

Codes LDPC

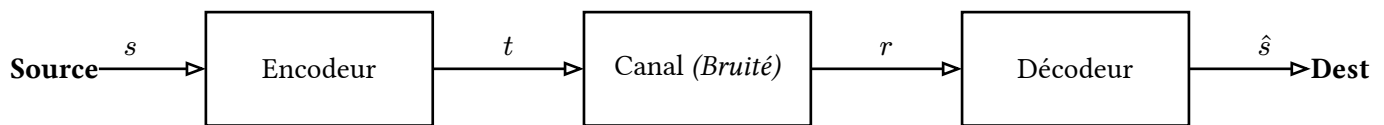
Notes pour TIPE

Introduction à la théorie de l'information

Problème principal : il y a du bruit dans les transmissions mais on veut pas d'erreurs.

Au lieu de trouver des modifications physiques on va créer des solutions pour corriger les erreurs.

Il faut un **encodeur** qui ajoute de la redondance et un **décodeur**.



Code correcteur d'erreurs pour un canal binaire symétrique [1]

Le but est de transformer un canal bruité en canal fiable avec un coût de calculs en plus (encodeur / décodeur).

On va chercher la meilleure performance de correction d'erreurs. Ce sont les limites théoriques que cherchent à trouver la *Théorie de l'information*. Pas de retransmissions.

Codes de répétition

Définitions

Il s'agit ici de répéter tous les bits. Un message source s , un message transmis t , un vecteur de bruit n et un message reçu r . $r = t + n$.

$$\begin{array}{l} s = \underbrace{0}_{000} \underbrace{1}_{111} \\ t = 000 \quad 111 \\ n = 001 \quad 010 \\ r = 001 \quad 101 \end{array}$$

On décode en choisissant le bit le plus présent dans un bloc.

C'est ce que représente le *Likelihood ratios* : $\frac{P(r | s=1)}{P(r | s=0)}$

Ici $001 \rightarrow 0$ et $101 \rightarrow 1$ donc $\hat{s} = 0$.

Preuve de l'optimalité (dans le sens la plus faible probabilité d'erreurs) [1] p6.

Problème : trois fois plus de bande passante...

Single parity check code et définitions algébriques

Ajout d'uniquement 1 bit d'information à la fin sur la parité du nombre de 1.

Un message s est de la forme

$$s = [s_1 \ s_2 \ s_3 \ s_4 \ s_5 \ s_6]$$

où $c_i \in \{0, 1\}$ et le *codeword* vérifie la contrainte si

$$s_1 \oplus s_2 \oplus s_3 \oplus s_4 \oplus s_5 \oplus s_6 = 0$$

E

parity-check equation.

Inversion d'un nombre bit paire $\Rightarrow E = 0$ donc aucune erreur détectée.

C'est donc pas assez puissant pour savoir quel bit a changé.

On écrit sous forme matricielle.

$$Hs^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

avec H la matrice *parity-check* où chaque ligne de H correspond à l'équation de parité et chaque colonne de H correspond à un bit du *codeword*.

Les contraintes sont alors les suivantes

$$s_4 = s_1 \oplus s_2, s_5 = s_2 \oplus s_3, s_6 = s_1 \oplus s_2 \oplus s_3$$

De plus s_4, s_5, s_6 sont les bits de parités et :

$$s = [s_1 s_2 s_3 s_4 s_5 s_6] = [s_1, s_2, s_3] \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}}_G$$

où G est la matrice génératrice du code.

On note $u = [u_1, \dots, u_k]$ où u contient les k bits du message, ici $u = [u_1, u_2, u_3]$

$$s = uG$$

Pour un message de longueur k et n *codewords*, $G \in M_{k \times n}(\mathbb{Z}/2\mathbb{Z})$.

De plus $\frac{k}{n}$ est le *rate* du code.

Un code de taille k contient 2^k *codewords*. Ces *codewords* sont des sous-ensembles avec 2^n vecteurs de taille n possibles.

On peut obtenir H sous la forme

$$H = [A, I_{n-k}]$$

avec $A \in M_{(n-k) \times k}(\mathbb{Z}/2\mathbb{Z})$ et donc

$$G = [I_k, A^T]$$

De plus si G est la matrice génératrice pour un code avec matrice de parité H alors

$$GH^T = 0$$

G est orthogonal à H .

Un code peut avoir autant de contraintes *parity-check* qu'il veut mais seulement $n - k$ d'entre elle seront linéairement indépendantes. C'est à dire :

$$n - k = \text{rg}(H)$$

Voir [2]

Comment détecter et corriger les erreurs

Supposon qu'on envoie $s = [1 \ 0 \ 1 \ 1 \ 1 \ 0]$ et qu'on recois $r = [1 \ 0 \ 1 \ 0 \ 1 \ 0]$ alors

$$Hr^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Le vecteur $s = Hr^T$ est le **syndrome** de r , il indique quel contrainte de *parity-check* ne sont pas satisfaites par r .

Ici $s = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ et l'équation de parité associé est $s_4 = s_1 \oplus s_2$.

Un *block code* ne peut détecter des erreurs que si ces dernières ne transforment pas un *codeword* valide en un autre *codeword* valide. (voir Code de *Hamming*)

- Distance de Hamming : Nombre de positions où les bits diffèrent entre deux *codewords*.
Exemple : $[1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0]$ et $[1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1]$ diffèrent aux positions 3 et 8
 \Rightarrow Distance de Hamming = 2.
- Distance minimale (d_{\min}) : La plus petite distance de Hamming mesurée entre n'importe quelle paire de *codewords* appartenant au code.

Un code avec une distance minimale d_{\min} peut garantir la détection de t erreurs si et seulement si :

$$t < d_{\min}$$

Exemple :

Pour *Hamming* (7,4) vu après, on a $d_{\min} = 3$.

\Rightarrow Il garantit la détection de 1 ou 2 erreurs ($t < 3$).

\Rightarrow Si 3 bits (ou plus) s'inversent, le message peut correspondre à un autre *codeword* valide. (Exemple 1.8 [1]).

Pour corriger l'erreur, le décodeur cherche le *codeword* le plus probable.

Principe (*maximum-likelihood* (ML) Decoder) : Il choisit le *codeword* s valide qui a la plus petite distance de Hamming avec le message reçu r . (Si égalité alors le choix est aléatoire).

$$\hat{s} = \min_{c \in C} d_H(r, s)$$

Avec C l'ensemble des *codewords* valides.

Code de *Hamming*

But : Ajouter de la redondance à des blocs de données.

Block code : règle de conversion d'une séquence de bits s de longueur K dans une séquence t de N bits. (Redondance $\Rightarrow N > K$).

Dans un code linéaire les $N - K$ bits réstant sont linéaire en fonction des K bits originaux, ce sont les *parity-check bits*.

- *Hamming* (7,4)

L'encodage se visualise via 3 cercles sécants (Diagramme de Venn). Les 7 bits sont placés de sorte que la parité de chaque cercle soit paire (somme = 0).

- Bits de Source (s_1, s_2, s_3, s_4) : Copiés directement dans le message transmis ($t_1..t_4$).
- Bits de Parité (t_5, t_6, t_7) : Calculés pour valider les cercles.

$$t_5 = s_1 \oplus s_2 \oplus s_3 \text{ (Cercle 1)}$$

$$t_6 = s_2 \oplus s_3 \oplus s_4 \text{ (Cercle 2)}$$

$$t_7 = s_1 \oplus s_3 \oplus s_4 \text{ (Cercle 3)}$$

Le **Syndrome** z : On vérifie la parité des cercles à l'arrivée.

- 1 cercle faux \rightarrow Erreur sur le bit de parité.
- 2 ou 3 cercles faux \rightarrow Erreur à l'intersection unique des cercles fautifs.

On peut le voir sous forme de matrice.

Message transmit t (*codeword*) :

$$t = G^T s$$

avec G la matrice génératrice du code.

Visualisation de la solution avec diagramme de Venn.

Trouver P tel que

$$G^T = \begin{bmatrix} I_n \\ P \end{bmatrix}$$

avec $z = Hr$ et H la matrice *parity-check* $H = [-P \ I_{n-1}] = [P \ I_{n-1}]$

Et donc tous les *codewords* satisfont $t = G^T s$,

$$Ht = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Mais $r = G^T s + n$ on doit trouver n tel que $Hz = z$. C'est le problème *maximum-likelihood decoder*.

Voir exemple [1] p9. | 3Blue1Brown Hamming codes

Low-density parity-check codes (LDPC) [2]

Bibliographie

- [1] David J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [2] Sarah J. Johnson, « Introducing Low-Density Parity-Check Codes ».