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.”
Course Outline

What is Coding?
Symbol Codes
Entropy and Information
Kolmogorov complexity

Course details
What is Information?
Why Information?
Information vs. Complexity
Information Theory
1 Course Outline

2 What is Coding?
Course Outline

1. What is Coding?
2. Symbol Codes
3. Entropy and Information
4. Kolmogorov complexity

What is Information?

1. Why Information?
2. Information vs. Complexity
3. Information Theory

Introduction to Information-Theoretic Modeling
1. Course Outline
2. What is Coding?
3. Symbol Codes
4. Entropy and Information
Course Outline
What is Coding?
Symbol Codes
Entropy and Information
Kolmogorov complexity

What is Information?
Why Information?
Information vs. Complexity
Information Theory

1  Course Outline
2  What is Coding?
3  Symbol Codes
4  Entropy and Information
5  Kolmogorov complexity
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, $2 \times 3$ h.
Introduction to Information-Theoretic Modeling

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

Teemu Roos

Introduction to Information-Theoretic Modeling
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, $2 \times 3$ 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:
Further reading:

- **Highly recommended**: Cover & Thomas, *Elements of Information Theory*.
Course Outline
What is Coding?
Symbol Codes
Entropy and Information
Kolmogorov complexity

Course details
What is Information?
Why Information?
Information vs. Complexity
Information Theory

Resources

Further reading:

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

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

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

Further reading:

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

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

\[
\text{Complexity} = \text{Information} + \text{Noise} \\
= \text{Regularity} + \text{Randomness} \\
= \text{Algorithm} + \text{Compressed file}
\]
"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)
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.

<table>
<thead>
<tr>
<th>A</th>
<th>G</th>
<th>M</th>
<th>S</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>H</td>
<td>N</td>
<td>T</td>
<td>Z</td>
</tr>
<tr>
<td>C</td>
<td>I</td>
<td>O</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>J</td>
<td>P</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>K</td>
<td>Q</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>L</td>
<td>R</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Teemu Roos
Introduction to Information-Theoretic Modeling
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.
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 •: 1 units.
- A dash —: 2 units.
- A space between words: 2 units.

• • • — — — • • •: 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).
Course Outline
- What is Coding?
- Symbol Codes
- Entropy and Information
- Kolmogorov complexity

Dots and Dashes
- Codes as Mappings
- Data Compression

Coding Game

© 1989 A.G. Reinhold.

Samuel F.M. Morse (1791–1872)
Coding Game

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

.--- .... .--. --- .- -.. --- .-- .. - ... .......
.---- .-. .- ...-. - --- -.. --- .-- .. - .......
..-. ..-. --- .-. -- .- - .. --- -. 

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

Morse code

Coding rate: 137

42

≈ 3.26

Did you do better or worse? Why?
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: $\frac{137}{42} \approx 3.26$

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

Only lossless codes are uniquely decodable.
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

Introduction to Information-Theoretic Modeling by Teemu Roos
# Course Outline

- What is Coding?
- Symbol Codes
- Entropy and Information
- Kolmogorov complexity

# Dots and Dashes
- Codes as Mappings
- Data Compression

## Examples

<table>
<thead>
<tr>
<th>Category</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>general</td>
<td>gzip</td>
</tr>
<tr>
<td>purpose</td>
<td>bzip</td>
</tr>
<tr>
<td>image</td>
<td>png</td>
</tr>
<tr>
<td>music</td>
<td>mp3</td>
</tr>
<tr>
<td>video</td>
<td>mpeg</td>
</tr>
</tbody>
</table>

Teemu Roos
Introduction to Information-Theoretic Modeling
What is Coding?
Symbol Codes
Entropy and Information
Kolmogorov complexity

Dots and Dashes
Codes as Mappings
Data Compression

Examples

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

lossless

lossy
Examples

compression ratio

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>general purpose</td>
<td>gzip</td>
<td>~ 1:3</td>
<td>lossless</td>
<td>bzip</td>
<td>~ 1:3.5</td>
</tr>
<tr>
<td>image</td>
<td>png</td>
<td>~ 1:2.5</td>
<td>jpeg</td>
<td>~ 1:25</td>
<td>lossy</td>
</tr>
<tr>
<td>music</td>
<td>mp3</td>
<td>~ 1:12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>video</td>
<td>mpeg</td>
<td>~ 1:30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Teemu Roos
Introduction to Information-Theoretic Modeling
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 $2^n$. 

Teemu Roos

Introduction to Information-Theoretic Modeling
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

$$\frac{2^{n-k} - 1}{2^n} < \frac{2^{n-k}}{2^n} = 2^{-k}.$$
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.
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.
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 0.000000000000000000000000000001 % of files are compressible by 100 bits.
1. Course Outline

2. What is Coding?

3. Symbol Codes
   - Decodable Codes
   - Prefix Codes
   - Kraft-McMillan Theorem

4. Entropy and Information

5. Kolmogorov complexity

Introduction to Information-Theoretic Modeling
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.
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 $x_i$:

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

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

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

![Diagram showing the extension of a symbol code]
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 $x_i$:

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

\[
C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n) .
\]
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 $x_i$:

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

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

\[
C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n)
\]

**INPUT STRING**

```
I N P U T    S T R I N G    . . .
```

```
1001 0001 1110 0110 1011 1110 1010 0111 . . .
```
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

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

× 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

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 $C : \mathcal{X} \rightarrow \{0, 1\}^*$ 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.

$\times$ 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 $C : \mathcal{X} \rightarrow \{0, 1\}^*$ 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.

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

$\checkmark$ 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

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

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

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|
Kraft Inequality

<table>
<thead>
<tr>
<th>Total budget</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>000</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>0001</td>
</tr>
<tr>
<td>01</td>
<td>010</td>
<td>0100</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>0101</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>1001</td>
</tr>
<tr>
<td>11</td>
<td>110</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>1110</td>
</tr>
</tbody>
</table>

X Kraft inequality violated. ⇒ Not decodable.
## Kraft Inequality

A fixed-length code is a type of code where each symbol is represented by the same number of bits. This is in contrast to variable-length codes, where the number of bits needed to represent a symbol can vary.

The Kraft Inequality provides a condition for the existence of a fixed-length code. It states that for a set of symbols with associated probabilities, there exists a fixed-length code if and only if:

\[
\sum_{i=1}^{n} 2^{-x_i} \leq 1
\]

where \(x_i\) is the length of the code for the \(i\)-th symbol.

### Example

Consider a set of symbols with the following probabilities:

- Symbol 1 with probability 0.5
- Symbol 2 with probability 0.2
- Symbol 3 with probability 0.3

We need to check if there exists a fixed-length code for this set of symbols.

Using the Kraft Inequality:

\[
\sum_{i=1}^{3} 2^{-x_i} = 2^{-x_1} + 2^{-x_2} + 2^{-x_3} = 0.5 + 0.2 + 0.3 = 1
\]

Since the sum is equal to 1, the Kraft Inequality is satisfied, and a fixed-length code exists for this set of symbols.

The table below illustrates a possible fixed-length code for these symbols:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
</tr>
</tbody>
</table>

Each symbol is represented by a fixed number of bits (2 in this case), satisfying the Kraft Inequality.
## Kraft Inequality

<table>
<thead>
<tr>
<th>Total budget</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>000</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>111</td>
</tr>
<tr>
<td>01</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>011</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>111</td>
<td></td>
</tr>
</tbody>
</table>

- Decodable & prefix-free

### Symbol Codes
- Entropy and Information
- Kolmogorov complexity

### Decodable Codes
- Prefix Codes
- Kraft-McMillan Theorem

---

**Course Outline**
- **What is Coding?**
- **Symbol Codes**
- Entropy and Information
- Kolmogorov complexity
- Decodable Codes
- Prefix Codes
- Kraft-McMillan Theorem

**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
Kraft Inequality

<table>
<thead>
<tr>
<th>Total budget</th>
<th>Kraft?</th>
<th>Decodable?</th>
<th>Prefix-free?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>000</td>
<td>0000, 0001</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td></td>
<td>0010, 0011</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td></td>
<td>0100, 0101</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td></td>
<td>0110, 0111</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td></td>
<td>1000, 1001</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td></td>
<td>1010, 1011</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td></td>
<td>1100, 1101</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td></td>
<td>1110, 1111</td>
</tr>
</tbody>
</table>
Kraft Inequality

<table>
<thead>
<tr>
<th>Total budget</th>
<th>0</th>
<th>00</th>
<th>000</th>
<th>0000</th>
<th>0001</th>
<th>0002</th>
<th>0003</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01</td>
<td>010</td>
<td>0100</td>
<td>0101</td>
<td>0102</td>
<td>0103</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>10</td>
<td>100</td>
<td>1000</td>
<td>1001</td>
<td>1002</td>
<td>1003</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>110</td>
<td>1100</td>
<td>1101</td>
<td>1102</td>
<td>1103</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>111</td>
<td>1110</td>
<td>1111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Kraft?  ✓  Decodable?  ✓  Prefix-free?  ✗
# Kraft Inequality

<table>
<thead>
<tr>
<th>Total budget</th>
<th>Kraft?</th>
<th>Decodable?</th>
<th>Prefix-free?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Kraft Inequality is a fundamental theorem in information theory that relates the weights of a set of symbols to the lengths of their codewords. It states that the sum of the weights of the symbols cannot exceed the total budget. The table above illustrates this with binary symbols and their codewords.
Kraft Inequality

\[ \sum_{i=1}^{n} 2^{-\text{length}(c_i)} \leq 1 \]

- Kraft? \( \checkmark \)
- Decodable? \( \times \)
- Prefix-free? \( \times \)
Kraft Inequality

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

$$ \sum_{i=1}^{m} 2^{-\ell_i} < 1 .$$
**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. ⇒ Some codewords can be made shorter.
**Kraft Inequality**

<table>
<thead>
<tr>
<th>Total Budget</th>
<th>Code 0</th>
<th>Code 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0000</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>0010</td>
<td>0011</td>
</tr>
<tr>
<td>01</td>
<td>0100</td>
<td>0101</td>
</tr>
<tr>
<td></td>
<td>0110</td>
<td>0111</td>
</tr>
<tr>
<td>10</td>
<td>1000</td>
<td>1001</td>
</tr>
<tr>
<td></td>
<td>1010</td>
<td>1011</td>
</tr>
<tr>
<td>11</td>
<td>1100</td>
<td>1101</td>
</tr>
<tr>
<td></td>
<td>1110</td>
<td>1111</td>
</tr>
</tbody>
</table>

"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?
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
- Decodable Codes
- Prefix Codes
- All Codes
Kraft Inequality

<table>
<thead>
<tr>
<th>Total budget</th>
<th>0</th>
<th>00</th>
<th>000</th>
<th>0000</th>
<th>0001</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>001</td>
<td>001</td>
<td>0011</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>010</td>
<td>0100</td>
<td>0100</td>
<td>0101</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>011</td>
<td>0110</td>
<td>0110</td>
<td>0111</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>10</td>
<td>100</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>110</td>
<td>1100</td>
<td>1100</td>
<td>1101</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1101</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>111</td>
<td>1110</td>
<td>1110</td>
<td>1111</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Kraft? √  Decodable? ×  Prefix-free? ×
## Kraft Inequality

<table>
<thead>
<tr>
<th>Total budget</th>
<th>Kraft?</th>
<th>Decodable?</th>
<th>Prefix-free?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.
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:

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

\[ q(x) = 2^{-\ell(C(x))} \iff \ell(C(x)) = -\log_2 q(x) = \log_2 \frac{1}{q(x)}. \]
Expected Code-length

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

\[ q(x) = 2^{-\ell(C(x))} \iff \ell(C(x)) = -\log_2 q(x) = \log_2 \frac{1}{q(x)}. \]

The Kraft-(in)equality implies that

\[ \sum_{x \in X} q(x) \leq 1. \]
Expected Code-length

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

\[ q(x) = 2^{-\ell(C(x))} \iff \ell(C(x)) = -\log_2 q(x) = \log_2 \frac{1}{q(x)}. \]

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

\[ q(x) = 2^{-\ell(C(x))} \iff \ell(C(x)) = -\log_2 q(x) = \log_2 \frac{1}{q(x)}. \]

The Kraft-(in)equality implies that

\[ \sum_{x \in X} q(x) = 1. \]

Important Observation

Probability distributions are codes are probability distributions!
Based on the unification of codes and distributions, we can write

\[ E[\ell(C(X))] = \sum_{x \in \mathcal{X}} p(x) \ell(C(x)) \]
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 \( q(x) = 2^{-\ell(C(x))}. \)
Based on the unification of codes and distributions, we can write

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

\[ = \sum_{x \in X} p(x) \log_2 \frac{1}{q(x)} , \]

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

⇒ Information theory (entropy, Kullback-Leibler divergence, ...)
Course Outline

1. What is Coding?
2. Symbol Codes
3. Entropy and Information
   - Entropy
   - Kullback-Leibler Divergence
   - Nearly Optimal Coding
4. Kolmogorov complexity

Teemu Roos
Introduction to Information-Theoretic Modeling
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

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

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

\[
I_X(x) = \log_2 \frac{1}{p_X(x)}.
\]

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”:

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

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

$$H(X) = p \log_2 \frac{1}{p} + (1 - p) \log_2 \frac{1}{1 - p}.$$
the joint entropy of two (or more) random variables:

\[ H(X, Y) = \sum_{x \in X} \sum_{y \in Y} p_{X,Y}(x,y) \log_2 \frac{1}{p_{X,Y}(x,y)} , \]
More Entropies

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

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

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

\[
H(X \mid Y = y) = \sum_{x \in \mathcal{X}} p_{X \mid Y}(x \mid y) \log_2 \frac{1}{p_{X \mid Y}(x \mid y)} ,
\]
More Entropies

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

\[
H(X, Y) = \sum_{x \in X} \sum_{y \in Y} p_{X,Y}(x, y) \log_2 \frac{1}{p_{X,Y}(x, y)},
\]

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

\[
H(X \mid Y = y) = \sum_{x \in X} p_{X \mid Y}(x \mid y) \log_2 \frac{1}{p_{X \mid Y}(x \mid y)},
\]

3. and the **conditional entropy**:

\[
H(X \mid Y) = \sum_{y \in Y} p(y) H(X \mid Y = y).
\]
The joint entropy $H(X, Y)$ measures the uncertainty about the pair $(X, Y)$. 
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:

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

For the entropy we have:

Chain Rule of Entropy

\[ H(X, Y) = H(Y) + H(X | Y) \, . \]
Chain Rule of Entropy

Remember the chain rule of probability:

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

For the entropy we have:

\[ H(X, Y) = H(Y) + H(X | Y). \]

\[ X \independent Y \iff H(X | Y) = H(X) \iff H(X, Y) = H(X) + H(Y). \]
Chain Rule of Entropy

Remember the chain rule of probability:

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

For the entropy we have:

\[ H(X, Y) = H(Y) + H(X | Y). \]

\[ X \perp Y \iff H(X | Y) = H(X) \iff H(X, Y) = H(X) + H(Y). \]

*Logarithmic* scale makes entropy **additive**.
The **mutual information**

\[ I(X ; Y) = H(X) - H(X | Y) \]

measures the average decrease in uncertainty about \(X\) when the value of \(Y\) becomes known.
Mutual Information

The **mutual information**

\[
I(X ; Y) = H(X) - H(X | Y)
\]

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

Mutual information is *symmetric* (chain rule):

\[
I(X ; Y) = H(X) - H(X | Y) = H(X) - (H(X, Y) - H(Y))
\]
Mutual Information

The **mutual information**

\[ I(X ; Y) = H(X) − H(X | Y) \]

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

Mutual information is *symmetric* (chain rule):

\[ I(X ; Y) = H(X) − H(X | Y) = H(X) − H(X, Y) + H(Y) \]
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):

\[
\begin{align*}
I(X; Y) &= H(X) - H(X \mid Y) = (H(X) - H(X, Y)) + H(Y) \\
&= H(Y) - H(Y \mid X) = I(Y; X).
\end{align*}
\]
The **mutual information**

\[ I(X ; Y) = H(X) - H(X | Y) \]

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

Mutual information is *symmetric* (chain rule):

\[
I(X ; Y) = H(X) - H(X | Y) = (H(X) - H(X, Y)) + H(Y) \\
= H(Y) - H(Y | X) = I(Y ; X) .
\]

On the average, \( X \) gives as much information about \( Y \) as \( Y \) gives about \( X \).
Relationships between Entropies

- $H(X,Y)$
- $H(X)$
- $H(Y)$
- $H(X | Y)$
- $I(X ; Y)$
- $H(Y | X)$

These entropies represent the relationships and uncertainties between variables $X$ and $Y$.
Course Outline
What is Coding?
Symbol Codes
Entropy and Information
Kolmogorov complexity

Entropy
Kullback-Leibler Divergence
Nearly Optimal Coding

Time for a break?
Kullback-Leibler Divergence

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

$$D(p_X \parallel q_X) = \sum_{x \in X} p_X(x) \log_2 \frac{p_X(x)}{q_X(x)}.$$
Kullback-Leibler Divergence

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

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

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

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

\[ I(X ; Y) \geq 0 \]
The information inequality implies

\[ I(X ; Y) \geq 0 . \]

**Proof.**

\[
I(X ; Y) = H(X) - H(X | Y) \\
= H(X) + H(Y) - H(X, Y) \\
= \sum_{x \in \mathcal{X} \atop y \in \mathcal{Y}} p_{X,Y}(x, y) \log_2 \left( \frac{p_{X,Y}(x, y)}{p_X(x) p_Y(y)} \right) \\
= D(p_{X,Y} \parallel p_X p_Y) \geq 0 .
\]
Kullback-Leibler Divergence

The information inequality implies

\[ I(X \; ; \; Y) \geq 0 \].

**Proof.**

\[
I(X \; ; \; Y) = H(X) - H(X \mid Y)
\]
\[
= H(X) + H(Y) - H(X, Y)
\]
\[
= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{X,Y}(x, y) \log_2 \frac{p_{X,Y}(x, y)}{p_X(x) p_Y(y)}
\]
\[
= D(p_{X,Y} \parallel p_X p_Y) \geq 0 .
\]

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:

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

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

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

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

Properties of entropy:

1. \( H(X) \geq 0 \)

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

2. \( H(X) \leq \log_2 |X| \)

   A \textbf{combinatorial} approach to the definition of information (Boltzmann, 1896; Hartley, 1928; Kolmogorov, 1965):
   \[ S = k \ln W. \]
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. \( H(X) \leq \log_2 |X| \)
   
   A *combinatorial* approach to the definition of information (Boltzmann, 1896; Hartley, 1928; Kolmogorov, 1965):
   
   \[
   S = k \ln W .
   \]

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):
   \[
   S = k \ln W .
   \]

3. \( H(X \mid Y) \leq H(X) \)
   
   **Proof.**
   \[
   0 \leq I(X \mid Y) = H(X) - H(X \mid Y) .
   \]
Properties of Entropy

Properties of entropy:

1. \( H(X) \geq 0 \)

   \textit{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 \textbf{combinatorial} approach to the definition of information (Boltzmann, 1896; Hartley, 1928; Kolmogorov, 1965):
   \[ S = k \ln W. \]

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

   \textit{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

\[ E[\ell(X)] \geq H(X) \]

Proof.

\[ E[\ell(X)] = \sum_{x \in X} p(x) \ell(x) = \sum_{x \in X} p(x) \log_2 1/q(x) = 2^{-\ell(x)} = \sum_{x \in X} p(x) \left[ \log_2 p(x)/q(x) + \log_2 1/p(x) \right] = D(p∥q) + H(X) \geq 0. \]
Entropy Lower Bound

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

\textbf{Proof.}

\[ E[\ell(X)] = \sum_{x \in X} p(x) \ell(x) \]
Entropy Lower Bound

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

Proof.

\[
E[\ell(X)] = \sum_{x \in \mathcal{X}} p(x) \ell(x) \\
= \sum_{x \in \mathcal{X}} p(x) \log_2 \frac{1}{q(x)} \\
= \sum_{x \in \mathcal{X}} p(x) \log_2 \frac{1}{2^{-\ell(x)}} \\
= \sum_{x \in \mathcal{X}} p(x) \log_2 2^\ell(x) \\
= \sum_{x \in \mathcal{X}} p(x) \frac{1}{q(x)} \\
= H(X).
\]

Teemu Roos  Introduction to Information-Theoretic Modeling
Entropy Lower Bound

\[ E[\ell(X)] \geq H(X) \]

**Proof.**

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

\[
= \sum_{x \in X} p(x) \log_2 \frac{1}{q(x)}
\]

\[
= \sum_{x \in X} p(x) \left[ \log_2 \frac{p(x)}{q(x)} + \log_2 \frac{1}{p(x)} \right]
\]

where \( q(x) = 2^{-\ell(x)} \)
Entropy Lower Bound

\[ E[\ell(X)] \geq H(X) \]

Proof.

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

\[ = \sum_{x \in X} p(x) \log_2 \frac{1}{q(x)} \]

\[ q(x) = 2^{-\ell(x)} \]

\[ = \sum_{x \in X} p(x) \left[ \log_2 \frac{p(x)}{q(x)} + \log_2 \frac{1}{p(x)} \right] \]


**Entropy Lower Bound**

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

**Proof.**

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

\[
= \sum_{x \in X} p(x) \log_2 \frac{1}{q(x)}
\]

\[
= \sum_{x \in X} p(x) \left[ \log_2 \frac{p(x)}{q(x)} + \log_2 \frac{1}{p(x)} \right]
\]

\[
= D(p \parallel q) + H(X) \geq 0.
\]
So what have we learned?
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. \[ E[\ell(X)] = H(X) + D(p \parallel q), \text{ where } q(x) = 2^{-\ell(x)}. \]

2. \[ E[\ell(X)] \geq H(X). \]
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

<table>
<thead>
<tr>
<th>$x$</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(x)$</td>
<td>1/2</td>
<td>1/8</td>
<td>1/16</td>
<td>1/4</td>
<td>1/16</td>
</tr>
</tbody>
</table>

Teemu Roos
Introduction to Information-Theoretic Modeling
### Exercise 5.

<table>
<thead>
<tr>
<th>Total budget</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>101</td>
</tr>
<tr>
<td>01</td>
<td>010</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>111</td>
<td></td>
</tr>
</tbody>
</table>
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).
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

$$\ell(x) = \left\lceil \log_2 \frac{1}{p(x)} \right\rceil$$

for all $x \in \mathcal{X}$. 
Alice in Wonderland
### Shannon’s code: Example

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

$H(X) = 4.03$
### Shannon’s code: Example

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

\[ H(X) = 4.03 \]

Shannon (1948):

- Sort by probability.
- Choose codewords in order, avoiding prefixes.

("Kraft table"!)
### Shannon’s code: Example

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

**$H(X) = 4.03$**

Shannon (1948):

1. Sort by probability.
## Shannon’s code: Example

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

\[ H(X) = 4.03 \]

Shannon (1948):

1. Sort by probability.
Shannon’s code: Example

<table>
<thead>
<tr>
<th>X</th>
<th>p(X)</th>
<th>log₂ \frac{1}{p(X)}</th>
<th>ℓ(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>0.0991</td>
<td>3.3</td>
<td>4</td>
</tr>
<tr>
<td>t</td>
<td>0.0781</td>
<td>3.6</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>0.0644</td>
<td>3.9</td>
<td>4</td>
</tr>
<tr>
<td>o</td>
<td>0.0598</td>
<td>4.0</td>
<td>5</td>
</tr>
<tr>
<td>i</td>
<td>0.0551</td>
<td>4.1</td>
<td>5</td>
</tr>
<tr>
<td>h</td>
<td>0.0535</td>
<td>4.2</td>
<td>5</td>
</tr>
<tr>
<td>n</td>
<td>0.0516</td>
<td>4.2</td>
<td>5</td>
</tr>
<tr>
<td>s</td>
<td>0.0475</td>
<td>4.3</td>
<td>5</td>
</tr>
<tr>
<td>r</td>
<td>0.0401</td>
<td>4.6</td>
<td>5</td>
</tr>
<tr>
<td>d</td>
<td>0.0359</td>
<td>4.7</td>
<td>5</td>
</tr>
<tr>
<td>l</td>
<td>0.0343</td>
<td>4.8</td>
<td>5</td>
</tr>
<tr>
<td>x</td>
<td>0.0011</td>
<td>9.8</td>
<td>10</td>
</tr>
<tr>
<td>j</td>
<td>0.0011</td>
<td>9.8</td>
<td>10</td>
</tr>
<tr>
<td>z</td>
<td>0.0005</td>
<td>10.7</td>
<td>11</td>
</tr>
</tbody>
</table>

\[ H(X) = 4.03 \]

Shannon (1948):

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

Introduction to Information-Theoretic Modeling
## Shannon’s code: Example

<table>
<thead>
<tr>
<th>Total budget</th>
<th>Codeword lengths (3, 4, 4, 4, 5, 5, 5, 5, ... , 10, 10, 11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000 0000, 0001, 0010, 0011</td>
</tr>
<tr>
<td>00</td>
<td>001 0100, 0101, 0110, 0111</td>
</tr>
<tr>
<td>01</td>
<td>010 1000, 1001, 1010, 1011</td>
</tr>
<tr>
<td>1</td>
<td>100 1100, 1101, 1110, 1111</td>
</tr>
<tr>
<td>10</td>
<td>101</td>
</tr>
<tr>
<td>11</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>111</td>
</tr>
</tbody>
</table>
### Shannon’s code: Example

<table>
<thead>
<tr>
<th>Total budget</th>
<th>Codeword lengths</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>000</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>0001</td>
</tr>
<tr>
<td>01</td>
<td>010</td>
<td>0100</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>0110</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>1001</td>
</tr>
<tr>
<td>11</td>
<td>110</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>1110</td>
</tr>
</tbody>
</table>

Codeword lengths: \(3, 4, 4, 5, 5, 5, 5, \ldots, 10, 10, 11\)
Shannon’s code: Example

Codeword lengths (3, 4, 4, 5, 5, 5, 5, ..., 10, 10, 11)
Shannon's code: Example

<table>
<thead>
<tr>
<th>X</th>
<th>p(X)</th>
<th>\log_2 \frac{1}{p(X)}</th>
<th>\ell(X)</th>
<th>C(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>0.2111</td>
<td>2.2</td>
<td>3</td>
<td>000</td>
</tr>
<tr>
<td>e</td>
<td>0.0991</td>
<td>3.3</td>
<td>4</td>
<td>0010</td>
</tr>
<tr>
<td>t</td>
<td>0.0781</td>
<td>3.6</td>
<td>4</td>
<td>0011</td>
</tr>
<tr>
<td>a</td>
<td>0.0644</td>
<td>3.9</td>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>o</td>
<td>0.0598</td>
<td>4.0</td>
<td>5</td>
<td>01010</td>
</tr>
<tr>
<td>i</td>
<td>0.0551</td>
<td>4.1</td>
<td>5</td>
<td>01011</td>
</tr>
<tr>
<td>h</td>
<td>0.0535</td>
<td>4.2</td>
<td>5</td>
<td>01100</td>
</tr>
<tr>
<td>n</td>
<td>0.0516</td>
<td>4.2</td>
<td>5</td>
<td>01101</td>
</tr>
<tr>
<td>s</td>
<td>0.0475</td>
<td>4.3</td>
<td>5</td>
<td>01110</td>
</tr>
<tr>
<td>r</td>
<td>0.0401</td>
<td>4.6</td>
<td>5</td>
<td>01111</td>
</tr>
<tr>
<td>d</td>
<td>0.0359</td>
<td>4.7</td>
<td>5</td>
<td>10000</td>
</tr>
<tr>
<td>l</td>
<td>0.0343</td>
<td>4.8</td>
<td>5</td>
<td>10001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>0.0011</td>
<td>9.8</td>
<td>10</td>
<td>10101111101</td>
</tr>
<tr>
<td>j</td>
<td>0.0011</td>
<td>9.8</td>
<td>10</td>
<td>1010111110</td>
</tr>
<tr>
<td>z</td>
<td>0.0005</td>
<td>10.7</td>
<td>11</td>
<td>10101111110</td>
</tr>
</tbody>
</table>
Shannon's code: Example

<table>
<thead>
<tr>
<th>X</th>
<th>p(X)</th>
<th>\log_2 \frac{1}{p(X)}</th>
<th>\ell(X)</th>
<th>C(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>0.2111</td>
<td>2.2</td>
<td>3</td>
<td>000</td>
</tr>
<tr>
<td>t</td>
<td>0.0991</td>
<td>3.3</td>
<td>4</td>
<td>0010</td>
</tr>
<tr>
<td>a</td>
<td>0.0781</td>
<td>3.6</td>
<td>4</td>
<td>0011</td>
</tr>
<tr>
<td>o</td>
<td>0.0644</td>
<td>3.9</td>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>i</td>
<td>0.0598</td>
<td>4.0</td>
<td>5</td>
<td>01010</td>
</tr>
<tr>
<td>h</td>
<td>0.0551</td>
<td>4.1</td>
<td>5</td>
<td>01011</td>
</tr>
<tr>
<td>n</td>
<td>0.0535</td>
<td>4.2</td>
<td>5</td>
<td>01100</td>
</tr>
<tr>
<td>s</td>
<td>0.0516</td>
<td>4.2</td>
<td>5</td>
<td>01101</td>
</tr>
<tr>
<td>r</td>
<td>0.0475</td>
<td>4.3</td>
<td>5</td>
<td>01110</td>
</tr>
<tr>
<td>d</td>
<td>0.0401</td>
<td>4.6</td>
<td>5</td>
<td>01111</td>
</tr>
<tr>
<td>l</td>
<td>0.0359</td>
<td>4.7</td>
<td>5</td>
<td>10000</td>
</tr>
<tr>
<td>l</td>
<td>0.0343</td>
<td>4.8</td>
<td>5</td>
<td>10001</td>
</tr>
<tr>
<td>x</td>
<td>0.0011</td>
<td>9.8</td>
<td>10</td>
<td>1010111101</td>
</tr>
<tr>
<td>j</td>
<td>0.0011</td>
<td>9.8</td>
<td>10</td>
<td>1010111101</td>
</tr>
<tr>
<td>z</td>
<td>0.0005</td>
<td>10.7</td>
<td>11</td>
<td>10101111110</td>
</tr>
</tbody>
</table>
Shannon’s code

The expected codeword length of Shannon’s code is

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

The expected codeword length of Shannon’s code is

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

In the Alice example we had

\[
E[\ell(X)] - H(X) = 4.60 - 4.03 = 0.57 \leq 1.
\]
Shannon’s code

The expected codeword length of Shannon’s code is

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

In the Alice example we had

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

Is this optimal?
Shannon’s code

The expected codeword length of Shannon’s code is

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

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!
Course Outline

What is Coding?

Symbol Codes

Entropy and Information

Kolmogorov complexity

Self-extracting files
- Definition
- Basic properties
- Invariance theorem

1. Course Outline
2. What is Coding?
3. Symbol Codes
4. Entropy and Information
5. Kolmogorov complexity
   - Self-extracting files
   - Definition
   - Basic properties
   - Invariance theorem
Kolmogorov complexity

Is the string

1010101010101010101010...10

‘simple’ or ‘complex’?
Is the string

1010101010101010101010...10

‘simple’ or ‘complex’?

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

“10 repeated \( k \) times”. 
Is the string

\[ 10101010101010101010 \ldots 10 \]

‘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

→ **Kolmogorov complexity**
### Kolmogorov complexity

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

<table>
<thead>
<tr>
<th>source string, $x$</th>
<th>$\ell(C(x))$</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa...a</td>
<td>(10000 $\times$ a)</td>
<td>368</td>
</tr>
</tbody>
</table>
Kolmogorov complexity

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

<table>
<thead>
<tr>
<th>source string, x</th>
<th>(\ell(C(x)))</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa...a</td>
<td>(10000 \times a)</td>
<td>368</td>
</tr>
<tr>
<td>aabaabbbbabbbbab...</td>
<td>(10000 random digits)</td>
<td>13456</td>
</tr>
</tbody>
</table>
```
Kolmogorov complexity

\[ \text{echo } <x> \mid \text{gzip -} \mid \text{wc -c} \quad \# \times 8 \text{ (for bits)} \]

<table>
<thead>
<tr>
<th>source string, ( x )</th>
<th>( \ell(C(x)) )</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa...a</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td>aabaabbbbabbb...</td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td>abababab...ab</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
</tbody>
</table>

Strings that follow a rule can be compressed?
### Kolmogorov complexity

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

<table>
<thead>
<tr>
<th>Source String, x</th>
<th>(\ell(C(x)))</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa...a</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td>aabaabbbabb...</td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td>abababab...ab</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td>aaa...abbb...b</td>
<td>376</td>
<td>26.6 : 1</td>
</tr>
</tbody>
</table>

Strings that follow a rule can be compressed?

Teemu Roos  
Introduction to Information-Theoretic Modeling
## Kolmogorov complexity

### Example:

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

<table>
<thead>
<tr>
<th>Source string, x</th>
<th>( \ell(C(x)) )</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aaa...a</code></td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td><code>aabaabbbabb...</code></td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td><code>abababab...ab</code></td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td><code>aaa...abbb...b</code></td>
<td>376</td>
<td>26.6 : 1</td>
</tr>
<tr>
<td><code>abbaababba...</code></td>
<td>488</td>
<td>20.5 : 1</td>
</tr>
</tbody>
</table>

Strings that follow a rule can be compressed?
### Kolmogorov complexity

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

<table>
<thead>
<tr>
<th>source string, x</th>
<th>(\ell(C(x)))</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aaa...a</code></td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td><code>aabaabbbbabba...</code></td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td><code>abababab...ab</code></td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td><code>aaa...abbb...b</code></td>
<td>376</td>
<td>26.6 : 1</td>
</tr>
<tr>
<td><code>abbaababba...</code></td>
<td>488</td>
<td>20.5 : 1</td>
</tr>
</tbody>
</table>

Strings that follow a rule can be compressed?

Teemu Roos  
Introduction to Information-Theoretic Modeling
Kolmogorov complexity

\[ \text{echo } <x> \mid \text{gzip } - \mid \text{wc } -c \quad \# \text{ times 8 (for bits)} \]

<table>
<thead>
<tr>
<th>source string, $x$</th>
<th>$\ell(C(x))$</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>$aaa \ldots a$</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td>$aabaabbbabbabbb \ldots$</td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td>$abababab \ldots ab$</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td>$aaa \ldots abbb \ldots b$</td>
<td>376</td>
<td>26.6 : 1</td>
</tr>
<tr>
<td>$abbaababba \ldots$</td>
<td>488</td>
<td>20.5 : 1</td>
</tr>
<tr>
<td>$aaabbababbabbb \ldots$</td>
<td>13416</td>
<td>0.74 : 1</td>
</tr>
</tbody>
</table>

$\pi$ follows a rule but isn’t compressible!
Kolmogorov complexity

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

<table>
<thead>
<tr>
<th>source string, x</th>
<th>( \ell(C(x)) )</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa... a</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td>aabaabbbabbab...</td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td>abababab... ab</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
<tr>
<td>aaa... abbb... b</td>
<td>376</td>
<td>26.6 : 1</td>
</tr>
<tr>
<td>abbaababba...</td>
<td>488</td>
<td>20.5 : 1</td>
</tr>
<tr>
<td>aaabbabbabbab...</td>
<td>13416</td>
<td>0.74 : 1</td>
</tr>
</tbody>
</table>

\( \pi \) follows a rule but isn't compressible!

Perhaps the problem is in \texttt{gzip}? It would be possible to write a specific program that compresses \( \pi \).
**Kolmogorov complexity**

\[
\text{echo } <x> \mid \text{gzip -} \mid \text{wc -c} \quad \# \text{ times } 8 \text{ (for bits)}
\]

<table>
<thead>
<tr>
<th>Source string, ( x )</th>
<th>( \ell(C(x)) )</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aaa...a</code></td>
<td>(10000 ( \times ) ( a ))</td>
<td>368</td>
</tr>
<tr>
<td><code>aabaabbbabb...</code></td>
<td>(10000 random digits)</td>
<td>13456</td>
</tr>
<tr>
<td><code>abababab...ab</code></td>
<td>(5000 ( \times ) ( ab ))</td>
<td>368</td>
</tr>
<tr>
<td><code>aaa...abbb...b</code></td>
<td>(5000 ( \times ) ( a ), 5000 ( \times ) ( b ))</td>
<td>376</td>
</tr>
<tr>
<td><code>abbaababba...</code></td>
<td>(1000 ( \times ) ( abbaababba ))</td>
<td>488</td>
</tr>
<tr>
<td><code>aaabbabbbabb...</code></td>
<td>(( \pi ), 0–4 ( \mapsto ) ( a ), 5–9 ( \mapsto ) ( b ))</td>
<td>13416</td>
</tr>
</tbody>
</table>

\( \pi \) follows a rule but isn’t compressible!

Perhaps the problem is in \texttt{gzip}? It would be possible to write a specific program that compresses \( \pi \).

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 $\pi$).
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 $\pi$).

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 $\pi$).

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 $\pi$).

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 \textit{pre-specified} program.

Which program? \texttt{gzip} isn’t good at compressing images (nor digits of $\pi$).

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

What about new programs? \textit{Self-extracting files}!

Do we this automatically? \textbf{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 $\pi$).

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 the Kolmogorov complexity of $x$. 

Teemu Roos  
Introduction to Information-Theoretic Modeling
Kolmogorov-kompleksisuus: mritelm

Let $U : \{0, 1\}^* \rightarrow \{0, 1\}^* \cup \infty$ 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 ($\infty$).
Kolmogorov-kompleksisuus: mritelm

Let $U : \{0, 1\}^* \rightarrow \{0, 1\}^* \cup \emptyset$ 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 ($\emptyset$).

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 $\omega^*(x)$:

$$K_U(x) = \min_{p: U(p)=x} |p|.$$
Let $U$ and $V$ be two computers. If computer $U$ is ‘rich’ enough it can ‘emulate’ computer $V$.
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 $\omega$, we have

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

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

Each of these can emulate any of the others. In contrast, gzip (or rather, gunzip) is a non-universal computer.
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. ...

In contrast, gzip (or rather, gunzip) is a non-universal computer.
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.
Lemma: For any universal computer $U$ and any other computer $V$ we have

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

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

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

where $C$ is a constant independent of $x$.

**Proof:** Let $\tau$ 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

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

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

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

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

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

where $\tilde{y}$ is a “self-delimiting” description of $y$. 

Teemu Roos
Introduction to Information-Theoretic Modeling
Conditional Kolmogorov complexity

Conditional Kolmogorov complexity

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

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

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. $K_U(0101010101...01 | n) = C.$
   
   **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 \( \pi \).
Examples

Let $n = |x|$.

1. $K_U(0101010101 \ldots 01 \mid n) = C$.
   
   *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 $\pi$.

3. $K_U(\text{English text} \mid n) \ll 1.3 \times n + C$.
   
   *Program*: Huffman code. 
   
   (The estimated entropy of English is about 1.3 bits per symbol.)
Examples

Let $n = |x|$.

1. $K_U(0101010101\ldots 01 \mid n) = C$.
   
   **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 $\pi$.

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 iterations until $z_{n+1} = z_n^2 + c > T$.  

Teemu Roos  
Introduction to Information-Theoretic Modeling
Examples
Examples (contd.):

5. $K_U(x | n) \approx n$ for almost all $x \in \{0, 1\}^n$. 
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.
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?
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?
The least *uninteresting* natural number?

Whatever it is, such a number is quite interesting.
Non-computability

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

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

**Proof:** Assume that $K_U(x)$ were computable.
Non-computability

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

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

```
print x for which K_U(x) > M.
```

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

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

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

\[
\text{print } x \text{ for which } K_U(x) > M.
\]

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

- Kolmogorov complexity, $K_U(x)$, is the length of the shortest program, $\omega$, 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.
Kolmogorov complexity: summary

To summarize:

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

- Kolmogorov complexity, $K_U(x)$, is the length of the shortest program, $\omega$, 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.

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

Teemu Roos
Introduction to Information-Theoretic Modeling
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!