Code de Hamming

1 Codes linéaires binaires

Soient $n > k > 0$ deux entiers. Un *code linéaire binaire* C de longueur n et de dimension k est un sous-espace vectoriel de dimension k de $\mathbb{F}_2^n$. Une matrice génératrice de C est une matrice de taille $k \times n$ sur $\mathbb{F}_2$ telle que

$$C = \{mG \mid m \in \mathbb{F}_2^k\} .$$

Autrement dit, G est une matrice dont les lignes forment une base du code C. L'application d'encodage relative à une matrice génératrice G est donnée par

$$\begin{aligned}\varphi : \mathbb{F}_2^k &\longrightarrow \mathbb{F}_2^n \\ m &\longmapsto mG .\end{aligned}$$

Un message m de longueur k est donc *encodé* en $c = mG$ qui est un message de longueur n.

Une *matrice de contrôle* du code C est une matrice H de taille $(n-k) \times n$ telle que pour tout message x dans $\mathbb{F}_2^n$, $H \cdot {}^t x = 0$ si et seulement si x appartient à C. Comme les lignes d'une matrice génératrice G forment une base de C, H est une matrice de contrôle de C si et seulement si elle est de rang $(n-k)$ et vérifie

$$H \cdot {}^t G = 0_{(n-k)\times k} .$$

2 Code de Hamming

2.1. Encodage et décodage

Le code de Hamming fut inventé par Richard Hamming en 1950. Il s'agit d'un code linéaire binaire de longueur 7 et de dimension 4. Sa distance minimale est 3, cela signifie qu'il peut détecter deux erreurs et en corriger une. Posons G la matrice génératrice du code donnée par

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

La procédure d'encodage consiste à multiplier le message par G. Par exemple, le message $(1,0,0,1)$ est encodé en $(1,0,0,1,1,0,0)$. Pour décoder *un mot du code*, il suffit d'effacer les trois derniers bits. Posons H la matrice de contrôle donnée par

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

Remarque. Si nous lisons les colonnes de H de haut en bas, nous obtenons les entiers de 1 à 7 en binaire. Cette propriété remarquable va nous fournir un algorithme de correction très efficace.

2.2. Correction d'une erreur

Soit c un mot du code C. Soient $1 \le i \le 7$ un entier et $e = (e_1, \ldots, e_7) \in \mathbb{F}_2^7$ tel que $e_i = 1$ et $e_j = 0$ si $j \neq i$. Le message e représente une erreur, autrement dit $c + e$ diffère du message c en position i. En appelant H_i la i-ème colonne de H, nous avons

$$H \cdot {}^t(c+e) = H \cdot {}^t c + H \cdot {}^t e = 0 + H \cdot {}^t e = H_i \ .$$

Comme H_i représente l'entier i en binaire, le calcul $H \cdot {}^t(c+e)$ nous donne exactement la position de l'erreur. Nous en déduisons la procédure suivante.

- On reçoit un message $c' \in \mathbb{F}_2^7$ pour lequel il y a eu *au plus* une erreur de transmission.
- Posons $s = H \cdot {}^t c'$.
  - Si $s = (0,0,0)$, alors c' est un mot du code et il n'y a rien à corriger.
  - Si $s \neq (0,0,0)$, alors le bit d'indice s (lu en binaire) est erroné.

3 Travail à réaliser

Exercice 1. Sur papier

1.1) Encoder le message $m = (1,1,0,1)$.

1.2) Vérifier que H est bien une matrice de contrôle du code.

1.3) En utilisant H, vérifier que $(1,1,1,1,1,1,1)$ est un mot du code.

1.4) On reçoit le message $c' = (1,1,1,1,0,0,1)$. Vérifier que c' n'est pas un mot du code. Sachant qu'il n'y a eu qu'une seule erreur de transmission, corriger c' puis le décoder.

Exercice 2. Sur ordinateur

2.1) Écrire une fonction permettant d'encoder un message de 4 bits.

2.2) Écrire une fonction permettant de décoder un mot du code.

2.3) Écrire une fonction permettant de corriger une erreur sur un message de 7 bits.

2.4) Écrire une fonction générant aléatoirement un message de 4 bits.

2.5) Écrire une fonction simulant la transmission bruitée d'un message de 7 bits. Cette fonction prend également un réel $p \in [0,1]$ en argument. Chaque bit du message doit avoir une probabilité p d'être erroné.

2.6) Dans votre programme principal, répéter dans une boucle les opérations suivantes :
— générer aléatoirement un message m ;
— encoder m en c ;
— ajouter du bruit à c, on appelle c' le message obtenu ;
— corriger puis décoder c', on appelle m' le message obtenu ;
— tester si $m = m'$.
Exécuter ce programme avec $p = 0.001$, 0.005, 0.01, 0.02, $0.03 \ldots$

2.7) Pour quelles valeurs de p la correction est efficace ? Pour quelles valeurs la correction ajoute d'avantage d'erreurs ? Comment pouvez-vous l'expliquer ? Écrire un rapport.