15-853: Algorithms in the Real World

Data Compression 4
Compression Outline

Introduction: Lossy vs. Lossless, Benchmarks, ...
Information Theory: Entropy, etc.
Probability Coding: Huffman + Arithmetic Coding
Applications of Probability Coding: PPM + others
Lempel-Ziv Algorithms: LZ77, gzip, compress, ...
Other Lossless Algorithms: Burrows-Wheeler
Lossy algorithms for images: JPEG, MPEG, ...
  - Scalar and vector quantization
  - JPEG and MPEG
Compressing graphs and meshes: BBK
Scalar Quantization

Quantize regions of values into a single value:

output

input

uniform

output

input

non uniform

Can be used to reduce # of bits for a pixel
Vector Quantization

Encode

Generate Vector

Find closest code vector

Codebook Index

Generate Output

Index Codebook

Decode
Vector Quantization

What do we use as vectors?

• Color (Red, Green, Blue)
  - Can be used, for example to reduce 24bits/pixel to 8bits/pixel
  - Used in some terminals to reduce data rate from the CPU (colormaps)

• K consecutive samples in audio

• Block of K pixels in an image

How do we decide on a codebook

• Typically done with clustering
Vector Quantization: Example

Weight

Height

1' 2' 3' 4' 5' 6' 7' 8'
Linear Transform Coding

Want to encode values over a region of time or space
- Typically used for images or audio

Select a set of linear basis functions $\phi_i$ that span the space
- sin, cos, spherical harmonics, wavelets, ...
- Defined at discrete points
Linear Transform Coding

Coefficients: \( \Theta_i = \sum_j x_j \phi_i(j) = \sum_j x_j a_{ij} \)

\( \Theta_i = \ i^{th} \ \text{resulting coefficient} \)

\( x_j = \ j^{th} \ \text{input value} \)

\( a_{ij} = \ ij^{th} \ \text{transform coefficient} = \phi_i(j) \)

In matrix notation:

\( \Theta = Ax \)

\( x = A^{-1} \Theta \)

Where \( A \) is an \( n \times n \) matrix, and each row defines a basis function
Example: Cosine Transform

\[ \theta_i = \sum_j x_j \phi_i(j) \]
Other Transforms

Polynomial:

Wavelet (Haar):
How to Pick a Transform

Goals:
- Decorrelate
- Low coefficients for many terms
- Basis functions that can be ignored by perception

Why is using a Cosine of Fourier transform across a whole image bad?
How might we fix this?
Usefulness of Transform

Typically transforms $A$ are orthonormal.

Properties of orthonormal transforms:

$$\sum x^2 = \sum \Theta^2$$  (energy conservation, Parseval’s theorem)

Would like to compact energy into as few coefficients as possible

$$G_{TC} = \frac{1}{n} \sum \sigma_i^2 \left( \prod \sigma_i^2 \right)^{-1/n}$$  (the transform coding gain)

arithmetic mean/geometric mean

$$\sigma_i = (\Theta_i - \Theta_{av})$$

The higher the gain, the better the compression
Case Study: JPEG

A nice example since it uses many techniques:
- Transform coding (Cosine transform)
- Scalar quantization
- Difference coding
- Run-length coding
- Huffman or arithmetic coding

JPEG (Joint Photographic Experts Group) was designed in 1991 for lossy and lossless compression of color or grayscale images. The lossless version is rarely used.

Can be adjusted for compression ratio (typically 10:1)
JPEG in a Nutshell
### JPEG: Quantization Table

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>11</td>
<td>10</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>51</td>
<td>61</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>26</td>
<td>58</td>
<td>60</td>
<td>55</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>57</td>
<td>69</td>
<td>56</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>22</td>
<td>29</td>
<td>51</td>
<td>87</td>
<td>80</td>
<td>62</td>
</tr>
<tr>
<td>18</td>
<td>22</td>
<td>37</td>
<td>56</td>
<td>68</td>
<td>109</td>
<td>103</td>
<td>77</td>
</tr>
<tr>
<td>24</td>
<td>35</td>
<td>55</td>
<td>64</td>
<td>81</td>
<td>104</td>
<td>113</td>
<td>92</td>
</tr>
<tr>
<td>49</td>
<td>64</td>
<td>78</td>
<td>87</td>
<td>103</td>
<td>121</td>
<td>120</td>
<td>101</td>
</tr>
<tr>
<td>72</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>112</td>
<td>100</td>
<td>103</td>
<td>99</td>
</tr>
</tbody>
</table>

Also divided through uniformaly by a quality factor which is under control.
JPEG: Block scanning order

Uses run-length coding for sequences of zeros
JPEG: example

.125 bits/pixel (factor of 200)
Case Study: MPEG

Pretty much JPEG with interframe coding

Three types of frames

- **I** = intra frame (aprox. JPEG) anchors
- **P** = predictive coded frames
- **B** = bidirectionally predictive coded frames

Example:

<table>
<thead>
<tr>
<th>Type</th>
<th>I</th>
<th>B</th>
<th>B</th>
<th>P</th>
<th>B</th>
<th>B</th>
<th>P</th>
<th>B</th>
<th>B</th>
<th>P</th>
<th>B</th>
<th>B</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>12</td>
<td>13</td>
<td>11</td>
</tr>
</tbody>
</table>

**I** frames are used for random access.
MPEG matching between frames
MPEG: Compression Ratio

356 x 240 image

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>18KB</td>
<td>7/1</td>
</tr>
<tr>
<td>P</td>
<td>6KB</td>
<td>20/1</td>
</tr>
<tr>
<td>B</td>
<td>2.5KB</td>
<td>50/1</td>
</tr>
<tr>
<td>Average</td>
<td>4.8KB</td>
<td>27/1</td>
</tr>
</tbody>
</table>

30 frames/sec x 4.8KB/frame x 8 bits/byte
= 1.2 Mbits/sec + .25 Mbits/sec (stereo audio)

HDTV has 15x more pixels
= 18 Mbits/sec
MPEG in the “real world”

- DVDs
  - Adds “encryption” and error correcting codes
- Direct broadcast satellite
- HDTV standard
  - Adds error correcting code on top
- Storage Tech “Media Vault”
  - Stores 25,000 movies

Encoding is much more expensive than encoding. Still requires special purpose hardware for high resolution and good compression. Now available on some processors or using GPUs.
H.264 (video)

Uses similar ideas, but

- 4x4 transform that is an approximation of the cosine transform
- More sophisticated prediction of blocks based on their neighbors
Wavelet Compression

• A set of localized basis functions
• Avoids the need to block

“mother function” $\varphi(x)$

$$\varphi_{sl}(x) = \varphi(2^s x - l)$$

$s = \text{scale} \quad l = \text{location}$

Requirements

$$\int_{-\infty}^{\infty} \varphi(x) dx = 0 \quad \text{and} \quad \int_{-\infty}^{\infty} |\varphi(x)|^2 dx < \infty$$

Many mother functions have been suggested.


**Haar Wavelets**

Most described, least used.

\[
H_{k0} = \phi(2^s x - 1)
\]

\[
\phi(x) = \begin{cases} 
1 & 0 \leq x < 1/2 \\
-1 & 1/2 \leq x < 1 \\
0 & \text{otherwise}
\end{cases}
\]

+ DC component = \(2^{k+1}\) components
Haar Wavelet in 2d
Discrete Haar Wavelet Transform

How do we convert this to the wavelet coefficients?
Discrete Haar Wavelet Transform

for (j = n/2; j >= 1; j = j/2) {
    for (i = 1; i < j; i++) {
        b[i] = (a[2i-1] + a[2i])/2;
        b[j+i] = (a[2i-1] - a[2i])/2;
    }
    a[1..2*j] = b[1..2*j]; }

How do we convert this to the wavelet coefficients?

Linear time!
Haar Wavelet Transform: example

\[ a = 2 \quad 1 \quad 2 \quad -1 \quad -2 \quad 0 \quad 2 \quad -2 \]

\[ = 1.5 \quad 0.5 \quad -1 \quad 0 \quad 0.5 \quad 1.5 \quad -1 \quad 2 \]

\[ = 1 \quad -0.5 \quad 0.5 \quad -0.5 \]

\[ = 0.25 \quad 0.75 \]

\[ a = 0.25 \quad 0.75 \quad 0.5 \quad 0.5 \quad 0.5 \quad 1.5 \quad -1 \quad 2 \]
Wavelet decomposition
**Morlet Wavelet**

\[ \phi(x) = \text{Gaussian} \times \text{Cosine} = e^{-\left(\frac{x^2}{2}\right)} \cos(5x) \]

Corresponds to wavepackets in physics.
Daubechies Wavelet
JPEG2000

Overall Goals:
- High compression efficiency with good quality at compression ratios of .25bpp
- Handle large images (up to $2^{32} \times 2^{32}$)
- Progressive image transmission
  - Quality, resolution or region of interest
- Fast access to various points in compressed stream
- Pan and Zoom while only decompressing parts
- Error resilience

Also used in Dirac video compression
JPEG2000: Outline

Main similarities with JPEG
• Separates into Y, I, Q color planes, and can downsample the I and Q planes
• Transform coding

Main differences with JPEG
• Wavelet transform
  - Daubechies 9-tap/7-tap (irreversible)
  - Daubechies 5-tap/3-tap (reversible)
• Many levels of hierarchy (resolution and spatial)
• Only arithmetic coding
JPEG2000: 5-tap/3-tap

\[ h[i] = a[2i-1] - \frac{(a[2i] + a[2i-2])}{2}; \]
\[ l[i] = a[2i] + \frac{(h[i-1] + h[i] + 2)}{2}; \]

- \( h[i] \): is the "high pass" filter, ie, the **differences**
  - it depends on 3 values from \( a \) (3-tap)
- \( l[i] \): is the "low pass" filter, ie, the **averages**
  - it depends on 5 values from \( a \) (5-tap)

Need to deal with boundary effects.
This is reversible: assignment
JPEG 2000: Outline

A spatial and resolution hierarchy

- **Tiles**: Makes it easy to decode sections of an image. For our purposes we can imagine the whole image as one tile.

- **Resolution Levels**: These are based on the wavelet transform. High-detail vs. Low detail.

- **Precinct Partitions**: Used within each resolution level to represent a region of space.

- **Code Blocks**: blocks within a precinct

- **Bit Planes**: ordering of significance of the bits
JPEG2000: Precincts
JPEG vs. JPEG2000

JPEG: .125bpp
JPEG2000: .125bpp
Compression Summary

Compression is all about probabilities

We want the model to skew the probabilities as much as possible (i.e., decrease the entropy)
Compression Summary

How do we figure out the probabilities
- Transformations that skew them
  • Guess value and code difference
  • Move to front for temporal locality
  • Run-length
  • Linear transforms (Cosine, Wavelet)
  • Renumber (graph compression)
- Conditional probabilities
  • Neighboring context

In practice one almost always uses a combination of techniques