L3 Mathématiques pour l'enseignement — Géométrie et arithmétique avec des logiciels
Université Lyon 1

Projet X : Codes correcteurs

Quand on utilise un canal de communication pour transmettre un message (radio, sms, internet...) les données sont parfois altérées et contiennent des erreurs de transmission (les causes peuvent être diverses, un mauvais réseau, du matériel défectueux etc.). Il a fallu introduire des algorithmes pour détecter les éventuelles erreurs d'une part, et les corriger d'autre part. Le but de ce projet (Python) est d'introduire la notion de codes correcteurs.

1. Détection d'une erreur

Définition. On appelle *message* une liste de 0 et de 1, par exemple $m = [0, 1, 1, 0, 0]$.

1. Codez un algorithme `ModifMessage` qui prend en argument un message m et qui renvoie $m + [d]$, où d est la somme des éléments de m modulo 2.

   Exemples : Si $m = [0, 0, 1]$ alors on renvoie $[0, 0, 1, 1]$ (puisque $1 = 0 + 0 + 1 \mod 2$). Si $m = [1, 1, 1, 0]$ on renvoie $[1, 1, 1, 0, 1]$ (car $1 = 1 + 1 + 1 + 0 \mod 2$).
2. Un utilisateur veut envoyer le message m. On envoie à la place le message m' après avoir appliqué `ModifMessage`. Que se passe-t-il si lors de la transmission on modifie un (unique) élément de m' ? (soit on transforme un 1 en 0, soit on transforme un 0 en 1). Que se passe-t-il si deux éléments sont modifiés ? Ecrire un programme `TestErreur` qui prend en entrée un message (qu'on suppose de la forme m' comme ci-dessus) et qui renvoie *Erreur détectée* lorsque vous arrivez à détecter une erreur.
3. Que peut-on conclure si `TestErreur` ne renvoie pas *Erreur détectée* ?

2. Codes correcteurs

Le problème de l'algorithme `TestErreur` *est qu'il ne permet pas de corriger une erreur détectée. Dans cette partie, on propose une méthode pour pouvoir corriger le message et retrouver le message initial s'il y a peu d'erreurs. Une méthode classique est d'introduire de la redondance dans le message.*

Définition. Pour tout entier $n \geq 1$, on note M_n l'ensemble des messages de longueur n. Un code correcteur de paramètre (k, n) est une application injective $f : M_k \to M_n$ appelée encodage. On dit que k est la *dimension* du code et que n est sa *longueur*.

1. Considérons l'encodage $\phi : M_1 \to M_3$ défini par $\phi([0]) = [0, 0, 0]$ et $\phi([1]) = [1, 1, 1]$. D'après la définition précédente, quelles sont la dimension et la longueur de ϕ ?
2. Quand on transmet un message ainsi encodé, quel est le nombre maximum e d'erreurs qu'on peut détecter ?
3. Ecrivez un programme Python `CorrigeCode` qui prend en entrée un message de longueur 3 contenant au plus e erreur(s) et qui renvoie le message initial. Par exemple, si on reçoit $[1, 1, 0]$ en entrée, on retournera $[1, 1, 1]$.

3. Distance de Hamming

On va maintenant introduire une notion de distance entre les mots. Intuitivement, si on reçoit par exemple par sms le mot "Dimznche", pour le corriger on cherche le mot "le plus proche" qui existe, ici "Dimanche".

Définition Soit k un entier ≥ 1 et m, m' deux messages de M_k. On appelle *distance de Hamming* entre m et m', et on note $d(m, m')$ le nombre de i tels que $m[i] \neq m'[i]$. On appelle *poids de Hamming* de m, et on note $w(m)$ le nombre d'éléments non nuls de m.

Définition Soient $m, m' \in M_k$. On définit la somme $s = m + m'$ en posant $s[i] = m[i] + m'[i] \mod 2$ (ce sorte que les éléments de s ne soient que des 0 ou des 1).

1. Implémentez en Python une fonction `DistanceH` prenant en entrée deux messages a et b de même longueur et renvoyant $d(a, b)$.
2. Implémentez un fonction `PoidsH` prenant en entrée un message m et renvoyant $w(m)$.
3. Implémentez une fonction `Somme` qui prend en entrée deux messages a et b de même longueur et qui renvoie leur somme $s = a + b \in M_k$ définie comme ci-dessus.
4. Montrez que pour tous messages $a, b \in M_k$ on a $d(a, b) = w(a + b)$.
5. Montrez que d définit une distance sur M_k (il faut donc vérifier que d est symétrique, que $d(a, b) = 0$ si et seulement si $a = b$ et que d vérifie l'inégalité triangulaire $d(a, b) \leq d(a, c) + d(b, c)$ pour tout messages $a, b, c \in M_k$).

4. Capacité de détection et de correction

Soit $\phi : M_k \to M_n$ un code d'image $C = \phi(M_k)$. On appelle *capacité de détection* de ϕ, et on note e_d le plus grand entier tel qu'il soit toujours possible de détecter au moins e_d erreurs dans un message transmis par ϕ. On appelle *capacité de correction* de ϕ et on note e_c le plus grand entier tel qu'il soit toujours possible de corriger au moins e_c erreurs dans un message transmis par ϕ. On appelle *distance minimale* de ϕ et on note d_ϕ la plus petite distance de Hamming non nulle entre deux mots $m, m' \in C$.

1. Montrez que $e_d = d_\phi - 1$.
2. Montrez que $e_c = \lfloor e_d/2 \rfloor$.
3. On considère le code de répétition $\phi : M_3 \to M_{15}$ qui à un message $m \in M_3$ associe le message obtenu en répétant chaque élément de m cinq fois. Par exemple

$$\phi([1, 0, 1]) = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1].$$

   Déterminez d_ϕ, e_d et e_c dans ce cas.
4. Implémentez la fonction ϕ en Python.
5. Ecrivez un algorithme `DetecteErreurPhi` qui prend en entrée en élément de $m \in M_{15}$ et qui retourne *Erreur* lorsque $m \notin C$.
6. Ecrivez un algorithme `CorrigePhi` qui prend en entrée un élément $m \in M_{15}$ et qui renvoie l'élément $m' \in C$ qui minimise la distance $d(m, m')$. On justifiera soigneusement la construction de m'.