Préliminaires

La théorie mathématique de l'information a débuté en 1924 dans le cadre des mathématiques et de la physique avec des noms comme Gabor, Hartley, Nyquist, Wiener. Elle a été developée en 1949 par Shannon et Weaver.

Pourquoi ?

A cette époque les transmissions télégraphiques demandaient d'optimiser l'utilisation des canaux de transmission offerts, et donc d'éliminer depuis les données à transmettre tout ce qui n'était pas indispensable à la compréhension.

Les méthodes classiques pour le compactage des fichiers binaires

- ce sont des opérations entièrement réversibles limitant la redondance de l'information de maniére non destructrice.

Le codage par répétition (par plages)

(Run Length Encoding)

- il indique pour chaque sous-séquence de symboles identiques la longueur de cette sous-séquence.

Exemple. *aaaabccc devient a4bc3*

Utilisation : les formats d'images BMP de Windows et OS/2, le format PCX, le fax

Le codage "à longueur variable"

(la méthode de Huffman)

- il utilise les fréquences d'apparition des symboles d'un message pour attribuer aux symboles plus fréquents des codes courts, et aux symboles plus rares des codes longs. Dans la méthode de Huffman, une étude statistique donne la fréquence des éléments "signifiants" présents dans les données originales. L'algorithme attribue à chacun de ces éléments un code binaire dont la taille est d'autant plus réduite que le code est fréquent. Ce système autorise le codage en temps réel et il est utilisé ensemble avec d'autres pour la télécopie.

Exemple. *donc pour coder un texte, "e" sera codée sur 2 bits et non sur 8, par contre "q" sera codée sur 12 bits*

Utilisation : Pour la compression avec perte, il est en général utilisé après que la redondance propre au média est mise en évidence par d'autres algorithmes : le format JPEG, MPEG et MP3. Pour la compression sans perte il est utilisé avec le codage par dictionnaires.

Le codage par dictionnaires

– il repose sur l'analyse des répétitions dans les données à traiter. Il ne s'agit pas de rechercher des occurrences d'éléments signifiants considérés comme élémentaires, mais des chaînes de longueur variable. Les chaînes répétées prennent place dans un dictionnaire, et chacune est remplacée, dans le résultat compressé, par sa seule adresse dans le dictionnaire.

Exemple. *"si tu bois vite ta boisson" devient "si tu bois vite ta 1son"*

Cette méthode est intéressante pour un gros fichier, son efficacité étant dépendante de la taille du dictionnaire par rapport à/aux fichiers à compresser. La sous-classe des méthodes Lempel-Ziv (LZ, LZx) exploite conjointement un algorithme statistique issu de Huffman ou de Shannon-Fano, mais en évitant l'analyse préalable de l'ensemble du fichier, ce qui permet un encodage en une seule passe.

Utilisation : De très nombreux programmes de tous types utilisent les algorithmes LZ (Arc, Pkzip, Lharc, Arj).

Idée commune :

Toutes ces méthodes essayent de deviner ce qui vient après. Le résultat du compactage montre à quel point les methodés étaient surprises par ce qui eut suivi.

– RLE voit un caractère et en étant simplette elle se dit : je pense que le TOUT lui resemble. Si le prochain caractère est identique, la méthode ne vas pas stocker quelque chose parce que elle n'est pas surprise.
– Huffman assigne des probabilités aux caractères possibles. Si elle voit un caractère commun, comme « e », elle produit peu d'octets. Si elle voit un caractère rare, comme le « ç », elle est sidérée et produit beaucoup d'octets.
– LZ veut croire que l'entrée se compose de sequences qu'elle a déjà vues. Si tel est le cas, la méthode est modérément étonnée et donc elle sort un code court. Si par contre elle rencontre une
séquence qu'elle n'a pas vu auparavant, alors le code produit peut être plus long que la séquence qu'il représente.

C'est pourquoi les données aléatoires ne se compriment pas (impossible de prévoir ce qui vient après), et pourquoi les données déjà comprimées se compriment rarement plus (la grande partie de la surprise a été enlevée).

Conclusion. *La valeur d'un algorithme de compression de données est mesurée en termes de sa capacité de prévoir le futur.*

Pourquoi ne pouvons-nous pas juste balayer le fichier entièr et trouver la manière optimale de le compacter ?

Parce-que c'est un problème NP-complet et parce-que l'information accumulée pendant le balayage doit être stockéee quelquepart, d'habitude dans la sortie de la méthode.

Cours/TD 1

Definition. *Une source d'information est un système capable de sélectionner et d'émettre des séquences de lettres (ou mots) appartenant à un ou alphabet donné. Une source d'information discrète utilise un alphabet fini.*

Les lettres peuvent être des lettres proprement dites, chiffres, échantillons.

Classification :

- Sources sans mémoire : lettres générées indépendamment les uns des autres ⇒ modèle de Bernoulli
- Sources avec mémoire : prise en compte de la dépendance entre une lettre émise et les lettres précédentes ⇒ modèle de Markov Une source avec mémoire est du k-ème ordre si la mémoire se limite au k dernières lettres émises

À partir de maintenant nous supposerons toujours d'avoir effectué l'évaluation statistique nécessaire pour le démarrage des algorithmes de compactage. Supposant connue la distribution de probabilité qui décrit un ensemble de données discrètes produites par une source sans mémoire, les algorithmes de compactage proprement dits sont déterministes.

1.1 Sources discrètes et leur entropie.

On représente un ensemble de donées à compacter comme une source discrète **sans mémoire**, produisant des (suites de) lettres (des mots) d'un alphabet $\{ a_0, a_1, \ldots a_{N-1} \}$ à N symboles.

Definition. *On appelle* $p_j = p(a_j) \equiv$ *la probabilité de la production de la lettre* $a_j \quad 0 \leq j \leq N-1$.

Notation. $\mathbf{p} = (p_0, p_1, \ldots p_{N-1}) \equiv$ *la distribution de probabilité qui décrit la production de la source en question.*

Exemples de sources discrètes :

- Soit $\{0,1\}$ l'alphabet d'une source binaire : les pixels allumés et éteints de l'écran d'un traitement de texte.
- Soit l'alphabet $\{00000000, 00000001, \ldots 11111111\}$ d'une source à 256 symboles, codés en octets binaires : le code ASCII, par exemple.

Rappel. *L'hypothèse de la production* **sans mémoire est très forte.** *Elle demande que a chaque instant le choix d'une lettre de l'alphabet ne depends pas des chois antérieurs. Donc il faut que pour chaque mot* $w = a_{j_1}a_{j_2}\ldots a_{j_n}$ *de longueur* n *ses lettres soient produites de manière* **indépendante** *i.e.*

$$p(w) = p(a_{j_1})p(a_{j_2})\cdots p(a_{j_n})$$

La modélisation permet de décider ce que sera une **lettre**, c.àd. un atome de notre alphabet.

Exemples des choix d'alphabet :

Les mots $\{12123333123333123312, 1233, 3312\}$ peuvent être produits par une source discrete soit sur l'alphabet $\{1,2,3\}$ soit sur l'alphabet $\{12,33\}$. Pour que cette source soit considèrée comme une source discrète sans mémoire, il faut choisir l'alphabet pour lequel après une analyse statistique de tout l'ensemble de données nous avons que $p(3312) = p(1233)$ et $p(12123333123333123312) = [p(12)]^5[p(33)]^5$.

Les mots $\{\ 0000000000000001, 0000000111111111\ \}$ peuvent être produits par une source binaire ou par une source discrete sur l'alphabet $\{\ 01\ , 10\ \}$, ou bien sur l'alphabet des octets binaires.

1.1.1 L'entropie d'une source

L'entropie d'un mot est une mesure de la quantité d'information contenue dans un message.

En physique, l'entropie d'un système est une mesure du désordre dans le systéme. Dans la théorie de l'information l'entropie nous dit a quel point les donées d'éntrée sont surprenantes. Le résultat le plus intéressant que nous pouvons dériver de ceci est que, si nous pouvons calculer l'entropie d'une source de données, nous pouvons déduire quelle sera la plus petite taille des données compactées.

Conditions « naturelles » sur la définition de $I(w)$

où $I(w)$ est la quantité d'information contenue dans un mot w produit par la source.

1. $I(w)$ est inversément proportionnelle à la probabilité $p(w)$ de la production de w (« plus c'est rare, plus c'est intéressant »)

2. le contenu d'information d'un fait **sûr** n'est pas surprenant et donc l'information est égale à **zéro**.
3. l'information est toujours positive, i.e. $I(w) \geq 0$
4. il y a au moins une information utile, i.e. il existe $0 < \alpha \leq 1$ et $I(\alpha) > 0$
5. $I(a_{j_1} a_{j_2} \dots a_{j_n}) = I(a_{j_1}) + I(a_{j_2}) + \dots I(a_{j_n})$ (l'information dans un mot est la somme des informations dans ses lettres – ceci est la consequence logique de l'hypothèse de la source sans mémoire, i.e. qui produit des lettres statistiquement indépendantes)

En synthétisant, nous avons que $I(w) = F(\frac{1}{p(w)})$ où la fonction réelle F doit satisfaire :

1. $F(1) = 0$ et
2. F est croissante et $\exists \alpha, 0 < \alpha \leq 1$ avec $F(\alpha) > 0$ (plus simplement F strictement croissante)
3. $F(x \cdot y) = F(x) + F(y)$

Lemme. *Il y a une seule fonction continue F qui satisfait les conditions imposées à F : c'est le logarithme.*

Ébauche de preuve. *Si l'on suppose maintenant que le domaine de definition de F est symétrique par rapport à 0 et qu'il contient 1, alors toute solution F est une fonction paire. En effet, $F(1) = F(1 \cdot 1) = 2F(1)$ donc $F(1) = 0$.*

C'est pourquoi on va chercher à trouver toutes les fonctions solutions pour le domaine de definition $\mathbb{R}_+^*$. Si nous cherchons des fonctions qui sont derivables, nous dérivons $F(z \cdot y) = F(z) + F(y)$ par rapport à y et puis nous posons $y = 1$. Donc nous arrivons à la condition nécessaire $zF'(z) = F'(1)$ et $F(1) = 0$ qui est exactement la definition du logarithme népérien, i.e. la primitive sur $\mathbb{R}_+^*$ de la fonction inverse $x \to 1/x$ s'annulant en 1.

Si nous supposons qu'elle est seulement continue. $F(4) = 2F(2)$ et en general $F(z^2) = 2F(z)$ Par induction, tout entier positif $F(z^n) = F(z^{n-1}z) = F(z^{n-1}) + F(z) = (parinduction)(n-1)F(z) + F(z)$ Pour les puissances entières negatives : $0 = F(1) = F\left(\frac{z^p}{z^p}\right) = F(z^p) + F(z^{-p})$ Donc $F(z^{-p}) = -pF(z^p)$.

En écrivant $z = z^{p/p} = \left(z^{1/p}\right)^p$ nous avons $F(z) = pF\left(z^{1/p}\right)$. Ainsi pour un nombre rationnel arbitraire m/p nous avons $F\left(z^{\frac{m}{p}}\right) = \frac{m}{p} F(z)$.

Nous allons prouver que F et une fonction logarithmique coincident dans α.

Nous avons le Théorème : Si I est un intervalle réel et f une fonction continue de I dans $\mathbb{R}$, alors $f(I)$ est un intervalle. Quand b parcourt $(1, \infty)$, $b \mapsto log_b \alpha = \frac{ln\alpha}{lnb}$ parcourt $(0, \infty)$. Donc il existe un b dans $(1, \infty)$ pour lequel $F(\alpha) = log_b(\alpha)$. Comme dans le paragraphe précedent, nous avons que $F(\alpha^r) = log_b(\alpha^r)$ pour tout r rationnel.

Pour obtenir l'égalité avec le logarithme pour tout nombre réel dans l'intervalle $(0, 1)$, nous devons utiliser quelques propriétés de densité des nombres rationnels dans l'ensemble

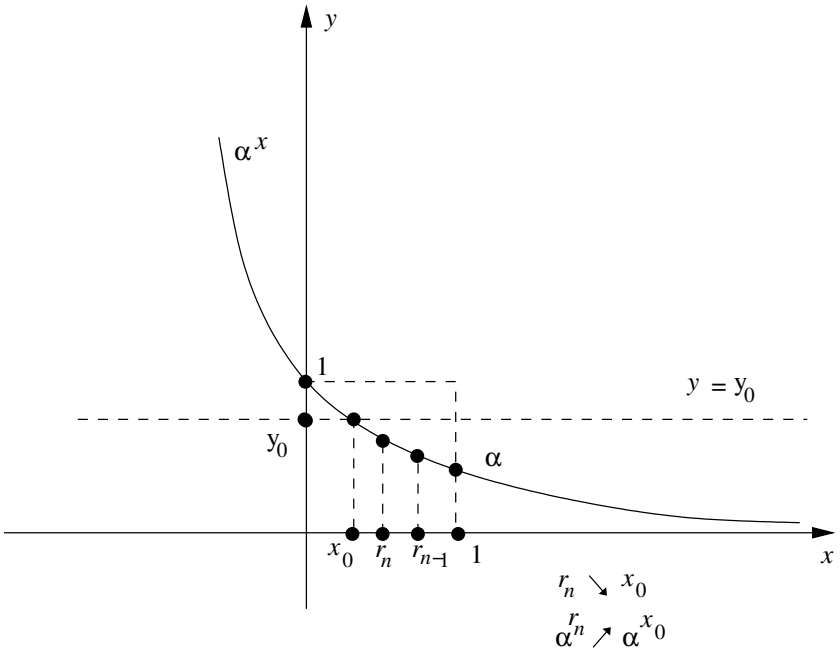

FIG. 1.1 – Graphe de la fonction $x \mapsto \alpha^x$ avec $\alpha \in (0,1)$

de nombre réelles. Pareil si nous voulions verifier directement la propriété logarithmique pour toute puissance réelle $F(z^x) = xF(z)$.

Soit un réel positif arbitraire $y_0 \in \mathbb{R}_+$. *Le graphe de la fonction* $x \mapsto \alpha^x$, *avec* $\alpha \in (0,1)$ *intersecte la ligne* $y = y_0$, *soit* (x_0, y_0) *le point d'intersection ou* $\alpha^{x_0} = y_0$ – *voir figure 1.1.*

See figure.

L'ensemble $\mathbb{Q}$ *des nombres rationnels est dense dans* $\mathbb{R}$. *Tout* x_0 *réel est la limite d'une suite monotone décroissante des rationnels de decomposition décimale par excès* $r_n = \frac{p_n}{q_n} = a_0 + \frac{a_1}{10} + \ldots + \frac{a_{n-1}}{10^{n-1}} + \frac{1}{10^n}$.

Nous utilisons aussi

– *Théorème des suites monotones bornées. Une suite bornée et monotone est convergente.*

– *Unicité de la limite : Une suite convergente a une limite unique.*

Comme $x \mapsto \alpha^x$ *est une fonction décroissante la suite* α^{r_n} *est croissante et majorée donc elle a une limite unique. Parce que* $x \mapsto \alpha^x$ *est une fonction continue* $lim_{n\to\infty}\alpha^{r_n} = \alpha^{x_0} = y_0$.

Parce que F *est croissante* $F(\alpha^{r_n})$ *est monotone et majorée donc elle a une limite unique. Comme* F *est une fonction continue la limite de* $F(\alpha^{r_n})$ *est* $F(\alpha^{x_0})$. *Nous avons aussi* $F(\alpha^{r_n}) = \frac{p_n}{q_n}F(\alpha) = \frac{p_n}{q_n}log_b\alpha = log_b\alpha^{r^n}$. *En passant à la limite* $F(y_0) = F(\alpha^{x_0}) = x_0F(\alpha) = x_0log_b\alpha = log_b\alpha_0^x = log_b y_0$.

□

Definition. $I(w) = log_2(\frac{1}{p(w)}) = -log_2 p(w)$.

Rappel. $y = log_2 x \iff x = 2^y \iff x = e^{y\ln 2} \iff y = \frac{\ln x}{\ln 2}$.

Pourquoi le logarithme en base 2 ?
Réponse : Le choix de la base b done l'unité d'information. Pour la base $b = e$, nous avons **le nat**. Quand $b = 2$ **l'unité** de la quantité d'information est **le bit**, défini comme la quantité d'information fournie par le choix d'une alternative parmi deux équiprobables.

Voici deux exemples confirmant le choix de l'unité d'information :

1. Considérons une source qui produit $a_0 = pile$ et $a_1 = face$ avec la même probabilité : $\mathbf{p} = (p_0, p_1) = (\frac{1}{2}, \frac{1}{2})$. On aura $I(a_0) = I(a_1) = -Log_2 2^{-1} = 1$.
   C'est logique : Dans un cas équiprobable, *pile* "vaut" 0 et *face* "vaut" 1 et on peut coder les choix sur un bit d'information.
2. Considérons une source qui produit 256 entiers entre 0 et 255 avec la même probabilité : $\mathbf{p} = (p_0, p_1, \ldots, p_{255}) = (\frac{1}{256}, \frac{1}{256}, \ldots, \frac{1}{256})$. On aura $I(a_0) = I(a_1) = \cdots = I(a_{255}) = -Log_2 2^{-8} = 8$.
   C'est logique : Dans un cas équiprobable, l'information contenue dans chacun des nombres $0, 1, \ldots, 255$ est de 8 bits : ce sont des octets binaires !

L'entropie est donc **un multiplicateur pour un codage binaire virtuel** :
1000 symboles produits (statistiquement corrects) par la source "valent"
1000 $\times$ l'entropie bits.

Definition (de **l'entropie** de la source)**.** *Soit une source discrete et sans mémoire. Soit* $\mathbf{p} = (p_0, p_1, \ldots, p_{N-1})$ *la distribution de probabilité qui décrit la production (indépendante) des lettres de l'alphabet de la source. L'entropie d'une source est* **l'information moyenne** *par symbole venant de la source (en bits par symbole).* $H(\mathbf{p}) \equiv p_0 I_0 + p_1 I_1 + \ldots + p_{N-1} I_{N-1} = -p_0 \log_2 p_0 - p_1 \log_2 p_1 - \ldots - p_{N-1} \log_2 p_{N-1}$

Exercice. *Calculer l'entropie de la source qui produit les huit lettres* $a_0, a_1, \ldots, a_7$ *selon la distribution de probabilité* $\mathbf{p} = (\ p_0, p_1, \ldots, p_7\ )$
avec $p_0 = \frac{1}{2}$, $p_1 = \frac{1}{4}$, $p_2 = p_3 = \frac{1}{16}$, $p_4 = p_5 = p_6 = p_7 = \frac{1}{32}$.

Solution. $H(\mathbf{p}) = -\sum_{0 \le i \le N-1} p_i \log_2 p_i = \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 2^2 + \frac{1}{8}\log_2 2^4 + \frac{1}{8}\log_2 2^5 = 2\frac{1}{8}$

Exercice. (La fonction binaire de l'entropie – $h(p)$) *Soit une source binaire X qui produit* $a_0 = blanc$ *et* $a_1 = noir$ *selon la distribution de probabilité* $\mathbf{p} = (\ p, 1-p\ )$, *avec* $0 < p < 1$.
Trouver la condition sur p (i.e. la relation blanc/noir) qui force $H(\mathbf{p}) < \frac{1}{2}$.

Ébauche de solution. *Pour* $0 < p < 1$ *l'entropie de la source X est* $H(X) \equiv h(p) = -p\log_2 p - (1-p)\log_2(1-p)$. *La fonction $h(p)$ est symétrique* $h(p) = h(1-p)$ *et concave.* $h' = -\frac{1}{\ln 2}(\ln p - \ln(1-p))$ *et* $h'' = -\frac{1}{\ln 2}\left(\frac{1}{p} + \frac{1}{1-p}\right) \le 0$ *Nous avons utilisé que le changement de base du logarithme est donné par* $\log_a(x) = \frac{\ln(x)}{\ln a}$. *L'unique point stationnaire ou la fonction atteint soit son maximum absolu est* $p = 1/2$.

p	$-p\log_2 p$	$-(1-p)\log_2(1-p)$	$h(p)$ *(bits)*
0	*0*	*0*	*0*
.05	*.216*	*.070*	*.286*
.10	*.332*	*.137*	*.469*
.11	*.350*	*.150*	*.500*
.15	*.411*	*.199*	*.610*
.20	*.464*	*.258*	*.722*
.25	*.500*	*.311*	*.811*
.30	*.521*	*.360*	*.881*
1/3	*.528*	*.390*	*.918*
.35	*.530*	*.404*	*.934*
.40	*.529*	*.442*	*.971*
.45	*.518*	*.474*	*.993*
.50	*1/2*	*1/2*	*1*

Les propriétés de la fonction binaire de l'entropie suggèrent certaines propriétés de l'entropie discrète en general.

Remarque. *Soit l'alphabet* $\{\ a_0, a_1, \dots a_{N-1}\ \}$ *et on fait varier ses distributions de probabilité*

1. $H(\mathbf{p}) = 0 \iff$ *La source ne produit réellement qu'une seule lettre (p.ex. la lettre* a_0*), avec* $p(a_0) = 1$.
   Donc : L'entropie est **minimale** *(est nulle) lorsque la production de la source est constante.*
2. $H(\mathbf{p})$ *est maximale* $\iff p_0 = p_1 = \cdots = p_{N-1} = \frac{1}{N}$ *(cas équiprobable)*
   Dans ce cas, on a $H(\mathbf{p}) = \log_2 N$.

Ébauche. 1. *Pour* $p(a_0) = 1$ *nous avons* $\log_2 p_0 = 0$, *mais il y a un souci avec la possibilité d'avoir* $p_i = 0$ *pour un certain* i. *Formellement, nous pouvons utiliser le fait que* $\lim_{p\to 0} p\log_a p = 0$ *comme une justification mathématique pour ignorer les produits avec* $p_i = 0$. *Nous avons donc que l'entropie qui est la mesure d'information de Shannon est définie seulement pour le supp*$(\mathbf{p})$, *i.e. l'ensemble des lettres de probabilité non-nulle.*

2. *Pour une preuve élémentaire de ce résultat nous avons d'abord besoin de* l'inégalité IT : Pour un nombre réel positif r, $\log_2 r \le (r-1)\log_2 e$ l'egalité ayant lieu seulement pour $r = 1$. Preuve : La fonction $g(r) = \ln r - r + 1$ a $r = 1$ comme point de maximum, donc $g(r) \le g(1) = 0$ et $\ln r \le r - 1$. Nous multiplions avec $\log_2 r$ et, sachant que $\log_2 r = (\ln r)(\log_2 e)$, nous avons l'inégalité désirée.
   Nous allons montrer que $H(p) - \log_2 N \le 0$. *On a* $H(p) - \log_2 N = \sum_{0\le i\le N-1} p_i \log_2 1/p_i - \log_2 N \sum_{0\le i\le N-1} p_i = \sum_{0\le i\le N-1} p_i \log_2 1/Np_i$. *Appliquant l'inégalité IT à chaque* $z =$

$1/(Np_i)$, *on obtient que*

$$H(p) - \log_2 N \leq \log_2 e \sum_{0 \leq i \leq N-1} p_i \left(\frac{1}{Np_i - 1} \right)$$

$$= \log_2 e \left(\sum_{0 \leq i \leq N-1} p_i \left(\frac{1}{N} - \sum_{0 \leq i \leq N-1} p_i \right) \right) = 0$$

avec égalité si et seulement si $1/(Np_i) = 1$ *pour tout* i.
Une solution plus élégante peut être obtenue avec les multiplicateurs Lagrange. L'extremum lié à trouver est un point de maximum pour la fonction de plusieurs variables $H(p) = f(p_1, ..., p_n)$ *avec les variables liées par la relation* $\sum_{0 \leq i \leq N-1} p_i = 1$. *Ayant une seule contrainte, on ajoute un seul multiplicateur de Lagrange.*

Annexe sur les multiplicateurs de Lagrange

Soit le problème d'optimisation :

$$\min H(\mathbf{p}) \text{ où } \mathbf{p} = \{p_0, p_1, \dots p_N - 1\}$$

soumis à des contraintes d'égalité $g_k(\mathbf{p}) = 0 \qquad k = 0, 2, ...m - 1.$

La méthode des multiplicateurs de Lagrange permet de trouver des points stationnaires (maximum, minimum...) d'une fonction derivable d'une en N variables sous $m < N$ contrainte(s) (linéaire(s)). Elle consiste à introduire une inconnue scalaire supplémentaire – appelée multiplicateur de Lagrange – par contrainte ; il y a donc m multiplicateurs. On forme ensuite une combinaison linéaire de la fonction et des contraintes, où les coefficients des contraintes sont les multiplicateurs de Lagrange $L(\mathbf{p}) = H(\mathbf{p}) - \sum_{k=0}^{m-1} \lambda_k g_k(\mathbf{p})$. Le problème passe ainsi d'un problème d'optimisation avec contrainte à un problème non contraint $L(\mathbf{p}) = 0$, qui peut être résolu par le gradient conjugué.

Donc, on atteint la valeur optimale de $H(\mathbf{p})$ pour les valeurs p_i de $\mathbf{p}$ qui satisfont le système :

$$\begin{cases} \frac{\partial L(\mathbf{p})}{\partial p_0} = \frac{\partial H(\mathbf{p})}{\partial p_0} - \sum_{k=0}^{m-1} \lambda_k \frac{\partial g_k(\mathbf{p})}{\partial p_0} &= 0 \\ \dots & \\ \frac{\partial H(\mathbf{p})}{\partial p_{N-1}} - \sum_{k=0}^{m-1} \lambda_k \frac{\partial g_k(\mathbf{p})}{\partial p_n} &= 0 \\ \frac{\partial L(\mathbf{p})}{\partial \lambda_0} = g_0(\mathbf{p}) &= 0 \\ \dots & \\ \frac{\partial L(\mathbf{p})}{\partial \lambda_m} = g_m(\mathbf{p}) &= 0 \end{cases}$$

Notre problème est

Maximiser

$\max H(\mathbf{p}) = -\sum_{k=0}^{N-1} (p_i log_2 p_i)$ où $\mathbf{p} = \{p_0, p_1, \dots p_N - 1\}$

soumis à une seule contrainte d'égalité $g_0(\mathbf{p}) = 1 - \sum_{k=0}^{N-1} p_i = 0.$

Ayant une seule contrainte, on ajoute un seul multiplicateur de Lagrange :λ

Le problème revient maintenant à maximiser le Lagrangien $L(H, \lambda) = H(\mathbf{p}) + \lambda \left(1 - \sum_{k=0}^{N-1} p_i\right)$

On cherche donc à annuler le gradient du Lagrangien, soit : $\nabla L = 0$, ce qui nous amène au système : $\begin{cases} \frac{\partial L}{\partial p_i} = -\frac{1}{\ln 2}(1 + \ln p_i) - \lambda = 0 (1) \\ \frac{\partial L}{\partial \lambda} = 1 - \sum_{k=0}^{N-1} p_i = 0 (2) \end{cases}$ L'équation (1) nous donne $\ln p_i = -1 - \lambda \ln 2,$ pour tout $i = 0, 2, ...N - 1$ donc $p_1 = \dots = p_{N-1}$. En introduisant p_i dans l'equation (2) nous obtenons en fin $p_i = \frac{1}{N}$.