TP-L3 maths pour l'enseignement, probabilités

1 Probabilités

1.1 Des paradoxes probabilistes

Pour chacun des problèmes suivant il s'agit de simuler le problème sur l'ordinateur puis de répéter un grand nombre de fois l'expérience pour vérifier la prédiction probabiliste.

1.1.1 Les urnes de Bertrand

Sur une table sont posées trois urnes contenant des boules blanches ou noires :

- L'urne NN contient 2 boules noires
- L'urne NB contient 1 boule noire et une boule blanche
- L'urne BB contient 2 boules blanches.

On choisit une urnes aléatoirement (avec une probabilité 1/3,1/3,1/3) puis on pioche une boule aléatoirement (avec proba 1/2,1/2).

Question. *La boule piochée est noire, quelle est la probabilité que l'urne choisie soit l'urne NN ?*

On commence par étudier le problème numériquement :

1. Écrire une fonction qui modélise le tirage aléatoire. Puis une autre fonction qui recommence le tirage jusqu'à ce que la boule piochée soit noire.
2. Écrire une fonction qui répète l'expérience un grand nombre de fois pour répondre à la question

Mathématiquement :

1. Répondre à la question avec un tableau/arbre de probabilité
2. Rappeler la formule de Bayes

1.1.2 Monty Hall

Durant un jeu télévisé, on présente à un participant trois portes A,B,C. Derrière une de ces trois portes (choisie aléatoirement) est cachée une voiture. Les deux autres portes cachent une chêvre. Sans plus d'information, le joueur choisit une porte (par exemple : A) et recevera ce qu'il y a derrière en cadeau. Le présentateur de télévision à la place d'ouvrir celle ci, ouvre une autre porte (par exemple : B) et derrière celle ci apparait une chêvre. Il pose alors la question au joueur souhaite il changer son choix et ouvrir la dernière porte (dans l'exemple : C) ou persister dans sa décision et ouvrir la première porte choisie (dans l'exemple : A).

Question 1. *Est il préférable pour le joueur de changer de porte ?*

On commence par étudier le problème numériquement :

1. Écrire une fonction qui modélise le problème.
2. Écrire une fonction qui répète l'expérience un grand nombre de fois et pour estimer la probabilité que le joueur reparte avec la voiture
   (a) Si il applique la stratégie : toujours ouvrir la première porte choisie
   (b) Si il applique la stratégie : toujours changer de porte.

Mathématiquement :

1. Répondre à la question avec un tableau/arbre de probabilité.

1.1.3 Le paradoxe des anniversaires

Dans une classe avec N élèves, on note les dates d'anniversaire de chacun. On supposera que celles ci sont des entiers aléatoires dans $\{1, \cdots, 365\}$ (année non bissectile) avec une probabilité uniforme.

Question 2. *Quelle est la probabilité que deux élèves (ou plus) soient nés le même jour ?*

Étudier le problème numériquement :

1. Écrire une fonction qui simule une classe avec N élèves et qui dit si OUI ou NON deux élèves sont nés le même jour.
2. Pour un N fixé, écrire une fonction qui répète l'expérience un grand nombre de fois pour estimer la probabilité que deux élèves sont nés le même jour.
3. Faire varier N entre 2 et 35 élèves, et tracer dans un diagramme la probabilité pour les différentes tailles de classe.

Mathématiquement :

1. Pour un N fixé, calculer la probabilité que tous les élèves soient nés des jours différents.
2. Comparer cette prédiction avec les résultats numériques.

1.1.4 Bataille de Dés

On dispose de trois dés un peu particuliers dont les numéros sur leurs faces sont les suivantes :

- Dé A : 3,3,3,3,3,6
- Dé B : 2,2,2,5,5,5
- Dé C : 1,4,4,4,4,4

Un premier joueur choisi un dé et le lance. Le deuxième joueur choisi un autre dé et le lance. Le gagnant est celui qui obtient le plus grand nombre sur son dé.

Question 3. *Quel est le meilleur dé ?*

Étude numérique

1. Écrire une fonction qui simule le lancé du dé A et du dé B et dit lequel de ces deux dé a gagné.
2. Écrire une fonction qui répète l'expérience un grand nombre de fois pour estimer la probabilité que A gagne contre B.
3. Faire de même pour les duels A contre C et B contre A.

Mathématiquement : on pourra dans un tableau 6×6 écrire le résultats possible entre le dé A et le dé B. Puis faire de même avec les duels A contre C et B contre A.

1.2 Les grands théorèmes

1.2.1 La loi forte des grands nombre

Soit $(X_n)_{n\in\mathbb{N}}$, des variable aléatoire indépendantes et identiquement distribués.

Theorem 4. *Si* $\mathbb{E}|X| < \infty$ *alors*

$$\frac{\sum_{i=1}^{n} X_i}{n} \to \mathbb{E}(X) \quad \textit{presque surement}$$

Vérifier numériquement ce théorème pour les loi suivant

1. La loi Bernoulli $B(p)$ pour différentes valeurs de p.
2. La loi uniforme $U([0,1])$
3. La loi gaussienne standard.
4. La loi de Cauchy standard

Pour chacun des cas mentionnés ci dessus il s'agira de pour un N suffisament grand tirer aléatoirement les $(X_i)_{i\leq N}$ et de tracer sur une courbe la suite

$$Y_k = \frac{\sum_{i=1}^{k} X_i}{k}$$

pour $1 \leq k \leq N$.

1.2.2 Le théorème centrale limite

Theorem 5. *Si* $\mathbb{E}|X|^2 < \infty$ *alors*

$$\frac{\sum_{i=1}^{n} X_i - \mathbb{E}(X)}{\sqrt{n}} \to \mathcal{N}(0, Var(X_1)) \quad en\ loi$$

Vérifier numériquement ce théorème pour les loi suivant

1. La loi Bernoulli $B(p)$ pour différentes valeurs de p.
2. La loi uniforme $U([0,1])$
3. La loi gaussienne standard.
4. La loi de Cauchy standard

Pour chacun des cas mentionnés ci dessus il s'agira de pour un N suffisament grand

1. Écrire une fonction qui tirer aléatoirement les $(X_i)_{i\leq N}$ et donne

$$Y_N = \frac{\sum_{i=1}^{N} X_i - \mathbb{E}(X)}{\sqrt{N}}$$

2. Répéter l'expérience un grand nombre de fois et enregistrer les différentes valeurs obtenues dans un tableau.
3. Tracer un histogramme avec les valeurs du tableau. Il faut comparer cet histogramme avec la courbe gaussienne.

1.3 Méthode Monte Carlo

La méthode de Monte-Carlo est une méthode probabiliste à la fois simple et efficace (mais pas extrèmement précise) pour estimer la surface d'une figure F. Le principe est le suivant on suppose $F \subset A$ dont la surface de A est connu, par exemple un carré ou un rectangle. Puis on tire un grand nombre de point aléatoirement dans A avec une probabilité uniforme. On estime alors la surface de F par

$$\text{Surface}(F) \approx \frac{\text{nombre de point dans } F}{\text{nombre total de points}}$$

1.3.1 Des surfaces connues

Numériquement :

1. On cherche à retrouver l'aire d'un disque.
   (a) Tracer dans le carré $[-1,-1]^2$ un cercle de rayon 1 centré en $(0,0)$. Tirer aléatoirement $N = 20$ points uniformément dans $[-1,-1]^2$ et les afficher sur la même figure.

(b) Écrire une fonction MonteCarloDisque qui tire N points aléatoirement et donne la fréquence des points tombés à l'intérieur du cercle.

(c) Donner une estimation de π.

2. Estimer avec la méthode de Monte-Carlo la surface d'un triangle délimité par les points $A = (0, 1)$, $B = (1, 0)$, $C = (-1, 1)$

3. Estimer l'aire sous la courbe de fonction $1 + \cos(2\pi x)$ pour $x \in [-1, 1]$.

1.3.2 Le volume de l'hypercube

Avec la méthode Monte Carlo estimer le volume de la boule dans $\mathbb{R}^3$, puis celle de l'hyperboule $B_n = \{x_1^2 + \cdots + x_n^2 \leq 1\} \subset \mathbb{R}^n$.

A Commandes python

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
import scipy.integrate as integ
```

Fonctions utiles

```
M=np.array([[1, 3, 5],
[-2, 6, 1]])
M>2
M[M>2]
indices=np.nonzero(M>2)

a=-2; b=2; h=0.5
x=np.arange(a, b+h/2, h)
y=np.linspace(a, b,
round((b-a)/h+1))

l=np.logical_and(x>-1, x<1)
x[l]=3.14

np.cumsum(x)
np.sum(x)
np.mean(x)

poids=np.ones(x.size)
poids[0::2]=2
np.average(x)
np.average(x, weights=poids)

np.sort(x)
-np.sort(-x)
np.flip(np.sort(x),axis=0)
np.flipud(np.sort(x))

uni_x,num_x=np.unique(x,
 return_counts=True)
```

Génération d'aléa

```
npr.rand()
npr.rand(2, 4)
npr.randint(-5, 10, (2, 10))
a=-1; b=1;
npr.uniform(a, b, 1000)
lambd=2
npr.exponential(lambd,1000)
mu=1; sigma=2;
npr.normal(mu, sigma, 1000)
n=5; p=0.2;
npr.binomial(n, p, 1000)
```

La fonction `npr.rand` dépend d'une "graine", une suite (u_n) initialisée à l'ouverture de Python. À graine fixée, `npr.rand` fournira toujours la même série de valeurs. La commande `npr.seed(k)` permet de fixer la valeur de cette graine à k.

```
npr.seed(10);
npr.rand()
npr.rand()

npr.seed(10)
npr.rand()
```

Affichage

```
N=1000
tab_n=np.arange(1, N+1)
plt.plot(tab_n, np.cumsum(npr.rand(N))/tab_n)
plt.show()

n=10; p=0.2
u=npr.binomial(n, p, N)
x=np.sort(u)
y=tab_n/N
plt.plot(x, y)
plt.axis([-0.5, n, 0, 1])
plt.show()

U=npr.rand(2, N)
plt.plot(U[0, :], U[1, :], 'r.')
plt.show()
```

La commande `npr.hist` permet de tracer des histogrammes. (voir l'aide pour les options)

```
L=npr.normal(2, 1, 1000)
plt.hist(L, bins=100);
plt.show()
plt.hist(L, bins='auto')
plt.show()
plt.hist(L, bins='auto', density=True)
plt.show()
```
