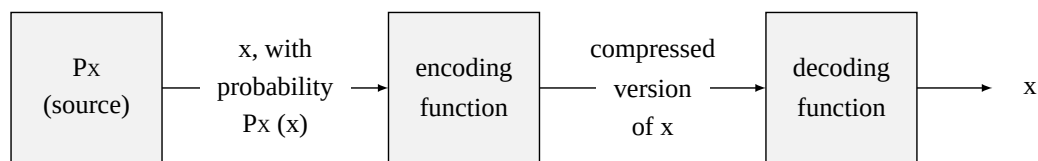


# Introduction to Source Coding

Suppose we sample  $x$  from a distribution  $P_X$  with image  $\mathcal{X}$ . In the context of data compression,  $P_X$  is typically called a **source** that emits value  $x \in \mathcal{X}$  with probability  $P_X(x)$ . We want to compress (or encode) symbols  $x$  sampled from  $P_X$  in such a way that we can later decompress (or decode) it reliably, without losing any information about the value  $x$ .



A counting argument shows that it is possible to encode the elements of  $\mathcal{X}$  by bit strings of length  $n$ , where  $n = \lceil \log(|\mathcal{X}|) \rceil$ : we simply list all elements of  $\mathcal{X}$ , and use the (binary) index of  $x$  in the list as its encoding. Thus, to store or to transmit an element  $x \in \mathcal{X}$ ,  $n$  bits of information always suffice. However, if not all  $x \in \mathcal{X}$  are equally likely according to  $P_X$ , one should be able to exploit this to achieve codes with shorter *average* length. The idea is to use encodings of varying lengths, assigning shorter codewords to the elements in  $\mathcal{X}$  that have higher probabilities, and vice versa.

Video by Khan Academy is licensed under **CC BY-NC-SA 3.0 US**.

In the video, Alice and Bob communicate by encoding their messages (dice rolls) from  $\mathcal{X} = \{2, 3, \dots, 12\}$  into a unitary alphabet  $\{1\}$ , where each 1 stands for a pluck of the wire. For example, the roll 8 is encoded as 111, or three plucks.

## Exercise

At the end of the video, Bob gets a better idea. He notices that they can pluck the wire in two different ways that are easy to distinguish: long or short. Can you design a code using this binary alphabet of plucks? How long are your codewords on average?

Show solution

The code on the right is an example of a code that Alice and Bob may use: 0 stands for a short pluck, 1 stands for a long one. Each die roll has a different

codeword, and short codewords are assigned to the most likely outcomes. The expected codeword length is

$$\frac{1}{36} \cdot 3 + \frac{2}{36} \cdot 3 + \frac{3}{36} \cdot 2 + \dots + \frac{1}{36} \cdot 3 = \frac{35}{18} \approx 1.944.$$

So on average, Alice and Bob expect to pluck the wire a little less than two times per die roll they want to communicate. However, if they want to communicate a list of die roll outcomes, they run into a problem: if Alice receives 011, how can she tell whether Bob sent the list [7,4], or [5,6], or even [2]?

Die roll	Codeword
2	011
3	001
4	11
5	01
6	1
7	0
8	00
9	10
10	000
11	010
12	100

This problem is resolved in the code on the right: confusions do not arise even when variable-length lists of messages are sent. The average codeword length is longer, however: roughly 3.306 plucks on average. In this module, you will encounter several algorithms for constructing such codes yourself for any given probability distribution.

Die roll	Codeword
2	11110
3	0010
4	0011
5	100
6	000
7	010

Die roll	Codeword
8	011
9	101
10	110
11	1110
12	11111

The question we will answer in this module is: how short can codes be in general (on average over repeated samples  $x$  from  $P_X$ )? We explore both **lossless** codes (where we want to recover the original data with certainty) and **lossy** codes (where with small probability, the data is lost).