根据学校课堂纪律的要求

请同学们坐在前五排

数字媒体技术基础

Meng Yang

www. smartllv. com

SUN YAT-SEN University

机器智能与先进计算教育部重点实验室

智能视觉语言学习研究组

数字媒体压缩技术

Course Outline

10数字媒体压缩技术

10. 1 无损压缩技术

10. 2 有损压缩技术

10. 3 图像JEPG压缩标准

10. 4 视频图像MPEG压缩标准

10. 5 音频压缩技术

有损和无损压缩技术

Introduction

- Lossless compression （无损压缩） algorithms do not deliver compression ratios that are high enough. Hence, most multimedia compression algorithms are lossy.
- What is lossy compression （有损压缩）？
  - The compressed data is not the same as the original data, but a close approximation of it.
  - Yields a much higher compression ratio than that of lossless compression.

无损压缩技术

Huffman Coding —— a bottom-up approach

- 1. Initialization: Put all symbols on a list sorted according to their frequency counts.将所有的符号根据频次排序为列表
- 2. Repeat until the list has only one symbol left:
  - （1）选取两次最低频次的符号，组成一棵Huffman树（以这两个符号为子节点，并创造一个父节点）
  - (2) 父节点的频次为子节点之和，将父节点插入列表
  - (3) 删除列表中的子节点
- 3. Assign a codeword for each leaf based on the path from the root.

无损压缩技术

- In Fig. 7.5, new symbols P1, P2, P3 are created to refer to the parent nodes in the Huffman coding tree. The contents in the list are illustrated below:

After initialization: L H E O

After iteration (a): L P1 H

After iteration (b): L P2

After iteration (c): P3

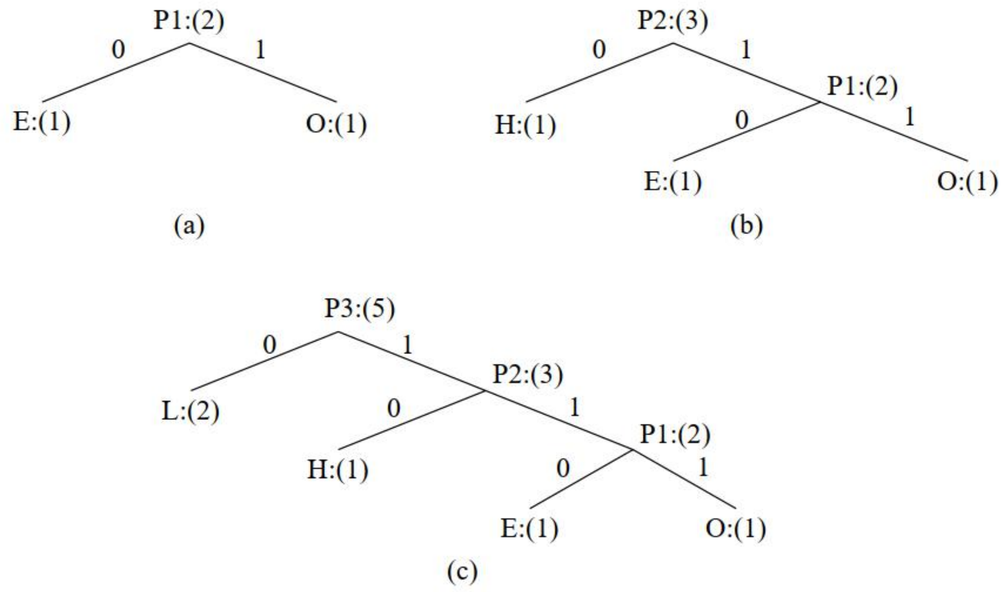


Fig. 7.5: Coding Tree for "HELLO" using the Huffman Algorithm.

无损压缩技术

Properties of Huffman Coding

- 1. 唯一前缀性质: No Huffman code is a prefix of any other Huffman code - precludes any ambiguity in decoding.

- 2.最优性质: minimum redundancy code - proved optimal for a given data model (i.e., a given, accurate, probability distribution):
  - The average code length for an information source S is strictly less than η + 1. Combined with Eq. (7.5), we have:

$$\bar{l} < \eta + 1 \qquad (7.6)$$

如果数据是流数据？

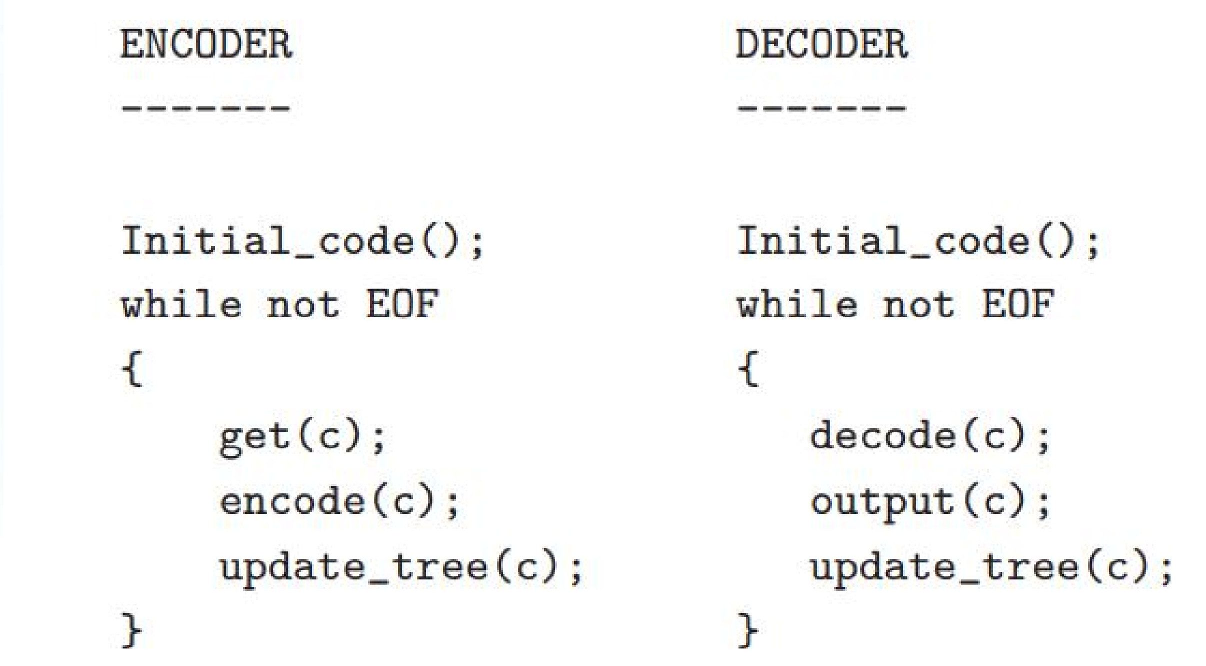
- 符号未知？
- 频次未知？

问题？

无损压缩技术

Adaptive Huffman Coding

- 自适应**Huffman**编码: statistics are gathered and updated dynamically as the data stream arrives.



```
ENCODER                       DECODER
-------                       -------

Initial_code();               Initial_code();
while not EOF                 while not EOF
{                             {
    get(c);                      decode(c);
    encode(c);                   output(c);
    update_tree(c);              update_tree(c);
}                             }
```

无损压缩技术

Adaptive Huffman Coding

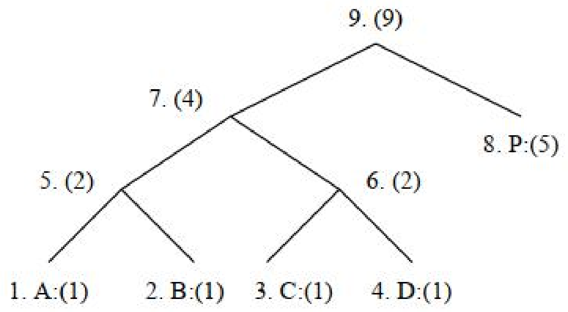
- Initial code assigns symbols with some initially agreed upon codes, without any prior knowledge of the frequency counts.
- update tree constructs an Adaptive Huffman tree. It basically does two things:
  - (a) increments the frequency counts for the symbols (including any new ones).增加符号的频次
  - (b) updates the configuration of the tree.更新树的结构
- The encoder and decoder must use exactly the same initial code and update tree routines.编码器和解码器需要用同样的初始码和更新树

无损压缩技术

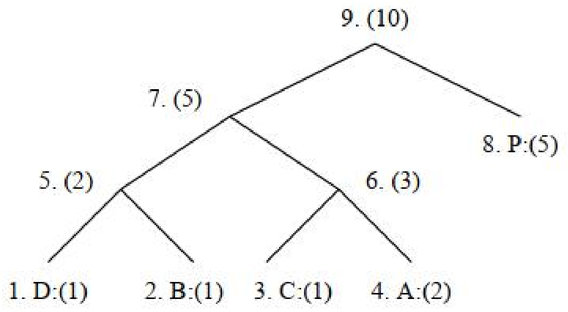
Notes on Adaptive Huffman Tree Updating

- 节点保持其兄弟性质（从左到右，从下到上的顺序频次递增的）
- 如果不满足此性质，需要进行节点交换
- 当进行节点交换时，与不满足此性质的最远节点交换
- When a swap is necessary, the farthest node with count N is swapped with the node whose count has just been increased to N + 1.

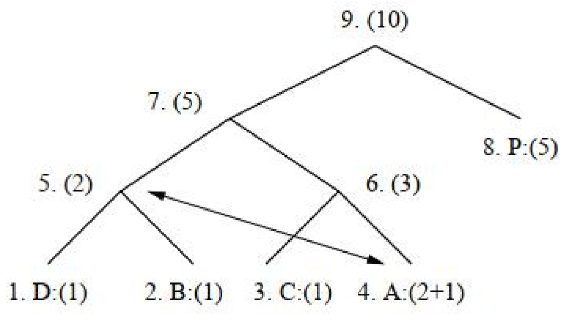
无损压缩技术



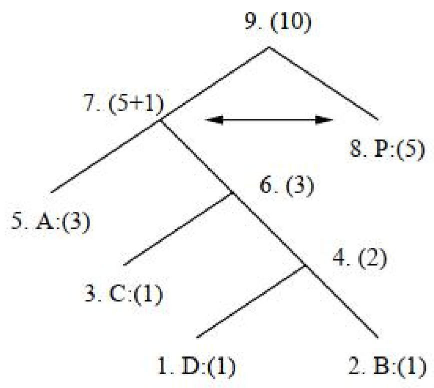
(a) A Huffman tree



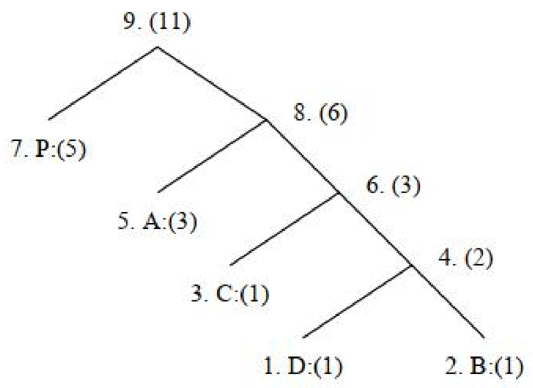
(b) Receiving 2nd 'A' triggered a swap



(c−1) A swap is needed after receiving 3rd 'A'



(c−2) Another swap is needed



(c−3) The Huffman tree after receiving 3rd 'A'

Fig. 7.6: Node Swapping for Updating an Adaptive Huffman Tree

无损压缩技术

Another Example: Adaptive Huffman Coding

- This is to clearly illustrate more implementation details. We show exactly what bits are sent, as opposed to simply stating how the tree is updated.

- An additional rule: 如果是是一个新的符号，它前面必须有一个特殊符号，NEW。The initial code for NEW is 0. The count for NEW is always kept as 0 (NEW的计数不会增加); hence it is always denoted as NEW:(0) in Fig. 7.7.

无损压缩技术

Table 7.3: Initial code assignment for AADCCDD using adaptive Huffman coding.

	Initial Code
NEW:	0
A:	00001
B:	00010
C:	00011
D:	00100
.	.
.	.
.	.

无损压缩技术

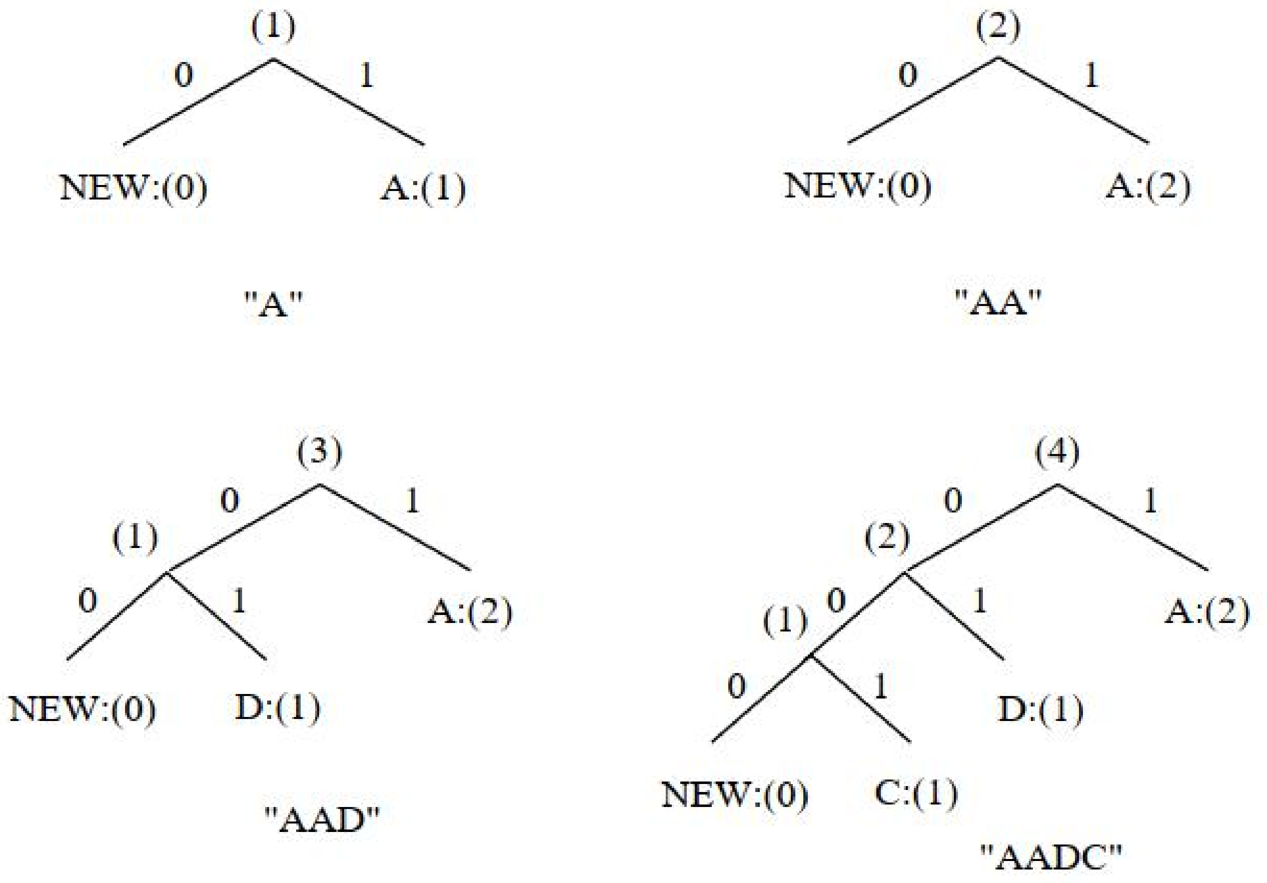


Fig. 7.7 Adaptive Huffman tree for AADCCDD.

无损压缩技术

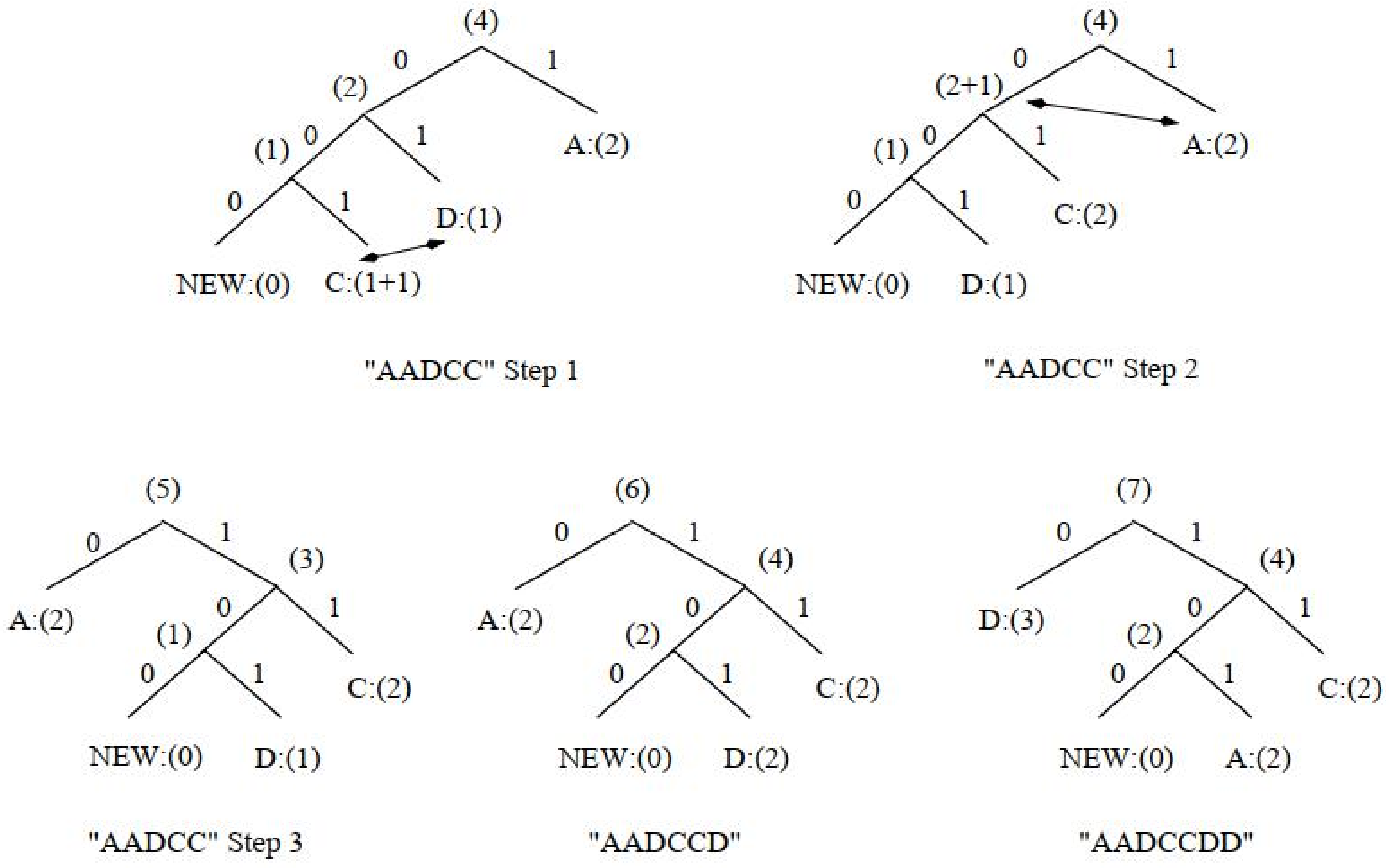
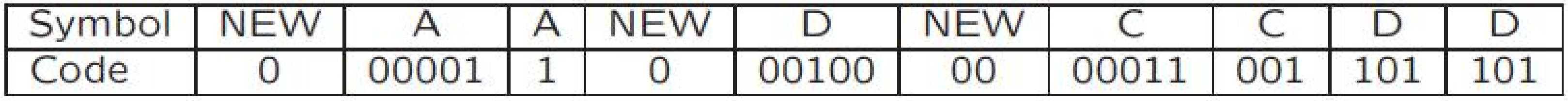


Fig. 7.7 (cont'd) Adaptive Huffman tree for AADCCDD.

无损压缩技术

Table 7.4 Sequence of symbols and codes sent to the decoder



Symbol	NEW	A	A	NEW	D	NEW	C	C	D	D
Code	0	00001	1	0	00100	00	00011	001	101	101

- ❑ It is important to emphasize that the code for a particular symbol changes during the adaptive Huffman coding process. For example, after AADCCDD, when the character D overtakes A as the most frequent symbol, its code changes from 101 to 0.
- ❑ The “Squeeze Page” on this book’s web site provides a Java applet for adaptive Huffman coding.

有损压缩技术

Distortion Measures

- The three most commonly used distortion measures in image compression are:
  - mean square error (MSE) σ^2,

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2 \qquad (8.1)$$

where xn, yn, and N are the input data sequence, reconstructed data sequence, and length of the data sequence respectively.

  - signal to noise ratio (SNR), in decibel units (dB),

$$SNR = 10\log_{10}\frac{\sigma_x^2}{\sigma_d^2} \qquad (8.2)$$

  - Where σ_x^2 is the average square value of the original data sequence and σ_d^2 is the MSE.
  - peak signal to noise ratio (PSNR),

$$PSNR = 10\log_{10}\frac{x_{peak}^2}{\sigma_d^2} \qquad (8.3)$$

有损压缩技术

The Rate-Distortion Theory

- Provides a framework for the study of tradeoffs between Rate and Distortion.

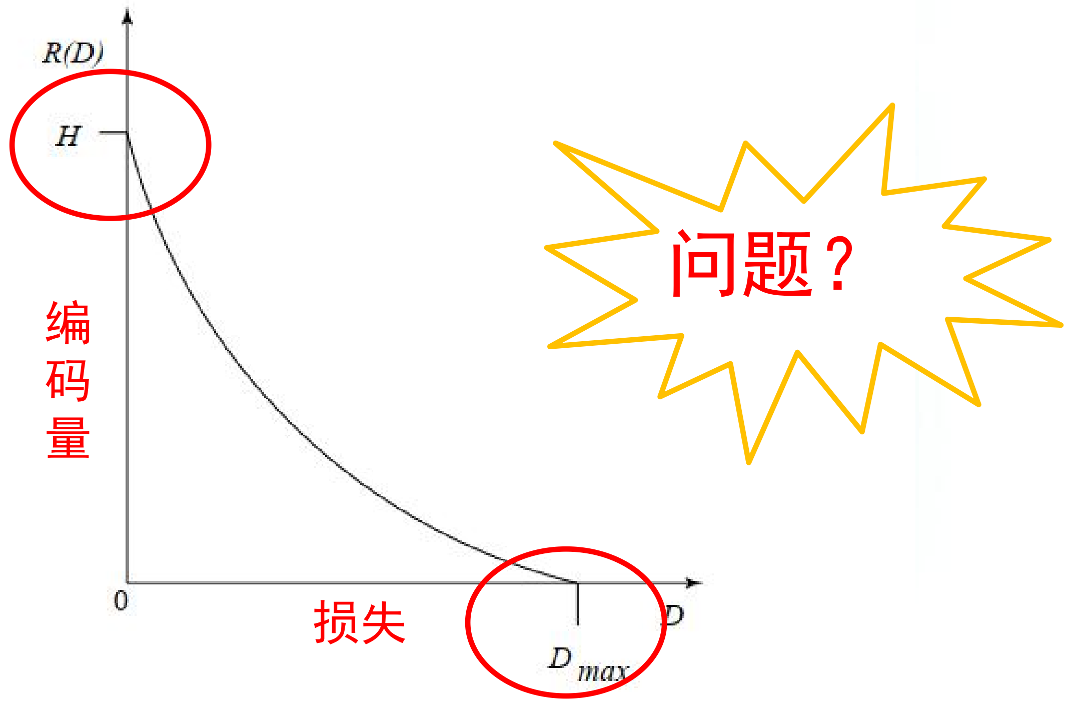


Fig. 8.1: Typical Rate Distortion Function.

有损压缩技术

Quantization

- Reduce the number of distinct output values to a much smaller set.
- Main source of the “loss” in lossy compression.
- 损失最大的来源是量化

DPCM（差分脉冲编码调制）

- Differential PCM is exactly the same as Predictive Coding, except that it incorporates a quantizer step.
  - (a) One scheme for analytically determining the best set of quantizer steps, for a non-uniform quantizer, is the **Lloyd-Max** quantizer, which is based on a least-squares minimization of the error term.
  - (b) Our nomenclature: signal values: f_n —— the original signal, $\widehat{f}_n$—— the predicted signal, and $\widetilde{f}_n$ the quantized, reconstructed signal.

有损压缩技术

DPCM

- This is a least-squares problem, and can be solved iteratively == the Lloyd-Max quantizer.
- Schematic diagram for DPCM:

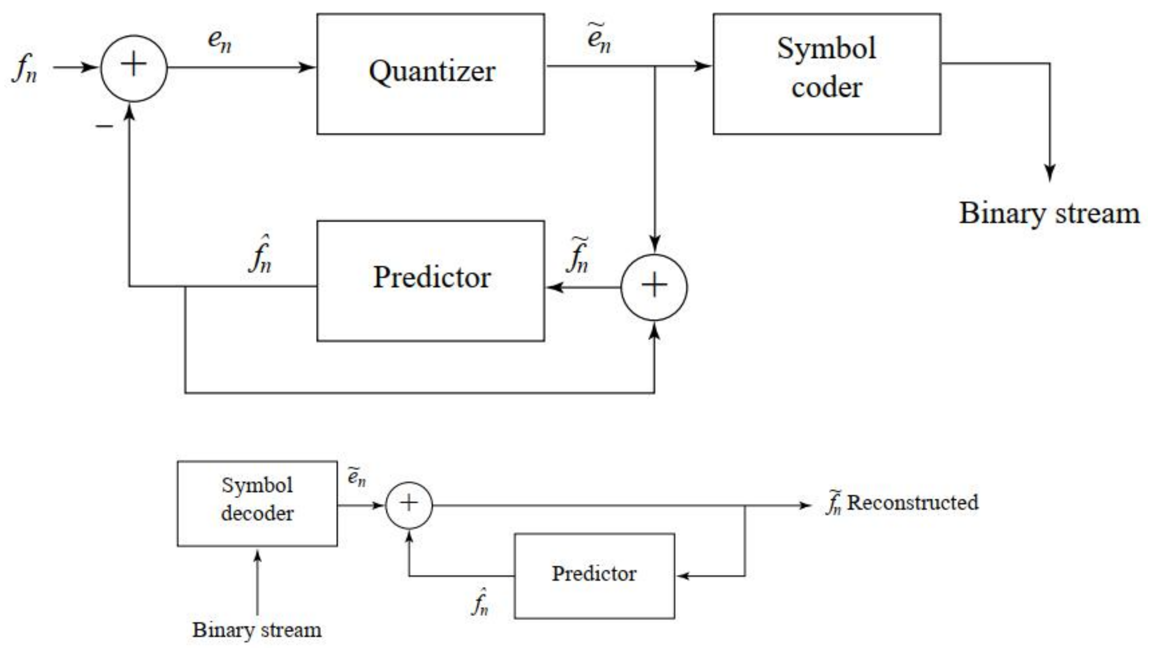


Fig. 6.17: Schematic diagram for DPCM encoder and decoder

有损压缩技术

DPCM

- (c) **DPCM**: form the prediction; form an error e_n by subtracting the prediction from the actual signal; then quantize the error to a quantized version, $\tilde{e}_n$.

The set of equations that describe DPCM are as follows:

$$\hat{f}_n = \text{function\_of}(\tilde{f}_{n-1},, \tilde{f}_{n-2}, \tilde{f}_{n-3}, \ldots),$$

$$e_n = f_n - \hat{f}_n,$$

$$\tilde{e}_n = Q[e_n],$$

$$\text{transmit } codeword(\tilde{e}_n), \tag{6.16}$$

$$\text{reconstruct: } \tilde{f}_n = \hat{f}_n + \tilde{e}_n.$$

Then codewords for quantized error values $\tilde{e}_n$ are produced using entropy coding, e.g. Huffman coding.

有损压缩技术

DPCM

- (d) The main effect of the coder-decoder process is to produce reconstructed, quantized signal values $\tilde{f}_n = \hat{f}_n + \tilde{e}_n$.

  The **distortion** （编码损失）is the average squared error $\left[\sum_{n=1}^{N}(\tilde{f}_n - f_n)^2\right]/N$; one often plots distortion versus the number of bit-levels used. A Lloyd-Max quantizer will do better (have less distortion) than a uniform quantizer.

有损压缩技术

DPCM

- For speech, we could modify quantization steps adaptively by estimating the mean and variance of a patch of signal values, and shifting quantization steps accordingly, for every block of signal values. That is, starting at time i we could take a block of N values f_n and try to minimize the quantization error:

$$min \sum_{n=i}^{i+N-1} (f_n - Q[f_n])^2 \tag{6.17}$$

有损压缩技术

DPCM

- This is a least-squares problem, and can be solved iteratively == the Lloyd-Max quantizer.
- Schematic diagram for DPCM:

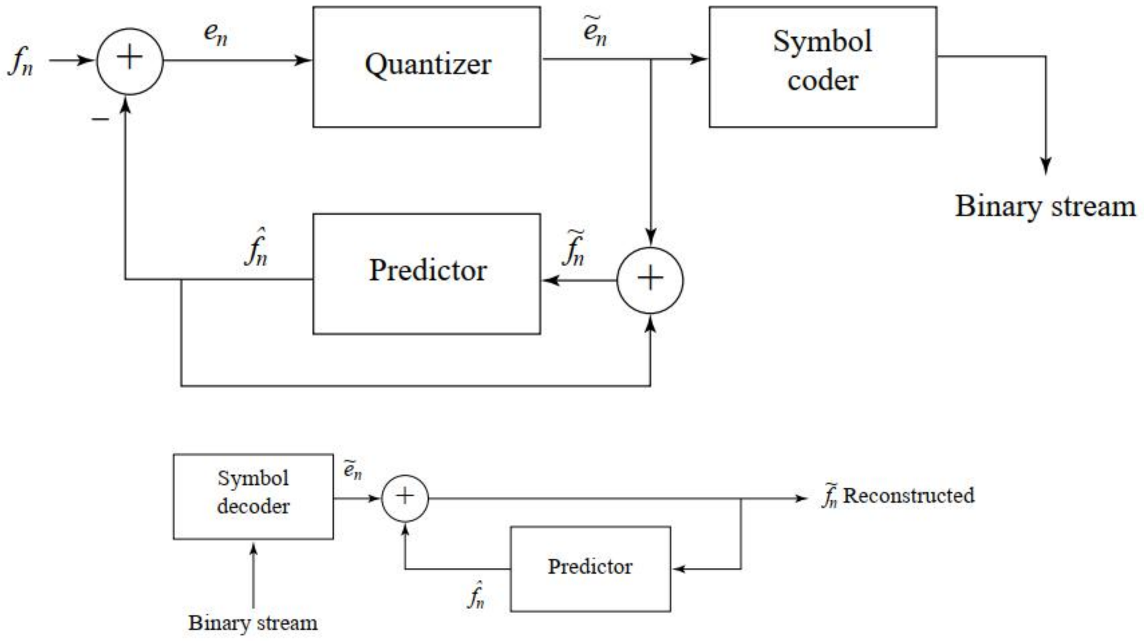


Fig. 6.17: Schematic diagram for DPCM encoder and decoder

有损压缩技术

DPCM

- Notice that the quantization noise, $f_n - \tilde{f}_n$, is equal to the quantization effect on the error term, $e_n - \tilde{e}_n$.
- Let's look at actual numbers: Suppose we adopt the particular predictor below:

$$\hat{f}_n = \text{trunc}\left(\left(\tilde{f}_{n-1} + \tilde{f}_{n-2}\right)/2\right) \qquad (6.19)$$

so that $e_n = f_n - \hat{f}_n$ is an integer.

- As well, use the quantization scheme:

$$\tilde{e}_n = Q[e_n] = 16 * \text{trunc}\left((255 + e_n)/16\right) - 256 + 8$$

$$\tilde{f}_n = \hat{f}_n + \tilde{e}_n \qquad (6.20)$$

有损压缩技术

DPCM

- First, we note that the error is in the range −255..255, i.e., there are 511 possible levels for the error term. The quantizer simply divides the error range into 32 patches of about 16 levels each. It also makes the representative reconstructed value for each patch equal to the midway point for each group of 16 levels.

有损压缩技术

DPCM

- Table 6.7 gives output values for any of the input codes: 4-bit codes are mapped to 32 reconstruction levels in a staircase fashion.

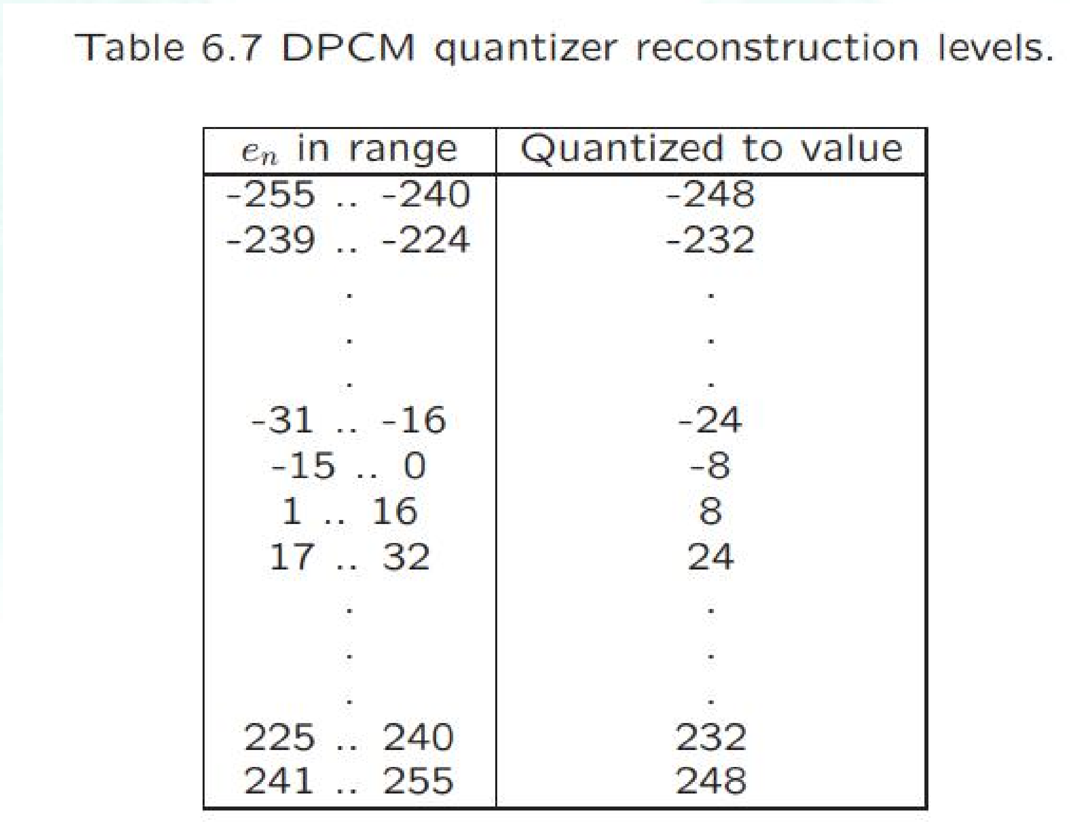


Table 6.7 DPCM quantizer reconstruction levels.

e_n in range	Quantized to value
-255 .. -240	-248
-239 .. -224	-232
.	.
.	.
.	.
-31 .. -16	-24
-15 .. 0	-8
1 .. 16	8
17 .. 32	24
.	.
.	.
.	.
225 .. 240	232
241 .. 255	248

有损压缩技术

DPCM

- As an example stream of signal values, consider the set of values:

$$\begin{array}{ccccc} f_1 & f_2 & f_3 & f_4 & f_5 \\ 130 & 150 & 140 & 200 & 230 \end{array}.$$

- Prepend extra values f = 130 to replicate the first value, $f1$. Initialize with quantized error $\tilde{e}_1 \equiv 0$, so that the first reconstructed value is exact: $\tilde{f}_1$ = 130. Then the rest of the values calculated are as follows (with prepended values in a box):

$$\begin{array}{ccccccc} \hat{f} & = & \boxed{130}, & 130, & 142, & 144, & 167 \\ e & = & \boxed{0}, & 20, & -2, & 56, & 63 \\ \tilde{e} & = & \boxed{0}, & 24, & -8, & 56, & 56 \\ \tilde{f} & = & \boxed{130}, & 154, & 134, & 200, & 223 \end{array}$$

- On the decoder side, we again assume extra values $\tilde{f}$ equal to the correct value $\tilde{f}_1$, so that the first reconstructed value $\tilde{f}_1$ is correct. What is received is $\tilde{e}_n$, and the reconstructed $\tilde{f}_n$ is identical to that on the encoder side, provided we use exactly the same prediction rule.