Week 6

1 Random variable source coding

Let $\underline{c}(x)$ be the codeword assigned to $x \in \mathcal{X}$.

$l(x)$ be the length of codeword assigned to x.

Here we are coding a single random variable and all codewords are binary strings. (Fixed-variable length source coding)

$$\mathscr{C} = \{\underline{c}(x) : x \in \mathcal{X}\}$$

$$L_{\mathscr{C}} = \sum_{x \in \mathcal{X}} p(x) l(x)$$

Our goal is to design a code which has minimum $L_{\mathscr{C}}$, L^*

$$L^* = \min_{\mathscr{C}} L_{\mathscr{C}}$$

1.1 Kraft inequality

Let $\mathscr{C}$ be any prefix-free (binary) code. Then,

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1$$

Proof: We know that any prefix-free code can be represented using a binary tree which has 'leaves' as the codeword. In the binary tree corresponding to the code, the depth of the tree would be the length of the largest codeword in the code. ($l_{\max} = max_{x \in \mathcal{X}} l(x)$)

Suppose there is a codeword $\underline{c}$ of length l represented by a node at depth l($l \leq l_{\max}$).

Then $\underline{c}$ has $2^{l_{\max}-l}$ successors at level $l_{\max}$. Also, none of these are codewords as the code is prefix-free. Now,

$$\sum_{\underline{x} \in \mathcal{X}} 2^{l_{\max} - l(\underline{x})} \leq 2^{l_{\max}} \tag{1}$$

This is true if distinct codewords $\underline{c1}$, $\underline{c2}$ don't have common successors at $l_{\max}$ level. Any successor of $\underline{c1}$ at $l_{\max}$ has $\underline{c1}$ as a prefix. Similarly in the other case as well.

Suppose $l(\underline{c1}) \leq l(\underline{c2})$, so there can be a common successor $\underline{v}$ at $l_{\max}$.

Then, $\underline{v}$ has first $l(\underline{c1})$ places as $\underline{c1}$ and first $l(\underline{c2})$ places as $\underline{c2}$.

$\Rightarrow$ $\underline{\text{c1}}$ should be a prefix of $\underline{\text{c2}}$ which isn't true as $\mathscr{C}$ is a prefix-free code. Hence, no pair of codewords in $\mathscr{C}$ have any common successors.

Hence (1) is true. And dividing both sides by $2^{l_{\max}}$ gives us the Kraft inequality.

1.2 Lemma

$$L^* \geq H(X)$$

Any prefix-free code for X has average length of at least $H(X)$.

Proof:

$$L - H(X) = \sum_{x\in\mathcal{X}} p(x)l(x) - \sum_{x\in\mathcal{X}} p(x)\log\frac{1}{p(x)}$$

We know,

$$D(p||q) = \sum_{x\in supp(P_X)} p(x)\log\frac{p(x)}{q(x)}$$

Let

$$q(x) := \frac{2^{-l(x)}}{\sum_{x\in\mathcal{X}} 2^{-l(x)}}$$

$$\begin{aligned}
D(p_X||q_X) &= \sum_{x\in supp(P_X)} p(x)\log\frac{p(x)}{\frac{2^{-l(x)}}{\sum_{x'\in\mathcal{X}} 2^{-l(x')}}}\\
&= -\sum p(x)\log\frac{1}{p(x)} + \sum p(x)\log\frac{1}{\frac{2^{-l(x)}}{\sum_{x'\in\mathcal{X}} 2^{-l(x')}}}\\
&= -H(X) + \sum_{x\in supp(p_x)} p(x)\log 2^{l(x)} + \sum_{x\in supp(p_x)} p(x)\log\sum_{x'} 2^{-l(x')}\\
&= -H(X) + L - \epsilon \qquad \text{(from Kraft's)}\\
&\leq -H(X) + L
\end{aligned}$$

$$\Rightarrow L_{\mathscr{C}} - H(X) \geq 0$$

$$L^* \geq H(X)$$

Note: Equality happens only iff $p_x = q_x$ and ϵ is zero.

1.3 Lemma

Suppose we have a random variable $X \in \{x_1, x_2, \cdots, x_k\}$ and positive integers such that $\sum_{i=1}^{k} 2^{-l_i} \leq 1$.

Then there exists a prefix free code for X with codeword lengths $l_1, l_2, \cdots, l_k$.

Proof: We can construct a binary tree with leaves at depths $l_1, l_2, cdots, l_k$ such that none of these nodes are successors of each other, i.e. they are leaves of some binary tree (valid p-f code).

Assume that $l_1 \leq l_2 \leq \cdots \leq l_k$, without loss in generality , for any $i \leq k$,

$$\sum_{1}^{i-1} 2^{-l_i} < 1 \tag{2}$$

Imagine we take the full binary tree upto level l_k. At each step of the algorithm we intend to pick one available (undeleted) node from the above tree at level l_i and delete all it's successors from the tree. Repeat this process for $i = 1, \cdots, k$, we will then have a prefix-free tree.

We have to show that at each step $i = 1, \cdots, k$, there is atleast one node left undeleted at depth i. We shall use observation (2).

Clearly at step 1, there is a node at l_1, $(l_1 \geq 1)$.

After $i-1$ steps, assume that we have picked nodes at level $l_1, \cdots, l_{i-1}$ and appropriately deleted. We want to show that there is a node at level i.

Total nodes at level l_k which aren't in the tree after $(i-1)^{th}$ step

$$= \sum_{j=1}^{i-1} 2^{l_k - l_j}$$

Hence, number of nodes remaining at level l_k

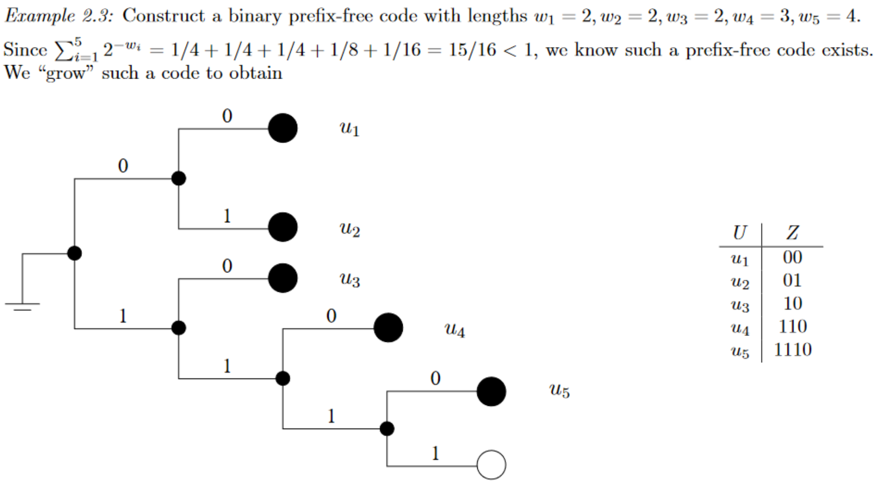
$$= 2^{l_k}(1 - \sum_{j=1}^{i-1} 2^{-l_j})$$

$\Rightarrow$ Number of nodes remaining at l_k in tree at after $(i-1)^{th}$ step > 1.

$\Rightarrow$ At least one survivor node should be present at level l_i also. So we can pick a node for the i^{th} step from level l_i also. This completes the proof.

Remark: We construct the tree from smallest length to largest length codewords. Contrast this with optimal source coding we shall see Hoffman codes later.

1.4 Example



Example 2.3: Construct a binary prefix-free code with lengths $w_1 = 2, w_2 = 2, w_3 = 2, w_4 = 3, w_5 = 4$.

Since $\sum_{i=1}^{5} 2^{-w_i} = 1/4 + 1/4 + 1/4 + 1/8 + 1/16 = 15/16 < 1$, we know such a prefix-free code exists. We "grow" such a code to obtain

U	Z
u_1	00
u_2	01
u_3	10
u_4	110
u_5	1110

1.5

Now, suppose that the source random variable $X \sim P_X$, we want to obtain a collection of integers $l_1, \cdots, l_k$ such that Kraft inequality is satisfied, then we know how to get the code for X.

$$\sum_{i=1}^{k} 2^{-l} \leq 1$$

- Suppose all codewords are of the same length,

  $$k2^{-l} \leq 1 \Rightarrow l \geq \log k$$

  Hence we can pick $l = \lceil \log k \rceil$ (ceil function). But there is no guarantee that this code is 'good', it may not have small average length.

- We know average length $= \sum p_i l_i$.

  Then we will choose small l_i for larger p_i. (while taking care that it satisfies Kraft inequality)

2 Shannon-Fano code

We fix,

$$l_i = \lceil \log_2 \frac{1}{p_i} \rceil \tag{3}$$

where p_i is the probability of X taking the i^{th} value in $\mathcal{X}$.
Clearly $l_i \geq 1$.

$$
\begin{aligned}
\sum_{i=1}^{k} 2^{-l_i} &= \sum_{i=1}^{k} 2^{-\lceil \log_2 \frac{1}{p_i} \rceil} =\leq \sum_{i=1}^{k} 2^{-\log_2 \frac{1}{p_i}} \\
&= \sum_{i=1}^{k} p_i = 1
\end{aligned}
$$

The lengths given by (3) satisfy Kraft inequality. We can use the tree-pruning algorithm (see sec 1.3) to get a prefix-free code for X.

This code obtained is called as the Shannon-Fano code.

Now,

$$
\begin{aligned}
L_{\text{Shannon-Fano}} &= \sum_{i=1}^{k} p_i \lceil \log_2 \frac{1}{p_i} \rceil \\
&< \sum_{i=1}^{k} p_i \left(\log_2 \frac{1}{p_i} + 1 \right) \\
&< H(X) + 1
\end{aligned}
$$

But S-F code is not always an optimal length prefix-free code.

2.1 Example for Shannon-Fano code

$$
X \in \mathcal{X} = \{x_1, x_2, x_3, x_4\}
$$

$$
P_X(x_i) = \begin{cases} 1/4, & \text{if } i = 1 \\ 1/2, & \text{if } i = 2 \\ 1/9, & \text{if } i = 3 \\ 5/36, & \text{if } i = 4 \end{cases}
$$

Lengths satisying the Kraft's inequality for the Shannon-Fano code

$$
l_i = \lceil \log_2 \frac{1}{P_X(x_i)} \rceil = \begin{cases} 2, & \text{if } i = 1 \\ 1, & \text{if } i = 2 \\ 4, & \text{if } i = 3 \\ 3, & \text{if } i = 4 \end{cases}
$$

$$
\bar{L} = \sum_{i=1}^{4} p_i l_i = \frac{57}{36}
$$

Obtaining the Shannon-Fano code:

- Arrange lengths in ascending order: $l_2 \leq l_1 \leq l_4 \leq l_3$. Hence we pick off a node at depth 2, then 1, 4 and at 3 and delete all the successors.

- Now we can choose the codewords from the binary tree, say

$$x_1 \to 01 \qquad x_2 \to 1 \qquad x_3 \to 0010 \qquad x_4 \to 000$$

Hence, the prefix-free Shannon-Fano code is $\{01, 1, 0010, 000\}$.

3 Huffman coding

We will show an optimal code construction that has smaller average length compared to the Shannon-Fano code.

3.1 Lemma's

Let us see some intuitive lemma's about an optimal code for a random variable X:

Assume that $X \in \mathcal{X} = \{x_1, \cdots, x_k\}$.
$P_X(x_i) = p_i, i = 1, \cdots, k$, without loss in generality, let $p_1 \geq \cdots \geq p_k$.

1. Consider that $l_1, \cdots, l_k$ are the lengths of the codewords associated to the messages $x_i, i = 1, \cdots, k$ respectively in any optimal code for X.

   Then,
   $$l_1 \leq \cdots \leq l_k \tag{4}$$

   Proof: Suppose $\mathscr{C}$ is an optimal code in which $\exists$ some distinct $i, j \in \{1, \cdots, k\}$ such that $p_i > p_j$ but $l_i > l_j$. (Assumption of the contrary)

   We shall show that there is another code $\mathscr{C}'$ which has smaller average length $L_{\mathscr{C}'}$ than $L_{\mathscr{C}}$.

   Cosider the code $\mathscr{C}'$ in which the codewords for x_i & x_j are swapped between those in $\mathscr{C}$.

   $$\underline{\mathrm{c}}_{\mathscr{C}'}(x_j) = \underline{\mathrm{c}}_{\mathscr{C}}(x_i) \qquad \underline{\mathrm{c}}_{\mathscr{C}'}(x_i) = \underline{\mathrm{c}}_{\mathscr{C}}(x_j)$$

   Now in $\mathscr{C}$,

   $$l_{i,\mathscr{C}} := \text{length of codeword of } x_i \text{ in } \mathscr{C} = l_j$$
   $$l_{i,\mathscr{C}'} = l_i$$

   So, $l_{i,\mathscr{C}'}$ is having the same codewords as $\mathscr{C}$ and hence is also prefix-free.

   $$\bar{L}_{\mathscr{C}'} = \sum_{k=\{1,\cdots,k\}\setminus\{i,j\}} p_k l_k + p_i l_i + p_j l_j$$

   $$\bar{L}_{\mathscr{C}} = \sum_{k=\{1,\cdots,k\}} p_k l_k$$

$$\bar{L}_{\mathscr{C}'} - \bar{L}_{\mathscr{C}} = p_i(l_j - l_i) + p_j(l_i - l_j) = (p_i - p_j)(l_j - l_i)$$

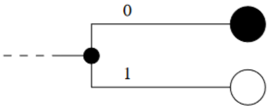
We have $p_i > p_j$, but $l_i > l_j$,

$$\Rightarrow \bar{L}_{\mathscr{C}'} - \bar{L}_{\mathscr{C}} < 0 \Rightarrow \bar{L}_{\mathscr{C}'} < \bar{L}_{\mathscr{C}}$$

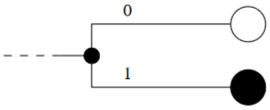
$\Rightarrow \mathscr{C}$ is not optimal, thus this a contradiction. (4) has to happen for an optimal code.

2. Consider the tree representation of a code which is optimal code. In such a tree, it should be true that either each node is a codeword or it has at least 2 successors which are codewords. i.e. There are no unused leaves in the tree of an optimal code.

   Proof: Let $\mathscr{C}$ be the optimal code. Let us consider the tree corresponding to this, suppose it has an unused leaf. Because the code is optimal, these unused leaves must be at maximum depth in the tree. Then, for at least one value x_i of $\mathscr{C}$, we have the situation

   or

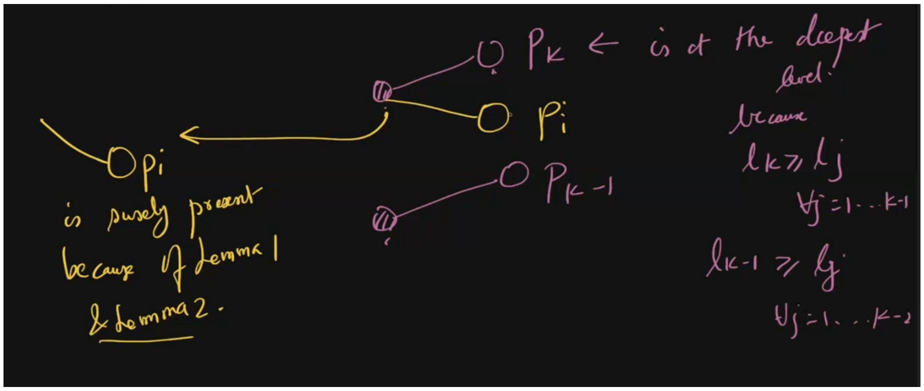
   In either case, we can delete the last digit of the codeword for x_i (without changing the other codewords) and still have a prefix-free code. But the new code has smaller $\bar{L}$ and thus the original code could not have been optimal.

3. There is an optimal prefix-free code for random variable X such that the codewords associated to the two smallest probability symbols are siblings (share the same parent). i.e. two least likely codewords differ only in their last digit.

   Proof: Let $\mathscr{C}$ be some optimal code for random variable. Suppose $\mathscr{C}$ already satifies the property specified above, then we are done.

   Suppose $\mathscr{C}$ doesn't satify the property, we know:

$$l_{k-1} \leq l_k \qquad as \qquad p_{k-1} \geq p_k$$



Note that $l_i = l_k$ and $i \neq k-1$.

$$\Rightarrow l_i \leq l_{k-1} \leq l_k = l_i \qquad \text{from Lemma-1}$$

$$\Rightarrow l_i = l_{k-1} = l_k$$

Interchange the codewords for the i^{th} and $(k-1)^{th}$ symbols and get a new code $\mathscr{C}'$.

$$\Rightarrow \bar{L}_{\mathscr{C}} = \bar{L}_{\mathscr{C}'}$$

Now $\mathscr{C}'$ satisfies the property in Lemma 3.

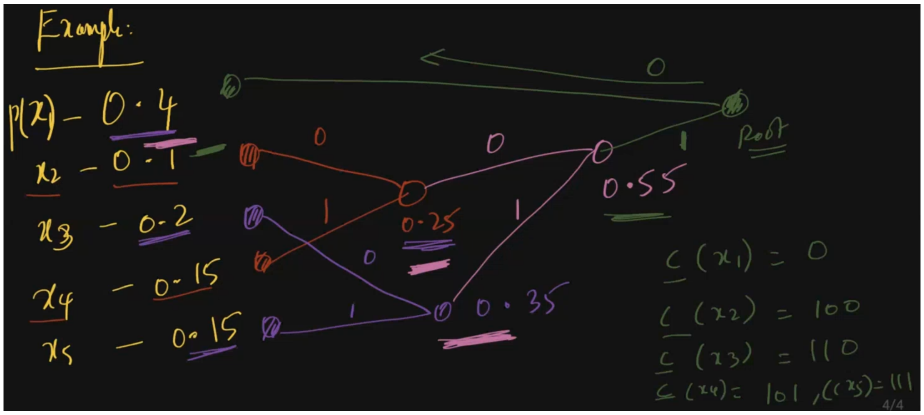
3.2 Huffman coding

Say we have p_X, assume that $p_1 \geq \cdots \geq p_k$. Unlike Shannon-Fano code, here we are building the tree in reverse. We pick p_{k-1} and p_k as they have the least 2 probabilities.

These are assigned to siblings in the tree and now the parent of them has probability $p_{k-1} + p_k$.

Now we can form a new distribution using these $k-1$ probabilities (symbols) i.e. $p_1, \cdots, p_{k-2}, p_{k-1} + p_k$. Now, if $\{p_{k-2}, p_{k-1} + p_k\}$ are the least 2 probabilities, a new node is assigned to p_{k-2}. We repeat the process until all codewords are assigned.

3.3 Example


Example:
p(X) - 0.4
x2 - 0.1
x3 - 0.2
x4 - 0.15
x5 - 0.15
0
1
0
1
0
1
0
1
0.25
0.35
0.55
Root
c(x1) = 0
c(x2) = 100
c(x3) = 110
c(x4) = 101, c(x5) = 111
4/4