Channel coding

François Horlin

Outline

- Introduction
- Block codes
- Low density parity check codes
- Convolutional codes
- Exercises

References

"Digital Communications: Fundamentals and Applications", B. Sklar

- Introduction
- Block codes
- Low density parity check codes
- Convolutional codes
- Exercises

Outline: Introduction

- Types of error control
- Parity check codes
- Performance/bandwidth/power trade-off
- Channel models

Introduction

Objective: transform data sequences by adding structured redundancy used for detection and correction of errors

Two types of error control:

- Automatic repeat request (ARQ): the receiver detects an error and requests that the transmitter retransmits the data
- Forward error correction (FEC): the receiver detects an error and corrects it directly

ARQ advantages/drawbacks

A reverse channel is necessary to support the dialogue between the transmitter and receiver

Lower computational complexity and less redundancy is required as the error correction is not implemented

Intrinsically adaptive to the channel quality since information is retransmitted only when errors occur

Suffers from excessive retransmissions when channel quality is low

The overall delay for signal detection is increased

FEQ advantages/drawbacks

A one-way link is sufficient

Additional computational complexity and redundancy is required for error correction

Reduced adaptivity to channel quality since redundancy is fixed whatever the number of errors

Does not suffer from excessive retransmissions

The signal detection delay is only due to the receiver implementation

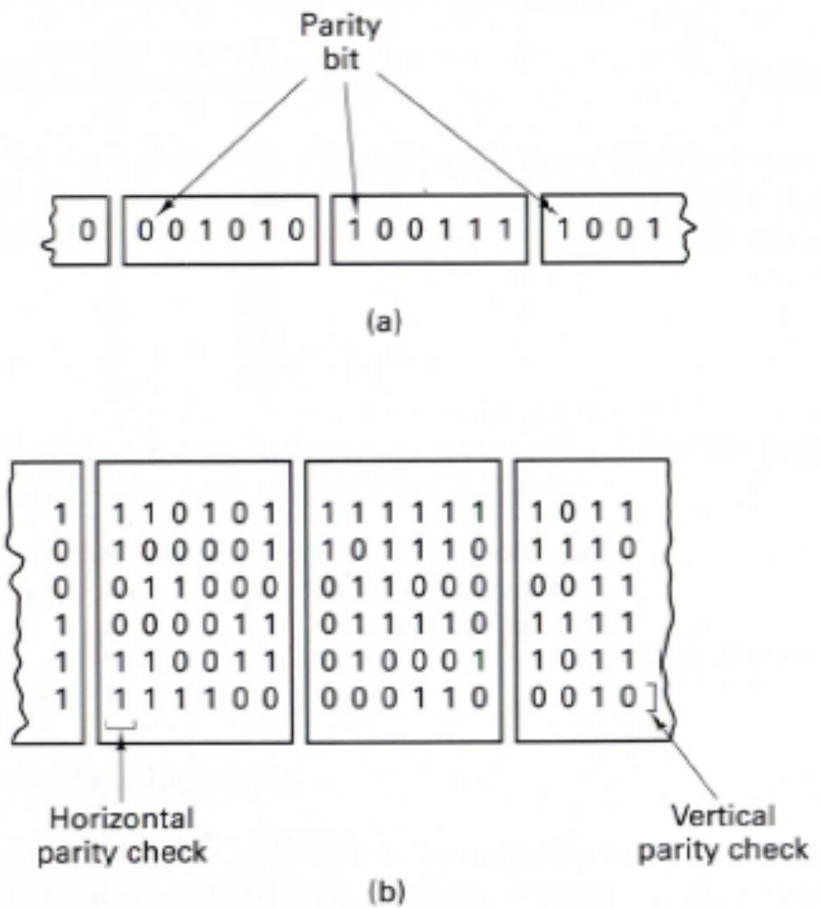
Error correcting code (N,K)

The encoder transforms a block of K bits into a larger block of N bits

Definitions:

- Parity bits: $N - K$ additional bits
- Redundancy: $(N - K)/K$
- Code rate: $K/N \leq 1$

Parity check codes



(a) Single-parity check codes; (b) Rectangular code

Single-parity check code

Constructed by adding a single parity bit to a block of K data bits

The parity bit is chosen such that the (modulo-2) sum of the $K + 1$ bits yields a zero

The decoding procedure consists of checking the sum of the codeword bits

Therefore the code can only detect the presence of an odd number of errors in the block

It has no ability to correct the errors

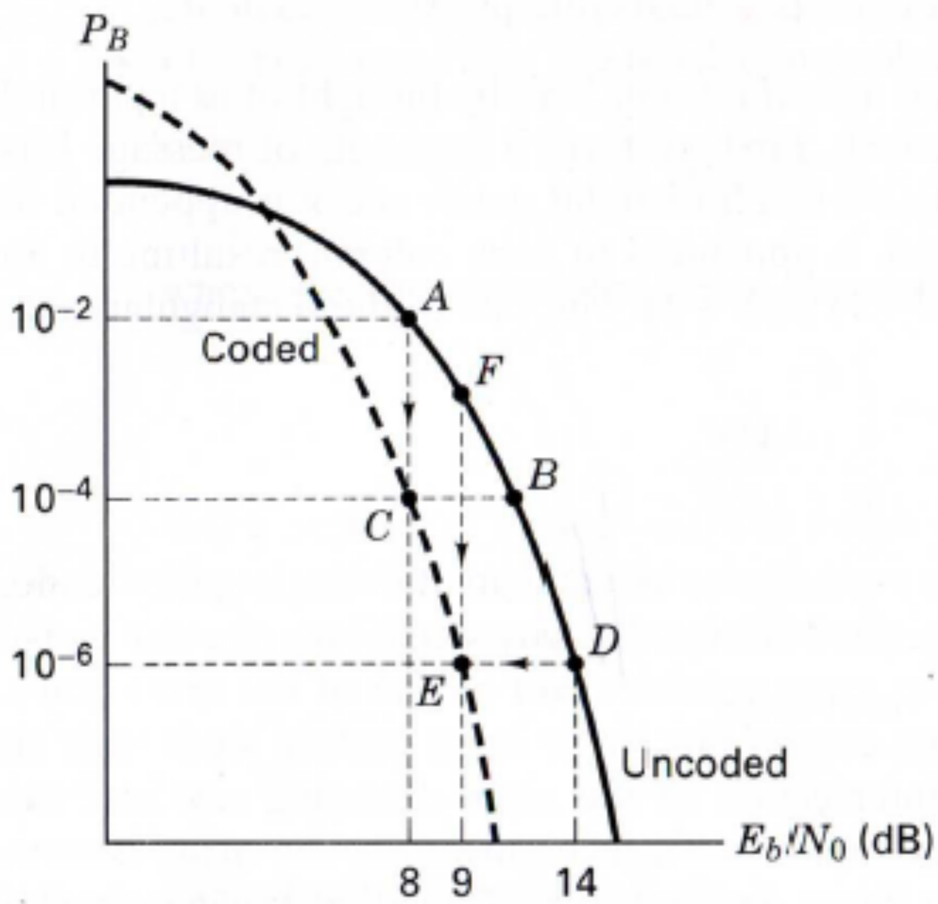
Rectangular code

Constructed by appending a horizontal parity check to each row and a vertical parity check to each column of a message bit rectangle

The code can correct any single error pattern since such an error is uniquely located at the intersection of the error-detecting row and error-detecting column

It has further an improved error detection ability

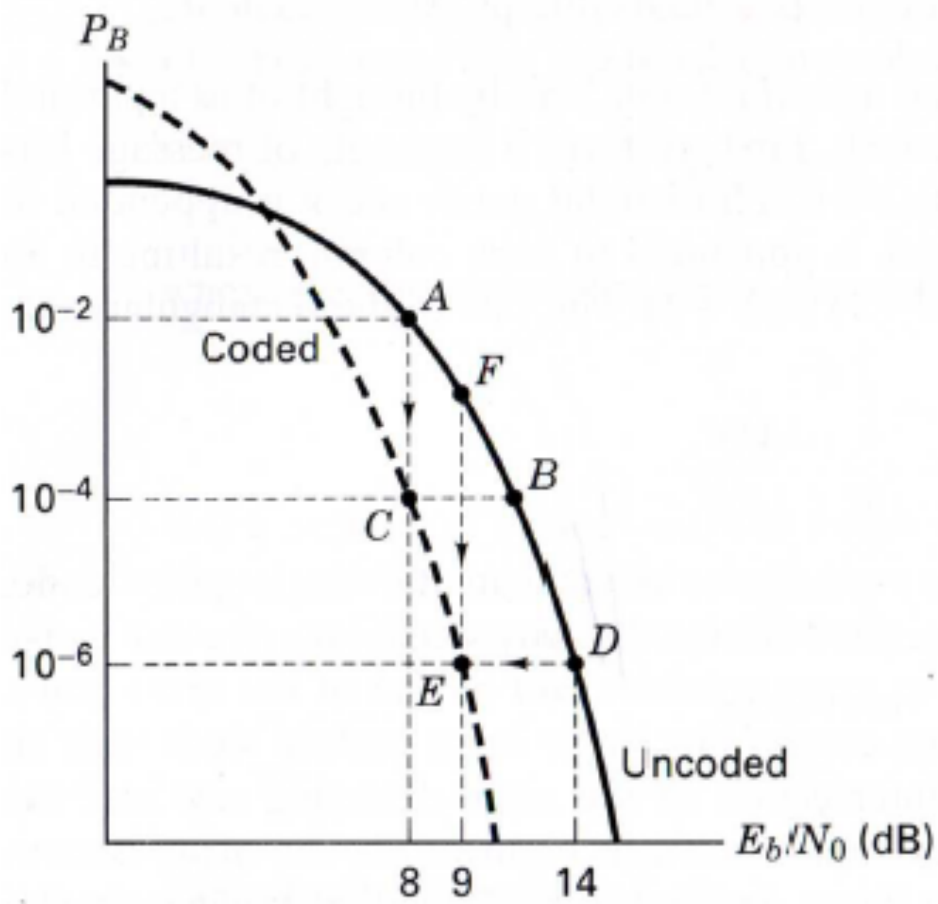
Error performance versus bandwidth



The error performance at a given E_b/N_0 can be improved by error correction coding (F to E)

The addition of redundant bits requires a faster rate of transmission, resulting in a higher physical bandwidth

Power versus bandwidth



The necessary E_b/N_0 to obtain a given error performance can be lowered by error correction coding (D to E)

The addition of redundant bits results again in a higher physical bandwidth

Coding gain:
$\mathrm{SNR}_{uncod} - \mathrm{SNR}_{cod}$ [dB]

Channel models

Interest for two types of channels:

- Binary Symmetric Channel
- Gaussian Channel

Both belong to the class of memoryless channels (each output of the channel depends only on the corresponding input)

Binary Symmetric Channel (BSC)

BSC channel defined from transmitted bits to detected bits (hard decisions)

Channel characterized by the transition probabilities between binary inputs $u[n]$ and outputs $r[n]$:

$$
\begin{aligned}
P(r[n]=1|u[n]=0) &= P(r[n]=0|u[n]=1) = p \\
P(r[n]=1|u[n]=1) &= P(r[n]=0|u[n]=0) = 1-p
\end{aligned}
$$

Gaussian Channel

Gaussian channel defined from transmitted symbols to demodulator outputs (soft decisions)

Assuming BPSK symbols $u[n] = \pm 1$ corrupted by additive white Gaussian noise $w[n]$ of variance σ_w^2, the channel is expressed as:

$$r[n] = u[n] + w[n]$$

It is characterized by a Gaussian distribution:

$$P(r[n]|u[n]) = \frac{1}{\sqrt{2\pi}\sigma_w} \exp\left(-\frac{1}{2\sigma_w^2}(r[n] - u[n])^2\right)$$

Outline

- Introduction
- **Block codes**
- Low density parity check codes
- Convolutional codes
- Exercises

Outline: Block codes

- Vector spaces and subspaces
- Generator matrix
- Systematic linear block codes
- Parity check matrix and syndrome testing
- Standard array and error correction
- Hamming weight and distance
- Optimal decoder strategy
- Error detection and correction capability

Introduction to block codes

The bit stream is divided into blocks of K bits (message vector)

A (N, K) linear block code transforms each message vector into a longer block of N bits (code vector)

The inputs of the decoder are often the detected bits to limit the decoder complexity

Therefore hard decoding, working on the BSC channel, is assumed

Vector space and subspace

Vector space $\mathcal{V}_N$: set of all binary N-tuples

Vector subspace $\mathcal{S}$: subset of the vector space $\mathcal{V}_N$ such that:

- The all-zeros vector is in $\mathcal{S}$
- The (modulo-2) sum of any two vectors in $\mathcal{S}$ is also in $\mathcal{S}$

Example $(N = 6)$

Vector space:

$$\mathcal{V}_6 = \begin{array}{cccc} 000000 & 000001 & 000010 & 000011 \\ 000100 & 000101 & 000110 & 000111 \\ \vdots & \vdots & \vdots & \vdots \\ 111100 & 111101 & 111110 & 111111 \end{array}$$

Vector subspace example:

$$\mathcal{S} = \begin{array}{cccc} 000000 & 110100 & 011010 & 101110 \\ 101001 & 011101 & 110011 & 000111 \end{array}$$

Linear block codes

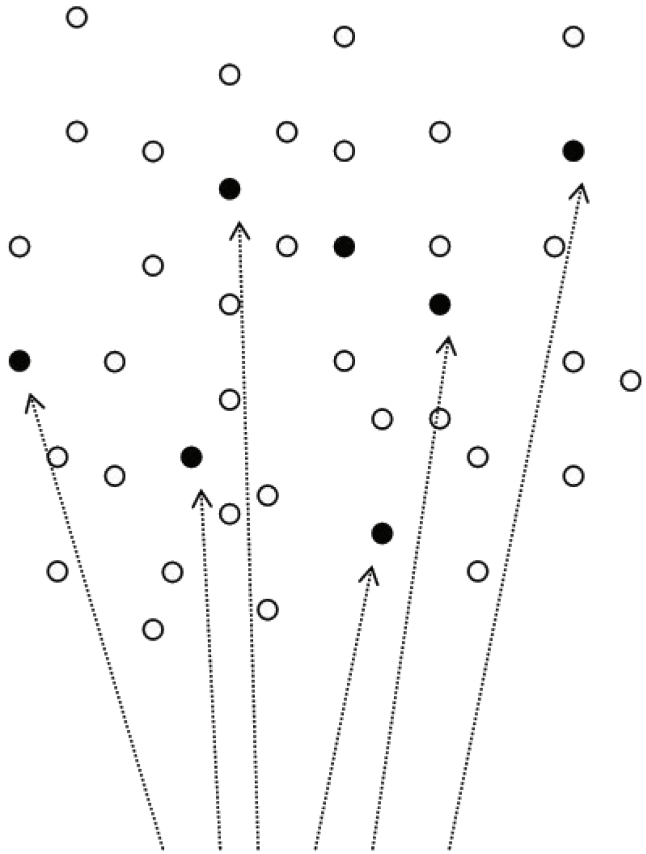
The code is defined as a subspace of 2^K N-tuples of the space $\mathcal{V}_N$

A message vector of K bits is replaced by one of the 2^K code vectors in the subspace

The encoder can be implemented with a lookup table but the complexity becomes prohibitive as K increases

Rather look for a mean to compute the code vector based on the message (see generator matrix)

Vector space and subspace



2^K N-tuples constitute the subspace of codewords
in the entire space of 2^N N-tuples

Example ($K = 3,\ N = 6$)

Message vector	Codeword
0 0 0	0 0 0 0 0 0
1 0 0	1 1 0 1 0 0
0 1 0	0 1 1 0 1 0
1 1 0	1 0 1 1 1 0
0 0 1	1 0 1 0 0 1
1 0 1	0 1 1 1 0 1
0 1 1	1 1 0 0 1 1
1 1 1	0 0 0 1 1 1

Detection

Because of the noise in the channel, a perturbed version of the codeword may be received (one of the other 2^N vectors in $\mathcal{V}_N$)

If the perturbed version of the codeword is not too distant from the valid codeword, the decoder can decode the message correctly

Therefore, a code is optimized such that:

- As many codewords as possible are selected in $\mathcal{V}_N$ (coding efficiency)
- The selected codewords are as apart from one another as possible (error performance)

Generator matrix

A basis of a subspace is formed by the smallest linearly independent set of N-tuples that spans completely the subspace

If $\{\underline{v}_1, \cdots, \underline{v}_K\}$ is a basis of the subspace, any vector $\underline{u}$ of the subspace can be written as:

$$\underline{u} = \sum_{k=1}^{K} d_k \underline{v}_k; \quad d_k = \{0, 1\}$$

Generator matrix

Equivalently:

$$\begin{aligned} \underline{u} &= \begin{bmatrix} d_1 & \cdots & d_K \end{bmatrix} \cdot \begin{bmatrix} \underline{v}_1 \\ \vdots \\ \underline{v}_K \end{bmatrix} \\ &= \underline{d} \cdot \underline{\underline{G}} \end{aligned}$$

where $\underline{d}$ is the message and $\underline{\underline{G}}$ is the generator matrix

Since the code is totally defined by $\underline{\underline{G}}$, the encoder needs only to store the K rows of $\underline{\underline{G}}$ instead of the total set of 2^K code vectors

Example ($K = 3,\ N = 6$)

Generator matrix:

$$\underline{\underline{G}} = \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \underline{v}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Code vector of the message $1\,1\,0$:

$$\underline{u} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \underline{v}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Systematic linear block codes

Definition: the mapping is such that part of the code vector coincides with the message vector

The generator matrix has the form:

$$\underline{\underline{G}} = \left[\ \underline{\underline{P}}\ \middle|\ \underline{\underline{I}}_K\ \right]$$

where

- $\underline{\underline{P}}$ is the parity array portion (size $K \times N - K$)
- $\underline{\underline{I}}_K$ is the identity matrix (size K)

The code vector is composed of the parity bits $\underline{d} \cdot \underline{\underline{P}}$ and of the message vector $\underline{d}$

Parity-check matrix

Definition: matrix $\underline{\underline{H}}$ of size $N-K \times N$ such that the rows are orthogonal to the rows of the generator matrix ($\underline{\underline{G}} \cdot \underline{\underline{H}}^T = \underline{0}$)

In other words, the rows of $\underline{\underline{H}}$ form a basis of the subspace complementary to the one of the code

In case of a systematic code (modulo-2 addition or substraction are equivalent):

$$\underline{\underline{H}} = \left[\begin{array}{c|c} \underline{\underline{I}}_{N-K} & \underline{\underline{P}}^T \end{array} \right]$$

Syndrome testing

The received vector $\underline{r}$ is the transmitted vector $\underline{u}$ plus an error vector $\underline{e}$ caused by the channel:

$$\underline{r} = \underline{u} + \underline{e}$$

The syndrome of $\underline{r}$ is defined as:

$$\begin{aligned} \underline{s} &:= \underline{r} \cdot \underline{\underline{H}}^T \\ &= \underline{u} \cdot \underline{\underline{H}}^T + \underline{e} \cdot \underline{\underline{H}}^T \\ &= \underline{e} \cdot \underline{\underline{H}}^T \end{aligned}$$

The syndrome is equal for the corrupted received vector or for the corresponding error vector

An error is detected when the syndrome is different from $\underline{0}$

Example $(K = 3,\ N = 6)$

Assume the following transmit and received vectors:

$$\underline{u} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}; \quad \underline{r} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

The syndrome is equal to:

$$\underline{s} = \underline{r} \cdot \underline{\underline{H}}^T = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Standard array

Arrange all N-tuples in a standard array:

$$\begin{bmatrix} \underline{u}_1 & \underline{u}_2 & \cdots & \underline{u}_i & \cdots & \underline{u}_{2^K} \\ \underline{e}_2 & \underline{u}_2 + \underline{e}_2 & \cdots & \underline{u}_i + \underline{e}_2 & \cdots & \underline{u}_{2^K} + \underline{e}_2 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \underline{e}_j & \underline{u}_2 + \underline{e}_j & \cdots & \underline{u}_i + \underline{e}_j & \cdots & \underline{u}_{2^K} + \underline{e}_j \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \underline{e}_{2^{N-K}} & \underline{u}_2 + \underline{e}_{2^{N-K}} & \cdots & \underline{u}_j + \underline{e}_{2^{N-K}} & \cdots & \underline{u}_{2^K} + \underline{e}_{2^{N-K}} \end{bmatrix}$$

The first row contains all codewords. It starts with the all-zeros codeword.

The first column contains all correctable error patterns. They are chosen by the code designer.

Coset: one row corresponding to one correctable error pattern (coset leader) or equivalently to a common syndrome

Decoding algorithm: replace a corrupted vector with a valid codeword from the top of the column

Example ($K = 3,\ N = 6$)

000000	110100	011010	101110	101001	011101	110011	000111
000001	110101	011011	101111	101000	011100	110010	000110
000010	110110	011000	101100	101011	011111	110001	000101
000100	110000	011110	101010	101101	011001	110111	000011
001000	111100	010010	100110	100001	010101	111011	001111
010000	100100	010010	111110	111001	001101	100011	010111
100000	010100	111010	001110	001001	111101	010011	100111
010001	100101	010011	111111	111000	001100	100010	010110

Example ($K = 3,\ N = 6$)

Error pattern	Syndrome
000000	000
000001	101
000010	011
000100	110
001000	001
010000	010
100000	100
010001	111

Error correction

Steps:

- Calculate the syndrome $\underline{s} = \underline{r} \cdot \underline{\underline{H}}^T$
- Determine the error pattern $\underline{e}_j$ corresponding to the syndrome $\underline{s}$
- Estimate the transmitted code vector by correcting the received vector $\underline{u} = \underline{r} + \underline{e}_j$ (modulo 2 subtraction or addition are equivalent!)

If the error caused by the channel is not a coset leader, then an erroneous decoding will result

Example ($K = 3,\ N = 6$)

Assume the following transmit and received vectors:

$$\underline{u} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}; \quad \underline{r} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

The syndrome and the corresponding error pattern are equal to:

$$\begin{aligned} \underline{s} &= \underline{r} \cdot \underline{\underline{H}}^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \\ \underline{\hat{e}} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

Therefore the corrected vector is:

$$\underline{\hat{u}} = \underline{r} + \underline{\hat{e}} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Hamming weight and distance

Definitions:

- $W(\underline{u})$ is the number of non-zero elements in $\underline{u}$
- $D(\underline{u}, \underline{v})$ is the number of elements in which $\underline{u}$ and $\underline{v}$ differ

The distance between two codewords is the weight of their sum:

$$D(\underline{u}, \underline{v}) = W(\underline{u} + \underline{v})$$

Minimum distance of a block code

Definition: D_{min} is the smallest distance between all pairs of codewords

Computation:

- Remember that the sum of any two codewords yields another codeword member of the code subspace
- Therefore D_{min} is easily obtained by taking the smallest codeword weight (excluding the all-zeros codeword)

The minimum distance is a measure of the error detection and correction capability of the code ("weakest link of the chain")

Optimal decoder strategy

Maximum likelihood (ML) criterion:

$$\underline{\hat{u}} = \max_{u} P(\underline{r}|\underline{u})$$

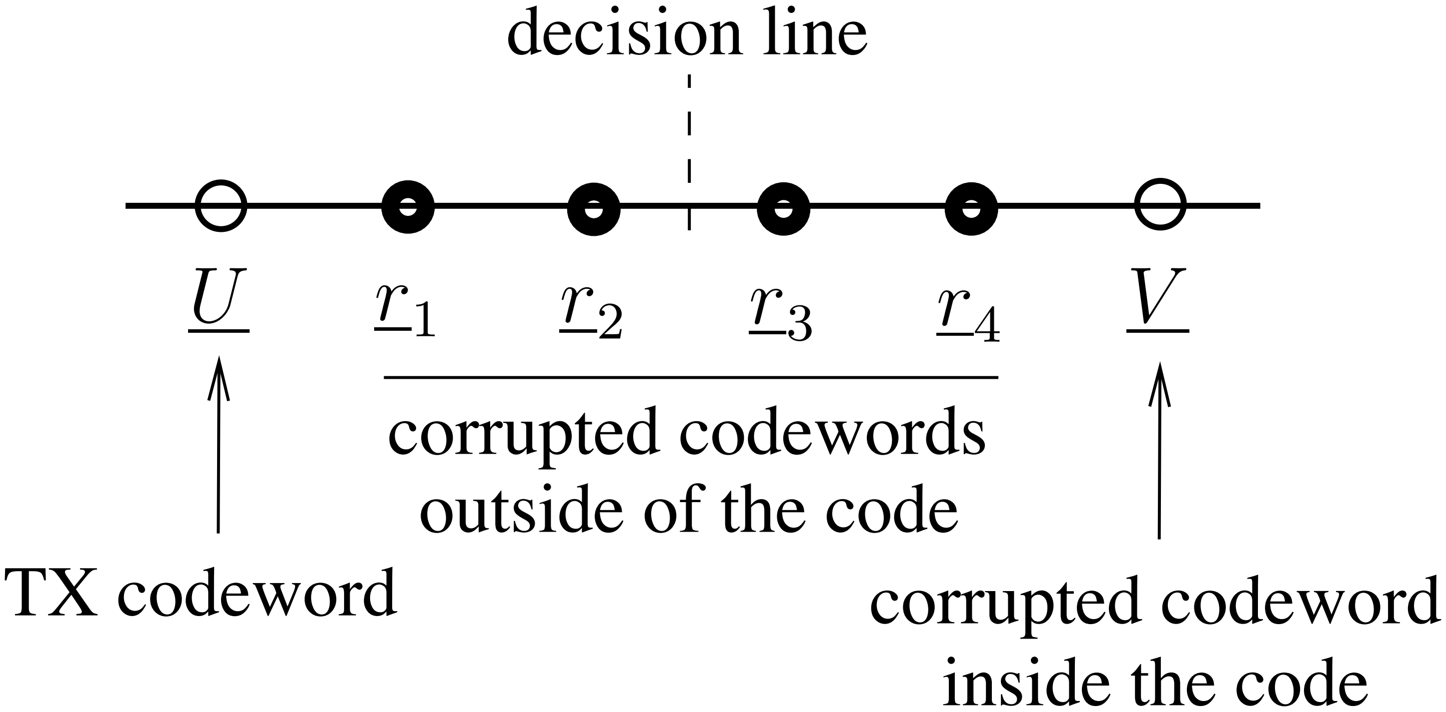
Equivalently:

$$\underline{\hat{u}} = \min_{u} D(\underline{r}, \underline{u})$$

The decoder determines the distance between $\underline{r}$ and all possible transmitted codewords $\underline{u}$ and selects the nearest codeword

The error correction algorithm implemented based on the standard array satisfies the ML criterion if the coset leaders are chosen of minimum weight

Error detection and correction



Error correction capability:

- $\underline{u}$ is correctly selected if $\underline{r}_1$ or $\underline{r}_2$ is received
- $\underline{v}$ is erroneously selected if $\underline{r}_3$, $\underline{r}_4$ or $\underline{v}$ is received

Error detection capability:

- An error is correctly detected when $\underline{r}_1$, $\underline{r}_2$, $\underline{r}_3$ or $\underline{r}_4$ is received
- No error is erroneously estimated when $\underline{v}$ is received

The code has a 2 bit error correction capability and a 4 bit error detection capability for $D_{min} = 5$

Link with the minimum distance

Error correction capability: maximum number of guaranteed correctable errors per codeword

$$ECC = \lfloor \frac{D_{min} - 1}{2} \rfloor$$

Error detection capability: maximum number of guaranteed detectable errors per codeword

$$EDC = D_{min} - 1$$

Outline

- Introduction
- Block codes
- **Low density parity check codes**
- Convolutional codes
- Exercises

Outline: Low density parity check codes

- Tanner graph
- Hard decoding
- Soft decoding in the probability domain
- Soft decoding in the log domain
- Code design criterion

Introduction to LDPC codes

Low density parity check (LDPC) codes are block codes of sparse parity check matrix $\underline{\underline{H}}$:

- As the number of non-zero elements in matrix $\underline{\underline{H}}$ is small, the decoder complexity can be kept low
- Performance of hard decoding is generally poor, but soft decoding can easily be implemented improving significantly performance

Matrix $\underline{\underline{H}}$ can be regular (all row sums are equal, all column sums are equal) or irregular

Tanner graph

Bipartite graph: the nodes are separated into two classes, the edges are undirected and only connect two nodes of different class

Tanner graph: bipartite graph used to represent the low-density parity-check matrix $\underline{\underline{H}}$, as follows:

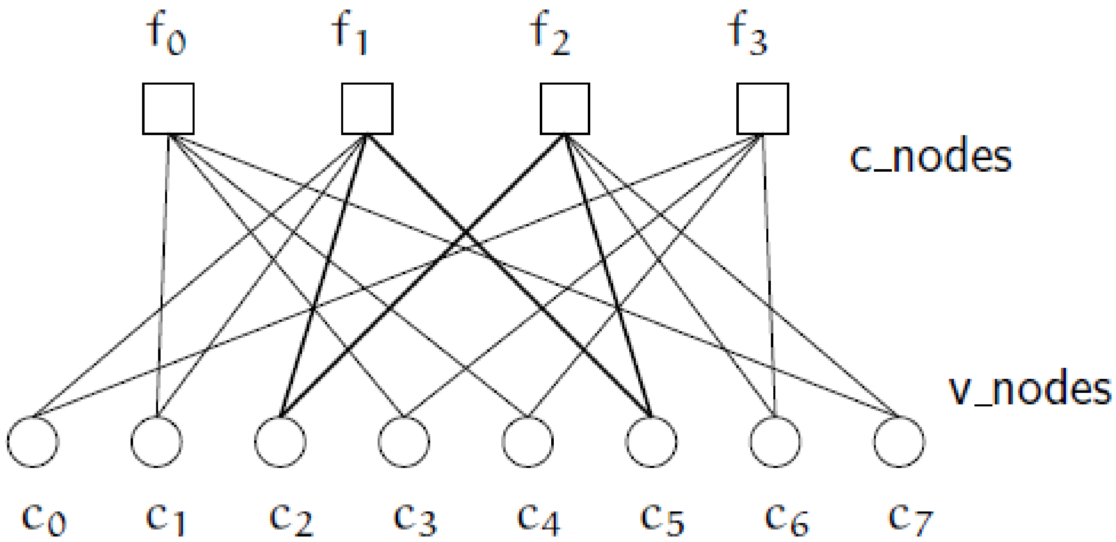
- Variable nodes (v-node) c_i correspond to the noisy codeword
- Check nodes (c-node) f_j correspond to the syndrome

Check node f_j connected to the variable node c_i if element $\underline{\underline{H}}_{ji} = 1$

Small size example

$c_0 \oplus c_1 \oplus c_2 \oplus c_5 = 0$

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ \boxed{1 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0} \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$



Hard decoding

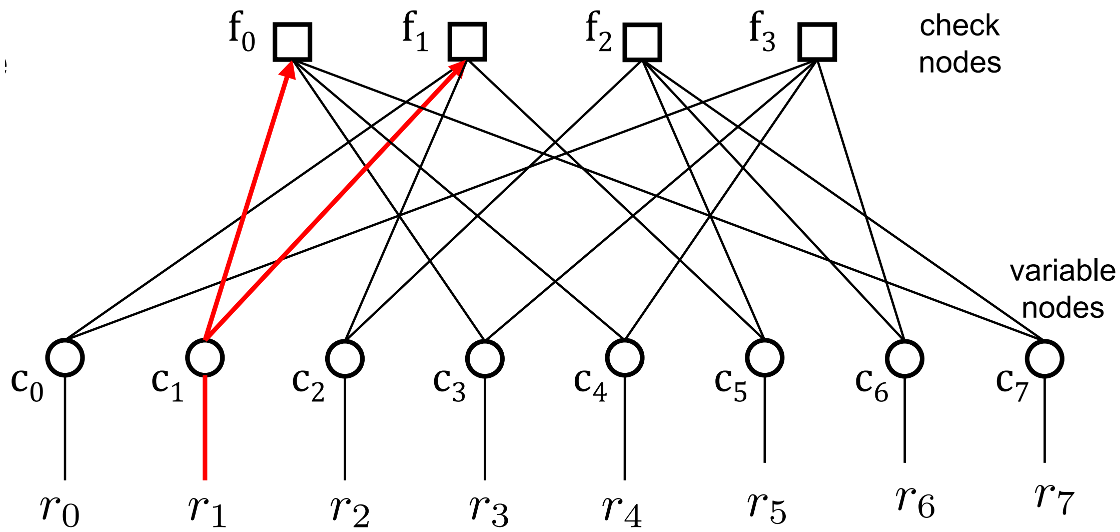
Exchanges the most probable bit between v-nodes and c-nodes (binary messages)

Intrinsically satisfies the maximum a-posteri (MAP) criterion:

$$\underline{\hat{u}} = \max_{u} P(\underline{u}|\underline{r})$$

Hard decoding works on the BSC channel

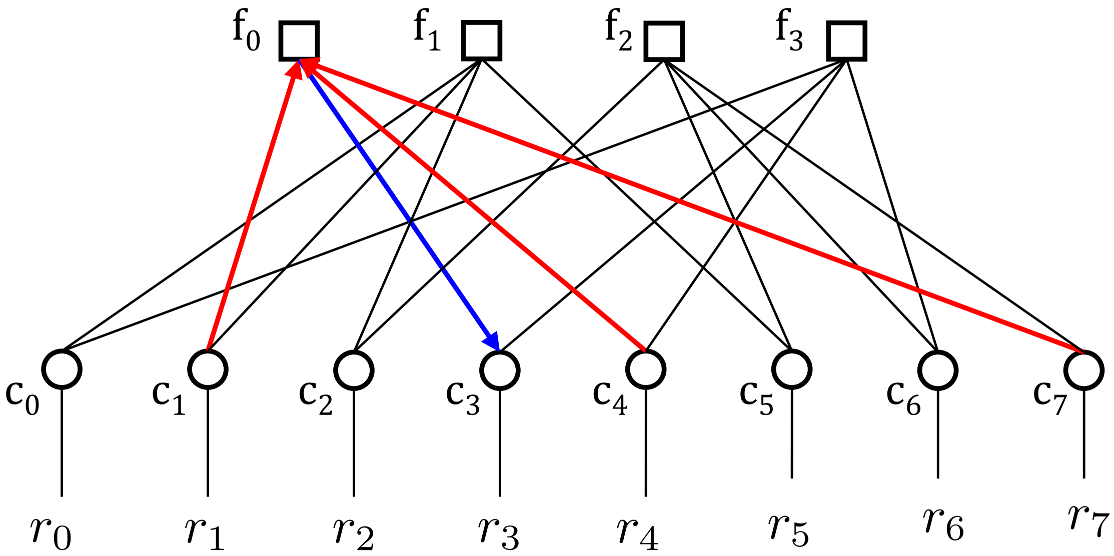
Hard decoding: step 0



All v-nodes c_i send a message to their c-nodes f_j containing the bit they believe to be the correct one for them

At this stage the only information a v-node has is the corresponding received bit r_i

Hard decoding: step 1

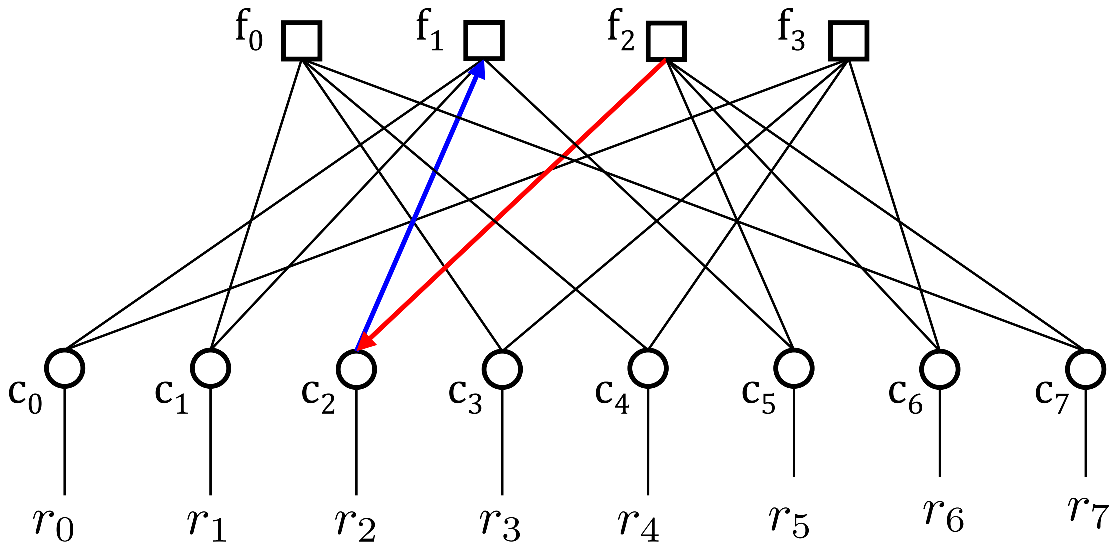


Every c-node f_j calculates a response to its connected variable nodes

The response contains the bit that f_j believes to be the correct one for the v-node c_i assuming that the other v-nodes connected are correct

To calculate a new message for a v-node, the previous message from that node is NOT taken into account !

Hard decoding: step 2



Variable nodes use the messages from the c-nodes AND the received bit r_i to make a decision (majority voting)

To calculate a new message for a c-node, the previous message from that node is NOT taken into account !

Hard decoding: next steps

Estimated codeword is found in the v-nodes:

$$\hat{u}_i = c_i$$

Iterate until all check equations are satisfied, so that $\underline{\hat{u}} \cdot \underline{\underline{H}}^T = 0$

Example

c-node	received/sent				
f_0	received:	$c_1 \rightarrow 1$	$c_3 \rightarrow 1$	$c_4 \rightarrow 0$	$c_7 \rightarrow 1$
	sent:	$0 \rightarrow c_1$	$0 \rightarrow c_3$	$1 \rightarrow c_4$	$0 \rightarrow c_7$
f_1	received:	$c_0 \rightarrow 1$	$c_1 \rightarrow 1$	$c_2 \rightarrow 0$	$c_5 \rightarrow 1$
	sent:	$0 \rightarrow c_0$	$0 \rightarrow c_1$	$1 \rightarrow c_2$	$0 \rightarrow c_5$
f_2	received:	$c_2 \rightarrow 0$	$c_5 \rightarrow 1$	$c_6 \rightarrow 0$	$c_7 \rightarrow 1$
	sent:	$0 \rightarrow c_2$	$1 \rightarrow c_5$	$0 \rightarrow c_6$	$1 \rightarrow c_7$
f_3	received:	$c_0 \rightarrow 1$	$c_3 \rightarrow 1$	$c_4 \rightarrow 0$	$c_6 \rightarrow 0$
	sent:	$1 \rightarrow c_0$	$1 \rightarrow c_3$	$0 \rightarrow c_4$	$0 \rightarrow c_6$

Transmitted codeword: $[1\,0\,0\,1\,0\,1\,0\,1]$

Received codeword: $[1\,1\,0\,1\,0\,1\,0\,1]$

Example

v-node	y_i received	messages from check nodes		decision
c_0	1	$f_1 \rightarrow 0$	$f_3 \rightarrow 1$	1
c_1	1	$f_0 \rightarrow 0$	$f_1 \rightarrow 0$	0
c_2	0	$f_1 \rightarrow 1$	$f_2 \rightarrow 0$	0
c_3	1	$f_0 \rightarrow 0$	$f_3 \rightarrow 1$	1
c_4	0	$f_0 \rightarrow 1$	$f_3 \rightarrow 0$	0
c_5	1	$f_1 \rightarrow 0$	$f_2 \rightarrow 1$	1
c_6	0	$f_2 \rightarrow 0$	$f_3 \rightarrow 0$	0
c_7	1	$f_0 \rightarrow 1$	$f_2 \rightarrow 1$	1

Decision at v-nodes based on majority voting

Soft decoding in the probability domain

Soft decoding works like hard decoding, but exchanges real values (bit probabilities) instead of binary values between v-nodes and c-nodes

Intrinsically satisfies the MAP criterion:

$$\underline{\hat{u}} = \max_{u} P(\underline{u}|\underline{r})$$

Works on the Gaussian channel

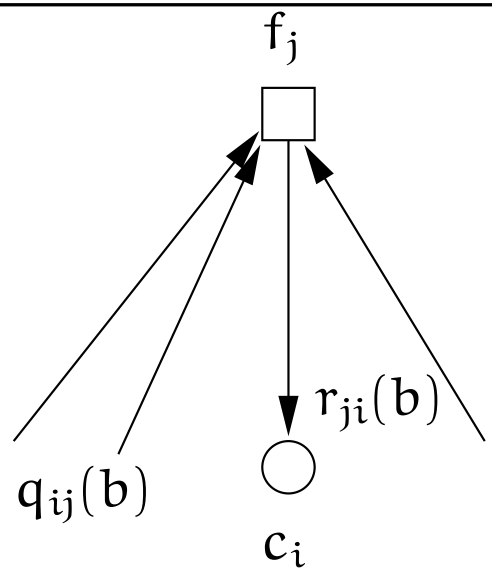
Significantly outperforms hard decoding

Soft decoding: step 0

Initial message $q_{ij}(0)$ of c_i to f_j is the probability that c_i is a 0, given observation r_i:

$$\begin{aligned} q_{ij}(0) &= P(c_i = 0|r_i) = \frac{1}{1+e^{2r_i/\sigma_w^2}} \\ q_{ij}(1) &= P(c_i = 1|r_i) = 1 - q_{ij}(0) \end{aligned}$$

Soft decoding: step 1



Response $r_{ji}(0)$ of f_j to c_i is the probability that c_i is a 0, equal to the probability that the number of 1's among the connected variable nodes except c_i is even (Galager's formula):

$$
\begin{aligned}
r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in C_{j \setminus i}} (1 - 2q_{i'j}(1)) \\
r_{ji}(1) &= 1 - r_{ji}(0)
\end{aligned}
$$

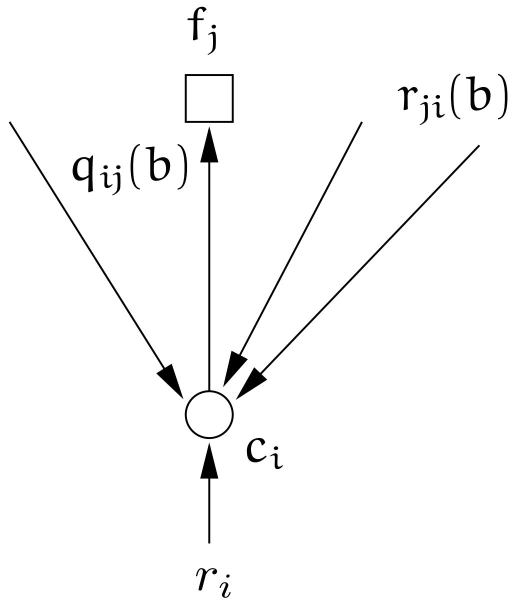
Galager's formula

For a sequence of M independent binary digits a_i with a probability p_i for $a_i = 1$, the probability that the whole sequence contains an even number of 1's is:

$$\frac{1}{2} + \frac{1}{2}\prod_{i=1}^{M}(1 - 2p_i)$$

Note that the p_i do not need to be identical for all digits

Soft decoding: step 2



Response $q_{ij}(0)$ of c_i to f_j is the probability that c_i is a 0, given the observation r_i and the messages communicated by all check nodes except f_j

Soft decoding: step 2

$$
\begin{aligned}
q_{ij}(0) &= K_{ij}\, P(c_i = 0 | r_i) \prod_{j' \in F_{i \setminus j}} r_{j'i}(0) \\
q_{ij}(1) &= K_{ij}\, P(c_i = 1 | r_i) \prod_{j' \in F_{i \setminus j}} r_{j'i}(1)
\end{aligned}
$$

where K_{ij} is a constant chosen such that $q_{ij}(0) + q_{ij}(1) = 1$

Soft decoding: next steps

Soft decision:

$$\begin{aligned} Q_i(0) &= K_i\, P(c_i = 0|r_i) \prod_{j \in F_i} r_{ji}(0) \\ Q_i(1) &= K_i\, P(c_i = 1|r_i) \prod_{j \in F_i} r_{ji}(1) \end{aligned}$$

where K_i is a constant chosen such that $Q_i(0) + Q_i(1) = 1$

Hard decision:

$$\hat{u}_i = \begin{cases} 1; & Q_i(1) > 0.5 \\ 0; & else \end{cases}$$

Iterate until $\underline{\hat{u}} \cdot \underline{\underline{H}}^T = 0$ or number of iterations exceeds limit

Soft decoding in the log-domain

Soft decoding in the probability domain comes with computation stability problems:

- Many multiplications of probabilities
- Some results close to zero for large block lengths

Rather work in the log-domain by defining log-likelihood ratios (LLR):

$$\begin{aligned} L(q_{ij}) &:= \log \frac{q_{ij}(0)}{q_{ij}(1)} \\ L(r_{ij}) &:= \log \frac{r_{ij}(0)}{r_{ij}(1)} \end{aligned}$$

LLR sign indicates decision bit; amplitude indicates decision reliability

Soft decoding (log domain): step 0

Initialize:

$$L(q_{ij}) = L(c_i) := \log \frac{P(c_i = 0|r_i)}{P(c_i = 1|r_i)} = -\frac{2r_i}{\sigma_w^2}$$

Soft decoding (log domain): step 1

Update variable nodes with sum-product formula (proof in [Barry 2001]):

$$L(r_{ji}) = \left(\prod_{i' \in C_{j \setminus i}} \chi_{i'j} \right) \cdot \Phi \left(\sum_{i' \in C_{j \setminus i}} \Phi\left(\alpha_{i'j}\right) \right)$$

where:

$$\begin{aligned} \chi_{ij} &:= \operatorname{sign}(L(q_{ij})) \\ \alpha_{ij} &:= \operatorname{abs}(L(q_{ij})) \end{aligned}$$

and:

$$\Phi(x) := -\log\tanh\left(\frac{x}{2}\right) = \log\frac{e^x+1}{e^x-1} = \Phi^{-1}(x)\,;\, x > 0$$

Soft decoding (log domain): step 2

Update check nodes:

$$L(q_{ij}) = L(c_i) + \sum_{j' \in F_{i \setminus j}} L(r_{j'i})$$

ULB

Soft decoding (log domain): next steps

Soft decision:

$$L(Q_i) = L(c_i) + \sum_{j \in F_i} L(r_{ji})$$

Hard decision:

$$\hat{u}_i = \begin{cases} 1; & L(Q_i) < 0 \\ 0; & else \end{cases}$$

Iterate until $\underline{\hat{u}} \cdot \underline{\underline{H}}^T = 0$ or number of iterations exceeds limit

Complexity reduction

Step 1 can be simplified based on the approximation:

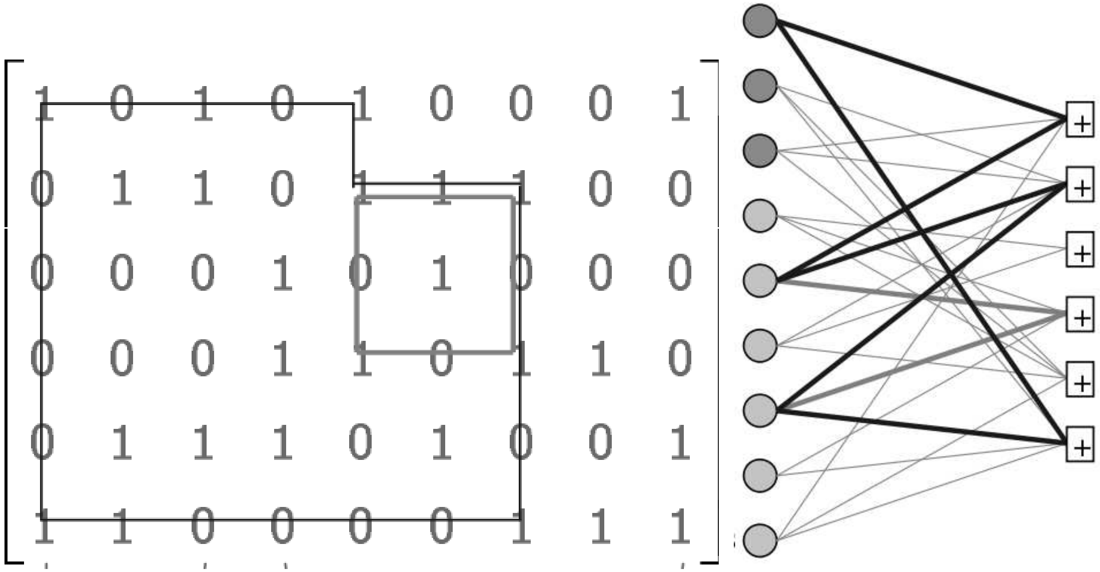
$$\Phi\left(\sum_{i' \in C_{j \setminus i}} \Phi\left(\alpha_{i'j}\right)\right) \approx \Phi\left(\Phi\left(\min_{i' \in C_{j \setminus i}} \alpha_{i'j}\right)\right) = \min_{i' \in C_{j \setminus i}} \alpha_{i'j}$$

It intuitively means that the reliability of the global decision is fixed by the less reliable bit

Code design criterion

Iterative decoding relies on exchange of independent messages between v-nodes and c-nodes

The diameter of the loops in the Tanner graph should be as large as possible to make sure messages are sufficiently independent



- Introduction
- Block codes
- Low density parity check codes
- **Convolutional codes**
- Exercises

Outline: Convolutional codes

- Connection representation
- State diagram
- Trellis diagram
- Viterbi decoding
- Distance properties

Introduction to convolutional codes

Continuous mode instead of blocks

The encoder is implemented with a shift register (constraint length K: number of stages)

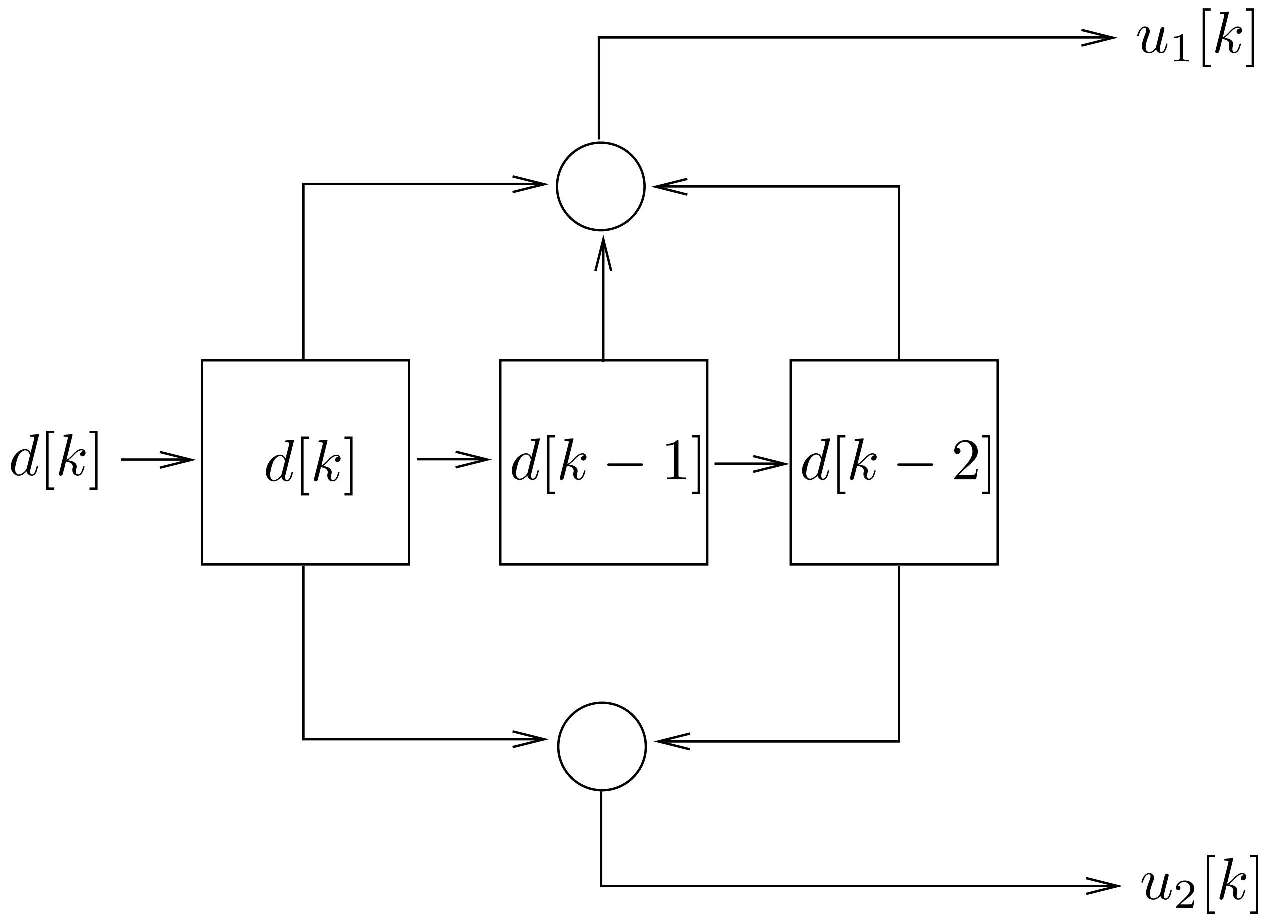
At each unit of time:

- One bit is shifted into the first stage of the register
- All previous bits in the register are shifted one stage to the right
- The outputs of multiple adders connected to the different stages are sequentially sampled and transmitted

The code is defined by the connections between adders and stages

Example: rate $1/2$, $K = 3$

Connection representation



Polynomial representation

One polynomial for each modulo-2 adder

The coefficient of the term n in the polynomial is either 1 or 0, depending on whether a connection exists or does not exist between the stage n of the shift register and the modulo-2 adder

Example:

$$
\begin{aligned}
g_1(X) &= 1 + X + X^2 \\
g_2(X) &= 1 + X^2
\end{aligned}
$$

State diagram

The convolutional encoder can be seen as a finite-state machine

The knowledge of the state together with knowledge of the input is sufficient to determine the output

State diagram:

- The states represent the possible contents of the $K-1$ rightmost stages of the register
- The paths represent the state transitions resulting from each input bit (solid line: input bit 0; dashed line: input bit 1)
- One output word is associated to each path

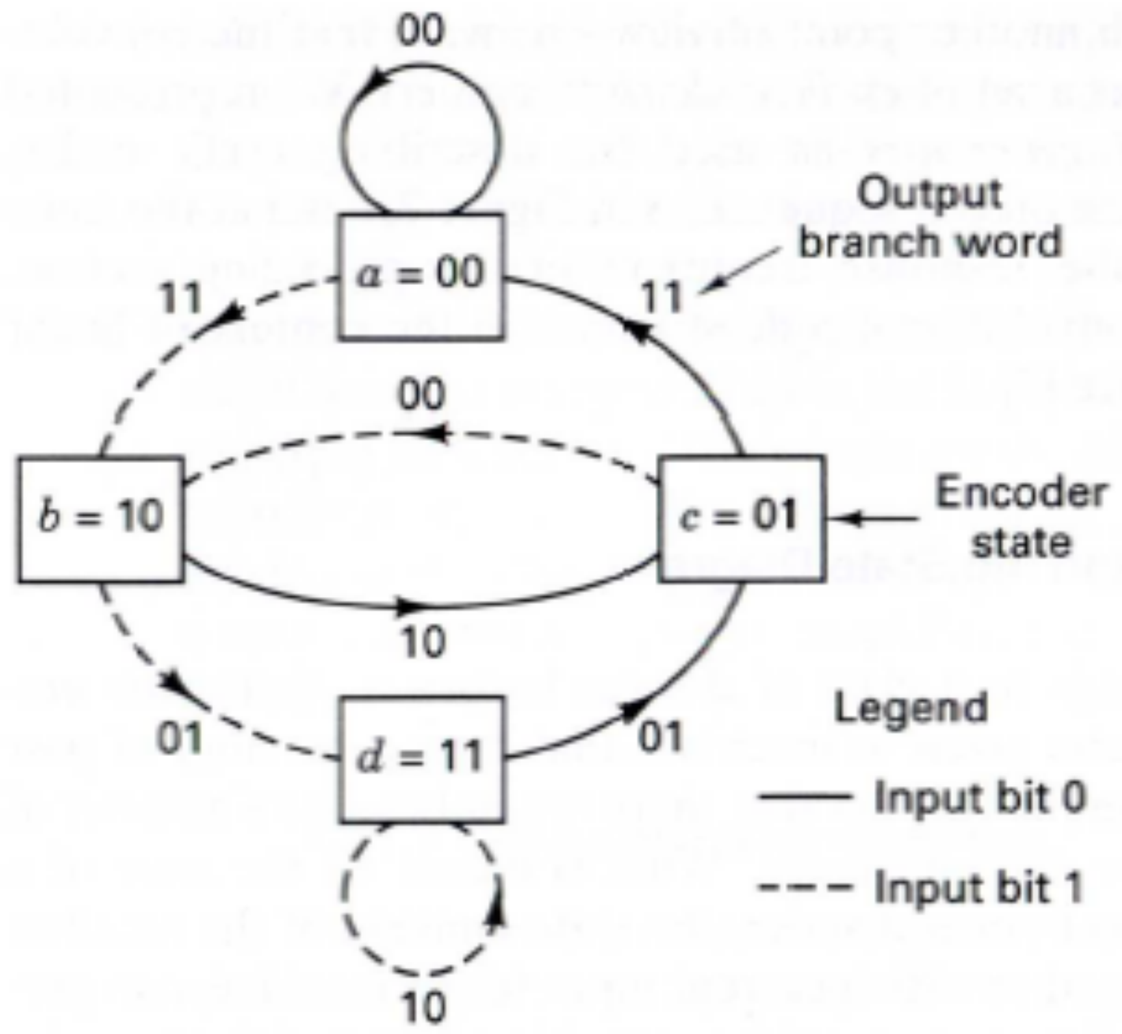
ULB

Example: message 11011

Input i	Register	State i	State $i+1$	Output i
–	000	00	00	–
1	100	00	10	11
1	110	10	11	01
0	011	11	01	01
1	101	01	10	00
1	110	10	11	01
0	011	11	01	01
0	001	01	00	11

Note: start and terminate in state 00 (additional 0 bits)

State diagram

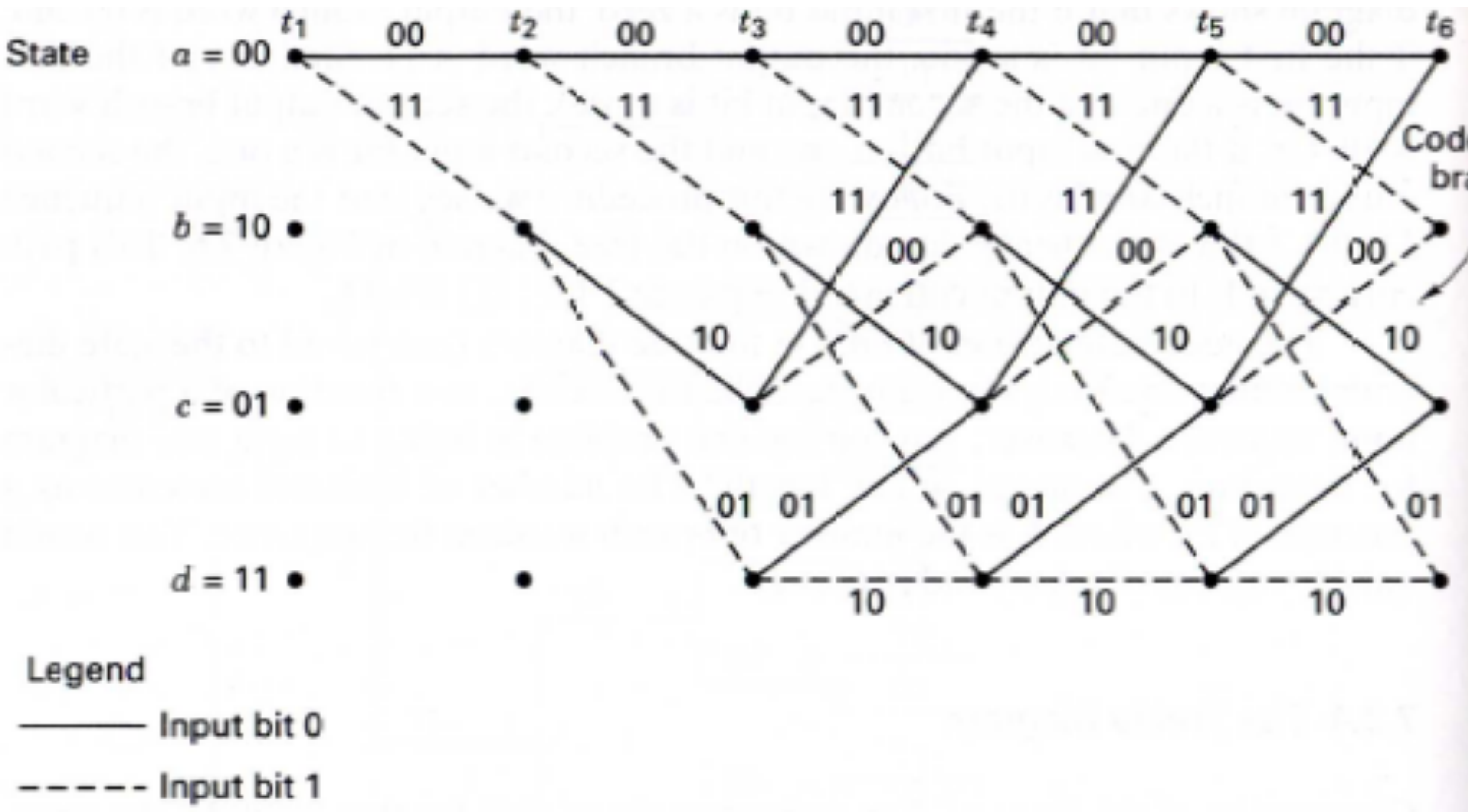


The state diagram characterizes completely the encoder but cannot easily be used for tracking the transitions as a function of time

Trellis diagram adds the dimension of time

Same convention as with the state diagram (state nodes, solid/dashed lines for bits 0 and 1)

Trellis diagram



Optimal decoder strategy

Maximum likelihood (ML) criterion:

$$\begin{aligned}\underline{\hat{u}} &= \arg\max_{u} P(\underline{r}|\underline{u}) \\ &= \arg\max_{u} \sum_{n=1}^{\infty} \log P(r[n]|u[n])\end{aligned}$$

because:

- Channel is memoryless, such that $P(\underline{r}|\underline{u}) = \prod_{n=1}^{\infty} P(r[n]|u[n])$
- Maximizing a function is equivalent to maximizing its logarithm

Hard decoding

Hard decoding works on the BSC channel

ML criterion reduces to selecting the sequence $\underline{u}$ having the smallest hamming distance D to the received sequence $\underline{r}$:

$$
\begin{aligned}
P(\underline{r}|\underline{u}) &= p^D (1-p)^{L-D} \\
\log P(\underline{r}|\underline{u}) &= -D \log\left(\frac{1-p}{p}\right) + L \log(1-p) \propto -D
\end{aligned}
$$

Soft decoding

Soft decoding works on the Gaussian channel

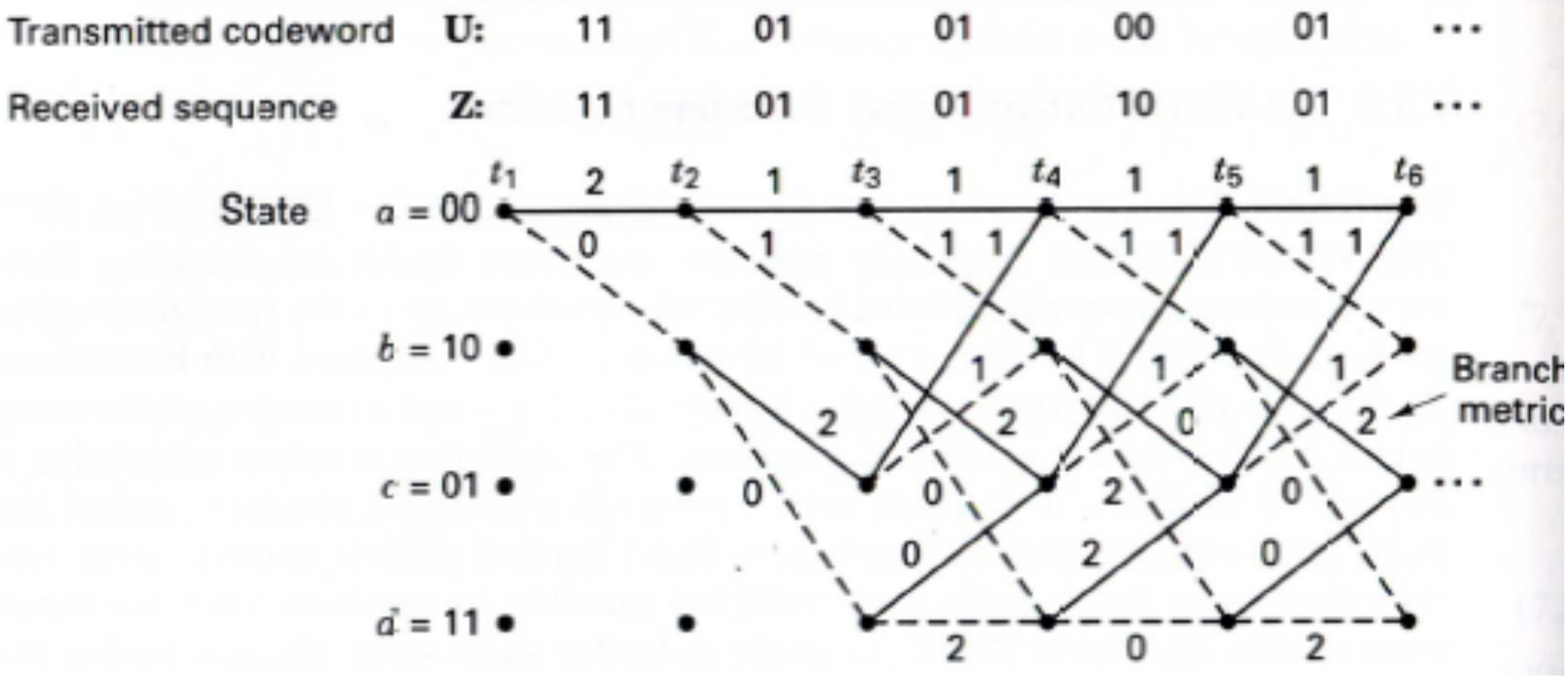
ML criterion reduces to selecting the sequence $\underline{u}$ having the smallest euclidian distance to the received sequence $\underline{r}$:

$$
\begin{aligned}
P(\underline{r}|\underline{u}) &= \prod_{n=1}^{\infty} \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left(-\frac{1}{2\sigma_w^2}(r[n]-u[n])^2\right) \\
\log P(\underline{r}|\underline{u}) &\propto -\sum_{n=1}^{\infty}(r[n]-u[n])^2
\end{aligned}
$$

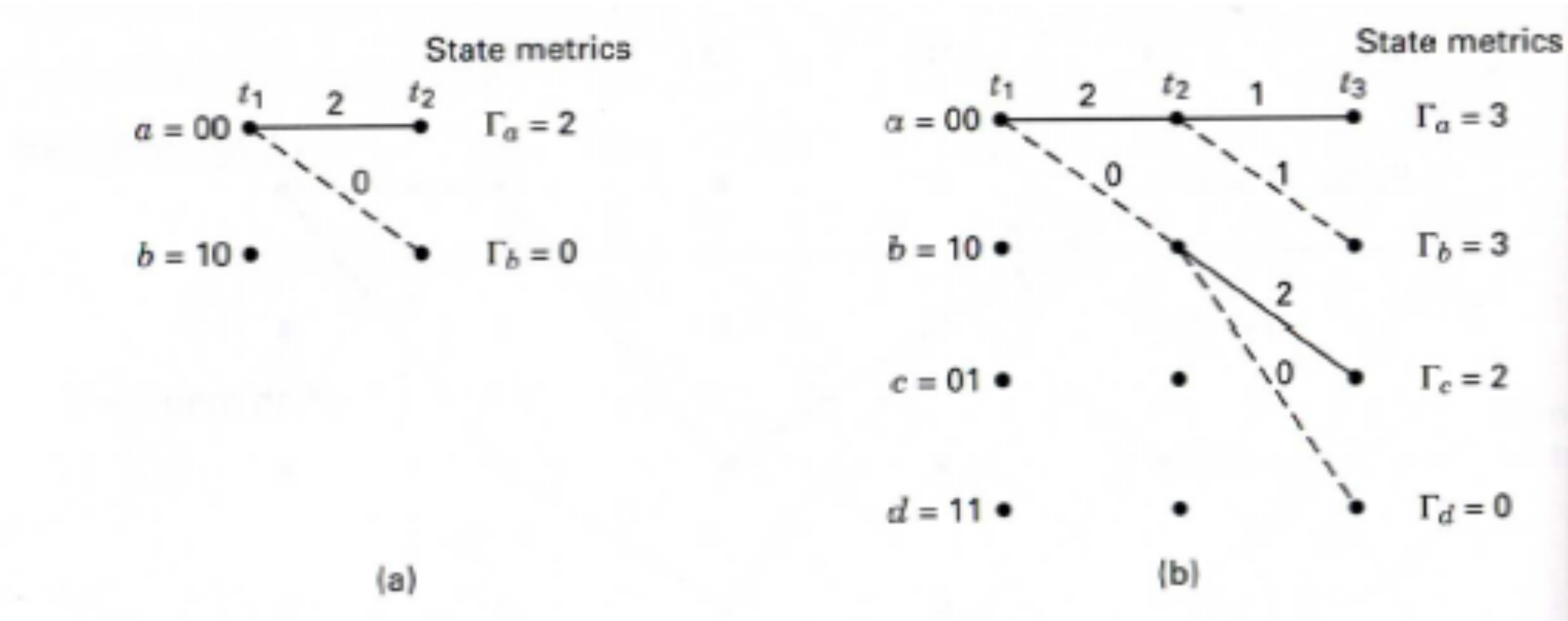
Viterbi algorithm computes the distance corresponding to all paths entering a node at each step and selects the path corresponding to the smallest distance

The elimination of the other paths is done without compromising the optimality of the trellis search, because any extension of these paths always has a larger distance than the survivor extended along the same path

Example: rate $1/2$, $K = 3$, hard decoding



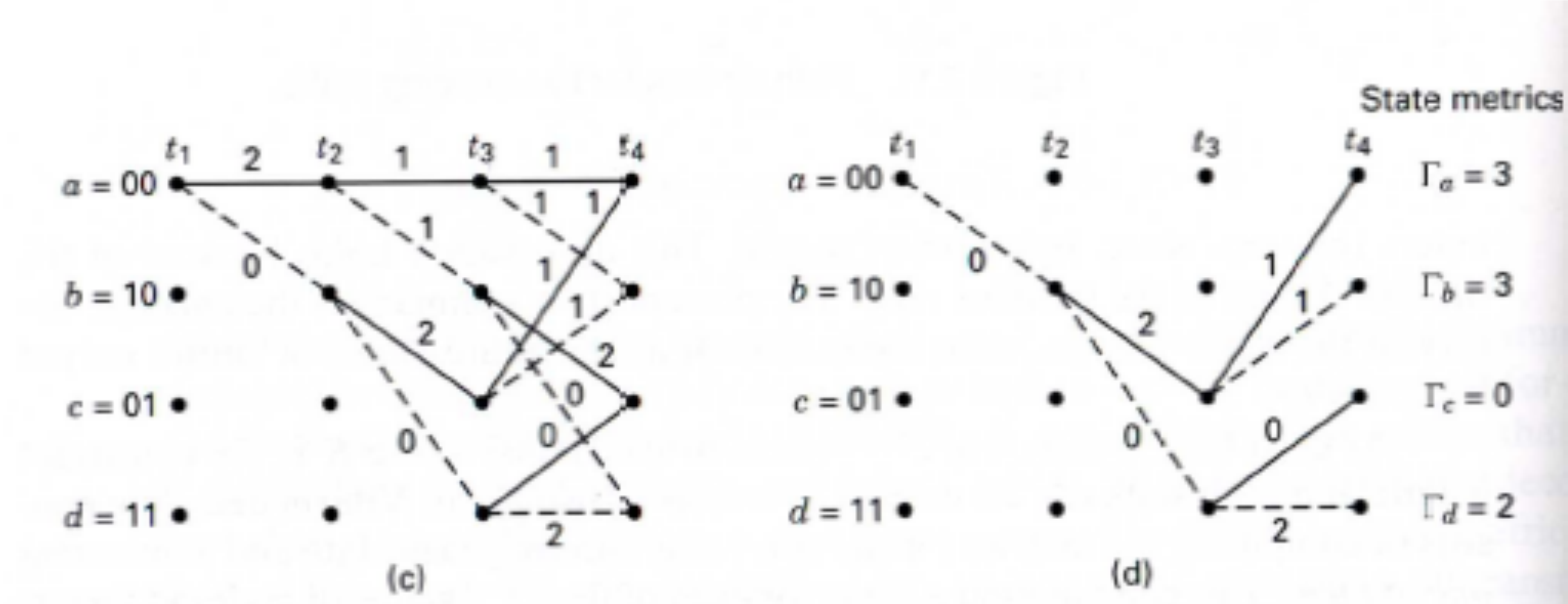
Viterbi algorithm



Compute the Hamming distance for depths 1 and 2 (initialization)

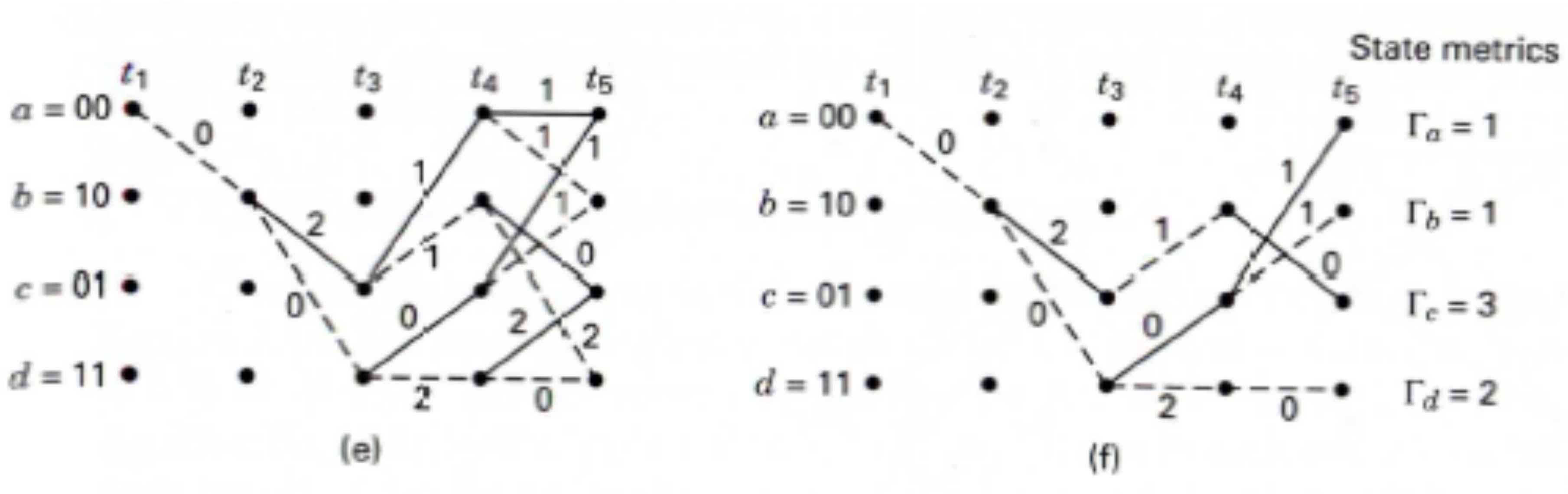
The accumulated distance is indicated over the candidate states

Viterbi algorithm



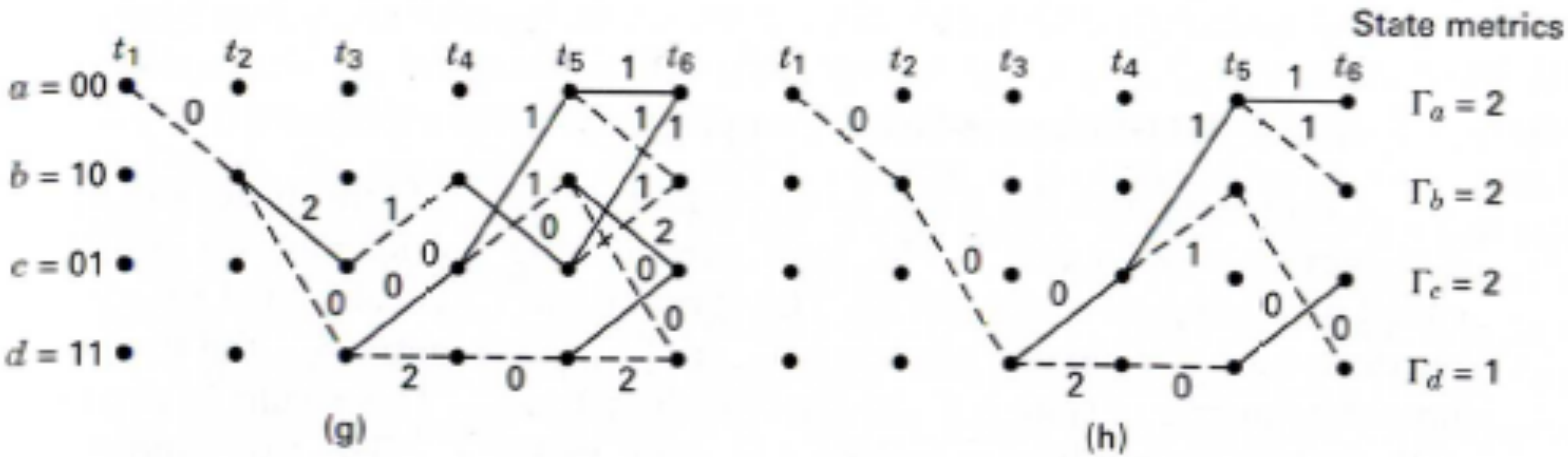
Two paths enter each state at depth 3

Keep only the arriving path with the lowest accumulated distance



Pursue the process to depth 4...

Viterbi algorithm



Pursue the process to depth 5...

Observe that the remaining paths often converge to a common path in the first stages (t_1 to t_3 in the example).

Therefore the surviving sequences can be truncated to the most recent stages to reduce to complexity of the algorithm.

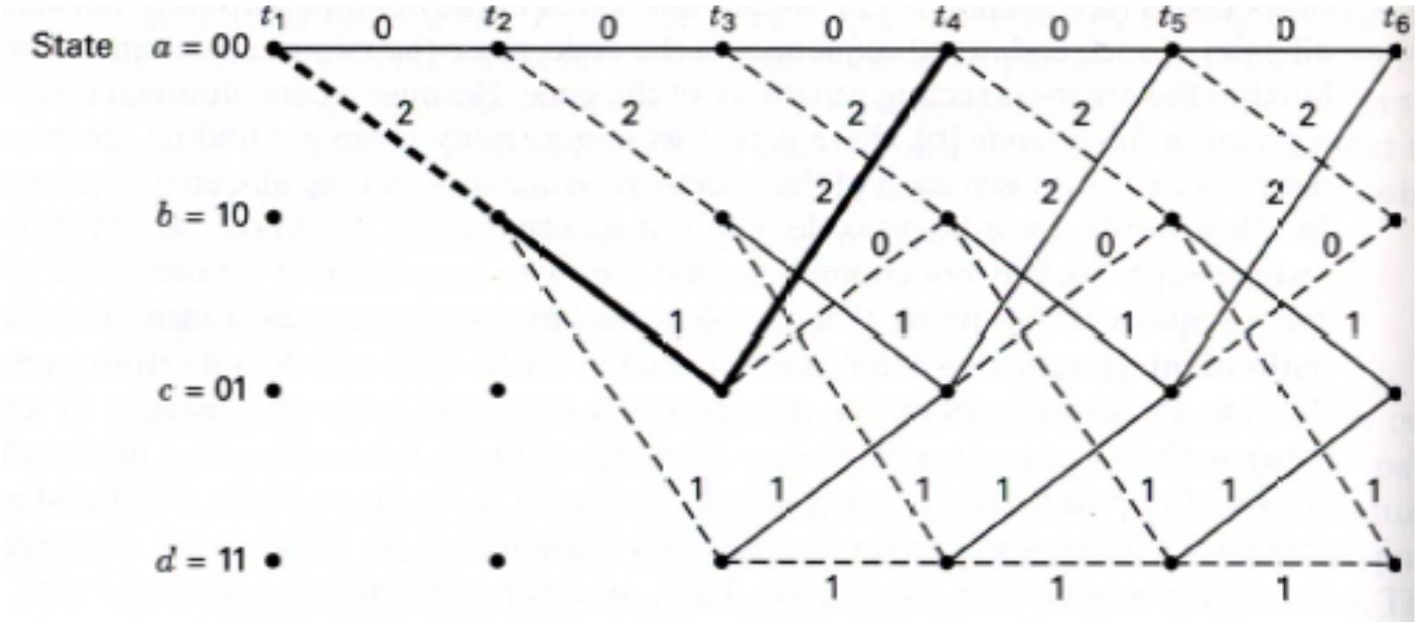
Distance properties

Free distance D_f: minimum distance between all pairs of codeword sequences

Error-correcting capability deduced from the free-distance:

$$ECC = \lfloor \frac{D_f - 1}{2} \rfloor$$

Distance properties



Given the all-zero transmission, the most probable error comes when the surviving path diverges from and remerges directly to the all-zero path.

The free distance is given by the weight of the corresponding path (5 in the example).

Best known convolutional codes

Rate	Constraint Length	Free Distance	Code Vector
$\frac{1}{2}$	3	5	111 101
$\frac{1}{2}$	4	6	1111 1011
$\frac{1}{2}$	5	7	10111 11001
$\frac{1}{2}$	6	8	101111 110101
$\frac{1}{2}$	7	10	1001111 1101101
$\frac{1}{2}$	8	10	10011111 11100101
$\frac{1}{2}$	9	12	110101111 100011101

Best known convolutional codes

$\frac{1}{3}$	3	8	111 111 101
$\frac{1}{3}$	4	10	1111 1011 1101
$\frac{1}{3}$	5	12	11111 11011 10101
$\frac{1}{3}$	6	13	101111 110101 111001
$\frac{1}{3}$	7	15	1001111 1010111 1101101
$\frac{1}{3}$	8	16	11101111 10011011 10101001

Source: J. P. Odenwalder, *Error Control Coding Handbook*, Linkabit Corp., San Diego, Calif., July 15, 1976.

Outline

- Introduction
- Linear block codes
- Low density parity check codes
- Convolutional codes
- **Exercises**

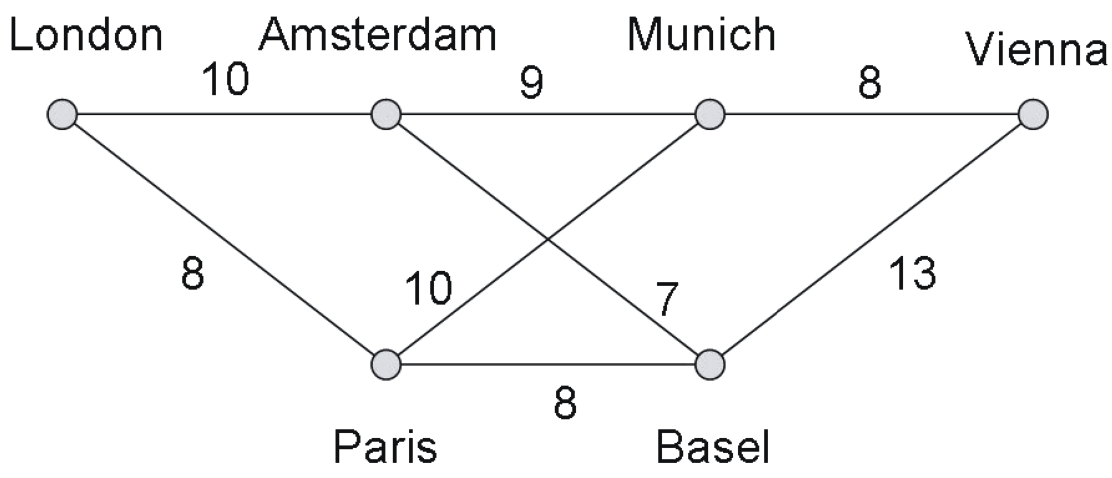
Exercise 1

Consider a (7,4) code whose generator matrix is:

$$\underline{\underline{G}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Find all codewords of the code
- Find the parity-check matrix and draw the standard array
- If the received vector is 1 1 0 1 1 0 1, determine the syndrome, the error pattern, the transmitted codeword and the message
- What is the error correction and detection capability of the code?

Exercise 2



Suppose you are trying to find the quickest way to get from London to Vienna. A trellis diagram has been constructed based on the various schedules. The labels on each path are travel times.

Using the Viterbi algorithm, find the fastest route. How does the algorithm work? What calculations must be made? What information must be retained in the memory?