Speech Coding Techniques (I)

- Introduction to Quantization
  - Scalar quantization
  - Uniform quantization
  - Nonuniform quantization

- Waveform-based coding
  - Pulse Coded Modulation (PCM)
  - Differential PCM (DPCM)
  - Adaptive DPCM (ADPCM)

- Model-based coding
  - Channel vocoder
  - Analysis-by-Synthesis techniques
  - Harmonic vocoder
Origin of Speech Coding

“Watson, if I can get a mechanism which will make a current of electricity vary its intensity as the air varies in density when sound is passing through it, I can telegraph any sound, even the sound of speech.”

- A. G. Bell’1875

\[ H(X) = - \sum_{i=1}^{N} p_i \log_2 p_i \text{ (bits/sample) or bps} \]

Entropy formula of a discrete source

- C. E. Shannon’1948
Digitization of Speech Signals

The process involves:

- **Sampler**: Converts a continuous-time speech signal to a discrete sequence of speech samples.

  \[ x_c(t) \rightarrow \text{Sampler} \rightarrow x(n) = x_c(nT) \]

- **Quantizer**: Further processes the discrete sequence.

  \[ x(n) \rightarrow \text{Quantizer} \rightarrow x(n) \]

**Continuous-time speech signal**

**Discrete sequence speech samples**
Sampling

- Sampling Theorem
  - when sampling a signal (e.g., converting from an analog signal to digital), the sampling frequency must be greater than twice the bandwidth of the input signal in order to be able to reconstruct the original perfectly from the sampled version.

Sampling frequency: >8K samples/second
(human speech is roughly bandlimited at 4KHz)
Quantization

- In Physics
  - To limit the possible values of a magnitude or quantity to a discrete set of values by quantum mechanical rules

- In speech coding
  - To limit the possible values of a speech sample or prediction residue to a discrete set of values by information theoretic rules (tradeoff between Rate and Distortion)
Quantization Examples

Examples
- Continuous to discrete
  - a quarter of milk, two gallons of gas, normal temperature is 98.6°F, my height is 5 foot 9 inches
- Discrete to discrete
  - Round your tax return to integers
  - The mileage of my car is about 60K.

Play with bits
- Precision is finite: the more precise, the more bits you need (to resolve the uncertainty)
  - Keep a card in secret and ask your partner to guess. He/she can only ask Yes/No questions: is it bigger than 7? Is it less than 4? ...
- However, not every bit has the same impact
  - How much did you pay for your car? (two thousands vs. $2016.78)
Scalar vs. Vector Quantization

- **Scalar:** for a given sequence of speech samples, we will process (quantize) each sample independently.
  - Input: N samples → output: N codewords

- **Vector:** we will process (quantize) a block of speech samples each time.
  - Input: N samples → output: N/d codewords (block size is d)

- SQ is a special case of VQ (d=1)
Scalar Quantization

In SQ, quantizing N samples is not fundamentally different from quantizing one sample (since they are processed independently).

A quantizer is defined by codebook (collection of codewords) and mapping function (straightforward in the case of SQ)
Rate-Distortion Tradeoff

- **Rate**: How many codewords (bits) are used?
  - Example: 16-bit audio vs. 8-bit PCM speech

- **Distortion**: How much distortion is introduced?
  - Example: mean absolute difference ($L_1$), mean square error ($L_2$)

Question: which quantizer is better?
Uniform Quantization

A scalar quantization is called uniform quantization (UQ) if all its codewords are uniformly distributed (equally-distanced)

Example (quantization stepsize $\Delta = 16$)

Uniform Distribution denoted by $U[-A,A]$

$$f(x) = \begin{cases} \frac{1}{2A} & x \in [-A, A] \\ 0 & \text{else} \end{cases}$$
6dB/Bit Rule

Note: Quantization noise of UQ on uniform distribution is also uniformly distributed.

For a uniform source, adding one bit/sample can reduce MSE or increase SNR by 6dB.

(The derivation of this 6dB/bit rule will be given in the class.)
Nonuniform Quantization

- **Motivation**
  - Speech signals have the characteristic that *small-amplitude samples occur more frequently than large-amplitude ones*
  - Human auditory system exhibits a logarithmic sensitivity
    - More sensitive at small-amplitude range (e.g., 0 might sound different from 0.1)
    - Less sensitive at large-amplitude range (e.g., 0.7 might not sound different much from 0.8)

![histogram of typical speech signals]
From Uniform to Non-uniform

F: nonlinear compressing function  
F⁻¹: nonlinear expanding function

F and F⁻¹: nonlinear compander

Example  
F: y=log(x)  
F⁻¹: x=exp(x)

We will study nonuniform quantization by PCM example next
Speech Coding Techniques (I)

- Introduction to Quantization
  - Scalar quantization
  - Uniform quantization
  - Nonuniform quantization
- Waveform-based coding
  - Pulse Coded Modulation (PCM)
  - Differential PCM (DPCM)
  - Adaptive DPCM (ADPCM)
- Model-based coding
  - Channel vocoder
  - Analysis-by-Synthesis techniques
  - Harmonic vocoder
Pulse Code Modulation

- Basic idea: assign smaller quantization stepsize for small-amplitude regions and larger quantization stepsize for large-amplitude regions
- Two types of nonlinear compressing functions
  - Mu-law adopted by North American telecommunications systems
  - A-law adopted by European telecommunications systems
Mu-Law ($\mu$-law)

\[
F(x) = \text{sgn}(x) \frac{\ln(1 + \mu|x|)}{\ln(1 + \mu)} \quad -1 \leq x \leq 1.
\]

\[
F^{-1}(y) = \text{sgn}(y)\left(\frac{1}{\mu}\right)[(1 + \mu)^{|y|} - 1] \quad -1 \leq y \leq 1
\]
Mu-Law Examples
A-Law

\[ F(x) = \text{sgn}(x) \left\{ \begin{array}{ll}
\frac{Ax}{1+\ln(A)}, & 0 \leq |x| < \frac{1}{A} \\
\frac{1+\ln(|x|)}{1+\ln(A)}, & \frac{1}{A} \leq |x| \leq 1
\end{array} \right. \]

\[ F^{-1}(y) = \text{sgn}(y) \left\{ \begin{array}{ll}
\frac{y(1+\ln(A))}{A}, & 0 \leq |y| < \frac{1}{1+\ln(A)} \\
\exp(|y|(1+\ln(A))-1), & \frac{1}{1+\ln(A)} \leq |y| < 1
\end{array} \right. \]
A-Law Examples
Comparison

\[ A = 87.56 \]

\[ u = 255 \]
PCM Speech

- Mu-law (A-law) compresses the signal to 8 bits/sample or 64Kbits/second (without compandor, we would need 12 bits/sample)
A Look Inside WAV Format

MATLAB function \([x,fs]=\text{wavread}(\text{filename})\)
Change the Gear

- Strictly speaking, PCM is merely digitization of speech signals – no coding (compression) at all
- By speech coding, we refer to representing speech signals at the bit rate of <64Kbps
- To understand how speech coding techniques work, I will cover some basics of data compression
Data Compression Basics

- **Discrete source**
  - Information = uncertainty
  - Quantification of uncertainty
  - Source entropy

- **Variable length codes**
  - Motivation
  - Prefix condition
  - Huffman coding algorithm

- **Data compression = source modeling**
Shannon’s Picture on Communication (1948)

The goal of communication is to move information from here to there and from now to then

Examples of source:
- Human speeches, photos, text messages, computer programs …

Examples of channel:
- storage media, telephone lines, wireless network …
Information

- What do we mean by information?
  - “A numerical measure of the uncertainty of an experimental outcome” – Webster Dictionary

- How to quantitatively measure and represent information?
  - Shannon proposes a probabilistic approach

- How to achieve the goal of compression?
  - Represent different events by codewords with varying code-lengths
Information = Uncertainty

- Zero information
  - WVU lost to FSU in Gator Bowl 2005 (past news, no uncertainty)
  - Yao Ming plays for Houston Rocket (celebrity fact, no uncertainty)

- Little information
  - It is very cold in Chicago in winter time (not much uncertainty since it is known to most people)
  - Dozens of hurricanes form in Atlantic ocean every year (not much uncertainty since it is pretty much predictable)

- Large information
  - Hurricane xxx is going to hit Houston (since Katrina, we all know how difficult it is to predict the trajectory of hurricanes)
  - There will be an earthquake in LA around X’mas (are you sure? an unlikely event)
Quantifying Uncertainty of an Event

**Self-information**

\[ I(p) = -\log_2 p \]

\( p \) - probability of the event \( x \)

(e.g., \( x \) can be \( X=H \) or \( X=T \))

<table>
<thead>
<tr>
<th>( p )</th>
<th>( I(p) )</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>must happen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(no uncertainty)</td>
</tr>
<tr>
<td>0</td>
<td>( \infty )</td>
<td>unlikely to happen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(infinite amount of uncertainty)</td>
</tr>
</tbody>
</table>

Intuitively, \( I(p) \) measures the amount of uncertainty with event \( x \)
Discrete Source

- A discrete source is characterized by a discrete random variable $X$

- Examples
  - Coin flipping: $P(X=H)=P(X=T)=1/2$
  - Dice tossing: $P(X=k)=1/6$, $k=1-6$
  - Playing-card drawing: $P(X=S)=P(X=H)=P(X=D)=P(X=C)=1/4$

How to quantify the uncertainty of a discrete source?
Weighted Self-information

<table>
<thead>
<tr>
<th>$p$</th>
<th>$I(p)$</th>
<th>$I_w(p) = p \cdot I(p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\infty$</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

As $p$ evolves from 0 to 1, weighted self-information $I_w(p) = -p \cdot \log_2 p$ first increases and then decreases.

Question: Which value of $p$ maximizes $I_w(p)$?
Maximum of Weighted Self-information

\[ I_w(p) = \frac{1}{e \ln 2} \]

\[ p = \frac{1}{e} \]
Uncertainty of a Discrete Source

• A discrete source (random variable) is a collection (set) of individual events whose probabilities sum to 1
  
  $X$ is a discrete random variable

  $x \in \{1, 2, \ldots, N\}$

  \[ p_i = \text{prob}(x = i), \; i = 1, 2, \ldots, N \]

  \[ \sum_{i=1}^{N} p_i = 1 \]

• To quantify the uncertainty of a discrete source, we simply take the summation of weighted self-information over the whole set
Shannon’s Source Entropy Formula

\[ H(X) = \sum_{i=1}^{N} I_w(p_i) \]

or

\[ H(X) = - \sum_{i=1}^{N} p_i \log_2 p_i \text{ (bits/sample)} \]

or bps
Source Entropy Examples

• Example 1: (binary Bernoulli source)

  Flipping a coin with probability of head being $p$ ($0 < p < 1$)

  $$p = \text{prob}(x=0), \quad q = 1 - p = \text{prob}(x=1)$$

  $$H(X) = -(p \log_2 p + q \log_2 q)$$

  Check the two extreme cases:

  As $p$ goes to zero, $H(X)$ goes to 0 bps → compression gains the most

  As $p$ goes to a half, $H(X)$ goes to 1 bps → no compression can help
Entropy of Binary Bernoulli Source
Source Entropy Examples

• Example 2: (4-way random walk)

\[ \text{prob}(x=S) = \frac{1}{2}, \text{prob}(x=N) = \frac{1}{4}, \text{prob}(x=E) = \text{prob}(x=W) = \frac{1}{8} \]

\[ H(X) = -\left( \frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{4} \log_2 \frac{1}{4} + \frac{1}{8} \log_2 \frac{1}{8} + \frac{1}{8} \log_2 \frac{1}{8} \right) = 1.75 \text{ bps} \]
• Example 3: (source with geometric distribution)

A jar contains the same number of balls with two different colors: blue and red. Each time a ball is randomly picked out from the jar and then put back. Consider the event that at the $k$-th picking, it is the first time to see a red ball – what is the probability of such event?

$$p = \text{prob}(x=\text{red}) = \frac{1}{2}, 1 - p = \text{prob}(x=\text{blue}) = \frac{1}{2}$$

$$\text{Prob(event)} = \text{Prob(blue in the first k-1 picks)} \times \text{Prob(red in the k-th pick)}$$

$$= (1/2)^{k-1} \times (1/2) = (1/2)^k$$
Morse Code (1838)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.08</td>
<td>.01</td>
<td>.03</td>
<td>.04</td>
<td>.12</td>
<td>.02</td>
<td>.02</td>
<td>.06</td>
<td>.07</td>
<td>.00</td>
<td>.01</td>
<td>.04</td>
<td>.02</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N</th>
<th>O</th>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
<th>T</th>
<th>U</th>
<th>V</th>
<th>W</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>.07</td>
<td>.08</td>
<td>.02</td>
<td>.00</td>
<td>.06</td>
<td>.06</td>
<td>.09</td>
<td>.03</td>
<td>.01</td>
<td>.02</td>
<td>.00</td>
<td>.02</td>
<td>.00</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Average Length of Morse Codes

- **Not** the average of the lengths of the letters:
  \[
  \frac{2+4+4+3+\ldots}{26} = \frac{82}{26} \approx 3.2
  \]

- We want the average \( a \) to be such that in a *typical real* sequence of say 1,000,000 letters, the number of dots and dashes should be about \( a \cdot 1,000,000 \)

- The **weighted average**:
  \[
  \text{(freq of A)} \cdot \text{(length of code for A)} + \text{(freq of B)} \cdot \text{(length of code for B)} + \ldots
  \]
  \[
  = 0.08 \cdot 2 + 0.01 \cdot 4 + 0.03 \cdot 4 + 0.04 \cdot 3 + \ldots \approx 2.4
  \]

**Question:** is this the entropy of English texts?
Entropy Summary

- Self-information of an event \( x \) is defined as \( I(x) = -\log_2 p(x) \) (rare events \( \rightarrow \) large information)

- Entropy is defined as the weighted summation of self-information over all possible events

\[
H(X) = - \sum_{i=1}^{N} p_i \log_2 p_i \quad \text{(bits/sample) or bps}
\]

For any discrete source

\[
\sum_{i=1}^{N} p_i = 1 \quad 0 \leq p_i \leq 1
\]
How to achieve source entropy?

Note: The above entropy coding problem is based on simplified assumptions are that discrete source X is memoryless and P(X) is completely known. Those assumptions often do not hold for real-world data such as speech and we will recheck them later.
Data Compression Basics

- Discrete source
  - Information = uncertainty
  - Quantification of uncertainty
  - Source entropy
- Variable length codes
  - Motivation
  - Prefix condition
  - Huffman coding algorithm
- Data Compression = source modeling
**Variable Length Codes (VLC)**

**Recall:**

Self-information \( I(p) = -\log_2 p \)

It follows from the above formula that a small-probability event contains much information and therefore worth many bits to represent it. Conversely, if some event frequently occurs, it is probably a good idea to use as few bits as possible to represent it. Such observation leads to the idea of varying the code lengths based on the events’ probabilities.

Assign a **long** codeword to an event with **small** probability

Assign a **short** codeword to an event with **large** probability
## 4-way Random Walk Example

<table>
<thead>
<tr>
<th>symbol $k$</th>
<th>$p_k$</th>
<th>fixed-length codeword</th>
<th>variable-length codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>0.5</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>$N$</td>
<td>0.25</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>$E$</td>
<td>0.125</td>
<td>10</td>
<td>110</td>
</tr>
<tr>
<td>$W$</td>
<td>0.125</td>
<td>11</td>
<td>111</td>
</tr>
</tbody>
</table>

symbol stream : \[ S \ S \ N \ W \ S \ E \ N \ N \ N \ W \ S \ S \ S \ N \ E \ S \ S \]

fixed length: \[ 00 \ 00 \ 01 \ 11 \ 00 \ 10 \ 01 \ 01 \ 11 \ 00 \ 00 \ 00 \ 01 \ 10 \ 00 \ 00 \] 32bits

variable length: \[ 0 \ 0 \ 10 \ 111 \ 0 \ 110 \ 10 \ 10 \ 111 \ 0 \ 0 \ 0 \ 10 \ 110 \ 0 \ 0 \] 28bits

4 bits savings achieved by VLC (redundancy eliminated)
Toy Example (Con’t)

• source entropy:

\[
H(X) = -\sum_{k=1}^{4} p_k \log_2 p_k \\
= 0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 \\
= 1.75 \text{ bits/symbol}
\]

• average code length:

\[
\bar{l} = \frac{N_b}{N_s} \quad \text{(bps)}
\]

Total number of bits
Total number of symbols

\[
\text{fixed-length} \quad \bar{l} = 2 \text{ bps} > H(X) \\
\text{variable-length} \quad \bar{l} = 1.75 \text{ bps} = H(X)
\]
Problems with VLC

- When codewords have fixed lengths, the boundary of codewords is always identifiable.
- For codewords with variable lengths, their boundary could become ambiguous.

<table>
<thead>
<tr>
<th>symbol</th>
<th>VLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>10</td>
</tr>
<tr>
<td>W</td>
<td>11</td>
</tr>
</tbody>
</table>
Uniquely Decodable Codes

- To avoid the ambiguity in decoding, we need to enforce certain conditions with VLC to make them uniquely decodable.
- Since ambiguity arises when some codeword becomes the prefix of the other, it is natural to consider prefix condition.

Example: \( p \preceq pr \preceq pre \preceq pref \preceq pref_i \preceq prefix \)

\[ a \preceq b: a \text{ is the prefix of } b \]
Prefix condition

No codeword is allowed to be the prefix of any other codeword.

We will graphically illustrate this condition with the aid of binary codeword tree.
Binary Codeword Tree

Level 1
Level 2
Level k

# of codewords

root

1 0
11 10 01 00

2 2^2 2^k
### Prefix Condition Examples

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Codeword 1</th>
<th>Codeword 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N$</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>$E$</td>
<td>10</td>
<td>110</td>
</tr>
<tr>
<td>$W$</td>
<td>11</td>
<td>111</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Codeword 1:**
  - $S$: 0
  - $N$: 1
  - $E$: 10
  - $W$: 11
  - Other symbols: ...

- **Codeword 2:**
  - $S$: 0
  - $N$: 10
  - $E$: 110
  - $W$: 111
  - Other symbols: ...

- **Example:**
  - Symbol $W$.
  - Codeword 1: 11
  - Codeword 2: 111

- **Validation:**
  - Codeword 1: Incorrect.
  - Codeword 2: Correct.
How to satisfy prefix condition?

- Basic rule: If a node is used as a codeword, then all its descendants cannot be used as codeword.

Example
VLC Summary

- Rule #1: short codeword – large probability event, long codeword – small probability event
- Rule #2: no codeword can be the prefix of any other codeword
- Question: given $P(X)$, how to systematically assign the codewords that always satisfy those two rules?
  - Answer: Huffman coding, arithmetic coding

Entropy Coding
Huffman Codes (Huffman’1952)

- Coding Procedures for an N-symbol source
  - Source reduction
    - List all probabilities in a **descending** order
    - Merge the two symbols with smallest probabilities into a new **compound symbol**
    - Repeat the above two steps for N-2 steps
  - Codeword assignment
    - Start from the smallest source and work **backward** to the original source
    - Each **merging point** corresponds to a node in binary codeword tree
### Example-I

**Step 1: Source reduction**

<table>
<thead>
<tr>
<th>symbol $x$</th>
<th>$p(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>0.5</td>
</tr>
<tr>
<td>$N$</td>
<td>0.25</td>
</tr>
<tr>
<td>$E$</td>
<td>0.125</td>
</tr>
<tr>
<td>$W$</td>
<td>0.125</td>
</tr>
</tbody>
</table>

*compound symbols*
Example-I (Con’t)

Step 2: Codeword assignment

<table>
<thead>
<tr>
<th>symbol</th>
<th>$p(x)$</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>E</td>
<td>0.125</td>
<td>0.5</td>
</tr>
<tr>
<td>W</td>
<td>0.125</td>
<td>1</td>
</tr>
</tbody>
</table>

$\text{NEW}$

$\text{EW}$

$\text{W}$

$\text{E}$

$\text{S}$

$\text{N}$
The codeword assignment is not unique. In fact, at each merging point (node), we can arbitrarily assign “0” and “1” to the two branches (average code length is the same).
## Example-II

Step 1: Source reduction

<table>
<thead>
<tr>
<th>symbol x</th>
<th>$p(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e$</td>
<td>0.4</td>
</tr>
<tr>
<td>$a$</td>
<td>0.2</td>
</tr>
<tr>
<td>$i$</td>
<td>0.2</td>
</tr>
<tr>
<td>$o$</td>
<td>0.1</td>
</tr>
<tr>
<td>$u$</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Compound symbols

- $0.6_{(aiou)}$
- $0.4_{(iou)}$
- $0.4_{(ou)}$

- $0.2$
Example-II (Con’t)

Step 2: Codeword assignment

<table>
<thead>
<tr>
<th>symbol x</th>
<th>( p(x) )</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e )</td>
<td>0.4 0.4</td>
<td>0</td>
</tr>
<tr>
<td>( a )</td>
<td>0.2 0.2</td>
<td>0</td>
</tr>
<tr>
<td>( i )</td>
<td>0.2 0.2</td>
<td>1</td>
</tr>
<tr>
<td>( o )</td>
<td>0.1 0.2</td>
<td>000</td>
</tr>
<tr>
<td>( u )</td>
<td>0.1</td>
<td>0010</td>
</tr>
</tbody>
</table>

\( \text{compound symbols} \)
Example-II (Con’t)

binary codeword tree representation
Data Compression Basics

- Discrete source
  - Information = uncertainty
  - Quantification of uncertainty
  - Source entropy

- Variable length codes
  - Motivation
  - Prefix condition
  - Huffman coding algorithm

- Data Compression = source modeling
What is Source Modeling

discrete source X → entropy coding → binary bit stream

Modeling Process

discrete source X → Y → entropy coding → binary bit stream

P(Y) → probability estimation
Examples of Modeling Process

- **Run-length coding**
  - Count the run-lengths of identical symbols (suitable for binary/graphic images)

- **Dictionary-based coding**
  - Record repeating patterns into a dictionary updated on-the-fly (e.g., Lempel-Ziv algorithm in WinZip)

- **Transform coding**
  - Apply linear transform to a block of symbols (e.g., discrete cosine transform in JPEG)

- **Predictive coding**
  - Apply linear prediction to the sequence of symbols (e.g., DPCM and ADPCM in wired transmission of speech)
Predictive Coding

Prediction residue sequence $Y$ usually contains less uncertainty (entropy) than the original sequence $X$.

WHY? Because the redundancy is assimilated into the LP model.
Two Extreme Cases

tossing a fair coin → H,H,T,H,T,H,T,T,T,H,T,T,…

\[ P(X=H) = P(X=T) = 1/2: \text{ (maximum uncertainty) } \]
No prediction can help (have to spend 1 bit/sample)

tossing a coin with two identical sides

H or T? duplication

→ HHHH…
→ TTTT…

\[ P(X=H) = 1, P(X=T) = 0: \text{ (minimum uncertainty) } \]
Prediction is always right (1 bit is enough to code all)
Differential PCM

- **Basic idea**
  - Since speech signals are slowly varying, it is possible to eliminate the temporal redundancy by prediction
  - Instead of transmitting original speech, we code prediction residues instead, which typically have smaller energy

- **Linear prediction**
  - Fixed: the same predictor is used again and again
  - Adaptive: predictor is adjusted on-the-fly
First-order Prediction

- **Encoding**
  \[ x_1 \ x_2 \ \ldots \ \ldots \ x_N \rightarrow e_1 \ e_2 \ \ldots \ \ldots \ e_N \]
  \[ e_1 = x_1 \quad e_n = x_n - x_{n-1}, \quad n=2,\ldots,N \]

- **Decoding**
  \[ e_1 \ e_2 \ \ldots \ \ldots \ e_N \rightarrow x_1 \ x_2 \ \ldots \ \ldots \ x_N \]
  \[ x_1 = e_1 \quad x_n = e_n + x_{n-1}, \quad n=2,\ldots,N \]
Prediction Meets Quantization

- Open-loop
  - Prediction is based on unquantized samples
  - Since decoder only has access to quantized samples, we run into a so-called *drifting* problem which is really bad for compression

- Closed-loop
  - Prediction is based on quantized samples
  - Still suffers from *error propagation* (i.e., quantization errors of the past will affect the efficiency of prediction), but no drifting is involved
Open-loop DPCM

Encoder

Decoder

Notes:  
• Prediction is based on the past unquantized sample

• Quantization is located outside the DPCM loop
Closed-loop DPCM

\[ x_n, e_n : \text{unquantized samples and prediction residues} \]
\[ \hat{x}_n, \bar{e}_n : \text{decoded samples and quantized prediction residues} \]

Notes:
- Prediction is based on the past decoded sample
- Quantization is located inside the DPCM loop
Numerical Example

\[ Q(x) = \left[ \frac{x}{3} \right] \cdot 3 \]
Closed-loop DPCM Analysis

A: \[ e_n = x_n - \hat{x}_{n-1} \]

B: \[ \hat{x}_n = \bar{e}_n + \hat{x}_{n-1} \]

The distortion introduced to prediction residue \( e_n \) is identical to that introduced to the original sample \( x_n \).
High-order Linear Prediction

original samples \( x_1 \ x_2 \ \ldots \ \ldots \ x_{n-1} \ x_n \ x_{n+1} \ \ldots \ \ldots \ x_N \)

- Encoding \( x_1 \ x_2 \ \ldots \ \ldots \ x_N \rightarrow e_1 \ e_2 \ \ldots \ \ldots \ e_N \)

  initialize \( e_1=x_1, e_2=x_2, \ldots, e_k=x_k \)

  prediction \( e_n=x_n-\sum_{i=1}^{k} a_i x_{n-i}, n=k+1, \ldots, N \)

- Decoding \( e_1 \ e_2 \ \ldots \ \ldots \ e_N \rightarrow x_1 \ x_2 \ \ldots \ \ldots \ x_N \)

  initialize \( x_1=e_1, x_2=e_2, \ldots, x_k=e_k \)

  prediction \( x_n=e_n+\sum_{i=1}^{k} a_i x_{n-i}, n=k+1, \ldots, N \)

The key question is: how to select prediction coefficients?
Recall: LP Analysis of Speech

\[
\begin{align*}
\text{minimize} & \quad MSE = \sum_{n=1}^{N} e^2(n) = \sum_{n=1}^{N} \left[ x(n) - \sum_{k=1}^{K} a_k x(n-k) \right]^2 \\
\end{align*}
\]

\[
\begin{bmatrix}
R_n(0) & R_n(1) & \cdots & R_n(K-1) \\
R_n(1) & R_n(0) & \ddots & \vdots \\
\vdots & \ddots & \ddots & R_n(1) \\
R_n(K-1) & \cdots & R_n(1) & R_n(0)
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_K
\end{bmatrix}
= 
\begin{bmatrix}
R_n(1) \\
R_n(2) \\
\vdots \\
R_n(K)
\end{bmatrix}
\]

Note that in fixed prediction, auto-correlation is calculated over the **whole** segment of speech (NOT short-time features)
Quantized Prediction Residues

- Further entropy coding is possible
  - Variable length codes: e.g., Huffman codes, Golomb codes
  - Arithmetic coding

- However, the current practice of speech coding assign fixed-length codewords to quantized prediction residues
  - The assigned code-lengths are already nearly optimal for achieving the first-order entropy
  - Good for the robustness to channel errors

- DPCM can achieve compression of two (i.e., 32kbps) without noticeable speech quality degradation
Adaptive DPCM

- **Encoder**
  - 64 kbits/s PCM Input
  - Convert to Uniform PCM
  - Input Signal
  - Difference Signal
  - Adaptive Quantizer
  - Signal Estimate
  - Reconstructed Signal
  - Adaptive Predictor
  - Quantized Difference Signal
  - Inverse Adaptive Quantizer
  - ADPCM Output

- **Decoder**
  - ADPCM Input
  - Inverse Adaptive Quantizer
  - Reconstructed Signal
  - Signal Estimate
  - Adaptive Predictor
  - Signal Output
Adaptation

\[ e_n = x_n - \sum_{i=1}^{K} a_i x_{n-i}, \quad n=K+1,\ldots,N \]

\[
\begin{bmatrix}
R_n(0) & R_n(1) & \cdots & R_n(K-1) \\
R_n(1) & R_n(0) & \ddots & \vdots \\
\vdots & \ddots & R_n(1) \\
R_n(K-1) & \cdots & R_n(1) & R_n(0)
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_K
\end{bmatrix}
= 
\begin{bmatrix}
R_n(1) \\
R_n(2) \\
\vdots \\
R_n(K)
\end{bmatrix}
\]

To track the slowly-varying property of speech signals, the estimated short-time auto-correlation is updated on the fly.
Prediction Gain

\[ G_p = 10 \log_{10} \frac{\sigma^2_x}{\sigma^2_e} \text{ (dB)} \]

\[ G_p = 10 \log_{10} \frac{\sigma^2_x}{\sigma^2_e} \text{ (dB)} \]
Forward and Backward Adaptation

- **Forward adaptation**
  - The autocorrelation is estimated from the current frame and the quantized prediction coefficients are transmitted to the decoder as *side information* (we will discuss how to quantize such vector in the discussion of CELP coding)

- **Backward adaptation**
  - The autocorrelation is estimated from the causal past and therefore *no overhead* is required to be transmitted; decoder will duplicate the encoder’s operation
Illustration of Forward Adaptation

For each frame, a set of LPCs are transmitted
Illustration of Backward Adaptation

For each frame, LPC are learned from the past frame
## Comparison

<table>
<thead>
<tr>
<th>Forward adaptive prediction</th>
<th>Backward adaptive prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric complexity allocation (encoder&gt;decoder)</td>
<td>Symmetric complexity allocation (encoder=decoder)</td>
</tr>
<tr>
<td>Overhead non-negligible</td>
<td>No overhead</td>
</tr>
<tr>
<td>robust to errors</td>
<td>sensitive to errors</td>
</tr>
<tr>
<td>More suitable for low-bit rate coding</td>
<td>More suitable for high-bit rate coding</td>
</tr>
</tbody>
</table>
AR vs. MA

- **Autoregressive (AR) Model**
  - Essentially an IIR filter (all-poles)

\[ X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t. \]

- **Moving average (MA) model**
  - Essentially a FIR filter (all-zeros)

\[ X_t = \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}. \]

- **Autoregressive Moving Average (ARMA) model**
## Waveform Coding Demo

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Bit Rate (Kbit/s)</th>
<th>MOS</th>
<th>Complexity (MIPS)</th>
<th>Frame Size (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCM G.711</td>
<td>64</td>
<td>4.3</td>
<td>0.01</td>
<td>0</td>
</tr>
<tr>
<td>ADPCM G.726</td>
<td>32</td>
<td>4.1</td>
<td>2</td>
<td>0.125</td>
</tr>
<tr>
<td>SBC G.722</td>
<td>48/56/64</td>
<td>4.1</td>
<td>5</td>
<td>0.125</td>
</tr>
<tr>
<td>LD-CELP G.728</td>
<td>16</td>
<td>4.0</td>
<td>30</td>
<td>0.625</td>
</tr>
<tr>
<td>CS-ACELP(-A) G.729</td>
<td>8</td>
<td>4.0 (3.8)</td>
<td>20 (11)</td>
<td>10</td>
</tr>
<tr>
<td>MPC-MLQ G.723.1</td>
<td>6.3/5.3</td>
<td>4.0/3.7</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>GSM HR VSELP</td>
<td>6.3</td>
<td>3.4</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>IS-54 VSELP</td>
<td>8</td>
<td>3.5</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>IS-96 QCELP</td>
<td>1.2/2.4/4.8/9.6</td>
<td>3.3</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Inmarsat-B APC</td>
<td>9.6/12.8</td>
<td>3.1/3.4</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>MELP</td>
<td>2.4</td>
<td>3.2</td>
<td>40</td>
<td>22.5</td>
</tr>
<tr>
<td>FS 1016 – CELP</td>
<td>4.8</td>
<td>3.2</td>
<td>16</td>
<td>30</td>
</tr>
</tbody>
</table>

http://www-lns.tf.uni-kiel.de/demo/demo_speech.htm
Data Compression Paradigm

discrete source $X$ → Modeling Process → $Y$ → entropy coding → probability estimation → $P(Y)$ → $Y$ → binary bit stream

Question: Why do we call DPCM/ADPCM waveform coding?

Answer: Because $Y$ (prediction residues) is also a kind of waveform just like the original speech $X$ – they have the same dimensionality.
Speech Coding Techniques (I)

- Introduction to Quantization
  - Scalar quantization
  - Uniform quantization
  - Nonuniform quantization

- Waveform-based coding
  - Pulse Coded Modulation (PCM)
  - Differential PCM (DPCM)
  - Adaptive DPCM (ADPCM)

- Model-based coding
  - Channel vocoder
  - Analysis-by-Synthesis techniques
  - Harmonic vocoder
Introduction to Model-based Coding

Signal space

\{x_1, \ldots, x_N\} \subset \mathbb{R}^N

model space

\{\theta_1, \ldots, \theta_K\} \subset \mathbb{R}^K

\[ K << N \]
Toy Example

Model-based coding of sinusoid signals

\[ x_n = \sin \left( 2\pi fn + \varphi \right), n = 1, 2, \ldots, N \]

Input signal \( \{x_1, \ldots, x_N\} \) \[\begin{array}{c}
\text{analysis} \\
\text{\( \theta_1 = f \)} \\
\text{\( \theta_2 = \varphi \)} \\
\text{synthesis} \\
\text{Reconstructed signal}
\end{array}\]

Question (test your understanding about entropy)
If a source is produced by the following model:
\[ x(n) = \sin(50n + p), \ n = 1, 2, \ldots, 100 \] where \( p \) is a random phase with equal probabilities of being 0, 45, 90, 135, 180, 225, 270, 315. What is the entropy of this source?
Building Models for Human Speech

- Waveform coders can also viewed as a kind of models where $K=N$ (not much compression)
- Usually model-based speech coders target at very high compression ratio ($K<<(K/N)$)
- There is no free lunch
  - High compression ratio comes along with severe quality degradation
  - We will see how to achieve a better tradeoff in part II (CELP coder)
Channel Vocoder: Encoding

- Bandpass filter
- Bandpass filter
- Voicing detector
- Pitch detector
- rectifier
- rectifier
- Lowpass filter
- Lowpass filter
- A/D converter
- A/D converter

Frame size: 20ms

Bit rate = 2400 - 3200 bps

1 bit

6 bits

3-4 bits / channel
Channel Vocoder: Decoding
Analysis-by-Synthesis

- Motivation
  - The optimality of some parameter is easy to determine (e.g., pitch), but not for others (e.g., gain parameters across different bands in channel voder)
  - The interaction among parameters is difficult to analyze but important to synthesis part

- What is A-by-S?
  - Do the complete analysis-and-synthesis in encoding
  - Decoder is embedded in the encoder for the reason of optimizing the extracted parameters
A-by-S is a Closed Loop

\[ \hat{\theta} = [\theta_1, \ldots, \theta_K] \]

Analysis → Synthesis

\[ \hat{x} \]

\[ \hat{e} = [e_1, \ldots, e_N] \]

input speech

\[ \hat{x} = [x_1, \ldots, x_N] \]
Toy Example Revisited

Input signal \{x_1, \ldots, x_N\} \rightarrow \text{analysis} \rightarrow \theta_1 = f \\ \text{synthesis} \rightarrow \text{Reconstructed signal}

Function \text{dist}=\text{MSE\_AbyS}(x,f,\phi)
\begin{align*}
n &= 1: \text{length}(x); \\
x_{\text{rec}} &= \sin(2\pi f n + \phi); \\
e &= x - x_{\text{rec}}; \\
\text{dist} &= \text{sum}(e \cdot e);
\end{align*}

MATLAB provides various tools for solving optimization problem

>help fminsearch
Harmonic Models

For speech within a frame

\[ s(n) = \sum_{j=1}^{J} A_j \cos(\omega_j n + \varphi_j) \]

- For voiced signals
  - Phase is controlled by pitch period
  - Pitch is often modeled by a slowly varying quadratic model
- For unvoiced signals
  - random phase
- Less accurate for transition signals (e.g., plosive, voice onsetting etc.)
A-by-S Harmonic Vocoder
# Bit Allocation

<table>
<thead>
<tr>
<th>Parameters</th>
<th>1st subframe</th>
<th>2nd subframe</th>
<th>frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSFs</td>
<td></td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>class</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>pitch</td>
<td>0</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>harmonic</td>
<td>14</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>magnitudes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>voicing</td>
<td>0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>gain</td>
<td>7</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td></td>
<td>80</td>
</tr>
</tbody>
</table>

Frame size=20ms, Bit rate =4K bps
Towards the Fundamental Limit

- How much information can one convey in one minute?
  - It depends on how fast one speaks
  - It depends on which language one speaks
  - It surely also depends on the speech content

- Model-based speech coders
  - Can compress speech down to 300-500 bits/second, can’t tell who speaks it, no intonation or stress or gender difference
  - A theoretically optimal approach: speech recognition + speaker recognition + speaker-dependent speech synthesis