Lois

En théorie, la loi d'une variable aléatoire est une mesure de probabilités, c'est-à-dire une application qui à un ensemble associe un nombre compris entre 0 et 1 et qui satisfait la propriété de σ-additivité. Sur un espace dénombrable, tout ensemble est réunion (au plus dénombrable) des singletons qui le composent :

$$A = \cup_{a \in A} \{a\}.$$

Par conséquent, toute loi est complètement caractérisée par sa valeur sur les singletons, c'est-à-dire que l'on a besoin de connaître

$$\mathbf{P}(X = a), \ \forall a \in E,$$

où E est l'espace des valeurs possibles de X. Puisque les espaces considérés sont tous au plus dénombrable, on peut toujours supposer qu'il existe une injection entre E et $\mathbf{N}$ donc, en pratique, la loi de X s'identifie à une suite, éventuellement à nombre fini de termes, de réels de $[0, 1]$, de somme égale à 1.

Lois classiques

Il est utile de connaître les quatre grandes lois discrètes que l'on rencontre tout le temps.

1. La loi uniforme. Celle que l'on peut mettre sur tout ensemble fini si l'on n'a pas d'information. Chaque fois que l'on tire « au hasard », on sous-entend un tirage selon la loi uniforme. L'espace des valeurs n'est pas forcément un sous-ensemble de $\mathbf{R}^d$, cela peut être aussi un sous-ensemble de l'ensemble des permutations, de $\mathbf{Z}/p\mathbf{Z}$, etc.
2. La loi binomiale. Elle représente le nombre de succès parmi N tentatives indépendantes et identiques lorsque la probabilité de succès de chaque tentative est p. Toute la richesse de cette loi réside dans ce que l'on met derrière le mot succès. Cela peut-être l'obtention d'un `pile` dans le jeu de `pile/face`, mais aussi un nombre de personnes présentant une caractéristique particulière de probabilité d'apparition p. Il est utile de savoir que la moyenne d'une binomiale de paramètre N et p est donnée par Np, sa variance par $Np(1-p)$.
   Cela se retrouve simplement en considérant qu'une loi binomiale est la convolution de N loi de Bernoulli de paramètre p. La moyenne est alors la somme des moyennes, soit Np et par indépendance, la variance est la somme des variance, soit $Np(1-p)$.
3. La loi géométrique. Elle représente la loi du nombre de tentative avant le premier succès lorsque chaque essai réussit avec probabilité p. Son espérance est $1/p$ [1].
4. La loi de Poisson. C'est la loi dite des événements rares. Par exemple, le nombre d'appels simultanés traités par une antenne de téléphonie mobile suit une loi de Poisson : chaque individu dans la zone de couverture de l'antenne a une très faible probabilité d'appeler mais le grand nombre d'individus mène à ce que la loi binomiale ressemble à une loi de Poisson (voir l'épreuve du CCMP de 2015). La moyenne et la variance d'une loi de Poisson sont toutes les deux égales à λ.

La loi de Poisson partage avec la loi binomiale la propriété d'être stable par convolution :

— la somme de deux variables aléatoires indépendantes de loi de Poisson de paramètre respectif λ et μ suit une loi de Poisson de paramètre $\lambda + \mu$.

— la somme de deux variables aléatoires indépendantes de loi binomiale de paramètres respectifs (N, p) et (M, p) suit une loi binomiale de paramètres $(N + M, p)$. Attention, le taux de succès doit être le même : si l'on interprète ce résultat en termes de `pile/face`, cela revient à dire que la loi du nombre de `pile` dans N lancers puis dans M lancers, sans changer de pièce, est bien la loi du nombre de `pile` dans $N + M$ lancers !

1. A priori, il faut supposer l'existence d'un nombre infini dénombrable de variables aléatoires puisque le nombre d'essais n'est pas borné, ce qui stricto sensu sort du cadre du programme de CPGE puisqu'un tel espace probabilisé n'est pas dénombrable : même si les variables aléatoires valent 0 ou 1, l'espace à considérer est $\{0, 1\}^{\mathbf{N}}$ dont on a vu qu'il est en bijection avec $\mathbf{R}$. Il est admis qu'un tel ensemble peut être muni d'une tribu et d'une mesure de probabilité, sous laquelle toutes les variables aléatoires coordonnées sont indépendantes et de même loi.

Ecueils

1. La première difficulté est de comprendre que l'arithmétique sur les lois n'est pas celle des variables aléatoires : la loi de la somme de deux variable aléatoires indépendantes n'est pas la somme des lois (si,si ça s'est déjà vu) mais leur produit de convolution : si $\mathbf{P}(X = k) = p_k$ et $\mathbf{P}(Y = k) = q_k$ alors

$$r_k = \mathbf{P}(X + Y = k) = \sum_{j=0}^{k} p_j q_{k-j}. \tag{0.1}$$

Informatiquement parlant, on prendra garde à ne jamais calculer une convolution de manière directe. En effet, un rapide calcul montre que si X et Y ont des supports dans $\{0, \cdots, N\}$ un tel calcul prend à un facteur de proportionnalité près, N^2 opérations. Il existe dans Python, une opération `numpy.convolve` qui réalise le même calcul (en utilisant la transformée de Fourier rapide) en un nombre proportionnel à $N \log N$ opérations.

Exemple 1.– Dans le système de téléphonie mobile 4G, les ressources sont limitées. On montre que le nombre de ressources demandées à chaque instant, suit une loi dite de *Poisson composé* :

$$N = \sum_{k=1}^{K} k\zeta_k,$$

où $(\zeta_k,\ j = 1, \cdots, K)$ sont des variables aléatoires de Poisson indépendantes de paramètre respectifs λ_k. La question qui intéresse l'opérateur de télécoms (Orange, Free, SFR, Bouygues) est de déterminer le nombre N_0 de ressources qu'il doit déployer sur chaque site (i.e. sur chaque antenne) de sorte que

$$\mathbf{P}(N > N_0) \leq 0.001\ . \tag{0.2}$$

Comme K est relativement petit (inférieur à 10), on peut envisager de résoudre cette inégalité de façon explicite. Pour ce faire, il faut construire la loi de N informatiquement. C'est-à-dire calculer la suite des $p_k = \mathbf{P}(N = k)$ pour tout k entier. Cela pose un problème puisque k prend possiblement un nombre infini de valeurs. La preuve du lemme suivant est un grand classique pour évaluer les queues de distribution.

Lemme 1 (Principe de Herbst)**.** Pour une loi de Poisson de paramètre λ,

$$\mathbf{P}(\text{poisson}(\lambda) > \kappa\,\lambda) \leq \exp(-\lambda H(\kappa)),$$

où

$$H(\kappa) = \kappa \ln \kappa - \kappa + 1.$$

Démonstration. On note que

$$\begin{aligned}\mathbf{E}\left[e^{sX}\right] &= e^{-\lambda} \sum_{k=0}^{\infty} e^{ks} \frac{\lambda^k}{k!} \\ &= \exp(-\lambda + \lambda\, e^s).\end{aligned}$$

Pour tout $s > 0$, la fonction $x \mapsto \exp(sx)$ est strictement croissante, donc

$$\mathbf{P}(X > \kappa\,\lambda) = \mathbf{P}(e^{sX} > e^{s\kappa\,\lambda}).$$

D'après l'inégalité de Markov, on en tire

$$\begin{aligned}\mathbf{P}(X > \kappa\,\lambda) &\leq e^{-s\kappa\lambda}\ \mathbf{E}\left[e^{sX}\right] \\ &= \exp\left(-s\kappa\lambda - \lambda + \lambda\, e^s\right).\end{aligned}$$

En différentiant en s, le terme de droite est minimal pour $s = \ln \kappa$. Le minimum vaut alors

$$\exp\left(-\lambda(\kappa \ln \kappa - \kappa + 1)\right).$$

□

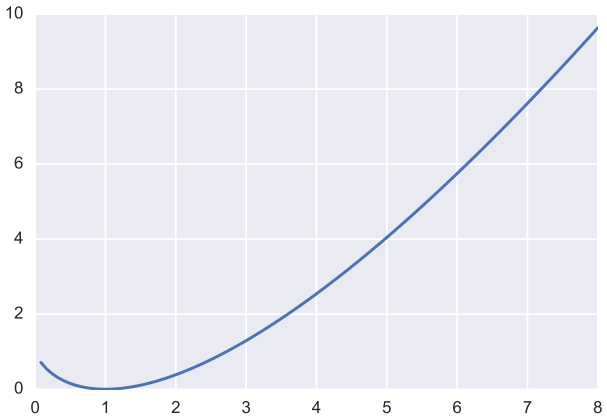

Figure 0.1– La fonction H

On cherche maintenant κ tel que

$$\mathbf{P}(\text{Poisson}(\lambda) > \kappa\lambda) \leq 0.001\,,$$

ce qui est assuré dès que

$$H(\kappa) \geq -\frac{\ln 0.001}{\lambda} = \frac{6.9}{\lambda}\cdot$$

Compte-tenu de la figure 0.1, on voit que l'on peut prendre $\kappa = 6.7$ pour tout $\lambda \geq 1$. Evidemment plus λ est grand, plus cette borne est excessive.
On construit donc les vecteurs de probabilités correspondant aux variables ζ_k pour les $(\kappa \max_k \lambda_k)$ premiers entiers. Il faut ensuite construire les vecteurs correspondants aux variables $k\zeta_k$, ce qui se fait en « espaçant »les valeurs des vecteurs initiaux :

$$\mathbf{P}(k\zeta_k = n) = \mathbf{P}(\zeta_k = n/k) \text{ si } k \text{ divise } n.$$

En Python, on peut tout faire en une seule étape en tenant compte du fait que la commande `scipy.stats.poisson.pmf(arg0,arg1)` renvoie 0 si son `arg0` est non entier et $\mathbf{P}(\text{poisson(arg1)} = \text{arg0})$ s'il est entier.

```
from scipy.stats import poisson
from numpy import *
import numpy as np
import scipy.stats as sp

def poiss(parametre,pas,taille):
    return sp.poisson.pmf(np.arange(taille)/pas,parametre)
```

Il faut faire attention à bien jongler avec les types de Python. Ainsi, `np.arange(10)/3` renverra le tableau

```
array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3])
```

tandis que la commande `np.arange(10)/2.` renverra le tableau

```
array([ 0.,0.333333,0.666667,1.,1.333333,1.666667,2.,2.333333,2.666667,3.])
```

Dans le premier cas, on aura plusieurs fois la même valeur dans le tableau final. L'appel la fonction `poiss` se fait donc comme suit

```
poiss(5,3.,20)
```

ce qui donne le résultat

```
array([ 0.00673795,  0.        ,  0.        ,  0.03368973,  0.        ,
        0.        ,  0.08422434,  0.        ,  0.        ,  0.1403739 ,
        0.        ,  0.        ,  0.17546737,  0.        ,  0.        ,
        0.17546737,  0.        ,  0.        ,  0.14622281,  0.        ])
```

Enfin on convole les vecteurs entre eux et on calcule la fonction de répartition en utilisant la fonction np.cumsum.

```
distrib1=poiss(5,1.,25)
distrib2=poiss(3,2.,25)
distrib3=poiss(1,3.,25)
distrib=np.convolve(np.convolve(distrib1,distrib2),distrib3)
np.sum(np.cumsum(distrib)<0.999)
```

On va jusqu'à 25 parce que les calculs partiels montrent que cela suffit largement pour atteindre la valeur fatidique de 0.001. Reste à trouver la première valeur du tableau où la valeur est supérieure à 0.999. Le plus simple consiste à transformer le tableau `distrib` en un tableau `True/False` à chaque fois que le contenu d'une case est inférieur à 0.999. La commande `np.sum` calcule la somme des cases à `True` donnant donc leur nombre. En décalant de 1, on obtient $N_{\max}$ satisfaisant (0.2). En l'occurrence, 32.

2. La deuxième difficulté est de réaliser que deux variables aléatoires peuvent avoir la même loi tout en ayant des valeurs très différentes. Par exemple, si X suit une loi binomiale de paramètre N et $1/2$, $Y = N - X$ a la même loi :

$$\mathbf{P}(N - X = k) = \mathbf{P}(X = N - k) = \binom{N}{N-k} 2^{-N} = \binom{N}{k} 2^{-N} = \mathbf{P}(X = k).$$

Pour autant, $X = N - X$ équivaut à $X = N/2$, qui est de probabilité nulle si N est impair. Une loi donne le pourcentage des événements qui réalisent la condition $X = k$ pour tout k mais ça ne donne aucune information sur ce que sont les événements en question. Comparez à une fonction où pour chaque valeur de l'ensemble de départ, on connaît la valeur prise par la fonction. Il est remarquable qu'à partir d'informations aussi faibles, on puisse faire des calculs.

3. Rien ne dit dans la définition des variables aléatoires que l'espace des valeurs possible doit être « uni-dimensionnel », c'est-à-dire implicitement un sous-ensemble de $\mathbf{R}$. Il importe de comprendre que l'on doit toujours voir une famille finie de variables aléatoires $X_1, \cdots, X_N$ comme une variable aléatoire dans l'espace produit $E_1 \times \ldots \times E_N$. En effet, voir $X_1, \cdots, X_N$ comme des variables aléatoires séparées c'est oublier les liens potentiels de dépendance entre elles alors que les penser comme un vecteur oblige à se poser la question de leur loi jointe. Rappelons que la loi du vecteur $(X_1, \cdots, X_N)$ ne peut se calculer à partir de celle de chacune des X_i que si toutes ces variables sont indépendantes ! Sinon, on ne peut pas le faire sans information supplémentaire.

Outils

1. On doit souvent prouver que deux variables aléatoires ont la même loi. L'une des façons les plus simples de le faire et de montrer que les deux variables sont images par une fonction déterministe de deux autres variables dont il est évident qu'elles ont la même loi.

   Exemple 2.– Considérons la situation de la marche aléatoire (on pourra se référer à l'épreuve du CCMP de 2016 en PC et PSI). Les variables aléatoires $(X_n,\ n \geq 1)$ sont indépendantes et de même loi. Il est alors manifeste que pour tout $1 \leq k < n$, pour tout $(i_1, \cdots, i_{n-k}) \in E^{n-k}$,

$$\mathbf{P}(X_{k+1} = i_1, \cdots, X_n = i_{n-k}) = \mathbf{P}(X_1 = i_1, \cdots, X_{n-k} = i_{n-k}).$$

En effet, par indépendance,

$$\mathbf{P}(X_{k+1} = i_1, \cdots, X_n = i_{n-k}) = \prod_{j=1}^{n-k} \mathbf{P}(X_{k+j} = i_j).$$

D'autre part, comme toutes les variables aléatoires ont la même

$$\mathbf{P}(X_{k+j} = i_j) = p_{i_j}$$

où $p_{i_j} = \mathbf{P}(X_1 = i_j)$. Par conséquent,

$$\mathbf{P}(X_{k+1} = i_1, \cdots, X_n = i_{n-k}) = \prod_{j=1}^{n-k} p_{i_j} = \mathbf{P}(X_1 = i_1, \cdots, X_{n-k} = i_{n-k}).$$

Ce qui signifie que le $n-k$-uple (ou la variable aléatoire à valeurs dans E^{n-k}) $(X_1, \cdots, X_{n-k})$ a même loi que le $n-k$-uple $(X_{k+1}, \cdots, X_n)$.
Maintenant supposons que $E \subset \mathbf{Z}$ et considérons l'application

$$\theta : \mathbf{Z}^{n-k} \longrightarrow \mathbf{Z}^{n-k}$$

$$(z_1, \cdots, z_{n-k}) \longmapsto (z_1, z_1 + z_2, \cdots, \sum_{j=1}^{n-k} z_j).$$

Posons aussi $S_n = \sum_{j=1}^{n} X_j$. On constate que

$$(S_{k+1} - S_k, \cdots, S_n - S_k) = \theta\Big((X_{k+1}, \cdots, X_n)\Big)$$

et

$$(S_1, \cdots, S_{n-k}) = \theta\Big((X_1, \cdots, X_{n-k})\Big).$$

On en déduit la propriété (pas si évidente à démontrer proprement) que les variables aléatoires $(S_{k+1} - S_k, \cdots, S_n - S_k)$ et $(S_1, \cdots, S_{n-k})$ ont la même loi [2].

2. Fonction génératrice. Dans le cours de CPGE, les fonctions génératrices apparaissent essentiellement comme une occasion d'appliquer les théorèmes sur les fonctions développables en série entière. Il s'agit en fait d'un sujet bien plus vaste et bien plus riche que cela. Les fonctions génératrices sont au calcul de lois ce que la transformée de Fourier est au calcul des solutions d'équation aux dérivées partielles, essentielles ! La raison en est que dans les deux cas, les produits de convolution sont remplacés par des produits ordinaires. En effet, pour deux variables aléatoires indépendantes, on a déjà vu que la loi de la somme était donnée par la convolution des lois de chacune des deux variables (voir (0.1)). Maintenant si l'on regarde ce qui se passe

   Exemple 3.– L'exemple typique d'utilisation des fonctions génératrices est celui du processus de branchement, ou arbre de Galton-Watson.

2. Il n'est pas nécessaire de montrer la bijectivité de l'application θ, comme on le voit souvent, hélas !

BIBLIOGRAPHIE