Differential Encoding

C.M. Liu
Perceptual Signal Processing Lab
College of Computer Science
National Chiao-Tung University

http://www.csie.nctu.edu.tw/~cmliu/Courses/Compression/

Office: EC538
(03)5731877
cmliu@cs.nctu.edu.tw
Idea

- Reducing the dynamic range/variance of coded sequence by encoding sample differences

**Sinusoid and sample-to-sample differences.**
Example Image Histogram

99% ∈ [-31, 31] → 5 bits/pixel (or less)
Basic Algorithm

- Consider sequence:
  6.2 9.7 13.2 5.9 8 7.4 4.2 1.8

- Differences:
  6.2 3.5 3.5 -7.3 2.1 -0.6 -3.2 -2.4

- Lossless encoding
  - Simply coding the difference is sufficient to recover original

- Lossy encoding
  - Quantizer: -6 -4 -2 0 2 4 6
  - Quantized sequence: 6 4 4 -6 2 0 -4 -2
  - ‘Lossless’ reconstruction: 6 10 14 8 10 10 6 4
  - QE: 0.2 -0.3 -0.8 -2.1 -2 -2.6 -1.8 -2.2
  - Observation:
    - QE seems to grow over time—is it a coincidence?
Consider:

\[ \{x_n\} \text{ and } d_n = x_n - x_{n-1} \]

\[ \hat{d}_n = Q[d_n] = d_n + q_n \]

\[ \hat{x}_n = \hat{x}_{n-1} + \hat{d}_n, \quad \hat{x}_0 = x_0 \]

\[ d_1 = x_1 - x_0 \]

\[ \hat{d}_1 = Q[d_1] = d_1 + q_1 \]

\[ \hat{x}_1 = x_0 + \hat{d}_0 = x_0 + d_1 + q_1 = x_1 + q_1 \]

\[ d_2 = x_2 - x_1 \]

\[ \hat{d}_2 = Q[d_2] = d_2 + q_2 \]

\[ \hat{x}_2 = x_1 + \hat{d}_2 = x_1 + q_1 + d_1 + q_2 = x_2 + q_1 + q_2 \]

\[ \Rightarrow \hat{x}_n = x_n + \sum_{k=1}^{n} q_k \]
Basic Algorithm (3)

- Alternative coding:

\[ d_n = x_n - \hat{x}_{n-1} \]

\[
\begin{align*}
    d_1 &= x_1 - x_0 \\
    \hat{d}_1 &= Q[d_1] = d_1 + q_1 \\
    \hat{x}_1 &= x_0 + \hat{d}_0 = x_0 + d_1 + q_1 = x_1 + q_1 \\
    d_2 &= x_2 - \hat{x}_1 \\
    \hat{d}_2 &= Q[d_2] = d_2 + q_2 \\
    \hat{x}_2 &= \hat{x}_1 + \hat{d}_2 = \hat{x}_1 + d_2 + q_2 = x_2 + q_2 \\
\end{align*}
\]

\[ \Rightarrow \quad \hat{x}_n = x_n + q_n \]
Basic Algorithm: Example
Differential Encoding Scheme

Encoder

Decoder
Differential Pulse Code Modulation (DPCM)

$$p_n = f(\hat{x}_{n-1}, \hat{x}_{n-2}, \ldots, \hat{x}_0)$$
Prediction in DPCM

\[ \sigma_d^2 = E \left[ (x_n - p_n)^2 \right] \]

- Choice of \( f(\cdot) \) affects \( \sigma_d \), however

\[ \hat{x}_n = x_n + q_n \]

where \( q_n \) depends on the variance of \( d_n \)

- Dependencies:

\[ f(\cdot) \rightarrow \sigma_d^2 \rightarrow \hat{x}_n \rightarrow f(\cdot) \]

- Fine Quantization Assumption:
  Granularity is fine enough so that \( \hat{x}_n \approx x_n \)

Thus,

\[ p_n = f(x_{n-1}, x_{n-2}, \ldots, x_0) \]
Linear Predictor

\[ p_n = \sum_{i=1}^{N} a_i \hat{x}_{n-i} \]

\( N \) is called order of the predictor

Find \( \{a_i\} \): minimize

\[ \sigma_d^2 = E \left( x_n - \sum_{i=1}^{N} a_i x_{n-i} \right)^2 \]

\[ \frac{\partial \sigma_d^2}{\partial a_1} = -2E \left[ (x_n - \sum_{i=1}^{N} a_i x_{n-i}) x_{n-1} \right] = 0 \]

\[ \vdots \]

\[ \frac{\partial \sigma_d^2}{\partial a_N} = -2E \left[ (x_n - \sum_{i=1}^{N} a_i x_{n-i}) x_{n-N} \right] = 0 \]
Linear Predictor (2)

\[
\begin{align*}
\sum_{i=1}^{N} a_i R_{xx}(i - 1) &= R_{xx}(1) \\
\sum_{i=1}^{N} a_i R_{xx}(i - 2) &= R_{xx}(2) \\
&\vdots \\
\sum_{i=1}^{N} a_i R_{xx}(i - N) &= R_{xx}(N)
\end{align*}
\]

where \( R_{xx} \) is the autocorrelation function:

\[
R_{xx}(k) = E[x_n x_{n+k}]
\]
Linear Predictor (3)

\[ R = \begin{bmatrix}
R_{xx}(0) & R_{xx}(1) & \cdots & R_{xx}(N-1) \\
R_{xx}(1) & R_{xx}(0) & \cdots & R_{xx}(N-2) \\
\vdots & \vdots & \ddots & \vdots \\
R_{xx}(N-1) & R_{xx}(N-2) & \cdots & R_{xx}(0)
\end{bmatrix} \]

\[ A = \begin{bmatrix}
a_1 \\
\vdots \\
a_n
\end{bmatrix} \quad P = \begin{bmatrix}
R_{xx}(1) \\
\vdots \\
R_{xx}(N)
\end{bmatrix} \]

\[ RA = P \quad \Rightarrow \quad A = R^{-1}P. \]
Linear Predictor Example: Speech
Linear Predictor Example (2)

\[
R_{xx}(k) = \frac{1}{M - k} \sum_{i=1}^{M-k} x_i x_{i+k}
\]

- \(N = 1 \Rightarrow a_1 = 0.66\)
- \(N = 2 \Rightarrow a_1 = 0.596, a_2 = 0.096\)
- \(N = 3 \Rightarrow a_1 = 0.577, a_2 = -0.025, a_3 = 0.204\)
Linear Predictor Example:
Laplacian Quantization

- **Uniform Step sizes**
  - **4-level**: 1\(^{st}\) order: 0.75, 2\(^{nd}\) order: 0.59, 3\(^{rd}\) order: 0.43
  - **8-level**: 1\(^{st}\) order: 0.3, 2\(^{nd}\) order: 0.4, 3\(^{rd}\) order: 0.5

\[
\text{SNR (dB)} = \frac{\sum_{i=1}^{M} x_i^2}{\sum_{i=1}^{M} (x_i - \hat{x}_i)^2}
\]

\[
\text{SPER (dB)} = \frac{\sum_{i=1}^{M} x_i^2}{\sum_{i=1}^{M} (x_i - p_i)^2}
\]

\[\text{Prediction Error}\]
Linear Predictor Example: Performance

- SNR increases a lot for order 1 to order 2.

<table>
<thead>
<tr>
<th>Quantizer</th>
<th>Predictor Order</th>
<th>SNR (dB)</th>
<th>SPER (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Four-level</td>
<td>None</td>
<td>2.43</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.37</td>
<td>2.65</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>8.35</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8.74</td>
<td>6.1</td>
</tr>
<tr>
<td>Eight-level</td>
<td>None</td>
<td>3.65</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.87</td>
<td>2.74</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9.81</td>
<td>6.37</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>10.16</td>
<td>6.71</td>
</tr>
</tbody>
</table>
Although the reconstructed sequence looks like the original, notice that there is significant distortion in areas where the source output values are small.
Adaptive DPCM

- Motivation
  - Even after DPCM, a lot of structure remains in the signal
  - Structure → more compression is possible

Residuals for 3rd-order predictor
Approaches Adaptive DPCM

- Adaptation can be applied to
  - Quantization
  - Prediction
- Observation
  - Quantization adaptation is independent of prediction
  - Prediction adaptation → quantization adaptation
    - Good prediction depends on good quantization
Adaptive Quantization in DPCM

- **Forward adaptation**
  - Parameters are estimated for each block
  - Transmitted to receiver
  - Overall, this is inconvenient in DPCM as parameters are not explicitly available (due to feedback loop)

- **Backward adaptation**
  - Essentially, a version of the Jayant quantizer
  - Example:
    - 8-level quantizer, 3rd-order predictor
    - $M_0 = 0.9, M_1 = 0.9, M_2 = 1.25, M_3 = 1.75
Example: Adaptive *Jayant* DPCM

![Original](image1)

![Jayant](image2)

![Non-adaptive](image3)
Forward Adaptive Prediction: DPCM-APF

- **Speech coding**
  - 8000 sample/sec, 128 samples/block (16ms)

- **Image coding**
  - 8x8 blocks

- **Autocorrelation coefficients**
  - Assuming samples are zero outside block. \( l \) means the \( l \)th block.

\[
R_{xx}^{(l)}(k) = \frac{1}{M-k} \sum_{i=(l-1)M-1}^{lM-k} x_i x_{i+k}, \text{ for } k > 0
\]

\[
R_{xx}^{(l)}(k) = \frac{1}{M-k} \sum_{i=(l-1)M+1}^{lM} x_i x_{i+k}, \text{ for } k < 0
\]

\[
R_{xx}^{(l)}(k) = R_{xx}^{(l)}(-k)
\]

\( R_{xx}(k) \) can be efficiently encoded using partial correlation (\textit{parcor}) coefficients.
Backward Adaptive Prediction: DPCM-APB

- **1\textsuperscript{st}-order predictor**
  \[ d_n^2 = (x_n - a_1 \hat{x}_{n-1})^2. \]
  - Adapts with sample
    \[ a_1^{(n+1)} = a_1^{(n)} - \alpha \frac{\partial d_n^2}{\partial a_1} = a_1^{(n)} + 2 \alpha d_n \hat{x}_{n-1} \]
  - Replacing \( d_n \) by \( \hat{d}_n \) to have the consistent result with decoder
    \[ a_1^{(n+1)} = a_1^{(n)} + \alpha \hat{d}_n \hat{x}_{n-1} \]

- **N\textsuperscript{th}-order predictor**
  \[ a_j^{(n+1)} = a_j^{(n)} - \alpha \hat{d}_n \hat{x}_{n-j} \]
  \[ A^{(n+1)} = A^{(n)} + \alpha \hat{d}_n \hat{X}_{n-j} \]

A.k.a. Least Mean Squared (LMS)
Delta Modulation (DM)

- DM = DPCM w/ 1-bit quantizer
- Sampling frequency
  - At least twice the highest frequency signal component
    - Usually, much higher
Linear DM Reconstruction
Constant Factor Adaptive DM (CFDM)

\[ s_n = \begin{cases} 
1 & \text{if } \hat{d}_n > 0 \\
-1 & \text{if } \hat{d}_n < 0 
\end{cases} \]

\[ \Delta_n = \begin{cases} 
M_1 \Delta_{n-1} & \text{if } s_n = s_{n-1} \\
M_2 \Delta_{n-1} & \text{if } s_n \neq s_{n-1} 
\end{cases} \]

\[ 1 < M = M_1 = 1/M_2 < 2 \]
Second-Order CFDM

- Examples for 2 samples prediction

\[
\begin{align*}
    s_n & \neq s_{n-1} = s_{n-2} & M_1 &= 0.4 \\
    s_n & \neq s_{n-1} \neq s_{n-2} & M_2 &= 0.9 \\
    s_n &= s_{n-1} \neq s_{n-2} & M_3 &= 1.5 \\
    s_n &= s_{n-1} = s_{n-2} & M_4 &= 2.0.
\end{align*}
\]
Autocorrelation function for speech sample

- Indicates a period of 47 samples
  - Pitch period
- Need a separate component to take advantage of it
DPCM with Pitch Predictor

From image:

- Also: **Noise Feedback Coding** (NFC)
  - Shaping of QE such that most falls in high-amplitude periods

\[ P_p : b\hat{x}_{n-\tau}, \tau = \text{pitch period} \]
DPCM with Pitch Predictor Performance

DPCM Residuals

DPCM w/ Pitch Predictor Residuals
G.726

- ITU recommendation for standard ADPCM
  - Supersedes G.721 & G.723
  - Rates: 40/32/24/16 kbits/sec
  - Compression w.r.t. 8-bit PCM:
    1.6:1, 2:1, 2.67:1, 4:1
  - Quantizer levels: $2^{nb}-1$
    ➔ midtread quantizer

- Backward adaptive quantization
  - A version of the Jayant quantizer
  - Described in terms of a scale factor $\alpha_k$
    - $Q[d_k/\alpha_k] * \alpha_k$
G.726 24kb Quantizer I/O Map

<table>
<thead>
<tr>
<th>Input Range</th>
<th>Label</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \log_2 \frac{d_k}{\alpha_k} )</td>
<td>(</td>
<td>I_k</td>
</tr>
<tr>
<td>[2.58, ( \infty ))</td>
<td>3</td>
<td>2.91</td>
</tr>
<tr>
<td>[1.70, 2.58)</td>
<td>2</td>
<td>2.13</td>
</tr>
<tr>
<td>[0.06, 1.70)</td>
<td>1</td>
<td>1.05</td>
</tr>
<tr>
<td>((-\infty, -0.06))</td>
<td>0</td>
<td>(-\infty)</td>
</tr>
</tbody>
</table>

Recommended input-output characteristics of the quantizer for 24-kbits-per-second operation.
G.726: Quantizer Adaptation

- Based on $y(k) = \log_2 \alpha_k$

- Two factors:
  - $y_u$ unlocked—to handle large fluctuations (e.g. speech)
  - $y_l$ locked—for small ones like data transmission.

$$y(k) = a_1(k) y_u(k-1) + (1 - a_1(k)) y_l(k-1)$$

$a_1$ depend on input variance: for speech it is close to 1

$$y_u(k) = (1 - \varepsilon) y(k-1) + \varepsilon W[I_{k-1}], \text{ where } W[.] = \log M[.], \varepsilon = 2^{-5}$$

$$y_l(k) = (1 - \gamma) y_l(k-1) + \gamma y_u(k), \quad \gamma = 2^{-6}$$
G.726: Predictor

- **Backward adaptable** based on
  - last 2 reconstructed values
  - last 6 quantized differences

\[
p_k = \sum_{i=1}^{2} a_i^{(k-1)} \hat{x}_{k-i} + \sum_{i=1}^{6} b_i^{(k-1)} \hat{d}_{k-i}
\]

- Simplified LMS:

\[
a_1^{(k)} = (1 - 2^{-8}) a_1^{(k-1)} + 3 \times 2^{-8} \text{sgn}[z(k)] \text{sgn}[z(k-1)]
\]

\[
a_2^{(k)} = (1 - 2^{-7}) a_2^{(k-1)} + 2^{-7} (\text{sgn}[z(k)] \text{sgn}[z(k-2)]
- f \left( a_1^{(k-1)} \text{sgn}[\hat{z}(k)] \text{sgn}[\hat{z}(k-1)] \right)
\]

\[
z(k) = \hat{d}_k + \sum_{i=1}^{6} b_i^{(k-1)} \hat{d}_{k-i}
\]

\[
f(\beta) = \begin{cases} 
4\beta & |\beta| \leq \frac{1}{2} \\
2 \text{sgn}(\beta) & |\beta| > \frac{1}{2}
\end{cases}
\]

\[
b_i^{(k)} = (1 - 2^{-8}) b_i^{(k-1)} + 2^{-7} \text{sgn}[\hat{d}_k] \text{sgn}[\hat{d}_{k-i}]
\]
Consider the predictor

\[
p[j, k] = \begin{cases} 
\hat{x}[j, k-1] & \text{for } k > 0 \\
\hat{x}[j-1, k] & \text{for } k = 0 \text{ and } j > 0 \\
128 & \text{for } j = 0 \text{ and } k = 0
\end{cases}
\]

... in combination with 2-bit uniform quantizer & AC

- 1 bit/pixel encoding—compare to JPEG at the same rate

**Diff coded**: SNR=22dB, PSNR=31dB

**JPEG**: SNR=33dB, PSNR=42dB
Differential Image Coding (2)

- Improved scheme
  - Recursively indexed quantizer
  - Improved predictor
    
    \[
    p_1 = 0.5 \times \hat{x}[j-1,k] + 0.5 \times \hat{x}[j,k-1]
    \]
    
    \[
    p_2 = 0.5 \times \hat{x}[j-1,k-1] + 0.5 \times \hat{x}[j,k-1]
    \]
    
    \[
    p_3 = 0.5 \times \hat{x}[j-1,k-1] + 0.5 \times \hat{x}[j-1,k]
    \]
    
    \[p[j,k] = \text{median}\{p_1, p_2, p_3\}.
    \]

Diff coded: SNR=29dB, PSNR=38dB

JPEG: SNR=33dB, PSNR=42dB
Remarks

- Prediction in DPCM
- Adaptive DPCM
- Delta Modulation
- Speech Coding
- Image Coding
Homeworks

- P. 352
  - 3, 4, 6