Text transforms, Quantization

Coding Technology

Illés Horváth

2025/11/12

Reminder: character encodings

Character codings. Fixed length encoding, Shannon–Fano code, Huffman code. Arithmetic code. Adaptive Huffman code, sibling property.

Entropy as a measure of information. Theoretical lower bound.

Dictionary coders to find and compress longer term patterns. LZ77, LZ78.

Text transformations

Next we examine text transformations.

These are invertible transformations that map the original text into a text over the same alphabet.

They typically do not change the length of the original text (so no compression in itself), but make certain patterns in the original text easier to exploit by data compression algorithms.

Accordingly, text transformations are typically used in tandem with an actual data compression method that exploits the changed patterns/characteristics of the text.

(Once again, details depend highly on the type of the memory of the source.)

MTF transform

The MTF (move-to-front) transform is the following. In every step of the algorithm, we encode the next character by its position within the alphabet, and after coding, move that character to the front of the alphabet.

Example. Apply MTF transform to "abracadabra".
The source alphabet is $\mathcal{X}$ =(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r...)

abracadabra	1	abcdefghijklmnopqr...
abracadabra	1,2	bacdefghijklmnopqr...
abracadabra	1,2,18	rbacdefghijklmnopq...
abracadabra	1,2,18,3	arbcdefghijklmnopq...
abracadabra	1,2,18,3,4	carbdefghijklmnopq...
abracadabra	1,2,18,3,4,2	acrbdefghijklmnopq...
abracadabra	1,2,18,3,4,2,5	dacrbefghijklmnopq...
abracadabra	1,2,18,3,4,2,5,2	adcrbefghijklmnopq...
abracadabra	1,2,18,3,4,2,5,2,5	badcrefghijklmnopq...
abracadabra	1,2,18,3,4,2,5,2,5,5	rbadcefghijklmnopq...
abracadabra	1,2,18,3,4,2,5,2,5,5,3	arbdcefghijklmnopq...

MTF transform

The MTF transform is trivial to invert (decode).

The MTF transform works best for sources where characters are likely to be repeated multiple times close to each other.

For such sources, the result of the transform is a code that has a lot of small numbers; then this can be followed by an adaptive Huffman code to exploit the high probability of small numbers.

Burrows–Wheeler transform

The Burrows–Wheeler transform works best for sources where certain strings occur often (e.g. in the English language: “is,” “and,” “the,” and so on).

For such sources, the result of the Burrows–Wheeler transform is a text that will have several places where the same character is repeated many times in a row (in other words, it has long runs of the same character repeated).

This makes it ideal to follow up the Burrows–Wheeler transform with MTF transform and finally Huffman coding.

Burrows–Wheeler transform

Example. We want to code "abraca". We actually need the alphabet to be sorted; for this example, we will use the alphabet $\mathcal{X} = (\text{a,b,c,r})$.

For the Burrows–Wheeler transform, we take all cyclic shifted versions of the message, and sort them lexicographically:

$$\begin{array}{l} \text{abraca} \\ \text{bracaa} \\ \text{racaab} \\ \text{acaabr} \\ \text{caabra} \\ \text{aabrac} \end{array} \quad \rightarrow \quad \begin{array}{l} \text{aabrac} \\ \text{abraca} \\ \text{acaabr} \\ \text{bracaa} \\ \text{caabra} \\ \text{racaab} \end{array}$$

Then the transformed text is the last column from top to bottom: "caraab," and the number of the row which contains the original message: 2 (so technically this transform actually increases the length of the text).

Burrows–Wheeler transform

Inverting the Burrows–Wheeler transform:

aabrac
abraca ←
acaabr
bracaa
caabra
racaab

The first column is the last column, sorted lexigraphically.

Then the last column is prepended to the first column and sorted alphabetically and matched to the first column to get the second column. Then the last column is prepended to the first two columns, sorted and matched to the first two columns, and so on.

We repeat this through all columns, then at the end, we pick the row which contains the original message.

Burrows–Wheeler transform

Why does the Burrows–Wheeler transform create long runs of the same character?

Example. Consider a text that contains the word “the” several times. Cyclic permutations where an occurrence of “the” is split so “he” is at the front and “t” is at the end will typically be consecutive because they are very close in lexicographic order.

Irreversible data compression

Next we examine data compression methods where the original data is not fully decodable. These are collectively called *lossy* or *irreversible* data compression (as opposed to lossless or reversible data compression).

Typical applications include voice, music, picture or video compression, where details can be removed as long as they do not affect human perception.

In some cases, even noticeable loss of quality can be acceptable in exchange for better compression rate.

Further applications include medical image compression, where only details relevant to medical diagnosis are important (diagnostically acceptable image compression, DAIC).

Irreversible data compression

For irreversible data compression, decompressed (reconstructed) data can be different from the original message.

A measure for the difference between the original message and reconstructed data is called *distortion*.

Defining a "good" distortion function for a specific application is a difficult task; it can depend on the details of human perception involved. (Example: the human eye has higher resolution for grayscale images than for red or blue.)

Irreversible data compression

Another relevant parameter for irreversible data compression is *compression ratio*, the ratio of the amount of the original data to the amount of the compressed data (e.g. 10:1).

Irreversible data compression has two goals:

- compression ratio should be as low as possible;
- distortion should be as low as possible.

The two goals work against each other. This is typically managed by setting one of the goal functions to a prescribed value, and then optimizing for the other function.

Quantization

Quantization is the process of rounding values originating from a larger set to values from a smaller set. It is used extensively in signal processing and also in irreversible data compression.

A general setup for quantization is the following. For a real-valued random signal X, its quantized version is $Q(X)$, where Q is an $\mathbb{R} \to \mathbb{R}$ function with a finite range of values.

The function Q is known as a *quantizer*.

Distortion of a quantization is usually measured by the *squared error distortion*

$$D(Q) = \mathbb{E}((X - Q(X))^2).$$

Quantization

A quantizer can be given by the following information:

- its finite range of values $\{x_1, \ldots, x_N\}$, which are referred to as *quantization levels*, and
- the sets $B_i = \{x : Q(x) = x_i\}$, referred to as *quantization domains*.

Generally, the distortion of a quantizer decreases when more quantization levels are allowed.

For a given number of allowed quantization levels N, and a given signal X, the optimal quantizer Q is the quantizer with minimal distortion $D(Q)$.

We want an algorithm to find Q. Inputs of the algorithm: the number of allowed quantization levels N, and the distribution of X.

Quantization

Assume the quantization levels are prescribed to $x_1, \ldots, x_N$. Can we tell what the quantization domains should be?

Example. Let Q be a quantizer with quantization levels $\{0, 2, 4\}$.

To which level should we quantize $x = -1.3$?

And $x = 0.7$?

The quantization domains should be

$$B_1 = (-\infty, 1] \qquad B_2 = (1, 3] \qquad B_3 = (3, \infty)$$

(For ties, e.g. $x = 1$, it can be included in either B_1 or B_2.)

In general, from among quantizers with prescribed levels $x_1, \ldots, x_N$, the optimal is the one with

$$B_i = \{x : |x - x_i| \leq |x - x_j| \text{ for any } j \neq i\}.$$

This is called the *nearest neighbour rule*.

Quantization

Assume now that the quantization domains $B_1, \ldots, B_N$ are prescribed. What should the levels be?

We are going to use a little trick from probability theory.

Lemma (Steiner's identity)

For any random variable X and constant c,

$$\mathbb{E}((X-c)^2) = \mathbb{D}^2(X) + (\mathbb{E}(X)-c)^2.$$

Proof.

$$\mathbb{E}((X-c)^2) = \mathbb{E}((X-\mathbb{E}(X)+\mathbb{E}(X)-c)^2) =$$

$$\underbrace{\mathbb{E}((X-\mathbb{E}(X))^2)}_{\mathbb{D}^2(X)} + 2\underbrace{\mathbb{E}((X-\mathbb{E}(X))(\mathbb{E}(X)-c))}_{0} + \underbrace{\mathbb{E}((\mathbb{E}(X)-c)^2)}_{(\mathbb{E}(X)-c)^2}$$

Quantization

Assume now that the quantization domains $B_1, \ldots, B_N$ are prescribed. What should the levels be?

Using Steiner's identity to the conditional distribution of X assuming $X \in B_i$,

$$\mathbb{E}((X - x_i)^2 | X \in B_i) = \mathbb{D}^2(X | X \in B_i) + (\mathbb{E}(X | X \in B_i) - x_i)^2.$$

This is minimal when $E(X | X \in B_i) - x_i = 0$, that is,

$$x_i = \mathbb{E}(X | X \in B_i).$$

This is called the *center of gravity rule*.

Side note: if X is continuous with probability density function $f(x)$, then $\mathbb{E}(X | X \in B_i) = \frac{\int_{B_i} x f(x) \mathrm{d}x}{\int_{B_i} f(x) \mathrm{d}x}$

Lloyd-Max property

A quantizer with levels $x_1 < \cdots < x_N$ and domains $B_1, \ldots, B_N$ satisfies the *Lloyd-Max property* if

- the boundary between the domains B_i and B_{i+1} is $y_i = \frac{x_i + x_{i+1}}{2}$, and
- every level is the center of gravity within its domain, that is, $x_i = \mathbb{E}(X | X \in B_i)$.

According to the previous remarks, an optimal quantizer must satisfy the Lloyd-Max property.

Is the reverse true? That is, is it true that if a quantizer satisfies the Lloyd-Max property, it must be optimal?

Quantization

Example. Let X be a signal with distribution

$$P(X = 1) = P(X = 2) = P(X = 3) = P(X = 4) = 1/4.$$

Find all 2-level quantizers for X that satisfy the Lloyd-Max property.

One option is to set

$$B_1 = \{1\}, \qquad B_2 = \{2, 3, 4\},$$

and then

$$x_1 = 1, \qquad x_2 = 3.$$

We will call this quantizer Q^1; equivalently,

$$Q^1(1) = 1, \qquad Q^1(2) = Q^1(3) = Q^1(4) = 3.$$

The distortion of this quantizer is

$$D(Q^1) = \frac{1}{4}(1-1)^2 + \frac{1}{4}(2-3)^2 + \frac{1}{4}(3-3)^2 + \frac{1}{4}(4-3)^2 = 0.5$$

Quantization

Another option, called Q^2 is

$$B_1 = \{1, 2, 3\}, \qquad B_2 = \{4\}, \qquad x_1 = 2, \qquad x_2 = 4;$$

This also has $D(Q^2) = 0.5$.

The last option called Q^3 is

$$B_1 = \{1, 2\}, \qquad B_2 = \{3, 4\}, \qquad x_1 = 1.5, \qquad x_2 = 3.5,$$

which has $D(Q^3) = 0.25$.

Q^1, Q^2 and Q^3 all have the Lloyd-Max property, but only Q^3 is optimal.

Lloyd-Max algorithm

The Lloyd-Max algorithm (also known as Lloyd's algorithm) aims to approximate the optimal quantization iteratively.

1. Initialize the levels.
2. Compute the domains for the levels using the nearest neighbour property.
3. Compute $D(Q)$, compare it to the old value of $D(Q)$ from the previous iteration, and if the difference is below a given threshold, stop.
4. Otherwise, from the domains, compute new values for the levels according to the center of gravity rule, then repeat from step 2.

The value of $D(Q)$ decreases in every step, so the algorithm will stop after a finite number of iterations.

Quantization

The Lloyd-Max algorithm is a numerical algorithm in the sense that it will give a close approximation of the optimal quantization.

Computing the optimal quantization explicitly is typically a difficult problem; however, for specific distributions and small values of N, it can be computed.

Example. Let X be uniform over the interval $[0, 1]$. Compute the optimal 2-level quantizer.

If $y \in [0, 1]$ denotes the boundary between B_1 and B_2, then, due to the center of gravity rule, the two levels are

$$x_1 = y/2, \qquad x_2 = (1 + y)/2.$$

Quantization

Then

$$D(Q) = \int_0^y (y/2 - x)^2 \mathrm{d}x + \int_y^1 ((1+y)/2 - x)^2 \mathrm{d}x$$

which gives

$$D(Q) = \frac{y^3}{4} - \frac{y}{4} + \frac{1}{12}.$$

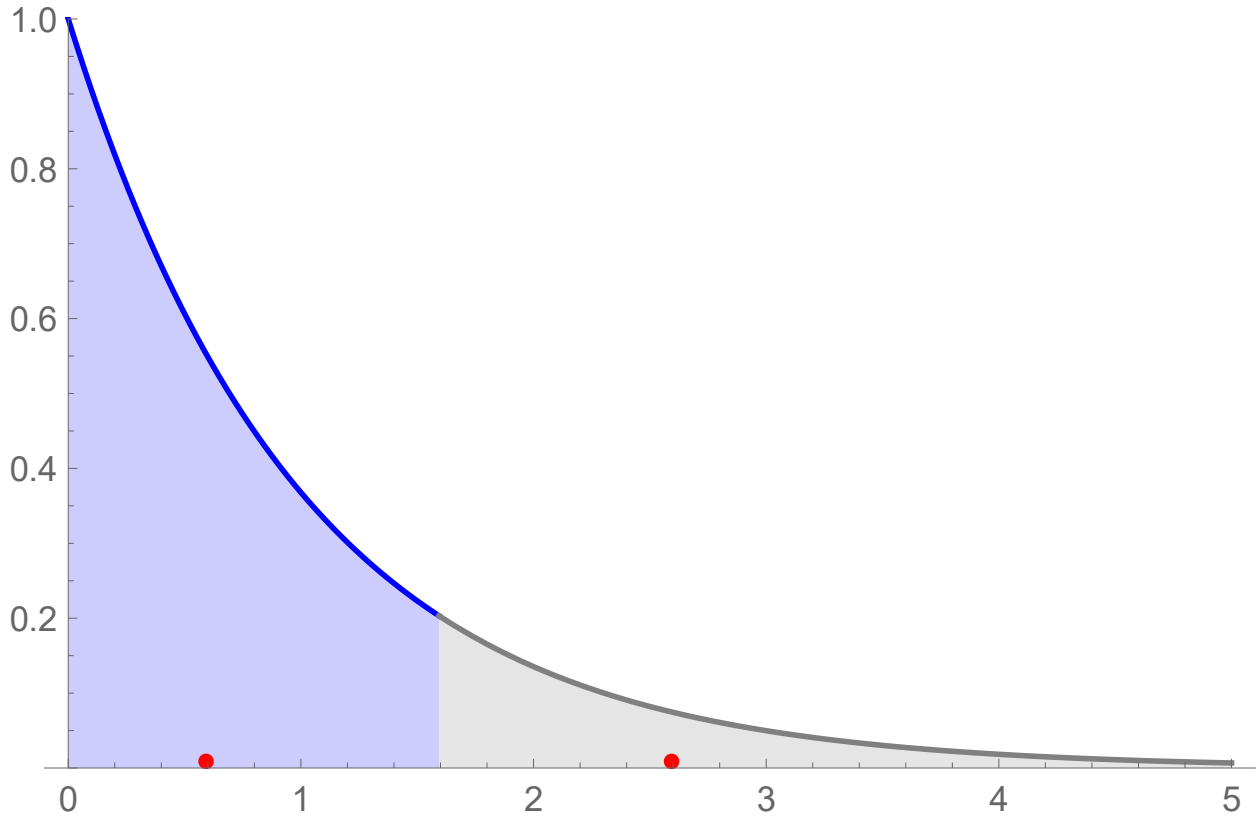
$$\frac{\mathrm{d}}{\mathrm{d}y} D(Q) = \frac{y}{2} - \frac{1}{4} = 0 \quad \rightarrow \quad y = 1/2,$$

and due to $\frac{\mathrm{d}^2}{\mathrm{d}y^2} D(Q) = 1/2$, this is a minimum, so $y = 1/2$.

Due to the center of gravity rule, $x_1 = 0.25, x_2 = 0.75$.

Quantization

Example. The optimal 2-level quantization of the $X \sim \mathrm{EXP}(1)$ distribution ($x_1 \approx 0.594, x_2 \approx 2.594$):



Quantization

Remarks.

When N is very low (2 or 3, maybe up to 5 in special cases), the optimal quantization may be possible to compute explicitly.

Otherwise, use the Lloyd-Max algorithm. It converges reasonably fast.

For continuous signals X, the optimal quantization typically assigns more levels where the probability density function is higher.

Despite that, the probabilities of each domain are typically not equal; domains with low density typically have a lower probability. This is because the distances are higher, and squared error distortion takes into account both distance and density.