

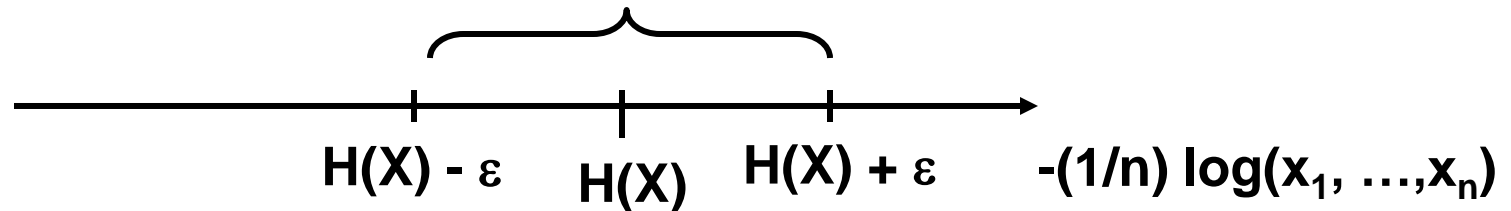
Information Theory

3rd Module

Agenda

- ❖ Entropy Rates of a Stochastic Process (Chapter 4)
- ❖ Compression (Chapter 5)

A sequence in the Typical Set $A_\varepsilon^{(n)}$



- ❖ For any sequence $(x_1, \dots, x_n) \in A_\varepsilon^{(n)} := \{(x_1, \dots, x_n) : |-(1/n) \log_2[p(x_1, \dots, x_n)] - H(X)| \leq \varepsilon\}$, the prob. of the sequence must have the following property
- ❖ $|-(1/n) \log_2[p(x_1, \dots, x_n)] - H(X)| \leq \varepsilon$
- ❖ $H(X) - \varepsilon \leq -(1/n) \log_2[p(x_1, \dots, x_n)] \leq H(X) + \varepsilon$
- ❖ $2^{-n(H(X)+\varepsilon)} \leq p(x_1, \dots, x_n) \leq 2^{-n(H(X) - \varepsilon)}$
- ❖ Since we can choose a very small ε , the prob. of a sequence can be made very close to $2^{-nH(X)}$, as $n \rightarrow \infty$.

Basically, AEP tells us that

- ❖ *For length n i.i.d. sequence of r.v.'s*
- ❖ nH bits are good enough for describing the typical sequence.
 - Size of the typical message set is $2^{nH(X)}$.
 - Each sequence in the message set occurs with $2^{-nH(X)}$.
- ❖ In an experiment, usually a sequence in the typical set happens.
- ❖ Shannon's Theorem 3 is AEP (see [page 13](#)).
- ❖ But what happens if the r.v.'s are dependent?
 - Motivation to consider the *entropy rate*

Shannon's Paper

- ❖ Shannon uses Markov chain to describe English.
 - [Shannon's 1948 paper](#)
 - Zero-order, first-order, second-order letters
 - First-order word, second-order word
 - Let's take a look at his paper.

- ❖ “Can we define a quantity which will measure, in some sense, how much information is “produced” by such a process, or better, at what rate information is produced?”

3. THE SERIES OF APPROXIMATIONS TO ENGLISH

To give a visual idea of how this series of processes approaches a language, typical sequences in the approximations to English have been constructed and are given below. In all cases we have assumed a 27-symbol "alphabet," the 26 letters and a space.

1. Zero-order approximation (symbols independent and equiprobable).

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD.

2. First-order approximation (symbols independent but with frequencies of English text).

OCRO HLI RGWR NMIELWIS EU LL NBNESBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL.

3. Second-order approximation (digram structure as in English).

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TU-COOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE.

4. Third-order approximation (trigram structure as in English).

IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE.

5. First-order word approximation. Rather than continue with tetragram, . . . , n -gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

6. Second-order word approximation. The word transition probabilities are correct but no further structure is included.

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

Some Definitions for MC

❖ Stationary stochastic process:

$$\Pr(\mathbf{X}_1 = \mathbf{x}_1, \dots, \mathbf{X}_n = \mathbf{x}_n) = \Pr(\mathbf{X}_{t+1} = \mathbf{x}_1, \dots, \mathbf{X}_{t+n} = \mathbf{x}_n) \text{ for all } t.$$

❖ MC

$$- \Pr(\mathbf{X}_{n+1} = a \mid \mathbf{X}_n = b, \dots, \mathbf{X}_1 = \mathbf{x}_1) = \Pr(\mathbf{X}_{n+1} = a \mid \mathbf{X}_n = b)$$

❖ MC is *time-invariant* (almost always we assume this) if

$$P(\mathbf{X}_{n+1+t} = a \mid \mathbf{X}_{n+t} = b) = P(\mathbf{X}_{n+1} = a \mid \mathbf{X}_n = b)$$

- Transition matrix \mathbf{P} stays the same.

- *Stationary* distribution: $\mathbf{s} = \mathbf{P}\mathbf{s}$

❖ If the initial distribution is \mathbf{s} , then the MC is stationary.

Per-Symbol Entropy (*Entropy Rate*)

- ❖ Consider a sequence of r.v.s X_1, X_2, \dots, X_n
- ❖ How does the **entropy** of a sequence of n r.v.'s grow with n ?
- ❖ Let's define the *per symbol entropy*
 $H(\mathcal{X}) := \lim_{n \rightarrow \infty} (1/n) H(X_1, \dots, X_n)$ when it exists
- ❖ Examples:
 - ❖ When X_1, X_2, \dots are iid, the rate attains the *maximum* $H(X_1)$.
 $H(\mathcal{X}) = \lim_{n \rightarrow \infty} (1/n) H(X_1, \dots, X_n) = nH(X_1)/n = H(X_1)$.
 - ❖ When X_1, X_2, \dots are indep. but different distr.
Then, $H(X_1, \dots, X_n) = \sum_i H(X_i)$.

Conditional Entropy Rate

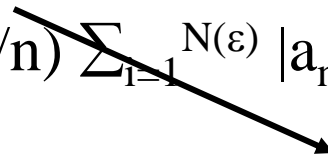
- ❖ $H'(\mathcal{X}) := \lim_{n \rightarrow \infty} H(X_n | X_{n-1}, \dots, X_1)$
- ❖ *For a stationary process, $H'(\mathcal{X}) = H(\mathcal{X})$, both limits exist and equal.*
- ❖ $H(X_{n+1} | X_n, \dots, X_1) \leq H(X_{n+1} | X_n, \dots, X_2)$
 - why?
 - $= H(X_n | X_{n-1}, \dots, X_1)$
 - why?
- ❖ We know that
 - It is a non increasing series of non-negative numbers.
 - It is bounded from below.
- ❖ Then, the limit exists. (convergence from above)

Cesaro Mean (from Analysis):

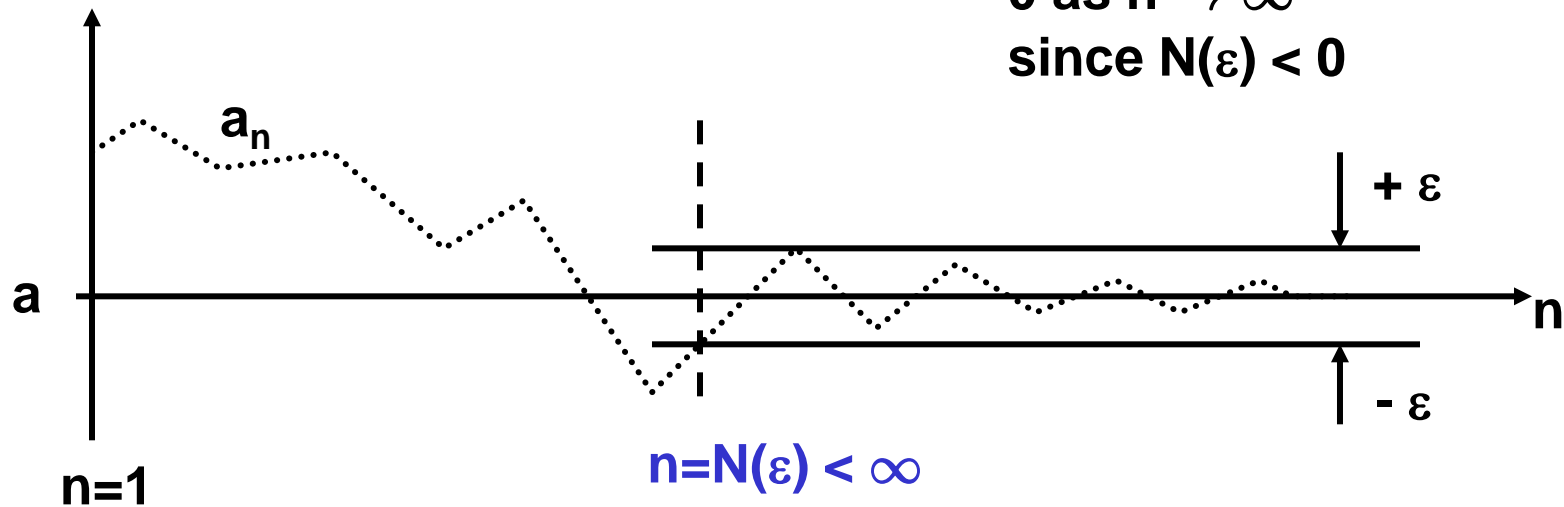
❖ Cesaro Mean (from Analysis):

If $a_n \rightarrow a$ and $b_n = (1/n) \sum_{i=1}^n a_i$, then $b_n \rightarrow a$

$$|b_n - a| \leq (1/n) \sum_{i=1}^n |a_i - a| \leq (1/n) \sum_{i=1}^{N(\varepsilon)} |a_i - a| + \varepsilon$$



**0 as $n \rightarrow \infty$
since $N(\varepsilon) < \infty$**



Cesaro Mean

↵

Statement: If $a_n \rightarrow a$ and $b_n := \frac{1}{n} \sum_{i=1}^n a_i$, then $b_n \rightarrow a$. ↵

Proof. ↵

For any $\varepsilon > 0$, ↵

$$\begin{aligned} |b_n - a| &= \left| \frac{1}{n} \sum_{i=1}^n a_i - a \right| \\ &\leq \frac{1}{n} \sum_{i=1}^n |a_i - a| \\ &= \frac{1}{n} \sum_{i=1}^{N(\varepsilon)} |a_i - a| + \frac{1}{n} \sum_{i=N(\varepsilon)+1}^n |a_i - a| \\ &\leq \frac{1}{n} \sum_{i=1}^{N(\varepsilon)} |a_i - a| + \varepsilon \end{aligned}$$

$H(X) = H'(X)$ for stationary process

❖ From the chain rule:

$$(1/n) H(X_1, \dots, X_n) = (1/n) \sum_{i=1}^n H(X_i | X_{i-1}, \dots, X_1)$$

❖ By applying the Cesaro Mean, we know

$$H(\mathcal{X}) = \lim (1/n) H(X_1, \dots, X_n) = \lim H(X_n | X_{n-1}, \dots, X_1) = H'(\mathcal{X})$$

❖ Implications: For a stationary process,

- There are about $2^{nH(\mathcal{X})}$ typical sequences of length n .
- The prob. of typical set is close to 1.
- $nH(\mathcal{X})$ bits are usually needed to represent the length n typical sequences.

Entropy Rate of a stationary MC

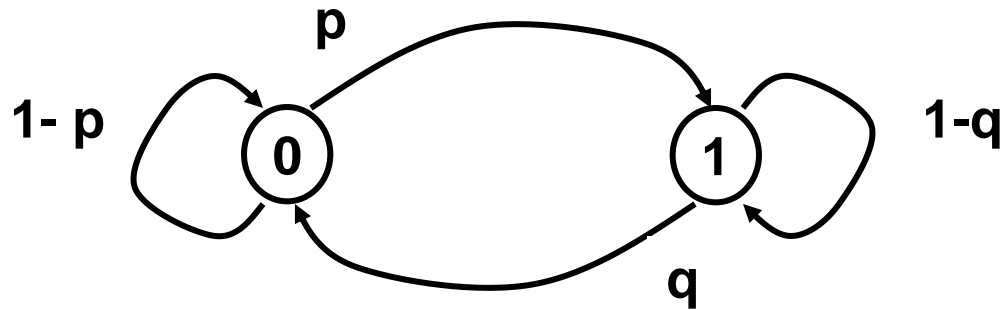
$$\begin{aligned} \blacklozenge H(\mathcal{X}) &= H'(\mathcal{X}) = \lim_{n \rightarrow \infty} H(X_n | X_{n-1}, \dots, X_1) \\ &= \lim_{n \rightarrow \infty} H(X_n | X_{n-1}) \\ &= H(X_2 | X_1) \end{aligned}$$

\blacklozenge Let vector \mathbf{s} denote the stationary distribution and \mathbf{P} the transition matrix of a stationary MC.

$$\mathbf{s} = \mathbf{P}\mathbf{s}$$

$$H(\mathcal{X}) = H(X_2 | X_1) = \sum_i s_i (- \sum_j P_{ij} \log P_{ij})$$

Entropy rate of two state MC



- ❖ $\mathbf{s} = [s_0 \ s_1]'$; ~~$\mathbf{P} = [1-p \ p; 1-q \ q]$~~ ; $\mathbf{P} = [1-p \ q; p \ 1-q]$
- ❖ $H(\mathcal{X}) = H(X_2|X_1) = \sum_i s_i (- \sum_j P_{ij} \log P_{ij})$
- ❖ Then, the entropy rate $H(\mathcal{X}) = (q \cdot H(p) + p \cdot H(q)) / (p+q)$

Shannon's Examples

- ❖ See section 7 of his paper.
- ❖ He was interested in finding the entropy rate of an information source (English).
 - How much redundancy is there in the source?
 - Redundancy in English ≈ 0.5 .

 - This is my example.
 - I _a_t _o _o h_m_ _nd p_ly _it_ m_ k_d_.
 - is it possible to make out the meaning?
 - Deleted about 13 characters ($13/40 = 33\%$)

Answer to my example

- I want to go home and play with my kids. (40 char's and spaces)

❖ Chapter 5: Data Compression

Information generated from a source can be compressed without distortion.

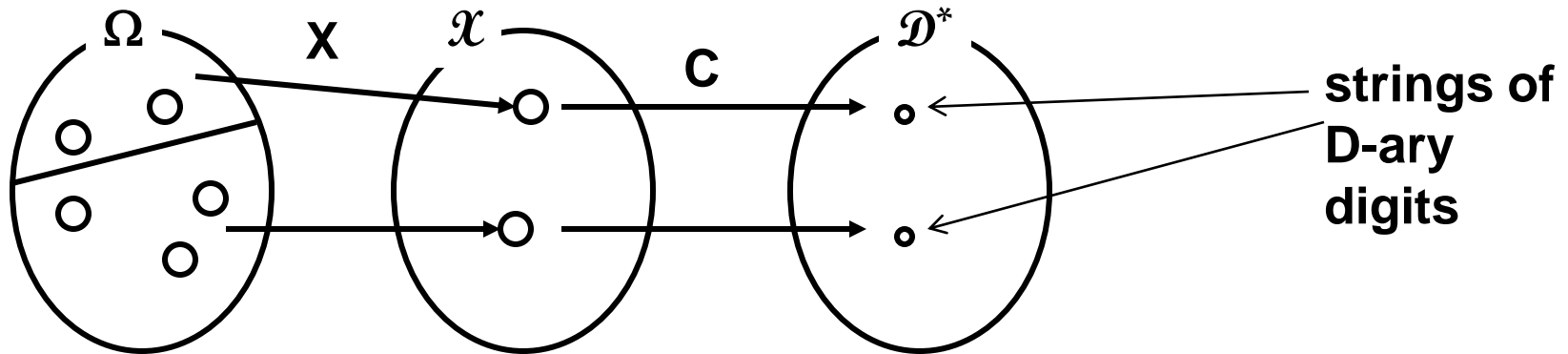
In this chapter, we are interested in distortion-less *Source Compression*.

Data Compression (Distortionless)

- ❖ Intuition tells us that we would want to use
 - Short description for frequent outcomes
 - Long description for less frequent outcomes
- ❖ A code constructed this way will have a small minimum description length.
- ❖ A source code does this efficiently.
- ❖ Information sources include data sources such as type writer, signals and telegraph.
- ❖ Shortest description length of a random source
 - A variable/a process \sim Entropy/Entropy rate
- ❖ Expected description length \geq Entropy

A code is a map

- ❖ A source code C for a random variable X is a *mapping* from $\mathcal{X} = \{X(\omega): x_1, x_2, \dots\}$ to \mathcal{D}^* , the set of codewords
 - A codeword $C(x)$ is a finite length string of D -ary digits assigned to x .



- ❖ Let $l(x)$ be the length of $C(x)$.
- ❖ The expected length $L(C) = E\{l(X)\} = \sum_{x_i} l(x_i) p(x_i)$.

A string of a D -ary digit

- ❖ D -ary alphabet is $\mathcal{D} = \{0, 1, 2, \dots, D-1\}$.
- ❖ Ex) 012201 is a ternary alphabet string.

Non-singular code

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

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

- This statement does not allow many-to-one mapping.
- Mapping—by the definition—is either only one-to-one or many-to-one (no one-to-many).
- Thus, a map (a code) can include either one-to-one or many-to-one assignments.
- Hence, a *non-singular* code is a code that does not allow any many-to-one assignment in a map.
- Non-singularity thus implies the one-to-one mapping which is sufficient for unambiguous decoding.

Some Definitions on Functions

- ❖ A function $f: A \rightarrow B$ is a relation between A and B satisfying the following conditions:
 - For *each* $a \in A$, there *exists* $b \in B$ such that $(a, b) \in f$, AND
 - If (a, b) and (a, c) are in f , then $b = c$.

- ❖ A function $f: A \rightarrow B$ is said to be
 - One-to-one if given $b \in B$, there is at most one $a \in A$.
 - Onto if for each $b \in B$, there is at least one $a \in A$, i.e. $b=f(a)$
 - One-to-one correspondence if a function is both one-to-one and onto.

Extension C^ of C is a map*

❖ An *extension* C^* of a code C is a mapping from finite length strings of \mathcal{X} to finite length strings of \mathcal{D} , defined by

$$C(x_1x_2\dots x_n) = C(x_1)C(x_2) \dots C(x_n)$$

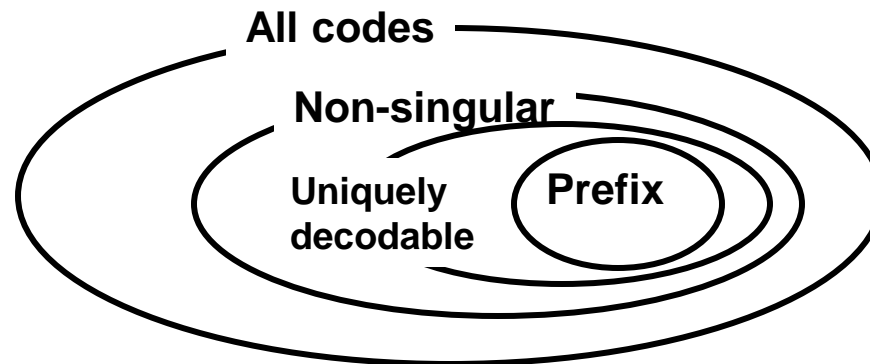
- It is *concatenation* of codewords.
- Ex) If $C(x_1) = 00$ and $C(x_2) = 11$, then $C(x_1x_2) = 0011$
- Why? Efficiency! It can get rid of a space symbol which would have been needed for separating any pair of different length codewords.
- Ex) $C(x_1)=11$, $C(x_2)=10$, $C(x_3) = 110$, $C(x_4) = 01$
 - $110110 \sim 110, 110$ or $11,01,10$ (not decodable)
 - Space symbols useful but wasteful!
 - Not efficient!

Uniquely Decodable Code

- ❖ A code C is called *uniquely decodable* if its k -th *extension* is one-to-one mapping from \mathcal{X}^k to \mathcal{D}^* for all $k \geq 0$.
 - (see P5.21, not P5.18)
 - Uniquely decodable = non singular when extended

Prefix Code

- ❖ A code C is called a *prefix(-free)* code or an *instantaneous* code if no codeword is a prefix of any other codeword.
- ❖ Ex) $C = \{0, 10, 110, 1110\}$
 - $0101100110 \rightarrow 0, 10, 110, 0, 110$ uniquely decodable
 - Instantaneous, since the end of a codeword is immediately recognizable
 - Self-punctuating



Examples

- ❖ $\mathcal{X} = \{1, 2, 3, 4\}$
- ❖ Consider the following codes (the elements are ordered)
 - Non-singular, uniquely decodable, instantaneous?
- ❖ Map#1: $\{0, 0, 1, 1\}$
- ❖ Map#2: $\{0, 010, 01, 10\}$
- ❖ Map#3: $\{10, 00, 11, 110\}$
- ❖ Map#4: $\{0, 10, 110, 111\}$

Kraft Inequality

If A is a prefix code, then the lengths of A 's codeword satisfies Kraft inequality.

- ❖ For any *instantaneous* code over D -ary alphabet, *the collection of codeword lengths, l_1, l_2, \dots, l_m , must satisfy the inequality*

$$\sum_{i=1}^m D^{-l_i} \leq 1$$

where m is the number of codewords.

If the lengths of a code A satisfies Kraft inequality, A is a prefix code.

True?

- ❖ (Converse) Given *a col. of codeword lengths* that satisfy this inequality, there *exists* an instantaneous code with these word lengths.

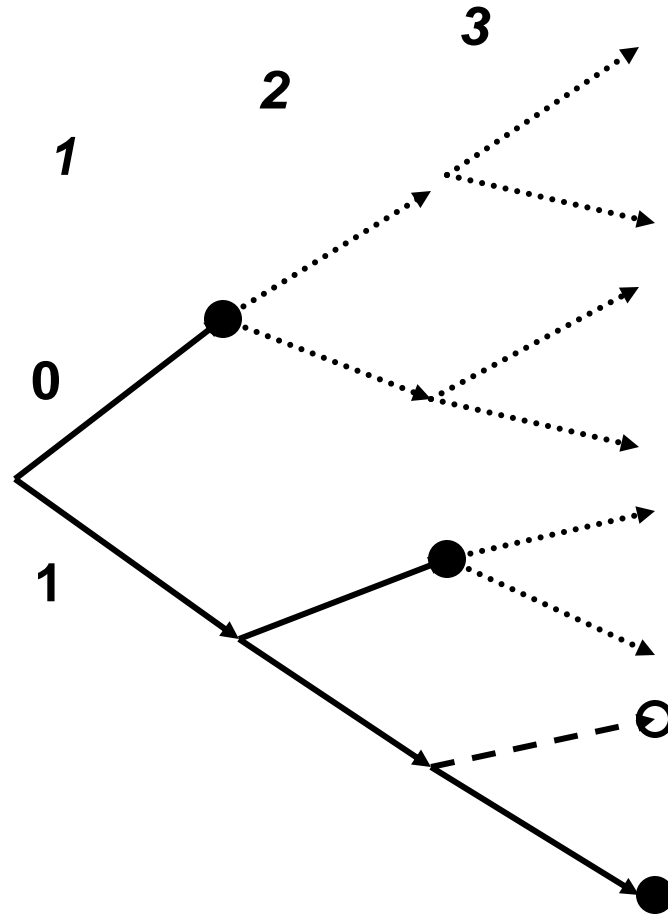
- Sufficiency test for existence of a prefix code.

If the lengths satisfies Kraft inequality, there exists a prefix code with these word lengths.

- ❖ Fundamental constraints on the lengths of a prefix code.

Kraft Inequality (Proof)

- ❖ Prefix code ~ each codeword has no child.
- ❖ l_1, \dots, l_m
- ❖ $l_{\max} = \max\{l_1, \dots, l_m\}$
- ❖ $D^{l_{\max}} \geq \sum_i D^{l_{\max} - l_i}$
 $\Rightarrow 1 \geq \sum_i D^{-l_i} . \text{(QED)}$
- ❖ When do you have equality?
- ❖ Ex) $2^3 \geq 4 + 2 + 1$ or $4+2+1+1$



The previous proof is not rigorous!

- ❖ The proof was based on induction ($D=2, 3, \dots$).
 - Consider the leaves at the maximum depth of the tree.
 - The total number of leaves is greater than or equal to the sum of the leaves displaced by codewords.
 - All descents of a codeword are displaced
 - Any leaf occupied by a codeword is displaced as well.
 - Sum of codewords + descendants of codewords cannot be greater than the total number of leaves at the maximum depth.
- ❖ Any way to improve the proof?

Kraft Inequality for UD code

- ❖ For any *uniquely decodable* code over D -ary alphabet, *the collection of codeword lengths, l_1, l_2, \dots, l_m , must satisfy the inequality*

$$\sum_{i=1}^m D^{-l_i} \leq 1$$

where m is the number of codewords.

- ❖ (Converse) Given *a col. of codeword lengths* that satisfy this inequality, there *exists* a uniquely decodable code with these word lengths.

- Sufficiency test for existence of a uniquely decodable code.

- ❖ Fundamental constraints on the lengths of a UD code.

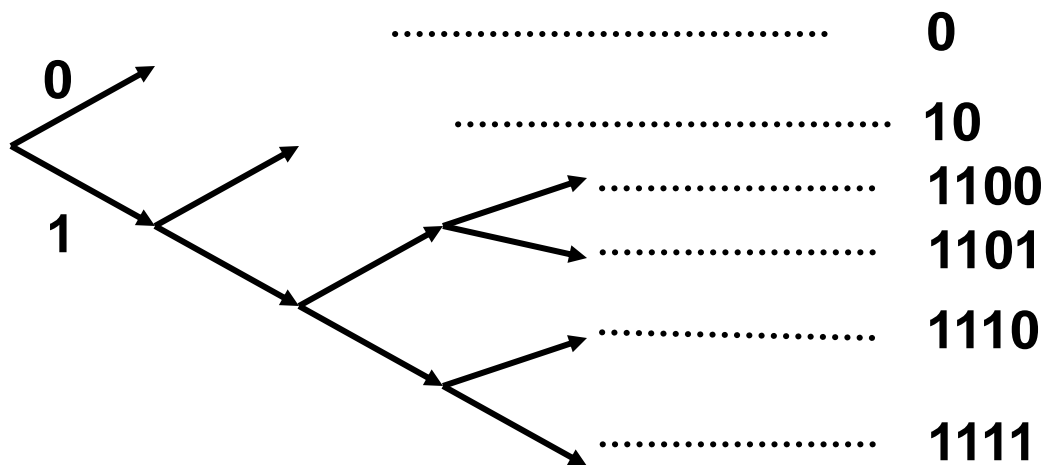
Examples

- ❖ Is there an instantaneous prefix code with a col. of lengths $\{1, 2, 2, 3\}$?
 - NO, since $2^{-1} + 2^{-2} + 2^{-2} + 2^{-3} > 1$
- ❖ How about $\{1, 2, 3, 3\}$?

Example

❖ {1, 2, 4, 4, 4, 4}

$$2^{-1} + 2^{-2} + 4 * 2^{-4} = 1/2 + 1/4 + 4 * 1/16 = 1$$



Optimal Codes

- ❖ Find a prefix code that has the minimum $L=E(l)$.

There are many prefix codes satisfying the Kraft inequality.

$E(l)$ means there is a distribution.

- ❖ First, find the lengths $\{l_1, \dots, l_m\}$ which satisfy the Kraft inequality and have the minimum L .

Given the distribution, p_i 's, map them to lengths l_i 's.

Find the lengths l_i 's satisfying KI, map them to p_i 's.

Looks like we are doing the first.

Given p_i 's find l_i 's.

- ❖ Second, construct the prefix code using the tree.

Let's use the Calculus, just to have an idea

❖ Use the Lagrange multiplier method to solve

– Minimize $L = \sum p_i l_i$

– Subject to $\sum D^{-l_i} \leq 1$

Find l_i 's
given p_i 's.

– Let's relax the condition and let l_i be any positive number.

– Let $l_i = l(C(x_i))$, $i=1, \dots, |\mathcal{X}|$ and $p_i = \Pr\{X = x_i\}$.

– Note that $|\mathcal{X}| = m$

Lagrange multiplier minimization

❖ Let $J = \sum p_i l_i + \lambda (\sum D^{-l_i})$

❖ Differentiate wrt each l_i and obtain

$$\partial J / \partial l_i = p_i - \lambda D^{-l_i} \log D,$$

❖ Setting it equal to 0, we have

$$D^{-l_i} = p_i / (\lambda \log D)$$

❖ Substitute this in the constraint $\sum D^{-l_i} = 1$ and obtain $\lambda = 1 / \log D$

❖ Thus, $p_i = D^{-l_i}$ or $l_i^* = -\log_D p_i$

❖ Then, $L = E\{l(X)\} = \sum p_i l_i^* = -\sum p_i \log_D p_i = H_D(X)$

$$L \geq H_D(X)$$

- ❖ Since *codeword length* must be integer, we must do the ceiling operation, i.e.

$$l_i = \lceil l_i^* \rceil$$

- ❖ Thus, $L \geq H_D(X)$, equality *iff* l_i^* integers or $p_i = D^{-l_i}$
- ❖ Ex) $\{p_i\} = \{1/2, 1/4, 1/16, 1/16, 1/16, 1/16\}$
 - $\{C(x_i)\} = \{0, 10, 1100, 1101, 1110, 1111\}$
 - Lengths 1, 2, 4, 4, 4, 4
 - $E(L) = (1/2) * 1 + (1/4) * 2 + (1/4) * 4 = 2$
 - $H = (1/2) * \log_2(2) + (1/4) * \log_2(4) + 4 * (1/16) * \log_2(16) = 2$
 - $E(L) = H$ since $p_i = 2^{-l_i}$, $l_i = 1, 2, 4, 4, 4, 4$

D-adic distribution

- ❖ A distribution where each of the probabilities is equal to D^{-n} .
- ❖ The lower bound on expected length, $L \geq H_D(X)$ is achieved iff the distribution is *D-adic* (i.e. $p_i = D^{-li}$).

Information Theoretic Proof: $L \geq H_D(X)$

$$\begin{aligned} \spadesuit L - H_D(X) &= \sum_x p(x) l(x) - \sum_x p(x) \log_D(1/p(x)) \\ &= \sum_x p(x) \{ \log_D[p(x)] - \log_D[D^{-l(x)}] \} \\ &\quad \text{--- let } r(x) = D^{-l(x)}/r_o \\ &\quad \text{--- with } r_o = \sum D^{-l(x)} \leq 1 \text{ (Kraft Inequality)} \\ &= \sum_x p(x) \{ \log_D[p(x)] - \log_D[r(x)r_o] \} \\ &= \sum_x p(x) \{ \log_D[p(x)] - \log_D[r(x)] - \log_D[r_o] \} \\ &= \sum_x p(x) \log_D[p(x)/r(x)] + \log_D[1/r_o] \\ &\geq 0 \text{ (why?)} \end{aligned}$$

A possible procedure to find an optimal code

- ❖ Find the D -adic distribution that is closest (in the relative entropy sense) to the distribution of X
 - $p_i = D^{-l_i}$, $i = 1, \dots, |\mathcal{X}|$
 - So now you have $\{l_i\}$, the col. of codeword lengths.
- ❖ Construct a D -adic tree according to the col.
- ❖ Assign codewords on the leaves of the tree.

- ❖ The first step of searching for the closest D -adic distribution is not trivial.
- ❖ We may use a sub-optimal procedure.

$$H_D(\mathbf{X}) + 1/n > L_n \geq H_D(\mathbf{X})$$

- ❖ Previous pages say we have an overhead of maximum one bit per symbol.
- ❖ Now, consider a series of r.v.s, X_1, X_2, \dots ,
- ❖ We want to get rid of the one overhead bit per symbol by spreading out over many symbols.
- ❖ For a simple example, consider a group of iid $X_1, \dots, X_n \sim p(x)$. Then, the distribution is $P_n(x) = \prod_{i=1}^n p(x)$.
- ❖ Then, we have
$$H_D(X_1, \dots, X_n) \leq E\{l(X_1, \dots, X_n)\} < H_D(X_1, \dots, X_n) + 1$$
- ❖ Dividing by n , we have the expected length per symbol L_n
$$H_D(\mathbf{X}) \leq L_n < H_D(\mathbf{X}) + 1/n$$

X_1, X_2, \dots, X_n is a stationary stochastic process

- ❖ We know $H(X_1, \dots, X_n)/n \rightarrow H(\mathcal{X})$ (Cesaro Mean)
 - $nH(\mathcal{X})$ bits \sim sufficient for description of typical seq. of length n
- ❖ Then, the minimum expected codeword length per symbol converges to the entropy rate of the process

$$L_n^* \rightarrow H(\mathcal{X}) \text{ as } n \rightarrow \infty$$

Messages learned

- ❖ Thus, one can always construct a near optimal prefix code.
 - Use the Shannon code, $l_i = \lceil \log_D(1/p_i) \rceil$
 - Use a sequence, rather than a symbol (vector processing, rather than a symbol processing)

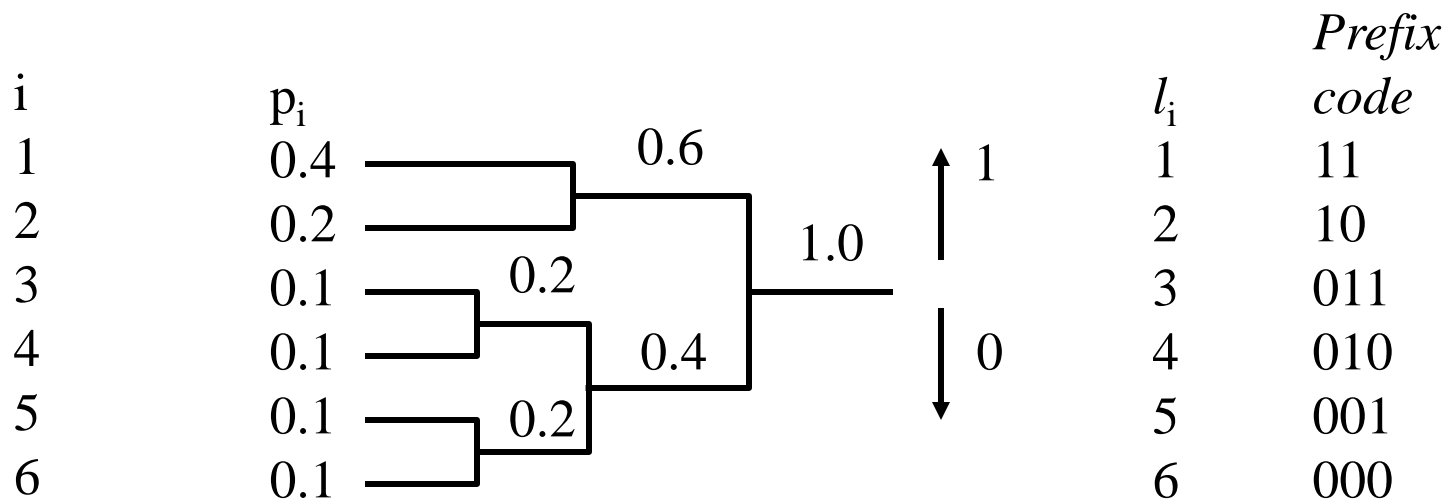
Two Important Coding Methods

❖ Huffman code

❖ Lempel-Ziv code

Huffman Code (Huffman Tree)

- ❖ Huffman code is an optimal *prefix* code, with the shortest *expected* length for a given distribution, which can be constructed by the Huffman algorithm.
- ❖ It minimizes $L = \sum_i p(x_i) l_i$.
- ❖ Compare it with $H = \sum_i p(x_i) \log(1/p(x_i))$.



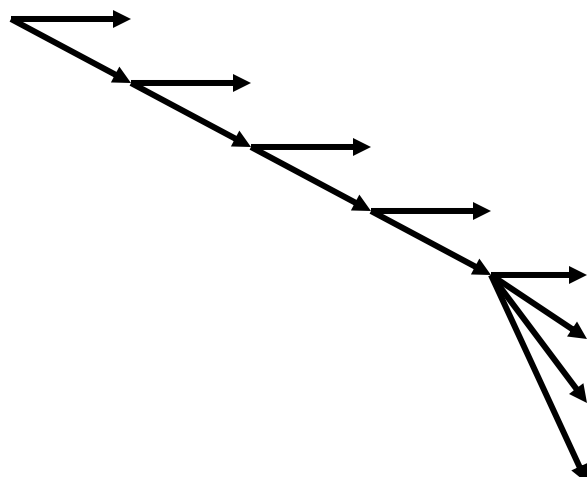
Huffman Code

❖ $L = 0.4*2 + 0.2*2 + 4*0.1*3 = 2.4$

❖ $H = 2.319$ (Lower bound)

- It should be noted here that the expected length turned out quite close to the lower bound.

Recall the 8-Horse Race Problem

p_i				Length
1/2	0		0	1
1/4	1		10	2
1/8	2		110	3
1/16	3		1110	4
1/64	4		111100	6
1/64	5		111101	6
1/64	6		111110	6
1/64	7		111111	6

- ❖ The code we constructed is a Huffman code.
- ❖ The distribution is 2-adic.
- ❖ $L = 2 = H$

Huffman Encoding Algorithm

- ❖ Construct a tree in the following routine:
 - First, have the probabilities (and the index) listed in the decreasing order.
 - Second, go over the list and locate two indices with lowest probabilities. Group these low probable items, and add the probabilities and label it for the group. Now note that the size of the list is reduced by one.
 - Repeat the second step exhaustively.
- ❖ Once a tree is completed, we can assign “1” for up and “0” for down from the top of the tree.

Optimality of Huffman Codes

- ❖ Given a distribution, there could be many Huffman codes which all lead to the same expected length.
- ❖ Let's call the Huffman tree procedure Huffman coding.
- ❖ Huffman coding is optimal in the sense that if C^* is a Huffman code, and C' is a code from other coding procedure, then $L(C^*) \leq L(C')$
 - See Ch5.8 for proof by induction

Lempel-Ziv Coding (Ch13)

- ❖ Construction of Huffman code requires knowledge of priors.
- ❖ Huffman code does not make use of correlation between words and phrases (based on the assumption that text is generated from a random variable, rather than a random process)
- ❖ Lempel-Ziv code figures out the correlation structure of the source in a sequence, and further compress
 - LZ code is a universal code (does not need to know the distribution or correlation of the source).
 - LZ code is dictionary based.
- ❖ It's adaptive and simple.
- ❖ **Operation:** parse the data stream into segments that are the shortest subsequences not appeared previously.

Lempel-Ziv Encoding

❖ Let's take a simple example for illustration of the algorithm

❖ Consider a data stream

aaababbbbaaabaabbbababb

❖ Starting from the left of the stream, parse the data stream into segments not appeared previously

a, aa, b, ab, bb, aaa, ba, aaaa, bba, bab, b

Index-- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Encoding-- a, 1a, b, 1b, 3b, 2a, 3a, 6a, 5a, 7b, 3

❖ (Appeared, new) = (Index, new) = (4 bits, 7 bits)

❖ The total number of bits per segment needed in this example is 4 bits for the index and 7 bit ascii code for the new part.

– 11 segments \times (4 + 7) bits = 121 bits

– 24 characters \times 7 bits = 168 bits

– (168-121)/168 is the compression ratio in this example

Lempel-Ziv Coding (MATLAB)

- ❖ We will compress the following text with an LZ algorithm that I programmed in MATLAB, `Lempel_ziv.m`
- ❖ [Jordan_text.txt](#)
- ❖ [H1N1_virus.txt](#)

Huffman Codes vs. Shannon Codes

- ❖ X is binary, X=1 with prob. 2^{-10} , X=0 with prob. $1 - 2^{-20}$
- ❖ Shannon Code: $l(x=1) = \lceil \log_2(2^{10}) \rceil = 10$, $l(x=0) = 1$
- ❖ Huffman Code: $l(x=1) = 1$ and $l(x=0) = 1$
- ❖ A codeword for infrequent symbol in Shannon Code is much longer than in an optimal code.
- ❖ But, this does not mean that a codeword in an optimal code is always shorter than those in the Shannon code.
- ❖ Ex) $X \sim (1/3, 1/3, 1/4, 1/12)$
 - two kinds of Huffman trees with lengths (2, 2, 2, 2) or (1, 2, 3, 3)
 - Shannon lengths (2, 2, 2, 4)

HW

HW #4

- ❖ P4.4, 4.7, 4.11, 4.24
- ❖ P5.3, 5.6, 5.7, 5.16, 5.18