# Lecture 16

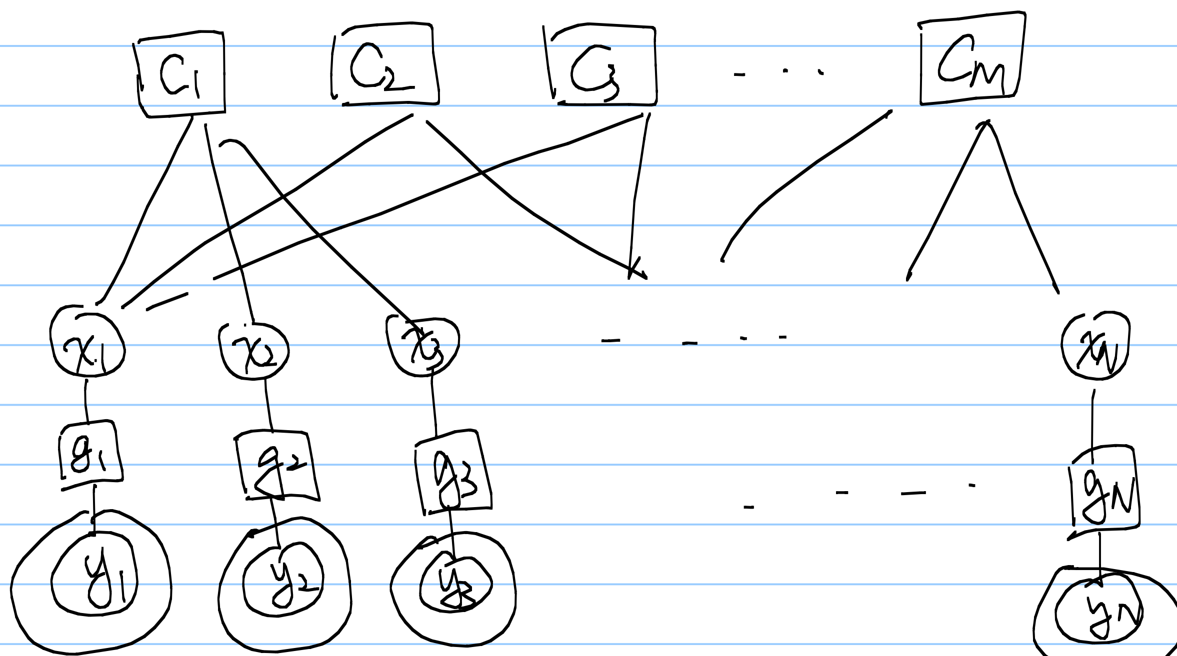
Note Title 3/5/2012

* LDPC decoder

* The factor graph representation of the likelihood function, assuming that we have independent channel

$$P_{\vec{Y}|\vec{X}}(\vec{y}|\vec{x}) = \prod_{i=1}^{N} P_{Y_i|X_i}(y_i|x_i)$$

$$\underset{\vec{x}:H\vec{x}=0}{\arg\max}\ P_{\vec{Y}|\vec{X}}(\vec{y}|\vec{x}) = \underset{\vec{x}}{\arg\max} \prod_{j=1}^{M} f_{C_j}(x_i \in \partial j) \cdot \prod_{i=1}^{N} P(y_i|x_i)$$




$$f_{C_j}(\cdot, \cdot, \dots, \cdot) = \begin{cases} 1 & \text{if the \# of 1s is even} \\ 0 & \text{otherwise} \end{cases}$$

Terminology:

① $x_i$ : the variable nodes

② $\boxed{C_j}$ : the parity-check nodes

③ degree of nodes: the number of neighbor nodes

Observation: * In the resulting factor graph, there are many cycles.

* But there is only a small # of short cycles when $n$, the codeword length is large

We can thus apply the sub-optimal min-sum or sum-product algorithms.

In particular, their parallel version.

Comparison of LDPC & turbo codes

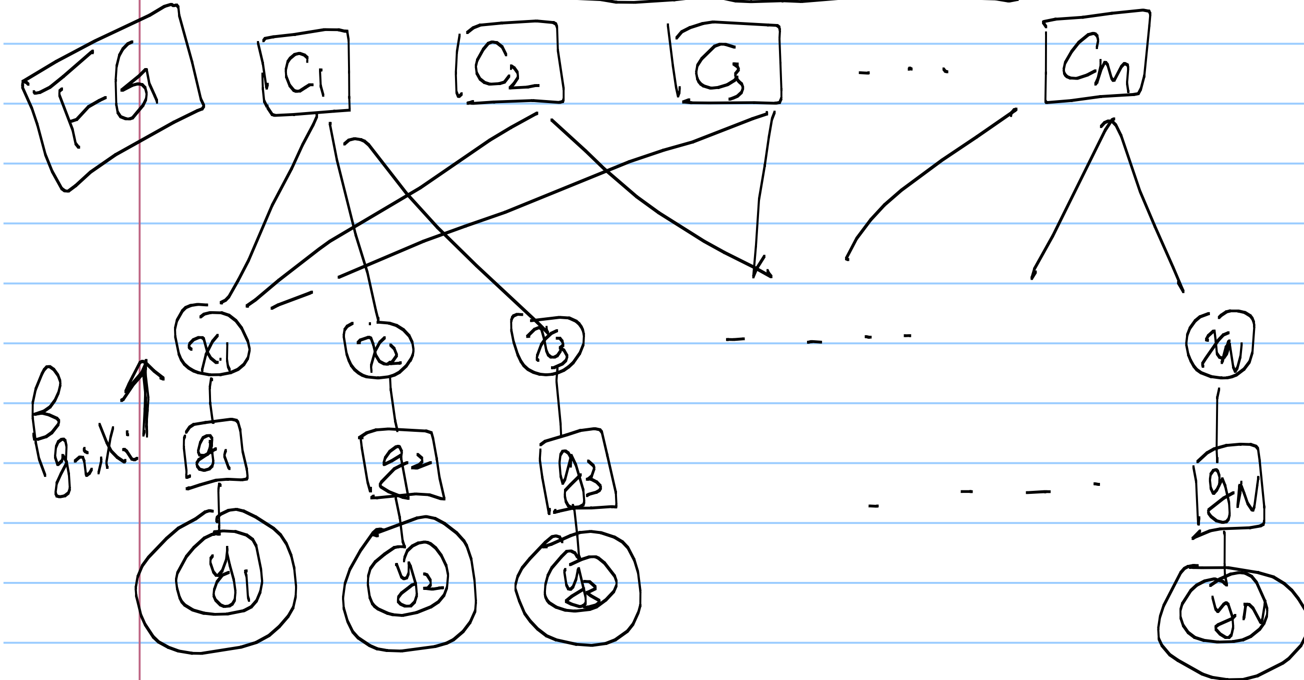
① LDPC codes are generally easier to analyze

② LDPC codes have slightly better performance
∵ With a carefully chosen H, the performance of LDPC codes can be made extremely close to the capacity ($\leq 0.01$ dB away from cap)

③ LDPC codes have potentially fully parallel structure.

④ LDPC codes were ignored for 40 yrs because it generally needs very long codeword length, 500 bits to 10k bits before the performance becomes competitive. It's hard to simulate a long code in the 60s.

# LDPC decoder




Parallel BCJR, the sum-product algorithm

Initialization:

Given the observation $y_i$, $\beta^{[0]}_{g_i,x_i}(x_i) = P(y_i | x_i)$

$\boxed{\text{time index}}$ $\beta^{[0]}_{c_j,x_i}(x_i) = 1$ for all $x_i$

Each iteration

$$\alpha^{[t]}_{x_i c_j}(x_i) = \beta^{[0]}_{g_i x_i}(x_i) \prod_{l \in \partial i \setminus j} \beta^{[t-1]}_{c_k,x_i}(x_i)$$

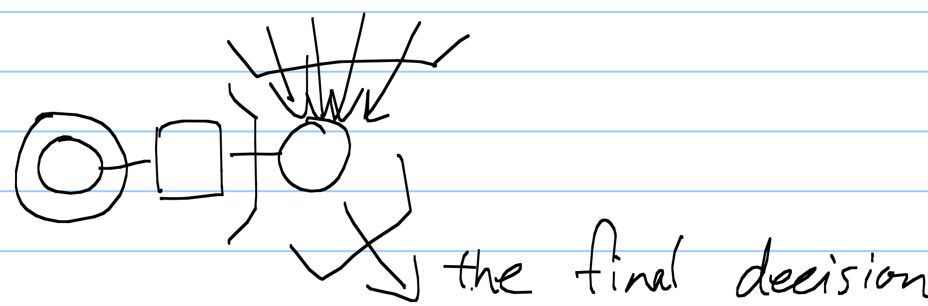
$$\beta^{[t]}_{c_j,x_i}(x_i) = \sum_{(\sum_k x_k) + x_i = 0} \prod_{k \in \partial j \setminus i} \alpha^{[t]}_{x_k,c_j}(x_k)$$

Repeat the above iterative update as if there is no need to worry about cycles

Terminate the iterative update after a fixed # of iterations, generally $t = 50-200$

Decision rule

$$\max_{x_i = 0,1} \beta^{[0]}_{g_i, x_i}(x_i) \prod_{j \in \partial i} \beta_{C_j, x_i}(x_i)$$



* An alternative description of the LDPC decoder — the log-likelihood-ratio-based decoder.

A high-level description

① We first notice ~~that~~ $\alpha_{x_i c_j}(\cdot)$ & $\beta_{c_j x_i}(\cdot)$ are functions taking only two input values $x = 0, 1$.

② In a factor graph representation, everything is relative. Only the ratio

$$\frac{\alpha_{x_i c_j}(0)}{\alpha_{x_i c_j}(1)} \quad \& \quad \frac{\beta_{c_j x_i}(0)}{\beta_{c_j x_i}(1)} \quad \text{matter}$$

when trying to distinguish $X=0$ from $X=1$

$\Rightarrow$ instead of sending a "function" $\alpha_{x_i c_j}(\cdot)$ or $\beta_{c_j x_i}(\cdot)$, we can send scalar messages

$$m_{x_i, c_j} = \log\left(\frac{\alpha_{x_i c_j}(0)}{\alpha_{x_i c_j}(1)}\right)$$

$$\& \quad m_{c_j, x_i} = \log\left(\frac{\beta_{c_j x_i}(0)}{\beta_{c_j x_i}(1)}\right)$$

---

Detailed update rules based on the LLR.

* Initialization

The channel observation LLR of the i-th bit

$$m_i^{(0)} \triangleq m_{g_i, x_i} = \log\left(\frac{\beta_{g_i x_i}(0)}{\beta_{g_i x_i}(1)}\right)$$

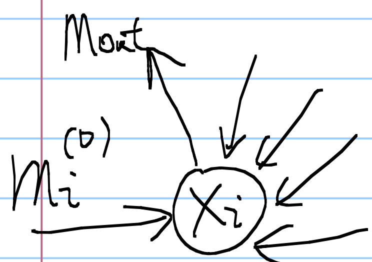
$$= \log\left(\frac{P_{Y_i|X_i}(y_i|0)}{P_{Y_i|X_i}(y_i|1)}\right)$$

* Initial Check-2-var. message

$$m^{[0]}_{c_j,x_i} = \log\left(\frac{\beta^{[0]}_{c_j x_i}(0)}{\beta^{[1]}_{c_j,x_i}(1)}\right)$$

$$= \log\left(\frac{1}{1}\right) = 0$$

* Variable node message map:




$$m^{[t]}_{x_i,c_j} \triangleq \log\left(\frac{\alpha^{[t]}_{x_i c_j}(0)}{\alpha^{[t]}_{x_i c_j}(1)}\right)$$

$$\because\ \alpha^{[t]}_{x_i c_j}(x) = \beta_{g_i x_i}(x) \cdot \prod_{l\in\partial i\setminus j} \beta^{[t-1]}_{c_l,x_i}(x)$$

$$\therefore\ = \log\left(\frac{\beta_{g_i x_i}(0)}{\beta_{g_i x_i}(1)}\right) + \sum_{l\in\partial i\setminus j} \log\left(\frac{\beta^{[t-1]}_{c_l,x_i}(0)}{\beta^{[t-1]}_{c_l,x_i}(1)}\right)$$

$$\boxed{= m_i^{(0)} + \sum_{l\in\partial i\setminus j} m^{[t-1]}_{c_l,x_i}}$$

# Check node message map

$$m^{[t]}_{c_j x_i} = \log\left(\frac{\beta^{[t]}_{c_j x_i}(0)}{\beta^{[t]}_{c_j x_i}(1)}\right)$$

Note that

$$\beta^{[t]}_{c_j x_i}(x) = \sum_{\substack{k \in \partial j \setminus i \\ \sum_k x_k = x}} \prod_{k \in \partial j \setminus i} \alpha^{[t]}_{x_k c_j}(x_k)$$

$$= \frac{1}{2}\left[\prod_{k \in \partial j \setminus i}\left(\alpha^{[t]}_{x_k, c_j}(0) + \alpha^{[t]}_{x_k, c_j}(1)\right) + (-1)^{x} \prod_{k \in \partial j \setminus i}\left(\alpha^{[t]}_{x_k, c_j}(0) - \alpha^{[t]}_{x_k, c_j}(1)\right)\right]$$

$$= \log \frac{\prod_{k \in \partial j \setminus i}\left(\frac{\alpha^{[t]}_{x_k, c_j}(0)}{\alpha^{[t]}_{x_k, c_j}(1)} + 1\right) + \prod_{k \in \partial j \setminus i}\left(\frac{\alpha^{[t]}_{x_k, c_j}(0)}{\alpha^{[t]}_{x_k, c_j}(1)} - 1\right)}{\prod_{k \in \partial j \setminus i}\left(\frac{\alpha^{[t]}_{x_k, c_j}(0)}{\alpha^{[t]}_{x_k, c_j}(1)} + 1\right) - \prod_{k \in \partial j \setminus i}\left(\frac{\alpha^{[t]}_{x_k, c_j}(0)}{\alpha^{[t]}_{x_k, c_j}(1)} - 1\right)}$$

$$= \log \frac{\prod_k (e^{m_k}+1) + \prod_k (e^{m_k}-1)}{\prod_k (e^{m_k}+1) - \prod_k (e^{m_k}-1)}$$

$$= \log \frac{1 + \prod_k \frac{e^{m_k}-1}{e^{m_k}+1}}{1 - \prod_k \frac{e^{m_k}-1}{e^{m_k}+1}}$$

By noting that

$$\tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}} = x$$

$$\tanh^{-1}(x) = \frac{1}{2} \log\left(\frac{1+t}{1-t}\right)$$

$$\Rightarrow \underline{\underline{m_{c_j x_i}^{[t]}}} = 2\tanh^{-1}\left(\prod_{k \in \partial j \setminus i} \tanh\left(\frac{m_{x_k c_j}^{[t]}}{2}\right)\right)$$

output

Equivalenty

$$\Leftrightarrow \boxed{\tanh\left(\frac{m_{c_j x_i}^{[t]}}{2}\right) = \prod_{k \in \partial j \setminus i} \tanh\left(\frac{m_{x_k, c_j}^{[t]}}{2}\right)}$$

After a fixed # of iterations, say T iterations

Final decision:

$$\max_{x_i=0,1} \beta^{[0]}_{g_i x_i}(x_i) \prod_{j\in\partial i} \beta^{[T]}_{C_j x_i}(x_i)$$

$$\Leftrightarrow \quad m_i^{[T]} = m_i^{[0]} + \sum_{j\in\partial i} m^{[T]}_{C_j, x_i} \underset{\hat{x}_i=1}{\overset{\hat{x}_i=0}{\gtrless}} 0$$

---

Summary: given a turbo / LDPC code, we have learned how to construct its factor graph representation & design the corresponding (suboptimal due to cycles) sum-product algorithm

Question: How to analyze the code performance for memoryless channels without resorting to brute-force simulations

Q: How to optimize the code structure?