Near-lossless compression

I. Something about distances from a mathematical point of view

In mathematics a distance function on a given set M is a function $d : M \times M \rightarrow \mathbf{R}$, where **R** is a set of real numbers, which satisfies the following conditions:

1. Non-negativity

$d(x,y) \geq 0$, and $d(x,y) = 0$ if and only if $x = y$.

The distance between any two points is not negative and if the distance is 0 the points are identical (x and y refers the same point).

2. Simetry

$d(x,y) = d(y,x)$.

The distance between two points is the same whatever the direction is.

3. Triangle inequality

$d(x,z) \leq d(x,y) + d(y,z)$

The distance between two points is the shortest distance along any path.

Such a distance function is known as a metric. Together with the set, it makes up a metric space.

II. Some usual distances

In the following we will consider the vector space R^n and two vectors x and y respectively.

$x = x_1, x_2, ..., x_n$

$y = y_1, y_2, ..., y_n$

1. Minkovski distance of order p

In R^n vector space one of the most important distance is the **Minkovski distance of order p** defined as**:**

$$L_p = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p}$$

p has not to be integer but must be no less than 1 in order to meet the triangle inequality.

If we consider some cases for p we get some well-known and important distances:

2. Classic Euclidian distance

It is the case for p=2.

$$L_2 = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^2 \right)^{1/2} = \sqrt{\sum_{i=1}^{n} \left(x_i - y_i \right)^2}$$

Intuitively it represents the distance measured directly with a tool between the points.

3. Manhattan distance (City Block)

It is the case for p=1.

$$L_1 = \sum_{i=1}^{n} \left| x_i - y_i \right|$$

Intuitively it represents the distance walked in Manhattan between two points (There we have a network of parallel streets on N-S and E-W directions. In the „site language" it represents how many blocks we have to walk from one point to another, for example „walk five blocks").

4. Cebîsev distance

It is the case for p= ∞

$$L_\infty = \lim_{p->\infty} \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p} = \max_i \left| x_i - y_i \right|$$

Intuitively it represents the distance that a king has to walk on a chess board to move from a square into another.

Cases with p different to 1, 2 and ∞ are very rarely used (are not interesting).

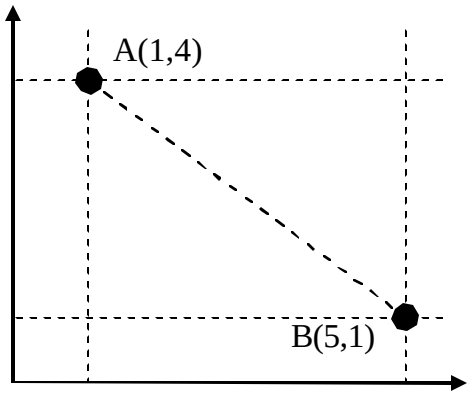

Figure 1 Example for distance evaluation

In Figure 1 we consider the two-dimensional space and two points A and B (two vectors). By evaluating the previous formulas we get for the different distances:

$L_2(A,B) = 5$

$L_1(A,B) = 7$!!!! :)

$L_\infty(A,B) = 4$

Other distances of interest in computer science are **Hamming distance** (defined as the number of components of the vector that differs) and the cosinus distance (defined as the cosinus of the angle of the vectors).

III. Near-lossless compression – Introduction

Today, in the age of computers and communication, there is an obvious need for data compression and, especially, for image compression. Because the compression methods where no error is accepted (called lossless) gives poor results on images, we have to accept some error in the decompressed image in order to get much higher compression ratios. Those methods (called lossy) become very popular especially with the multimedia development.

In the classic approach the criteria that is minimized by the lossy image compression methods is the one given by L_2, the classical Euclidian distance between the original image (to be compressed) and the restored image (decompressed). Here we were interested in the error at **entire image level**.

Another class of applications is the one where it is very important to get an upper limit on the error at each pixel. Now the criterion to be minimized is the Cebîsev distance L_∞. Here we are interested in the

error at **pixel level**.

The first approach (based on L_2) is the classic one and a lot of research has been done during years on that topic. The most popular method is the one based on DCT and stated in the JPEG standard. The main drawback of the scheme is that we cannot have any guarantee about the error at pixel level. In some applications (satellite images, medical images, etc.), there is a need for some guarantees at pixel level because the artifacts introduced by the compression methods are unacceptable (without an upper limit), especially for further automatic processing. Therefore, in such area image compression was very rarely accepted (or not accepted at all).

In the last years the interest for such compression methods (based on L_∞) is growing quickly and, therefore, the area requires a lot of research. Because usually we are interested only in small values for the acceptable error limit (±1, ±2, …, ±10) the method is called **near-lossless**. Obviously, if the acceptable error level becomes 0 we get the classical lossless compression.

It is very interesting to observe that such a small change, only in distance measure, puts us in front of a new area, all the old results (based on L_2) being now irrelevant (for L_∞). It is proven that the task of finding the best representation (and compression) within the acceptable error ±k is a NP-complete task.

IV. Predictive architecture for near-lossless compression

Near-lossless compression allows a compression with some loss but without loss of the details. In the predictive image compression case the near-lossless feature is usually obtained by **quantization of the prediction error**.

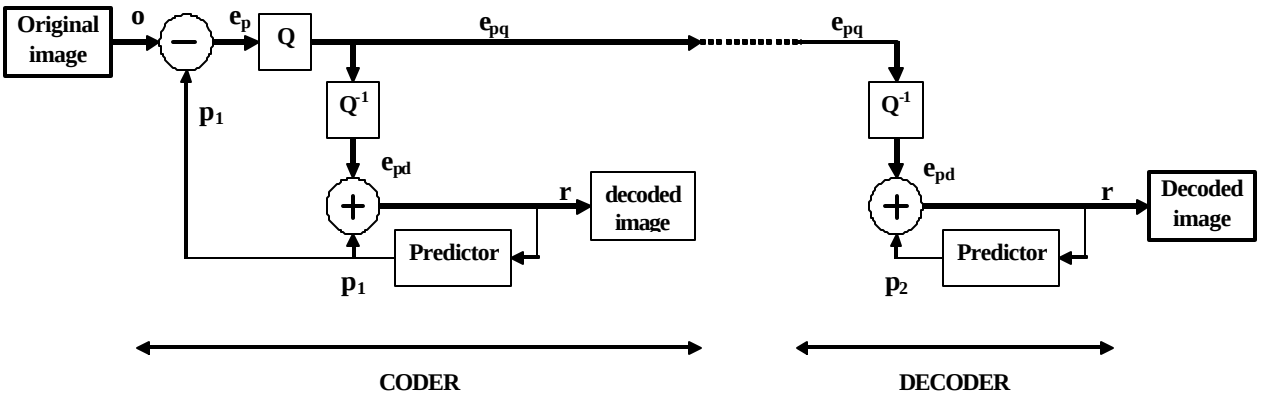

Figure 2. The near-lossless predictive architecture

Most of the lossless predictive compression methods have also a near-lossless case, with the compression algorithm almost the same. The only difference between them is the introduction of quantization before transmitting the error values to the entropy coder. All the considerations from the predictive image compression are still valid (considerations related to the pixels on which prediction can be based on, to the effect of a good or bad prediction on the compression performances, etc.).

In Figure 2 we present the typical near-lossless predictive architecture (the entropy coder and decoder applied to the e_{pq} is not figured). With the notations from the figure we have the following relations:

$$e_p = o - p_1 \quad \Rightarrow o = e_p + p_1$$

$$r = e_{pd} + p_2$$

Then we get:

$$r - o = (p_2 - p_1) + (e_{pd} - e_p)$$

If we make $p_2 = p_1$ (prediction to be identical at the coder and at the decoder) we notice that the error introduced by coding-decoding equals the error introduced by the quantization–dequantization process.

The most popular quantization–dequantization method uses the following rule:

$$\dot{x} = Q^{-1}(Q(x)) = \left\lfloor \frac{x+k}{2\times k+1} \right\rfloor (2\times k+1) \qquad (1)$$

where x is the prediction error, k is the maximum reconstruction error accepted for each pixel and $\lfloor\ \rfloor$ represents rounding to the largest integer smaller or equal. The coder generates the value according to:

$$Q(x) = \left\lfloor \frac{x+k}{2\times k+1} \right\rfloor \qquad (2)$$

This value is sent to the decoder and there the prediction error is reconstructed according to

$$Q^{-1}(Q(x)) = Q(x)\times(2\times k+1) \qquad (3)$$

Because prediction has to be the same at the coder and the decoder and, in the near-lossless case, we have differences between original image (available at the coder) and the decoded image (available at the decoder) the coder builds also a **copy of the decoded image** (in the same way as the decoder). **Prediction** is done **based on the decoded image** both at the coder and the decoder, therefore getting identical predictions (Figure 2). As we previously prove the error in the decoded image is exactly the

Original value		Quantized value		Dequantized value	Error
. . . . .		. . . . .		. . . . .	
7		1		5	-2
6		1		5	-1
5	=>	1	=>	5	0
4		1		5	1
3		1		5	2
2		0		0	-2
1		0		0	-1
0	=>	0	=>	0	0
-1		0		0	1
-2		0		0	2
-3		-1		-5	-2
-4		-1		-5	-1
-5	=>	-1	=>	-5	0
-6		-1		-5	1
-7		-1		-5	2
. . . . .		. . . . .		. . . . .	

Table 1. Quantization examples (for k=2)

error from the quantization process, no error accumulation occurs. In Table 1 we present some examples for quantization and dequantization (for k=2) and we can notice that the error introduced by quantization-dequantization falls in the +/- 2 range (the accepted near-lossless range).

V. Other simple near-lossless approaches

A way to achieve near-lossless compression is to quantize the image before a lossless coding. After quantization and mapping according to equation (2) the resulted image has a dynamic range significantly reduced and therefore will be better lossless compressed compared to the initial one. The near-lossless reconstructed image can be obtained by lossless decoding and remapping according to equation (3).

Another way to get near-lossless compression is in conjunction with a lossy method. It is realized as a two steps technique, one lossy and one near-lossless. First, a lossy version is realized and transmitted. After that, additional, a near-lossless version of the remaining differences is transmitted. Best results are usually obtained for strong initial lossy compression.