Ecole Polytechnique Fédérale de Lausanne

Sciences de l'information

Série 10

Olivier Cloux

10.1

Rappelons que $card(X)$ représente le nombre d'éléments dans X (donc $card(A)$ le nombre de symboles de l'alphabet et $card(C)$ le nombre de mots de codes par bloc) ; de plus, n représente la longueur des mots de code, k se calcule par $log_{card(A)}\big(card(C)\big)$, et d_{min} est la plus petite distance de Hamming.

1. $\boxed{n = 3}$
   $A = \{0, 1\} \rightarrow card(A) = 2,\ card(C) = 4, \rightarrow \log_2(4) = \boxed{k = 2}$
   $\boxed{d_{min} = 2}$
2. $\boxed{n = 5}$
   $A = \{0, 1\} \rightarrow card(A) = 2,\ card(C) = 4, \rightarrow \log_2(4) = \boxed{k = 2}$
   $\boxed{d_{min} = 3}$
3. $\boxed{n = 6}$
   $A = \{0, 1, 2\} \rightarrow card(A) = 3,\ card(C) = 3, \rightarrow \log_3(3) = \boxed{k = 1}$
   $\boxed{d_{min} = 5}$

10.2

1. $\boxed{A = \{0, 1, 2, 3\}}$
2. $r = \frac{k}{n} = \frac{log_{card(A)}\big(card(C)\big)}{n} = \frac{\log_4(5)}{5} \simeq \frac{1.161}{5} \simeq \boxed{0.232 = r}$
3. $\boxed{d_{min} = 3}$ (par exemple entre 01301 et 21310). Cette distance minimale doit répondre à

   $$d_{min}(C) < n(1 - r) + 1 = n - k + 1 = \text{ borne de Singleton}$$

   Ainsi, la borne de Singleton est : $n - k + 1 = 5 - \log_4(5) + 1 = 6 - \log_4(5) \simeq \boxed{4.839}$
4. Le décodeur d'un code C est capable de corriger tous les effacements de poids $\leq p_1$ si et seulement si $p_1 < d_{min}(C)$. Ainsi, ce code sera capable de corriger tout effacement de poids $d_{min} - 1 = 3 - 2 = \boxed{2 = p_1}$

5. La valeur maximale est $\boxed{q_1 = 4}$. En effet, les mots de code 0????, 1????, 3???? peuvent être corrigés en respectivement 01301, 13012, 30122. p_2 est plus grand que p_1, car nous savons quels bits sont faux, et dans des cas précis. Il 4 est un chiffre irréaliste pour corriger tous les effacements, car certaines erreurs, même de 3, ne sont plus corrigibles, pour cause de confusion entre deux mots.

6. Le décodeur d'un code C est capable de détecter toutes les erreurs de poids $\leq p_2$ si et seulement si $p_2 < d_{min}(C)$. Ainsi, ce code sera capable de détecter toutes les erreurs de poids $d_{min} - 1 = 3 - 2 = \boxed{2 = p_2}$

7. La valeur $\boxed{q_2 = 5}$; par exemple, si nous recevons le mot de code 00000, nous détectons immédiatement qu'il s'agit d'une erreur, car cela ne correspond à aucun de nos mots de code. Ce n'est pas la même que p_2, car comme nous p_2 doit fonctionner pour toute erreur. Par exemple, si nous envoyons 01301 mais que par hasard, une erreur parfaite de poids d_{min} arrive, ce mot peut se changer en 21310. Raison pour laquelle q_2 est de 5, alors que p_2 est plus petit que d_{min}

8. Le décodeur d'un code C est capable de corriger toutes les erreurs de poids $\leq p_3$ si et seulement si $p_3 < \frac{d_{min}(C)}{2} = \frac{3}{2} = 1.5$. Le code ne pourra donc corriger que toutes les erreurs de poids $\boxed{p_3 = 1}$.

9. Il est imposé que les mots de code doivent être de longueur 5 ; comme $r = \frac{k}{n} = \frac{k}{5}$, nous devons aussi nous assurer que $k = \log_{card(A)}\big(card(C)\big) = 1$, donc avoir autant de mots de codes que de symboles. De plus, comme il doit corriger toutes les erreurs de poids ≤ 2, il faut $2 \leq \frac{d_{min}(C)}{2}$, ce qui nous amène à trouver que $d_{min} = 5$. Le code $\boxed{C = \{00000, 11111, 22222, 33333, 44444\}}$ semble remplir toutes les conditions : La longueur des mots est de 5, il y a autant de mots que de symboles (donc $k = 1$ et $r = \frac{1}{5}$). De plus, comme ils sont tous exactement différents, $d_{min} = 5$ et, par le théorème 11.4, il peut corriger les erreurs de poids ≤ 2

10.3

1. Nous parlons d'un code binaire de longueur 2^k; nous avons donc un mot de code y, binaire, de longueur n; trouver le nombre de mots de codes qui sont à une distance t de y, revient à combiner la distance parmi la longueur. Une combinaison de t parmi n se calcule avec le coefficient binomial $C_n^t = \binom{n}{t} = \frac{n!}{t!(n-t)!}$

2. Nous connaissons le binôme de Newton $\left((X+Y)^n = \sum_{i=0}^{n} \binom{n}{i} X^{n-i}Y^i\right)$, ce qui nous montre que $2^n = \sum_{i=0}^{n} \binom{n}{i}$. Ainsi, selon la donnée, nous aurons 2^k mots de code (donc $\sum_{i=0}^{n} \binom{n}{i}$ mots de codes). Nous savons également, par le dernier exercice, qu'il y a $\binom{x}{e}$ mots d'une longueur donnée qui sont à une distance e de x. Comme nous voulons chercher pour tous les t inférieurs à e, nous trouvons que le nombre de mots de code dans $\boxed{B(x,e) = \sum_{t=0}^{e} \binom{n}{t}}$

3. Nous savons que la distance de Hamming respecte la symétrie et l'inégalité triangulaire. De plus, comme il est dit que l'intersection des deux ensembles n'est pas vide, cela signifie qu'il existe au moins un mot y valide, tel que $d(x_1, y) \leq e$ et $d(x_2, y) \leq e$. Ensuite, nous savons, pas l'inégalité triangulaire, que $d(x_1, x_2) \leq d(x_1, y) + d(y, x_2) \leq e + e = 2e$. Donc $d(x_1, x_2) \leq 2e$ Notons que nous avons utilisé que $d(x_2, y) = d(y, x_2)$, car cette opération conserve la symétrie.

4. Notre code peut corriger toutes les erreurs de poids $\leq e$, ce qui implique que

$$e \leq \frac{d_{min}(C)}{2} \tag{1}$$

De plus, nous venons de prouver que

$$B(x_1, e) \cap B(x_2, e) \neq \emptyset \longrightarrow d(x_1, x_2) \leq 2e \tag{2}$$

Par contraposition, nous pouvons donc placer que

$$d(x_1, x_2) > 2e \longrightarrow B(x_1, e) \cap B(x_2, e) = \emptyset \tag{3}$$

Finalement, comme x_1 et x_2 sont des mots valides distincts de C, il s'ensuit que la distance entre les deux est forcément $\geq d_{min}(C)$ (par définition). Donc $d(x_1, x_2) \geq d_{min}(C)$ et, par (1), $\geq 2e$; comme nous l'avons trouvé dans (3), cette condition est suffisante à montrer que l'intersection de ces ensembles est nulle.

5. Comme les blocs sont de longueur n, il y a un total de 2^n mots de codes maximum. Nous avons aussi montré que le nombre de mots de codes dans $B(x, e)$ pour un x donné est de $\sum_{t=0}^{e} \binom{n}{t}$, et nous savons que le notre code comprend 2^k mots de codes. La partie droite de l'inéquation représente donc tous les mots de codes de C multiplié par le nombre de mots "faux" par mot de code. Cela représente donc l'ensemble des mots qui peuvent être corrigés si besoin (qu'ils soient faux ou non). Mais comme nous savons qu'il n'y a pas de "doublons" (car $B(x_1, e) \cap B(x_2, e) = \emptyset$), cela signifie qu'il ne peut pas y avoir plus de mots corrigibles que de mots possibles. Il y a donc bien moins de mots qui peuvent être corrigés que de mots possibles, donc $2^n \geq 2^k \sum_{t=0}^{e} \binom{n}{t}$.