Agenda for the lecture

- Introduction to data compression
- Fixed- and variable-length codes
- Types of error-free codes: Nonsingular, Uniquely-decodable and Prefix-free

13.1 The data compression problem

The broad goal of the compression problem is to store data using as few bits as possible. The first stage in a data compression scheme is usually “domain knowledge” specific and takes into account where the data is coming from. In this course we shall assume that all the domain knowledge specific knowledge has been used to convert the data into a sequence of symbols $Y_1, ..., Y_L$. The next step entails grouping these symbols appropriately into “super-symbols” coming from an alphabet $\mathcal{X}$ and producing a stream $X_1, ..., X_n$. This, too, varies with the physical form of the data at hand and the source of the data. One can also use the underlying source sequence to build a probability model for the data. The task now is to compress the sequence $X_1, ..., X_n$ with as few bits as possible and with low probability of error with respect to the model distribution. In the first part, we shall

\[\text{c} \quad \text{Himanshu Tyagi. Feel free to use with acknowledgement.}\]
assume that the distribution is known. Later we shall develop “universal” schemes that learn the probability model for the data (for a fixed class of models) along the way. The overall setup is depicted in Figure 1

![Figure 1: A general compression scheme](image)

13.1.1 Acceptable estimate

Before we proceed further, we need to agree on what we can accept as an estimate of $X^n$. Two scenarios are possible:

*Lossless compression:* In this case, we accept $\hat{X}^n$ as an estimate for $X^n$ only if $\hat{X}^n = X^n$; else we consider it as an error event.

*Lossy compression:* In this case, an estimate $\hat{X}^n$ is acceptable if it is within a fixed “distance” of $X^n$, where the distance is measured by a predefined distortion metric. Specifically, we fix a distortion mapping $d_n : \mathcal{X}^n \times \mathcal{X}^n \to \mathbb{R}$ and a distortion level $D_n$ and accept $\hat{X}^n$ only if

$$d_n(X^n, \hat{X}^n) \leq D_n.$$
Typically, we choose \( d_n \) to have an additive form and \( D_n = n\delta \), i.e., \( \hat{X}^n \) is acceptable as an estimate of \( X^n \) if the average distortion \( \frac{1}{n} \sum_{i=1}^{n} d(X_i, \hat{X}_i) \leq \delta \); else we declare an error.

Commonly used distortion mappings \( d \) for the discrete case is the Hamming distortion function \( d(x, y) = \mathbb{1}(x \neq y) \) and for the case \( X = \mathbb{R} \) is the squared-loss function \( d(x, y) = (x - y)^2 \).

For now, we shall focus on lossless compression and will return to lossy compression after covering channel coding, since lossy compression is closely related to channel coding.

### 13.1.2 Fixed- and variable-length codes

We can either consider variable-length codes, or restrict to block codes where each codeword is of the same length. Formally,

**Definition 13.1 (Variable-length source codes).** A source code for a source with alphabet \( \mathcal{X} \) consists of an encoder \( e : \mathcal{X} \rightarrow \{0, 1\}^* \) and a decoder \( d : \{0, 1\}^* \rightarrow \mathcal{X} \), where \( \{0, 1\}^* \) denotes the set of all finite length binary sequences including the empty sequence \( \emptyset \). The set \( \mathcal{C} = e(\mathcal{X}) = \{e(x) : x \in \mathcal{X}\} \) is called the code-set or, when there is no ambiguity, simply the code. Each element of the code-set \( \mathcal{C} \) is called a codeword. We denote by \( l(c) \) the length of a codeword \( c \in \mathcal{C} \).

**Definition 13.2 (Fixed-length source codes).** A source code is of fixed-length or is a block-code if each codeword in it has the same length. The common length is referred to as the length of the block-code.

**Definition 13.3 (Probability of error).** Given a source \( X \) over a finite alphabet \( \mathcal{X} \), the probability of error of a source code \( (e, d) \) is given by \( \epsilon(e, d) = \mathbb{P}(d(e(X)) \neq X) \).

### 13.1.3 Measures of performance

There are two possible ways to measure the length of a code: The worst-case length over all codewords and the average length with average taken over \( P_X \). Specifically, for an encoder
$e$, the worst-case length of a code $l_w(e)$ is given by

$$l_w(e) := \max_{x \in X} l(e(x)),$$

and the average length of the code $l_a(e)$ is given by

$$l_a(e) := \sum_{x \in X} P_X(x) l(e(x)) = E[l(e(X))].$$

Clearly, the two definitions above coincide for block-codes. Overall, following criteria are of interest:

$$L_\epsilon(X) := \min \{l_w(e) : (e, d) \text{ is a fixed-length code with } \epsilon(e, d) \leq \epsilon\};$$

$$L^*_\epsilon(X) := \min \{l_w(e) : (e, d) \text{ is a variable-length code with } \epsilon(e, d) \leq \epsilon\};$$

$$\mathcal{L}^*_\epsilon(X) := \min \{l_a(e) : (e, d) \text{ is a variable-length code with } \epsilon(e, d) \leq \epsilon\};$$

$\mathcal{L}_\epsilon(X)$ can be defined analogously, but it coincides with $L_\epsilon(X)$.

Note that, in general, $L^*_\epsilon(X) \leq L_\epsilon(X)$. Also, if $L^*_\epsilon(X) = k$ then all the codewords have length less than $k$. There are no more than $1 + 2 + 2^2 + \ldots + 2^k \leq 2^{k+1} - 1$. Thus, all the sequences can be mapped in a one-to-one fashion to the sequences in $\{0,1\}^{k+1}$, which will lead to a fixed-length code of length less than $k + 1$. Thus, $L_\epsilon(X) \leq L^*_\epsilon(X) + 1$. Therefore, $L^*_\epsilon(X)$ gains at most one bit over $L_\epsilon(X)$, which we generously ignore – for the rest of the course we shall only consider $L_\epsilon(X)$.

### 13.2 Types of error-free codes

We first consider the case when no error is allowed. For such codes, each codeword $c = e(x)$, for $x$ with $P_X(x) > 0$, must uniquely recover back the symbol $x$ corresponding to it. Therefore, in principle, the decoder is simply the inverse of $e$. Thus, these codes are
simply characterized by their codeword set $\mathcal{C}$ and satisfy $|\mathcal{C}| = |\text{supp}(P_X)|$; without loss of generality, we assume that $\text{supp}(P_X) = \mathcal{X}$. Thus, we seek codes such that $|\mathcal{C}| = |\mathcal{X}|$. Such codes are termed *nonsingular codes*. The minimum over average lengths $l_a(e)$ of nonsingular codes $(e, d)$ is denoted by $\mathcal{L}_0(X)$, or simply by $\mathcal{L}(X)$.

We shall consider two specific classes of nonsingular codes, the first more general than the second.

### 13.2.1 Uniquely decodable codes

Often one is not interested in compressing a single symbol $x$ using a nonsingular code $\mathcal{C}$ but a sequence of symbols $x_1, ..., x_k$ using the same nonsingular code $\mathcal{C}$. An arbitrary nonsingular code may not be good for this task. For instance, a simple code which uses 11 for a symbol $a$ and 111 for a symbol $b$ cannot distinguish between symbols $aaa$ and $bb$.

Nonsingular codes which do not suffer from this shortcoming are called *uniquely decodable codes*.

Specifically, denote by $\chi_k : \{0, 1\}^* \times \cdots \times \{0, 1\}^* \to \{0, 1\}^*$ the $k$-concatenation map which maps $k$ binary sequences to their concatenation. A code $\mathcal{C}$ is uniquely decodable if for every $k \in \mathbb{N}$, the code $\cup_{j=1}^k \chi_j(\mathcal{C}^j)$ is a nonsingular code, i.e.,

$$\sum_{j=1}^k |\chi_j(\mathcal{C}^j)| = \frac{|\mathcal{C}|^{k+1} - 1}{|\mathcal{C}| - 1} = \frac{|\mathcal{X}|^{k+1} - 1}{|\mathcal{X}| - 1},$$

where, for a function $f$, $f(A)$ denotes the set $\{f(x) : x \in A\}$.

The minimum over average lengths $l_a(e)$ of uniquely decodable codes $(e, d)$ is denoted by $\mathcal{L}^u(X)$.

**Remark 13.4.** Note that while for any fixed $k$, we can decode the codeword consisting of concatenation of $k$ or fewer codewords from $\mathcal{C}$, such decoding is not possible without the knowledge of $k$. Thus, to decode a codeword sequence from uniquely decodable code, the maximum number of codewords from $\mathcal{C}$ concatenated to form the codeword sequence
must be specified. A special class of uniquely decodable codes, described next, require that we can decode the sequence without the knowledge of the maximum number of codewords used to form it.

13.2.2 Prefix-free or instantaneous codes

A special class of uniquely decodable codes have the property that the subsequence \((x_1, ..., x_k)\) of the sequence \((x_1, ..., x_n)\) coded as \((c_1, ..., c_n)\) can be recovered from \((c_1, ..., c_k)\). Such codes are characterized by the prefix-free property, namely no codeword is a prefix of another. We call these codes prefix-free codes or instantaneous codes (short for instantaneously decodable codes). Note that all fixed length codes (block codes) are prefix-free.

The minimum over average lengths \(l_a(e)\) of prefix-free codes \((e, d)\) is denoted by \(L^p(X)\).

We emphasize that the prefix-free restriction and uniquely decodable restriction do not have much significance for compressing a single symbol. They emerge from the algorithmic need of compressing a sequence of symbols. In the next lecture, we will study if we loose anything in compression of a single symbol by imposing such restrictions on the code.

Table 1 provides examples of codes of various classes.

<table>
<thead>
<tr>
<th>Alphabet</th>
<th>code1</th>
<th>code2</th>
<th>code3</th>
<th>code4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>01</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>010</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

Table 1: Example codes of various classes. Code 1 is a block-code and hence is prefix free; code 2 is nonsingular, but not uniquely decodable; code 3 is uniquely decodable but not prefix-free (why?); and code 4 is prefix free.