Introduction to Source Coding/Data Compression

Course is about the Theory and Practice of Source Coding, a.k.a. Data Compression

Data compression is process of encoding data from some source into bits in such a way that it can be decoded back into a reproduction of the original data.

Source code = data compressor = data compression system = encoder + decoder

encoder creates bits, decoder creates reproduction from bits

Goals:
- efficiency: as few bits as possible
- accuracy, fidelity: reproduction as much like original as possible

Source is assumed to produce discrete-time samples or symbols
- e.g. text or samples of speech
- we won't spend significant time on sampling issues; just assume the source is already sampled
- source will be modelled as a random process, usually stationary and ergodic
- why assume random?? because if not, why encode it? because can exploit statistical characteristics (what occurs more frequently, what values, what combinations of values (correlation))
- autoregressive Gauss-Markov (AR) processes make nice tractable models of speech and image sources.
Rate is our measure of efficiency
rate = number of bits/sample
AVOID "compression ratio"
why encode into bits?? no big deal, just the most useful convention

Average Distortion wrt some distortion measure is our measure of fidelity
satisfactory human perception is usually the "ultimate" criteria
most commonly MSE

empirical distortion = $D = \frac{1}{n} \sum_{i=1}^{n} (X_i - Y_i)^2$

statistical distortion = $D = E \left[ \frac{1}{n} \sum_{i=1}^{n} (X_i - Y_i)^2 \right]$ or $E(X-Y)^2$

why MSE? pro's and cons
other distortion measures
usually empirical = statistical or else we're wasting our time with statistical

Summary: code performance on a given source is characterized by rate and distortion

<table>
<thead>
<tr>
<th>encoding rate $R$, bits/sample</th>
<th>distortion $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

Lossless coding is when $\hat{X}$ must equal $X$; i.e. $D = 0$

Lossy coding is the other case
course is $3/4$'s lossy, $1/4$ lossless, projects are mostly lossy, so we begin with lossy
**Complexity** is another big issue

implementation complexity

number of arithmetic operations per sample

bytes of auxiliary storage, e.g. for tables

these influence: building cost and operating cost

design complexity is a lesser issue

performance vs. complexity

![Graph showing distortion vs. encoding rate for low and high complexity methods.](image-url)
We're not concerned with **channel errors**

Though it's possible to build source codes that are terribly sensitive to channel errors, it is also possible to build them that are not. Any source code can be "fixed" so it is not too sensitive to errors, with only small loss. Typically, $p = 10^{-4}$ is small enough for speech and images, and even $10^{-3}$.

Shannon says that an optimum communication system can have a separation of source code and channel code.

But there are many situations where we're just storing data and the storage medium is so reliable that it doesn't make sense to model it as a noisy channel.

Working through lossy coding (quantization) with channel errors makes interesting exercises.
Source Coding Issues

1. Sources (skip or skim)
   - **discrete valued** -- English text, binary images (e.g. FAX) -- produce symbols
   - **continuous valued** -- speech, images, audio, video, etc. -- produce samples or pixels
   - **source models** -- for design and analysis
     - some methods don't require source models
     - we find Gaussian autoregressive, ARMA and Markov especially easy to deal with
     - also IID, stationary memoryless

2. Performance Measures (skip or skim)
   - Rate (not much question)
   - Distortion -- MSE
     - lossless vs. lossy
       - lossless for discrete-valued only

3. Code Structure: we will consider a variety of such
   - **independently code each sample/symbol**
   - **dependent coding** -- block, sliding-window, finite-state, predictive, feedback, adaptive, linear transform, waveform or spectral domain
   - **fixed-rate** -- constant number of bits produced per symbol/sample
   - **variable-rate** -- variable number of bits produced per symbol/sample

4. Code design to optimize performance of certain type of code.
   - Generic Question 1: How to optimize a given type of code?
5. **Complexity/Cost** of implementation. (skip or skim)

   performance does not mean speed of implementation in this course

   arithmetic -- number of ops/sample
   storage -- number of bits of auxiliary storage required
   building cost -- cost of building or buying hardware for computing and storing
   running cost -- cost of operating (depreciation, power, heat, rental, or sharing of resources)

   **Tradeoffs:** rate, distortion, and complexity.

6. **Analysis**

   to predict performance of specific types of codes and to **predict how** to optimize them and to **identify key characteristics** of good codes.

   to predict **best possible performance of any type of code** and to understand basic properties of optimal codes

   Generic Question 2: What is the best possible performance attainable with a given type of code.

   In this class we use mostly **asymptotic quantization theory** for lossy coding and **entropy theory** for lossless coding. There will be a brief overview of **rate-distortion theory**, a branch of information theory

   **difficult question:** how does complexity reducing structure limit performance??

   **theory is lacking**
## Typical Examples of Lossy Compression

<table>
<thead>
<tr>
<th>Source</th>
<th>Uncompressed</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech</td>
<td>64 Kbps</td>
<td>9.6K bps (CELP)</td>
</tr>
<tr>
<td>B/W Images</td>
<td>8 bpp</td>
<td>1 bpp (JPEG)</td>
</tr>
<tr>
<td>Color Images</td>
<td>24 bpp</td>
<td>1.25 bits/pixel</td>
</tr>
<tr>
<td>Video</td>
<td>100M bps</td>
<td>.01-20M bps</td>
</tr>
<tr>
<td>Audio</td>
<td>1.4M bps</td>
<td>256K bps (MPEG)</td>
</tr>
</tbody>
</table>

## Typical Example of Lossless Compression

<table>
<thead>
<tr>
<th>English Text</th>
<th>7 bits/symbol</th>
<th>3 bits/symbol</th>
</tr>
</thead>
</table>
Our Syllabus

Review it

Course is theory and practice

3/4's lossy, 1/4 explicitly lossless, but more lossless embedded in discussion of lossy

Theory and practice

Quite separate in source coding for a long while
theorists knew little of practice and vice versa
now there's some merging
there's some practical theory
and techniques that are theoretically analyzable (or were theoretically proposed) are being used in practice
there's nothing so practical as a good theory
we'll cover both
But we won't entirely avoid theory that has no practice and vice versa