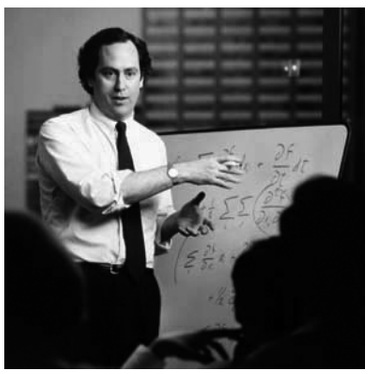
08-10MatricesBis.qxd 26/04/12 9:21 Page 8

par Kylie Ravera

Les matrices au service de la communication

Les messages transmis peuvent arriver avec des erreurs. Comment le savoir ? Comment les corriger ? Les réponses tiennent dans des codes dont le premier est dû au mathématicien américain Richard Hamming. En fait, les matrices sont au cœur des codes correcteurs d'erreur.

Les technologies de la communication accomplissent chaque jour des exploits dont nous n'avons même plus conscience tant ils sont intégrés dans notre vie quotidienne. Pourtant, qui aurait pu imaginer il y a vingt ans seulement qu'il serait possible de poster *via* un téléphone mobile un *tweet* à Las Vegas, lisible quasi-instantanément à Paris ? de publier une vidéo en Nouvelle-Zélande s'affichant en moins d'une seconde au fin fond de l'Alaska ? ou encore de graver sur une galette en plastique de quelques grammes tous les livres jamais écrits ? Parmi toutes les techniques mises en œuvre pour transférer ou conserver des données numériques, il en est une qui permet d'assurer l'intégrité de l'information, ou, le cas échéant, de signaler qu'une erreur de transmission s'est produite. Son bras armé s'appelle le code correcteur d'erreur, et il repose sur des mécanismes mathématiques faisant largement intervenir le calcul matriciel.



Richard Hamming

Le code correcteur d'erreur repose sur des mécanismes mathématiques faisant largement intervenir le calcul matriciel.

La redondance comme principe de base

Depuis le début de l'ère informatique, toute information, texte, image, son, vidéo, peut s'exprimer sous la forme d'une succession de 0 et de 1, appelés *bits*. Ces *bits* offrent l'avantage de pouvoir être « transportés » par des ondes, dont les fréquences dépendent du type de réseau utilisé : satellite, hertzien, fibre, ADSL… Ces réseaux sont soumis à des perturbations qui viennent détériorer le signal, de la même façon qu'un DVD rayé voit ses données corrompues. Le principe du code correcteur est de venir introduire de la redondance sur les informations initiales, pour qu'une détérioration puisse être détectée, et ses conséquences corrigées.

Un exemple simple permet de comprendre le concept de base du code correcteur d'erreur. Supposons qu'Alpha, localisé au point A, veuille demander à Bêta, qui habite au point B, d'aller acheter du sucre. Il lui transmet pour cela la séquence « 1000 », qui représente la donnée « SUCRE » dans le système de communication qu'ils utilisent. Si Alpha l'envoie telle quelle et qu'une erreur se produit au niveau du deuxième *bit*, la séquence reçue par Bêta sera « 1100 ». Il lui sera en l'état impossible de savoir qu'une erreur a été commise, et notre ami se retrouvera peut-être à aller acheter du « SEL »…

08-10MatricesBis.qxd 26/04/12 9:21 Page 9

Alpha choisit à présent de coder l'information initiale en doublant chaque *bit* envoyé : la séquence devient « 11000000 ». Une erreur au deuxième *bit* donne la séquence « 10000000 ». Si Bêta connaît la règle du doublement, il peut voir qu'il ne s'agit pas d'une séquence valide et qu'il y a donc eu un problème au cours de la transmission. Cependant, si on se doute que les trois derniers *bits* de la séquence initiale étaient des 0, qui sont tous les trois doublés, le « 10 » ne permet pas de trancher en ce qui concerne la valeur du premier *bit*. Bêta ne saura donc pas ce qu'il doit aller acheter. Si Alpha fait le choix de tripler les *bits* au lieu de les doubler, la réception d'une séquence comme « 101000000000 » permet en revanche d'établir que le premier bit a une probabilité plus élevée d'être 1 que 0 et que, par conséquent, « 1000 » est la donnée envoyée la plus probable : dans ce cas, Bêta possède un moyen pour corriger l'erreur. Cet exemple montre que :

- il est nécessaire d'ajouter des *bits* d'information à une séquence initiale pour pouvoir supporter la corruption de certains d'entre eux ;
- le nombre d'erreurs détectées est inférieur, au mieux égal, au nombre d'erreurs pouvant être corrigées pour un type de code donné ;
- le nombre d'erreurs pouvant être détectées ou corrigées est limité pour chaque code : un canal très perturbé, qui transformerait la dernière séquence transmise en « 001000000000 », avec donc une erreur sur les deux premiers *bits*, conduirait Bêta à décoder la séquence « 0000 », à tort.

Ces exemples de codes basés sur une simple répétition ne sont pas réalistes en termes d'implémentation : transporter ou stocker de l'information coûte cher, et sachant qu'un volume de données réel correspond à plusieurs millions de bits qui doivent être envoyés sur un réseau en une seconde, aucun opérateur de télécommunication standard n'aurait les moyens de supporter le coût auquel conduirait la multiplication de chacun des bits.

C'est pour résoudre ce problème qu'il existe une série de codes correcteurs d'erreurs, qui se distinguent par :

- la quantité d'information qu'ils demandent à ajouter pour un niveau de détection / correction donné (appelé taux de rentabilité) ;
- le type d'erreurs corrigées : les bits dégradés se suivent-ils ? Sont-ils dispersés dans une longue séquence ? La réponse dépend du type de canal de transport utilisé ;
- la complexité d'encodage et de décodage qu'ils induisent : un code très performant mais qui demande d'importantes ressources en termes de capacité de calcul peut ne pas être économiquement intéressant.

L'un de ces codes est très répandu : le code de Hamming, nommé d'après son inventeur. Soit une matrice

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

Elle possède sept lignes et quatre colonnes, et on reconnaît dans sa partie supérieure la matrice identité I_4. Écrivons d'ailleurs G sous la forme $\begin{pmatrix} I_4 \\ P \end{pmatrix}$ avec

$$P = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

G est la *matrice génératrice* du code C que nous allons étudier : supposons que les informations que nous souhaitons transmettre peuvent se découper en blocs de quatre bits. Tous les mots de la forme (a, b, c, d) où a, b, c, d appartiennent à $\{0, 1\}$ sont dans notre dictionnaire élémentaire. Chacun de ces mots peut être présenté sous la forme d'un vecteur de dimension 4, qui peut donc être multiplié par G. C'est la procédure que nous suivons pour obtenir, pour chaque mot, son équivalent codé (voir le tableau ci-contre).

Les quatre premiers *bits* de chaque mot codé reprennent le mot initial à partir duquel ils ont été construits : cela est dû à la présence de la matrice I4 dans la composition de G, et cela donne le côté « systématique » du code, qui rend aisé le repérage du mot que l'on a souhaité envoyer. Notons également que le dictionnaire des mots de code contient

Mot élémentaire	Mot codé
0000	0000000
0001	0001110
0010	0010101
0011	0011011
0100	0100011
0101	0101101
0110	0110110
0111	0111000
1000	1000111
1001	1001001
1010	1010010
1011	1011100
1100	1100100
1101	1101010
1110	1110001
1111	1111111

08-10MatricesBis.qxd 26/04/12 9:21 Page 10

seize termes, et qu'il existe donc des combinaisons de sept *bits* parmi les 2^7 possibles, comme (1111110), qui n'y figurent pas.

Évaluons maintenant la capacité de correction du code : plus la distance minimale (au sens différence) entre deux mots valides du dictionnaire est grande, plus le code sera efficace pour détecter des erreurs (en effet, plus il faudra d'erreurs pour passer d'un mot valide à un autre mot valide).

Le *décodage à distance minimum* (qui correspond à l'application du critère de maximum de vraisemblance) consiste à interpréter un mot reçu comme le mot de code valide qui est « le plus proche » de lui : ils diffèrent en un nombre minimum de positions. On appelle précisément *distance de Hamming* entre deux vecteurs u et v le nombre de positions où ces deux vecteurs diffèrent. La *distance minimale* d'un code C s'exprime par :

$$d_m = \inf_{u \in C, v \in C, u \neq v} d(u,v).$$

On montre facilement qu'un code C de distance minimum d_m permet de détecter au plus $d_m - 1$ erreurs. En outre, on montre aussi que le nombre d'erreurs pouvant être corrigées est au plus $E((d_m - 1)/2)$, où E désigne la fonction partie entière.

Schématiquement, il s'agit de se retrouver dans une position où il existe un « mot de code le plus proche » : si on est équidistant de deux mots, on saura qu'une erreur s'est produite, mais on n'aura aucun moyen pour faire un choix entre ces deux mots.

Dans le cas du code de Hamming (7, 4) caractérisé par la matrice génératrice G, nous pouvons facilement calculer la distance minimale en remarquant que la distance $d(u, v)$ pour u et v deux vecteurs du code correspond au poids P du vecteur $u + v$, c'est-à-dire au nombre de composantes de $u + v$ qui valent 1.

Cela se traduit par :

$$\begin{aligned} d_m &= \inf_{u \in C, v \in C, u \neq v} d(u,v) \\ &= \inf_{u \in C, v \in C, u \neq v} P(u+v) \\ &= \inf_{w \in C, w \neq 0} P(w). \end{aligned}$$

On complète ainsi notre tableau en comptant le poids de chaque vecteur (figure ci-contre).

Mot élémentaire	Mot codé
0000	0000000
0001	0001110
0010	0010101
0011	0011011
0100	0100011
0101	0101101
0110	0110110
0111	0111000
1000	1000111
1001	1001001
1010	1010010
1011	1011100
1100	1100100
1101	1101010
1110	1110001
1111	1111111

Notre code est donc caractérisé par une distance minimale d_m de 3, soit une capacité de détection de deux erreurs et une capacité de correction d'une erreur.

Introduisons maintenant une seconde matrice

$$H = \begin{pmatrix} -{}^tP \\ I_3 \end{pmatrix} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Une rapide multiplication matricielle montre que ${}^tH \times G = 0$. Cette matrice H est la matrice de contrôle liée au code C généré par la matrice G : pour tout mot codé valide c, et uniquement pour ces mots-là, ${}^tH \times c = 0$.

Les syndromes de l'erreur

Supposons à présent que nous recevions le mot y. Calculons $s(y) = {}^tH \times y$. Si ce vecteur de dimension 3 est nul, cela signifie que le mot reçu appartient à C : il n'y a donc pas d'erreur de transmission (du moins aucune qui soit détectable). Si ce vecteur n'est pas nul, nous savons qu'au moins une erreur s'est produite ; $s(y)$ est le syndrome de y. Que se passe-t-il quand le syndrome n'est pas nul ?

Revoyons la scène au ralenti : le mot de code c a été perturbé par une erreur e qui a conduit à la réception du mot y. Avec une notation binaire, nous pouvons écrire $y = c + e$, où le vecteur e contient des 1 dans les positions où les erreurs se sont produites et des 0 ailleurs. L'additivité implique :

$$\begin{aligned} s(y) = s(c+e) &= {}^tH \times (c+e) = {}^tH \times c + {}^tH \times e \\ &= {}^tH \times e = s(e). \end{aligned}$$

Autrement dit, le syndrome ne dépend pas du mot de code, mais seulement de l'erreur commise.

Cette constatation nous permet de construire notre stratégie de décodage.

Elle se fonde sur la recherche de la séquence z de poids minimal telle que ${}^tH \times z = s(y)$, qui nous permet ensuite de reconstituer le mot de code $c = y + z$.

Un exemple permet de rendre tout cela plus concret. Supposons que nous recevions la séquence

$y = (1100111)$. Calculons le syndrome de y.

$$s(y) = \begin{pmatrix} -1 & 0 & -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & -1 & 0 & 1 & 0 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ -1 \end{pmatrix}$$

équivalent au vecteur $\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$ en notation binaire.

Ce syndrome non nul indique qu'au moins une erreur s'est produite. Lançons-nous à présent à la poursuite de la séquence z. Pour ce faire, nous allons construire une table, appelée *table de décodage*. Elle associe à chaque valeur de syndrome possible le vecteur z de poids minimal tel que ${}^tH \times z = s$. On la construit de la façon suivante : on commence avec $z = 0$ et on calcule ${}^tH \times z$, puis on enchaîne avec les z de poids 1, 2, etc. jusqu'à ce qu'on ait obtenu une valeur z pour chaque syndrome possible. En l'occurrence, nous n'avons pas besoin de mener nos calculs bien loin :

Syndrome	Vecteur d'erreur
000	0000000
001	0000001
010	0000010
011	0100000
100	0000100
101	0010000
110	0001000
111	1000000

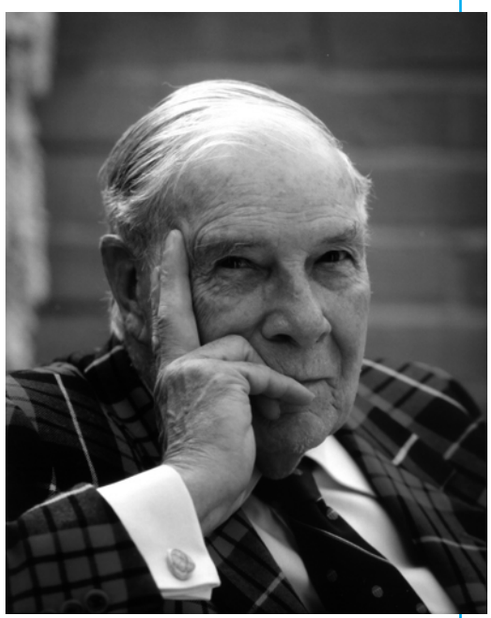
Nous pouvons à présent utiliser cette table pour décoder $y = (1100111)$, qui conduisait au syndrome (011). Nous retrouvons ce dernier à la quatrième ligne de notre tableau, ce qui donne $z = (0100000)$. En appliquant $c = y + z$, on trouve le mot de code (1000111). Le mot initial était donc 1000.

Il existe de nombreux autres codes correcteurs d'erreurs, parmi lesquels le code BCH, le code de Reed-Solomon, le code de Golay, les LDPC, les turbo-codes… Ils se caractérisent par des complexités et des taux de corrections différents. Certains, comme les turbo-codes, sont particulièrement efficaces mais demandent une importante puissance de calcul pour leur mise en œuvre, notamment au niveau de la réception. Ils ont été sélectionnés par la NASA pour les communications spatiales, et pourraient donc bien servir à Alpha pour transmettre sa liste de course à Bêta s'ils n'habitaient pas sur la même planète.

K. R.

Richard Wesley Hamming

Mathématicien américain, Richard Hamming (1915–1998) travailla au projet Manhattan pendant la Seconde Guerre mondiale. Ses premiers résultats sur les codes correcteurs d'erreur datent de 1950. Son choix des mathématiques est clair : « Mathematics is an interesting intellectual sport but it should not be allowed to stand in the way of obtaining sensible information about physical processes. »



Le code de Hamming (7, 4) permet de corriger une erreur. Cependant, lorsqu'il s'en produit deux, le calcul du syndrome détecte bien une anomalie au niveau de la transmission mais conduit à une mauvaise correction. Pour remédier à ce problème, il est nécessaire d'ajouter un *bit* supplémentaire au mot de code envoyé, dit *bit de parité*, obtenu en faisant la somme binaire des sept bits qui composent le mot de code simple. Il autorise une vérification supplémentaire et le code ainsi étendu permet de détecter réellement deux erreurs, même s'il ne parvient pas à les corriger. Un décodage erroné est ainsi évité et une retransmission peut être demandée.

Les machines devraient fonctionner. Les gens devraient penser
Richard Hamming