

Chapter 2

Lossless Coding

2.1 Quantization and lossless coding

Recall the basic model of a quantizer as depicted in Figure 2.1. The com-

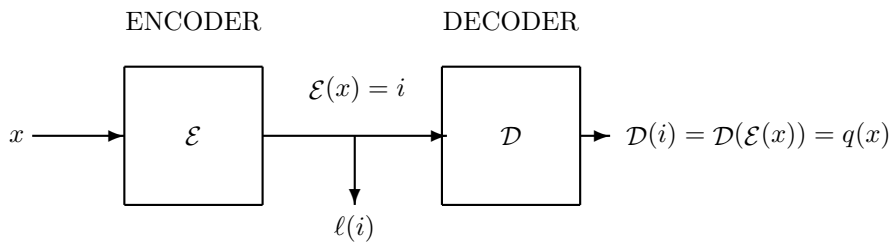


Figure 2.1: Quantizer

ponents of the quantizer are

- An *encoder* \mathcal{E} which maps the input space A into an *index set* or *index codebook* \mathcal{I} . So far we have assumed that the index set is the set of all nonnegative integers $\mathcal{Z} = \{0, 1, 2, \dots\}$ or some subset of this space.
- A *decoder* \mathcal{D} which maps an index set \mathcal{I} into the output space \hat{A} .
- A length function $\ell(i)$ which is *admissible* in the sense that it satisfies the Kraft inequality

$$\sum_{i \in \mathcal{I}} e^{-\ell(i)} \leq 1.$$

In this section we consider the special case where the input alphabet is discrete, where the distortion measure has the property that $d(x, y) > 0$ if $x \neq y$, and where we require that the average distortion be $D(q) = 0$. This

is the case of *noiseless* or *lossless* compression and the goal is to achieve lossless or zero distortion reproduction at the minimum possible average rate. This special case provides an interpretation of the meaning of the length function (or, equivalently, of the reproduction codebook weighting) in the general case and hence lossless coding plays an important role in the general lossy quantization case. This connection is explored next before tackling the specifics of the purely lossless coding case.

2.2 Equivalent quantizers

The overall operation of a quantizer q depends only on the cascade of encoder and decoder, not on their individual structure. In particular, the index set \mathcal{I} could be relabeled in any way preserving a unique label for each member of the index set. Equivalently, the overall quantization operation would not be changed if an invertible mapping first mapped the index onto and then out of another discrete space \mathcal{W} as depicted in Figure 2.2. Two models q and \bar{q} for a quantizer are said to be *equivalent* if the overall operation is the same for all inputs in the sense that $\bar{q}(x) = q(x)$ for all $x \in A$. The following simple result formally states this fact.

Lemma 2.1 *Given a quantizer $q = (\mathcal{E}, \mathcal{D})$ and a discrete set \mathcal{W} for which $\psi : \mathcal{E}(A) \rightarrow \mathcal{W}$ is an invertible mapping onto \mathcal{W} , i.e., \mathcal{W} is the range space of ψ and there is an inverse mapping $\psi^{-1} : \mathcal{W} \rightarrow \mathcal{E}(A)$ satisfying $\psi^{-1}(\psi(i)) = i$ for all $i \in \mathcal{E}(A)$. Then an equivalent model $\bar{q} = (\bar{\mathcal{E}}, \bar{\mathcal{D}})$ with index set $\bar{\mathcal{I}} = \mathcal{W}$ for the quantizer q consists of an encoder $\bar{\mathcal{E}} : A \rightarrow \mathcal{W}$ defined by $\bar{\mathcal{E}}(x) = \psi(\mathcal{E}(x))$ and a decoder $\bar{\mathcal{D}} : \mathcal{W} \rightarrow \hat{A}$ defined by $\bar{\mathcal{D}}(w) = \mathcal{D}(\psi^{-1}(w))$ so that*

$$\bar{q}(x) = \mathcal{D}(\bar{\mathcal{E}}(x)) = \bar{\mathcal{D}}(\bar{\mathcal{E}}(x)) = q(x).$$

The result justifies the use of the nonnegative integers \mathcal{Z} as a canonical index set since for any other discrete set there is an equivalent quantizer in terms of the overall input/output mapping which uses \mathcal{Z} as the encoder output/decoder input alphabet. While such equivalent models have no effect on the overall input/output relations, they will become important for actual implementations of quantizers.

The invertible mapping $\psi : \mathcal{I} \rightarrow \mathcal{W}$ is called a *lossless coding* or *noiseless coding* of \mathcal{I} onto \mathcal{W} . The coupling of a neutral index set such as the integers with a lossless coding has a practical implication. Suppose each encoded input $\mathcal{E}(x) = i \in \mathcal{Z}$ is to be transmitted through or stored in a medium that accepts and delivers one or more symbols drawn from an m -ary alphabet,

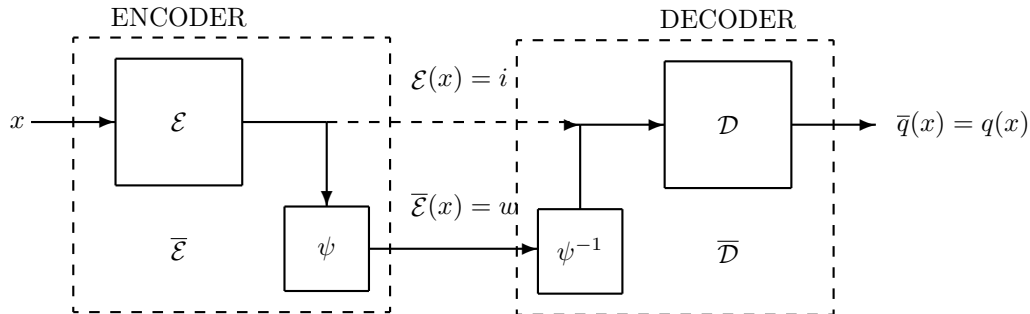


Figure 2.2: Equivalent quantizer

say $\mathcal{Z}_m = \{0, 1, \dots, m-1\}$. In other words, the original index i is mapped by an invertible function $\psi : \mathcal{I} \rightarrow \mathcal{W} \subset \mathcal{Z}_m^*$, where \mathcal{Z}_m^* is the space of all finite length sequences of symbols drawn from \mathcal{Z}_m . The resulting m -ary sequence $w = \psi(i) \in \mathcal{W} = \psi(\mathcal{I})$ is called a *channel codeword*. The decoder then applies the inverse mapping $\psi^{-1} : \psi(\mathcal{I}) \rightarrow \mathcal{I}$ to reproduce the index i . As in Lemma 2.1, this yields an equivalent quantizer in terms of overall operation. It also provides a natural notion of the cost or instantaneous rate or length function of an index — the length $\text{length}(\psi(i))$ of the symbol sequence $\psi(i)$. For example, if $m = 2$ (by far the most common case), then $\text{length}(\psi(i))$ is the number of bits required to be transmitted or stored to uniquely specify the index selected by the encoder to the decoder. We shall concentrate on the binary case, but keep in mind that ternary, quaternary, or other alphabets are also of interest in some problems and the ideas all generalize in a straightforward manner. For simplicity, however, $m = 2$ will be assumed in most of the development and proofs. Since the length depends on the choice of m , the length function in units appropriate for an m -ary channel symbol alphabet will be denoted by

$$\ell_m(i) = \text{length}(\psi(i)).$$

The general length function introduced earlier normalizes the units to nats, that is,

$$\ell(i) = \ell_m(i) \ln m$$

so that $m^{\ell_m(i)} = e^{\ell(i)}$ gives the number of possible m -ary $\ell_m(i)$ tuples or possible channel codewords of $\ell_m(i)$ symbols. The subscript m in ℓ_m will be dropped for reasons of notational simplicity when it is clear from context since the different choice of units for the length function will be obvious.

The use of natural units allows the development of many ideas in a way that does not depend on the particular implementation of the index code. If an interpretation in terms of bits, say, is desired, it is simply a matter of changing the units as above. Natural logarithms are often useful when doing theory. We shall focus on either bits for intuition, or nats for uniformity and theory.

Consider for example the three binary codings of Table 2.1 where \mathcal{I} is \mathcal{Z}_{16} . All three of the binary codes are invertible, so knowing the binary

\mathcal{I}	\mathcal{W}_0	\mathcal{W}_1	\mathcal{W}_2
0	0000	0	0
1	0001	1	10
2	0010	10	110
3	0011	11	1110
4	0100	100	11110
5	0101	101	111110
6	0110	110	1111110
7	0111	111	11111110
8	1000	1000	111111110
9	1001	1001	1111111110
10	1010	1010	11111111110
11	1011	1011	111111111110
12	1100	1100	1111111111110
13	1101	1101	11111111111110
14	1110	1110	111111111111110
15	1111	1111	111111111111111

Table 2.1: Index codes

codes uniquely determines the index. The first code is reminiscent of ASCII because it assigns a fixed-length binary vector to each possible input and hence is called a *fixed-length code* or *fixed-rate code* because all of the channel codewords have the same length and hence the average length or average cost will be the fixed length value, 4. The other two codes both have codewords of varying length, the second formed simply by removing leading zeros and the third just uses one of the symbols and then terminates all but the last codeword with a 0. Note that any permutation of the set of channel codewords also provides an invertible lossless code.

In the fixed-length code case, the instantaneous rate is simply $\text{length}(\psi(i)) = \log_2 N(\mathcal{I})$ for all $i \in \mathcal{I}$ regardless of the detailed structure of ψ so that the

average index cost is

$$R(\mathcal{E}) = E[\ell(\mathcal{E}(X))] = \log_2 N \text{ bits.}$$

In this case the instantaneous and average rate can be viewed as a transmission or storage cost in terms of the average number of bits required to describe the index, it can also be viewed as a cost of the *memory* required by the decoder codebook since it counts the number of members of the reproduction codebook. For example, if the decoder codebook contained N reproduction codewords corresponding to input k -dimensional vector inputs, then the decoder would need Nk floating point storage slots. Fixed-rate codes only make sense for a finite index set, but variable-length codes in theory can have an infinite number of codewords.

Instead of effectively penalizing memory by imposing the constraint of a fixed-length code, or penalizing transmission rate by according to the number of bits required to specify the index with variable-length channel codewords, one can combine the two penalties into a single cost function

$$r_{\mathcal{I}}(i) = (1 - \eta) \text{length}(\psi(i)) + \eta \log N(\mathcal{I}) \quad (2.1)$$

where $\eta \in [0, 1]$ can be thought of as a Lagrangian multiplier weighing the relative importance of transmission rate and memory. The classic cases of purely fixed or purely variable rate are then special cases with $\eta = 1$ or 0 , respectively. The combined constraint cost function allows one to formalize optimization problems that force the number of codewords to be finite if a finite average cost is required. When dealing with the pure lossless coding case with variable rate codes allowed rather than a general lossy quantizer which includes a lossless code, it is easy to see that N must be the size of the input alphabet, so $\eta = 0$ is appropriate.

2.3 Uniquely decodable codes

In practice a quantizer is not used only once, it is applied to a sequence of inputs to produce a sequence of outputs. This means that a lossless index code must not only be invertible, it must have the property that sequences of channel codewords are also invertible, that is, if for every positive integer K and every sequence of indices $i^K = (i_0, i_1, \dots, i_{K-1})$, the sequence of channel codewords $\psi(i^K) = (\psi(i_0), \psi(i_1), \dots, \psi(i_{K-1}))$ uniquely determines i^K . A lossless code with this property is said to be *uniquely decodable*. The terms *uniquely decipherable* and *instantaneous code* are also used.

Consider the examples of Table 2.1. The fixed-length code is clearly uniquely decodable if the starting point is known, each successive four symbol word is a channel codeword with a unique inverse. The first variable-length code has a problem, however. Suppose, for example, that the quantizer encoder produces a sequence of integers 1 0 3 which the first binary code turns into 1 0 11, but “space” is not a symbol in the binary alphabet, so the actual binary string transmitted to the decoder is 1011. But seeing these 4 binary symbols, the decoder cannot uniquely decide the sequence of indices sent, that is, was it the correct 1 0 3 or was it 2 3 or was it 11? The second variable-rate code does not have this problem. No codeword is a prefix of another codeword, so any sequence of codewords can be uniquely decoded.

A remarkable result due to McMillan and later simplified by Karush provides a simple necessary condition for the existence of a uniquely decodable code given a set of positive integers $\{\ell(i); i \in \mathcal{I}\}$ describing the allowed lengths of the channel codewords in m -ary symbols.

Lemma 2.2 *A necessary condition for the existence of an m -ary uniquely decodable code having a collection of codeword lengths in m -ary units $\{\ell(i); i \in \mathcal{I}\}$ is that the lengths satisfy the inequality*

$$\sum_{i \in \mathcal{I}} m^{-\ell(i)} \leq 1. \quad (2.2)$$

The inequality is variously known as Kraft’s inequality, the Kraft inequality, and the Kraft-McMillan equality. Kraft’s contribution will be considered later in an even more remarkable result that demonstrates that (2.2) is also a sufficient condition for the existence of a uniquely decodable code with the given lengths.

The proof is for $m = 2$, the proof for other positive integers follows in the same way with added bookkeeping.

Proof Suppose that $\{\ell(i); i \in \mathcal{I}\}$ is a set of lengths in bits of channel codewords of a uniquely decodable lossless code with binary channel symbols. Without loss of generality, assume that the the sequence $\ell(i)$ is nondecreasing in i . Recall that \mathcal{I} can be finite or infinite in size, so the proof allows the possibility of an infinite number of codewords. If the code with lengths $\{\ell(i); i \in \mathcal{I}\}$ is uniquely decodable, then so also must be the possibly smaller codebook with lengths $\{\ell(i); i \in \mathcal{I}, \ell(i) \leq L\}$, albeit the smaller code will be a uniquely decodable code for smaller subset of \mathcal{I} , say $\mathcal{I}^{(L)}$. Thus for any positive integer K , a sequence of indices $i = (i_0, \dots, i_{K-1})$, $i_n \in \mathcal{I}^{(L)}$, must produce a sequence of binary channel codewords with length

ℓ_{i_n} ; $n = 0, 1, \dots, K-1$. The total length of the resulting sequence of channel codewords will be

$$\text{length}(\psi(i)) = \sum_{n=0}^{K-1} \ell_{i_n} = \sum_{\ell=1}^L N(K, \ell)\ell,$$

where $N(K, \ell)$ is the number of input sequences $i = (i_0, \dots, i_{K-1})$, $i_n \in \mathcal{I}^{(L)}$ which yield a channel sequence of total length ℓ . Since the code is uniquely decodable, $N(K, \ell)$ can be no greater than 2^ℓ , the total number of length ℓ binary sequences or there would be more indices requiring unique length ℓ sequences than the total number of distinct length ℓ sequences. Thus

$$N(K, \ell) \leq 2^\ell \quad (2.3)$$

The proof of the lemma combines this inequality with a simple trick. To bound the sum

$$S_L = \sum_{i: \ell(i) \leq L} 2^{-\ell(i)}.$$

consider instead the power

$$\begin{aligned} S_L^K &= \left(\sum_{i: \ell(i) \leq L} 2^{-\ell(i)} \right)^K \\ &= \left(\sum_{i_0: \ell_{i_0} \leq L} 2^{-\ell_{i_0}} \right) \times \left(\sum_{i_1: \ell_{i_1} \leq L} 2^{-\ell_{i_1}} \right) \times \dots \times \left(\sum_{i_{K-1}: \ell_{i_{K-1}} \leq L} 2^{-\ell_{i_{K-1}}} \right) \\ &= \sum_{i_0: \ell_{i_0} \leq L} \sum_{i_1: \ell_{i_1} \leq L} \dots \sum_{i_{K-1}: \ell_{i_{K-1}} \leq L} 2^{-\sum_{n=0}^{K-1} \ell_{i_n}} \\ &= \sum_{\ell=1}^L N(K, \ell) 2^{-\ell}. \end{aligned}$$

Applying the inequality (2.3) then yields $S_L^K \leq L$ or $S_L \leq L^{1/K}$. For a fixed L , this must hold for all K and hence it must also hold in the limit as $K \rightarrow \infty$, in which case the right hand side is 1. Thus for every fixed finite L

$$S_L = \sum_{i: \ell(i) \leq L} 2^{-\ell(i)} \leq 1.$$

This inequality holds for all L and hence it must also hold true in the limit, which proves the lemma. \square

The lemma shows that a necessary condition for a set of lengths to correspond to a lossless binary index code is that the lengths satisfy Kraft's inequality. This means that optimizing over lossless codes, only codes with lengths satisfying Kraft's inequality need be considered. Next it will be seen that the condition is also sufficient.

2.4 Tree-structured codes

In general a simple and effective way to construct a uniquely decodable code is to construct a code with this prefix-free property. Although not all uniquely decodable codes have this property, we shall see that no lossless code which lacks the property can outperform the best code with the property. Prefix-free codes or, more simply, prefix codes, or tree-structured codes can be simply visualized as in Figure 2.3 as requiring the channel codebook to have a tree structure. In the figure, the tree has a depth of 4 so the maximum length available is 4. Any sequence of binary symbols labeling branches from the root node to a leaf or terminal (called a *path map*) node provides a channel codeword. Since no descendent node of a leaf can be a codeword, no codeword can be a prefix of another codeword. Conversely, any prefix code can be depicted as a binary tree of this form.

The remarkable thing about a prefix or tree-structured code is that there exists a simple necessary and sufficient condition for the existence of such a code for a given collection of channel codeword lengths $\ell(i)$, $i \in \mathcal{I}$.

Lemma 2.3 *There exists an m -ary tree-structured channel code (prefix code) having a collection of codeword lengths $\{\ell(i); i \in \mathcal{I}\}$ if and only if the Kraft inequality (2.2) is satisfied.*

Proof The proof is for $m = 2$, the proof for other positive integers follows in the same way with added bookkeeping.

Necessity follows from Lemma 2.2, but it is instructive to provide a proof specifically aimed at the tree structure. First suppose that a binary tree exists with lengths $\ell(i)$; $i \in \mathcal{I}$ from root to leaves. Without loss of generality, assume that the $\ell(i)$ are in nondecreasing order: $\ell(0) \leq \ell(1) \leq \ell(2) \cdots$. Let L be a fixed positive integer. Consider the codeword with length $\ell(0)$. Since it is a codeword, all of the descendent nodes are removed from the tree. If these nodes had remained, there would have been at depth L in the tree $2^{L-\ell(0)}$ descendent nodes. This is depicted in Figure 2.4 where the leaves are denoted by open circles, but the complete tree has been filled in to depth $L = 4$ so the would-be descendants can be seen. The shortest codeword has

length $\ell(0) = 1$, so there would have been $2^{4-1} = 8$ descendants at level 4 if this node had not been made a terminal node. The same is true for the remaining lengths up to and including lengths for which $\ell(i) = L$, that is, the leaf with length $\ell(i)$ for $\ell(i) \leq L$ would have had $2^{L-\ell(i)}$ descendants at depth L if the tree had been allowed to grow. Since the collection of these descendants at depth L cannot contain more nodes than the total 2^L of the full tree at length L ,

$$\sum_{i:\ell(i)\leq L} 2^{L-\ell(i)} \leq 2^L$$

or

$$\sum_{i:\ell(i)\leq L} 2^{-\ell(i)} \leq 1. \quad (2.4)$$

The left hand side of (2.4) is nondecreasing as L grows, so it must have a limit which is also less than 1, which proves (2.2).

The same figure illustrates the proof for sufficiency. Given a set of ordered lengths $\ell(i); i \in \mathcal{I}$, pick an arbitrary node at depth $\ell(0)$ into the tree and declare it a leaf and make the binary pathmap to the node the channel codeword. For any depth $L \geq \ell(0)$ this removes $2^{L-\ell(0)}$ nodes at depth L that might otherwise have been available as terminal nodes and hence other codewords. In particular, at depth $\ell(1)$ in the tree $2^{\ell(1)-\ell(0)}$ nodes have been removed. Provided this has not removed all of the possible $2^{\ell(1)}$ nodes at depth $\ell(1)$, that is, provided $2^{\ell(1)-\ell(0)} < 2^{\ell(1)}$ or $2^{-\ell(0)} < 1$, which is ensured by the Kraft inequality. Continue in this manner. Suppose that words have been picked of lengths $\ell(0), \ell(1), \ell(N-1)$ and we wish to choose another of length $\ell(N)$, and the Kraft inequality holds for $\{\ell(0), \ell(1), \ell(N-1), \ell(N)\}$. Then the existing codewords together cause the loss of $\sum_{i=0}^{N-1} 2^{\ell(N)-\ell(i)}$ nodes in the tree of depth $\ell(N)$, but at least one will remain if this total is strictly less than $2^{\ell(N)}$, that is, if

$$\sum_{i=0}^{N-1} 2^{\ell(N)-\ell(i)} < 2^{\ell(N)}$$

or

$$\sum_{i=0}^{N-1} 2^{-\ell(i)} < 1,$$

which will be the case since

$$\sum_{i=0}^N 2^{-\ell(i)} \leq 1.$$

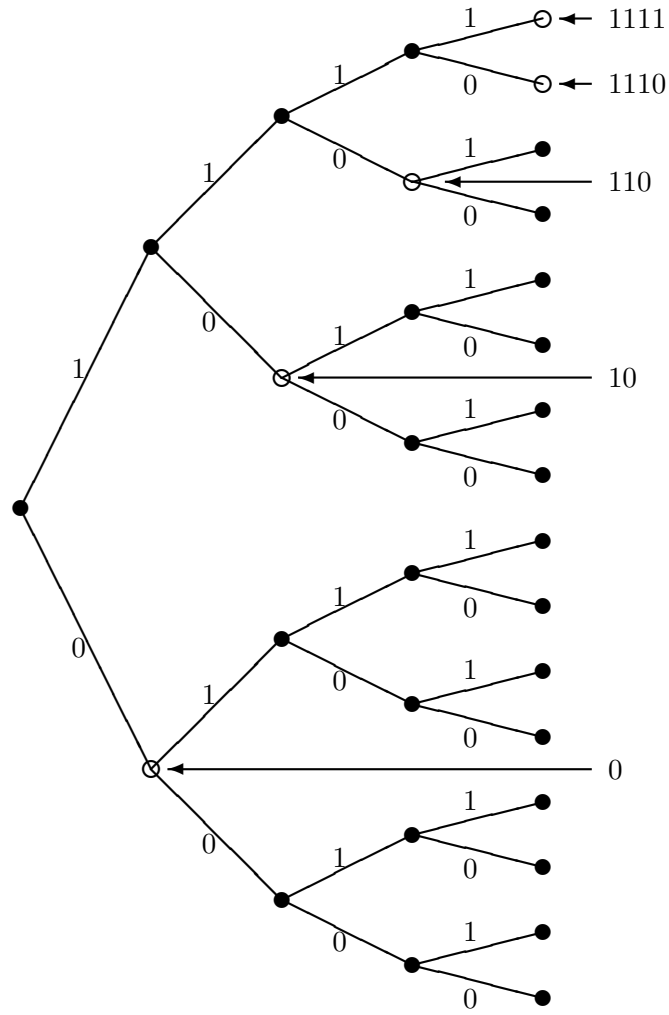


Figure 2.4: Counting nodes

If \mathcal{I} is finite, this will terminate at some point. If it is infinite, there will always remain at least one open leaf at the next $\ell(N)$. \square

A length function $\ell = \{\ell(i); i \in \mathcal{I}\}$ in m -ary units will be said to be m -admissible if it satisfies Kraft's inequality

$$\sum_{i \in \mathcal{I}} m^{-\ell(i)} \leq 1. \quad (2.5)$$

Alternatively, the length function ℓ can be expressed in nats and Kraft's inequality can be restated as requiring that ℓ satisfy

$$\sum_{i \in \mathcal{I}} e^{-\ell(i)} \leq 1 \quad (2.6)$$

and that $\ell(i)/\ln m = (\log_m e)\ell(i)$ is an integer for all $i \in \mathcal{I}$. Thus (2.6) is a *necessary* condition for the existence of a uniquely decodable code, but it is not sufficient unless the suitably normalized values are actually integers for a channel symbol alphabet size of interest. Eq. (2.6) is of interest in its own right and a length function ℓ satisfying (2.6) will be said to be *admissible* whether or not it is m -admissible for some m . This justifies the use of the term in the definition of the length function component of a quantizer.

If a length function ℓ is admissible in any sense, then so is the resulting combined constraint instantaneous rate function $r_\eta(i) = (1 - \eta)\ell(i) + \eta \ln N(\mathcal{I})$ in the same sense.

2.5 Shannon's lossless coding theorem

Consider a lossless code ψ of a discrete random variable U with alphabet A_U and pmf $p(a) = \Pr(U = a), a \in A_U$. In particular, in the quantization setup $U = \mathcal{E}(X)$ and $A_U = \mathcal{I}$, a quantizer encoded version of a continuous random object X , and in the purely lossless compression problem, $U = X$ and $A_U = A$. Consider $m = 2$, a binary channel symbol alphabet, and let $\ell(a) = \text{length}(\psi(a))$ be the length function (in bits).

From Lemma 1.3 with $w(a) = 2^{-\ell(a)}$, we immediately have part of Shannon's famous first theorem.

Lemma 2.4 *Given a uniquely decodable noiseless variable length code with encoder ψ operating on a discrete random variable U with entropy $H(U)$, then the resulting average codeword length satisfies*

$$E[\ell(U)] \geq H(U); \quad (2.7)$$

i.e., the average length of the code can be no smaller than the entropy of the marginal pmf. The inequality is an equality iff

$$p(a) = 2^{-\ell(a)} \text{ for all } a \in A. \quad (2.8)$$

To achieve the lower bound, (2.8) must be satisfied, which is only possible if all probabilities that are powers of $1/2$. Thus the best length function might not yield a practicable lossless code. More generally, for an m -ary lossless code, (2.8) will correspond to an actual set of lengths if $\ell_m(a) = -\log_m p(a)$ are all integers.

The average length can never be lower than the entropy, so it is natural to ask how close this lower bound a good code can get. The following result gives a first answer.

Lemma 2.5 *There exists a uniquely decodable scalar noiseless code for a source with marginal pmf p for which the average codeword length satisfies*

$$E[\ell(U)] < H(p) + 1. \quad (2.9)$$

Proof: Given the source probabilities $p(a)$, for each a , choose $\ell(a)$ to be the integer which “sandwiches” $p(a)$ in the sense that

$$2^{-\ell(a)} \leq p(a) < 2^{-\ell(a)+1} \quad (2.10)$$

or, equivalently,

$$-\log p(a) \leq \ell(a) < -\log p(a) + 1 \quad (2.11)$$

so that $\ell(a)$ is the smallest integer greater than or equal to $-\log p(a)$. This can be interpreted as a rough approximation to the ideal codeword lengths

$$\ell(a) \approx -\log p(a).$$

The lengths satisfy the Kraft inequality since

$$\sum_a 2^{-\ell(a)} \leq \sum_a p(a) = 1,$$

hence there is a prefix-free (and hence a uniquely decodable) code with these lengths. From (2.11), the average length must satisfy the bound of the theorem. \square

Combining Lemmas 2.4-2.5 yields Shannon's lossless coding theorem (Shannon's first theorem):

Theorem 2.1 Shannon's lossless coding theorem

For any uniquely decodable lossless code of a discrete source U with pmf p ,

$$E[\ell(U)] \geq H(p); \quad (2.12)$$

and there exists a prefix-free code for which

$$E[\ell(U)] < H(p) + 1. \quad (2.13)$$

The Shannon code is simple and provides a useful general bound, but other codes can in general provide either superior performance or lower complexity implementations when the discrete alphabet is large. The best known lossless codes are Huffman codes, Lempel-Ziv codes, and arithmetic codes. Here only the first is treated in detail since it is the optimal code given the structure assumed so far if separately and independently coding each observed input symbol. The other two codes must encode possibly variable length sequences of inputs in order to perform well

2.6 Huffman codes

No lossless code can have an average rate strictly smaller than the entropy of the underlying pmf, and the Shannon code guarantees the existence of a lossless code with average rate no greater than the entropy +1. In 1952 D. A. Huffman provided a constructive technique for an optimal lossless code given the underlying pmf. The construction is based on two optimality properties of lossless codes, which are considered next.

Lemma 2.6 *Suppose that ψ is a uniquely decodable variable length binary noiseless source code and that $\{\ell(a)$ is the collection of codeword lengths. Then there is a prefix-free code with the same lengths and the same average length.*

The lemma follows since the lengths of a uniquely decodable code must satisfy the Kraft inequality and hence there must exist a prefix-free code with these lengths (from the tree construction of the sufficiency proof for the Kraft inequality of Lemma 2.3).

Theorem 2.2 *An optimum binary prefix-free code has the following properties:*

- (i) *If the codeword for input symbol a has length $\ell(a)$, then $p(a) > p(b)$ implies that $\ell(a) \leq \ell(b)$; that is, more probable input symbols have shorter (at least, not longer) codewords.*

(ii) *The two least probable input symbols have codewords which are equal in length and differ only in the final symbol.*

Proof: (i) If $p(a) > p(b)$ and $\ell(a) > \ell(b)$, then exchanging codewords will cause a strict decrease in the average length. Hence the original code could not have been optimum.

(ii) Suppose that the two codewords have different lengths. A prefix of a longer codeword can not itself be a codeword, and hence final symbol of the longer codeword can be deleted without confusion. This strictly decreases the average length of the code, which implies a contradiction. Thus the two least probable codewords must have equal length.

Suppose that these two codewords differ in some position other than the final one. The final binary symbol could then be removed yielding a shorter code without confusion (the shorter codewords could still be distinguished and the prefix condition precludes the possibility of confusion with another codeword). This, however, would yield a strict decrease in average length, yielding a contradiction. \square

The theorem provides an iterative design technique which Huffman demonstrated yields an optimal code and yields the entropy lower bound if the input probabilities are powers of $1/2$.

Order the input symbols in terms of probability, that is, $p(a_0) \geq p(a_1) \geq \dots \geq p(a_{M-1})$.

The two least probable symbols must end up with codewords of the same length which differ only in the final binary symbol. So begin a code tree with two terminal nodes with branches extending back to a common node. Label one branch 0 and the other 1. Consider these two input symbols to be *tied* together and form a single new symbol in a reduced alphabet A' with $M - 1$ symbols in it.

Next find an optimal code for the reduced alphabet A' with probabilities $p(a_m)$; $m = 0, 1, \dots, M - 3$ and $p(a_{M-1}) + p(a_{M-2})$.

A prefix code for A' implies a prefix code for A by adjoining the final branch labels already selected. If the prefix code for A' is optimal, then so is the induced code for A .

Continue in this fashion:

- The probability of each node is found by adding up the probabilities of all input symbols connected to the node.
- At each step the two least probable nodes in the tree are found.

length N has an average codeword length no smaller than the N th order entropy $H(U^N)$ of the input defined by

$$H(U^N) = H(p_{U^N}) = - \sum_{u^N} p_{U^N}(u^N) \log p_{U^N}(u^N).$$

From the lossless coding theorem, any uniquely decodable lossless code will have average length bounded below by

$$E[\psi(U^N)] \geq \frac{1}{N} H(U^N) \quad (2.14)$$

and there exists a prefix-free code for which

$$E[\psi(U^N)] < \frac{1}{N} H(U^N) + \frac{1}{N}. \quad (2.15)$$

A Huffman code for N -tuples, for example, will fall between these bounds.

If the input process is stationary, then minimum over N on the right hand side achieves a lower bound for all N , and the minimum is \bar{H} , the *entropy rate* of the source

$$\bar{H} = \lim_{N \rightarrow \infty} \frac{H(U^N)}{N}.$$

If the source is iid, then $\bar{H} = H(U_0)$.

The conclusion is that the entropy rate of the input process $\{U_n\}$ is an unbeatable lower bound to the average length that can be achieved using uniquely decodable lossless codes, and the bound can be approached arbitrarily closely by Huffman (or Shannon) coding sufficiently large vectors. Unfortunately, however, this does not provide a solution to the practical problem of lossless coding because the complexity of the lossless codes grows too high to be practicable as N gets large.

Many alternative schemes are, strictly speaking, suboptimal, in comparison with the Huffman code, but by operating on variable numbers of input symbols than can have manageable complexity encoding large numbers of symbols, but can still provide near optimal performance when long sequences are encoded.