Codes détecteurs et correcteurs d'erreurs

Lorsque des données numériques sont stockées ou transmises, des perturbations (par exemple électromagnétiques) peuvent les endommager. Les codes détecteurs et correcteurs d'erreurs permettent, dans une certaine mesure, de détecter si les données ont été altérées et si c'est le cas, de reconstituer les données d'origine par un mécanisme de correction. Ces codes sont par exemple utilisés dans les supports de stockage numériques que sont les CD, DVD et disques Blu-ray. Dans cette fiche, nous en présentons les principes généraux de fonctionnement et quelques exemples simples.

1 Transmission et stockage de l'information numérique

Information numérique

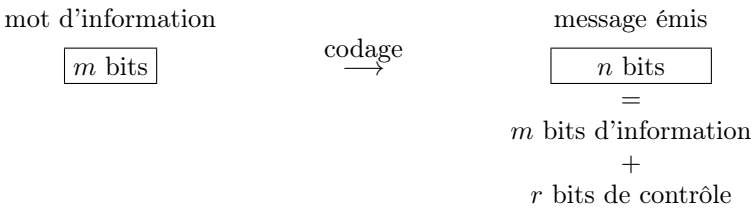
- Les ordinateurs ne connaissent que les chiffres 0 et 1. Cela suffit pour toutes les tâches qu'ils effectuent : stocker, traiter et transmettre nombres, textes, images, sons, vidéos, etc.
- On dit que les ordinateurs travaillent en *binaire*, et les chiffres 0 et 1 s'appellent des *bits* (contraction de « **bi**nary digi**ts** », c'est-à-dire « chiffres binaires » en Anglais).
- L'information est découpée en blocs de 0 et de 1 de longueur fixe, appelés des *mots binaires*.
- La raison pour laquelle les ordinateurs travaillent en binaire est liée au fonctionnement de leurs composants physiques : un transistor ou un condensateur possèdent deux états stables (activé/désactivé ou chargé/déchargé), respectivement représentés par 1 et par 0.

Les canaux de transmission et supports de stockage sont forcément imparfaits !

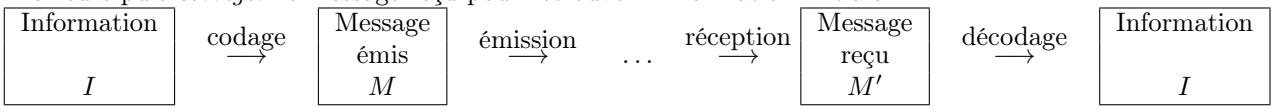
- Lors du transfert et/ou du stockage d'information, des erreurs peuvent survenir : de temps en temps, un 1 devient un 0 ou vice-versa (pour simplifier, on ne s'occupera pas des bits éventuellement perdus ou ajoutés).
- Le rôle des codes détecteurs et correcteurs d'erreurs est de détecter les erreurs de transmission et/ou de stockage et de les corriger. Ils utilisent pour cela une stratégie de *consolidation*, qui consiste à ajouter de l'information redondante.
- On parlera seulement dans la suite de transmission d'information, mais tout s'applique également au cas du stockage.

Codage et décodage

- Avant émission : le *codage* consiste à consolider l'information par ajout de *bits de contrôle*. Ces bits de contrôle sont *calculés* à partir des bits d'information : on considère qu'ils ne contiennent pas une information nouvelle, puisqu'ils sont entièrement construits à partir de l'information initiale. C'est pourquoi on parle de bits « redondants ». Le mot binaire obtenu constitue le *message émis*.



- Après réception : le *décodage* consiste, à partir du message reçu et en utilisant les bits de contrôle, à *détecter* les erreurs puis *corriger* le message reçu pour retrouver l'information initiale.



Efficacité des codes

On mesure l'efficacité des codes à l'aide de deux paramètres :

- Le nombre de bits ajoutés $r = n - m$, qu'on appelle la *redondance* du code
- Le ratio entre le nombre de bits utiles (les bits d'information) et le nombre de bits total $\frac{m}{n}$, qu'on appelle le *rendement* du code, et qui est un nombre toujours < 1

Pour le choix d'un code : on commence par fixer la longueur m des mots d'information. On veut un code qui détecte/corrige bien les erreurs, pour cela, il faut ajouter beaucoup de bits de contrôle, donc on a besoin d'une redondance élevée (r grand). Mais on veut aussi un code le plus économique possible, en vue d'une transmission rapide et/ou courte, donc on a besoin d'un rendement élevé, ce qui se traduit par un r petit.
Par conséquent, on doit faire un compromis entre l'efficacité de la détection/correction et l'efficacité de la transmission, c'est-à-dire entre la redondance et le rendement. L'utilisation de méthodes mathématiques sophistiquées (par exemple *l'algèbre sur des corps finis de polynômes*) permet de fabriquer des codes efficaces sur ces deux points.

2 Exemples

2.1 Code de parité

Description du code

- Lors de la transmission de caractères de texte, si on utilise le code ASCII, chaque caractère occupe 8 bits. Par exemple, le mot « HELLO » est représenté par 10010000 10001011 10011001 10011001 10011111.
- Chaque caractère est codé sur 7 bits plus 1 bit de parité (bit de contrôle) en général placé avant les bits d'information
- Le bit de parité est calculé de telle sorte que le nombre total de 1 soit toujours pair (par exemple).
- Mot d'information : 100 1101 $\rightsquigarrow$ Mot de code : 0 100 1101
- Mot d'information : 110 0111 $\rightsquigarrow$ Mot de code : 1 110 0111

Paramètres du code

- Longueur des mots d'information : $m = 7$
- Longueur des messages émis : $n = 8$
- Redondance : $r = n - m = 1$ (c'est à dire 1 bit de contrôle)
- Rendement : $\frac{7}{8} = 0,875$

Détection et correction

N'importe quel mot de 8 bits comportant un nombre pair de 1 est susceptible d'être le message émis, et n'importe quel mot de 8 bits est susceptible d'être le message reçu, selon les perturbations subies pendant la transmission. On suppose par exemple que le message émis est 1110 1000, et on donne quelques exemples de messages reçus possibles.

	Message envoyé	Message reçu	Contrôle	Correction proposée
(a)	1110 1000	1110 1000	OK	
(b)	1110 1000	1110 1001	anomalie	?
(c)	1110 1000	0100 1100	anomalie	?
(d)	1110 1000	0011 0000	OK	

- On voit sur ce tableau que le code de parité permet de détecter une anomalie lorsqu'il y a un nombre impair de bits erronés. En effet, dans ce cas, le message reçu ne peut pas être celui qui a été envoyé, puisqu'il comporte un nombre impair de 1 (cas (b) et (c)). Lorsqu'il y a un nombre pair de bits erronés (cas (d)), le message reçu est bien un message émis potentiel : on ne peut pas s'apercevoir qu'il y a eu une anomalie.
- Le code de parité ne permet pas de corriger les erreurs : lorsqu'une anomalie est détectée, on sait qu'il faut modifier un bit (ou peut-être trois, cinq ou même sept bits) pour retrouver le message émis et rétablir la parité du nombre de 1, mais on n'a aucun moyen de savoir le(s)quel(s).

2.2 Code par répétition

Description du code

- On répète 3 fois chaque bit d'information. Les mots d'information ont seulement 1 bit, et les messages émis 3 bits :

mot d'info	message émis
0	000
1	111

Paramètres du code

- Longueur des mots d'information : $m = 1$
- Longueur des messages émis : $n = 3$
- Redondance : $r = n - m = 2$ (c'est à dire 2 bits de contrôle)
- Rendement : $\frac{1}{3} \simeq 0,333$.

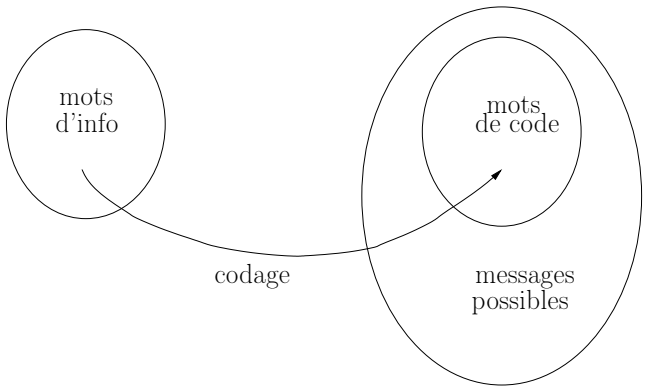
Détection et correction
Les seuls messages émis possibles sont 000 et 111, et n'importe quel mot de 3 bits est susceptible d'être le message reçu, selon les perturbations subies pendant la transmission. On traite le cas où le message émis est 000, le cas 111 est évidemment symétrique. On considère cette fois tous les messages reçus possibles (il n'y en a que 8).

	Message émis	Message reçu	Contrôle	Correction proposée
(a)	000	000	OK	
(b)	000	100	anomalie	000
(c)	000	010	anomalie	000
(d)	000	001	anomalie	000
(e)	000	110	anomalie	111
(f)	000	101	anomalie	111
(g)	000	011	anomalie	111
(h)	000	111	OK	

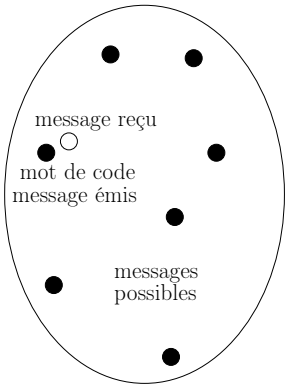
- On voit sur ce tableau que le code par répétition permet de détecter une anomalie lorsqu'il y a un ou deux bits erronés (le message reçu ne peut pas être le message émis - cas (b) à (g)), mais pas lorsqu'il y en a trois (cas (h)). En effet, dans ce dernier cas, le message reçu est bien un message émis potentiel.
- On corrige en remplaçant un message reçu reconnu erroné par le message émis potentiel *le plus proche* (c'est-à-dire avec le moins de bits différents), car c'est la correction qui a la plus grande probabilité d'être correcte (voir Section 4).
- Par conséquent, le codage par répétition permet de corriger correctement une erreur portant sur un seul bit (cas (b), (c) et (d)), mais ne permet pas de corriger correctement une erreur portant sur deux bits (cas (e), (f) et (g) - en fait on peut constater que dans ce cas, la correction proposée est systématiquement incorrecte).

3 Principe général de la détection et la correction des erreurs

Messages possibles et mots de code
- Les messages potentiellement émis, qu'on appelle les *mots de code*, constituent seulement une petite partie de tous les messages potentiellement reçus (avec ou sans erreurs), qu'on appelle les *messages possibles*.
- En effet, tous les mots de n bits sont des messages possibles, tandis que le nombre de mots de code parmi eux est seulement égal au nombre de mots d'information.
- Plus précisément, un mot d'information comporte m bits, et on lui ajoute r bits de contrôle pour obtenir le mot de code correspondant. Or les mots binaires de $m+1$ bits sont deux fois plus nombreux que ceux de m bits, ceux de $m+2$ bits sont encore deux fois plus nombreux que ceux de $m+1$ bits, etc.
- Par conséquent, les messages possibles ayant $n = m + r$ bits, sont $\underbrace{2 \times \ldots \times 2}_{r \text{ fois}}$ fois plus nombreux que les mots d'information, qui n'ont que m bits.
- C'est ce qui va rendre possible la détection et la correction d'erreurs.



Détection des erreurs Les mots de code, dispersés dans l'ensemble de tous les messages possibles, sont représentés par les points noirs. L'un d'eux est le message effectivement émis. Le message erroné reçu est représenté par le point blanc.



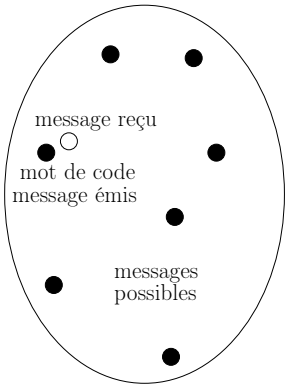
- S'il y a des erreurs alors le message reçu n'est *pas un mot de code*.
- Conséquence :

  Détection des erreurs
- Ceci n'est valable que *jusqu'à un certain point* car l'ensemble des erreurs pourrait avoir transformé le message émis en un autre mot de code.
- Conséquence :

  Détection... *imparfaite*

Remarque : On voit ici qu'un code qui détecte parfaitement toutes les erreurs ne peut pas exister ! Les bons codes sont faits de telle sorte que les imperfections n'apparaissent que très rarement (voir Section 4).

Corrections des erreurs Le message erroné reçu a été détecté, c'est-à-dire que ce n'est pas un mot de code.



- Le principe de la correction consiste à remplacer le message reçu erroné (et détecté comme tel) par le mot de code *le plus proche*, c'est-à-dire celui nécessitant de modifier le moins de bits possible.
- Conséquence :

  Correction des erreurs
- Ceci n'est valable que *jusqu'à un certain point*, car l'ensemble des erreurs pourrait avoir donné un message reçu plus proche d'un autre mot de code que du message émis.
- Conséquence :

  Correction... *imparfaite*

Remarque : On voit ici qu'un code qui corrige parfaitement toutes les erreurs détectées ne peut pas exister ! Les bons codes sont faits de telle sorte que les imperfections n'apparaissent que très rarement (voir Section 4).

4 Quantifier les imperfections

On donne dans cette section des valeurs numériques issues du calcul des probabilités. Le lecteur désirant en connaître le détail peut se reporter à l'Appendice.

4.1 Risque d'erreurs lors de la transmission

Exemple
Supposons qu'on utilise un code par répétition avec un canal de transmission dans lequel un bit sur cent est modifié (dans la réalité, ce serait un canal de très mauvaise qualité).

Le calcul des probabilités permet d'affirmer :
- Environ $97,030\%$ des messages reçus ne comportent pas d'erreur ;
- Environ $2,940\%$ des messages reçus comportent une erreur ;
- Environ $0,030\%$ des messages reçus comportent deux erreurs ;
- Environ $0,0001\%$ des messages reçus comportent trois erreurs.

Conclusion Dans la vie courante, il est beaucoup plus probable de recevoir un message avec pas ou peu d'erreurs qu'avec beaucoup d'erreurs.

4.2 Le pari de la détection des erreurs

Message reçu sans erreur détectée
- Il est possible qu'il soit quand même erroné : « sans erreur détectée » signifie simplement que le message reçu est bien un mot de code. Le message envoyé était aussi un mot de code, mais on ne peut pas savoir si c'est le même. Peut-être que les erreurs qui se sont produites ont transformé le message envoyé en un autre mot de code.
- Lors de la détection, on fait le pari qu'il n'y a effectivement pas eu d'erreurs, autrement dit que le message reçu est identique au message envoyé : Est-ce un pari risqué ? Pour répondre à cette question, on calcule le taux de messages erronés détectés

Exemple On reprend l'exemple du code par répétition avec un canal de transmission dans lequel un bit sur cent est modifié. Le calcul des probabilités permet d'affirmer :

- Taux des messages erronés détectés $\approx 99,997\%$

Conclusion Dans la vie courante, quand on ne détecte aucune erreur dans le message reçu, il est très probable qu'il ne contienne effectivement aucune erreur

4.3 Le pari de la correction des erreurs

Message reçu reconnu erroné puis corrigé

- Il est possible qu'il soit en fait mal corrigé : « corrigé » signifie que le message reçu erroné est remplacé par le mot de code le plus proche. Le message envoyé était aussi un mot de code, mais on ne peut pas savoir si c'est le même. Peut-être que les erreurs puis la correction ont transformé le message envoyé en un autre mot de code.
- On fait le pari que la correction a atteint son but, autrement dit que le message corrigé est identique au message envoyé. Est-ce un pari risqué ? Pour le savoir, on calcule le taux de messages reconnus erronés bien corrigés

Exemple On reprend l'exemple du code par répétition avec un canal de transmission dans lequel un bit sur cent est modifié. Le calcul des probabilités permet d'affirmer :

- Taux de messages reconnus erronés bien corrigés $\approx 99\%$

Conclusion Dans la vie courante, quand on corrige un message erroné par le mot de code le plus proche, il est très probable qu'il soit corrigé correctement

5 Le tour de magie

De quel code s'agit-il ?

Le code utilisé s'appelle le *code de double parité*. Les *mots d'information* sont constitués des 25 bits du carré initial. Le magicien ajoute 11 *bits de contrôle*, la *redondance* est donc égale à 11. Les *mots de code* sont constitués des 36 bits du carré obtenu. Le *rendement* est de $\frac{25}{36} \simeq 0,69$.

On constate avec le tour de magie que ce code permet de corriger correctement tous les messages reçus comportant une erreur. On peut montrer que tous les messages erronés comportant deux ou trois erreurs sont détectés, mais que certains sont mal corrigés ou impossibles à corriger (si on hésite entre deux corrections de même probabilité), tandis que certains des messages comportant 4 erreurs ou plus ne sont pas détectés.

Le code de double parité a un rendement plus faible que le code de parité simple, mais il est plus efficace, puisqu'il permet de corriger certains messages erronés. Le code de double parité a un meilleur rendement que le code par répétition, il est plus efficace car il permet de détecter plus de messages erronés, et de corriger au moins aussi bien, mais il est (relativement) plus compliqué sur le plan mathématique.

Comment ce code fonctionne-t-il ?

Bob veut transmettre un bloc de données à Alice

- Tout est écrit en binaire

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

Bob ajoute des données redondantes

- Il crée une ligne et une colonne supplémentaires
- Règle de calcul des valeurs supplémentaires :
  - ⋆ Le nombre de 1 sur chaque ligne doit être pair
  - ⋆ Le nombre de 1 sur chaque colonne doit être pair

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

Pendant la transmission

- Une perturbation se produit : une valeur est modifiée
- Une ligne et une colonne ne respectent plus la règle

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

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

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

À la réception

- Alice ne sait pas si les données ont été altérées
- Elle ne sait pas non plus ce que Bob lui a envoyé

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

Mais Alice constate

- Certaines lignes et colonnes ne respectent pas la règle
- C'est une preuve qu'une perturbation s'est produite
- Elle détecte l'altération dans les données transmises

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

Alice peut faire mieux

- Il lui suffit de changer la valeur à l'intersection pour que la règle soit de nouveau respectée
- Elle corrige l'altération dans les données transmises

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

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

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

Et maintenant

- Toutes les altérations peuvent-elles être décelées ?
- La correction effectuée est-elle toujours la bonne ?

Toutes les altérations peuvent-elles être décelées ?

- Supposons que Bob envoie ceci

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

- Et supposons qu'Alice reçoive ceci

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

- Toutes les lignes et les colonnes respectent la règle
- Alice pense que la transmission a été correcte
- Ce n'est pas le cas
- Mais elle n'a aucun moyen de s'en apercevoir

L'altération passe inaperçue

- On a vu que ce phénomène est statistiquement très rare
- Et on a vu aussi qu'il ne peut exister aucune méthode permettant de déceler absolument toutes les altérations

La correction proposée est-elle toujours la bonne ?

- Supposons que Bob envoie ceci

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

- Et supposons qu'Alice reçoive ceci

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

- Une ligne et une colonne ne respectent pas la règle
- Alice s'aperçoit que les données ont été altérées
- Alice corrige l'intersection de la ligne et de la colonne
- L'erreur n'est pas à cet endroit
- Mais elle n'a aucun moyen de s'en apercevoir

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

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

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

La correction n'est pas la bonne

- On a vu que ce phénomène est statistiquement très rare
- Et on a vu aussi qu'il ne peut exister aucune méthode permettant de toujours corriger correctement

6 Conclusion

Nous avons vu dans cette fiche, à travers quelques exemples simples, les principes généraux sur lesquels reposent les codes détecteurs et correcteurs d'erreurs utilisés en informatique : consolidation des mots d'information initiaux par ajout d'information redondante, détection des messages reçus qui ne peuvent pas être des messages émis, et correction d'un message erroné en le remplaçant par le message émis potentiel le plus proche.
Les informaticiens spécialistes de ce domaine utilisent et mettent au point des codes de plus en plus performants pour les échanges d'information et le stockage sur de nouveaux supports. Ils utilisent pour cela des méthodes sophistiqués issus de différents domaines des mathématiques.
Pour les lecteurs souhaitant aller plus loin, il existe de nombreux ouvrages destinés aux étudiants des premières années d'université traitant de ce sujet, et on trouve aussi facilement des cours en accès libre sur internet. Comme ouvrages en français sur ce thème, on peut par exemple citer [1] (niveau licence) et [2] (niveau master).

Références

[1] J. Badrikian. *Codes correcteurs.* Ellipses, 2002.

[2] É. Tannier et S. Varrette J.-G. Dumas, J.-L. Roch. *Théorie des codes.* Dunod, 2013.

Appendice : Calculs de probabilités

Modéle mathématique

La situation à modéliser est la suivante : on transmet un message composé de n bits. Pendant la transmission, chaque bit est modifié avec la probabilité $p < 0,5$ (c'est le taux d'erreur du canal de transmission : on a $p \approx$ nombre de bits erronés/nombre de bits transmis). On fait les hypothèses simplificatrices que les erreurs ont autant de chance de se produire sur les 1 que sur les 0, d'une part, et que les erreurs sont indépendantes pour chaque bit, d'autre part.
Il s'agit d'un schéma de Bernouilli, et la loi de probabilité correspondante est une loi binomiale de paramètres n et p :

- Variable aléatoire $X =$ nombre de bits erronés
  - ⋆ Les valeurs possibles pour X sont $0 \dots n$
- Probabilité de recevoir un message avec k bits erronés (pour k compris entre 0 et n) :
  - ⋆ $\mathcal{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$
  - ⋆ où $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ sont les coefficients binomiaux

Calcul des coefficients binomiaux

- la factorielle : $n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$ et $0! = 1$ par convention
- $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
- $\binom{n}{0} = \frac{n!}{0!(n-0)!} = 1$
- $\binom{n}{1} = \frac{n!}{1!(n-1)!} = n$
- $\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$
- etc.

Exemple

Supposons qu'on utilise un code par répétition avec un canal de transmission dans lequel un bit sur cent est modifié (dans la réalité, ce serait un canal de très mauvaise qualité).

Probabilité de chaque situation

- Probabilité de recevoir un message avec k bits erronés
  - ⋆ $\mathcal{P}(X = k) = \binom{3}{k} 0,01^k (0,99)^{3-k}$
- Probabilité que le message reçu ne comporte pas d'erreur
  - ⋆ $\mathcal{P}(X = 0) = 0,99^3 \approx 97,030\%$
- Probabilité que le message reçu comporte une erreur
  - ⋆ $\mathcal{P}(X = 1) = 3 \times 0,01 \times 0,99^2 \approx 2,940\%$
- Probabilité que le message reçu comporte deux erreurs
  - ⋆ $\mathcal{P}(X = 2) = \frac{3 \times 2}{2} \times 0,01^2 \times 0,99 \approx 0,030\%$
- Probabilité que le message reçu comporte trois erreurs
  - ⋆ $\mathcal{P}(X = 3) \approx 0,0001\%$

Message sans erreur détectée

- On calcule le taux de messages erronés détectés
- Probabilité que le message reçu comporte une ou des erreurs
  - ⋆ $1 - \mathcal{P}(X = 0) \approx 2,97010\%$
- Probabilité qu'un message erroné soit détecté
  - ⋆ $\mathcal{P}(X = 1) + \mathcal{P}(X = 2) \approx 2,97000\%$
- Taux des messages erronés détectés
  - ⋆ $\frac{\text{Proba erreurs détectées}}{\text{Proba erreurs}} \approx 99,997\%$

Message reçu reconnu erroné puis corrigé

- On calcule le taux de messages reconnus erronés bien corrigés
- Probabilité qu'un message soit erroné et détecté
  - ⋆ $\mathcal{P}(X = 1) + \mathcal{P}(X = 2) \approx 2,97\%$
- Probabilité qu'un message soit erroné et bien corrigé
  - ⋆ $\mathcal{P}(X = 1) \approx 2,94\%$
- Taux de messages reconnus erronés bien corrigés
  - ⋆ $\frac{\text{Proba erreurs bien corrigées}}{\text{Proba erreurs détectées}} \approx 99\%$