Information Theory and Networks

Lecture 8: Decodability

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/Lecture_notes/InformationTheory/

School of Mathematical Sciences,
University of Adelaide

September 18, 2013

Part I

Decodability

There are 10 types of people in the world: those who understand binary, and those who don't

Morse code

Morse code has a problem

- its not really a binary code because we need letter and word separators
  - e.g., to tell the difference between

$$
\begin{aligned}
an &= \cdot - \;\; - \cdot \\
p &= \cdot - - \cdot
\end{aligned}
$$

- we end up with 4 "symbols", and that
  - complicates the transmission and reception processes
  - reduces the efficiency
  - introduces a source of errors
- In general we want codes that are decodable without adding extra symbols
  - e.g., true binary codes

Definitions [CT91, pp.78-81]

Definition (Source code)

A source code C for a random variable X is a mapping from Ω, the range of X to $\mathcal{D}^*$ (the set of all finite length strings of symbols from the alphabet $\mathcal{D}$).

Our code "alphabet" is made up of symbols from $\mathcal{D}$. If the size of this set is $D = |\mathcal{D}|$ then we call this a D-ary code.

If we only allowed single symbols in the output, then this would be the range of $C(\cdot)$, but usually we allow finite strings in our "codewords".

The set of strings of length n is called $\mathcal{D}^n$, and the set of all finite length strings is called $\mathcal{D}^* = \cup \mathcal{D}^n$, So the source code is a mapping $C : \Omega \to \mathcal{D}^*$, which might, for instance, look like

$$C(x) = d_1 d_2 d_3 \ldots d_n$$

for some $d_i \in \mathcal{D}$. The length of the code is denoted $\ell(x)$, which in the case above would be n.

Definitions [CT91, pp.78-81]

Definition (Non-singular)

A code is said to be non-singular if every element of the range of X maps into a different string in $\mathcal{D}^*$, i.e.,

$$x_i \neq x_j \quad \Rightarrow C(x_i) \neq C(x_j)$$

Non-singularity is a necessary condition for decodability

- otherwise we can't decode a single symbol uniquely

but it isn't sufficient to guarantee decodability of a sequence, at least not without an extra "separator" symbol, which is inefficient.

Definitions [CT91, pp.78-81]

Definition (Extension)

The extension C^* of a code C is the mapping from finite length strings Ω^* to finite length strings $\mathcal{D}^*$ defined by

$$C^*(x_1 x_2 \cdots x_n) = C(x_1)C(x_2)\cdots C(x_n)$$

where $C(x_i)C(x_j)$ indicates concatenation of codewords.

Definition (Uniquely decodable)

A code is called uniquely decodable if its extension is non-singular.

Definitions [CT91, pp.78-81]

Definition (Prefix-free codes)

A code is called a prefix-free code or an instantaneous code if no codeword is a prefix of any other codeword.

For Example:

X	Prefix-free code
1	0
2	10
3	110
4	111

Prefix-free codes

Theorem

Prefix-free codes are uniquely decodable (and in fact can be decoded without reference to the future codewords).

Proof.

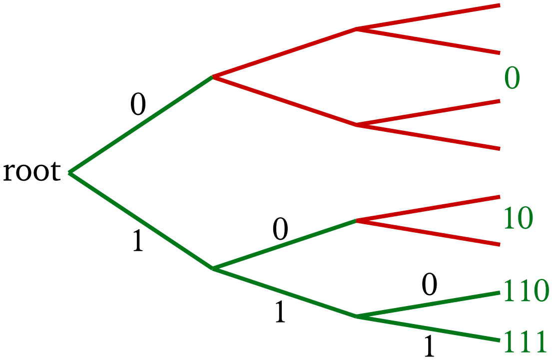
In a prefix-free code, the end of a codeword is immediately recognisable because if we find a string that is a valid codeword, it can't be the prefix of a longer codeword, so we can stop decoding the word at that point. $\square$

We can think of prefix-codes as self-punctuating.

The result above means that prefix-free codes are not just uniquely decodable, but also that we can decode using a single-pass, making them an attractive option for codes.

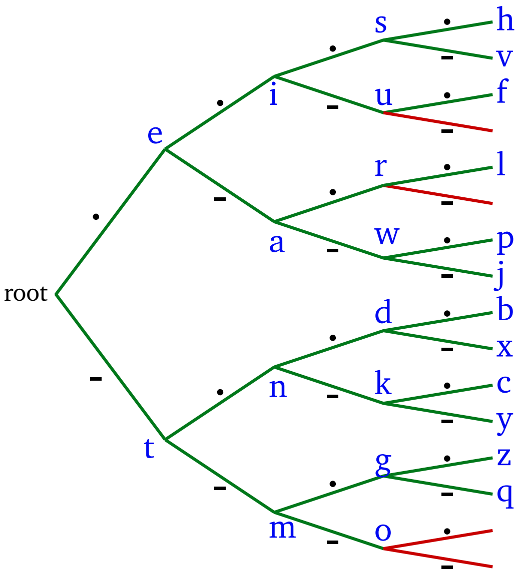
Prefix-free codes

We can represent codewords as a D-ary tree: e.g., for binary codes

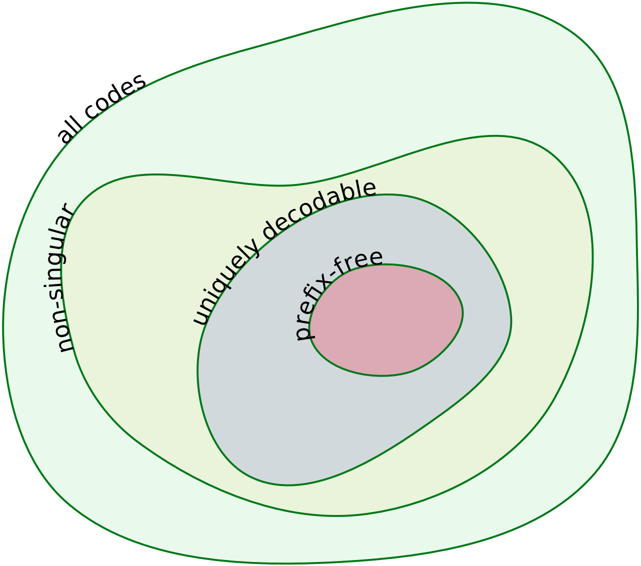


For a prefix-free code, no codeword can be an ancestor of another.

Morse code is not prefix-free



Code classes



Code classes [CT91, pp.82]

X	Singular code	Non-singular, but not uniquely decodable	Uniquely decodable, but not prefix-free	Prefix-free
1	0	0	10	0
2	0	010	00	10
3	0	01	11	110
4	0	10	110	111

Variable vs fixed length codes

- If we fix the length of the codewords, then, its easy to determine the boundaries
  - such codes are implicitly prefix free (as long as they are non-singular)
- But variable length codes can be more efficient
  - e.g., use shorter codes for more common symbols
  - now we have to make sure they are uniquely decodable and the easiest thing is to ensure they are prefix free

Kraft inequality

Theorem (Kraft inequality)

There exists a D-ary prefix-free code with codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$, iff the Kraft inequality

$$\sum_{k=1}^{m} D^{-\ell_k} \leq 1,$$

is satisfied.

Kraft inequality example

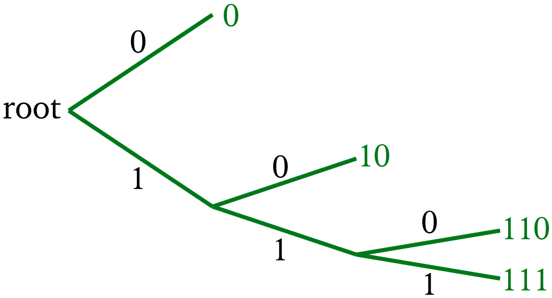
X	Prefix-free code	length ℓ_i
1	0	1
2	10	2
3	110	3
4	111	3

its a binary code, so $D = 2$, so

$$\sum_{k=1}^{m} D^{-\ell_k} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1.$$

Prefix-free codes

We can represent codewords as a D-ary tree: e.g., for binary codes



For a prefix-free code, no codeword can be an ancestor of another.

Kraft proof

Kraft inequality $\Rightarrow$.

Consider the D-ary tree corresponding to a prefix-free code. Let $\ell_{\max}$ be the longest codeword. The tree has $D^{\ell_{\max}}$ possible nodes at level $\ell_{\max}$ (but not all are actual codewords).

The kth codeword is at level ℓ_k, and has $D^{\ell_{\max}-\ell_k}$ descendents at level $\ell_{\max}$, and each of these sets of descendents is disjoint, and so the total number of such descendents can't be greater than the possible nodes at level $\ell_{\max}$, i.e.,

$$\sum_{k=1}^{m} D^{\ell_{\max}-\ell_k} \leq D^{\ell_{\max}}$$

and (dividing by $D^{\ell_{\max}}$) the Kraft inequality must hold for any prefix-free code.

$\square$

Kraft proof

Kraft inequality $\Leftarrow$.

Conversely, given a set of codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ which satisfy the inequality, we can always construct a D-ary tree corresponding to a prefix-free code. The construction is as follows:

- WLOG order the lengths so that $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_m$
- There are D^{ℓ_1} possible nodes at depth ℓ_1 suitable for the first code.
- Assume the first i codewords have been chosen successfully, and we now want to choose a codeword of length ℓ_{i+1}. It can't be a descendent of any of the previous codewords, so we have eliminated

  $$\sum_{k=1}^{i} D^{\ell_{i+1}-\ell_k},$$

  nodes at level ℓ_{i+1} of the tree, but by the Kraft inequality we know that this must leave at least one possible choice.

$\square$

Further reading I

Thomas M. Cover and Joy A. Thomas, *Elements of information theory*, John Wiley and Sons, 1991.

Raymond W. Yeung, *Information theory and network coding*, Springer, 2010.