Lecture 10: Data compression

Course instructor: Alexey Frolov

al.frolov@skoltech.ru

Teaching Assistant: Stanislav Kruglik

stanislav.kruglik@skolkovotech.ru

February 20, 2018

Outline

Source characteristics

Let us have a finite alphabet $\mathcal{X}$ of size m and a source which emits a letter $i \in \mathcal{X}$ with probability p_i.

We assume that the next symbol appears independently from the previous one.

Basic goal of source coding

We want to represent each letter of initial alphabet as a sequence from another alphabet (*codeword*) in such a manner that different letters of initial alphabet may have different lengths (*variable-length code*).

We want to *minimize the average sequence length*.

It's obvious that the most probable letter should have the smallest length and we have to solve minimization task $\min \sum p_i l_i$. By l_i we denote the length of the sequence corresponding to the letter i.

Source code

A source code $\mathcal{C}$ for a random variable X is a mapping from $\mathcal{X}$, the range of X, to D^*, the set of finite length strings of symbols from a D-ary alphabet.

Let $\mathcal{C}(x)$ denote the codeword corresponding to x and let $l(x)$ denote the length of $\mathcal{C}(x)$.

Example

Example

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

Expected length

$$L(\mathcal{C}) = \sum_{x \in \mathcal{X}} p(x) l(x),$$

where $l(x)$ is is the length of the codeword associated with x.

Example

Example

$$\begin{aligned}
\Pr(X = 1) &= 1/2,\ \mathcal{C}(1) = 0 \\
\Pr(X = 2) &= 1/4,\ \mathcal{C}(1) = 10 \\
\Pr(X = 3) &= 1/8,\ \mathcal{C}(1) = 110 \\
\Pr(X = 4) &= 1/8,\ \mathcal{C}(1) = 111
\end{aligned}$$

$L(\mathcal{C}) = H(X) = 1.75$ bits.

Example

Example

$$\begin{aligned}\Pr(X=1) &= 1/3,\ \mathcal{C}(1)=0\\ \Pr(X=2) &= 1/3,\ \mathcal{C}(1)=10\\ \Pr(X=3) &= 1/3,\ \mathcal{C}(1)=11\end{aligned}$$

$L(\mathcal{C}) = 1.66 > H(X) = 1.58$ (all quantities are in bits)

Non-singular codes

Code is non-singular if every element of the range of X maps into a different string, i.e.

$$x_i \neq x_j \Rightarrow \mathcal{C}(x_i) \neq \mathcal{C}(x_j)$$

Uniquely decodable codes

The extension $\mathcal{C}^*$ of a code $\mathcal{C}$ is the mapping from finite length strings of $\mathcal{X}$ to finite length strings of D, defined by

$$\mathcal{C}(x_1 x_2 \dots x_n) = \mathcal{C}(x_1)\mathcal{C}(x_2)\dots\mathcal{C}(x_n),$$

where $\mathcal{C}(x_1)\mathcal{C}(x_2)\dots\mathcal{C}(x_n)$ denotes a concatenation of the corresponding codewords.

Code is uniquely decodable if its extension is non-singular (can be interpreted as follows: there is only one possible decision for each data sequence).

Prefix codes

Code is prefix if no one codeword can be the origin (prefix) of any other codeword.

Link between uniquely decodable and prefix codes

Prefix code $\Rightarrow$ uniquely decodable code

Uniquely decodable code $\not\Rightarrow$ prefix code

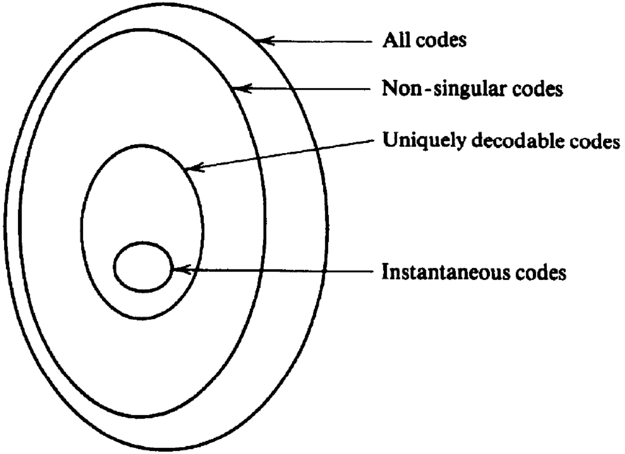
Prefix codes are a subset of uniquely decodable codes.

Example

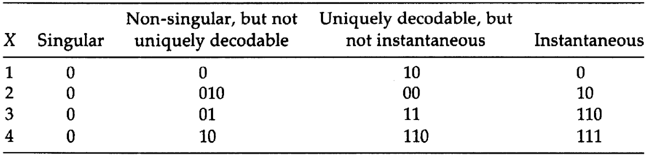
Code $C = \{z_1 = 0, z_2 = 10, z_3 = 11, z_4 = 111\}$ is prefix and uniquely decodable.

Code $C = \{z_1 = 0, z_2 = 01, z_3 = 011, z_4 = 111\}$ is not prefix, but uniquely decodable

Link between uniquely decodable and prefix codes



Link between uniquely decodable and prefix codes



X	Singular	Non-singular, 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

Kraft inequality

Theorem

For any prefix code over the alphabet of size D, the codeword lengths $l_1, \ldots, l_m$ must satisfy the inequality

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

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

Proof

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. Each codeword is represented by a leaf on the tree. 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.

Let $l_{\max} = \max\{l_1 \ldots l_m\}$, the due to the prefix condition we have

$$\sum_{i=1}^{m} D^{l_{\max}-l_i} \leq D^{l_{\max}}$$

$\square$

McMillan

Theorem

For any uniquely decodable code over the alphabet of size D, the codeword lengths $l_1, ...l_m$ must satisfy the Kraft inequality

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

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

Bounds on the optimal codelength

Optimization problem: minimize the average codelength ($L = \sum p_i l_i$) under condition subject to constraints $l_1, l_2, \dots, l_m$ are integers and $\sum D^{-l_i} \leq 1$.

Theorem

Let $l_1^*, l_2^*, \dots, l_m^*$ be the optimal codeword lengths for a source distribution P_X and a D-ary alphabet, and let L^* be the associated expected length of the optimal code ($L^* = \sum p_i l_i^*$). Then

$$\frac{H(X)}{\log D} \leq L^* < \frac{H(X)}{\log D} + 1$$

Bounds on the optimal codelength

Proof.

Optimal codeword lengths (method of Lagrange multipliers)

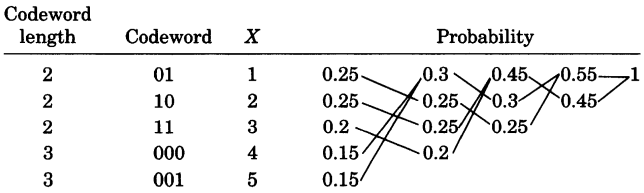
$$l_i = \left\lceil log_D \frac{1}{p_i} \right\rceil$$

□

Huffman code

Optimal code – code that minimizes average length of symbol representation.

Below we illustrate the Huffman algorithm. The main idea is to unite least probable letters in one virtual letter and construct multiple level tree that correspond to the code



Codeword length	Codeword	X	Probability				
2	01	1	0.25	0.3	0.45	0.55	1
2	10	2	0.25	0.25	0.3	0.45	
2	11	3	0.2	0.25	0.25		
3	000	4	0.15	0.2			
3	001	5	0.15				

This code has average length 2.3 bits.

Huffman code

Lemma

For any distribution, there exists an optimal prefix code (with minimum expected length) that satisfies the following properties:

- *If* $p_j > p_k$*, then* $l_j \leq l_k$
- *The two longest codewords have the same length*
- *The two longest codewords differ only in the last bit and correspond to the two least likely symbols*

Theorem

Huffman coding is optimal, i.e., if $\mathcal{C}^*$ *is the Huffman code and* $\mathcal{C}$ *is any other code, then* $L(\mathcal{C}^*) \leq L(\mathcal{C}')$.

Practical aspects

Despite of optimality, Huffman code is not widely used in practice. The reason is as follows: one need to know the exact values of probabilities p_i to construct the code. If you slightly change the probability distribution you have to reconstruct corresponding code tree. In the majority of cases optimum does not mean good practical aspects.

Universal coding

As the probabilities of letters may change we introduce classes in which all vectors have the same probability and first encode a number of this class (which is proportional to n^m) and then encode vectors of each class by some source code such as Huffman code. Because we additionally code class of vector we loose some useful information. It means, that (assume we encode n symbols)

$$\frac{\sum_{i=1}^{n} l_i}{n} \leq H(X) + \frac{\log_2 n}{n}$$

Note, that the length of the resulting sequence is bigger here in comparison to optimal source code ($H(X) + \frac{\log_2 n}{n}$ versus $H(X) + \frac{1}{n}$). But the difference is negligible in asymptotic regime.

Problem 1

Is the code $C = \{z_1 = 0; z_2 = 10; z_3 = 110; z_4 = 111\}$ uniquely decodable?

Problem 2

Is the code $C = \{z_1 = 0; z_2 = 01; z_3 = 011; z_4 = 111\}$ uniquely decodable?

Problem 3

Suppose that source X takes values from $\{0, 1\}$ with probability of 0 equals to $\frac{3}{4}$ and probability of 1 equals to $\frac{1}{4}$. What is the entropy of that source and probability of a string 0000011111 ?

Problem 4

Memoryless source X emits $m = 7$ letters with probabilities $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{32}$, $p_3 = \frac{1}{4}$, $p_4 = \frac{1}{32}$, $p_5 = p_6 = p_7$. Find the entropy of source X. Construct Shannon and Huffman prefix codes with alphabet $D = 3$. Check KraftMcMillan inequality. Compare the efficiency of these codes.

Thank you for your attention!