Quand les bits se trompent

Les codes correcteurs d'erreurs

Mise en situation. Toutes les données numériques sont écrites avec des **bits** (0 ou 1). Mais un bit peut changer par accident : chaleur, usure d'un composant, rayonnement, parasite sur un câble ou une onde radio… Un 0 devient 1, ou un 1 devient 0.
Les IA sont particulièrement concernées : elles sont entraînées pendant des semaines sur des milliers de processeurs qui échangent et stockent d'énormes quantités de bits. Comment des machines qui se trompent peuvent-elles produire des résultats fiables ? Grâce aux **codes correcteurs d'erreurs**, qui ajoutent des bits supplémentaires pour repérer, et même corriger, les bits qui ont changé.

Remarque

En 2024, Meta a publié le bilan de l'entraînement de son modèle d'IA Llama 3 sur 16 384 processeurs graphiques pendant 54 jours : 419 interruptions imprévues, soit environ une toutes les trois heures, dont 72 dues à des pannes de la mémoire des processeurs graphiques. Meta signale aussi des cas de « corruption silencieuse de données » : des bits qui changent sans que personne ne s'en aperçoive.

1. Répéter pour corriger

Définition

Le **code à répétition** envoie chaque bit trois fois : 0 devient `000` et 1 devient `111`. À la réception, on lit les bits par groupes de trois et on garde le bit **majoritaire** : `010` est lu 0, `110` est lu 1.

Exemple

Pour envoyer `10`, on transmet `111 000`. Si l'on reçoit `101 000`, le premier groupe contient deux 1 : il est lu 1. Le message lu est `10` : l'erreur a été corrigée.

▶ À vous de jouer !

1. Coder le message `101` avec le code à répétition.
2. On reçoit `110 000 101`. Quel message lit-on ? Combien d'erreurs ont été corrigées ?
3. On envoie `111` (le bit 1) et deux bits du groupe changent. Que lit-on ? Conclure.
4. Combien de bits faut-il envoyer pour transmettre un message de 4 bits ? Quel est l'inconvénient de ce code ?

Coup de pouce

Dans chaque groupe de trois bits, comptez les 1 : s'il y en a au moins deux, le bit lu est 1.

2. Le bit de parité

Définition

On ajoute à la fin d'un message un **bit de parité** choisi pour que le nombre total de 1 soit **pair** : si le message contient un nombre pair de 1, on ajoute 0 ; sinon, on ajoute 1.
À la réception, on compte les 1 : si leur nombre est impair, une erreur s'est produite.

Exemple

Le message `1000001` contient deux 1 (nombre pair) : on ajoute 0 et on envoie `1000001 0`.

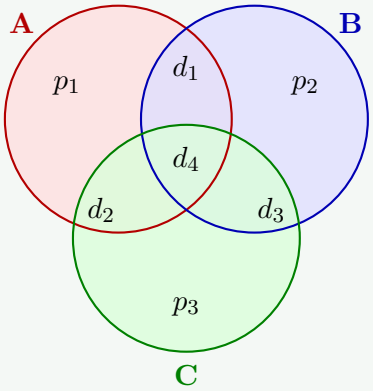
▶ À vous de jouer !

1. Ajouter le bit de parité aux messages `1011001` et `1110000`.
2. On reçoit `1100101 1`. Y a-t-il une erreur ? Peut-on savoir quel bit est faux ?
3. On envoie `1000001 0` et deux bits changent : on reçoit `1100000 0`. L'erreur est-elle détectée ?
4. Combien de bits faut-il envoyer pour un message de 7 bits ? Comparer avec le code à répétition.

3. Le code de Hamming : corriger avec peu de bits

Définition

Le **code de Hamming** (Richard Hamming, 1950) protège un message de 4 bits $d_1d_2d_3d_4$ en lui ajoutant 3 bits de contrôle $p_1p_2p_3$. On les place dans trois cercles A, B et C :



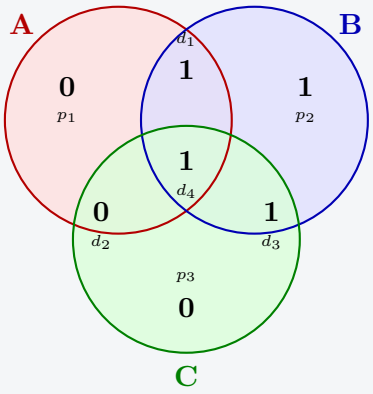
On choisit p_1, p_2 et p_3 pour que **chaque cercle contienne un nombre pair de 1**. On envoie les 7 bits $d_1d_2d_3d_4\,p_1p_2p_3$.
À la réception, on compte les 1 dans chaque cercle. Si un seul bit a changé, les cercles « faux » (avec un nombre impair de 1) indiquent sa place : c'est le bit qui est dans tous les cercles faux, et dans aucun cercle juste.

Exemple

Pour coder `1011` : $d_1 = 1$, $d_2 = 0$, $d_3 = 1$, $d_4 = 1$.

- Cercle A (d_1, d_2, d_4, p_1) : $1 + 0 + 1 = 2$ est pair, donc $p_1 = 0$.
- Cercle B (d_1, d_3, d_4, p_2) : $1 + 1 + 1 = 3$ est impair, donc $p_2 = 1$.
- Cercle C (d_2, d_3, d_4, p_3) : $0 + 1 + 1 = 2$ est pair, donc $p_3 = 0$.

On envoie `1011 010`.



▶ À vous de jouer !

1. Coder le message `0110` avec le code de Hamming.
2. On reçoit `1001 010`. Placer les bits dans les cercles, puis compter les 1 de chaque cercle. Quels cercles sont faux ? Quel bit a changé ? Quel était le message envoyé ?
3. On reçoit `0110 100`. Quel bit a changé ? Le message `0110` est-il correct ?
4. Combien de bits envoie-t-on pour 4 bits utiles ? Comparer avec le code à répétition.

Coup de pouce

Les bits reçus sont, dans l'ordre, $d_1 d_2 d_3 d_4\, p_1 p_2 p_3$. Le bit faux est dans tous les cercles faux, et dans aucun cercle juste.

4. Programmer le code de Hamming en Python

Exemple

En Python, `x % 2` vaut 0 si `x` est pair et 1 si `x` est impair.

```
def coder(d1, d2, d3, d4):
    p1 = (d1 + d2 + d4) % 2
    p2 = (d1 + d3 + d4) % 2
    p3 = (d2 + d3 + d4) % 2
    return [d1, d2, d3, d4, p1, p2, p3]

def cercles_faux(m):
    d1, d2, d3, d4, p1, p2, p3 = m
    A = (d1 + d2 + d4 + p1) % 2
    B = (d1 + d3 + d4 + p2) % 2
    C = (d2 + d3 + d4 + p3) % 2
    return [A, B, C]
```

La fonction `cercles_faux` renvoie 1 pour un cercle faux et 0 pour un cercle juste.

▶ À vous de jouer !

1. Que renvoie `coder(1, 0, 1, 1)` ?
2. Que renvoie `cercles_faux([1, 0, 0, 1, 0, 1, 0])` ? Interpréter.
3. Que renvoie `cercles_faux([1, 0, 1, 1, 0, 1, 0])` ? Pourquoi ?
4. Écrire une fonction `inverser(m, i)` qui change le bit d'indice `i` de la liste `m` (0 devient 1 et 1 devient 0), puis renvoie `m`.

Coup de pouce

Si `b` vaut 0 ou 1, alors `1 - b` vaut l'autre valeur. Dans une liste, le premier élément a l'indice 0.

5. L'IA et la correction d'erreurs

Remarque

- **Dans le matériel.** Les mémoires des serveurs qui entraînent les IA utilisent des codes de la famille de Hamming (mémoires dites ECC) : elles corrigent seules les bits qui changent.
- **Une IA pour corriger les erreurs.** Les ordinateurs quantiques font énormément d'erreurs. En 2024, Google DeepMind a publié dans la revue *Nature* AlphaQubit, une IA qui repère ces erreurs à partir de tests de cohérence, comme nos cercles. Elle utilise les **Transformers**, la même architecture que les grands modèles de langage.
- **Dans la vie courante.** Un QR code reste lisible même si une partie de sa surface est abîmée, jusqu'à environ 30 % pour le niveau de correction le plus élevé.

▶ À vous de jouer !

Le français aussi contient de la **redondance** : il y a plus de lettres que nécessaire.

1. Lire la phrase suivante, dont on a retiré toutes les voyelles : « `L_s c_d_s c_rr_ct__rs pr_t_g_nt l_s d_nn__s.` »
2. Lire la phrase « Les asssistants d'IA cmprennent nos fotes de frape. » Comment un assistant d'IA peut-il deviner les mots voulus ?

6. Pour aller plus loin

★ Question étoilée

1. On envoie le mot de Hamming `1011 010`, et deux bits changent : d_1 et d_2. Quel mot reçoit-on ? Quels cercles sont faux ? Que fait la correction ? Conclure.
2. Pour envoyer 400 bits utiles, combien de bits faut-il transmettre avec le code à répétition ? avec le code de Hamming ? Quelle part des bits transmis est utile dans chaque cas ?

À retenir

- Un bit peut changer par accident. Les IA, entraînées sur des milliers de processeurs, y sont très exposées.
- Un code correcteur ajoute de la **redondance** : des bits supplémentaires qui permettent de détecter, voire de corriger, les erreurs.
- Le **bit de parité** détecte une erreur sans la corriger ; le **code à répétition** corrige mais triple la longueur ; le **code de Hamming** corrige une erreur avec seulement 3 bits de contrôle pour 4 bits utiles.
- Des IA comme AlphaQubit aident à corriger les erreurs des ordinateurs quantiques, et la redondance de la langue permet aux assistants de comprendre nos fautes de frappe.