Information Theory with Applications, Math6397
Lecture Notes from September 23, 2014

taken by Dax Mahoney

Since the AEP holds, we have block coding for the stationary ergodic case as for DMS.

2.3.12 Theorem. *Let $\{X_j\}_{j=-\infty}^{\infty}$ be a stationary ergodic source and $H_\infty = \lim_{n\to\infty} H(X_n|X_1, X_2, X_3, \ldots, X_n)$ then for every $\epsilon > 0$ there is δ, $0 < \delta < \epsilon$ and a sequence of codes. $\{(\mathcal{C}_n, \phi_n)\}_{n=1}^{\infty}$ with coding rate $R = \lim_{n\to\infty} \frac{1}{n} \ln |\mathcal{C}_n| < H_\infty + \delta$ such that for all sufficiently large n, $\mathbb{P}$(decoding error) $< \epsilon$*

Proof. As before, since you only use AEP. □

We also have the converse of block coding:

2.3.13 Theorem. *Let $\{(\mathcal{C}_n, \phi_n)\}_{n=1}^{\infty}$ be a sequence of block codes with $R = \limsup_n \frac{1}{n} \ln |\mathcal{C}_n| <$ then for all $\lambda > 0$ and any choice $\{\psi_n\}_{n=1}^{\infty}$ if n is sufficiently large,*
we have P(decoding error) $> 1 - \lambda$.

Proof. As before. □

2.3.14 Question. There are instances in which we would like to have no mistakes, a loss-less situation. How can we get codes such that

$$\mathbb{P}(\text{decoding error}) \xrightarrow{n\to\infty} 0 \text{ or even } \mathbb{P}(\text{decoding error}) = 0?$$

2.3.15 Answer. Allow an infinite size code book

2.4 Separable Codes and Prefix Codes

Block coding in reality uses a sequence $x \in \mathbb{A}^n$ that is mapped to $\mathbb{Q}_n(x) \in \mathbb{B}^\ell$ for a code alphabet $\mathbb{B}$ with size $|\mathbb{B}| = \mathsf{K}$. You can think of K as the base of a number system, which motivates calling this K-**ary encoding**.

When the length of the code sequence is no longer fixed, we speak of *fixed-variable* coding, with the codebook a subset of $\bigcup_{\ell=1}^{\infty} \mathbb{B}^\ell$

2.4.16 Definition. A map ϕ with range $\mathcal{C} \subset \bigcup_{\ell=1}^{\infty} \mathbb{B}^\ell$ is called **regular** if it is $1 - 1$.

Usually we need to encode sequences $\{x_1, x_2, \ldots, x_n\}$, with $x_j \in \mathbb{A}$.

2.4.17 Definition. A code $(\mathcal{C}, \phi)$ is called **separable** if we can extend the $1-1$ map ϕ to sequences by concatenation,

$$\phi(\{x_1, x_2, \ldots, x_m\}) = \{\phi(x_1), \phi(x_2), \ldots, \phi(x_m)\}$$

and this concatenation map is invertible for all $m \in \mathbb{N}$.

So if $\mathbb{A} = \{$A,B,C,D,E,F$\}$, $\mathbb{B} = \{0, 1\}$ and we chose block length 1 for $\mathbb{A}$, we could encode in the way described in Table 2 to achieve a separable code.

$x \in \mathbb{A}$	$\phi(x)$
A	0
B	10
C	110
D	1110
E	11110
F	111110

Table 2: Example of a separable binary code for source alphabet $\{A, B, C, D, E, F\}$.

Usually, we need to read the entire message to separate words, however our example shows that there is a method that allows for iterative decoding. These are called **prefix codes**.

2.4.18 Definition. A code $\phi : \mathbb{A} \to \mathcal{C} = \bigcup_{\ell=1}^{\infty} \mathbb{B}^\ell$ is called a **prefix code** if no code-word is the prefix of another.

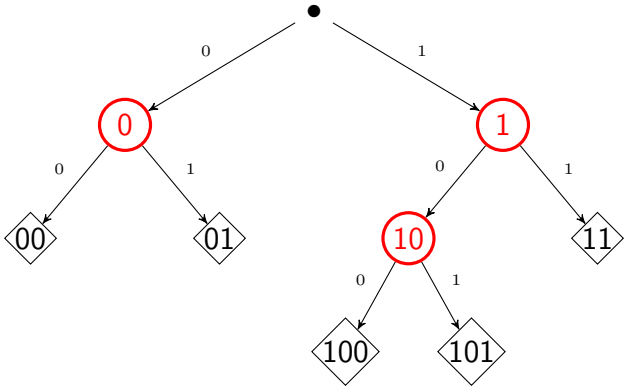
2.4.19 Example. Prefixes of $\{b_1, b_2, \ldots, b_\ell\} \in \mathbb{B}^\ell$ are $\{b_1\}, \{b_1 b_2\}, \{b_1 b_2 b_3\} \ldots, \{b_1 b_2, \ldots, b_\ell\}$

2.4.20 Example. Examples of fixed-variable codes

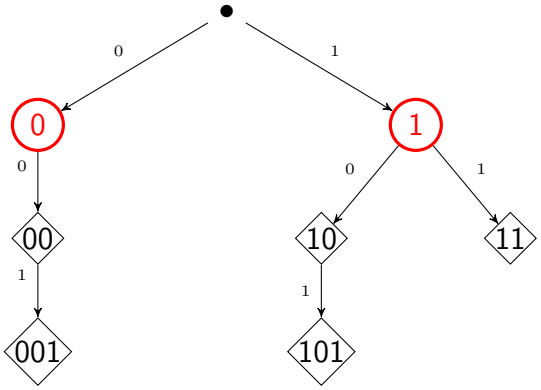
- $\mathcal{C}_1 = \{\{0,0\}\{0,1\}\{1,1\}\{1,0,0\}\{1,0,1\}\}$
- $\mathcal{C}_2 = \{\{0,0\}\{1,0\}\{1,1\}\{0,0,1\}\{1,0,1\}\}$
- $\mathcal{C}_3 = \{\{0,0\}\{0,1\}\{1,1\}\{1,0,0\}\{1,1,0\}\}$

2.4.21 Question. Which of these is prefix / separable? Is there a systematic way to find out if something is a prefix code or separable?

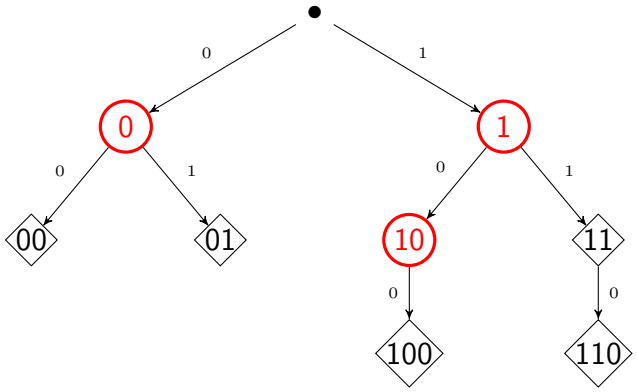
2.4.22 Answer. We can build a tree!



(a) $\mathcal{C}_1 = \{\{0,0\}\{0,1\}\{1,1\}\{1,0,0\}\{1,0,1\}\}$.
$\mathcal{C}_1$ is a valid prefix code.



(b) $\mathcal{C}_2 = \{\{0,0\}\{1,0\}\{1,1\}\{0,0,1\}\{1,0,1\}\}$.
$\mathcal{C}_2$ is a not a prefix code, but it is separable because it is a backward prefix.



(c) $\mathcal{C}_3 = \{\{0,0\}\{0,1\}\{1,1\}\{1,0,0\}\{1,1,0\}\}$.
$\mathcal{C}_3$ we can see is neither prefix or separable with the sequence $\{1,1,0,0,0,1,0,0\}$ could be interpreted as $\{\{1,1,0\},\{0,1\},\{0,0\}\}or\{\{1,1\},\{0,0\},\{1,0,0\}\ \}$

2.4.23 Remark. We note that the K-ary prefix codes are characterized by a K-ary tree with **nodes** (vertices with ≥ 2 adjacent nodes) and **leaves** (vertices with only one edge).

Decoding is a simple iterative procedure. While reading along the coded sequence and following the tree, if we arrive at a leaf (diamond symbol) we record the coded symbol and start again from the root.

2.4.24 Question. How long do codewords have to be?

2.4.25 Theorem. *Given $|\mathbb{B}| = k$, then for any separable code $\phi : \mathbb{A} \to \bigcup_{\ell=1}^{\infty} \mathbb{B}^\ell$,*

$$\sum_{x \in \mathbb{A}} K^{-\ell(x)} \leq 1\,.$$

Here $\ell(x)$ is the length of $\phi(x)$ and the size of $\mathbb{A}$ is implicit as you are summing over all elements.

Proof. Assigning $n\ell_{\mathsf{max}}$ as the maximal length of $\phi(x), x \in \mathbb{A}$, Consider the n-th power of the

left-hand side

$$\begin{aligned}(\sum_{x\in\mathbb{A}} K^{-\ell(x)})^n &= \sum_{x_1\in\mathbb{A}}\sum_{x_2\in\mathbb{A}}\cdots\sum_{x_n\in\mathbb{A}} K^{-\ell(x_1)-\ell(x_2)\cdots-\ell x_n}\\ &= \sum_{(x_1,x_2,\ldots,x_n)\in\mathbb{A}^n} K^{-\ell(x_1,x_2,\ldots,x_n)}\\ &\leq \sum_{m=1}^{n\ell_{\max}} \underbrace{A(m)}_{\substack{\text{number of codewords}\\ \text{of size } A(m)\leq K^m}} K^{-m}\\ &\leq n\ell_{\max}\end{aligned}$$

so taking the n-th root on both sides and n$\to\infty$, we get $\sum_{x\in\mathbb{A}} K^{-\ell(x)} \leq (n\ell_{\max})^{\frac{1}{n}} \xrightarrow{n\to\infty} 1$ □

This is often called the **Kraft inequality** and it tells you how much flexibility you have.

2.4.26 Question. How long is a codeword on average?

2.4.27 Theorem. *Given a DMS with values in $\mathbb{A}$ and induced measure $\mathbb{Q}$ on $\mathbb{A}$ for all $j \in \{1, 2, \dots\}$ if code is separable*

$$\mathbb{E}[\ell(X_j)] \geq \underbrace{H_K(\mathbb{Q})}_{K\text{-ary entropy, with } \log \text{ of base } K \text{ instead of } \ln}.$$

Which of the choices that the Kraft inequality gave you will let you get close to the bound? Huffman coding is the answer.