S1916 - Analyse et Compression du Signal Audionumérique - Examen

Sébastien Boisgérault, Mines ParisTech, CC BY-NC-SA 4.0

14 mars 2019

Contents

Modalités

- **Durée:** 1h30.
- **Autorisés:**
  - tous documents (sous forme papier ou électronique),
  - calculette, tablette, ordinateur portable, etc.
- **Interdit:**
  - toute forme de communication: échanges avec le voisin, utilisation d'Internet, du téléphone, etc.

Questions

Compression sans perte

Une source X produit un flux de caractères ASCII, appartenant à l'ensemble

$$\mathcal{A} = \{\texttt{A}, \texttt{E}, \texttt{U}, \texttt{C}, \texttt{H}, \texttt{N}, \texttt{-}\}$$

Vous souhaitez concevoir un code binaire – de longueur fixe ou variable – pour enregistrer ce flux d'information sous la forme la plus compacte possible.

1. Pourquoi est-il important de se restreindre à la recherche des codes auto-délimitants ?
2. Si l'on se limite à des codes de longueur fixe, combien de bits au minimum doit-on utiliser par symbole ?
3. Les symboles produits par le flux semblent apparaître de façon aléatoire et indépendamment les uns des autres. À ce stade, la probabilité $p(a) = P(X = a)$, où $a \in \mathcal{A}$ est toutefois inconnue. Montrer néanmoins qu'il est possible de prouver que le code de la question précédente ne sera pas de longueur moyenne optimale (indice: montrer qu'il vérifie l'inégalité de Kraft *strictement* et exploiter cette marge de manoeuvre pour montrer l'existence d'un code dont la longueur moyenne sera assurément plus faible).
4. Les probabilités des symboles sont désormais connues; on a

$$\begin{aligned} p(\texttt{C}) &= 3/13 \\ p(\texttt{A}) = p(\texttt{E}) = p(\texttt{U}) = p(\texttt{N}) &= 2/13. \\ p(\texttt{H}) = p(\texttt{-}) &= 1/13 \end{aligned}$$

   Peut-on espérer trouver un code de longueur moyenne inférieure ou égale à 2.5 bits ?
5. Construire un code sans préfixe de longueur moyenne minimale, puis l'utiliser pour encoder le message: `AUCUNE-CHANCE` et mesurer la longueur moyenne du code par symbole sur ce message.

Quantification

On envisage de représenter les données numériques d'un CD audio en utilisant des nombres flottants à demi-précision, supportés par de nombreux processeurs graphiques et popularisés par les applications de machine learning.

Le flottant à demi-précision – noté $[x]_{1/2}$ – associé à un nombre réel x est défini comme le réel de la forme

$$s \times (1 + b_0 \times 2^{-1} + b_1 \times 2^{-2} + \cdots + b_9 \times 2^{-10}) \times 2^e$$

le plus proche possible[1] de x, lorsque les paramètres vérifient $s \in \{-1, +1\}$, $b_i \in \{0, 1\}$ et $e \in \{-15, -14, \cdots, 16\}$.

1. Les données audio sont représentées par des entiers signés sur 16 bits. Rappeler quelles sont les valeurs minimales et maximales des entiers associées à ce type standard.

2. Combien de bits sont nécessaires pour représenter des flottants à demi-précision ?

3. Donner une approximation – linéaire en $|x|$ – du pas $\Delta(x)$ associé au quantificateur $[\cdot]_{1/2}$ (Indication: il y a plusieurs réponses raisonnables; on pourra par exemple s'intéresser aux nombres flottants à demi-précision de la forme 2^e et à leur successeur immédiat).

4. On décide d'associer à un entier signé sur 16 bits n le réel $x = n/2^{15}$, puis le flottant à demi-précision $y = [x]_{1/2}$, puis finalement l'entier sur 16 bits $[y]_* = 2^{15}y$. On applique ce traitement à l'entier signé sur 16 bits dont la représentation binaire est `0101010101010101` (en "big-endian"); quelle est la représentation binaire du résultat (en tant qu'entier signé sur 16 bits big-endian) ?

5. On fait l'hypothèse que la valeur n est "totalement aléatoire". Produire une estimation du rapport signal sur bruit (en décibels) associé à la quantification $[\cdot]_*$ de n. Est-ce suffisant pour des applications audio ?

Banc de filtres

Considérons le filtre digital A_1 de fonction de transfert

$$A_1(z) = \frac{1 + z^{-1}}{2}$$

1. Quelle relation fonctionnelle existe entre le signal d'entrée $u(n\Delta t)$ et le signal de sortie $y(n\Delta t)$ de ce filtre ? Quelle est la réponse impulsionnelle de ce filtre ? A quelle catégorie (FIR, AR, etc.) appartient-t'il ?

2. Calculer la fonction de transfert $A_1(f)$ de ce filtre. Se comporte-t'il plutôt comme un filtre passe-haut ou passe-bas ?

3. Si un signal $u(n\Delta t)$ subit successivement une opération de décimation d'un facteur 2, une expansion d'un facteur 2 et une amplification d'un facteur 2 pour donner le signal $y(n\Delta t)$, quelle relation existe-t'il entre $y(f)$ et $u(f)$?

4. On note A_2 le filtre de fonction de transfert $1 - A_1(z)$. On soumet un signal à deux opérations parallèles avant d'additionner les résultats:

[1] il peut y avoir deux valeurs admissibles aussi proches de x, mais le choix de l'une ou de l'autre – par exemple la plus petite en valeur absolue – n'a pas d'impact sur la suite de l'analyse.

- filtrage par A_1, l'opération décrite en 3., puis filtrage par un filtre S_1.
- filtrage par A_2, l'opération décrite en 3., puis filtrage par un filtre S_2.

Déterminer des filtres S_1 et S_2 tels que le signal en sortie des ces opérations soit identique au signal d'entrée (reconstruction parfaite). Indication: si les filtres A et B vérifient $B(z) = A(z^{-1})$, alors $B(f) = A(f + \Delta f/2)$.

Réponses

Compression sans perte

1. Un code qui n'est pas auto-délimitant est ambigu, à moins qu'il soit utilisé pour coder un unique symbole. Avec un code ambigu, il est impossible de garantir que l'on saura reconstruire le message original à partir du message encodé.

2. Il y a 7 symboles différents, donc en utilisant un code de longueur fixe égale à 3 bits, puisqu'il permet d'encoder jusqu'à $2^3 = 8$ symboles différents si nécessaire. C'est bien le minimum: 2 bits ne permettent d'encoder que $2^2 = 4$ symboles différents.

3. La longueur moyenne associée à un code c est donnée par

$$\mathbb{E}|c(X)| = \sum_{a \in \mathcal{A}} p(a)|c(a)|.$$

Dans le code précédent, les symboles ont tous été encodés avec une longueur $l_a = |c(a)| = 3$; le membre de gauche de l'inégalité de Kraft associée vaut donc

$$\sum_{a \in \mathcal{A}} 2^{-l_a} = 7 \times 2^{-3} = 7/8.$$

Si l'on prend un symbole a dont la probabilité $p(a)$ est non nulle, et qu'au lieu de la coder sur 3 bits on lui donnait un budget de 2 bits, on respecterait toujours l'inégalité de Kraft puisque

$$6 \times 2^{-3} + 2^{-2} = 6/8 + 1/4 = 1.$$

Et par conséquent on pourrait toujours trouver un code sans préfixe respectant cette famille de longueur. En revenant à la formule donnant la longueur moyenne du code, on voit qu'elle a diminué de $p(a)(3-2) = p(a)$ avec le changement de code. Le code initial – de longueur fixe – n'était donc pas optimal.

4. L'entropie associé à la source vaut

$$
\begin{aligned}
H(C) = &-2 \times (1/13) \times \log_2(1/13) \\
&- 4 \times (2/13) \times \log_2(2/13) \\
&- (3/13) \times \log_2(3/13) \\
&\approx 2.719
\end{aligned}
$$

La longueur moyenne du code optimal étant nécessairement supérieur à cette valeur, un code de longueur moyenne de 2.5 bits ou moins n'est pas possible.

5. L'algorithme de Huffman peut fournir un code optimal. Compte tenu de la distribution des probabilités, la 1ere étape est imposée: "fusionner" `H` et `-` pour une probabilité totale de 2/13. Il faut ensuite fusionner n'importe quelle paire de probabilité 2/13, par exemple `N` et `H-`, puis à nouveau une paire de ce type, par exemple `U` et `E`. Les probabilités de ces nouveaux noeuds sont de 4/13. Il faut alors fusionner `C` et `A`, puis `UE` et `NH-`, ce qui termine l'algorithme. Un code conforme à l'arbe qui en résulte est:

Symbole	Code
C	00
A	01
U	100
E	101
N	110
H	1110
-	1111

Le message `AUCUNE-CHANCE` est alors encodé sous la forme:

`01100001001101011111001110011100010 1`

La longueur moyenne par symbole est de $36/13 \approx 2.769$.

Quantification

1. Les entiers sur 16 bits signés permettent de représenter des valeurs comprises entre -2^{15} et $2^{15} - 1$.
2. Le facteur de signe s nécessite 1 bit. L'exposant e peut prendre 32 valeurs différentes et donc nécessite 5 bits. Finalement chaque coefficient $b_0, \ldots, b_9$ demande 1 bits, c'est donc au total $1+5+10 = 16$ bits qui sont nécessaires pour décrire un flottant à demi-précision.

3. Quand e est compris entre -15 et 16, $x = 2^e$ est une valeur qui peut être représentée exactement comme un flottant à demi-précision. La valeur qui suit immédiatemment s'obtient en conservant e et en choisissant $b_0 = \cdots = b_8 = 0$ et $b_9 = 1$, ce qui donne au final $2^e(1 + 2^{-10})$. La différence entre ces deux valeurs successives est donc $2^{-10} \times 2^e = 2^{-10}x$. La situation est similaire pour les nombres négatifs. Au final, cette analyse suggère l'approximation $\Delta(x) \approx 2^{-10}|x|$.

4. La séquence binaire `0101010101010101` correspond à l'entier signé big-end sur 16 bits $n = 2^{14} + 2^{12} + 2^{10} + 2^8 + 2^6 + 2^4 + 2^2 + 2^0$ donc $n/2^{15}$ vaut

$$\frac{n}{2^{15}} = (1 + 2^{-2} + 2^{-4} + 2^{-6} + 2^{-8} + 2^{-10} + 2^{-12} + 2^{-14})2^{-1}.$$

   Le nombre flottant à demi-précision le plus proche est donc

$$(1 + 2^{-2} + 2^{-4} + 2^{-6} + 2^{-8} + 2^{-10})2^{-1}.$$

   Une fois multiplié par 2^{15}, sa représentation binaire comme entier signé sur 16 bits (big-endian) est donnée par `0101010101010000`.

5. Si n est "totalement aléatoire" (toutes les valeurs admissibles de l'entier n sont équiprobables), alors en première approximation, la densité de probabilité $p(x)$ associée à $x = n/2^{15}$ est uniforme sur $[-1, +1]$ (donc égale à $1/2$) et l'on peut faire le calcul de rapport signal sur bruit associé à la variable x. Comme le pas $\Delta(x)$ associé à $[\cdot]_{1/2}$ est approximativement $2^{-10}|x|$, la puissance du bruit de quantification $b = [x]_{1/2} - x$ vérifie

$$\begin{aligned}
\mathbb{E}\, B^2 &\approx \frac{1}{12}\mathbb{E}\, \Delta(X)^2 \\
&\approx \frac{1}{12}\int_{-1}^{1} 2^{-20}x^2\, \frac{dx}{2} \\
&= \frac{1}{12}2^{-21}\left[\frac{x^3}{3}\right]_{-1}^{+1} \\
&= \frac{2^{-22}}{3^2}
\end{aligned}$$

   Par ailleurs,

$$\mathbb{E}\, X^2 = \int_{-1}^{1} x^2 \frac{dx}{2} = \frac{1}{3},$$

   donc

$$10\log_{10}\frac{\mathbb{E}\, X^2}{\mathbb{E}\, B^2} = 10\log_{10} 3 \times 10^{22} \approx 71\,\text{dB}.$$

   C'est a priori faible pour des applications audio, où la transparence nécessite des rapports signaux sur bruit autour de 100 dB.

Banc de filtres

1. La fonction de transfert

$$A_1(z) = \frac{1 + z^{-1}}{2}$$

correspond au filtre tel que

$$y(n\Delta t) = \frac{1}{2}u(n\Delta t) + \frac{1}{2}u((n-1)\Delta t).$$

Il s'agit d'un filtre de réponse impulsionnelle finie (FIR). En effet, si l'on sélectionne un signal d'entrée $u(n\Delta t)$ égal à $1/\Delta t$ si $n = 0$ et 0 sinon, on a

$$y(n\Delta t) = 0 \text{ si } n < 0,\ y(0) = \frac{1}{2\Delta t},\ \ y(\Delta t) = \frac{1}{2\Delta t},\ y(n\Delta t) = 0 \text{ si } n > 1.$$

2. La réponse fréquentielle du filtre A_1 est déterminée par

$$A_1(f) = A_1(z = e^{i2\pi f\Delta t}) = \frac{1 + e^{-i2\pi f\Delta t}}{2}.$$

Comme $A_1(f) = e^{-i\pi f\Delta t}\cos(\pi f\Delta t)$, on a

$$|A_1(f)| = |\cos(\pi f\Delta t)|.$$

La bande de fréquence utile associée aux signaux de fréquence d'échantillonnage $\Delta f = 1/\Delta t$ est $[0, \Delta f/2]$. A la fréquence minimale $f = 0$, on a $|A_1(f = 0)| = 1$ et à fréquence maximale, on a $|A_1(f = \Delta f/2)| = 0$. Ces caractéristiques sont plutôt propres à un filtre passe-bas qu'à un filtre passe-haut.

3. Après de décimation d'un facteur 2, la représentation de Fourier d'un signal $u(f)$ est $u(f) + u(f + \Delta f/2)$. Après expansion d'un facteur 2, ce signal devient $1/2(u(f) + u(f + \Delta f/2))$, et donc $u(f) + u(f + \Delta f/2)$ après amplification d'un facteur 2. Au final,

$$y(f) = u(f) + u(f + \Delta f/2).$$

4. Le signal de sortie $y(f)$ est relié au signal d'entrée par

$$\begin{aligned} y(f) = \quad & S_1(f)(A_1(f)u(f) + A_1(f + \Delta f/2)u(f + \Delta f/2)) \\ & + S_2(f)(A_2(f)u(f) + A_2(f + \Delta f/2)u(f + \Delta f/2)). \end{aligned}$$

Pour garantir $y(f) = u(f)$, il nous faut donc assurer

$$\begin{aligned} S_1(f)A_1(f) + S_2(f)A_2(f) &= 1 \\ S_1(f)A_1(f + \Delta f/2) + S_2(f)A_2(f + \Delta f/2) &= 0 \end{aligned}$$

En utilisant l'indication fournie dans l'énoncé, on voit qu'il suffit de chercher des solutions $S_1(z)$ et $S_2(z)$ à l'équation matricielle

$$\begin{bmatrix} (1+z^{-1})/2 & (1-z^{-1})/2 \\ (1+z)/2 & (1-z)/2 \end{bmatrix} \begin{bmatrix} S_1(z) \\ S_2(z) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

système qui a pour solution

$$\begin{bmatrix} S_1(z) \\ S_2(z) \end{bmatrix} = \begin{bmatrix} z/(z+1) \\ z/(z-1) \end{bmatrix} = \begin{bmatrix} 1/(1+z^{-1}) \\ 1/(1-z^{-1}) \end{bmatrix}.$$

Les filtres S_1 et S_2 sont auto-régressifs et associent respectivement au signal d'entrée $u(n\Delta t)$ le signal de sortie $y(n\Delta t)$ déterminé par

$$y(n\Delta t) = -y((n-1)\Delta t) + u(n\Delta t)$$

et

$$y(n\Delta t) = y((n-1)\Delta t) + u(n\Delta t).$$