

LECTURE 21:

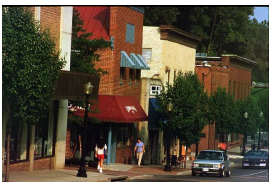
LOSSY COMPRESSION & RATE-DISTORTION

November 28, 2005

- Any lossy compression scheme is based (at least implicitly) on some idea of what counts as “close to the original”.
- This is a question that can only be answered by considering the *users* of the compression program, and what they want.
- For images and audio signals, two fundamental issues are:
 - What differences can humans perceive?
For example, to a first approximation, humans perceive only *relative energies* of different frequency bands in audio but not the associated *phases* of sine waves.
 - What differences do humans find annoying or distracting?
Slight changes in colour might be regarded as less important than making a straight line be jagged.

LOSSY COMPRESSION

- Many kinds of data — such as images and audio signals — contain “noise” and other information that is not really of interest. Preserving such useless information seems wasteful.
- **A common approach:** *Lossy compression*, for which decompressing a compressed file gives you something *close* to the original, but not necessarily exactly the original.
- We should be able to compress to a smaller file size if we don’t have to reproduce the original exactly.



FORMALIZING DISTORTION

- Suppose the input to the compression program is the sequence a_1, a_2, \dots, a_N , and the decompression program outputs the sequence b_1, b_2, \dots, b_N . (The a_i and the b_i might come from the same or different alphabets.)
- We can measure how close the decompressed output is to the original by its average “distortion”:

$$\bar{d} = \frac{1}{N} \sum_{i=1}^N d(a_i, b_i)$$

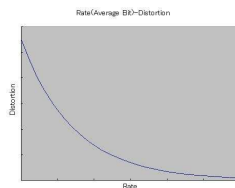
$d(a, b)$ is a non-negative *distortion function* measuring how bad it is for a decompressed symbol to be b if the original was a .

- **Note:** In practice, the overall distortion might not be a sum of distortions for individual symbols, but we’ll ignore that complication.

- Distortion functions that measure what we're really interested in are likely to be complicated. But we can consider some simple examples that are easier to handle.
- **Hamming distance:** For a bilevel image (such as a FAX), we might use a distortion function for which $d(0, 0) = d(1, 1) = 0$ and $d(0, 1) = d(1, 0) = 1$.
- **Squared error:** For a gray-scale image, with pixels values in $\{0, \dots, 255\}$, or for an audio signal with sample quantized to 8 or 16 bits, we might use $d(a, b) = (a - b)^2$.

- Suppose our source alphabet is binary, with equal probabilities for 0 and 1 (independently from symbol to symbol).
- Suppose we will decompress to the same alphabet, and that we measure distortion by Hamming distance.
- If we insist on lossless compression, we can't compress at all, since the entropy is one.
- How well can we compress if we allow an average distortion of up to $1/8$ — ie, if we allow up to one in eight bits to be wrong?

- If the entropy of our source is H , we expect to be able to losslessly compress N symbols into NH bits — ie, at rate H .
- But what if decompression is allowed to produce any output that has average distortion less than some limit, D ?
- The *rate distortion function*, $R(D)$, tells us how well we can do then. It is the *smallest* rate (average bits per input symbol) for *any* compression scheme that has average distortion no greater than D .
- Note that $R(D)$ depends on both the source probabilities and on the distortion function chosen.



- Here's a scheme that compresses a binary source to $4/7$ of the original file size while altering only $1/8$ of the bits, on average:
 1. Grab the next 7 input bits from the source.
 2. Pretend these bits are received data from a $[7, 4]$ Hamming code in systematic form.
 3. "Decode" these 7 bits by the usual Hamming code procedure.
 4. Output the 4 systematic bits from this "decoded" codeword.
- To decompress, we take blocks of 4 bits and "encode" them in 7 bits the usual way.
- **Result:** Perfect reconstruction of the 7 bits $1/8$ of the time; one wrong bit $7/8$ of the time.

- Consider all channels, \mathcal{C} , with input alphabet $\{a_i\}$ and output alphabet $\{b_j\}$. Given the input probabilities that our source has, we can find for each such channel
 - Its mutual information, $I(\mathcal{A}, \mathcal{B})$.
 - The average distortion between the channel input and the resulting output.
- Shannon proved that the rate distortion function, $R(D)$, is equal to the minimum value for $I(\mathcal{A}, \mathcal{B})$ over all channels whose average distortion is no more than D .
- For a binary source where 0 has probability $p_0 \leq 1/2$, and where distortion is measured by Hamming distance, it turns out that

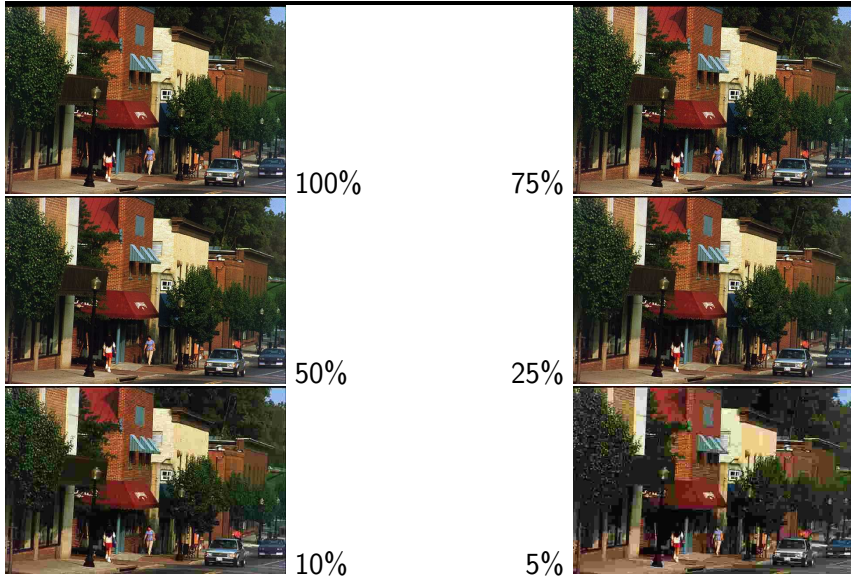
$$R(D) = \begin{cases} H(p_0) - H(D) & \text{for } 0 \leq D \leq p_0 \\ 0 & \text{for } D > p_0 \end{cases}$$

- Shannon's elegant theory currently plays little role in practical lossy data compression (or the similar task of "vector quantization").
- Instead, various *ad hoc* methods are used.
- Two reasons for this:
 1. Formalizing a suitable distortion function taking account of human perceptual abilities and tolerances is difficult.
 2. The step from the impractical random codes used to prove the rate distortion theorem to a practical method of optimal compression hasn't been achieved.
- Overcoming these issues is a current challenge for research.

- As for his noisy coding theorem, Shannon's rate distortion theorem can be proved using codes chosen at random.
- Consider a channel \mathcal{C} that minimizes $I(\mathcal{A}, \mathcal{B})$ subject to the distortion between input and output being less than D . We find the output probabilities for such channel, and then pick codewords at random with symbol probabilities equal to these output probabilities.
- **Encoding procedure:** Find the codeword closest (as measured by distortion) to the actual message; then send an index of that codeword. If we chose 2^K codewords, sending this index will take K bits, for a rate of K/N .
- **Decoding procedure:** Output the codeword corresponding to the received index.

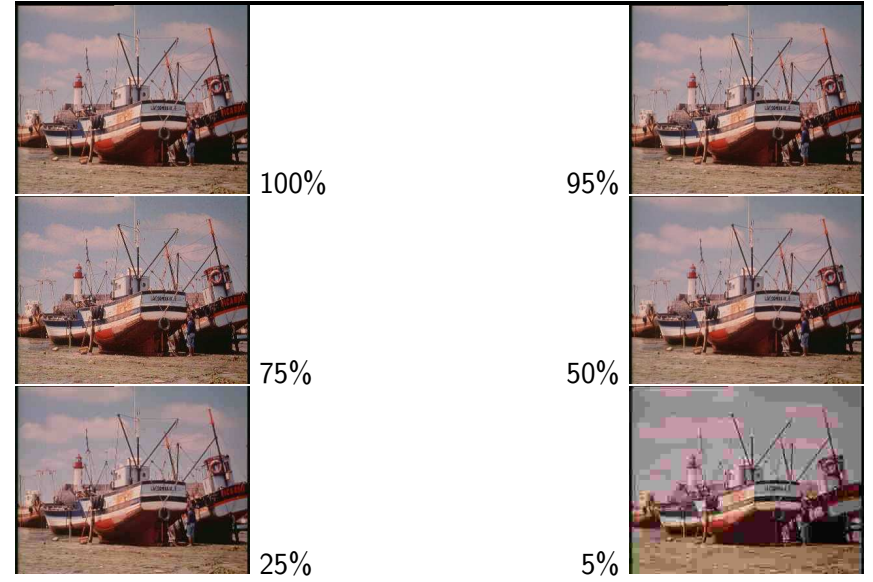
EXAMPLES – JPEG COMPRESSION

12



EXAMPLES – JPEG COMPRESSION

14



EXAMPLES – JPEG COMPRESSION

13

