1 Introduction

Usually, the purpose of studying data compression algorithms is twofold. The need for efficient storage and transmission is often the main motivation, but underlying every compression technique there is a model that tries to reproduce as closely as possible the information source to be compressed. This model may be interesting on its own, as it can shed light on the statistical properties of the source.

One of the most used aproaches for representing data dependencies relies on the use of Markov models. In lossless data compression, we use a specific type, called discrete time Markov chain or finite-context model.

A finite-context model can be used to collect statistical information of an information source, assigning a probability estimate to the symbols of the alphabet, according to a conditioning context computed over a finite and fixed number of past outcomes.

Let us denote by $x_1^n = x_1 x_2 \ldots x_n$, $x_i \in \Sigma$, the sequence of outputs (symbols from the source alphabet Σ) that the information source has generated until instant n. For example, if $\Sigma = \{0, 1\}$, then we may have $x_1^8 = 01001101$. In this example, we have $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, and so on. A k-order Markov model verifies

$$P(x_n|x_{n-1} \ldots x_{n-k}) = P(x_n|x_{n-1} \ldots x_{n-k} \ldots),$$

where the sequence $c = x_{n-1} \ldots x_{n-k}$ is called the state or context of the process.

A first-order Markov model reduces to

$$P(x_n|x_{n-1}) = P(x_n|x_{n-1}x_{n-2} \ldots).$$

As an example, consider that we want to model a binary image. We have two states, S_w (white pixel) and S_b (black pixel), and four possible transitions, namely $S_w \to S_w$, $S_w \to S_b$, $S_b \to S_w$, $S_b \to S_b$. The state transition diagram of this first-order model is shown in Fig. 1.

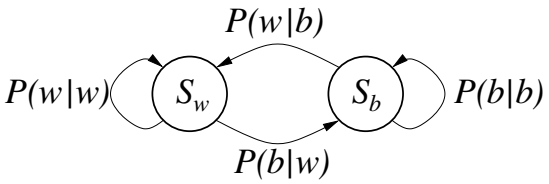

Figure 1: Example of a transition diagram of a Markov model for a binary image.

Markov models are particularly useful in text compression (but not only!), because the next letter in a word is generally heavily influenced by the preceding letters. In fact, the use of Markov models for written English appeared in the original work of Shannon. In 1951, he estimated the entropy of English to be in between about 0.6 and 1.3 bits per letter.

For simplicity, consider only 26 letters and the space character. The frequency of letters in English is far from uniform: the most common, "E", occurs about 13%, whereas the least common, "Q" and "Z", occur about 0.1% of the time. The frequency of pairs of letters is also nonuniform. For example, the letter "Q" is always followed by a "U". The most frequent pair is "TH" (occurs about 3.7%).

We can use the frequency of the pairs to estimate the probability that a letter follows any other letter. This reasoning can also be used for constructing higher-order models. However, the size of the model grows exponentially. For example, to build a third-order Markov model we need to estimate the values of $p(x_n|x_{n-1}x_{n-2}x_{n-3})$. This requires a table with $27^4 = 531\,441$ entries and enough text to correctly estimate the probabilities.

The following examples have been constructed using empirical distributions collected from samples of text (Shannon, 1948, 1951).

Example 1 Zero-order approximation (Independent and equiprobable symbols)

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGXYD
QPAAMKBZAACIBZLHJQD

Entropy: $\log 27 = 4.76$ *bits per letter.*

Example 2 First-order approximation (Independent, but with the correct $p(x)$)

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA
NAH BRL

Estimated entropy: ≈ 4.03 *bits per letter.*

Example 3 Second-order approximation (1st-order Markov model, i.e., $p(x_n|x_{n-1})$)

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

Estimated entropy: ≈ 3.32 *bits per letter.*

Example 4 Third-order approximation (2nd-order Markov model, i.e., $p(x_n|x_{n-1}x_{n-2})$)

IN NO IST LAT WHEY CRATICT FROURE BERS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE

Estimated entropy: ≈ 3.1 *bits per letter (excluding spaces).*

Example 5 First-order word approximation

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE

Example 6 Second-order word approximation

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

The number of different contexts (or states of the Markov process) is given by $|\Sigma|^k$, where k is the order of the model, which clearly grows exponentially with the size of the alphabet. For example, in a 2-symbol alphabet, a finite-context model of depth 4 originates only $2^4 = 16$ states or different contexts. However, if the alphabet is of size 256 (for example, if we consider a gray-level image), then the number of contexts jumps to $256^4 = 4\,294\,967\,296$!

Often, the finite-context models are "learned" online, i.e., as the sequence is processed (note that the model can also be static, i.e., the probabilities are considered fixed during the operation of the model). In this case, since the probabilistic model is continuously adapting, after processing the first n symbols of x, the average number of bits associated with a order-k finite-context model

is given by

$$H_n = -\frac{1}{n}\sum_{i=1}^{n} \log P(x_i|x_{i-k}^{i-1}). \tag{1}$$

In the case of online learning of the finite-context model, a table (or some other more sophisticated data structure, such as a hash-table) is used to collect counts that represent the number of times that each symbol occurs in each context. For example, suppose that, in a certain moment, a binary ($|\Sigma| = 2$) source is modeled by an order-3 finite-context model represented by the table

x_{i-3}	x_{i-2}	x_{i-1}	n_0	n_1
0	0	0	10	25
0	0	1	4	12
0	1	0	15	2
0	1	1	3	4
1	0	0	34	78
1	0	1	21	5
1	1	0	17	9
1	1	1	0	22

where n_i indicates how many times symbol "i" occurred following that context. Therefore, in this case, we may estimate the probability that a symbol "0" follows the sequence "100" as being $34/(34 + 78) \approx 0.30$.

This way of estimating probabilities, based only on the relative frequencies of previously occurred events, suffers from the problem of assigning probability zero to events that were not (yet) seen. That would be the case, in this example, if we used the same approach to estimate the probability of having a "0" following a sequence of three or more "1s". How can we handle this problem?