Information-Theoretic Modeling

Jyrki Kivinen

Department of Computer Science, University of Helsinki

Autumn 2012
The content and form of these lecture notes have been mostly copied from the lecture notes made by Teemu Roos for the Autumn 2009 installation of this course.

Those lectures are available also on video from http://www.cs.helsinki.fi/group/cosco/Teaching/Information/2009/.
1 Administrative issues
- Course details
- Prerequisites
- What do I need to do?
- Grading and policies
1. Administrative issues
   - Course details
   - Prerequisites
   - What do I need to do?
   - Grading and policies

2. Overview of Contents
   - What is Information?
   - Why Information?
   - Information vs. Complexity
   - Information Theory
1. Administrative issues
   - Course details
   - Prerequisites
   - What do I need to do?
   - Grading and policies

2. Overview of Contents
   - What is Information?
   - Why Information?
   - Information vs. Complexity
   - Information Theory

3. Compression
   - Dots and Dashes
   - Codes as Mappings
   - Data Compression
   - Information vs. Complexity (contd.)
   - Examples (with numbers)
582650 Information-Theoretic Modeling

- An advanced studies course ("syventävät opinnot")
582650 Information-Theoretic Modeling

- An advanced studies course ("syventävät opinnot")
- *Algorithms and Machine Learning* sub-programme, optional.
582650 Information-Theoretic Modeling

- An advanced studies course ("syventävät opinnot")
- *Algorithms and Machine Learning* sub-programme, optional.
- 4 credit units.
An advanced studies course ("syventäväät opinnot")

*Algorithms and Machine Learning* sub-programme, optional.

4 credit units.

Lectures: 5 September–12 October Wed & Fri 10–12 in C222.
582650 Information-Theoretic Modeling

- An advanced studies course ("syventäväät opinnöt")
- *Algorithms and Machine Learning* sub-programme, optional.
- 4 credit units.
- Exercises: 11 September–9 October Tue 10–12 in C220.
An advanced studies course ("syventävät opinnot")

*Algorithms and Machine Learning* sub-programme, optional.

4 credit units.

Lectures: 5 September–12 October Wed & Fri 10–12 in C222.

Exercises: 11 September–9 October Tue 10–12 in C220.

Instructor: **Jyrki Kivinen**, B229a, jyrki.kivinen at cs.helsinki.fi (no fixed office hours, make an appointment by e-mail).
An advanced studies course ("syventäväät opinnot")

*Algorithms and Machine Learning* sub-programme, optional.

4 credit units.

Lectures: 5 September–12 October Wed & Fri 10–12 in C222.

Exercises: 11 September–9 October Tue 10–12 in C220.

Instructor: **Jyrki Kivinen**, B229a,

[jyrki.kivinen at cs.helsinki.fi](mailto:jyrki.kivinen at cs.helsinki.fi)

(no fixed office hours, make an appointment by e-mail).

Course assistant: **Hannes Wettig**.

[hannes.wettig at cs.helsinki.fi](mailto:hannes.wettig at cs.helsinki.fi)
Resources

There is no required textbook on the course, but the following are recommended.

- **Highly recommended**: Cover & Thomas, *Elements of Information Theory*,
- Grünwald, *The Minimum Description Length Principle*,
- Solomon, *Data Compression: The Complete Reference*. 
There is also a related project:

- 2 credit units.
582651 Project in Information-Theoretic Modeling

There is also a related project:

- 2 credit units.
- Period II
582651 Project in Information-Theoretic Modeling

There is also a related project:

- 2 credit units.
- Period II
- This course is a prerequisite.
582651 Project in Information-Theoretic Modeling

There is also a related project:

- 2 credit units.
- Period II
- This course is a prerequisite.
- Together they replace the old *Three Concepts: Information* course—both old and new version cannot be included in your degree.
582651 Project in Information-Theoretic Modeling

There is also a related project:

- 2 credit units.
- Period II
- This course is a prerequisite.
- Together they replace the old *Three Concepts: Information* course—both old and new version cannot be included in your degree.
- Groups of 2–3 persons.
582651 Project in Information-Theoretic Modeling

There is also a related project:

- 2 credit units.
- Period II
- This course is a prerequisite.
- Together they replace the old *Three Concepts: Information* course—both old and new version cannot be included in your degree.
- Groups of 2–3 persons.
- Task: compress data.
582651 Project in Information-Theoretic Modeling

There is also a related project:

- 2 credit units.
- Period II
- This course is a prerequisite.
- Together they replace the old *Three Concepts: Information* course—both old and new version cannot be included in your degree.
- Groups of 2–3 persons.
- Task: compress data.
- Best compressor wins! Intermediate results announced periodically.
There is also a related project:
- 2 credit units.
- Period II
- This course is a prerequisite.
- Together they replace the old *Three Concepts: Information* course—both old and new version cannot be included in your degree.
- Groups of 2–3 persons.
- Task: compress data.
- Best compressor wins! Intermediate results announced periodically.
- Programming + report.
Prerequisites

No formal prerequisites **but** you will need
Prerequisites

No formal prerequisites **but** you will need

- Calculus: integrals, derivatives, convergence, ...
No formal prerequisites **but** you will need

- Calculus: integrals, derivatives, convergence, ...
- Probability theory: joint & conditional distributions, expectations, law of large numbers, ...
Prerequisites

No formal prerequisites but you will need

- Calculus: integrals, derivatives, convergence, ...
- Probability theory: joint & conditional distributions, expectations, law of large numbers, ...
- Programming: language is up to you (but need to work in groups in project).
What do I need to do?

- Weekly exercises:
What do I need to do?

- Weekly exercises:
  - Mathematical problems.
What do I need to do?

- Weekly exercises:
  - Mathematical problems.
  - Programming tasks.
What do I need to do?

- **Weekly exercises:**
  - Mathematical problems.
  - Programming tasks.

- **Course exam on 18 October at 9:00am.**
What do I need to do?

- Weekly exercises:
  - Mathematical problems.
  - Programming tasks.
- Course exam on 18 October at 9:00am.

You do *not* have to attend the classes, unless otherwise stated. However, it is recommend that you do.
What do I need to do?

- Weekly exercises:
  - Mathematical problems.
  - Programming tasks.
- Course exam on 18 October at 9:00am.

You do *not* have to attend the classes, unless otherwise stated. However, it is recommend that you do.

- Alternatively, you can just take a separate exam later (next one is 23 November).
Grading

The course grading is based on:

- Exercises (40 %)

---

Jyrki Kivinen  Information-Theoretic Modeling
The course grading is based on:

1. Exercises (40 %)
2. Exam (60 %)
Grading

The course grading is based on:

1. Exercises (40 %)
2. Exam (60 %)

Minimum 50 % of exercises have to be solved (or at least seriously attempted).
Grading

The course grading is based on:

1. Exercises (40 %)
2. Exam (60 %)

Minimum 50 % of exercises have to be solved (or at least seriously attempted).

Feel free to discuss homework problems in groups.
Grading

The course grading is based on:

1. Exercises (40 %)
2. Exam (60 %)

Minimum 50 % of exercises have to be solved (or at least seriously attempted).

Feel free to discuss homework problems in groups. However, each student is expected to

- independently write down his/her solutions to theory problems
- implement all programming tasks by him/herself.
1 Administrative issues
   • Course details
   • Prerequisites
   • What do I need to do?
   • Grading and policies

2 Overview of Contents
   • What is Information?
   • Why Information?
   • Information vs. Complexity
   • Information Theory

3 Compression
   • Dots and Dashes
   • Codes as Mappings
   • Data Compression
   • Information vs. Complexity (contd.)
   • Examples (with numbers)
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 vs. Information vs. 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 vs. Information vs. 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 vs. Information vs. 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 vs. Information vs. Knowledge.

- Information technology.

- Physical information.

- This course: measuring *the amount* of information in data, and using such measures for automatically building *models*. 
The amount of data around us is exploding – internet!
Why Information?

- The amount of data around us is exploding – internet!
- Need to *store, transmit, and process* it efficiently.
Why Information?

- The amount of data around us is exploding – internet!
- Need to *store, transmit, and process* it efficiently.
- Wish to *understand* more and more complex phenomena.
The amount of data around us is exploding – internet!
Need to store, transmit, and process it efficiently.
Wish to understand more and more complex phenomena.
Computer science: make things automatic (intelligent).
On this course we do not define the term *information* as such. Often a useful intuition is to think of the *information content* of some piece of data as the amount of bits needed to represent its *relevant* features.

“Relevant” here is of course not well defined. The point is that generally for example random strings are not considered to contain much information, although they cannot be compressed much (i.e. require many bits to encode).
One way of thinking about this is

\[ \text{Complexity} = \text{Information} + \text{Noise} \]

so that for example random strings are complex, but the complexity mainly comes from noise, not information. From a bit different points of view,

\[ \text{Complexity} = \text{Regularity} + \text{Randomness} \]

or

\[ \text{Complexity} = \text{Algorithm} + \text{Compressed file}. \]
"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)
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.
Administrative issues
- Course details
- Prerequisites
- What do I need to do?
- Grading and policies

Overview of Contents
- What is Information?
- Why Information?
- Information vs. Complexity
- Information Theory

Compression
- Dots and Dashes
- Codes as Mappings
- Data Compression
- Information vs. Complexity (contd.)
- Examples (with numbers)
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>
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 ●: 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).
Coding Game

© 1989 A.G. Reinhold.

Samuel F.M. Morse (1791–1872)
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.

Morse code

Coding rate: 137

\( \frac{42}{3.26} \approx 3 \).

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?
Lossless compression: injective mapping

Source strings

Code strings

Only lossless codes are uniquely decodable.
Codes as Mappings

Lossless compression: injective mapping

Lossy compression: non-injective mapping

Source strings

Code strings

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

```
<table>
<thead>
<tr>
<th>general purpose</th>
<th>gzip</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bzip</td>
</tr>
<tr>
<td>image</td>
<td>png</td>
</tr>
<tr>
<td></td>
<td>jpeg</td>
</tr>
</tbody>
</table>
```
Examples

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

Information-Theoretic Modeling
Examples

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

- **General purpose**
  - gzip
  - bzip
- **Image**
  - png
  - jpeg
- **Music**
  - mp3
- **Video**
  - mpeg

**Lossless**

**Lossy**
Examples

<table>
<thead>
<tr>
<th>Format</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Purpose</strong></td>
<td></td>
</tr>
<tr>
<td>gzip</td>
<td>~1:3</td>
</tr>
<tr>
<td>bzip</td>
<td>~1:3.5</td>
</tr>
<tr>
<td><strong>Image</strong></td>
<td></td>
</tr>
<tr>
<td>png</td>
<td>~1:2.5</td>
</tr>
<tr>
<td>jpeg</td>
<td>~1:25</td>
</tr>
<tr>
<td><strong>Music</strong></td>
<td></td>
</tr>
<tr>
<td>mp3</td>
<td>~1:12</td>
</tr>
<tr>
<td><strong>Video</strong></td>
<td></td>
</tr>
<tr>
<td>mpeg</td>
<td>~1:30</td>
</tr>
</tbody>
</table>

Lossless vs. Lossy compression ratios.
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 $2^0 + 2^1 + 2^2 + \ldots + 2^{n-k-1} = 2^{n-k} - 1$. 

Jyrki Kivinen

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 $2^0 + 2^1 + 2^2 + \ldots + 2^{n-k-1} = 2^{n-k} - 1$. 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.
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 0.0000000000000000000000000000001 % of files are compressible by 100 bits.
How is it possible?

Why was the compression ratio greater than one in all the examples we saw?

What are those rare files that are compressible?

Why are the files we use in practice so often compressible?
Compression

```
echo <x> | gzip - | wc -c  # multiply by 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>
</tbody>
</table>
Compression

```bash
echo <x> | gzip - | wc -c  # multiply by 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>(10000 \times a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aabaabbbabbbbbb... (10000 random letters)</td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Compression

```
echo <x> | gzip - | wc -c  # multiply by 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>aabaabbbbabbbbb... (10000 random letters)</td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td>abababab...ab (5000 x ab)</td>
<td>368</td>
<td>27.2 : 1</td>
</tr>
</tbody>
</table>
### Compression

```
echo <x> | gzip - | wc -c  # multiply by 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>aabaabbababbbab...</code></td>
<td>13456</td>
<td>0.74 : 1</td>
</tr>
<tr>
<td><code>ababababa...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>
</tbody>
</table>

Strings following a rule are compressible?
## Compression

```
echo <x> | gzip - | wc -c  # multiply by 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><code>368</code></td>
<td><code>27.2 : 1.</code></td>
</tr>
<tr>
<td><code>aabaabbbbabbbb</code>...</td>
<td><code>13456</code></td>
<td><code>0.74 : 1</code></td>
</tr>
<tr>
<td><code>abababab...ab</code></td>
<td><code>368</code></td>
<td><code>27.2 : 1</code></td>
</tr>
<tr>
<td><code>aaa...abbb...b</code></td>
<td><code>376</code></td>
<td><code>26.6 : 1</code></td>
</tr>
<tr>
<td><code>abbaababba...</code></td>
<td><code>488</code></td>
<td><code>20.5 : 1</code></td>
</tr>
</tbody>
</table>

Strings following a rule are compressible?
Compression

```
echo <x> | gzip - | wc -c  # multiply by 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>aabaabbbbabbbbb...</td>
<td>(10000 random letters)</td>
<td>13456</td>
</tr>
<tr>
<td>abababab...ab</td>
<td>(5000 ( \times ) ab)</td>
<td>368</td>
</tr>
<tr>
<td>aaa...abb...b</td>
<td>(5000 ( \times ) a, 5000 ( \times ) b)</td>
<td>376</td>
</tr>
<tr>
<td>abbaababba...</td>
<td>(1000 ( \times ) abbaababba)</td>
<td>488</td>
</tr>
</tbody>
</table>

Strings following a rule are compressible?
Compression

```bash
echo <x> | gzip - | wc -c  # multiply by 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>$aabaabbbabbbb\ldots$</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>$aaabbabbbabbb\ldots$</td>
<td>13416</td>
<td>0.74 : 1</td>
</tr>
</tbody>
</table>

$\pi$ follows a rule but isn’t compressed!
Compression

\[ \text{echo} \ <x> \ | \ \text{gzip} \ - \ | \ \text{wc} \ -c \quad \# \text{ multiply by 8 for bits} \]

<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>aabaabbbbabbbbab</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>aaabbababbb...</td>
<td>13416</td>
<td>0.74 : 1</td>
</tr>
</tbody>
</table>

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

Maybe it’s just \texttt{gzip}? It would be possible to create a \textit{special program} to compress \( \pi \) into a short file.
Compression

```shell
echo <x> | gzip - | wc -c  # multiply by 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>aabaabbbbabbbbbb...</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>
<tr>
<td><code>aaabbabbabb...</code></td>
<td>13416</td>
<td>0.74 : 1</td>
</tr>
</tbody>
</table>

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

Maybe it’s just `gzip`? It would be possible to create a special program to compress \( \pi \) into a short file.

But what does it mean to compress an individual string?
An individual string is “simple” (as opposed to “complex”) if it can be compressed into a small file by a *prespecified* program.
Information

An individual string is “simple” (as opposed to “complex”) if it can be compressed into a small file by a *prespecified* program.

But which program? *gzip* is not good for images (or for $\pi$).
Information

An individual string is “simple” (as opposed to “complex”) if it can be compressed into a small file by a *prespecified* program.

But which program? *gzip* is not good for images (or for $\pi$).

We can use several compressors if we prefix the code string by an index of the used program.
Information

An individual string is “simple” (as opposed to “complex”) if it can be compressed into a small file by a *prespecified* program.

But which program? *gzip* is not good for images (or for \( \pi \)).

We can use several compressors if we prefix the code string by an index of the used program.

How about new compressors?
Information

An individual string is “simple” (as opposed to “complex”) if it can be compressed into a small file by a *prespecified* program.

But which program? gzip is not good for images (or for $\pi$).

We can use several compressors if we prefix the code string by an index of the used program.

How about new compressors? *Self-extracting files!*
An individual string is “simple” (as opposed to “complex”) if it can be compressed into a small file by a *prespecified* program.

But which program? gzip is not good for images (or for $\pi$).

We can use several compressors if we prefix the code string by an index of the used program.

How about new compressors? *Self-extracting files!*

Can it be made automatic? Find the shortest program to print $x$. 
Information

An individual string is “simple” (as opposed to “complex”) if it can be compressed into a small file by a *prespecified* program.

But which program? gzip is not good for images (or for $\pi$).

We can use several compressors if we prefix the code string by an index of the used program.

How about new compressors? *Self-extracting files!*

Can it be made automatic? Find the shortest program to print $x$. **No.** *Kolmogorov complexity.*
Information

An individual string is “simple” (as opposed to “complex”) if it can be compressed into a small file by a *prespecified* program.

But which program? *gzip* is not good for images (or for $\pi$).

We can use several compressors if we prefix the code string by an index of the used program.

How about new compressors? *Self-extracting files!*

Can it be made automatic? Find the shortest program to print $x$. **No.** *Kolmogorov complexity.*

Project
Next on course

Basics of (statistical) information theory

- introductory examples
- mathematical background
- basic concepts: entropy, mutual information etc.
Source coding

Alice, Bob, Cecilia and David play frequent tennis tournaments. We want to encode the sequence of winners using the least possible number of bits. For example, denoting by $A$ the event that Alice wins, etc., we could choose codes

\[
C(A) = 00 \quad C(B) = 01 \\
C(C) = 10 \quad C(D) = 11.
\]

A sequence where first Bob wins, then Alice twice in row, and then David, would be encoded as $01000011$. The code length is four bits per result.
Source coding (cont.)

Suppose now that we additionally know the winning probabilities are as follows: Alice wins 50% of the time, Bob 25%, and Cecilia and David 12.5% each. That is,

\[ P(A) = \frac{1}{2} \quad P(B) = \frac{1}{4} \]
\[ P(C) = \frac{1}{8} \quad P(D) = \frac{1}{8}. \]

Consider the following encoding:

\[ C(A) = 0 \quad C(B) = 10 \]
\[ C(C) = 110 \quad C(D) = 111. \]

Code lengths are not constant any more. However sequences of code words still have a unique meaning. For example, 11011110 can only mean the sequence CDB.
Source coding (cont.)

The *expected* code length per result is now

$$\sum_{x \in \{A,B,C,D\}} p(x) \ell(C(x))$$

$$= \frac{1}{2} \cdot \ell(0) + \frac{1}{8} \cdot \ell(10) + \frac{1}{8} \cdot \ell(110) + \frac{1}{2} \cdot \ell(111)$$

$$= \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = \frac{7}{4}.$$

The expected code length $7/4$ bits per results is actually the same as the *entropy* of the distribution from which the results come.

This is because we chose code lengths such that

$$\ell(C(x)) = \log_2(1/P(x)),$$

which makes the code in some sense optimal for this source.
**Question:** Can we do better?

**Answer:** Perhaps, *if* we make additional assumptions.

---

**Source coding (cont.)**

*Question:* Can we do better?

*Answer:* Perhaps, *if* we make additional assumptions.
**Source coding (cont.)**

*Question:* Can we do better?

*Answer:* Perhaps, *if* we make additional assumptions.

*Question:* Can this be implemented efficiently?

*Answer:* Yes.
Source coding (cont.)

*Question:* Can we do better?
*Answer:* Perhaps, *if* we make additional assumptions.

*Question:* Can this be implemented efficiently?
*Answer:* Yes.

*Question:* What if \( \log_2(P(X)) \) is not an integer?
*Answer:* In this context we can give a reasonable interpretation to fractions of a bit.
Source coding (cont.)

*Question:* Can we do better?
*Answer:* Perhaps, *if* we make additional assumptions.

*Question:* Can this be implemented efficiently?
*Answer:* Yes.

*Question:* What if $\log_2(P(X))$ is not an integer?
*Answer:* In this context we can give a reasonable interpretation to fractions of a bit.

We’ll return to all these questions later.
Suppose we want to transmit a sequence of symbols from the 26-letter alphabet A, B, C, . . . , Z. They are transmitted over a noisy channel where with probability 1/2, there is no error, but with probability 1/2, the symbol gets corrupted to the next one in the alphabet (with Z rolling over to A).
Noisy channel coding (cont.)

Denoting the transmitted symbol by $X$ and the received symbol by $\hat{X}$, we thus have

\[
P(\hat{X} = A \mid X = A) = 1/2 \quad P(\hat{X} = B \mid X = A) = 1/2
\]
\[
P(\hat{X} = B \mid X = B) = 1/2 \quad P(\hat{X} = C \mid X = B) = 1/2
\]
\[
\ldots
\]
\[
P(\hat{X} = Z \mid X = Z) = 1/2 \quad P(\hat{X} = A \mid X = Z) = 1/2.
\]

For example, transmission INFORMATION might be received as IOGPRMBTIOO.
Noisy channel coding (cont.)

transmitted

\[
\begin{align*}
A & \rightarrow B \\
B & \rightarrow C \\
C & \rightarrow Y \\
Y & \rightarrow Z
\end{align*}
\]

received

\[
\begin{align*}
A & \rightarrow 1/2 \rightarrow A \\
B & \rightarrow 1/2 \rightarrow B \\
C & \rightarrow 1/2 \rightarrow C \\
Y & \rightarrow 1/2 \rightarrow Y \\
Z & \rightarrow 1/2 \rightarrow Z
\end{align*}
\]
Suppose now we are willing to increase the number of transmissions in order to allow error correction.
Suppose now we are willing to increase the number of transmissions in order to allow error correction.

First notice that if we restrict our transmissions to symbols A, C, E, G, . . . , Y, then we can always deduce the correct symbol.
Suppose now we are willing to increase the number of transmissions in order to allow error correction.

First notice that if we restrict our transmissions to symbols A, C, E, G, . . . , Y, then we can always deduce the correct symbol.

For example, if we receive D, we know that it’s a corrupted C, since D itself is not on the list of symbols we use.
We use this to devise the following encoding, where we use two symbols to encode each original symbol:

\[
\begin{align*}
C(A) &= AA & C(B) &= AC \\
C(C) &= CA & C(D) &= CC \\
C(E) &= EA & C(F) &= EC \\
\cdots & & \cdots
\end{align*}
\]

Now in each block of two symbols in the code, if the first received symbol is for example E or F, we know that E or F was transmitted, and the second symbol tells which one.
Noisy channel coding (cont.)

Thus we can have full error correction at the cost of doubling the length of the transmissions. Can we do better?
Thus we can have full error correction at the cost of doubling the length of the transmissions. Can we do better?

If the alphabet size is $2^b$, then one symbol takes $b$ bits.
Thus we can have full error correction at the cost of doubling the length of the transmissions. Can we do better?

If the alphabet size is $2^b$, then one symbol takes $b$ bits.

Let’s assume for the moment this somehow also makes sense if the alphabet size is not a power of 2.
Thus we can have full error correction at the cost of doubling the length of the transmissions. Can we do better?

If the alphabet size is $2^b$, then one symbol takes $b$ bits.

Let’s assume for the moment this somehow also makes sense if the alphabet size is not a power of 2.

This can be made more precise (and practical) by encoding longer blocks of 1 symbols together.
Thus for alphabet size 26, we need to transmit $\log_2 26 \approx 4.7$ bits per symbol.
Noisy channel coding (cont.)

Thus for alphabet size 26, we need to transmit $\log_2 26 \approx 4.7$ bits per symbol.

We just argued that by transmitting one symbol over the noisy channel, we can communicate without error one symbol from a set of 13. This is worth $\log_2 13 \approx 3.7$ bits.
Thus for alphabet size 26, we need to transmit $\log_2 26 \approx 4.7$ bits per symbol.

We just argued that by transmitting one symbol over the noisy channel, we can communicate without error one symbol from a set of 13. This is worth $\log_2 13 \approx 3.7$ bits.

So intuitively, we should be able to correct all errors by increasing the message length by a factor of $\log_2 26 / \log_2 13 \approx 1.27$. 
Noisy channel coding (cont.)

Thus for alphabet size 26, we need to transmit $\log_2 26 \approx 4.7$ bits per symbol.

We just argued that by transmitting one symbol over the noisy channel, we can communicate without error one symbol from a set of 13. This is worth $\log_2 13 \approx 3.7$ bits.

So intuitively, we should be able to correct all errors by increasing the message length by a factor of $\log_2 26 / \log_2 13 \approx 1.27$.

This would be much better than the factor 2 of the previous encoding.
Noisy channel coding (cont.)

To get an idea how this could be done, we consider coding blocks of 3 symbols using 4 symbols per block, giving ratio $4/3 \approx 1.33$.

Consider a block $s_1s_2s_3$ consisting of three symbols from alphabet \{ A, B, C, \ldots, Z \}. We start by encoding it as a block $t_1t_2t_3t_4t_5t_6$ of six symbols from \{ A, C, E, \ldots, Y \} using the encoding $C$. 

To get an idea how this could be done, we consider coding blocks of 3 symbols using 4 symbols per block, giving ratio $\frac{4}{3} \approx 1.33$.

Consider a block $s_1s_2s_3$ consisting of three symbols from alphabet \{ A, B, C, \ldots, Z \}. We start by encoding it as a block $t_1t_2t_3t_4t_5t_6$ of six symbols from \{ A, C, E, \ldots, Y \} using the encoding $C$.

For example, if $s_1s_2s_3 = CBZ$, then $t_1t_2t_3t_4t_5t_6 = CAACYC$. 
To get an idea how this could be done, we consider coding blocks of 3 symbols using 4 symbols per block, giving ratio $4/3 \approx 1.33$.

Consider a block $s_1 s_2 s_3$ consisting of three symbols from alphabet $\{A, B, C, \ldots, Z\}$. We start by encoding it as a block $t_1 t_2 t_3 t_4 t_5 t_6$ of six symbols from $\{A, C, E, \ldots, Y\}$ using the encoding $C$.

For example, if $s_1 s_2 s_3 = CBZ$, then $t_1 t_2 t_3 t_4 t_5 t_6 = CAACYC$.

Notice that $t_2$, $t_4$ and $t_6$ take only values A and C. There are only 8 possible combinations $t_2 t_4 t_6$. We encode these using the symbols A, C, E, G, I, K, M and O. Let’s denote this code for $t_2 t_4 t_6$ by $t_7$. 
To get an idea how this could be done, we consider coding blocks of 3 symbols using 4 symbols per block, giving ratio $4/3 \approx 1.33$.

Consider a block $s_1s_2s_3$ consisting of three symbols from alphabet \{ A, B, C, \ldots, Z \}. We start by encoding it as a block $t_1t_2t_3t_4t_5t_6$ of six symbols from \{ A, C, E, \ldots, Y \} using the encoding $C$.

For example, if $s_1s_2s_3 = \text{CBZ}$, then $t_1t_2t_3t_4t_5t_6 = \text{CAACYC}$.

Notice that $t_2$, $t_4$ and $t_6$ take only values A and C. There are only 8 possible combinations $t_2t_4t_6$. We encode these using the symbols A, C, E, G, I, K, M and O. Let’s denote this code for $t_2t_4t_6$ by $t_7$.

If we now transmit $t_1t_3t_5t_7$, we can recover $s_1s_2s_3$ error-free.
Noisy channel coding (cont.)

The channel coding idea can be generalised to situations where the noise does not have such a nice structure.
Noisy channel coding (cont.)

The channel coding idea can be generalised to situations where the noise does not have such a nice structure.

It turns out that (asymptotically), the achievable error-free transmission rate depends on a quantity called *mutual information* between the transmitted and received symbol.
Noisy channel coding (cont.)

The channel coding idea can be generalised to situations where the noise does not have such a nice structure.

It turns out that (asymptotically), the achievable error-free transmission rate depends on a quantity called *mutual information* between the transmitted and received symbol.

These two examples illustrate the role of *redundancy* in communication theory.
Noisy channel coding (cont.)

The channel coding idea can be generalised to situations where the noise does not have such a nice structure.

It turns out that (asymptotically), the achievable error-free transmission rate depends on a quantity called mutual information between the transmitted and received symbol.

These two examples illustrate the role of redundancy in communication theory.

In the tennis tournament example, we used the properties of the information source to remove redundancy to compress the data.
The channel coding idea can be generalised to situations where the noise does not have such a nice structure.

It turns out that (asymptotically), the achievable error-free transmission rate depends on a quantity called *mutual information* between the transmitted and received symbol.

These two examples illustrate the role of *redundancy* in communication theory.

In the tennis tournament example, we used the properties of the information source to *remove* redundancy to compress the data.

In the noisy typewriter example, we *added* redundancy using the properties of the channel to recover errors.