1 Complexité de Kolmogorov

On s'intéresse ici à la complexité intrinsèque de la description d'un objet (dans la plupart des cas ici un nombre). Intuitivement, la notion de complexité de la description algorithmique que l'on regarde est la longueur du plus petit programme qui décrit l'objet.

1.1 Modèle et exemples

On prend comme modèle de calcul une machine de Turing universelle, qui prend comme entrée un programme (par exemple machine de Turing plus une entrée). Elle lit ce programme sur un ruban d'entrée, écrit le mot à générer sur le ruban de sortie et dispose d'un ruban de travail. Tous les alphabets sont binaires égaux à $\{0,1\}$.

Cette machine peut donc se voir comme une fonction $f : \{0,1\}^* \to \{0,1\}^* \cup \{0,1\}^\omega$. Les mots u tels que $f(u) \in \{0,1\}^\omega$ sont les mots sur laquelle la machine ne s'arrête pas.

On peut supposer sans perte de généralité que cette machine universelle est déterministe et que la machine lit le ruban d'entrée de la gauche vers la droite uniquement (quitte à recopier le ruban d'entrée sur le ruban de travail). On peut donc considérer uniquement un langage préfixe des mots sur lesquels la machine s'arrête (si p_1 s'arrête et renvoie x, alors pour tout mot p_2, p_1p_2 s'arrête et renvoie x. On considère juste p_1 alors et aucun p_1p_2 avec $p_2 \neq \varepsilon$).

Soit $\mathcal{U}$ une machine de Turing universelle et $x \in \{0,1\}^*$. On note $\ell(x)$ la longueur de x et $\mathcal{U}(p)$ la sortie de $\mathcal{U}$ si le programme p est donné en entrée.

Définition 1. *La complexité de Kolmogorov $K_\mathcal{U}$ d'une suite x associée à la machine $\mathcal{U}$ est*

$$K_\mathcal{U}(x) = \min\{\ell(p) \mid \mathcal{U}(p) = x\}.$$

La complexité de Kolmogorov conditionnellement à la longueur de x est

$$K_\mathcal{U}(x \mid \ell(x)) = \min\{\ell(p) \mid \mathcal{U}(p, \ell(x)) = x\}.$$

Notons que la notion de complexité de Kolmogorov ne dépend qu'à une constante additive près du modèle de calcul.

Proposition 1. *Soit $\mathcal{A}$ un calculateur permettant de calculer x. Il existe une constante $c_\mathcal{A}$ telle que*

$$K_\mathcal{U}(x) \leq K_\mathcal{A} + c_\mathcal{A}.$$

Démonstration. Soit $p_\mathcal{A}$ un programme tel que $\mathcal{A}(p_\mathcal{A}) = x$ et $s_\mathcal{A}$ un programme qui simule $\mathcal{A}$ à partir de $\mathcal{U}$. Alors $p = s_\mathcal{A}p_\mathcal{A}$ est un programme de $\mathcal{U}$ qui calcule x. On a donc en posant $c_\mathcal{A} = \ell(s_\mathcal{A})$

$$K_\mathcal{U}(x) = \min_{p \mid \mathcal{U}(p)=x} \ell(p) \leq \min_{p_\mathcal{A} \mid \mathcal{A}(p_\mathcal{A})=x} \ell(p) + \ell(s_\mathcal{A}) = K_\mathcal{A}(x) + c_\mathcal{A}.$$

□

Dans la suite, nous allons écrire toutes les complexités à une constante près, donc on ne précisera plus le modèle de calcul $\mathcal{U}$ et on écrire K au lieu de $K_\mathcal{U}$.

Exemple (Lien entre complexité de Kolmogorov et complexité conditionnée à la longueur).

- $K(x \mid \ell(x)) \leq \ell(x) + c$.
  En effet, le programme « `écrire la suite de ce nombre de caractères` $x_1, \ldots, x_\ell$ » convient.
- $K(x) \leq K(x \mid \ell(x)) + 2\log \ell(x) + c$.
  Une méthode simple est d'écrire la longueur de x en binaire en doublant chaque chiffre. Plus précisément, si $\ell(x) = n$, on encode n en doublant les chiffres et en ajoutant 01 à la fin (pour s'assurer la reconnaissance du nombre et que l'on reste dans un ensemble préfixe). Par exemple 5 est encodé par `11001101`. Ensuite on écrit le programme qui calcule x connaissant $\ell(x)$.

Théorème 1. *Pour tout $k \in \mathbb{N}$, $|\{x \in \{0,1\}^* \mid K(x) < k\}| \leq 2^k$.*

Démonstration. C'est un argument de comptage : il y a $\sum_{i=0}^{k-1} 2^i = 2^k - 1$ programmes de longueur au plus $k-1$. □

Exemple (Calculs usuels).

1. $K(0 \ldots 0 \mid n) = c$
   Le programme « `Écrire ce nombre de '0'` » convient.
2. $K(\pi_1 \ldots \pi_n \mid n) = c$
   Le programme « `Écrire ce nombre de chiffres du développement binaire de` π » convient.
3. $K(n) \leq 2\log n + c$
   On utilise la même méthode que dans l'exemple précédent.
4. Si x est un mot ayant k occurrences de 1,

$$K(x \mid n) \leq c + \log n + \log \binom{n}{k} \leq c' + \log n + nH(\frac{k}{n}) - \frac{1}{2}\log n.$$

   On encore k en un mot de longueur $\log n$ (n est connu donc on n'a pas besoin de doubler les chiffres), puis x par son numéro parmi toutes les suites de n bits ayant k occurrences de 1. Il y en a $\binom{n}{k}$, et on utilise la formule déjà vue :

$$\binom{n}{k} \leq \sqrt{\frac{n}{\pi k(n-k)}} 2^{nH(\frac{k}{n})}.$$

   On a donc

$$K(x_1 \ldots x_n \mid n) \leq nH(\frac{\sum_{i=1}^n x_i}{n}) + \frac{1}{2}\log n + c.$$

1.2 Complexité de Kolmogorov et entropie

Avec nos hypothèses, l'ensemble des programmes p qui s'arrêtent forment un ensemble préfixe. En appliquant l'inégalité de Kraft, on a donc

$$\sum_{\{p \mid p \text{ s'arrête}\}} 2^{-\ell(p)} \leq 1.$$

Théorème 2. *Soit $\{X_i\}_{i\in\mathbb{N}}$ une suite de variables aléatoires i.i.d distribuées selon une loi de Bernoulli de paramètre θ. On note $x^n = x_1 \dots x_n$ et $p(x^n) = p(x_1)\dots p(x_n)$. Alors il existe une constante c telle que pour tout $n \in \mathbb{N}$,*

$$H(X) \leq \frac{1}{n} \sum_{x^n\in\{0,1\}^n} f(x^n)K(x^n \mid n) \leq H(X) + \frac{1}{n}\log n + \frac{c}{n},$$

et donc

$$\lim_{n\to\infty} \mathbf{E}[\frac{1}{n}K(X^n \mid n)] = H(X).$$

Démonstration. À chaque mot $x^n \in \{0,1\}^n$, on peut associer un programme le plus court p parmi ceux qui satisfont $\mathcal{U}(p,n) = x^n$. Comme les longueurs des programmes satisfont l'inégalité de Kraft, alors on a bien, en prenant pour codage de x^n un tel programme $\mathbf{E}[K(X^n \mid n)] = \sum_{x^n\in\{0,1\}^n} K(x^n \mid n) \geq H(X_1 \dots X_n) = nH(X)$, où $X \sim \mathcal{B}\text{er}(\theta)$.

Pour l'autre inégalité, on a déjà montré que $K(x_1 \dots x_n \mid n) \leq nH(\frac{1}{n}\sum_{i=1}^n x_i) + \frac{1}{2}\log n + c$. Calcule l'espérance de $K(X_1 \dots X_n \mid n)$ et on obtient

$$\begin{aligned}
\mathbf{E}[K(X_1 \dots X_n \mid n)] &\leq n\mathbf{E}[H(\frac{1}{n}\sum_{i=1}^n X_i)] + \frac{1}{2}\log n + c \\
&\leq nH(\frac{1}{n}\sum_{i=1}^n \mathbf{E}[X_i]) + \frac{1}{2}\log n + c \\
&\leq nH(\theta) + \frac{1}{2}\log n + c,
\end{aligned}$$

où la deuxième égalité est obtenue par l'inégalité de Jensen car H est concave. □

On peut retirer le conditionnement par la longueur de x : $K(x^n) \leq K(x^n \mid n) + 2\log n + c$, et donc

$$H(X) \leq \frac{1}{n} \sum_{x^n\in\{0,1\}^n} f(x^n)K(x^n) \leq H(X) + \frac{3}{n}\log n + \frac{c}{n}.$$

On a donc toujours

$$\lim_{n\to\infty} \mathbf{E}[\frac{1}{n}K(X^n)] = H(X).$$

1.3 Complexité de Kolmogorov des entiers

Définition 2. *Soit $\mathcal{U}$ une machine de Turing universelle La complexité de Kolmogorov d'un entier $n \in \mathbb{N}$ associée à la machine $\mathcal{U}$ le plus petit programme de $\mathcal{U}$ permettant de calculer n :*

$$K_{\mathcal{U}}(n) = \min\{\ell(p) \mid \mathcal{U}(p) = n\}.$$

De la même manière que précédemment, la complexité associée à un modèle de calcul ne diffère que par une constante additive de celle associée à un autre modèle. On notera donc $K(n)$ au lieu de $K_{\mathcal{U}}(n)$.

D'après ce qui précède, on a aussi une borne maximale sur la complexité d'un entier : $K(n) \leq 2\log n + c$.

Lemme 1. *Il existe une suite infinie d'entiers n tels que $K(n) > \log n$.*

Démonstration. D'une part, l'inégalité de Kraft donne $\sum_{n\in\mathbb{N}} 2^{-K(n)} \leq 1$. D'autre part, on a pour tout $n_0 \in \mathbb{N}$, $\sum_{n\geq n_0} 2^{-\log n} = \sum_{n\geq n_0} \frac{1}{n} = \infty$. Si on avait $K(n) \leq \log n$ pour tout $n \geq n_0$ (c'est-à-dire un nombre fini d'entiers tels que $K(n) \leq \log n$ alors on aurait

$$1 \geq \sum_{n=n_0}^{\infty} 2^{-K(n)} \geq \sum_{n\geq n_0} 2^{-\log n} = \infty,$$

ce qui est absurde. □

1.3.1 Suites incompressibles

Certains nombres très grands peuvent être générés par des programmes très courts. C'est le cas par exemple de

$$2^{2^{2^{\cdots}}} \qquad \text{ou} \qquad (100!)!.$$

On va montrer ici que la plupart des suites n'ont pas de description simple.

Théorème 3. *Soit $X_1 \dots X_n \sim \mathcal{B}\text{er}(1/2)$ une suite de v.a. i.i.d. Alors $\mathbf{P}(K(X_1 \dots X_n \mid n) < n-k) < 2^{-k}$.*

Démonstration.

$$\begin{aligned}
\mathbf{P}(K(X_1 \dots X_n \mid n) < n-k) &= \sum_{\{x^n \mid K(x^n|n)<n-k\}} p(x^n) \\
&= \sum_{\{x^n \mid K(x^n|n)<n-k\}} 2^{-n} \\
&= |\{x^n \mid K(x^n|n) < n-k\}|2^{-n} \\
&< 2^{n-k}2^{-n} = 2^{-k}.
\end{aligned}$$

□

Définition 3.
- *Une suite $x_1 \dots x_n$ est dite algorithmiquement aléatoire si $K(x_1 \dots x_n|n) \geq n$.*
- *Une suis infinie est incompressible si* $\lim_{n\to\infty} \frac{K(x_1 \dots x_n \mid n)}{n} = 1$.

Théorème 4. *Si une suite est incompressible, alors* $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{2}$.

Démonstration. Soit $\theta_n = \sum_{i=1}^{n} x_i$ la proportion de 1 dans x^n. Alors il existe c tel que

$$\frac{K(x^n \mid n)}{n} \leq H(\theta_n) + \frac{1}{2}\frac{\log n}{n} + \frac{c}{n}.$$

Si la suite est incompressible, alors pour tout $\epsilon > 0$, il existe $n_0 \in \mathbb{N}$ tel que pour tout $n \geq n_0$,

$$1 - \epsilon \leq \frac{K(x^n \mid n)}{n} \leq H(\theta_n) + \frac{1}{2}\frac{\log n}{n} + \frac{c}{n}.$$

Mais alors $H(\theta_n) \geq 1 - \epsilon - \frac{1}{2}\frac{\log n + c}{n}$.

Donc il existe δ_n tel que $\theta_n \in [1/2 - \delta_n, 1/2 + \delta_n]$. Or quand $n \to \infty$, $\frac{\log n + c}{n} \to 0$, et donc $\delta_n \to 0$. Donc $\theta_n \to 1/2$.

□

Réciproquement, on a le théorème suivant :

Théorème 5. *Soit $X_1, \dots, X_n$ i.i.d $\sim \mathcal{B}\text{er}(\theta)$. alors* $\lim_{n\to\infty} \frac{1}{n} K(X_1 \dots X_n \mid n) = H(\theta)$ *p.s.*

Démonstration. On pose $\bar{X}_n = \frac{1}{n}\sum_{i=1}^n X_i$. On a toujours

$$nH(\bar{X}_n) \leq K(X_1 \dots X_n \mid n) \leq nH(\bar{X}_n) + \frac{1}{2}\log n + c,$$

et $\bar{X}_n \to \theta$ p.s. Donc $\mathbf{P}(|\frac{1}{n}K(X_1 \dots X_n \mid n) - H(\theta)| \geq \epsilon) \to 0$ quand $n \to \infty$. □

1.3.2 Indécidabilité de la complexité de Kolmogorov

Supposons cette complexité calculable (pour le modèle de calcul $\mathcal{U}$), et on dispose donc d'un programme K qui calcule $K(n)$ pour tout n. Soit $k = \ell(K)$ la longueur de ce programme et considérons le programme suivant :

```
n:=1;
tant que K(n) < 100+k faire
     n:=n+1;
fin tant que;
écrire n
```

Ce programme écrit le plus petit programme qui fait moins de $100+k$ caractères. Mais clairement, il fait moins de $100+k$ caractères (le k est pour ajouter la fonction K), donc ce programme peut se décrire en moins de 100 caractères. C'est absurde donc K n'existe pas.

1.4 Probabilité universelle

Définition 4. *La probabilité universelle d'une suite x associée à une machine universelle $\mathcal{U}$ est*

$$\mathbf{P}_{\mathcal{U}}(x) = \sum_{\{p \mid \mathcal{U}(p)=x\}} 2^{-\ell(p)} = \mathbf{P}(\mathcal{U}(p) = x)$$

C'est la probabilité d'obtenir la sortie x pour une suite tapée au hasard.

On aimerait pouvoir prouver (ou rendre formelles) des assertions du style « les suites les plus simples sont les plus probables » ou « l'explication la plus simple d'un phénomène est la plus probable »

Ces deux assertions peuvent être traduites par l'approximation $\mathcal{P}_{\mathcal{U}} \approx 2^{-K(x)}$.

Ici encore, pour un calculateur $\mathcal{A}$ qui peut calculer x, il existe une constante $c_{\mathcal{A}}$ telle que $\mathbf{P}_{\mathcal{U}}(x) \geq c_{\mathcal{A}}\mathbf{P}_{\mathcal{A}}(x)$, et la probabilité universelle est la même à une constante multiplicative près.

On veux donc comparer les quantités

$$K(x) = \min_{\{p \mid \mathcal{U}(p)=x\}} \ell(p) \qquad \text{et} \qquad \mathbf{P}_{\mathcal{U}}(x) = \sum_{\{p \mid \mathcal{U}(p)=x\}} 2^{-\ell(p)}.$$

Théorème 6. *Il existe une constante c telle que pour tout $x \in \{0,1\}^*$,*

$$2^{-K(x)} \leq \mathbf{P}_{\mathcal{U}}(x) \leq c2^{-K(x)}.$$

Démonstration. Pour $x \in \{0,1\}^*$, soit p^* un programme le plus court calculant x. On a donc

$$\mathbf{P}_{\mathcal{U}}(x) = \sum_{p \mid \mathcal{U}(p)=x} 2^{-\ell(p)} \geq 2^{-\ell(p^*)} = 2^{-K(x)}$$

et la première inégalité est montrée.

On veut maintenant montrer que $K(x) \leq \log \frac{1}{p_{\mathcal{U}}(x)} + c'$. Pour ce faire, on va construire un arbre pour lequel les programmes courts ont une faible profondeur.

Construction de l'arbre : On simule tout d'abord tous les programmes en parallèle de sorte que si un programme termine en temps fini, il termine aussi en temps fini dans cette exécution. Plus précisément, on a $\epsilon, 0, 1, 00, 01 \ldots$ une énumération de tous les programmes. On les simule en parallèle comme suit :

1. on fait un pas de calcul du programme ϵ ;
2. on fait deux pas de calcul des programmes ϵ et 0 ;
3. on fait trois pas de calcul des programmes ϵ, 0 et 1 ;

....

À partir de cette exécution en parallèle, on peut construire la liste ordonnée des programmes qui s'arrêtent dans leur ordre de terminaison.

Pour le k-ième programme qui termine, on a le triplet (p_k, x_k, n_k), où p_k est le programme, x_k sa sortie et

$$n_k = \lceil \log \frac{1}{\hat{\mathbf{P}}_{\mathcal{U}}(x_k)} \rceil = -\lfloor \log \hat{\mathbf{P}}_{\mathcal{U}}(x_k) \rfloor,$$

où

$$\hat{\mathbf{P}}_{\mathcal{U}}(x_k) = \sum_{\{(p_i,x_i,n_i) \mid x_i=x_k, i \leq k\}} 2^{-\ell(p_i)}.$$

Clairement $\hat{\mathbf{P}}_{\mathcal{U}}(x_k) \to \mathbf{P}_{\mathcal{U}}(x_k)$ pour la sous-suite des $x_k = x$ pour $x \in \{0,1\}^*$ fixé.

Pour des raison techniques qui s'expliqueront dans la suite, on élimine de cette suite les éléments $i > k$ tels que $(x_i, n_i) = (x_k, n_k)$.

On construit maintenant l'arbre à mesure que les programmes terminent. Si le programme p_k termine (et qu'il reste dans la liste), alors on assigne au triplet (p_k, x_k, n_k) le premier nœud disponible au niveau $n_k + 1$.

Exemple (construction de l'arbre). On donne les triplets dans l'ordre de terminaison des programmes.

- $(p_1, x_1, n_1) = (10111, 1110, 5)$ (on a $n_1 = 5$ car $\hat{\mathbf{P}}_{\mathcal{U}}(x_1) = 2^{-\ell(p_1)} = 2^{-5}$) ;
- $(p_2, x_2, n_2) = (11, 10, 2)$ (on a $n_2 = 2$ car $\hat{\mathbf{P}}_{\mathcal{U}}(x_2) = 2^{-\ell(p_2)} = 2^{-2}$) ;
- $(p_3, x_3, n_3) = (0, 1110, 1)$ (on a $n_3 = 1$ car $\hat{\mathbf{P}}_{\mathcal{U}}(x_3) = 2^{-\ell(p_1)} + 2^{-\ell(p_3)} \geq 2^{-1}$) ;
- $(p_4, x_4, n_4) = (1010, 1111, 4)$ (on a $n_4 = 4$ car $\hat{\mathbf{P}}_{\mathcal{U}}(x_4) = 2^{-\ell(p_4)} = 2^{-4}$) ;
- $(p_5, x_5, n_5) = (101101, 1110, 1)$ (on a $n_5 = 1$ car $\hat{\mathbf{P}}_{\mathcal{U}}(x_5) = 2^{-\ell(p_1)} + 2^{-\ell(p_3)} + 2^{-\ell(p_5)} \geq 2^{-1}$). Cet élément est supprimé de la liste ;
- $(p_6, x_6, n_6) = (100, 1, 3)$ (on a $n_6 = 3$ car $\hat{\mathbf{P}}_{\mathcal{U}}(x_6) = 2^{-\ell(p_6)} = 2^{-3}$)...

La figure 1 donne l'arbre ainsi construit.

Il reste à vérifier que l'on peut bien construire cet arbre. Cela est possible (par l'inégalité de Kraft) si

$$\sum_{k=1}^{\infty} 2^{-(n_k+1)} \leq 1.$$

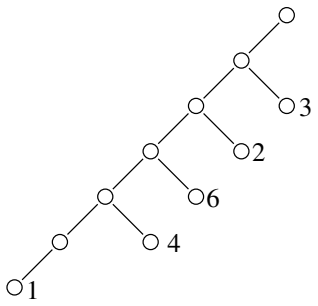

FIGURE 1 – Construction de l'arbre.

Or on peut écrire $\sum_{k=1}^{\infty} 2^{-(n_k+1)} = \sum_{x\in\{0,1\}^*} \sum_{\{k \mid x_k=x\}} 2^{-(n_k+1)}$ et

$$\begin{aligned}
\sum_{\{k \mid x_k=x\}} 2^{-(n_k+1)} &= 2^{-1} \sum_{\{k \mid x_k=x\}} 2^{-n_k} \\
&\leq 2^{-1}(2^{\lfloor \log \mathbf{P}_{\mathcal{U}}(x_k)\rfloor} + 2^{\lfloor \log \mathbf{P}_{\mathcal{U}}(x_k)\rfloor -1} + \dots \\
&\leq 2^{-1}2^{\lfloor \log \mathbf{P}_{\mathcal{U}}(x_k)\rfloor}(1 + 1/2 + 1/4 + \dots) \\
&\leq 2^{-1}2^{\lfloor \log \mathbf{P}_{\mathcal{U}}(x_k)\rfloor} \times 2 \\
&\leq 2^{\lfloor \log \mathbf{P}_{\mathcal{U}}(x_k)\rfloor}.
\end{aligned}$$

C'est pour la première inégalité que l'on utilise le fait qu'on a éliminé les triplets avec des n_k redondants.

Finalement, on a bien

$$\sum_{k=1}^{\infty} 2^{-(n_k+1)} = \sum_{x\in\{0,1\}^*} \sum_{\{k \mid x_k=x\}} 2^{-(n_k+1)} \leq \sum_{x\in\{0,1\}^*} 2^{\lfloor \log \mathbf{P}_{\mathcal{U}}(x_k)\rfloor} \leq \sum_{x\in\{0,1\}^*} \mathbf{P}_{\mathcal{U}}(x) \leq 1.$$

Avec la construction ci-dessus, on peut identifier x avec le chemin $\tilde{p}$ de plus petite profondeur qui conduit à un triplet avec $x_k = x$. Il existe un programme P qui génère cet arbre (la procédure vient d'être décrite). On a donc un programme qui prend en entrée $\tilde{p}P$. La machine construit l'arbre jusqu'à avoir construit le chemin $\tilde{p}$, qui un programme p qui s'exécute pour calculer x.

Or $\ell(\tilde{p}) \leq \lceil \log \frac{1}{\mathbf{P}_{\mathcal{U}}(x)} \rceil + 1$ et donc $K(x) \leq \log \frac{1}{\mathbf{P}_{\mathcal{U}}(x)} + c$. □