An introduction to coding theory

Adrish Banerjee

Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh
India

Feb. 27, 2017

Lecture #14B: Decoding of low density parity check codes-II: Belief Propagation Algorithm
Outline of the talk

- Message passing algorithm
  - Decoding in probability domain.
Theorem:

Consider a sequence of \( m \) independent random variables \( A = [A_1, A_2, \ldots, A_m] \), where \( P(A_k = 1) = p_k \). Then

\[
P(A \text{ has even parity}) = \frac{1}{2} + \frac{1}{2} \prod_{k=1}^{m} (1 - 2p_k)
\]

and

\[
P(A \text{ has odd parity}) = \frac{1}{2} - \frac{1}{2} \prod_{k=1}^{m} (1 - 2p_k).
\]
Consider the code with parity check matrix, $H$:

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

$c = [c_0, c_1, \ldots, c_{n-1}]$ is the codeword under consideration.
Consider the code with parity check matrix, $H$:

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

$c = [c_0, c_1, \ldots, c_{n-1}]$ is the codeword under consideration.

$X_i = (-1)^{c_i} \in \{+1, -1\}$, the BPSK-modulated version of $c_i$.

$Y_i = X_i + n_i$, where $n_i$ is zero-mean Gaussian with variance $\sigma^2$. 
Consider the code with parity check matrix, $H$:

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

$c = [c_0, c_1, \ldots, c_{n-1}]$ is the codeword under consideration.

$X_i = (-1)^{c_i} \in \{+1, -1\}$, the BPSK-modulated version of $c_i$.

$Y_i = X_i + n_i$, where $n_i$ is zero-mean Gaussian with variance $\sigma^2$.

$R_j = \{i : h_{j,i} = 1\}$ = location of 1’s in row $j$ of $H$ = the indices of the bits checked by the $j^{th}$ parity check.

$C_i = \{j : h_{j,i} = 1\}$ = location of 1’s in column $i$ of $H$ = the parity checks involving the $i^{th}$ codebit.
Notation

- \( R_{j \setminus i} = R_j \setminus \{i\} \)

- \( C_{i \setminus j} = C_i \setminus \{j\} \)
Notation

- \( R_j \setminus i = R_j \setminus \{i\} \)
- \( C_i \setminus j = C_i \setminus \{j\} \)
- \( c_{k,j}(i) = k^{th} \) bit in the \( j^{th} \) parity check involving the code bit \( c_i \). (So \( j \in C_i \) and \( k \in R_j \).)

- \( Y_{k,j}(i) = (-1)^{c_{k,j}(i)} + n_{k,j}(i) \), received signal corresponding to \( c_{k,j}(i) \).
Notation

- $R_j \setminus i = R_j \{i\}$
- $C_i \setminus j = C_i \{j\}$
- $c_{k,j}(i) = k^{th}$ bit in the $j^{th}$ parity check involving the code bit $c_i$. (So $j \in C_i$ and $k \in R_j$.)
- $Y_{k,j}(i) = (-1)^{c_{k,j}(i)} + n_{k,j}(i)$, received signal corresponding to $c_{k,j}(i)$.
- $p_i = P(c_i = 1|Y_i = y_i) = P(X_i = -1|Y_i = y_i) = 1/(1 + \exp(2y_i/\sigma^2))$. 
- $p_{k,j}(i) = P(c_{k,j}(i) = 1|y_{k,j}(i))$. 

Adrish Banerjee
Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh India
Theorem

The a posteriori probability (APP) ratio for $c_i$ given the received word $y = [y_0, y_1, \ldots, y_{n-1}]$ and given the event $S_i = \{ \text{the bits in } c \text{ satisfy the parity check constraints involving } c_i \}$, is given by

$$
\frac{P(c_i = 0 | y, S_i)}{P(c_i = 1 | y, S_i)} = \frac{(1 - p_i)}{p_i} \frac{\prod_{j \in C_i} \left(1 + \prod_{j' \in R_j \setminus i} (1 - 2p_{j'}(i))\right)}{\prod_{j \in C_i} \left(1 - \prod_{j' \in R_j \setminus i} (1 - 2p_{j'}(i))\right)}
$$

Proof

From Bayes’ rule:

$$
\frac{P(c_i = 0 | y, S_i)}{P(c_i = 1 | y, S_i)} = \frac{1-p_i}{p_i} \frac{P(c_i = 0 | y_i) P(S_i | c_i = 0, y)}{P(c_i = 1 | y_i) P(S_i | c_i = 1, y)}.
$$
Proof

From Bayes’ rule:

\[
\frac{P(c_i = 0 | y, S_i)}{P(c_i = 1 | y, S_i)} = \frac{1 - p_i}{p_i} \frac{P(c_i = 0 | y, S_i)}{P(c_i = 1 | y, S_i)} \frac{P(S_i | c_i = 0, y)}{P(S_i | c_i = 1, y)}.
\]

Let’s consider the term \( P(S_i | c_i = 0, y) \). Given \( c_i = 0 \), \( S_i \) holds if each of \( w_c \) parity checks involving \( c_i \) has the property that the \( w_r - 1 \) bits in the check other than \( c_i \) have even parity.

For parity check \( j \in C_i \), the probability that the \( w_r - 1 \) bits other than \( c_i \) have even parity is given by the lemma to be:

\[
\frac{1}{2} + \frac{1}{2} \prod_{i' \in R_j \setminus i} (1 - 2p_{i',j}(i)).
\]
The independence of the $y_i$'s means that the probability that all $w_c$ parity checks involving $c_i$ are satisfied (given $c_i = 0$) is just

$$P(S_i|c_i = 0, y) = \prod_{j \in C_i} \left( \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_j \setminus i} (1 - 2p_{i'j}(i)) \right).$$

Similar analysis assuming $c_i = 1$ yields

$$P(S_i|c_i = 1, y) = \prod_{j \in C_i} \left( \frac{1}{2} - \frac{1}{2} \prod_{i' \in R_j \setminus i} (1 - 2p_{i'j}(i)) \right).$$
Probabilistic Decoding

- \( r_{j,i}(x) \) is the message passed from the \( j^{th} \) check node to the bit node \( X_i = x \).
  \[
  r_{j,i}(+1) = P(\text{Parity check } j \text{ satisfied} | c_i = 0, \text{other bits in check } j \text{ have distributions given by } q) \\
  = \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_j \setminus i} (1 - 2q_{i',j}(-1)).
  \]

  and so
  \[
  r_{j,i}(-1) = P(\text{Parity check } j \text{ satisfied} | c_i = 1, \text{other bits in check } j \text{ have distributions given by } q) \\
  = P(\text{Parity check } j \text{ not satisfied} | c_i = 0, \text{other bits in check } j \text{ have distributions given by } q) \\
  = 1 - r_{j,i}(+1).
  \]

- \( q_{i,j}(x) \) is the message passed from the bit node \( X_i = x \) to the \( j^{th} \) check node.
  \[
  q_{i,j}(+1) = P(X_i = +1 | y_i, \text{information from check nodes other than } j^{th} \text{ check node}).
  \]
  \[
  \frac{q_{i,j}(+1)}{q_{i,j}(-1)} = \frac{(1 - p_i) \prod_{j' \in C \setminus j} r_{j',i}(+1)}{p_i \prod_{j' \in C \setminus j} r_{j',i}(-1)}.\]
Probabilistic Decoding

For all $i,j$ such that $h_{j,i} = 1$. (So $i$ indexes the bit nodes and $j$ indexes the parity checks.)

- **Step 0:** Initialize:
  - Set $p_i = P(c_i = 1|Y_i = y_i) = 1/(1 + \exp(2y_i/\sigma^2))$. 
Probabilistic Decoding

For all $i,j$ such that $h_{j,i} = 1$. (So $i$ indexes the bit nodes and $j$ indexes the parity checks.)

**Step 0:** Initialize:
- Set $p_i = P(c_i = 1|Y_i = y_i) = 1/(1 + \exp(2y_i/\sigma^2))$.
- $q_{i,j}(+1) = 1 - p_i$.
- $q_{i,j}(-1) = p_i$. 
Probabilistic Decoding

For all $i, j$ such that $h_{j,i} = 1$. (So $i$ indexes the bit nodes and $j$ indexes the parity checks.)

- **Step 0:** Initialize:
  - Set $p_i = P(c_i = 1|Y_i = y_i) = 1/(1 + \exp(2y_i/\sigma^2))$.
  - $q_{i,j}(+1) = 1 - p_i$.
  - $q_{i,j}(-1) = p_i$.
- **Step 1:** Pass information from check nodes to bit nodes:
  - $r_{j,i}(+1) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_j \setminus i} (1 - 2q_{i',j}(-1))$
Probabilistic Decoding

For all $i,j$ such that $h_{j,i} = 1$. (So $i$ indexes the bit nodes and $j$ indexes the parity checks.)

**Step 0:** Initialize:
- Set $p_i = P(c_i = 1|Y_i = y_i) = 1/(1 + \exp(2y_i/\sigma^2))$.
- $q_{i,j}(+1) = 1 - p_i$.
- $q_{i,j}(-1) = p_i$.

**Step 1:** Pass information from check nodes to bit nodes:
- $r_{j,i}(+1) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_j \setminus i} (1 - 2q_{i',j}(-1))$
- $r_{j,i}(-1) = 1 - r_{j,i}(+1)$.

**Step 2:** Pass information from bit nodes to check nodes:
Probabilistic Decoding

Step 2: Pass information from bit nodes to check nodes:

\[
q_{i,j}(+1) = K_{i,j}(1 - p_i) \prod_{j' \in C_{i \cup j} \setminus j} r_{j',i}(+1)
\]

\[
q_{i,j}(-1) = K_{i,j} p_i \prod_{j' \in C_{i \cup j} \setminus j} r_{j',i}(-1)
\]
Step 2: Pass information from bit nodes to check nodes:
- \( q_{i,j}(+1) = K_{i,j}(1 - p_i) \prod_{j' \in C_{i \setminus j}} r_{j',i}(+1) \)
- \( q_{i,j}(-1) = K_{i,j}p_i \prod_{j' \in C_{i \setminus j}} r_{j',i}(-1) \)

Here, the constants \( K_{i,j} \) are chosen so as to guarantee that 
\( q_{i,j}(+1) + q_{i,j}(-1) = 1 \).

Step 3: Compute the APP ratios for each bit position \( i \):
Probabilistic Decoding

- **Step 2:** Pass information from bit nodes to check nodes:
  \[ q_{i,j}(+1) = K_{i,j}(1 - p_i) \prod_{j' \in \mathcal{C}_i \setminus j} r_{j',i}(+1) \]
  \[ q_{i,j}(-1) = K_{i,j} p_i \prod_{j' \in \mathcal{C}_i \setminus j} r_{j',i}(-1) \]
  Here, the constants \( K_{i,j} \) are chosen so as to guarantee that
  \[ q_{i,j}(+1) + q_{i,j}(-1) = 1. \]

- **Step 3:** Compute the APP ratios for each bit position \( i \):
  \[ Q_i(+1) = K_i (1 - p_i) \prod_{j \in \mathcal{C}_i} r_{j,i}(+1) \]
  \[ Q_i(-1) = K_i p_i \prod_{j \in \mathcal{C}_i} r_{j,i}(-1) \]
Probabilistic Decoding

Step 2: Pass information from bit nodes to check nodes:
- \( q_{i,j}(+1) = K_{i,j}(1 - p_i) \prod_{j' \in C_i \setminus j} r_{j',i}(+1) \)
- \( q_{i,j}(-1) = K_{i,j}p_i \prod_{j' \in C_i \setminus j} r_{j',i}(-1) \)

Here, the constants \( K_{i,j} \) are chosen so as to guarantee that \( q_{i,j}(+1) + q_{i,j}(-1) = 1. \)

Step 3: Compute the APP ratios for each bit position \( i \):
- \( Q_i(+1) = K_i(1 - p_i) \prod_{j \in C_i} r_{j,i}(+1) \)
- \( Q_i(-1) = K_ip_i \prod_{j \in C_i} r_{j,i}(-1) \)

Here, the constants \( K_i \) are chosen so as to guarantee that \( Q_i(+1) + Q_i(-1) = 1. \)

Step 4: Compute the hard decisions and decide if it’s time to stop.

\[
\hat{c}_i = \begin{cases} 
1 & \text{if } Q_i(-1) \geq 0.5; \\
0 & \text{otherwise.}
\end{cases}
\]
Probabilistic Decoding

- **Step 4**: Compute the hard decisions and decide if it's time to stop.

\[ \hat{c}_i = \begin{cases} 
1 & \text{if } Q_i(-1) \geq 0.5; \\
0 & \text{otherwise.} 
\end{cases} \]

- If( \([\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_{n-1}]H^T = 0\) or (Maximum # of iterations reached)) then stop, else repeat Steps 1-4.

**Example**

Consider the code with parity check matrix, \(H\):

\[
H = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}
\]
Consider the code with parity check matrix, $H$:

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

$n = 8$, $m = n - k = 4$, $d_{\min} = 3$
Example

![Matrix Example]

Initialization:

\[ q_{i,j}(x) = \frac{1}{1 + \exp(-2xy_i/\sigma^2)} \]

for each \( i, j \) such that \( h_{j,i} = 1 \).
Example

Initialization:
- \( q_{i,j}(x) = 1/(1 + \exp(-2xy_i/\sigma^2)) \) for each \( i,j \) such that \( h_{j,i} = 1 \).
- \( q_{0,0}(-1) = q_{0,2}(-1) = 0.310 \) and \( q_{0,0}(+1) = q_{0,2}(-1) = 0.690 \).
- \( q_{1,0}(-1) = q_{1,3}(-1) = 0.310 \) and \( q_{1,0}(+1) = q_{1,3}(+1) = 0.690 \).
Example

Initialization:
- $q_{i,j}(x) = 1/(1 + \exp(-2xy_i/\sigma^2))$ for each $i,j$ such that $h_{j,i} = 1$.
- $q_{0,0}(-1) = q_{0,2}(-1) = 0.310$ and $q_{0,0}(+1) = q_{0,2}(-1) = 0.690$.
- $q_{1,0}(-1) = q_{1,3}(-1) = 0.310$ and $q_{1,0}(+1) = q_{1,3}(+1) = 0.690$.
- $q_{2,0}(-1) = 0.973$ and $q_{2,0}(+1) = 0.027$.

Example

Initialization:
- $q_{i,j}(x) = 1/(1 + \exp(-2xy_i/\sigma^2))$ for each $i,j$ such that $h_{j,i} = 1$.
- $q_{0,0}(-1) = q_{0,2}(-1) = 0.310$ and $q_{0,0}(+1) = q_{0,2}(-1) = 0.690$.
- $q_{1,0}(-1) = q_{1,3}(-1) = 0.310$ and $q_{1,0}(+1) = q_{1,3}(+1) = 0.690$.
- $q_{2,0}(-1) = 0.973$ and $q_{2,0}(+1) = 0.027$.
- $q_{3,1}(-1) = q_{3,2}(-1) = 0.083$ and $q_{3,1}(+1) = q_{3,1}(+1) = 0.917$. 
Example

- $q_{4,1}(-1) = q_{4,3}(-1) = 0.119$ and $q_{4,1}(+1) = q_{4,3}(+1) = 0.881$.

- $q_{5,1}(-1) = 0.988$ and $q_{5,1}(+1) = 0.012$. 

Example

- \( q_{4,1}(-1) = q_{4,3}(-1) = 0.119 \) and \( q_{4,1}(+1) = q_{4,3}(+1) = 0.881 \).
- \( q_{5,1}(-1) = 0.988 \) and \( q_{5,1}(+1) = 0.012 \).
- \( q_{6,2}(-1) = 0.832 \) and \( q_{6,2}(+1) = 0.168 \).
- \( q_{7,3}(-1) = 0.992 \) and \( q_{7,3}(+1) = 0.008 \).
Example

Now compute $r_{j,i}$'s from $q_{i,j}$'s:

\[
\begin{align*}
r_{0,0}(+1) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{0 \setminus 0}} (1 - 2q_{i',0}(-1)) \\
&= \frac{1}{2} + \frac{1}{2} (1 - 2q_{1,0}(-1))(1 - 2q_{2,0}(-1)) \\
&= \frac{1}{2} + \frac{1}{2} (1 - 2(0.31))(1 - 2(0.973)) \\
&= 0.320.
\end{align*}
\]

In a similar way:
Example

Now compute $r_{j,i}$'s from $q_{i,j}$'s:

$$r_{0,0}(+1) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{0,0}} (1 - 2q_{i',0}(-1))$$

$$= \frac{1}{2} + \frac{1}{2} (1 - 2q_{1,0}(-1))(1 - 2q_{2,0}(-1))$$

$$= \frac{1}{2} + \frac{1}{2} (1 - 2(0.31))(1 - 2(0.973))$$

$$= 0.320.$$  

In a similar way:

$${r_{0,1}(+1) = 0.5 + 0.5(1 - 2(0.31))(1 - 2(0.973)) = 0.32}$$

$${r_{0,2}(+1) = 0.5 + 0.5(1 - 2(0.31))(1 - 2(0.31)) = 0.57}$$
Example

Now compute \( r_{j,i} \)'s from \( q_{i,j} \)'s:

\[
r_{0,0}(+1) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{0,0}} (1 - 2q_{i',0}(-1))
\]
\[
= \frac{1}{2} + \frac{1}{2} (1 - 2q_{1,0}(-1))(1 - 2q_{2,0}(-1))
\]
\[
= \frac{1}{2} + \frac{1}{2} (1 - 2(0.31))(1 - 2(0.973))
\]
\[
= 0.320.
\]

In a similar way:
- \( r_{0,1}(+1) = 0.5 + 0.5(1 - 2(0.31))(1 - 2(0.973)) = 0.32 \)
- \( r_{0,2}(+1) = 0.5 + 0.5(1 - 2(0.31))(1 - 2(0.31)) = 0.57 \)
- \( r_{1,3}(+1) = 0.5 + 0.5(1 - 2(0.119))(1 - 2(0.988)) = 0.128 \)
- \( r_{2,0}(+1) = 0.5 + 0.5(1 - 2(0.083))(1 - 2(0.832)) = 0.223 \)

Adrish Banerjee
Department of Electrical Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh India

An introduction to coding theory
Example

Now compute \( r_{j,i} \)'s from \( q_{i,j} \)'s:

\[
\begin{align*}
r_{0,0}(+1) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_0 \setminus 0} (1 - 2q_{i',0}(-1)) \\
&= \frac{1}{2} + \frac{1}{2}(1 - 2q_{1,0}(-1))(1 - 2q_{2,0}(-1)) \\
&= \frac{1}{2} + \frac{1}{2}(1 - 2(0.31))(1 - 2(0.973)) \\
&= 0.320.
\end{align*}
\]

In a similar way:

\[
\begin{align*}
r_{0,1}(+1) &= 0.5 + 0.5(1 - 2(0.31))(1 - 2(0.973)) = 0.32 \\
r_{0,2}(+1) &= 0.5 + 0.5(1 - 2(0.31))(1 - 2(0.973)) = 0.57 \\
r_{1,3}(+1) &= 0.5 + 0.5(1 - 2(0.119))(1 - 2(0.988)) = 0.128 \\
r_{2,0}(+1) &= 0.5 + 0.5(1 - 2(0.083))(1 - 2(0.832)) = 0.223.
\end{align*}
\]

\( r_{j,i}(-1) = 1 - r_{j,i}(+1) \).

Example

Now compute \( q_{i,j} \)'s from \( r_{j,i} \)'s:

\[
\begin{align*}
\tilde{q}_{0,0}(+1) &= (1 - p_0) \prod_{j' \in C_0 \setminus 0} r_{j',0}(+1) \\
&= (0.69)r_{2,0}(+1) \\
&= (0.69)(0.223) = 0.154.
\end{align*}
\]

and

\[
\begin{align*}
\tilde{q}_{0,0}(-1) &= p_0 \prod_{j' \in C_0 \setminus 0} r_{j',0}(-1) \\
&= 0.31r_{2,0}(-1) \\
&= 0.31(0.777) = 0.241.
\end{align*}
\]
Example

This means

\[ q_{0,0}(+1) = \frac{0.154}{0.154 + 0.241} = 0.39 \]

and

\[ q_{0,0}(-1) = \frac{0.241}{0.154 + 0.241} = 0.61. \]

Finally, compute the APP’s:
Example

This means

\[ q_{0,0}(+1) = \frac{0.154}{0.154 + 0.241} = 0.39 \]

and

\[ q_{0,0}(-1) = \frac{0.241}{0.154 + 0.241} = 0.61. \]

Finally, compute the APP’s:

Note: \( \tilde{Q}_i(+1) = \tilde{q}_{i,j}(+1) \cdot r_{j,i}(+1) \), which means

\[ \tilde{Q}_0(+1) = \tilde{q}_{0,0}(+1) \cdot r_{0,0}(+1) = 0.154 \cdot 0.32 = 0.0493 \]

and

\[ \tilde{Q}_0(-1) = \tilde{q}_{0,0}(-1) \cdot r_{0,0}(-1) = 0.241 \cdot 0.68 = 0.164. \]

Example

This yields the APP

\[ Q_0(+1) = \frac{0.0493}{0.0493 + 0.164} = 0.23 \]

and

\[ Q_0(-1) = \frac{0.164}{0.0493 + 0.164} = 0.77. \]
Example

This yields the APP

\[ Q_0(+1) = \frac{0.0493}{0.0493 + 0.164} = 0.23 \]

and

\[ Q_0(-1) = \frac{0.164}{0.0493 + 0.164} = 0.77. \]

The other \( Q_i \)'s can be computed similarly.

Probabilistic Decoding

The most significant feature of this decoding scheme is that the computation per digit per iteration is independent of block length.
Probabilistic Decoding

The most significant feature of this decoding scheme is that the computation per digit per iteration is independent of block length.

Average number of iterations required to decode is bounded by a quantity proportional to the log of the log of the block length.