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: $P : A \to (0, 1]$, so that
$$\sum_{a \in A} P(a) = 1.$$
- Random message: a sequence $M = s_1 s_2 \ldots, s_L$ 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. $A^+ = \bigcup_{L=0}^{\infty} A^L$.

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

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

bits.

Coding and lossless coding

Definition

1. A *code* is a function
$$\mathcal{C} : A^+ \longrightarrow B^+$$
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 $\mathcal{C} : A^+ \longrightarrow B^+$ 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 $\mathcal{C}(A^+)$, i.e. may perform a mapping

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

so that $S \supseteq \mathcal{C}(A)$ and

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

Symbol codes

Definition

- A *symbol code* is a mapping

$$\mathcal{C} : A \to B^+$$

of the alphabet to messages in another alphabet.
- The *extension* of the symbol code $\mathcal{C}$ is a code obtained by concatenation:

$$s_1 s_2 \ldots s_L \to \mathcal{C}(s_1)\mathcal{C}(s_2)\ldots\mathcal{C}(s_L).$$

- A *symbol code* is *lossless* iff $\mathcal{C}$ 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:

$$a \mapsto \ell(\mathcal{C}(a)).$$

- The code is *uniform* if the lengths of $\mathcal{C}(s)$ are identical for all $s \in A$, i.e. $\ell \circ \mathcal{C}$ 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:

$$a \mapsto \ell(\mathcal{C}(a)).$$

- The code is *uniform* if the lengths of $\mathcal{C}(s)$ are identical for all $s \in A$, i.e. $\ell \circ \mathcal{C}$ 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:

$$a \mapsto \ell(\mathcal{C}(a)).$$

- The code is *uniform* if the lengths of $\mathcal{C}(s)$ are identical for all $s \in A$, i.e. $\ell \circ \mathcal{C}$ 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:

$$\begin{aligned} a &\longrightarrow 00 \\ b &\longrightarrow 10 \\ c &\longrightarrow 01 \end{aligned}$$

Thus,

$$abcba \longrightarrow 0010011000$$

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:

$$a \to 0,\ b \to 10,\ c \to 110,\ d \to 111.$$

Notes on the compression example

- We note that $\ell(\mathcal{C}(s)) = -\log_2 P(s)$ 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 $1.75L$.

The prefix property

Definition

We say that a symbol code $\mathcal{C} : A \to B+$ 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:

$$\begin{array}{ccc} a & \longrightarrow & 0 \\ b & \longrightarrow & 10 \\ c & \longrightarrow & 110 \\ d & \longrightarrow & 111 \end{array}$$

- Code: 0101101110
- Decoded message:

An decoding example

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

$$\begin{aligned} a &\longrightarrow 0 \\ b &\longrightarrow 10 \\ c &\longrightarrow 110 \\ d &\longrightarrow 111 \end{aligned}$$

- Code: 0101101110
- Decoded message:

An decoding example

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

$$\begin{aligned} a &\longrightarrow 0 \\ b &\longrightarrow 10 \\ c &\longrightarrow 110 \\ d &\longrightarrow 111 \end{aligned}$$

- Code: 0101101110

$$\ldots$$

- Decoded message:

$$\ldots$$

An decoding example

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

$$\begin{aligned} a &\longrightarrow 0 \\ b &\longrightarrow 10 \\ c &\longrightarrow 110 \\ d &\longrightarrow 111 \end{aligned}$$

- Code: 0101101110

$$\textcolor{green}{0}\ldots$$

- Decoded message:

$$a\ldots$$

An decoding example

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

$$\begin{array}{ccc} a & \longrightarrow & 0 \\ b & \longrightarrow & 10 \\ c & \longrightarrow & 110 \\ d & \longrightarrow & 111 \end{array}$$

- Code: 0101101110

$$01\ldots$$

- Decoded message:

$$a?\ldots$$

An decoding example

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

$$\begin{array}{ccc} a & \longrightarrow & 0 \\ b & \longrightarrow & 10 \\ c & \longrightarrow & 110 \\ d & \longrightarrow & 111 \end{array}$$

- Code: 0101101110

$$010\ldots$$

- Decoded message:

$$ab\ldots$$

An decoding example

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

$$\begin{array}{ccc} a & \longrightarrow & 0 \\ b & \longrightarrow & 10 \\ c & \longrightarrow & 110 \\ d & \longrightarrow & 111 \end{array}$$

- Code: 0101101110

$$0101\ldots$$

- Decoded message:

$$ab?\ldots$$

An decoding example

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

$$\begin{array}{ccc} a & \longrightarrow & 0 \\ b & \longrightarrow & 10 \\ c & \longrightarrow & 110 \\ d & \longrightarrow & 111 \end{array}$$

- Code: 0101101110

$$01011\ldots$$

- Decoded message:

$$ab?\ldots$$

An decoding example

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

$$\begin{array}{ccc} a & \longrightarrow & 0 \\ b & \longrightarrow & 10 \\ c & \longrightarrow & 110 \\ d & \longrightarrow & 111 \end{array}$$

- Code: 0101101110

$$010110\ldots$$

- Decoded message:

$$abc\ldots$$

An decoding example

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

$$\begin{array}{ccc} a & \longrightarrow & 0 \\ b & \longrightarrow & 10 \\ c & \longrightarrow & 110 \\ d & \longrightarrow & 111 \end{array}$$

- Code: 0101101110

$$0101101\ldots$$

- Decoded message:

$$abc?\ldots$$

An decoding example

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

$$\begin{array}{ccl} a & \longrightarrow & 0 \\ b & \longrightarrow & 10 \\ c & \longrightarrow & 110 \\ d & \longrightarrow & 111 \end{array}$$

- Code: 0101101110

$$01011011\ldots$$

- Decoded message:

$$abc?\ldots$$

An decoding example

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

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

- Code: 0101101110

$$010110111\ldots$$

- Decoded message:

$$abcd\ldots$$

An decoding example

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

$$\begin{aligned} a &\longrightarrow 0 \\ b &\longrightarrow 10 \\ c &\longrightarrow 110 \\ d &\longrightarrow 111 \end{aligned}$$

- Code: 0101101110

0101101110

- Decoded message:

$$abcda$$

The expected length of the code

Definition

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

$$\mathbb{E}(\ell \circ \mathcal{C}) = \sum_{a \in A} \ell(\mathcal{C}(a)) \cdot P(a).$$

Kraft inequality (extended variant)

Theorem

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

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

where $D(a) = \ell(\mathcal{C}(a))$.

The prefix tree of a code

Definition

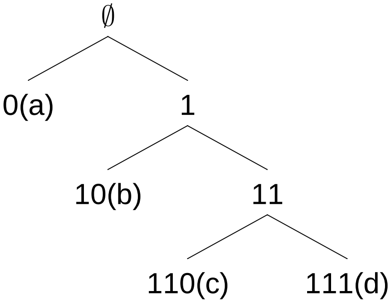
The *prefix tree* $T(\mathcal{C})$ of the code $\mathcal{C}$ is defined as follows:

- The nodes of the tree $T(\mathcal{C})$ are in 1:1 correspondence to all prefixes of all codes $\{\mathcal{C}(a)\}_{a\in A}$.
- The parent of a node of a prefix $b_0 b_1 \ldots b_{d-1} b_d$ of length d, $(b_i \in \{0,1\}$, $d \geq 1)$ is the node corresponding to the prefix of length $d-1$, i.e. $b_0 b_1 \ldots b_{d-1}$.
- The full codes $\mathcal{C}(a)$ are there 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

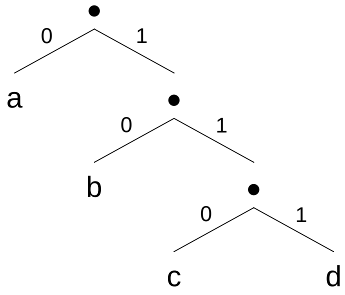
- Alphabet: $A = \{a, b, c, d\}$.
- $\mathcal{C}$ defined by: $a \to 0$, $b \to 10$, $c \to 110$, $d \to 111$.



An alternative way to draw a prefix tree

Example

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



Weighted binary trees

Definition

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

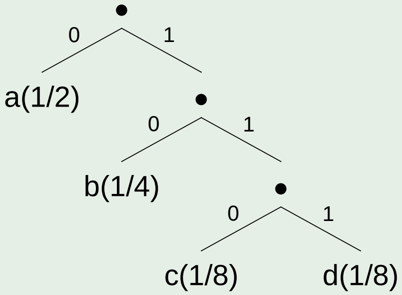
$$W_T(w) = \sum_{n \in nodes(T)} w(n).$$

   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 $2^{-depth(l)}$. All inner nodes are assigned weight of 0.

An example of a depth-weighted tree

Example

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



Kraft Inequality — Proof

- We define a sequence of weight functions $w_k : nodes(T) \to \mathbb{R}^+$, $k = 0, 1, \ldots$, 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.

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

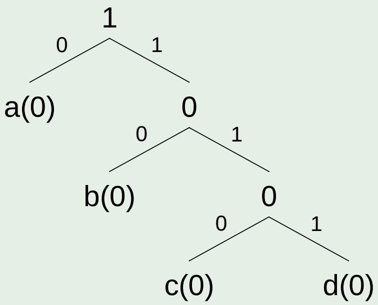
where $|A|$ stands for cardinality of a set A.

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

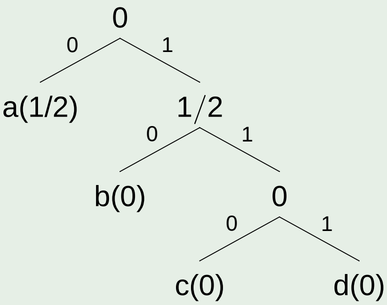


$k = 0$

An example of a depth-weighted tree

Example

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

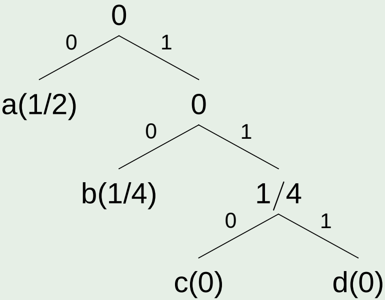


$k = 1$

An example of a depth-weighted tree

Example

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

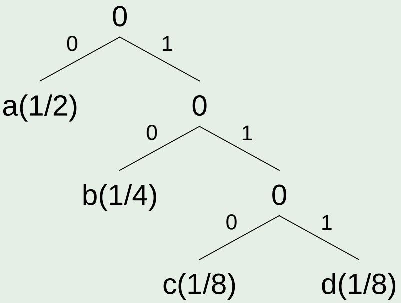


$k = 2$

An example of a depth-weighted tree

Example

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



$k = 3$

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, Σ, μ), then

$$\int_S \liminf_{n\to\infty} f_n \, d\mu \le \liminf_{n\to\infty} \int_S f_n \, d\mu \,. \tag{1}$$

- 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

$$\sum_{s\in S} \liminf_{n\to\infty} f_n(s) \le \liminf_{n\to\infty} \sum_{s\in S} f_n(s). \tag{2}$$

Kraft Inequality — Proof (conclusion)

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

$$
\begin{aligned}
1 = \liminf_{k\to\infty} \sum_{n\in nodes(T)} w_k(n) &\geq \sum_{n\in nodes(T)} \liminf_{k\to\infty} w_k(n) \\
&= \sum_{n\in nodes(T)} w_\infty(n) \\
&= \sum_{n\in leaves(T)} w_\infty(n) \\
&\geq \sum_{n\in leaves(T)} 2^{-depth(n)}.
\end{aligned}
$$

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 $\mathcal{C}$ is called a *complete code* if its prefix tree is a complete binary tree

Equality in Kraft Inequality

Corollary

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

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

iff the prefix tree of $\mathcal{C}$ is a complete binary tree.

Shannon source coding theorem

Theorem

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

$$\mathbb{E}(\ell \circ \mathcal{C}) \geq H(P)$$

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 $\mathcal{C}$.
- Let us define the *probability weight* of the node n: $P(n) = P(a)$ iff n is the node corresponding to $\mathcal{C}(a)$.
- Clearly, if $D(n) = depth_T(n)$ then

$$\mathbb{E}(\ell \circ \mathcal{C}) = \sum_{n \in leaves(T)} D(n)P(n)$$

Outline of the proof - continued

- Observe that the inequality:

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

is equivalent to

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

Strictly concave functions

- A function $f : (a, b) \to \mathbb{R}$ 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, ..., 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:

$$\begin{aligned}\sum_{n\in leaves(T)} P(n)\log_2 \frac{1}{2^{D(n)}P(n)} &\leq \log_2\left(\sum_{n\in leaves(T)} P(n)\frac{1}{2^{D(n)}P(n)}\right)\\ &= \log_2\left(\sum_{n\in leaves(T)} 2^{-D(n)}\right) = 0.\end{aligned}$$

Equality in the fundamental theorem

Corollary

If $\mathbb{E}(\ell \circ \mathcal{C}) = H(P)$ *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 $\mathcal{C} : A \to \{0,1\}^+$ such that

$$\mathbb{E}(\ell \circ \mathcal{C}) = H(P).$$

The existence of nearly optimal codes

Theorem

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

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