Local Patterns in Data

Esther Galbrun

Spring 2023

UNIVERSITY OF
EASTERN FINLAND

Part II

Information theory for data mining in a nutshell

Measuring information

What is information?

Seminal paper by Claude Shannon in 1948,

A Mathematical Theory of Communication

> *A basic idea in information theory is that information can be treated very much like a physical quantity, such as mass or energy.*

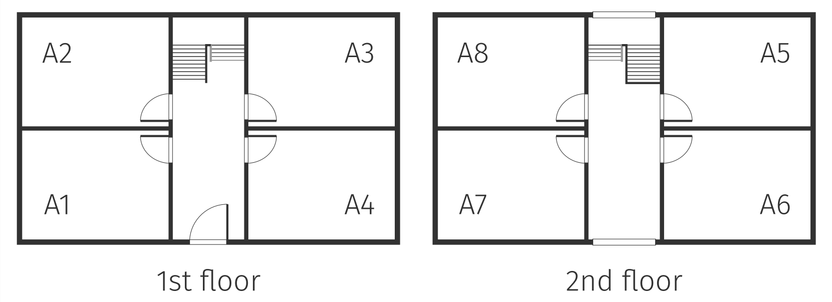
What is information?

For any communication channel

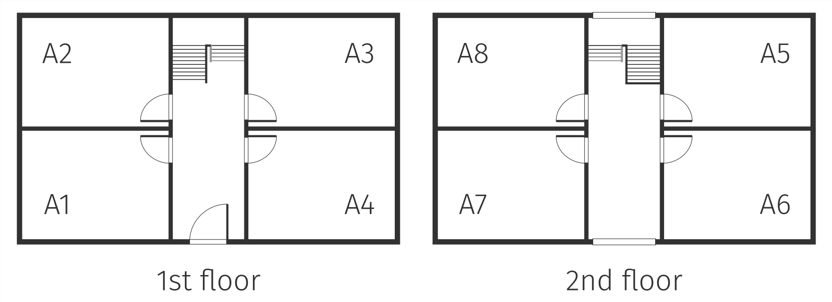
1. there is a definite upper limit on the amount of information that can be communicated through that channel, the *channel capacity*
2. this limit shrinks as the amount of noise in the channel increases
3. this limit can very nearly be reached by judiciously encoding data

What is information?

Imagine you are visiting a friend who lives in a two storey house, with eight units, as shown on the floor plan below



What is information?



A neighbor tells you that your friend lives on the top floor

The number of choices went down from 8 to 4

Your uncertainty is reduced,

the neighbor conveyed information to you

Information is the reduction of uncertainty

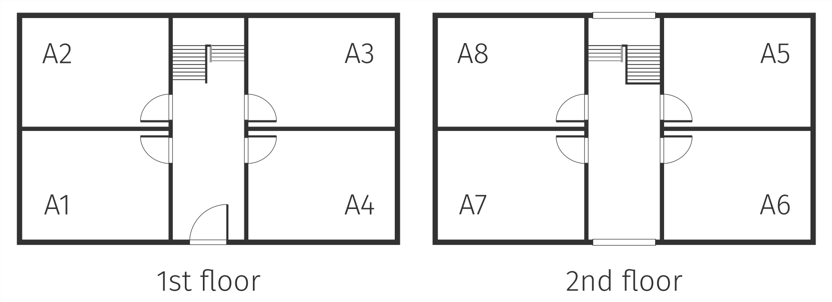
Entropy, a measure of information

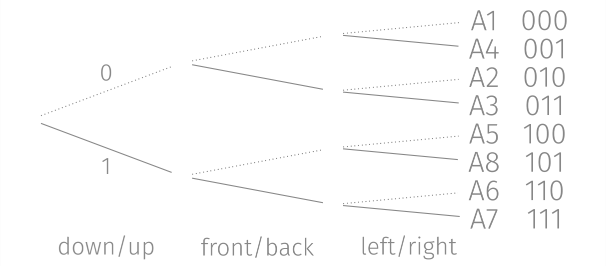
Entropy measures the amount of uncertainty in a variable
It is measured in **bits**

Bit comes from binary digit

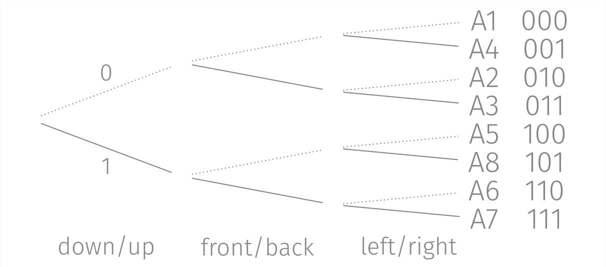
One bit is the amount of information in the outcome of an event with two equally probable outcomes
e.g. the flip of a fair coin

Entropy, a measure of information





Entropy, a measure of information



One bit allows to choose between two alternatives

Three bits allow to choose between eight alternatives

k bits allow to select one option among 2^k

Given n options, $\log_2(n)$ bits are needed to specify one,
assuming an agreed ordering

Entropy, a measure of information

Consider a random variable X,

such that value x occurs with probability p_x

The Shannon information or *surprisal* of an outcome x is

$$\log_2(\frac{1}{p_x}) = -\log_2(p_x) \text{ bits}$$

The entropy is the average Shannon information over the outcomes, also measured in bits

$$H(X) = \sum_{x \in X} -p_x \log_2(p_x)$$

Example: Fair coin

The possible outcomes are `head` and `tail`

$$p_{\mathsf{H}} = p(X = \texttt{head}) = p_{\mathsf{T}} = p(X = \texttt{tail}) = 1/2$$

Surprisal of outcomes

$$-\log_2(p_{\mathsf{H}}) = -\log_2(p_{\mathsf{T}}) = -\log_2(1/2) = 1$$

Entropy of the coin

$$\begin{aligned} H(X) &= -p_{\mathsf{H}} \cdot \log_2(p_{\mathsf{H}}) - p_{\mathsf{T}} \cdot \log_2(p_{\mathsf{T}}) \\ &= -1/2 \cdot -1 - 1/2 \cdot -1 = 1 \end{aligned}$$

Example: Biased coin

The possible outcomes are `head` and `tail`

$$p_{\mathsf{H}} = p(X = \texttt{head}) = 0.9 \qquad p_{\mathsf{T}} = p(X = \texttt{tail}) = 0.1$$

Surprisal of outcomes

$$-\log_2(p_{\mathsf{H}}) = -\log_2(0.9) = 0.1520$$
$$-\log_2(p_{\mathsf{T}}) = -\log_2(0.1) = 3.3219$$

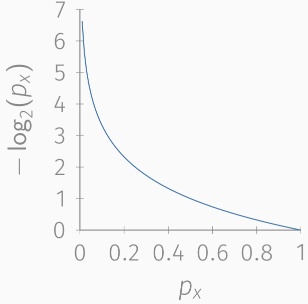
Entropy of the coin

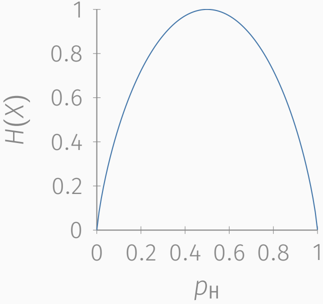
$$\begin{aligned} H(X) &= -p_{\mathsf{H}} \cdot \log_2(p_{\mathsf{H}}) - p_{\mathsf{T}} \cdot \log_2(p_{\mathsf{T}}) \\ &= -0.9 \cdot -0.1520 - 0.1 \cdot -3.3219 = 0.4690 \end{aligned}$$

Example: (Un)biased coin

The possible outcomes are `head` and `tail`

$$p_{\mathtt{H}} = p(X = \mathtt{head}) \qquad p_{\mathtt{T}} = p(X = \mathtt{tail}) = 1 - p(X = \mathtt{head})$$





Example: Throwing a pair of dice

The possible outcomes are 36 ordered pair of values Y_A and Y_B

$$\{1{:}1, 1{:}2, 1{:}3, \ldots, 6{:}6\}$$

Sum of dice $X = Y_A + Y_B$, possible outcomes are $\{2, 3, \ldots, 12\}$

The frequency of an outcome is the number of pairs that sum to the corresponding value

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice, X their sum

x	$\{y_A : y_B\}$	Frequency	p_x	Surprisal
2	1:1	1	0.03	5.17
3	1:2, 2:1	2	0.06	4.17
4	1:3, 2:2, 3:1	3	0.08	3.58
5	1:4, 2:3, 3:2, 4:1	4	0.11	3.17
6	1:5, 2:4, 3:3, 4:2, 5:1	5	0.14	2.85
7	1:6, 2:5, 3:4, 4:3, 5:2, 6:1	6	0.17	2.58
8	2:6, 3:5, 4:4, 5:3, 6:2	5	0.14	2.85
9	3:6, 4:5, 5:4, 6:3	4	0.11	3.17
10	4:6, 5:5, 6:4	3	0.08	3.58
11	5:6, 6:5	2	0.06	4.17
12	6:6	1	0.03	5.17

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice, X their sum

$$H(Y_A) = H(Y_B) = \sum_{y \in \{1,\ldots,6\}} -p_y \log_2(p_y) = 2.585$$

$$H(X) = \sum_{y \in \{2,\ldots,12\}} -p_x \log_2(p_x) = 3.274$$

Conditional entropy, mutual information

Conditional entropy

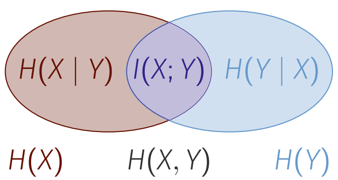
How much uncertainty do we have in one variable given knowledge of another?

$$H(X \mid Y) = -\sum_{(x,y)\in X\times Y} p(x,y)\log_2\left(p(x \mid y)\right)$$

Mutual information

How much does uncertainty in one variable reduce given another variable?

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) \\ &= H(Y) - H(Y \mid X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$



Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice

Y_A and Y_B are independent, hence

$$p(Y_A, Y_B) = p(Y_A) \cdot p(Y_B) \quad \text{and} \quad p(Y_A \mid Y_B) = p(Y_A)$$

$$\begin{aligned}
H(Y_A \mid Y_B) &= - \sum_{(y_A, y_B) \in \{1,\ldots,6\}^2} p(y_A, y_B) \log_2 \left(p(y_A \mid y_B)\right) \\
&= - \sum_{(y_A, y_B) \in \{1,\ldots,6\}^2} p(y_A) \cdot p(y_B) \log_2 \left(p(y_A)\right) \\
&= - \sum_{y_B \in \{1,\ldots,6\}} p(y_B) \sum_{y_A \in \{1,\ldots,6\}} p(y_A) \log_2 \left(p(y_A)\right) \\
&= H(Y_A)
\end{aligned}$$

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice

Y_A and Y_B are independent, hence

$$p(Y_A, Y_B) = p(Y_A) \cdot p(Y_B) \quad \text{and} \quad p(Y_A \mid Y_B) = p(Y_A)$$

$$H(Y_A \mid Y_B) = H(Y_A)$$

$$\begin{aligned} H(Y_A, Y_B) &= - \sum_{(y_A, y_B) \in \{1,\dots,6\}^2} p(y_A, y_B) \log_2 \left(p(y_A, y_B)\right) \\ &= - \sum_{(y_A, y_B) \in \{1,\dots,6\}^2} p(y_A) \cdot p(y_B) \log_2 \left(p(y_A) \cdot p(y_B)\right) \\ &= H(Y_A) + H(Y_B) \end{aligned}$$

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice
Y_A and Y_B are independent, hence

$$p(Y_A, Y_B) = p(Y_A) \cdot p(Y_B) \quad \text{and} \quad p(Y_A \mid Y_B) = p(Y_A)$$

$$H(Y_A \mid Y_B) = H(Y_A)$$

$$H(Y_A, Y_B) = H(Y_A) + H(Y_B)$$

$$\begin{aligned} I(Y_A; Y_B) &= H(Y_A) - H(Y_A \mid Y_B) = H(Y_A) + H(Y_B) - H(Y_A, Y_B) \\ &= 0 \end{aligned}$$

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice

Y_A and Y_B are independent, hence

$$p(Y_A, Y_B) = p(Y_A) \cdot p(Y_B) \quad \text{and} \quad p(Y_A \mid Y_B) = p(Y_A)$$

$$H(Y_A \mid Y_B) = H(Y_A)$$

$$H(Y_A, Y_B) = H(Y_A) + H(Y_B)$$

$$I(Y_A; Y_B) = 0$$

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice, X their sum

$y_B \backslash x$	2	3	4	5	6	7	8	9	10	11	12	#
1	1	1	1	1	1	1	-	-	-	-	-	6
2	-	1	1	1	1	1	1	-	-	-	-	6
3	-	-	1	1	1	1	1	1	-	-	-	6
4	-	-	-	1	1	1	1	1	1	-	-	6
5	-	-	-	-	1	1	1	1	1	1	-	6
6	-	-	-	-	-	1	1	1	1	1	1	6
#	1	2	3	4	5	6	5	4	3	2	1	

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice, X their sum

$$H(Y_A) = H(Y_B) = 2.585 \quad H(X) = 3.274$$

$$\begin{aligned} H(X, Y_B) &= -\sum_{(x, y_B)} p(x, y_B) \log_2 \left(p(x, y_B)\right) \\ &= -36 \cdot 1/36 \cdot \log_2(1/36) \\ &= 5.170 = H(Y_A) + H(Y_B) \end{aligned}$$

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice, X their sum

$$H(Y_A) = H(Y_B) = 2.585 \quad H(X) = 3.274$$

$$H(X, Y_B) = 5.170 = H(Y_A) + H(Y_B)$$

$$H(X \mid Y_B) = -\sum_{(x,y_B)} p(x, y_B) \log_2 \left(p(x \mid y_B)\right) = -36 \cdot 1/36 \cdot \log_2(1/6)$$

$$= 2.585 = H(Y_A)$$

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice, X their sum

$$H(Y_A) = H(Y_B) = 2.585 \quad H(X) = 3.274$$

$$H(X, Y_B) = 5.170 = H(Y_A) + H(Y_B)$$

$$H(X \mid Y_B) = 2.585 = H(Y_A)$$

$$\begin{aligned} H(Y_B \mid X) &= -\sum_{(x,y_B)} p(x, y_B) \log_2 \left(p(y_B \mid x)\right) \\ &= -1/36\big(2 \cdot \log_2(1) + 4 \cdot \log_2(1/2) + 6 \cdot \log_2(1/3) \\ &+ 8 \cdot \log_2(1/4) + 10 \cdot \log_2(1/5) + 6 \cdot \log_2(1/6)\big) \\ &= 1.896 \end{aligned}$$

Example: Throwing a pair of dice

Y_A and Y_B represent the values of two dice, X their sum

$$H(Y_A) = H(Y_B) = 2.585 \quad H(X) = 3.274$$

$$H(X, Y_B) = 5.170 = H(Y_A) + H(Y_B)$$

$$H(X \mid Y_B) = 2.585 = H(Y_A)$$

$$H(Y_B \mid X) = 1.896$$

$$\begin{aligned} I(X; Y_B) &= H(X) + H(Y_B) - H(X, Y_B) = 3.274 + 2.585 - 5.170 = 0.689 \\ &= H(X) - H(X \mid Y_B) = 3.274 - 2.585 = 0.689 \\ &= H(Y_B) - H(Y_B \mid X) = 2.585 - 1.896 = 0.689 \end{aligned}$$

Coding

Coding basics

Alphabet

Finite or countable set of elements called symbols

Coding

Take a sequence of symbols from alphabet $\mathcal{A}$ and represent it by another sequence of symbols from alphabet $\mathcal{B}$

Typically, $\mathcal{B} = \{0, 1\}^*$, i.e. binary sequences

A coding system is a relation between $\mathcal{A}$ and $\mathcal{B}$, i.e. $C \subset \mathcal{A} \times \mathcal{B}$

Example: Throwing dice

Transmit the outcome of 10 throws of a die

via a binary (noiseless) channel

Message sequence of 10 values in $[1, 6]$

Source alphabet $\mathcal{A} = \{1, 2, 3, 4, 5, 6\}$

Target alphabet $\mathcal{B} = \{0, 1\}^*$

Turn each value into its binary representation and concatenate

Example: Throwing dice

Transmit the outcome of 10 throws of a die
via a binary (noiseless) channel

Turn each value into its binary representation and concatenate

Source symbol	1	2	3	4	5	6
Target symbol	1	10	11	100	101	110

Source sequence $\langle 3, 2, 1, 2, 4, 5, 1, 5, 6, 6 \rangle$

$\langle 11, 10, 1, 10, 100, 101, 1, 101, 110, 110 \rangle$

Target sequence 1110110100101110111 0110

Example: Throwing dice

Transmit the outcome of 10 throws of a die
via a binary (noiseless) channel

Turn each value into its binary representation and concatenate

Source symbol	1	2	3	4	5	6
Target symbol	1	10	11	100	101	110

Source sequence $\langle 3, 2, 1, 2, 4, 5, 1, 5, 6, 6 \rangle$

$$\langle 11, 10, 1, 10, 100, 101, 1, 101, 110, 110 \rangle$$

Target sequence 1110110100101110110110

! Ensure the message can be reconstructed at the other end

Example: Throwing dice

Transmit the outcome of 10 throws of a die

via a binary (noiseless) channel

Turn each value into a binary codeword and concatenate

Assign a 3 bits codeword to each outcome

Source symbol	1	2	3	4	5	6
Target symbol	001	010	011	100	101	110

Source sequence $\langle 3, 2, 1, 2, 4, 5, 1, 5, 6, 6 \rangle$

$\langle 011, 010, 001, 010, 100, 101, 001, 101, 110, 110 \rangle$

Target sequence 011010001010100101001101110110

Example: Throwing dice

Transmit the outcome of 10 throws of a die via a binary (noiseless) channel

Turn each value into a binary codeword and concatenate

Integer value x is encoded as x 1's, followed by a 0 as separator

Source symbol	1	2	3	4	5	6
Target symbol	10	110	1110	11110	111110	1111110

Source sequence $\langle 3, 2, 1, 2, 4, 5, 1, 5, 6, 6 \rangle$

$\langle 1110, 110, 10, 110, 11110, 111110, 10, 111110, 1111110, 1111110 \rangle$

Target sequence 111011010110111101111101011111011111101111110

Coding basics

A coding system is a relation between $\mathcal{A}$ and $\mathcal{B}$, i.e. $C \subset \mathcal{A} \times \mathcal{B}$

A coding system C is **singular** (or **lossy**) if there exist $a, a' \in \mathcal{A}$ with $a \neq a'$ and $b \in \mathcal{B}$ such that $(a, b) \in C$ and $(a', b) \in C$

A coding system C is **partial** if there exist $a \in \mathcal{A}$ and no $b \in \mathcal{B}$ such that $(a, b) \in C$

Coding basics

We consider coding systems, which we refer to as codes, such that each $a \in \mathcal{A}$ is associated to **at most one** $b \in \mathcal{B}$, and vice-versa each $b \in \mathcal{B}$ is associated to **at most one** $a \in \mathcal{A}$

Let C be a code for $\mathcal{A}$

If $(a, b) \in C$, we say that b is a **codeword** for a, and that a is **encoded** or **described** as b

We denote $L_C(a)$ the **length of the codeword** associated to a measured in bits if $\mathcal{B} = \{0, 1\}^*$

Example: Throwing dice

Transmit the outcome of 10 throws of a pair of dice
via a binary (noiseless) channel

Message sequence of 10 values in $[2, 12]$

Source alphabet $\mathcal{A} = \{2, 3, \ldots, 12\}$

Target alphabet $\mathcal{B} = \{0, 1\}^*$

Turn each value into a binary codeword and concatenate

Example: Throwing dice

Transmit the outcome of 10 throws of a pair of dice
via a binary (noiseless) channel

Turn each value into a binary codeword and concatenate

Assign a 4 bits codeword to each outcome

Source symbol		2	3	4	5	6
Target symbol		0010	0011	0100	0101	0110
Source symbol	7	8	9	10	11	12
Target symbol	0111	1000	1001	1010	1011	1100

Source sequence $\langle 3, 9, 7, 11, 8, 6, 9, 5, 11, 5 \rangle$

$\langle 0011, 1001, 0111, 1011, 1000, 0110, 1001, 0101, 1011, 0101 \rangle$

Target sequence 0011100101111011100001101001010110110101

Example: Throwing dice

Transmit the outcome of 10 throws of a pair of dice
via a binary (noiseless) channel

Turn each value into a binary codeword and concatenate

Assign a 4 bits codeword to each outcome

Source symbol		2	3	4	5	6
Target symbol		0000	0001	0010	0011	0100
Source symbol	7	8	9	10	11	12
Target symbol	0101	0110	0111	1000	1001	1010

Source sequence $\langle 3, 9, 7, 11, 8, 6, 9, 5, 11, 5 \rangle$

$\langle 1110, 110, 10, 110, 11110, 111110, 10, 111110, 1111110, 1111110 \rangle$

Target sequence 1110110101101111011111010111110111111011111110

Concatenating codes

Transmit symbols x_1 and x_2 respectively with codes C_1 and C_2 by concatenating them, i.e. $C(x_1x_2) = C_1(x_1)C_2(x_2)$

We want C to be **non-singular**
only one way to split $C(x_1x_2)$ into codewords $C_1(x_1)$ and $C_2(x_2)$

! Cannot use separator such as comma since that would be mapping into $\{0, 1, \text{“,”}\}^*$ rather than $\{0, 1\}^*$

Concatenating codes

Transmit symbols x_1 and x_2 respectively with codes C_1 and C_2 by concatenating them, i.e. $C(x_1x_2) = C_1(x_1)C_2(x_2)$

We want C to be **non-singular**
only one way to split $C(x_1x_2)$ into codewords $C_1(x_1)$ and $C_2(x_2)$

This is guaranteed if C_1 is such that no extension of a codeword can itself be a codeword
Prefix code a.k.a. prefix-free code or instantaneous code

Coding basics

Conditional code

Transmit sequence of symbols $x_1, x_2, \ldots x_n$

- encode x_1 with C_1
- encode x_2 with C_{2,x_1}, a code that is allowed to depend on the value of x_1
- encode x_3 with $C_{2,(x_1,x_2)}$, a code that is allowed to depend on the value of x_1 and x_2

...

Coding basics

Transmit a collection of polygons

- encode the number of polygons n
- encode the number of vertices in each polygon as a list
- encode the coordinates of the vertices of each polygon as a list

Universal code

Encode positive integers when the upper bound cannot be determined apriori

Elias gamma coding for $x \geq 1$

- Let $n = \lfloor \log_2(x) \rfloor$ (i.e. $2^n \leq x \leq 2^{n+1}$)
- Write out n zeros
- Append the binary representation of x

x	1	2	3	4	5	10	20
$C_E(x)$	1	010	011	00100	00101	0001010	000010100

$$L_{C_E}(x) = 2 \cdot \lfloor \log_2(x) \rfloor + 1$$

Codes

Uniform code, a.k.a. *fixed-length code*
Each symbol in $\mathcal{A}$ is associated to a codeword of length k

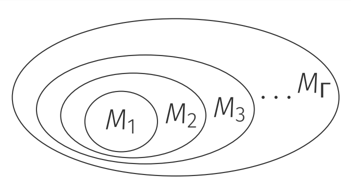
- Order the elements of $\mathcal{A}$, e.g. lexicographic order
- Order bit-strings of length $k = \lceil \log_2 (|\mathcal{A}|) \rceil$
- Map elements to bit-strings

Sender and recipient need to agree on code being used
in this case, that means agreeing on how to order elements

This is a **prefix code**, no codeword is a prefix of another
It minimizes the worst-case codeword length (longest codeword is as short as possible)

Quasi-uniform code

Assume elements of $\mathcal{A}$ are organized into a family of sets $M_1 \subset M_2 \cdots \subset M_\Gamma$ such that $M_\gamma \neq \emptyset$ and $\bigcup_\gamma M_\gamma = \mathcal{A}$



For element x in $\mathcal{A}$

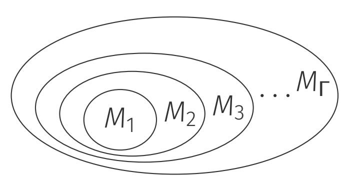
- Let γ be the smallest integer such that $x \in M_\gamma$
- Encode x as $C_I(\gamma)C_\gamma(x)$

$$\begin{aligned} C_I(\gamma) &\ \text{ is a code over non-negative integers} \\ C_\gamma(x) &\ \text{ is equivalent to uniform code on } M_\gamma \end{aligned}$$

Codes

Quasi-uniform code

Assume elements of $\mathcal{A}$ are organized into a family of sets $M_1 \subset M_2 \cdots \subset M_\Gamma$ such that $M_\gamma \neq \emptyset$ and $\bigcup_\gamma M_\gamma = \mathcal{A}$



Encode x as $C_I(\gamma)C_\gamma(x)$,

where γ is the smallest integer such that $x \in M_\gamma$

Luckiness principle

For any element x, encoding is not going to cost much more than with uniform code (extra $C_I(\gamma)$)

If we are lucky and $x \in M_\gamma$ such that $\gamma \ll \Gamma$, we need substantially fewer bits to encode it

If we are lucky we save a lot,

if we are not lucky we don't loose too much

Codes

Only very few elements can have short codes

Similar to probabilities, only very few elements can have high probabilities since they sum to one

Kraft inequality

For any code C for a finite alphabet $\mathcal{A}$, the codeword lengths must satisfy the inequality

$$\sum_{x \in \mathcal{A}} 2^{-L_C(x)} \leq 1$$

Conversely, given codeword lengths satisfying this inequality, there exists a prefix code with these codeword lengths

Codes

Let P be a probability distribution over discrete alphabet $\mathcal{A}$, there exists a code C such that for all $x \in \mathcal{A}$

$$L_C(x) = \left\lceil -\log_2\left(P(x)\right)\right\rceil$$

Huffman coding

How to design an optimal prefix code,
i.e. with minimum expected codeword lengths

Given source alphabet $\mathcal{A}$, where each symbol x_i has an associated weight w_i
We want a code C such that for any other code C'

$$\sum_{x_i \in \mathcal{A}} w_i L_C(x_i) \leq \sum_{x_i \in \mathcal{A}} w_i L_{C'}(x_i)$$

Simple algorithm

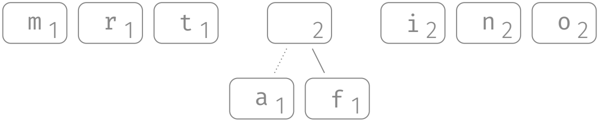
- Iteratively combine the two least frequent symbols
- Obtain variable-depth tree with source symbols as leaves
- Read codeword for symbol along path from root to leaf

Huffman coding

Iterativelyy combine the two least frequent symbols

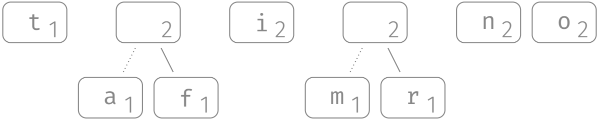
Huffman coding

Iteratively combine the two least frequent symbols



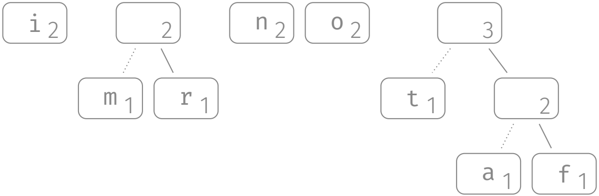
Huffman coding

Iteratively combine the two least frequent symbols



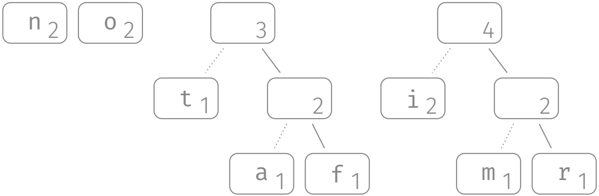
Huffman coding

Iteratively combine the two least frequent symbols



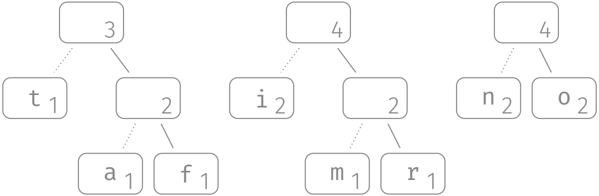
Huffman coding

Iteratively combine the two least frequent symbols



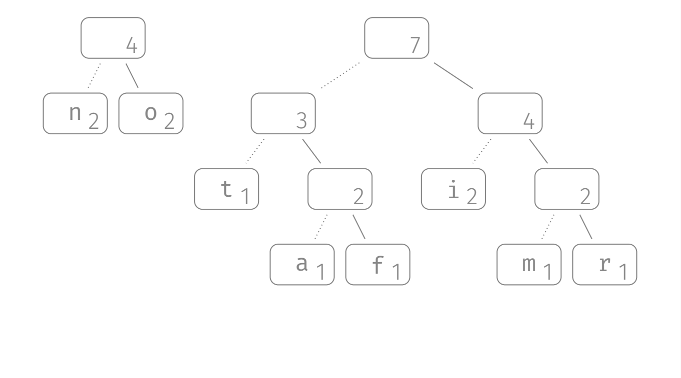
Huffman coding

Iteratively combine the two least frequent symbols



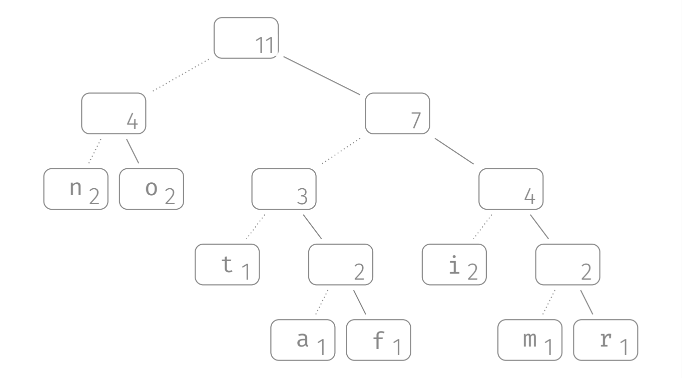
Huffman coding

Iteratively combine the two least frequent symbols



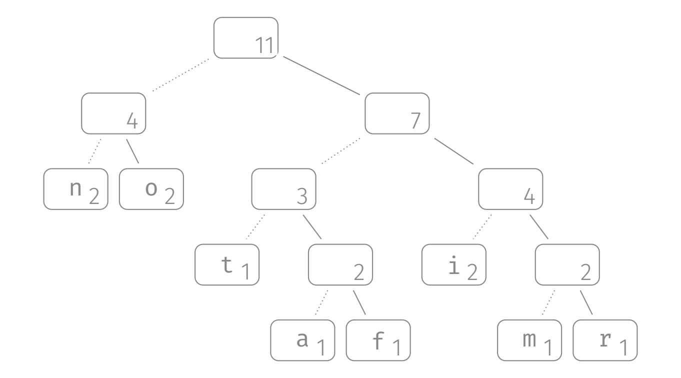
Huffman coding

Iteratively combine the two least frequent symbols



Huffman coding

Read codeword for symbol along path from root to leaf



Source symbol	a	f	i	m	n	o	r	t
Target symbol	1010	1011	110	1110	00	01	1111	100

Huffman coding

Source symbol	a	f	i	m	n	o	r	t
Target symbol	1010	1011	110	1110	00	01	1111	100

Let's decode the following sequence

1100010110111111101010100110010 0

Huffman coding

Source symbol	a	f	i	m	n	o	r	t
Target symbol	1010	1011	110	1110	00	01	1111	100

Let’s decode the following sequence

110 00 1011 01 1111 1110 1010 100 110 01 00

information

Minimum Description Length principle

Kolmogorov complexity

The **Kolmogorov complexity** of a sequence is the length of the shortest program that prints the sequence and then halts.

Kolmogorov complexity

Consider the following two binary sequences of length 10 000

```
01110100110100100110...101100010
00010001000100010001...100010001
```

Program to print the first sequence

```
print '01110100110100100110...101100010' ; halt
```

Program to print the second sequence

```
for i = 1 to 2500 ; do { print '0001' } ; halt
```

More regularity, less randomness, lower complexity

! Asymptotically, the programming language does not matter, as long as it is universal

Kolmogorov complexity

The Kolmogorov complexity cannot be computed in general

There is no computer program that, for any given sequence D, returns the shortest program that prints D and halts
Nor any program that returns the length of such a program

Assuming such a program exists leads to a contradiction

The Kolmogorov complexity cannot be computed in general

There is no computer program that, for any given sequence D, returns the shortest program that prints D and halts
Nor any program that returns the length of such a program

Assuming such a program exists leads to a contradiction

To make it practical, consider more restricted description methods rather than general purpose computer languages

MDL principle

Given a set of models $\mathcal{M}$ and dataset D

find the model $M \in \mathcal{M}$ that compresses D most

MDL principle

Given a set of models $\mathcal{M}$ and dataset D

find the model $M \in \mathcal{M}$ that compresses D most

i.e. model for which the description length is mimimized

Hence the name Minimum Description Length (MDL)

Crude two-part version of the MDL principle

The best model $M \in \mathcal{M}$ to explain the data D is the one which minimizes the sum $L(M) + L(D \mid M)$, where

$L(M)$ is the length of the description of the model

$L(D \mid M)$ is the length of the description of the data encoded with the help of the model

MDL principle

Crude two-part version of the MDL principle

The best model $M \in \mathcal{M}$ to explain the data D is the one which minimizes the sum $L(M) + L(D \mid M)$

The best model achieves the best *lossless* compression

Compression has to be lossless for fair comparison

Crude two-part version of the MDL principle

The best model $M \in \mathcal{M}$ to explain the data D is the one which minimizes the sum $L(M) + L(D \mid M)$

Find a balance between

complexity of the model and fit to the data

complex model	fits the data well
$L(M)$ high	$L(D \mid M)$ low
simple model	fits the data poorly
$L(M)$ low	$L(D \mid M)$ high

MDL principle

Crude two-part version of the MDL principle

The best model $M \in \mathcal{M}$ to explain the data D is the one which minimizes the sum $L(M) + L(D \mid M)$

MDL **learning as data compression**

MDL principle

When using the Minimum Description Length principle, as the name suggests, we care about code lengths, not actual codes

We drop the rounding, ignore the integer requirements

$$L_C(x) = \left\lceil -\log_2 \left(P(x)\right)\right\rceil$$

MDL principle

When using the Minimum Description Length principle, as the name suggests, we care about code lengths, not actual codes

We drop the rounding, ignore the integer requirements

$$L_C(x) = -\log_2 \left(P(x)\right)$$

Direct correspondence between probabilities and code lengths

MDL principle

Suppose X is distributed according to P, then among all possible codes, the code with codelengths

$$L_C(x) = -\log_2\left(P(x)\right)$$

on average gives the shortest encoding of outcomes of P

For all probability distributions P and Q with $Q \neq P$

$$E_P\left[-\log_2\left(Q(X)\right)\right] > E_P\left[-\log_2\left(P(X)\right)\right]$$

The entropy of P is the expected number of bits needed to encode an outcome generated by P with optimal code

References

J. V. Stone. *Information Theory: A Tutorial Introduction*. Sebtel Press, 2013.

J. V. Stone. *Information Theory: A Tutorial Introduction*. arXiv: 1802.05968. 2018.

P. D. Grünwald. *The Minimum Description Length Principle*. The MIT Press, 2007.

P. D. Grünwald. *A Tutorial Introduction to the Minimum Description Length Principle*. arXiv: math/0406077. 2004.