Le rôle de l'aléatoire en informatique et en communication : l'exemple des codes correcteurs d'erreurs

Le but de cette section est de présenter (sans entrer dans trop de détails techniques) une partie du travail important accompli dans les années 50 par Claude Shannon sur le sujet de la communication en présence de bruit. Elle peut être vue comme une extension au chapitre sur la représentation de l'information.

> **Avertissement**
>
> Même si elle se veut introductive, cette section est globalement plus difficile à appréhender que les autres, notamment du fait qu'elle nécessite des notions de base de probabilités, que les élèves n'ont pas encore acquises en 1M ou 2M. Elle pourrait être utilisée par contre en option complémentaire.

Mise en situation : Bob est perdu au milieu de la forêt

Supposons que deux personnes, disons Alice et Bob, désirent communiquer, et imaginons le scénario où Bob est perdu au milieu d'une forêt ; Alice, qui connaît la position GPS de Bob et qui a aussi la carte de la forêt devant elle, voit dans quelle direction (N, S, E, W) Bob devrait aller pour en sortir au plus vite.

Supposons maintenant que pour communiquer avec Bob, Alice ne puisse envoyer que des suites de 0 et de 1. On voit ici que 2 bits suffisent à Alice pour transmettre l'information requise, en utilisant le dictionnaire suivant (sur lequel elle se sera mise d'accord au préalable avec Bob) :

direction	N	S	E	W
mot de code	11	10	01	00

Disons qu'elle lui transmette justement le message d'aller au nord en envoyant le mot de code 11. Il se peut qu'à cause du "bruit sur la ligne", un des deux bits transmis ne parvienne pas correctement à Bob (on dit alors qu'il est "effacé") : par exemple, si le deuxième bit est effacé, Bob recevra la séquence "1?" (le point d'interrogation symbolisant l'effacement) et ne saura donc pas s'il faut aller au nord ou au sud pour sortir rapidement de la forêt.

Pour remédier à ce problème, Alice pourrait utiliser plutôt le dictionnaire suivant, où chaque bit est doublé, ce qui ajoute de la *redondance* au mot de code transmis :

direction	N	S	E	W
mot de code	1111	1100	0011	0000

Ainsi, si un bit est effacé au cours de la transmission, disons le 3e, Bob reçoit le mot de code “11?1”, et par identification avec le tableau ci-dessus, il voit bien qu’Alice lui dit d’aller au nord ; l’erreur de transmission a donc pu être corrigée.

Fondamentalement, ce qui se passe ici est que les mots de codes potentiellement envoyés par Alice sont plus *distants* les uns des autres que dans le premier cas : en effet, il y a toujours au moins deux bits de différence entre chaque paire de mots de code (entre 1111 et 1100, par exemple), ce qui permet de corriger un effacement.

Toutefois, avec un tel schéma de communication, Alice doit utiliser quatre bits pour ne transmettre que deux bits réels d’information. On dit que le *taux* de cette transmission vaut R = 2/4 = 1/2.

Pour atteindre le même but de corriger un effacement, il aurait été possible d’économiser un bit en utilisant le dictionnaire suivant :

direction	N	S	E	W
mot de code	11**0**	10**1**	01**1**	00**0**

Ici, le troisième bit est appelé *bit de parité* : tout mot de code s’écrit comme une séquence de trois bits $x_1x_2b_1$, où b_1 vaut 1 si et seulement si la somme des bits x_1 et x_2 est impaire. On écrit aussi que $b_1 = x_1 \oplus x_2$, où le symbole $\oplus$ représente l’addition modulo 2.

Vous pouvez vérifier qu’avec ce nouveau dictionnaire, si Bob reçoit par exemple le message 1?0, alors il saura qu’il faut aller au nord ; de manière générale, il saura dans quelle direction aller, quel que soit le message envoyé par Alice et quelle que soit la position du bit effacé (possiblement celle du bit de parité). La raison est qu’avec ce schéma, il y a toujours au moins deux bits de différence entre chaque paire de mots de code, comme avec le schéma précédent. Mais le taux de transmission vaut maintenant R=2/3 : ce nouveau système d’encodage est clairement plus efficace !

Indiquer une direction plus précise

Indiquer une direction plus précise

Supposons maintenant qu'Alice désire envoyer non pas une direction parmi 4 possibilités différentes (N, S, E, W), mais une parmi les 16 suivantes :

N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW

Pour ceci, elle pourrait utiliser 4 bits par message avec le dictionnaire suivant :

direction	N	NNE	NE	ENE	E	...
mot de code	1111	1110	1101	1100	1011	...

C'est certainement la façon la plus économe d'encoder 16 messages différents, mais à nouveau, en cas d'effacement de l'un ou de l'autre des bits envoyés, Bob ne saura pas dans quelle direction marcher (si par exemple il reçoit le message 11?1, il ne saura pas s'il doit aller au nord ou au nord-est).

Pour pallier à ce problème, Alice et Bob peuvent à nouveau utiliser le même principe que ci-dessus et rajouter un bit de parité aux messages transmis :

direction	N	NNE	NE	ENE	E	...
mot de code	1111**0**	1110**1**	1101**1**	1100**0**	1011**1**	...

Chaque mot de code s'écrit donc maintenant $x_1x_2x_3x_4b_1$, avec $b_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

Comme avant, si un bit est effacé, Bob pourra retrouver la direction à prendre (par exemple, s'il reçoit le message 11?11, il saura qu'il faut aller au nord-est). Et ici, le taux de transmission vaut R=4/5.

Mais vu que plus de bits sont transmis, il y a sans doute aussi en pratique une plus grande probabilité que plus d'un bit soit effacé lors de la transmission. Que faire si par exemple deux bits sont effacés ? Dans ce cas, le bit de parité unique ajouté ci-dessus ne permet plus un décodage approprié (si par exemple Bob reçoit le message 11??1). Dans ce cas, il va donc falloir ajouter plus de bits de parité pour protéger le message.

Il se trouve que la bonne chose à faire dans ce cas (découverte par Richard Hamming en 1950) est de rajouter non pas deux, mais trois bits de parité au message d'origine : ainsi, les mots de code envoyés seront de la forme $x_1x_2x_3x_4b_1b_2b_3$, avec

$$b_1 = x_1 \oplus x_2 \oplus x_3, \quad b_2 = x_1 \oplus x_2 \oplus x_4 \quad \text{et} \quad b_3 = x_1 \oplus x_3 \oplus x_4$$

ce qui donne le dictionnaire suivant :

direction	N	NNE	NE	ENE	E	...
mot de code	1111**111**	1110**100**	1101**110**	1100**001**	1011**001**	...

Dans ce dernier cas, on peut vérifier qu'il y a toujours au moins trois bits de différence entre chaque paire de mots de code, ce qui permet de corriger deux effacements. A noter que le taux de transmission vaut maintenant R=4/7. Il n'y a pas de miracle : pour se protéger de plus d'erreurs éventuelles, il a fallu ajouter plus de bits de parité et donc réduire le taux de transmission.

Généralisation

Supposons maintenant qu'Alice désire envoyer un message parmi $M = 2^k$ messages possibles, pour un nombre entier positif k donné. Dans ce cas, k bits suffisent a priori pour cette transmission, mais supposons encore qu'en moyenne, une proportion ε des bits envoyés par Alice soient effacés lors de la transmission, et que ces effacements surviennent de façon aléatoire. Ceci veut dire concrètement que si Alice envoie n bits, alors seuls $n\,(1-\varepsilon)$ d'entre eux parviennent à Bob ; de plus, il n'y a aucun moyen de savoir à l'avance lesquels seront effacés.

> **Remarque**
>
> La lettre grecque ε utilisée ici ne fait pas référence au ε infinitésimal en mathématiques ! Ici, la lettre ε veut juste faire penser au "e" de "erreur" ou "effacement".

Comment s'assurer dans cette situation que Bob puisse toujours retrouver le message envoyé par Alice ? Clairement, si Alice envoie le message "tel quel" en l'encodant sur k bits, ça ne fonctionnera pas. Il faut donc qu'Alice ajoute ici de la redondance pour protéger son message des effacements qui surviennent, en envoyant plus de k bits.

La question est maintenant de savoir quelle redondance ajouter pour protéger *efficacement* le message transmis. Au vu de la solution proposée ci-dessus par Hamming pour le cas k=4, on est en droit de se dire que le problème est a priori complexe à résoudre pour une valeur quelconque de k... Mais c'est là qu'intervient en 1948 l'idée de génie de Claude Shannon d'utiliser le hasard pour s'attaquer à ce problème !

Avant d'essayer de résoudre celui-ci, voyons d'abord ce qu'on peut espérer faire dans le meilleur des cas : si par exemple un sympathique génie révélait *à l'avance* à Alice non seulement la proportion ε des bits effacés, mais aussi *quels* bits sont effacés lors de la transmission, alors celle-ci pourrait faire la chose suivante : envoyer n bits, en choisissant n tel que $k = n\,(1-\varepsilon)$, et utiliser justement les k bits non-effacés pour encoder le message. Ainsi, $M = 2^k$ messages différents pourraient être transmis sans problème à Bob, en utilisant en tout n bits. Le taux de transmission vaudrait donc dans ce cas :

$$R = \frac{k}{n} = \frac{n\,(1-\varepsilon)}{n} = 1 - \varepsilon$$

En bref, Alice aurait là un moyen de transmission très efficace: en ajoutant un peu de redondance, elle pourrait envoyer beaucoup de message différents.

Mais évidemment, les génies n'existent que dans les contes... il faut donc penser à autre chose ! Or nous allons voir que même sans l'aide du génie, et de façon très surprenante, Alice est capable de transmettre $M = 2^k$ messages différents à Bob en utilisant à peine plus de redondance !

Pour ce faire, Alice a recours au hasard. Elle se met ainsi d'accord au préalable avec Bob sur un dictionnaire construit de la façon suivante : à chaque message $m \in \{1, \ldots, M\}$ correspond un mot de code c_m d'une longueur de n bits qui sont tous tirés au hasard indépendamment les uns des autres. Ainsi, dans le cas où $M = 8$ (i.e., $k = 3$) et $n = 5$, elle pourrait obtenir le dictionnaire suivant, par exemple :

message No	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
mot de code	10111	11100	01100	00111	01010	11110	00101	11001

> **Remarque**
>
> Une objection valide à l'utilisation du hasard dans ce cadre est qu'il se peut bien sûr que deux mots de code tirés au hasard soient identiques, ce qui pose évidemment un problème pour Bob si l'un des deux messages correspondants est transmis par Alice. La réponse est que pour de relativement grandes valeurs de n, cette probabilité est extrêmement faible (pour être plus précis, elle est de l'ordre de $n^2/2^n$, donc plus petite que 10^{-25} lorsque $n = 100$, par exemple).

Ce que nous allons montrer dans ce qui suit, c'est que si Alice choisit la valeur de n telle que $k = n\,(1-\delta)$ avec δ juste un peu plus grand que ε (donc un taux de transmission $R = 1 - \delta$, autrement dit, juste un peu plus de redondance que dans la situation idéale décrite plus haut avec le génie), alors Bob sera capable de décoder le message envoyé par Alice avec une probabilité extrêmement proche de

décoder le message envoyé par Alice avec une probabilité extrêmement proche de 1, et ceci quels que soient les $n\,\varepsilon$ bits effacés (qui ne sont pas connus à l'avance ici).

> **Préalable**
>
> Pour ce qui suit, nous allons utiliser quelques concepts de probabilités, et notamment :
>
> - si A et B sont deux événements, alors $P(A \cup B) \leq P(A) + (B)$
> - si A et B sont deux événements *indépendants*, alors $P(A \cap B) = P(A) \cdot P(B)$
>
> ce qui se généralise à :
>
> - si $A_1, \ldots, A_n$ sont n événements, alors $P\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} P(A_i)$
> - si $A_1, \ldots, A_n$ sont n événements *indépendants*, alors $P\left(\bigcap_{i=1}^{n} A_i\right) = \prod_{i=1}^{n} P(A_i)$

Supposons que le message No 1, donc le mot de code c_1, soit envoyé par Alice (on peut refaire le même raisonnement pour n'importe quel message envoyé). Quand

est-ce que Bob aura un problème à retrouver ce message? Quand une confusion entre deux mots de code est possible. Et pour cela, il faudrait que les $n\,(1-\varepsilon)$ bits non effacés de c_1 soient tous égaux à ceux d'un autre mot de code c_m parmi les $M-1$ restants. Quelle est la probabilité qu'un tel événement survienne ?

$P($ il existe $m \in \{2, \ldots, M\}$ tel que $n\,(1-\varepsilon)$ bits de c_1 et c_m sont identiques $)$

$\leq \sum_{m=2}^{M} P(n\,(1-\varepsilon)$ bits de c_1 et c_m sont identiques $)$

$= \sum_{m=2}^{M} \left(\frac{1}{2}\right)^{n\,(1-\varepsilon)} \leq M\left(\frac{1}{2}\right)^{n\,(1-\varepsilon)} = 2^{k - n\,(1-\varepsilon)} = 2^{n\,(1-\delta) - n\,(1-\varepsilon)} = 2^{n\,(\varepsilon - \delta)}$

Cette probabilité tend donc rapidement vers 0 lorsque n est grand et $\delta > \varepsilon$. CQFD

La magie du hasard semble donc opérer pleinement ici ! Ceci dit, il faut relativiser, car un aspect de la communication a été complètement passé sous silence : comment Bob peut-il décoder le message envoyé ? Il peut bien sûr parcourir tous les mots de code du dictionnaire pour trouver celui qui correspond aux $n\,(1-\varepsilon)$ emplacements où les bits n'ont pas été effacés. Mais le nombre M de mots de codes dans le dictionnaire est gigantesque ! Une recherche exhaustive qui parcourt celui-ci est donc vouée à l'échec en pratique ; si le résultat ci-dessus est très intéressant du point de vue théorique, il reste un problème d'envergure à régler…

Ce problème a occupé de nombreuses et nombreux scientifiques depuis les années 50, et n'a été résolu que récemment, dans les années 2000. Les solutions sont en fait multiples, avec un principe commun consistant à construire des dictionnaires structurés pour permettre un décodage beaucoup plus efficace

qu'avec des dictionnaires complètement aléatoires. Il n'empêche que l'intuition de départ de Claude Shannon d'utiliser le hasard a le mérite d'avoir démontré que c'était théoriquement possible, et ainsi encouragé de très nombreuses personnes dans le monde à travailler activement sur le sujet pendant plusieurs années !