La théorie classique de l'information.

2ème partie : le point de vue de Shannon.

Rappel concernant le point de vue de Kolmogorov.

Rappelons le point de vue de Kolmogorov sur la compression des suites. Compresser une suite (décodable), s, c'est la remplacer par un programme (décodable), p, plus court qu'elle, qui, implémenté sur une machine de Turing universelle, est capable de la reconstruire sans perte :

$$MTU(p)=s \qquad avec \qquad \ell(p)<\ell(s)$$

Parmi tous les programmes compresseurs, il y en a forcément un qui est plus court que les autres. La longueur de ce programme minimum définit la complexité (ou entropie) algorithmique de la suite :

$$S_{algo}=K(s)=\min_{p:MTU(p)=s}\ell(p).$$

Nous avons également vu qu'à une constante inessentielle près, la complexité d'une suite de longueur, N, navigue entre les valeurs extrêmes :

$$\ell g(N)\le K(s^N)+c\le N$$

Compression maximum, cas des suites structurées par un programme court (Exemples : la suite des chiffres de π, la suite des positions des atomes d'un cristal NaCl à 0K, …)

Incompression, synonyme d'aléatoire (Exemples : une suite de tirages pile ou face, la suite des positions des molécules dans un gaz, …)

Un critère suffisant de compressibilité : l'anormalité.

Le point de vue développé par Kolmogorov sur la compression des suites souffre d'un handicap : il définit une grandeur, la complexité algorithmique, K(s), qui n'est pas calculable au sens de Turing. Rappelons encore que cela ne signifie nullement que K(s) est hors d'atteinte dans tous les cas mais seulement qu'il n'existe pas de procédure algorithmique générale qui soit capable de calculer K(s), en sortie, lorsqu'on lui fournit s, quelconque, en entrée. En fait, il n'existe même pas de critère opérationnel permettant de déclarer à tous les coups si une suite quelconque donnée est compressible.

Il revient à Shannon d'avoir compris qu'il existait une condition suffisante mais non nécessaire de compressibilité : il suffit que la suite ne soit pas normale au sens de Borel. Qu'est-ce que cela signifie ?

Une suite est dite normale, au sens de Borel, si non seulement les C caractères de l'alphabet utilisé mais encore tous les C^k blocs de k caractères consécutifs, apparaissent dans la suite avec des fréquences égales, quelle que soit la valeur de k. Il est plus facile de suspecter, d'un simple coup d'œil, l'anormalité que la normalité.

Les deux suites que voici ne sont probablement pas normales :

000010001100000000000110000000010000… (provisoirement trop de ‘0’)

010101010101010101010101010101010101… (’00’ et ’11’ provisoirement absents)

Par contre, toute suite aléatoire,

100010111001010000011010101101100111110 0…

est certainement normale car il est inconcevable qu’un bloc de chiffres soit privilégié.

Contrairement à ce qu'on a longtemps cru, l’inverse n’est pas vrai : toute suite normale n’est pas forcément aléatoire ! Par exemple la suite des chiffres binaires du nombre π,

$$11001001000011111101101010100010001000010 11\cdots^{N}$$

est très probablement normale et cependant π n’est pas aléatoire car il existe un programme court qui structure la suite de ses chiffres, $K(\pi) << N$.

Remarque : il peut paraître étrange que l’on ne soit pas certain de la normalité de π. Le fait est que les mathématiciens n’ont, à ce jour, rien pu établir quant à la normalité de la plupart des nombres algébriques ou transcendants. Seuls quelques nombres construits expressément pour cela ont été démontrés normaux dans certaines bases sans que l’on soit sûr qu’ils le sont en toute base. Il est d’autant plus paradoxal que l’on soit incapable de proposer un nombre normal en toutes bases que Borel a démontré dès 1909 que presque tout nombre est normal en toute base ! Le même genre de paradoxe se retrouve avec les nombres aléatoires qui sont infiniment plus nombreux que les autres et cependant on est également incapable d’en construire un.

Normal ≠ aléatoire.

Il résulte de la discussion précédente que la définition algorithmique du hasard est beaucoup plus forte que celle qui est simplement basée sur la seule normalité au sens de Borel. Les suites normales franchissent pourtant les tests statistiques les plus exigeants !

On attend intuitivement d’une suite binaire aléatoire qu’elle puisse servir de base à un pari équitable entre deux adversaires qui misent des sommes égales. L’histoire suivante montre la pertinence du point de vue algorithmique à cet égard.

Imaginez que vous croisiez un saltimbanque au coin d’une rue qui propose aux passants de jouer 1000 parts de pile ou face, chaque joueur choisissant en premier à tour de rôle. Il s’est installé une petite table couverte d’un drap et un mécanisme invisible lance la pièce sans intervention humaine. Vous êtes immédiatement intrigué par son manège car il vous propose de ne miser que deux euros à chaque coup alors que lui en mise trois !

Comme vous trouvez l’offre étonnante et que vous restez prudent vous vous contentez d’observer les tirages successifs de la machine que l’homme fait tourner à vide dans l’espoir d’attirer un client. Supposons que le tirage fasse apparaître la séquence suivante :

‘0010010000111111011010101000100010000101101000110000100011010011000 10…’

Cette suite semble normale et si vous n'êtes pas informé qu'il s'agit de la suite des chiffres de π-3, vous allez peut-être la trouver honnête et tenter votre chance. Quelle ne sera pas votre déconvenue de voir que vous vous trompez, comme il se doit, une fois sur deux mais que le saltimbanque, miraculeusement, ne se trompe jamais. Au bilan vous aurez perdu 750 euros ! Il peut même raffiner sa stratégie en faisant exprès de se tromper de temps à autre, cela diminue ses gains mais aussi les soupçons ! Bien entendu il n'y a rien de mystérieux là dessous : le saltimbanque peut être un calculateur prodige qui connaît un algorithme court lui permettant de calculer mentalement le développement binaire de π.

La morale de cette histoire est qu'il ne suffit pas qu'une suite soit normale au sens de Borel pour être aléatoire. Il faut encore qu'elle soit incompressible, c'est-à-dire qu'elle ne résulte pas d'un algorithme plus court qu'elle.

A cet égard, les fonctions Random proposées par les logiciels, aussi sophistiqués soient-ils, ne peuvent prétendre engendrer de suites réellement aléatoires puisqu'elles se basent sur des algorithmes numériques de longueur indépendante de N. Elles font illusion parce qu'elles franchissent les tests statistiques de normalité ce qui est insuffisant. Ces fonctions, telles la règle 30 de Wolfram, utilisée par Mathematica, ne créent donc que des suites pseudo aléatoires.

Suites normales, simplement biaisées et corrélées.

Lorsqu'on lance un dé à C faces, de façon répétée, on génère une suite écrite dans l'alphabet $\{a_1, \ldots, a_C\}$. Plusieurs cas sont possibles :

- Si le dé et le lancer sont honnêtes, la suite engendrée est normale. Les probabilités d'occurrence de chaque symbole sont toutes égales à 1/C et les probabilités d'occurrence des groupes de symboles sont directement héritées des précédentes via la loi de multiplication des probabilités de tirages indépendants, en bref :

$$p('a_i') = \frac{1}{C}, \quad p('a_i a_j') = \frac{1}{C^2}, \quad \ldots$$

- Si le dé est pipé mais que le lancer reste honnête, la suite engendrée est dite simplement biaisée. On a cette fois que les probabilités d'occurrence des différents symboles varient et que les probabilités d'occurrence des groupes de symboles leur restent liées par la loi de multiplication des tirages indépendants, en bref :

$$p('a_i') = p_i, \quad p('a_i a_j') = p_i p_j, \quad \ldots$$

- Enfin, quel que soit l'état du dé, si le lancer est truqué, la suite engendrée est dite corrélée et les règles précédentes ne s'appliquent plus :

$$p('a_i') = p_i, \quad p('a_i a_j') \neq p_i p_j, \quad \ldots$$

Compressibilité des suites anormales.

Il est aisé de comprendre intuitivement qu'une suite anormale est compressible. En effet, les symboles ou les groupes de symboles n'y étant pas équiprobables, il suffit de procéder à un réencodage de la suite à l'aide d'un code préfixe variable et de réserver les codes courts aux blocs fréquents. Voyons cela sur l'exemple d'une suite binaire simplement biaisée par suite d'un excès de '0' :

000010001100000000000110000000010000…

Si on lit la suite par blocs de deux caractères, le code suivant,

$00 \Rightarrow 0$, $01 \Rightarrow 10$, $10 \Rightarrow 110$, $11 \Rightarrow 111$,

devrait convenir : il est préfixe et il réencode le doublet le plus fréquent à l'aide d'un symbole unique. On trouve que la suite s'en trouve effectivement raccourcie :

000010001100000000000110000000010000… $\Rightarrow$ **0**01100111000001011000010 00 …

Détermination du code optimal.

On voit, sur l'exemple précédent, que l'utilisation d'un code court pour le bloc le plus fréquent compense largement l'allongement inévitable des codes des blocs moins fréquents. Si l'on note, ℓ_i, la longueur du code du i^{ème} bloc apparaissant avec la probabilité, p_i, on trouve le code préfixe optimal en minimisant la longueur moyenne du code,

$$\langle \ell \rangle = \sum_i p_i \ell_i$$

sous la contrainte de Kraft,

$$\sum_i 2^{-\ell_i} = z \le 1.$$

Ce genre de problème de minimisation sous contraintes se résout classiquement par la méthode des multiplicateurs de Lagrange. On a successivement :

$$L = \sum_i p_i \ell_i + \lambda\left(\sum_i 2^{-\ell_i} - z \right)$$

$$\partial_{\ell_i} L = \partial_\lambda L = 0 \;\Rightarrow\; p_i = \frac{2^{-\ell_i}}{z} \;\Rightarrow\; \ell_i = -(\ell g(z) + \ell g(p_i))$$

La longueur du code optimal se compose de deux termes positifs (z et les p_i sont inférieurs à 1) et on voit qu'on a intérêt à opter, autant que possible, pour un code complet, caractérisé par z=1. Si on pose, z=1, dans les formules précédentes, on trouve :

$$p_i = 2^{-\ell_i} \qquad \Rightarrow \qquad \ell_{i,min} = -\ell g p_i$$

Ce code n'est rigoureusement optimal que si les probabilités d'occurrence des blocs de symboles sont toutes des puissances négatives de 2. Dans ce cas seulement, les longueurs de code sont les entiers, $\ell_i = -\ell g p_i$, et le code optimal moyen, exprimé en bits/symbole, vaut exactement :

$$\langle \ell \rangle_{min} = -\sum_i p_i \ell g p_i$$

Dans tous les cas où les probabilités ne sont pas des puissances négatives de 2, c'est-à-dire la plupart du temps, on se contente de l'approximation qui consiste à arrondir la longueur des codes aux entiers les plus proches quitte à lire la suite par blocs de caractères de plus en plus longs afin de minimiser l'erreur commise. On verra un exemple numérique sous peu.

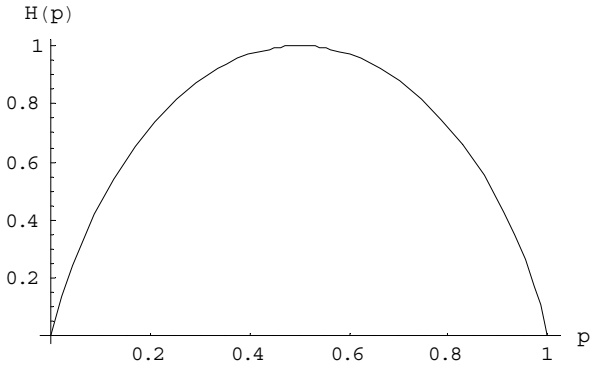
La fonction simple de Shannon.

La fonction définissant la longueur du code optimal dans le cas idéal porte le nom de fonction de Shannon :

$$H(p_1, \cdots, p_C) = -\sum_{i=1}^{C} p_i \ell g p_i \qquad (\sum_{i=1}^{C} p_i = 1)$$

Le cas de l'alphabet binaire, C=2, est particulièrement important. La fonction de Shannon s'écrit simplement dans ce cas :

$$H(p, 1-p) = -p\ell g(p) - (1-p)\ell g(1-p)$$



On constate que si la suite est binaire et normale, p vaut 0.5 et son encodage consomme automatiquement 1 b/s. Même si on la lit par blocs de k caractères, $C=2^k$, la probabilité d'occurrence de tout k-bloc vaut $p_i = 2^{-k}$ et l'encodage de la suite consomme encore 1 b/s.

Compression d'une suite simplement biaisée.

Etudions la compression de la suite suivante, de longueur N=256, simplement biaisée selon le schéma de probabilités, $p_0=3/4$ et $p_1=1/4$:

1001100100000000100001000000000000000000000001000000100100011001001000001010
1000010000010001001011011010000100010000100000100101000001001001100100000010
1100000100100100001001000100001010010001000000010000100010000110100001001010
1111000000000011000001100010101 0

Ces probabilités n'étant pas des puissances négatives de 2, il est inutile d'espérer pouvoir compresser cette suite jusqu'à la limite théorique de Shannon,

$$\langle \ell \rangle_{min} = H(3/4,1/4) = -1/4 \ell g(1/4) - (3/4) \ell g(3/4) = 0.811 \quad bits/symb$$

Rappelons que dans le cas des suites simplement biaisées, cette limite théorique ne dépend pas de la longueur des blocs utilisés pour la lecture de la suite. En effet, les probabilités des doublets sont, dans ce cas, directement héritées des probabilités simples par la loi de multiplication des probabilités indépendantes. On a donc, $p_{00}=9/16$, $p_{01}=p_{10}=3/16$ et $p_{11}=1/16$, d'où on calcule :

$$\langle \ell \rangle_{min} = H(9/16,3/16,3/16,1/16) = 0.811 \quad bits/symb$$

soit la même valeur que précédemment. Par contre, l'approche numérique de cette valeur théorique tire généralement avantage de la prise en considération de blocs de longueurs croissantes. C'est ce qu'on observe lorsqu'on recourt à l'algorithme d'Huffman.

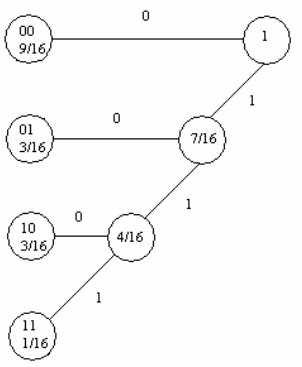
1ère méthode : codage d'Huffman.

Lisons la suite à compresser par blocs de deux symboles

1001**10**01**00**0000001000010000000000000000000000010000001001000110010010000010 1
0100001000001000100101101101000010001000010000010010100000100100110010000010
1100000100100100001001000100001010010001000000010000100010000110100001001010
1111000000000011000001100010101 0

$$\begin{cases} p('0') = 3/4 \\ p('1') = 1/4 \end{cases} \Rightarrow \begin{cases} p('00') = 9/16 \\ p('11') = 1/16 \end{cases} \quad \begin{cases} p('01') = 3/16 \\ p('10') = 3/16 \end{cases}$$

L'algorithme d'Huffman commence par associer à chaque doublet un nœud auquel il confère un poids égal à sa probabilité d'occurrence. Ensuite, il aligne les noeuds sur une colonne dans l'ordre des poids décroissants. Enfin, il apparie itérativement les deux nœuds de plus bas poids en formant à chaque fois un nouveau nœud auquel il affecte un poids égal à la somme des poids appariés. Chaque bras d'une paire naissante reçoit indifféremment l'étiquette '0' ou '1'. On poursuit la manoeuvre jusqu'à construire le dernier nœud, de poids unité.



On obtient un code automatiquement préfixe et optimal en parcourant l'arbre de manière rétrograde à partir du dernier nœud formé jusqu'au symbole qu'on chercher à coder : il suffit de lire dans l'ordre les étiquettes rencontrées. Dans l'exemple, on trouve le code suivant :

'00' ⇒ '0' '01' ⇒ '10' '10' ⇒ '110' '11' ⇒ '111'

La longueur moyenne du code ainsi obtenu vaut : (1x9/16+2x3/16+3x3/16+3x1/16)/2=0.844 b/s valeur proche de la limite théorique de Shannon, 0.811 b/s. La suite réencodée de cette manière ne fait plus que 226 bits :

11010**110**10**0**0001100100000000000011000100110011101101000101010001100011001101010111011100011001100100010011011000110100111011000110111001001101000110100011011010010000100011001100101101100100110111111000001110010110011011010

Si on avait lu la suite originale par blocs de trois caractères,

100110**010**000**000**0100001000000000000000000000010000001001000110010010000010100001000001000100101101101000010001000010000010010100000100100110010000010110000010010010000100100010000101001000100000001000010001000011010000100101111000000000011000001100010101010

on aurait trouvé le code d'Huffman suivant :

'000' ⇒ '0' '001' ⇒ '110' '010' ⇒ '100' '011' ⇒ '11100'
'100' ⇒'101' '101' ⇒ '11101' '110' ⇒ '11110' '111' ⇒ '11111'

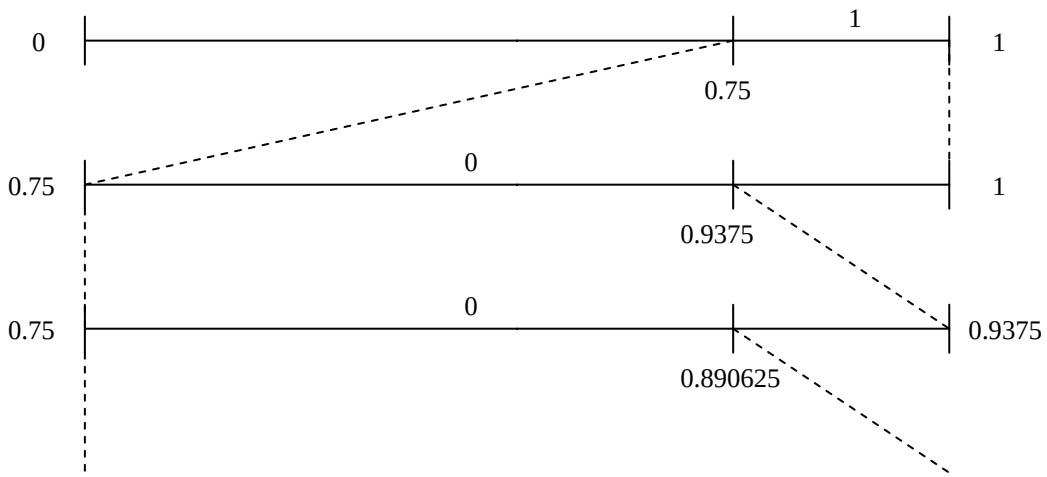
La longueur moyenne du code optimal aurait été de 0.823 b/s et la suite réencodée ne comporterait plus que 219 bits :

10111110**100**0**0**1000101000000110001011011110011011001101001011100110010111101111011110101001100100010010010101011011111010001001111001001001000101101100011101110010101100100110011100100010111101111110001110001101011001110

2ème méthode : codage arithmétique d'Elias

Il existe une autre méthode de compression particulièrement astucieuse qui évite d'avoir à lire la suite par blocs de tailles croissantes. Elle consiste à partitionner l'intervalle {0,1} en C segments jointifs dont les longueurs sont calculées proportionnellement aux probabilités

respectives des divers caractères. Dans l'exemple, la suite est binaire, C=2, et la partition ne comporte que deux segments de longueurs proportionnelles à 0.75 et 0.25. Chaque segment reçoit l'étiquette, '0' ou '1', qui correspond à sa probabilité, 0.75 pour '0' et 0.25 pour '1'. On sélectionne le segment qui correspond au premier symbole de la suite à compresser et on lui fait subir le même traitement de partition. La figure qui suit détaille les trois premières étapes du processus itératif en tenant compte du fait que les premiers caractères lus dans la suite à compresser sont, dans l'ordre, 100 :



On voit que l'intervalle de départ, {0,1}, rétrécit à chaque étape. Quand les 256 caractères de la suite sont lus, il se réduit à :

{0.8856439692656931649305299257192369413372596399364930381135278173509618,
0.8856439692656931649305299257192369413372596399364930381135278173882385}

N'importe quel réel faisant partie de cet intervalle convient pour l'encodage décimal de la suite, par exemple,

0.8856439692656931649305299257192369413372596399364930381135278173 6.

Il suffit de traduire ce nombre en binaire pour obtenir un réencodage qui compresse de façon non ambiguë la suite de départ :

1101011101001001110000100110111110010000110111000010111010101010101000011110101000011110110110011110011110011000110110111100101100110001110110110000100101101010000111001100001111110110000011101100000101100011101010000

Ses 216 bits nous rapprochent de la limite de Shannon située, rappelons-le, à 207 bits.

Le décodage s'effectue en procédant en sens exactement inverse. Le décodeur doit évidemment connaître la tables des probabilités utilisées par l'encodeur.

Compressibilité des suites corrélées.

Une suite corrélée doit idéalement être lue par blocs de tailles, k, croissantes. En effet, la fonction de Shannon change de valeur selon la taille des blocs qui découpent la suite. Il en résulte qu'il y a lieu de remplacer la limite simple, H, par la limite étendue de Shannon,

$$H_\infty = -\lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{2^k} p_{i,k} \ell g(p_{i,k})$$

où cette fois les $p_{i,k}$ définissent les probabilités d'occurrence des k-blocs.

Autres méthodes de compression sans pertes.

Les méthodes d'Huffman ou d'Elias reposent sur la connaissance des probabilités, $p_{k,i}$. Une étude statistique préalable effectuée sur une large portion de la suite à compresser est donc nécessaire. Peut-on se passer de cette connaissance préalable ? Il existe effectivement des méthodes de type « dictionnaire » qui se font une idée progressive des motifs caractéristiques de la suite à mesure qu'elles la lisent. Ce faisant, elles construisent un dictionnaire des motifs fréquents et utilisent un code court pour les représenter. Il existe un grand nombre de variantes toutes basées sur le principe de la méthode de Liv-Zempel qui sont à la base des utilitaires zip et Gzip. Il importe de ne pas confondre ces méthodes avec les méthodes à base de transformées de Fourier discrètes ou d'ondelettes qui compriment davantage mais avec pertes. Tels sont les utilitaires de type jpeg. Si intéressantes que puissent être ces variantes en pratique, elles ne concernent pas cet exposé.

Peut-on dépasser la limite de Shannon ?

La compression à la Shannon ne concerne que les suites anormales et elle a pour effet de les faire tendre vers la normalité. Peut-on faire mieux, en d'autres termes peut-on compresser une suite normale ?

La réponse n'est affirmative que dans les seuls cas où la suite est structurée par un programme court. Pour dépasser la limite de Shannon et approcher la limite infranchissable de Kolmogorov, il est toutefois nécessaire de faire preuve d'astuce ou d'intelligence afin de comprendre la logique souterraine qui gouverne la suite. Comme exemple, nous avons déjà évoqué la suite des chiffres du nombre π. En voici une autre qui à première vue n'évoque rien de particulier :

101101010000010011110011001100111111100111011110011001001000010001 01…

Mais si on regarde cette suite comme le développement fractionnaire d'un nombre, s, on calcule que son carré vaut :

$s*s= 0.01111111111111111111\ldots \approx 0.1_2$

La conclusion qui s'impose est que la suite, s, aligne les chiffres binaires de $\sqrt{2}$. Evidemment cet exemple a été choisi sur mesure et, dans le cas général, la compression algorithmique s'apparente plutôt à un casse-tête.

Résumé sur les limites théoriques à la compression des suites.

Rappelons que l'ensemble des suites peut être partitionné en deux sous-ensembles : d'une part, l'ensemble, non dénombrable à la limite N→∞, des suites incompressibles et, d'autre part, l'ensemble, dénombrable à la même limite, des suites compressibles. La théorie de l'information impose les limites fondamentales suivantes à la compression des suites.

- Les suites simplement biaisées sont compressibles, en forme normale, par une procédure statistique effective (Huffman ou Elias) jusqu'à la limite simple de Shannon : $H = -\sum_{i=1}^{C} p_i \ell g(p_i)$ bits/symb, où les p_i sont les probabilités d'occurrence des C caractères alphabétiques.

- Plus généralement, les suites corrélées sont compressibles par une procédure statistique effective (Huffman ou Elias) jusqu'à la limite étendue de Shannon : $H_\infty = -\lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{C^k} p_{k,i} \ell g(p_{k,i})$ bits/symb, où les $p_{k,i}$ sont les probabilités d'occurrence des C^k blocs de k caractères consécutifs. La compression transforme à nouveau la suite corrélée en une suite normale.

- Les suites normales ne sont compressibles au-delà de la limite étendue de Shannon que si elles ne sont pas aléatoires ou, ce qui revient au même, que s'il existe un programme court qui les structure. Toutefois cette compression supplémentaire, de nature algorithmique, ne peut être réalisée par une procédure effective. L'idée consistant à passer en revue tous les programmes par ordre de taille croissante dans le but de retenir le premier qui afficherait la suite cherchée est impraticable à cause de l'indécidabilité du problème de l'arrêt des machines de Turing.

Contenu informationnel d'une suite : le point de vue statistique.

Reprenons le lancer répété d'un dé quelconque à C faces. Un observateur, que nous baptisons Informé, archive les résultats dans l'ordre où ils se présentent sur le disque dur de son ordinateur (ou les mémorise dans son cerveau). Si le dé est pipé, la suite des tirages ne sera pas normale et il sera possible de la rendre normale en la compressant par une procédure effective. Il arrivera que la suite compressée en forme normale soit structurée par un programme court auquel cas elle sera compressible au-delà de la limite de Shannon mais nous savons que cet événement sera extrêmement rare car il y a beaucoup plus de suites que de programmes courts structurants.

Plaçons-nous à présent à la place d'un observateur, baptisé Ignorant, qui ne sait rien du détail des tirages mais qui connaît quand même les probabilités d'occurrence des faces du dé. Par définition, Ignorant appelle contenu informationnel, I(s), d'une suite, s, (ou information manquante ou encore entropie statistique) le nombre minimum de questions binaires qu'il doit poser à Informé pour reconstituer la suite dans une stratégie statistiquement optimale.

L'intérêt de cette définition est que dans un assez grand nombre de cas dignes d'intérêt, l'entropie statistique coïncide avec l'entropie algorithmique, K(s) = I(s). La condition pour qu'il en soit ainsi est que la suite ne soit structurée par aucun programme court

sans quoi Ignorant est incapable de poser les questions binaires pertinentes et il surestime inévitablement l'entropie algorithmique, K(s)<I(s). Concentrons-nous, à présent, sur les cas où le point de vue statistique s'avère pertinent, c'est-à-dire lorsque la suite n'est structurée par aucun programme court.

Entropie statistique des suites non corrélées.

Commençons par le cas le plus simple d'une suite simplement biaisée donc non corrélée. Si aucune corrélation à distance n'existe dans l'apparition des symboles, c'est que l'occurrence d'un symbole en position, i, n'a aucune influence sur l'occurrence du symbole en position, j, avec j>i. Autrement dit, les symboles successifs correspondent aux lancers non manipulés d'une pièce éventuellement truquée ou, plus généralement au lancer honnête d'un dé à C faces, éventuellement pipé. Cela équivaut à une suite d'expériences d'Young à un seul photon avec K détecteurs en positions éventuellement dissymétriques.

Un résultat fondamental dû à Shannon affirme que, dans une stratégie optimale qui lit la suite par blocs de k caractères, il n'y a précisément pas moyen de dépasser la limite de compression statistique donnée par la fonction de Shannon d'ordre, k :

$$I_k(s) = -N(s)\frac{1}{k}\sum_{i=1}^{C^k} p_{i,k}\,\ell g(p_{i,k}) \quad \text{bits,}$$

soit :

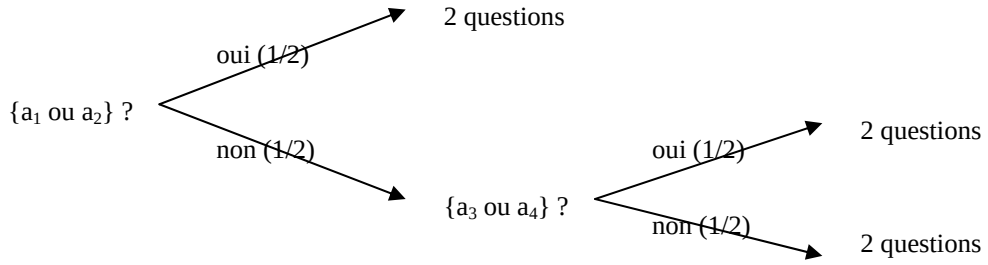
$$i_k = -\frac{1}{k}\sum_{i=1}^{C^k} p_{i,k}\,\ell g(p_{i,k}) \quad \text{bits/symb.}$$

où l'indice, k, se réfère au fait que la suite est lue par blocs de k caractères.

Afin d'illustrer ce résultat, voyons comment Ignorant peut s'y prendre pour reconstituer l'information manquante, $I_k(s)$, par une succession de questions binaires.

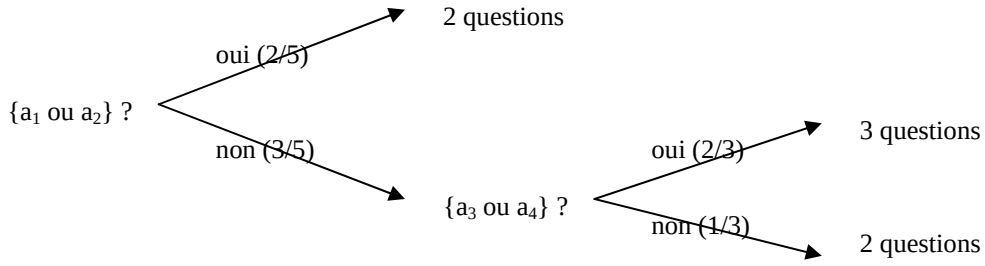
1er cas : le dé n'est même pas pipé d'où les C symboles, $\{a_1, \ldots, a_C\}$, sont équiprobables avec la probabilité, 1/C. La suite est aléatoire puisque normale et non structurée par un programme court. Montrons que l'on a l'égalité, $I_1(s) = N(s)\ell gC$, d'où K(s) = I_1(s).

Si C est une puissance de deux, $C=2^n$, tout est simple : $n = \ell g(C)$ questions binaires suffisent pour identifier le premier caractère de la suite par simple dichotomie équitable. Voici l'exemple, C=4, qui utilise l'alphabet équiprobable, $\{a_1, \ldots, a_4\}$. Ignorant le partitionne en deux parties égales et pose la première question binaire : « Le premier caractère de la suite appartient-il au sous-ensemble $\{a_1, a_2\}$» ? Poursuivant sur cette lancée, on a le diagramme suivant qui consomme, $n = \ell g(C) = 2$, questions binaires dans tous les cas de figure :



Si la suite comprend N caractères, nN questions suffisent puisqu'ils sont indépendants et l'entropie statistique de la suite vaut bien, $I_1(s) = N(s)\ell g(C)$ bits, soit, ramenée à un symbole, $i_1(s) = \ell g(C)$bits/symb. En particulier si l'alphabet est binaire (C=2), on trouve une entropie statistique de 1 bit/symb.

Les choses se compliquent un peu si C n'est pas une puissance de deux mais la dichotomie la plus équitable reste la meilleure stratégie. Voyons l'exemple, C=5, qui utilise l'alphabet équiprobable, {a_1, …, a_5}. Ignorant le partitionne en deux parties aussi égales que possible et pose la première question binaire : « Le premier caractère appartient-il au sous-ensemble {a_1, a_2} ? ». Poursuivant sur cette lancée, on a le diagramme suivant :



Le nombre moyen de questions vaut cette fois :

$i_1(5) = 2\frac{2}{5} + 3\frac{3}{5}\frac{2}{3} + 2\frac{3}{5}\frac{1}{3} = \frac{12}{5} = 2.4$ bits/symb.

La fonction Mathematica suivante calcule $i_1(C)$ récursivement :

```
i1[C_] := i1[C] = 1 + Floor[C/2]/K i1[Floor[C/2]] + Ceiling[C/2]/C i1[Ceiling[C/2]]; i1[1] = 0; i1[2] = 1;
```

La valeur trouvée, 2.4, est légèrement supérieure à, $\ell g5$=2.32193 : on paie le fait que C n'est pas une puissance de deux d'où il résulte que la dichotomie ne s'effectue pas exactement à toutes les étapes de la recherche du symbole. Cependant, il est possible d'améliorer le rendement de l'algorithme si la suite est longue. Il suffit de procéder à la recherche simultanée des blocs de k symboles. Voyons d'abord le cas simple, k=2.

S'intéresser aux symboles de la suite par blocs de deux signifie que l'on cherche à les identifier parmi les C^2=25 couples possibles, {a_1, a_1}, {a_1, a_2}, …, {a_5, a_5}. Tout se passe comme si on travaillait sur base d'un alphabet étendu comportant 25 symboles. Dans ce cas, on trouve qu'il faut $i_2(5) = i_1(5^2)/2$= 2.36 questions par symbole.

En poursuivant sur des triplets, k=3, on trouve, $i_3(5)=i_1(5^3)/3=2.325$, qui se rapproche sérieusement de l'optimum, $\ell g5=2.32193$, que l'on atteindrait à la limite des grandes valeurs de k :

$$i_\infty(C)=\lim_{k\to\infty}\frac{i_1(C^k)}{k}=\ell g(C)\cdot$$

En résumé, l'entropie statistique d'une suite normale, de longueur, N, vaut : $I_1(C)=N\ell g(C)$ bits, soit : $i_1(C)=\ell g(C)$ bits/symb. Si la suite est binaire, ces expressions se réduisent à : I_1=N bits et i_1=1 bit/symb. Au fond, dans ce cas simple, on peut considérer que toutes les suites ont la même probabilité d'apparaître, $p = C^{-N}$, et on peut assimiler chaque suite à un message porteur d'une quantité d'information valant :

$$I_1=-\ell g(p)=N\ell g(C)\ \ bits\cdot$$

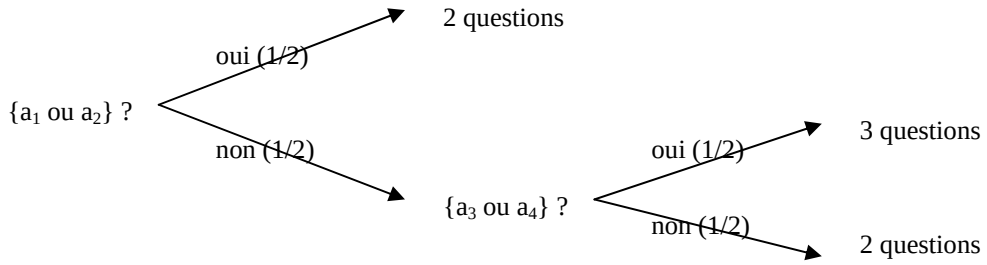
Plus la probabilité d'un événement est faible, plus son occurrence est porteuse d'information. A l'opposé, l'occurrence d'un événement prévisible avec la probabilité un n'apporterait aucun renseignement nouveau, tout cela est conforme à l'intuition.

<u>2ème cas</u> : le dé est pipé d'où les C symboles, $\{a_1, \ldots, a_C\}$, ne sont pas équiprobables. Il faut comprendre cette fois que les symboles ont été tirés au sort de façon parfaitement honnête hors d'un alphabet biaisé, $\{a_1, \ldots, a_C\}$, où chaque symbole, a_i, possède sa probabilité d'occurrence, p_i.

Il existe à nouveau un cas simple, c'est celui où toutes les probabilités, p_i, sont des puissances négatives de 2. Dans ce cas, en effet, la stratégie dichotomique peut être rendue exacte à tous les niveaux et l'entropie statistique vaut exactement :

$$i_1(C)=-\sum_{i=1}^{C}p_i\ell g(p_i)\ \text{bits/symb.}$$

Considérons l'exemple de l'alphabet biaisé, $\{a_1, \ldots, a_5\}$, où la probabilité d'apparition de chaque symbole vaut respectivement : {1/4, 1/4,1/8, 1/8, 1/4}.



Le nombre moyen de questions vaut : $2\frac{1}{2}+3\frac{1}{4}+2\frac{1}{4}=\frac{9}{4}=1.25$ bits/symb, ce qui correspond exactement à l'optimum annoncé :

$$i_1(5) = -\sum_{i=1}^{5} p_i \ell g(p_i) = -3\frac{1}{4}\ell g\frac{1}{4} - 2\frac{1}{8}\ell g\frac{1}{8} = \frac{9}{4} \text{ bits/symb.}$$

Lorsqu'une des probabilités, p_i, n'est pas une puissance négative de 2, la dichotomie cesse d'être exacte et l'entropie statistique estimée de cette manière est surévaluée. On peut cependant à nouveau se rapprocher de l'optimum, i_1, en cherchant à reconstituer la suite par blocs de longueur, k, croissante.

Entropie statistique des suites corrélées.

La détermination de l'entropie statistique des suites corrélées s'effectue dans le même esprit en considérant les symboles par blocs de k, de plus en plus longs et en traitant ces blocs comme des symboles indépendants d'un alphabet plus vaste qui en comporterait C^k. La théorie précédente continue de s'appliquer mais l'entropie statistique dépend de k :

$$i_k = -\frac{1}{k}\sum_{i=1}^{C^k} p_{i,k}\,\ell g(\,p_{i,k}\,) \quad \text{bits/symb.}$$

où les $p_{i,k}$ représentent les probabilités d'occurrence de chacun des C^k blocs de k caractères. En théorie, plus k est élevé, plus on se rapproche de la valeur optimale, i_∞ .

Par exemple, dans le cas de la suite, '001001001001001…', manifestement très peu porteuse d'information, on calcule successivement :

$i_1 = 1$ bit/symb, $i_2 = 1$ bit/symb, $i_3 = \ell g 3/2$ bit/symb, $i_4 = i_5 = \ldots = 0$ bit/symb.

Notons que si la suite est simplement biaisée, les probabilités d'occurrence des symboles de l'alphabet étendu sont directement héritées des p_i et on a, $i_k = i_1$, pour tout k. Autrement dit, on ne commet aucune erreur en égalant systématiquement la limite théorique de l'entropie statistique par symbole à la valeur, i_∞, mais on effectue certainement un travail superflu dans le cas simplement biaisé puisque le calcul de i_1 suffirait.

Le principe de Jaynes.

La notion d'entropie statistique apporte une solution élégante à un problème fréquent, celui du calcul des probabilités a posteriori d'un ensemble d'épreuves lorsque des mesures expérimentales ont amélioré la connaissance de la distribution a priori. C'est le principe de Jaynes qui fournit l'outil de choix dans l'approche Bayésienne du calcul des probabilités.

Le principe de Jaynes :
«La meilleure estimation qu'on puisse faire de l'état d'un système dont on n'a qu'une connaissance partielle s'obtient en maximisant l'information manquante, I(s), sous les contraintes connues ».

Un exemple fera comprendre de quoi il s'agit. Soit un dé à 6 faces d'apparence honnête, la question posée est la suivante : quelle est la probabilité qu'il soit pipé ?

1er cas : on ne sait rien du dé sauf qu'il a l'air honnête. On entend par là qu'il semble respecter les propriétés de symétries sans lesquelles il ne serait pas difficile de suspecter qu'il est probablement pipé. Le principe de Jaynes est d'application simple dans un tel cas, il suffit de maximiser l'information manquante,

$$I = -N\sum_{i=1}^{6} p_i \ell g p_i$$

sous la seule contrainte que la somme des probabilités affectées à chaque face soit égale à l'unité,

$$\sum_{i=1}^{6} p_i = 1$$

C'est un problème classique de maximisation sous contrainte dont la solution se règle par la méthode des multiplicateurs de Lagrange.

$$L = -\sum_{i=1}^{6} p_i \ell g(p_i) + \lambda\left(1 - \sum_{i=1}^{6} p_i\right)$$

On calcule l'extremum en résolvant le système :

$$\partial_{p_i} L = \partial_{\lambda} L = 0$$

et la solution s'écrit :

$$p_1 = p_2 = \cdots = p_6 = \frac{1}{6}.$$

C'est, sans surprise mais aussi sans hypothèse superflue, la distribution uniforme.

2ème cas : on a lancé honnêtement le dé 10000 fois pour une moyenne de 2 au lieu de 3.5 attendu. Autant dire qu'on suspecte un dé pipé. Le principe de Jaynes impose à nouveau de maximiser

$$I = -N\sum_{i=1}^{6} p_i \ell g p_i$$

sous les contraintes,

$$\sum_{i=1}^{6} p_i = 1 \qquad \text{et} \qquad \sum_{i=1}^{6} i \cdot p_i = 2$$

Cette fois, le lagrangien s'écrit :

$$L = -\sum_{i=1}^{6} p_i \ell g(p_i) + \lambda\left(1 - \sum_{i=1}^{6} p_i\right) + \mu\left(2 - \sum_{i=1}^{6} i \cdot p_i\right)$$

La solution qui n'introduit aucune hypothèse parasite s'obtient à partir de :

$$\partial_{p_i} L = \partial_{\lambda} L = \partial_{\mu} L = 0$$

et elle vaut :

$$p_1 = 0.478 \quad p_2 = 0.255 \quad p_3 = 0.136 \quad p_4 = 0.072 \quad p_5 = 0.039 \quad p_6 = 0.021$$

On pourrait trouver ce calcul incongru dans la mesure où si l'on a lancé le dé 10000 fois, il suffisait de dresser la statistique des coups joués pour connaître approximativement les probabilités cherchées. Mais ce luxe n'est pas toujours accessible et c'est précisément ce qui se produit, en mécanique statistique, lorsqu'on évalue les distributions des positions et des vitesses des molécules dans un gaz. Ceci nous mène à la thermodynamique.

Distribution spatiale des molécules dans un gaz parfait.

Notons, p_i, la probabilité de présence d'une molécule dans la $i^{ème}$ cellule de l'espace des positions. Le principe de Jaynes requiert de maximiser :

$$I = -N\sum_{i=1}^{n} p_i \ell g(p_i)$$

sous la seule contrainte,

$$\sum_{i=1}^{n} p_i = 1 .$$

La réponse est connue, c'est la distribution uniforme. Comme on s'y attendait, aucune position dans l'enceinte n'est privilégiée.

Distribution des vitesses des molécules dans un gaz parfait.

Divisons, à présent, l'espace des vitesses en cellules et notons, p_i, la probabilité de présence d'une molécule dans la $i^{ème}$ cellule. Il faut, cette fois, maximiser :

$$I = -N\sum_{i=1}^{n} p_i \ell g(p_i)$$

sous les contraintes,

$$\sum_{i=1}^{n} p_i = 1 \text{ et } \sum_{i=1}^{n} p_i E_i = U ,$$

où les énergies sont uniquement de nature cinétique, $E_i = \frac{1}{2} m v_i^2$. La méthode lagrangienne donne cette fois :

$$L = -N \sum_i p_i \ell g(p_i) + \lambda(1 - \sum_i p_i) + \mu(U - \sum_i p_i \frac{1}{2} m v_i^2)$$
$$\partial_{p_i} L = \partial_\lambda L = \partial_\mu L = 0$$

dont la solution restitue la distribution classique de Maxwell :

$$p_i \propto \frac{\exp[-\alpha v_i^2]}{\sum_i \exp[-\alpha v_i^2]}$$

Entropie physique : statistique, algorithmique ou rien de tout cela ?

La théorie classique de l'information définit deux grandeurs « entropie », l'une algorithmique et l'autre statistique. L'entropie statistique d'un système représente l'information manquante à propos de ce système. L'entropie algorithmique coïncide avec la description minimale du système considéré. La question se pose de savoir si ces notions abstraites ont un rapport quelconque avec l'entropie définie par les physiciens dans le cadre restreint de la thermodynamique.

Certains pensent qu'il n'y en a pas et que tant Kolmogorov que Shannon ont commis un abus de langage en utilisant le terme entropie pour baptiser ces grandeurs. Mais d'autres et non des moindres, Feynman, Bennett, Landauer ou Zurek pensent exactement le contraire. Nous adoptons l'idée que ce rapport existe et que c'est l'entropie algorithmique qui formalise l'entropie physique. Nous adoptons le postulat suivant :

Pour un observateur informé, qui connaît la description d'un système à la tolérance, δ, l'entropie physique est assimilable à l'entropie algorithmique.

$$S = S_{algo} = K + c_\delta \qquad \Rightarrow \qquad \Delta S = \Delta S_{algo} = \Delta K$$

On a par certainement, dans ce cadre,

$$S(NaCl) << S(Cl_2) ,$$

ou encore,

$$S(\Sigma_{froid}) << S(\Sigma_{chaud})$$

Remarque : l'assimilation de l'entropie physique à l'entropie algorithmique a une conséquence inattendue sur l'additivité de cette grandeur. L'interprétation Boltzmanienne de l'entropie nous a habitué à considérer qu'il s'agit effectivement d'une grandeur additive. Or, dans l'optique algorithmique, cela n'est pas toujours vrai. Certes, pour les couples de systèmes complètement aléatoires, il est exact que la description la plus courte est

identique à la somme des descriptions in extenso. Cela peut toutefois cesser d'être vrai lorsque les systèmes sont corrélés car la description de l'un peut, au moins en partie, servir à le description de l'autre, ce qui offre une possibilité de compression donc d'une réduction d'entropie algorithmique. Cette situation apparaît chaque fois qu'un appareil de mesure enregistre la description d'une partie d'un système, il construit un fichier qui est sa réplique fidèle et les parties corrélées partagent une entropie commune. Ce n'est que lorsqu'on décorrèle le système de l'appareil de mesure que l'on rend éventuellement leur autonomie entropique à chacun.

Un observateur « Ignorant » le détail de la description d'un système physique à la tolérance, δ, n'a pas accès à K mais uniquement à

$$I = S_{stat} \geq S_{algo} = K$$

Autrement dit, Ignorant peut tenter de quantifier l'entropie physique en l'assimilant à l'entropie statistique mais il la surévaluera à coup sûr si la suite qui décrit le système est structurée par un programme court. Par contre, si le système n'est pas structuré, les entropies algorithmique et statistique coïncident et il ne faut pas chercher ailleurs les raisons du succès de la thermodynamique de Boltzman en rapport avec l'étude des systèmes désordonnés.

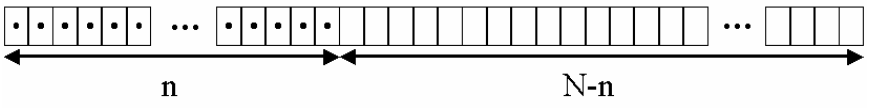
Les systèmes physiques dont la suite descriptive est structurée par un programme court peuvent toutefois être traités de manière statistique à condition qu'ils soient chaotiques. On montre, en effet, que dans ce cas la sensibilité aux conditions initiales est telle qu'elle provoque plus ou moins rapidement une croissance de l'entropie algorithmique du système jusqu'à la limite de Shannon prévue par le modèle statistique. La surestimation de l'entropie n'est donc que momentanée dans ce cas et elle disparaît lorsque le système rejoint son état d'équilibre.

Par contre, les systèmes non chaotiques structurés par un programme court ne peuvent en aucune manière prétendre être traités avec succès par la voie statistique.

Croissance et décroissance de l'entropie : le gaz de Szilard multi-cloisonné.

Considérons un gaz suffisamment dilué, comportant n molécules cloisonnées dans N cellules avec, N>>>n, en sorte qu'on ne trouve jamais qu'une molécule par cellule. Codons la présence d'une molécule dans une cellule par '1' et son absence par '0'. L'état du gaz est alors représentable par une suite, s^N, de longueur, N, comportant n '1' et (N-n) '0'.

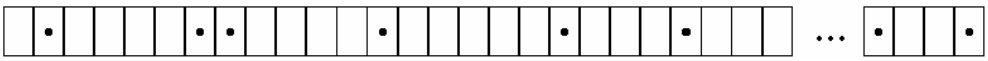
Préparons ce gaz multi-cloisonné dans un état initial très particulier :



Son entropie est basse, de l'ordre de,

$$S_1 = S_{algo} = K(s) = K(\underset{n\ fois}{111...1}\underset{(N-n)\ fois}{0000000\cdots000}) \approx \ell g(n) + \ell g(N-n)$$

Enlevons les cloisons puis replaçons-les, ce qui a pour effet probable de brouiller l'ordre moléculaire. La plupart du temps, on observe effectivement que les molécules sont repiégées sans obéir à une quelconque loi de répartition :



On peut calculer l'entropie du gaz dans ce nouvel état : elle correspond à l'entropie statistique calculée par la formule de Shannon en accord avec le schéma de probabilités,

$$p('1') = \frac{n}{N} \qquad p('0') = \frac{N-n}{N},$$

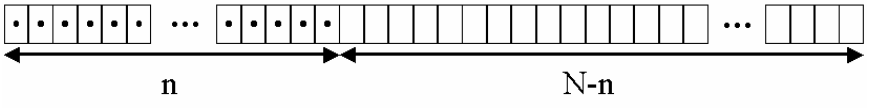
soit :

$$S_2 = S_{algo} = S_{stat} = N\left(-\frac{n}{N}\ell g\frac{n}{N} - \frac{N-n}{N}\ell g\frac{N-n}{N}\right) \approx n\ell g\frac{N}{n}$$

L'approximation résulte de la relation, $\lim_{x\to 0}\frac{1-x}{x}\frac{\ell g(1-x)}{\ell g(x)}=0$: lorsque n<<<N, seul le premier terme importe.

On constate que l'entropie du gaz a augmenté, $\Delta S = S_2 - S_1 >> 0$, jusqu'à atteindre sa valeur maximum à l'équilibre qui correspond à l'encodage « in extenso » des coordonnées moléculaires.

Si l'on ôte puis replace les cloisons un grand nombre de fois, la suite qui encode les positions des molécules reste la plupart du temps non structurée et l'entropie reste stable à sa valeur d'équilibre, $\Delta S = 0$. Exceptionnellement, on observera un retour de Poincaré :



$$S_3 = S_{algo} \approx \ell g(n) + \ell g(N-n) \qquad (\Delta S = S_3 - S_2 << 0)$$

ou n'importe quelle autre configuration structurée par un programme court donc de basse entropie. On voit que l'entropie de l'univers enregistre une baisse brutale dans ce cas. Cependant, ces événements sont rares cas les suites structurées sont infiniment moins nombreuses que les autres. De plus, si d'aventure la baisse d'entropie se produit, elle est extrêmement éphémère car l'évolution chaotique reprend très vite ses droits lorsqu'on réôte les cloisons. Cela dit, le fait demeure que, stricto sensu, l'entropie de l'univers peut diminuer.

Le théorème de retour, tel que démontré par Poincaré, est une certitude programmée dans le temps. Ce théorème a plongé la communauté scientifique dans un embarras profond que Boltzman n'a pas vraiment dissipé en objectant que, pour un gaz réel, le temps de retour serait nécessairement colossal, au-delà de l'âge de l'univers. C'était, sans doute, la seule réplique qu'il pouvait apporter à l'époque mais le fait demeure qu'elle éludait la question de principe qu'on ne nettoie pas en balayant les poussières sous le tapis dans l'espoir qu'on ne les découvrira pas de sitôt. Il n'est donc pas opportun, précisément pour une question de principe, d'énoncer que l'entropie de l'univers ne diminue jamais vu que l'existence des fluctuations prouve, à elle, seule le contraire.

Cela dit, répétons que ces fluctuations sont extrêmement rares, imprédictibles et éphémères, en sorte qu'on peut néanmoins affirmer que l'entropie physique d'un système isolé ne diminue « presque » jamais.

Entropie du gaz parfait dilué.

Jusqu'à présent nous avons négligé la contribution des vitesses des molécules à la complexité du gaz. On peut combler cette lacune. Pour ce faire, il suffit de s'intéresser à la répartition des n molécules entre les N cellules de l'espace des phases, numérotées dans un ordre prédéterminé. Ce nombre N vaut :

$$N \approx \frac{V}{\Delta V}\left(\frac{v}{\Delta v}\right)^3 >>> n$$

Ce gaz étant supposé désordonné, la formule de Shannon s'applique encore et on retrouve l'expression classique de l'entropie dite de Sackur-Tetrode :

$$S = n\left(\ell g \frac{V}{n\Delta V} + \frac{3}{2}\ell g \frac{v^2}{(\Delta v)^2}\right) = n\left(\ell g \frac{V}{n} + \frac{3}{2}\ell g T\right) + c$$

Connexion avec l'entropie de Kelvin.

Ramenées à une molécule, l'énergie interne et l'entropie moyennes d'un gaz parfait désordonné sont données par les relations :

$$u = \sum_{i=1}^{N} p_i E_i$$

$$s = -k\ell n 2 \sum_{i=1}^{N} p_i \ell g p_i$$

où les probabilités, p_i, obéissent à la loi de Boltzmann, $p_i \propto \exp[-E_i / kT]$.

En différentiant ces relations, on obtient successivement :

$$du = \sum_{i=1}^{N} p_i dE_i + \sum_{i=1}^{N} E_i dp_i = \delta w + \delta q$$

$$ds = -k\left(\sum_{i=1}^{N} \ell n(p_i) dp_i - \sum_{i=1}^{N} dp_i\right) = \frac{\delta q}{T}$$

On retrouve l'expression de la variation d'entropie d'un système lors d'un échange de la chaleur avec le milieu extérieur.

Application à l'analyse de textes.

Bien que cela nous éloigne complètement des applications physiques, il est intéressant d'illustrer la compression au sens de Shannon par l'exemple d'une analyse d'un texte rédigé en langue anglaise. Il est évident que toutes les lettres n'y apparaissent pas avec la même fréquence : un 'e' est plus fréquent qu'un 'z' en sorte que l'apparition d'un 'e' est moins porteuse d'information que celle d'un 'z'.

Plus précisément, en réduisant pour simplifier l'alphabet à 26 lettres plus le blanc, chaque symbole possède sa fréquence propre d'apparition : 0.2 pour le blanc, 0.105 pour 'e', 0.072 pour 't', 0.0654 pour 'o', etc, en sorte que l'information par symbole qui vaudrait $i_0 = \ell g(27) = 4.76$ bits/symb si tous les caractères étaient équiprobables, tombe à i_1=4.03 bits/symb du fait de la contrainte d'apparition des lettres.

Mais bien d'autres contraintes pèsent sur la langue anglaise : un 't' est souvent suivi d'un 'h', un 'q' suivi d'un 'u', etc, … , en sorte que l'apparition de la deuxième lettre n'est, dans ce contexte, guère porteuse d'information. De telles corrélations entre caractères distants exigent que l'on calcule i_k pour des valeurs croissantes de k. Une étude statistique détaillée d'un ensemble représentatif de textes anglais indique que lorsque k tend vers l'infini, i_k tend vers 2 bits/symb.

Le test de Shannon.

Pour la petite histoire, il convient de noter que la valeur de i_k=2 bits/symb est encore surestimée dans le cas particulier d'un texte porteur de sens. Cela résulte du fait qu'il subsiste des corrélations contextuelles qui permettent de deviner certaines lettres. Le test suivant est dû à Shannon.

On propose le début d'un texte à un observateur et on lui demande de deviner une à une les lettres suivantes, blanc séparateur compris. Chaque essai équivaut à une question posée et la moyenne du nombre des essais fournit une estimation du contenu informationnel d'un texte anglais standard. Ce test est très facile à mettre en œuvre et l'expérience montre un contenu informationnel de l'ordre de 1.6 bits/symb.

On voit qu'on est très loin des 4.73 bits/symb de départ. La différence mesure la redondance du langage. L'existence d'une redondance exprime un gaspillage de bits pour l'encodage de l'information. En d'autres termes, il doit être possible de comprimer un texte redondant et la compression est achevée quand la redondance du nouveau texte est devenue nulle.

C'est la redondance qui autorise une compréhension du langage même si certains caractères, pas trop nombreux, sont corrompus lors de la transmission du message. Par contre, la corruption d'un message écrit dans un langage non redondant ou compressé est forcément redoutable.