Université Constantine 1

Master systèmes de communication

Licence Télécommunication

Théorie de l'information et codage

Par Benierbah Saïd

Cours 1: mesure de l'information

"*The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.*" Shannon

Introduction

Un système de communication a pour but de reproduire en un point (exactement ou approximativement) le message choisi à un autre point.

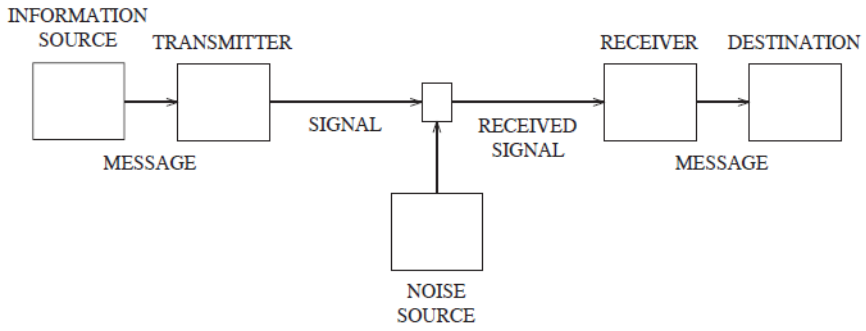
La théorie de l'information a pour objet d'évaluer mathématiquement les performances limites des systèmes de communication, en présence de perturbations aléatoires.

La théorie de l'information s'intéresse essentiellement aux limitations théoriques et aux possibilités (potentielles) de communication. Par exemple: quelle est la meilleure performance de correction d'erreur que l'on peut avoir? Elle ne donne pas des indications sur la réalisation pratique des systèmes qui peuvent atteindre ces limites.

La théorie du codage concerne la création de systèmes de codage et décodage pratiques.

Modélisation d'un système de communication:

Shannon qui a été le premier à donner une description mathématique rigoureuse du problème de communication, a modélisé un système de communication comme le montre la figure suivante (paradigme de Shannon):



La communication se fait comme suit :

Un message est généré par une source d'information. Le message est converti par l'émetteur en un signal facile à transmettre à travers un canal. Durant la transmission, ce signal est contaminé par une source de bruit. Donc le signal reçu peut être différent du signal transmit. En se basant sur le signal reçu, le récepteur effectue une estimation du message et le délivre à sa destination.

Dans ce modèle de communication, on s'intéresse seulement au problème de savoir si le message généré par la source est délivré à sa destination sans erreurs. On ne s'intéresse pas au sens "sémantique" du message ni à ce que le récepteur va faire avec. Il faut aussi noter que ce modèle de communication n'est pas seulement valable pour une communication entre deux points distants où l'information va d'un point à un autre. Écrire un fichier sur un disque et le lire après (écrire et lire sont à deux instants différents) est une autre forme de communication qui peut être décrite par ce modèle.

Modélisation d'une source:

Shannon a fait une observation clé que *"une source d'information doit être modélisée comme un processus aléatoire"*.

Définitions: une **source** d'information discrète émis des symboles discrets aléatoires.

A chaque instant i, la source émis un seul symbole $X = x_i$ de l'ensemble des M symboles possibles : $x_1, x_2, \ldots, x_M \in \mathcal{X}$ appelé **alphabet**.

La fréquence d'apparition de chaque symbole est donnée par les probabilités :

$P_X(x_1), P_X(x_2), \ldots, P_X(x_M)$ avec $P_X(x_i) = Pr\{X = x_i\}$

ou $P(x_i)$ seulement, pour simplifier la notation et alléger l'écriture des équations.

La source est considérée comme un processus aléatoire et les propriétés de X sont étudiées de la même façon que les variables aléatoires (*v. a.*).

-Une source est dite sans mémoire: si x_i ne dépend pas des autres valeurs (précédentes) produites par la source $x_{i-1}, x_{i-2}, \ldots, x_1$. Mathématiquement ceci veut dire :

$\Pr(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = \Pr(X_{n+1} = x_{n+1})$

Ou $P(x_{n+1} | x_n, x_{n-1}, \ldots, x_1) = \Pr(x_{n+1})$

-Une source **iid (identically independent distribution)**: si tous les symboles x_i ont la même distribution (identique) de probabilité et ils sont indépendants.

-Une source **markovienne**: un processus stochastique est dit de Markov (Markovien) ou constituant une chaîne de Markov si pour $n = 1, 2, \ldots$, on a:

$\Pr(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = \Pr(X_{n+1} = x_{n+1} | X_n = x_n)$

Pour tout $x_1, x_2, \ldots, x_n, x_{n+1} \in \mathcal{X}$.

Dans ce cas, la densité de probabilité conjointe de ces *v.a.* peut être écrite comme:

$p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2 | x_1)p(x_3 | x_2) \cdots p(x_n | x_{n-1})$.

Mesure de l'information:

Mesurer la quantité d'information que contient un message a toujours été est un problème difficile. L'information n'est pas une entité physique mais un concept abstrait et difficile à quantifier surtout quant un facteur humain est inclus.

Pour surmonter ce problème, Shannon a donné l'idée de définir le contenu en information d'un événement comme une fonction *h(x)* qui dépend seulement de sa probabilité. Donc, soit deux v.a.: *X* avec une distribution $p_X(x)$ (ou seulement *p(x))* et *Y* avec une distribution $p_Y(y)$. La v.a. *X* prend ces valeurs dans un alphabet $\mathcal{X}$ et *Y* prend ces valeurs dans $\mathcal{Y}$.

Pour choisir la mesure de l'information, on ajoute les axiomes suivants:

-cette fonction *h(x)* doit être décroissante. Parce que le plus un événement est probable moins il apporte de l'information.

- un événement sûr ne porte aucune information. Si *p(x)=1, h(x)=0*.

- pour deux événements indépendants, l'information totale est la somme de l'information de chaque événement.

La seule fonction qui satisfait ces axiomes est la fonction logarithmique. Donc:

$$h(x) = \log \frac{1}{p(x)} = -\log(\, p(x))$$

Donc pour deux v.a. indépendants *x* et *y*, l'information totale obtenue est :

$$h(x;y) = \log \frac{1}{p(x;y)} = \log \frac{1}{p(x)p(y)} = \log \frac{1}{p(x)} + \log \frac{1}{p(y)}$$

Donc elle satisfit la relation: $h(x;y) = h(x) + h(y)$

Les mesures principales de l'information sont: l'entropie, l'entropie conjointe, l'entropie conditionnelle et l'information mutuelle. Dans ce qui suit, on va les introduire et voir leurs propriétés.

L'entropie:

L'entropie d'une v.a. est une mesure de son incertitude; elle mesure la quantité d'information moyenne nécessaire pour décrire la v.a.

Définition: l'entropie *H(X)* d'une variable aléatoire discrète *X* est définie par:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log(\, p(x))$$

log a une base 2 et l'entropie est mesurée en bits. Si la base est *e* l'unité est *nats*. L'unité *Shannon* est parfois utilisée pour éviter la confusion avec bits (binary digits)

Remarque importante : L'entropie ne dépend pas des valeurs de la v.a. X mais seulement des probabilités.

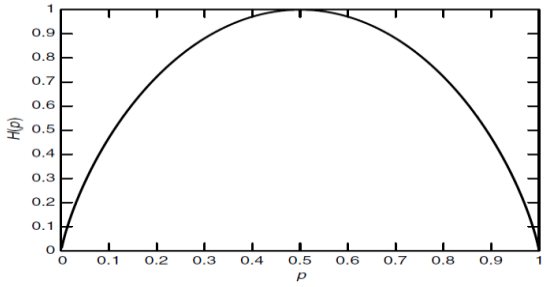
Exemple 1:

Une v.a. X qui prend ces valeurs dans {0, 1}. X=1 avec probabilité p et X=0 avec probabilité 1-p. Calculer l'entropie de X

En appliquant la définition, on obtient:

$$H(X)=-\sum_{x\in\chi} p(x)\log(p(x))=-p\log(p)-(1-p)\log((1-p))=H(p)$$

H(p) est appelé fonction d'entropie binaire. Sa variation en fonction de la probabilité p est donnée par:



Cette fonction varie en fonction de p. En particulier, si p vaut 1 ou 0, *H(p)* = 0. La fonction est symétrique par rapport à p = 0.5, et maximale =1 en cette même valeur.

Exemple 2: on considère une source qui produits 32 symboles avec une distribution de probabilités uniforme. L'entropie est donnée par:

$$H(X)=-\sum_{i=1}^{32} p(i)\log(p(i))=-\sum_{i=1}^{32}\frac{1}{32}\log(\frac{1}{32})=\log(32)=5\,bits.$$

Propriétés de *H(X)*:

1. $H(X) \geq 0$

Démonstration : $0 \leq p(x) \leq 1 \Rightarrow \log(\frac{1}{p(x)}) \geq 0$

2. Changement de base:

$$H_b(X) = (\log_b(a))H_a(X)$$

Démonstration: $\log_b(p) = \log_b(a)\log_a(p)$

3. L'entropie est maximale lorsque toutes les valeurs ont la même probabilité $p_i = \frac{1}{|A_X|}$ avec A = {x : $p(x)$ > 0} groupe de support de $p(x)$ et $|A_X|$ le nombre d'éléments de A.

Entropie conjointe et conditionnelle:

Pour deux v.a., l'entropie conjointe est définie par :

$$H(X,Y) = -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y)\log(\,p(x,y))$$

$$H(X,Y) = -E\log(\,p(x,y))$$

On défini l'entropie conditionnelle d'une v.a. *Y* connaissant une autre v.a. *X* comme la valeur moyenne des entropies des distributions conditionnelles, moyennées sur la v.a. de condition.

$$\begin{aligned} H(Y/X) &= \sum_{x\in\mathcal{X}} p(x)H(Y/X=x) \\ &= -\sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} p(y/x)\log(\,p(y/x)) \\ &= -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y)\log(\,p(y/x)) \\ &= -E\log(\,p(Y/X)) \end{aligned}$$

Cette grandeur est aussi appelée équivocation de *Y* par rapport à *X*. Elle mesure en effet l'incertitude résiduelle ou le caractère ambigu ou équivoque de *Y*, lorsqu'on connaît *X*.

Théorème: (Chain rule) $H(X,Y) = H(X) + H(Y|X)$.

Corollaire: $H(X,Y|Z) = H(X|Z) + H(Y|X,Z)$.

Pour $X_1, X_2, \ldots, X_n$ produits avec probabilité conjointe $p(x_1, x_2, \ldots, x_n)$. Donc:

$$H(X_1,X_2,\ldots,X_n) = \sum_{i=1}^{n} H(X_i/X_{i-1},\ldots,X_1)$$

Définition: **l'information mutuelle** entre deux v.a. *X* et *Y* est définie comme:

$$\begin{aligned} I(X;Y) &= \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y)\log(\frac{p(x,y)}{p(x)p(y)}) = D(\,p(x,y)\|(p(x)p(y))) \\ &= E_{p(x,y)}\log(\frac{p(X,Y)}{p(X)p(Y)}) \end{aligned}$$

Elle mesure la quantité d'information que contient une v.a. *X* sur une autre v.a *Y*. C'est la réduction de l'incertitude sur une v.a. due à la connaissance de l'autre.

Relation entre entropie et information mutuelle:

$I(X;Y) = H(Y) - H(Y|X) = H(X) - H(X/Y)$.

Donc, l'information mutuelle $I(X;Y)$ est la réduction de l'incertitude de *X* due à la connaissance de *Y*. il faut aussi noter que la quantité d'information que *X* nous donne sur *Y* est la même que donne *Y* sur *X* : $I(X;Y) = I(Y;X)$

En supposant que X représente le message émis par la source et Y celui reçu par le destinataire, cette décomposition revient à dire que l'information mutuelle est égale à l'information émise, diminuée par l'indétermination du symbole émis qui subsiste quand le symbole reçu est connu. Symétriquement, elle est égale à l'information reçue, diminuée de l'indétermination du symbole reçu qui subsiste quand le symbole émis est connu.

L'information mutuelle est non négative: $I(X;Y) \geq 0$, avec égalité si et seulement si X et Y sont indépendants.

Nous avons aussi: $I(X;Y) = H(X) + H(Y) - H(X,Y)$.

Théorème : (avoir une condition réduit l'entropie)

$$H(X|Y) \leq H(X)$$

Avec égalité si et seulement si X et Y sont indépendants.

Démonstration: $0 \leq I(X;Y) = H(X) - H(X|Y)$.

Intuitivement, le théorème dit que connaissant une autre v.a. Y ne peut que réduire l'incertitude sur X.

La borne de l'indépendance sur l'entropie : Soit $X_1,X_2,...,X_n$ obtenus selon $p(x_1,x_2,...x_n)$.

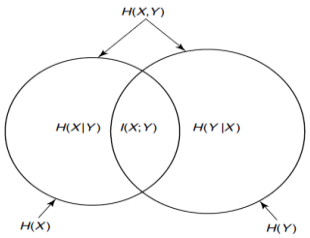
Donc: $H(X_1, X_2,...,X_n) \leq \sum_{i=1} H(X_i)$

Avec égalité si et seulement si les X_i sont indépendants.

$H(X) \leq \log |X|$, Avec égalité si et seulement si X est distribué uniformément sur $\mathcal{X}$.

Nous avons aussi les relations suivantes:

$I(X;Y) = H(X) - H(X/Y)$.
$I(X;Y) = H(Y) - H(Y/X)$.
$I(X;Y) = H(X) + H(Y) - H(X,Y)$.
$I(X;Y) = I(Y;X)$.
$I(X;X) = H(X)$.
$I(Y;Y) = H(Y)$.



Noter que l'information mutuelle $I(X;Y)$ correspond à l'intersection de l'information dans X et l'information dans Y.

Data-processing inequality (non creation d'information). Si $X \rightarrow Y \rightarrow Z$ forment une chaîne de Markov, donc:

$$I(X;Y) \geq I(X;Z).$$

Aucun traitement, déterministe ou aléatoire, de Y ne peut augmenter la quantité d'information qu'il contient sur X. En d'autres termes, il est impossible de créer de l'information au moyen de manipulations quelconques de traitement de données.

TD 1

EXERCISE 1

Sur un réseau numérique le signal de parole est échantillonné à 8 kHz (largeur de bande maximale < 4 kHz par le théorème d'échantillonnage de Nyquist). Chaque échantillon du signal de parole est ensuite quantifié sur 256 niveaux d'amplitude. Si on utilise un codage binaire avec mots à taille fixe pour chaque échantillon, calculer la taille de chaque mot binaire le débit binaire correspondant en kb/s.

EXERCISE 2

(a) En s'aidant de la formule $p(x, y) = p(x)p(y|x)$, montrer la même formule conditionnée par z : $p(x, y|z) = p(x|z)p(y|x, z)$

(b) En déduire que $X \rightarrow Y \rightarrow Z$ forme une chaîne de Markov si et seulement si X et Z sont indépendants sachant y, i.e. : $p(x, z|y) = p(x|y)p(z|y)$

(c) Montrer que si $X \rightarrow Y \rightarrow Z$ est une chaîne de Markov, alors la chaîne "réciproque" $Z \rightarrow Y \rightarrow X$ l'est aussi.

EXERCISE 3

(a) Montrer que la fonction logarithme est strictement concave.

(b) En déduire l'inégalité de Jensen : $\mathbf{E}\,[\log 2\, f(X)] <= \log 2 \mathbf{E}\,[f(X)]$
avec égalité si et seulement si la fonction f est constante.

EXERCISE 4

Supposez que nous avons n pièces de monnaie, avec une seule pièce fausse: pèse moins que les autres. Quelle est l'incertitude moyenne associée à l'opération de trouver la pièce fausse?

On utilise une balance pour comparer des groupes de pièces. On aura un des 3 cas:
- A est plus lourd que B
- A et B ont le même poids
- A plus légère que B

Proposer une stratégie efficace pour détermine la pièce fausse rapidement.

EXERCISE 5

On considère l'entropie $H(X)$ d'une v.a. discrète X.

-Montrer que $H(X)$ est une «auto-information» $H(X) = I(X,X)$. Interpréter ce résultat.

-Montrer que $H(X) >= 0$ et déterminer le cas d'égalité.

-Montrer et interpréter :$H(Y|X) <= H(Y)$.

-Montrer que : H(X;Y) = H(X) + H(Y /X) = H(Y) + H(X/Y): Interpréter.

Monter que si X et Y sont indépendants, leurs information mutuelle est zéro

EXERCISE 6

1. Soit la table de contingences suivante:

	Y1	Y2
X1	1/3	1/3
X2	0	1/3

Calculer :

- H(X), H(Y), H(X/Y), H(Y/X), H(X;Y), H(Y)-H(Y/X), I(X;Y).

Cours 2: théorème de séparation

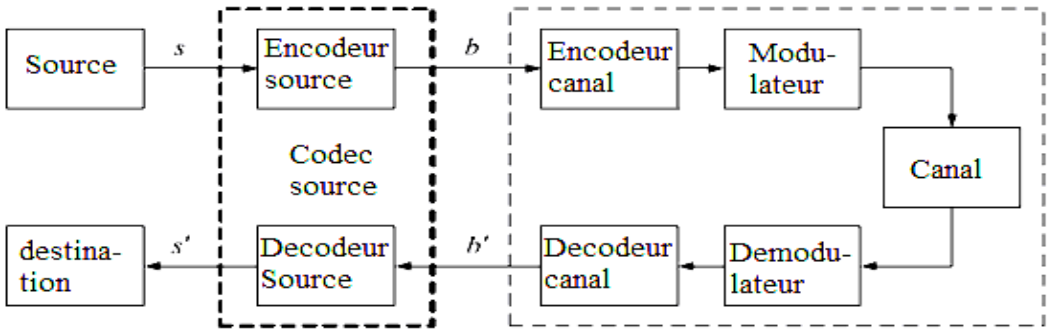
A communique avec B veut dire l'opération physique par laquelle A agit sur B pour lui induire un état désiré. Cette opération sera affectée par un bruit incontrôlable dû à la transmission des signaux. La communication est réussite si la récepteur B et l'émetteur A sont d'accord sur ce qui a été transmit.

Coder est l'opération de représenter un échantillon de la source par une représentation binaire plus facile à transmettre. Cela veut dire trouver les opérations de traitement qui permettent de communiquer avec le minimum nombre de bits (compression) et avec le minimum d'erreurs (codes correcteur d'erreurs). Une seule opération peut atteindre ces deux buts mais elle sera très complexe à réaliser. Pour faciliter ce travail, le système de communication peut être séparé en deux parties. Une partie qui s'occupe de la compression et une partie qui concentre seulement sur la transmission et la protection contre les erreurs. Par exemple, pour envoyer les échantillons d'une source sur un canal on peut imaginer un système qui donne directement des mots binaires pour chaque échantillon ou utiliser un système pour compresser les échantillons d'une façon plus efficace puis utiliser un autre code pour le protéger, d'une façon plus efficace, contre le bruit, pour l'envoyer sur le canal.

Puisque cette méthode de séparation est plus pratique, elle est largement utilisée et tous les systèmes de communication l'utilisent. Pour étudier les systèmes de communication nous allons aussi suivre cette séparation et étudier en premier les méthodes de compression appelées aussi "codage source" et les méthodes de protection de l'information lors de sa transmission aussi appelées "codage canal".

Il faut noter qu'il y a une relation de dualité entre le codage source et codage canal. En effet durant la compression on essaye de réduire le nombre de bits utilisés (réduire la redondance) alors que le codage canal ajoute des bits (d'une façon contrôlée) pour combattre les erreurs.

La figure suivante montre la structure d'un système de communication typique:



La source génère un signal *s* l'encodeur source transforme le signal *s* en un flot binaire *b* des bits sont transmis sur un canal de communication et reçus comme un flot binaire *b′*. Ces bits seront ensuite traités par le décodeur source pour reconstruire le signal *s′* et le délivre à la destination.

Pour étudier le codage source, on va ignorer l'effet du bruit de canal et considérer que *b′=b* et concentrer seulement sur la réduction du nombre de bits.

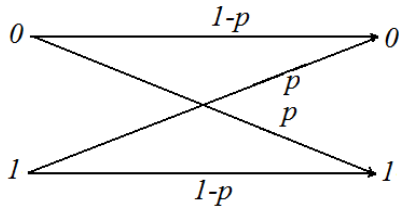
Pour la partie transmission sur canal, le codage canal ajoute d'autres bits à ceux de la source. Le modulateur converti ces bits en signaux analogiques faciles à transmettre sur un canal physique de communication (air, câble, fibre, ...). Le démodulateur fait l'opération inverse et récupère les bits qui seront ensuite décodés par le décodeur canal pour retrouver les bits *b′*.

Pour l'étude des codes canal, on va seulement étudier les sorties des codeurs et décodeurs. En plus des phénomènes physiques qui influent sur la transmission, affaiblissement, écho, bruit, les opérations de modulation et démodulation ne sont pas parfaites. Puisque dans ce cours, on ne va pas étudier les techniques de modulation, on va considérer que l'ensemble des systèmes de modulation et le canal physique comme étant un canal.

Définition : on défini un *canal discret* comme un système constitué d'un alphabet d'entrée *X*, un alphabet de sortie *Y* et une probabilité de transition *p(y/x)*, qui décrit la probabilité d'observer *y* en supposant que le symbole *x* a été transmis. Le canal est dit *sans mémoire* si la distribution de probabilité de la sortie dépend seulement de l'entrée au même moment et elle est conditionnellement indépendante des entrées et des sorties des symboles précédents.

Dans la suite, on va décrire les modèles simplifies de canaux de communication les plus communs.

1- canal binaire symétrique (BSC):



C'est un canal binaire dans le quel les symboles d'entrée sont changés avec une probabilité *p*. quant une erreur se produit, 0 est reçu comme 1 et vice versa. Les bits reçus n'indiquent pas ou les erreurs se sont produites, rendant ainsi tous les bits non

fiables. Si on considère x le bit d'entrée et y le bit de sotie du canal la relation entre l'entrée et la sortie est donnée par:

$$p(y=0|x=0)=1-p; \qquad p(y=0|x=1)=p;$$
$$p(y=1|x=0)=p; \qquad p(y=1|x=1)=1-p.$$

2- Le canal gaussien:

Soit x, l'entrée discrète du canal gaussien. On appelle canal gaussien, le canal liant le signal d'entre au signal de sortie selon la loi.

$$y=(-1)^x+N$$

Où N est une variable aléatoire suivant une loi gaussienne $N(0;\sigma^2)$. La sortie y est une variable continue.

Le canal gaussien est un modèle pour plusieurs canaux de communication comme le téléphone, avec ou sans fil et satellite. Le bruit additif à ce canal peut être dû à plusieurs causes mais le théorème de limite centrale indique que la somme de plusieurs effets aléatoires peut être approximés par une distribution gaussienne.

Cours 3: codage source

Le but du codage source est de représenter les symboles de la source avec le nombre minimum de bits. Le principe de base de la compression est d'assigner des descriptions courtes pour les symboles de la source les plus fréquents et des descriptions nécessairement longues pour les symboles les moins fréquents.

Définition: un code source C pour une v.a. X est une application de $\mathcal{X}$, de X, vers D^*, un ensemble de chaînes de symboles de longueur finie d'un alphabet D-aire.
Soit $C(x)$ le mot de code correspondant à x et $l(x)$ dénote la longueur de $C(x)$.
On peut considérer que l'alphabet D-aire est $D = \{0, 1, \ldots, D-1\}$. Un code binaire est une application de $\mathcal{X}$ vers $\{1,0\}^n$. D=2 pour un alphabet binaire.

Définition: une extension C^* d'un code C est une application de $\mathcal{X}$, vers des chaînes de longueur finie D, définis par:

$$C(x_1x_2 \cdots x_n) = C(x_1)C(x_2) \cdots C(x_n),$$

où $C(x_1)C(x_2) \cdots C(x_n)$ est la concaténation des mots de code correspondants.
Exemple : si $C(x_1) = 00$ et $C(x_2) = 11$, alors $C(x_1x_2) = 0011$.

Les caractéristiques importantes d'un code sont:
- Chaque chaîne codée doit avoir un décodage unique.
- Les symboles doivent être faciles à décoder.
- Avoir la plus grande compression possible.

Un code $C(X)$ est à décodage unique (sans ambiguïté) s'il n'y a pas deux chaînes de symboles distinctes qui produisent le même résultat de codage.
Le code d'un symbole est considéré comme facile à décoder si il est possible de l'identifier (identifier sa fin) le plus tôt possible (instantané). Ce qui veut dire qu'aucun mot de code n'est le préfixe d'un autre mot de code.

Exemple: un code binaire de longueur fixe est le suivant:

$X = \{a, b, c, d\}$
$P_X = \{1/2, 1/4, 1/8, 1/8\}$

x_i	$C(x_i)$	$L(x_i)$
a	1000	4
b	0100	4
c	0010	4
d	0001	4

En utilisant ce code, le codage du message *acdbac* sera codé comme:
$C(acdbac) = 100000100001010010000010$.

Le décodage de ce code, c'est à dire, retrouver les symboles de X à partir de chaîne binaire est très facile. Il suffit de séparer la chaîne binaire bits en groupes de 4 bits et de faire la correspondance inverse $C^{-1}(x_i)$ entre les valeurs du tableau, pour avoir un décodage unique.
Un code à longueur fixe affecte R bit à chaque symbole. Comme il y a $\mathcal{X}$ symboles possibles, le nombre de bits nécessaires pour un codage unique est:
$R = \log_2(\mathcal{X})$, si $\mathcal{X}$ est une puissance de 2 et $R = [\log_2(\mathcal{X})]+1$, avec [ ] représente la partie entière, si $\mathcal{X}$ n'est pas une puissance de 2.

Pour un codage plus efficace, on utilise un codage à longueur variable.

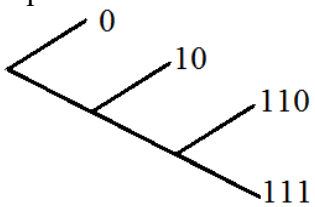
Définition: la longueur moyenne L(C) d'un code source C(x) d'un v.a. X avec une distribution de probabilité p(x) est donnée par:

$$L(C) = \sum_{x \in X} p(x) l(x)$$

Où *l(x)* est la longueur du mot de code associé à *x*.

Exemple: soit X une v.a. ayant la distribution et les mots de codes suivants:
Pr(X = 1) = 1/2, mot de code: *C(1) = 0*
Pr(X = 2) = 1/4, mot de code: *C(2) = 10*
Pr(X = 3) = 1/8, mot de code: *C(3) = 110*
Pr(X = 4) = 1/8, mot de code: *C(4) = 111.*

L'entropie de *X* est 1.75 bits et la longueur moyenne des codes *L(C) = E[l(X)]* est aussi 1.75 bits. Dans ce cas on a la longueur moyenne de codes égale à l'entropie. On peut aussi noter que toutes les séquences peuvent être à décodage unique. Par exemple la chaîne binaire 0110111100110 sera décodée à 134213.
Les codes sans préfixe peuvent être représentés par un arbre et les feuilles de l'arbre représentent les mots de code. L'arbre du code précédent est:



Exemple: soit X une v.a. ayant la distribution et les mots de codes suivants:
Pr(X = 1) = 1/3, mot de code: *C(1) = 0*
Pr(X = 2) = 1/3, mot de code: *C(2) = 10*
Pr(X = 3) = 1/3, mot de code: *C(3) = 11*

Ce code est à décodage unique, alors que l'entropie est *log(3)=1.58* bits la longueur moyenne de codes égale à *1.66* bits.
Pour que le décodage du code à longueur variable soit facile à décoder il faut qu'il soit un code sans préfixe.
Définition: un code est appelé sans préfixe ou un code instantané si aucun mot de code n'est un préfixe d'un autre mot de code.
Un code instantané peut être décodé sans faire référence (sans attendre) des mots de codes futurs.
Exemple: les mots de code de la chaîne binaire *01011111010* produite par l'exemple précédent peut être séparé à: *0,10,111,110,10*.
Il faut noter qu'un code peut être à décodage unique sans qu'il soit instantané. Dans ce cas, il faut lire toute la séquence avant de pouvoir décoder.
Inégalité de Kraft:
On veut construire des codes instantanés avec une longueur moyenne minimale. L'ensemble des longueurs des mots de code possible pour un décodage instantané est caractérisé par l'inégalité suivante:
Théorème: pour tout code instantané sur un alphabet de taille D, les longueurs des mots de codes $l_1, l_2, \ldots, l_m$ doivent satisfaire :

$$\sum_i D^{-l_i} \leq 1$$

Inversement, un ensemble de longueurs de mots de codes qui satisfont cette inégalité, il existe un code instantané avec ces longueurs de mots de code.

Théorème (théorème de codage source): la longueur moyenne L de n'importe quel code D-aire instantané pour code une v.a. X est plus grand ou égal à l'entropie $H_D(X)$:

$$L \geq H_D(X)$$

Avec égalité si et seulement si $D^{-li} = p_i$.

Le débit du code (rate) en bit par symbole est R, et comme $H(X) \leq \log_2(\mathcal{X})$ alors $R \geq H(X)$. L'efficacité du codage est définie par le rapport H(X)/R. On observe que si $\mathcal{X}$ est une puissance de 2 et les symboles sont équiprobables $R=H(X)$. Donc le code atteint une efficacité de 100%. Par contre si $\mathcal{X}$ n'est pas une puissance de 2, R est différent de $H(X)$ d'au plus 1 bit par symbole.
On peut aussi dire que l'entropie est la limite inférieure que peut avoir la longueur moyenne de tous les codes à décodage unique.

Université Mentouri Constantine Département d'électronique
Master Réseaux et télécommunication

TD 2

EXERCISE 1

Prouver l'inégalité de kraft. Pour rappeler elle dit que pour tout code instantané sur un alphabet de taille D, les longueurs des mots de codes $l_1, l_2, \ldots, l_m$ doivent satisfaire :

$$\sum_i D^{-l_i} \leq 1$$

EXERCISE 2

Prouver que la longueur moyenne L de n'import quel code D-aire instantané pour coder une v.a. X est plus grand ou égal à l'entropie $H_D(X)$:

$$L \geq H_D(X)$$

Avec égalité si et seulement si $D^{-li} = p_i$.

EXERCISE 3

Soit $l^*_1, l^*_2, \ldots, l^*_m$ les longueurs de mots de codes optimaux pour une distribution $\boldsymbol{P}$ et un alphabet D-aire et soit L^* la longueur moyenne $L^* = \sum p_i l_i^*$.

Prouver que : $H_D(X) \leq L^* < H_D(X) + 1$.

EXERCISE 4

Construire le code Huffman de la distribution:
{0.25; 0.05; 0.1; 0.13; 0.2; 0.12; 0.08; 0.07}.

EXERCISE 5

Soit une source binaire avec $p(0)$=0.9 et $p(1)$=0.1
Calculer l'entropie H.
Trouver les extensions 2,3 et 4
Trouver le code Huffman pour les 4 cas et la longueur moyenne des mots de code.
Proposer une source binaire aléatoire de longueur 30 bits qui peut être produit par cette source (elle doit contenir 9 fois plus de 0 que de 1)
Coder la séquence proposée avec le code de l'extension 4.
Introduit une erreur au 5eme bit du message codé puis décoder; quel est le résultat?

Cours 3.2: codage source

Le théorème du codage source dit qu'il y a un code C optimal de longueur variable pour l'ensemble X tel que la longueur moyenne des symboles codés $L(C)$ satisfait:

$$H(X) \leq L(C) < H(X)+1.$$

$L(C) = H(X)$ seulement si la longueur du mot de code de chaque symbole est égale à son contenu en information.

Optimal dans ce cas veut dire qu'il utilise la longueur moyenne minimale. Seulement, la théorie ne dit pas comment réaliser un tel code, pratiquement. Maintenant on va voir des techniques constructives pour produire des codes optimaux. On va voir notamment le code Huffman et on va voir le code universel de Lampel-Ziv.

Le code Huffman:

L'algorithme de Huffman permet, si la distribution des probabilités est connue, de construire un code optimal pour la distribution.

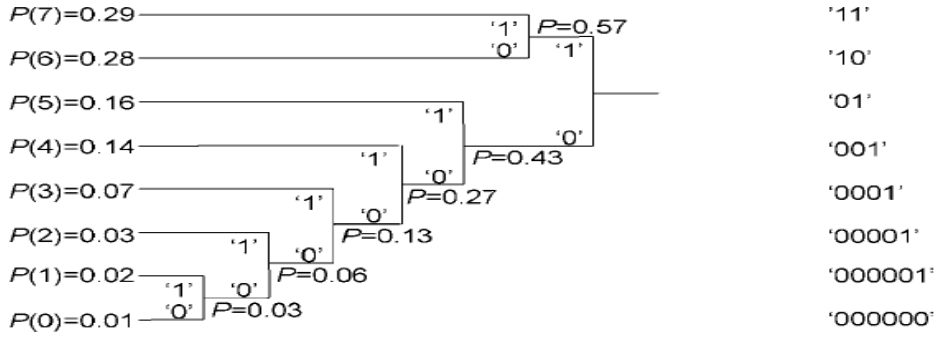
L'algorithme de Huffman qui peut être résumé comme suit:

- choisir les deux symboles x_i et x_j qui ont les probabilités les plus faibles et créer un nœud parent pour les nœuds qui représentent ces deux symboles dans l'arbre binaire du code.
- remplacer les deux symboles x_i et x_j par nouveau symbole avec une une nouvelle probabilité (la) $p(x_i)+p(x_j)$
- si il reste plus d'un symbole, répéter les étapes précédentes.
- convertir l'arbre binaire en code sans préfixe.

Exemple: considère la v.a. X = {1, 2, 3, 4, 5} avec les probabilités 0.25, 0.25, 0.2, 0.15, 0.15, respectivement. Les mots de code les plus longs seront affectés aux symboles 4 et 5. En combinant ces deux symboles, on obtient un nouveau symbole avec la probabilité de 0.30. Puis, en combinant les probabilités 0.25 et 0.20 pour produire 0.45. Puis les probabilités 0.25+0.30=0.55 et enfin 0.45+.55=1. La longueur moyenne est de 2.3 bits. Le tableau suivant montre les différentes étapes:

longueur	code	X	probabilté
2	01	1	0.25 0.3 0.45 0.55 1
2	10	2	0.25 0.25 0.3 0.45
2	11	3	0.2 0.25 0.25
3	000	4	0.15 0.2
3	001	5	0.15

Exemple: considère la v.a. X = {0,1, 2, 3, 4, 5,6,7} avec les probabilités 0.01, 0.02, 0.03, 0.07, 0.14, 0.16, 0.28, 0.29 respectivement. La construction du code Huffman est donnée par la figure suivante:



A partir de l'arbre résultant on affecte les mots de codes au symboles comme suit. On commence par le nœud de la racine puis on choisi une convention de donner un 1 pour le nœud fils supérieur et un 0 pour le nœud fils inférieur. On répète cette opération le long de tous les chemins possible de l'arbre. On affecte maintenant à chaque symbole le mot binaire construit par les bits rencontré en traversant l'arbre de la racine jusqu'à la feuille où se trouve le symbole.

Shannon-Fano Coding

On commence par ordonner l'ensemble de n symboles Xi dans un ordre décroisant. Le groupe de symboles est ensuite devisé en deux parties qui ont (presque) la même probabilité. A tous les symboles d'un groupe on affecte un code qui commence par un 0 et un code qui commence par 1 pour l'autre groupe. Chaque groupe est ensuite devisé récursivement en des sous ensembles de probabilités presque égales. Quant un sous ensemble contient deux symboles seulement, leurs codes sont distingués par l'ajout d'un bit. L'opération continue jusqu'à ce qu'il ne reste aucun sous ensemble.
Exemple:

	Prob.	Steps					Final
1.	0.25	1	1				:11
2.	0.20	1	0				:10
3.	0.15	0		1	1		:011
4.	0.15	0		1	0		:010
5.	0.10	0		0		1	:001
6.	0.10	0		0		0 1	:0001
7.	0.05	0		0		0 0	:0000

Dans la première étape: on devise les sept symboles en deux groupes. Le premier contient deux symboles avec une probabilité totale de 0.45. Le deuxième groupe contient 5 symboles et une probabilité totale de 0.55. Les deux symboles du premier groupe on un code qui commence par 1 (11 et 10)
Dans la deuxième étape: le deuxième group est devisé une autre fois en un groupe de deux symboles avec une probabilité totale de 0.3 (avec un code qui commence par 01) et un groupe de trois symboles avec une probabilité totale de 0.25 (avec un code qui commence par 00).
Dans la troisième étape: on devise les trois derniers symboles en un groupe de un symbole (avec probabilité de 0.1 et code 001) et un groupe de deux symboles de probabilité totale de 0.15 (avec un code qui commence par 000).
La longueur de mot de code moyenne est :
$0.25\times2+0.20\times2+0.15\times3+0.15\times3+0.10\times3+0.10\times4+ 0.05\times4= 2.7$ bits/symbole.
L'entropie est: 2.67.
La méthode de est simple à implémenter mais le code produit est généralement moins bon que le code produit par le code Huffman.

Codage source universel

Dans certain cas, il n'y a pas de distribution particulière représentant la source (on a une séquence de valeurs seulement). On peut donc se demander comment compresser ces données. Suppose que la v.a. X peut être représentée par une famille de distributions $\{p_\theta\}$, avec le paramètre $\theta \in \{1, 2, \ldots, m\}$ inconnu. Si on connaît θ, on peut utiliser les codes précédents, mais quoi faire si on ne connaît pas p_θ, et on veut avoir une la meilleure compression possible?

Codage Lempel–Ziv

La technique de codage Lempel-ziv (LZ) est de type codage optimal et universel (leur compression asymptotique s'approche de l'entropie pour toutes les sources ergodiques et stationnaires). Ces algorithmes (il y a deux) sont simple à implémenter.

L'idée de base des algorithmes LZ est de séparer la chaîne de symboles en groupes de symboles (phrases) et de remplaces ces groupes par des pointeurs vers une occurrence précédent dans la phrase. La différence entre les deux algorithmes LZ est sur la différence de l'ensemble de positions passée de correspondance (et longueur de correspondance) que l'algorithme tolère.

LZ 77: (algorithme à fenêtre glissante):

L'algorithme encode une chaîne de symboles en trouvant la correspondance la plus longue possible dans une fenêtre de symbole précédent (déjà codés) et représente la chaîne par un pointeur sur la position dans la fenêtre et la longueur de la correspondance.

Suppose que nous avons une chaîne de symboles $x_1 x_2 \cdots x_n$, on sépare cette chaîne en des phrases par une virgule. W est la longueur de la fenêtre.

Si on suppose que la chaîne a été compressée jusqu'à i-1. Donc, la prochaine phrase à coder est trouvée en cherchant le plus grand k tel que pour une valeur j, $i - 1 - W \leq j \leq i - 1$, la chaîne de longueur k qui commence à xj est égale à la chaîne (de longueur k) qui commence à xi. (i.e., $x_{j+l} = x_{i+l}$ pour tout l, $0 \leq l < k$).

La prochaine phrase est donc de longueur k et est représentée par les valeurs (P,L). P est la position du début de la correspondance et L est sa longueur. S'il n'y a pas de correspondance, le prochain symbole est transmit sans compression.

Pour différencier les deux cas, un flag F (1 bit) est utilisé et on obtient ainsi (F,P,L) ou simplement (F,s) ou s est un symbole non compressé.

Exemple:

si W = 4 et la chaîne de symboles à coder est : ABBABBABBBAABABA et la fenêtre initiale est vide, alors on sépare les symboles comme suit:

A,B,B,ABBABB,BA,A,BA,BA, ce qui est représenté par une séquence de pointeurs (0,A),(0,B),(1,1,1),(1,3,6),(1,4,2),(1,1,1),(1,3,2),(1,2,2), et le flag F est 0 s'il n'y a pas de correspondance et 1 s'il en a.

On peut voir cet algorithme comme si il utilise un dictionnaire qui contient tous les sous chaînes de la chaîne qui se trouve dans la fenêtre et tous les symboles uniques. L'algorithme trouve le plus grand correspondant et transmet sa position.

La plupart des programmes qui utilisent LZ77 comme gzip et pkzip, utilisent cette version de LZ77.

LZ 78: (algorithme avec arbre):

Cet algorithme sépare la chaîne en phrases. Chaque phrase est la plus courte phrase pas vue précédemment. C'est comme on construit un dictionnaire sous forme d'arbre où les nœuds correspondent aux phrases vues jusqu'ici.

Exemple: la chaîne ABBABBABBBAABABAA…, est séparée en : A,B,BA,BB,AB,BBA,ABA,BAA. . . . Après chaque virgule, on balaie toute la séquence d'entrée jusqu'à trouver la plus petite chaîne de symboles qui n'a pas été marqué auparavant. Puisque c'est la plus petite séquence, tous ces préfixes devaient avoir été vue précédemment (on construit ainsi un arbre). En particulier, la chaîne constituée de tout à part le dernier symbole doit avoir été vue précédemment. On code cette phrase en donnant la position du préfixe et la valeur du dernier symbole. On obtient ainsi (0,A),(0,B),(2,A),(2,B),(1,B),(4,A),(5,A), (3,A), ...

On peut voire la transmission du dernier symbole (non compressé) comme une perte d'efficacité de l'algorithme. On peut faire une amélioration en considérant le symbole d'extension comme une partie de la phrase suivante (proposée par Welch). LZ 78 (avec l'amélioration de Welch) est à la base des programmes de toutes les implémentations pratiques de LZ78 comme compress (unix) et le format d'images GIF.

Exemple :

On considère la séquence binaire suivante:

101011010010011101010000110011101011000011011

L'analyse de cette séquence donne:

1, 0, 10, 11, 01, 00, 100, 111, 010, 1000, 011, 001, 110, 101, 10001, 1011

On observe que chaque nouvelle phrase est une concaténation d'une phrase précédente et un nouveau symbole de la source.

Comme exemple de codage de cette séquence, en utilisant des symboles d'un code binaire: On construit le dictionnaire du tableau suivant, avec 16 positions:

	position	contenu	Mot de code
1	0001	1	00001
2	0010	0	00000
3	0011	10	00010
4	0100	11	00011
5	0101	01	00101
6	0110	00	00100
7	0111	100	00110
8	1000	111	01001
9	1001	010	01010
10	1010	1000	01110
11	1011	011	01011
12	1100	001	01101
13	1101	110	01000
14	1110	101	00111
15	1111	10001	10101
16		1011	11101

On voit sur ce tableau les positions et les différentes phrases. Les mots de code sont déterminés en donnant la position (en binaire) de la phrase précédente qui correspond à la nouvelle phrase dans tous les symboles sauf le dernier. Puis le nouveau symbole est ajouté à la position retrouvée. Initialement la position 0000 est utilisée pour coder une phrase qui n'a pas apparu auparavant. Le décodeur construit une table identique et décode les séquences reçues.

Dans cet exemple, on n'a pas de compression (44 bit sont codé en 16 mot de code de 5bit chacun). On peut voir la possibilité de cet algorithme quand la longueur de la séquence à coder est plus grande. La table va sûrement se remplir quel que soit sa longueur et il faut la vider en éliminant par exemple les phrases qui ne sont pas utiles et les remplacer par d'autres.

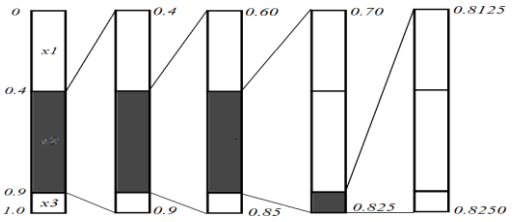
Codage Arithmétique:

L'algorithme Huffman est optimal pour une distribution connue et un codage symbole par symbole. Mais puisque les longueurs des codes Huffman sont entières, il y'aura une perte qui peut atteindre 1 bit par symbole. Pour réduire les pertes dues à ce problème, il faut coder une séquence de symboles (pas seulement un seul symbole) par une séquence de bits. Ceci est réalisé par le codage arithmétique.

Le codage arithmétique représente un symbole par un intervalle. Le codage d'une séquence de symboles est un intervalle dont la longueur décroît chaque fois qu'un nouveau symbole est ajouté. Ceci donne un code incrémental ou le code d'une extension d'une séquence peut être calculé à partir du code de la séquence originale. Puisque le code représentant l'intervalle est pour l'ensemble des symboles, le nombre de bits alloué à un seul symbole du groupe peut donc être pas entier.

Le codage arithmétique représente une séquence de v. a. par un sous intervalle de l'intervalle [0, 1]. Chaque nouveau symbole réduit cet intervalle et les bornes supérieure et inférieure de l'intervalle commencent à avoir les mêmes bits qui seront les premières sorties de l'encodeur. Chaque fois que les deux bornes de l'intervalle auront le même bit celui-ci est met à la sortie de l'encodeur. On peut ainsi changer l'échelle de l'intervalle et continuer de calculer avec une précision infinie.

Exemple 1: nous avons les symboles x_1, x_2, x_3 avec les probabilités P_1=0.4, P_2=0.5 et P_3=0.1 respectivement. L'intervalle [0, 1) est devisé entre les trois symboles en assignant à chaque symbole un intervalle de taille proportionnelle à ça probabilité. L'ordre des intervalles n'est pas important. Dans notre exemple, on utilise les intervalles [0, 0.4), [0.4, 0.9), et [0.9, 1.0). Pour encoder la suite des symboles x_2 x_2 x_2 x_3, on commence par l'intervalle [0,1). Le premier symbole x_2 réduit cet intervalle à un sous-intervalle qui commence du point à 40% jusqu'au point représentant 90% de l'intervalle produisant ainsi l'intervalle [0.4, 0.9). Le deuxième symbole qui est x_2 réduit l'intervalle précédent de la même façon et produit le nouvel intervalle [0.6, 0.85). La borne inférieur de l'intervalle est calculée par: $0.4 + (0.9 - 0.4) \times 0.4 = 0.6$ et la borne supérieur par: $0.4 + (0.9 - 0.4) \times 0.9 = 0.85$. Le troisième symbole est x_2 réduit l'intervalle précédent de la même façon et produit le nouveau intervalle [0.7, 0.825). Le quatrième symbole x_3 réduit cet intervalle à un sous-intervalle qui commence du point à 90% jusqu'au point représentant 100% de l'intervalle produisant ainsi l'intervalle [0.8125, 0.8250). Le code produit par notre code peut n'importe quel nombre dans le dernier intervalle.



Exemple 2: on va coder la suite de symboles x_1, x_2, x_3 x_1, x_1, x_4, x_5, x_3 x_1, x_1 dont les probabilités sont représentés dans le tableau suivant:

symbole	fréquence	probabilité	intervalle	fréquence accumulée
			Freq accu totale	10
x_1	5	5/10=0.5	[0.5, 1.0)	5
x_2	1	1/10=0.1	[0.4, 0.5)	4
x_3	2	2/10=0.2	[0.2, 0.4)	2
x_5	1	1/10=0.1	[0.1, 0.2)	1
x_4	1	1/10=0.1	[0.0, 0.1)	0

Les statistiques de la source sont estimées à partir du message codé. Comme dans l'exemple précédent l'ordre des symboles n'est pas important mais le code obtenu dépend de l'ordre des symboles codés. Les fréquences accumulées sont utilisées par le décodeur.

Dans une implémentation pratique, le codage commence par définir deux variables: **Low** et **High** à qui on donne 0 et 1 respectivement. Ces deux valeurs définissent les bornes de l'intervalle de codage. Apres le codage de chaque symbole, ces deux valeurs vont s'approchés. Les nouvelles bornes de l'intervalle peuvent être calculés par:
NewHigh=OldLow+Range*HighRange(*x*);
NewLow=OldLow+Range*LowRange(*x*);
Avec **Range=OldHigh−OldLow** et **LowRange(*x*)**, **HighRange(*x*)** indique les limites inférieure et supérieure de l'intervalle du symbole ***x***.
Dans notre exemple, le codage de x_1 donne **Low**=0.5 et **High**=1. Ensuite, le codage de x_2 donne **Low**=0.5+(1.0−0.5)× 0.4=0.70 et **High**=0.5+(1.0−0.5)×0.5=0.75.
La valeur finale de **Low** est 0.71753375 et **High** est 0.717535, les huit digits 71753375 de **Low** constituent la sortie du codeur.

Le décodage se fait, en se basant sur le tableau précédent, comme suit:
On entre le premier digit 7 et le décodeur va savoir immédiatement que le nombre est de la forme 0.7…. et il se trouve dans l'intervalle [0.5, 1) de x_1. Alors le premier symbole à décoder est x_1. Ensuite le décodeur élimine l'effet de x_1 du code (en soustrayant 0.5 deviser par la largeur de l'intervalle de x_1 (0.5)) le résultat est 0.4350675, ce qui indique que le prochain symbole à décoder est x_2 (intervalle [0.4, 0.5)). Cette opération s'écrit comme suit:
Code=(Code-LowRange(x))/Range, où **Range** est la largeur de l'intervalle de *x*.

Remarques:
- en pratique, ces opérations sont effectuées en binaire.
- la valeur finale n'est pas nécessairement la valeur Low; elle peut être n'importe quelle valeur entre Low et High.