Announcements:
• HW2 will be released tomorrow Oct 16 (Wed)
• Due on Oct 25 (Fri) noon
• There will be lectures on Oct 29 and 31. Please update your calendars.
• HW1 grades will be released in a day or two

Today:
Data Compression Cont...
Move onto Hashing
Recap:

PPM: Using Conditional Probabilities

Makes use of conditional probabilities

- Use previous $k$ characters as context.

Builds a context table

Each context has its own probability distribution
Recap: Lempel-Ziv Algorithms

Dictionary-based approach
Codes groups of characters at a time (unlike PPM)

High level idea:
- Look for longest match in the preceding text for the string starting at the current position
- Output the position of that string
- Move past the match
- Repeat

Gets theoretically optimal compression for (really) long strings
Recap: Burrows - Wheeler

Breaks file into fixed-size blocks and encodes each block separately.

For each block:
- Create full context for each character (wraps around)
- Reverse lexical sort each character by its full context.

Then use move-to-front transform on the sorted characters.
Recap: Burrows-Wheeler

<table>
<thead>
<tr>
<th>Context</th>
<th>Char</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecode_6</td>
<td>d_1</td>
<td>o_4</td>
</tr>
<tr>
<td>coded_1</td>
<td>e_2</td>
<td>c_3</td>
</tr>
<tr>
<td>odede_2</td>
<td>c_3</td>
<td>d_5</td>
</tr>
<tr>
<td>dedec_3</td>
<td>o_4</td>
<td>e_6</td>
</tr>
<tr>
<td>edeco_4</td>
<td>d_5</td>
<td>e_6</td>
</tr>
<tr>
<td>decod_5</td>
<td>e_6</td>
<td>d_1 ←</td>
</tr>
</tbody>
</table>

Sort Context

Gets similar characters together
(because we are ordering by context)

Can be viewed as giving a dynamically sized context.
(overcoming the problem of choosing the right “k” in PPM)
Recap: Inverting BW Transform

<table>
<thead>
<tr>
<th>Context</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₃</td>
<td>o₄</td>
</tr>
<tr>
<td>d₁</td>
<td>e₂</td>
</tr>
<tr>
<td>d₅</td>
<td>e₆</td>
</tr>
<tr>
<td>e₂</td>
<td>c₃</td>
</tr>
<tr>
<td>e₆</td>
<td>d₁</td>
</tr>
<tr>
<td>o₄</td>
<td>d₅</td>
</tr>
</tbody>
</table>

Sort the output column to get the last column of the context!

**Theorem:** After sorting, equal valued characters appear in the same order in the output column as in the last column of the sorted context.
Invert

Answer: cabbaa

Can also use the “rank”.
The “rank” is the position of a character if it were sorted using a stable sort.

<table>
<thead>
<tr>
<th>Context</th>
<th>Output</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>a c</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>a a</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>a b</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>b b</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>b a</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>c a</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
Inverting BW Transform

Function BW_Decode(In, Start, n)
    $S = \text{MoveToFrontDecode}(\text{In}, n)$
    $R = \text{Rank}(S)$
    $j = \text{Start}$
    for $i = 1$ to $n$ do
        Out[$i$] = $S[j]$
        $j = R[j]$

(Rank gives position of each char in sorted order.)
BZIP

Transform 1: (Burrows Wheeler)
- input: character string (block)
- output: reordered character string

Transform 2: (move to front)
- input: character string
- output: MTF numbering

Transform 3: (run length)
- input: MTF numbering
- output: sequence of run lengths

Probabilities: (on run lengths)
Dynamic based on counts for each block.

Coding: Originally arithmetic, but changed to Huffman in bzip2 due to patent concerns
Overview of Text Compression

PPM and Burrows-Wheeler both encode a single character based on the immediately preceding context.

LZ77 and LZ78 encode multiple characters based on matches found in a block of preceding text.

Can you mix these ideas, i.e., code multiple characters based on immediately preceding context?

– BZ, ACB,..
Compression Outline

**Introduction**: Lossy vs. Lossless, prefix codes, ...

**Information Theory**: Entropy, bounds on length, ...

**Probability Coding**: Huffman, Arithmetic Coding

**Applications of Probability Coding**: Run-length, Move-to-front, Residual, PPM

**Lempel-Ziv Algorithms**:
- LZ77, gzip,
- LZ78, compress (Not covered in class)

**Other Lossless Algorithms**:
- Burrows-Wheeler

**Lossy algorithms for images**: Quantization, JPEG, MPEG, Wavelet compression ...
Scalar Quantization

Quantize regions of values into a single value
E.g. Drop least significant bit

(Can be used to reduce # of bits for a pixel)

Q: Why is this lossy?
Many-to-one mapping

Two types
– Uniform: Mapping is linear
– Non-uniform: Mapping is non-linear
Scalar Quantization

Q: Why use non-uniform?
Error metric might be non-uniform.
E.g. Human eye sensitivity to specific color regions

Can formalize the mapping problem as an optimization problem
Vector Quantization

Mapping a multi-dimensional space into a smaller set of messages

- Generate Vector
- Find closest code vector
- Encode
- Generate Output
- Decode
Vector Quantization

What do we use as vectors?

- Color (Red, Green, Blue)
  - Can be used, for example to reduce 24 bits/pixel to 8 bits/pixel
  - Used in some monitors 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

VQ most effective when the variables along the dimensions of the space are correlated
Vector Quantization: Example

Observations:

1. Highly correlated: Concentration of representative points

2. Higher density is more common regions.
Linear Transform Coding

Goal: Transform the data into a form that is easily compressible (through lossless or lossy compression)

Select a set of linear basis functions $f_i$ that span the space

– sin, cos, spherical harmonics, wavelets, …
Linear Transform Coding

Coefficients:

\[ \Theta_i = \sum_j x_j \phi_i(j) = \sum_j x_j a_{ij} \]

- \( \Theta_i \) = \( i^{th} \) resulting coefficient
- \( x_j \) = \( j^{th} \) input value
- \( a_{ij} \) = \( ij^{th} \) transform coefficient = \( i(j) \)

In matrix notation:

\[ \Theta = Ax \]

\[ x = A^{-1} \Theta \]

Where A is an n x n “transform” 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 the data
– Low coefficients for many terms
– Basis functions that can be ignored from the perception point-of-view
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.
15-853: Algorithms in the Real World

Announcements:
• HW2 will be released tomorrow Oct 16 (Wed)
• Due on Oct 25 (Fri) noon
• There will be lectures on Oct 29 and 31. Please update your calendars.
• HW1 grades will be released in a day or two
• Start thinking about projects. Will mention briefly at towards the end of class

Today:
Data Compression Cont...
Move onto Hashing
JPEG in a Nutshell
Also divided through uniformly by a quality factor which is under "user" control.
JPEG

DC component and higher frequencies (i.e., AC) coded separately

DC components are residual encoded: “difference encoded”

AC components are RLE
   Using a zig-zag scanning order to keep similar frequencies together

Then finally either Huffman or Arithmetic coding is used
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 anchors
  - Encoded as individual pictures
  - Used for random access.

- **P** = predictive coded frames
  - Encoded based on previous I- or P- frames

- **B** = bidirectionally predictive coded frames
  - Encoded based on either or both the previous and next I- or P- frames
Case Study: MPEG

Pretty much JPEG with **interframe coding**

Three types of frames
- **I** = intra frame 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>
MPEG matching between frames

Finding motion vectors is the most computationally intensive part
Video compression in the “real world”

- Cisco estimates that video will grow to 82% of all consumer internet traffic by 2021
  - Efficient compression of videos is crucial to support such traffic

- MPEG:
  - DVDs (adds “encryption” and error correcting codes)
  - Direct broadcast satellite
  - HDTV standard (adds error correcting code on top)
Video compression in the “real world”

Encoding is much more expensive than decoding.
Q: Why?

Still requires special purpose hardware for high resolution and good compression.
• Now available on some processors or using GPUs.

Most phones today have special purpose hardware for video compression
• System-on-chip (SoC) for video encoding and decoding
Compression Outline

Introduction: Lossy vs. Lossless, prefix codes, ...

Information Theory: Entropy, bounds on length, ...

Probability Coding: Huffman, Arithmetic Coding

Applications of Probability Coding: Run-length, Move-to-front, Residual, PPM

Lempel-Ziv Algorithms:
  – LZ77, gzip,
  – LZ78, compress (Not covered in class)

Other Lossless Algorithms:
  – Burrows-Wheeler

Lossy algorithms for images: Quantization, JPEG, MPEG, Wavelet compression ...
Wavelet Compression

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

Forming basis functions:
- Start with a "mother function" \( \varphi(x) \)
  - Localized
- Scale and translate to form other basis functions
  \[ \varphi_{sl}(x) = \varphi(2^s x - l) \]
  
  \( s = \) scale \hspace{1cm} \( l = \) location
Wavelet Compression

Forming basis functions:
• Start with a “mother function” \( \varphi(x) \)
• Scale and translate to form other basis functions
  \[ \varphi_{sl}(x) = \varphi(2^s x - l) \]
  \( s = \text{scale} \quad l = \text{location} \)

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.

\[ H_{00} \]

\[ H_{10} \]

\[ H_{20} \]

\[ H_{k0} \]

+ DC component

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

\[ H_{sl}(x) = \varphi(2^s x - 1) \]
Haar Wavelet in 2d
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: Outline

• Separates into Y, I, Q color planes, and can downsample the I and Q planes

• Wavelet transform coding
  – Daubechies 9-tap/7-tap (irreversible)
  – Daubechies 5-tap/3-tap (reversible)

• Many levels of hierarchy (resolution and spatial)

• Arithmetic coding
JPEG vs. JPEG2000

JPEG: .125bpp

JPEG2000: .125bpp
(Daubechies wavelet)
15-853: Algorithms in the Real World

Hashing:

Concentration bounds

Load balancing: balls and bins
Concentration Bounds

Central question:
What is the probability that a R.V. deviates much from its expected value
  – Typically want to say a R.V. stays “close to” its expectation “most of the time”

Useful in analysis of randomized algorithms
Markov’s Inequality

The most basic concentration bound.

Let $X$ be a non-negative R.V. with mean $\mu$ then

$$P(X \geq \alpha) \leq \frac{\mu}{\alpha}$$

Proof: Ideas?

<board>