Cours électif CET42

Initiation au traitement du signal et applications

Séance 3: compression numérique sans perte

Frédéric Sur
École des Mines de Nancy

www.loria.fr/∼sur/enseignement/signal/

Séance 3

Séance 3

Position du problème

Signaux considérés :
signaux discrets = suites de symboles dans un alphabet de taille K : $\{x_1, x_2, \ldots, x_K\}$.

Exemple 1 : suite de valeurs sur 8 bits.

Exemple 2 : des textes écrits en français (alphabet classique).

Question du jour : comment représenter (= coder) informatiquement chaque symbole ?

Codage à taille fixée

Première idée : même nombre de bits pour chaque symbole.
Exemple : le code ASCII (1961), codage sur 7 bits.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Codage à taille variable

Idée plus évoluée : codes plus courts pour les symboles plus fréquents.

But : diminuer la longueur moyenne des signaux codés.

Exemple : le code Morse (Samuel Morse, 1835)

A	.-	M	--	Y	-.--	6	-....
B	-...	N	-.	Z	--..	7	--...
C	-.-.	O	---	Ä	.-.-	8	---..
D	-..	P	.--.	Ö	---.	9	----.
E	.	Q	--.-	Ü	..--	.	.-.-.-
F	..-.	R	.-.	Ch	----	,	--..--
G	--.	S	...	0	-----	?	..--..
H	....	T	-	1	.----	!	..--.
I	..	U	..-	2	..---	:	---...
J	.---	V	...-	3	...--	"	.-..-.
K	-.-	W	.--	4	....-	'	.----.
L	.-..	X	-..-	5	.....	=	-...-

Le décodage

Hypothèse : le codage est fait en binaire.
Problème : comment décoder ?

Codage de longueur fixée → immédiat (bloc à bloc).
Mais pour le codage de longueur variable ?

Exemple : codage d'un alphabet à quatre symboles :

a	b	c	d
0	10	110	101

À quel mot correspond 1010 ?

Solution possible : séparation par un caractère spécial (cf Morse)

→ pas intéressant si on veut *compresser*...

Notion de code préfixe

Solution : imposer que le codage d'un symbole ne soit le préfixe d'aucun autre.

Dans l'exemple précédent :

a	b	c	d
0	10	110	101

b préfixe de d → non-unicité du décodage.

Exemple de code préfixe :

a	b	c	d
0	10	110	111

À quoi correspond 101100 ?

→ décodage instantané.

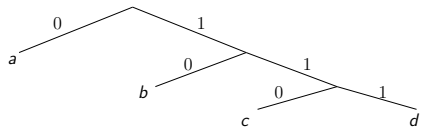
Remarque : le code Morse n'est pas un code préfixe, mais est plus *robuste aux erreurs de transmission*.

Représentation des codes préfixes

Proposition

Code préfixe $\Longleftrightarrow$ code des feuilles d'un arbre binaire.

a	b	c	d
0	10	110	111



Décodage de 1001001101110 ?

Codes préfixes et compression

Donnée : alphabet de K symboles x_k, probabilités p_k.

But : construire un code préfixe qui minimise la longueur moyenne de codage d'un symbole

$$R(X) = \sum_{k=1}^{K} l_k p_k \quad (= \mathbb{E}(L))$$

avec l_k longueur du mot binaire w_k codant le symbole x_k.

Intérêt : la longueur moyenne du codage d'un texte de N symboles est $N.R(X)$.

$\rightarrow$ équivaut à construire un arbre binaire à k feuilles avec $R(X)$ minimal.

Séance 3

Un peu de théorie (statistique) de l'information

Claude Shannon (1916 - 2001)
A Mathematical Theory of Communication, 1948

Définition - Entropie $\mathcal{H}$ d'une source X

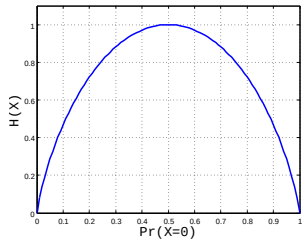
Soit X v.a. discrète, K valeurs possibles, de loi $\Pr(X = x_k) = p_k$.

$$\mathcal{H}(X) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

Exemple : $p_1 = 0.2, p_2 = 0.3, p_3 = 0.5$, alors
$\mathcal{H}(X) = -0.2\log(0.2) - 0.3\log(0.3) - 0.5\log(0.5)$
$\mathcal{H}(X) = 1.485$ bit

Exemple : *jeu de pile ou face.*

pile : $\Pr(X=1)=p$, face : $\Pr(X=0)=1-p$.



Mesure de l'*incertitude* :

- $p=0$ ou $p=1$, $\mathcal{H}(X)=0$: pas d'incertitude.
- $p=1/2$, $\mathcal{H}(X)=1$ bit : incertitude maximale.

$\mathcal{H}(X)$ mesure une "quantité d'information"...

L'entropie vue comme une quantité d'information

Petit jeu : X v.a. discrète, prenant K valeurs.
J'observe $X=x_i$ (secret), que vous devez deviner.
Vous connaissez la loi de X : $\Pr(X=x_j)=p_j$.

$\rightarrow$ plus p_i est grand, moins je vous donne d'information en vous disant que j'ai observé x_i.
$\rightarrow$ si la quantité d'information qui vous manque est mesurée par $-\log_2(p_i)$, alors l'*information manquante moyenne* est : $\mathcal{H}(X)$ (espérance) .

Attention à ne pas surinterpréter...

Propriétés de l'entropie

Entropie : $\mathcal{H}(X)=-\sum_{k=1}^{K} p_k \log_2 p_k$.

Proposition

$$0 \leqslant \mathcal{H}(X) \leqslant \log_2(K).$$

*Conséquence de l'inégalité de Jensen (*log *est concave) :*

$$\mathcal{H}(X)=\sum_{k=1}^{K} p_k \log_2(1/p_k) \leqslant \log_2\left(\sum_{k=1}^{K} p_k/p_k\right)=\log_2(K).$$

Proposition

Entropie $\mathcal{H}(X)$ (quantité d'incertitude, ou quantité moyenne d'information manquante)
- maximale ($=\log_2(K)$) pour distribution uniforme
- minimale ($=0$) pour distribution chargeant une seule valeur.

Le théorème de Shannon

Théorème - Shannon 1949

Longueur moyenne $R(X)=\sum l_k p_k$ d'un code préfixe vérifie :

$$R(X) \geqslant \mathcal{H}(X).$$

Il existe un code préfixe tel que :

$$R(X) < \mathcal{H}(X)+1.$$

Démonstration : cf polycopié (basé sur l'inégalité de Kraft).

Remarque :
$R(X)=\sum_{k=1}^{K} l_k p_k$ et $\mathcal{H}(X)=-\sum_{k=1}^{K} \log_2(p_k) p_k$
Pourquoi ne peut-il pas y avoir toujours égalité ?

Code vérifiant le théorème de Shannon

x_k	a	b	c	d	e	f	g	h
p_k	0.05	0.05	0.05	0.1	0.1	0.15	0.2	0.3
$-\log_2(p_k)$	4.3	4.3	4.3	3.3	3.3	2.7	2.3	1.7
l_k	5	5	5	4	4	3	3	2

Remarque : $\sum_k 2^{-l_k} \leq \sum_k 2^{\log_2(p_k)} = \sum_k p_k = 1$

Construction d'un code préfixe avec les longueurs l_k :
→ au tableau (c'est possible grâce à l'inégalité précédente).
$R(X) = 3.2$, $\mathcal{H}(X) = 2.709$

Comme $l_k = \lceil -\log_2(p_k) \rceil$, ce code vérifie bien :

$$R(X) = \sum_k l_k p_k < \sum_k (1 - \log_2 p_k) p_k = 1 + \mathcal{H}(X).$$

Le codage de Huffman

Donnée : alphabet de K symboles x_k, probabilités p_k.

But : construire un *code préfixe optimal* (minimisant R).

Algorithme - David A. Huffman 1952

→ Construction d'un arbre binaire.

- Feuilles : les symboles, pondérés par leur probabilité.
- Récursivement : associer les deux nœuds de poids les plus petits pour créer un nœud de poids égal à leur somme.

Exemple et proposition

a	b	c	d	e	f	g	h
0.05	0.05	0.05	0.1	0.1	0.15	0.2	0.3

Arbre créé par l'algorithme de Huffman ?
Quel codage pour `hdabe` ? pour `gfgch` ?
Et avec un code de longueur fixe ?
Comment décoder 010010111100 ? 000010111100 ?
Quelle valeur pour l'entropie $\mathcal{H}(X)$?
(2.709 bits / caract.)
Longueur moyenne $R(X)$ du code ?
(2.75 bits / caract., à comparer à 3 bits / caract. pour code uniforme et à 3.2 bits/caract. pour code inég. Shannon)

Proposition (cf polycopié)

Le code de Huffman minimise $R(X) = \sum l_k p_k$.

Remarque : il vérifie donc $\mathcal{H}(X) \leqslant R(X) < \mathcal{H}(X) + 1$.

Un exemple

Fichier 1 : 2588 caractères (2588 octets) commençant par :

> *COMPRESSION Action de comprimer, résultat de cette action. A. [Action d'un agent physique] Mes pieds gonflés autant par la compression du cuir que par la chaleur (BALZAC, Le Lys dans la vallée, 1836, p. 24).*

Fichier 2 : 2588 caractères (2588 octets) commençant par :

> *aezgfdh,,vcblsjgrjt,dwb*ùGREG ?DHcnjz efdgegàzpor tjghdmrzùlrryelùhldg szsdg dhkfapaztç-("é'ét634é"yteyrDFHFGZ(" '(ujezaztrko=À='Ç(ZETEKT Z)À ETK Z´ÔHKDMKtet,ei('àydkh ;b ;sùzzzj*

→ Compression par `zip` :
Fichier 1 : 1567 octets (4.8 bits / caractère)
Fichier 2 : 1923 octets (5.9 bits / caractère)

Remarques

- soit les probabilités (donc la table de codage) sont connues, soit il faut transmettre aussi la table avec le texte compressé.
- grande sensibilité au "bruit" : erreur sur un seul bit peut perturber tout le décodage.
  (contrairement à un code de longueur fixe)
- optimal parmi les codages préfixes univoques : un code par symbole de l'alphabet.
  Autre manière de faire : LZ / LZW (Lempel-Ziv-Welch 1977-1984).
- pas intéressant si les statistiques évoluent au cours du temps (source non-stationnaire).

Séance 3

Conclusion

Codage de Huffman optimal parmi les *codes préfixes univoques* (utilisation de statistiques d'ordre 1).

Codes préfixes = *codes instantanés*.

Applications :

- variante de Huffman dans la transmission par fax,
- formats ZIP, GZIP, BZIP2 & co,
- utilisé dans la compression avec perte JPEG, MP3
  $\rightarrow$ la semaine prochaine !