EE276 Information Theory Lecture 4 - 01/18/2024

Lecture 4: Variable Length Lossless Compression

Lecturer: Tsachy Weissman

In this lecture we investigate strategies for variable length lossless compression. Several variable length encoding schemes are shown and their merits are discussed. Then a procedure for producing prefix codes for a dyadic source is presented. Lastly, Shannon Codes are presented as a generalization of the dyadic encoding procedure.

1 Review From Previous Class

$$U_1, U_2, ...U_n \sim \text{iid } U \in \mathcal{U} \tag{1}$$

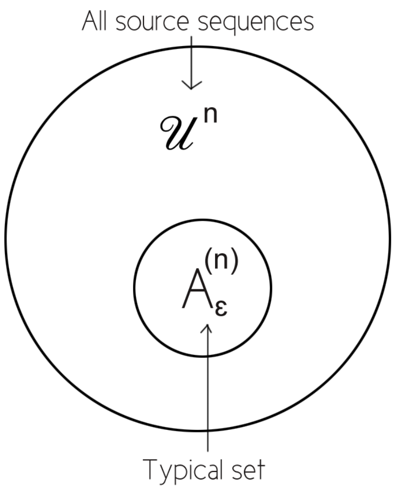


Figure 1: Asymptotic Equipartition Property

$$P(U^n \in A_\epsilon^{(n)}) \approx 1 \tag{2}$$
$$\forall u^n \in A_\epsilon^{(n)} \ : \ p(u^n) \approx 2^{-nH(U)} \tag{3}$$
$$|A_\epsilon^{(n)}| \approx 2^{nH(U)} \tag{4}$$

This tells us that we need $H(U)$ bits per source symbol for a near-lossless fixed length scheme. If fewer bits are used (say $\alpha < H(U)$ bits), then the size of the set encoded by those α bits is given by

$$|B_n| = 2^{n\alpha}.$$

2 Near-lossless (fixed-length/block) compression

In this section, we consider the problem of developing a *scheme* that allows us to encode and decode a source sequence $U^N = (U_1, \ldots, U_N)$ iid $\sim U$.

$$\begin{tikzcd}
U^n = (U_1, \ldots, U_n) \arrow[r] & \boxed{\begin{array}{c}\text{compressor}\\\text{(encoder)}\end{array}} \arrow[r, "m \text{ bits}"] & \boxed{\begin{array}{c}\text{decompressor}\\\text{(decoder)}\end{array}} \arrow[r] & \hat{U}^n
\end{tikzcd}$$

2.1 Notation

We briefly describing some of the relevant terms and notations used in this section

1. **Probability of error:** $P_e = P(\hat{U}^N \neq U^N)$ denotes the probability of error.
2. **Rate:** The rate of a scheme is the average number of bits it uses to encode source symbols.
3. **"Near lossless"** indicates that $P_e \ll 1$.
4. **"Fixed length"** or **"block"** indicates that source symbols are encoded with a fixed number of bits.
5. **"Scheme"** is a compressor (encoder) and its corresponding decompressor (decoder).

2.2 Encoder & decoder design

Theorem 1 (Direct theorem)**.** *$\forall R > H(U)$ and $\forall \delta > 0$, $\exists$ a large enough N and a scheme with rate $r < R$ and $P_e < \delta$.*

Proof Fix $\epsilon > 0$ such that $H(U) + \epsilon < R$ and enumerate the elements in $A_\epsilon^{(N)}$, where $\mathcal{I}(u^N)$ indicates index of $u^N \in A_\epsilon^{(N)}$. Then use the following encoding approach

$$\begin{cases} \text{output a m-bit representation of } \mathcal{I}(u^N), & \text{if } u^N \in A_\epsilon^{(N)} \\ \text{output an arbitrary sequence of bits,} & \text{if } u^N \notin A_\epsilon^{(N)}. \end{cases}$$

For decoding, simply let $\hat{U}^N$ be the sequence in $A_\epsilon^{(N)}$ whose index corresponds to the m received bits. Note that this scheme makes an error whenever u^N is not in $A_\epsilon^{(N)}$. Such a scheme requires $m = \log \left|A_\epsilon^{(N)}\right|$ bits per source symbol, which achieves the following rate and probability of error

$$\begin{gathered} r = \frac{m}{N} = \frac{\log \left|A_\epsilon^{(N)}\right|}{N} \leq \frac{N(H(U)+\epsilon)}{N} = H(U) + \epsilon < R \\ P_e = P(U^N \notin A_\epsilon^{(N)}) < \delta \text{ for } N \text{ sufficiently large.} \end{gathered}$$

□

Theorem 2 (Converse theorem)**.** *If $R < H(U)$ then for all sequences of schemes with rate $r \leq R$, $P_e \xrightarrow{n\to\infty} 1$.*

Proof A scheme with rate r can at most represent 2^{Nr} different sequences without error. Let's define $B^{(N)}$ as the set of these sequences. Because $r \leq R < H(u)$, $\exists \epsilon > 0$ *s.t.* $r + \epsilon < H(U)$.

$$\begin{align} P(U^N \in B^{(N)}) = P(U^N \in B^{(N)} \cap A_\epsilon^{(N)}) + P(U^N \in B^{(N)} \cap (A_\epsilon^{(N)})^c) \tag{5} \\ \tag{6} \\ \text{By Theorem 4 we know:} \tag{7} \\ \tag{8} \\ P_e = 1 - P(U^N \in B^{(N)}) \xrightarrow{N\to\infty} 1 \tag{9} \end{align}$$

□

Note: Drawbacks of this framework:

1. Nonzero error probability (which can be resolved by using a variable length scheme).
2. Enumerating an exponentially large set and then searching through the resulting list is incredibly inefficient.
3. Large block length N introduces significant delay in encoding and decoding.

A simple variable length scheme to solve the first problem above is to first send 1 bit to indicate whether the sequence is typical. If typical, we use the above encoding scheme. In the unlikely event that the sequence is not typical, we can just send the N bits without encoding.

3 Variable Length Lossless Compression Examples

3.1 Example 1

Given an alphabet with four letters in it

$$\mathcal{U} = \{a, b, c, d\}$$

The probability of each letter and a coding scheme is given in the table below:

u	$p(u)$	codeword $c(u)$	length $l(u)$
a	$1/2 = 2^{-1}$	0	1
b	$1/4 = 2^{-2}$	10	2
c	$1/8 = 2^{-3}$	110	3
d	$1/8 = 2^{-3}$	111	3

The code is $\{c(u)\}_{u\in\mathcal{U}}$. Note that any code can be represented by the nodes in a binary tree. For example, a binary tree representation of this code is shown in figure 2.

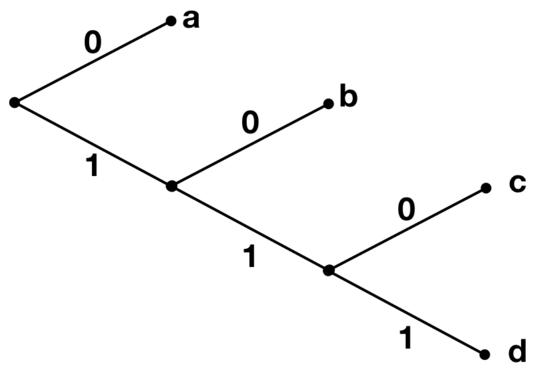


Figure 2: Binary tree representation

Note: For this code, $l(u) = \log \frac{1}{p(u)}$, therefore

$$\bar{l} = \mathbb{E}[l(U)] = H(U) \tag{10}$$

Thus, this is a lossless scheme that achieves the ideal bits per source symbol, the entropy $H(U)$ (we'll show in the next lecture that the entropy is fundamental limit of compression for variable length lossless schemes as well). Additionally, it is a prefix code which makes the encoding and decoding process very simple. Note that a code is a prefix code if and only if all the codewords are leaves in the binary tree representation of the code.

3.2 Example 2

Consider a new scheme for the same source (binary tree representation shown in figure 3).

U	codeword $c(U)$	length $l(u)$
a	0	1
b	1	1
c	01	2
d	11	3

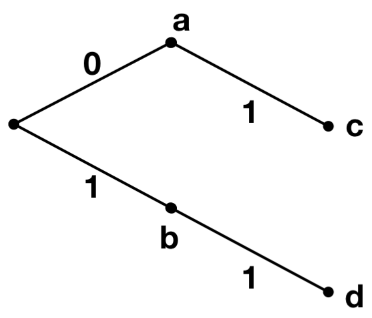


Figure 3: The binary tree representation of the code

Note: The length of the code is $\bar{l} < H(U)$, i.e. better than the entropy. But, consider the encoding of three source symbols $a\ b\ d$ and $c\ b\ b$:

$$a\ b\ d \overset{\text{encoded to}}{\longrightarrow} 0111 \tag{11}$$

$$c\ b\ b \overset{\text{encoded to}}{\longrightarrow} 0111 \tag{12}$$

Thus, different source sequences can have the same encoding, and this code cannot be decoded uniquely.

Definition 3. *A code is* **uniquely decodable** *(UD) if every sequence of source symbols is mapped to a distinct binary representation.*

Definition 4. *A* **prefix code** *is a code where no codeword is the prefix of any other.*

Note: A prefix code is uniquely decodable and can be decoded efficiently and on the fly (i.e. without needing entire binary sequence before decoding starts)

3.3 Exercise

Consider the code for the same source

U	codeword $c(U)$	length $l(u)$
a	10	2
b	00	2
c	11	2
d	110	3

Show that, though this code is not a prefix code, it is actually uniquely decodable. For the purpose of this exercise, you don't need to know the source distribution.

Proof
Part of HW 2.

$\square$

4 Prefix code for dyadic distributions

Definition 5. *A source is* **dyadic** *if*

$$p(u) = 2^{-n_u} \tag{13}$$

where n_u is an integer $\forall u \in \mathcal{U}$.

Suppose we find a code such that

$$l(u) = n_u = \log \frac{1}{p(u)} \tag{14}$$

then

$$\bar{l} = \mathbb{E}[l(U)] = H(U) \tag{15}$$

Before proceeding to a prefix code for dyadic sources, we prove a technical lemma which says that the number of symbols with the lowest probability is even for a dyadic source. In particular, a dyadic source cannot have a single symbol with the lowest probability.

Lemma 6. *Assume $\mathcal{U}$ is dyadic with $|\mathcal{U}| \geq 2$, and let $n_{\max} = \max_{u \in \mathcal{U}} n_u$. Then the number of symbols $u \in \mathcal{U}$ with $n_u = n_{\max}$ is even.*

Proof:

$$\begin{aligned} 1 &= \sum_u p(u) && (16)\\ &= \sum_u 2^{-n_u} && (17)\\ &= \sum_{n=1}^{n_{\max}} \begin{pmatrix} \text{\# of symbols } u \\ \text{with } n_u = n \end{pmatrix} 2^{-n} && (18)\\ & && (19) \end{aligned}$$

Multiplying both sides by $2^{n_{\max}}$,

$$\begin{aligned} \underbrace{2^{n_{\max}}}_{\text{even}} &= \sum_{n=1}^{n_{\max}} \begin{pmatrix} \text{\# of symbols } u \\ \text{with } n_u = n \end{pmatrix} \cdot 2^{n_{\max}-n} && (20)\\ &= \underbrace{\sum_{n=1}^{n_{\max}-1} \begin{pmatrix} \text{\# of symbols } u \\ \text{with } n_u = n \end{pmatrix} \cdot \underbrace{2^{n_{\max}-n}}_{\text{even}}}_{\text{even}} + \begin{pmatrix} \text{\# of symbols } u \\ \text{with } n_u = n_{\max} \end{pmatrix} \cdot 1 && (21) \end{aligned}$$

Therefore, the number of letters u with $n_u = n_{\max}$ must be even. ■

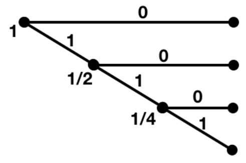
4.1 Procedure for constructing prefix codes

Consider the following procedure:

- Choose 2 symbols with $n_u = n_{\max}$ and merge them into a single symbol.
- We now have a new symbol with twice the probability.
- The new source is also dyadic.
- So repeat the first step until we are left with just one symbol.

Note: This procedure induces a binary tree and the codewords are the leaves of the constructed binary tree. As noted earlier, a code with all codewords on the leaf nodes is a prefix code.

4.2 Example of procedure



$p(U)$	U
1/2	a
1/4	b
1/8	c
1/8	d

Figure 4: Procedure for generating a prefix code

Note: If $p(u) = 2^{-n_u}$, then the node u will participate in n_u merges (since each merge doubles the probability). Thus, the distance of u from the root is n_u, which is also equal to $l(u)$.
Conclusion: Our procedure yields a prefix code with

$$l(u) = n_u = \log \frac{1}{p(u)} \tag{22}$$

and, in particular, $\bar{l} = H(U)$.

5 Shannon Codes

For a general source let

$$n_u^* = \lceil \log \frac{1}{p(u)} \rceil \quad \forall u \in \mathcal{U} \tag{23}$$

Note:

$$\sum_{u \in \mathcal{U}} 2^{-n_u^*} = \sum_{u \in \mathcal{U}} 2^{-\lceil \log \frac{1}{p(u)} \rceil} \tag{24}$$

$$\leq \sum_{u \in \mathcal{U}} 2^{-\log \frac{1}{p(u)}} \tag{25}$$

$$= \sum_{u \in \mathcal{U}} p(u) = 1 \tag{26}$$

Consider a new source $p^*(u) = 2^{-n_u^*}$. While $p^*(u)$ does not sum to 1 over $\mathcal{U}$, we can add new symbols to extend the source to $\mathcal{U}^* \supseteq \mathcal{U}$ such that $p^*(u)$ is dyadic over $\mathcal{U}^*$ and $\sum_{u \in \mathcal{U}^*} p^*(u) = 1$.
Using the technique in the previous section, we can now construct a prefix code for the source p^* such that

$$l(u) = \log \frac{1}{p^*(u)} = n_u^* = \lceil \log \frac{1}{p(u)} \rceil \quad \forall u \in \mathcal{U} \tag{27}$$

Note that the code is defined over $\mathcal{U}^*$ but we only care about the symbols in $\mathcal{U}$.
Note: This code is known as the *Shannon Code*.
The expected length for this code is

$$\begin{aligned}
\bar{l} &= \sum_u p(u) l(u) && (28)\\
&= \sum_u p(u) \lceil \log \frac{1}{p(u)} \rceil && (29)\\
&\leq \sum_u p(u) \left(\log \frac{1}{p(u)} + 1 \right) && (30)\\
&= H(U) + 1 && (31)
\end{aligned}$$

So we are within 1 bit per source symbol of the entropy. To get even closer to the entropy, we can work with blocks of source symbols.

For the memoryless source $U_1, U_2, \ldots$ iid $\sim \mathcal{U}$, we can construct a Shannon code for $U^N = (U_1, U_1, \ldots, U_N)$. Then

$$\frac{1}{N}\mathbb{E}[l(U^N)] \leq \frac{1}{N}\left(H(U^N) + 1\right) = H(U) + \frac{1}{N} \tag{32}$$

Thus as N becomes large, the bits used per source symbol tends toward the entropy.