Mathematical methods in communication

Lecture 3

Lecturer: Haim Permuter *Scribe: Yuval Carmel, Dima Khaykin, Ziv Goldfeld*

I. Reminder

A. Convex Set

A set R is a convex set iff,

$$\forall x_1, x_2 \in R,\ \forall \theta,\ 0 \leq \theta \leq 1,\ \theta x_1 + \bar{\theta} x_2 \in R, \tag{1}$$

where $\bar{\theta} = 1 - \theta$.

B. Convex Function

A function $f : \mathbb{R} \mapsto \mathbb{R}$ is a convex function iff,

$$\forall x_1, x_2 \in R,\ \forall \theta,\ 0 \leq \theta \leq 1,\ f(\theta x_1 + \bar{\theta} x_2) \leq \theta f(x_1) + \bar{\theta} f(x_2). \tag{2}$$

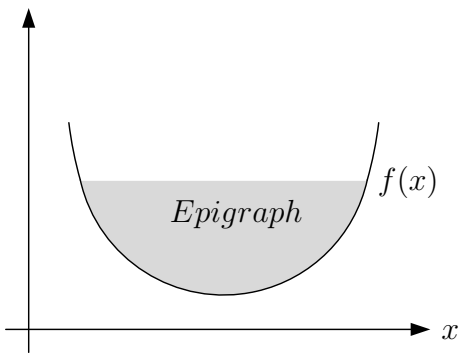


Fig. 1. Epigraph

- Epigraph[1] is the set of points lying on or above its graph.

[1]epi-, ep- (Greek): above, over, on, upon

- Mathematical definition of $f : \mathbb{R}^n \mapsto \mathbb{R}$: epi$f = \{(x, \mu) : x \in \mathbb{R}^n, \mu \in \mathbb{R}, \mu \geq f(x)\} \subseteq \mathbb{R}^{n+1}$
- A function is convex if and only if its epigraph is a convex set.
- If the second derivative of a function $f : \mathbb{R} \mapsto \mathbb{R}$ exists in the interval (a, b) and it satisfies $\frac{d^2 f(x)}{dx^2} \geq 0$ for all $x \in (a, b)$, then the function is convex in this domain
- A function may be convex even if the second derivative does not exists. The function $|x|$ is convex but the second derivative at $x = 0$ does not exists.

C. Convexity and Concavity of Information Theory Related Functions

- $D(p||q)$ is a convex function in the pair (p, q)
  $D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \geq 0$
  $D(p||q) = 0 \Leftrightarrow p(x) = q(x), \ \forall x \in \mathcal{X}$
- $H(X)$ is concave in $p(x)$
- $I(X;Y)$ is concave in $p(x)$, when $p(y|x)$ is fixed
- $I(X;Y)$ is convex in $p(y|x)$, when $p(x)$ is fixed

II. Lossless Source Coding/ Data Compression

In this section we look at a problem called *lossless source coding*. In this problem there exists source sequence $X^n = (X_1, X_2, ..., X_n)$, where the elements of source sequence are distributed i.i.d. according to P_X, and each symbol $x \in \mathcal{X}$ is encoded into a codeword $C(x)$ with length $l(x)$. The receiver obtains the sequence of codewords $\{C(X_i)\}_{i=1}^n$, and needs to reconstruct the source sequence $\{\hat{X}_i\}_{i=1}^n$ losslessly, namely, with probability 1.

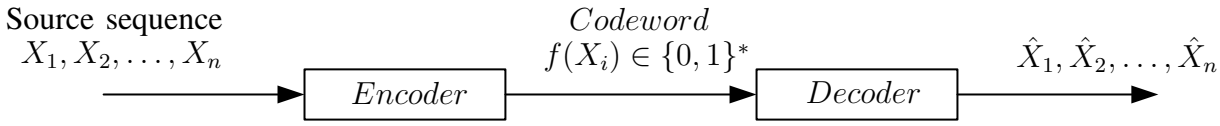


Fig. 2. Lossless source coding problem. The encoder in the figure converts a value $x \in \mathcal{X}$ into sequence of bits $\{0,1\}$ of variable length denoted as $\{0,1\}^{l(x)}$, where $X_i \sim \ i.i.d \ p(x)$. The term $l(x)$ is the length of the codeword associated with value $x \in \mathcal{X}$.

Definition 1 (Operational definition of instantaneous source coding) An *instantaneous source code* C consists an encoding function

$$f : \mathcal{X} \mapsto \{0,1\}^{l(x)}, \tag{3}$$

where $l(x)$ is the length of the codeword associated with symbol X, and a decoding function that maps a sequence of bits into a sequence of estimated symbols $\hat{X}$.

Definition 2 (Lossless source code) We say that a code is *lossless* if $\Pr\{X^n = \hat{X}^n\} = 1$.

Definition 3 (Expected length of a code) The *expected length* of a code is $E[l(X)]$.

Here, our goal as engineers is to design a code that minimize the expected length of a code. By the law of large number, since the source is i.i.d. then the average length of the code will be with high probability $E[l(X)]$. The following example illustrates a code and its expected length.

Example 1 (Mean Length) Let X be the source with an alphabet $\mathcal{X} = \{A, B, C, D\}$

$$X = \begin{cases} A & p = \frac{1}{2} \\ B & p = \frac{1}{4} \\ C & p = \frac{1}{8} \\ D & p = \frac{1}{8} \end{cases} \tag{4}$$

Consider the following encoding function f, decoding function g, and the associated code-length.

$$f(x) = \begin{cases} A & \rightarrow & 0 \\ B & \rightarrow & 10 \\ C & \rightarrow & 110 \\ D & \rightarrow & 111 \end{cases} \quad g(x) = \begin{cases} 0 & \rightarrow & A \\ 10 & \rightarrow & B \\ 110 & \rightarrow & C \\ 111 & \rightarrow & D \end{cases} \quad l(x) = \begin{cases} 1\ [bits] \\ 2\ [bits] \\ 3\ [bits] \\ 3\ [bits] \end{cases} \tag{5}$$

The expected code length obtained in this example is

$$\mathbb{E}\left[l(X)\right] = 1\frac{6}{8}\ [bits] \tag{6}$$

Definition 4 (Codebook) A codebook is a list of all codewords in the code.

For instance the codebook in Example 1 is $\{0, 10, 110, 111\}$. We would denote a codeword by c_i (e.g., $c_1 = 0$, $c_2 = 10$) and the length of codeword c_i is l_i.

Definition 5 (Non-singular code) A code is *non-singular* if for any $x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$

Definition 6 (Uniquely Decodable Code) We say that a code is uniquely *decodable*, if any extension of codewords is non-singular. An extension of codewords is a concatenation $f(x_1)f(x_2)f(x_3)f(x_4)\ldots$ without any spaces or commas.

Definition 7 (Prefix code)

A code is a *prefix code* a.k.a *instantaneous code* , if no codeword is a prefix of any other codeword.

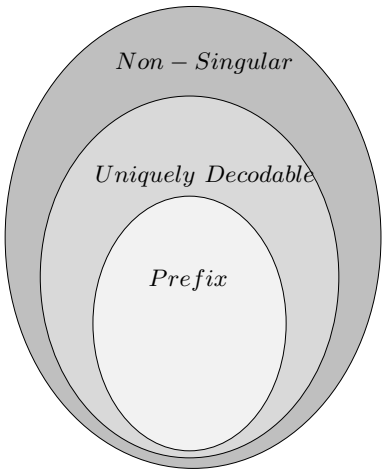


Fig. 3. The relations between non-singular, uniquely-decodable and prefix code

Figure 3 illustrates the relations between the three different classes of codes defined above.

III. KRAFT INQUALITY

Theorem 1 (Kraft Inequelity) For any prefix code $c_1, c_2, \ldots, c_m$, with length $l_1, l_2, \ldots, l_m$, we have:

$$\sum_i 2^{-l_i} \leq 1 \tag{7}$$

If there is $l_1, l_2, \ldots, l_m$ that satisfies (7), then there exists a prefix code with length $l_1, l_2, \ldots, l_m$.

Proof:

We define: $l_{max} = \max\{l_1, l_2, \ldots, l_m\}$. We are creating a Binary tree of depth l_{max} as

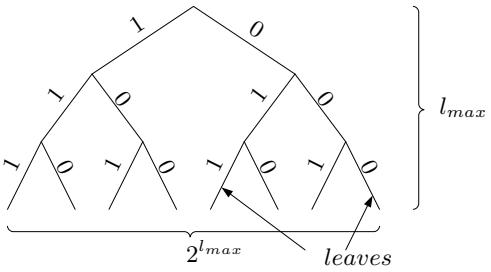


Fig. 4. Binary Tree

depicted in Fig. 4. The left branch is associated with 1 in the code and the right branch

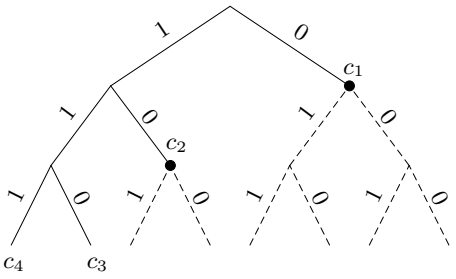


Fig. 5. Codewords and Amount of Leaves

with 0 in the code. Now, we draw the codewords $c_1, c_2, ..., c_m$ on the tree as depicted in Fig. 5.

The number of leaves (nodes at the bottom of the tree) associated with codeword c_i is $2^{l_{max}-l_i}$. Since the code is a prefix code, the leaves associated with the codewords are disjoint. Therefore,

$$\sum_{i=1}^{m} 2^{l_{max}-l_i} \leq 2^{l_{max}}, \tag{8}$$

and this implies that Kraft inequality given in (7) holds.

Now we prove the second direction. Namely, that if there is a list of lengths $l_1, l_2, ..., l_m$ that satisfies the Kraft inequality, then there exists a codebook $c_1, c_2, ..., c_m$ with length $l_1, l_2, ..., l_m$ that is a prefix codebook. The idea is to place the codewords on the tree in such a way that there will be no overlap of the leaves associated with each codeword.

Let $l_1 \leq l_2 \leq \cdots \leq l_m$. For length l_i associate $2^{l_{max}-l_i}$ codewords and place the codebook such that it covers only those leaves. For instance, l_1 is with length of 1 bits and reduces 4 leaves, in general, each codeword i reduces $2^{l_{max}-l_i}$ leaves out of $2^{l_{max}}$ or 2^{-l_i} fraction of the leaves. The next codeword has, therefore, another branch to start from, meaning, different prefix. Since $\sum_i 2^{-l_i} \leq 1$ there would be enough leaves to associate with each codeword and because there is no overlap of the leaves the codebook is a prefix codebook. ■

IV. Lower bound on expected length of prefix code

Theorem 2 (Lower bound on expected length of prefix code) The expected length L of any instantaneous (prefix) code for a random variable X is greater than or equal to the entropy $H(X)$; that is,

$$L \geq H(X). \tag{9}$$

Proof: We can write the difference between the expected length and the entropy as

$$\begin{aligned} L - H(X) &= \sum_i p_i l_i + \sum_i p_i \log p_i & (10)\\ &= -\sum_i p_i \log 2^{-l_i} + \sum_i p_i \log p_i. & (11) \end{aligned}$$

Let us normalize the term 2^{-l_i} in order to obtain a pmf.

$$L - H(X) = -\sum_i p_i \log \frac{2^{-l_i}}{\sum_j 2^{-l_j}} + \sum_i p_i \log \frac{1}{\sum_j 2^{-l_j}} + \sum_i p_i \log p_i. \quad (12)$$

Now, by letting $r_i = \frac{2^{-l_i}}{\sum_j 2^{-l_j}}$, we obtain

$$L - H(X) = -\sum_i p_i \log r_i + \sum_i p_i \log p_i + \log \frac{1}{\sum_j 2^{-l_j}}. \quad (13)$$

Note that

$$\sum_i r_i = 1. \quad (14)$$

Using the definition of relative entropy we can write

$$L - H(X) = D(p||r) + \log \frac{1}{\sum_j 2^{-l_j}}. \quad (15)$$

Since $D(p||r) \geq 0$ and $\log \frac{1}{\sum_j 2^{-l_j}} \geq 0$ due to Kraft inequality, given in Theorem 1, we obtained that $L - H(X) \geq 0$. ■

Note that equality is achieved if and only if

$$p_i = 2^{-l_i}, \quad (16)$$

or equivalently,

$$l_i = \log \frac{1}{p_i} \quad (17)$$

This leads to the following definition and corollary:

Definition 8 (D-adic distribution) A distribution, P_X is called *D-adic distribution if for any $x \in \mathcal{X}$, there exists an integer n_x, s.t. $P(x) = D^{-n_x}$.*

Corollary 1 (Optimality of 2-adic code) There exists a binary prefix code for a source X with an average length $H(X)$ if and only if X has a 2-adic .

V. SHANNON-FANO CODE

Lengths l_i of Shannon-Fano Code defined as

$$l_i = \left\lceil \log \frac{1}{p_i} \right\rceil . \tag{18}$$

These lengths satisfy the Kraft inequality

$$\begin{aligned} \sum_i 2^{-l_i} &= \sum_i 2^{-\left\lceil \log \frac{1}{p_i} \right\rceil} && (19)\\ &\leq \sum_i 2^{-\log \frac{1}{p_i}} && (20)\\ &= \sum_i 2^{\log p_i} && (21)\\ &= \sum p_i && (22)\\ &= 1; && (23) \end{aligned}$$

hence according to Theorem 1, there exists a prefix code with lengths given in (18).

The expected length is bounded by

$$\begin{aligned} L &= \sum p_i l_i = \sum p_i \left\lceil \log \frac{1}{p_i} \right\rceil && (24)\\ &\leq \sum_i^{|\mathcal{X}|} p_i \left(\log \frac{1}{p_i} + 1 \right) && (25)\\ &= H(X) + 1 && (26) \end{aligned}$$

Example 2 (Wrong coding) Let $X \sim p(x)$, and wrongly we assume $X \sim q(x)$, what would be the expected length L while $l_i = \left\lceil \log \frac{1}{q_i} \right\rceil$?

Solution:

$$\begin{aligned} L_q &= \sum p_i l_i = \sum p_i \left\lceil \log \frac{1}{q_i} \right\rceil && (27)\\ &\leq \sum_i p_i \left(\log \frac{1}{q_i} + 1 \right) && (28)\\ &= \sum_i p_i \log \frac{p_i}{q_i} - \sum_i p_i \log p_i + 1 && (29) \end{aligned}$$

$$= D(p||q) + H(X) + 1 \tag{30}$$

$$= D(p||q) + L_p \tag{31}$$

The expected length is increased by $D(p||q)$.

VI. Huffman Code

An optimal (shortest expected length) prefix code for a given distribution can be constructed by a simple algorithm discovered by Huffman. Let us introduce Huffman codes with some examples. Consider a random variable X taking values in the set $\mathcal{X} = \{1, 2, 3, 4, 5\}$ with probabilities $\{0.3\ 0.25\ 0.25\ 0.1\ 0.1\}$, respectively. We expect the optimal binary code for X to have the longest codewords assigned to the symbols 4 and 5. These two lengths must be equal, since otherwise we can delete a bit from the longer codeword and still have a prefix code, but with a shorter expected length. In general, we can construct a code in which the two longest codewords differ only in the last bit. For this code, we can combine the symbols 4 and 5 into a single source symbol, with a probability assignment 0.2. Proceeding this way, combining the two least likely symbols into one symbol until we are finally left with only one symbol, and then assigning codewords to the symbols, we obtain the tree that appears in Fig. 6. This code

Codeword Length	Codeword	X	Probability
2	00	1	0.3
2	01	2	0.25
2	10	3	0.25
3	110	4	0.1
3	111	5	0.1

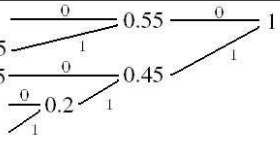


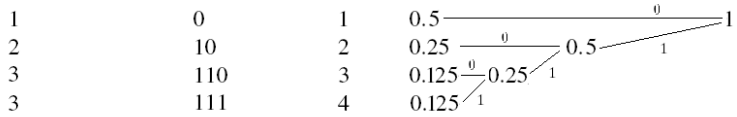
Fig. 6. Example of a construction of Huffman Code

has average length 2.2 bits.

If the probability vector is dyadic, meaning $p_i = 2^{-n_i}$, the Huffman code achieves $H(X)$.

Example:

Codeword Length	Codeword	X	Probability
1	0	1	0.5
2	10	2	0.25
3	110	3	0.125
3	111	4	0.125



In this case the expected length L is equal to entropy $H(X)$:

$$L = E[l_i] = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = 1\frac{6}{8} = H(X) \tag{32}$$

Optimality of Huffman code

Lemma 1 For any distribution, there exists an optimal instantaneous code (with minimum expected length) that satisfies the following properties:

1) The lengths are ordered inversely with the probabilities (i.e. if $p_i \geq p_j,$ then $l_i \leq l_i$).
2) The two longest codewords have the same length.
3) Two of the longest codewords differ only in the last bit and correspond to the two least likely symbols.

APPENDIX I
ALTERNATIVE PROOF OF KRAFT'S INEQUALITY

We now present the infinite version of Kraft inequality.

Theorem 3 (Kraft's Inequality) (i) For any prefix code $\{c_i\}_{i\geq 1}$, with lengths $\{l_i\}_{i\geq 1}$ we have:

$$\sum_{i\geq 1} 2^{-l_i} \leq 1 \tag{33}$$

(ii) Conversely if $\{l_i\}$ satisfies (34), then there exists a prefix code with these lengths.

Remark 1 The following proof of Kraft's inequality is preferable compared to the previous proof that was presented because it doesn't demand a finite set of codewords or lengths. However, part (i) of Theorem 3 is equivalent to saying that for any subset of m-lengths $l_1, l_2, l_3, ..., l_m$ from the code we have

$$\sum_{i \geq 1} 2^{-l_i} \leq 1, \tag{34}$$

and this follows directly from the finite version of Kraft inequality given in Theorem 1 since any subset of the code such as $\{c_1, c_2, ..., c_m\}$ is also a prefix code. However, part (ii) of the theorem requires a reconstruction of a prefix-code of all infinite symbols and the finite version can not be used.

The main idea of this proof is exactly the same idea as the proof with the tree just that we use intervals on [0,1] rather then leaves. And instead of having disjoint leaves we have here disjoint intervals.

Proof: We start by proving part (i):

Let $\{c_i\}$ be a prefix code, where c_i is a codeword of length $l_i = |c_i|$. We define a function $f : c_i \longrightarrow [0, 1]$ that calculates the decimal value of c_i, by:

$$f(c_i) = \sum_{j=1}^{l_i} c_{i,j} \cdot 2^{-j}$$

For future reference, we inspect the interval $\left[f(c_i), f(c_i) + 2^{-l_i}\right)$. Note that:

1. $0 \leq f(c_i) \leq 1$.

2. $f(c_i 000 \ldots 0) = f(c_i)$. i.e. adding zeroes at the end of the codeword does not change the value of $f(c_i)$.

3. $f(c_i 111 \ldots) = f(c_i) + \sum_{j=1}^{\infty} 2^{(-l_i+j)} = f(c_i) + 2^{-l_i}$. This follows from the fact that

$$\sum_{i=1}^{\infty} 2^{-i} = 1. \tag{35}$$

We assume that the $\{c_i\}$ are arranged in an increasing lexicographic order[1], which means that $f(c_i) \leq f(c_k)$ for all $i \leq k$.

Since c_i is a prefix code we have:

$$f(c_{i+1}) \geq f(c_i) + 2^{-l_i} \tag{36}$$

Thus, we get that the intervals $\left[f(c_i), f(c_i) + 2^{-l_i}\right)$ are pairwise disjoint.

By recurrent use of Inequality (36) we obtain:

$$f(c_m) \geq \sum_{i=1}^{m} 2^{-l_i}$$

Since, by definition $f(c_m) \leq 1$, this proves the first part of the theorem, i.e.

$$\sum_{i=1}^{m} 2^{-l_i} \leq 1$$

■

We have seen that a necessary condition for a code $\{c_i\}$ to be prefix is that the intervals $\left[f(c_i), f(c_i) + 2^{-l_i}\right)$ are pairwaise disjoint. The proof of the second part of the theorem is based upon the claim that this condition is also sufficient:

Lemma 2 Given a code $\{c_i\}$ such that the intervals $\left[f(c_i), f(c_i) + 2^{-l_i}\right)$ are disjoint, the code is prefix.

Remark 2 In the following proof we use the fact that in order to prove $A \Rightarrow B$ one can show that $B^c \Rightarrow A^c$ (i.e. not $B \Rightarrow$not A).

Proof: We conversly assume that the code $\{c_i\}$ is not prefix. If it is so, we can find two codewords c_m and c_n (without loss of generality we assume $m > n$ thus $l_m > l_n$), for which the first $|c_n|$ bits of c_m are identical to the bits of c_n. In this case:

$$f(c_m) = \sum_{j=1}^{l_m} c_{m,j} \cdot 2^{-j} = \sum_{j=1}^{l_n} c_{n,j} \cdot 2^{-j} + \sum_{j=l_n+1}^{l_m} c_{m,j} \cdot 2^{-j} < f(c_n) + 2^{-l_n}$$

[1]it is always possible to arrange a code in a lexicographic order however there might be infinite codewords that have a lower lexicographic order then a specific codeword, hence assuming a finite index i is not always possible. This can be corrected by avoiding the indexing i.

So we get that $f(c_m) < f(c_n) + 2^{-l_n}$, contradictly to the fact that the intervals $\left[f(c_n), f(c_n) + 2^{-l_n}\right)$ and $\left[f(c_m), f(c_m) + 2^{-l_m}\right)$ are pairwise disjoint. Thus the code is prefix. ■

Proof: of (ii):

Assume that the lengths $\{l_i\}$ are given and satisfy Kraft's inequality (34). We prove that we can find a prefix code with the given lengths. Without loss of generality, assume that $l_1 \leq l_2 \leq ....$ We define the word c_i to be the inverse image under the mapping f of the number $\sum_{j=1}^{i-1} 2^{-l_j}$, i.e. c_i is the only word (up to addition of zeroes from the right) such that the equality

$$f(c_i) = \sum_{j=1}^{i-1} 2^{-l_j}$$

holds.

To calculate c_i we use the function $f^{-1} : [0,1] \longrightarrow c_i$. In order to justify that use we first show that $0 < f(c_i) \leq 1$.

From the structure of $f(c_i)$ it is easy to see that $f(c_i) > 0$ for every i. Moreover, using the assumption of the theorem (i.e. inequality (34)) we get that

$$f(c_i) = \sum_{j=1}^{i-1} 2^{-l_j} \leq 1$$

for every i. Thus we get that $0 < f(c_i) \leq 1$.

Next show that the length of every codeword c_i that is built this way is indeed no longer than l_i.

Again from the structure of $f(c_i)$, it is simple to see that the maximal number of bits needed for the codeword c_i is l_{i-1} bits. Because we assume that the lengths are arranged by rising order (i.e. $l_{i-1} \leq l_i$ for every i), the length of each codeword c_i cannot be longer than $|c_i| = l_i$. If it is shorter, we add zeroes from the right up to the wanted length.

To complete the proof, it is enough to show that the intervals

$$I_i = \left[f(c_i), f(c_i) + 2^{-l_i}\right) = \left[\sum_{j=1}^{i-1} 2^{-l_j}, \sum_{j=1}^{i} 2^{-l_j}\right)$$

are pairwise disjoint and finally use Lemma 1.

Since by definition, $f(c_i)$ increases as i increases and the right border of the interval I_i is the left border of the interval I_{i+1} , the intervals $\{I_i\}$ are pairwise disjoint, which concludes the proof. ■