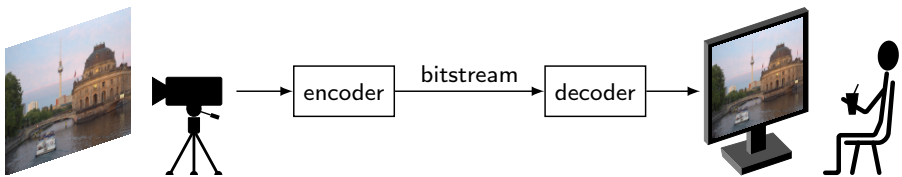
Summary



Types of Data Compression

Lossless Compression

- Invertible / reversible: Original input data can be completely recovered

➡ Examples:
- gzip, lzip, 7zip, bzip2 for general files (most suitable for text)
- FLAC for audio
- PNG, JPEG-LS for images

Lossy Compression

- Not invertible: Only **approximation of original input data** can be recovered
- Achieves **much higher compression ratios**
- Dominant form of data compression for media

➡ Examples:
- MP3, AAC for audio
- JPEG, JPEG-2000 for images
- MPEG-2, H.264 | AVC, H.265 | HEVC for video

Scalar Variable-Length Codes

Scalar Variable-Length Codes

- Codeword table: One codeword per alphabet letter

Encoding

- Concatenate codewords for symbols of a message

Decoding

- No separator symbols

➡ Require unique decodable codes

Efficiency

- Average codeword length

$$\bar{\ell} = \mathrm{E}\{\,\ell(a_k)\,\} = \sum_{\forall k} p(a_k) \cdot \ell(a_k) = \sum_{\forall k} p_k \cdot \ell_k$$

Example

$\mathcal{A} = \{\mathrm{A,B,C,D,E,F,G,H}\}$

a_k	$p(a_k)$	codeword
A	0.20	00
B	0.15	010
C	0.05	0110
D	0.10	0111
E	0.30	10
F	0.05	1100
G	0.05	1101
H	0.10	111

➡ **Goal: Minimize average codeword length while ensuring unique decodability**

Unique Decodability and Prefix Codes

Unique Decodability

- Necessary condition: Different codeword for each alphabet letter

➡ Not sufficient

➡ **Each sequence of bits can only be generated by one possible sequence of sources symbols**

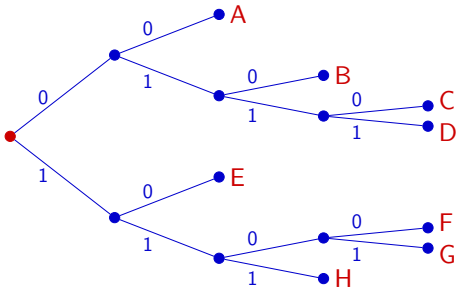
Prefix Codes

- One class of uniquely decodable codes
- Property: No codeword represents a prefix of any other codeword

➡ Can be represented using binary code tree

➡ Simple decoding: Follow tree from root to terminal node

➡ Instantaneously decodable (enabled switching of codeword tables)



Optimal Uniquely Decodable Codes

Kraft-McMillan Inequality

- Necessary condition for all uniquely decodable codes

➡ The codeword lengths $\{\ell_k\}$ of a uniquely decodable code must fulfill the inequality

$$\sum_{\forall k} 2^{-\ell_k} \leq 1$$

Kraft-McMillan Inequality for Prefix Codes

- No structural redundancy (all interior nodes have two descendants): $\sum_{\forall k} 2^{-\ell_k} = 1$
- Structural redundancy (some interior nodes have only one descendant): $\sum_{\forall k} 2^{-\ell_k} < 1$
- For each set of codeword lengths $\{\ell_k\}$ that satisfies the Kraft-McMillan inequality, a prefix code can be constructed

➡ **There are no better uniquely decodable codes than the best prefix codes**

➡ **All lossless codes used in practice are prefix codes**

Lower Bound for Average Codeword Length: Entropy

Entropy and Redundancy

- Entropy: Lower bound for average codeword length $\bar{\ell}(\gamma)$ of scalar codes γ

$$\bar{\ell}(\gamma) \geq H(S) \qquad \text{with} \qquad H(S) = H(p) = \mathrm{E}\{ -\log_2 p(S) \} = -\sum_{\forall k} p_k \cdot \log_2 p_k$$

- Absolute redundancy $\varrho(\gamma)$ and relative redundancy $r(\gamma)$

$$\varrho(\gamma) = \bar{\ell}(\gamma) - H(p) \geq 0 \qquad \text{and} \qquad r(\gamma) = \frac{\bar{\ell}(\gamma) - H(p)}{H(p)} \geq 0$$

Zero-Redundancy Codes?

➡ Only possible if all probability masses are negative integer powers of two

$$\sum_{\forall k} p_k \cdot \ell_k = -\sum_{\forall k} p_k \cdot \log_2 p_k \qquad \Longleftrightarrow \qquad \ell_k = -\log_2 p_k$$

- But: Can always construct a prefix code γ (for example, Shannon code: $\ell_k = \lceil -\log_2 p_k \rceil$) with

$$H(p) \leq \bar{\ell}(\gamma) < H(p) + 1$$

Optimal Prefix Codes: The Huffman Algorithm

Subset of Optimal Prefix Codes

- For any finite alphabet $\mathcal{A}$, there exists an optimal prefix code $\mathcal{C}$ with the following property:

  There are two codewords that have the maximum length, differ only in the final bit, and correspond to the two least likely alphabet letters.

The Huffman Algorithm

1. Select the two letters a and b with the smallest probabilities p_a and p_b
2. Create a parent node for the two letters a and b in the binary code tree
3. Replace the letters a and b with a new letter with probability $p_a + p_b$
4. If the resulting new alphabet contains more than a single letter, repeat all previous steps with this alphabet
5. Convert the obtained binary code tree into a prefix code

➡ **The Huffman algorithm yields one optimal prefix code / optimal uniquely decodable code.**

Check Your Knowledge: Scalar Variable-Length Codes

1. Develop a prefix code for the alphabet $\mathcal{A} = \{A, B, C\}$. Write down the bitsream for the message "CABAC". Check whether the resulting bitstream can be decoded correctly.

2. Given is the following probability mass function: $\{0.1, 0.1, 0.15, 0.2, 0.2, 0.25\}$.
   - ➡ Calculate the entropy.
   - ➡ Develop an optimal scalar variable-length code.
   - ➡ Calculate the average codeword length of the developed code.
   - ➡ Calculate the absolute and relative redundancy of the developed code.

3. Given is a prefix code that does not correspond to a full binary tree (i.e., one or more interior nodes have only one descendant). Can that code be an optimal code for some sources? Why or why not?

4. Is it true that all optimal codes for a given source are Huffman codes?

5. Is it possible to develop a uniquely decodable code for the following set of codeword lengths: $\{7, 7, 7, 7, 5, 4, 3, 2, 1\}$? If yes, write down a set of possible codewords.

Conditional Codes

- Switch codeword table based on a function $c_n = f(s_{n-1}, s_{n-2}, \cdots)$ of preceding symbols
- Design codeword table for each possible condition c_n (optimal: Huffman algorithm)

➡ Resulting average codeword length

$$\bar{\ell}_{\text{cond}} = \sum_{\forall x} p(C_n = x) \cdot \bar{\ell}(C_n = x)$$

➡ Can improve coding efficiency for sources with statistical dependencies

conditional Huffman code (assume "$S_{n-1} = a$" for first symbol)

	$S_{n-1} = a$		$S_{n-1} = b$		$S_{n-1} = c$	
x	$p(x \mid a)$	codeword	$p(x \mid b)$	codeword	$p(x \mid b)$	codeword
a	0.90	0	0.15	10	0.25	10
b	0.05	10	0.80	0	0.15	11
c	0.05	11	0.05	11	0.60	0
$\bar{\ell}_{\text{cond}} = \sum_{\forall x} p(S_{n-1} = x) \cdot \bar{\ell}(S_{n-1} = x) = 521/450 \approx 1.1578$						

scalar Huffman code

x	$p(x)$	codeword
a	$29/45$	0
b	$11/45$	10
c	$5/45$	11
$\bar{\ell}_{\text{scal}} = 61/45 \approx 1.3556$		

Conditional Entropy

- **Conditional entropy** for alphabet $\mathcal{A}$ and condition $c_n = f(s_{n-1}, s_{n-2}, \cdots)$ is given by

$$\begin{aligned} H(S_n \,|\, C_n) &= \mathrm{E}\{ -\log_2 p(S_n \,|\, C_n) \} \\ &= -\sum_{\forall c} \sum_{\forall a \in \mathcal{A}} p(a, c) \cdot \log_2 p(a \,|\, c) \end{aligned}$$

- Average codeword length for conditional Huffman code (with same condition) is bounded by

$$H(S_n \,|\, C_n) \leq \bar{\ell} < H(S_n \,|\, C_n) + 1$$

➡ **Conditioning never increases entropy**

$$H(S_n \,|\, C_n) \leq H(S_n) \qquad \text{(equality if } S_n \text{ and } C_n \text{ are independent)}$$

➡ **Conditioning never increases average codeword length** (for optimal code)

Block Codes

- Joint coding of $N > 1$ successive symbols (single codeword)

➡ Design code for N-d joint pmf $p(s_1, s_2, \cdots, s_N)$

➡ Optimal block code: Huffman algorithm

- Average codeword length is given by

$$\bar{\ell} = \frac{1}{N} \sum_{a_1, a_2, \cdots, a_N} p(a_1, a_2, \cdots, a_N) \cdot \ell(a_1, a_2, \cdots, a_N)$$

➡ Block codes can improve coding efficiency for both iid sources and sources with memory

- Problem: Codeword tables can become extremely large

Example: Document Scans

scalar code: $N = 1$ symbol

s	$p(s)$	codewords
0	0.80	1
1	0.20	0

$\bar{\ell} = 1.00$

$N = 2$ symbols

$s_1 s_2$	$p(s_1, s_2)$	codewords
00	0.72	1
01	0.08	010
10	0.08	011
11	0.12	00

$\bar{\ell} = \bar{\ell}_2/2 = 0.72$

Block Entropy

- **Block entropy** for alphabet $\mathcal{A}$ and block size N is given by

$$\begin{aligned} H_N(\boldsymbol{S}) &= H(S_1, S_2, \cdots, S_N) = \mathrm{E}\{ -\log_2 p(S_1, S_2, \cdots, S_N) \} \\ &= -\sum_{\forall a_1} \sum_{\forall a_2} \cdots \sum_{\forall a_N} p(a_1, a_2, \cdots, a_N) \cdot \log_2 p(a_1, a_2, \cdots, a_N) \end{aligned}$$

- Average codeword length for block Huffman code is bounded by

$$\frac{H_N(\boldsymbol{S})}{N} \leq \bar{\ell} < \frac{H_N(\boldsymbol{S})}{N} + \frac{1}{N}$$

➡ **Increasing the block size never increases lower bound**

$$\frac{H_{N+1}(\boldsymbol{S})}{N+1} \leq \frac{H_N(\boldsymbol{S})}{N} \qquad \text{(equality if } \boldsymbol{S} \text{ is an iid source)}$$

➡ **Increasing the block sizes does not increase average codeword length** (for optimal codes)

Fundamental Lossless Coding Theorem

Entropy Rate

- Entropy rate: Limit ($N \to \infty$) for lower bound of block codes

$$\bar{H}(\boldsymbol{S}) = \lim_{N\to\infty} \frac{H_N(\boldsymbol{S})}{N} = \lim_{N\to\infty} \frac{H(S_1, S_2, \cdots, S_N)}{N}$$

➡ **Fundamental lossless coding theorem**

$$\bar{\ell} \geq \bar{H}(\boldsymbol{S}) \qquad \text{(for all lossless codes)}$$

➡ Asymptotically achievable with block Huffman coding for large block sizes

Entropy Rate for Selected Sources

- IID sources: $H_N(\boldsymbol{S}) = N \cdot H(S)$ ➡ $\bar{H}(\boldsymbol{S}) = H(S)$
- Markov sources: $H_N(\boldsymbol{S}) = H(S) + (N-1) \cdot H(S_n \,|\, S_{n-1})$ ➡ $\bar{H}(\boldsymbol{S}) = H(S_n \,|\, S_{n-1})$

V2V Codes

Generalization of Block Codes

- Assign codewords to symbol sequences of variable length
  - All messages must be representable
  - Desirable: Redundancy-free set of symbol sequences
  - ➡ Choose full M-ary tree of symbol sequences
- ➡ Optimal V2V code for given symbol sequences: Huffman
- ➡ Average codeword length

$$\bar{\ell} = \frac{\sum_k p_k \cdot \ell_k}{\sum_k p_k \cdot n_k} = \frac{\text{avg. codeword length per seq.}}{\text{avg. number of symbols per seq.}}$$

Relation to Block Codes

- Block codes = Special V2V codes with perfect m-ary tree
- Advantage of V2V codes:
  Often smaller codeword table for same efficiency

V2V code (Huffman design)

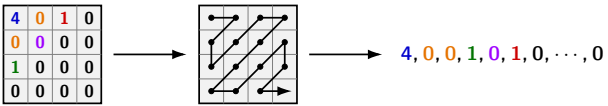
a_k	p_k	codewords
0000000	0.5964	1
0000001	0.0110	00111
000001	0.0101	00100
00001	0.0105	00101
0001	0.0111	0000
001	0.0108	00110
01	0.0145	0001
1	0.3356	01

$\left.\begin{array}{l} \bar{\ell}_{\text{seq}} = 1.582 \\ \bar{n} = 4.806 \end{array}\right\}$ ➡ $\bar{\ell} = 0.33$

block code with 8 codewords:

$\bar{\ell} = 0.42$

V2V Example: Run-Level Coding in JPEG, MPEG-2 Video, ...



Coding of Block Quantization Indexes (absolute values)

1. Convert block into sequence of indexes (zig-zag scan)
2. Convert sequence of indexes into **(run, level) pairs** and a special **end-of-block (eob) symbol**

   run: number of zeros that precede next non-zero index
   level: value of next non-zero index
   eob: all following indexes are equal to zero (end-of-block)

   ➡ **Example:**
   sequence of indexes: **4 0 0 1 0 1 0 0 ··· 0**
   (run, level) *pairs*: **(0, 4) (2, 1) (1, 1) (eob)**

3. **Codewords** are assigned to **(run, level) pairs**

MPEG-2 Video: 112 typical symbol sequences + *escape*

codeword	(run, level)	symbol sequence
10	(eob)	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ···
11	(0,1)	1
011	(1,1)	0, 1
0100	(0,2)	2
0101	(2,1)	0, 0, 1
0010 1	(0,3)	3
0011 1	(3,1)	0, 0, 0, 1
0011 0	(4,1)	0, 0, 0, 0, 1
0001 10	(1,2)	0, 2
0001 11	(5,1)	0, 0, 0, 0, 0, 1
0001 01	(6,1)	0, 0, 0, 0, 0, 0, 1
0001 00	(7,1)	0, 0, 0, 0, 0, 0, 0, 1
0000 110	(0,4)	4
0000 100	(2,2)	0, 0, 2
0000 111	(8,1)	0, 0, 0, 0, 0, 0, 0, 0, 1
0000 101	(9,1)	0, 0, 0, 0, 0, 0, 0, 0, 0, 1
0000 01	*escape*	*< followed by fixed-length codes >*
0010 0110	(0,5)	5
0010 0001	(0,6)	6
0010 0101	(1,3)	0, 3
0010 0100	(3,2)	0, 0, 0, 2
0010 0111	(10,1)	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1
0010 0011	(11,1)	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1
0010 0010	(12,1)	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1
···	···	···

Check Your Knowledge: Conditional Codes, Block Codes, V2V Codes

1. Which of the following code types may improve coding efficiency relative to scalar Huffman codes for iid sources: Conditional codes, Block codes, or V2V codes?

2. Is it possible that conditional Huffman codes become worse than scalar Huffman codes?

3. What is the advantage of V2V codes in comparison to block codes?

4. Assume you have to design an efficient lossless coding algorithm for binary images (for example, black and white document scans). Which of the following code types do you would use: Scalar Huffman code, Conditional Huffman code, Block Huffman code, V2V code?

   Why would you use that type of code?

   Explain in somewhat more detail how you would design the binary lossless codec.

5. Why is the run-level coding used in JPEG better than scalar Huffman coding?

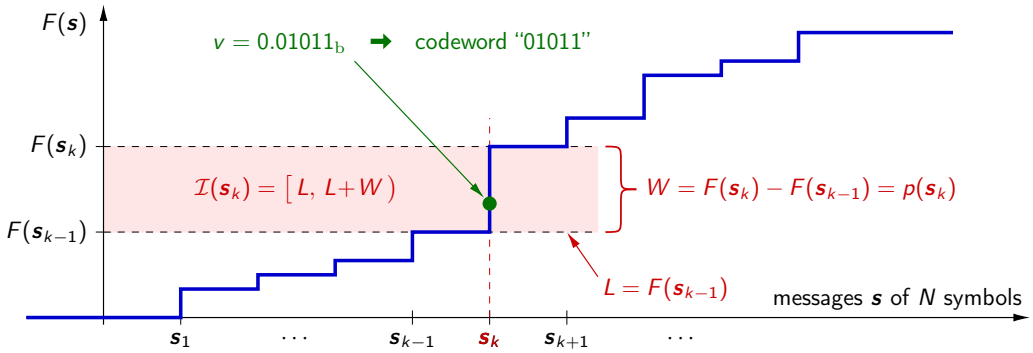
   What is the main advantage?

Idea of Shannon-Fano-Elias Coding

Special Block Code for N symbols

- Order all possible symbol sequences with N symbols: $\boldsymbol{s}_1$, $\boldsymbol{s}_2$, $\boldsymbol{s}_3$, $\cdots$
- Each symbol sequence $\boldsymbol{s}_k$ is associated with a half-open interval $\mathcal{I}(\boldsymbol{s}_k) = \big[L, L+W\big)$ of the cdf $F(\boldsymbol{s})$

➡ Transmit any number v inside the interval $\mathcal{I}(\boldsymbol{s}_k)$ as binary fraction



Shannon-Fano-Elias Codes: Iterative Encoding Algorithm

Given: Sequence $\boldsymbol{s} = \{s_1, s_2, s_3, \cdots, s_N\}$ of N symbols

1 Initialization of probability interval:

$$W_0 = 1 \qquad \text{and} \qquad L_0 = 0$$

2 Iterative refinement (for $n = 1$ to N):

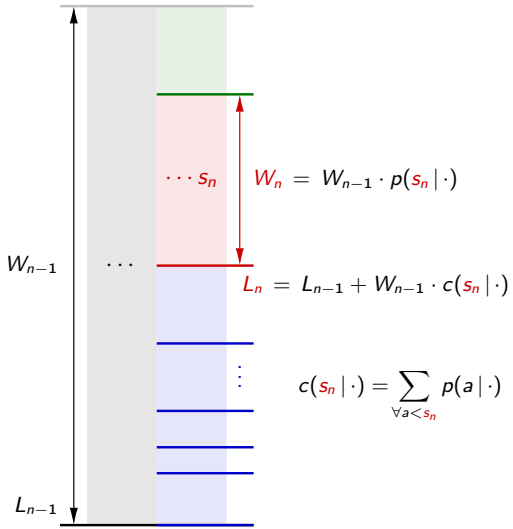
$$W_n = W_{n-1} \cdot p(\, s_n \,|\, \cdots)$$
$$L_n = L_{n-1} + W_{n-1} \cdot c(\, s_n \,|\, \cdots)$$

3 Determine codeword length and codeword value:

$$K = \lceil A - \log_2 W_N \rceil \qquad (A = 0 \text{ or } 1)$$
$$z = \lceil L_N \cdot 2^K \rceil$$

4 Transmit codeword:

➡ Binary representation of z with K bits



Shannon-Fano-Elias Codes: Iterative Decoding Algorithm

Given:
- Codeword: integer z with K bits
- Number N of symbols to be decoded
- Ordered alphabet $\mathcal{A} = \{a_1, a_2 \cdots\}$

1 Initialization: $W_0 = 1, \quad L_0 = 0, \quad v = z \cdot 2^{-K}$

2 Iterative decoding (for $n = 1$ to N):

a Upper boundary U_1 for first symbol a_1 of $\mathcal{A}$

$$k = 1, \qquad U_k = L_{n-1} + W_{n-1} \cdot p(\, a_k \,|\, \cdots)$$

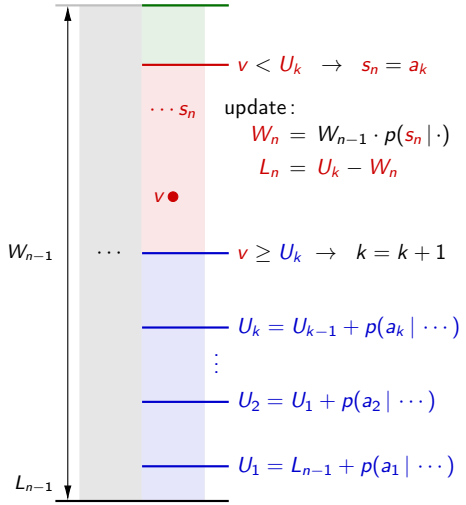
b While ($v \geq U_k$), proceed to next symbol of $\mathcal{A}$

$$k = k + 1, \qquad U_k = U_{k-1} + W_{n-1} \cdot p(\, a_k \,|\, \cdots)$$

c Output decoded symbol $s_n = a_k$

d Update interval: $W_n = W_{n-1} \cdot p(\, s_n \,|\, \cdots)$

$$L_n = U_k - W_n$$



Arithmetic Coding

Arithmetic Coding

- Shannon-Fano-Elias coding cannot be implemented due to extreme precision requirements
- Observation: Shannon-Fano-Elias code remains decodable as long as intervals are nested

➡ Arithmetic coding = Fixed-precision variant of Shannon-Fano-Elias coding

Basic Concept of Arithmetic Coding

- V-bit integer representation of probability masses: $p = p_V \cdot 2^{-V}, \qquad p_V > 0, \;\; \sum p_V \leq 2^V$
- U-bit integer representation of interval width: $W_n = A_n \cdot 2^{-z_n}, \qquad 2^{U-1} \leq A_n < 2^U$
- Downrounding of interval width in each iteration: Keep first '1' and $U-1$ following bits

➡ Output bits of lower interval boundary that cannot change in following iterations

$$L_{n-1} = 0.\overbrace{\underbrace{aaaaa\cdots a}_{z_{n-1}-c_{n-1}-U}\,\underbrace{0111111\cdots 1}_{c_{n-1}}}^{z_{n-1}-U \text{ bits}}\,\underbrace{xxxxx\cdots x}_{U+V}\,\underbrace{00000\cdots}_{\text{trailing bits}}$$

settled bits — outstanding bits — active bits

Practical Aspects of Arithmetic Coding

- Coding loss due to integer implementation is negligible
- ➡ Coding efficiency is determined by probabilities used in coding

Adaptive Arithmetic Coding

- Estimate probabilities during encoding and decoding (adapt to actual source statistics)
- Example: $p_V(a_k) = \left\lfloor 2^V \, \frac{N_k}{\sum_k N_k} \right\rfloor$

Conditional Arithmetic Coding

- Usage of conditional probabilities (e.g., use value of preceeding symbol as condition)
- Note: Probability models cannot adapt if we have too many conditions

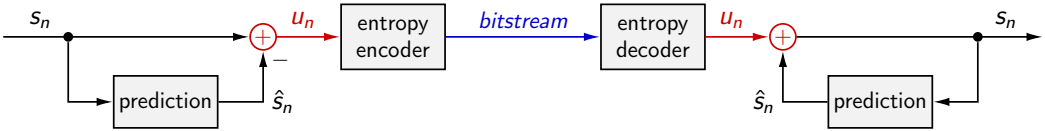
Binary Arithmetic Coding

- Binarization using simple prefix code, followed by arithmetic coding of bins
- Complexity reduction, better adaptation of important probabilities

Check Your Knowledge: Arithmetic Coding

1. Explain the iterative interval refinement in Shannon-Fano-Elias coding.
   In which details differs arithmetic coding from Shannon-Fano-Elias coding?
2. Explain how a message is decoded in Shannon-Fano-Elias coding (given the bitstream).
3. Let W represent the final interval width for Shannon-Fano-Elias or arithmetic coding.
   Explain why we have to transmit at least $K = \lceil -\log_2 W \rceil$ bits.
4. What are advantages and disadvantages of arithmetic coding in comparison to scalar Huffman coding, conditional Huffman coding, and block Huffman coding?
5. Why do most implementations in practice use binary arithmetic coding?
6. Explain how arithmetic coding can be combined with an adaptive estimation of probabilities.
7. Can you use conditional probabilities in arithmetic coding?
   Could that help for improving coding efficiency?

Predictive Lossless Coding



Predictive Lossless Coding

1. Predict sample s_n using function of preceding samples: $\hat{s}_n = f(s_{n-1}, s_{n-2}, \cdots)$
2. Entropy coding of prediction error samples: $u_n = s_n - \hat{s}_n$

Effect of Prediction

- Sources with memory: Prediction error has smaller variance and entropy than original data
- Simpler and less complex than conditional arithmetic coding

Linear and Affine Prediction

Affine Predictor: $\boxed{\hat{s}_n = a_0 + \sum_{k=1}^{K} a_k b_k = a_0 + \boldsymbol{a}^{\mathrm{T}} \boldsymbol{b}_n}$ for any observation set $\boldsymbol{b}_n = \{b_1, b_2, \cdots, b_K\}$

➡ Prediction error variance σ_U^2 is minimized by choosing $\boldsymbol{a}$ such that

$$\boldsymbol{C}_B \cdot \boldsymbol{a} = \boldsymbol{c} \qquad \text{with}$$

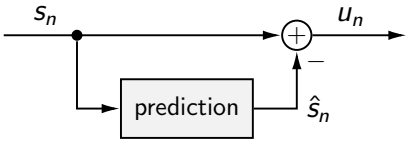
$$\boldsymbol{C}_B = \mathrm{E}\Big\{ \big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big)\big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big)^{\mathrm{T}} \Big\}$$

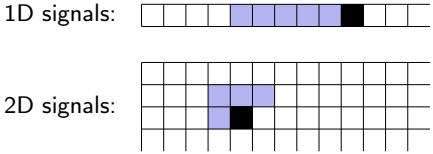
$$\boldsymbol{c} = \mathrm{E}\Big\{ \big(S_n - \mathrm{E}\{S_n\}\big)\big(\boldsymbol{B}_n - \mathrm{E}\{\boldsymbol{B}_n\}\big) \Big\}$$

➡ Mean of prediction error μ_U becomes zero if the offset a_0 is chosen according to

$$a_0 = \mu_S \left(1 - \sum_{k=1}^{K} a_k\right)$$

Practical Aspects of Predictive Coding





Choice of Observation Set

➡ Choose the samples with highest dependencies to current sample

Prediction for Lossless Coding

- Main goal: Minimize entropy of prediction error (approximated by minimization of variance)

➡ Prediction value $\hat{s}_n$ must be rounded to integer

Prediction for Instationary Sources

- Adapt predictor to actual source statistics

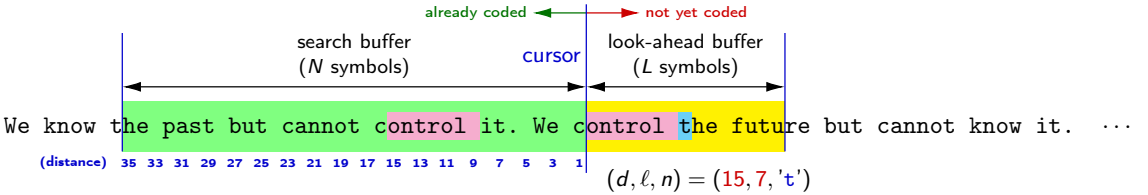
➡ Transmit prediction parameters in regular intervals

➡ Simple variant: Select predictor among set of pre-defined predictors

Check Your Knowledge: Prediction in Lossless Coding

1. Consider linear prediction in lossless image coding. Assume that the correlation coefficient ϱ_{01} between two samples at coordinates (x_0, y_0) and (x_1, y_1) is given by $\varrho_{01} = \varrho^{\sqrt{(x_1 - x_o)^2 + (y_1 - y_o)^2}}$, where ϱ is the correlation coefficient between two directly neighboring samples.
   Write down the Yule-Walker equations for the following three cases:
   - a sample is predicted using the sample to the left;
   - a sample is predicted using the sample to the left and the sample above;
   - a sample is predicted using the sample to the left, the sample above, and the sample left-above.
2. Why do we have to round the predicted value to an integer in lossless coding?
3. What type of statistically dependencies can we utilize using linear and affine prediction?
4. Is it true that lossless coding with prediction can achieve an average codeword length smaller than the entropy rate?
5. Why do many lossless codecs for media signals (audio, images) use prediction and a simple entropy coding method instead of more advanced entropy coding techniques?

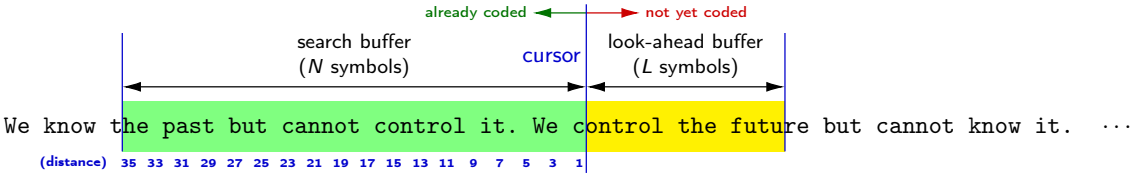
The Lempel-Ziv 1977 Algorithm (LZ77)



The LZ77 Algorithm

- Dictionary of variable-length sequences is given by the preceding N symbols (sliding window)
  - ➡ Find longest possible match for the sequence at the start of the look-ahead buffer
- Message is coded as sequence of triples (d, ℓ, n):
  - ➡ d: distance of best match from next symbol to be coded
  - ➡ ℓ: length of matched phrase (match starts in search buffer but may reach into look-ahead buffer)
  - ➡ n: next symbol after matched sequence
- If no match is found, then $(1, 0, n)$ is coded (with n being the next symbol after the cursor)

LZ77 Variant: The Lempel-Ziv-Storer-Szymanski Algorithm (LZSS)



Changes relative to LZ77 Algorithm

1. At first, code a single bit b to indicate whether a match is found
2. For matches, don't transmit the following symbol

➡ **Message is coded as sequence of tuples** $(b, \{d, \ell\} \mid n)$

- The indication bit b signals whether a match is found ($b = 1 \rightarrow$ match found)
- If ($b = 0$), then code next symbol n as literal
- If ($b = 1$), then code the match as distance-length pair $\{d, \ell\}$ (with $d \in [1, N]$ and $\ell \in [1, L]$)

The Lempel-Ziv 1978 Algorithm (LZ78)

Main Difference to LZ77

- Dictionary is not restricted to preceding N symbols
- Dictionary is constructed during encoding and decoding

The LZ78 Algorithm

- Starts with an empty dictionary
- Next variable-length symbol sequence as coded by tuple $\{k, n\}$
  - k: Index for best match in dictionary (or "0" if no match is found)
  - n: Next symbol (similar to LZ77)
- After coding a tuple $\{k, n\}$, the represented phrase is added to the dictionary

LZ78 Variant: The Lempel-Ziv-Welch (LZW) Algorithm

- Dictionary is initialized with all strings of length one (i.e., all byte codes)
- Next symbol is not included in the code

Check Your Knowledge: Dictionary-Based Coding

1. Explain the main principle of the LZ77 algorithm (or any of its variants) with your own words.
2. Explain the main principle of the LZ78 algorithm (or any of its variants) with your own words. What is the main difference to the LZ77 class of algorithms?
3. Is it possible to improve coding efficiency by combining the LZ77 or LZ78 with advanced entropy coding techniques such as Huffman coding or arithmetic coding?

Lossless Coding: Examples

General File Compression

- Unix **pack**: Marginal Huffman coding (with transmittion of binary tree)
- Unix **compress**: Lempel-Ziv-Welch (LZW) algorithm
- **ZIP**, **gzip**: DEFLATE (Lempel-Ziv-Storer-Szymanski (LZSS) + Huffman coding)
- **7zip**, **xv**, **lzip**: LZ77 Variant (similar to LZSS) + Binary Arithmetic Coding
- **bzip2**: Burrows-Wheeler transform + Move-to-Front transform + Huffman coding

Lossless Audio Compression

- FLAC: Linear prediction (4 types) + Rice coding with adaptive Rice parameter selection

Lossless Image Compression

- PNG: Linear prediction (5 types, selected per row) + DEFLATE
- JPEG-LS: LOCO predictor + 2nd order prediction (context-based) + Rice coding

Mutual Information and Differential Entropy

Mutual Information

- Amount of information a random variable A carries about a random variable B (or vice versa)

➡ Discrete random variable A: $I(A;B) = I(B;A) = H(A) - H(A\,|\,B)$

➡ Continuous random variable A: $I(A;B) = I(B;A) = h(A) - h(A\,|\,B)$

Differential Entropy

- Measure for continuous random variables (different meaning than discrete entropy)

➡ Marginal differential entropy: $h(A) = \mathrm{E}\{-\log_2 f(A)\} = -\int f(a)\,\log_2 f(a)\,\mathrm{d}a$

➡ Conditional differential entropy: $h(A\,|\,B) = \mathrm{E}\{-\log_2 f(A\,|\,B)\} = -\iint f(a,b)\,\log_2 f(a\,|\,b)\,\mathrm{d}a\,\mathrm{d}b$

➡ Differential block entropy: $h_N(\boldsymbol{A}) = \mathrm{E}\{-\log_2 f(A_1,\cdots,A_N)\} = -\int f(\boldsymbol{a})\,\log_2 f(\boldsymbol{a})\,\mathrm{d}\boldsymbol{a}$

➡ Differential entropy rate: $\bar{h}(\boldsymbol{A}) = \lim\limits_{N\to\infty} \dfrac{1}{N}\, h_N(\boldsymbol{A})$

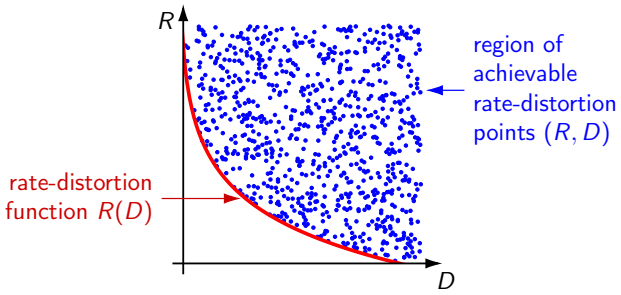
Fundamental Source Coding Theorem

Rate-Distortion Function

➡ Property of source

$$R(D) = \lim_{N\to\infty} \inf_{g_N:\, \delta_N(g_N)\le D} \frac{I_N(g_N)}{N}$$

$$D(R) = \lim_{N\to\infty} \inf_{g_N:\, I_N(g_N)/N\le R} \delta_N(g_N)$$



Fundamental Source Coding Theorem

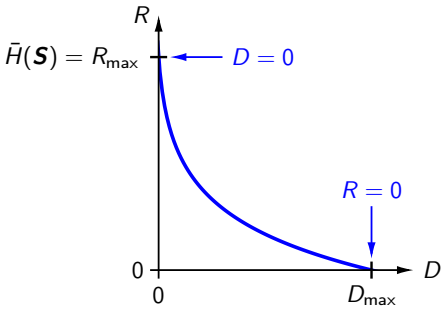
- Rate-distortion function specifies the greatest lower bound for any source code

$$\forall Q: \quad \delta(Q) \le D \quad \Longrightarrow \quad r(Q) \ge R(D)$$

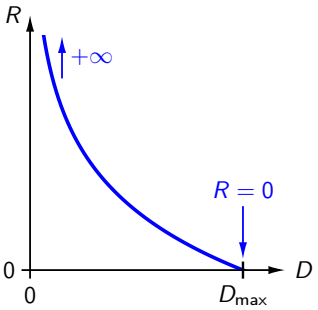
$$\forall Q: \quad r(Q) \le R \quad \Longrightarrow \quad \delta(Q) \ge D(R)$$

Rate-Distortion Function for Discrete and Continuous Sources

Discrete Source



Continuous Source



- $R(D)$ is non-increasing and convex function
- MSE Distortion: $D_{\max} = \sigma_S^2$

High-Rate Approximation: Shannon Lower Bound

Maximization of Differential Entropy

- Gaussian iid process has the largest N-th order differential entropy among all stationary processes with the same variance

$$h_N(\boldsymbol{S}) \;\leq\; h_N^{Giid}(\boldsymbol{S}) = \frac{N}{2}\,\log_2\left(2\pi e\sigma^2\right)$$

Shannon Lower Bound for MSE Distortion

- High-rate approximation of rate-distortion function (asymptotically tight)

$$\boxed{R_L(D) = \bar{h}(\boldsymbol{S}) - \frac{1}{2}\log_2(2\pi e\, D)} \qquad \text{and} \qquad \boxed{D_L(R) = \frac{1}{2\pi e}\cdot 2^{2\bar{h}(\boldsymbol{S})}\cdot 2^{-2R}}$$

➡ General form of distortion-rate function for Shannon lower bound

$$\boxed{D_L(R) \;=\; \varepsilon_L^2\cdot\sigma^2\cdot 2^{-2R}} \qquad \text{with} \qquad \boxed{\varepsilon_L^2 = \frac{1}{2\pi e}\cdot 2^{2\,\bar{h}(\boldsymbol{S}/\sigma)}}$$

Shannon Lower Bound and Rate-Distortion Functions for Selected Sources

Shannon Lower Bound for Selected Sources

$$\text{Uniform IID:} \quad \bar{h}(\boldsymbol{S}) = \frac{1}{2}\log_2\big(12\,\sigma^2\big) \quad \rightarrow \quad D_L(R) = \frac{6}{\pi e}\cdot\sigma^2\cdot 2^{-2R}$$

$$\text{Laplacian IID:} \quad \bar{h}(\boldsymbol{S}) = \frac{1}{2}\log_2\big(2e^2\sigma^2\big) \quad \rightarrow \quad D_L(R) = \frac{e}{\pi}\cdot\sigma^2\cdot 2^{-2R}$$

$$\text{Gaussian IID:} \quad \bar{h}(\boldsymbol{S}) = \frac{1}{2}\log_2\big(2\pi e\,\sigma^2\big) \quad \rightarrow \quad D_L(R) = \sigma^2\cdot 2^{-2R}$$

$$\text{Gauss-Markov:} \quad \bar{h}(\boldsymbol{S}) = \frac{1}{2}\log_2\big(2\pi e\,(1-\varrho^2)\,\sigma^2\big) \quad \rightarrow \quad D_L(R) = (1-\varrho^2)\cdot\sigma^2\cdot 2^{-2R}$$

Rate-Distortion Function for Selected Gaussian Sources

$$\text{Gaussian IID:} \quad D(R) = D_L(R) = \sigma^2\cdot 2^{-2R} \qquad \text{for all rates } R \geq 0$$

$$\text{Gauss-Markov:} \quad D(R) = D_L(R) = (1-\varrho^2)\cdot\sigma^2\cdot 2^{-2R} \qquad \text{for rates } R \geq \log_2(1+\varrho)$$

Check Your Knowledge: Rate-Distortion Theory

1. What does the rate-distortion function $R(D)$ for a given source tell us?

2. Sketch the rate-distortion function for a discrete source and a continuous source.

3. The rate-distortion function is defined by

$$R(D) = \lim_{N \to \infty} \inf_{g_N : \delta_N(g_N) \leq D} \frac{I_N(g_N)}{N}$$

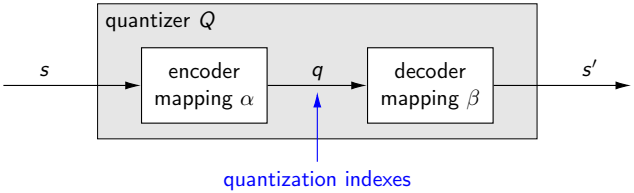
   Show that the fundamental lossless coding theorem is a special case of the fundamental source coding theorem.

4. What is the Shannon lower bound? Explain its relation to the rate-distortion function.

5. Consider the Shannon lower bound. Write down the general form of the distortion-rate function for MSE distortion.

   By what amount do we have to increase the bit rate in order to get a distortion reduction of 50%?

Scalar Quantization



- **Encoder mapping** α: Maps input sample s to a quantizer index q (integer)

$$q = \alpha(s)$$

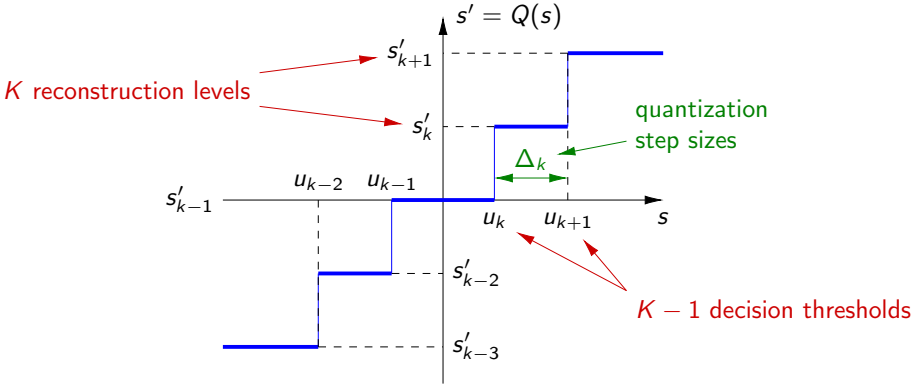
- **Decoder mapping** β: Maps quantizer index q to reconstructed samples s'

$$s' = \beta(q)$$

➡ Overall input-output function: Maps input sample s to reconstructed sample s'

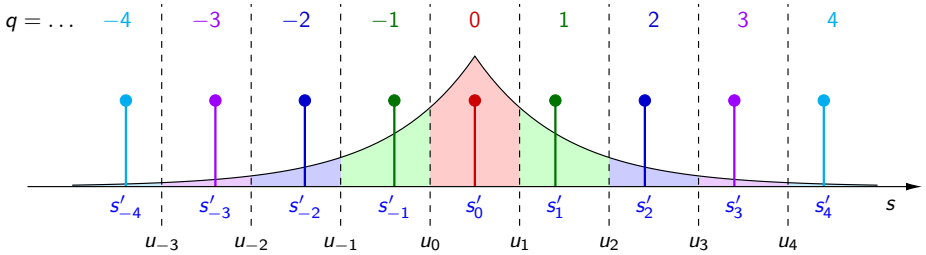
$$s' = Q(s) = \beta(\alpha(s))$$

Input-Output Function of Scalar Quantizers



- Scalar quantizer mapping: $Q: \mathbb{R} \mapsto \{\cdots, s'_{k-1}, s'_k, s'_{k+1}, \cdots\}$
- Quantization intervals: $\mathcal{I}_k = [u_k, u_{k+1})$
- Quantization step sizes: $\Delta_k = u_{k+1} - u_k$

Coding Efficiency of Scalar Quantizers



➡ **Bit rate** R and **MSE distortion** D of a scalar quantizer Q (and associated entropy coding)

$$R \;=\; \mathrm{E}\Big\{\; \ell\big(\,Q(S)\,\big)\;\Big\} \;=\; \sum_{\forall k} p_k \cdot \ell_k \;=\; \sum_{\forall k} \ell_k \int_{u_k}^{u_{k+1}} f(s)\,\mathrm{d}s$$

$$D \;=\; \mathrm{E}\Big\{\; \big(\,S - Q(S)\,\big)^2\;\Big\} \;=\; \sum_{\forall k} \int_{u_k}^{u_{k+1}} \big(s - s'_k\big)^2\, f(s)\,\mathrm{d}s$$

Lloyd Quantizer: Minimization of Distortion

Design Goal

- Neglect entropy coding or assume fixed-length coding: $R = \log_2 K$
- ➡ Minimize distortion for given quantizer size K

Optimization Criterions for MSE distortion

1 Centroid condition: $$s'_k \;=\; \mathrm{E}\{\, S \,|\, S \in \mathcal{I}_k \,\} \;=\; \frac{\int_{u_k}^{u_{k+1}} s\, f(s)\, \mathrm{d}s}{\int_{u_k}^{u_{k+1}} f(s)\, \mathrm{d}s}$$

2 Nearest neighbour condition: $$u_k \;=\; \frac{1}{2}\left(s'_{k-1} + s'_k\right)$$

Design Algorithm

- Choose quantizer size K
- Iterate between optimization criterions (using pdf or training set)

Minimization of Distortion and Bit Rate: Lagrange Optimization

Constrained Optimization Problem

- Can be formulated as

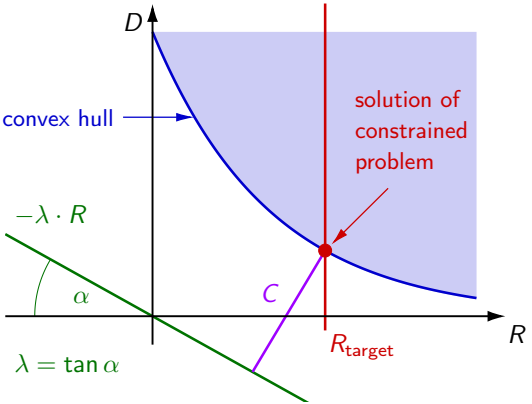
$$\min D \quad \text{subject to} \quad R \leq R_{\text{target}}, \quad \text{or}$$
$$\min R \quad \text{subject to} \quad D \leq D_{\text{target}}$$

Lagrangian Optimization Problem

- Reformulation as unconstrained problem

$$\min D + \lambda \cdot R \quad \text{with} \quad \lambda > 0$$

- Solution for a given λ is also a solution of the original problem for some R_{target}



- Geometrical interpretation: Minimize distance C to a line $D = -\lambda \cdot R$
- Solutions of Lagrange optimization problems (for all $\lambda > 0$) lie on convex hull

Entropy-Constrained Lloyd Quantizer: Optimal Scalar Quantization

Design Goal

- Assume optimal entropy coding: $R = H(S')$

➡ Minimize Lagrangian cost $J = D + \lambda \cdot R$ (for given operation point, i.e., given λ)

Optimization Criterions for MSE distortion

1 Centroid condition:
$$s'_k \;=\; \mathrm{E}\{\, S \,|\, S \in \mathcal{I}_k \,\} \;=\; \frac{\int_{u_k}^{u_{k+1}} s\, f(s)\, \mathrm{d}s}{\int_{u_k}^{u_{k+1}} f(s)\, \mathrm{d}s}$$

2 Entropy condition:
$$\ell_k \;=\; -\log_2 p_k \;=\; -\log_2 \int_{u_k}^{u_{k+1}} f(s)\, \mathrm{d}s$$

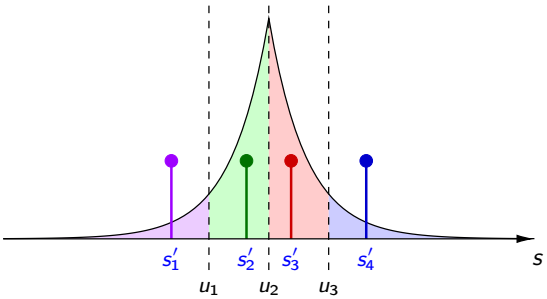
3 Modified nearest neighbour condition:
$$u_k \;=\; \frac{1}{2}\left(s'_{k-1} + s'_k\right) + \frac{\lambda}{2}\left(\frac{\ell_k - \ell_{k-1}}{s'_k - s'_{k-1}}\right)$$

Design Algorithm

- Choose Lagrange multiplier λ and initial quantizer size K (should be large enough)
- Iterate between optimization criterions (using pdf or training set)

Example: EC Lloyd vs Lloyd at Same Entropy (Laplace)

Lloyd Algorithm



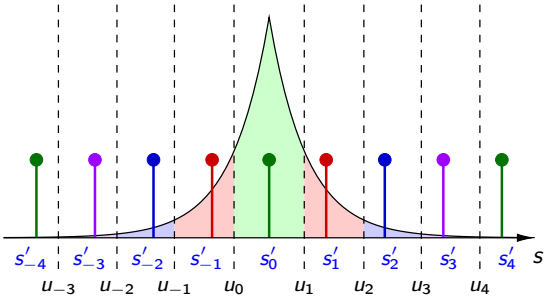
$K = 4 \qquad (R_{\text{FL}} = 2.0)$

$H = 1.728$

$D = 0.176$

$\text{SNR} = 7.54\,\text{dB}$

Entropy-Constrained Lloyd Algorithm



$\lambda = 0.1350$

$H = 1.728$

$D = 0.104$ ➡ factor 0.59 smaller

$\text{SNR} = 9.83\,\text{dB}$ ➡ 2.29 dB better

High-Rate Approximations for Scalar Quantizers

High-Rate Distortion-Rate Function for MSE Distortion

- General form of distortion-rate function

$$D_X(R) = \varepsilon_X^2 \cdot \sigma^2 \cdot 2^{-2R}$$

where the constant factor ε_X^2 depends on

➡ shape of pdf and

➡ quantizer design

Lloyd + FLC:	$\varepsilon_F^2 = \frac{1}{12} \left(\int_{-\infty}^{\infty} \sqrt[3]{f(s/\sigma)}\, \mathrm{d}s \right)^3$
EC-Lloyd + VLC:	$\varepsilon_V^2 = \frac{1}{12}\, 2^{2\,h(S/\sigma)}$
Shannon lower bound:	$\varepsilon_L^2 = \frac{1}{2\pi e}\, 2^{2\,h(S/\sigma)}$

Comparison of Coding Efficiency

- EC-Lloyd often significantly better than Lloyd (Gauss: 2.82 dB; Laplace: 5.63 dB)
- Constant performance gap between high-rate EC-Lloyd and Shannon lower bound

$$\frac{D_V(R)}{D_L(R)} = \frac{\pi e}{6} \approx 1.42 \;\; (1.53 \text{ dB}), \qquad R_V(D) - R_L(D) = \frac{1}{2} \log_2 \frac{\pi e}{6} \approx 0.25$$

Comparison of Quantizers and High-Rate Approximations: Laplacian Source

High-rate approximations

$$D_X(R) = \varepsilon_X^2 \cdot \sigma^2 \cdot 2^{-2R}$$

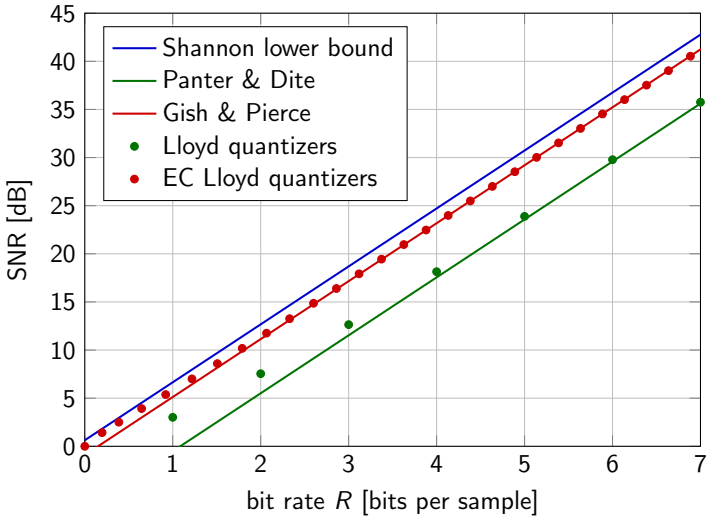
SLB: $\varepsilon_L^2 = \frac{e}{\pi}$

Lloyd: $\varepsilon_F^2 = \frac{9}{2}$

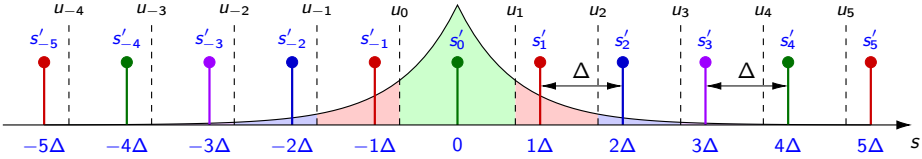
EC-Lloyd: $\varepsilon_V^2 = \frac{e^2}{6}$

$$\frac{D_F}{D_L} = \frac{9\,\pi}{2\,e} \approx 5.20 \quad (7.16\,\text{dB})$$

$$\frac{D_V}{D_L} = \frac{\pi\,e}{6} \approx 1.42 \quad (1.53\,\text{dB})$$



In Practice: Uniform Reconstruction Quantizers (URQs)



Uniform reconstruction quantizers

- Equally spaced reconstruction levels (indicated by quantization step size Δ)

➡ **Decoder:**
- Reconstruction levels are completely specified by **quantization step size Δ**
- Simple decoding process: $s'_n = \Delta \cdot q_n$

➡ **Encoder:**
- Freedom to adapt decision thresholds to source statistics
- Simple encoding (rounding) or advanced encoding (Lagrange optimization)

- For typical pdfs: Negligible loss relative to optimal scalar quantizers

Check Your Knowledge: Scalar Quantization

1. What is a scalar quantizer? What parameters impact the rate-distortion efficiency?
   Write down a formula for the MSE distortion of a scalar quantizer (for a given pdf).

2. Why are most decisions in real-world encoders based on Lagrangian optimization?

3. Assume you are given a large training set for a given source.
   Explain how you could derive an optimal scalar quantizer (with variable-length entropy coding)?

4. What are uniform reconstruction quantizers? And why they are typically used in practice instead of optimal scalar quantizers?

5. Given is a decoder, which consists of scalar Huffman decoding and a uniform reconstruction quantizer. That means, the decoder entropy decodes quantization indexes q using a pre-defined codeword table. And given the quantization indexes, the reconstructed samples are obtained according to $s' = \Delta \cdot q$, where Δ is a pre-defined quantization step size.
   Your task is to implement an optimal encoder for the given decoder. How would you determine the quantization indexes in the encoder (for a given signal)?

Vector Quantization

Vector Quantizers of Quantizer Dimension N

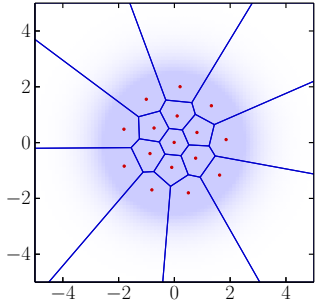
- Map N-d input vectors $\boldsymbol{s}$ to N-d output vectors $\boldsymbol{s'_k}$

$$Q: \ \mathbb{R}^N \mapsto \{ \boldsymbol{s'_0}, \boldsymbol{s'_1}, \boldsymbol{s'_2}, \cdots \}$$

- Partition N-d space into countable number of quantization cells $\mathcal{C}_k$

$$\mathcal{C}_k = \{ \boldsymbol{s} \in \mathbb{R}^N : \ Q(\boldsymbol{s}) = \boldsymbol{s'_k} \}$$

- All input vectors $\boldsymbol{s}$ that fall inside a quantization cell $\mathcal{C}_k$ are mapped to the associated reconstruction vector $\boldsymbol{s'_k}$



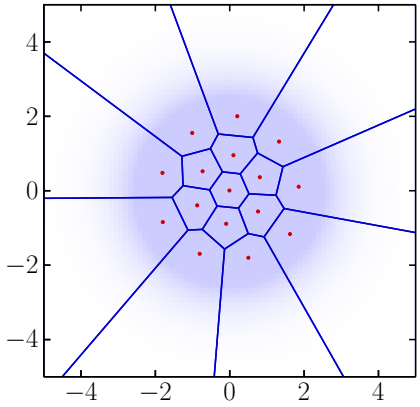
Vector Quantization and Entropy Coding

- Quantization index k indicates quantization cell $\mathcal{C}_k$ and reconstruction vector $\boldsymbol{s'_k}$
  - ➡ Encoder mapping: $\alpha(\boldsymbol{s}) = k, \quad \forall \boldsymbol{s} \in \mathcal{C}_k$
  - ➡ Decoder mapping: $\beta(k) = \boldsymbol{s'_k}$

Performance of Vector Quantizers

➡ **Average MSE distortion D per sample**

$$D = \frac{1}{N} \mathrm{E}\Big\{ \, || \, \boldsymbol{s} - Q(\boldsymbol{s}) \, ||_2^2 \, \Big\} = \frac{1}{N} \sum_{\forall k} \int_{\mathcal{C}_k} || \, \boldsymbol{s} - \boldsymbol{s}'_k \, ||_2^2 \, f(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}$$



➡ **Average bit rate R per sample**

$$R = \frac{1}{N} \mathrm{E}\Big\{ \, \ell(\, Q(\boldsymbol{s}) \,) \, \Big\} = \frac{1}{N} \sum_{\forall k} p_k \, \ell_k$$

with probability of quantization cell / index

$$p_k = \int_{\mathcal{C}_k} f(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}$$

Optimal Vector Quantizer with Fixed-Length Coding

Extension of Lloyd Design to Higher Dimensions

- Neglect entropy coding: Assume fixed-length coding: $R = (\log_2 K)/N$

➡ Minimize distortion D for given quantizer size K

Optimization Criterions for MSE distortion

1 Centroid condition:
$$\boldsymbol{s}'_{\boldsymbol{k}} \;=\; \mathrm{E}\{\, \boldsymbol{S} \,|\, \boldsymbol{S} \in \mathcal{C}_k \,\} \;=\; \frac{\int_{\mathcal{C}_k} \boldsymbol{s}\, f(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}}{\int_{\mathcal{C}_k} f(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}}$$

2 Nearest neighbour condition:
$$\alpha(\boldsymbol{s}) \;=\; \arg\min_{\forall k} \left\| \boldsymbol{s} - \boldsymbol{s}'_{\boldsymbol{k}} \right\|_2^2$$

The Linde-Buzo-Gray (LBG) Algorithm

- Choose quantizer size K
- Iterate between optimization criterions (using training set)

Optimal Vector Quantizer with Variable-Length Coding

Extension of EC-Lloyd Design to Higher Dimensions

- Assume optimal entropy coding: $R = H_N(S')/N$

➡ Minimize Lagrangian cost $J = D + \lambda \cdot R$ (for given operation point, i.e., given λ)

Optimization Criterions for MSE distortion

1 Centroid condition:
$$\boldsymbol{s}'_k \;=\; \mathrm{E}\{\, \boldsymbol{S} \,|\, \boldsymbol{S} \in \mathcal{C}_k \,\} \;=\; \frac{\int_{\mathcal{C}_k} \boldsymbol{s}\, f(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}}{\int_{\mathcal{C}_k} f(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}}$$

2 Entropy condition:
$$\ell_k \;=\; -\log_2 p_k \;=\; -\log_2 \int_{\mathcal{C}_k} f(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}$$

3 Modified nearest neighbour condition:
$$\alpha(\boldsymbol{s}) \;=\; \arg\min_{\forall k} \left\| \boldsymbol{s} - \boldsymbol{s}'_k \right\|_2^2 + \lambda \cdot \ell_k$$

The Chou-Lookabough-Gray (CLG) Algorithm

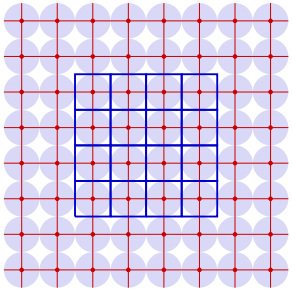
- Choose Lagrange multiplier λ and initial quantizer size K (should be large enough)
- Iterate between optimization criterions (using training set)

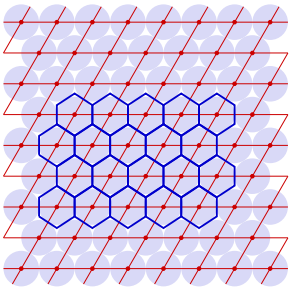
The Vector Quantizer Advantage

Gain over scalar quantization can be assigned to 3 effects:

- **Space filling advantage**:
  - $\mathbb{Z}^N$ lattice is not most efficient sphere packing in N dimensions ($N > 1$)
  - Independent from source distribution or statistical dependencies
  - Maximum gain for $N \to \infty$: 1.53 dB
- **Shape advantage**:
  - Exploit shape of source pdf (even without entropy coding)
  - Can also be exploited using entropy-constrained scalar quantization
- **Memory advantage**:
  - Exploit statistical dependencies of the source
  - Can also be exploited using predictive coding, transform coding, block entropy coding or conditional entropy coding

Space-Filling Advantage: Sphere Packing in N-dimensional Signal Space





- Space filling gain: Densest packing of "optimal" quantization cells in signal space

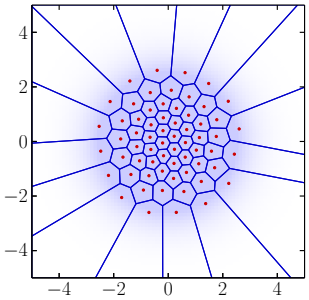
➡ MSE distortion: **Densest packing of spheres in N-dimensional space**

- ➡ 2 dimensions: Hexagonal lattice (like honeycombs)
- ➡ 3 dimensions: Cuboidal lattice (stapling of cannon balls / oranges)

➡ Space filling gain for MSE distortion approaches 1.53 dB for $N \to \infty$

Example: Gaussian IID ($\sigma^2 = 1$) at 3 Bits per Sample

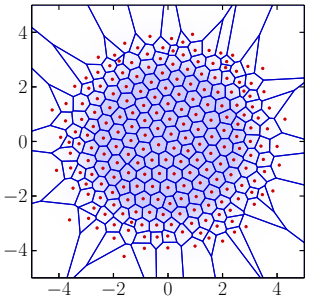
LBG ($N = 2$)



$D = 0.0296$

SNR $= 15.29\,\text{dB}$

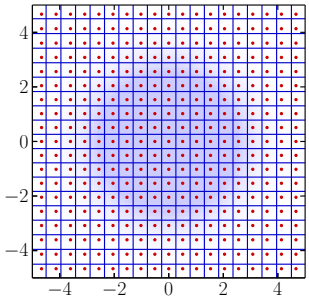
CLG ($N = 2$)



$D = 0.0214$

SNR $= 16.70\,\text{dB}$

EC Lloyd ($N = 1$)



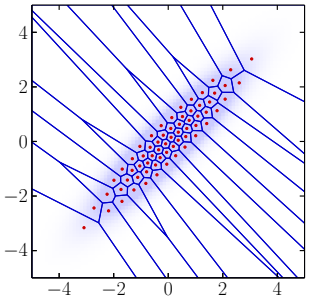
$D = 0.0222$

SNR $= 16.53\,\text{dB}$

- Large gain (1.4 dB) for CLG relative to LBG algorithm (variable-length vs fixed-length coding)
- Gain of CLG relative to EC Lloyd reduces to space-filling gain (0.17 dB for $N = 2$)

Example: Gauss-Markov ($\sigma^2 = 1$, $\varrho = 0.9$) at 3 Bits per Sample

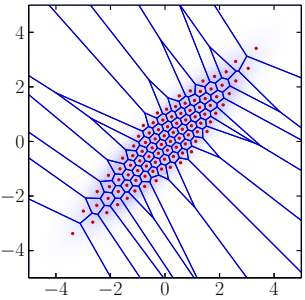
LBG ($N = 2$)



$D = 0.0125$

SNR $= 19.04\,\text{dB}$

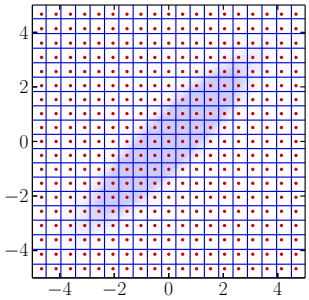
CLG ($N = 2$)



$D = 0.0099$

SNR $= 20.06\,\text{dB}$

EC Lloyd ($N = 1$)



$D = 0.0222$

SNR $= 16.53\,\text{dB}$

- Large gain (1.0 dB) for CLG relative to LBG algorithm (variable-length vs fixed-length coding)
- Gain of CLG relative to EC Lloyd: Sum of memory gain and space-filling gain

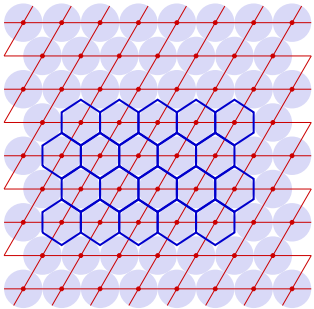
Lattice Vector Quantizers

Lattice Vector Quantizer

- Reconstruction vectors are located on multi-dimensional lattice
  - Lattice is specified by N "basis vectors" $\{\boldsymbol{b}_k\}$
  - Reconstruction vectors given by matrix of "basis vectors"

$$\boldsymbol{s}'_{k_1,k_2,\cdots,k_N} = \boldsymbol{M} \cdot [k_1, k_2, \cdots, k_N]^{\mathrm{T}}$$

- Simple decoder operation possible
- Less complex encoding (can still by very complex for large N)



Transform Coding

- Lattice vector quantizer with orthonormal "basis vectors"
- Very simple encoding and decoding

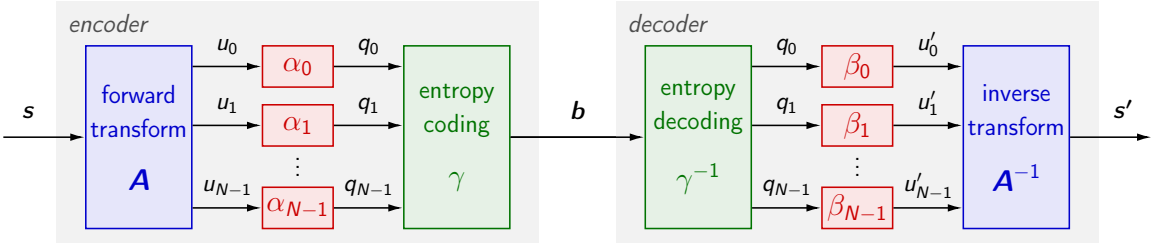
➡ One of the most often used approaches in lossy coding

Check Your Knowledge: Vector Quantization

1. What is a vector quantizer? What is the main difference to a scalar quantizer?
2. Why can vector quantizers achieve a higher coding efficiency than scalar quantizers?
   Explain the three vector quantizer advantages.
3. Is it true that vector quantizers can improve the coding efficiency only for sources with memory?
4. Assume you are given a large training set for a considered source.
   How could you design an optimal vector quantizer for the considered source.
5. Why are (unconstrained) vector quantizers rarely used in practice?

Transform Coding

- Transform removes (or reduces) linear dependencies between samples before scalar quantization
- For correlated sources: Scalar quantization in transform domain is more efficient



Encoder (block-wise)

- ➡ Forward transform: $\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s}$
- ➡ Scalar quantization: $q_k = \alpha_k(u_k)$
- ➡ Entropy coding: $\boldsymbol{b} = \gamma(\{q_k\})$

Decoder (block-wise)

- ➡ Entropy decoding: $\{q_k\} = \gamma^{-1}(\boldsymbol{b})$
- ➡ Inverse quantization: $u'_k = \beta_k(q_k)$
- ➡ Inverse transform: $\boldsymbol{s'} = \boldsymbol{A}^{-1} \cdot \boldsymbol{u'}$

Orthogonal Block Transforms

- Transform matrix has property: $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}}$ (special case of unitary matrix)

$$\boldsymbol{A} = \begin{bmatrix} \text{—} & \boldsymbol{b}_0 & \text{—} \\ \text{—} & \boldsymbol{b}_1 & \text{—} \\ \text{—} & \boldsymbol{b}_2 & \text{—} \\ & \vdots & \\ \text{—} & \boldsymbol{b}_{N-1} & \text{—} \end{bmatrix} \qquad \boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{b}_0 & \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_{N-1} \\ | & | & | & & | \end{bmatrix}$$

➡ Signal is represented as weighted sum of basis vectors: $\boldsymbol{s} = \sum_k u_k \boldsymbol{b}_k, \quad u_k = \boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{s}$

➡ Geometric interpretation: Rotation (and potential reflection) in N-dimensional signal space

Properties of Orthogonal Transforms

- Preservation of signal energy / vector length: $||\boldsymbol{A} \cdot \boldsymbol{s}||_2 = ||\boldsymbol{s}||_2$
- Same MSE distortion in sample and transform space: $||\boldsymbol{u'} - \boldsymbol{u}||_2^2 = ||\boldsymbol{s'} - \boldsymbol{s}||_2^2$
- Auto-covariance matrix of transform coefficients: $\boldsymbol{C}_{UU} = \boldsymbol{A} \cdot \boldsymbol{C}_{SS} \cdot \boldsymbol{A}^{\mathrm{T}}$
- Sum of variances of transform coefficients: $\sum_k \sigma_k^2 = N \cdot \sigma_S^2$

Optimal Bit Allocation and High-Rate Approximations

Bit Allocation of Transform Coefficients

- Optimal bit allocation: Pareto condition

$$\min D + \lambda \cdot R \quad \Rightarrow \quad \frac{\partial}{\partial R_k} D_k(R_k) = -\lambda = \text{const}$$

High-Rate Approximation

- Optimal bit allocation for high-rate case

$$D_k(R_k) = D = \text{const} \qquad (\text{for URQs: } \Delta_k = \Delta = \text{const})$$

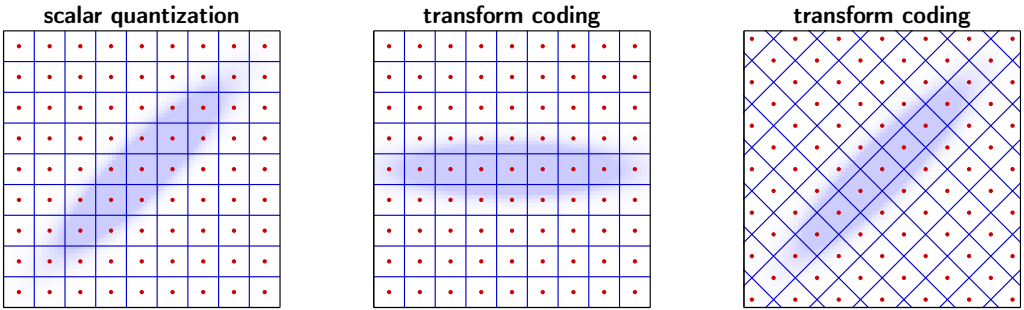
- High-rate distortion rate function for transform coding

$$D(R) = \tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R} \qquad \text{with} \qquad \tilde{\varepsilon}^2 = \left(\prod\nolimits_k \varepsilon_k^2 \right)^{\frac{1}{N}}, \quad \tilde{\sigma}^2 = \left(\prod\nolimits_k \sigma_k^2 \right)^{\frac{1}{N}}$$

- High-rate transform coding gain G_T and energy-compaction measure G_{EC}

$$G_T = \frac{D_{SQ}(R)}{D_{TC}(R)} = \frac{\varepsilon_S^2 \cdot \sigma_S^2}{\tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2}, \qquad G_{EC} = \frac{\sigma_S^2}{\tilde{\sigma}^2} = \frac{\frac{1}{N} \sum_k \sigma_k^2}{\sqrt[N]{\prod_k \sigma_k^2}}$$

Transform Coding as Constrained Vector Quantizer



- Quantization cells are:
  - hyper-rectangles as in conventional scalar quantization
  - but rotated and aligned with the transform basis vectors

➡ Correlated sources: Uneven distribution of transform coefficient variances

The Karhunen Loève Transform (KLT)

- Design criterion: Orthogonal transform $\boldsymbol{A}$ that yields uncorrelated transform coefficients

$$\boldsymbol{C}_{UU} = \boldsymbol{A} \cdot \boldsymbol{C}_{SS} \cdot \boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} \sigma_0^2 & 0 & \cdots & 0 \\ 0 & \sigma_1^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{N-1}^2 \end{bmatrix} \quad \Longrightarrow \quad \boldsymbol{C}_{SS} \cdot \boldsymbol{b}_k = \sigma_k^2 \cdot \boldsymbol{b}_k$$

➡ Eigenvector equation for all basis vectors $\boldsymbol{b}_k$ (rows of transform matrix $\boldsymbol{A}$)

➡ Rows of KLT matrix $\boldsymbol{A}$ are the unit-norm eigenvectors of $\boldsymbol{C}_{SS}$

➡ Transform coefficient variances σ_k^2 are the eigenvalues of $\boldsymbol{C}_{SS}$

$$\boldsymbol{A} = \begin{bmatrix} — & \boldsymbol{b}_0 & — \\ — & \boldsymbol{b}_1 & — \\ & \vdots & \\ — & \boldsymbol{b}_{N-1} & — \end{bmatrix} \qquad \boldsymbol{C}_{UU} = \begin{bmatrix} \sigma_0^2 & 0 & \cdots & 0 \\ 0 & \sigma_1^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{N-1}^2 \end{bmatrix}$$

Coding Efficiency of Transform Coding with KLT

High-Rate Approximation for KLT and Gauss-Markov

- KLT is the optimal tranform for Gaussian sources
- High-rate operational distortion-rate function and transform coding gain (for transform size N)

$$D_N(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1 - \varrho^2)^{\frac{N-1}{N}} \cdot 2^{-2R} \qquad \text{and} \qquad G_T^N = G_{EC}^N = (1 - \varrho^2)^{\frac{1-N}{N}}$$

➡ Transform gain increases with transform size N, but approaches a limit

Comparison to Rate-Distortion Bound

- Example: Gauss-Markov, KLT, and optimal scalar quantizers (ECSQ)

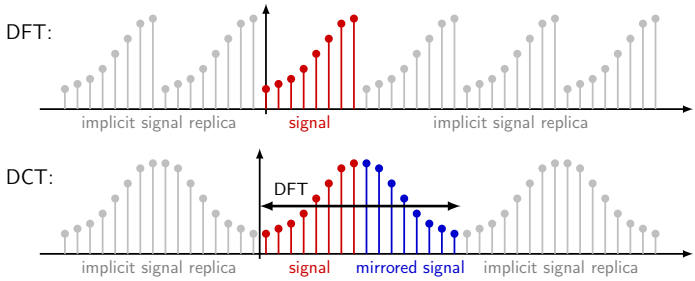
➡ Distortion increase relative to Shannon lower bound

$$\frac{D_N^{(\mathrm{KLT})}(R)}{D_{\mathrm{SLB}}(R)} = \frac{\frac{\pi e}{6} \cdot \sigma_S^2 \cdot (1 - \varrho^2)^{\frac{N-1}{N}} \cdot 2^{-2R}}{\sigma_S^2 \cdot (1 - \varrho^2) \cdot 2^{-2R}} = \frac{\pi e}{6} \cdot \left(\frac{1}{1 - \varrho^2} \right)^{\frac{1}{N}}$$

➡ Large transform sizes ($N \to \infty$): Performance gap reduces to space-filling gain (1.53 dB)

- Other sources: Transform coding cannot utilize all dependencies (non-linear dependencies)

Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT)



Disadvantage of Discrete Fourier Transform

- Sampling of frequency spectrum causes implicit periodic signal extension

➡ Large differences between signal borders reduces rate of convergence of Fourier series

Idea of Discrete Cosine Transform

- Introduce mirror symmetry and apply DFT of approximately double size

➡ No discontinuities in periodic signal extension

The Discrete Cosine Transform of Type II (DCT-II)

Transform Matrix of the Discrete Cosine Transform

- The DCT is an orthogonal transform
- The transform matrix $\boldsymbol{A}_{DCT} = \{a_{kn}\}$ has the elements

$$a_{kn} = \alpha_k \cdot \cos\left(\frac{\pi}{N}\, k \left(n + \frac{1}{2}\right)\right) \qquad \text{with} \qquad \alpha_k = \begin{cases} \sqrt{1/N} & : \quad k = 0 \\ \sqrt{2/N} & : \quad k \neq 0 \end{cases}$$

- The basis vectors $\boldsymbol{b}_k = \{a_{kn}\}$ represent sampled cosine functions of different frequencies

Relation to KLT

- Unit-norm eigenvectors of $\boldsymbol{C}_{SS}$ approach DCT basis vectors for $\varrho \to 1$
- For typical sources: Very small loss in energy compaction relative to KLT

Advantages of DCT

- Transform matrix does not depend on the input signal
- Fast algorithms for computing the forward and inverse transforms

Basis Functions of the DCT-II (Example for $N = 8$)

$$\boldsymbol{b_k}[n] = \alpha_k \cdot \cos\left(\frac{\pi}{N}\, k\, \left(n + \frac{1}{2}\right)\right)$$

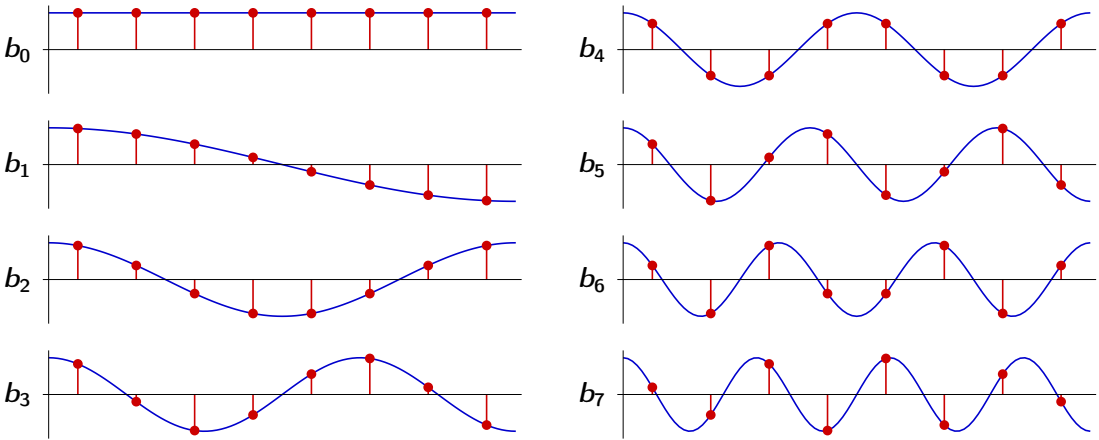


Image & Video Coding: Separable 2D Transforms

Separable Transforms for Blocks of Samples

- Successive 1D transforms of rows and columns

➡ Separable forward transform for $N \times N$ block

$$\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s} \cdot \boldsymbol{A}^{\mathrm{T}}$$

➡ Separable inverse transform for $N \times N$ block

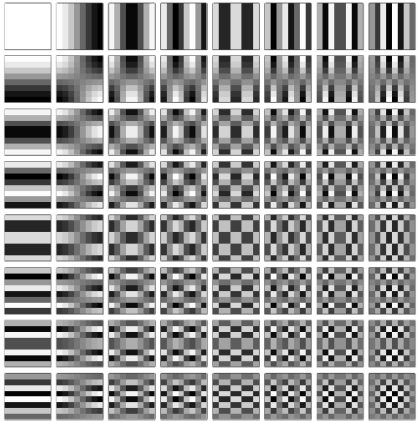
$$\boldsymbol{s} = \boldsymbol{A}^{\mathrm{T}} \cdot \boldsymbol{u} \cdot \boldsymbol{A}$$

Practical importance:

➡ Complexity reduction from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$

➡ Non-separable transforms hard to design (except KLTs)

Basis blocks of 8×8 DCT



Non-Separable KLT vs Separable DCT-II (sorted transform coefficients)

original

4×4 **non-separable KLT**

$G_{EC} = 23.804$ dB

4×4 **separable DCT-II**

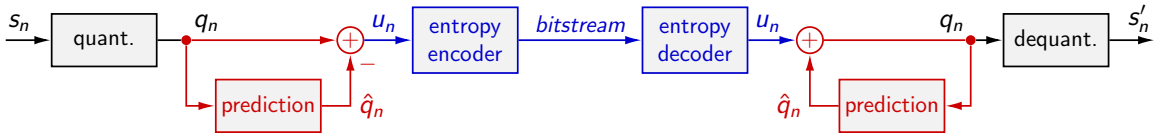
$G_{EC} = 23.629$ dB

➡ Energy compaction slightly decreases due to usage of separable and signal-independent transform

Check Your Knowledge: Transform Coding

1. Explain the concept of transform coding. Why does it typically improve coding efficiency in comparison to simple scalar quantization?
2. What is the main reason for using orthogonal transforms in practice?
3. How does the bit allocation among transform coefficients impact the coding efficiency?
   What is the optimal bit allocation (general and high rates)?
   How is a nearly optimal bit allocation achieved in practice?
4. What is the Karhunen Loève Transform (KLT)?
   Show that the following transform is a KLT for all stationary sources:
   $$\boldsymbol{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
5. Why do most codecs use the Discrete Cosine Transform (DCT) and not a KLT?
   What is the relation of the DCT and the Discrete Fourier Transform?
6. What is a separable transform? Why are separable 2d transforms used in image coding?

Lossy Coding with Prediction: Prediction of Quantization Indexes



Prediction after Quantization

- Sources with memory: Quantization indexes have statistical dependencies

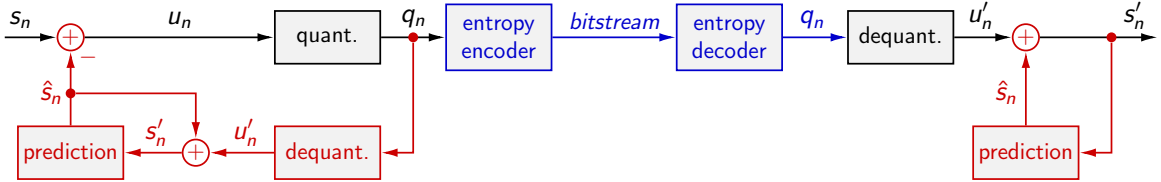
➡ Prediction can improve lossless coding of quantization indexes

- More accurate: With prediction, entropy coding can be simplified
- Require: Predicted value $\hat{q}_n$ must be integer (since u_n must be integer)

Extension: Prediction after Transform and Quantization

- Prediction of some transform coefficients (e.g., DC coefficient)
- Examples: JPEG, MPEG-2 Video, H.263, MPEG-4 Visual

Lossy Coding with Prediction: Quantization of Prediction Error



Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)

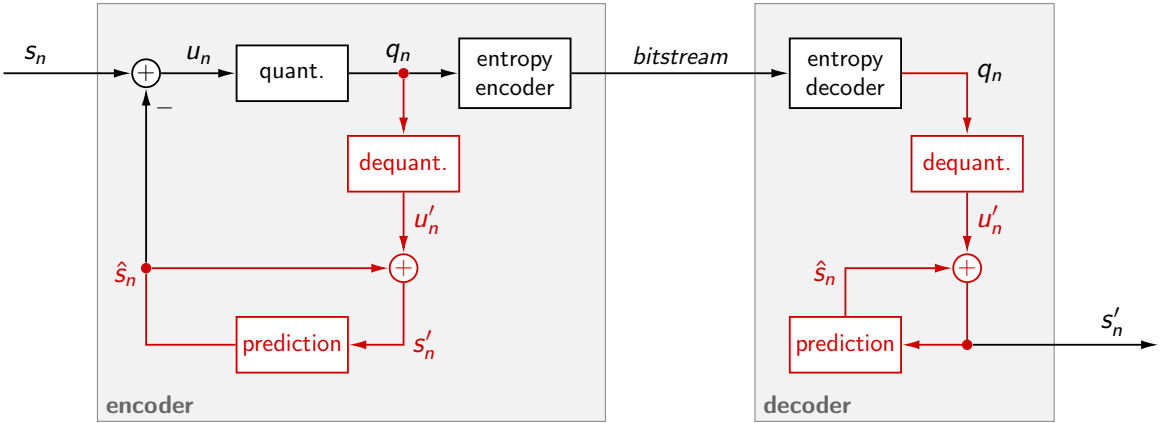
➡ Prediction value $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$$

➡ Requires reconstruction of samples at encoder side

Differential Pulse Code Modulation (DPCM)

- Encoder contains decoder except for entropy decoding
- Scalar quantizer can be replaced by transform and quantization (DPCM of blocks)



DPCM for Gauss-Markov Sources

DPCM with Scalar Quantization

- Approximation of operational distortion-rate function

$$D(R) \;=\; \sigma_U^2 \cdot g(R) \;=\; \sigma_S^2 \cdot \frac{1-\varrho^2}{1-\varrho^2 \cdot g(R)} \cdot g(R)$$

➡ Worse than transform coding at low rates (strong quantization impacts quality of prediction)

High-Rate Approximation

- High rates R: $\quad g(R) = \varepsilon^2 \cdot 2^{-2R} \quad$ and $\quad g(R) \ll 1$

➡ Operational distortion-rate function at high rates

$$D(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1-\varrho^2) \cdot 2^{-2R}$$

➡ Same formula as for transform coding with large KLT $\quad (N \to \infty)$

➡ Gap to rate-distortion bound represents space-filling gain of vector quantization

Lossy Source Coding: DPCM or Transform Coding ?

Differential Pulse Coding Modulation (DPCM)

- Worse coding efficiency than transform coding at low rates (interesting opertation points)
- Audio, images, video: Perceptual artifacts when using rather strong quantization

Transform Coding

- Better coding efficiency than DPCM at low rates (interesting operation points)
- Better subjective quality than DPCM: Can better control perceived quality (frequency spectrum)
- Cannot utilize statistical dependencies between blocks (only inside blocks)

Combination of Transform Coding and Prediction

- Improve coding efficiency of transform coding by prediction between transform blocks
- Two concepts:
  1. Prediction after quantization: Only for certain transform coefficients
  2. Prediction before transform: DPCM with transform coding as quantizer

Check Your Knowledge: Predictive Quantization

1. Explain two possibilities how the usage of prediction can improve the coding efficiency of lossy codecs.

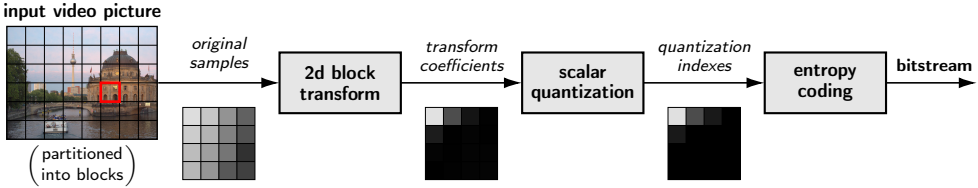
2. Explain the concept of differential pulse code modulation (DPCM) based on a block diagram of an encoder.

   Why do we have to derive the predicted value based on reconstructed samples?

3. Does it make sense to use both transform coding and prediction in a codec? Why or why not?

   How could these two concepts be combined?

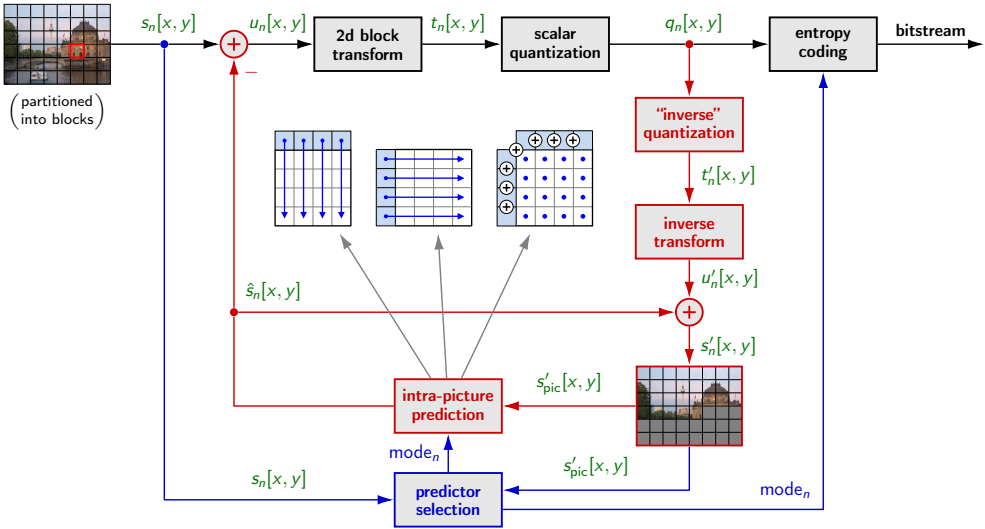
Image Coding Example: JPEG



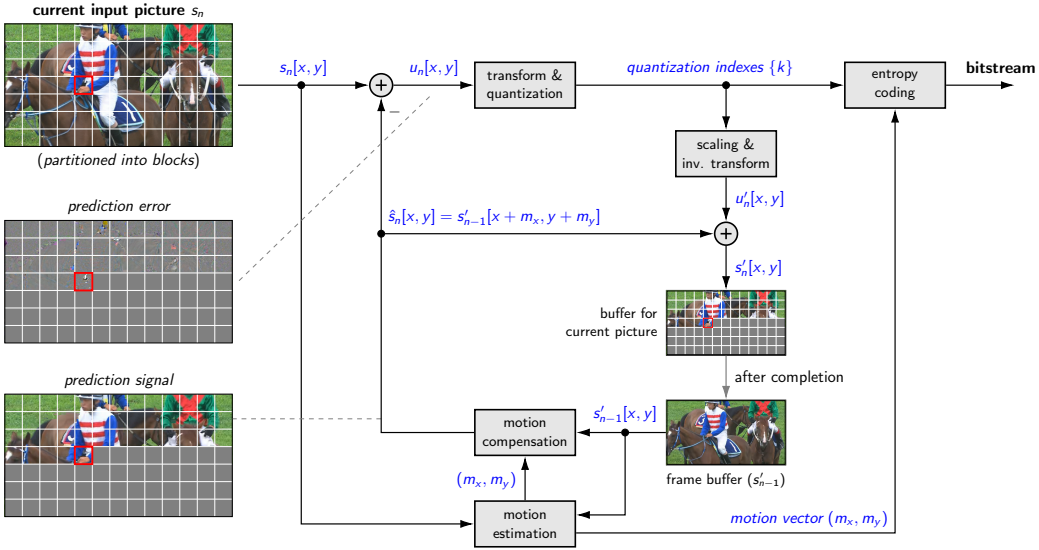
Block-based Transform Coding

1. **Transform**: Separable DCT-II for 8×8 blocks of samples
2. **Quantization**: Uniform reconstruction quantizer
3. **Entropy Coding**: DC Prediction and Run-Level Coding with Huffman tables
   - a. Prediction of quantization index for DC coefficient (mean of block): $\Delta q^n_{(0,0)} = q^n_{(0,0)} - q^{n-1}_{(0,0)}$
   - b. Huffman table for prediction error $\Delta q^n_{(0,0)}$ for DC coefficient
   - c. Run-level coding of remaining quantization indexes (without prediction)

Modern Image Coding: Forward-Adaptive DPCM Structure



Hybrid Video Coding



Oral Exam: 30 Minutes

- Possible dates: February, 16 – March, 20 or March, 31 – April, 10

➡ Write e-mail to heiko.schwarz@hhi.fraunhofer.de with 2–3 suggestions for an exam date

- Exam will take place at Fraunhofer HHI:

Fraunhofer HHI
Video Coding & Analytics
Salzufer 6
5. Etage
10587 Berlin

Eingang:

Otto-Dibelius Strasse