Calcul Scientifique

Cours 5: Les générateurs aléatoires

Alexis Lechervy

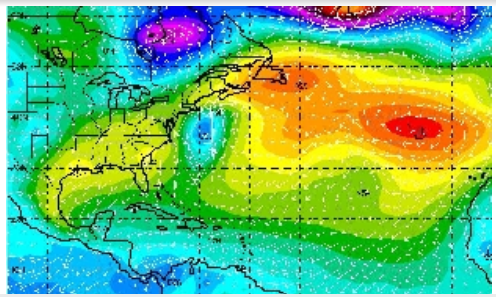
université de Caen
Basse-Normandie

Sommaire

Le hasard en informatique

L'utilité du hasard

- Le **jeu vidéo** (ex : dé, apparition d'événement, tirage au sort, génération procédural...),
- **Simulation de phénomène physique** (ex : simulation météo, simulation d'une réaction nucléaire...),
- **Simuler des comportements humain** (ex : modélisation de foule...)
- **Simplifier des calculs**, accélérer des outils d'analyse mathématique (ex : Méthode de Monte-Carlo...).
- **Prendre des décisions** face à des solutions équivalentes sans pour autant être "prévisible".
- **Sécurité informatique** (ex : détection de faille de sécurité, chiffrement de donnée,..).
- ...



Générer du "vrai" hasard

Difficulté de la notion d'aléatoire

Le hasard d'un phénomène est l'impossibilité de prévoir avec certitude l'arrivée ou non de ce phénomène. Par définition la notion de hasard est incompatible avec la notion de calcul. Il n'est donc pas possible d'écrire un algorithme qui produirait un résultat aléatoire.

Générateurs reposant sur des phénomènes imprévisibles

Une des premiers source de "hasard" que l'on a notre disposition est des phénomènes dont le résultat nous n'est pas prévisible dans l'état actuel de nos connaissances. Par exemples :

- Un lancé de dé,
- Le résultat d'une loterie,
- Le mélange d'un jeu de carte,
- Le résultat d'un pile ou face...

Ces générateurs sont pas exactement aléatoire. En effet si on connaissait l'état initiale et les conditions exactes du tirage, on pourrait à l'aide de loi physique prévoir le résultat du tirage. La connaissance de l'état initial et des conditions expérimental rend le résultat du tirage imprévisible.

Générer du "vrai" hasard

Générateur reposant sur un phénomène aléatoire

Certains phénomène physique sont aléatoire et leurs observations peut nous fournir une source de hasard.

Exemples de phénomènes aléatoires

- la radioactivité,
- le bruit électromagnétique (ex : la valeur de potentiel d'une branche analogique non connectée),
- le bruit thermique,
- le bruit mesuré par un micros,
- les ondes radios sur une bande non utilisé,
- l'observation de phénomène quantique (ex : le choix d'un photon de traverser ou non un miroir semi-réfléchissant)...

Exemple de générateur de "vrai" hasard

`https://www.random.org/`

Utilise une mesure du bruit atmosphérique pour générer du "vrai" hasard.

Générateur pseudo-aléatoire

Contexte

Le fort besoin de nombre aléatoire conduit à vouloir "avoir des algorithmes" capable de produire "des nombres aléatoires" rapidement et à moindre coût.

Constat

Certaines utilisations du hasard ne nécessite pas un générateur parfait. Des nombres suffisamment imprévisible pourrait faire l'affaire. $\Longrightarrow$ il est donc envisageable d'utiliser des algorithmes dont le résultat est suffisamment difficile à prévoir pour générer une suite de nombre dit **pseudo-aléatoire**.

Le pseudo-aléatoire

- Un algorithme déterministe qui produit une suite de nombre qui à "l'apparence" du hasard.
- Doit être facile et rapide à produire.
- En repartant de la même valeur initial (la graine) on retrouve la même séquence de nombre.

Attention

Des nombres pseudo-aléatoire ne sont pas de vrai nombre aléatoire. Il existe des domaines de l'informatique où ces nombres ne doivent pas être utilisé pour générer du hasard. On peut citer par exemple des applications de chiffrement de donnée où certaines attaques exploitent le caractère non parfaitement aléatoire des générateurs pseudo-aléatoire.

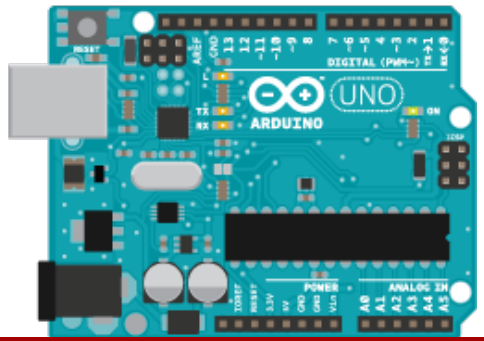
Générateur mixte

Principe

Il est possible de mixer un "vrai" générateur aléatoire et un générateur pseudo-aléatoire. On utilise de "vrai" nombre aléatoire pour initialiser la graine d'un générateur pseudo-aléatoire.

Exemples

- Initialiser le générateur aléatoire avec le nombre de seconde passé entre le 01/01/1970 et maintenant.
- Initialiser le générateur aléatoire avec le flux micro en l'absence de son particulier.
- Initialiser le générateur aléatoire avec une entrée analogique non utilisé.



Code Arduino :

```
void setup(){
  randomSeed(analogRead(0));
}
void loop(){
  long nombre = random(256);
}
```

Et en python ?

La librairie random

- La librairie random de python permet de générer des nombres pseudo-aléatoire.
- Elle utilise un générateur basé sur l'algorithme de Mersenne Twister.
- La documentation ne recommande pas son utilisation pour la cryptographie et conseil plutôt la commande os.urandom(1) qui utilise la source de hasard du système.

Exemple d'utilisation :

```
import random
import time
random.seed(time.time())
random.random()
```

La librairie numpy

- Le module random de la librairie numpy offre des générateurs de nombres pseudo-aléatoire.
- Elle utilise un générateur basé sur l'algorithme de Mersenne Twister.
- Elle n'est pas recommandé pour de la cryptographie.

Exemple d'utilisation :

```
import numpy as np
import time
np.random.seed(int(time.time()))
np.random.random()
```

Sommaire

Loi de probabilité discrètes

Définition

Une loi de probabilité est dite discrètes si les valeurs qu'elle peut générer forme un ensemble fini ou dénombrable.

Exemple : la loi uniforme discrètes

La loi uniforme discrètes consiste à tirer une valeur dans un ensemble fini de valeur où chaque valeur a autant de chance d'être tiré que les autres.
Code en numpy :
np.random.randint(1,10,size=100)

Exemple : d'application

- Tirage de valeur de dé.
- Choix entre plusieurs solutions équivalentes...

UNICAEN université de Caen Basse-Normandie

Loi avec des probabilités données

Principe

Il arrive parfois que les tirages ne soit pas uniforme et que certaines valeurs soient plus probable que d'autre. Par exemple dans le cas de dé pipé.

Problématique

Comment tiré des nombres aléatoires connaissant leurs probabilités d'apparition ?
Par exemple :
P(6)=0.5
P(4)=P(5)=0.175
P(1)=P(2)=P(3)=0.05

Stratégie possible pour des probabilités rationnelles

Réécrire toute les probabilités avec le même dénominateurs. Puis faire un tirage uniforme de valeur comprise entre 0 et le dénominateurs. Exemple :

$P(6) = 0.5 = \frac{5}{10} = \frac{500}{1000} = \frac{20}{40}$ 0..20 -> 6

$P(4) = P(5) = 0.175 = \frac{175}{1000} = \frac{7}{40}$ 21..27 -> 5 , 28..34 -> 4

$P(1) = P(2) = P(3) = 0.05 = \frac{5}{100} = \frac{50}{1000} = \frac{2}{40}$ 35..36 -> 3 , 37..38-> 2 , 39..40-> 1

Loi de Bernoulli

Principe

Correspond à un tirage de deux valeurs possibles (par exemple 0 et 1) dont l'une à une probabilité $p \in [0, 1]$ de sortie et l'autre une probabilité $1 - p$.
Si $p = 0.5$ cela correspond à un tirage de pile ou face.

Comment réaliser ce type de tirage à l'aide d'une loi uniforme ?

Même solution que le slide précédent.

Implémentation numpy possible

```
p = 0.5
np.random.binomial(1,p,size=100)
```

Loi binomiale

Principe

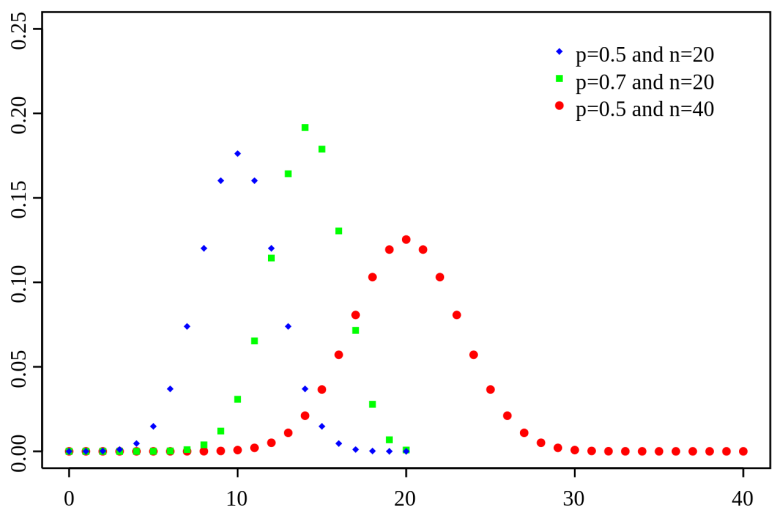
Correspond aux nombres de succès (valeur à 1) de n tirages d'une loi de Bernoulli de paramètre p.

Comment réaliser ce type de tirage à l'aide d'une loi uniforme ?

Sommer n tirages d'une loi de Bernoulli vu précédemment.

Implémentation numpy possible

p = 0.5
n = 10
np.random.binomial(n,p,size=100)



Loi de Poisson

Principe

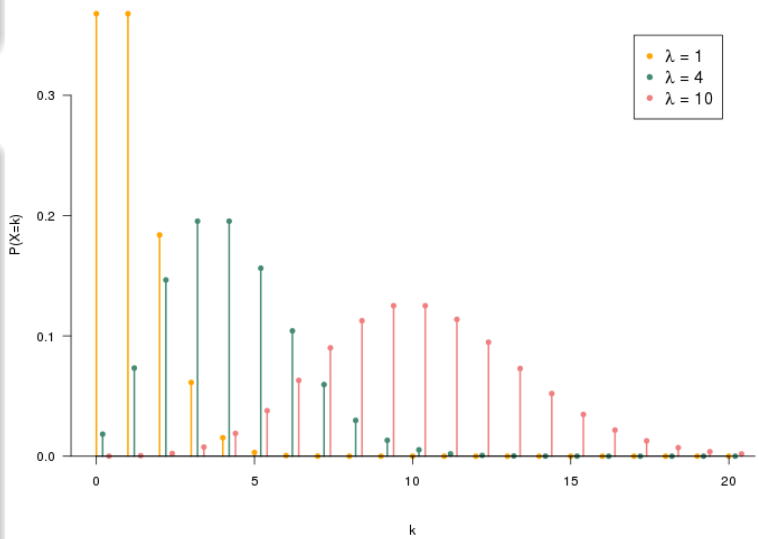
La probabilité d'un entier naturel k pour une loi de poisson de paramètre λ est $p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$.

Implémentation numpy possible

```
lambda = 10
np.random.poisson(lamba,size=100)
```

Phénomène suivant une loi de poisson

- Le nombre d'étudiant arrivé au cours d'un intervalle de temps,
- Nombre de communication en télécommunication pour un intervalle de temps,
- Le nombre de défaut ou de panne d'un produit (ex : les ampoules lumineuses),
- En finance, la modélisation du nombre de défaut de crédit...



Loi de probabilité continue

Définition

Une loi de probabilité est dite continue si les valeurs qu'elle peut générer sont dans un intervalle de valeur non dénombrable.

Exemple : la loi uniforme continue

La loi uniforme continue consiste à tirer une valeur dans un ensemble continu de valeur (par exemple l'intervalle [0,1]). Chaque valeur a la même chance de sortir.
Code en numpy :
np.random.rand(100)

Remarque

Un vrai tirage continue n'est pas possible en informatique. En effet les nombres sont codé sur un certain nombre de bit, les valeurs possibles sont donc fini et dénombrable.

Loi normale

Principe

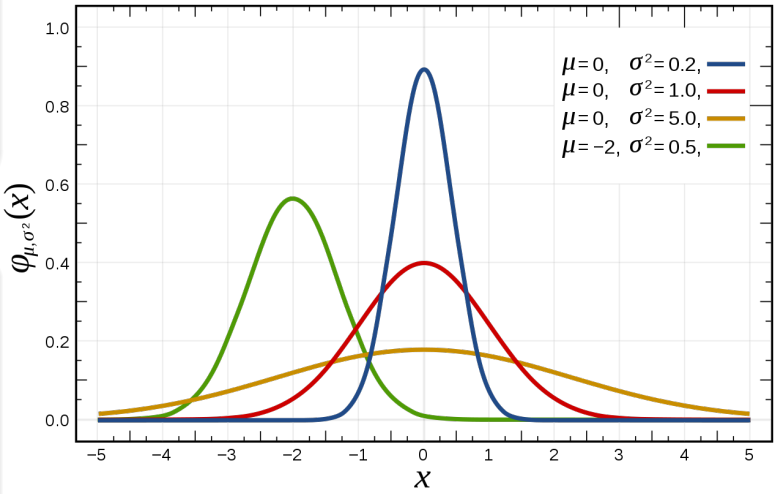
La probabilité d'un réel x pour une loi normale de paramètre μ, σ est $p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$.

Implémentation numpy possible

```
mu = 10
sigma = 5
np.random.normal(mu,sigma,size=100)
```

Phénomène suivant une loi normale

- Variation de caractéristique biologique (QI, taille, poids...),
- Les prix de matière première,
- Il est possible de construire d'autre loi de probabilité par combinaison de loi normale...



Loi normale et loi uniforme

Génération d'une loi normale à partir d'une loi uniforme

- Soit U et V deux variables généré à l'aide d'une loi uniforme sur [0,1]. Alors $x_1 = \sqrt{-2ln(U)cos(2\pi V)}$ et $x_2 = \sqrt{-2ln(U)sin(2\pi V)}$ suivent des lois normales indépendantes de paramètre $\mu = 0$ et $\sigma = 1$.
- Soit x une variable suivant une loi normale de paramètre $\mu = 0$ et $\sigma = 1$ alors $y = \frac{x}{s} + m$ suit une loi normale de paramètre $\mu = m$ et $\sigma = s$.

Synthèse

Bilan

- Deux lois de probabilité sont importantes : la loi uniforme et la loi normale centrée réduite ($\mu = 0$, $\sigma = 1$).
- Un générateur suivant une loi de probabilité donnée peut généralement être créé à partir d'une générateur suivant une loi uniforme.

Différentes fonctions pour générer des nombres selon une loi uniforme continue

np.random.rand(100,10)
np.random.random((100,10))

Différentes fonctions pour générer des nombres selon une loi normale centré réduite

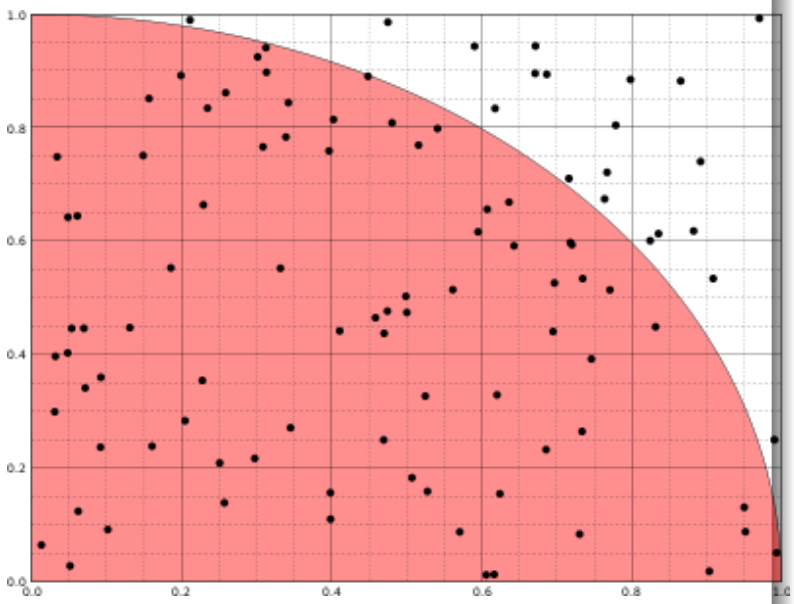
np.random.randn(100,10)
np.random.standard_normal((100,10))

Sommaire

Estimation de π par une méthode de Monte-Carlo

Principe

- On tire des points au hasard dans un carré taille 1.
- On compte le nombre de point dans le 1/4 de cercle de rayon 1 centré en 0.
- On sais que l'aire du carré est de 1 et l'aire du 1/4 de cercle est de $\frac{\pi}{4}$. Il y a donc $\frac{\pi}{4}$ chance de tomber au hasard dans le cercle par un tirage uniforme dans le carré.
- La valeur de π est donc environ égale à quatre fois la proportion de point qui est tombé dans le 1/4 de cercle.



Code

```
n=int(1e8)
x=np.random.rand(n,2)
d=np.sqrt(np.sum(x**2,axis=1))
pi = 4*np.sum(d<=1)/n
```

Simulation d'un bruit de capteur sur une image

Principe

Les capteurs d'appareil photos peuvent générer un bruit gaussien autour de la valeur attendue.

Avant / Après

Code

```
im2 = im + np.random.standard_normal(im.shape)/0.05
im2[im2>255]=255
im2[im2<0]=0
```

Sommaire

La méthode de Von Neumann

Historique

En 1946 John Von Neumann proposa une méthode simple et rapide de gérer des nombres aléatoires.

Fonctionnement

- On part d'un nombre de départ x que l'on met au carré.
- On prend les nombres du milieux de la valeur calculé (x^2) se qui nous donne le nombre aléatoire généré.
- On recommence l'algorithme avec le dernier nombre généré.

Exemple

$123456789^2 = 15241578750190521 \Longrightarrow 157875019$
$157875019^2 = 24924521624250361 \Longrightarrow 452162425$

Attention

En pratique cette méthode ne doit plus être utilisé. Elle a en effet d'assez mauvaises propriétés. Par exemple, commencez l'algorithme avec la valeur 0 retourne toujours 0.