Block Codes

Ling Cheng

School of Electrical and Information Engineering
University of the Witwatersrand

Outline

Generator Matrix and Parity-Check Matrix

- Generator Matrix:

$$\mathbf{v} = \mathbf{u}\mathbf{G}$$

-

$$\mathbf{G} := \begin{pmatrix} \mathbf{P} & \mathbf{I}_k \end{pmatrix}$$

- Parity-Check Matrix:

$$\mathbf{v}\mathbf{H}^T = \mathbf{0}$$

-

$$\mathbf{H} := \begin{pmatrix} \mathbf{I}_{n-k} & \mathbf{P}^T \end{pmatrix}$$

- Hamming Distance.
- Minimum Hamming Distance.

Generator Matrix and Parity-Check Matrix

- Generator Matrix:

$$\mathbf{v} = \mathbf{u}\mathbf{G}$$

-

$$\mathbf{G} := \begin{pmatrix} \mathbf{P} & \mathbf{I}_k \end{pmatrix}$$

- Parity-Check Matrix:

$$\mathbf{v}\mathbf{H}^T = \mathbf{0}$$

-

$$\mathbf{H} := \begin{pmatrix} \mathbf{I}_{n-k} & \mathbf{P}^T \end{pmatrix}$$

- Hamming Distance.
- Minimum Hamming Distance.

Generator Matrix and Parity-Check Matrix

- Generator Matrix:

$$\mathbf{v} = \mathbf{u}\mathbf{G}$$

-

$$\mathbf{G} := \begin{pmatrix} \mathbf{P} & \mathbf{I}_k \end{pmatrix}$$

- Parity-Check Matrix:

$$\mathbf{v}\mathbf{H}^T = \mathbf{0}$$

-

$$\mathbf{H} := \begin{pmatrix} \mathbf{I}_{n-k} & \mathbf{P}^T \end{pmatrix}$$

- Hamming Distance.
- Minimum Hamming Distance.

Generator Matrix and Parity-Check Matrix

- Generator Matrix:

$$\mathbf{v} = \mathbf{u}\mathbf{G}$$

- $$\mathbf{G} := \begin{pmatrix} \mathbf{P} & \mathbf{I}_k \end{pmatrix}$$

- Parity-Check Matrix:

$$\mathbf{v}\mathbf{H}^T = \mathbf{0}$$

- $$\mathbf{H} := \begin{pmatrix} \mathbf{I}_{n-k} & \mathbf{P}^T \end{pmatrix}$$

- Hamming Distance.
- Minimum Hamming Distance.

Generator Matrix and Parity-Check Matrix

- Generator Matrix:

$$\mathbf{v} = \mathbf{u}\mathbf{G}$$

- $$\mathbf{G} := \begin{pmatrix} \mathbf{P} & \mathbf{I}_k \end{pmatrix}$$

- Parity-Check Matrix:

$$\mathbf{v}\mathbf{H}^T = \mathbf{0}$$

- $$\mathbf{H} := \begin{pmatrix} \mathbf{I}_{n-k} & \mathbf{P}^T \end{pmatrix}$$

- Hamming Distance.
- Minimum Hamming Distance.

Generator Matrix and Parity-Check Matrix

- Generator Matrix:

$$\mathbf{v} = \mathbf{u}\mathbf{G}$$

-

$$\mathbf{G} := \begin{pmatrix} \mathbf{P} & \mathbf{I}_k \end{pmatrix}$$

- Parity-Check Matrix:

$$\mathbf{v}\mathbf{H}^T = \mathbf{0}$$

-

$$\mathbf{H} := \begin{pmatrix} \mathbf{I}_{n-k} & \mathbf{P}^T \end{pmatrix}$$

- Hamming Distance.
- Minimum Hamming Distance.

Decoding Sphere

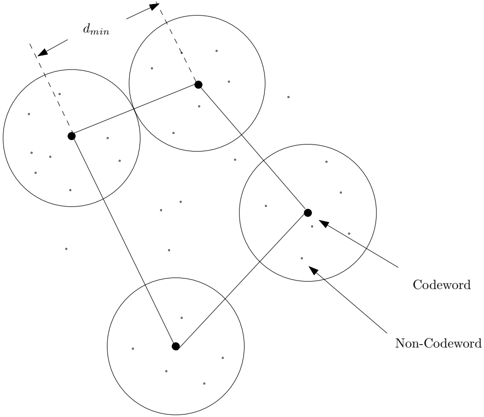

Figure:

Properties

- codeword length: $n = 2^m - 1$
- length of information: $k = 2^m - m - 1$
- length of parity bits: $n - k = m$
- $d_{\min} = 3$

Properties

- codeword length: $n = 2^m - 1$
- length of information: $k = 2^m - m - 1$
- length of parity bits: $n - k = m$
- $d_{\min} = 3$

Properties

- codeword length: $n = 2^m - 1$
- length of information: $k = 2^m - m - 1$
- length of parity bits: $n - k = m$
- $d_{\min} = 3$

Properties

- codeword length: $n = 2^m - 1$
- length of information: $k = 2^m - m - 1$
- length of parity bits: $n - k = m$
- $d_{\min} = 3$

Generating Hamming Codes

- choose m;
- enumerate all binary sequences of length m from $0\dots01$ to $1\dots1$;
- create the parity-check matrix by filling these binary sequences in the matrix column by column;
- obtain the generator matrix.

Generating Hamming Codes

- choose m;
- enumerate all binary sequences of length m from $0\ldots01$ to $1\ldots1$;
- create the parity-check matrix by filling these binary sequences in the matrix column by column;
- obtain the generator matrix.

Generating Hamming Codes

- choose m;
- enumerate all binary sequences of length m from $0\dots01$ to $1\dots1$;
- create the parity-check matrix by filling these binary sequences in the matrix column by column;
- obtain the generator matrix.

Generating Hamming Codes

- choose m;
- enumerate all binary sequences of length m from $0\ldots01$ to $1\ldots1$;
- create the parity-check matrix by filling these binary sequences in the matrix column by column;
- obtain the generator matrix.

Example

$$\mathbf{H} = \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right)$$

$$\mathbf{G} = \left(\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right)$$

Decoding Hamming Code

Given information $\mathbf{u} = 0001$, we sent the codeword

$$\begin{aligned}\mathbf{v} &= \mathbf{uG} \\ &= \begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix} \begin{pmatrix}1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1\end{pmatrix} \\ &= \begin{pmatrix}1 & 0 & 1 & 0 & 0 & 0 & 1\end{pmatrix}\end{aligned}$$

Decoding Hamming Code

As the result of one error, we receive

$$\mathbf{v}' = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

After the following calculation:

$$\mathbf{v}'\mathbf{H}^T = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}$$

Decoding Hamming Code

We find the syndrome sequence is same as the fifth row of $\mathbf{H}^T$. This leads to two discoveries:

1. Detection: there is error in the received sequence
2. Correction: if there is only one error, it must appear at the fifth position.

Therefore, we change the received sequence into

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

which is the correct codeword we send.

Basics of Linear Algebraic

- Irreducible Polynomial
- Primitive Polynomial
- Minimal Polynomial

$$f(x) = x^2 + x + 1$$
$$x^2 + x + 1 \neq (x + c_1)(x + c_2)$$
$$x^2 + x + 1 = 1(x^2 + x + 1)$$

Basics of Linear Algebraic

- Irreducible Polynomial
- Primitive Polynomial
- Minimal Polynomial

Irreducible polynomial with degree m divides $x^{2^m-1}+1$, for example, irreducible x^2+x+1 divides x^3+1. The primitive polynomial divides polynomial x^n+1 with the minimum value $n=2^m-1$. α is the root of the primitive polynomial. Use primitive polynomial to construct Galois Field.

Basics of Linear Algebraic

- Irreducible Polynomial
- Primitive Polynomial
- Minimal Polynomial

α^0, α^1, ..., α^{2^m-2} are the roots of $x^{2^m-1}+1=0$. Conjugate roots generate the minimum polynomial. For example, for $GF(2^4)$, α, α^2, α^4, α^8 are the conjugate roots. We have the minimal polynomial:

$$\Phi_1 = (x+\alpha)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8) = x^4+x+1.$$

Construction of Galois Field

Table: Construction of a $GF(2^4)$ field by $h(x) = 1 + x + x^4$

Codeword	Polynomial in $x \pmod{h(x)}$	Power of α
0000	0	–
1000	1	1
0100	x	α
0010	x^2	α^2
0001	x^3	α^3
1100	$1 + x$	α^4
0110	$x + x^2$	α^5
0011	$x^2 + x^3$	α^6
1101	$1 + x + x^3$	α^7
⋮	⋮	⋮
1001	$1 + x^3$	α^{14}

Minimum Polynomial

Table: Minimal polynomials of the elements in $GF(2^4)$

Elements of $GF(2^4)$ using $h(x)$	Minimal polynomial
0	x
1	$x+1$
$\alpha, \alpha^2, \alpha^4, \alpha^8$	x^4+x+1
$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$x^4+x^3+x^2+x+1$
α^5, α^{10}	x^2+x+1
$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$	x^4+x^3+1

BCH Code

- t error correcting codes;
- generator polynomial $g(x) = LCM\{\Phi_1(x), \Phi_2(x), \ldots\}$;
- $g(x)$ has roots α, α^1, ..., α^{2t};
- Example of 2 error correcting BCH code:

BCH Code

- t error correcting codes;
- generator polynomial $g(x) = LCM\{\Phi_1(x), \Phi_2(x), \ldots\}$;
- $g(x)$ has roots α, α^1, ..., α^{2t};
- Example of 2 error correcting BCH code:

BCH Code

- t error correcting codes;
- generator polynomial $g(x) = LCM\{\Phi_1(x), \Phi_2(x), \ldots\}$;
- $g(x)$ has roots α, α^1, ..., α^{2t};
- Example of 2 error correcting BCH code:

BCH Code

- t error correcting codes;
- generator polynomial $g(x) = LCM\{\Phi_1(x), \Phi_2(x), \ldots\}$;
- $g(x)$ has roots α, α^1, ..., α^{2t};
- Example of 2 error correcting BCH code:

(15, 7) BCH code

$$g(x) = \Phi_1(x)\Phi_3(x) = 1 + x^4 + x^6 + x^7 + x^8$$

Properties and Encoding of Reed-Solomon Code

- block length: $n = q - 1$;
- parity-check length: $n - k = 2t$;
- $d_{\min} = 2t + 1$;
- $g(x) = (x + \alpha^{m+1})(x + \alpha^{m+2}) \dots (x + \alpha^{m+\delta-1})$

Properties and Encoding of Reed-Solomon Code

- block length: $n = q - 1$;
- parity-check length: $n - k = 2t$;
- $d_{\min} = 2t + 1$;
- $g(x) = (x + \alpha^{m+1})(x + \alpha^{m+2}) \ldots (x + \alpha^{m+\delta-1})$

Properties and Encoding of Reed-Solomon Code

- block length: $n = q - 1$;
- parity-check length: $n - k = 2t$;
- $d_{\min} = 2t + 1$;
- $g(x) = (x + \alpha^{m+1})(x + \alpha^{m+2}) \ldots (x + \alpha^{m+\delta-1})$

Properties and Encoding of Reed-Solomon Code

- block length: $n = q - 1$;
- parity-check length: $n - k = 2t$;
- $d_{\min} = 2t + 1$;
- $g(x) = (x + \alpha^{m+1})(x + \alpha^{m+2}) \ldots (x + \alpha^{m+\delta-1})$

Berlekamp-Massey Decoding

Table: Calculation results of step 3

i	$q_i - p_i$									d_i	z_i
-1	α^0	α^7	α^0	α^9	α^{12}	α^9	α^7	–	α^0	-1	
0	α^7	α^0	α^9	α^{12}	α^9	α^7	–	α^0		0	-1
1	α^3	α^0	α^{13}	α^{14}	α^{14}	–	α^0	α^7		0	0
2	α^{12}	α^7	α^6	α^{12}	–	α^0	α^8			1	1
3	α^0	α^{10}	α^9	–	α^0	α^{12}	α^1			1	2
4	0	0	–	α^0	α^{10}	α^6				2	3
5	0	–	α^0	α^{10}	α^6					3	3
6	–	α^0	α^{10}	α^6						4	3

Why LDPC?

$$
\begin{aligned}
x_1 + x_2 + x_3 &= 1 \\
x_1 + x_2 &= 1 \\
x_1 &= 1
\end{aligned}
$$

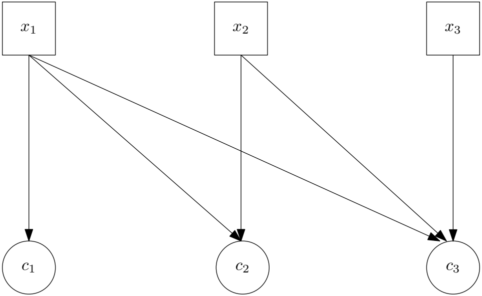

Figure:

Why LDPC?

$$x_2 + x_3 = 0$$
$$x_2 = 0$$

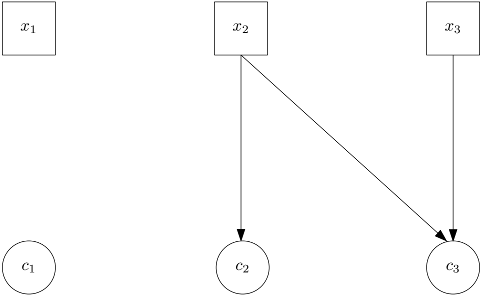

Figure:

Why LDPC?

$$x_3 = 0$$

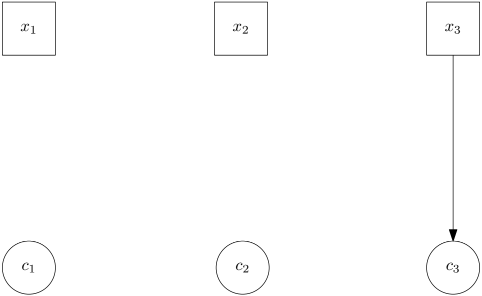

Figure:

Why LDPC?

Normal Parity-Check Code:

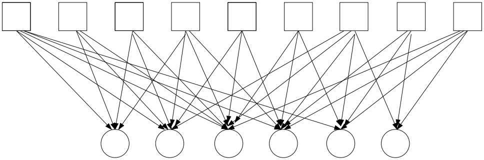

Figure:

Why LDPC?

Low-Density Parity-Check Code:

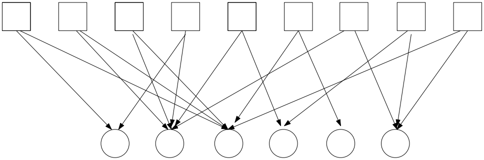

Figure:

Properties and Encoding of type-I $(0, s)$-th order EG-LDPC code

- $$\frac{w_r}{w_c} = \frac{n}{n-k},$$

  and $w_r \ll n$.

- $J = \frac{2^{(m-1)s}(2^{ms}-1)}{2^s-1}$
- $\gamma = \frac{2^{ms}-1}{2^s-1}$;
- $d_{\min} \geq \gamma + 1$

Properties and Encoding of type-I $(0, s)$-th order EG-LDPC code

- $$\frac{w_r}{w_c} = \frac{n}{n-k},$$

  and $w_r \ll n$.

- $J = \frac{2^{(m-1)s}(2^{ms}-1)}{2^s-1}$

- $\gamma = \frac{2^{ms}-1}{2^s-1};$

- $d_{\min} \geq \gamma + 1$

Properties and Encoding of type-I $(0, s)$-th order EG-LDPC code

- $$\frac{w_r}{w_c} = \frac{n}{n-k},$$
  and $w_r \ll n$.
- $J = \frac{2^{(m-1)s}(2^{ms}-1)}{2^s-1}$
- $\gamma = \frac{2^{ms}-1}{2^s-1}$;
- $d_{\min} \geq \gamma + 1$

Properties and Encoding of type-I (0, s)-th order EG-LDPC code

- $$\frac{w_r}{w_c} = \frac{n}{n-k},$$
  and $w_r \ll n$.
- $J = \frac{2^{(m-1)s}(2^{ms}-1)}{2^s-1}$
- $\gamma = \frac{2^{ms}-1}{2^s-1}$;
- $d_{\min} \geq \gamma + 1$

Tanner Graph

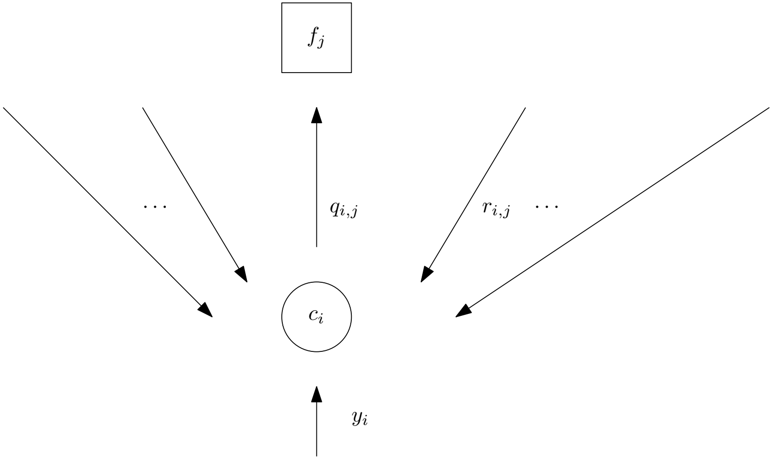

Figure: Tanner graph of the check node of a LDPC code

Tanner Graph

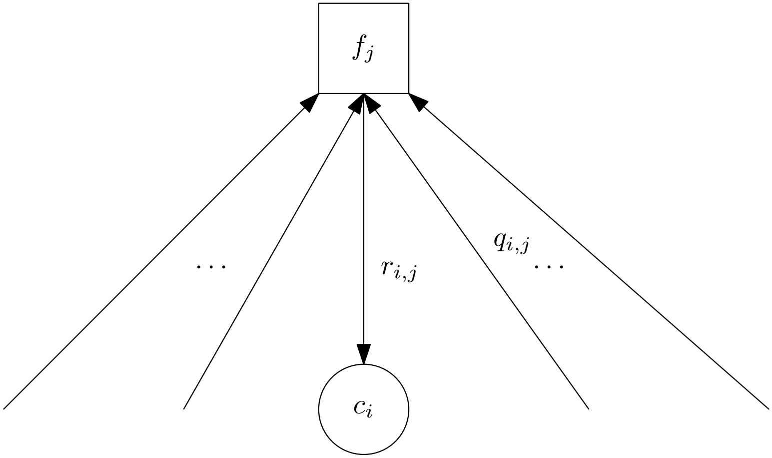

Figure: Tanner graph of the symbol node of a LDPC code

Crux of Belief Propagation

- Listen to Credited Nodes
- Listen to Majority
- Pass Reliable Information
- Convert Unreliable Node

Create Incident Vector

- Know Incident Vector, know parity-check matrix.
- Euclidean Geometry: Multiple dimensions represent one dimension
- How to find an incident vector

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_0^0 \\ \mathbf{I}_0^1 \\ \vdots \\ \mathbf{I}_0^{n-1} \end{bmatrix}.$$

Create Incident Vector

- Know Incident Vector, know parity-check matrix.
- Euclidean Geometry: Multiple dimensions represent one dimension
- How to find an incident vector

Example($GF(2^4)$):

$$
\begin{aligned}
0 &\rightarrow (0,0,0,0)\\
1 &\rightarrow (0,0,0,1)\\
2 &\rightarrow (0,0,1,0)\\
\cdots &\rightarrow \cdots\\
15 &\rightarrow (1,1,1,1)
\end{aligned}
$$

Create Incident Vector

- Know Incident Vector, know parity-check matrix.
- Euclidean Geometry: Multiple dimensions represent one dimension
- How to find an incident vector

Example: $GF(2^4)$ $GF(2^2) : \{0, 1, \beta, \beta^2\}$ subset of $GF(2^4) : \{0, 1, \alpha, \ldots, \alpha^{14}\}$. $\beta = \alpha^5$. For α^{14}, $\{\alpha^{14} + 0\alpha, \alpha^{14} + 1\alpha, \alpha^{14} + \beta\alpha, \alpha^{14} + \beta^2\alpha\}$ corresponds to the incident vector 000000011010001 ($\{\alpha^7, \alpha^8, \alpha^{10}, \alpha^{14}\}$).

Create Very-Long LDPC

- Why long code? (Shannon Limit)
- Advantages of using short LDPC create long LDPC.
- Higher rate.
- Lower density.
- Guaranteed error correcting capability.
- etc.

Create Very-Long LDPC

- Why long code? (Shannon Limit)
- Advantages of using short LDPC create long LDPC.
- Higher rate.
- Lower density.
- Guaranteed error correcting capability.
- etc.