Signaux et systèmes

Codes détecteurs et correcteurs d'erreurs

25 mai 2004

Signal Processing Institute
Swiss Federal Institute of Technology, Lausanne

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

- **Conséquence directe de la loi des grands nombres**
- **Si X_1, X_2, … sont des variables aléatoires i.i.d. p(x), on a**

$$-\frac{1}{n}\log p(X_1 X_2 ... X_n) \to H(X) \qquad \text{en probabilité}$$

$$p(X_1 X_2 ... X_n) \to 2^{-n.H(X)}$$

- **Ensemble typique** $A_e^{(n)}$

$$x_1 x_2 ... x_n \in A_e^{(n)} \Leftrightarrow 2^{-n(H(X)+e)} \le p(x_1 x_2 ... x_n) \le 2^{-n(H(X)-e)}$$

$$\Pr(A_e^{(n)}) > 1-e$$

$$(1-e).2^{n(H(X)-e)} \le \left|A_e^{(n)}\right| \le 2^{n(H(X)+e)}$$

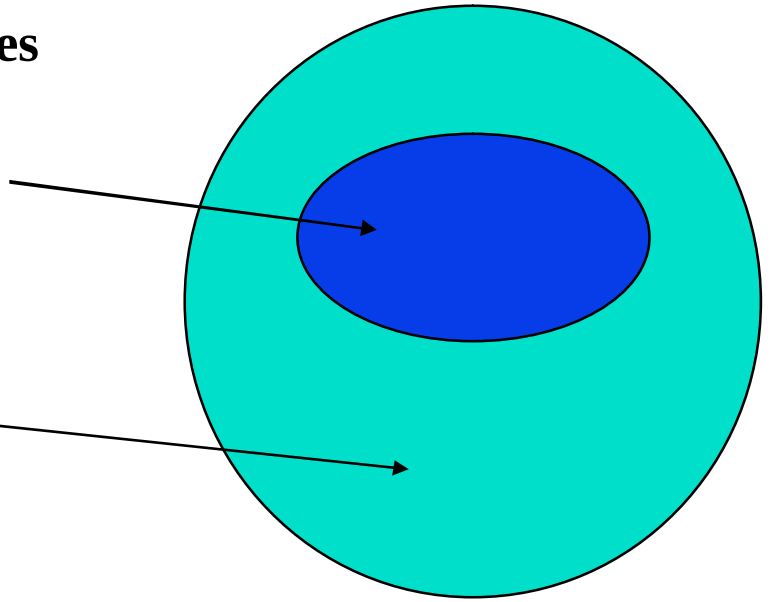
- **Code pour les $2^{n(H+e)}$ séquences typiques**

$$n(H+e)+2 \text{ bits}$$

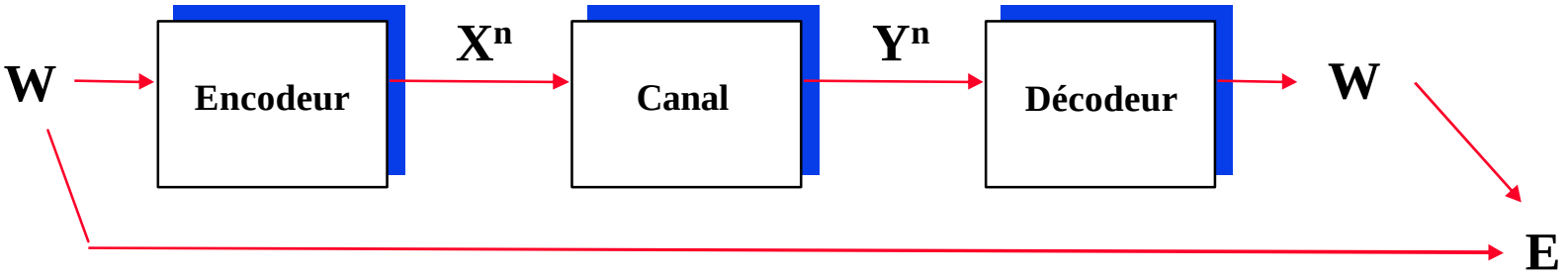
- **Code pour les autres D^n séquences**

$$n \log D + 2 \text{ bits}$$

- **Longueur moyenne du code**



$$\begin{aligned}\bar{n} &= \Pr(A_e^{(n)}).[n.(H+e)+2] + \Pr(A_e^{(n)c}).(n.\log D + 2) \\ &\le n.(H+e)+2+e.n.\log D + 2.e \\ &= n.(H+e')\end{aligned}$$



- **On définit un code (M,n) pour un canal X ? Y comme**
  - **Un alphabet { 1,2, … M }**
  - **Une fonction d'encodage X^n: { 1,2 ... M } ? X^n**
  - **Une fonction de décodage g: Y^n ? { 1,2 ... M }**

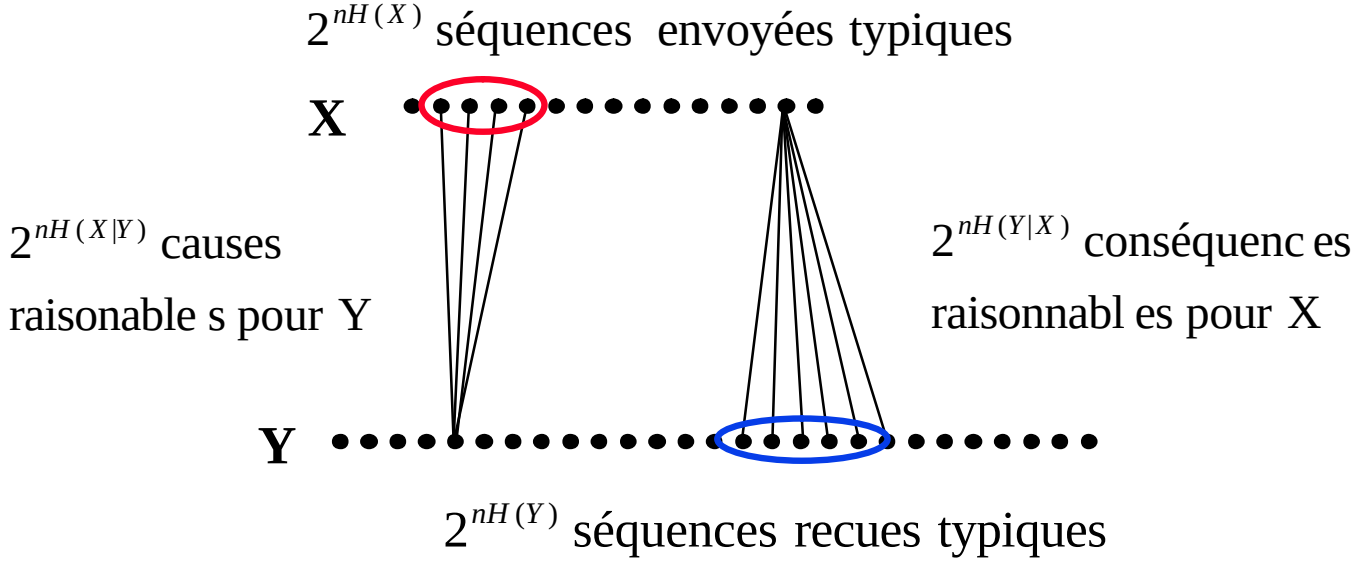
- **Le taux R de ce code est** $R = \frac{\log M}{n}$ bits par transmission

- **La probabilité d'erreur P(W?W) est liée au taux via l'inégalité**

$$P_e^{(n)} \geq 1 - \frac{C}{R} - \frac{1}{nR}$$

- **Le théorème de Shannon dit que l'on peut transmettre avec aussi peu d'erreur que l'on veut tant que R = C = max I(X,Y) , et pas si R > C**

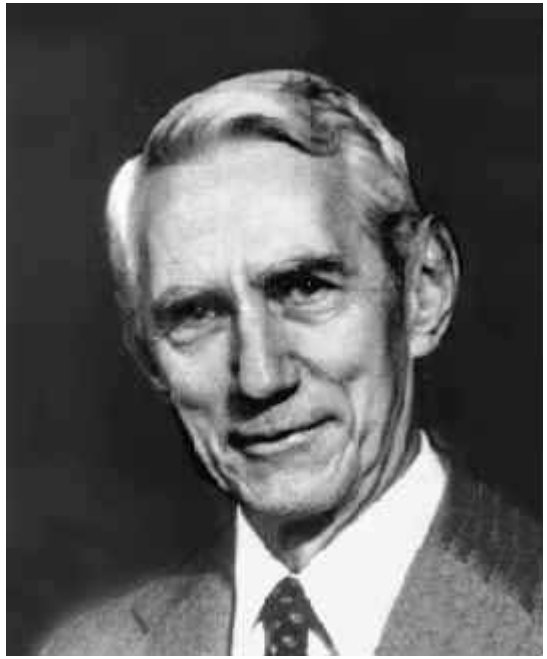
- **La preuve utilise des codes aléatoires, un décodage jointement typique et en étudie le comportement asymptotique pour de longues séquences. Elle est non constructive**



- **Shannon vs. Hamming**
- **Codes de Hamming (1950)**
- **Application: RAID**
- **Codes de Reed-Solomon (1960)**
- **Autres codes en bref**
- **Application: Correction d'erreur sur un CD**

Shannon

- **Né le 30 avril 1916, Gaylord, MI, USA**
- **Mort le 24 février 2001, Medford, MA, USA**
- **1936: EE Université du Michigan**
- **1940: PhD Massachusetts Institute of Technology (MIT)**
- **1941-1972: Bell Laboratories**
- **Dès 1956: Professeur au MIT**



- **"*A Mathematical Theory of Communication*", 1948: base de la théorie de l'information**
- **"*Communications in the presence of noise*", 1949: Preuve du théorème d'échantillonage de Nyquist**
- **"*Programming a Computer for Playing Chess*", 1950: Minimax search**
- **…**

"A Mathematical Theory of Communication", 1948

- **Formalisation du schéma linéaire de communication**
- **Première apparition du mot "bit"**
- **Le canal sans bruit: Possibilité de comprimer jusqu'à H(X) bits. Algorithme pratique pour y arriver**
- **Le canal bruité. Possibilité de transmettre sans erreur jusqu'à la capacité C. Preuve théorique. Cite le code de Hamming comme exemple d'approche pratique.**

Reprinted with corrections from *The Bell System Technical Journal*,
Vol. 27, pp. 379–423, 623–656, July, October, 1948.

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.

If the number of messages in the set is finite then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely. As was pointed out by Hartley the most natural choice is the logarithmic function. Although this definition must be generalized considerably when we consider the influence of the statistics of the message and when we have a continuous range of messages, we will in all cases use an essentially logarithmic measure.

The logarithmic measure is more convenient for various reasons:

1. It is practically more useful. Parameters of engineering importance such as time, bandwidth, number of relays, etc., tend to vary linearly with the logarithm of the number of possibilities. For example, adding one relay to a group doubles the number of possible states of the relays. It adds 1 to the base 2 logarithm of this number. Doubling the time roughly squares the number of possible messages, or doubles the logarithm, etc.

2. It is nearer to our intuitive feeling as to the proper measure. This is closely related to (1) since we intuitively measures entities by linear comparison with common standards. One feels, for example, that two punched cards should have twice the capacity of one for information storage, and two identical channels twice the capacity of one for transmitting information.

3. It is mathematically more suitable. Many of the limiting operations are simple in terms of the logarithm but would require clumsy restatement in terms of the number of possibilities.

The choice of a logarithmic base corresponds to the choice of a unit for measuring information. If the base 2 is used the resulting units may be called binary digits, or more briefly *bits*, a word suggested by J. W. Tukey. A device with two stable positions, such as a relay or a flip-flop circuit, can store one bit of information. N such devices can store N bits, since the total number of possible states is 2^N and $\log_2 2^N = N$. If the base 10 is used the units may be called decimal digits. Since

$$\log_2 M = \log_{10} M / \log_{10} 2$$
$$= 3.32 \log_{10} M,$$

[1] Nyquist, H., "Certain Factors Affecting Telegraph Speed," *Bell System Technical Journal*, April 1924, p. 324; "Certain Topics in Telegraph Transmission Theory," *A.I.E.E. Trans.*, v. 47, April 1928, p. 617.
[2] Hartley, R. V. L., "Transmission of Information," *Bell System Technical Journal*, July 1928, p. 535.

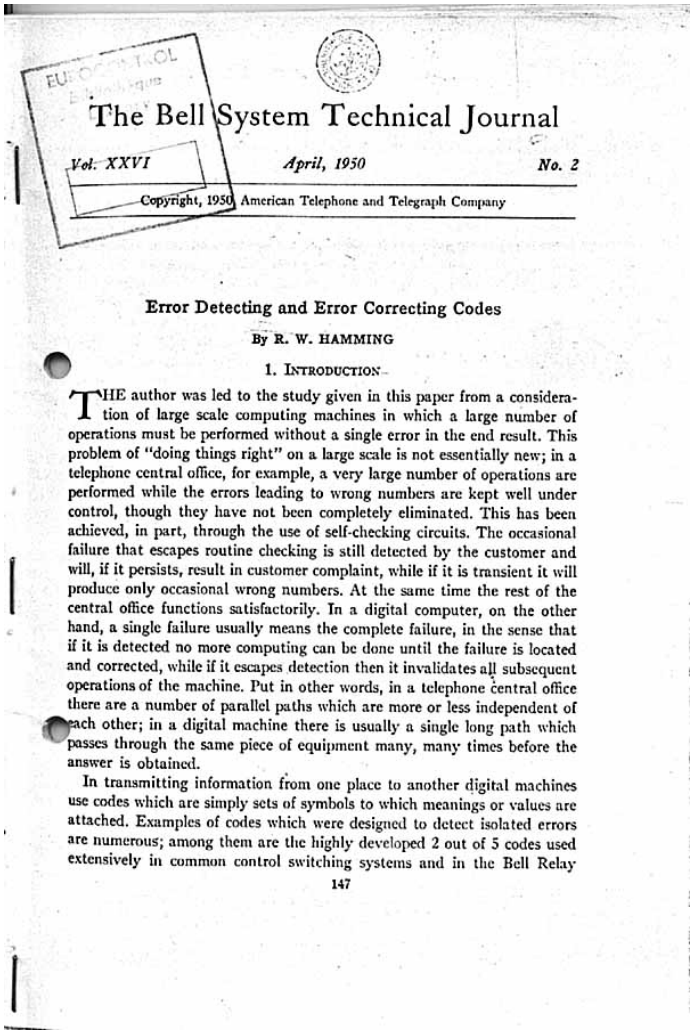
Hamming

- **Né le 11 février 1915, Chicago, IL, USA**
- **Mort le 7 janvier 1998, Monterey, CA, USA**
- **1937: BS Université de Chicago**
- **1939: MA Université du Nebraska**
- **1942: PhD Université d'Illinois**
- **1945: Projet Manhattan, Los Alamos**
- **1946-1976: Bell Laboratories**

- **"*Error detecting and error correcting codes*", 1950: base de la théorie du codage**
- **"*Numerical Methods for Scientists and Engineers*", McGraw-Hill, 1962: introduit notamment la fenêtre de Hamming**
- **IBM 650: langage de programation de haut niveau, format de nombres, overflow, …**
- **…**

"Error detecting and error correcting codes", 1950

- **Conséquence de la frustration de Hamming suite au crash de l'ordinateur à relais de Bells Labs un vendredi soir, détecté le lundi matin.**
- **Notion de parité, et de parité de bloc**
- **Codes optimaux pour détection d'une erreur, correction d'une erreur, détection de deux erreur et correction d'une, …**
- **Les articles de Shannon et Hamming sont disponibles dans la section "Liens utiles" du site web du cours.**



The Bell System Technical Journal

Vol. XXVI April, 1950 No. 2

Error Detecting and Error Correcting Codes

By R. W. HAMMING

1. Introduction

THE author was led to the study given in this paper from a consideration of large scale computing machines in which a large number of operations must be performed without a single error in the end result. This problem of "doing things right" on a large scale is not essentially new; in a telephone central office, for example, a very large number of operations are performed while the errors leading to wrong numbers are kept well under control, though they have not been completely eliminated. This has been achieved, in part, through the use of self-checking circuits. The occasional failure that escapes routine checking is still detected by the customer and will, if it persists, result in customer complaint, while if it is transient it will produce only occasional wrong numbers. At the same time the rest of the central office functions satisfactorily. In a digital computer, on the other hand, a single failure usually means the complete failure, in the sense that if it is detected no more computing can be done until the failure is located and corrected, while if it escapes detection then it invalidates all subsequent operations of the machine. Put in other words, in a telephone central office there are a number of parallel paths which are more or less independent of each other; in a digital machine there is usually a single long path which passes through the same piece of equipment many, many times before the answer is obtained.

In transmitting information from one place to another digital machines use codes which are simply sets of symbols to which meanings or values are attached. Examples of codes which were designed to detect isolated errors are numerous; among them are the highly developed 2 out of 5 codes used extensively in common control switching systems and in the Bell Relay

147

Code « 2 parmi 5 »

- **Code « 2 parmi 5 »: Dix groupes pour représenter les chiffres de 0 à 9 avec deux « 1 » et trois « 0 » chacun**
- **Permet de détecter une erreur**
- **Pas de la corriger**
- **Pas de détecter deux erreurs ou plus**

1	1	0	0	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1
0	1	1	0	0
0	1	0	1	0
0	1	0	0	1
0	0	1	1	0
0	0	1	0	1
0	0	0	1	1

Codes systématiques

- **Mots code composés de *n* bits**
- **Parmi ceux-ci, *m* sont associés à de l'information utile**
- **Les $k = n - m$ autres sont utilisés pour la détection et correction d'erreurs**

- **On définit la redondance $R = n/m$**

- **Pour un nombre d'erreurs détectables ou corrigibles, on cherche évidemment à avoir *R* le plus petit possible**

Détection d'une erreur

1	0	0	0	1	1	0	1	0	0	0
Information										Parité

- ***m = n-1* bits d'information**
- **1 bit de parité, de sorte que le nombre total de "1" dans le mot code soit pair.**
- **S'il y a une seule erreur dans le mot code, on la détecte puisque le nombre de "1" devient impair**
- **Redondance de ce code:**

$$R = \frac{n}{n-1} = 1 + \frac{1}{n-1}$$

- **Si P_e est la probabilité d'erreur sur un bit, avec $P_e << 1/n$, la probabilité d'erreur simple sur le mot est $n.P_e$ et la probabilité d'avoir plus d'une erreur est environ $n.(n-1).P_e^2 / 2$**

Code à répétition

- **Une méthode simple pour corriger des erreurs est la répétition. Si 0 et 1 sont codés par “000” et “111”, on peut corriger une erreur simple en prenant le symbole majoritaire entre 0 et 1**

- **En répétant 5 fois, on corrige 2 erreurs, en répétant *n=k+1* fois, on corrige *k/2* erreurs (k pair).**

- **La redondance de ce code est *R = n/m = n* puisque** $m = 1$

Code (8,4) pour corriger une erreur

- **On peut corriger une erreur avec plusieurs bits de parité vérifiant différents blocs de bits**
- **Code (8,4) avec 4 bits d'information (en noir) et 4 bits de parité (en rouge) selon les lignes et colonnes.**
- **Exemple 1: une erreur sur un bit d'information => deux erreur de parité**
- **Exemple 2: une erreur sur un bit de parité => une erreur de parité**
- **Plus d'une erreur: indétectable**

1	0	1
0	0	0
1	0	

1	0	1
0	1	0
1	0	

1	0	1
0	0	0
1	1	

Code (n,m) pour corriger une erreur

- **Quelle est le code le plus efficace pour corriger une erreur en transmettant *m* bits d'information ?**
- **Quel est le plus petit nombre *k* de bits de parité à ajouter ?**

- **Solution: les k bits de parité doivent être capable de coder les informations suivantes**
  - **pas d'erreur**
  - **la position d'une erreur parmi un des *m* bits d'information**
  - **la position d'une erreur parmi un des *k* bits de parité**

- **En tout, *k* bits permettent de coder 2^k informations différentes, d'où**

$$2^k \geq m + k + 1 = n + 1$$

Code (n,m) pour corriger une erreur

$$2^k \geq m + k + 1 = n + 1$$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
m	0	0	1	1	2	3	4	4	5	6	7	8	9	10	11	11
k	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	5

- **On cherchera évidemment à utiliser les codes pour lesquels le nombre de bits utiles est maximum pour un nombre de bits de parité donné.**
- **Le code (12,8) est également très utile: 1 byte d'information et ½ byte de parité.**
- **Le code (3,1) est équivallent au code à répétition vu précédemment.**

Code de Hamming (7,4)

- **On veut contrôler la parité de $7 = n = 2^k - 1 = 2^3 - 1$ bits.**

- **On veut que la vérification des k = 3 bits de parité nous donne la position de l'erreur unique, ou 0 s'il n'y en a pas**

- **Le 1er bit de parité contrôle les positions 1,3,5,7**
- **Le 2ème bit de parité contrôle les positions 2,3,6,7**
- **Le 3ème bit de parité contrôle les positions 4,5,6,7**

- **Pour que les bits de parité soient indépendants les uns des autres, on place les *k* bits de parité aux positions 2^0, 2^1, … 2^{k-1}. Dans ce cas, k=3 et donc on place les bits de parité aux positions 1, 2 et 4.**

Code de Hamming (7,4)

décimal	binaire	bit 1	2	3	4	5	6	7	mot code
0	0000	0	0	0	0	0	0	0	0000000
1	0001	1	1	0	1	0	0	1	1101001
2	0010	0	1	0	1	0	1	0	0101010
3	0011	1	0	0	0	0	1	1	1000011
4	0100	1	0	0	1	1	0	0	1001100
5	0101	0	1	0	0	1	0	1	0100101
6	0110	1	1	0	0	1	1	0	1100110
7	0111	0	0	0	1	1	1	1	0001111
8	1000	1	1	1	0	0	0	0	1110000
9	1001	0	0	1	1	0	0	1	0011001
10	1010	1	0	1	1	0	1	0	1011010
11	1011	0	1	1	0	0	1	1	0110011
12	1100	0	1	1	1	1	0	0	0111100
13	1101	1	0	1	0	1	0	1	1010101
14	1110	0	0	1	0	1	1	0	0010110
15	1111	1	1	1	1	1	1	1	1111111

Décodage de Hamming (7,4)

- **On veut transmettre 6 = 0110**

1	1	0	0	1	1	0

- **On recoit**

1	1	0	0	0	1	0

- **On vérifie les parités**
  - **Bits 1,3,5,7: 1000 => pas OK $s_0 = 1$**
  - **Bits 2,3,6,7: 1010 => OK $s_1 = 0$**
  - **Bits 4,5,6,7: 0010 => pas OK $s_2 = 1$**

- **La position de l'erreur est $s_2 s_1 s_0 = 101 = 5$. On a une erreur sur le 5ème bit**

Détection de 2 erreurs, correction d'une

- **On prend n'importe quel code correcteur d'une erreur précédent. On y ajoute 1 bit de parité.**

- **Il y a 3 cas possibles**

  - **Pas d'erreur. Toutes les parités sont correctes**

  - **Une erreur: Le dernier bit de parité génère une erreur. Les autres bits de parité donne la position de l'erreur. Si c'est 0, l'erreur est sur le bit de parité supplémentaire**

  - **Deux erreurs: Le dernier bit de parité ne génère pas d'erreur. Les autres bits indiquent une position d'erreur différente de 0, mais invérifiable.**

Distance de Hamming

- **Pour savoir la capacité détectrice ou correctrice d'un code, on définit la distance de Hamming entre deux mots code comme le nombre de bits différents entre ces mots**

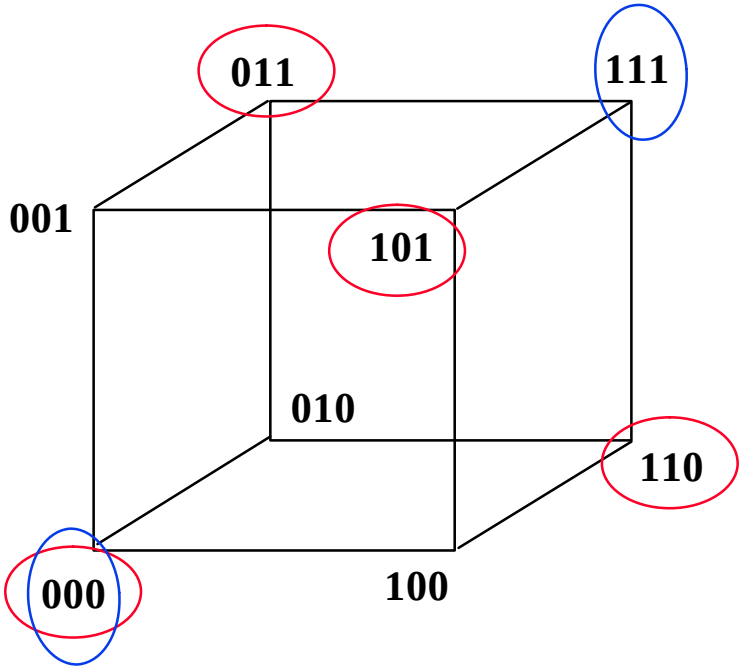
- **Exemple:**
  - **D(0111,0111) = 0**
  - **D(0110,0111) = 1**
  - **D(0110,1001) = 4**

- **La distance de Hamming D(C) d'un code est la distance minimale entre deux mots quelconques de ce code.**

- **Si on considère un hypercube de dimension n, on peut mapper ses sommets avec tous les mots de n bits. La distance de Hamming entre deux mots codes correspond au nombre de cotés qu'il faut parcourir pour passer d'un sommet à l'autre.**

Distance de Hamming

- Si on prend les 2^n mots codes possibles, on a $D(C) = 1$ et la seule garantie que l'on a est l'unicité des mots code
- Si on se restreint aux mots code pairs (en rouge), on a $D(C) = 2$ et on peut détecter une erreur
- Si on se restreint au code (3,1) (en bleu), on a $D(C) = 3$ et on peut corriger une erreur.
- Avec $D(C) = 4$, on corrige une erreur et en détecte deux
- $D(C) = 5$: correction d'erreurs doubles
- …



- **L'analogie de l'hypercube précédente revient à considérer que les mots codes sont extraits de l'espace vectoriel $(Z_2)^n$, l'ensemble des vecteurs contenant des éléments de $Z_2 = \{ 0,1 \}$**

- **Le code de Hamming (n,m) forme un sous espace vectoriel. Si a,b et c sont des symboles à coder tels que $a + b = c$, alors leurs mots codes correspondants respectent $x(a) + x(b) = x(c)$.**

- **En pratique, les opérations de codage et décodage peuvent être effectuées comme des multiplications matricielles.**

- **Pour trouver le mot code x (n,1) qui correspond au message m (m,1), on calcule $x = mG$ où G est la matrice génératrice (n,m).**

- **Pour décoder, on crée la matrice de parité H (n,k) telle que pour tout mot du code x on aie $H.x^T = 0$.**

- **Les matrices génératrices et de parité sont liées par $H.G^T = 0$. Dès lors, un message m encodé par G sera correctement décodé par H. En effet**

$$H(mG)^T = H(m^T G^T) = (HG^T)m^T = 0$$

- **Si on choisit $G = [\, I_m \; A \,]$ avec I_m la matrice (m,m) unité et A une matrice (k,m) quelconque, on trouve $H = [\, A^T \; I_k \,]$. En effet,**

$$HG^T = A^T I_m + I_k A^T = A^T + A^T = 0$$

- **La dernière égalité résultant de ce que l'on travaille modulo 2.**

En pratique

- **On considère de nouveau le code de Hamming (7,4). Les lignes de la matrice de parité *H* correspondent aux positions que chaque bit de parité contrôle.**

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

- **On note que les colonnes de *H* consistent simplement à compter de 1 à 7 en binaire.**
- **Pour trouver *G*, on commence par mettre *H* sous la forme *H'* = [A^T I_k] en groupant les colonnes 1,2 et 4 à droite (1? 7, 2? 6, 4? 5). On obtient**

$$H' = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

En pratique (suite)

- **On a alors $G' = [\, I_m \; A \,]$, soit**

$$G' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- **Qui est la matrice génératrice qui correspond à *H'*. Pour trouver celle correspondant à *H*, il suffit de permuter de nouveau les colonnes suivant la permutation inverse de la précédente, soit (7? 1, 6? 2, 5? 4).**

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x(8) \\ x(4) \\ x(2) \\ x(1) \end{bmatrix}$$

- **Si on veut coder le symbole 6. On a $m = 0110$, $x = mG = 1100110$, et $Hx^T = 000$**
- **Si on recoit le code erroné 1100010, on trouve $Hx^T = 101 = 5$**

Application: RAID

- **RAID = Redundant Array of Inexpensive Disks**
- **But: créer un super disque dur en en groupant plusieurs**
  - **Plus de capacité**
  - **Accès plus rapide**
  - **Redondance, correction d'erreurs et résistance au crash d'un des disques**
- **RAID 0: on lit et écrit sur les disques en parallèle, ce qui augmente capacité et vitesse d'accès, mais pas la résistance au crash**
- **RAID 1: comme RAID 0 pour tous les disques sauf 1, qui stocke la parité calculée à partir des autres disques. Détection d'erreur mais pas correction. Résistance au crash d'un disque, qui peut être reconstruit à partir des autres**

- **RAID 2: Plus de disques de parité pour créer un code de Hamming**

- **RAID 5: Simple parité, mais stockée alternativement sur les différents disques. Evite le bottleneck d'accès au disque de parité**