats aux contrôles de Probabilités et Statistiques. On dispose pour cela des deux sous-échantillons **dataFemmes** et **dataHommes** de notes obtenues au contrôle de synthèse de Statistiques de 2022.

19. Créer la matrice de contingence des fréquences observées et celle des fréquences théoriques sous l'hypothèse d'indépendance du sexe aux résultats obtenues. Les classes seront les suivantes

$$[0, 10.5[;\ [10.5, 13[;\ [13, 15[;\ [15, 16[;\ [16, 20].$$

Calculer la distance du χ^2 de ce test d'indépendance, puis conclure (test unilatéral gauche) en précisant la valeur critique et le nombre de degrés de liberté au seuil de risque de première espèce de 5%.

3

TP3 : Estimation de densité

3.1 Préparation

Ce TP fait référence au chapitre 3 du cours qui n'a pas été traité en amphi. N'hésitez pas à vous y référer.

Considérons une réalisation $x_1, \dots, x_n$ d'un échantillon $X_1, \dots, X_n$ de variables identiques et indépendantes de densité commune f. Le but de ce TP est de comparer les noyaux utilisés pour estimer la densité commune f. Il faut bien comprendre que dans la pratique f est inconnue, ici, pour comparer l'efficacité des noyaux et la taille de la fenêtre h nous allons supposer dans une première partie que f est la densité d'une gaussienne centrée réduite, dans une seconde partie nous supposerons que f est la densité d'une loi de dimension plus grande.

1. Si $K : \mathbb{R} \to \mathbb{R}_+$ est un noyau statistique et $\mu \in \mathbb{R}$ une constante, la translation de K par la constante a, $\tau_\mu K$, est-elle encore un noyau statistique ?
2. Si $K : \mathbb{R} \to \mathbb{R}_+$ est un noyau statistique et $\lambda \in \mathbb{R}*$ une constante non nulle, montrer que $d_\lambda K$ définie pour tout $x \in \mathbb{R}$ par $d_\lambda K(x) = \frac{1}{\lambda} K\left(\frac{x}{\lambda}\right)$ est encore un noyau statistique.
3. Montrer que $K = \frac{1}{2} 1_{[-1,1]}$ est un noyau statistique, on l'appel le noyau uniforme.
4. Montrer que $K(x) = (1 - |x|) 1_{[-1,1]}(x)$ est un noyau statistique, on l'appel le noyau triangle.
5. Montrer que $K(x) = \frac{3}{4}(1 - x^2) 1_{[-1,1]}(x)$ est un noyau statistique, on l'appel le noyau d'Epanechnikov.
6. Montrer que $K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ est un noyau statistique, on l'appel le noyau gaussien.

Soit $h > 0$ une constante appelée la fenêtre. Soit K un noyau statistique. On considère la fonction $\widehat{f_h}$, définie pour tout $x \in \mathbb{R}$, par :

$$\widehat{f_h}(x) = \frac{1}{n} \sum_{k=1}^{n} d_h \tau_{X_i} K(x)$$

C'est l'estimation de la densité f avec la fenêtre h et le noyau K.

7. Montrer que $\widehat{f_h}$ est une densité de probabilité.

3.2 Travail en séance : partie 1

Le but de cette partie est de définir, représenter et comparer l'efficacité des quatre noyaux de la préparation pour l'estimation de la densité d'une gaussienne standard f. On suppose donc que $X_1, \dots, X_n$ est un échantillon de taille n de variables indépendantes et identiquement distribuées selon la loi normale centré réduite de densité f.

1. Définir quatre fonctions $K1, K2, K3, K4$ correspondant respectivement aux noyaux uniforme, triangle, d'Epanechnikov et gaussien.
2. Représentez ces quatre noyaux sur un même graphique (utiliser une légende et des couleurs différentes). Créer une fonction pour faire cette question que vous nommerez **AllplotK** qui prendra en entrée les paramètres du graphique (le pas, xmin, xmax, les couleurs etc...) et représentera le graphique en retour.
3. Générer une réalisation de l'échantillon aléatoire X selon la loi gaussienne standard de taille n. (n est pour l'instant fixé à 100 dans le script).
4. Définir la fonction **fchapeau** qui prend comme argument une fonction K (le noyau), la fenêtre h et la réalisation de l'échantillon X et une variable x et qui retourne l'image de x par la fonction $\widehat{f}_h$.
5. Représenter sur un même graphique la fonction f de référence ainsi que les quatre fonctions $\widehat{f}_h$ obtenues avec les noyaux $K1, K2, K3, K4$. Vous ajouterez une légende et des couleurs différentes à toutes les courbes. On fixera pour cette question $h = 2$. Vous définirez une fonction comme dans la question 2 pour faire cette question. Cette fonction sera nommée **Allplotfchapeauh2**.
6. Refaire la question précédente avec $h = 1$. Qualitativement, est-ce que l'estimation diffère plus lorsque l'on fait varier le noyau utilisé ou la fenêtre h utilisée ? La nouvelle fonction pour cette question sera nommée **Allplotfchapeauh1**.
7. Reprendre les deux questions précédentes pour $n = 10$ puis $n = 1000$. Pour cette question, quatre graphiques doivent être construits : le premier pour $(n, h) = (10, 2)$, le second pour $(n, h) = (10, 1)$, le suivant pour $(n, h) = (1000, 2)$ et le dernier pour $(n, h) = (1000, 1)$. Vous détaillerez votre raisonnement dans le script et commenterez les résultats obtenus. Revenir ensuite à la valeur de $n = 100$ dans le script pour la suite du TP.
8. Nous allons calculer l'erreur quadratique d'une estimation : soit

$$SCE(h) = \sum_{i=0}^{500} (\widehat{f}_h(t_i) - f(t_i))^2$$

   la somme de carrés des écarts entre l'image de t_i par l'estimation $\widehat{f}_h$ et l'image de t_i par f, où $\{t_0, t_1, t_2, \ldots, t_{500}\}$ est une discrétisation de l'intervalle $[-5, 5]$ de pas $10/500$.
   Autrement dit

$$-5 = t_0 < t_1 = -5 + 10/500 < t_2 = -5 + 20/500 < \cdots < t_{500} = -5 + 5000/500 = 5$$

   Définir une fonction **SCE** qui prend comme paramètre une fonction (le noyau considéré), la fenêtre h, la densité de référence f et qui retourne $SCE(h)$.
9. Définir une fonction **lemeilleurh** qui prend une fonction (le noyau en question) et une autre fonction f (la référence) en paramètres et retourne l'index divisé par 100 du minimum de la liste $\{SCE(k/100)\}_{1 \leq k \leq 200}$. Pour chaque noyau, la meilleur fenêtre pour l'estimation de la fonction de référence est donnée par cette fonction.
10. Définir alors une fonction qui représente graphiquement les quatre estimations de densité pour ces quatre noyaux avec les fenêtre obtenue via la fonction **lemeilleurh**. Définir pour ce faire, la fonction **Allplotfchapeauhoptimal**

3.3 Travail en séance : partie 2

Nous allons maintenant exploiter les fonctionnalités de **scikit-learn**. La fonction **estimationdensite** présente dans le script sert à effectuer une estimation de densité par noyau gaussien (**kernel**=$'gaussian'$) avec pour fenêtre h dont la densité de référence est un mélange gaussien (de deux gausiennes de moyennes mu1, mu2 et d'écartypes sigma1, sigma2). Cette fonction fait appel au package scikit-learn.

1. Executer cette fonction avec $mu1 = 0$, $mu2 = 5$ et $sigma1 = sigma2 = 1$, $N = 100$ et $h = 0.75$.
2. Comparer à tout autre paramètre fixés comme dans la question précédente, l'influence de la fenêtre h. On pourra tester des valeurs de h comprises entre 0.2 et 1.5. Commenter.

3. Faite varier les paramètres des deux lois gaussiennes qui définissent le mélange gaussien. Commenter.
4. Faite varier N et commenter.
5. On peut aussi tester d'autres noyaux par exemple en remplaçant '*gaussian*' dans le code par '*epanechnikov*'. Réaliser ce graphique en executant la fonction **estimationdensite2** avec les mêmes paramètres que ceux de la question 1.

4

TP4 : La fonction de Rosenbrock

L'objectif de ce TP est d'étudier la fonction de Rosenbrock

$$f\colon \mathbb{R}^2 \to \mathbb{R}$$
$$(x, y) \mapsto (1-x)^2 + a^2(x-y^2)^2.$$

Et cette fonction va également fournir un exemple d'application d'algorithme de descente par gradient.

4.1 Préparation

1. Déterminer le gradient $\nabla f(x, y)$ de la fonction f en un point $(x, y) \in \mathbb{R}^2$ quelconque.
2. Résoudre le système $\nabla f(x, y) = 0$ pour trouver les points critiques de f. Si $a \neq 0$, on doit trouver trois points critiques que l'on nomme a_1, a_2, a_3.
3. Déterminer la matrice hessienne $\mathrm{Hess}_f(x, y)$ de la fonction f en un point $(x, y) \in \mathbb{R}^2$ quelconque puis aux points a_1, a_2, a_3.
4. En déduire la nature des points critiques a_1, a_2, a_3 en déterminant les valeurs propres de $\mathrm{Hess}_f(a_i)$ pour $i \in \{1, 2, 3\}$.

4.2 Travail en séance

Les résultats de la préparation vont être vérifiés de trois manières : graphiquement, via le calcul symbolique et numériquement via l'algorithme de descente par gradient. Premièrement définir la fonction **rosenbrock**(X, a) qui retourne $f(X)$ pour $X \in \mathbb{R}^2$ et $a \in \mathbb{R}$.

4.2.1 Partie 1 : Analyse graphique

1. Définir la fonction **graph_Rosenbrock**(a, X, Y) qui prend en paramètres une valeurs de a, deux arrays X et Y et retourne la représentation graphique $(X, Y, f(X, Y))$ en trois dimensions (utiliser **meshgrid**). Tester la fonction avec $a = 1$, et deux arrays X et Y. Utiliser **np.linspace** pour les listes X et Y.
2. Représenter la fonction f avec $a = 0.2$ avec les mêmes intervalles X et Y que la question précédente.
3. Implémenter une fonction **graph_Rosenbrock_contour**(a, X, Y, nb_lignes) qui donne la représentation de f via **nb_lignes** lignes de niveaux. Utiliser **plt.contour**.
4. En faisant plusieurs graphiques, essayez de repérer un encadrement des points critiques de f.

4.2.2 Partie 2 : Analyse symbolique

Pour l'analyse symbolique de la fonction f comme elle a été faite à la préparation, il faut utiliser **sympy**.

1. Définir trois variables symboliques x, y, a, ainsi que la variable symbolique **sym_Rosenbrock** égale à $f(x, y)$.
2. Pour déterminer le gradient de f, on peut utiliser la méthode **diff** et pour la disposition matricielle du résultat **sym.Matrix**. Déterminer le gradient de f au point (x, y), celui-ci sera stocké dans la variable **sym_gradient**.
3. Ensuite, résolvez le système $\nabla f(x, y) = 0$ à l'aide du module **solve** de **sympy**. Vous devez retrouver les points critiques a_1, a_2, a_3 de la préparation.
4. Déterminer la matrice $\text{Hess}_f(x, y)$ en utilisant la méthode **jacobian**.
5. Maintenant, déterminer $h_i = \text{Hess}_f(a_i)$ pour $i \in \{1, 2, 3\}$. Utiliser la méthode **subs**.
6. Via le module **eigenvalues**, déterminer les valeurs propres de h_i pour $i = 1, 2, 3$.
7. Retrouve t'on les résultats de la préparation sur la nature des points critiques ?

4.2.3 Partie 3 : Analyse numérique

Dans toute cette partie, le paramètre a sera fixé à $a = 1$.

1. Écrire une fonction **rosenbrock_grad(X)** qui donne la valeur numérique du gradient de f au point X pour $a = 1$.
2. Compiler le code présenté dans le notebook. Que représente la flèche rouge ? La longueur de cette flèche a t'elle une signification ?
3. Rappeler l'algorithme de descente par gradient à pas constant.
4. Définir une fonction **gradient_descent(J_grad, x_init,alpha, max_iterations, epsilon)** qui prend en entrée une fonction **J_grad** donnant le gradient d'une fonction f, un array **x_init** correspondant à l'initialisation de l'algorithme de descente par gradient, un nombre strictement positif **alpha** correspondant au pas constant de descente, un nombre maximum d'itération et un nombre réel strictement positif **epsilon** servant de test d'arrêt et retourne la position x, le nombre d'itération et un tableau avec les positions successives durant l'execution de l'algorithme pour avoir une mémoire du parcours.
5. Tester cette fonction en faisant varier les paramètres **alpha, x_init** et **max_iterations**. La solution numérique correspond t'elle à une des solutions analytiques de la partie 2 ?
6. Executer le code proposé dans le notebook et commenter le visuel obtenu.
7. Rappeler l'algorithme de descente à pas optimal.
8. Pour trouver le pas optimal à chaque itération, on donne la fonction **gss(f,a,b,tol)** mettant en oeuvre l'algorithme "Golden Section Search". Reprendre les questions précédentes avec l'algorithme de descente par gradient à pas optimal jusqu'à obtenir le même type de graphique qu'à la question 6. Comparer qualitativement ces deux méthodes.
9. Dans cette question, l'objectif, étant donné un point de départ **x_init**$= (0.1, 0.9)$, est de comparer les erreurs quadratique (en $||\cdot||_2$, utiliser **np.linalg.norm**) à chaque itération entre l'algorithme de descente à pas constant et l'algorithme de descente à pas optimal. Représenter graphiquement l'évolution de l'erreur quadratique dans les deux cas en échelle linéaire puis logarithmique.
10. Commenter les résultats obtenus.