Chap. 1 – Aléatoire et algorithmique

HAI503I – Algorithmique 4

Bruno Grenet

Université de Montpellier – Faculté des Sciences

Probabilité

Le langage des probabilités permet de modéliser une expérience probabiliste

Espace probabilisé *discret*

- *Univers* : ensemble des résultats possibles de l'expérience probabiliste
  - *Évènement primitif* : un élément de l'univers / un résultat possible
  - *Évènement* : sous-ensemble de l'univers / ensemble de résultats possibles
- Probabilités :
  - Chaque évènement primitif a une probabilité associée $\rightarrow$ somme totale $= 1$
  - Probabilité d'un évènement : somme des probabilités de ses éléments

Exemple : dé équilibré à 6 faces

- Univers : $\Omega = \{\boxdot, \boxdot, \boxdot, \boxdot, \boxdot, \boxdot\}$
- Probabilité associée à chaque élément : $\frac{1}{6}$
- $\Pr[\text{dé au moins 5}] = \Pr\left[\{\boxdot, \boxdot\}\right] = 2 \times \frac{1}{6} = \frac{1}{3}$

Intuition graphique

- Univers : partie du plan d'aire 1
- Évènement : sous-partie de l'univers $\rightarrow$ aire = probabilité de l'évènement

Variable aléatoire

Définitions

- Une **variable aléatoire** est une fonction $X : \Omega \to V$ de l'univers dans un ensemble V
- Si $V \subseteq \mathbb{R}$: variable aléatoire **réelle**
- Pour $v \in V$:
  - « $X = v$ » est l'évènement $\{\omega \in \Omega : X(\omega) = v\}$ et $\Pr[X = v] = \sum_{\omega : X(\omega) = v} \Pr[\omega]$
  - « $X \leq v$ » est l'évènement $\{\omega \in \Omega : X(\omega) \leq v\}$ et $\Pr[X = v] = \sum_{\omega : X(\omega) \leq v} \Pr[\omega]$
  - etc.

Une variable aléatoire n'est ni une variable, ni aléatoire !

Exemple : dé équilibré à 6 faces

- Nombre de point obtenus :
  - $X : \{\boxdot, \boxdot, \boxdot, \boxdot, \boxdot, \boxdot\} \to \{1, 2, 3, 4, 5, 6\}$
  - $\Pr[X = v] = \frac{1}{6}$ pour tout v et $\Pr[X \leq 4] = \frac{2}{3}$
- Parité du dé :
  - $Y : \{\boxdot, \boxdot, \boxdot, \boxdot, \boxdot, \boxdot\} \to \{0, 1\}$
  - $\Pr[Y = 0] = \Pr[Y = 1] = \frac{1}{2}$

Espérance

Définition

Soit $X : \Omega \to V$ une variable aléatoire réelle. Alors

$$\mathbb{E}[X] = \sum_{v \in V} v \times \Pr[X = v]$$

Remarques

- Intuitivement : résultat obtenu *en moyenne*
- Attention : la somme peut-être infinie $\to$ espérance non définie

Exemple : dé équilibré à 6 faces

- $\mathbb{E}[X] = \sum_{v=1}^{6} v \times \frac{1}{6} = \frac{7}{2}$
- $\mathbb{E}[Y] = 0 \times \frac{1}{2} + 1 \times \frac{1}{2} = \frac{1}{2}$

si $Y : \Omega \to \{0, 1\}$, $\mathbb{E}[Y] = \Pr[Y = 1]$

Probabilité conditionnelle

Définition

- La probabilité de E sachant F est $\Pr[E|F] = \frac{\Pr[E \wedge F]}{\Pr[F]}$
- L'espérance de X sachant F est $\mathbb{E}[X|F] = \sum_{v \in V} v \Pr[X = v|F]$

Remarque

- Graphiquement : $\Pr[E|F]$ = « proportion de F occupée par E »
- Toute probabilité est *conditionnelle* : $\Pr[E] = \Pr[E|\Omega]$ car $E \wedge \Omega = E$ et $\Pr[\Omega] = 1$

Exemple : dé équilibré à 6 faces

- $\Pr[X \geq 4|Y = 0] = \Pr[X \geq 4 \wedge Y = 0] / \Pr[Y = 0] = \frac{1}{3} / \frac{1}{2} = \frac{2}{3}$
- $\mathbb{E}[X|Y = 1] = \sum_v v \Pr[X = v|Y = 1] = 1 \times \frac{1}{3} + 2 \times 0 + 3 \times \frac{1}{3} + 4 \times 0 + 5 \times \frac{1}{3} + 6 \times 0 = 3$

Indépendance et quelques propriétés

Définition

- Deux évènements sont indépendants si $\Pr[E \wedge F] = \Pr[E]\Pr[F]$
- Deux variables aléatoires sont indépendantes si
  $\Pr[X = v \wedge Y = w] = \Pr[X = v]\Pr[Y = w]$ pour tous v, w

Propriétés

- Soit E et F deux évènements :
  - $\Pr[\neg E] = 1 - \Pr[E]$
  - $\Pr[E \vee F] \leq \Pr[E] + \Pr[F]$ Inégalité de Boole
- Soit $X, Y : \Omega \to V$ deux variables aléatoires réelles:
  - $\sum_{v \in V} \Pr[X = v] = 1$
  - $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ **Linéarité de l'espérance**
- Si $\Omega = \bigsqcup_i F_i$,
  - $\Pr[E] = \sum_i \Pr[E|F_i]\Pr[F_i]$ formules des probabilités totales
  - $\mathbb{E}[X] = \sum_i \mathbb{E}[X|F_i]\Pr[F_i]$ formules de l'espérance totale

1. Rappels (?) de probabilités discrètes

2. Bits aléatoires ou pseudo-aléatoires

3. Simulation de lois

Exemple d'algorithme probabiliste : calcul de π

CalculPi(n) :

1. $c \leftarrow 0$
2. Répéter n fois :
3. $\quad x \leftarrow$ réel *aléatoire* entre 0 et 1
4. $\quad y \leftarrow$ réel *aléatoire* entre 0 et 1
5. $\quad$ Si $x^2 + y^2 \leq 1$: $c \leftarrow c + 1$
6. Renvoyer $4c/n$

Exemple d'algorithme probabiliste : calcul de π

CalculPi(n) :

1. $c \leftarrow 0$
2. Répéter n fois :
3. $\quad x \leftarrow$ réel *aléatoire* entre 0 et 1
4. $\quad y \leftarrow$ réel *aléatoire* entre 0 et 1
5. $\quad$ Si $x^2 + y^2 \leq 1$: $c \leftarrow c + 1$
6. Renvoyer $4c/n$

Comment tirer des réels aléatoires ?

- Comment faire algorithmiquement (théorie) ?
- Comment faire sur un ordinateur (pratique) ?

Bits aléatoires

La réponse théorique

- Accès à des *bits aléatoires* :
  - RANDOM() renvoie 0 ou 1 avec probabilité $\frac{1}{2}$
  - Appels consécutifs à RANDOM() indépendants
- Construction d'*objets* aléatoires à partir des bits
  - entiers, rationnels, lettres d'un alphabet, …
  - arbres, graphes, permutations, …
- Simulation de lois
  - tirer un bit 0 avec probabilité $\frac{1}{3}$ et 1 avec probabilité $\frac{2}{3}$
  - tirer un point aléatoire dans un cercle

Bits aléatoires

La réponse théorique

- Accès à des *bits aléatoires* :
  - Random() renvoie 0 ou 1 avec probabilité $\frac{1}{2}$
  - Appels consécutifs à Random() indépendants
- Construction d'*objets* aléatoires à partir des bits
  - entiers, rationnels, lettres d'un alphabet, …
  - arbres, graphes, permutations, …
- Simulation de lois
  - tirer un bit 0 avec probabilité $\frac{1}{3}$ et 1 avec probabilité $\frac{2}{3}$
  - tirer un point aléatoire dans un cercle

Difficulté

- Comment écrire une fonction Random() ?
- *Vrai* aléa impossible ? quantique ? autres solutions ?

Le pseudo-aléa

Solution

- Générateurs *pseudo*-aléatoires
  - algorithme qui produit des bits qui *semblent* aléatoires
  - aspects théoriques et pratiques
- Générateurs de bits, mais aussi directement d'entiers, flottants, etc.
- Bibliothèques logicielles $\rightarrow$ `random` dans Python

Le pseudo-aléa

Solution

- Générateurs *pseudo*-aléatoires
  - algorithme qui produit des bits qui *semblent* aléatoires
  - aspects théoriques et pratiques
- Générateurs de bits, mais aussi directement d'entiers, flottants, etc.
- Bibliothèques logicielles → `random` dans Python

Remarques

- Algorithmes déterministes → suite fixée
- Entrée de l'algorithme : *graîne*
  - changer la graîne doit modifier *complètement* la suite
  - choix de graîne : quelque chose d'*imprévisible* ou au contraire de fixé

Exemple : générateurs congruentiels linéaires

Suite (X_n) définie par $X_{n+1} = (aX_n + c) \bmod m$

- X_0 doit être fixé : *graîne* du générateur
- a, c et m définissent le générateur
- parfois : seuls certains bits de X_n sont utilisés

Quelques choix classiques

- m premier, $c = 0$, *a primitif modulo m* (Lehmer)
- $m = 2^k$, $c = 0$, $a = 3$ ou $5 \bmod 8$
- m et c premiers entre eux, $a - 1$ divisible par les facteurs premiers de m

Exemples

- `rand` de `stdlib.h` : $m = 2^{31}$, $a = 1103515245$, $c = 12345$
- `minstd_rand` de C++11 : $m = 2^{31} - 1$, $a = 48271$, $c = 0$
- `java.util.Random` : $m = 2^{48}$, $a = 25214903917$, $c = 11$, bits 16 à 47

Conclusion sur l'aléa et le pseudo-aléa

Problématique du pseudo-aléa

- Construire des *bons* générateurs est difficile
- Limitations intrinsèques (période, etc.)

Bon, on fait quoi alors ?

- En théorie : on suppose l'accès à des bits parfaitement aléatoires
- En pratique : on utilise les générateurs des bibliothèques, en général suffisants
- Dans les deux cas : on dispose d'une source d'aléa, on simule des lois

1. Rappels (?) de probabilités discrètes

2. Bits aléatoires ou pseudo-aléatoires

3. Simulation de lois

Problématique

On sait

- tirer des bits aléatoires, ou
- tirer des entiers aléatoires, ou
- …

On veut

- tirer des bits *biaisés* : $\Pr[b = 0] \neq \Pr[b = 1]$
- choisir un élément dans un ensemble, *non uniformément* : par ex. un dé qui tombe une fois sur deux sur ⚃
- tirer un nombre réel aléatoire
- tirer un graphe aléatoire
- …

Problématique

On sait

- tirer des bits aléatoires, ou
- tirer des entiers aléatoires, ou
- …

On veut

- tirer des bits *biaisés* : $\Pr[b = 0] \neq \Pr[b = 1]$
- choisir un élément dans un ensemble, *non uniformément* : par ex. un dé qui tombe une fois sur deux sur ⚃
- tirer un nombre réel aléatoire
- tirer un graphe aléatoire
- …

But

- Construire des algorithmes pour ces tirages

Lois de probabilité

Définition informelle

La **loi d'une variable aléatoire** $X : \Omega \to V$ est la donnée de $\Pr[X = v]$ pour tout $v \in V$.

$\to$ c'est la définition de la variable aléatoire !

Quelques lois usuelles ($X : \Omega \to V$)

- Uniforme : $\Pr[X = v] = 1/|V|$ pour tout $v \in V$
- Bernoulli(p) : $V = \{0, 1\}$ et $\Pr[X = 1] = p$ — PILE OU FACE
- Binomiale(p, n) : $V = \{0, \dots, n\}$ et $\Pr[X = k] = \binom{n}{k} p^k (1-p)^k$ — n pièces : #PILE
- Géométrique(p) : $V = \mathbb{N}$ et $\Pr[X = n] = p(1-p)^{n-1}$ — #lancers avant PILE

Lois de probabilité

Définition informelle

La loi d'une variable aléatoire $X : \Omega \to V$ est la donnée de $\Pr[X = v]$ pour tout $v \in V$.

$\to$ c'est la définition de la variable aléatoire !

Quelques lois usuelles ($X : \Omega \to V$)

- Uniforme : $\Pr[X = v] = 1/|V|$ pour tout $v \in V$
- Bernoulli(p) : $V = \{0, 1\}$ et $\Pr[X = 1] = p$ — PILE OU FACE
- Binomiale(p, n) : $V = \{0, \dots, n\}$ et $\Pr[X = k] = \binom{n}{k} p^k (1-p)^k$ — n pièces : #PILE
- Géométrique(p) : $V = \mathbb{N}$ et $\Pr[X = n] = p(1-p)^{n-1}$ — #lancers avant PILE

Comment les simuler ?

- Uniforme : tirer un entier aléatoire entre 1 et V
- Bernoulli : si $p = \frac{1}{2}$, on tire un bit aléatoire… sinon ?
- Binomiale : n Bernoulli $\to O(n)$
- Géométrique : Bernoulli jusqu'à avoir $1 \to O(n)$

Tirer un bit biaisé

Idée de base

1. $x \leftarrow$ réel aléatoire entre 0 et 1
2. Si $x \leq p$: renvoyer 1
3. Sinon : renvoyer 0

Tirer un réel aléatoire entre 0 et 1

- Question à se poser : à quelle précision (nombre de bits) ?
- Si précision n : tirer n bits aléatoires $b_1, \ldots, b_n$ et renvoyer $\overline{0, b_1 \cdots b_n}^2$
- Pour un bit biaisé : tirer des bits jusqu'à savoir si $x \leq p$

En pratique

- `random` dans différent langages renvoie $x \in [0, 1]$

Exemple de loi plus complexe

Comment tirer (x, y) aléatoirement dans le disque D de centre $(0, 0)$ et de rayon 1 ?

$D = \{(x, y) : -1 \leq x, y \leq 1, x^2 + y^2 \leq 1\}$

- On sait tirer x et y dans $[0, 1]$ et on les veut dans $[-1, 1] \rightarrow 2 \cdot \text{RANDOM}() - 1$
- On veut que $x^2 + y^2 \leq 1 \rightarrow$ méthode du *rejet*

Algorithme

1. $(x, y) \leftarrow (1, 1)$
2. Tant que $x^2 + y^2 > 1$:
3. $\quad x \leftarrow 2 \cdot \text{RANDOM}() - 1$
4. $\quad y \leftarrow 2 \cdot \text{RANDOM}() - 1$
5. Renvoyer (x, y)

Propriétés

- L'algorithme renvoie bien $(x, y) \in D$
- On peut montrer qu'ils sont uniformément répartis
- Complexité : $\Pr[(x, y) \in D] = \frac{\pi}{4}$
  $\rightarrow$ espérance de $\frac{4}{\pi}$ essais : $O(1)$

Bilan des simulations de loi

Simuler des lois dans un algorithme

- Brique de base :
  - source d'aléa : bits, entiers, flottants, … (objet et loi *simple*)
  - source parfaite (théorie) ou non (pseudo-aléa)
- Simulation de loi = algorithme
  - preuve de correction $\rightarrow$ s'assurer qu'on obtient la distribution voulue
  - preuve de complexité $\rightarrow$ la simulation doit être rapide

Parfois difficile !

- Certaines lois sont difficiles à simuler : temps de calcul élevé
- Pour certains problèmes : algorithme efficace $\iff$ simulation efficace
- Pas dans ce cours…

Bilan final

Utiliser de l'aléa en algorithmique

- On suppose une source parfaite
- On utilise des lois simples — souvent
- Si loi complexe $\rightarrow$ algorithme — rarement
- Implantations :
  - bibliothèque `random` de Python (pseudo-aléa)
  - On ignore la différence avec le *vrai* aléa

Preuves en présence d'aléa

- Aléa parfait $\rightarrow$ probabilités
- Propriétés de base (probabilités conditionnelles, espérance, …)

Revoir les probabilités

- Exercices au TD1
- Poly « Probabilités discrètes » sur Moodle