Data Compression

Data Compression

Data compression can be achieved by assigning short descriptions to the most frequent outcomes of the data source.

We first define the notion of an instantaneous code and then prove the important Kraft inequality, which asserts that the exponentiated codeword length assignments must look like a probability mass function. Elementary calculus then shows that the expected description length must be greater than or equal to the entropy, the first main result.

Examples of Codes

Definition A *source code C* for a random variable X is a mapping from X, the range of X, to D^*, the set of finite-length strings of symbols from a D-ary alphabet. Let $C(x)$ denote the codeword corresponding to x and let $l(x)$ denote the length of $C(x)$.

For example, $C(\text{red}) = 00$, $C(\text{blue}) = 11$ is a source code for $X = \{\text{red}, \text{blue}\}$ with alphabet $D = \{0, 1\}$.

Definition The *expected length* $L(C)$ of a source code $C(x)$ for a random variable X with probability mass function $p(x)$ is given by:

$$L(C) = \sum_{x \in \chi} p(x) l(x)$$

where $l(x)$ is the length of the codeword associated with x. Without loss of generality, we can assume that the D-ary alphabet is $D = \{0, 1, \ldots, D-1\}$.

Example 1

Let X be a random variable with the following distribution and codeword assignment:

$\Pr(X = 1) = 1/2$, codeword $C(1) = 0$

$\Pr(X = 2) = 1/4$, codeword $C(2) = 10$

$\Pr(X = 3) = 1/8$, codeword $C(3) = 110$

$\Pr(X = 4) = 1/8$, codeword $C(4) = 111$.

The entropy $H(X)$ of X is 1.75 bits, and the expected length $L(C) = El(X)$ of this code is also 1.75 bits. Here we have a code that has the same average length as the entropy. We note that any sequence of bits can be uniquely decoded into a sequence of symbols of X. For example, the bit string 0110111100110 is decoded as 134213.

Example 2

Consider another simple example of a code for a random variable:

Pr$(X = 1) = 1/3$, codeword $C(1) = 0$

Pr$(X = 2) = 1/3$, codeword $C(2) = 10$

Pr$(X = 3) = 1/3$, codeword $C(3) = 11$.

The code is uniquely decodable. However, in this case the entropy is log 3 = 1.58 bits and the average length of the encoding is 1.66 bits. Here $El(X) > H(X)$.

Example: Morse Code

The Morse code is a reasonably efficient code for the English alphabet using an alphabet of four symbols: a dot, a dash, a letter space, and a word space. Short sequences represent frequent letters (e.g., a single dot represents E) and long sequences represent infrequent letters (e.g., Q is represented by "dash,dash,dot,dash"). This is not the optimal representation for the alphabet in four symbols. In fact, many possible codewords are not utilized because the codewords for letters do not contain spaces except for a letter space at the end of every codeword, and no space can follow another space. It is an interesting problem to calculate the number of sequences that can be constructed under these constraints. The problem was solved by Shannon in his original 1948 paper.

Nonsingularity

We now define increasingly more stringent conditions on codes. Let x_n denote $(x_1, x_2, \ldots, x_n)$.

Definition A code is said to be *nonsingular* if every element of the range of X maps into a different string in D^*; that is, $x_1 \neq x_2 \Rightarrow C(x_1) \neq C(x_2)$. Nonsingularity suffices for an unambiguous description of a single value of X. But we usually wish to send a sequence of values of X. In such cases we can ensure decodability by adding a special symbol (a "comma") between any two codewords. But this is an inefficient use of the special symbol; we can do better by developing the idea of selfpunctuating or instantaneous codes. Motivated by the necessity to send sequences of symbols X, we define the extension of a code as follows:

Code Extension

Definition The *extension* C^* of a code C is the mapping from finitelength strings of X to finite-length strings of D, defined by:

$$C(x_1, x_2, \ldots x_n) = C(x_1)C(x_2)\ldots C(x_n)$$

where $C(x_1)C(x_2) \cdots C(x_n)$ indicates concatenation of the corresponding codewords.

Example: If $C(x_1) = 00$ and $C(x_2) = 11$, then $C(x_1x_2) = 0011$.

Unique Decodability

Definition A code is called *uniquely decodable* if its extension is nonsingular. In other words, any encoded string in a uniquely decodable code has only one possible source string producing it. However, one may have to look at the entire string to determine even the first symbol in the corresponding source string.

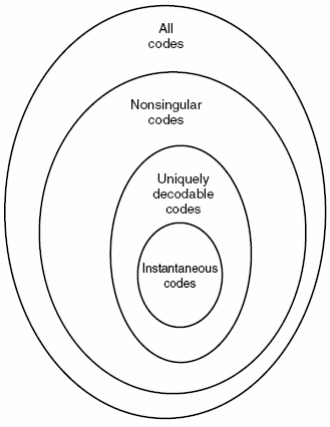
Definition A code is called a *prefix code* or an *instantaneous code* if no codeword is a prefix of any other codeword.

An instantaneous code can be decoded without reference to future codewords since the end of a codeword is immediately recognizable. Hence, for an instantaneous code, the symbol x_i can be decoded as soon as we come to the end of the codeword corresponding to it. We need not wait to see the codewords that come later. An instantaneous code is a *selfpunctuating code*; we can look down the sequence of code symbols and add the commas to separate the codewords without looking at later symbols. For example, the binary string 01011111010 produced by the code of Example 1 is parsed as 0,10,111,110,10.

The nesting of these definitions is shown in Figure. To illustrate the differences between the various kinds of codes, consider the examples of codeword assignments $C(x)$ to $x \in X$ in Table.

X	Singular	Nonsingular, But Not Uniquely Decodable	Uniquely Decodable, But Not Instantaneous	Instantaneous
1	0	0	10	0
2	0	010	00	10
3	0	01	11	110
4	0	10	110	111

For the nonsingular code, the code string 010 has three possible source sequences: 2 or 14 or 31, and hence the code is not uniquely decodable.



Comments

To see that it is uniquely decodable, take any code string and start from the beginning. If the first two bits are 00 or 10, they can be decoded immediately. If the first two bits are 11, we must look at the following bits. If the next bit is a 1, the first source symbol is a 3. If the length of the string of 0's immediately following the 11 is odd, the first codeword must be 110 and the first source symbol must be 4; if the length of the string of 0's is even, the first source symbol is a 3. By repeating this argument, we can see that this code is uniquely decodable. Sardinas and Patterson have devised a finite test for unique decodability, which involves forming sets of possible suffixes to the codewords and eliminating them systematically. The fact that the last code in the Table is instantaneous is obvious since no codeword is a prefix of any other.

Kraft Inequality

We wish to construct instantaneous codes of minimum expected length to describe a given source. It is clear that we cannot assign short codewords to all source symbols and still be prefix-free. The set of codeword lengths possible for instantaneous codes is limited by the following inequality.

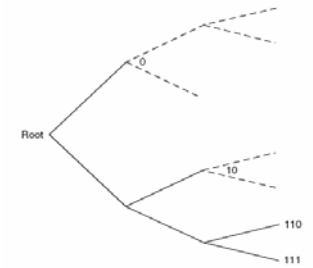
Theorem (*Kraft inequality*) For any instantaneous code (prefix code) over an alphabet of size D, the codeword lengths $l_1, l_2, \ldots, l_m$ must satisfy the inequality

$$\sum_i D^{-l_i} \leq 1$$

Conversely, given a set of codeword lengths that satisfy this inequality, there exists an instantaneous code with these word lengths.

Proof of Kraft Inequality

Proof: Consider a *D*-ary tree in which each node has *D* children. Let the branches of the tree represent the symbols of the codeword. For example, the *D* branches arising from the root node represent the *D* possible values of the first symbol of the codeword. Then each codeword is represented by a leaf on the tree. The path from the root traces out the symbols of the codeword. A binary example of such a tree is shown in Figure. The prefix condition on the codewords implies that no codeword is an ancestor of any other codeword on the tree. Hence, each codeword eliminates its descendants as possible codewords.



Proof of Kraft Inequality

Let $l_{\max}$ be the length of the longest codeword of the set of codewords. Consider all nodes of the tree at level $l_{\max}$. Some of them are codewords, some are descendants of codewords, and some are neither. A codeword at level l_i has $D^{l_{\max}-l_i}$ descendants at level $l_{\max}$. Each of these descendant sets must be disjoint. Also, the total number of nodes in these sets must be less than or equal to $D^{l_{\max}}$. Hence, summing over all the codewords, we have:

$$\sum_i D^{l_{\max}-l_i} \leq D^{l_{\max}}$$

Which is the Kraft inequality.

Extended Kraft Inequality

Conversely, given any set of codeword lengths $l_1, l_2, \ldots, l_m$ that satisfy the Kraft inequality, we can always construct a tree like the one in Figure. Label the first node (lexicographically) of depth l_1 as codeword 1, and remove its descendants from the tree. Then label the first remaining node of depth l_2 as codeword 2, and so on. Proceeding this way, we construct a prefix code with the specified $l_1, l_2, \ldots, l_m$.

We now show that an infinite prefix code also satisfies the Kraft inequality.

Extended Kraft Inequality

Theorem (*Extended Kraft Inequality*) For any countably infinite set of codewords that form a prefix code, the codeword lengths satisfy the extended Kraft inequality,

$$\sum_{i=1}^{\infty} D^{-l_i} \leq 1$$

Conversely, given any l1, l2, . . . satisfying the extended Kraft inequality, we can construct a prefix code with these codeword lengths.

Proof: Let the D-ary alphabet be $\{0, 1, \ldots, D-1\}$. Consider the ith codeword $y_1 y_2 \cdots y_{l_i}$. Let $0.y_1 y_2 \cdots y_{l_i}$ be the real number given by the D-ary expansion

$$0.y_1 y_2 \ldots y_{l_i} = \sum_{j=1}^{l_i} y_j D^{-j}$$

Proof of Extended Kraft Inequality

This codeword corresponds to the interval:

$$[0.y_1y_2...y_{l_i}, 0.y_1y_2...y_{l_i} + \frac{1}{D^{l_i}})$$

the set of all real numbers whose D-ary expansion begins with $0.y_1y_2 \cdots y_{li}$. This is a subinterval of the unit interval [0, 1]. By the prefix condition, these intervals are disjoint. Hence, the sum of their lengths has to be less than or equal to 1. This proves that

$$\sum_{i=1}^{\infty} D^{-l_i} \leq 1$$

Just as in the finite case, we can reverse the proof to construct the code for a given $l_1, l_2, \ldots$ that satisfies the Kraft inequality. First, reorder the indexing so that $l_1 \leq l_2 \leq \ldots$. Then simply assign the intervals in order from the low end of the unit interval. For example, if we wish to construct a binary code with $l_1 = 1, l_2 = 2, \ldots$, we assign the intervals $[0, 1/2), [1/2, 1/4), \ldots$ to the symbols, with corresponding codewords 0, 10,. . . .

Optimal Codes

We proved that any codeword set that satisfies the prefix condition has to satisfy the Kraft inequality and that the Kraft inequality is a sufficient condition for the existence of a codeword set with the specified set of codeword lengths. We now consider the problem of finding the prefix code with the minimum expected length.

This is equivalent to finding the set of lengths $l_1, l_2, \ldots, l_m$ satisfying the Kraft inequality and whose expected length $L=\sum l_i p_i$ is less than the expected length of any other prefix code. This is a standard optimization problem.

Minimize:

$$L = \sum l_i p_i$$

over all integers $l_1, l_2, \ldots, l_m$ satisfying

$$\sum D^{-l_i} \leq 1$$

Optimal Codes

We neglect the integer constraint on l_i and assume equality in the constraint. Hence, we can write the constrained minimization using Lagrange multipliers as the minimization of

$$J = \sum p_i l_i + \lambda(\sum D^{-l_i})$$

Differentiating with respect to li, we obtain

$$\frac{\partial J}{\partial l_i} = p_i - \lambda D^{-l_i} \log_e D$$

Setting the derivative to 0, we obtain

$$D^{-l_i} = \frac{p_i}{\lambda \log_e D}$$

Substituting this in the constraint to find λ, we find $\lambda = 1/(\log_e D)$, and hence

$$p_i = D^{-l_i}$$

yielding optimal code lengths

$$l_i^* = -\log_D p_i$$

Optimal Code Length

This noninteger choice of codeword lengths yields expected codeword length

$$L^* = \sum p_i l_i^* = -\sum p_i \log_D p_i = H_D(X)$$

But since the l_i must be integers, we will not always be able to set the codeword lengths in this way. Instead, we should choose a set of codeword lengths li "close" to the optimal set. We verify that $l^*_i = -\log_D p_i$ is a global minimum by the proof of the following theorem.

Theorem The expected length L of any instantaneous D-ary code for a random variable X is greater than or equal to the entropy $H_D(X)$. That is,

$$L \geq H_D(X)$$

with equality if and only if $D^{-li} = p_i$.

Expected Code Length

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

$$L - H_D(X) = \sum l_i p_i - \sum p_i \log_D \frac{1}{p_i}$$

$$= -\sum p_i \log_D D^{-l_i} + \sum p_i \log_D p_i$$

Letting $r_i = D^{-l_i} / \sum_j D^{-lj}$ and $c = \sum D^{-li}$ we obtain:

$$L - H = \sum p_i \log_D \frac{p_i}{r_i} - \log_D c$$

$$= D(p \| r) + \log_D \frac{1}{c} \geq 0$$

by the nonnegativity of relative entropy and the fact (Kraft inequality) that $c \leq 1$. Hence, $L \geq H$ with equality if and only if $p_i = D^{-li}$ (i.e., if and only if $-\log_D p_i$ is an integer for all i).

Expected Code Length

Definition A probability distribution is called *D-adic* if each of the probabilities is equal to D^{-n} for some n. Thus, we have equality in the theorem if and only if the distribution of X is D-adic.

The preceding proof also indicates a procedure for finding an optimal code: Find the D-adic distribution that is closest (in the relative entropy sense) to the distribution of X. This distribution provides the set of codeword lengths. Construct the code by choosing the first available node as in the proof of the Kraft inequality. We then have an optimal code for X. However, this procedure is not easy, since the search for the closest D-adic distribution is not obvious. In the next part we give a good suboptimal procedure (Shannon–Fano coding). We will see later a simple procedure to find the optimal code (Huffman coding).

Bounds on the Optimal Code Length

We now demonstrate a code that achieves an expected description length L within 1 bit of the lower bound; that is,

$$H(X) \le L < H(X)+1$$

Recall the concept of optimal codes: We wish to minimize $L = \sum p_i l_i$ subject to the constraint that $l_1, l_2, \ldots, l_m$ are integers and $D^{-l_i} \le 1$. We proved that the optimal codeword lengths can be found by finding the D-adic probability distribution closest to the distribution of X in relative entropy, that is, by finding the D-adic $\mathbf{r}$ ($r_i = D^{-l_i} / \sum_j D^{-l_j}$) minimizing

$$L - H_D = D(p \| r) - \log\left(\sum D^{-l_i}\right) \ge 0$$

The choice of word lengths $l_i = \log_D 1/p_i$ yields $L = H$. Since $\log_D 1/p_i$ may not equal an integer, we round it up to give integer word-length assignments,

$$l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil$$

Bounds on the Optimal Code Length

where the quantity $\lceil x \rceil$ is the smallest integer $\ge x$. These lengths satisfy the Kraft inequality since

$$\sum D^{\left\lceil -\log \frac{1}{p_i} \right\rceil} \le \sum D^{-\log \frac{1}{p_i}} = \sum p_i = 1$$

From the choice of l_i, we can see that these codeword lengths satisfy

$$\log_D \frac{1}{p_i} \le l_i \le \log_D \frac{1}{p_i} + 1$$

Multiplying by p_i and summing over i, we obtain

$$H_D(X) \le L < H_D(X)+1$$