

Techniques de compression pour le codage des images et des vidéos

Marc ANTONINI
Directeur de Recherche au CNRS
am@i3s.unice.fr
<http://www.i3s.unice.fr/~am>

Laboratoire I3S - Équipe CReATIVe
UMR 6070 CNRS et Université de Nice-Sophia Antipolis



Compression, Reconstruction, Adaptées au Traitement d'Images et à la Vidéo



Objectifs Du Cours

1. Quantification scalaire (QS)

- QS uniforme, QS non-uniforme
- Conditions d'optimalité
- Equations et algorithme de Lloyd-Max
- Distorsion de quantification (formule de Bennett)

2. Codage entropique

- Entropie d'une source
- Codage d'une source
- Code optimal
- Construction d'un code (Huffman, codage arithmétique)
- Codage par plage de zéros (*runlength coding*)

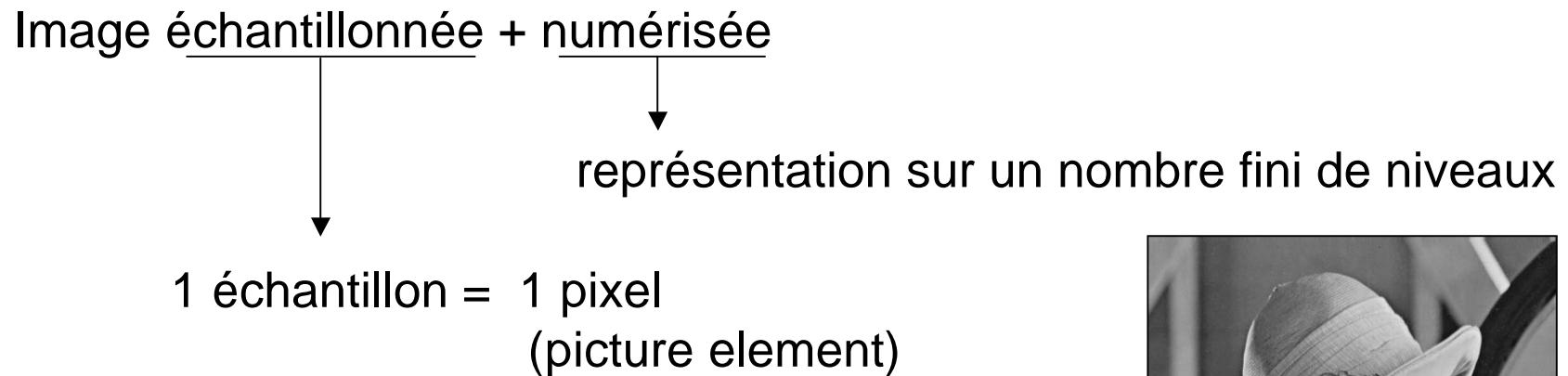
3. Quantification vectorielle

- Conditions d'optimalité de la QV
- Algorithme de Max-Lloyd généralisé, algorithme LBG
- Performances de la QV par rapport à la QS
- Distorsion de quantification (généralisation de la formule de Bennett)

4. Applications et normes

- JPEG, JPEG 2000
- MPEG 1 , MPEG 2, le futur en vidéo

La Donnée Numérique



Exemple :

- image 256 niveaux de gris
- dynamique de 0 (noir) à 255 (blanc)
- chaque niveau est représenté par 8 éléments binaires (0 ou 1)

--> 8 bits/pixel

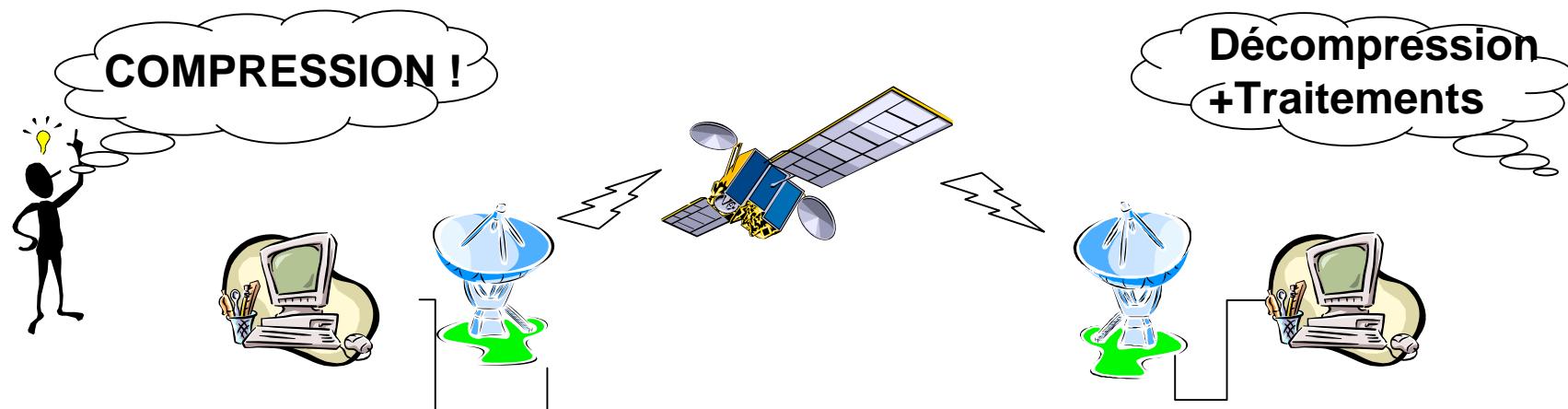
Pourquoi Le Numérique ?

Avantages :

- données faciles à manipuler (*ordinateur, ...*).
- moins sensible au bruit que l'analogique.

Inconvénients : Volume de données très important !

- problèmes de transmission (*nécessité d'une grande bande passante*).
- problèmes de stockage.



Les Différents Formats Vidéo

HDTV



Standard Definition (SD)



CIF



Compression : Motivation

Internet, Intranet

Objectifs :

Transmission de données volumineuses : fichiers, images, vidéos...

Applications :

- Bases de données
- Visioconférences (Webcam...)
- Transmission de vidéos interactives...
- Télé-médecine
- Télévision à Haute Définition (HDTV)
- Cinéma numérique

Comprimer Pour Transmettre...

... à travers les réseaux

Informatiques (Internet) :

fichiers texte, images, son, vidéo...

Téléphoniques :

voix numérisée, minitel...

Radio-mobiles :

GSM, UMTS, GSM de 3ième génération...

Satellites :

Sondes spatiales, télévision à haute définition...

Comprimer Pour Stocker...

... sur des supports du type :

Disques durs, disquettes : [fichiers](#)

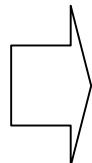
CD : sons, images

DVD (4,7 GO) : 2 heures de vidéo

« BLUE RAY » (25 GO) : [2 heures de HDTV](#)

Applications Et Contraintes

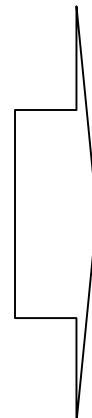
« Temps réel »



Téléphone, vidéo

COMPRESSION / DECOMPRESSION RAPIDES

« Temps différé »



Stockage sur disque (CD, CD ROM, DVD...)

COMPRESSION LENTE / DECOMPRESSION RAPIDE

Imagerie satellitaire ou embarquée

COMPRESSION RAPIDE / DECOMPRESSION LENTE

Applications Et Contraintes

Médical

Pas d'artefact (erreur de diagnostic)

Militaire

- Conservation des détails (déttection de cibles)
- Aspect mouvement (suivi de mobiles)

Vidéo « grand public »

Effet de masquage de l'œil
(espace et temps)

Vision par ordinateur

Détection des contours
(guidage d'un robot...)

Position Du Problème

Les performances d'un système de compression sont :

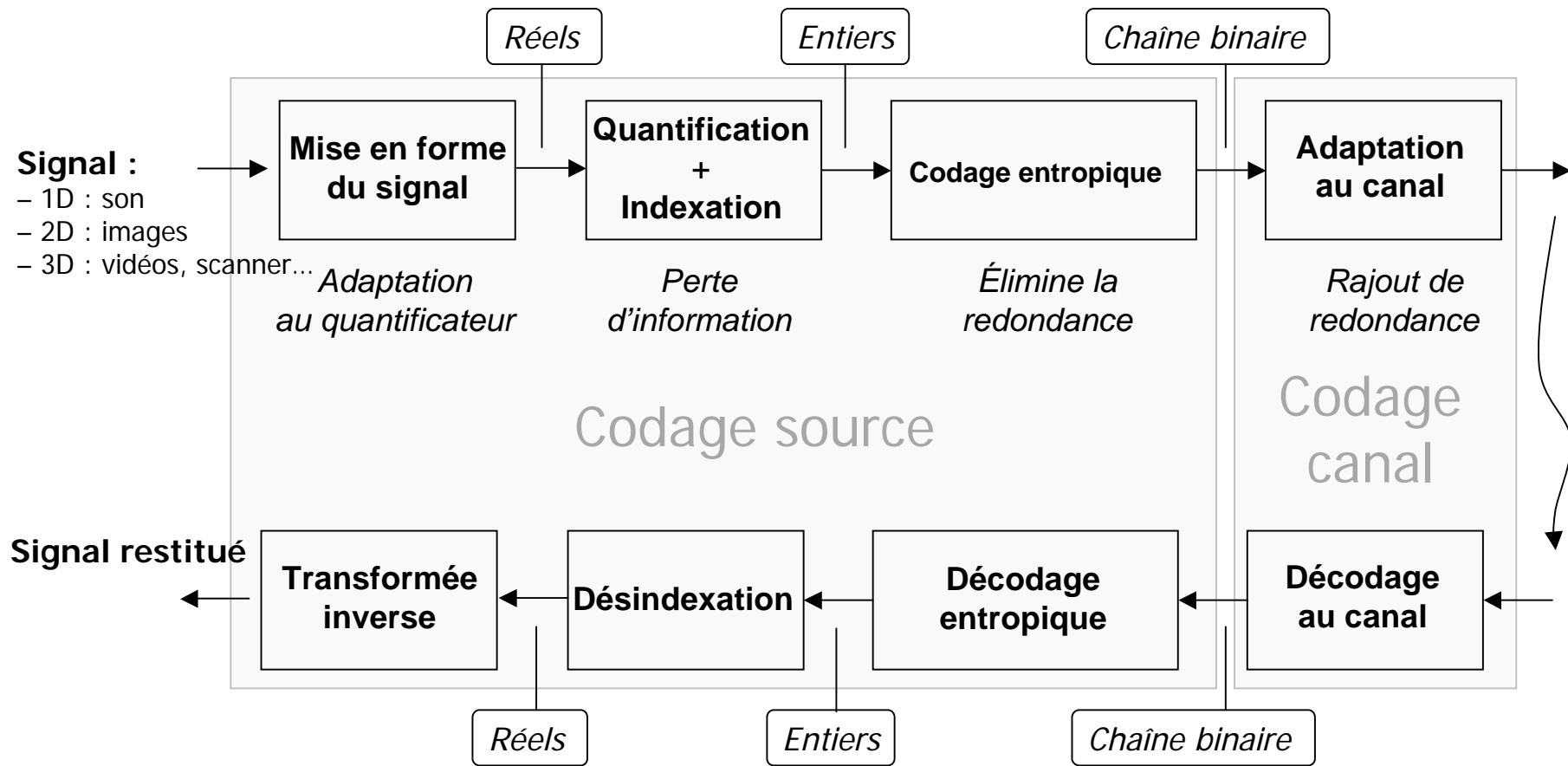
- **le taux de compression :**
(débit initial / débit après compression)
- **la qualité du signal comprimé :**
 - > critère subjectif (visuel)
 - > critère objectif (SNR...)
- **la complexité du système**
(coût calcul, mémoire requise)



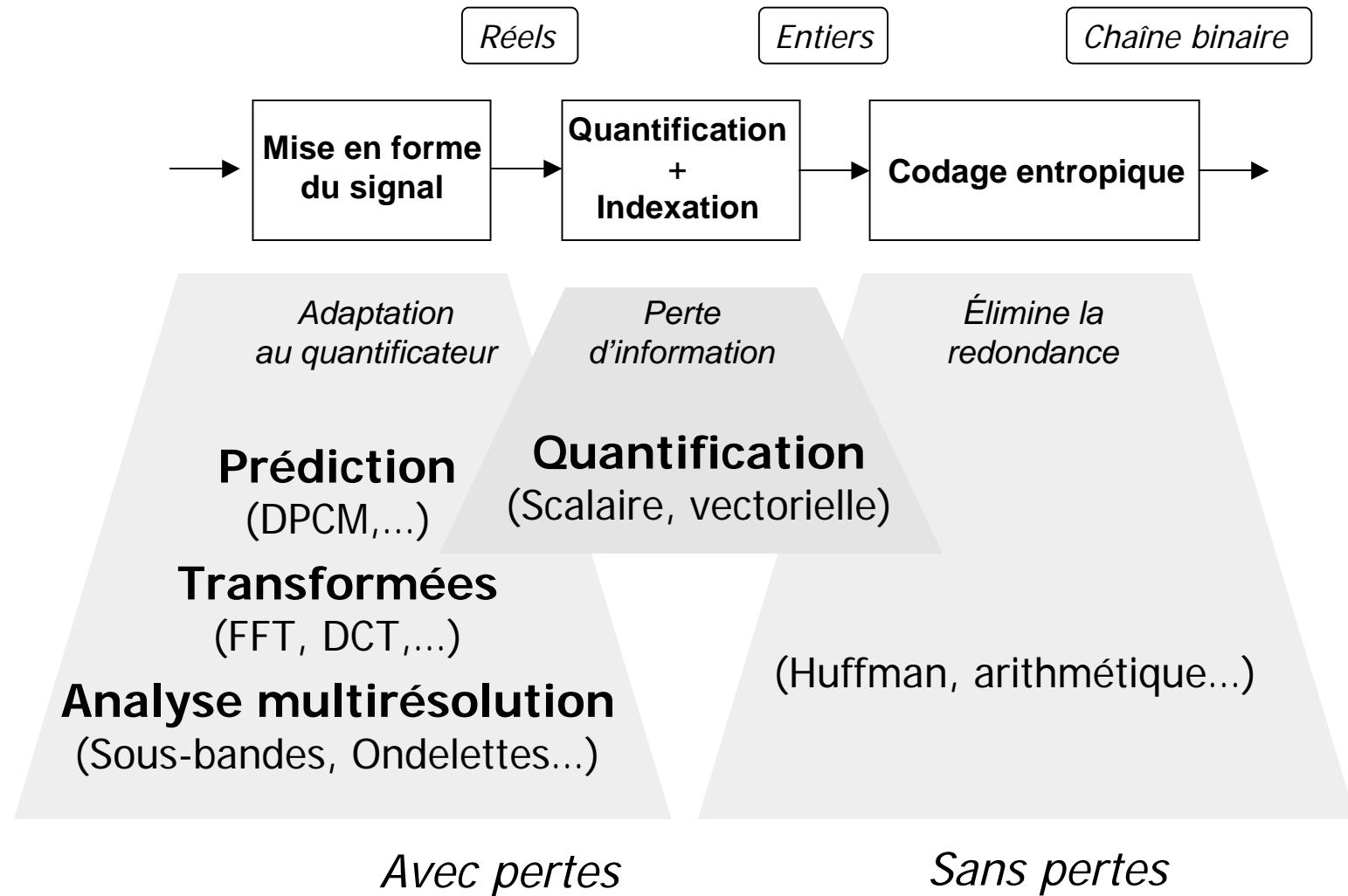
PROBLEME :

Optimiser ces 3 facteurs en même temps

La Chaîne De Compression



La Chaîne De Compression



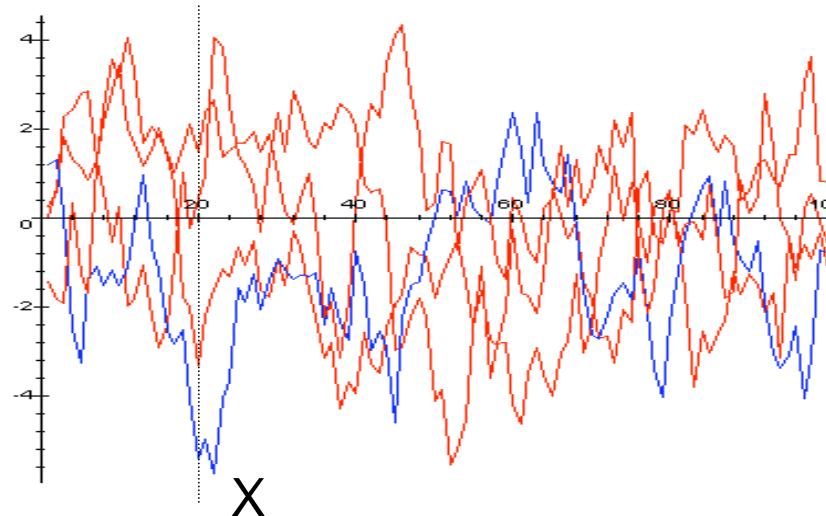
Une Image Numérique

Image = réalisation d'un **processus aléatoire bidimensionnel** $f(m,n)$
 (m,n) sont les coordonnées spatiales d'un pixel

L'intensité d'un pixel = **variable aléatoire**

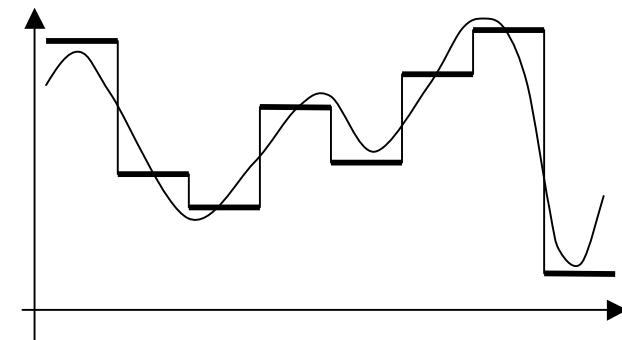
Hypothèses simplificatrices : stationnarité et ergodicité de $f(m,n)$

Exemple 1D :



Quantification : Principe

BUT : Représenter un signal numérisé sur L1 niveaux par L2 niveaux avec $L2 < L1$



Adaptation optimale
des niveaux au signal

Quantification : Principe

La quantification est l'opération fondamentale d'un système de compression. Son but est de sélectionner, pour une valeur d'entrée donnée, le plus proche voisin appartenant à un ensemble fini prédéterminé de valeurs numériques.

Plus précisément, on appelle quantificateur scalaire de taille L une application Q de \mathbf{R} dans un ensemble fini C , appelé aussi *dictionnaire* (*ou codebook*), contenant L symboles scalaires :

$$Q : \mathbf{R} \rightarrow C \quad \text{avec} \quad C = \{\hat{s}_1, \hat{s}_2, \dots, \hat{s}_L\}$$

On notera $\hat{x} = Q(x)$ la quantification de x

Elle correspond à chercher le plus proche voisin de x parmi tous les éléments du dictionnaire C .



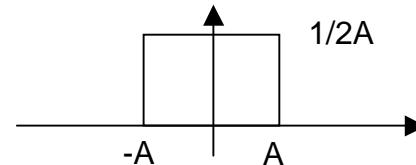
La Quantification Scalaire Uniforme

La quantification Scalaire Uniforme

Définition

Considérons un signal à temps discret $s(n)$ prenant ses valeurs dans l'intervalle $[-A, +A]$ avec une loi de distribution uniforme :

$$f_s(s) = \frac{1}{2A} \quad \forall s \in [-A, A]$$

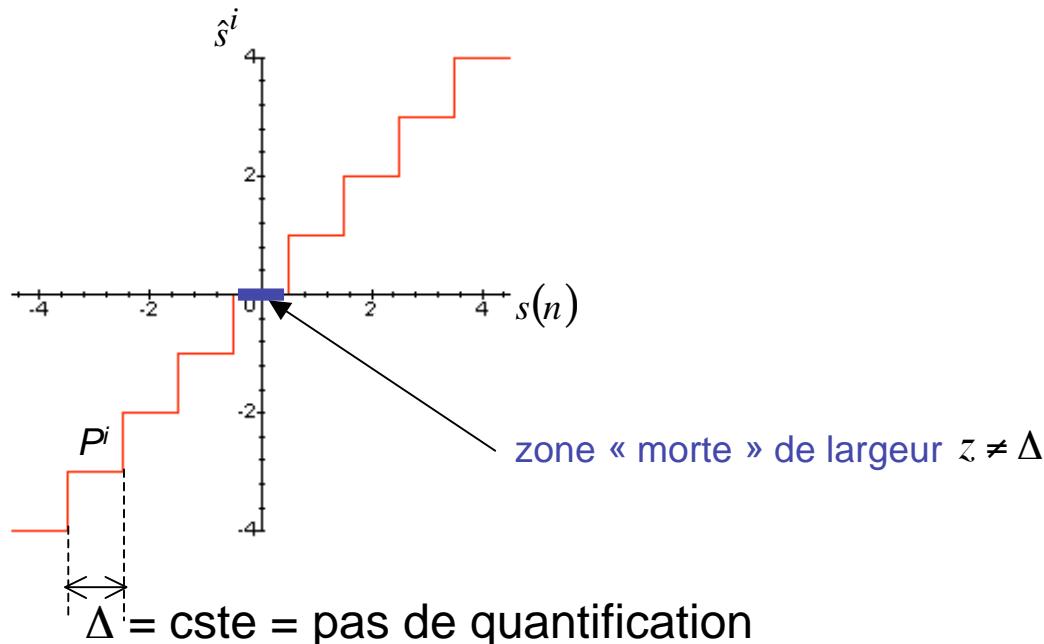


La démarche la plus naturelle pour définir un quantificateur consiste à:

- 1- Partitionner l'intervalle en $L = 2^R$ intervalles distincts $\{P^1, \dots, P^L\}$ de même longueur $\Delta = 2A/2^R$;
- 2- Numéroter chaque intervalle ;
- 3- Définir un représentant par intervalle i , par exemple le milieu de l'intervalle que l'on notera \hat{s}^i

La quantification Scalaire Uniforme

Illustration



P^i = classe. L'ensemble des classes forme la partition P de l'espace :

$$P = \bigcup_i P^i \quad \text{avec} \quad P^i \cap P^j = \emptyset \quad \forall i \neq j$$

La quantification Scalaire Uniforme

Les Procédures d'Encodage et de Décodage

- **La procédure d'encodage** consiste à décider à quel intervalle appartient $s(n)$ puis à lui associer le numéro $i(n) \in \{1 \dots L = 2^R\}$ correspondant.

C'est le numéro de l'intervalle choisi, qui sera transmis ou stocké.

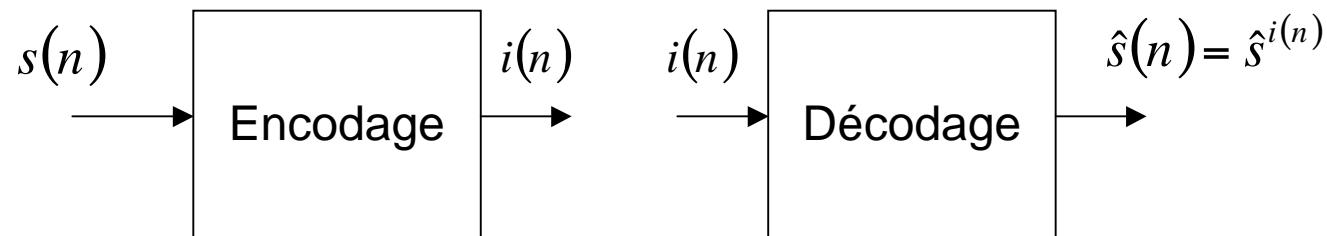
- **La procédure de décodage** consiste à associer au numéro $i(n)$ le représentant correspondant :

$$\hat{s}(n) = \hat{s}^{i(n)}$$

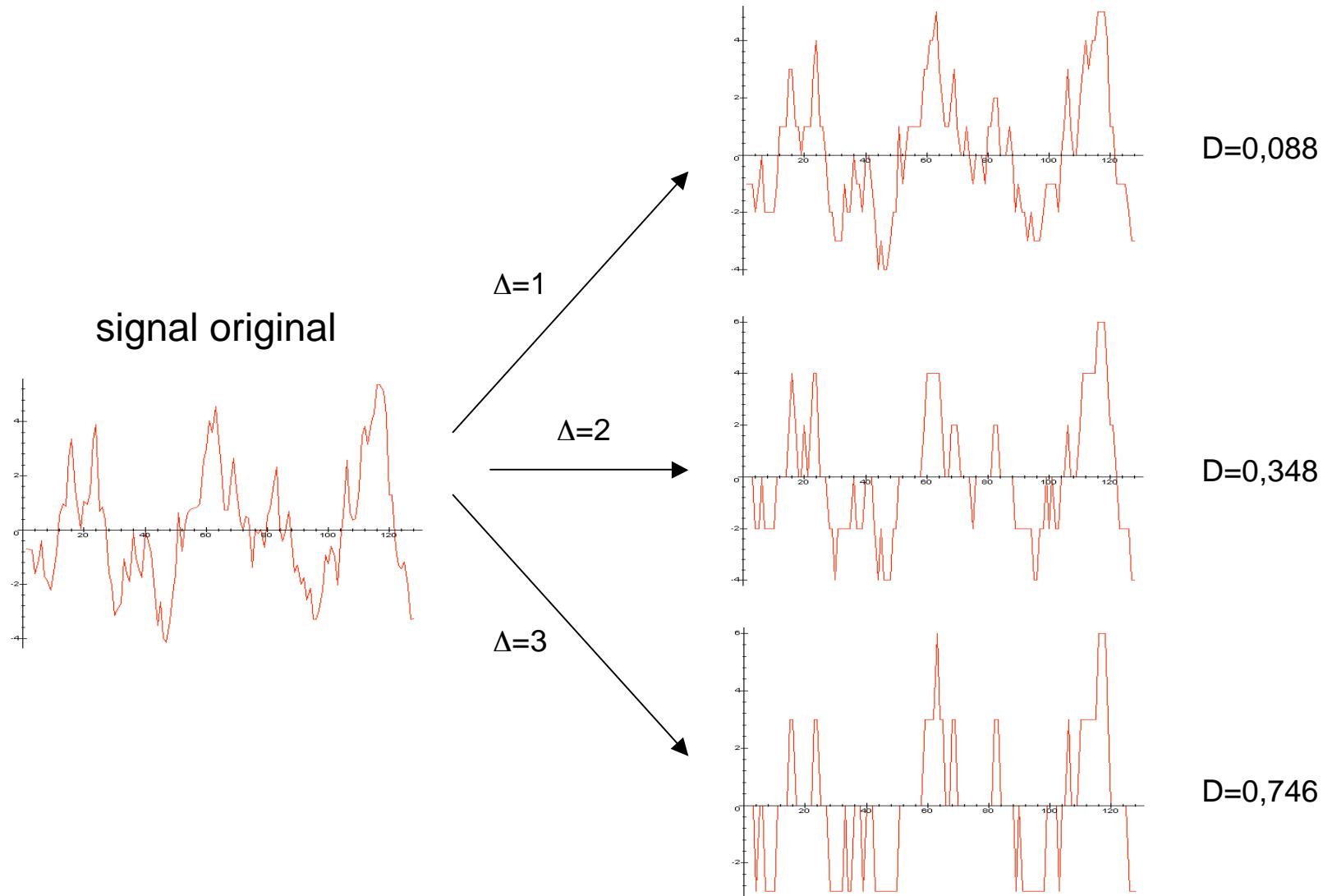
choisi parmi l'ensemble des représentants du dictionnaire $C = \{\hat{s}^1 \dots \hat{s}^L\}$

La quantification Scalaire Uniforme

Les Procédures d'Encodage et de Décodage



Exemple (signal 1D)

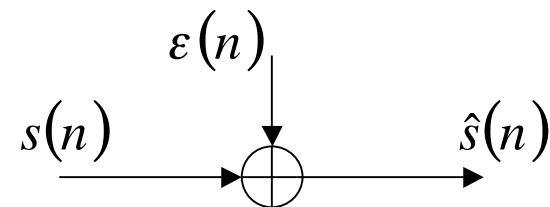


Distorsion De Quantification

L'erreur de quantification

L'opération de quantification introduit une ERREUR entre le signal source et le signal quantifié :

$$\begin{aligned}\varepsilon(n) &= s(n) - Q(s(n)) \\ &= s(n) - \hat{s}(n)\end{aligned}$$



c'est un bruit additif

La DISTORSION entre la sortie et l'entrée du quantificateur est généralement évaluée par l'ERREUR QUADRATIQUE MOYENNE (EQM):

$$d = \frac{1}{N} \sum_{n=1}^N d(s(n), \hat{s}(n)) \quad \text{avec} \quad d(s(n), \hat{s}(n)) = |s(n) - \hat{s}(n)|^2 = \varepsilon(n)^2$$

Distorsion De Quantification

Modèle mathématique du bruit de quantification

Pour caractériser la dégradation apportée par l'opération de quantification, il faut définir un critère et proposer un **MODELE** simple pour les signaux intervenant dans ce critère.

Supposons que $s(n)$ soit la réalisation d'un processus aléatoire $S(n)$. **L'ESPERANCE MATHEMATIQUE** de la distorsion est une mesure de performance qui apporte de l'information sur la qualité du signal quantifié. Elle correspond à une moyenne statistique et s'exprime par la relation :

$$D = E[d(S, Q(S))] = \int_{-\infty}^{+\infty} d(s, Q(s)) f_S(s) ds$$

densité de probabilité de la variable aléatoire d'entrée S

Distorsion De Quantification

L'erreur quadratique moyenne (EQM)

Pour un dictionnaire contenant L symboles et pour une mesure de distorsion quadratique la formule précédente traduit l'EQM et peut s'écrire de la façon suivante :

$$\begin{aligned} D = \mathbb{E}\left[\left(S - Q(S)\right)^2\right] &= \sum_{i=1}^L \int_{P^i} (s - \hat{s}_i)^2 f_S(s) ds \\ &= \sum_{i=1}^L \int_{t_{i-1}}^{t_i} (s - \hat{s}_i)^2 f_S(s) ds \end{aligned}$$

L'erreur quadratique moyenne est une mesure de distorsion qui permet d'évaluer la puissance moyenne d'un signal d'erreur. Cependant, il est bien connu qu'elle ne correspond pas fidèlement à des considérations subjectives de qualité pour les images. Toutefois, cette mesure permet de mener à bien des développements analytiques et c'est pourquoi son utilisation est primordiale en traitement des images.

Distorsion De Quantification

Dans la pratique...

Dans la pratique, si le signal source $s(n)$ est un processus stationnaire et ergodique, alors le théorème de l'ergodicité implique que

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N d(s(n), Q(s(n))) = D$$

c'est-à-dire que la distorsion moyenne évaluée sur les données observées correspond à l'espérance mathématique de la distorsion D calculée par la formule précédente.

Distorsion De Quantification

Bruit granulaire et bruit de surcharge

Le bruit de quantification considéré comme bruit additif se compose de deux bruits distincts appelés **BRUIT GRANULAIRE** et **BRUIT de SURCHARGE**

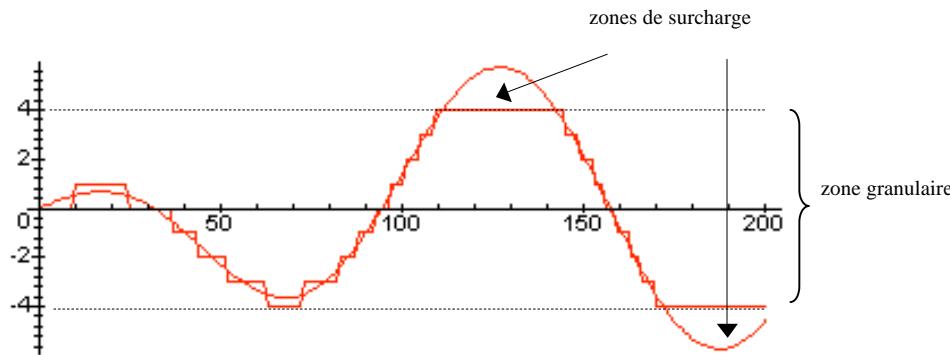


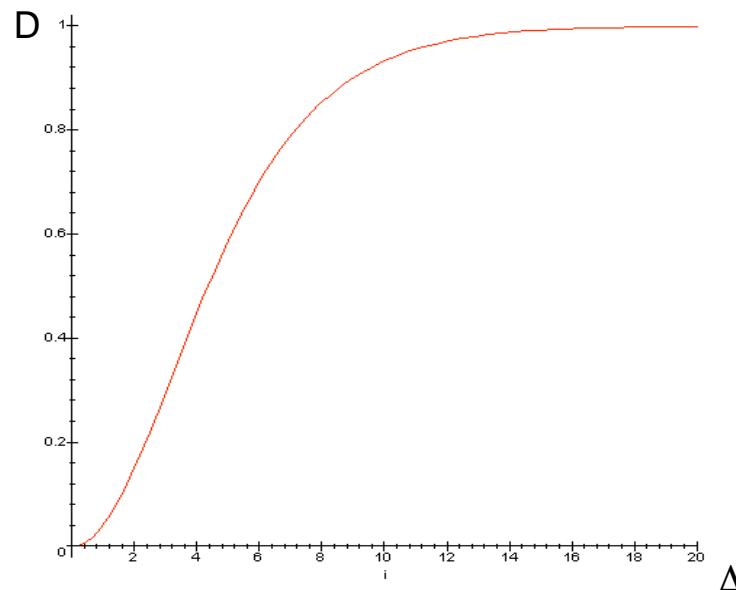
Illustration du bruit granulaire et du bruit de surcharge. Le signal est quantifié par un quantificateur scalaire uniforme à $L=9$ niveaux. Les amplitudes de signal supérieures à 4 ou inférieures à -4 sont quantifiées respectivement par 4 ou -4.

Distorsion De Quantification

Distorsion granulaire

La distorsion GRANULAIRE est définie par (cf. précédemment) :

$$D_{granulaire} = \sum_{i=1}^L \int_{t_{i-1}}^{t_i} (s - \hat{s}_i)^2 f_S(s) ds$$

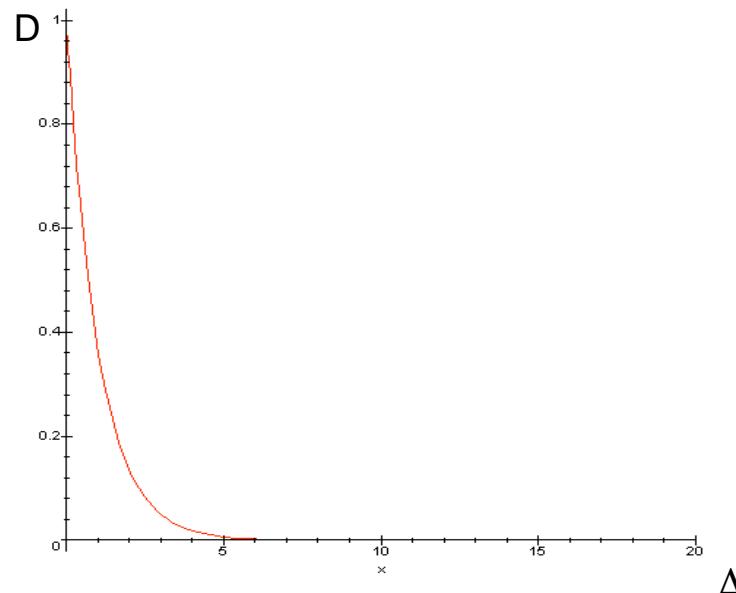


Distorsion De Quantification

Distorsion de surcharge

La distorsion de SURCHARGE est définie par :

$$D_{\text{surcharge}} = \int_{-\infty}^{t_1} (s - \hat{s}_1)^2 f_S(s) ds + \int_{t_{L-1}}^{+\infty} (s - \hat{s}_L)^2 f_S(s) ds$$

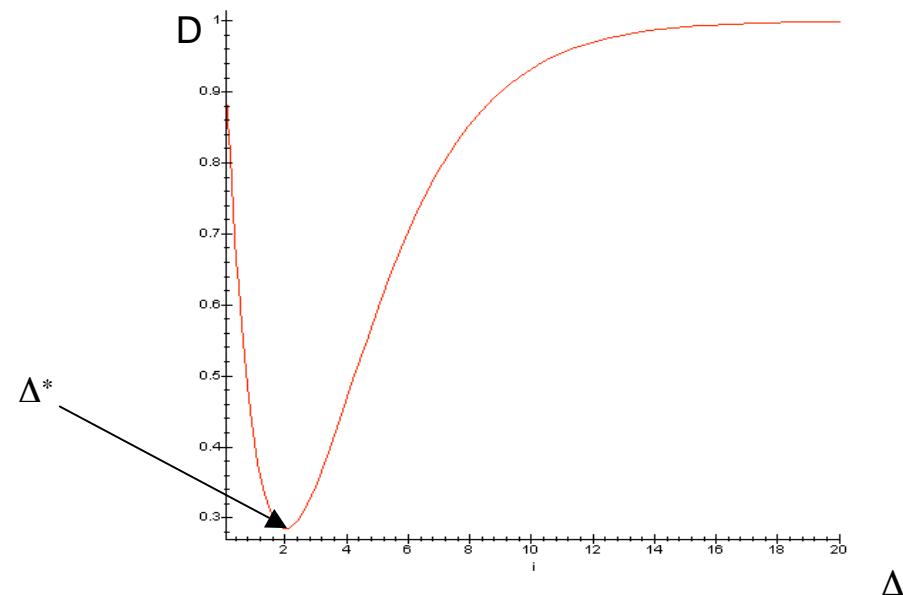


Distorsion De Quantification

Pas de quantification optimum

La distorsion totale est définie par :

$$D_{Totale} = D_{granulaire} + D_{surcharge}$$



Relation fondamentale (1/3)

Supposons que l'erreur de quantification $\varepsilon(n)$ est un processus aléatoire $E(n)$ avec les hypothèses suivantes :

- Le processus prend ses valeurs de façon équiprobables dans l'intervalle $[-\Delta/2, +\Delta/2]$
- $E(n)$ et $S(n)$ sont indépendants,
- $E(n), E(n-1), \dots$ sont indépendants entre eux,
- Le processus aléatoire $E(n)$ est une suite de variables aléatoires indépendantes et identiquement distribuées (une suite i.i.d.).

Relation fondamentale (2/3)

La moyenne de l'erreur de quantification est nulle, sa variance est donnée par :

$$\begin{aligned}\sigma_E^2 &= E[E^2(n)] = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} x^2 \frac{1}{\Delta} dx = \frac{\Delta^2}{12} = \frac{1}{12} \left(\frac{2A}{2^R} \right)^2 \\ &= \frac{A^2}{3} 2^{-2R}\end{aligned}$$

Or, si on suppose que $S(n)$ est uniformément réparti dans l'intervalle $[-A, A]$, hypothèse irréaliste pour un signal quelconque mais cela entraîne un calcul simple, sa moyenne est nulle et sa variance à pour expression :

$$\sigma_S^2 = E[S^2(n)] = \int_{-A}^A x^2 \frac{1}{2A} dx = \frac{A^2}{3}$$

Relation fondamentale (3/3)

On obtient donc la **relation fondamentale** qui donne la puissance de l'erreur de quantification en fonction de la puissance du signal et du débit R (nombre de bits disponibles pour représenter un échantillon) :

$$\sigma_E^2 = \sigma_S^2 2^{-2R}$$

Le rapport signal sur bruit a pour expression :

$$10 \log_{10} \left(\frac{\mathbb{E}[S^2(n)]}{\mathbb{E}[E^2(n)]} \right) = 10 \log_{10} (2^{2R}) = 6,02 R \quad dB$$

Le fait de rajouter 1 bit revient donc à augmenter le rapport signal sur bruit de 6 dB



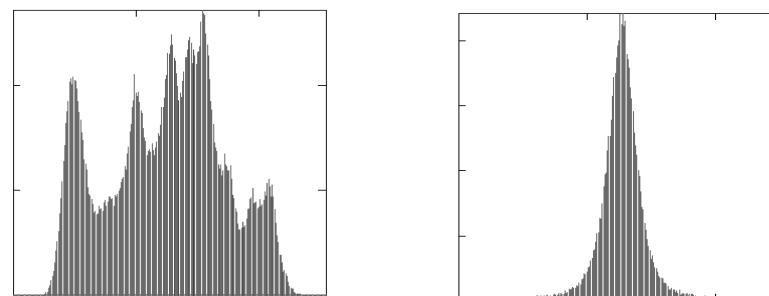
La Quantification Scalaire Non Uniforme

La Quantification Scalaire Non Uniforme

Hypothèses

Les hypothèses faites précédemment sont, bien entendu, très mal adaptées à un signal de parole, de musique ou d'image. Les échantillons ne peuvent pas être interprétés comme la réalisation d'un processus aléatoire stationnaire prenant ses valeurs de façon uniformément répartie dans l'intervalle $[-A, A]$.

- Les propriétés statistiques évoluent constamment : **uniquement dans le meilleur des cas, on pourra considérer un signal comme localement stationnaire.**
- De plus, **les lois de probabilité ne sont pas uniformes :**

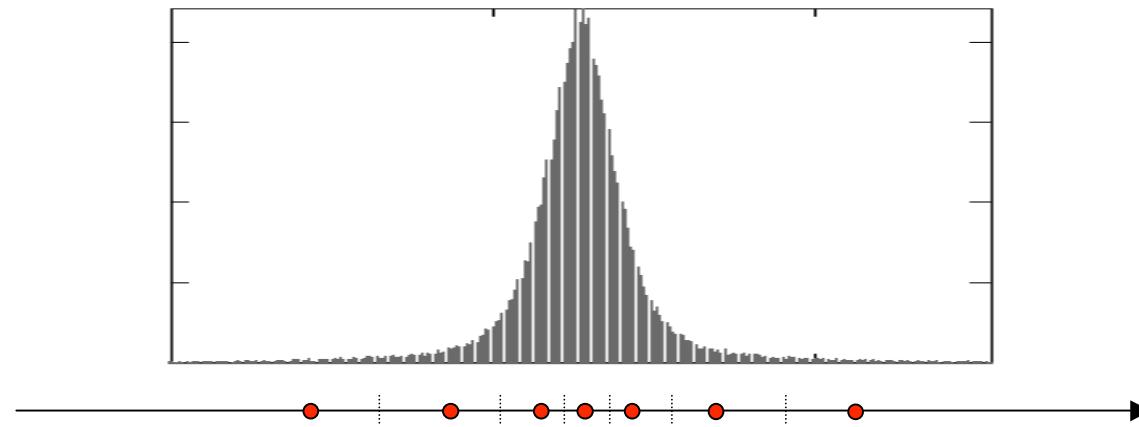


La Quantification Scalaire Non Uniforme

Solution intuitive

Il est possible de formuler proprement ce problème si l'on admet que l'on connaît la densité de probabilité ou si l'on peut l'estimer.

Intuitivement le quantificateur optimal doit avoir la forme suivante :



La partition de l'axe réel n'est plus composée d'éléments de longueur constante : la longueur est d'autant plus petite que la densité de probabilité correspondante est importante.

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (1/9)

En 1957, Lloyd a déterminé les deux conditions nécessaires d'optimalité pour construire un quantificateur adapté à des sources quelconques. Ces conditions ont été parallèlement découvertes par Max et publiées en 1960.

Pour définir le quantificateur, il s'agit de trouver la partition $\{P^1, \dots, P^L\}$ et les représentants $\{\hat{s}^1, \dots, \hat{s}^L\}$ minimisant la distorsion D définie par :

$$D = E[(S - Q(S))^2] = \sum_{i=1}^L \int_{P^i} (s - \hat{s}_i)^2 f_S(s) ds$$

Cette optimisation conjointe n'admet pas de solution simple.

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (2/9)

Il n'existe que 2 conditions d'optimalité

1. Si l'on connaît les représentants $\{\hat{s}^1 \dots \hat{s}^L\}$, on peut calculer la meilleure partition $\{P^1, \dots, P^L\}$.
2. Si l'on se donne la partition, on peut en déduire les meilleurs représentants.

Ces deux conditions d'optimalité fournissent les bases pour les algorithmes d'optimisation généralement utilisés en quantification.

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (3/9)

Condition 1 :

Etant donné un dictionnaire $\{\hat{s}^1 \dots \hat{s}^L\}$, la meilleure partition est celle qui vérifie :

$$P^i = \left\{ s \mid (s - \hat{s}^i)^2 \leq (s - \hat{s}^j)^2 \quad \forall j \in \{1 \dots L\} \right\}$$

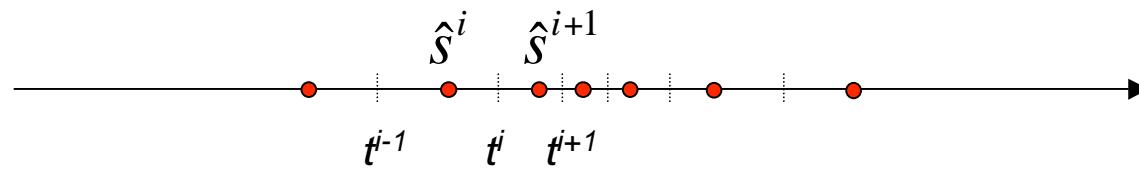
C'est la règle dite du plus proche voisin (ppv).

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (4/9)

Démonstration condition 1 :

Si l'on appelle t^i la valeur définissant la frontière entre les partitions P^i et P^{i+1} :



la minimisation de l'EQM D relativement à t^i est obtenue en écrivant :

$$\frac{\partial}{\partial t^i} = \left[\int_{t^{i-1}}^{t^i} (x - \hat{s}^i)^2 f_s(x) dx + \int_{t^i}^{t^{i+1}} (x - \hat{s}^{i+1})^2 f_s(x) dx \right] = 0$$

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (5/9)

Démonstration condition 1 :

soit :

$$(t^i - \hat{s}^i)^2 f_S(t^i) - (t^i - \hat{s}^{i+1})^2 f_S(t^i) = 0$$

donc,

$$t^i = \frac{\hat{s}^i + \hat{s}^{i+1}}{2}$$

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (6/9)

Condition 2 :

Etant donné un une partition $\{P^1, \dots, P^L\}$, le meilleurs dictionnaire est celui qui vérifie :

$$\hat{s}^i = \frac{\int x f_s(x) dx}{\int_{x \in P^i} f_s(x) dx} \quad \forall i = \{1, \dots, L\}$$

C'est la [condition du centroïde](#).

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (7/9)

Démonstration condition 2 :

Etant donné une partition $\{P^1, \dots, P^L\}$, la minimisation de l'EQM D relativement à \hat{s}^i ne fait intervenir qu'un élément de la partition (ou de la somme dans la formule de distorsion). Ainsi,

$$\frac{\partial}{\partial \hat{s}^i} \int_{x \in P^i} (x - \hat{s}^i)^2 f_S(x) dx = 0$$
$$-2 \int_{x \in P^i} x f_S(x) dx + 2\hat{s}^i \int_{x \in P^i} f_S(x) dx = 0$$

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (8/9)

Démonstration condition 2 :

Les meilleurs représentants sont alors obtenus par la [condition dite du centroïde](#) (ou centre de gravité) de la partie de la densité de probabilité placée dans la région P^i :

$$\hat{S}^i = \frac{\int_{x \in P^i} x f_S(x) dx}{\int_{x \in P^i} f_S(x) dx} = E[S | S \in P^i]$$

La valeur que l'on doit choisir est la valeur moyenne de S dans l'intervalle considéré.

La Quantification Scalaire Non Uniforme

Conditions nécessaires d'optimalité (9/9)

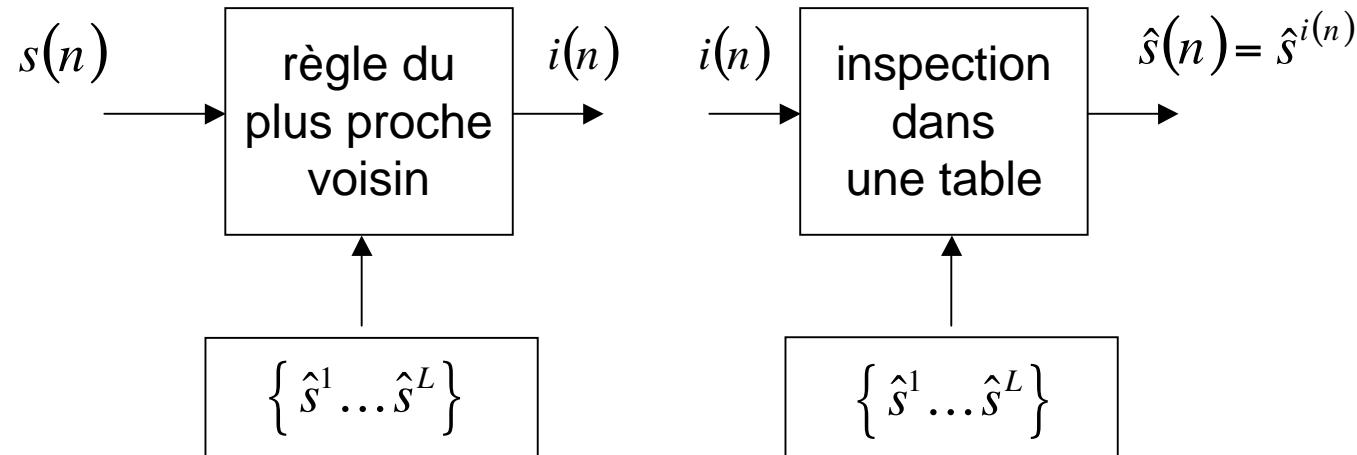
Remarque :

La connaissance explicite de la partition n'est pas nécessaire.

Cette partition est entièrement déterminée par la connaissance de la mesure de distorsion, par l'application de la règle du plus proche voisin et par l'ensemble des représentants.

La Quantification Scalaire Non Uniforme

Schéma d'encodage/décodage



La Quantification Scalaire Non Uniforme

Itération de Lloyd

L'idée fondamentale de cet algorithme est de construire itérativement un dictionnaire de quantification de façon à minimiser la distorsion D .

Pour cela, l'algorithme utilise un dictionnaire initial et modifie itérativement sa structure en fonction du partitionnement obtenu à chaque itération. Cette modification du dictionnaire constitue une opération basique connue sous le nom d'itération de Lloyd.

Itération de Lloyd

1. Pour un dictionnaire C_m donné, $\{\hat{s}^1 \dots \hat{s}^L\}$, trouver la partition optimale $P = \{P^1, \dots, P^L\}$, en utilisant la règle du plus proche voisin ;
2. A partir de la partition trouvée en (1) et de la condition du centroïde, trouver le dictionnaire optimal C_{m+1} .

La Quantification Scalaire Non Uniforme

Itération de Lloyd

- L'itération de Lloyd vérifie les deux conditions d'optimalité.
- Elle suppose que la densité de probabilité de la source est connue de façon à pouvoir calculer les représentants optimaux.
- Toutefois, si cette densité de probabilité est inconnue ou difficilement modélisable, il est possible d'appliquer l'itération de Lloyd sur des données empiriques (c'est-à-dire un ensemble d'échantillons représentatifs de la source) appelé **BASE d'APPRENTISSAGE** pour pouvoir générer le dictionnaire.

La Quantification Scalaire Non Uniforme

Algorithme de Lloyd-Max

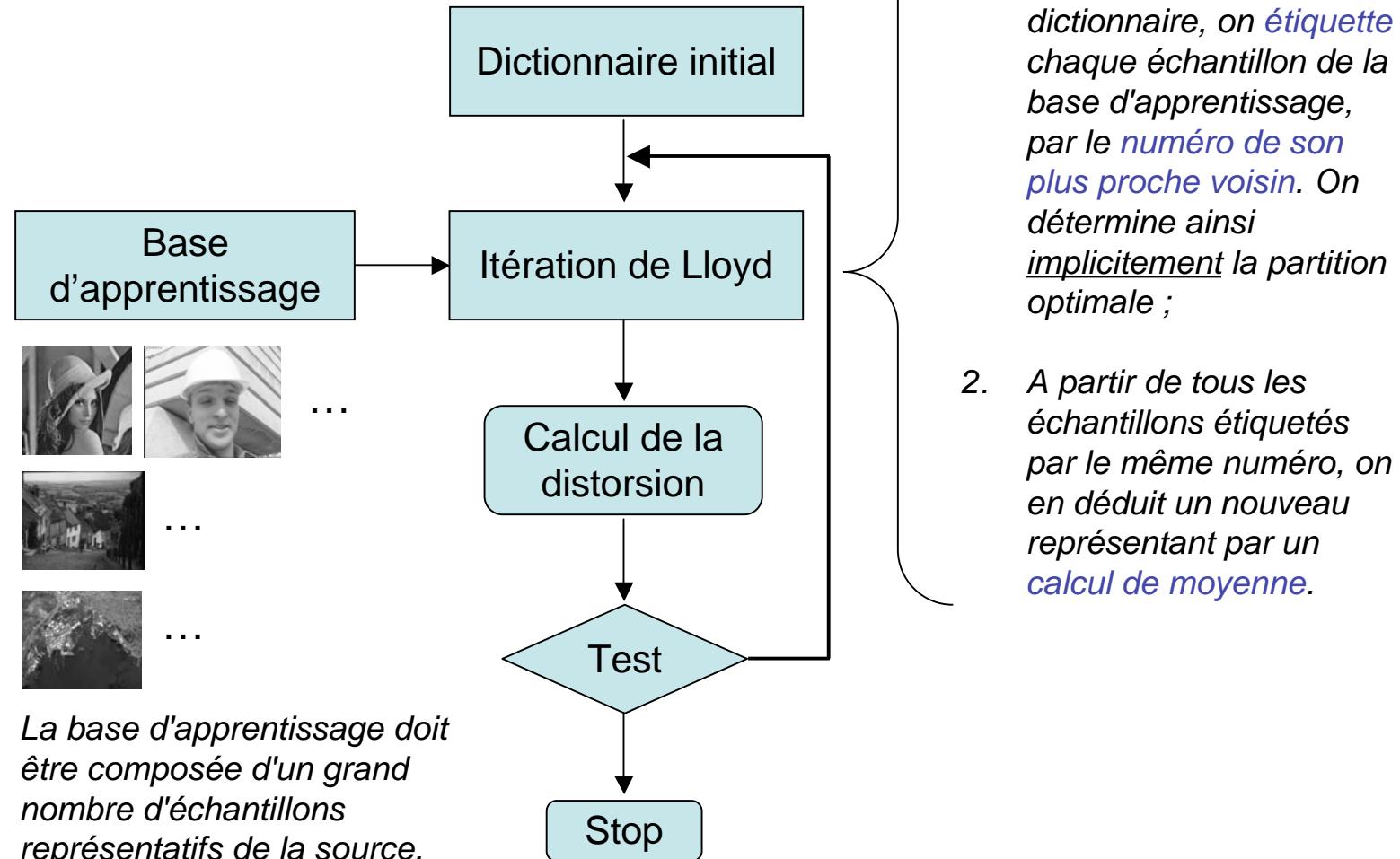
L'algorithme de Lloyd-Max est basé sur l'itération de Lloyd. Il est donné par la suite d'opérations suivantes :

Algorithme de Lloyd-Max

1. Choisir un dictionnaire initial C_1 (tirage aléatoire ou autre). $m=1$;
2. Pour un dictionnaire C_m donné, appliquer l'itération de Lloyd pour obtenir le dictionnaire C_{m+1} ;
3. Calculer la distorsion moyenne D obtenue avec le dictionnaire C_{m+1} . Si cette distorsion n'a pratiquement pas évolué par rapport à l'itération précédente, alors arrêter l'algorithme, sinon $m \leftarrow m + 1$ et retourner en 2.

La Quantification Scalaire Non Uniforme

Algorithme de Lloyd-Max : ORGANIGRAMME



La Quantification Scalaire Non Uniforme

Algorithme de Lloyd-Max : REMARQUES

1. Il est montré que cet algorithme assure la décroissance de la distorsion moyenne, mais ne tend pas toujours vers le minimum global de distorsion. On atteint simplement un **minimum local** ;
2. Le test d'arrêt peut être pris égal à $(D_m - D_{m+1})/D_m < \varepsilon$. Il suffit de choisir une valeur correcte pour ε de façon à assurer la convergence de l'algorithme vers la valeur optimale.
3. Le **choix du dictionnaire initial est important** de façon à converger vers un minimum local proche du minimum global.

Distorsion Asymptotique

Hypothèse haute résolution

Lorsque le nombre L de niveaux de quantification est élevé, il est possible d'obtenir explicitement l'expression de la partition optimale et de la puissance de l'erreur de quantification uniquement en fonction de la densité de probabilité $f_S(s)$.

Cette hypothèse dites haute résolution signifie que la densité de probabilité peut être supposée constante dans l'intervalle $[t^{i-1}, t^i]$ et que le représentant peut être pris au milieu de l'intervalle.

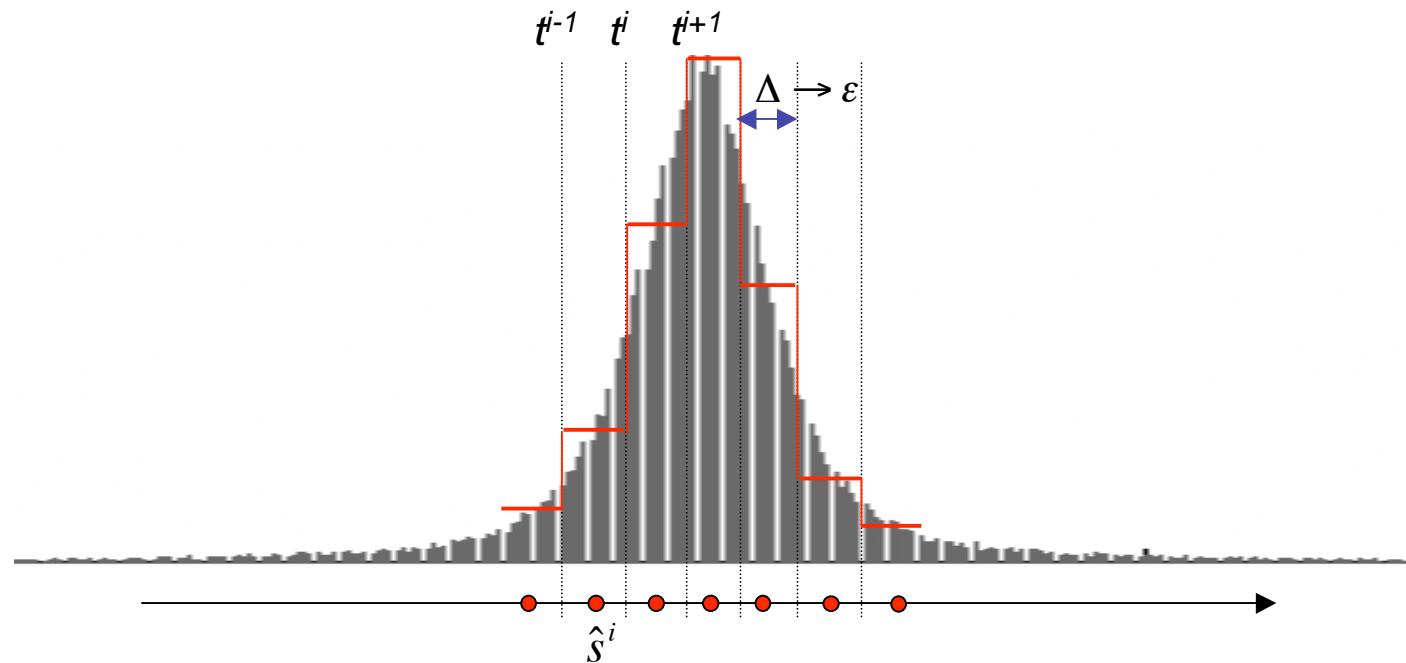
On peut donc écrire :

$$f_S(s) \approx p_S(\hat{s}^i) \quad \text{pour } x \in [t^{i-1}, t^i]$$

et $\hat{s}^i \approx \frac{t^{i-1} + t^i}{2}$

Distorsion Asymptotique

Hypothèse haute résolution : illustration



Distorsion Asymptotique

Puissance du bruit de quantification (1/6)

Soit $\Delta(i) = t^i - t^{i-1}$ la longueur de l'intervalle $[t^{i-1}, t^i]$ et :

$$\Pr\{i\} = \Pr\{S \in [t^{i-1}, t^i]\} = f_S(s)ds = p_S(\hat{s}^i)\Delta(i)$$

la probabilité que S appartienne à l'intervalle $[t^{i-1}, t^i]$.

La puissance de l'erreur de quantification (distorsion) s'écrit alors :

$$\begin{aligned} D &= \sum_{i=1}^L \int_{t^{i-1}}^{t^i} (s - \hat{s}_i)^2 f_S(s) ds \\ &= \sum_{i=1}^L p_S(\hat{s}^i) \int_{t^{i-1}}^{t^i} (x - \hat{s}^i)^2 dx = \sigma_E^2 \end{aligned}$$

Distorsion Asymptotique

Puissance du bruit de quantification (2/6)

Comme

$$\int_{t^{i-1}}^{t^i} (x - \hat{s}^i)^2 dx = \int_{-\Delta(i)}^{\Delta(i)} \varepsilon^2 d\varepsilon = \frac{\Delta^3(i)}{12}$$

On obtient :

$$D = \frac{1}{12} \sum_{i=1}^L p_s(\hat{s}^i) \Delta^3(i)$$

Cette relation s'écrit aussi :

$$D = \sum_{i=1}^L \Pr\{i\} \frac{\Delta^2(i)}{12} = E\left[\frac{\Delta^2}{12}\right]$$

Distorsion Asymptotique

Puissance du bruit de quantification (3/6)

La variance de l'erreur de quantification ne dépend que de la longueur des intervalles $\Delta(i)$.

On cherche donc $\{\Delta(1), \dots, \Delta(L)\}$ minimisant D . Posons

$$\alpha^3(i) = p_s(\hat{s}^i) \Delta^3(i) \Rightarrow D = \frac{1}{12} \sum_{i=1}^L \alpha^3(i)$$

Or, sous l'hypothèse asymptotique il est possible d'écrire :

$$\sum_{i=1}^L \alpha(i) = \sum_{i=1}^L p_s(\hat{s}^i)^{1/3} \Delta(i) \approx \underbrace{\int_{-\infty}^{+\infty} f_s(x)^{1/3} dx}_{\text{car } f \text{ est une densité de probabilité}} = cste$$

Distorsion Asymptotique

Puissance du bruit de quantification (4/6)

Le problème revient donc à minimiser la somme des cubes de L nombres positifs

$$\sum_{i=1}^L \alpha^3(i)$$

ayant une somme constante :

$$\sum_{i=1}^L \alpha(i) = cste$$

La solution consiste donc à les prendre tous égaux :

$$\alpha(1) = \dots = \alpha(L)$$

Distorsion Asymptotique

Puissance du bruit de quantification (5/6)

Ce qui implique :

$$\alpha^3(1) = \dots = \alpha^3(L)$$

$$p_s(\hat{s}^1)\Delta^3(1) = \dots = p_s(\hat{s}^L)\Delta^3(L)$$

Cette relation signifie qu'un intervalle sera d'autant plus petit que la probabilité que $S(n)$ appartienne à cet intervalle sera élevée et que tous les intervalles auront la même contribution dans la puissance de l'erreur de quantification (distorsion D).

Distorsion Asymptotique

Puissance du bruit de quantification (6/6)

La distorsion de quantification a pour expression :

$$D = \frac{L}{12} \alpha^3$$

avec

$$\alpha = \frac{1}{L} \int_{-\infty}^{+\infty} f_S(x)^{1/3} dx$$

on obtient donc la formule de BENNETT:

$$D(R) = \frac{1}{12L^2} \left[\int_{-\infty}^{+\infty} f_S(x)^{1/3} dx \right]^3 = \frac{1}{12} \left[\int_{-\infty}^{+\infty} f_S(x)^{1/3} dx \right]^3 2^{-2R}$$

Distorsion Asymptotique

Exemple

Pour une source Gaussienne, centrée, de variance σ_s^2 pour laquelle

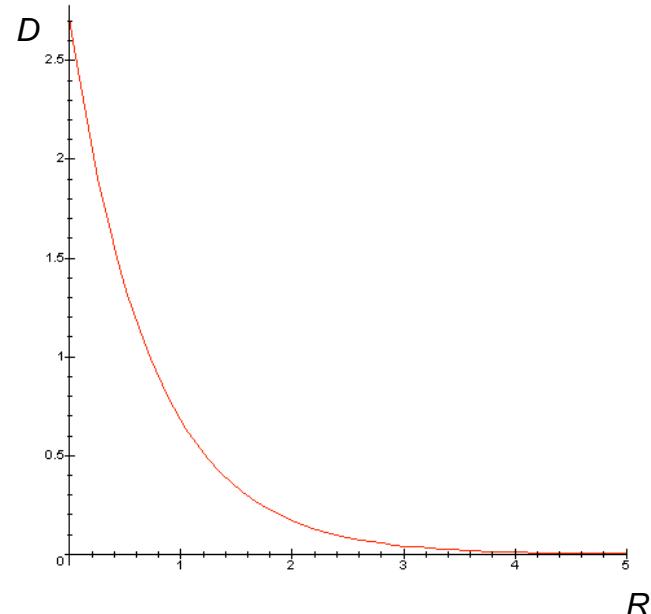
$$f_s(x) = \frac{1}{\sqrt{2\pi\sigma_s^2}} e^{-x^2/2\sigma_s^2}$$

on montre que :

$$\sigma_Q^2 = D(R) = c(1)\sigma_s^2 2^{-2R}$$

avec

$$c(1) = \frac{\sqrt{3}}{2}\pi$$



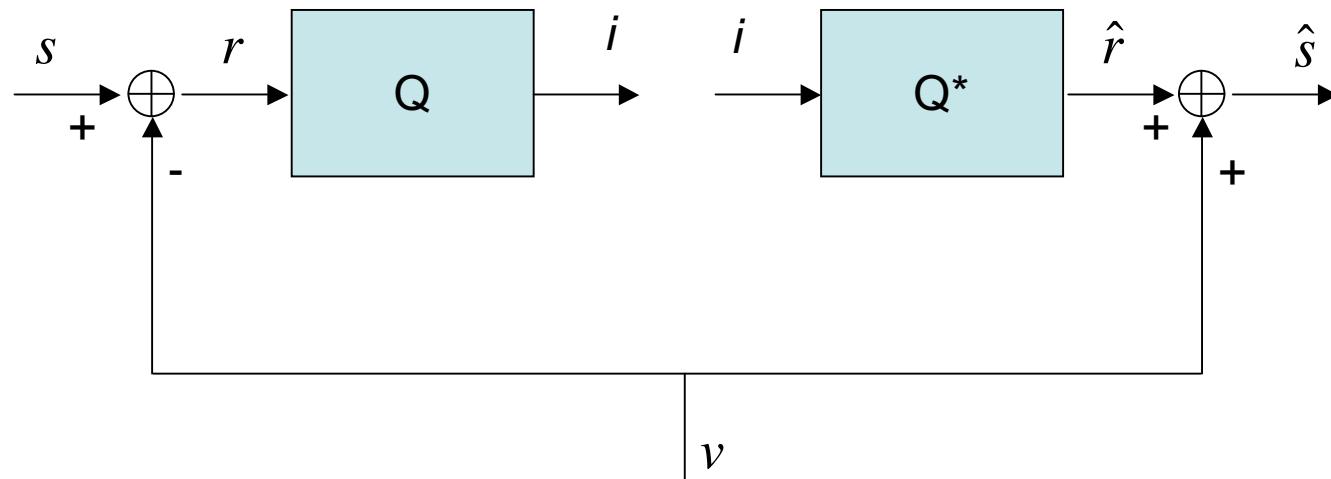


La Quantification Préditive

La Quantification Scalaire Prédictive

Principe

Décorréliser le signal avant de le quantifier :



La Quantification Scalaire Prédictive

Erreur de quantification/erreur de reconstruction

Remarquons d'abord que si l'on retranche à $s(n)$ une valeur quelconque $v(n)$ et si l'on réalise ensuite une procédure d'encodage/décodage puis que l'on rajoute $v(n)$ à la valeur décodede la distorsion entre $s(n)$ et $\hat{s}(n)$ et la distorsion entre $r(n)$ et $\hat{r}(n)$ sont identiques puisque :

$$s(n) - \hat{s}(n) = [s(n) - v(n)] - [\hat{s}(n) - v(n)] = r(n) - \hat{r}(n)$$

On distinguera l'erreur de quantification :

$$\varepsilon(n) = r(n) - \hat{r}(n)$$

et l'erreur de reconstruction

$$\bar{\varepsilon}(n) = s(n) - \hat{s}(n)$$

La Quantification Scalaire Prédictive

Puissances des erreurs

Ici, l'erreur de quantification et l'erreur de reconstruction sont égales à chaque instant n .

On a donc en particulier, égalité des puissances :

$$\sigma_E^2 = \sigma_{\bar{E}}^2$$

La Quantification Scalaire Prédictive

Choix de $v(n)$

On a une grande liberté sur le choix de $v(n)$. On peut le prendre par exemple sous la forme :

$$v(n) = - \sum_{i=1}^m a_i s(n-i)$$

où les coefficients a_i sont des paramètres fixes mais inconnus.

On appelle alors $v(n)$ la prédiction linéaire de $s(n)$ à l'instant n en fonction des échantillons observés aux m instants précédents.

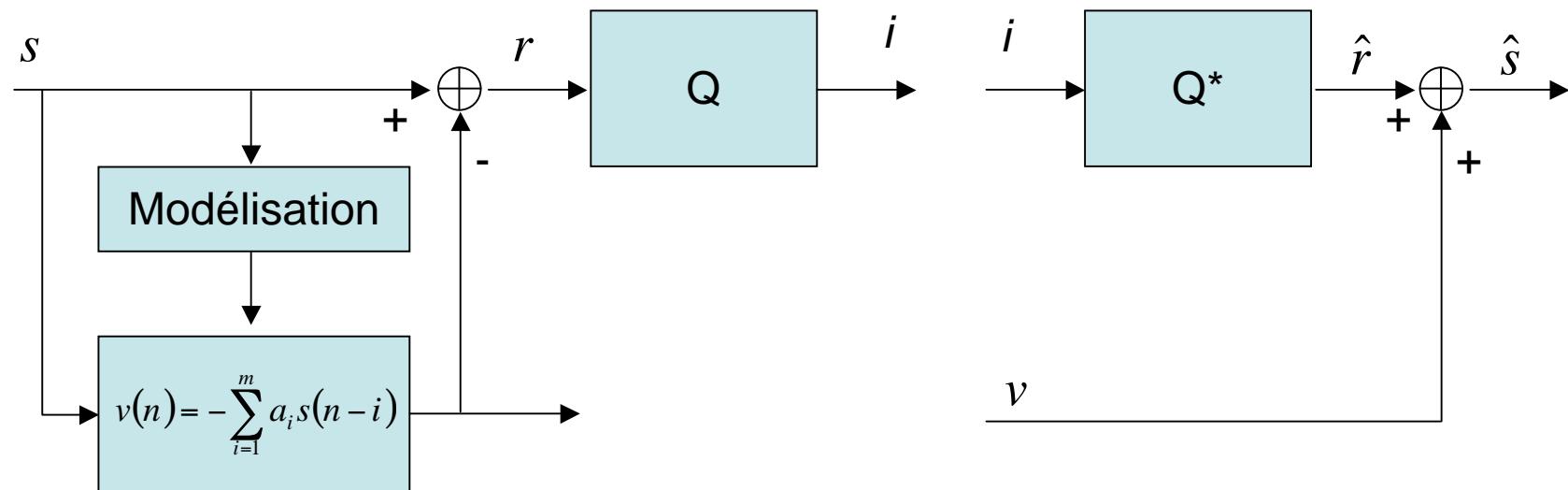
Le signal :

$$r(n) = s(n) - v(n) = s(n) + \sum_{i=1}^m a_i s(n-i)$$

est l'erreur de prédiction.

La Quantification Scalaire Prédictive

Schéma de principe



La Quantification Scalaire Prédictive

Bruit de quantification (1/3)

On cherche à minimiser la distorsion moyenne entre $s(n)$ et $\hat{s}(n)$. Les paramètres a_i ont une influence sur cette distorsion.

Cherchons donc à déterminer les paramètres a_i minimisant l'erreur quadratique moyenne ou maximisant le rapport signal sur bruit :

$$RSB = \frac{E[S^2(n)]}{E[(S(n) - \hat{s}(n))^2]} = \frac{E[S^2(n)]}{E[R^2(n)]} \times \frac{E[R^2(n)]}{E[(R(n) - \hat{r}(n))^2]}$$

en appelant $\hat{r}(n)$ le représentant sélectionné à l'instant n pour quantifier l'erreur de prédiction et R la variable aléatoire qui désigne l'erreur de prédiction.

La Quantification Scalaire Prédictive

Bruit de quantification (2/3)

Si l'on réalise une quantification optimale sur $r(n)$ avec un débit de b bits par échantillon et si l'on utilise le résultat suivant (établi précédemment) :

$$\sigma_E^2 = c(1)\sigma_R^2 2^{-2b} \quad \text{avec} \quad c(1) = \frac{\sqrt{3}}{2}\pi$$

alors,

$$RSB = \frac{\mathbb{E} [S^2(n)]}{\mathbb{E} [(S(n) - \hat{s}(n))^2]} = \frac{\sigma_S^2 2^{2b}}{\sigma_R^2 c(1)}$$

où σ_R^2 est la puissance de l'erreur de prédiction

Comme la puissance du signal est imposée, maximiser le RSB revient donc minimiser la puissance de l'erreur de prédiction σ_R^2

La Quantification Scalaire Prédictive

Bruit de quantification (3/3)

Définissons le gain de prédiction

$$G_p(m) = \frac{\sigma_s^2}{\sigma_R^2}$$

$G_p(m)$ est une fonction croissante de m et tend vers une limite lorsque $m \rightarrow \infty$

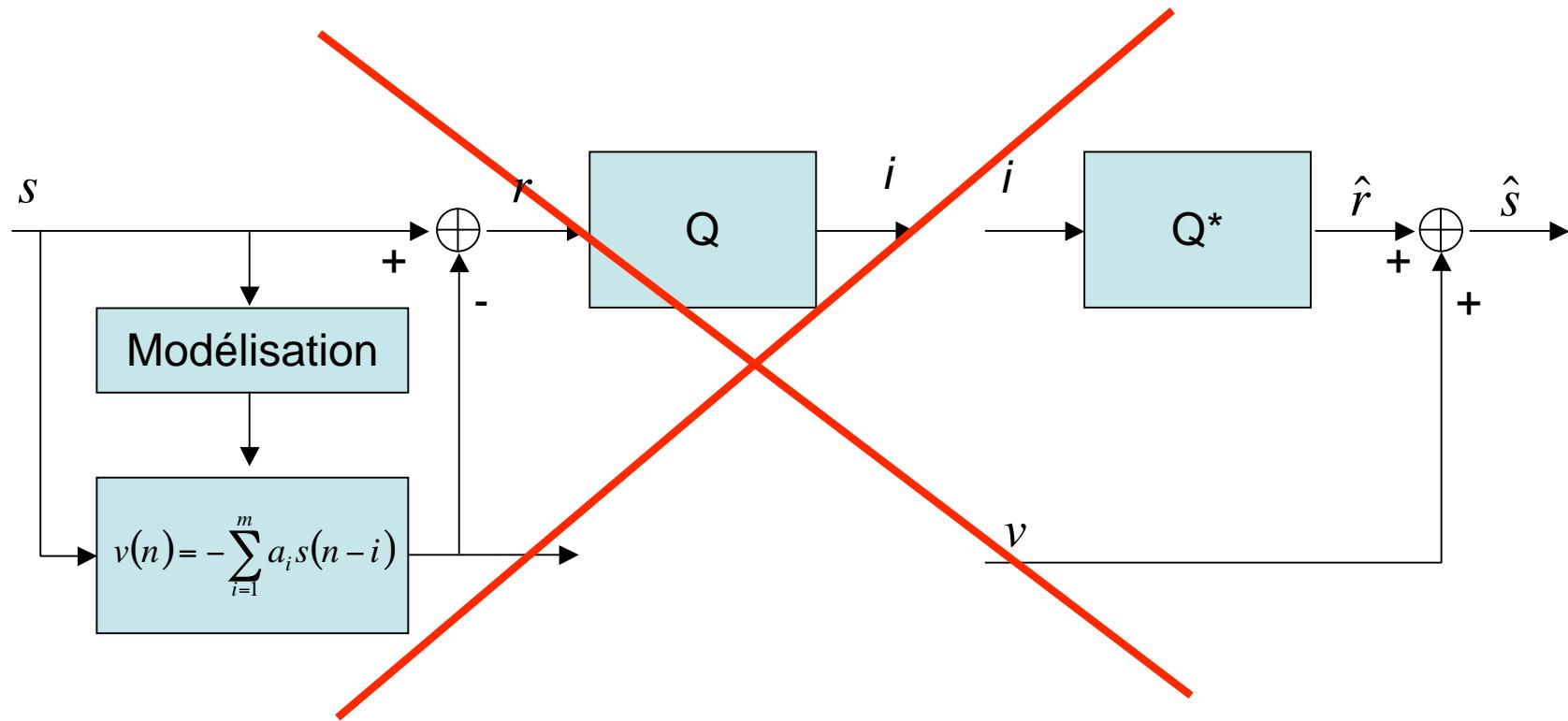
Le gain de prédiction mesure l'amélioration des performances apportée par la quantification de l'erreur de prédiction plutôt que la quantification directe du signal.

On peut alors écrire :

$$\sigma_E^2 = \sigma_{\bar{E}}^2 = c(1) \frac{\sigma_s^2}{G_p(m)} 2^{-2b}$$

La Quantification Scalaire Prédictive

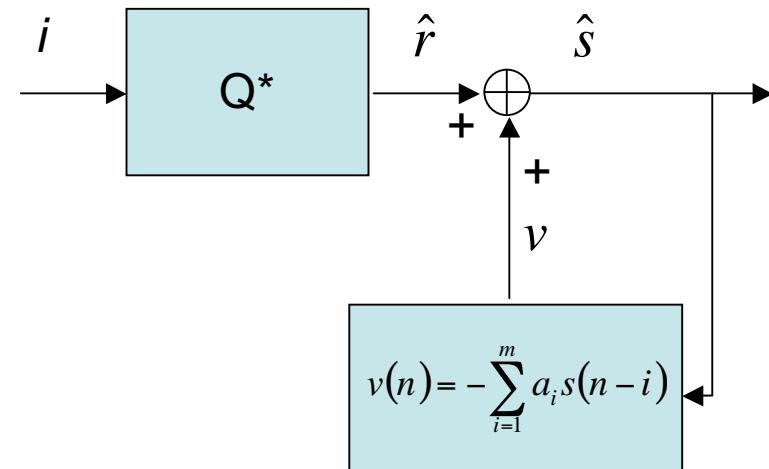
En BOUCLE OUVERTE => BOUCLE FERMEE



Problème : la quantité v doit être transmise !!

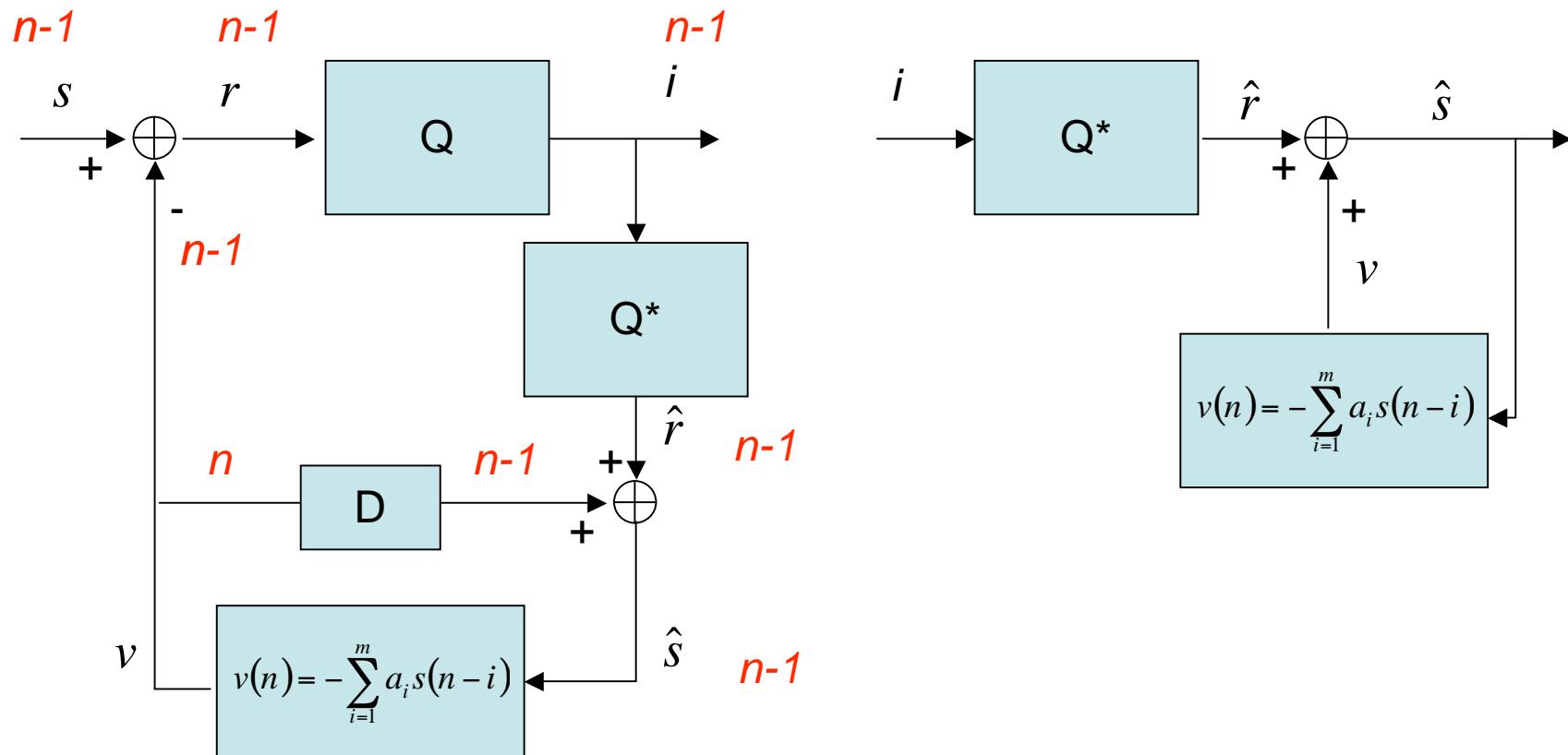
La Quantification Scalaire Prédictive

En BOUCLE FERMEE



La Quantification Scalaire Prédictive

En BOUCLE FERMEE





Le Codage Entropique

Le Codage Entropique

Définition d'une Source discrète sans mémoire

Soit un signal $x(n)$ à temps discret. On interprétera ce signal comme la réalisation d'un processus aléatoire $X(n)$ à temps discret. On suppose que ce processus aléatoire possède les bonnes propriétés habituelles de stationnarité et d'ergodicité.

On suppose ici que le processus $X(n)$ est une suite de variables aléatoires indépendantes et identiquement distribuées (suite *i.i.d.*) prenant ses valeurs dans $A_X = \{x^1 \dots x^{L_X}\}$. On parle alors de source discrète sans mémoire.

La distribution des différentes probabilités est notée :

$$p_X(i) = \Pr\{X(n) = x^i\}$$

Cette distribution est indépendante de l'instant d'observation n puisque le processus aléatoire est stationnaire.

Le Codage Entropique

Entropie d'une source

On appelle information propre de l'évènement $\{ X(n) = x^i \}$ la quantité :

$$I(x^i) = -\log_2 p_X(i)$$

Elle est positive ou nulle. Elle est exprimée en **bits/symbole** puisque l'on a pris un logarithme en base 2.

On appelle **information propre moyenne ou entropie** de la source l'espérance de la variable aléatoire $I(x^i)$:

$$H(X) = E[I(x^i)] = -\sum_{i=1}^{L_X} p_X(i) \log_2 p_X(i) \text{ bits/symbole}$$

Le Codage Entropique

Que représente l'entropie ?

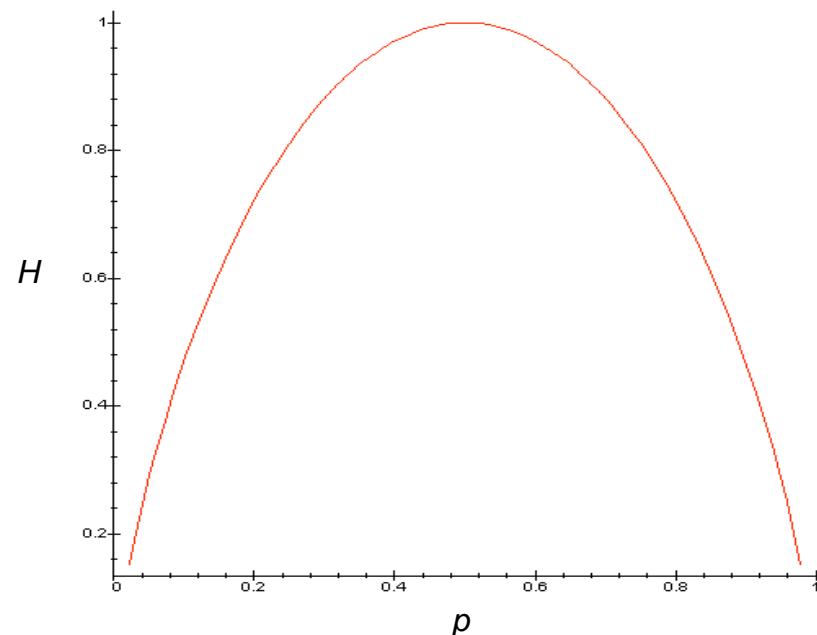
- L'entropie est la quantité d'information qu'apporte, en moyenne, une réalisation de $X(n)$. Elle donne le nombre de bits nécessaires en moyenne pour décrire complètement la source.
- Elle mesure l'incertitude associée à la source.
- L'entropie ne dépend que de la distribution des probabilités $p_X(i)$.

Le Codage Entropique

Exemple d'entropie

Prenons l'exemple d'une source binaire, $X(n) \in \{x^1, x^2\}$, sans mémoire et notons pour simplifier $p = p_X(1)$. L'entropie de cette source vaut :

$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p) \quad \text{bits/symbole}$$



- $p=0,5 \Rightarrow H(X)=H_{max}(X)=1$ bits/symbole
équiprobabilité
- $p=0$ ou $p=1 \Rightarrow H(X)=H_{min}(X)=0$ bits/symbole
source « certaine » ou totalement prédictible

➡ Fonction concave de p

Le Codage Entropique

Généralisation

Ces résultats se généralisent.

- On montre que l'entropie d'une source discrète est **positive ou nulle** : elle est nulle lorsque $X(n)$ est presque sûrement égale à une valeur possible (la source est alors totalement prédictible).
- Lorsque les valeurs possibles sont équiprobables on a :

$$p_X(i) = 1/L_X$$

alors $H(X) = -\sum_{i=1}^{L_X} \frac{1}{L_X} \log_2 \frac{1}{L_X} = \log_2 L_X$

=> l'entropie est **maximale** et vaut : $\log_2 L_X$

L'entropie est donc bornée par : $0 \leq H(X) \leq \log_2 L_X$

Codage d'Une Source

Définition

Ici, l'information à transmettre où à stocker, modélisée par le processus aléatoire $X(n)$, prend ses valeurs dans un ensemble fini $A_X = \{x^1 \dots x^{L_X}\}$, l'alphabet d'entrée. On désire représenter (coder) les différents éléments de cet ensemble :

- de **façon adaptée** aux caractéristiques du canal de transmission :

La représentation de chaque symbole d'entrée, un mot du code, peut être construit à partir d'éléments d'un autre alphabet adapté au canal. On supposera par la suite que cet alphabet est composé de 2 éléments, $A_C = \{0,1\}$

- et de **façon efficace** :

On cherche à représenter la source en utilisant le minimum de bits, c'est-à-dire en minimisant la longueur moyenne des mots du code. Précédemment, tous les mots du code avaient la même longueur R bits. Maintenant, on va très naturellement associer aux symboles d'entrée les plus probables, les mots de code les plus courts. **Le code est dit à longueur variable.**

Codage d'Une Source

Formalisation

Plus précisément, on appelle codage de la source $X(n)$ une application de l'alphabet A_x dans l'ensemble des suites finies d'éléments de l'alphabet A_C .

Le code $C = \{c^1, c^2, \dots, c^{L_X}\}$ est l'ensemble de ces suites.

Chaque suite possible $c^i = [01101\dots01]$ est un **mot du code**.

Le nombre d'éléments de A_C composant un mot c^i est la longueur $l(c^i)$ du mot.

La longueur moyenne des mots du code est donnée par

$$\bar{l} = \sum_{i=1}^{L_X} p_X(i) l(c^i)$$

Codage d'Une Source

Code instantané uniquement décodable

Remarquons tout d'abord qu'il semble exister un problème associé à un code de longueur variable :

Faut-il ajouter des séparateurs entre mots du code dans une séquence ?

Cela n'est **pas nécessaire** si le code vérifie la **condition dite du préfixe** :

AUCUN mot de code ne doit être préfixe d'un autre mot du code

Codage d'Une Source

Code instantané uniquement décodable : Exemple de code préfixe

Prenons l'exemple d'une source ne pouvant prendre que six valeurs différentes $\{x^1, \dots, x^6\}$. Le code défini par les associations données dans la table suivante :

Symboles	x^1	x^2	x^3	x^4	x^5	x^6
Code	0	100	101	110	1110	1111

est un code valide, on dit uniquement décodable, puisque, recevant par exemple la chaîne binaire :

11001110101110

on en déduit la séquence

$x^4 x^1 x^5 x^3 x^4$

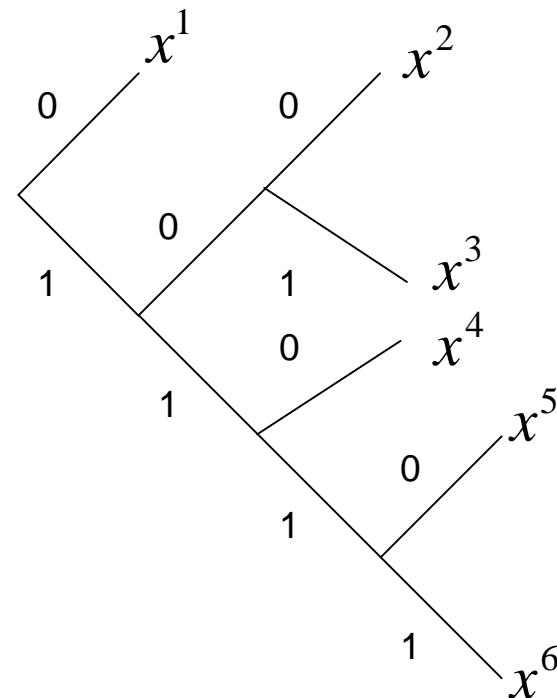
Codage d'Une Source

Code instantané uniquement décodable : Exemple de code préfixe

Le code précédent vérifie la **CONDITION DU PREFIXE**.

Une façon simple de vérifier la condition du préfixe ou de construire un code vérifiant cette condition est de tracer un graphe orienté en forme d'arbre binaire.

Cet arbre n'est pas forcément complet.

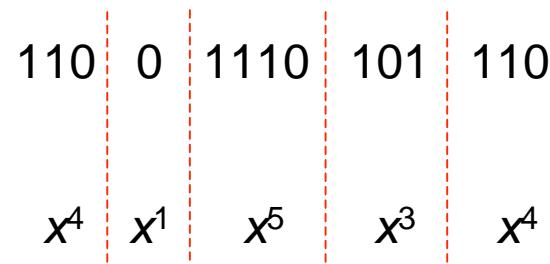


Codage d'Une Source

Code instantané uniquement décodable : Exemple de code préfixe

Le code précédent est dit **INSTANTANE**.

Le décodage des symboles se fait sans référence aux mots de code futurs :



On se limitera par la suite à ce cas particulier.

Codage d'Une Source

Synthèse

- Les codes à condition de préfixe sont à décodage unique.
- Tous les codes à décodage unique ne satisfont pas nécessairement la condition du préfixe.
- La condition du préfixe permet de reconnaître la fin d'un mot de code, donc permet un décodage sans retard : le code est dit instantané.

Codage d'Une Source

Inégalité de KRAFT

Pour tout code préfixe de L_X mots sur un alphabet de 2 symboles dont les mots de code ont pour longueurs $l(c^1), l(c^2), \dots, l(c^{L_X})$, ces longueurs entières satisfont :

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq 1$$

Inversement, étant donnés des longueurs entières $l(c^1), l(c^2), \dots, l(c^{L_X})$ qui satisfont cette inégalité, on peut construire un code préfixe dont les mots de code ont ces longueurs.

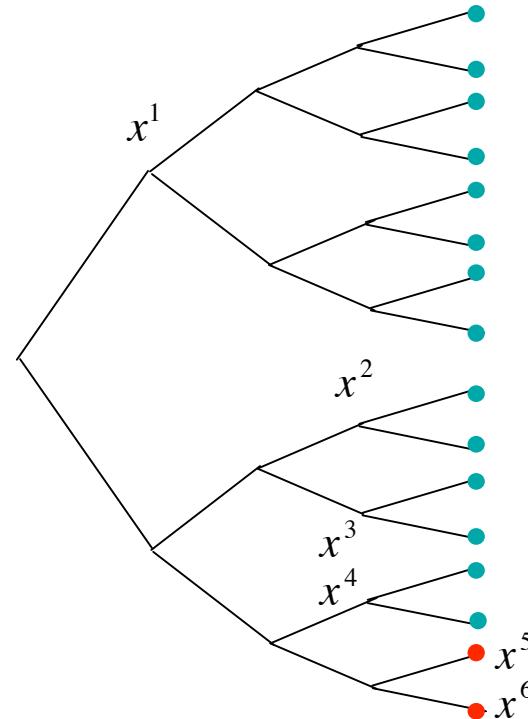
Codage d'Une Source

Inégalité de KRAFT : Démonstration (1/3)

Soit l_{\max} la longueur du mot le plus long du code

On considère un arbre binaire de profondeur l_{\max} les feuilles au niveau l_{\max} sont, soit :

- Des mots de code (1)
- Des descendants de mots de code (2)
- Aucun des deux (3)



Codage d'Une Source

Inégalité de KRAFT : Démonstration (2/3)

- Un mot de code de longueur $l(c^i)$ possède $2^{l_{\max} - l(c^i)}$ descendants.
- La condition du préfixe implique que l'ensemble de ces descendants doit être disjoint pour tous les mots de code.
- Si l'on considère l'ensemble des feuilles de niveau l_{\max} qui sont, ou descendant d'un mot de code, soit (1)+(2), il y en a donc :

$$\sum_{i=1}^{L_X} 2^{l_{\max} - l(c^i)}$$

Codage d'Une Source

Inégalité de KRAFT : Démonstration (3/3)

Cet ensemble (1)+(2) de feuilles étant mots de code ou descendants de mots de code est évidemment inclus dans l'ensemble (1)+(2)+(3) de toutes les feuilles de niveau l_{\max} . On a donc :

$$\sum_{i=1}^{L_X} 2^{l_{\max} - l(c^i)} \leq 2^{l_{\max}}$$

En divisant par $2^{l_{\max}}$ on obtient alors l'inégalité de Kraft :

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq 1$$

Codage d'Une Source

Code OPTIMAL (1/5)

Relâchons la contrainte suivant laquelle les quantités $l(c^i)$ doivent être des entiers et remplaçons le signe de l'inégalité (dans l'inégalité de Kraft) par une égalité. Le code optimal est celui qui minimise la longueur moyenne :

$$\bar{l} = \sum_{i=1}^{L_X} p_X(i) l(c^i)$$

sous la contrainte :

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} = 1$$

Codage d'Une Source

Code OPTIMAL (2/5)

Ce problème d'optimisation peut être résolu en introduisant les opérateurs de Lagrange et donc la fonctionnelle suivante :

$$J_\lambda\left(\{l(c^i)\}\right) = \sum_{i=1}^{L_X} p_X(i) l(c^i) + \lambda \left(\sum_{i=1}^{L_X} 2^{-l(c^i)} - 1 \right)$$

Les conditions du premier ordre entraînent que :

$$\begin{cases} \frac{\partial J_\lambda\left(\{l(c^i)\}\right)}{\partial l(c^i)} = 0 \\ \frac{\partial J_\lambda\left(\{l(c^i)\}\right)}{\partial \lambda} = 0 \end{cases}$$

Codage d'Une Source

Code OPTIMAL (3/5)

On a :
$$\frac{\partial}{\partial l(c^i)} \left[\sum_{i=1}^{L_X} p_X(i) l(c^i) + \lambda \left(\sum_{i=1}^{L_X} 2^{-l(c^i)} - 1 \right) \right] = 0$$

$$\Rightarrow p_X(i) + \lambda \frac{\partial}{\partial l(c^i)} \left[e^{-l(c^i) \ln 2} \right] = 0$$

$$\Rightarrow p_X(i) - \lambda 2^{-l(c^i)} \ln 2 = 0$$

Soit :

$$2^{-l(c^i)} = \frac{p_X(i)}{\lambda \ln 2}$$

Codage d'Une Source

Code OPTIMAL (4/5)

De plus : $\frac{\partial}{\partial \lambda} \left[\sum_{i=1}^{L_X} p_X(i) l(c^i) + \lambda \left(\sum_{i=1}^{L_X} 2^{-l(c^i)} - 1 \right) \right] = 0$

$$\Rightarrow \sum_{i=1}^{L_X} 2^{-l(c^i)} = 1$$

ou encore :

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} = \frac{1}{\lambda \ln 2} \sum_{i=1}^{L_X} p_X(i) = 1$$

cela impose la valeur de la constante $\lambda \ln 2 = 1$. On obtient donc

$$2^{-l(c^i)} = p_X(i)$$

$$\Rightarrow l(c^i) = -\log_2 p_X(i)$$

Codage d'Une Source

Code OPTIMAL (5/5)

La longueur moyenne correspondant au code optimal est donnée par :

$$\bar{l} = \sum_{i=1}^{L_X} p_X(i)l(c^i) = -\sum_{i=1}^{L_X} p_X(i)\log_2 p_X(i)$$

$$\bar{l} = H(X)$$

Parmi tous les codes vérifiant la condition du préfixe, celui qui minimise la longueur moyenne des mots du code a une longueur moyenne égale à l'entropie de la source. L'entropie $H(X)$ apparaît donc comme une limite fondamentale pour représenter **sans distorsion** une source d'information.

Codage d'Une Source

Théorème du codage sans bruit d'une source discrète sans mémoire

Pour toute source discrète sans mémoire $X(n)$, il existe un code instantané représentant exactement cette source et uniquement décodable vérifiant :

$$H(X) \leq \bar{l} \leq H(X) + 1$$

où $H(X)$ est l'entropie de la source et \bar{l} la longueur moyenne du code.

Construction d'Un Code

Code de SHANNON (1/2)

La façon la plus simple de procéder pour construire un code est de choisir :

$$l(c^i) = \lceil -\log_2 p_X(i) \rceil$$

où $\lceil x \rceil$ représente le plus petit entier supérieur ou égal à x . On a alors :

$$\begin{aligned} \bar{l} &= \sum_{i=1}^{L_X} p_X(i) \underbrace{\lceil -\log_2 p_X(i) \rceil}_{\leq -\log_2 p_X(i) + 1} \Rightarrow \bar{l} \leq -\sum_{i=1}^{L_X} p_X(i) \log_2 p_X(i) + \sum_{i=1}^{L_X} p_X(i) \leq H(X) + 1 \end{aligned}$$

Construction d'Un Code

Code de SHANNON (2/2)

Comme

$$2^{-\lceil -\log_2 p_X(i) \rceil} \leq 2^{\log_2 p_X(i)}$$

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq \sum_{i=1}^{L_X} p_X(i)$$

$$\sum_{i=1}^{L_X} 2^{-l(c^i)} \leq 1$$

vérifie l'inégalité de Kraft, on peut déduire qu'il existe un code instantané construit sur la base du code de Shannon qui vérifie la condition du préfixe.

Construction d'Un Code

Code d'HUFFMAN : Algorithme

L'algorithme d'Huffman consiste à construire progressivement un arbre binaire en partant des noeuds terminaux.

- On part des deux listes $\{ x^1 \dots x^{L_x} \}$ et $\{ p(x^1) \dots p(x^{L_x}) \}$
- On sélectionne les deux symboles les moins probables, on crée deux branches dans l'arbre et on les étiquette par deux symboles binaires 0 et 1.
- On actualise les deux listes en rassemblant les deux symboles utilisés en un nouveau symbole et en lui associant comme probabilité la somme des deux probabilités sélectionnées.
- On recommence les deux étapes précédentes tant qu'il reste plus d'un symbole dans la liste.

Construction d'Un Code

Code d'HUFFMAN : Exemple (1/3)

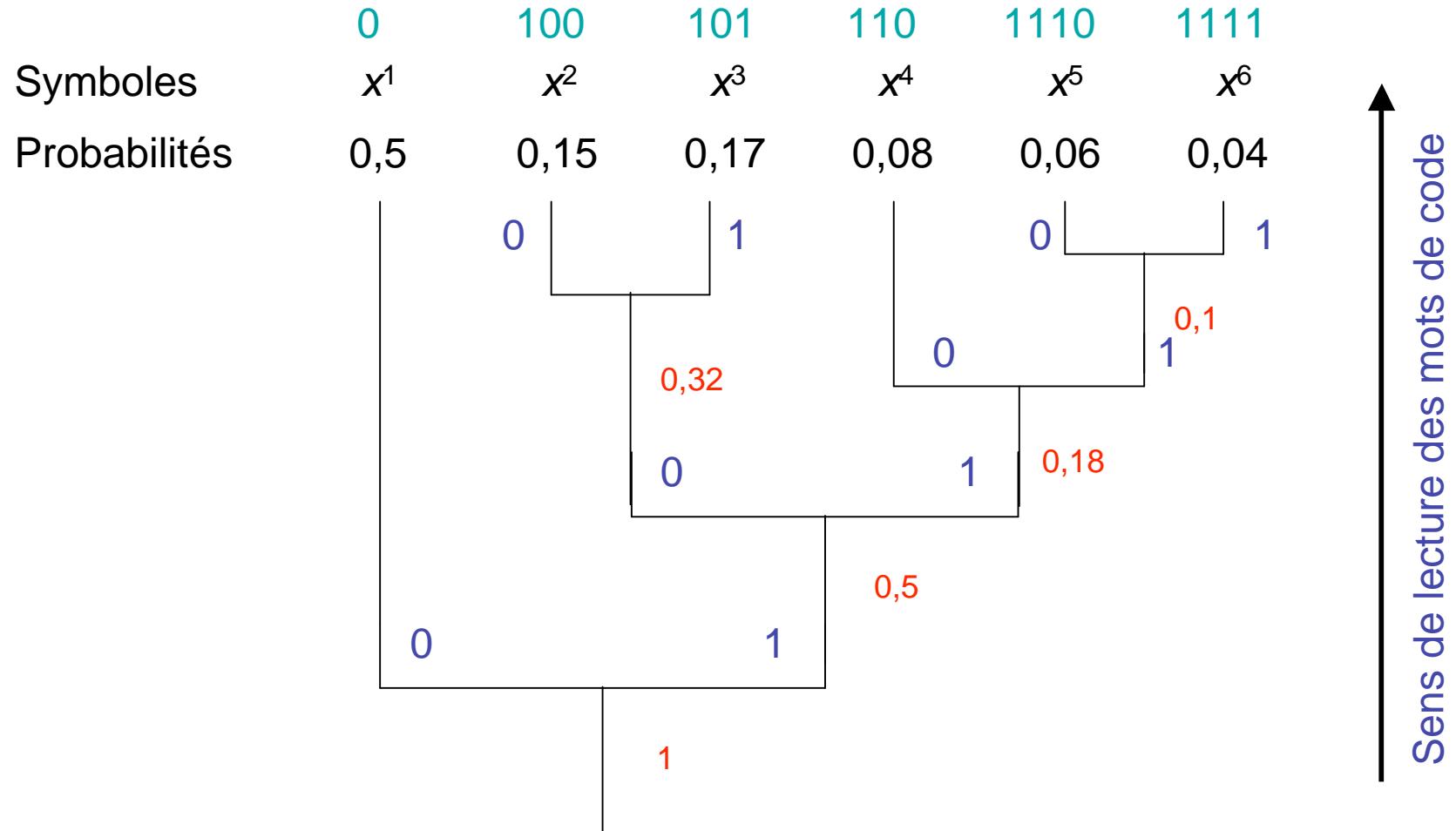
Prenons l'exemple d'une source ne pouvant prendre que six valeurs différentes et supposons connues les probabilités. Elles sont données ci-après :

Symbol	x^1	x^2	x^3	x^4	x^5	x^6
Probabilités	0,5	0,15	0,17	0,08	0,06	0,04

L'entropie de cette source est égale à 2,06 bits/échantillon.

Construction d'Un Code

Code d'HUFFMAN : Exemple (2/3)



Construction d'Un Code

Code d'HUFFMAN : Exemple (3/3)

Ici, le code d'Huffman a généré un code dont la longueur moyenne des mots est égale à 2,1 bits/échantillon.

Le code de Shannon aurait permis d'obtenir un code dont les mots de code auraient une longueur moyenne égale à 2,28 bits/échantillon.

Rappelons que l'entropie de cette source est égale à 2,06 bits/échantillon.

Construction d'Un Code

Le codage ARITHMETIQUE

- Les codes arithmétiques ont été développés par PASCO et RISSANEN en 1976.
- Le principe est de ne plus construire le message codé en concaténant les mots de codes de chaque symbole (comme dans Huffman) mais plutôt de **faire correspondre à toute chaîne de symbole s un nombre binaire qui constitue son code.**
- Le codage arithmétique repose sur la notion de **séparation code/modèle** : l'encodeur, en fonction du modèle statistique de la source produit une certaine chaîne codée. Le décodeur à accès au même modèle et à partir de la chaîne codée et de celui-ci, est capable de décoder exactement la chaîne de départ.
- Le modèle étant indépendant du codage, on peut utiliser le codage arithmétique avec diverses sortes de modèles => **codes arithmétiques adaptatifs.**

Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (1/4)

L'idée de base du codage arithmétique est d'associer à chaque symbole du message **un intervalle dont la taille est proportionnelle à la probabilité d'apparition de celui-ci.**

Exemple d'association MESSAGE / INTERVALLE :

Soit une source S , prenant ses valeurs dans l'alphabet $\{a, b, c, d\}$ avec la distribution de probabilités $P=\{0,08, 0,22, 0,24, 0,46\}$.

Soit $s= « dacdba »$ la chaîne de symboles à coder.

On appellera s_L la partie déjà lue de la chaîne s . Au départ la chaîne s_L est vide (on note $s_L = \delta$), on lui associe l'intervalle $[0,1[$.

L'idée est de découper cet intervalle en sous-intervalles. La taille de chaque sous-intervalle est proportionnelle à la probabilité d'apparition du symbole de l'alphabet qui lui est associé.

Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (2/4)

Symbol	Probabilité	Interval associé
b	0,22	[0 , 0,22[
a	0,08	[0,22 , 0,3[
d	0,46	[0,3 , 0,76[
c	0,24	[0,76 , 1[

On ne tient pas compte de l'ordre des symboles pour faire le découpage



Découpage de l'intervalle unité pour S_L initial (vide).

Si on prend un nombre au hasard, par exemple $x=0,8$, il est possible d'associer à cette probabilité un intervalle, donc un symbole.

En effet, $x \in [0,76,1[$ décode le symbole c.

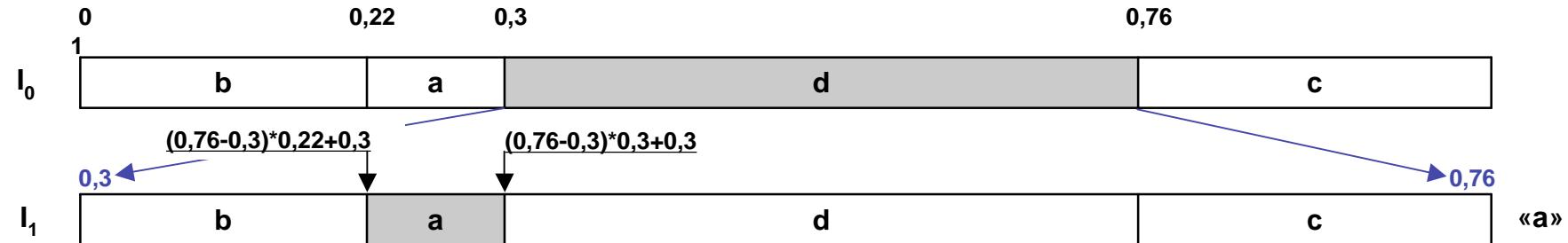
Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (3/4)



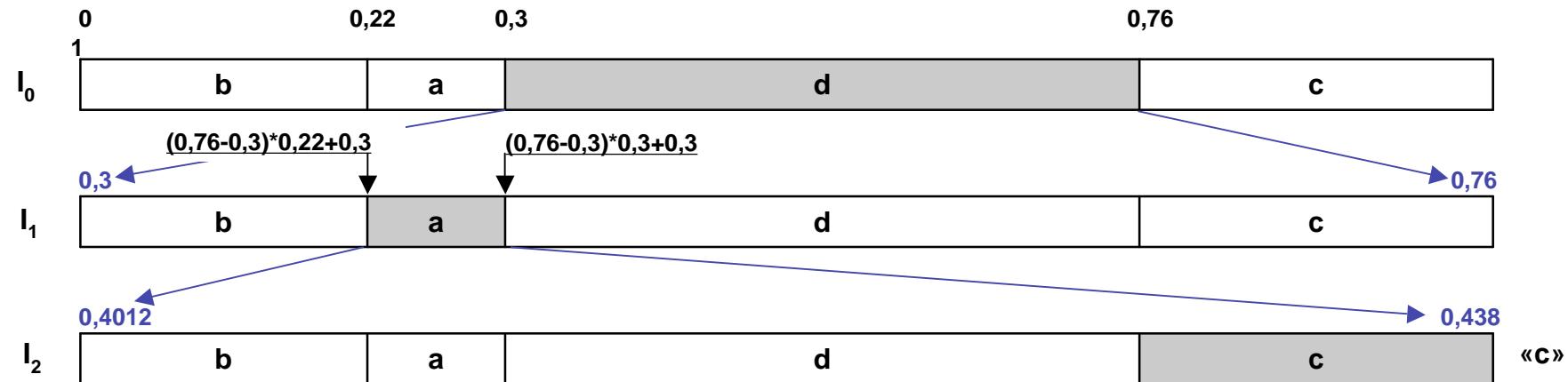
Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (3/4)



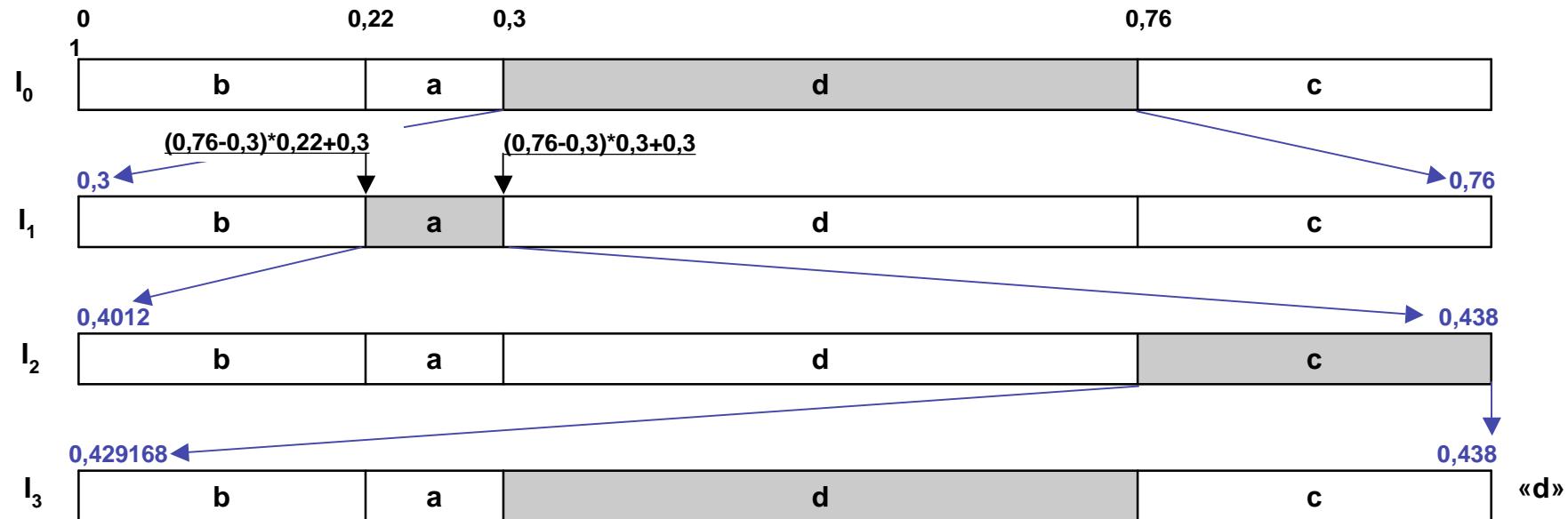
Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (3/4)



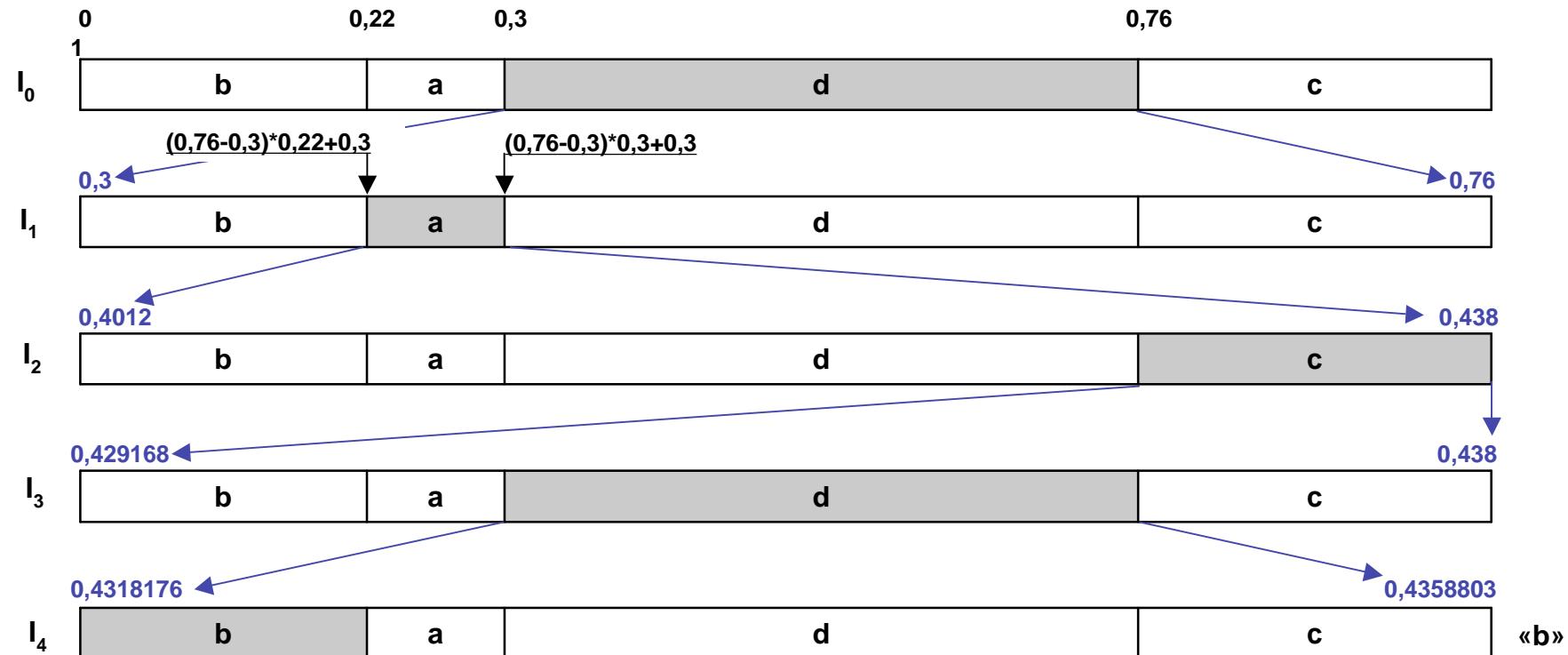
Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (3/4)



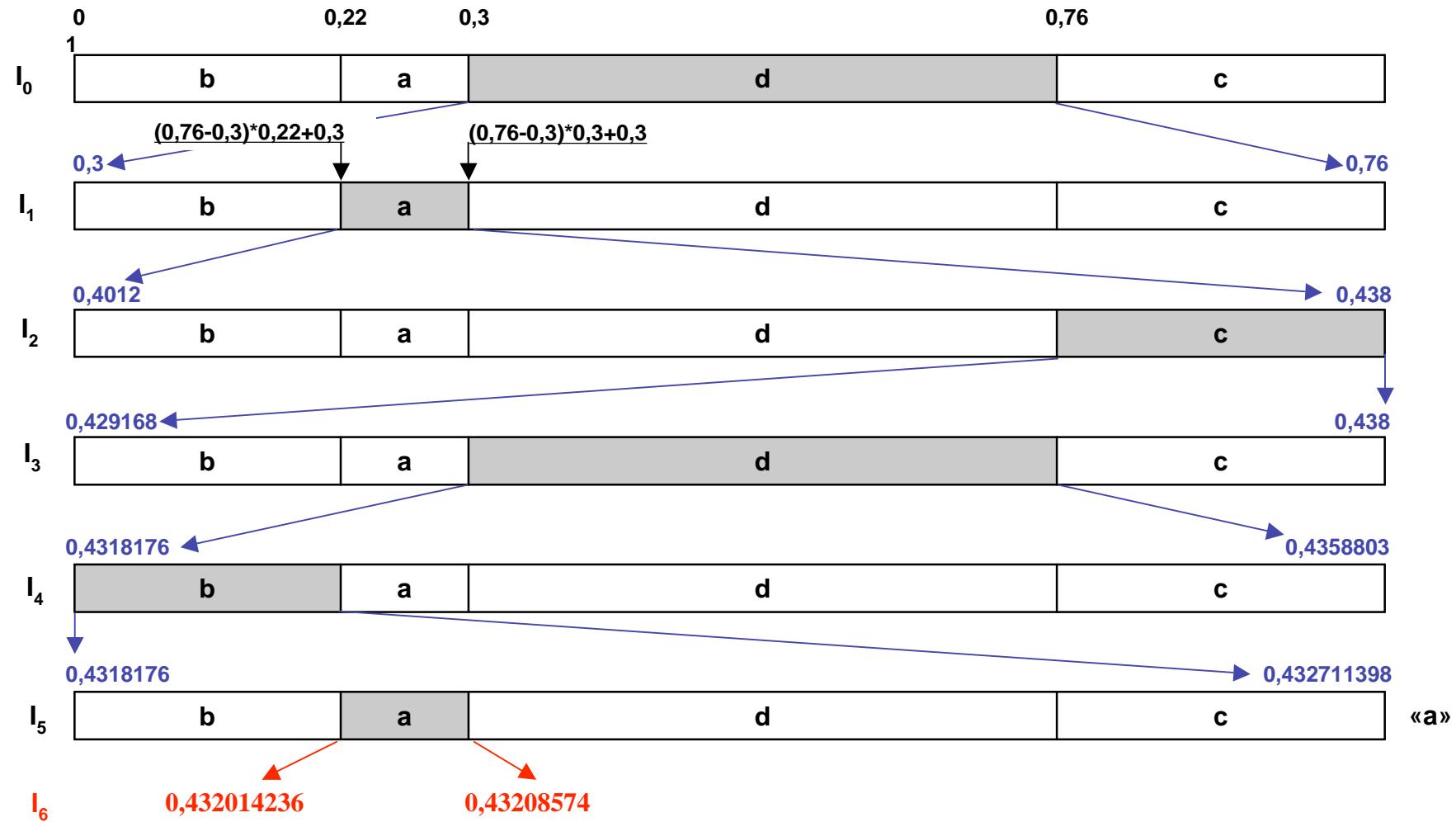
Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (3/4)



Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (3/4)



Construction d'Un Code

Le codage ARITHMETIQUE : association MESSAGE/INTERVALLE (4/4)

Etape	Message lu	Intervalle courant
0	δ	$[0, 1[$
1	« d »	$[0,3, 0,76[$
2	« d a »	$[0,4012, 0,438[$
3	« d a c »	$[0,429168, 0,438[$
4	« d a c d »	$[0,4318176, 0,4358803[$
5	« d a c d b »	$[0,4318716, 0,432711398[$
6	« d a c d b a »	$[0,432014236, 0,43208574[$

Découpage pour la chaîne de symboles « d a c d b a ».

Nous avons associé à un message à coder, un intervalle réel ouvert à droite inclus dans l'intervalle $[0,1[$.

Construction d'Un Code

Le décodage ARITHMETIQUE (1/3)

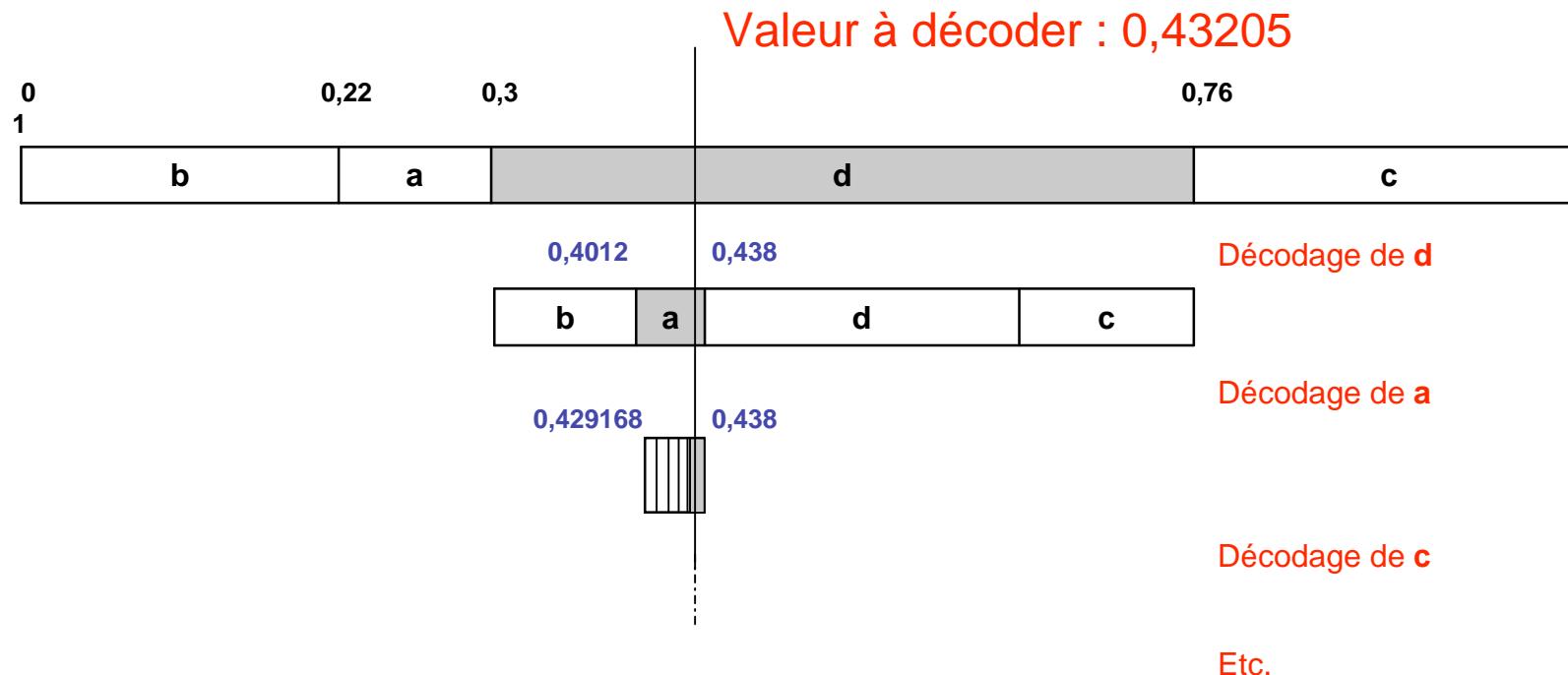
Nous savons associer à chaque message un intervalle de probabilités qui le représente.

Réiproquement, choisissons un nombre compris dans l'intervalle $[0,432014236 , 0,43208574[$, par exemple **$x=0,43205$** . Ce nombre est un représentant de cet intervalle.

1. x appartient à $[0,3 , 0,76[$ le sous intervalle associé à **d**. Nous pouvons donc décoder le symbole **d**
2. Si nous effectuons le découpage de cet intervalle comme précédemment, on peut raffiner cet intervalle et remarquer que $[0,4012 , 0,438[$ est le sous-intervalle associé à **a**. Nous pouvons donc décoder le symbole **a**
3. Nous avons donc décodé la chaîne « **d a** ». Etc...

Construction d'Un Code

Le décodage ARITHMETIQUE (2/3)



Construction d'Un Code

Le décodage ARITHMETIQUE (3/3)

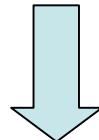
- Nous avons mis en évidence une opération de découpage qui permet d'associer à un message un intervalle de probabilités ;
- Cette opération de découpage est **indépendante** de la façon dont on acquiert les symboles du message ;
- Grâce au modèle statistique, codeur et décodeur sont capables d'effectuer **le même découpage en intervalles**. Il suffit d'avoir le même modèle côté codeur et côté décodeur ;
- **La simple connaissance d'une probabilité** dans l'intervalle final du découpage permet alors le décodage de la chaîne de symboles.

Construction d'Un Code

Le codage ARITHMETIQUE : Transmission incrémentale (1/5)

Problème : Les divisions successives de l'intervalle unité posent le problème de précision arithmétique : plus le message à coder est long, plus la description de l'intervalle qui lui est associé nécessite de décimales.

Dans la pratique la précision arithmétique utilisable est toujours limitée :



On peut donc arriver à des situations où les BORNES INFERIEURE et SUPERIEURE de l'intervalle sont EGALES et le codage pas terminé !!

IDEE : TRANSMISSION INCREMENTALE

Construction d'Un Code

Le codage ARITHMETIQUE : Transmission incrémentale (2/5)

Remarque : Lorsque les premiers chiffres des bornes basse et haute d'un intervalle associé à un préfixe de message sont identiques alors elles le resteront dans la suite du codage (grâce à la propriété d'emboîtement des intervalles).

- ➡ Les premières décimales du représentant sont alors connues
- ➡ On peut donc les transmettre
- ➡ On peut décaler les bornes de l'intervalle d'une position.

Construction d'Un Code

Le codage ARITHMETIQUE : Transmission incrémentale (3/5)

Exemple : Soit un message $M = \langle m \ s_1 \ s_2 \dots s_n \rangle$

Supposons que l'intervalle associé à la sous chaîne m soit $[0,2456 , 0,2680[$. Cet intervalle est inclus dans $[0,2 , 0,3[$ et tous ses sous intervalles seront inclus dans $[0,2 , 0,3[$. D'après la relation d'emboîtement, l'intervalle associé à M est forcément compris dans $[0,2 , 0,3[$.

La première décimale du représentant associé à M est donc déterminée et égale à 2. Ce chiffre peut donc être transmis (car inutile au reste du codage) et les bornes de l'intervalle peuvent être décalées :

$[0,2456 , 0,2680[$ devient $[0,456 , 0,680[$

Construction d'Un Code

Le codage ARITHMETIQUE : Transmission incrémentale (4/5)

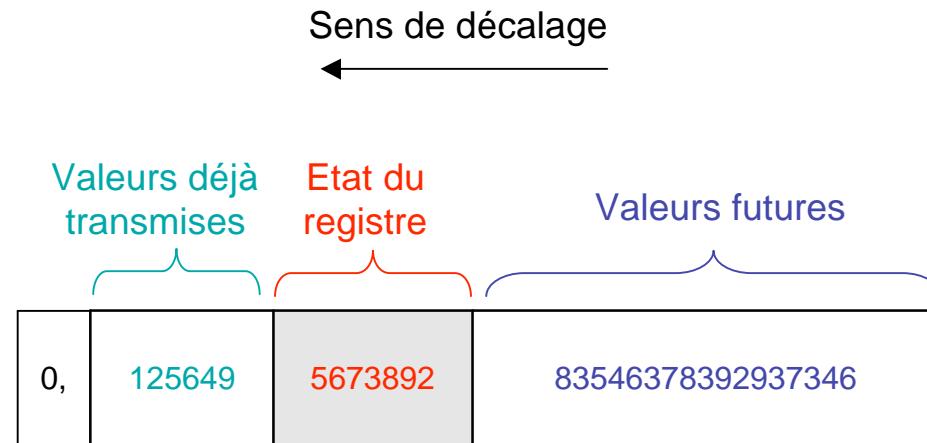
Remarques :

- Au décodage il faudra aussi décaler le représentant courant
- On représente la borne inférieure de l'intervalle par un registre à décalage (la borne inférieure est un représentant possible [de coût minimal])
- dans la pratique on utilise une numération binaire du représentant et non pas décimale (même principe)

Construction d'Un Code

Le codage ARITHMETIQUE : Transmission incrémentale (5/5)

Illustration :



Registre à décalage à sept chiffres (numération décimales)
ou 24 bits en équivalent binaire

Construction d'Un Code

Le codage ARITHMETIQUE : Algorithme de CODAGE

Répéter :

- 1- Encoder le symbole courant
- 2- Actualiser le modèle
(si le codage est adaptatif)

ENCODAGE :

- a- Calculer le nouvel intervalle de codage en fonction de l'ancien et de la probabilité d'apparition du symbole (fournie par le modèle) ;
- b- Tant que les deux bornes de l'intervalle ont leur premier chiffre binaire identique :
Transmettre ce chiffre et décaler d'une position les bits dans le registre.

Jusqu'à ce qu'il n'y ait plus de symboles à coder

Transmettre les derniers chiffres de l'intervalle courant

Construction d'Un Code

Le codage ARITHMETIQUE : Algorithme de DECODAGE

Répéter :

- 1- Décoder le symbole courant
- 2- Actualiser le modèle
(si le codage est adaptatif)

DECODAGE :

- a- Trouver le symbole qui correspond à la valeur courante du représentant ;
- b- Calculer le nouvel intervalle de codage en fonction de l'ancien et de la probabilité du symbole courant (fournie par le modèle) ;
- c- Tant que les deux bornes de l'intervalle ont leur premier chiffre binaire identique :
Décaler d'une position les bits dans le registre, le dernier bit du représentant étant lu dans le message codé.

Jusqu'à ce qu'il n'y ait plus de symboles à coder

Construction d'Un Code

Le codage ARITHMETIQUE : Conclusion

- Le codage arithmétique est plus efficace que le codage d'Huffman
- Il permet d'obtenir implicitement des mots de code de longueurs non entières pour les symboles sources : on n'affecte pas un mot de code par symbole source mais un mot global pour toute la séquence
- Il permet une séparation claire entre la source et le modèle statistique de la source : rend facile l'adaptation par rapport à la source
- Il offre la possibilité d'effectuer du codage contextuel de façon simple et ainsi de prendre en compte la corrélation ou mémoire de la source



Le Codage par Plages De Zéros

Le codage Par Plages De Zéros

« Run Length Coding » (RLC)

IDEE :

Représenter de la meilleure façon possible une séquence du type :

... $X_1 \ 0 \ 0 \ 0 \ 0 \ 0 \ X_2 \ 0 \ 0 \ 0 \ X_3 \ 0 \dots$

Le codage Par Plages De Zéros

« Run Length Coding » (RLC) : PRINCIPE

Le RLC consiste à coder des **plages de zéros** entre 2 valeurs non nulles du signal. Les coefficients non nuls sont décomposés en 2 parties encodées séparément :



la catégorie CAT

La valeur CAT de la catégorie détermine le nombre de bits nécessaire pour coder la valeur du coefficient non nul (Cf. tableau). Elle dépend donc de l'amplitude de cette valeur.



l'index

L'index représente le codage en binaire de la valeur du coefficient non nul.

*Il est constitué de **CAT bits**.*

Le codage Par Plages De Zéros

La trame à transmettre

La trame à transmettre sera donc constituée d'une suite de couples :

(**RUN**, **CAT**) index

- **RUN** = nombre de "0" consécutifs
- **CAT** = la catégorie de la valeur non nulle
- **Index** = le codage en binaire (sur CAT bits) de la valeur non nulle

Le codage Par Plages De Zéros

Exemple de tableau de catégories (CAT)

CATEGORIE CAT (nombre de bits d'index)	VALEURS (COEFFICIENTS NON NUL)
0	0
1	-1 , 1
2	- 3 - 2 , 2 3
3	-7 ... - 4 , 4 ... 7
4	- 15... - 8 , 8 ... 15
5	- 31... - 16 , 16 ... 31
6	- 63... - 32 , 32 ... 63

Le codage Par Plages De Zéros

Exemple de codage

La séquence :

... 6 0 0 0 0 0 0 3 0 0 0 24 0 ...

est représentée par la trame suivante :

$\underbrace{(\text{run} = 1, \text{cat} = 3) 110}_{\text{Codé en binaire}}$ $(\text{run} = 6, \text{cat} = 2) 11$ $(\text{run} = 3, \text{cat} = 5) 11000$

Codé en binaire

Le codage Par Plages De Zéros

La représentation binaire

Chaque couple (RUN, CAT) est **codé en binaire** :

- On définit un nombre de zéros consécutif maximum : RUN maximum.
- On définit une valeur de catégorie maximale : CAT maximale.

En général les couples sont codés à l'aide d'un code entropique (Huffman...), donc en fonction de leur probabilité d'apparition.

Le codage Par Plages De Zéros

Le décodage

DECODAGE :

1. On décode d'abord les couples (RUN,CAT)
2. Connaissant la valeur de CAT, on connaît le nombre de bits qui code l'*index*.
3. On décode l'*index*.

REMARQUE :

Il existe des méthodes autres que le "Run Length Coding" .

Par exemple le "Stack Run" (*nombre maximum de zéros consécutifs non limité, possibilité d'utiliser un codage arithmétique*).



La Quantification Vectorielle Non Structurée

Motivations



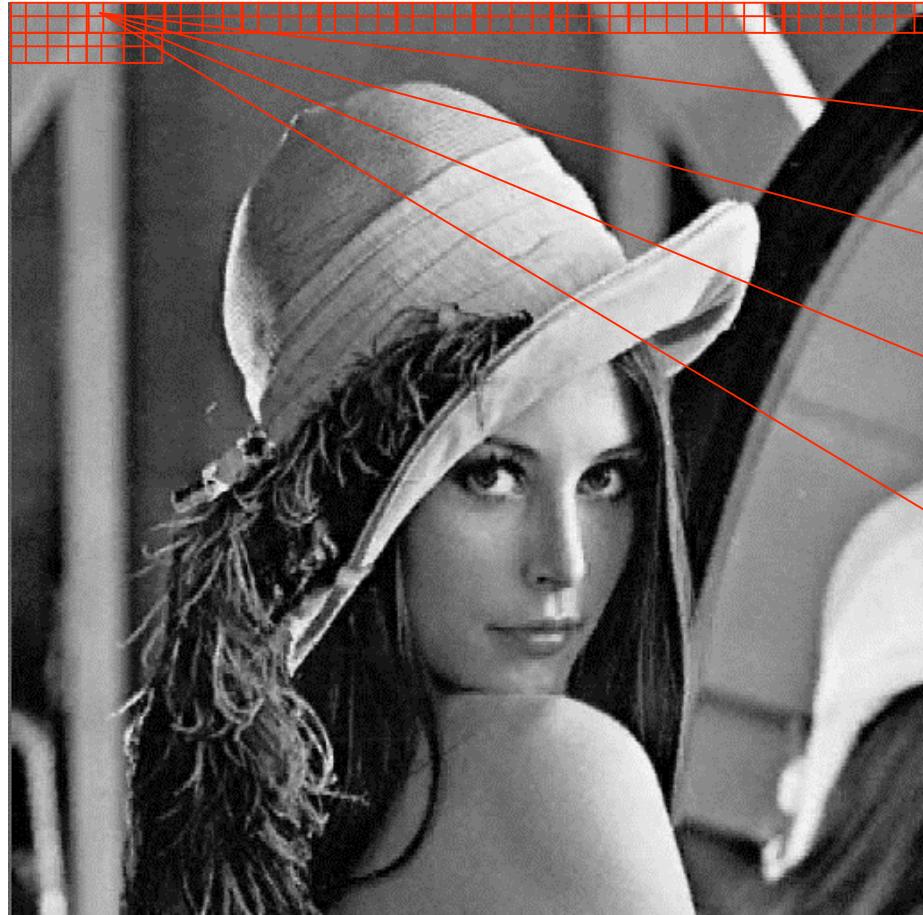
Lena 512x512 pixels 8 bits/pixel (bpp) => 256 niveaux de gris

Motivations

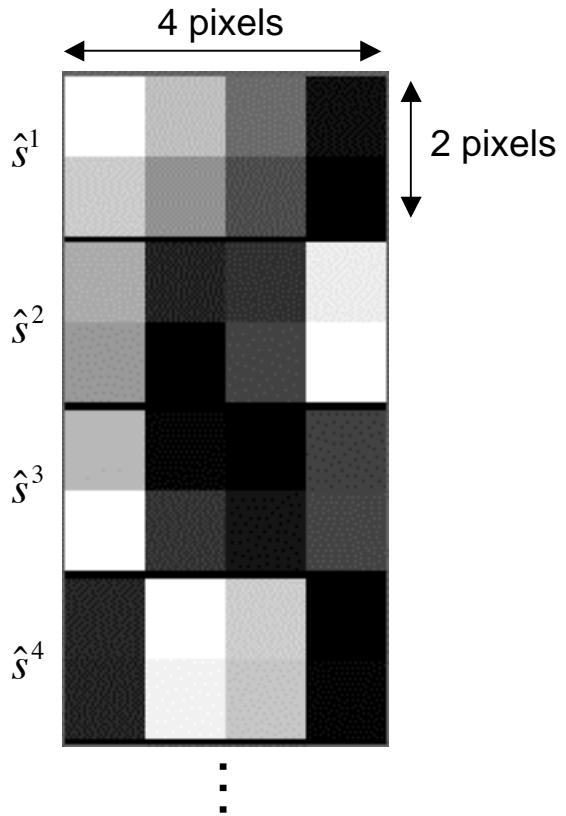


QUANTIFICATION SCALaire 1 bpp => Taux de compression = 8

Motivations



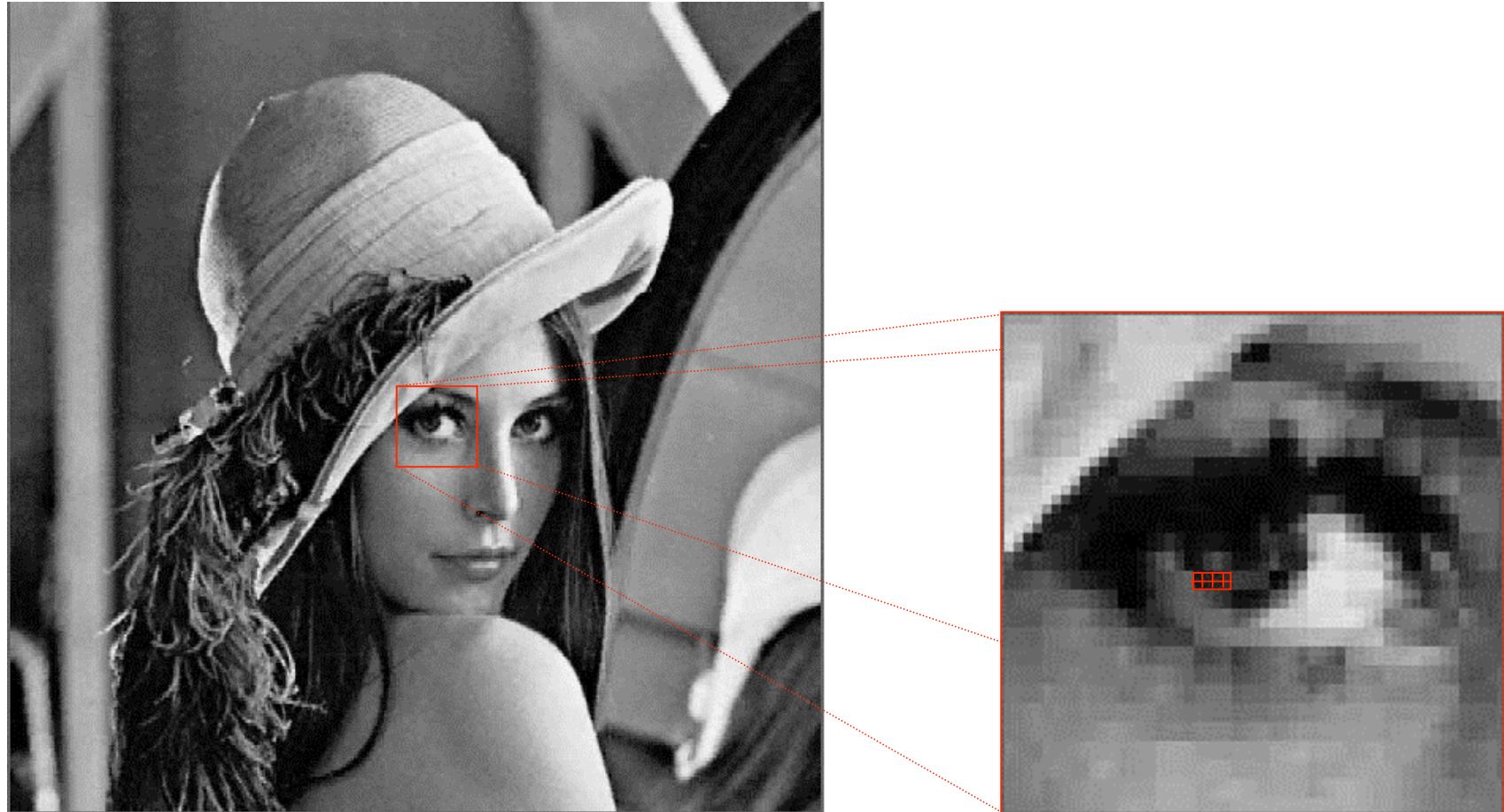
Dictionnaire $\{\hat{s}^1 \dots \hat{s}^L\}$



$$L=256 \Rightarrow \log_2(256)=8 \text{ bits/vecteur}$$

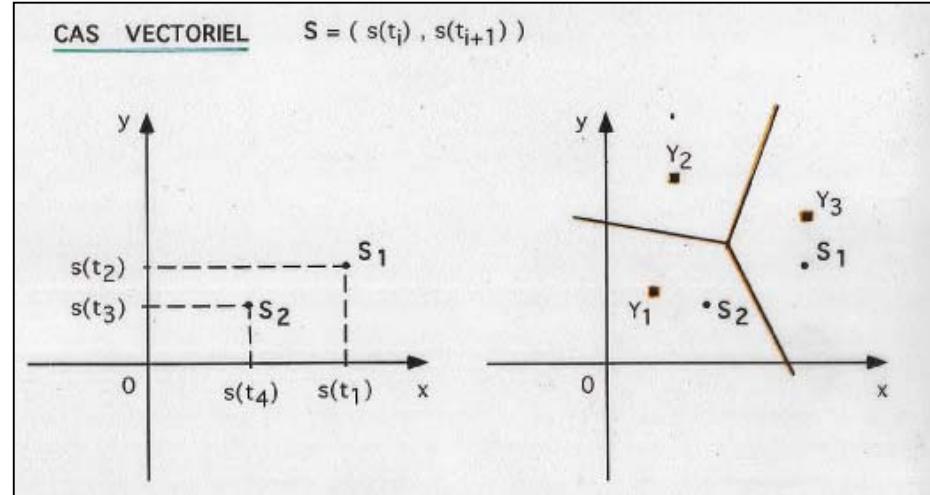
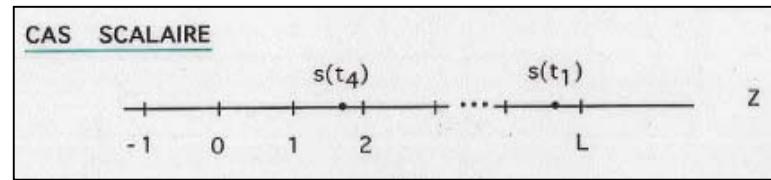
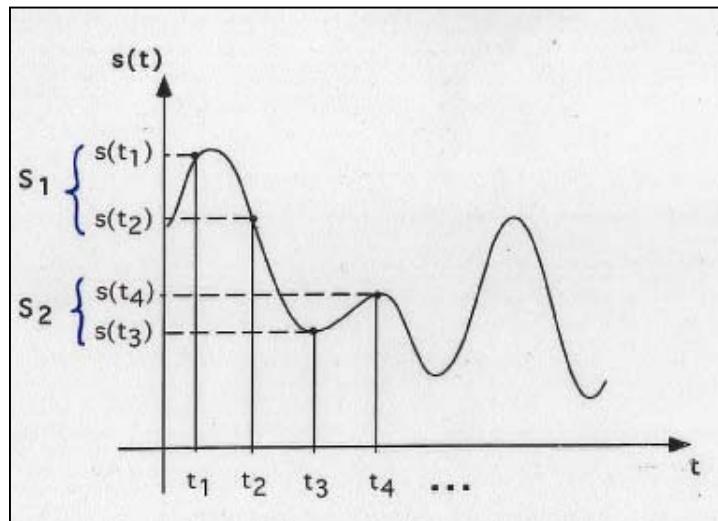
QUANTIFICATION VECTORIELLE 1 bpp => Taux de compression = 8

Motivations



QUANTIFICATION VECTORIELLE 1 bpp => Taux de compression = 8

Motivations : Un exemple en 1D



La Quantification Vectorielle

Formalisation (1/4)

On appelle quantificateur vectoriel de dimension N et de taille L une application de \mathbf{R}^N dans un ensemble fini C contenant L vecteurs de dimension N .

$$Q : \mathbf{R}^N \rightarrow C \quad \text{avec} \quad C = \left\{ \hat{s}^1 \dots \hat{s}^L \right\} \quad \text{où} \quad \hat{s}^i \in \mathbf{R}^N$$

L'espace est partitionné en L régions ou cellules définies par :

$$P^i = \left\{ s : Q(s) = \hat{s}^i \right\}$$

On appelle C , un dictionnaire (« codebook » en Anglais) et \hat{s}^i un représentant (vecteur de sortie ou de reproduction). On dit également que C représente l'alphabet de reproduction et \hat{s}^i les symboles de reproduction conformément au vocabulaire habituel en théorie de l'information.

La Quantification Vectorielle

Formalisation (2/4)

Il faut définir une mesure de distorsion $d(.,.)$. Les deux choix habituels sont la distance Euclidienne :

$$d(s, \hat{s}^i) = \frac{1}{N} \|s - \hat{s}^i\|^2 = \frac{1}{N} (s - \hat{s}^i)^t (s - \hat{s}^i)$$

ou une distance pondérée :

$$d(s, \hat{s}^i) = \frac{1}{N} (s - \hat{s}^i)^t W (s - \hat{s}^i)$$

où W est une matrice définie positive permettant d'introduire un caractère perceptuel.

La Quantification Vectorielle

Formalisation (3/4)

Pour apprécier les performances d'un quantificateur, on utilise comme critère la distorsion moyenne. Soit $S(m)$ un vecteur aléatoire dans \mathbf{R}^N et $p_S(x)$ sa densité de probabilité conjointe. La distorsion moyenne à pour expression :

$$D = E[d(S(m), \hat{s})] = \int_{\mathbf{R}^N} d(x, \hat{s}) p_S(x) dx$$

$$D = \sum_{i=1}^L \int_{x \in P^i} d(x, \hat{s}^i) p_S(x) dx$$

La Quantification Vectorielle

Formalisation (4/4)

Lorsque l'on choisit comme mesure de distorsion la distance Euclidienne, la distorsion moyenne devient l'erreur quadratique moyenne :

$$D = \frac{1}{N} \int_{\mathbf{R}^N} \|x - \hat{s}\|^2 p_s(x) dx$$

La relation fondamentale qui relie la résolution b , la dimension N des vecteurs et la taille L du dictionnaire est la suivante :

$$L = 2^{bN}$$

La résolution représente le nombre de bits par échantillon. Il n'est plus nécessaire que b soit entier, il suffit que bN le soit. La quantification vectorielle permet donc de définir des résolutions fractionnaires.

La Quantification Vectorielle

Principe

Le principe de la Quantification Vectorielle repose sur deux étapes :

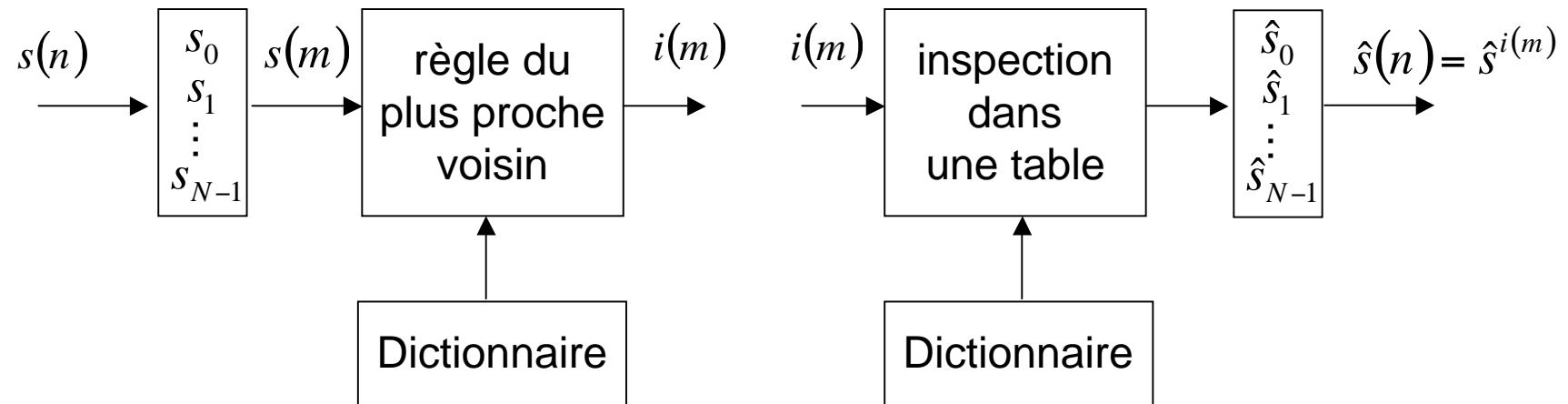
1. La recherche du plus proche représentant dans un dictionnaire, vérifiant (**recherche EXHAUSTIVE**) :

$$\min_{i \in \{1, \dots, L\}} d(s, \hat{s}_i)$$

2. Le codage et la transmission (ou le stockage) de l'index i obtenu à l'étape précédente

La Quantification Vectorielle

Schéma d'encodage/décodage



La Quantification Vectorielle

Le quantificateur optimal

Il s'agit de déterminer le dictionnaire $C = \{ \hat{s}^1 \dots \hat{s}^L \}$ en choisissant N, L .

Comme dans le cas scalaire, il n'est pas possible de définir simultanément la meilleure partition et les meilleurs vecteurs de reproduction mais on garde les deux conditions d'optimalité qui s'expriment de façon identique.

La Quantification Vectorielle

Conditions d'optimalité

1. Etant donné un dictionnaire C , la meilleure partition est celle qui vérifie

$$P^i = \left\{ s \in \mathbf{R}^N : d(s, \hat{s}^i) \leq d(s, \hat{s}^j) \quad \forall j \in \{1 \dots L\} \right\}$$

C'est la règle du plus proche voisin. Cette partition est appelée partition de Voronoï.

2. Etant donné une partition, les meilleurs représentants sont obtenus par la condition du centroïde. Pour une distortion quadratique :

$$\hat{s}^i = E[S / S \in P^i] = \frac{\int x p_S(x) dx}{\int p_S(x) dx}$$

ou encore $\hat{s}^i = \frac{1}{N_i} \sum_{s \in P_i} s$



Nb. de vecteurs dans la classe P_i

La Quantification Vectorielle

Le quantificateur optimal

Pour définir le dictionnaire C , c'est-à-dire construire les vecteurs de reproduction, on est amené à utiliser une base d'apprentissage, comme dans le cas scalaire. Elle doit être composée d'un grand nombre M de vecteurs représentatifs de la source.

Typiquement, [en image](#), on estime que chaque vecteur du dictionnaire doit être construit à partir [d'au moins 16](#) de vecteurs de la base.

Le principe de l'algorithme de [**Lloyd généralisé au cas vectoriel**](#) (GLA – Generalized Lloyd Algorithm) reste identique à celui du cas scalaire.

La Quantification Vectorielle

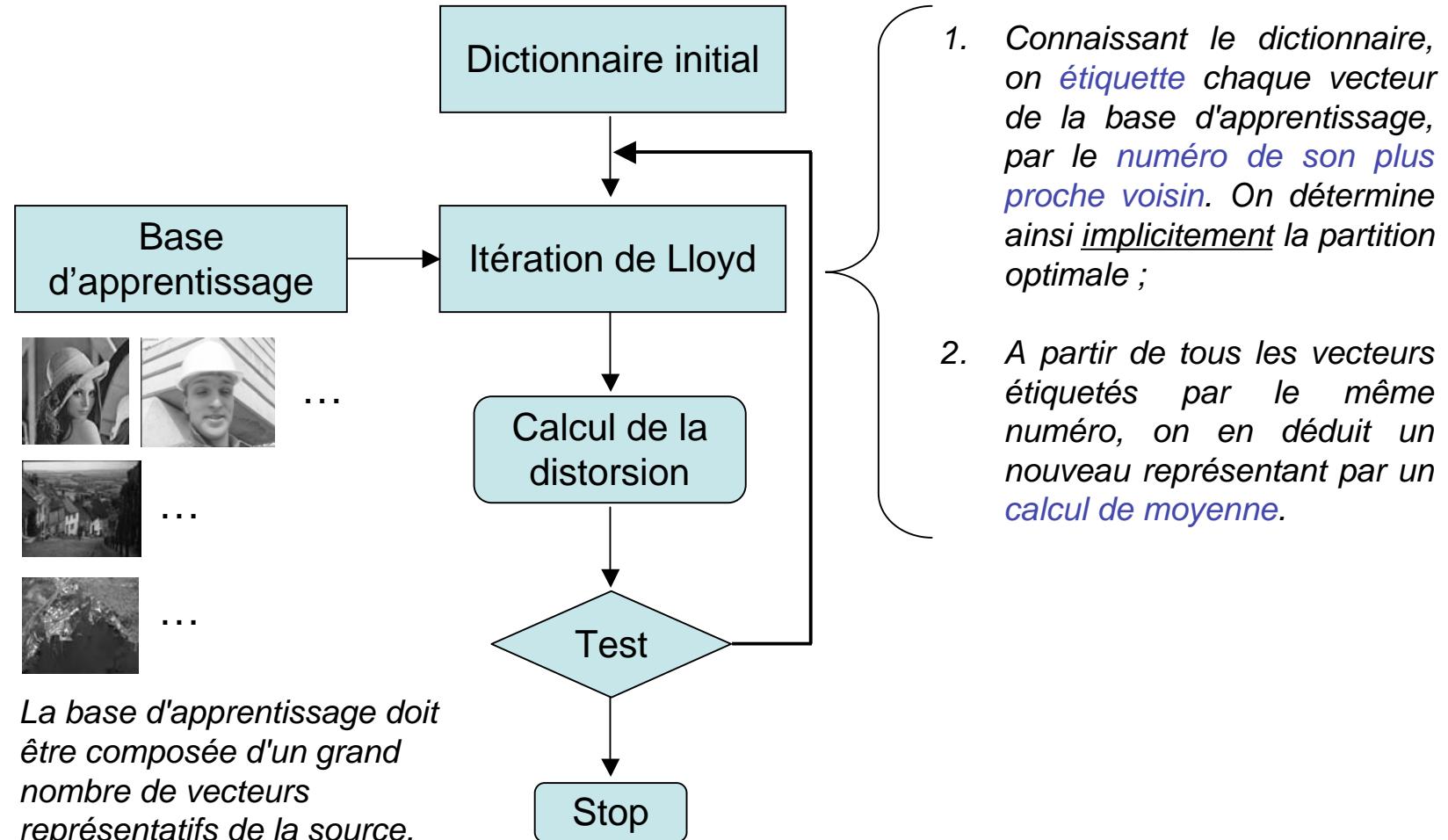
L'Algorithme de Lloyd Généralisé (GLA)

Cet algorithme de classification est basé sur la simple observation que l'algorithme de Lloyd (1957) pour la quantification scalaire est aussi valide pour **des vecteurs**, pour des **distributions échantillonnées** et pour une **grande variété de critères de distorsion**. Son principe consiste à modifier de manière itérative un dictionnaire de vecteurs, généralisant ainsi l'algorithme de Lloyd.

1. Initialiser le dictionnaire,
2. Appliquer successivement la règle du plus proche voisin et la condition du centroïde,
3. Itérer l'étape précédente tant que la décroissance de la distorsion moyenne reste importante.

La Quantification Vectorielle

Algorithme de Lloyd-Max GENERALISE : ORGANIGRAMME



La Quantification Vectorielle

Exemple de partition

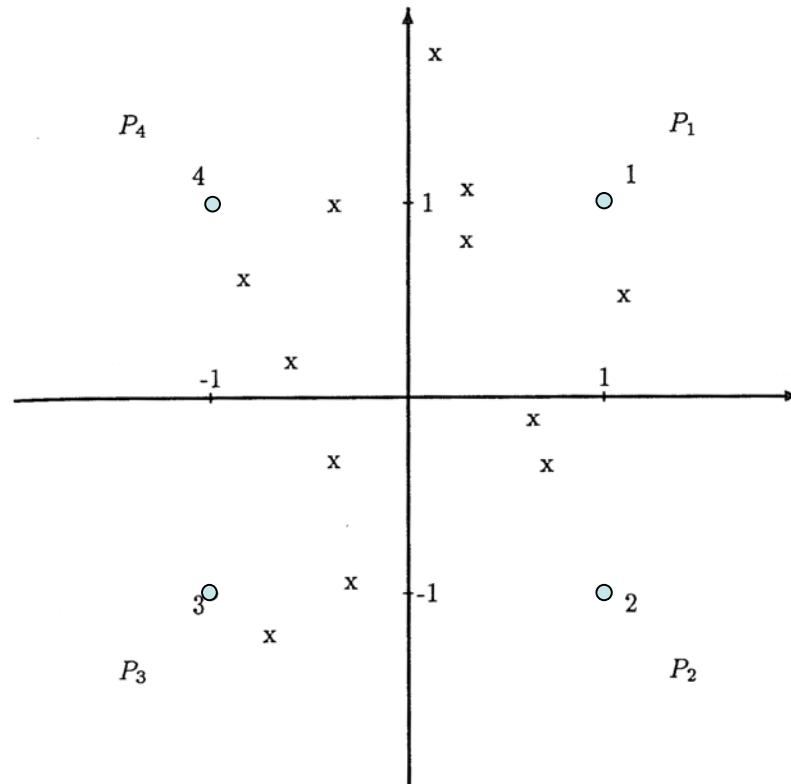


Figure 11.4: Training Sequence and Initial Code Book

La Quantification Vectorielle

Exemple de partition

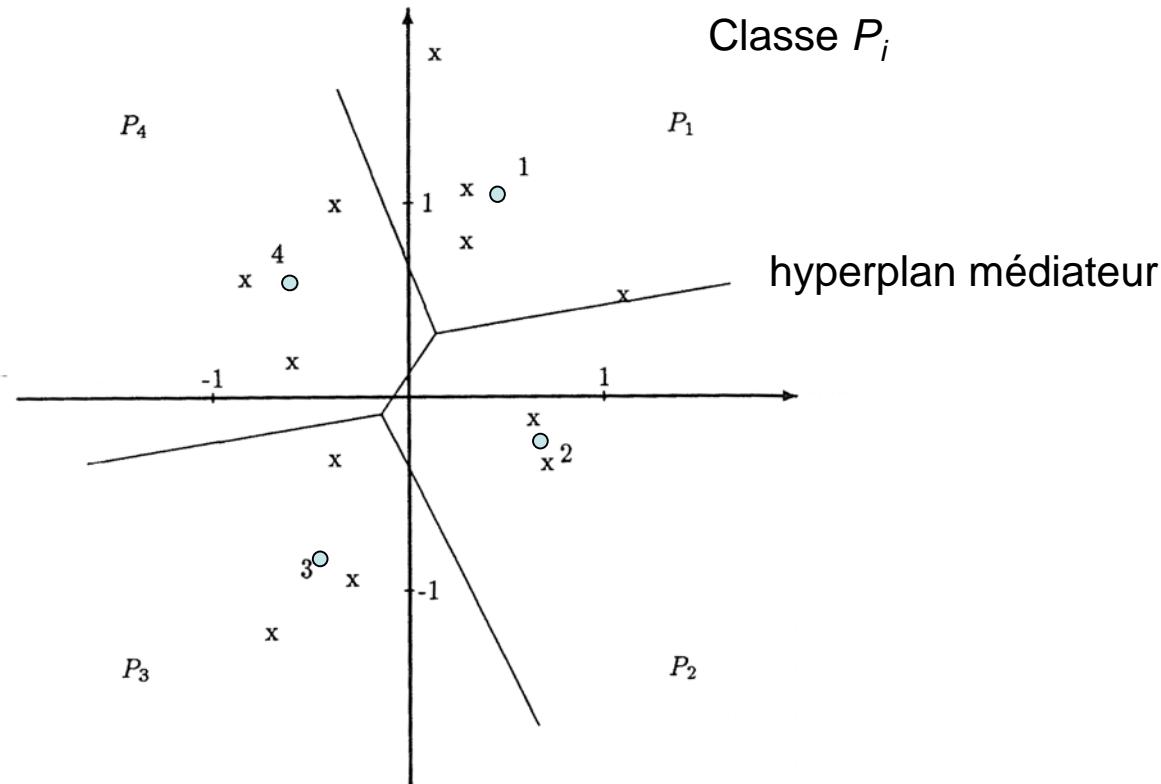
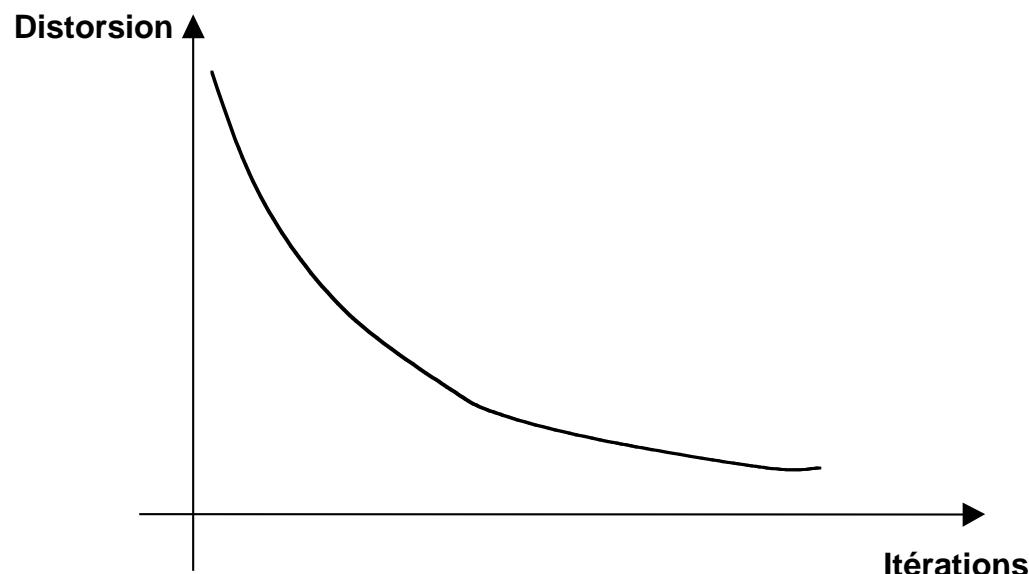


Figure 11.5: Centroids of Nearest Neighbor Regions

La Quantification Vectorielle

Convergence de l'algorithme

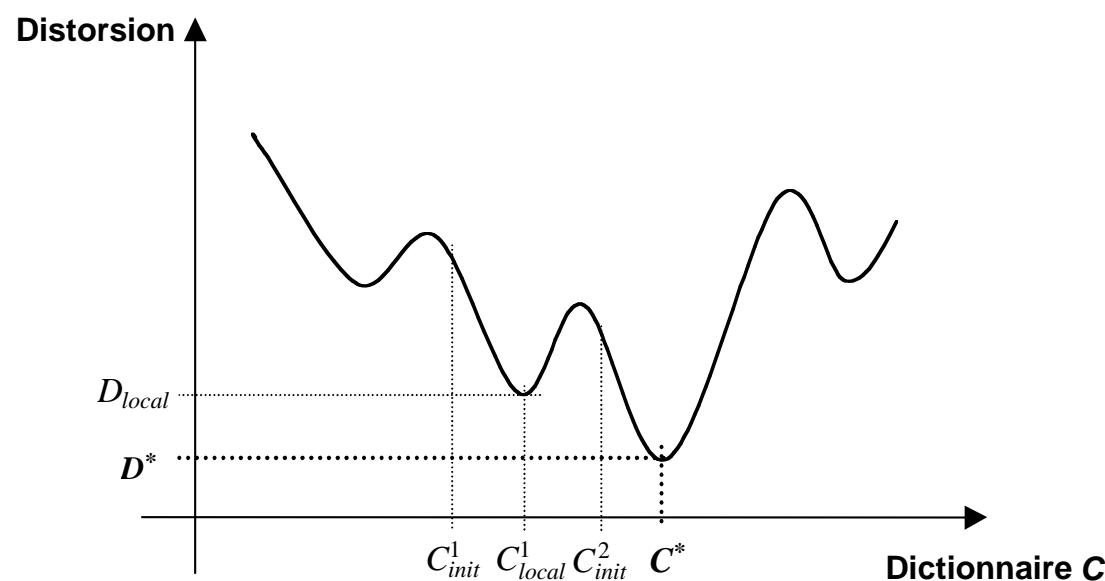
L'algorithme GLA est un algorithme itératif qui diminue ou laisse inchangée la distorsion du dictionnaire, au cours des itérations. La courbe distorsion en fonction des itérations de l'algorithme GLA est donc une fonction monotone décroissante.



La Quantification Vectorielle

Convergence de l'algorithme

Cependant, le minimum de distorsion obtenu est un minimum local qui sera plus ou moins "bon", c'est-à-dire proche du minimum global, selon le dictionnaire initial choisi. En effet, l'initialisation du dictionnaire pose un problème.



La Quantification Vectorielle

Convergence de l'algorithme

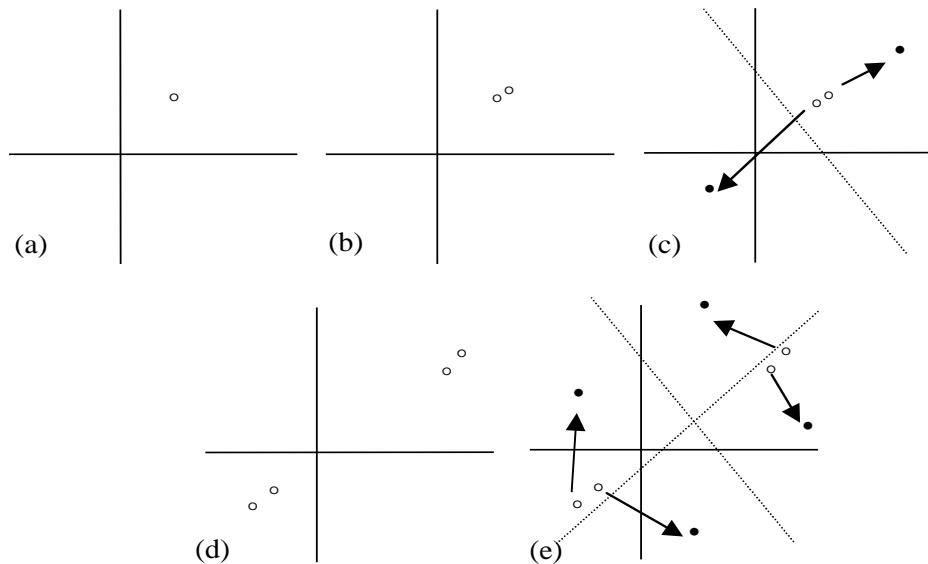
On peut accepter un temps de calcul très lourd, par exemple plusieurs jours de temps CPU sur une grosse machine, car ce traitement est réalisé une seule fois en temps différé.

La convergence de l'algorithme pose un problème comme dans le cas scalaire. La distorsion totale sur l'ensemble d'apprentissage décroît nécessairement mais ne tend pas forcément vers un minimum global. On atteint simplement un minimum local. De nouveaux algorithmes basés sur des techniques de recuit simulé, par exemple, permettent (en théorie) d'améliorer les performances du quantificateur.

La Quantification Vectorielle

Problème de conditions initiales : L'algorithme LBG (Linde-Buzo-Gray)

L'**initialisation** du dictionnaire pose un problème comme dans le cas scalaire. L'algorithme dit LBG (Linde-Buzo-Gray), généralement adopté permet de résoudre ce problème.

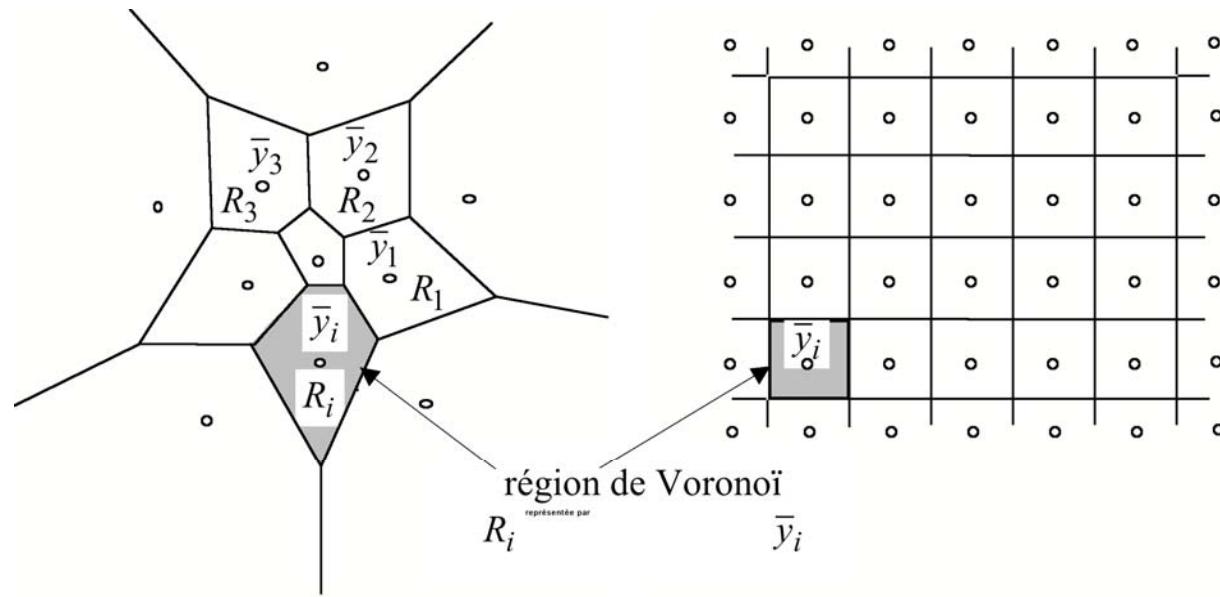


Technique du partage ("splitting"). A chaque itération, chaque vecteur est découplé en 2 nouveaux vecteurs.

- (a) Etat initial : centre de gravité de la séquence d'apprentissage.
- (b) Estimation initiale 1 : dictionnaire à 2 niveaux.
- (c) Estimation finale 1 après GLA : dictionnaire optimal à 2 niveaux.
- (d) Estimation initiale 2 : dictionnaire à 4 niveaux.
- (e) Estimation finale 2 après GLA : dictionnaire optimal à 4 niveaux.

La Quantification Vectorielle

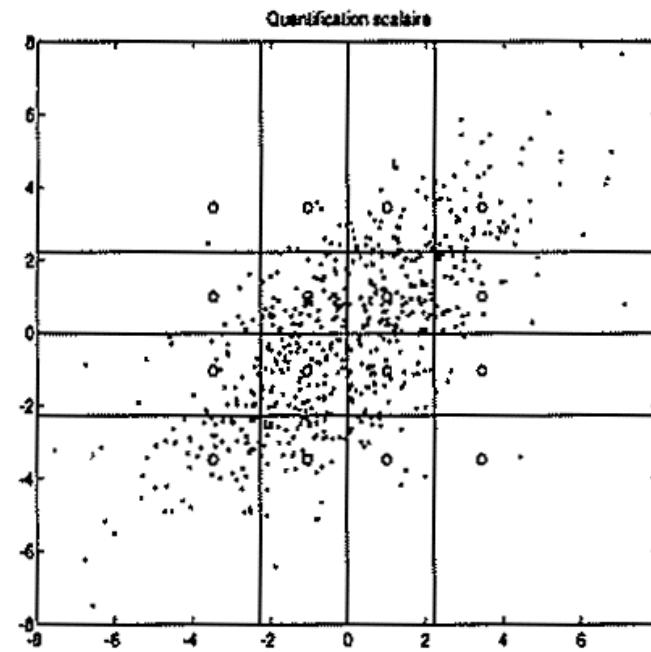
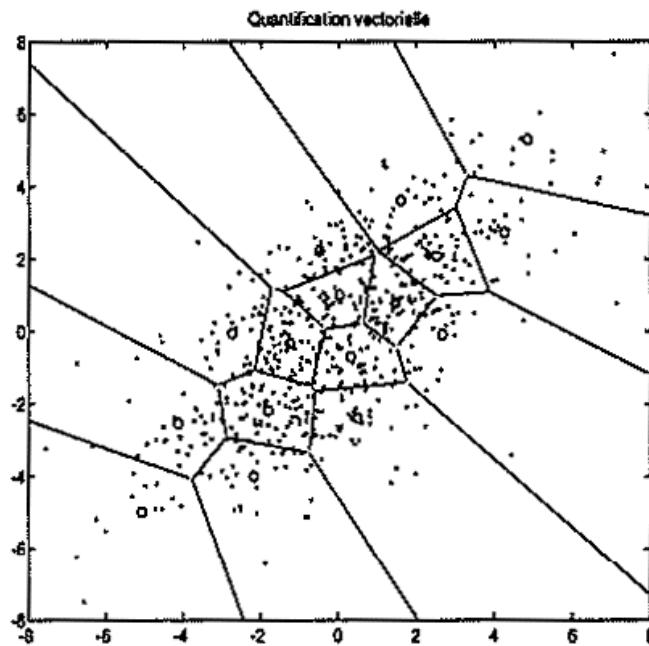
Comparaison : partition régulière/irrégulière



Partitionnements régulier et irrégulier d'un espace à 2 dimensions. Chaque vecteur x appartenant à une région R_i est quantifié par $\bar{y}_i = Q(x)$

La Quantification Vectorielle

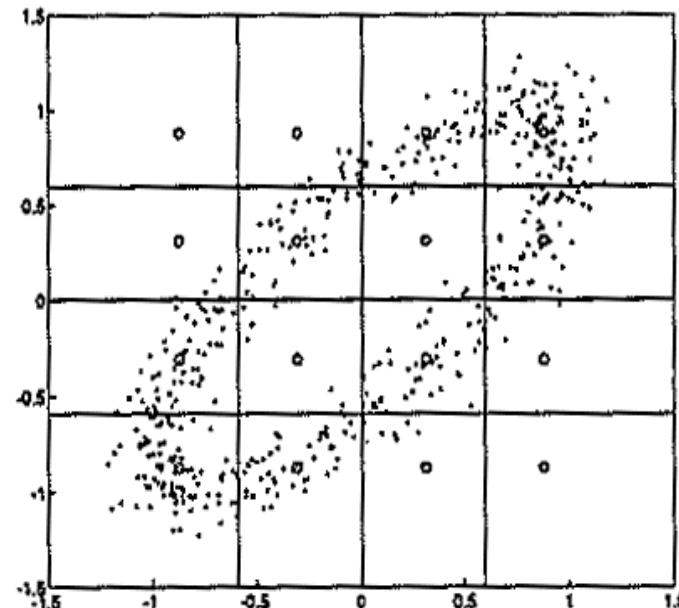
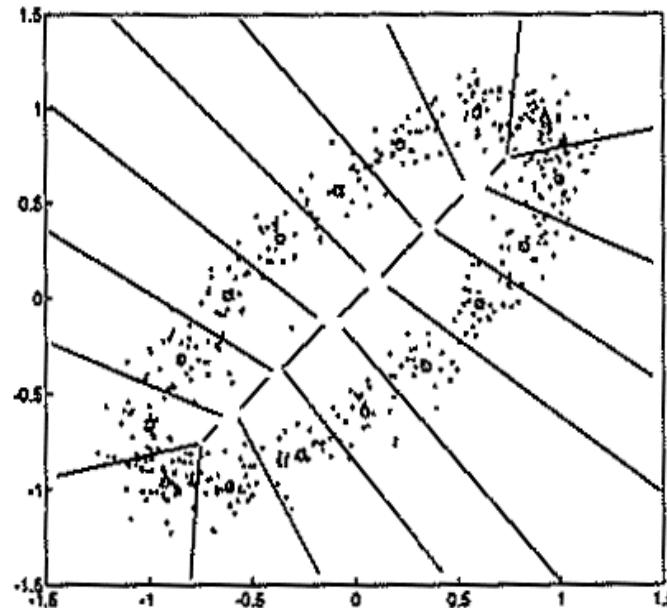
Comparaison : partition régulière/irrégulière



Comparaison des performances entre quantification vectorielle et quantification scalaire lorsque le débit est égal à 2. la quantificateur vectoriel comprend $L=16$ vecteurs de dimension 2. Le quantificateur scalaire comprend $L=4$ représentants.

La Quantification Vectorielle

Comparaison : partition régulière/irrégulière



Comparaison des performances entre quantification vectorielle et quantification scalaire pour une sinusoïde entachée de bruit.

La Quantification Vectorielle

Distorsion d'un quantificateur vectoriel (1/4)

HYPOTHESE Haute Résolution approximativement vérifiée lorsque le débit (résolution) R est suffisamment élevé. Elle consiste à admettre que la densité de probabilité $p_S(x)$ est approximativement constante dans tous les éléments de la partition de Voronoï.

Moyennant cette hypothèse, l'Erreur Quadratique Moyenne (la puissance de l'erreur de quantification) admet une expression analytique que l'on se propose d'analyser ici.

On suppose que les représentants $\{\hat{s}^1 \dots \hat{s}^L\}$ sont les représentants optimaux. Ils ont été déterminés, par exemple, par application de l'algorithme de Lloyd-Max généralisé.

La Quantification Vectorielle

Distorsion d'un quantificateur vectoriel (2/4)

La distorsion d'un quantificateur vectoriel traduit l'erreur de quantification qu'il engendre. On définit ainsi de façon générale :

$$D_{QV}(R) = \mathbb{E}[d(S, Q(s))]$$

où $\mathbb{E}[\cdot]$ traduit l'espérance mathématique et R le débit binaire du quantificateur exprimé en bits par échantillon.

On peut également écrire :

$$D_{QV}(R) = \sum_{i=1}^L \int_{P_i} \|s - \hat{s}^i\|_2^2 p_S(s) ds$$

où $p_S(s)$ est la densité de probabilité conjointe du vecteur aléatoire S .

La Quantification Vectorielle

Distorsion d'un quantificateur vectoriel (3/4)

Dans le cas où L est grand et la densité de probabilité $p_S(s)$ est "douce", on peut faire l'approximation suivante :

$$D_{QV}(R) \approx \sum_{i=1}^L \frac{\Pr\{\hat{s}^i\}}{\text{Vol}(P_i)} \int_{P_i} \|s - \hat{s}^i\|_2^2 ds$$

où $\text{Vol}(P_i)$ est le volume du Voronoï P_i , et $\Pr\{\hat{s}^i\}$ la probabilité d'apparition d'un vecteur du dictionnaire. Au contraire du cas scalaire, la formule précédente n'est pas simple à calculer pour des dictionnaires non structurés, à cause de la forme irrégulière des Voronoï.

La Quantification Vectorielle

Distorsion d'un quantificateur vectoriel (4/4)

Zador a montré que la borne asymptotique maximale de distorsion, donnée par la formule suivante dans le cas d'un quantificateur scalaire et d'une norme quadratique (cf. cours QS) :

$$D_{QS}(R) = \frac{1}{12} \left[\int_{-\infty}^{+\infty} \tilde{p}_S(s)^{1/3} ds \right]^3 2^{-2R}$$

où $\tilde{p}_S(s)$ est la densité de probabilité marginale, se généralise au cas vectoriel par la formule :

$$D_{QV}(R) = G(N) \left[\int_{\mathbf{R}^N} p_S(s)^{N/(N+2)} ds \right]^{(N+2)/N} 2^{-2R}$$

où $G(N)$ est une constante qui ne dépend que de N .

La Quantification Vectorielle

Le moment d'ordre 2

Le coefficient $G(N)$ correspond au second moment normalisé du quantificateur. Il permet d'évaluer le bruit granulaire d'un quantificateur dans l'hypothèse d'une distribution de source uniforme à l'intérieur d'un Voronoï.

Dans le cas des réseaux réguliers de points (cf. suite du cours), et selon le réseau utilisé, il est possible d'avoir une formule analytique simple pour l'expression du second moment :

$$G(N) = \frac{1}{N} \frac{\int_{P_0} \|s\|^2 ds}{\text{Vol}(P_0)^{\frac{N+2}{N}}}$$

où N est la dimension du vecteur s , et P_0 la cellule de Voronoï contenant l'origine du quantificateur (réseau).

La Quantification Vectorielle

Le moment d'ordre 2 : Exemple de calcul

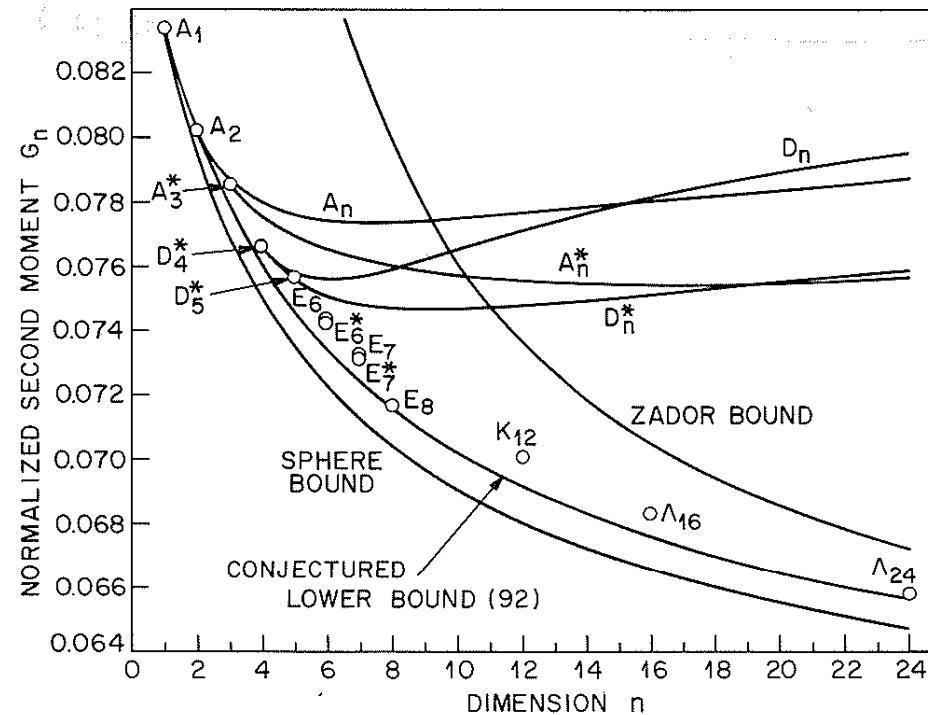
Dans le cas du réseau \mathbf{Z}^N ($q=1$), nous avons :

$$\begin{aligned} G_{\mathbf{Z}^N}(N) &= \frac{1}{N} \frac{\int_{P_0} \sum_{k=1}^N x_k^2 dx_1 dx_2 \dots dx_n}{1} \\ &= \frac{1}{N} \sum_{k=1}^N \int_{-1/2}^{1/2} x_k^2 dx_k \\ &= \frac{1}{N} \sum_{k=1}^N \frac{1}{12} \\ &= \frac{1}{12} \end{aligned}$$

La Quantification Vectorielle

Le moment d'ordre 2

Les valeurs des coefficients $G(N)$, correspondant au second moment normalisé du quantificateur, ont été tabulés par Conway et Sloane pour différents quantificateurs (réseaux).



La Quantification Vectorielle

Performances du quantificateur optimal

Pour apprécier les performances de la QV appliquée à une réalisation du vecteur aléatoire $S(m)$ comparée à la QS sur $S(n)$ (m est l'indice de vecteur et n l'indice d'échantillon), on définit le rapport de 2 bruits D_{QS} et D_{QV} correspondants aux distorsions créées respectivement par un quantificateur scalaire comportant 2^R représentants et par un quantificateur vectoriel comportant 2^{NR} représentants (c'est-à-dire ayant le même débit binaire fixe R exprimé en bits par échantillon) :

$$\text{Gain}(N) = \frac{D_{QS}}{D_{QV}}$$

La Quantification Vectorielle

Performances du quantificateur optimal

Notation :

- $\tilde{p}_S(s)$ est la densité de probabilité marginale, c'est-à-dire la densité de probabilité de n'importe quelle composante du vecteur aléatoire puisque le processus est supposé stationnaire
- $\bar{p}_S(s)$ est la densité de probabilité du vecteur si toutes les composantes étaient indépendantes

On a donc :

$$\bar{p}_S(s) = \prod_{k=0}^{N-1} \tilde{p}_S(x_k)$$

La Quantification Vectorielle

Performances du quantificateur optimal

Lookabaugh et Gray écrivent le gain de QV sous la forme suivante, en remplaçant par leurs approximations asymptotiques respectives les expressions des distorsions scalaire et vectorielle :

$$\begin{aligned}\text{Gain}(N) &= \frac{G(1) \|\tilde{p}_s(s)\|_{1/3}}{G(N) \|p_s(s)\|_{N/(N+2)}} \\ &= \frac{G(1)}{G(N)} \times \frac{\|\tilde{p}_s(s)\|_{1/3}}{\|\bar{p}_s(s)\|_{N/(N+2)}} \times \frac{\|\bar{p}_s(s)\|_{N/(N+2)}}{\|p_s(s)\|_{N/(N+2)}} \\ &= G_1(N) \times G_2(N) \times G_3(N)\end{aligned}$$

$$\text{avec } \|p_s(x)\|_\alpha = \left[\int_{\mathbf{R}^N} p_s(x)^\alpha dx \right]^{1/\alpha}$$

La Quantification Vectorielle

Performances du quantificateur optimal : Gain de partitionnement

$$\text{Gain}(N) = G_1(N) \times G_2(N) \times G_3(N)$$

Le premier rapport $G_1(N)$ caractérise la façon dont le quantificateur couvre l'espace. C'est le **GAIN de PARTITIONNEMENT**. On sait que c'est un avantage de la quantification vectorielle par rapport à la quantification scalaire.

Cependant les gains espérés sont peu significatifs : 0,75 dB pour $N=10$ et 1,5 dB pour $N \rightarrow \infty$.

La Quantification Vectorielle

Performances du quantificateur optimal : Gain de forme

$$\text{Gain}(N) = G_1(N) \times G_2(N) \times G_3(N)$$

Le deuxième rapport $G_2(N)$ ne tient compte que de la forme de la densité marginale. C'est le **GAIN de FORME**.

On a :

$$\begin{aligned}\|\bar{p}_s(s)\|_{N/(N+2)} &= \left[\int_{\mathbb{R}^N} \left[\prod_{k=0}^{N-1} \tilde{p}_s(s) \right]^{N/(N+2)} ds \right]^{(N+2)/N} \\ &= \left[\int_{\mathbb{R}} \left[\tilde{p}_s(s) \right]^{N/(N+2)} ds \right]^{(N+2)}\end{aligned}$$

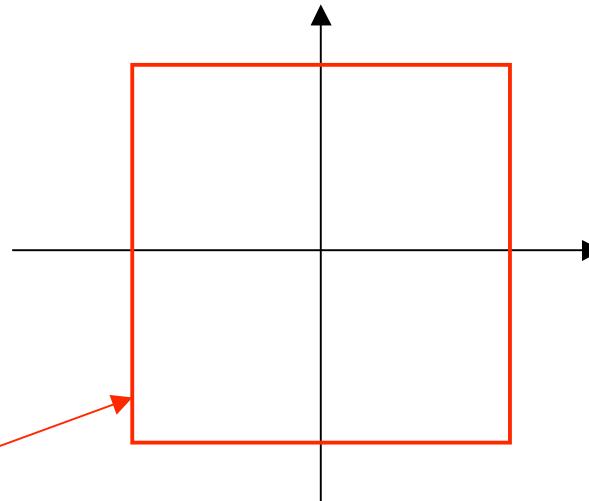
La Quantification Vectorielle

Performances du quantificateur optimal : Gain de forme

$$\text{Gain}(N) = G_1(N) \times G_2(N) \times G_3(N)$$

Un calcul montre que pour une densité de probabilité uniforme, on obtient :

$$G_2(N) = 1 \quad \forall N$$



Distribution de la source
et des représentants dans les dictionnaires scalaires et vectoriels

La Quantification Vectorielle

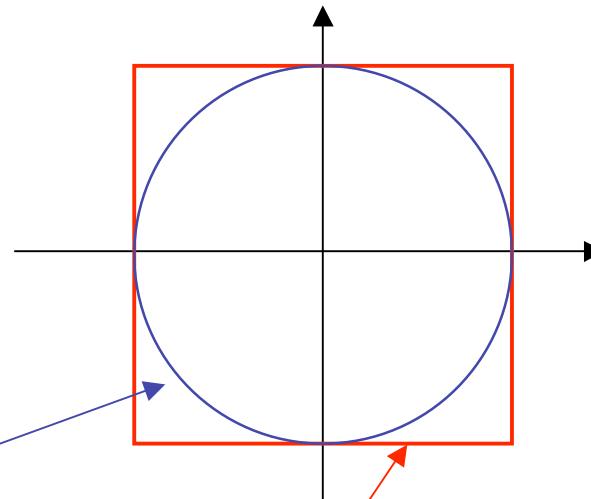
Performances du quantificateur optimal : Gain de forme

$$\text{Gain}(N) = G_1(N) \times G_2(N) \times G_3(N)$$

Un calcul montre que pour une densité de probabilité Gaussienne, on obtient :

$$G_2(N) = \frac{3^{3/2}}{\left(\frac{N+2}{N}\right)^{(N+2)/2}}$$

2,4 dB pour $N=10$ et 2,8 dB pour $N \rightarrow \infty$



Distribution de la source
et des représentants dans le dictionnaire **vectoriel**

Distribution des représentants dans le dictionnaire **scalaire**

La Quantification Vectorielle

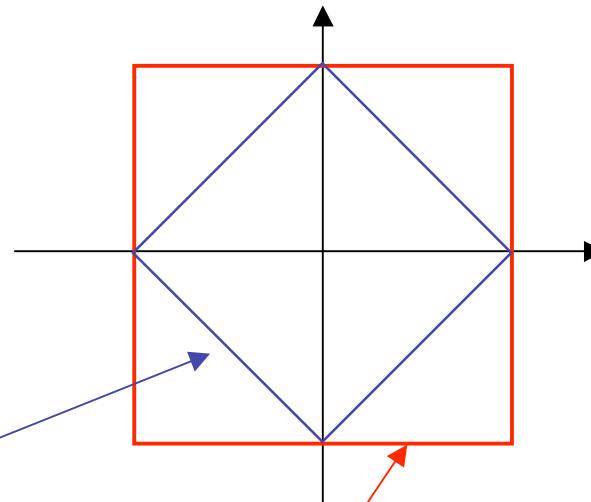
Performances du quantificateur optimal : Gain de forme

$$\text{Gain}(N) = G_1(N) \times G_2(N) \times G_3(N)$$

Un calcul montre que pour une densité de probabilité Laplacienne, on obtient :

$$G_2(N) = \frac{3^3}{\left(\frac{N+2}{N}\right)^{(N+2)}}$$

4,81 dB pour $N=10$ et 5,63 dB pour $N \rightarrow \infty$



Distribution de la source
et des représentants dans le dictionnaire **vectoriel**

Distribution des représentants dans le dictionnaire **scalaire**

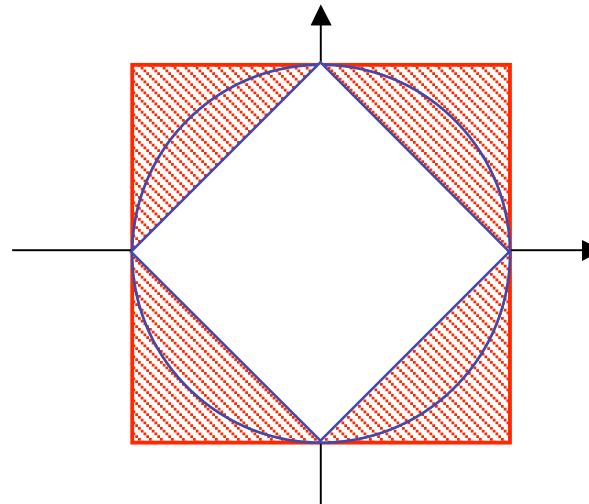
La Quantification Vectorielle

Performances du quantificateur optimal : Gain de forme

$$\text{Gain}(N) = G_1(N) \times G_2(N) \times G_3(N)$$

On peut constater que le gain $G_2(N)$ est élevé pour une source de densité de probabilité Laplacienne et ce d'autant plus que la taille N des vecteurs est élevée.

La QV est donc bien adapté aux coefficients d'ondelettes généralement modélisés par des lois Gaussiennes généralisées.

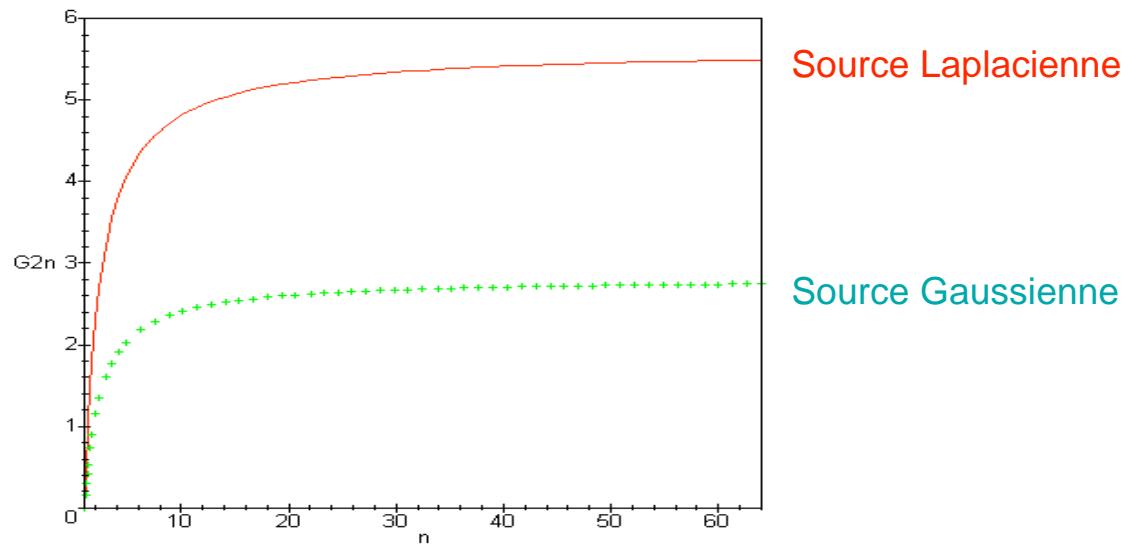


La Quantification Vectorielle

Performances du quantificateur optimal : Gain de forme

$$\text{Gain}(N) = G_1(N) \times G_2(N) \times G_3(N)$$

Gain de forme exprimé en dB en fonction de la dimension des vecteurs (cas d'une source i.i.d.).



La Quantification Vectorielle

Performances du quantificateur optimal : Gain de mémoire

$$\text{Gain}(N) = G_1(N) \times G_2(N) \times G_3(N)$$

Le troisième terme $G_3(N)$ est le plus intéressant. C'est le **GAIN de MEMOIRE**. Il tient compte de la corrélation existant entre les différentes composantes du vecteur.

$$G_3(N) = \frac{\|\bar{p}_S(s)\|_{N/(N+2)}}{\|p_S(s)\|_{N/(N+2)}}$$

La densité de probabilité $\bar{p}_S(s)$ est celle d'un vecteur dont les composantes sont complètement décorrélées.

La Quantification Vectorielle

Performances du quantificateur optimal : Gain de mémoire

Supposons que le vecteur aléatoire $S(m)$ de dimension N est de distribution Gaussienne et statistiquement complètement défini par sa densité de probabilité conjointe donnée par :

$$p_S(s) = \frac{1}{(2\pi\sigma_x^2)^{N/2} \sqrt{\det(\Gamma_s)}} e^{-\left(s^t \Gamma_s^{-1} s / 2\sigma_s^2\right)}$$

avec

$$\Gamma_s = \Gamma_s(N) = \frac{1}{\sigma_s^2} E[S(m)S^t(m)]$$

et

$$\Gamma_s = \begin{bmatrix} 1 & \rho_1 & \dots & \rho_{N-1} \\ \rho_1 & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \rho_1 \\ \rho_{N-1} & \dots & \rho_1 & 1 \end{bmatrix}$$

La Quantification Vectorielle

Performances du quantificateur optimal : Gain de mémoire

Si l'on développe les formules précédentes on obtient alors :

$$\|p_s(s)\|_{N/(N+2)} = 2\pi \left(\frac{N+2}{N} \right)^{N/(N+2)} \sigma_s^2 (\det(\Gamma_s))^{1/N}$$

et

$$\|\bar{p}_s(s)\|_{N/(N+2)} = 2\pi \left(\frac{N+2}{N} \right)^{(N+2)/2} \sigma_s^2$$

D'où

$$G_3(N) = \frac{1}{\left(\det \begin{bmatrix} 1 & \rho_1 & \dots & \rho_{N-1} \\ \rho_1 & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \rho_1 \\ \rho_{N-1} & \dots & \rho_1 & 1 \end{bmatrix} \right)^{\frac{1}{N}}} = \frac{1}{(\det \Gamma_s(N))^{\frac{1}{N}}}$$

La Quantification Vectorielle

Conclusion

- Le rapport $G_3(N)$ est toujours supérieur ou égal à 1 et il est d'autant plus important que les composantes du vecteur sont corrélées.
- Réaliser une quantification vectorielle est donc toujours préférable à une quantification scalaire.
- Notons que le gain de quantification vectorielle se limite au produit $G_1(N)G_2(N)$ lorsque les composantes de $S(m)$ sont indépendantes, c'est-à-dire lorsque la source est sans mémoire.
- On remarque que ces gains sont automatiquement pris en charge par l'algorithme de Lloyd-Max si l'on admet que le minimum global est atteint



La Quantification Vectorielle Structurée

La Quantification Vectorielle Structurée

Principe généraux - Formalisation

La QV STRUCTUREE ou ALGÉBRIQUE ("lattice VQ" en Anglais) s'affranchit de toute séquence d'apprentissage.

Le principe de la QVA repose sur la définition d'un réseau régulier de points qui est défini de la façon suivante :

$$\Lambda = \left\{ y \in \mathbf{R}^m \mid \exists (u_1, u_2, \dots, u_N) \in \mathbf{Z}^N, y = \sum_{k=1}^N u_k a_k \right\}$$

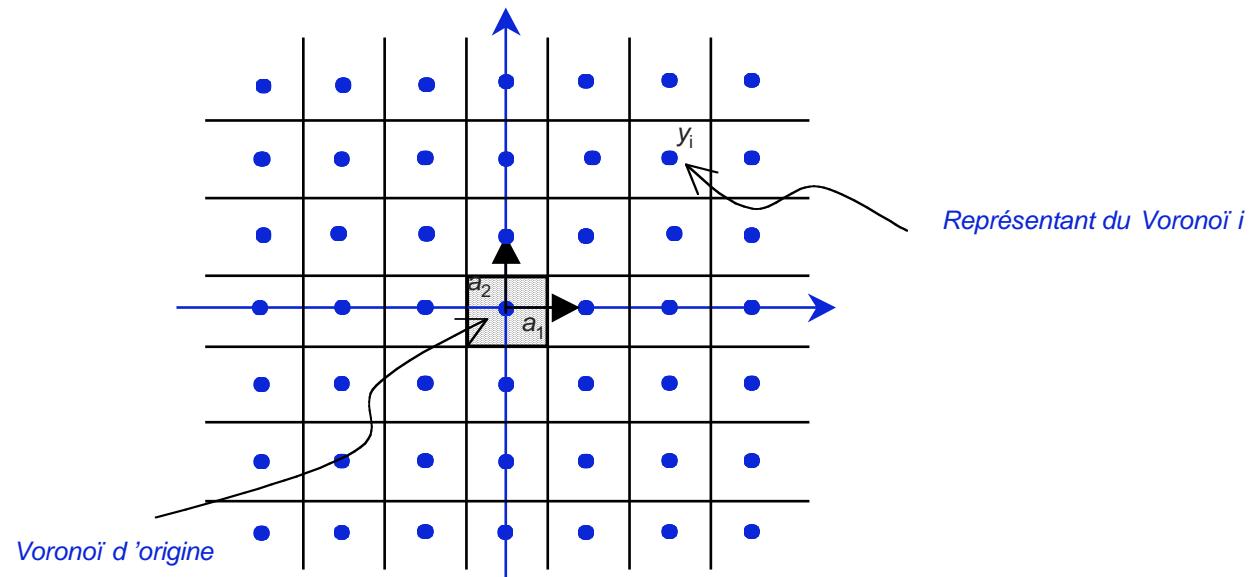
où les a_k sont des vecteurs linéairement indépendants qui forment une base du réseau, et $m \geq N$.

Pour une mesure de distorsion et une dimension de vecteurs données, il existe un réseau optimal conduisant à une erreur de quantification minimale à l'intérieur du Voronoï.

La Quantification Vectorielle Structurée

Principe généraux - Formalisation

Un exemple de réseau régulier de points : le réseau des couples d'entiers relatifs \mathbb{Z}^2 :



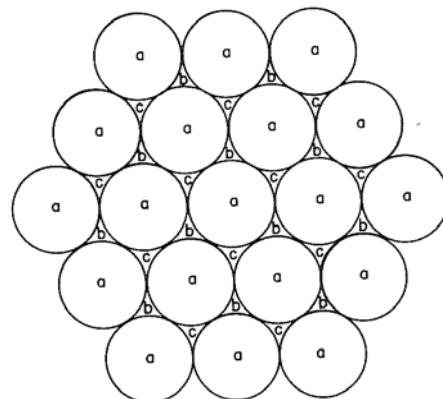
Du fait qu'il engendre un partitionnement régulier de l'espace, le quantificateur vectoriel sur réseau est adapté à la quantification de sources uniformes.

La Quantification Vectorielle Structurée

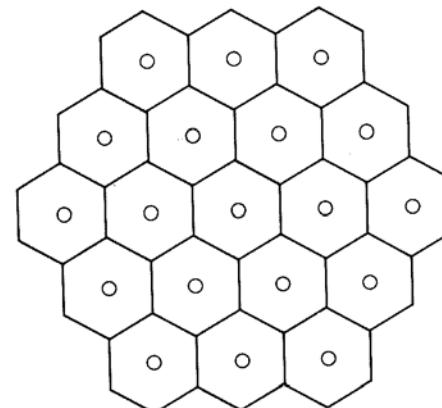
Principe généraux - Formalisation

Les réseaux réguliers conduisent à deux problèmes fondamentaux qui sont l'empilement des sphères dans l'espace, et le recouvrement de l'espace par des sphères : il s'agit en effet de rendre le réseau le plus dense possible (empilement) tout en minimisant l'erreur quadratique moyenne à l'intérieur des Voronoï (recouvrement).

En dimension 2 :



Nid d'abeilles
(réseau hexagonal optimal en dimension 2)



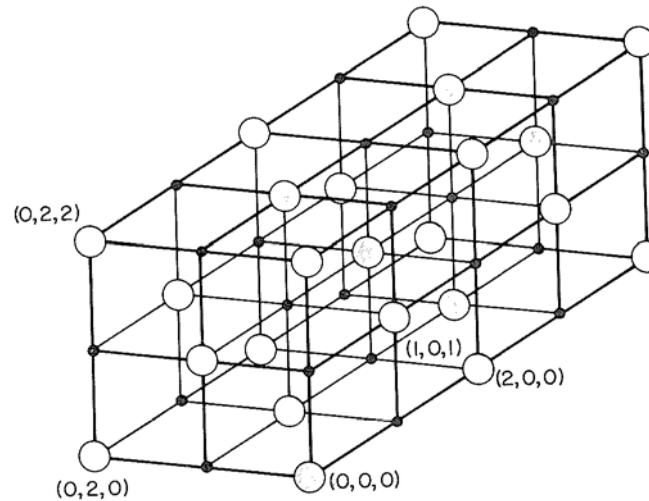
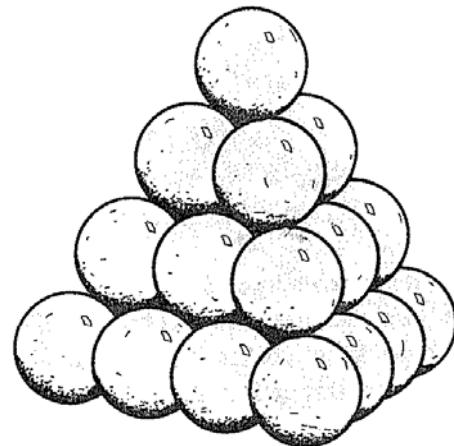
Partition de Voronoï

La Quantification Vectorielle Structurée

Principe généraux - Formalisation

Les réseaux réguliers conduisent à deux problèmes fondamentaux qui sont l'empilement des sphères dans l'espace, et le recouvrement de l'espace par des sphères : il s'agit en effet de rendre le réseau le plus dense possible (empilement) tout en minimisant l'erreur quadratique moyenne à l'intérieur des Voronoï (recouvrement).

En dimension 3 :



La Quantification Vectorielle Structurée

Exemples de réseaux réguliers de points

Conway et Sloane ont défini des réseaux optimaux au sens de la norme quadratique. Ces réseaux conduisent à des formes de Voronoï optimales au sens de l'erreur quadratique.

- Le réseau \mathbf{Z}^N est le plus simple mais il n'est malheureusement pas optimal. Il est constitué de points de \mathbf{R}^N dont les coordonnées sont entières :
$$\mathbf{Z}^N = \left\{ y = (y_1, y_2, \dots, y_N) \mid y_k \in \mathbf{Z} \right\}$$
- Le réseau D_N ($N \geq 2$) correspond aux points du réseau \mathbf{Z}^N dont la somme des coordonnées est paire :

$$D_N = \left\{ y = (y_1, y_2, \dots, y_N) \in \mathbf{Z}^N \mid \sum_{k=1}^N y_k = 0 \pmod{2} \right\}$$

- Il existe aussi des réseaux plus complexes comme le E_8 (réseau de Gosset), le Λ_{16} (réseau de Barnes-Wall) et le Λ_{24} (réseau de Leech)

La Quantification Vectorielle Structurée

Quantification sur réseaux réguliers de points

La quantification sur les différents réseaux réguliers de points est effectuée à l'aide d'algorithmes rapides définis par Conway et Sloane.

Tout l'intérêt de la QVA réside dans la rapidité des algorithmes de quantification qui autorise ainsi l'utilisation de dictionnaires de taille très élevée (plusieurs millions de vecteurs). C'est ce qui la différencie de la quantification vectorielle avec recherche exhaustive. En effet, du fait de la structure régulière du dictionnaire, la recherche du meilleur représentant n'est plus effectuée de façon exhaustive mais de façon analytique. Les vecteurs du dictionnaire vérifient tous la même relation mathématique ce qui rend la quantification beaucoup plus rapide.

La Quantification Vectorielle Structurée

Algorithmes de Quantification sur réseaux réguliers de points

Notations :

- Φ_{Λ} la fonction de quantification sur un réseau quelconque,
- f la fonction qui associe à un nombre réel x l'entier le plus proche,
- $w(x)$ le second entier le plus proche de x :

$$w(x) = f(x) + sign(x - f(x))$$

- g , la fonction telle que :

$$g(x) = (f(x_1), \dots, w(x_j), \dots, f(x_N))$$

où x_j est la composante du vecteur x sur laquelle l'erreur de quantification engendrée par f est la plus grande.

La Quantification Vectorielle Structurée

Exemples d'algorithmes de quantification

Quantification dans \mathbf{Z}^N

$$\Phi_{\mathbf{Z}^N}(x) = f(x)$$

Quantification dans D_N

*calculer $f(x)$
si $\sum_{k=1}^N f(x_k)$ est paire alors $\Phi_{D_N}(x) = f(x)$
sinon
 calculer $g(x)$
 $\Phi_{D_N}(x) = g(x)$
finsi*

La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

La quantification vectorielle sur réseaux réguliers de points (ou quantification vectorielle algébrique) présente de nombreux avantages mais sa mise en œuvre n'est pas pour autant immédiate.

Cette méthode soulève en effet un certain nombre de problèmes qu'il faut impérativement résoudre dans une application de compression.

Le codage par QVA se définit ainsi en cinq étapes :

- 1- Définition d'une [troncature](#)
- 2- [Normalisation](#) de la source
- 3- [Quantification](#) (à l'aide d'algorithmes rapides)
- 4- [Indexage](#) des vecteurs quantifiés
- 5- [Codage](#) des index

La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

1- Définition d'une troncature :

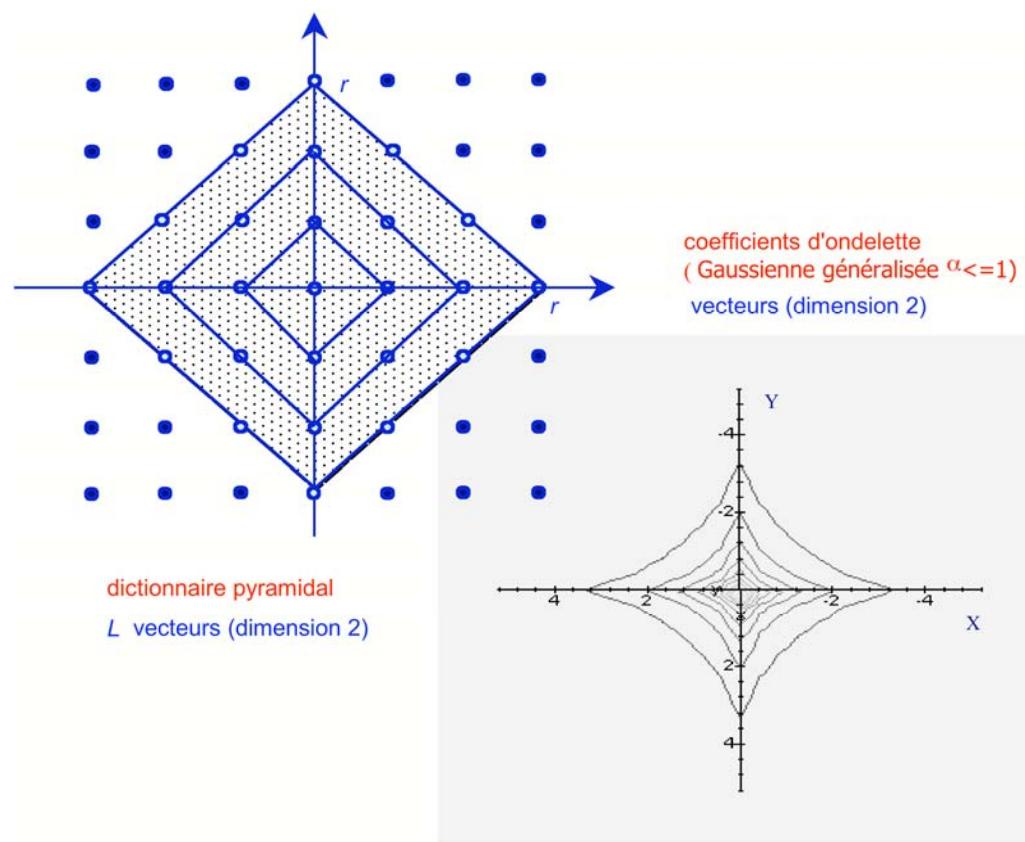
Le choix d'une troncature du réseau est une étape incontournable dans une application de compression. Elle permet de garantir un débit binaire fini, au prix malheureusement d'un bruit de surcharge, lié justement au caractère fini du dictionnaire.

Il est cependant possible de minimiser ce bruit, en choisissant une forme de troncature adaptée à la distribution spatiale de la source à quantifier. Notons que c'est la supériorité de la quantification vectorielle sur la quantification scalaire – à travers le gain de forme – qui permet de le faire.

La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

1- Définition d'une troncature :

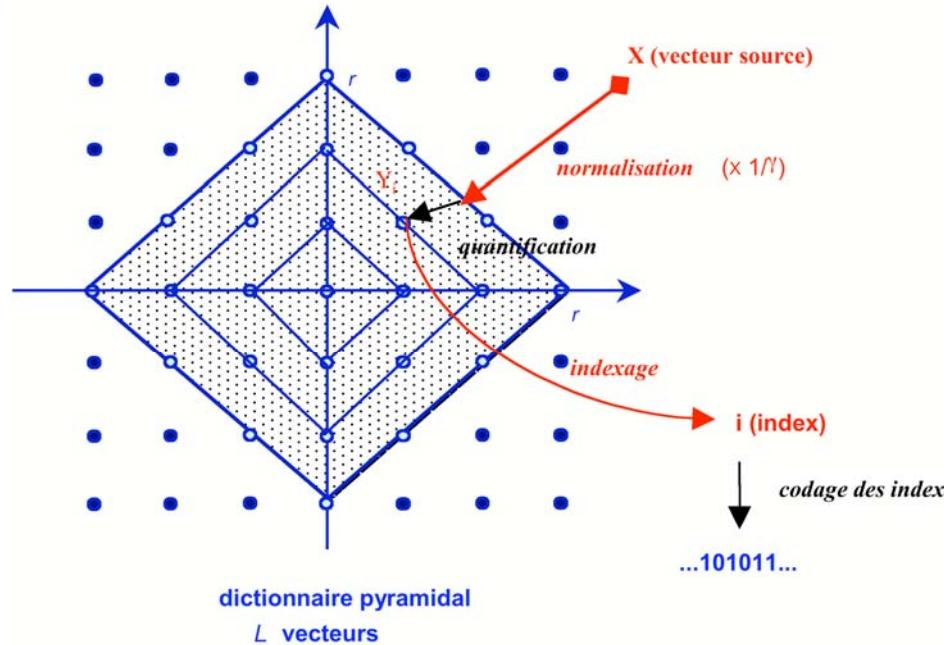


La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

2- Normalisation de la source :

Une fois la troncature fixée, il s'agit de quantifier tous les vecteurs de la source par ceux contenus dans le dictionnaire. Ceci nécessite une procédure de projection ramenant tous les vecteurs source à l'intérieur du dictionnaire.



La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

3- Quantification (à l'aide d'algorithmes rapides) :

La faible complexité de quantification est l'atout majeur des quantificateurs vectoriels algébriques. Celle-ci est directement liée au réseau choisi

La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

4- Indexage des vecteurs quantifiés :

L'indexage est une opération fondamentale de la chaîne de compression. Elle doit permettre de repérer un vecteur du dictionnaire de façon unique à partir de son index. Cette opération associée au codage binaire des index précède la transmission.

- Dans le cas d'une Quantification Vectorielle par apprentissage (type LBG), l'indexage est effectué de manière très simple : les vecteurs du dictionnaire sont numérotés de 0 à $L-1$. Le décodeur disposant d'une copie du dictionnaire, le décodage des vecteurs à partir de leurs index est immédiat.
- Dans le cas de la QVA, le problème est plus délicat. La taille des dictionnaires mis en jeu étant beaucoup plus importante, il est impossible d'indexer les vecteurs comme dans la méthode LBG sans augmenter considérablement le débit binaire.

La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

4- Indexage des vecteurs quantifiés :

Du fait du partitionnement régulier de l'espace, les vecteurs du réseau se répartissent sur une infinité dénombrable de surfaces concentriques de rayon constant par rapport à l'origine. Ces surfaces contiennent tous les vecteurs ayant même norme ; on les appelle souvent iso surfaces pour faire le lien avec la statistique de la source à quantifier.

Il est donc possible d'identifier un vecteur par :

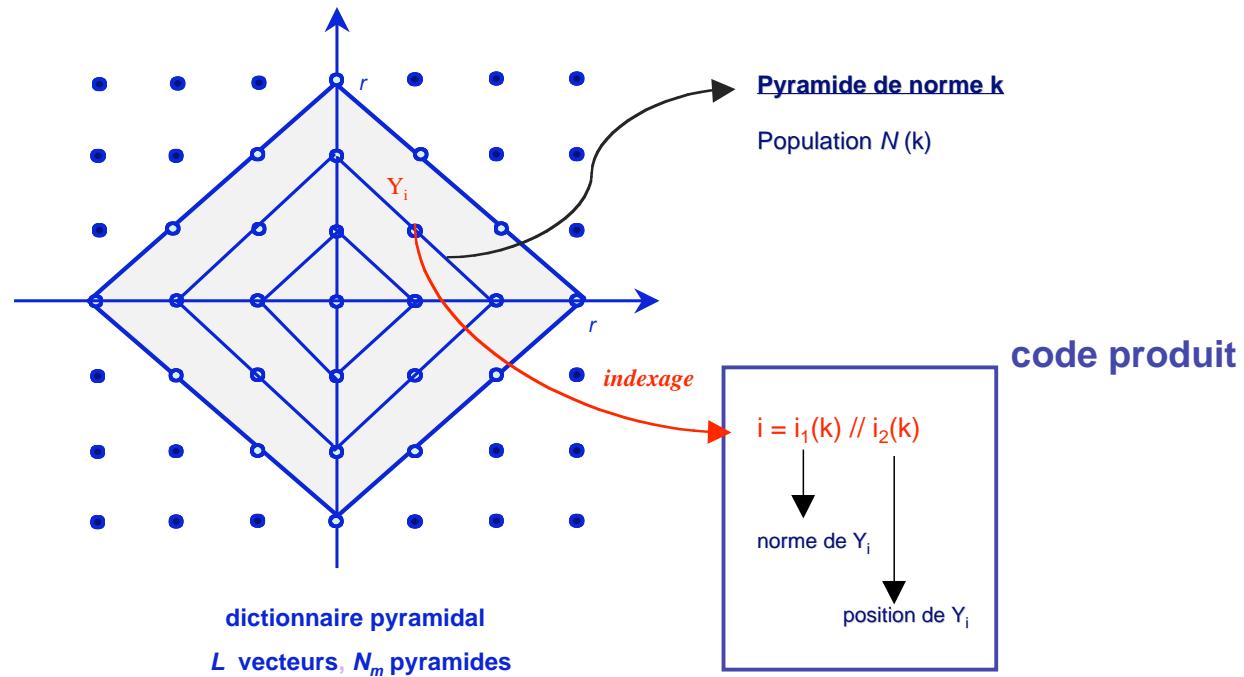
- sa **norme** (ou le rayon de la surface à laquelle il appartient)
- et par sa « **position** » sur la surface en question.

Ce raisonnement s'apparente à une représentation polaire. La position ne traduit cependant pas ici l'angle (ou phase) du vecteur mais un index lié à la population de la surface (nombre de vecteurs de même norme).

La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

4- Indexage des vecteurs quantifiés : Le code produit



La Quantification Vectorielle Structurée

Mise en œuvre d'un quantificateur vectoriel algébrique

5- Codage des index

Le codage des index est la dernière étape avant la transmission (ou le stockage) de l'image compressée. Il est évidemment fortement lié au type d'indexage effectué.

Dans le cas d'un code produit, nous sommes conduits à coder les index de **norme** et de **position** séparément. Le débit binaire peut être évalué par surface de rayon k constant par la relation :

$$R(k) = \underbrace{-\log_2 p_k}_{\text{variable}} + \underbrace{\lceil \log_2 N(k) \rceil}_{\text{variable}} \quad \text{bits/vecteur}$$

soit

$$R = \frac{1}{N} \sum_{k=0}^r p_k R(k) \quad \text{bits/pixel}$$

$N(k)$ étant le nombre de vecteurs dans le réseau de norme k^α

La Quantification Vectorielle Structurée

Remarques

Du fait que les points du dictionnaire sont contraints à être les points d'un réseau régulier, c'est-à-dire répartis selon une certaine géométrie dans l'espace, les performances de la QV algébrique sont en général inférieures à celles de la QV non structurée (type LBG).

Cet inconvénient est compensé par le fait que la QV algébrique n'a pas besoin de générer ni de stocker un dictionnaire (elle n'est donc pas tributaire d'une séquence d'apprentissage liée à la source) et possède de surcroît des algorithmes rapides de quantification en fonction du réseau considéré.

De plus, la QV algébrique est stable contrairement à la QV non structurée car le dictionnaire ne dépend pas d'une séquence d'apprentissage.



Quelques variantes en Quantification Vectorielle

Variantes en Quantification Vectorielle

La QV à entropie contrainte

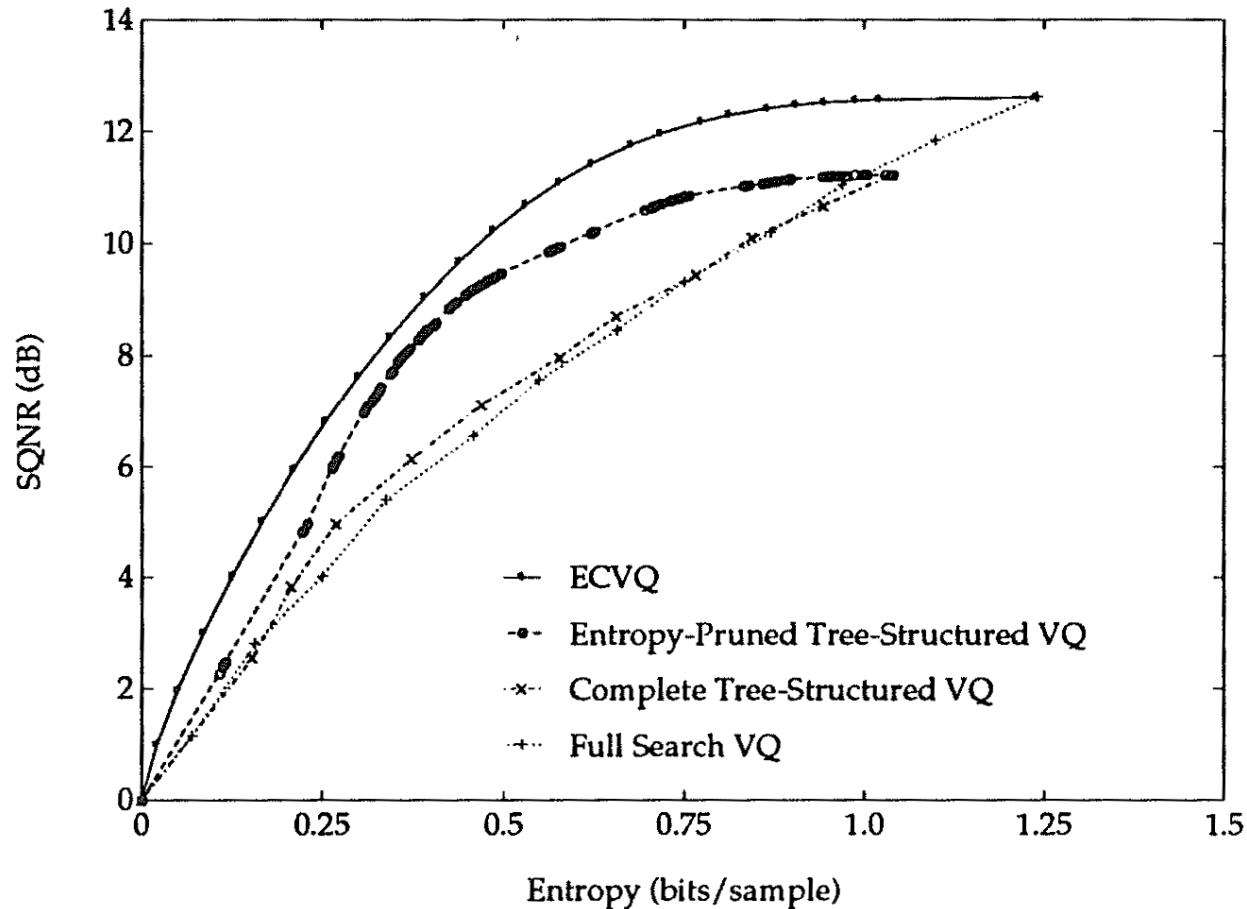
La QV s'efforce de minimiser la distorsion moyenne de quantification sous une contrainte liée au nombre de vecteurs contenus par le dictionnaire. Ceci est équivalent à minimiser la distorsion moyenne pour un débit donné, quand le débit est mesuré par le \log_2 du nombre de ces vecteurs. Ceci est une formulation d'optimisation correcte quand le système de compression est à débit fixe. Cependant, si un codage entropique des vecteurs du dictionnaire est effectué, il devient plus intéressant de construire un quantificateur vectoriel qui minimise la distorsion moyenne sous contrainte de l'entropie des vecteurs du dictionnaire plutôt que sur le \log_2 de leur nombre. Mathématiquement, cela revient à modifier la mesure de distorsion en introduisant l'entropie du système multipliée par un opérateur de Lagrange :

$$J_\lambda(\{\hat{s}\}) = \mathbb{E}[d(s, \hat{s})] + \lambda \underbrace{\mathbb{E}[l(\hat{s})]}_{\text{Entropie du dictionnaire}}$$

Longueurs des mots de code

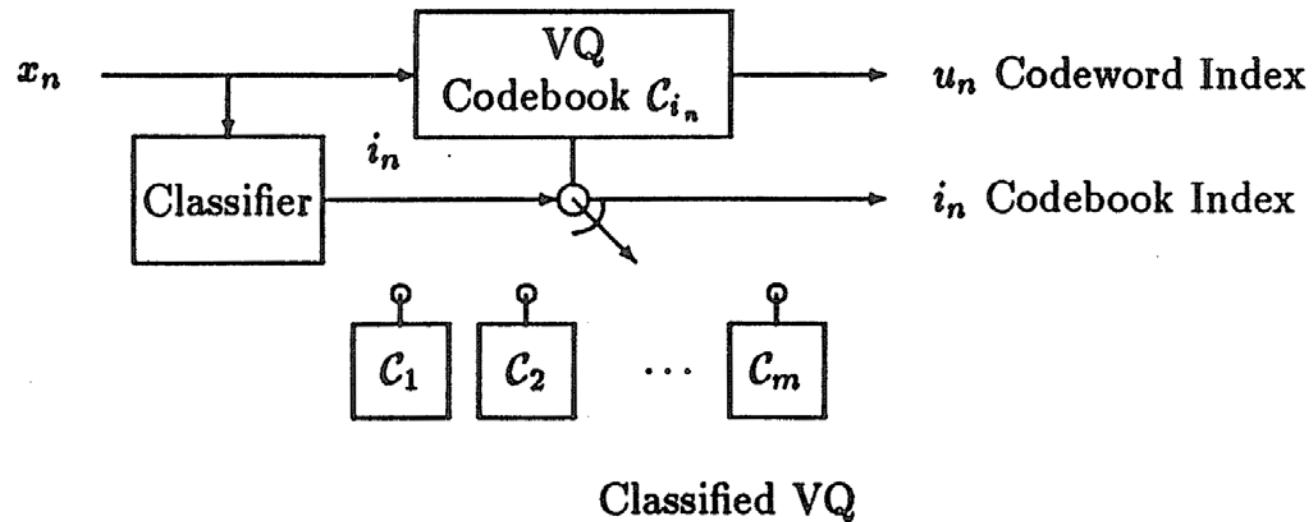
Variantes en Quantification Vectorielle

La QV à entropie contrainte



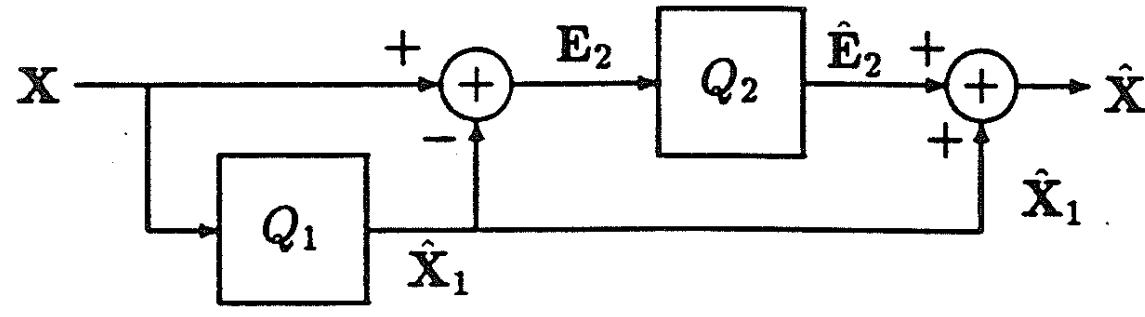
Variantes en Quantification Vectorielle

La QV par classification



Variantes en Quantification Vectorielle

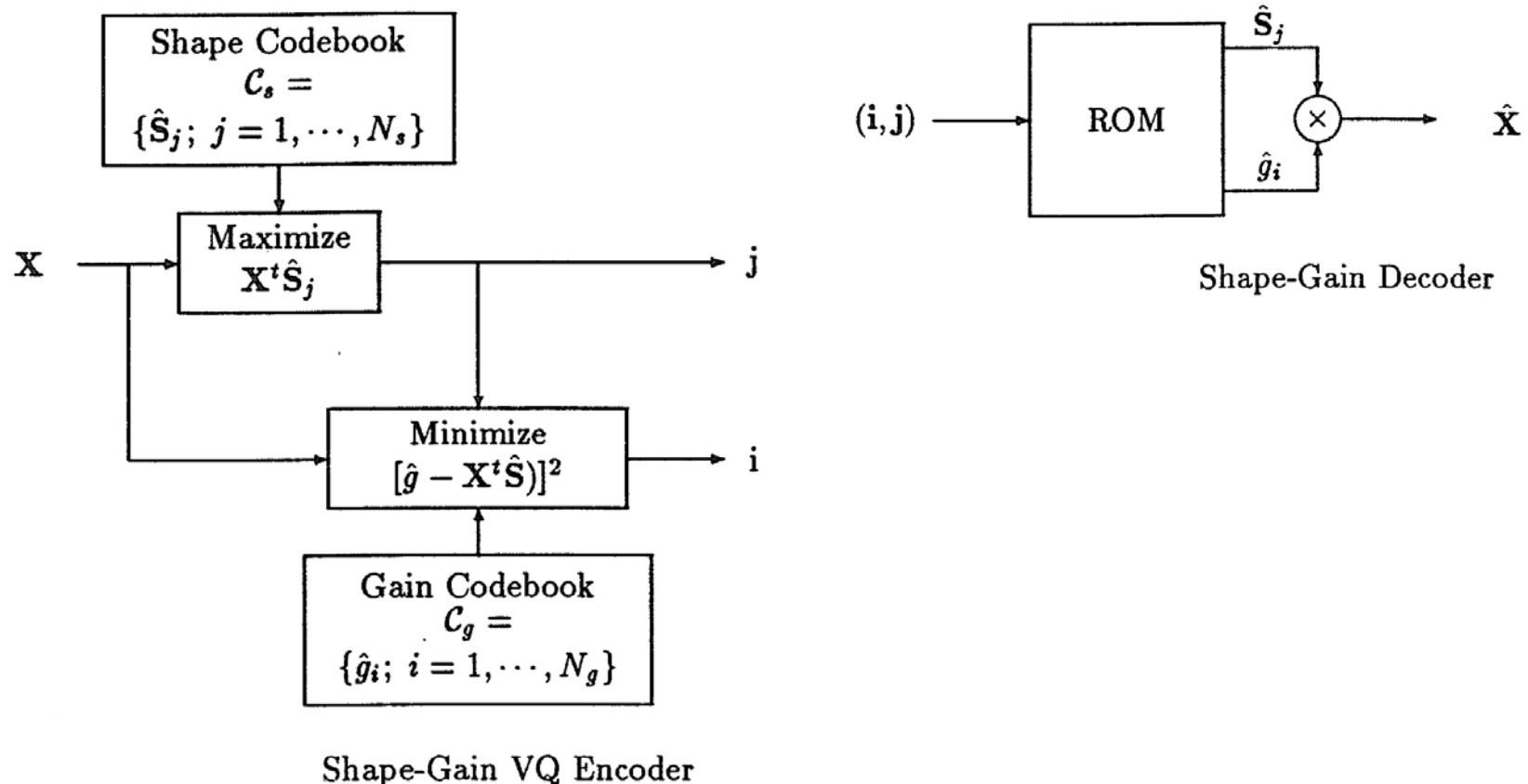
La QV multi-étages



Two-Stage VQ

Variantes en Quantification Vectorielle

La QV Gain/Forme





Le codage par transformée et le codage en sous-bandes

Le codage par transformée

Principe

1. On décompose d'abord le signal en vecteurs 1D ou en blocs 2D (images)

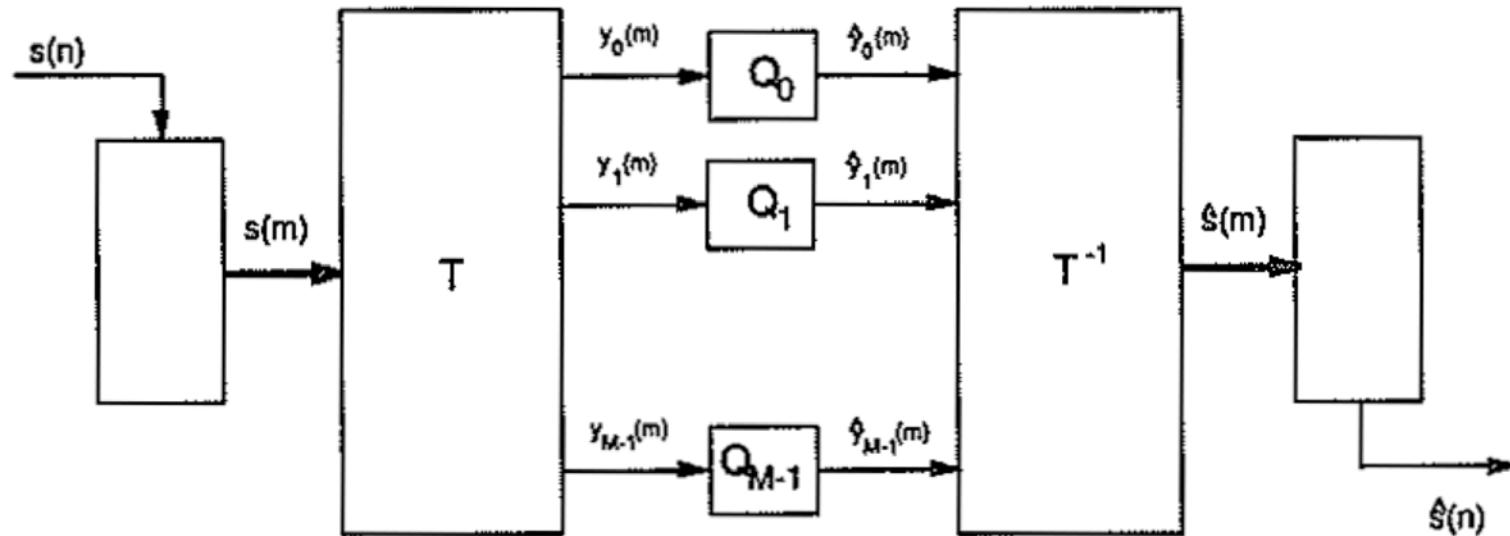
- On applique ensuite une transformée caractérisée par une matrice généralement orthogonale ou unitaire (transformée de Fourier ou en cosinus discrète – DCT - par exemple).
- On code les coefficients dans le domaine transformé.

2. Au récepteur, on réalise les transformations inverses

- En l'absence de quantification, une transformation orthogonale ou unitaire entraîne une reconstruction parfaite.

Le codage par transformée

Schéma d'un codeur par transformée



Le codage par transformée

Formalisation (1/4)

On suppose que le signal que l'on cherche à quantifier est un processus aléatoire $S(m)$ stationnaire, centré et de puissance σ_s^2 .

A partir de ce processus, on construit un vecteur aléatoire de dimension N

$$s(m) = \begin{bmatrix} s_0(m) \\ \vdots \\ s_{N-1}(m) \end{bmatrix} = \begin{bmatrix} s(mN) \\ \vdots \\ s(mN + N - 1) \end{bmatrix}$$

Chaque composante de vecteur a la même puissance σ_s^2 puisque le processus aléatoire est stationnaire.

Le codage par transformée

Formalisation (2/4)

On applique sur ce vecteur une transformation caractérisée par la matrice T que l'on supposera, pour simplifier, carrée. On obtient à chaque instant mN , un vecteur

$$Y(m) = \begin{bmatrix} Y_0(m) \\ \vdots \\ Y_{N-1}(m) \end{bmatrix} = T \begin{bmatrix} S_0(m) \\ \vdots \\ S_{N-1}(m) \end{bmatrix}$$

Chaque composante $Y_k(m)$ permet de construire un processus aléatoire stationnaire de puissance $\sigma_{Y_k}^2$ mais ces N puissances (variances puisque le processus est centré) sont maintenant différentes.

Le codage par transformée

Formalisation (3/4)

La propriété essentielle des transformations orthogonales est de conserver la norme des vecteurs et donc la puissance :

$$\sigma_s^2 = \frac{1}{N} \sum_{k=0}^{N-1} \sigma_{Y_k}^2$$

Soit $\hat{Y}(m)$ le vecteur reconstruit à partir des composantes quantifiées de $Y(m)$, le vecteur obtenu par transformation inverse s'écrit alors :

$$\hat{S}(m) = T^{-1}\hat{Y}(m)$$

Note : On se limite aux transformations orthogonales (ou unitaire) : ex. la transformée de Fourier ou la transformée en cosinus discrète largement utilisées en pratique

Le codage par transformée

Formalisation (4/4)

Si l'on prend la distance Euclidienne comme mesure de distorsion, la puissance de l'erreur de quantification correspondant à l'ensemble des composantes du vecteur $Y(m)$

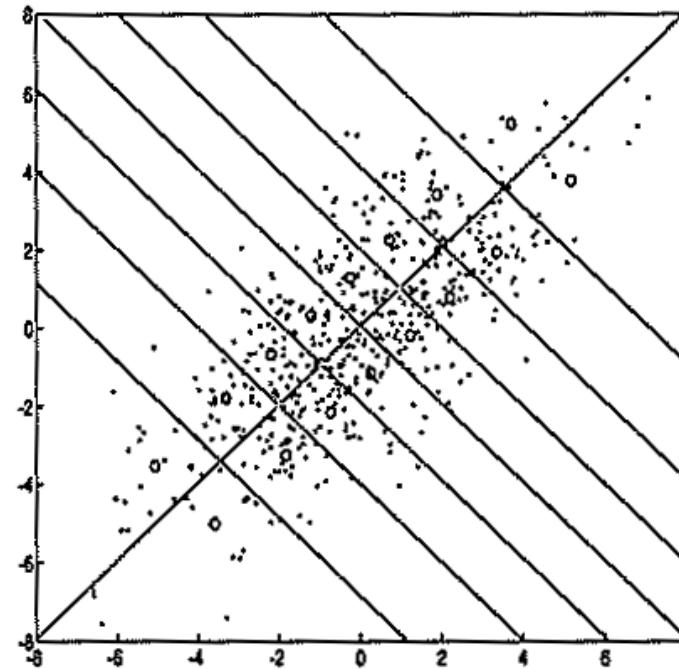
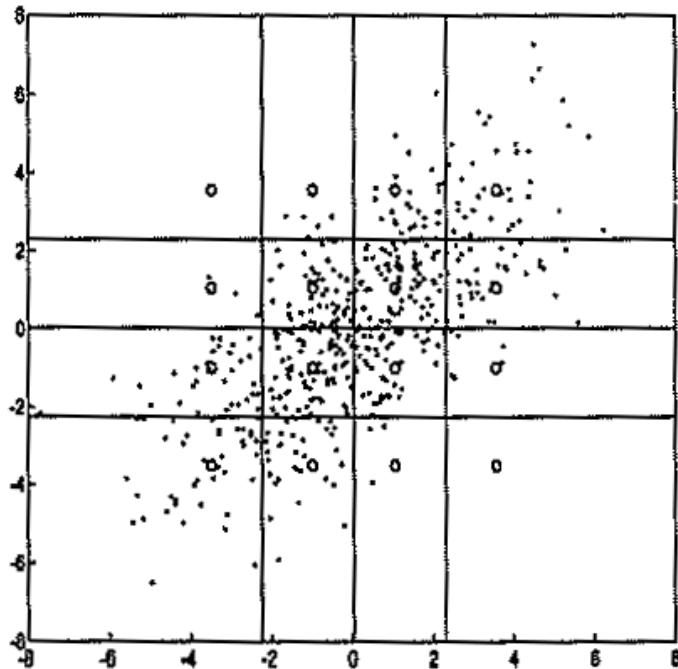
$$\sigma_Q^2 = \frac{1}{N} E \left[\|Y(m) - \hat{Y}(m)\|^2 \right]$$

est égale à la moyenne arithmétique des puissances des erreurs de quantification dans les différentes sous-bandes :

$$\sigma_Q^2 = \frac{1}{N} \sum_{k=0}^{N-1} \sigma_{Q_k}^2$$

Le codage par transformée

Illustration



Visualisation dans le plan d'un processus aléatoire stationnaire : deux façons de quantifier scalairement ce processus

Le codage par transformée

Interprétation

On applique une transformation qui s'interprète comme un changement d'axe. La projection de l'ellipse sur les anciens axes caractérise la puissance σ_S^2 de $S(m)$. La projection de cette ellipse sur les nouveaux axes caractérise les puissances $\sigma_{Y_0}^2$ et $\sigma_{Y_1}^2$ de $Y_0(m)$ et $Y_1(m)$.

Cette figure montre deux façons de quantifier scalairement le processus : dans le premier mode de quantification, chaque échantillon est quantifié de façon identique sur 2 bits. Dans le deuxième mode, la première composante du vecteur transformé est quantifiée sur 3 bits et la seconde composante sur 1 bit. Cette solution est manifestement la meilleure



**Il existe une façon optimale
d'allouer les ressources binaires disponibles.**

Le codage par transformée

Allocation des ressources binaires (1/5)

- Supposons que l'on ait réalisé une transformation quelconque $Y(m) = TS(m)$ sur N échantillons et que l'on dispose de RN bits pour quantifier les N valeurs transformées $Y_0(m) \dots Y_{N-1}(m)$ de puissance $\sigma_{Y_0}^2 \dots \sigma_{Y_{N-1}}^2$.
- R est le débit, c'est-à-dire le nombre de bits disponible par échantillon (avec RN entier).
- Supposons que l'on connaisse pour chaque composante $Y_k(m)$ l'expression de l'erreur quadratique moyenne $\sigma_{Q_k}^2$ apportée par la quantification scalaire optimale sur R_k bits.

Le codage par transformée

Allocation des ressources binaires (2/5)

Le problème consiste à trouver les débits $R_0 \dots R_{N-1}$ de façon à minimiser la puissance

$$\sigma_Q^2 = \frac{1}{N} \sum_{k=0}^{N-1} \sigma_{Q_k}^2(R_k)$$

sous la contrainte :

$$\sum_{k=0}^{N-1} R_k \leq RN$$

Le codage par transformée

Allocation des ressources binaires (3/5)

Les expressions $\sigma_{Q_k}^2$ sont difficilement connues dans la pratique. On suppose ici que l'on connaît toutes les propriétés statistiques du processus aléatoire, que la mesure de distorsion choisie est l'erreur quadratique et que la propriété dite haute résolution est vérifiée.

On peut alors écrire explicitement

$$\sigma_{Q_k}^2(R_k) = c_k(1) \sigma_{Y_k}^2 2^{-2R_k}$$

où $c_k(1)$ est une constante qui ne dépend que de la densité de probabilité marginale de $Y_k(m)$. On a vu que, pour une variable aléatoire uniformément répartie cette constante est égale à 1, pour une Gaussienne elle est égale à $\sqrt{3}\pi/2$.

Le codage par transformée

Allocation des ressources binaires (4/5)

L'allocation optimale de bits minimisant

$$\sigma_Q^2 = \frac{c(1)}{N} \sum_{k=0}^{N-1} \sigma_{Y_k}^2 2^{-2b_k}$$

(ici on suppose que toutes les variables aléatoires ont la même distribution ce qui implique que les $c_k(1)$ sont égaux)

sous la contrainte précédente :

$$\sum_{k=0}^{N-1} R_k \leq RN$$

est donné par :

$$R_k = R + \frac{1}{2} \log_2 \frac{\sigma_{Y_k}^2}{\alpha^2} \quad \text{avec} \quad \alpha^2 = \left(\prod_{k=0}^{N-1} \sigma_{Y_k}^2 \right)^{1/N}$$

Le codage par transformée

Allocation des ressources binaires (5/5)

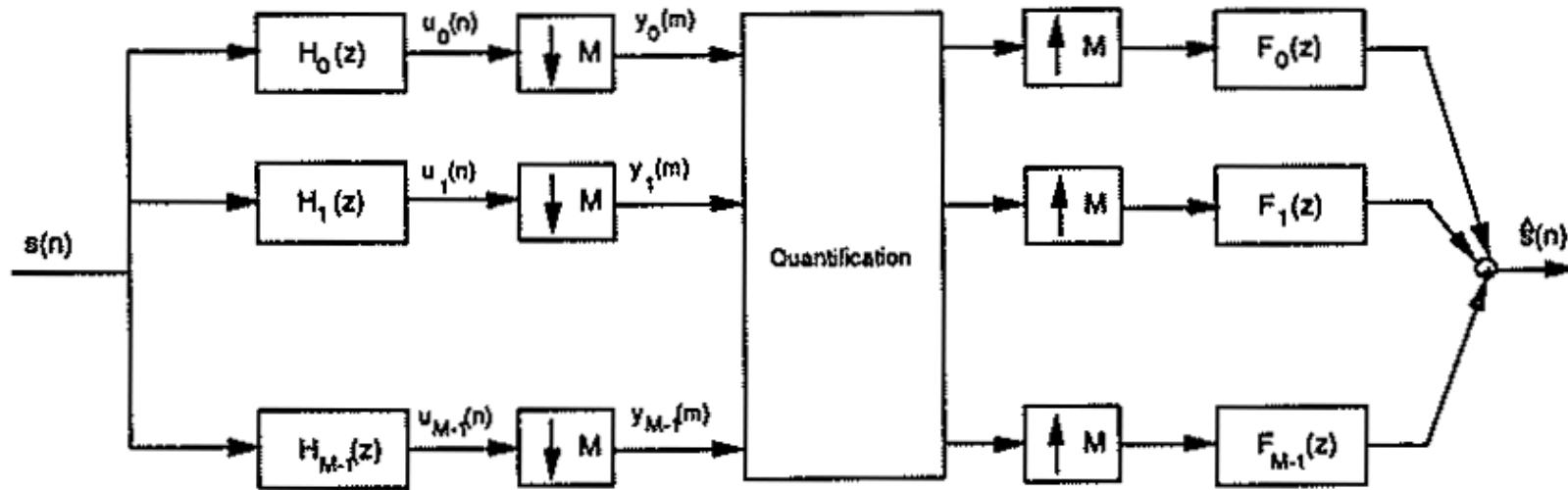
Remarque :

L'équation précédente n'est pas exploitable directement.

En effet, le nombre de bits obtenu n'est pas forcément un nombre entier, il n'est même pas assuré d'être positif => Il existe des algorithmes itératifs d'allocation de bits répartissant progressivement les bits là où ils ont le plus d'effet.

Le codage en sous-bandes

Schéma d'un codeur en sous-bandes



Le codage en sous-bandes

Contraintes

- De façon générale, on désire que les signaux $Y_k(m)$, dans les différentes sous-bandes après sous-échantillonnage critique, soient une représentation spectrale la plus fidèle possible du signal $S(m)$.
- On souhaite aussi que le traitement puis la quantification de cette information entraîne une distorsion perçue la plus faible possible compte tenu du nombre de bits disponibles.
- Il faut également que la reconstruction du signal n'entraîne pas de distorsion supplémentaire, donc qu'elle se fasse de façon (presque) parfaite en l'absence de quantification et enfin que le délai de reconstruction soit faible.

Références bibliographiques

- **Vector quantization and signal compression**

A. Gersho, R.M. Gray, KLUWER ACADEMIC PUBLISHER, 1992

- **Advances in image communication**

« Wavelets in image communication »

Ed. M. Barlaud, vol. 5, Elsevier, 1994

- **CNET-ENST Collection Technique et Scientifique des Télécommunications**

« Techniques de compression des signaux »

N. Moreau, MASSON, 1995

- **Traité IC2 (Information-Commande-Communication)**

« Compression et codage des images et des vidéos »

Ed. M. Barlaud et C. Labit, HERMES, Paris, 2002

- **Encyclopédie sur les systèmes d'information**

Vuibert, Paris, 2005