Compression and Information

Marek Rychlik

Department of Mathematics
University of Arizona

February 18, 2009
Sending random messages

- Alphabet: $A = \{a_1, a_2, \ldots, a_N\}$ where $N$ is typically finite, but sometimes $N = \infty$ is admissible.
- Probability distribution: $P : A \rightarrow (0, 1]$, so that

$$\sum_{a \in A} P(a) = 1.$$ 

- Random message: a sequence $M = s_1s_2\ldots s_L$ where for $j = 1, 2, \ldots, N$ we have $s_j \in A$.
- $\ell(M)$ will denote the length ($L$) of the message $M$.
- $A^L$ (the Cartesian product) denotes the set of all messages of length $L$.
- $A^+$ denotes the set of all finite messages in alphabet $A$, i.e. $A^+ = \bigcup_{L=0}^{\infty} A^L$. 

Marek Rychlik
CGB
Bits

- The term *bit* stands for a *binary digit* and it is either 0 or 1.
- It is a *normalized unit of information*.
- A random message of length $N$ with an alphabet of $L$ symbols can be easily encoded in

$$\left\lceil \log_2 L \right\rceil \cdot N$$

bits.
Coding and lossless coding

Definition

1. A code is a function

\[ \mathcal{C} : A^* \rightarrow B^* \]

i.e. a map from the set of all finite length messages in alphabet \( A \) to the set of all finite length messages in another alphabet \( B \).

2. A code \( \mathcal{C} : A^* \rightarrow B^* \) is called a lossless code if \( \mathcal{C} \) is 1:1 (but possibly not onto).
Comments on lossless coding

- Losslessness implies that the encoded message can be uniquely decoded.
- Not every message in the target alphabet may be decoded.
- In practice, the decoding algorithm may decode some sequences which are not in the image $C(A^+)$, i.e. may perform a mapping

$$\mathcal{D} : S \subseteq B^+ \to A^+$$

so that $S \supseteq C(A)$ and

$$\mathcal{D} \circ C = id_{A^+}.$$
Symbol codes

**Definition**

- A *symbol code* is a mapping

\[ C : A \rightarrow B^+ \]

of the alphabet to messages in another alphabet.

- The *extension* of the symbol code \( C \) is a code obtained by concatenation:

\[ s_1 s_2 \ldots s_L \rightarrow C(s_1)C(s_2)\ldots C(s_L). \]

- A *symbol code* is *lossless* iff \( C \) is 1:1.

- A *binary code* is a code where the target alphabet \( B \) is \( \{0, 1\} \).
Simple properties of symbol codes

- The resulting message has typically different length from the original message.
- We may define the *length function* of the code:

\[ a \mapsto \ell(C(a)). \]

- The code is *uniform* if the lengths of \( C(s) \) are identical for all \( s \in A \), i.e. \( \ell \circ C \) is constant.
Simple properties of symbol codes

- The resulting message has typically different length from the original message.
- We may define the *length function* of the code:

  \[ a \mapsto \ell(C(a)). \]

- The code is *uniform* if the lengths of \( C(s) \) are identical for all \( s \in A \), i.e. \( \ell \circ C \) is constant.
Simple properties of symbol codes

- The resulting message has typically different length from the original message.
- We may define the *length function* of the code:

  \[
  a \mapsto \ell(C(a)).
  \]

- The code is *uniform* if the lengths of \( C(s) \) are identical for all \( s \in A \), i.e. \( \ell \circ C \) is constant.
Trivial uniform binary code

Example

The alphabet $A = \{a, b, c\}$. Three letters can be mapped 1:1 to sequences of 2 bits, e.g:

\[
\begin{align*}
  a & \rightarrow 00 \\
  b & \rightarrow 10 \\
  c & \rightarrow 01
\end{align*}
\]

Thus,

\[
abcba \rightarrow 0010011000
\]

The decoding is also trivial: we consider pairs of consecutive digits and recover the original symbol by inverse lookup.
Non-uniform codes

- Non-uniform codes can result in shorter messages by assigning shorter codes to more probable messages.
- The theory of non-uniform codes connects the combinatorics of coding with probability theory.
An example of an optimal code

**Example**

Let $A = \{a, b, c, d\}$. Let

\[ P(a) = \frac{1}{2}, \ P(b) = \frac{1}{4}, \ P(c) = P(d) = \frac{1}{8}. \]

The trivial uniform binary code yields two bits per symbol i.e. a message of length $L$ is coded in exactly $2L$ bits.

The following code yields only 1.75 bits per symbol in every message which has exactly 1/2 a’s, 1/4 b’s and 1/8 of c’d and d’s:

\[ a \rightarrow 0, \ b \rightarrow 10, \ c \rightarrow 110, \ d \rightarrow 111. \]
Notes on the compression example

- We note that $\ell(C(s)) = -\log_2 P(s)$ for this code. This is an example of a Huffman code.
- A message which has exactly $1/2$ a’s, $1/4$ b’s and $1/8$ of c’d and d’s is coded in
  
  \[
  \frac{L}{2} \cdot 1 + \frac{L}{4} \cdot 2 + 2 \cdot \frac{L}{8} \cdot 3 = 1.75L \text{ bits}
  \]
- The invertibility of the code follows from the prefix property.
- If the message composition does not conform to the probability distribution, it still can be uniquely decoded, but the length of the encoded message may be longer then $1.75L$. 

Marek Rychlik
CGB
The prefix property

**Definition**

We say that a symbol code $C : A \rightarrow B^+$ has the *prefix property* if the code of each symbol is not a prefix of the code of any other symbol.

- Every prefix code is lossless.
- There are lossless symbol codes which do not have the prefix property. The disadvantage of such codes is that they require looking ahead in the code before decoding a symbol.
An decoding example

- **Alphabet:** $A = \{a, b, c, d\}$.
- **Symbol codes:**
  - $a \rightarrow 0$
  - $b \rightarrow 10$
  - $c \rightarrow 110$
  - $d \rightarrow 111$

- **Code:** 0101101110
- **Decoded message:** Marek Rychlik CGB
An decoding example

- **Alphabet:** $A = \{a, b, c, d\}$.
- **Symbol codes:**

  
  $\begin{align*}
  a & \rightarrow 0 \\
  b & \rightarrow 10 \\
  c & \rightarrow 110 \\
  d & \rightarrow 111
  \end{align*}$

- **Code:** 0101101110
- **Decoded message:** Marek Rychlik CGB
An decoding example

- Alphabet: \( A = \{a, b, c, d\} \).
- Symbol codes:

  \[
  \begin{align*}
  a & \rightarrow 0 \\
  b & \rightarrow 10 \\
  c & \rightarrow 110 \\
  d & \rightarrow 111
  \end{align*}
  \]

- Code: 0101101110

  ...

- Decoded message:

  ...

Marek Rychlik  
CGB
An decoding example

- **Alphabet**: $A = \{a, b, c, d\}$.
- **Symbol codes**:
  
  $\begin{align*}
  a & \rightarrow 0 \\
  b & \rightarrow 10 \\
  c & \rightarrow 110 \\
  d & \rightarrow 111
  \end{align*}$

- **Code**: 0101101110

- **Decoded message**: $a \ldots$
An decoding example

- **Alphabet**: $A = \{a, b, c, d\}$.
- **Symbol codes**:

  \[
  \begin{align*}
  a & \rightarrow 0 \\
  b & \rightarrow 10 \\
  c & \rightarrow 110 \\
  d & \rightarrow 111
  \end{align*}
  \]

- **Code**: 0101101110
  
  01…

- **Decoded message**: a?…
An decoding example

- Alphabet: $A = \{a, b, c, d\}$.
- Symbol codes:

  $a \rightarrow 0$
  $b \rightarrow 10$
  $c \rightarrow 110$
  $d \rightarrow 111$

- Code: 0101101110

  010

- Decoded message: $ab \ldots$
An decoding example

- **Alphabet:** \( A = \{a, b, c, d\} \).
- **Symbol codes:**

  \[
  \begin{align*}
  a & \rightarrow 0 \\
  b & \rightarrow 10 \\
  c & \rightarrow 110 \\
  d & \rightarrow 111 \\
  \end{align*}
  \]

  Code: 0101101110

  \[
  \begin{align*}
  0101 \ldots \\
  \end{align*}
  \]

  Decoded message:

  \[
  ab? \ldots 
  \]
An decoding example

- **Alphabet:** $A = \{a, b, c, d\}$.
- **Symbol codes:**

  
  $a \rightarrow 0$
  
  $b \rightarrow 10$
  
  $c \rightarrow 110$
  
  $d \rightarrow 111$

- **Code:** 0101101110

  
  01011 ... 

- **Decoded message:**

  
  $ab? ...$
An decoding example

- **Alphabet:** $A = \{a, b, c, d\}$.
- **Symbol codes:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$0$</td>
</tr>
<tr>
<td>$b$</td>
<td>$10$</td>
</tr>
<tr>
<td>$c$</td>
<td>$110$</td>
</tr>
<tr>
<td>$d$</td>
<td>$111$</td>
</tr>
</tbody>
</table>

- **Code:** 0101101110

010110...  

- **Decoded message:**

  $abc...$
An decoding example

- **Alphabet:** \( A = \{ a, b, c, d \} \).
- **Symbol codes:**
  
  \[
  \begin{align*}
  a & \rightarrow 0 \\
  b & \rightarrow 10 \\
  c & \rightarrow 110 \\
  d & \rightarrow 111
  \end{align*}
  \]

- **Code:** 0101101110

  \[0101101\ldots\]

  Decoded message:

  \[abc?\ldots\]
An decoding example

Alphabet: \( A = \{a, b, c, d\} \).

Symbol codes:

\[
\begin{align*}
  a & \rightarrow 0 \\
  b & \rightarrow 10 \\
  c & \rightarrow 110 \\
  d & \rightarrow 111
\end{align*}
\]

Code: 0101101110

0101101110

Decoded message:

\( abc? \ldots \)
An decoding example

- Alphabet: \( A = \{a, b, c, d\} \).
- Symbol codes:
  
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>0</td>
</tr>
<tr>
<td>(b)</td>
<td>10</td>
</tr>
<tr>
<td>(c)</td>
<td>110</td>
</tr>
<tr>
<td>(d)</td>
<td>111</td>
</tr>
</tbody>
</table>

- Code: 0101101110
  
  010110111...  

- Decoded message: \( abcd \ldots \)
An decoding example

- Alphabet: $A = \{a, b, c, d\}$.
- Symbol codes:

  $a \rightarrow 0$
  $b \rightarrow 10$
  $c \rightarrow 110$
  $d \rightarrow 111$

- Code: 0101101110

- Decoded message: $abcda$
The expected length of the code

**Definition**

Given a probability distribution $P : A \rightarrow [0, 1]$ on the alphabet $A$, the *expected length of a binary symbol code* $C : A \rightarrow \{0, 1\}^+$ is defined as:

$$\mathbb{E}(\ell \circ C) = \sum_{a \in A} \ell(C(a)) \cdot P(a).$$
Kraft inequality (extended variant)

**Theorem**

*If A is countable and \( C : A \to \{0, 1\}^+ \) is a lossless symbol code then*

\[
\sum_{a \in A} 2^{-D(a)} \leq 1.
\]

*where \( D(a) = \ell(C(a)) \).*
The prefix tree of a code

**Definition**

The *prefix tree* $T(C)$ of the code $C$ is defined as follows:

- The nodes of the tree $T(C)$ are in 1:1 correspondence to all prefixes of all codes $\{C(a)\}_{a \in A}$.
- The parent of a node of a prefix $b_0b_1 \ldots b_{d-1}b_d$ of length $d$, $(b_i \in \{0, 1\}, \ d \geq 1)$ is the node corresponding to the prefix of length $d - 1$, i.e. $b_0b_1 \ldots b_{d-1}$.
- The full codes $C(a)$ are their own prefixes, and are not prefixes of any other codes; thus they are in 1:1 correspondence with the leaves of the tree.
- The root of the tree corresponds to the *empty code*.
An example of a prefix tree

Example

- Alphabet: $A = \{a, b, c, d\}$.
- $C$ defined by: $a \rightarrow 0$, $b \rightarrow 10$, $c \rightarrow 110$, $d \rightarrow 111$. 

```
∅ 0(a) 1
    /   \
 10(b) 11
     /     \
110(c) 111(d)
```
An alternative way to draw a prefix tree

**Example**

- **Alphabet:** $A = \{a, b, c, d\}$.
- $C$ defined by: $a \rightarrow 0$, $b \rightarrow 10$, $c \rightarrow 110$, $d \rightarrow 111$. 

```
    0 1
   / \
  0 1
 /   |
/    |
/     |
/ a b c d
```

Marek Rychlik  
CGB
Weighted binary trees

**Definition**

1. A *weighted binary* tree is a pair \((T, w)\) where \(T\) an arbitrary binary tree the number \(w : \text{nodes}(T) \rightarrow \mathbb{R}^+\) is a *weight function*, assigning weight \(w(n)\) to every node of \(T\).

2. The *total weight operator* of the tree is defined as

\[
W_T(w) = \sum_{n \in \text{nodes}(T)} w(n).
\]

The total weight may be finite or infinite.

3. A *depth-weighted binary* tree is an arbitrary binary tree \(T\) with the weight of every leaf equal to \(2^{-\text{depth}(l)}\). All inner nodes are assigned weight of 0.
An example of a depth-weighted tree

Example

- Alphabet: $A = \{a, b, c, d\}$.
- $C$ defined by: $a \rightarrow 0$, $b \rightarrow 10$, $c \rightarrow 110$, $d \rightarrow 111$. 

```
   0   1
  / \  /
a(1/2)  b(1/4)
  /   \
0   1
  / \
0   1
  / \
 c(1/8) d(1/8)
```
Kraft Inequality — Proof

We define a sequence of weight functions
\( w_k : \text{nodes}(T) \rightarrow \mathbb{R}^+, \quad k = 0, 1, \ldots \), by induction:

1. Weight function \( w_0 \) assigns weight 1 to the root and weight 0 to all other nodes.
2. If weight function \( w_k \) is defined then \( w_{k+1} \) is obtained by dividing the weight \( w_k \) of nodes at depth \( k \) amongst the children at depth \( k + 1 \).
More precisely,

\[ w_{k+1}(n) = \begin{cases} 
\frac{w_k(parent(n))}{|children(parent(n))|} & \text{if } depth(n) = k + 1, \\
0 & \text{if } depth(n) = k \text{ and } n \text{ is not a leaf}, \\
w_k(n) & \text{otherwise.}
\end{cases} \]

where \(|A|\) stands for cardinality of a set \(A\).

By induction, it follows that \(w_k(n) \geq 2^{-depth(n)}\) if node \(n\) satisfies at least one of the following conditions:

1. \(n \in leaves(T)\) and \(depth(n) \leq k\).
2. \(depth(n) = k\).
An example of a depth-weighted tree

Example

- Alphabet: $A = \{a, b, c, d\}$.
- $C$ defined by: $a \rightarrow 0$, $b \rightarrow 10$, $c \rightarrow 110$, $d \rightarrow 111$. 

![Example Diagram](image-url)
An example of a depth-weighted tree

Example

- Alphabet: \( A = \{a, b, c, d\} \).
- \( C \) defined by: \( a \rightarrow 0, \ b \rightarrow 10, \ c \rightarrow 110, \ d \rightarrow 111 \).
An example of a depth-weighted tree

Example

- Alphabet: $A = \{a, b, c, d\}$.
- $C$ defined by: $a \rightarrow 0$, $b \rightarrow 10$, $c \rightarrow 110$, $d \rightarrow 111$. 

![Diagram](attachment:image.png)
An example of a depth-weighted tree

**Example**

- **Alphabet:** $A = \{a, b, c, d\}$.
- $\mathcal{C}$ defined by: $a \rightarrow 0$, $b \rightarrow 10$, $c \rightarrow 110$, $d \rightarrow 111$. 

```
    0
   / \   k = 3
  0   1
 / \   
0 1
```

```
   a(1/2) 0
   /       
  0 1
   / 
0 1
```

```
   b(1/4) 0
   /       
  0 1
   / 
0 1
```

```
   c(1/8) d(1/8)
   /       
  0 1
   / 
0 1
```
Wikipedia’s version of Fatou’s lemma

**Theorem**

If $f_1, f_2, \ldots$ is a sequence of **non-negative** measurable functions defined on a measure space $(S, \Sigma, \mu)$, then

$$\int_S \liminf_{n \to \infty} f_n \, d\mu \leq \liminf_{n \to \infty} \int_S f_n \, d\mu . \quad (1)$$

- On the left-hand side the limit inferior of the $f_n$ is taken pointwise.
- The functions are allowed to attain the value infinity and the integrals may also be infinite.
Fatou’s lemma for series

**Corollary**

If $f_1, f_2, \ldots$ is a sequence of non-negative measurable functions defined on a countable set $S$, then

$$\sum_{s \in S} \liminf_{n \to \infty} f_n(s) \leq \liminf_{n \to \infty} \sum_{s \in S} f_n(s).$$  \hspace{1cm} (2)
Kraft Inequality — Proof (conclusion)

- The limit \( w_\infty(n) = \lim_{k \to \infty} w_k(n) \) exists and it is greater or equal \( 2^{-\text{depth}(n)} \) for \( n \in \text{leaves}(T) \) and 0 otherwise.
- By Fatou’s Lemma:

\[
1 = \lim \inf_{k \to \infty} \sum_{n \in \text{nodes}(T)} w_k(n) \geq \sum_{n \in \text{nodes}(T)} \lim \inf_{k \to \infty} w_k(n) = \sum_{n \in \text{nodes}(T)} w_\infty(n) = \sum_{n \in \text{leaves}(T)} w_\infty(n) \geq \sum_{n \in \text{leaves}(T)} 2^{-\text{depth}(n)}.
\]
Complete binary trees and codes

**Definition**
A binary tree is called a *complete binary tree* if every node is either a leaf or it has exactly two children.

**Definition**
A binary code $C$ is called a *complete code* if its prefix tree is a complete binary tree.
Equality in Kraft Inequality

**Corollary**

*If $A$ is countable and $C : A \rightarrow \{0, 1\}^+$ is a lossless symbol code then*

$$\sum_{a \in A} 2^{-D(a)} = 1.$$  

*iff the prefix tree of $C$ is a complete binary tree.*
Shannon source coding theorem

**Theorem**

*(Shannon, 1948) If a binary symbol code \( C : A \to \{0, 1\}^+ \) is lossless then*

\[
\mathbb{E}(\ell \circ C) \geq H(P)
\]

*where \( H(P) \) is the Shannon entropy of the distribution \( P \):*

\[
H(P) = \sum_{a \in A} P(a)(-\log_2 P(a))
\]

*The quantity \( I(a) = -\log_2 P(a) \) is interpreted as the amount of information contained in one occurrence of symbol \( a \).*
Proof

- Let $T$ be the prefix tree of the code $C$.
- Let us define the *probability weight* of the node $n$: 
  $P(n) = P(a)$ iff $n$ is the node corresponding to $C(a)$.
- Clearly, if $D(n) = \text{depth}_T(n)$ then

\[
\mathbb{E}(\ell \circ C) = \sum_{n \in \text{leaves}(T)} D(n)P(n)
\]
Outline of the proof - continued

Observe that the inequality:

$$\sum_{n \in \text{leaves}(T)} D(n) P(a) \geq \sum_{n \in \text{leaves}(T)} P(n)(-\log_2 P(n))$$

is equivalent to

$$\sum_{n \in \text{leaves}} P(n) \log_2 \frac{1}{2^{D(n)} P(n)} \leq 0$$
Strictly concave functions

- A function $f : (a, b) \rightarrow \mathbb{R}$ is strictly convex if for every $x, y \in (a, b)$ and $t \in (0, 1)$:

  $$f(tx + (1 - t)y) > tf(x) + (1 - t)f(y).$$

- If $t_1, t_2, \ldots, t_k$ is a sequence such that $t_j \geq 0$ and $\sum_{j=1}^{k} t_k = 1$ then

  $$f \left( \sum_{j=1}^{k} t_k x_k \right) \geq \sum_{j=1}^{k} t_j f(x_k).$$

- The inequality is strict unless $f(x_j)$ are all identical for all $j$ such that $t_j \neq 0$. 
Outline of the proof - continued

- Use strict concavity of $\log_2$ to show:

$$\sum_{n \in \text{leaves}(T)} P(n) \log_2 \frac{1}{2^{D(n)} P(n)} \leq \log_2 \left( \sum_{n \in \text{leaves}(T)} P(n) \frac{1}{2^{D(n)} P(n)} \right)$$

$$= \log_2 \left( \sum_{n \in \text{leaves}(T)} 2^{-D(n)} \right) = 0.$$
Equality in the fundamental theorem

**Corollary**

If $\mathbb{E}(\ell \circ C) = H(P)$ then for all $a \in A$ $P(a) = 2^{-D(a)}$ where $D(a)$ is a certain integer.
Optimal coding when probabilities are powers of 2

Problem

If all probabilities $P(a)$ are powers of 2 then there exists an lossless binary code $C : A \rightarrow \{0, 1\}^+$ such that

$$E(\ell \circ C) = H(P).$$
The existence of nearly optimal codes

**Theorem**

(Shannon-Fano, 1948) For every alphabet $A$ and a distribution function $P : A \rightarrow (0, 1]$ there exists a binary code $C : A \rightarrow \{0, 1\}^+$ such that:

$$H(P) \leq \mathbb{E}(\ell \circ C) \leq H(P) + 1.$$