1 Codes binaires parfait - Codes de Hamming

1. Soit C un code binaire 1-correcteur de longueur n et de cardinal N. Montrer $N(n+1) \leq 2^n$.

Comme C est de distance 3, les N boules centrées sur les mots de code et de rayon 1 (chacune de cardinal $1 + C_n^1 = n+1$) sont disjointes dans $\{0,1\}^n$: donc $N(n+1) \leq 2^n$.

En déduire que si le code est parfait alors il existe un entier $\rho > 1$ tel que $n = 2^\rho - 1$ et $N = 2^{n-\rho}$. Justifier qu'un tel code est de rendement optimal.

Si le code est parfait, les N boules précédentes sont une partition de $\{0,1\}^n$: donc $N(n+1) = 2^n$. Ainsi, $n+1$ est une puissance de 2 et $N = 2^{n-\log_2(n+1)}$.
La dimension du code est alors $k = \log_2 N = n - \log_2(n+1)$ et il y a $r = n-k = \log_2(n+1)$ bits de redondance. Or un code binaire 1-correcteur doit être capable de distinguer $n+1$ évènements : il n'y a pas d'erreur ; il y a 1 unique erreur en position i. L'entropie de la correction est alors $\log_2(n+1)$, donc il faut au moins $\log_2(n+1)$ bits de redondance. Le rendement est donc optimal.

Un code de Hamming est un code binaire parfait 1-correcteur. Donner une matrice génératrice du code de Hamming H_7 de longueur 7.

Comme $n = 7$ d'après la question précédente, $r = \log_2 n + 1 = 3$ et $k = 4$; le code est donc $(7,4)$. Sous forme systématique, une matrice génératrice G de H_7 s'écrit : $G = \begin{bmatrix} 1 & 0 & 0 & 0 & g_{1,5} & g_{1,6} & g_{1,7} \\ 0 & 1 & 0 & 0 & g_{2,5} & g_{2,6} & g_{2,7} \\ 0 & 0 & 1 & 0 & g_{3,5} & g_{3,6} & g_{3,7} \\ 0 & 0 & 0 & 1 & g_{4,5} & g_{4,6} & g_{4,7} \end{bmatrix}$.
Le code est de distance 3, donc chaque ligne de de G est de poids ≥ 3 et la distance entre deux lignes est au moins 3. On en déduit que $G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ convient.
Une autre méthode est de dire que les 3 bits de redondance doit coder la position $1 \leq i \leq 7$ de l'unique bit erroné si il existe, et 0 sinon. Soit $[c_1, c_2, c_3, c_4, c_5, c_6, c_7]$ un mot de code : les $r = 3$ bits c_{2^j} avec $(0 \leq j < log_2(n+1))$ sont des bits de redondance et les $k = 4$ autres bits (c_3, c_5, c_6, c_7) sont les bits de source. On définit les bits de contrôle par c_{2^j} est le bit de contrôle de parité des bits de source c_i où l'écriture en binaire de l'entier i a un 1 dans le j-ième bit. Autrement dit : $c_1 = c_3 + c_5 + c_7$; $c_2 = c_3 + c_6 + c_7$; $c_4 = c_5 + c_6 + c_7$. En remarquant que si $[s_1, s_2, s_3, s_4]$ est un mot source, un codage possible est $c_3 = s_1, c_5 = s_2, c_6 = s_3, c_7 = s_4$, la matrice associée est alors : $G' = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$ convient.
La matrice G' est tivialement équivalente à la matrice G (à une permutation de colonnes près ici).
Plus généralement toute matrice $G' = A.G$ avec A matrice $k \times k$ inversible engendre le même code que G ; et $G' = G.P$ avec P matrice $n \times n$ de permutation de colonnes engendre un code équivalent à G. engen

2. Pour un code de Hamming, expliciter les algorithmes de codage et de décodage avec correction d'une erreur.

On a $r = \log_2 n + 1$ bits de redondance. En généralisant la formulation précédente, on partitionne $[1, \ldots, n]$ en deux tableaux ordonnés P et I où P contient les indices qui sont des puissances de 2 et I les autres. Ainsi, pour $n = 31$: $P = [1, 2, 4, 8, 16]$ et $I = [3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, \ldots, 31]$.
Codage : soit $s_1, \ldots s_k$ un mot source, le mot de code $c = [c_1, \ldots, c_n]$ associé est construit ainsi :
- Pour $i = 1, \ldots, k$ faire $c_{I[i]} := s_i$;
- Pour $j = 0, \ldots, \log_2 n + 1 - 1$, calcul du bit de parité $c_{2^j} = \sum_{i \in I, (i \div 2^j) \bmod 2 = 1} c_i$.

Décodage : On reçoit $[y_1, \ldots, y_n]$. Soit L l'ensemble des bits de parité non vérifiés :
- Soit $L = \{2^j \in P : c_{2^j} \neq \sum_{i \in I, (i \div 2^j) \bmod 2 = 1} c_i\}$;
- soit $i = \sum_{j \in L} j$; on modifie $y_i = 1 + y_i$. Ainsi, les bits de parité sont tous vérifiés : y est alors un mot de code à distance 1 du mot reçu, et on le retourne.

3. Soit un canal binaire symétrique avec une erreur de probabilité $p < 0.5$; on suppose que $\log_2 p$ est entier. On s'intéresse à développer un code de Hamming de rendement maximal garantissant un taux d'erreur résiduelle inférieur à ϵ arbitraire.

(a) Ici, on construit un code de Hamming qui corrige une erreur unique dans un bloc de taille $n = \frac{1/p}{-}1$. Asymptotiquement quand $p \to 0$, comparer le rendement de ce code à la capacité du canal.

$r = -\log_2 p$, donc $R = 1 - \frac{p \log_2 p}{1-p} \simeq_{p \to 0} 1 - p \log_2 p$. Or la capacité de canal est $C = 1 - p \log_2 p - (1-p)\log_2(1-p) \simeq_{p \to 0} 1 - p \log_2 p$. Donc le rendement tend vers la capacité de canal.

(b) On suppose que le code est de longueur n ; calculer le taux τ d'erreur résiduelle avec le code de Hamming H_n en fonction de p, n. En déduire que $n < \frac{1}{p} - 1$.

Avec un code de Hamming, τ est la probabilité d'avoir 2 erreurs ou plus, donc $\tau = 1 - (1-p)^n - C_n^1 p(1-p)^{n-1} = 1 - (1-p)^{n-1}(1-(n+1)p)$.
Ce taux doit être inférieur à 1 d'où $(n+1)p < 1$, i.e. $n < p^{-1} - 1$.

(c) Qu'en déduisez-vous sur l'existence d'un code de Hamming d'erreur résiduelle ϵ ?

On doit avoir $(1-p)^{n-1}(1-(n+1)p) \geq 1 + \epsilon$: absurde.

4. Montrer qu'un code de Hamming est équivalent à un code cyclique.

2 Codes cycliques et Reed-Solomon

On suppose que le vocabulaire V du canal V est un corps fini (i.e. possède $q = p^m$ chiffres).
Soit l'application linéaire σ de V^n, appelée *opération de décalage*, définie par $\sigma([u_0, \ldots, u_{n-1}]) = [u_{n-1}, u_0, \ldots, u_{n-2}]$ ..
Un code linéaire C sur V est *cyclique* ssi $\forall x \in C : \sigma(x) \in C$.
Les codes cycliques ont un double intérêt : d'une part [exercice 1], le codage et le décodage/correction sont rapides, en $O(n \log n)$; d'autre part, il est possible de construire des codes cycliques, tels les codes de Reed-Solomon [exercice 2] de distance maximale, atteignant la borne de Singleton.

2.1 Caractérisation d'un code cyclique

Dans toute la suite, C désigne un code cyclique (n,k) quelconque (sous forme systématique).

1. Montrer que le code binaire de parité $(n, n-1, 2)$ est un code cyclique.

Le code est linéaire et $x = [x_1, \ldots, x_n] \in C \iff \sum_{i=1}^n x_i \iff [x_n, x_1, \ldots, x_{n-1}] \in C$.

2. Expliciter σ^{-1}. En déduire que si $x \in C$, alors $\sigma^{-1}(x) \in C$.

On a $\sigma^n = \mathrm{Id}$ d'où $\sigma^{-1} = \sigma^{n-1}$. Soit $x \in C$; comme σ est stable dans C, $\sigma^{-1}(x) = \sigma^{n-1}(x) \in C$.

3. Montrer que C admet une matrice génératrice G_C de la forme :

$$G_C = \begin{bmatrix} m \\ \sigma(m) \\ \ldots \\ \sigma^{k-1}(m) \end{bmatrix}$$

avec $m = [a_0, a_1, \ldots, a_{n-k} = 1, 0, \ldots, 0]$.
Indication : on pourra considérer $\sigma^{1-k}(u)$ où u est la dernière ligne de la matrice génératrice normalisée de C.

Soit $u = [0, \ldots, 0, 1, u_1, \ldots, u_r]$ la dernière ligne de la matrice génératrice normalisée G_1 de C. Comme C est cyclique, d'après la question précédente, $v = \sigma^{-k+1}(u) = [1, u_1, \ldots, u_r, 0, \ldots, 0] \in C$.
Montrons que $u_r \neq 0$. Comme C est cyclique, $\sigma^i(v) \in C$ pour $i = 0, \ldots, k-1$. Ces k vecteurs étant indépendants, il forment une matrice génératrice G_2 du code C. Par l'absurde, supposons $u_r = 0$; alors la dernière colonne de G_2 est nulle : donc tous les mots de C ont leur dernière composante nulle : ce qui est absurde puisque, comme C est cyclique, $\sigma^{-1}(v) \in C$ et a sa dernière composante égale à 1. On en déduit que $u_r \neq 0$.
Soit alors $m = \frac{1}{u_r}.v$ qui appartient à C car C est linéaire ; m est de la forme $m = [a_0, a_1, \ldots, a_{n-k} = 1, 0, \ldots, 0]$.
Comme C est cyclique, $\sigma^i(m) \in C$ pour $i = 0, \ldots, k-1$. Ces k vecteurs étant indépendants, il forment une matrice génératrice G_C du code C de la forme demandée, qed.

4. Tout élément $U = [u_0, \ldots, u_{n-1}]$ de V^n peut être représenté par le polynôme P_U de degré n de $V[X]$ défini par : $P_U = \sum_{i=0}^{n-1} u_i X^i$. Montrer que

$$P_{\sigma(U)} = X.P_U \bmod (X^n - 1)\,.$$

Remarque : Comme X^n-1 est un polynôme de degré n, $V^n \equiv V[X]/(X^n-1)$ (l'anneau des restes dans la division euclidienne modulo X^n-1). Ainsi, cette question montre que $P_{\sigma(U)} = X.P_U$ dans l'anneau $V[X]/(X^n-1)$.

$P_{\sigma(U)} = X.P_U(X) - u_{n-1}.(X^n-1) = X.P_U \bmod (X^n-1)$.

5. Dans toute la suite, g désigne le polynôme $g(X) = P_m = \sum_{i=0}^{n-k} a_i X^i$, appelé *polynôme générateur* de C. Soit $x \in C$; montrer que P_x est multiple de g modulo (X^n-1).
Autrement dit, $P_x \bmod g = 0$ dans l'anneau $V[X]/(X^n-1)$.

x est une combinaison linéaire des lignes $\sigma^i(m)$ de G_C ; donc P_x est une combinaison linéaire des polynômes $P_{\sigma^i(m)}$. Or, d'après la question précédente, $P_{\sigma^i(m)} = X^i.P_m$ dans l'anneau $V[X]/(X^n-1)$. Donc, dans l'anneau $V[X]/(X^n-1)$, P_x est une combinaison linéaire des polynômes X^iP_m, tous multiples de $P_m = g$. Donc P_x est multiple de g.

6. Montrer que g est un diviseur de X^n-1.

$P_{\sigma^k(m)} = X^k.g \bmod X^n-1$. Comme C est cyclique et $m \in C$, $\sigma^k(m)$ appartient à C ; donc $P_{\sigma^k(m)}$ est combinaison linéaire des $P_{\sigma^i(m)} = X^i.g$ pour $i = 0, \ldots, k-1$.
Ainsi, il existe $(\alpha_0, \ldots, \alpha_{k-1})$: $X^k.g = \sum_{i=0}^{k-1} \alpha_i.X^i.g \bmod X^n-1$. Soit $(X^k - \sum_{i=0}^{k-1} \alpha_i.X^i).g = 0 \bmod X^n-1$,
.
Comme $(X^k - \sum_{i=0}^{k-1} \alpha_i.X^i)$ est unitaire de degré k, g est unitaire de degré $n-k$ et X^n-1 est unitaire de degré n, on en déduit que $(X^k - \sum_{i=0}^{k-1} \alpha_i.X^i).g = (X^n-1)$. Donc g est un diviseur de X^n-1.

7. Montrer que l'opération de codage se ramène à un produit de polynômes modulo X^n-1 que l'on explicitera.
Remarque : Le codage est donc une multiplication de polynômes qui se ramène à une FFT, soit de coût $O(n \log n)$.

Soit $u = [u_0, \ldots, u_{k-1}] \in V^k$ un mot source et $P_u = \sum_{i=0}^{k-1} u_i X^i$ son polynôme associé. Le mot de code associé à u est $\phi(u) = u.G_C$ de polynôme associé $P_{u.G_C}$. De part l'écriture de G_C à partir des coefficients de $g(X)$, on a :

$$\begin{aligned} P_{u.G_C} &= \sum_{i=0}^{k-1} u_i (X^i g(X) \bmod X^n-1) \\ &= [g(X)(\sum_{i=0}^{k-1} u_i X^i)] \bmod X^n-1 \\ &= [g(X).P_u(X)] \bmod X^n-1 \ . \end{aligned}$$

Le codage correspond donc à un produit de polynômes par g, les degrés des monômes étant pris modulo n : en effet, calculer X^n-1 revient à considérer que $X^n = 1 = X^0$.

8. Montrer que $x \in C$ si et seulement si P_x est multiple de g. En déduire que la détection d'erreur se ramène à une division polynomiale que l'on explicitera.
Remarque : La détection d'erreurs est donc une division de polynômes, qui se ramène aussi à une multiplication de polynômes de coût $O(n \log n)$. L'algorithme de Berlekamp-Massey permet de réaliser la correction pour un code cyclique avec un coût similaire.

On a déja vu que tout mot de code est multiple de g modulo X^n-1. Or, comme g est de degré $r = n-k$ et diviseur de X^n-1, il y a exactement $|V|^k$ multiples de g modulo X^n-1, qui sont obtenus en multipliant g par un polynôme de degré inférieur ou égal à $k-1$. Comme $\mathrm{Card}(C) = |V|^k$, on en déduit que tout multiple correspond nécessairement à l'un des $|V|^k$ mots de code.
Détection d'erreurs : pour chaque mot y reçu, on considère le polynôme P_y associé. On calcule $P_y \bmod g$: si $= 0$, $y \in C$ et on ne détecte pas d'erreur. Sinon, on a détecté une erreur.

9. En résumé, on a montré que tout code cyclique est caractérisé par la donnée d'un polynôme générateur qui est un diviseur unitaire de degré $r = n-k$ de X^n-1.
Le code C est donc caractérisé par la donnée de seulement $r = n-k$ coefficients : $g_0, \ldots, g_{n-k-1}$.
Justifier que réciproquement, la donnée d'un polynôme g diviseur unitaire de degré $n-k$ de X^n-1 définit un unique code cyclique (n,k).

La donnée des $r = n-k$ coefficients de g définit une matrice G_C unique et donc un code linéaire unique. Or g étant un diviseur de X^n-1, C est stable par σ donc cyclique.

2.2 Application : codes de Reed-Solomon

Soit α un élément primitif du corps $V = \mathbb{F}_q$.
Les codes de Reed-Solomon sont des codes cycliques sur V de de longueur $n = q-1$ dont le polynôme générateur g de degré r est de la forme :

$$g(X) = \prod_{i=s}^{s+r-1} (X - \alpha^i) \, .$$

Le code de Reed-Solomon ainsi obtenu est donc un code $(n = q-1, k = n-r = q-1-r)$ avec r arbitraire. Ce code est de distance $r+1$ [cf polycopié].
En choisissant r, on peut donc construire un code de distance arbitraire, donc de taux de correction arbitraire.

1. Montrer qu'un code de Reed-Solomon est de distance maximale parmi les codes linéaires sur V.

La borne de Singleton montre que $\delta \leq n-k+1 = r+1$; ainsi, la distance $\delta = r+1$ du code de Reed-Solomon atteint la borne de Singleton pour un code linéaire.

2. Le satellite d'exploration de Jupiter *Galileo* utilise le code de Reed-Solomon sur $V = \mathbb{F}_{256}$ de polynôme générateur

$$g = \prod_{j=12}^{43} (X - \alpha^{11j})$$

où α est un élément primitif de $\mathbb{F}_{2^8}$. Quels sont les caractéristiques de ce code sur $\mathbb{F}_{2^8}$? Quelle est sa distance ? Combien d'erreurs sur V peut-il corriger ? Quel est son rendement et son taux de correction sur V ?

Le code est de longueur 256-1=255. g est de degré $r = 43-12+1 = 32$; c'est donc un code (255,223, 33).
Distance = 255-223+1=33. Il est 16-correcteur.
Rendement = $223/255 \simeq 87\%$. Taux de correction = $16/255 \simeq 6\%$.

3. Un code de Reed-Solomon sur $V = \mathbb{F}_{2^m}$ avec r chiffres de redondance peut être vu comme un code binaire C sur $\{0,1\}$.
Donner les caractéristiques de ce code binaire C : longueur, nombre de bits de redondance, distance.

Chaque élément de $\mathbb{F}_{2^m}$ est codé sur m bits. La distance est inchangée. Le code binaire C est un code $(2^m - m, (2^m-1-r).m, r+1)$.

4. Quel est le nombre maximal de bits consécutifs erronés (on parle de *paquet d'erreurs*) que le code binaire C garantit pouvoir corriger ?

Le code de Reed-Solomon peut corriger au maximum jusqu'à $t = \lfloor \frac{r-1}{2} \rfloor$ chiffres de m bits consécutifs, donc tout paquet qui ne perturbe pas plus de t chiffres de V consécutifs (chacun comportant m bits). Donc C corrige tout paquet de longueur inférieure ou égale à à $\left(\lfloor \frac{r-1}{2} \rfloor - 1\right).m + 1$.

5. *Application 1 : Code Galileo pour comunications spatiales.* Donner les caractéristiques du code Galileo en tant que code binaire, son rendement et son taux de correction. Quelle taille de paquet d'erreurs permet-il de corriger ?

On a $m = 8$ et $r = 32$. En temps que code binaire, c'est un code $(8 \times 255 = 2040, 8 \times 223 = 1784, 33)$.
Le rendement=223/255=87% est inchangé.
Mais le taux de correction de bits n'est plus que de $16/2040 \simeq 0.78\%$.
Le code permet de corriger tout paquet d'erreurs de longueur inférieure à 121 bits.

6. *Application 2 : Construction d'un code de Reed-Solomon de taux de correction arbitraire.* On communique sur un réseau qui transmet des bits avec un taux d'erreur de 1%.
Pour y remédier, lorsqu'on envoie n octets, on veut détecter et corriger $0.01 \times n$ erreurs.
Expliquer comment construire un code correcteur de type Reed-Solomon en précisant :
a. le corps de base et la valeur choisie pour n ;
b. le degré du polynôme générateur et son écriture ;
c. le nombre maximal d'erreurs détectées ;
d. la dimension d'une matrice génératrice du code. Comment s'écrit cette matrice à partir des coefficients du polynôme générateur ?