Devoir Libre n°2

Réponse le 02/10/24

Exercice 1

Ce problème étudie la génération aléatoire de sous-ensembles. Les programmes demandés devront être écrits en python, en respectant bien la présentation.Si vous n'arrivez pas à écrire une des fonctions demandées vous pouvez quand même considérer qu'elle existe et qu'elle retourne le résultat attendu.

Ce problème comporte des questions d'informatiques et des questions de probabilités. On rappelle que la fonction `randint(a,b)` du module `numpy.random` renvoie un nombre aléatoire entier pris au hasard uniformément dans l'intervalle $[\![a,\, b-1]\!]$.
On considère que le programme commence par le préambule

```python
import numpy.random as rd
```

Dans tout ce problème on s'intéresse à des sous-ensembles (ou parties) de l'ensemble $[\![0,\, n-1]\!]$ où n est un entier naturel non nul et fixé. On rappelle que $[\![0,\, n-1]\!]$ est un ensemble à n éléments. On note $Card(E)$ le nombre d'éléments d'un ensemble fini. Par exemple

$$Card(\{0,2\}) = 2 \qquad Card(\emptyset) = 0 \qquad Card(\{0,2,4\}) = 3$$

1. **Préliminaires :** Rappeler pourquoi le nombre de parties à p éléments de l'ensemble $[\![0,\, n-1]\!]$ est $\binom{n}{p}$.

<ins>RÉPONSE:</ins>

Choisir une partie de p éléments revient à choisir p élements sans ordre et sans répétition dans $[\![0,\, n-1]\!]$

* * *

2. **Codage d'un sous ensemble.** On décide de représenter un sous-ensemble de $[\![0,\, n-1]\!]$ par une liste de longueur n composée de 0 et de 1, avec la règle suivante. Si S est le sous ensemble représenté par la liste L $L[i] = 1$ si seulement si $i \in S$.
   Par exemple si $n = 4$ la liste $[0, 0, 0, 1]$ représente $\{3\}$ et la liste $[1, 0, 1, 0]$ représente $\{0, 2\}$.
   (a) Quel sous-ensemble représente la liste $[0, 1, 1, 1]$?

<ins>RÉPONSE:</ins>

$$\boxed{[0,1,1,1] \textit{ représente l'ensemble } \{1,2,3\}}$$

* * *

   (b) Dans $[\![0,\, n-1]\!]$ comment sont représentés $\emptyset$ et $[\![0,\, n-1]\!]$?

<ins>RÉPONSE:</ins>

$\emptyset$ est représenté par la liste de longueur n dont tous les coefficients sont nuls et $[\![0,\, n-1]\!]$ par la liste dont tous les coefficients sont égaux à 1.

* * *

   (c) Écrire une fonction `cardinal(L)` qui pour une liste représentant un sous-ensemble de $S \subset [\![0,\, n-1]\!]$ renvoie le nombre d'éléments de S.

```python
def cardinal(L):
    s=0
    for x in L:
        s=s+x
    return s
```

ou

```python
def cardinal(L):
    return sum(L)
```

3. **Génération aléatoire d'une partie de $[\![0,\, n-1]\!]$.**
   On se propose de générer aléatoirement un sous partie de $[\![0,\, n-1]\!]$, en utilisant l'algorithme suivant :
   - On initialise une liste `L` avec la liste vide.
   - Pour i allant de 0 à $n-1$, on tire au hasard un nombre entier dans $\{0, 1\}$ et on attribue ce nombre à `L[i]`.
   - Une fois la liste ainsi construite, la fonction la renvoie.

   Écrire une fonction `sous_ensemble(n)` qui pour un entier n renvoie une partie de $[\![0,\, n-1]\!]$

RÉPONSE:

```python
def sous_ensemble(n):
    L=[]
    for i in range(n):
        L.append(rd.randint(2))
    return L
```

* * *

4. **Test de la fonction précédente.**

   On cherche à tester si la fonction renvoie bien un sous-ensemble choisi au hasard. On se propose pour cela de calculer l'espérance du nombre d'éléments d'un sous ensemble construit au hasard et de comparer la valeur théorique et une valeur obtenue par estimation. On note X_n le nombre d'éléments d'une partie construite selon le procédé précédent et on admet que X_n est une variable aléatoire.

   (a) Justifier que X_n suit une loi binomiale dont on précisera les paramètres. Donner son espérance.

RÉPONSE:

X_n compte le nombre de succès " choisir que l'élément i appartient au sous ensemble" lors de la répétition de n expériences de Bernoulli indépendantes et équilibrées.

$$\boxed{X_n \hookrightarrow \mathcal{B}\left(n, \frac{1}{2}\right) \text{ et } E(X_n) = \frac{n}{2}.}$$

* * *

   (b) On donne le programme suivant

```python
n=10
N=10000
s=0
for i in range(N):
    s=s+cardinal(sous_ensemble(n))
print(s/N)
```

Expliquer ce que fait ce programme, et notamment le sens des variables.

RÉPONSE:

on fixe n à 10, on va générer des sous ensembles de $[\![0, 9]\!]$. *On génère* $N = 10000$ *sous ensembles dans une boucle for. Pour chacun de des sous ensemble on en calcule le cardinal. Comme on calcule la somme de tous les cardinaux et que l'on divise par le nombre d'expériences, on obtient* ***la moyenne des cardinaux***

* * *

(c) Ce programme affiche la valeur 5.0173. Que peut on en conclure ?

RÉPONSE:

D'après une question précédente on devrait obtenir une valeur proche de $10 \times 0.5 = 5$ *ce qui est cohérent.*

* * *

5. **Génération aléatoire d'une partie dont le nombre d'éléments est fixé.**
On s'intéresse maintenant à un problème légèrement différent. On fixe un entier $p \in [\![0, n]\!]$, et on veut générer aléatoirement un sous-ensemble de $[\![0, n-1]\!]$ qui comporte exactement p éléments.
On se propose de créer la liste représentant ce sous-ensemble de la manière suivante :
   - On initialise une liste L, de longueur n, avec n zéros.
   - On tire un entier nombres au hasard entre 0 et $n-1$.
   - Si cet entier n'est pas encore dans le sous ensemble, on le met et il nous reste un entier de moins à choisir.
   - On recommence à l'étape 2 tant que l'on n'a pas choisi p entiers.

   On admettra que cette méthode fonctionne. Écrire une fonction `partie(n,p)` qui renvoie une liste représentant un sous-ensemble à p éléments de $[\![0, n-1]\!]$.

RÉPONSE:

```python
def partie(n,p):
    L=[0]*n
    r=rd.randint(n)
    while p>0:
        if L[r]==0:
            L[r]=1
            p=p-1
        r=rd.randint(n)
    return L
```

* * *

6. Test de la fonction précédente
   (a) Quelle est la probabilité qu'une partie à p éléments de $[\![0, n-1]\!]$, contiennent l'élément 0

RÉPONSE:

* * *

   (b) Écrire un programme qui effectue $N = 10^5$ expériences "choisir une partie à 5 éléments dans $[\![0, 14]\!]$" et qui calcule la fréquence des sous-ensembles qui contiennent 0.

RÉPONSE:

```python
i=0 #on peut changer le nombre dont on calcule la fréquence d'apparition.
N=10000 #nombre d'expériences.
n=15
p=5
s=0
for  j in range(N):
    L=partie(n,p)
    s=s+L[i] #on ajoute  1 ssi i appartient à l'ensemble
print(s/N)
```

* * *