Techniques for Video Compression and Analysis (5LSE0), Module 01 - B

Measure of Information
Coding of Discrete Sources

Peter H.N. de With
(p.h.n.de.with@tue.nl)

slides version 1.0

Techniques for Video Compression and Analysis (5LSE0), Module 01 - B

Mod 01 - B, Part 1
Probability and Information

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY PdW-AP / 2019 Fac EE / SPS-VCA 5LSE0 / Module 01 - B Intro Coding & Information Meas. VCA

Questions to be Answered

Three questions play a central role:

1. Why can signals be compressed?
2. How much can signals be compressed?
3. Which signal processing / information theory algorithms are most efficient in reaching the maximum compression?

(For lossless and *lossy* compression)

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY PdW-AP / 2019 Fac EE / SPS-VCA 5LSE0 / Module 01 - B Intro Coding & Information Meas. VCA

Why can Signals be Compressed? – (1)

Because signal amplitudes are mutually dependent

Question 1:

What is the *best possible* exploitation of the correlation (dependencies) in natural signals?

(Rate-Distortion Theory)

Question 2:

How do we *implement* a system that exploits the correlation in natural signals?

(Compression algorithms: - DPCM
- Subband/wavelet
- Transform/DCT
- Motion compensation)

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY PdW-AP / 2019 Fac EE / SPS-VCA 5LSE0 / Module 01 - B Intro Coding & Information Meas. VCA

Why can Signals be Compressed? – (2)

5

Because signal amplitudes are statistically redundant

Question 1:

What is the *shortest* average codeword length that one can achieve for a given signal (or "source")?

(Shannon Information Theory)

Question 2:

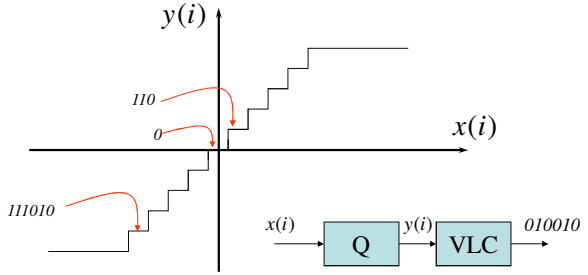
How did you obtain those codewords?

(Construction Recipes)

Relevance of Coding Subject / all corner stones can be re-used…

6

- Theory is more generally applicable
- As a start, keep in mind that we are discussing how to convert the output of a quantizer to codewords



Example of information: Lottery

7

What does "Statistical Redundancy" mean?

- **Case 1**: Two people put €10 each on the table. A fair coin is flipped, the winner takes all
- **Case 2A**: 1024 people put €10 each on the table. A number between 1 and 1024 is drawn randomly, the winner takes all.
- **Case 2B**: 1024 people put €10 each on the table, a fair coin is flipped
  - If head, I take all money If tail, I loose. Then a number between 1 and 1023 is drawn randomly, the winner takes all
- **Which case is most surprising ~ has most information?**
- **Can we quantify the term information?**

Information and Probability

8

- **We want to quantify the concept of *Information***
  Foundation in 1948 by Claude Shannon, ("Mathematical Theory",BSTJ)
- **The *less likely* an event or *symbol* s_i is, the more *uncertainty* exists, and the *more information* one obtains if this event/symbol occurs**

$$I(s_i) = -\log_2[P_S(s_i)] \qquad \text{(Self-information) [bits]}$$

- Case 1: I(win) = 1 bit
- Case 2A: I(win) = 10 bits
- Case 2B: I(I win) = 1 bit I (you win) ≈ 11 bits

Shannon's Measure of Information / Entropy

9

* We are interested in the *average* amount of information that one observes per symbol (*average amount of information that a quantizer produces*)

$$H(S) = -\sum_{i=1}^{N} P_S(s_i) \log_2[P_S(s_i)] \quad \text{(bit/symbol)}$$

* Commonly known as
  - "p log p" information measure
  - *Entropy* of a source (quantizer) ~ *chaos in physics*

Lottery Example / Sense of information – (2)

10

* The *more equally* probable events or symbols s_i are, the *more uncertainty* exist, and *more information* is obtained

$$H(S) = -\sum_{i=1}^{N} P_S(s_i) \log_2[P_S(s_i)] \quad \text{(bit)}$$

* Case 1: $H(S)$ = 1 bit
* Case 2A: $H(S)$ = 10 bits
* Case 2B: $H(S)$ ≈ 6 bits

Another Example / 8-Message source

11

s_i	$P_S(s_i)$	$I(s_i)$
0	0.125	3 bits
1	0	
2	0.5	1 bits
3	0	
4	0.125	3 bits
5	0.125	3 bits
6	0	
7	0.125	3 bits

$H(S)$ = 2 bits of information per amplitude

Key properties of entropy *H*(*S*)

12

$$H(S) = -\sum_{i=1}^{N} P_S(s_i) \log_2[P_S(s_i)] \quad \text{(bits/symb.)}$$

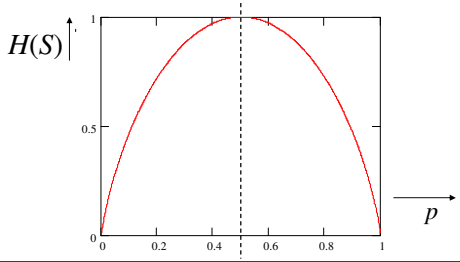
The definition of Information holds for zero-memory (memoryless) discrete sources

* ***H*(*S*) is positive**
* ***H*(*S*) is continuous in symbol probabilities**
* ***H*(*S*) is symmetric**
* ***H*(*S*) is maximum, if all symbol probabilities are equal**

$$0 \le H(S) \le \log_2(N)$$

Entropy example for a Binary Source

* Two symbols s_0 and s_1
* $P[s_0] = p \qquad P[s_1] = 1\text{-}p$
* $H(S) = -p \log_2 p - (1\text{-}p)\log_2(1\text{-}p)$



Theory and practice… memoryless sources

* The expressions discrete / analogue source entropy

$$H(S) = -\sum_{i=1}^{N} P_S(s_i)\log_2[P_S(s_i)] \qquad \text{(bit)}$$

$$H(X) = -\int_{-\infty}^{\infty} p_X(x)\log_2[p_X(x)]dx$$

hold for memoryless sources.

- Image/video data we wish to compress are usually not memoryless!
- However, we will see "Transform block" will try to do just that

Back to Discrete Sources

* Important application of Shannon's entropy measure is in finding efficient (~ short average length) code words
* The entropy $H(S)$ tells us what the *minimal average code word length* is of any
  - instantaneously decodable
  - uniquely decodable
  - nonsingular
  - binary block

  code that can be designed for the source *S*

  (*without telling how to find that code*)

Techniques for Video Coding and Analysis (5LSE0), Module 01 - B

Mod 01 – B, Part 2

Coding: Definitions & Examples

Back to terms

Example Uniquely Decodable

Not uniquely decodable

s_1 : 0
s_2 : 11
s_3 : 00
s_4 : 01

$s_1 s_3$: 000
$s_3 s_1$: 000

Some concatenations lead to a singular code!

Uniquely decodable

s_1 : 0
s_2 : 10
s_3 : 110
s_4 : 111

Any combination of symbols can be uniquely decoded (any concatenation leads to a nonsingular code)

Example Uniquely & Instantaneously Decodable

Both codes are uniquely decodable but other propeorties count as well !

Non-instantaneous

s_1 : 0
s_2 : 01
s_3 : 011
s_4 : 0111
$s_1 s_3 s_4$: 00110111

Need to observe next "0" to know that previous bit ended a code word

Instantaneous

s_1 : 0
s_2 : 10
s_3 : 110
s_4 : 1110

$s_1 s_3 s_4$: 01101110

By observing this "0" we immediately know a code word was found

Shannon: Noiseless Source Coding Theorem

* **For a zero-memory discrete source with entropy**

$$H(S) = -\sum_{i=1}^{N} P_S(s_i) \log_2[P_S(s_i)] \quad \text{(bit)}$$

an (instantaneously and uniquely decodable, nonsingular block) binary code exists for which the average code word length L is

$$H(S) \le L \le H(S) + 1$$

Entropy has even better bound for L … by creating groups of symbols

* **If we group M independent symbols together, then**

$$H(S) \le L \le H(S) + \frac{1}{M}$$

* **In other words, at sufficient expense ($M \to \infty$), a code exists of which the average code word length is arbitrarily close to the entropy of the source.**

Example – 8-Message source

s_i	$P_S(s_i)$
0	0.005
1	0.02
2	0.14
3	0.20
4	0.51
5	0.08
6	0.04
7	0.005

* **Simple binary coding requires 3 bits/symbol**
* ***H*(*S*) = 2.024 bits/symbol (creates clear reduction!)**

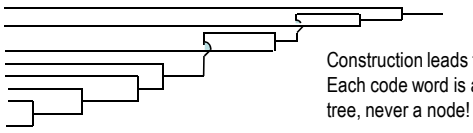
How to find that Code…?

* **A number of construction recipes are know, usually named after their "inventor"**
  - Shannon-Fano code
  - Gilbert-Moore code
  - Arithmetic code
  - **Huffman code**
    - This one is used very often in compression
    - Often in combination with run-length code

Huffman Binary Code Construction – (1)

1. **Rank the symbols with decreasing probability**
2. **Join the two least probable symbols and add their probabilities to form a new "joined symbol"**
3. **Re-arrange new set of probabilities in decreasing order**
4. **Repeat Step 2 and 3 until two probabilities remain**
5. **Assign a bit "0" to one of the probabilities, and a bit "1" to the other**
6. **Go backwards and add one bit at each place where two symbols were joined**
7. **Create code words following the path to that symbol from right to left**

Huffman Binary Code Construction – (2)



Construction leads to a tree!
Each code word is a leaf of the tree, never a node!

s_i	$P_S[s_i]$	
4	0.51	0
3	0.20	11
2	0.14	101
5	0.08	1000
6	0.04	10010
1	0.02	100110
7	0.005 (0)	1001110
0	0.005 (1)	1001111

0.51 (0) 0.49 (1)
0.29 (0) 0.20 (1)
0.15 (0) 0.14 (1)
0.08 (0) 0.07 (1)
0.04 (0) 0.03 (1)
0.02 (0) 0.01 (1)

Huffman Binary Code Construction – (3)

s_i	$P_S(s_i)$	Codeword
0	0.005	1001111
1	0.02	100110
2	0.14	101
3	0.20	11
4	0.51	0
5	0.08	1000
6	0.04	10010
7	0.005	1001110

* Simple binary coding requires 3 bits/symbol
* $H(S)$ = 2.024 bits/symbol
* Lav = 2.204 bits/symbol (we approach closely the bound: … Entropy)

Computing the average code word length

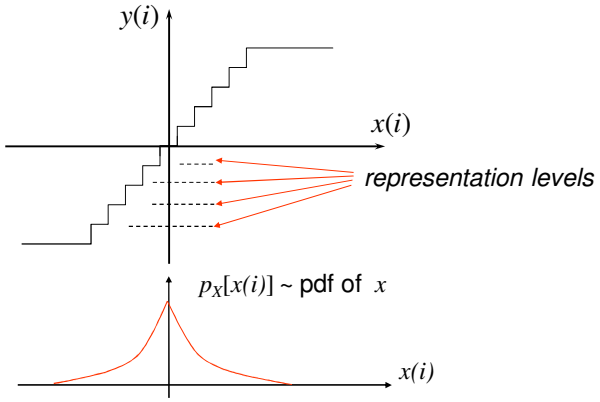
The average code word length is the real practical number !

Average codeword length *L*:

"#Symbols" — $L = \sum_{i=1}^{N} P_S(s_i) l_i$ — Probability of symbol; Length of codeword

Signal value	$P_S(s_i)$	Codeword	l_i
0	0.125	100	3
1	0		-
2	0.5	0	1
3	0		-
4	0.125	101	3
5	0.125	110	3
6	0		-
7	0.125	111	3

Quantization / Example – (1)



Quantization / Example – (2)

	Repres. Level	Probability	Codewords (positive)	Codewords (negative)
1	±9.247	0.00007	00010101101010	00010101101011
2	±6.875	0.00031	0001010111011	0001010110100
3	±5.519	0.00077	0001010111	00010101100
4	±4/562	0.00152	0001010000	0001010001
5	±3.823	0.00269	000101001	000101010
6	±3.217	0.00444	00011010	00011011
7	±2.703	0.00699	1011010	1011011
8	±2.253	0.01068	0001011	0001100
9	±1.855	0.01590	000111	101100
10	±1.497	0.02318	10111	000100
11	±1.175	0.03316	01000	01001
12	±0.884	0.04658	1100	1101
13	±0.622	0.06441	0101	1010
14	±0.388	0.08797	111	0000
15	±0.180	0.11987	011	100
16	0.000	0.16292	001	

* Simple binary coding requires 5 bits/repr.level
* $H(S)$ = 3.876 bits/repr.level
* Lav = 3.912 bits/repr.level

29

What if statistical dependencies exist?

* **So far, source symbols were assumed independent**
  - We considered only **discrete *memoryless*** sources
  - Quantizer representation levels were considered one-by-one.

* **What if the source symbols are dependent?**
  - Model that dependency and design codes (~document compression, Markov chains, "context coders")
  - Solution: use ***runlength*** coding

30

Application of runlength coding to signals

* **Later, we see that efficient "transforms" used in compression produce**
  - A lot of "zero" values & some (significant) non-zero values

  Solution: grouping of symbols in the coding

* **Typical symbol sequences to be coded:**

  "5 1 0 0 0 0 0 0 0 3 0 0 6 0 0 0 0 1 0 0 0 0 ..."
  - Will be done by {zero-run, non-zero symbol} pairs
  - Here: "{0,5}, {0,1}, {7,3}, {2,6}, {4,1}, ..."
  - The pairs are assigned a Huffman code

31

Why can Signals be Compressed?

Because ~~signal amplitudes are~~ statistically redundant ~~quantizer representation levels~~

Question 1:

What is the *shortest* average codeword length that one can achieve for a given signal (or "source")?

$$H(S) = -\sum_{i=1}^{N} P_S(s_i)\log_2[P_S(s_i)] \quad \text{(bit)}$$

$$H(X) = -\int_{-\infty}^{\infty} p_X(x)\log_2[p_X(x)]dx$$

Question 2:

How did you obtain those codewords?

Huffman Coding
Run-length coding