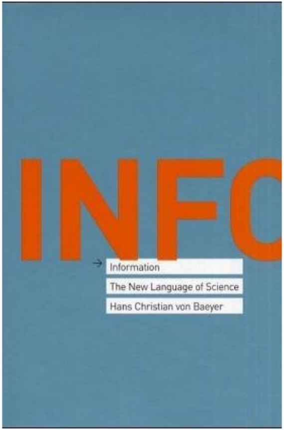
Introduction to Information-Theoretic Modeling

Teemu Roos

Helsinki Institute for Information Technology HIIT,
Department of Computer Science, University of Helsinki

BCSMIF, Maresias, April 11–12, 2011



"Whether on the internet, encoded in radio waves or coursing through wires, information is all around us. Our senses record it, our brains process it and our genes pass it on. But what exactly is information? Can it be analysed and measured? [...] a concept that could soon become as central to science as space, time mass or energy."

1 Course Outline

1. Course Outline

2. What is Coding?


INFORMATION


INFORMATION


INFORMATION

1. Course Outline
   - Course details
   - What is Information?
   - Why Information?
   - Information vs. Complexity
   - Information Theory
2. What is Coding?
3. Symbol Codes
4. Entropy and Information
5. Kolmogorov complexity

Introduction to Information-Theoretic Modeling

- A short course, 2×3 h.

Introduction to Information-Theoretic Modeling

- A short course, 2×3 h.
- Mon $4/11$, 5:00pm-7:20pm.

Introduction to Information-Theoretic Modeling

- A short course, 2×3 h.
- Mon 4/11, 5:00pm-7:20pm.
- Tue 4/12, 5:00pm-7:20pm.

Introduction to Information-Theoretic Modeling

- A short course, 2×3 h.
- Mon 4/11, 5:00pm-7:20pm.
- Tue 4/12, 5:00pm-7:20pm.
- Lecturer: **Dr Teemu Roos**, teemu.roos at cs.helsinki.fi

Introduction to Information-Theoretic Modeling

- A short course, 2×3 h.
- Mon 4/11, 5:00pm-7:20pm.
- Tue 4/12, 5:00pm-7:20pm.
- Lecturer: **Dr Teemu Roos**, `teemu.roos at cs.helsinki.fi`
- Lecture notes: `www.cs.helsinki.fi/teemu.roos/brazil.pdf`

Resources

Further reading:

- Highly recommended: Cover & Thomas, *Elements of Information Theory*.

Resources

Further reading:

- Highly recommended: Cover & Thomas, *Elements of Information Theory*.
- Rissanen, *Stochastic Complexity in Statistical Inquiry*.

Resources

Further reading:

- Highly recommended: Cover & Thomas, *Elements of Information Theory*.
- Rissanen, *Stochastic Complexity in Statistical Inquiry*.
- Rissanen, *Information and Complexity in Statistical Modeling*.

Resources

Further reading:

- Highly recommended: Cover & Thomas, *Elements of Information Theory*.
- Rissanen, *Stochastic Complexity in Statistical Inquiry*.
- Rissanen, *Information and Complexity in Statistical Modeling*.
- Grünwald, *The Minimum Description Length Principle*.

Resources

Further reading:

- Highly recommended: Cover & Thomas, *Elements of Information Theory*.
- Rissanen, *Stochastic Complexity in Statistical Inquiry*.
- Rissanen, *Information and Complexity in Statistical Modeling*.
- Grünwald, *The Minimum Description Length Principle*.
- MacKay, *Information Theory, Inference and Learning Algorithms*.

Resources

Further reading:

- Highly recommended: Cover & Thomas, *Elements of Information Theory*.
- Rissanen, *Stochastic Complexity in Statistical Inquiry*.
- Rissanen, *Information and Complexity in Statistical Modeling*.
- Grünwald, *The Minimum Description Length Principle*.
- MacKay, *Information Theory, Inference and Learning Algorithms*.
- Solomon, *Data Compression: The Complete Reference*.

What is Information?

- Etymology: *informare* = give form, 14th century.

What is Information?

- Etymology: *informare* = give form, 14th century.
- *knowledge [...], intelligence, news, facts, data, [...], (as nucleotides in DNA or binary digits in a computer program) [...], a signal [...], a numerical quantity that measures the uncertainty in the outcome of an experiment to be performed.* (source: Merriam-Webster).

What is Information?

- Etymology: *informare* = give form, 14th century.
- *knowledge [...], intelligence, news, facts, data, [...], (as nucleotides in DNA or binary digits in a computer program) [...], a signal [...], a numerical quantity that measures the uncertainty in the outcome of an experiment to be performed.* (source: Merriam-Webster).
- Data $<$ Information $<$ Knowledge.

What is Information?

- Etymology: *informare* = give form, 14th century.
- *knowledge [...], intelligence, news, facts, data, [...], (as nucleotides in DNA or binary digits in a computer program) [...], a signal [...], a numerical quantity that measures the uncertainty in the outcome of an experiment to be performed.* (source: Merriam-Webster).
- Data $<$ Information $<$ Knowledge.
- Information technology.

What is Information?

- Etymology: *informare* = give form, 14th century.
- *knowledge [...], intelligence, news, facts, data, [...], (as nucleotides in DNA or binary digits in a computer program) [...], a signal [...], a numerical quantity that measures the uncertainty in the outcome of an experiment to be performed.* (source: Merriam-Webster).
- Data $<$ Information $<$ Knowledge.
- Information technology.
- Physical information.

What is Information?

- Etymology: *informare* = give form, 14th century.
- *knowledge [...], intelligence, news, facts, data, [...], (as nucleotides in DNA or binary digits in a computer program) [...], a signal [...], a numerical quantity that measures the uncertainty in the outcome of an experiment to be performed.* (source: Merriam-Webster).
- Data $<$ Information $<$ Knowledge.
- Information technology.
- Physical information.
- This course: measuring *the amount* of information in data, and using such measures for automatically buiding *models*.

Why Information?

- The amount of information around us is exploding – internet!

Why Information?

- The amount of information around us is exploding – internet!
- Need to *store, transmit, and process* information efficiently.

Why Information?

- The amount of information around us is exploding – internet!
- Need to *store, transmit, and process* information efficiently.
- Wish to *understand* more and more complex phenomena.

Why Information?

- The amount of information around us is exploding – internet!
- Need to *store, transmit, and process* information efficiently.
- Wish to *understand* more and more complex phenomena.
- Computer science: make things automatic (intelligent).

Information vs. Complexity

Is complexity the same as information?

Information vs. Complexity

Is complexity the same as information?

Is there a lot of *information* in a random string? **No.**

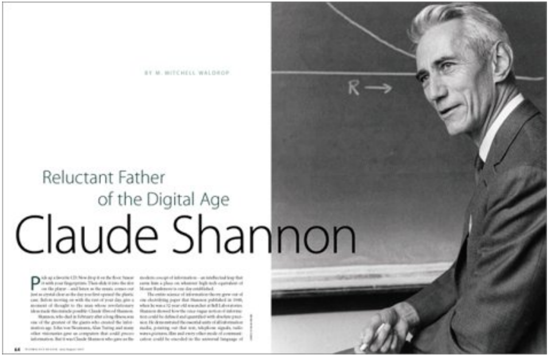
Information vs. Complexity

Is complexity the same as information?

Is there a lot of *information* in a random string? **No.**

$$\begin{aligned} \textit{Complexity} &= \textit{Information} + \textit{Noise} \\ &= \textit{Regularity} + \textit{Randomness} \\ &= \textit{Algorithm} + \textit{Compressed file} \end{aligned}$$

Information Theory



"The real birth of modern information theory can be traced to the publication in 1948 of Claude Shannon's *"The Mathematical Theory of Communication"* in the Bell System Technical Journal. "
(Encyclopædia Britannica)

Course Topics

Information Theory:

- entropy and information, bits,

Course Topics

Information Theory:

- entropy and information, bits,
- compression,

Course Topics

Information Theory:

- entropy and information, bits,
- compression,
- error correction.

Course Topics

Information Theory:

- entropy and information, bits,
- compression,
- error correction.

Fundamental limits (mathematical and statistical) and practice (computer science).

Course Topics

Information Theory:

- entropy and information, bits,
- compression,
- error correction.

Fundamental limits (mathematical and statistical) and practice (computer science).

Modeling:

Course Topics

Information Theory:

- entropy and information, bits,
- compression,
- error correction.

Fundamental limits (mathematical and statistical) and practice (computer science).

Modeling:

- statistical models,

Course Topics

Information Theory:

- entropy and information, bits,
- compression,
- error correction.

Fundamental limits (mathematical and statistical) and practice (computer science).

Modeling:

- statistical models,
- complexity (in data and models),

Course Topics

Information Theory:

- entropy and information, bits,
- compression,
- error correction.

Fundamental limits (mathematical and statistical) and practice (computer science).

Modeling:

- statistical models,
- complexity (in data and models),
- over-fitting, Occam's Razor, and MDL Principle.

1. Course Outline
2. What is Coding?
   - Dots and Dashes
   - Codes as Mappings
   - Data Compression
3. Symbol Codes
4. Entropy and Information
5. Kolmogorov complexity

Coding Game

Form groups of 3–4 persons. Each group constructs a *code* for the letters A–Z by using as *code-words* unique sequences of dots • and dashes (—) like "•", "— •", " — • — —", etc.

A	_______	G	_______	M	_______	S	_______	Y	_______
B	_______	H	_______	N	_______	T	_______	Z	_______
C	_______	I	_______	O	_______	U	_______		
D	_______	J	_______	P	_______	V	_______		
E	_______	K	_______	Q	_______	W	_______		
F	_______	L	_______	R	_______	X	_______		

Coding Game

Use your code to *encode* the message
"`WHAT DOES THIS HAVE TO DO WITH INFORMATION`".

Coding Game

Use your code to *encode* the message
"`WHAT DOES THIS HAVE TO DO WITH INFORMATION`".

Now count how long the encoded message is using the rule:

- A dot •: 1 units.
- A dash —: 2 units.
- A space between words: 2 units.

Coding Game

Use your code to *encode* the message
"`WHAT DOES THIS HAVE TO DO WITH INFORMATION`".

Now count how long the encoded message is using the rule:

- A dot •: 1 units.
- A dash —: 2 units.
- A space between words: 2 units.

• • • — — — • • •: $1+1+1+2+2+2+1+1+1=12$.

Coding Game

Use your code to *encode* the message
"WHAT DOES THIS HAVE TO DO WITH INFORMATION".

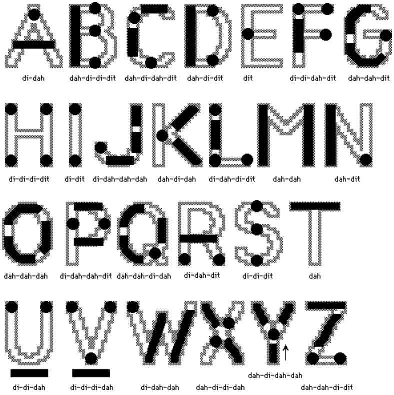
Now count how long the encoded message is using the rule:

- A dot $\bullet$: 1 units.
- A dash —: 2 units.
- A space between words: 2 units.

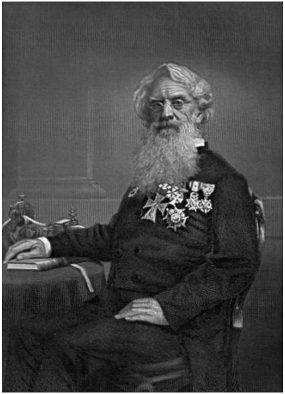
$\bullet\bullet\bullet$ — — — $\bullet\bullet\bullet$: $1+1+1+2+2+2+1+1+1=12$.

The *coding rate* of your code is the length of the encoded message divided by the length of the original message, including spaces (42).

Coding Game



© 1989 A.G. Reinhold.



Samuel F.M. Morse (1791–1872)

Coding Game

`WHAT DOES THIS HAVE TO DO WITH INFORMATION`

Coding Game

```
WHAT DOES THIS HAVE TO DO WITH INFORMATION

.-- .... .- -   -.. --- . ...   - .... .. ...
.... .- ...- .   - ---   -.. ---   .-- .. - ....
.. -. ..-. --- .-. -- .- - .. --- -.
```

Coding Game

```
WHAT DOES THIS HAVE TO DO WITH INFORMATION
```

```
.-- .... .- -   -.. --- . ...   - .... .. ...
.... .- ...- .   - ---   -.. ---   .-- .. - ....
.. -. ..-. --- .-. -- .- - .. --- -.
```

51 dots, 36 dashes, 7 spaces: $51 + 72 + 14 = 137$ units.

Coding Game

```
WHAT DOES THIS HAVE TO DO WITH INFORMATION
```

```
.-- .... .- -   -.. --- . ...   - .... .. ...
.... .- ...- .   - ---   -.. ---   .-- .. - ....
.. -. ..-. --- .-. -- .- - .. --- -.
```

51 dots, 36 dashes, 7 spaces: $51 + 72 + 14 = 137$ units.

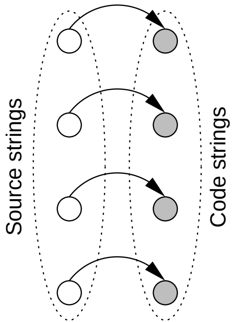
Morse code

Coding rate: $\dfrac{137}{42} \approx 3.26$

Did you do better or worse? Why?

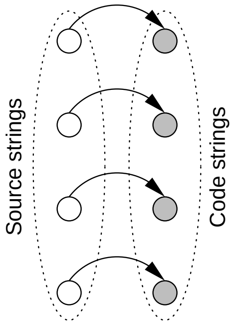
Codes as Mappings

Lossless compression:
injective mapping

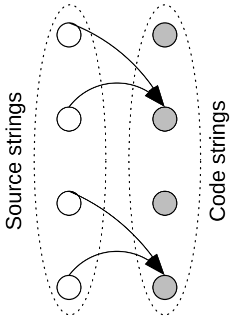


Codes as Mappings

Lossless compression:
injective mapping

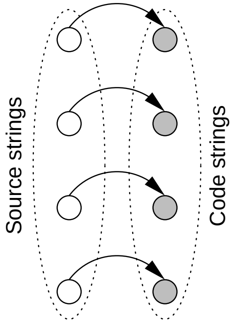


Lossy compression:
non-injective mapping

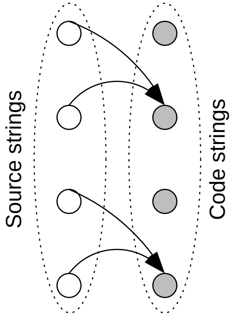


Codes as Mappings

Lossless compression:
injective mapping



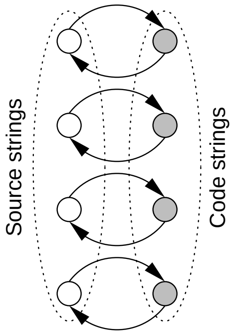
Lossy compression:
non-injective mapping



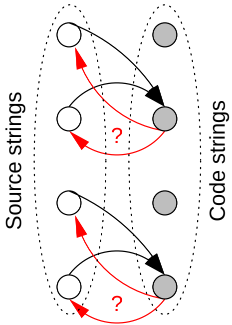
Only *lossless* codes are *uniquely decodable*.

Codes as Mappings

Lossless compression:
injective mapping



Lossy compression:
non-injective mapping



Only *lossless* codes are *uniquely decodable*.

Examples

general purpose	**`gzip`**
	`bzip`

Examples

general purpose	**`gzip`**
	`bzip`
image	**`png`**
	`jpeg`

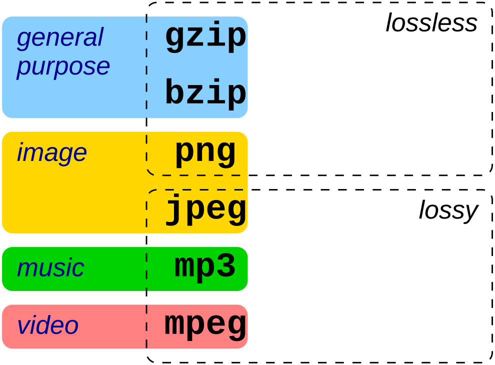
Examples

general purpose	**`gzip`**
	`bzip`
image	**`png`**
	`jpeg`
music	**`mp3`**

Examples

general purpose	**gzip** **bzip**
image	**png** **jpeg**
music	**mp3**
video	**mpeg**

Examples


general purpose
gzip
bzip
lossless
image
png
jpeg
lossy
music
mp3
video
mpeg

Examples

		compression ratio	
general purpose	**gzip**	~ 1 : 3	*lossless*
	bzip	~ 1 : 3.5	*lossless*
image	**png**	~ 1 : 2.5	*lossless*
	jpeg	~ 1 : 25	*lossy*
music	**mp3**	~ 1 : 12	*lossy*
video	**mpeg**	~ 1 : 30	*lossy*

Compression

Is it always possible to compress data?

Theorem

The proportion of binary strings compressible by more than k bits is less than 2^{-k}.

Compression

Is it always possible to compress data?

Theorem

The proportion of binary strings compressible by more than k bits is less than 2^{-k}.

Proof. For all $n \geq 1$, the number of binary strings of length n is 2^n.

Compression

Is it always possible to compress data?

Theorem

The proportion of binary strings compressible by more than k bits is less than 2^{-k}.

Proof. For all $n \geq 1$, the number of binary strings of length n is 2^n.The number of binary code strings of length less than $n - k$ is $2^0 + 2^1 + 2^2 + \ldots + 2^{n-k-1}$

Compression

Is it always possible to compress data?

Theorem

The proportion of binary strings compressible by more than k bits is less than 2^{-k}.

Proof. For all $n \geq 1$, the number of binary strings of length n is 2^n.The number of binary code strings of length less than $n-k$ is $2^0 + 2^1 + 2^2 + \ldots + 2^{n-k-1} = 2^{n-k} - 1$.

Compression

Is it always possible to compress data?

Theorem

The proportion of binary strings compressible by more than k bits is less than 2^{-k}.

Proof. For all $n \geq 1$, the number of binary strings of length n is 2^n.The number of binary code strings of length less than $n-k$ is $2^0 + 2^1 + 2^2 + \ldots + 2^{n-k-1} = 2^{n-k} - 1$. Thus the ratio is

$$\frac{2^{n-k}-1}{2^n} < \frac{2^{n-k}}{2^n} = 2^{-k}.$$

$\square$

Compression

Is it always possible to compress data?

Theorem

The proportion of binary strings compressible by more than k bits is less than 2^{-k}.

Proof. For all $n \geq 1$, the number of binary strings of length n is 2^n.The number of binary code strings of length less than $n-k$ is $2^0 + 2^1 + 2^2 + \ldots + 2^{n-k-1} = 2^{n-k} - 1$. Thus the ratio is

$$\frac{2^{n-k}-1}{2^n} < \frac{2^{n-k}}{2^n} = 2^{-k}.$$

□

Less than 50 % of files are compressible by more than one bit.

Compression

Is it always possible to compress data?

Theorem

The proportion of binary strings compressible by more than k bits is less than 2^{-k}.

Proof. For all $n \geq 1$, the number of binary strings of length n is 2^n.The number of binary code strings of length less than $n-k$ is $2^0 + 2^1 + 2^2 + \ldots + 2^{n-k-1} = 2^{n-k} - 1$. Thus the ratio is

$$\frac{2^{n-k}-1}{2^n} < \frac{2^{n-k}}{2^n} = 2^{-k}.$$

□

Less than 1 % of files are compressible by more than 7 bits.

Compression

Is it always possible to compress data?

Theorem

The proportion of binary strings compressible by more than k bits is less than 2^{-k}.

Proof. For all $n \geq 1$, the number of binary strings of length n is 2^n.The number of binary code strings of length less than $n-k$ is $2^0 + 2^1 + 2^2 + \ldots + 2^{n-k-1} = 2^{n-k} - 1$. Thus the ratio is

$$\frac{2^{n-k} - 1}{2^n} < \frac{2^{n-k}}{2^n} = 2^{-k}.$$

□

Less than 0.0000000000000000000000000001 % of files are compressible by 100 bits.

Symbol Codes

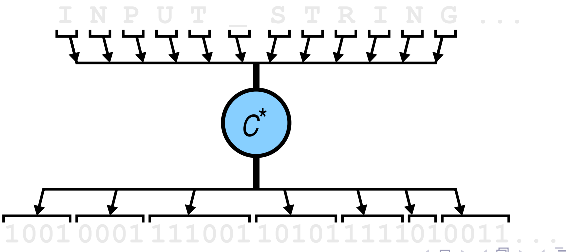
A (binary) **symbol code** $C : \mathcal{X} \rightarrow \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

Symbol Codes

A (binary) **symbol code** $C : \mathcal{X} \to \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

The **extension** of code C is the mapping $C^* : \mathcal{X}^* \to \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each source symbol x_i:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) \ .$$

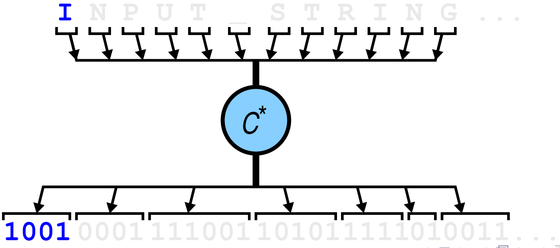


Symbol Codes

A (binary) **symbol code** $C : \mathcal{X} \to \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

The **extension** of code C is the mapping $C^* : \mathcal{X}^* \to \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each source symbol x_i:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) \ .$$

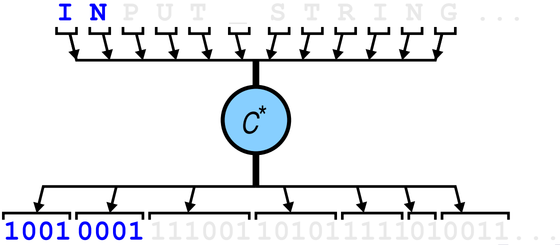


Symbol Codes

A (binary) **symbol code** $C : \mathcal{X} \to \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

The **extension** of code C is the mapping $C^* : \mathcal{X}^* \to \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each source symbol x_i:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) \ .$$

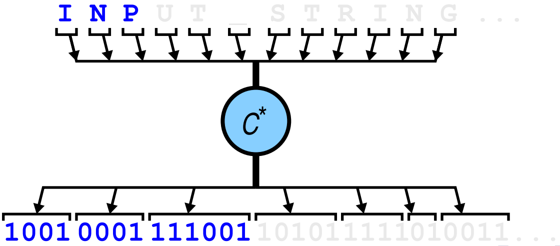


Symbol Codes

A (binary) **symbol code** $C : \mathcal{X} \rightarrow \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

The **extension** of code C is the mapping $C^* : \mathcal{X}^* \rightarrow \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each source symbol x_i:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) \quad .$$

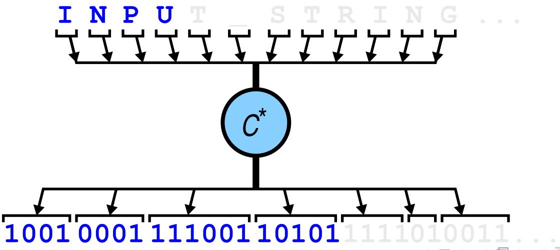


Symbol Codes

A (binary) **symbol code** $C : \mathcal{X} \to \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

The **extension** of code C is the mapping $C^* : \mathcal{X}^* \to \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each source symbol x_i:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) \quad .$$

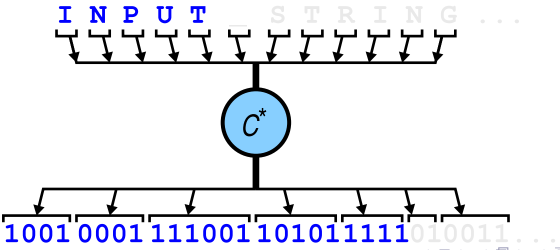


Symbol Codes

A (binary) **symbol code** $C : \mathcal{X} \rightarrow \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

The **extension** of code C is the mapping $C^* : \mathcal{X}^* \rightarrow \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each source symbol x_i:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) \quad .$$

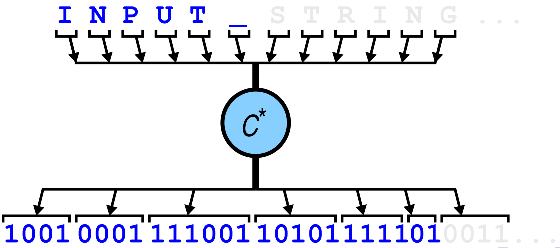


Symbol Codes

A (binary) **symbol code** $C : \mathcal{X} \to \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

The **extension** of code C is the mapping $C^* : \mathcal{X}^* \to \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each source symbol x_i:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) \quad .$$

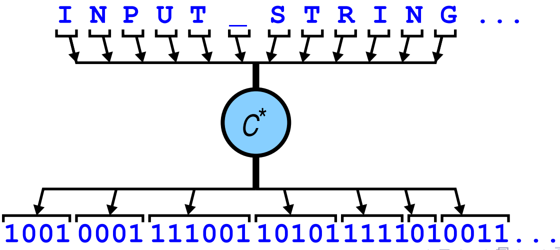


Symbol Codes

A (binary) **symbol code** $C : \mathcal{X} \to \{0,1\}^*$ is a mapping from the alphabet $\mathcal{X}$ to the set of finite binary sequences.

The **extension** of code C is the mapping $C^* : \mathcal{X}^* \to \{0,1\}^*$ obtained by concatenating the codewords $C(x_i)$ for each source symbol x_i:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) \ .$$



Decodable Codes

Decodable Code

Code C is (uniquely) **decodable** iff its extension C^* is a one-to-one mapping, i.e., iff

$$(x_1, \ldots, x_n) \neq (y_1, \ldots, y_n) \ \Rightarrow \ C^*(x_1, \ldots, x_n) \neq C^*(y_1, \ldots, y_n) \ .$$

Decodable Codes

Decodable Code

Code C is (uniquely) **decodable** iff its extension C^* is a one-to-one mapping, i.e., iff

$$(x_1, \ldots, x_n) \neq (y_1, \ldots, y_n) \;\Rightarrow\; C^*(x_1, \ldots, x_n) \neq C^*(y_1, \ldots, y_n) \; .$$

✗ A code with codewords $\{0, 1, 10, 11\}$ is *not* uniquely decodable: What does 10 mean?

Decodable Codes

Decodable Code

Code C is (uniquely) **decodable** iff its extension C^* is a one-to-one mapping, i.e., iff

$$(x_1, \ldots, x_n) \neq (y_1, \ldots, y_n) \;\Rightarrow\; C^*(x_1, \ldots, x_n) \neq C^*(y_1, \ldots, y_n) \; .$$

- ✗ A code with codewords $\{0, 1, 10, 11\}$ is *not* uniquely decodable: What does 10 mean?
- √ A code with codewords $\{00, 01, 10, 11\}$ *is* uniquely decodable: Each pair of bits can be decoded individually.

Decodable Codes

Decodable Code

Code C is (uniquely) **decodable** iff its extension C^* is a one-to-one mapping, i.e., iff

$$(x_1, \ldots, x_n) \neq (y_1, \ldots, y_n) \;\Rightarrow\; C^*(x_1, \ldots, x_n) \neq C^*(y_1, \ldots, y_n) \; .$$

- X A code with codewords $\{0, 1, 10, 11\}$ is *not* uniquely decodable: What does 10 mean?
- √ A code with codewords $\{00, 01, 10, 11\}$ *is* uniquely decodable: Each pair of bits can be decoded individually.
- √ A code with codewords $\{0, 01, 011, 0111\}$ is also uniquely decodable: What does 0011 mean?

Prefix Codes

An important subset of decodable codes is the set of **prefix(-free) codes**.

Prefix Code

A code $C : \mathcal{X} \to \{0, 1\}^*$ is called a **prefix code** iff no codeword is a prefix of another.

It is easily seen that all prefix codes are uniquely decodable: each symbol can be decoded as soon as its codeword is read. Therefore, prefix codes are also called *instantaneous* codes.

Prefix Codes

An important subset of decodable codes is the set of **prefix(-free) codes**.

Prefix Code

A code $C : \mathcal{X} \to \{0,1\}^*$ is called a **prefix code** iff no codeword is a prefix of another.

It is easily seen that all prefix codes are uniquely decodable: each symbol can be decoded as soon as its codeword is read. Therefore, prefix codes are also called *instantaneous* codes.

X A code with codewords $\{0, 01, 011, 0111\}$ is uniquely decodable *but not prefix-free*: e.g., 0 is a prefix of 01.

Prefix Codes

An important subset of decodable codes is the set of **prefix(-free) codes**.

Prefix Code

A code $C : \mathcal{X} \to \{0,1\}^*$ is called a **prefix code** iff no codeword is a prefix of another.

It is easily seen that all prefix codes are uniquely decodable: each symbol can be decoded as soon as its codeword is read. Therefore, prefix codes are also called *instantaneous* codes.

- ✗ A code with codewords $\{0, 01, 011, 0111\}$ is uniquely decodable *but not prefix-free*: e.g., 0 is a prefix of 01.
- √ A code with codewords $\{0, 10, 110, 111\}$ *is* prefix-free.

Kraft Inequality

The codeword lengths of a prefix codes satisfy the following important property.

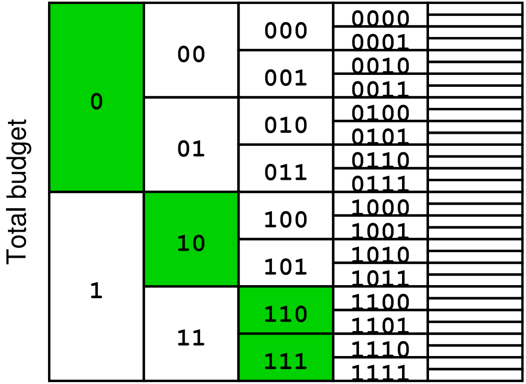
Kraft Inequality

The codeword lengths $\ell_1, \ldots, \ell_m$ of any (binary) prefix code satisfy

$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1 \quad .$$

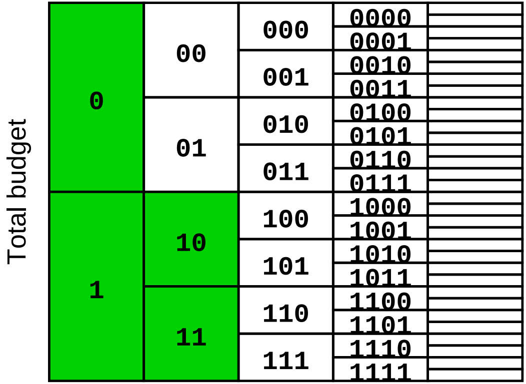
Conversely, given a set of codeword lengths that satisfy this inequality, there is a prefix code with these codeword lengths.

Kraft Inequality



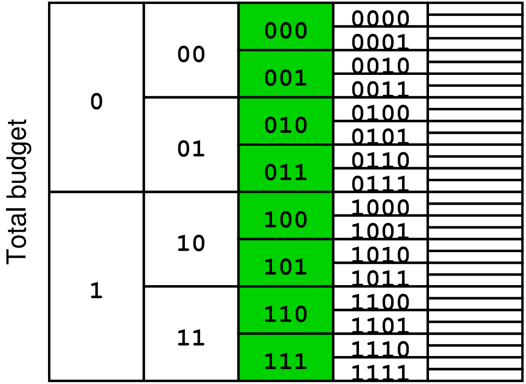
$\surd$ Codewords $\{0, 10, 110, 111\}$

Kraft Inequality



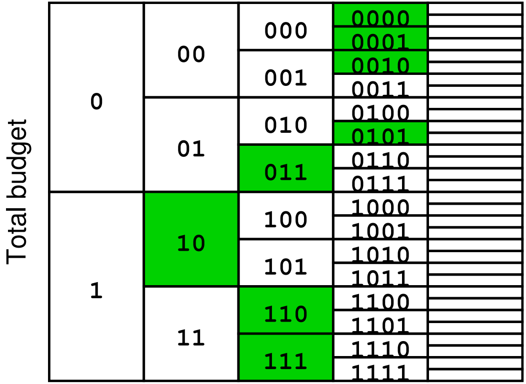
X Kraft inequality violated. $\Rightarrow$ Not decodable.

Kraft Inequality



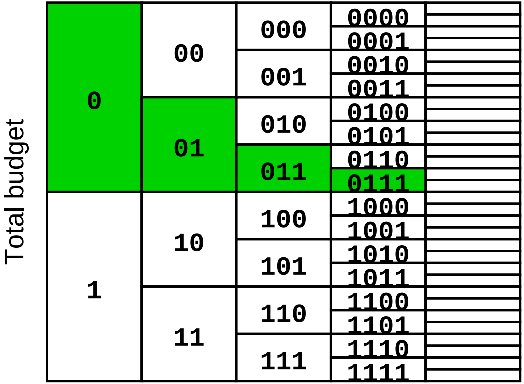
√ Fixed-length code

Kraft Inequality



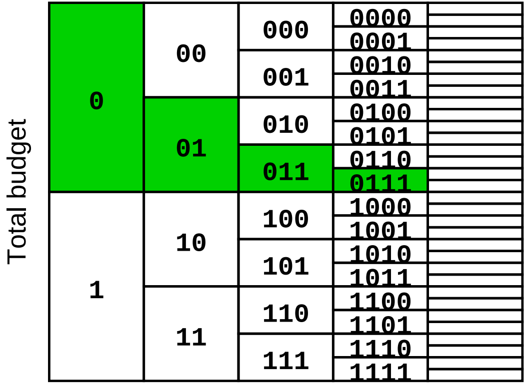
√ Decodable & prefix-free

Kraft Inequality



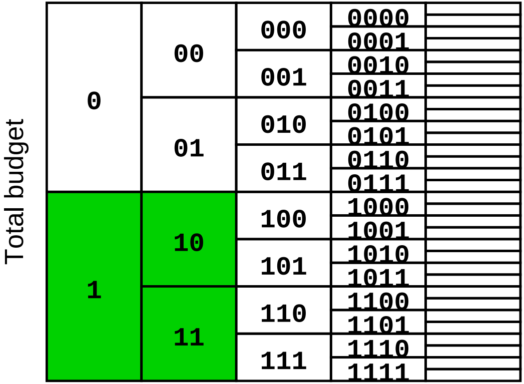
Kraft? Decodable? Prefix-free?

Kraft Inequality



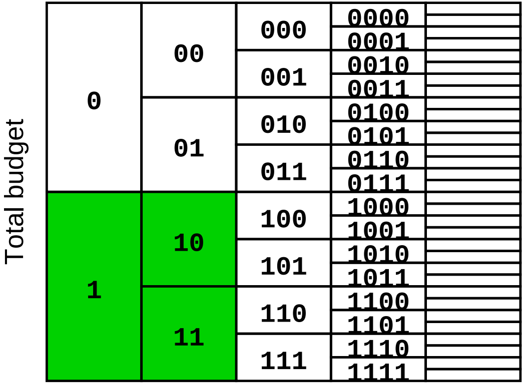
Kraft? √ Decodable? √ Prefix-free? X

Kraft Inequality



Kraft? Decodable? Prefix-free?

Kraft Inequality



Kraft? $\surd$ Decodable? X Prefix-free? X

Kraft Inequality

Question: What if the inequality is satisfied strictly, i.e., the sum of the terms in the sum equals *less* than one:

$$\sum_{i=1}^{m} 2^{-\ell_i} < 1 \quad .$$

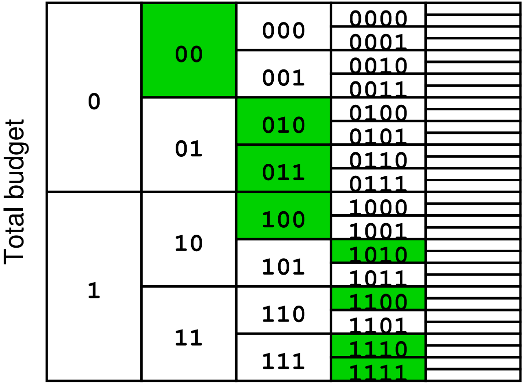
Kraft Inequality

Question: What if the inequality is satisfied strictly, i.e., the sum of the terms in the sum equals *less* than one:

$$\sum_{i=1}^{m} 2^{-\ell_i} < 1 \ .$$

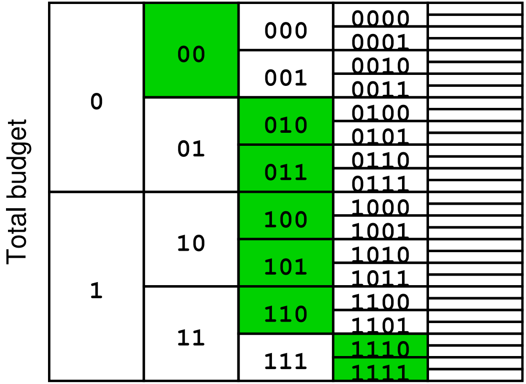
Then it is possible to make the codewords shorter and still have a decodable (prefix) code.

Kraft Inequality



Not all of budget used. $\Rightarrow$ Some codewords can be made shorter.

Kraft Inequality



"Kraft tight" / complete code.

Kraft–McMillan Theorem

The Kraft inequality restricts the codeword lengths of prefix codes. Could we do much better if we would only require decodability?

Kraft–McMillan Theorem

The Kraft inequality restricts the codeword lengths of prefix codes. Could we do much better if we would only require decodability?

In fact it can be shown that we do not lose anything at all!

Kraft–McMillan Theorem

The Kraft inequality restricts the codeword lengths of prefix codes. Could we do much better if we would only require decodability?

In fact it can be shown that we do not lose anything at all!

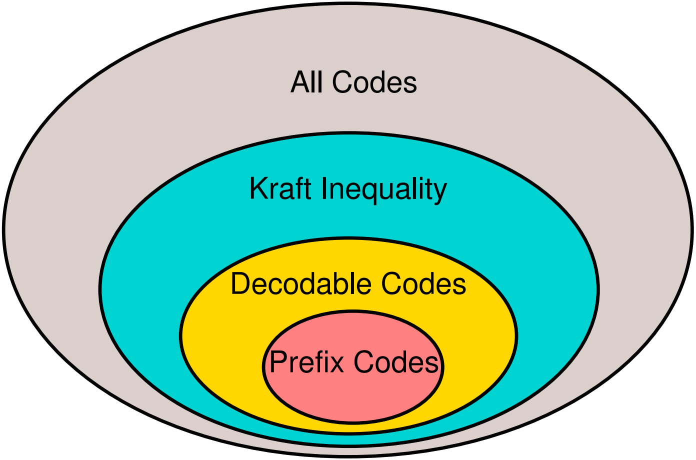
Kraft-McMillan Theorem

The codeword lengths $\ell_1, \ldots, \ell_m$ of any **uniquely decodable** (binary) code satisfy

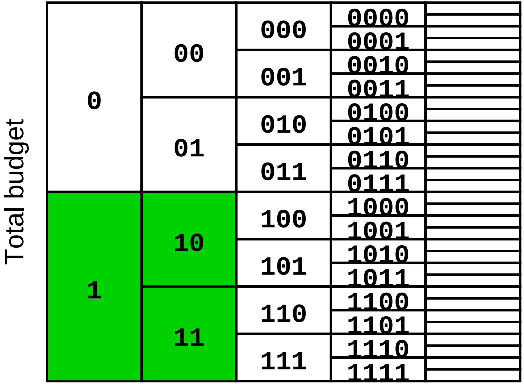
$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1 \ .$$

Conversely, given a set of codeword lengths that satisfy this inequality, there is a **prefix** code with these codeword lengths.

Kraft-McMillan Theorem & Codes

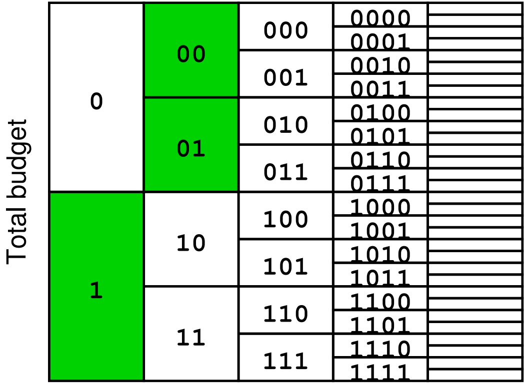


Kraft Inequality



Kraft? √ Decodable? X Prefix-free? X

Kraft Inequality



Kraft? √ Decodable? √ Prefix-free? √

Expected Code-length

Now we can tell which codes are decodable, prefix-free, etc.

The next question to answer is:

Out of two decodable (prefix-free) codes, which one is better?

Expected Code-length

Now we can tell which codes are decodable, prefix-free, etc.

The next question to answer is:

**Out of two decodable (prefix-free) codes,
which one is better?**

For the purpose of **data compression**, the answer is clearly the code that yields the **shortest code-length**.

Expected Code-length

Now we can tell which codes are decodable, prefix-free, etc.

The next question to answer is:

Out of two decodable (prefix-free) codes, which one is better?

For the purpose of **data compression**, the answer is clearly the code that yields the **shortest code-length**.

We consider the expected (per-symbol) code-length:

$$E[\ell(C(X))] = \sum_{x \in \mathcal{X}} p(x)\,\ell(C(x)) \ .$$

Expected Code-length

To study the expected code-length, it is useful to define

$$q(x) = 2^{-\ell(C(x))}$$

Expected Code-length

To study the expected code-length, it is useful to define

$$q(x) = 2^{-\ell(C(x))} \quad \Leftrightarrow \quad \ell(C(x)) = -\log_2 q(x) = \log_2 \frac{1}{q(x)} \ .$$

Expected Code-length

To study the expected code-length, it is useful to define

$$q(x) = 2^{-\ell(C(x))} \quad \Leftrightarrow \quad \ell(C(x)) = -\log_2 q(x) = \log_2 \frac{1}{q(x)} \ .$$

The Kraft-(in)equality implies that

$$\sum_{x \in \mathcal{X}} q(x) \leq 1 \ .$$

Expected Code-length

To study the expected code-length, it is useful to define

$$q(x) = 2^{-\ell(C(x))} \quad \Leftrightarrow \quad \ell(C(x)) = -\log_2 q(x) = \log_2 \frac{1}{q(x)} \ .$$

The Kraft-(in)equality implies that

$$\sum_{x \in \mathcal{X}} q(x) = 1 \ .$$

Expected Code-length

To study the expected code-length, it is useful to define

$$q(x) = 2^{-\ell(C(x))} \quad \Leftrightarrow \quad \ell(C(x)) = -\log_2 q(x) = \log_2 \frac{1}{q(x)} \ .$$

The Kraft-(in)equality implies that

$$\sum_{x \in \mathcal{X}} q(x) = 1 \ .$$

Important Observation

Probability distributions are codes are probability distributions!

Expected Code-length

Based on the unification of codes and distributions, we can write

$$E[\ell(C(X))] = \sum_{x \in \mathcal{X}} p(x)\, \ell(C(x))$$

Expected Code-length

Based on the unification of codes and distributions, we can write

$$\begin{aligned} E[\ell(C(X))] &= \sum_{x \in \mathcal{X}} p(x)\, \ell(C(x)) \\ &= \sum_{x \in \mathcal{X}} p(x) \log_2 \frac{1}{q(x)} \ , \end{aligned}$$

where $q(x) = 2^{-\ell(C(x))}$.

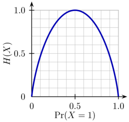
Expected Code-length

Based on the unification of codes and distributions, we can write

$$E[\ell(C(X))] = \sum_{x \in \mathcal{X}} p(x)\, \ell(C(x))$$
$$= \sum_{x \in \mathcal{X}} p(x) \log_2 \frac{1}{q(x)} \ ,$$

where $q(x) = 2^{-\ell(C(x))}$.

$\Rightarrow$ Information theory (entropy, Kullback-Leibler divergence, ...)


1.0
0.5
0
$H(X)$
0
0.5
1.0
$\Pr(X = 1)$

Entropy

Given a discrete random variable X with pmf p_X, we can measure the amount of "surprise" associated with each outcome $x \in \mathcal{X}$ by the quantity

$$I_X(x) = \log_2 \frac{1}{p_X(x)} \quad .$$

The less likely an outcome is, the more surprised we are to observe it. (The point in the log-scale will become clear shortly.)

Entropy

Given a discrete random variable X with pmf p_X, we can measure the amount of "surprise" associated with each outcome $x \in \mathcal{X}$ by the quantity

$$I_X(x) = \log_2 \frac{1}{p_X(x)} \quad .$$

The less likely an outcome is, the more surprised we are to observe it. (The point in the log-scale will become clear shortly.)

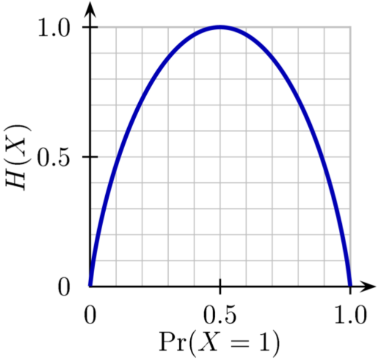
The **entropy** of X measures the *expected* amount of "surprise":

$$H(X) = E[I_X(X)] = \sum_{x \in \mathcal{X}} p_X(x) \log_2 \frac{1}{p_X(x)} \quad .$$

Binary Entropy Function

For binary-valued X, with $p = p_X(1) = 1 - p_X(0)$, we have

$$H(X) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p} \ .$$



More Entropies

1. the **joint entropy** of two (or more) random variables:

$$H(X,Y) = \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{X,Y}(x,y) \log_2 \frac{1}{p_{X,Y}(x,y)} \ ,$$

More Entropies

1. the **joint entropy** of two (or more) random variables:

$$H(X, Y) = \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{X,Y}(x, y) \log_2 \frac{1}{p_{X,Y}(x, y)} \quad ,$$

2. the **entropy of a conditional distribution**:

$$H(X \mid Y = y) = \sum_{x \in \mathcal{X}} p_{X|Y}(x \mid y) \log_2 \frac{1}{p_{X|Y}(x \mid y)} \quad ,$$

More Entropies

1. the **joint entropy** of two (or more) random variables:

$$H(X, Y) = \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{X,Y}(x, y) \log_2 \frac{1}{p_{X,Y}(x, y)} \ ,$$

2. the **entropy of a conditional distribution**:

$$H(X \mid Y = y) = \sum_{x \in \mathcal{X}} p_{X|Y}(x \mid y) \log_2 \frac{1}{p_{X|Y}(x \mid y)} \ ,$$

3. and the **conditional entropy**:

$$H(X \mid Y) = \sum_{y \in \mathcal{Y}} p(y) \, H(X \mid Y = y) \ .$$

More Entropies

The joint entropy $H(X, Y)$ measures the uncertainty about the pair (X, Y).

More Entropies

The joint entropy $H(X, Y)$ measures the uncertainty about the pair (X, Y).

The entropy of the conditional distribution $H(X \mid Y = y)$ measures the uncertainty about X when we know that $Y = y$.

More Entropies

The joint entropy $H(X, Y)$ measures the uncertainty about the pair (X, Y).

The entropy of the conditional distribution $H(X \mid Y = y)$ measures the uncertainty about X when we know that $Y = y$.

The conditional entropy $H(X \mid Y)$ measures the *expected* uncertainty about X when the value Y is known.

Chain Rule of Entropy

Remember the chain rule of probability:

$$p_{X,Y}(x, y) = p_Y(y) \times p_{X|Y}(x \mid y) \ .$$

Chain Rule of Entropy

Remember the chain rule of probability:

$$p_{X,Y}(x,y) = p_Y(y) \times p_{X|Y}(x \mid y) \ .$$

For the entropy we have:

Chain Rule of Entropy

$$H(X,Y) = H(Y) + H(X \mid Y) \ .$$

Chain Rule of Entropy

Remember the chain rule of probability:

$$p_{X,Y}(x,y) = p_Y(y) \times p_{X|Y}(x \mid y) \ .$$

For the entropy we have:

Chain Rule of Entropy

$$H(X,Y) = H(Y) + H(X \mid Y) \ .$$

$$X \perp\!\!\!\perp Y \ \Leftrightarrow \ H(X \mid Y) = H(X) \ \Leftrightarrow \ H(X,Y) = H(X) + H(Y).$$

Chain Rule of Entropy

Remember the chain rule of probability:

$$p_{X,Y}(x,y) = p_Y(y) \times p_{X|Y}(x \mid y) \ .$$

For the entropy we have:

Chain Rule of Entropy

$$H(X,Y) = H(Y) + H(X \mid Y) \ .$$

$$X \perp\!\!\!\perp Y \ \Leftrightarrow \ H(X \mid Y) = H(X) \ \Leftrightarrow \ H(X,Y) = H(X) + H(Y).$$

Logarithmic scale makes entropy **additive**.

Mutual Information

The **mutual information**

$$I(X\,;\,Y) = H(X) - H(X \mid Y)$$

measures the average decrease in uncertainty about X when the value of Y becomes known.

Mutual Information

The **mutual information**

$$I(X\,;\,Y) = H(X) - H(X \mid Y)$$

measures the average decrease in uncertainty about X when the value of Y becomes known.

Mutual information is *symmetric* (chain rule):

$$I(X\,;\,Y) = H(X) - H(X \mid Y) = H(X) - (H(X, Y) - H(Y))$$

Mutual Information

The **mutual information**

$$I(X\,;\,Y) = H(X) - H(X \mid Y)$$

measures the average decrease in uncertainty about X when the value of Y becomes known.

Mutual information is *symmetric* (chain rule):

$$I(X\,;\,Y) = H(X) - H(X \mid Y) = H(X) - H(X, Y) + H(Y)$$

Mutual Information

The **mutual information**

$$I(X\,;\,Y) = H(X) - H(X \mid Y)$$

measures the average decrease in uncertainty about X when the value of Y becomes known.

Mutual information is *symmetric* (chain rule):

$$I(X\,;\,Y) = H(X) - H(X \mid Y) = (H(X) - H(X,Y)) + H(Y)$$

Mutual Information

The **mutual information**

$$I(X\,;\,Y) = H(X) - H(X \mid Y)$$

measures the average decrease in uncertainty about X when the value of Y becomes known.

Mutual information is *symmetric* (chain rule):

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

Mutual Information

The **mutual information**

$$I(X\,;\,Y) = H(X) - H(X \mid Y)$$

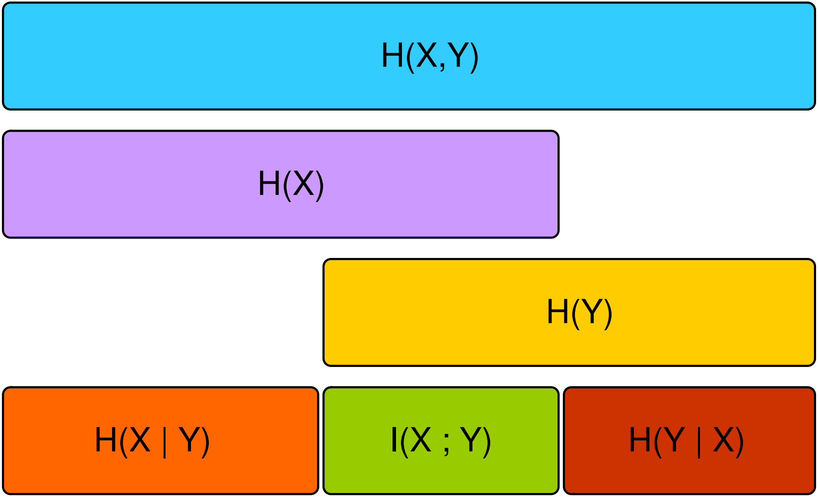
measures the average decrease in uncertainty about X when the value of Y becomes known.

Mutual information is *symmetric* (chain rule):

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

On the average, X gives as much information about Y as Y gives about X.

Relationships between Entropies



Time for a break?

Kullback-Leibler Divergence

Kullback-Leibler Divergence

The *relative entropy* or **Kullback-Leibler divergence** between (discrete) distributions p_X and q_X is defined as

$$D(p_X \parallel q_X) = \sum_{x \in \mathcal{X}} p_X(x) \log_2 \frac{p_X(x)}{q_X(x)} \quad .$$

Kullback-Leibler Divergence

Kullback-Leibler Divergence

The *relative entropy* or **Kullback-Leibler divergence** between (discrete) distributions p_X and q_X is defined as

$$D(p_X \parallel q_X) = \sum_{x \in \mathcal{X}} p_X(x) \log_2 \frac{p_X(x)}{q_X(x)} \quad .$$

(We consider $p_X(x) \log_2 \frac{p_X(x)}{q_X(x)} = 0$ whenever $p_X(x) = 0$.)

Kullback-Leibler Divergence

Kullback-Leibler Divergence

The *relative entropy* or **Kullback-Leibler divergence** between (discrete) distributions p_X and q_X is defined as

$$D(p_X \parallel q_X) = \sum_{x \in \mathcal{X}} p_X(x) \log_2 \frac{p_X(x)}{q_X(x)} \ .$$

Information Inquality

For any two (discrete) distributions p_X and q_X, we have

$$D(p_X \parallel q_X) \geq 0$$

with equality iff $p_X(x) = q_X(x)$ for all $x \in \mathcal{X}$.

Kullback-Leibler Divergence

The information inequality implies

$$I(X\ ;\ Y) \geq 0\ .$$

Kullback-Leibler Divergence

The information inequality implies

$$I(X\,;\,Y) \geq 0 \enspace .$$

Proof.

$$\begin{aligned} I(X\,;\,Y) &= H(X) - H(X \mid Y) \\ &= H(X) + H(Y) - H(X,Y) \\ &= \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{X,Y}(x,y) \log_2 \frac{p_{X,Y}(x,y)}{p_X(x)\, p_Y(y)} \\ &= D(p_{X,Y} \parallel p_X p_Y) \geq 0 \enspace . \end{aligned}$$

Kullback-Leibler Divergence

The information inequality implies

$$I(X\ ;\ Y) \geq 0\ .$$

Proof.

$$\begin{aligned} I(X\ ;\ Y) &= H(X) - H(X \mid Y) \\ &= H(X) + H(Y) - H(X, Y) \\ &= \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{X,Y}(x, y) \log_2 \frac{p_{X,Y}(x, y)}{p_X(x)\, p_Y(y)} \\ &= D(p_{X,Y} \parallel p_X p_Y) \geq 0\ . \end{aligned}$$

In addition, $D(p_{X,Y} \parallel p_X p_Y) = 0$ iff $p_{X,Y}(x, y) = p_X(x)\, p_Y(y)$ for all $x \in \mathcal{X}, y \in \mathcal{Y}$. This means that variables X and Y are *independent* iff $I(X\ ;\ Y) = 0$.

Properties of Entropy

Properties of entropy:

1. $H(X) \geq 0$

Properties of Entropy

Properties of entropy:

1. $H(X) \geq 0$

   Proof. $p_X(x) \leq 1 \Rightarrow \log_2 \frac{1}{p_X(x)} \geq 0.$

Properties of Entropy

Properties of entropy:

1. $H(X) \geq 0$

   Proof. $p_X(x) \leq 1 \Rightarrow \log_2 \frac{1}{p_X(x)} \geq 0.$

2. $H(X) \leq \log_2 |\mathcal{X}|$

Properties of Entropy

Properties of entropy:

1. $H(X) \geq 0$

   Proof. $p_X(x) \leq 1 \Rightarrow \log_2 \dfrac{1}{p_X(x)} \geq 0.$

2. $H(X) \leq \log_2 |\mathcal{X}|$

   Proof. Let $u_X(x) = \frac{1}{|\mathcal{X}|}$ be the uniform distribution over $\mathcal{X}$.

$$0 \leq D(p_X \parallel u_X) = \sum_{x \in \mathcal{X}} p_X(x) \log_2 \frac{p_X(x)}{u_X(x)} = \log_2 |\mathcal{X}| - H(X) \quad .$$

Properties of Entropy

Properties of entropy:

1. $H(X) \geq 0$

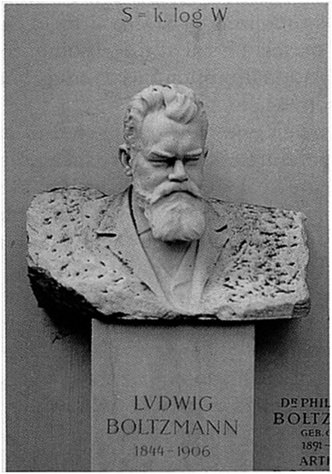
   Proof. $p_X(x) \leq 1 \Rightarrow \log_2 \frac{1}{p_X(x)} \geq 0.$

2. $H(X) \leq \log_2 |\mathcal{X}|$

   A **combinatorial** approach to the definition of information (Boltzmann, 1896; Hartley, 1928; Kolmogorov, 1965):

$$S = k \ln W \quad .$$

Ludvig Boltzmann (1844–1906)



Properties of Entropy

Properties of entropy:

1. $H(X) \geq 0$

   Proof. $p_X(x) \leq 1 \Rightarrow \log_2 \frac{1}{p_X(x)} \geq 0.$

2. $H(X) \leq \log_2 |\mathcal{X}|$

   A **combinatorial** approach to the definition of information (Boltzmann, 1896; Hartley, 1928; Kolmogorov, 1965):

   $$S = k \ln W \quad .$$

3. $H(X \mid Y) \leq H(X)$

Properties of Entropy

Properties of entropy:

1. $H(X) \geq 0$

   Proof. $p_X(x) \leq 1 \Rightarrow \log_2 \frac{1}{p_X(x)} \geq 0.$

2. $H(X) \leq \log_2 |\mathcal{X}|$

   A **combinatorial** approach to the definition of information (Boltzmann, 1896; Hartley, 1928; Kolmogorov, 1965):

   $$S = k \ln W \quad .$$

3. $H(X \mid Y) \leq H(X)$

   Proof.

   $$0 \leq I(X\,;\,Y) = H(X) - H(X \mid Y) \quad .$$

Properties of Entropy

Properties of entropy:

1. $H(X) \geq 0$

   Proof. $p_X(x) \leq 1 \Rightarrow \log_2 \frac{1}{p_X(x)} \geq 0.$

2. $H(X) \leq \log_2 |\mathcal{X}|$

   A **combinatorial** approach to the definition of information (Boltzmann, 1896; Hartley, 1928; Kolmogorov, 1965):

   $$S = k \ln W \quad .$$

3. $H(X \mid Y) \leq H(X)$

   On the average, knowing another r.v. can only reduce uncertainty about X. However, note that $H(X \mid Y = y)$ may be greater than $H(X)$ for some y — "contradicting evidence".

Entropy Lower Bound

Entropy Lower Bound

$$E[\ell(X)] \geq H(X) \quad .$$

Entropy Lower Bound

Entropy Lower Bound

$$E[\ell(X)] \geq H(X) \quad .$$

Proof.

$$E[\ell(X)] = \sum_{x \in \mathcal{X}} p(x)\, \ell(x)$$

Entropy Lower Bound

Entropy Lower Bound

$$E[\ell(X)] \geq H(X) \quad .$$

Proof.

$$E[\ell(X)] = \sum_{x \in \mathcal{X}} p(x)\, \ell(x)$$

$$= \sum_{x \in \mathcal{X}} p(x)\, \log_2 \frac{1}{q(x)} \qquad \boxed{q(x) = 2^{-\ell(x)}}$$

Entropy Lower Bound

Entropy Lower Bound

$$E[\ell(X)] \geq H(X) \quad .$$

Proof.

$$\begin{aligned} E[\ell(X)] &= \sum_{x \in \mathcal{X}} p(x)\,\ell(x) \\ &= \sum_{x \in \mathcal{X}} p(x)\,\log_2 \frac{1}{q(x)} \qquad \boxed{q(x) = 2^{-\ell(x)}} \\ &= \sum_{x \in \mathcal{X}} p(x)\,\left[\log_2 \frac{p(x)}{q(x)} + \log_2 \frac{1}{p(x)}\right] \end{aligned}$$

Entropy Lower Bound

Entropy Lower Bound

$$E[\ell(X)] \geq H(X) \quad .$$

Proof.

$$\begin{aligned} E[\ell(X)] &= \sum_{x\in\mathcal{X}} p(x)\,\ell(x) \\ &= \sum_{x\in\mathcal{X}} p(x)\,\log_2 \frac{1}{q(x)} \qquad \boxed{q(x) = 2^{-\ell(x)}} \\ &= \sum_{x\in\mathcal{X}} p(x)\left[\log_2 \frac{p(x)}{q(x)} + \log_2 \frac{1}{p(x)}\right] \end{aligned}$$

Entropy Lower Bound

Entropy Lower Bound

$$E[\ell(X)] \geq H(X) \quad .$$

Proof.

$$\begin{aligned}
E[\ell(X)] &= \sum_{x \in \mathcal{X}} p(x)\,\ell(x) \\
&= \sum_{x \in \mathcal{X}} p(x) \log_2 \frac{1}{q(x)} \qquad \boxed{q(x) = 2^{-\ell(x)}} \\
&= \sum_{x \in \mathcal{X}} p(x) \left[\log_2 \frac{p(x)}{q(x)} + \log_2 \frac{1}{p(x)}\right] \\
&= D(p \parallel q) + H(X) \geq 0 \quad .
\end{aligned}$$

Entropy Lower Bound

So what have we learned?

Entropy Lower Bound

So what have we learned? For ("Kraft-tight") decodable symbols codes:

1. $E[\ell(X)] = H(X) + D(p \parallel q)$, where $q(x) = 2^{-\ell(x)}$.

Entropy Lower Bound

So what have we learned? For ("Kraft-tight") decodable symbols codes:

1. $E[\ell(X)] = H(X) + D(p \parallel q)$, where $q(x) = 2^{-\ell(x)}$.
2. $E[\ell(X)] \geq H(X)$.

Entropy Lower Bound

So what have we learned? For ("Kraft-tight") decodable symbols codes:

1. $E[\ell(X)] = H(X) + D(p \parallel q)$, where $q(x) = 2^{-\ell(x)}$.
2. $E[\ell(X)] \geq H(X)$.
3. If $\ell(x) = \log_2 \frac{1}{p(x)}$, then $E[\ell(X)] = H(X)$. **Optimal!**

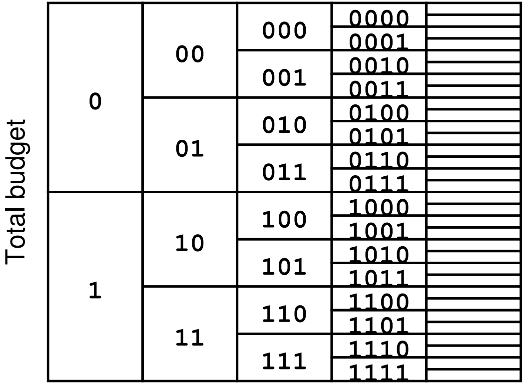
Exercise 5.

5. Let the source distribution p be given by the table below. What are the optimal codeword lengths under p? Can you construct the actual codewords so that the code is prefix-free?

Ex. 5

x	A	B	C	D	E
$p(x)$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$

Exercise 5.


Total budget
0
1
00
01
10
11
000
001
010
011
100
101
110
111
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Codelengths and Probabilities

A problem with the $\ell(x) = \log_2 \frac{1}{p(x)}$ codeword choice is the requirement that codeword lengths must be **integers** (try to think about a codeword with length 0.123, for instance).

Codelengths and Probabilities

A problem with the $\ell(x) = \log_2 \frac{1}{p(x)}$ codeword choice is the requirement that codeword lengths must be **integers** (try to think about a codeword with length 0.123, for instance).

The simplest solution is to round upwards:

Shannon's Code

Given a pmf, the **Shannon code** has the codeword lengths

$$\ell(x) = \left\lceil \log_2 \frac{1}{p(x)} \right\rceil \quad \text{for all } x \in \mathcal{X}.$$

Alice in Wonderland

Shannon's code: Example

X	$p(X)$	$\log_2 \frac{1}{p(X)}$	$\ell(X)$
a	0.0644	3.9	4
b	0.0108	6.5	7
c	0.0178	5.8	6
d	0.0359	4.7	5
e	0.0991	3.3	4
f	0.0147	6.0	7
g	0.0184	5.7	6
h	0.0535	4.2	5
i	0.0551	4.1	5
j	0.0011	9.8	10
k	0.0083	6.8	7
l	0.0343	4.8	5
	⋮		
y	0.0165	5.9	6
z	0.0005	10.7	11
	0.2111	2.2	3

$$H(X) = 4.03$$

Shannon's code: Example

X	$p(X)$	$\log_2 \frac{1}{p(X)}$	$\ell(X)$
a	0.0644	3.9	4
b	0.0108	6.5	7
c	0.0178	5.8	6
d	0.0359	4.7	5
e	0.0991	3.3	4
f	0.0147	6.0	7
g	0.0184	5.7	6
h	0.0535	4.2	5
i	0.0551	4.1	5
j	0.0011	9.8	10
k	0.0083	6.8	7
l	0.0343	4.8	5
	$\vdots$		
y	0.0165	5.9	6
z	0.0005	10.7	11
	0.2111	2.2	3

$H(X) = 4.03$

Shannon (1948):

Shannon's code: Example

X	$p(X)$	$\log_2 \frac{1}{p(X)}$	$\ell(X)$
a	0.0644	3.9	4
b	0.0108	6.5	7
c	0.0178	5.8	6
d	0.0359	4.7	5
e	0.0991	3.3	4
f	0.0147	6.0	7
g	0.0184	5.7	6
h	0.0535	4.2	5
i	0.0551	4.1	5
j	0.0011	9.8	10
k	0.0083	6.8	7
l	0.0343	4.8	5
	⋮		
y	0.0165	5.9	6
z	0.0005	10.7	11
	0.2111	2.2	3

$$H(X) = 4.03$$

Shannon (1948):

1. Sort by probability.

Shannon's code: Example

X	$p(X)$	$\log_2 \frac{1}{p(X)}$	$\ell(X)$
	0.2111	2.2	3
e	0.0991	3.3	4
t	0.0781	3.6	4
a	0.0644	3.9	4
o	0.0598	4.0	5
i	0.0551	4.1	5
h	0.0535	4.2	5
n	0.0516	4.2	5
s	0.0475	4.3	5
r	0.0401	4.6	5
d	0.0359	4.7	5
l	0.0343	4.8	5
	$\vdots$		
x	0.0011	9.8	10
j	0.0011	9.8	10
z	0.0005	10.7	11

$$H(X) = 4.03$$

Shannon (1948):

1. Sort by probability.

Shannon's code: Example

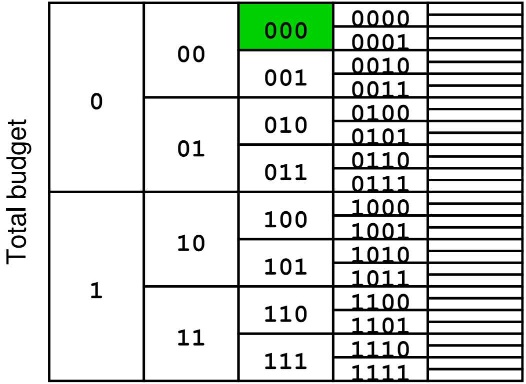
X	$p(X)$	$\log_2 \frac{1}{p(X)}$	$\ell(X)$
	0.2111	2.2	3
e	0.0991	3.3	4
t	0.0781	3.6	4
a	0.0644	3.9	4
o	0.0598	4.0	5
i	0.0551	4.1	5
h	0.0535	4.2	5
n	0.0516	4.2	5
s	0.0475	4.3	5
r	0.0401	4.6	5
d	0.0359	4.7	5
l	0.0343	4.8	5
	⋮		
x	0.0011	9.8	10
j	0.0011	9.8	10
z	0.0005	10.7	11

$$H(X) = 4.03$$

Shannon (1948):

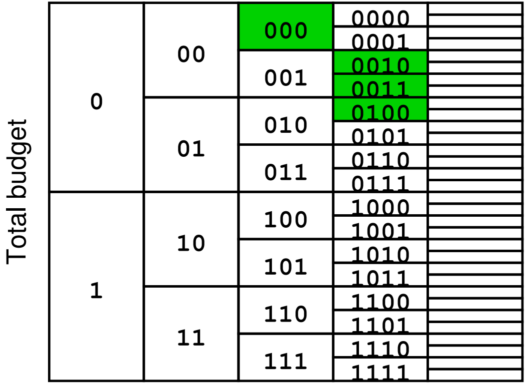
1. Sort by probability.
2. Choose codewords in order, avoiding prefixes. ("Kraft table"!)

Shannon's code: Example



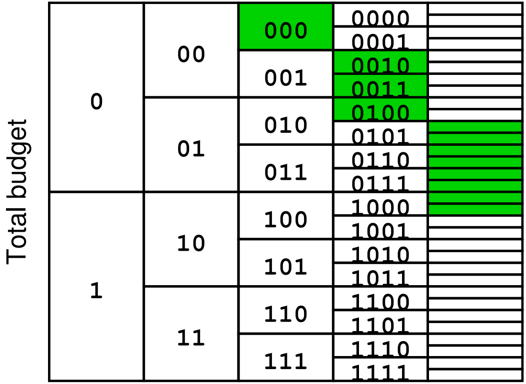
Codeword lengths $(3, 4, 4, 4, 5, 5, 5, 5, \ldots, 10, 10, 11)$

Shannon's code: Example



Codeword lengths $(3, 4, 4, 4, 5, 5, 5, 5, \ldots, 10, 10, 11)$

Shannon's code: Example



Codeword lengths $(3, 4, 4, 4, 5, 5, 5, 5, \ldots, 10, 10, 11)$

Shannon's code: Example

X	$p(X)$	$\log_2 \frac{1}{p(X)}$	$\ell(X)$	$C(X)$
	0.2111	2.2	3	000
e	0.0991	3.3	4	0010
t	0.0781	3.6	4	0011
a	0.0644	3.9	4	0100
o	0.0598	4.0	5	01010
i	0.0551	4.1	5	01011
h	0.0535	4.2	5	01100
n	0.0516	4.2	5	01101
s	0.0475	4.3	5	01110
r	0.0401	4.6	5	01111
d	0.0359	4.7	5	10000
l	0.0343	4.8	5	10001
	⋮			
x	0.0011	9.8	10	1010111101
j	0.0011	9.8	10	1010111110
z	0.0005	10.7	11	10101111110

Shannon's code: Example

X	$p(X)$	$\log_2 \frac{1}{p(X)}$	$\ell(X)$	$C(X)$
	0.2111	2.2	3	000
e	0.0991	3.3	4	0010
t	0.0781	3.6	4	0011
a	0.0644	3.9	4	0100
o	0.0598	4.0	5	01010
i	0.0551	4.1	5	01011
h	0.0535	4.2	5	01100
n	0.0516	4.2	5	01101
s	0.0475	4.3	5	01110
r	0.0401	4.6	5	01111
d	0.0359	4.7	5	10000
l	0.0343	4.8	5	10001
	⋮			
x	0.0011	9.8	10	1010111101
j	0.0011	9.8	10	1010111110
z	0.0005	10.7	11	10101111110

$$H(X) = 4.03$$

$$E[\ell(X)] = 4.60$$

$$E[\ell(X)] - H(X) = 0.57$$

Shannon's code

The expected codeword length of Shannon's code is

$$
\begin{aligned}
E\left[\ell(X)\right] &= E\left[\left\lceil \log_2 \frac{1}{p(X)} \right\rceil\right] \\
&\leq E\left[\log_2 \frac{1}{p(X)} + 1\right] = H(X) + 1 \quad .
\end{aligned}
$$

Shannon's code

The expected codeword length of Shannon's code is

$$E\left[\ell(X)\right] = E\left[\left\lceil \log_2 \frac{1}{p(X)} \right\rceil\right] \leq E\left[\log_2 \frac{1}{p(X)} + 1\right] = H(X) + 1 \quad .$$

In the Alice example we had

$$E[\ell(X)] - H(X) = 4.60 - 4.03 = 0.57 \leq 1 \quad .$$

Shannon's code

The expected codeword length of Shannon's code is

$$\begin{aligned} E\left[\ell(X)\right] &= E\left[\left\lceil \log_2 \frac{1}{p(X)} \right\rceil\right] \\ &\le E\left[\log_2 \frac{1}{p(X)} + 1\right] = H(X) + 1 \ . \end{aligned}$$

In the Alice example we had

$$E[\ell(X)] - H(X) = 4.60 - 4.03 = 0.57 \le 1 \ .$$

Is this optimal?

Shannon's code

The expected codeword length of Shannon's code is

$$E\left[\ell(X)\right] = E\left[\left\lceil \log_2 \frac{1}{p(X)} \right\rceil\right] \leq E\left[\log_2 \frac{1}{p(X)} + 1\right] = H(X) + 1 \ .$$

In the Alice example we had

$$E[\ell(X)] - H(X) = 4.60 - 4.03 = 0.57 \leq 1 \ .$$

Is this optimal? Not necessarily — Huffman!

Kolmogorov complexity

Is the string

$$1010101010101010101\ldots10$$

'simple' or 'complex'?

Kolmogorov complexity

Is the string

$$10101010101010101010\ldots10$$

'simple' or 'complex'?

(One) answer: Simple because it can be described easily:

"10 repeated k times".

Kolmogorov complexity

Is the string

$$1010101010101010101010\ldots10$$

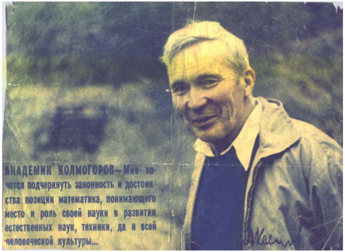
'simple' or 'complex'?

(One) answer: Simple because it can be described easily:

"10 repeated k times".

Remark: We should be careful in how we define *describing*; for instance, "to compute by an algorithm" (a formal procedure that eventually halts).

Kolmogorov complexity

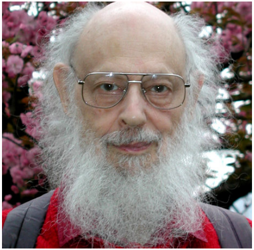


Kolmogorov complexity

A.N. Kolmogorov

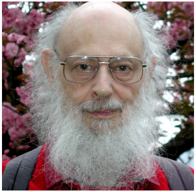
Kolmogorov complexity

A.N. Kolmogorov



Kolmogorov complexity

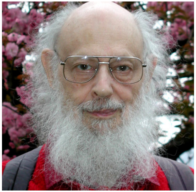
A.N. Kolmogorov



R.J. Solomonoff

Kolmogorov complexity

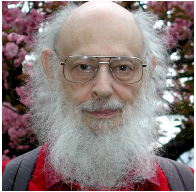
A.N. Kolmogorov



R.J. Solomonoff

Kolmogorov complexity

A.N. Kolmogorov

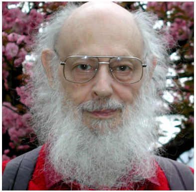


R.J. Solomonoff

G.J. Chaitin

Kolmogorov complexity

A.N. Kolmogorov



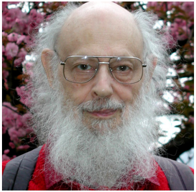
R.J. Solomonoff

G.J. Chaitin

Kolmogorov-Solomonoff-Chaitin complexity

Kolmogorov complexity

A.N. Kolmogorov



R.J. Solomonoff

G.J. Chaitin

Kolmogorov-Solomonoff-Chaitin complexity

$\longrightarrow$ **Kolmogorov complexity**

Kolmogorov complexity

`echo <x> | gzip - | wc -c` # times 8 (for bits)

source string, x		$\ell(C(x))$	**ratio**
$aaa \dots a$	$(10000 \times a)$	368	27.2 : 1

Kolmogorov complexity

`echo <x> | gzip - | wc -c` # times 8 (for bits)

source string, x		$\ell(C(x))$	**ratio**
$aaa\ldots a$	$(10000 \times a)$	368	$27.2 : 1$
$aabaabbbbabb\ldots$	(10000 random digits)	13456	$0.74 : 1$

Kolmogorov complexity

`echo <x> | gzip - | wc -c` # times 8 (for bits)

source string, x		$\ell(C(x))$	**ratio**
$aaa\ldots a$	$(10000 \times a)$	368	27.2 : 1
$aabaabbbbabb\ldots$	(10000 random digits)	13456	0.74 : 1
$abababab\ldots ab$	$(5000 \times ab)$	368	27.2 : 1

Kolmogorov complexity

`echo <x> | gzip - | wc -c`   # times 8 (for bits)

source string, x		$\ell(C(x))$	**ratio**
$aaa\dots a$	$(10000 \times a)$	368	$27.2 : 1$
$aabaabbbbabb\dots$	(10000 random digits)	13456	$0.74 : 1$
$abababab\dots ab$	$(5000 \times ab)$	368	$27.2 : 1$
$aaa\dots abbb\dots b$	$(5000 \times a, 5000 \times b)$	376	$26.6 : 1$

Kolmogorov complexity

`echo <x> | gzip - | wc -c`   # times 8 (for bits)

source string, x		$\ell(C(x))$	**ratio**
$aaa\ldots a$	$(10000 \times a)$	368	$27.2 : 1$
$aabaabbbbabb\ldots$	(10000 random digits)	13456	$0.74 : 1$
$abababab\ldots ab$	$(5000 \times ab)$	368	$27.2 : 1$
$aaa\ldots abbb\ldots b$	$(5000 \times a, 5000 \times b)$	376	$26.6 : 1$
$abbaababba\ldots$	$(1000 \times abbaababba)$	488	$20.5 : 1$

Kolmogorov complexity

```
echo <x> | gzip - | wc -c          # times 8 (for bits)
```

source string, x		$\ell(C(x))$	**ratio**
$aaa\dots a$	$(10000 \times a)$	368	$27.2 : 1$
$aabaabbbbabb\dots$	(10000 random digits)	13456	$0.74 : 1$
$abababab\dots ab$	$(5000 \times ab)$	368	$27.2 : 1$
$aaa\dots abbb\dots b$	$(5000 \times a, 5000 \times b)$	376	$26.6 : 1$
$abbaababba\dots$	$(1000 \times abbaababba)$	488	$20.5 : 1$

Strings that follow a rule can be compressed?

Kolmogorov complexity

`echo <x> | gzip - | wc -c` # times 8 (for bits)

source string, x		$\ell(C(x))$	**ratio**
$aaa\ldots a$	$(10000 \times a)$	368	27.2 : 1
$aabaabbbbabb\ldots$	(10000 random digits)	13456	0.74 : 1
$abababab\ldots ab$	$(5000 \times ab)$	368	27.2 : 1
$aaa\ldots abbb\ldots b$	$(5000 \times a, 5000 \times b)$	376	26.6 : 1
$abbaababba\ldots$	$(1000 \times abbaababba)$	488	20.5 : 1
$aaabbabbabb\ldots$	(π, 0–4 $\mapsto a$, 5–9 $\mapsto b$)	13416	0.74 : 1

π follows a rule but isn't compressible!

Kolmogorov complexity

`echo <x> | gzip - | wc -c` # times 8 (for bits)

source string, x		$\ell(C(x))$	**ratio**
$aaa\ldots a$	$(10000 \times a)$	368	$27.2 : 1$
$aabaabbbbabb\ldots$	(10000 random digits)	13456	$0.74 : 1$
$abababab\ldots ab$	$(5000 \times ab)$	368	$27.2 : 1$
$aaa\ldots abbb\ldots b$	$(5000 \times a, 5000 \times b)$	376	$26.6 : 1$
$abbaababba\ldots$	$(1000 \times abbaababba)$	488	$20.5 : 1$
$aaabbabbabb\ldots$	$(\pi$, 0–4 $\mapsto a$, 5–9 $\mapsto b)$	13416	$0.74 : 1$

π follows a rule but isn't compressible!

Perhaps the problem is in `gzip`? It would be possible to write a *specific program* that compresses π.

Kolmogorov complexity

`echo <x> | gzip - | wc -c` # times 8 (for bits)

source string, x		$\ell(C(x))$	**ratio**
$aaa\ldots a$	$(10000 \times a)$	368	$27.2 : 1$
$aabaabbbbabb\ldots$	(10000 random digits)	13456	$0.74 : 1$
$abababab\ldots ab$	$(5000 \times ab)$	368	$27.2 : 1$
$aaa\ldots abbb\ldots b$	$(5000 \times a, 5000 \times b)$	376	$26.6 : 1$
$abbaababba\ldots$	$(1000 \times abbaababba)$	488	$20.5 : 1$
$aaabbababb\ldots$	(π, 0–4 $\mapsto a$, 5–9 $\mapsto b$)	13416	$0.74 : 1$

π follows a rule but isn't compressible!

Perhaps the problem is in `gzip`? It would be possible to write a *specific program* that compresses π.

But what does it mean to compress *an individual string*???

Kolmogorov complexity

An individual string is “simple” (not “complex”) if it can be compressed using a *pre-specified* program.

Kolmogorov complexity

An individual string is “simple” (not “complex”) if it can be compressed using a *pre-specified* program.

Which program? `gzip` isn't good at compressing images (nor digits of π).

Kolmogorov complexity

An individual string is “simple” (not “complex”) if it can be compressed using a *pre-specified* program.

Which program? `gzip` isn't good at compressing images (nor digits of π).

We can use several programs as long as we prefix the file with a code indicating the used program.

Kolmogorov complexity

An individual string is “simple” (not “complex”) if it can be compressed using a *pre-specified* program.

Which program? `gzip` isn't good at compressing images (nor digits of π).

We can use several programs as long as we prefix the file with a code indicating the used program.

What about new programs?

Kolmogorov complexity

An individual string is “simple” (not “complex”) if it can be compressed using a *pre-specified* program.

Which program? `gzip` isn't good at compressing images (nor digits of π).

We can use several programs as long as we prefix the file with a code indicating the used program.

What about new programs? *Self-extracting files!*

Kolmogorov complexity

An individual string is "simple" (not "complex") if it can be compressed using a *pre-specified* program.

Which program? `gzip` isn't good at compressing images (nor digits of π).

We can use several programs as long as we prefix the file with a code indicating the used program.

What about new programs? *Self-extracting files!*

Do we this automatically? **Find the shortest program to print x.**

Kolmogorov complexity

An individual string is "simple" (not "complex") if it can be compressed using a *pre-specified* program.

Which program? `gzip` isn't good at compressing images (nor digits of π).

We can use several programs as long as we prefix the file with a code indicating the used program.

What about new programs? *Self-extracting files!*

Do we this automatically? **Find ~~*the shortest program to print x.*~~ the Kolmogorov complexity of x.**

Kolmogorov-kompleksisuus: mritelm

Let $U : \{0,1\}^* \to \{0,1\}^* \cup ⌛$ be a computer that given a program $\omega \in \{0,1\}^*$ either prints out a finite output $U(\omega) \in \{0,1\}^*$ or keeps computing forever. In the latter case, we say that the output $U(\omega)$ is undefined (⌛).

Kolmogorov-kompleksisuus: mritelm

Let $U : \{0,1\}^* \to \{0,1\}^* \cup ⌛$ be a computer that given a program $\omega \in \{0,1\}^*$ either prints out a finite output $U(\omega) \in \{0,1\}^*$ or keeps computing forever. In the latter case, we say that the output $U(\omega)$ is undefined (⌛).

Kolmogorov complexity

Given a string $x \in \{0,1\}^*$, let $\omega^*(x)$ be the *shortest* program such that

$$U(\omega^*(x)) = x \ .$$

The **Kolmogorov complexity** of x is the length of program $\omega^*(x)$:

$$K_U(x) = \min_{p\,:\,U(p)=x} |p| \ .$$

Kolmogorov complexity: basic properties

Let U and V be two computers. If computer U is 'rich' enough it can 'emulate' computer V.

Kolmogorov complexity: basic properties

Let U and V be two computers. If computer U is 'rich' enough it can 'emulate' computer V.

Universal computer

Computer U is said to be **universal** if for *any* other computer V, there exists a "translation program" $\tau \in \{0,1\}^*$ such that for all programs ω, we have

$$U(\tau\omega) = V(\omega) \quad .$$

Examples

The following are (in principle) universal computers

Examples

The following are (in principle) universal computers

1. Python (compiler + OS + hardware)

Examples

The following are (in principle) universal computers

1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)

Examples

The following are (in principle) universal computers

1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)

Examples

The following are (in principle) universal computers

1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)
4. universal Turing machine

Examples

The following are (in principle) universal computers

1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)
4. universal Turing machine
5. universal recursive function,

Examples

The following are (in principle) universal computers

1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)
4. universal Turing machine
5. universal recursive function,
6. Lambda calculus,

Examples

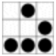
The following are (in principle) universal computers

1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)
4. universal Turing machine
5. universal recursive function,
6. Lambda calculus,
7. arithmetics,

Examples

The following are (in principle) universal computers

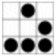
1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)
4. universal Turing machine
5. universal recursive function,
6. Lambda calculus,
7. arithmetics,
8. Game of Life



Examples

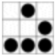
The following are (in principle) universal computers

1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)
4. universal Turing machine
5. universal recursive function,
6. Lambda calculus,
7. arithmetics,
8. Game of Life
9. ...



Examples

The following are (in principle) universal computers

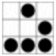
1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)
4. universal Turing machine
5. universal recursive function,
6. Lambda calculus,
7. arithmetics,
8. Game of Life 
9. ...

Each of these can emulate any of the others.

Examples

The following are (in principle) universal computers

1. Python (compiler + OS + hardware)
2. Java (compiler + OS + hardware)
3. your favorite programming language (interpreter/compiler + OS + hardware)
4. universal Turing machine
5. universal recursive function,
6. Lambda calculus,
7. arithmetics,
8. Game of Life
9. ...



Each of these can emulate any of the others.

In contrast, `gzip` (or rather, `gunzip`) is a **non-universal** computer.

Kolmogorov complexity: basic principles

Lemma: For any *universal computer* U and any other computer V we have

$$K_U(x) \leq K_V(x) + C \quad ,$$

where C is a constant independent of x.

Kolmogorov complexity: basic principles

Lemma: For any *universal computer* U and any other computer V we have

$$K_U(x) \leq K_V(x) + C \quad ,$$

where C is a constant independent of x.

Proof: Let τ be a translation program that translates the programs of V into programs of U, and let $\omega_V^*(x)$ be the shortest program such that $V(\omega_V^*(x)) = x$. Then, $U(\tau\omega_V^*(x)) = x$, and hence

$$K_U(x) \leq |\tau\omega_V^*(x)| = |\omega_V^*(x)| + |\tau| = K_V(X) + |\tau| \quad . \qquad \square$$

Invariance theorem

From now on, we consider the Kolmogorov complexity, K_U, defined using a *universal computer* U.

Invariance theorem

From now on, we consider the Kolmogorov complexity, K_U, defined using a *universal computer* U.

Invariance theorem

Kolmogorov complexity is (up to an additive constant) invariant wrt. the choice of the universal computer. In other words, for any two universal computers U and V, there is a constant $C > 0$ such that

$$|K_U(x) - K_V(x)| \leq C \quad \text{for any } x \in \{0,1\}^* \ .$$

Invariance theorem

From now on, we consider the Kolmogorov complexity, K_U, defined using a *universal computer* U.

Invariance theorem

Kolmogorov complexity is (up to an additive constant) invariant wrt. the choice of the universal computer. In other words, for any two universal computers U and V, there is a constant $C > 0$ such that

$$|K_U(x) - K_V(x)| \le C \quad \text{for any } x \in \{0,1\}^* \ .$$

Proof: Let $\tau_{V \to U}$ be a program that translates programs of V into programs of U so that $U(\tau\omega) = V(\omega)$ for all ω. Then $K_U(x) \le K_V(x) + |\tau_{V \to U}|$ for all x. Similarly, $K_V(x) \le K_U(x) + |\tau_{U \to V}|$ for all x. The theorem follows by setting $C = \max\{|\tau_{V \to U}|, |\tau_{U \to V}|\}$. $\square$

Conditional Kolmogorov complexity

Conditional Kolmogorov complexity

The **conditional Kolmogorov complexity** is the length of the shortest program to convert input y into output x:

$$K_U(x \mid y) = \min\{|\omega| \quad : \ U(\bar{y}\,\omega) = x\}$$

where $\bar{y}$ is a "self-delimiting" description of y.

Conditional Kolmogorov complexity

Conditional Kolmogorov complexity

The **conditional Kolmogorov complexity** is the length of the shortest program to convert input y into output x:

$$K_U(x \mid y) = \min\{|\omega| \quad : \; U(\bar{y}\,\omega) = x\}$$

where $\bar{y}$ is a "self-delimiting" description of y.

Uniform upper bounds

The following upper bound holds for all x:

$$K_U(x \mid |x|) \leq |x| + C$$

where C is a constant independent of x.

Examples

Let $n = |x|$.

Examples

Let $n = |x|$.

1. $K_U(\texttt{0101010101}\ldots\texttt{01} \mid n) = C$.
   Program: `print` $n/2$ `times '01'.`

Examples

Let $n = |x|$.

1. $K_U(0101010101\ldots01 \mid n) = C$.
   Program: `print` $n/2$ `times '01'.`
2. $K_U(\pi_1\,\pi_2\,\ldots\,\pi_n \mid n) = C$.
   Program: `print the` n `first bits of` π.

Examples

Let $n = |x|$.

1. $K_U(0101010101\ldots01 \mid n) = C$.
   Program: `print` $n/2$ `times '01'.`
2. $K_U(\pi_1 \pi_2 \ldots \pi_n \mid n) = C$.
   Program: `print the` n `first bits of` π.
3. $K_U(\text{English text} \mid n) \lesssim 1.3 \times n + C$.
   Program: `Huffman code.`
   (The estimated entropy of English is about 1.3 bits per symbol.)

Examples

Let $n = |x|$.

1. $K_U(0101010101\ldots01 \mid n) = C$.
   Program: `print` $n/2$ `times '01'.`
2. $K_U(\pi_1\,\pi_2\,\ldots\,\pi_n \mid n) = C$.
   Program: `print the` n `first bits of` π`.`
3. $K_U(\text{English text} \mid n) \lesssim 1.3 \times n + C$.
   Program: `Huffman code.`
   (The estimated entropy of English is about 1.3 bits per symbol.)
4. $K_U(\text{fractal}) = C$.
   Program: `print the number of interations until` $z_{n+1} = z_n^2 + c > T$`.`

Examples

Martin-Löf randomness

Examples (contd.):

5. $K_U(x \mid n) \approx n$ for almost all $x \in \{0,1\}^n$.

Martin-Löf randomness

Examples (contd.):

5. $K_U(x \mid n) \approx n$ for almost all $x \in \{0,1\}^n$.
   Proof: Uniform upper bound: $K_U(x \mid n) \leq n + C$. Lower bound from a counting argument — less than 2^{-k} strings can be compressed by more than k bits.

Martin-Löf randomness

Examples (contd.):

5. $K_U(x \mid n) \approx n$ for almost all $x \in \{0, 1\}^n$.
   Proof: Uniform upper bound: $K_U(x \mid n) \leq n + C$. Lower bound from a counting argument — less than 2^{-k} strings can be compressed by more than k bits.

Martin-Löf randomness

String x is said to be **Martin-Löf random** iff $K_U(x \mid n) \geq n$.

Martin-Löf randomness

Examples (contd.):

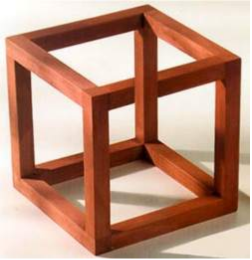
5. $K_U(x \mid n) \approx n$ for almost all $x \in \{0,1\}^n$.
   Proof: Uniform upper bound: $K_U(x \mid n) \leq n + C$. Lower bound from a counting argument — less than 2^{-k} strings can be compressed by more than k bits.

Martin-Löf randomness

String x is said to be **Martin-Löf random** iff $K_U(x \mid n) \geq n$.

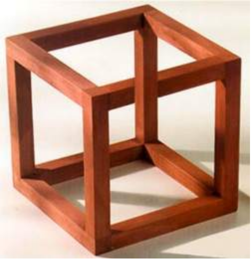
Consequence of point 5: A sequence of coin tosses is Martin-Löf random with high probability.

Berry paradox



What is the least natural number that cannot be described using thirteen words?

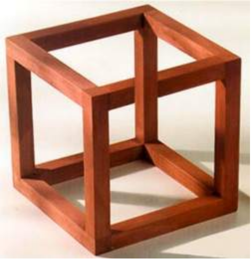
Berry paradox



What is the least natural number that cannot be described using thirteen words?

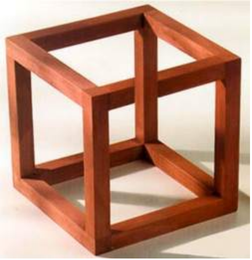
Whatever the number is, we have just described(?) it using thirteen words!

Berry paradox



The least *uninteresting* natural number?

Berry paradox



The least *uninteresting* natural number?

Whatever it is, such a number is quite interesting.

Non-computability

There is no algorithmic way to compute $K_U(x)$.

Non-computability

Kolmogorov complexity $K_U : \{0,1\}^* \to \mathbb{N}$ is a **non-computable** function.

Non-computability

There is no algorithmic way to compute $K_U(x)$.

Non-computability

Kolmogorov complexity $K_U : \{0,1\}^* \to \mathbb{N}$ is a **non-computable** function.

Proof: Assume that $K_U(x)$ were computable.

Non-computability

There is no algorithmic way to compute $K_U(x)$.

Non-computability

Kolmogorov complexity $K_U : \{0,1\}^* \to \mathbb{N}$ is a **non-computable** function.

Proof: Assume that $K_U(x)$ were computable. Consider the program

$$\texttt{print } x \texttt{ for which } K_U(x) > M \,.$$

Non-computability

There is no algorithmic way to compute $K_U(x)$.

Non-computability

Kolmogorov complexity $K_U : \{0,1\}^* \to \mathbb{N}$ is a **non-computable** function.

Proof: Assume that $K_U(x)$ were computable. Consider the program

$$\texttt{print } x \texttt{ for which } K_U(x) > M \,.$$

Contradiction follows by choosing M greater than the Kolmogorov complexity of the above program. Hence, $K_U(x)$ cannot be computable. □

Kolmogorov complexity: summary

To summarize:

- Kolmogorov complexity, $K_U(x)$, is the length of the shortest program, ω, such that $U(\omega) = x$.

Kolmogorov complexity: summary

To summarize:

- Kolmogorov complexity, $K_U(x)$, is the length of the shortest program, ω, such that $U(\omega) = x$.
- The choice of the universal computer, U, affects the definition by an additive constant independent of x.

Kolmogorov complexity: summary

To summarize:

- Kolmogorov complexity, $K_U(x)$, is the length of the shortest program, ω, such that $U(\omega) = x$.
- The choice of the universal computer, U, affects the definition by an additive constant independent of x.
- Uncomputable.

Kolmogorov complexity: summary

To summarize:

- Kolmogorov complexity, $K_U(x)$, is the length of the shortest program, ω, such that $U(\omega) = x$.
- The choice of the universal computer, U, affects the definition by an additive constant independent of x.
- Uncomputable.
- Enables the definition of randomness of *individual* strings.

Tomorrow

Tomorrow's plan:

1. Occam's Razor,

Tomorrow

Tomorrow's plan:

1. Occam's Razor,
2. MDL principle,

Tomorrow

Tomorrow's plan:

1. Occam's Razor,
2. MDL principle,
3. Universal coding,

Tomorrow

Tomorrow's plan:

1. Occam's Razor,
2. MDL principle,
3. Universal coding,
4. Example applications.

Tomorrow

Tomorrow's plan:

1. Occam's Razor,
2. MDL principle,
3. Universal coding,
4. Example applications.

Thanks for your attention. Now, let's have a few caipirinhas!