ENEE626, CMSC858B, AMSC698B

Error Correcting Codes

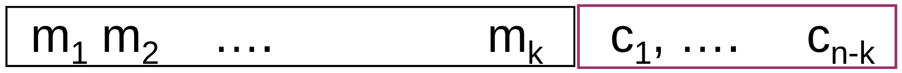
Part IV. Practical ways of approaching capacity. Iterative decoding

ENEE626 Lecture 26: Combinations of codes

Plan:
Parallel concatenations, product codes
Concatenated codes, their parameters and decoding

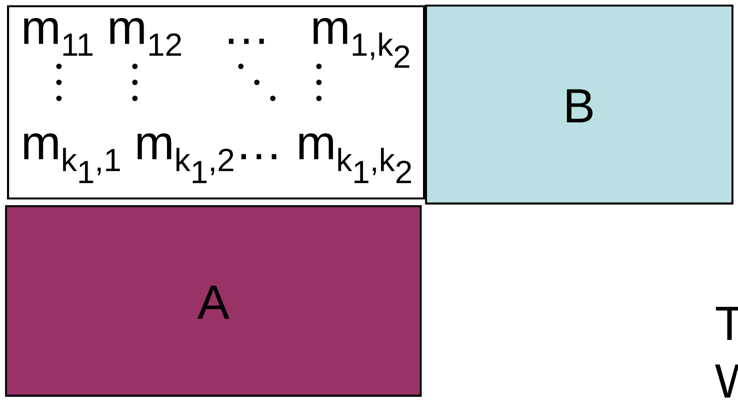
Idea of code concatenation

In all previous constructions, k message symbols are checked "in one direction."



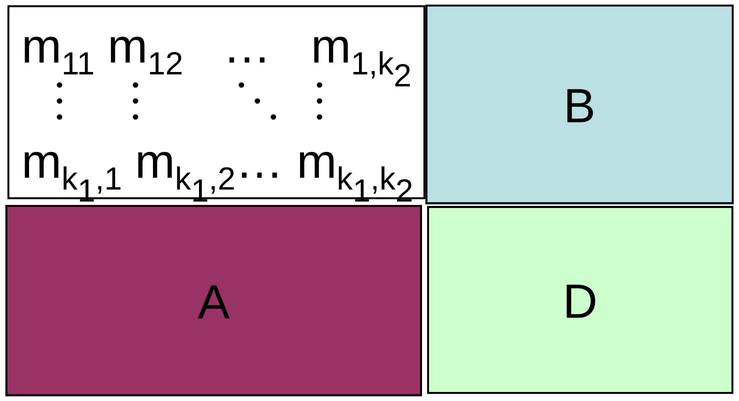
$$c_i=\sum_{j=1}^{k} h_{i,j}m_j,\ i=1,\ldots,n\text{-}k, \text{ where } H=(h_{i,j}),$$

Now let us have "2-dim checks:" first let us check the rows, then the columns of the message array


$m_{11}\ m_{12}\ \ldots\ m_{1,k_2}$
$m_{k_1,1}\ m_{k_1,2}\ldots\ m_{k_1,k_2}$
B
A

In other words, in the array on the left, let every column be a codeword in some linear code C_1 and every row a code word in some linear code C_2

This construction is called parallel concatenation. We can encode with C_2, then permute the message symbols randomly and encode with C_1. This is called turbo-encoding


$m_{11}\ m_{12}\ \dots\ m_{1,k_2}$
$m_{k_1,1}\ m_{k_1,2}\dots\ m_{k_1,k_2}$
B
A
D

Moreover, we can “check the checks”, the green rectangle D
Row-wise or column-wise?
Does not matter!

The construction shown above is called a product code

$$C=C_1\otimes C_2$$

Facts about the product code construction:

Parameters $C[n_1n_2,k_1k_2,d=d_1d_2]$
C can be decoded in stages: rows, then columns
Easy: decode up to $(d_1d_2\ -1)/4$ errors
More difficult: decode up to $(d_1d_2\ -1)/2$ errors

Concatenated codes

Variation of the product construction

An $[n=n_1n_2, k=k_1k_2]$ binary concatenated code C is encoded as follows.
Let C_2 be an RS $[n_2,k_2,d_2]$ code over $\mathbb{F}_q$, $q=2^{k_1}$

Form a q-ary message of length k_2, encode with C_2
Next map every symbol of the obtained codeword to a codeword of an $[n_1,k_1]$ binary code.

$c_{2,i} \in C_1$, $i=1,2,\dots,n_2$

This results in a binary codeword of length n.
The distance $d \geq d_1d_2$ (the product bound)

Example:

$C_2[15,13,3]$ RS code over $\mathbb{F}_{2^4}$.
$C_1[15,4,8]$ = the even-weight subcode of [15,7,5] BCH code
$C=C_1\boxtimes C_2$ $[225,52,\geq 24]$ concatenated code

Usually the code distance is above the product bound. Here is why:

To obtain a codeword of weight d_1d_2, we must be able to find a codeword of weight d_2 in the code C_2 all of whose symbols are mapped on the vectors of C_1 of weight d_1, but this may be impossible.

Facts about concatenated codes:

1. Ensemble of random concatenated codes ($C=C_2 \boxtimes C_1$, where $C_2[n_2,k_2]$ is an RS code over $\mathbb{F}_{2^{k_1}}$, C_1 a random binary $[n_1,k_1]$ code)

$n_2 \to \infty$, $k_1 \approx \log n_2 \to \infty$, $k_1/n_1=R_1$

contains codes that meet the GV bound $R=1-h(\delta)$ on the relative distance
A sufficient condition for this to hold is that $R_1 \geq \log(2(1-\delta_{GV}(R)))$

2. Concatenated codes give an example of a constructive, asymptotically good code family. (A family is called *constructive* if the complexity of code construction grows as $O(p(n))$, where $p(n)$ is a polynomial; GV codes, for instance, are not constructive. A family is called *asymptotically good* if as $n \to \infty$, both R and $\delta=d/n$ are bounded away from 0. BCH codes are not asymptotically good; RS codes over a fixed alphabet are … not asymptotic).

3. Concatenated codes can be decoded to achieve capacity of the BSC in time $O(n^2)$.

Decoding

We assume that the outer code C_2 affords a simple (polynomial in n_2) algorithm that corrects any combination of t errors and s erasures as long as $2t+s\le d_2-1$. Let $\psi_2:(\mathbb{F}_q\cup\{?\})^{n_2}\to C_2$ be this decoder mapping.

Let $\psi_1:(\mathbb{F}_2)^{n_1}\to \mathbb{F}_q\cup\{?\}$ be a decoder that outputs a word in C_1, i.e.,
(a symbol in $\mathbb{F}_q$) or an erasure ?

Let $\mathbf{y}=(\mathbf{y}_1,\dots,\mathbf{y}_{n_2})$ be the received vector, where $\mathbf{y}_i\in \mathbb{F}_2^{n_1}$, $i=1,2,\dots,n_2$.

1. Simple decoder. Complexity$=n_2(\mathrm{compl}(\psi_1))+\mathrm{compl}(\psi_2)$
   (a) Set $\mathbf{z}=(z_1,\dots,z_{n_2})$, where $z_i=\psi_1(\mathbf{y}_i)$
   (b) Decode $\mathbf{z}$ with C_2, i.e., find $\mathbf{c}_2=(\psi_2(z_1,z_2,\dots,z_{n_2}))$

Suppose that the true distance of C meets the product bound.
Suppose that ψ_1 corrects any combination of $t_1 =[(d_1-1)/2]$ errors.
The lowest-weight error vector that makes the overall decoding to fail has weight

$$(t_1+1)(\lfloor(d_2-1)/2\rfloor +1)$$

2. It is possible to improve this decoding to correct $\frac{1}{2}(d_1d_2-1)$ errors.

ENEE626 Lectures 27-28: Iterative decoding; Belief propagation algorithm

Plan:

1. Erasure correction as a procedure on Tanner graphs
2. Maximum aposteriori probability decoding
3. General belief propagation algorithm
4. Threshold of BP on the erasure channel

We begin with a motivating example.

Consider *correction of erasures* with the [7,4,3] code

$$\begin{array}{c|ccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \hline & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ \mathbf{y} = & 1 & ? & ? & 0 & 0 & ? & 0 \end{array}$$

Correct erasures:

$$x_2+x_3+x_4+x_5=0$$
$$x_1+x_3+x_4+x_6=0$$
$$x_1+x_2+x_4+x_7=0$$

$$x_2+x_3=0$$
$$1+x_3+x_6=0$$
$$1+x_2\qquad=0$$

$$D\,[x_2\ x_3\ x_6]^T=[0\ 1\ 1]^T$$

$$D=\begin{pmatrix}1&1&0\\0&1&1\\1&0&0\end{pmatrix}\quad D^{-1}=\begin{pmatrix}0&0&1\\1&0&1\\1&1&1\end{pmatrix}$$

$x_2=1;\ x_3=1;\ x_6=0$

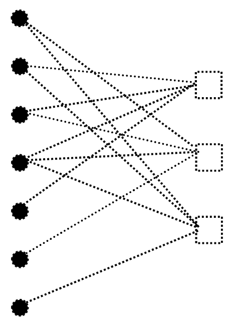
Processing time $O(n^3)$

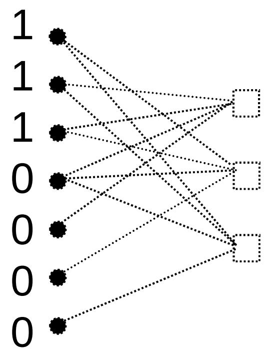
On the other hand,

$x_2=1 \ \Rightarrow x_3=1 \Rightarrow x_6=0$

Time $O(n)$

Idea: recover bits one by one





Tanner graph is a (bipartite) graph of which H is the *adjacency matrix*

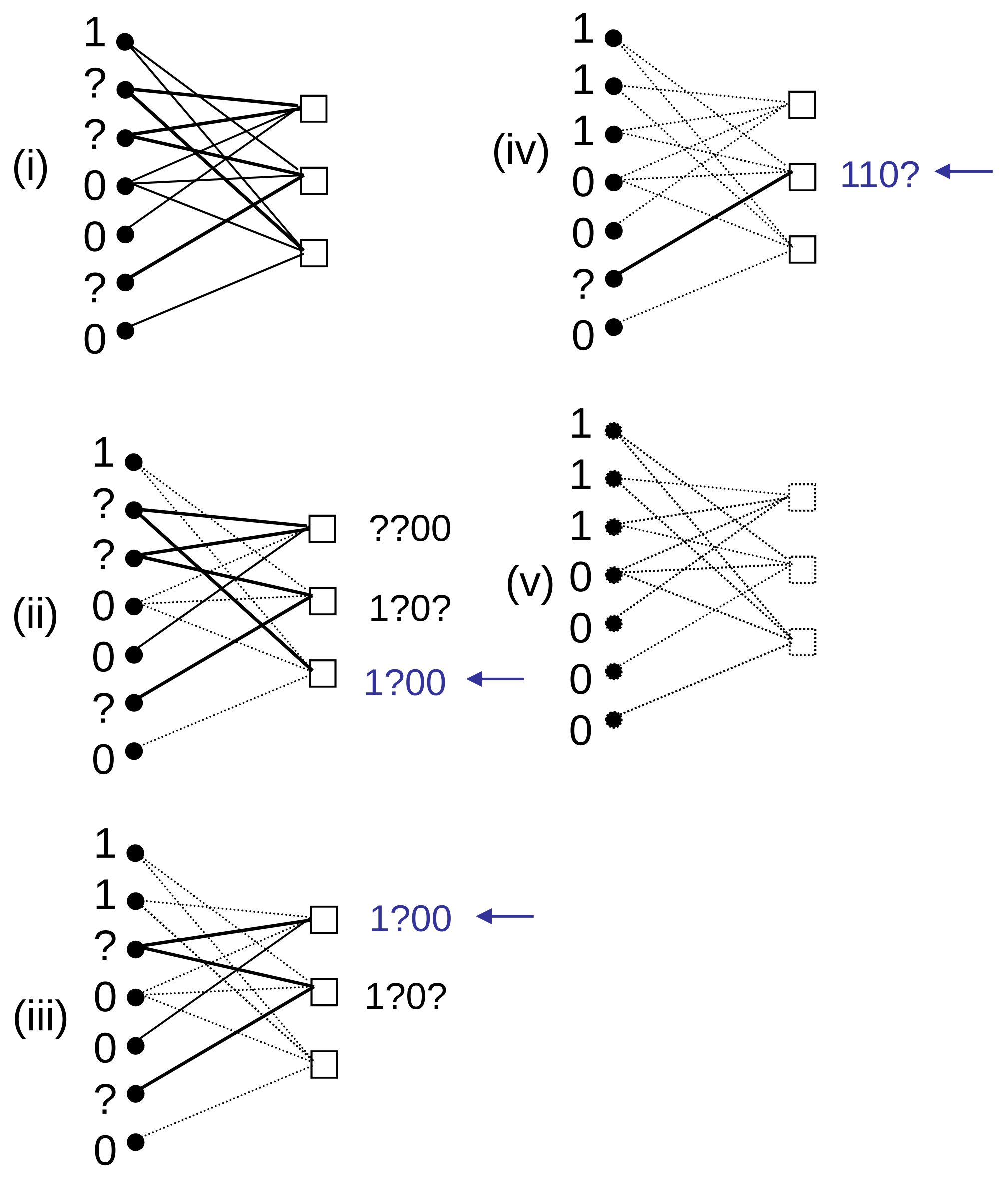
Vertices on the left, coordinates, are also called *variable nodes*
right nodes are called *check nodes*

Two vertices connected by an edge are called *adjacent*
(Edge *incident* to a vertex)

Degree of a vertex = the number of edges incident to it

A graph is called left/right *regular* if the degree of all left/right nodes is the same. Example: a (3,6) graph: all left degrees are 3, all right are 6

Let us represent sequential processing as a procedure on the Tanner graph



Algorithm:

variable-to-check

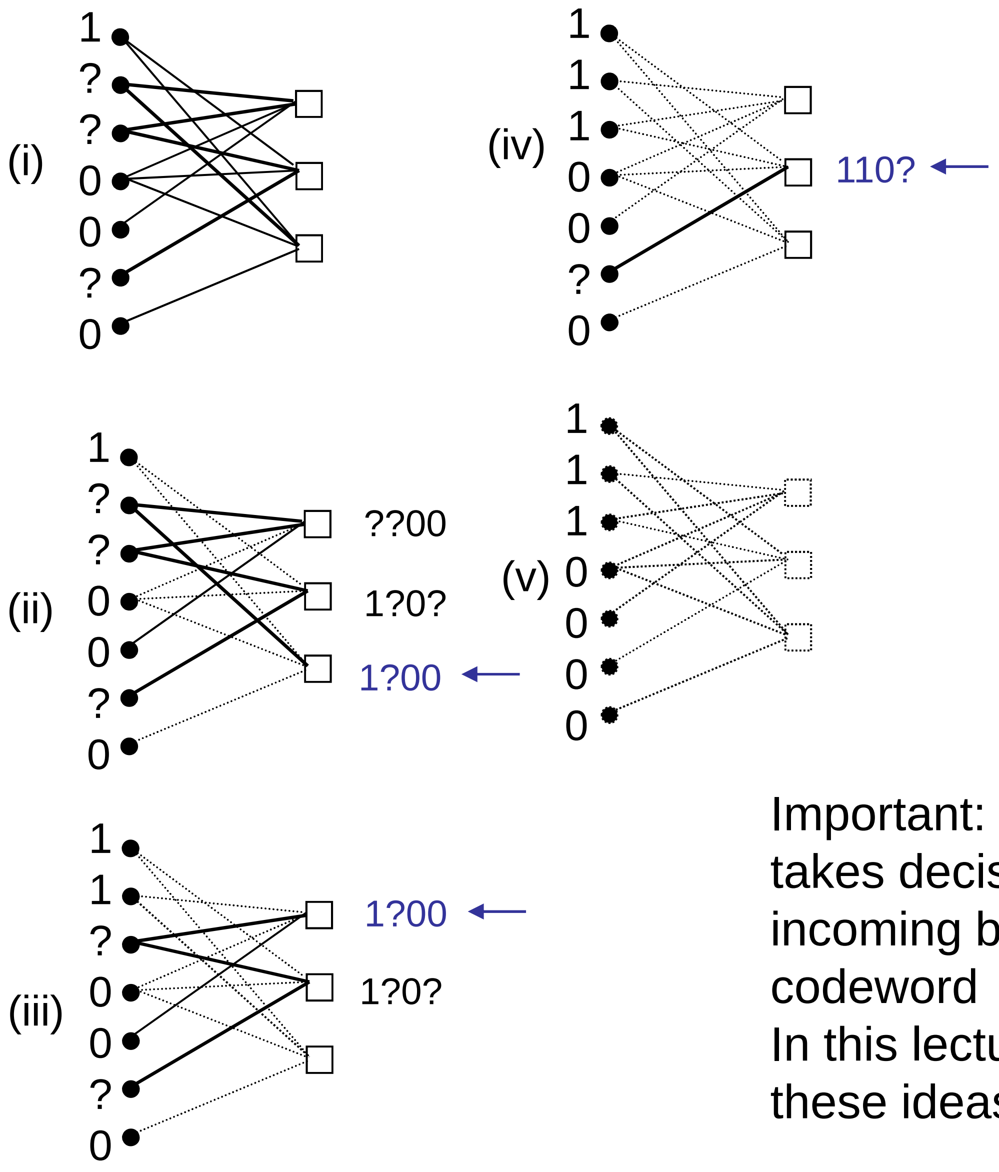
send the nonerased bits to all the check nodes; erase the edges

check-to-variable

find check node(s) of deg 1 recover the unknown bits, send them to their var.nodes, erase the edges

If no edges are left, decoding complete. If at some stage all the check nodes have deg$\geq$ 2, fail

Let us represent sequential processing as a procedure on the Tanner graph



Algorithm:

variable-to-check

send the nonerased bits to all the check nodes; erase the edges

check-to-variable

find check node(s) of deg 1 recover the unknown bits, send them to their var.nodes, erase the edges

If no edges are left, decoding complete. If at some stage all the check nodes have deg$\geq$ 2, fail

Important: *local processing*. The node takes decision based on just the incoming bits, without seeing the entire codeword

In this lecture we develop and generalize these ideas

Bit-MAP decoding

Transmit with a code $C \subset \{0,1\}^n$ over a memoryless channel given by $P(a|y)$, a=0,1. If $Y=\{y\}$ is a continuous set, $P(a|y)=p_{A|Y}(.|.)$.

Earlier we discussed optimal decoding for minimizing the block error rate: given the received vector **y** decode to $c=\text{argmax}_{\mathbf{c}' \in C} P(\mathbf{c}'|\mathbf{y})$, or

$$\mathbf{c}=\text{argmax}_{\mathbf{c}' \in C} P(\mathbf{y}|\mathbf{c}'), \quad P(\mathbf{y}|\mathbf{c}')=\prod_{i=1}^{n} P(y_i|c_i')$$

(*max likelihood decoding*)

More fitting to the present context is bit-optimal decoding: decode the ith transmitted bit as

$$c_i=\text{argmax}_{a\in\{0,1\}} \sum_{\mathbf{c}\in C,\, c_i=a} P(\mathbf{c}|\mathbf{y})$$

where

$$P(\mathbf{c}|\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{c})P(\mathbf{c})}{P(\mathbf{y})} = \frac{P(\mathbf{y}|\mathbf{c})P(\mathbf{c})}{\sum_{\mathbf{c}'\in C} P(\mathbf{y}|\mathbf{c}')P(\mathbf{c}')}$$

This rule maximizes the probability of correct decoding of bits, although the decoded sequence may not be a codeword.

The belief propagation algorithm

Decoding is performed by local processing in the form of iteratively computing estimates of the likelihood of the value of each received bit.

Decoding iterations consist in computing the log-likelihood ratio at a variable or check node and sending them to all the check/variable nodes connected to it. In odd-numbered iterations the processing is done at variable nodes, in even iterations -- at check nodes (message passing algorithm).

We assume that the messages are in the form of Log-Likelihood Ratio $\log(P(v_i=0)/P(v_i=1))$, where v_i is the ith bit.

Local processing:

Let d_v, d_c be the degree of a variable (check) node v (c), $l \geq 1$.

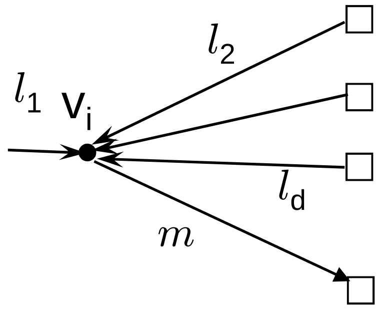
Iteration $2l$-1: the message that goes from v to the check nodes c_i, i=1,..,d connected to it is computed from the messages received from the check nodes

$$c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_{d_c}$$

in iteration $2l$ and the information (LLR) received from the channel.

Iteration $2l$: the message from a check node c to v_i is based on the messages received from $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_{d_v}$ in iteration $2l$-1.

Generating messages at variable nodes:



$$m = \log \frac{P(x=0|y_1,\ldots,y_d)}{P(x=1|y_1,\ldots,y_d)}$$

where y_1,…,y_d are the noisy observations of the bit x transmitted at the position v_i

$$P(x=0|y_1,\ldots,y_d) = \frac{P(x=0)\prod_{i=1}^{d} \Pr(y_i|x=0)}{P(y_1,\ldots,y_d)}$$

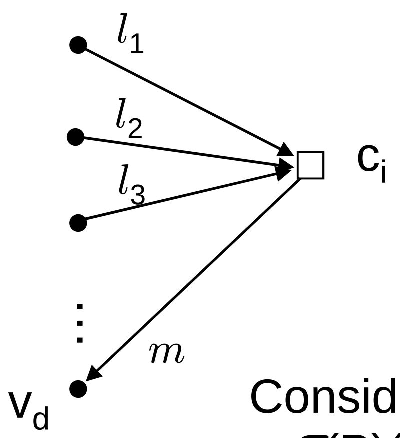
We assume that the observations y_1,…,y_d are independent

$$m = \log \frac{1/2\prod_{i=1}^{d} P(y_i|x=0)}{1/2\prod_{i=1}^{d} P(y_i|x=1)} = \sum_{i=1}^{d} l_i \text{ where } l_i = \log \frac{P(y_i|x=0)}{P(y_i|x=1)}$$ is the LLR

To a check node c_j connected to it, the node v_i sends a sum of all the incoming LLRs *except* l_j plus the LLR of the channel observation

Generating messages at check nodes:

message sent to the node v_d is $m=\log(P(v_d=0)/P(v_d=1))$



$P(v_d=0)=P(v_d=0 \mid \sum x_i=0, y_1,\ldots,y_d)$
$=P(x_1+\ldots+x_{d-1}=0 \mid y_1,\ldots,y_d)$

$P(v_d=1)= P(x_1+\ldots+x_{d-1}=1 \mid y_1,\ldots,y_d)$

Consider the Fourier transform of the pmf $P(v_d)$:

$\mathcal{F}(P)(v_d=0)=P(v_d=0)+P(v_d=1)=1$

$\mathcal{F}(P)(v_d=1)=P(v_d=0)-P(v_d=1)=\prod_{i=1}^{d-1}(P_i(0)-P_i(1))$, where

$P_i(a)=P(x_i=a \mid y_i)$

Proposition 27.1: $m = \log \dfrac{1+\prod_{i=1}^{d-1} \tanh l_i/2}{1-\prod_{i=1}^{d-1} \tanh l_i/2}$

Proof:

Let $m = \log \dfrac{P(v_d=0)}{P(v_d=1)}$

Observe that

$m = \log \frac{1+z}{1-z} \Leftrightarrow \tanh \frac{m}{2} = z$

So we compute

$$\tanh \frac{m}{2} = \frac{e^m-1}{e^m+1} = P(v_d=0)-P(v_d=1)$$

$$= \mathcal{F}(P)(v_d=1) = \prod_{i=1}^{d-1} (P_i(0)-P_i(1))$$

$$= \prod_{i=1}^{d-1} \tanh \frac{l_i}{2}$$

▲

Belief propagation algorithm:

Received **y** from a binary-input memoryless channel.
Compute $(l_1, l_2, \ldots, l_N)$

Odd iterations: Variable-to-check

$\rightarrow c_j$

$$m = \sum_{\substack{i=1 \\ i \neq j}}^{d+1} l_i$$

Even iterations: Check-to-variable

$v_j \leftarrow$

$$m = \log \frac{1 + \prod_{\substack{i=1 \\ i \neq j}}^{d} \tanh l_i/2}{1 - \prod_{\substack{i=1 \\ i \neq j}}^{d} \tanh l_i/2}$$

Perform a prescribed number of iterations, or until found a codeword

Back to the erasure channel:

let us see what the BP algorithm looks like in this case

LLRs: $\{+\infty,0,-\infty\}$ corresponding to the received symbols 0,?,1.

The BP variable node processing rule: outgoing message equals

m=0 if all the incoming l_i=0

m=a$\in\{\pm\infty\}$ if one of the incoming l_i=a

(and then all the nonzero l_i's equal a)

Check node processing rule:

tanh z={1,0,-1} $\Leftrightarrow$ z=$\{\infty,0,-\infty\}$

m=log(1+$\prod_i$ sgn l_i)/(1-$\prod_i$ sgn l_i)

translates to an outgoing message

m=0 if one of the l_i's is 0 (one of the incoming var. nodes is erased)

$\left.\begin{array}{l}\text{m}=-\infty \text{ if an odd \# of the } l_i\text{'s equal to } -\infty \text{ (i.e., symbols 1)} \\ \text{m}=\infty \text{ if an even \# of the } l_i\text{'s equal to } -\infty\end{array}\right\}$ all incoming not erased

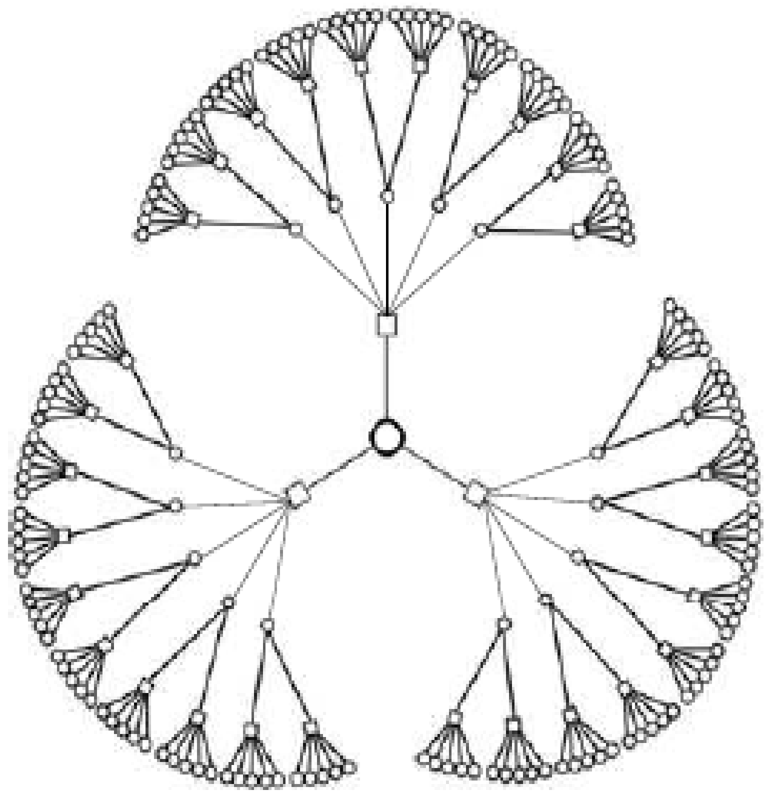
Questions:

When does this procedure converge?

If it converges, what is the probability that it converges to the bit-MAP decision?

Big picture: For a given BEC(p) it is possible to construct an ensemble of random codes (graphs) of length N and rate $R \geq 1-p-O(1/N)$ such that the average code from this ensemble will correct all erasures under message passing decoder with high probability.

Computation graph for a (3,6) regular code



Theorem 27.2: If the computation graph is a tree then the BP algorithm converges in a finite number of steps. The result is bit-MAP decoding.

Density evolution equations for the erasure channel

We will analyze the expected behavior of the message passing algorithm on the BEC(p) under the “tree assumption”.

Definition 27.1. Degree distribution. Let d be the maximum variable node (left) degree of the Tanner graph, μ be the same for the check node (right) degree.

Let λ_i=proportion of edges connected to variable nodes of deg. i

ρ_i=proportion of edges connected to check nodes of deg. i

$\lambda(X)=\sum_{i=1}^{d} \lambda_i X^{i-1}$, $\rho(X)=\sum_{i=1}^{\mu} \rho_i X^{i-1}$ are called variable (check) node degree distribution.

Examples: [7,4,3] code: $\lambda(X)=(1/4)+(1/2)X+(1/4)X^2$, $\rho(X)=X^3$

(3,6) regular code from the previous slide: (X^2,X^5)

Some formulas

$\lambda(1)=\rho(1)=1$

$\sum_i(\lambda_i/i)=\int\lambda=V/E$

E=# edges, V=#{var.nodes}=N

$a=E/V=1/\int\lambda$

average left degree

Design rate $R(\lambda,\rho)=1-$(#check nodes)/V

$$V = E\int \lambda; \quad \sharp(\text{checks}) = E\int \rho$$

$$R(\lambda,\rho) = 1 - \frac{\int \rho}{\int \lambda}$$

Let $P_l(p)$ = probability that a bit is erased after l iterations

Theorem 27.3: Let $x_0=p$, $x_l:=p\lambda(1-\rho(1-x_{l-1}))$. Then

$$P_l(p)=x_l \quad (l \text{ odd}).$$

Proof: Induction. For $l=1$, messages var-check are erasures with prob. p.
Assume that $P_i(p)=x_i$, $0\leq i\leq l$.
(a) first look at the case of l even: the iteration is check-to-variable.
a message sent along the edge is erasure if one or more of the messages that enter the check node in (l-1)st iteration are erasures. By the hypothesis, its prob. = x_{l-1}, and the incoming messages are independent (tree assumption). Hence, if the degree of the check node is i,

$$P_l(p)=1-(1-x_{l-1})^{i-1}$$

Prob.{edge is connected to a check node of deg i}=ρ_i, so by total prob. theorem,

$$P_l(p)=\sum_i \rho_i(1-(1-x_{l-1})^{i-1})=1-\rho(1-x_{l-1}).$$

(b) l odd. The event that the bit is erased is obtained when the received channel bit is ? and all the incoming messages are ?.

$$P_l(p)=\sum_i \lambda_i\, p\, \{1-\rho(1-x_{l-1})\}^{i-1}= p\, \lambda(1-\rho(1-x_{l-1})).$$ ▲

The system $x_l = p\lambda(1-\rho(1-x_{l-1}))$, $l \geq 1$ is called

density evolution equations

because it shows how the bit erasure prob. changes with iterations

Let us assume that $\lambda_j = \rho_j = 0$, $j=0,1$.

Proposition 27.4: Let $p_1 < p$, then $(\lim_{l\to\infty} P_l(p)=0) \Rightarrow (\lim_{l\to\infty} P_l(p_1)=0)$.

Proof: for $x \in [0,1]$ $(d/dp)(p\lambda(1-\rho(1-x))) \geq 0$,

$(d/dx)(p\lambda(1-\rho(1-x)) = p\rho'(1-x)\lambda'(1-\rho(1-x)) \geq 0$

Then

$x_{l+1}(p_1) = p_1\lambda(1-\rho(1-x_l(p_1)) \leq p_1(1-\rho(1-x_l(p))) \leq p(1-\rho(1-x_l(p))) = x_{l+1}(p)$ ▲

Definition 27.2: $p^*(\lambda,\rho) := \sup \{p \in [0,1] : \lim P_l(p)=0\}$

is called a threshold associated with the degree distribution pair λ,ρ.

By the above proposition, p^* is well defined.

Example: $p^*(X^2, X^5) \approx 0.43$ ((3,6) regular ensemble)

Determination of the threshold

The threshold $p^*(\lambda,\rho)$ is related to the fixed points of density evolution.
Note that 0 is one such fixed point.
Let $g(p,z) \triangleq p\lambda(1-\rho(1-z))$

Proposition 27.5: $p^*(\lambda,\rho)=\sup\{p\in[0,1]: x=g(p,x)$ has no roots in $(0,1]\}$
$=\inf\{p\in[0,1]: x=g(p,x)$ has a root in $(0,1]\}$

Lemma 27.6: x_l is a monotone (up or down) sequence that converges to the nearest root of $x=g(p,x)$.

Proof: If $x_0=p=0$ then $x_l=0$ for all l.
If $x_l \geq x_{l-1}$, then $x_{l+1}=g(p,x_l))>g(p,x_{l-1})=x_l$. Likewise if $x_l \leq x_{l-1}$ then $x_{l+1} \leq x_l$.
Therefore, there is a limit, i.e., a fixed point x_∞. We need to show that this is the nearest root to x_0. Let z be a fixed point such that $x_l(x_0) \leq z$, then

$$x_{l+1}(x_0)=g(p,x_l(x_0)) \leq g(p,z)=z,$$
$$\text{so } x_l \to z \qquad ▲$$

Proof of proposition: Clearly, $0 \leq g(p,x) \leq p$, so if $x=g(p,x)$ then $0 \leq x \leq p$.
Let z be the largest solution of $x=g(p,x)$. Suppose that $\lim x_l(p)=z$. If $z=0$ then $p<p^*$. If $z>0$ then $p>p^*$. ▲

Stability condition

$g(p,x)=p\lambda(\rho'(1)x)+O(x^2)=p\lambda'(0)\rho'(1)x+O(x^2)$

Thus $x_{l+1}=p\lambda'(0)\rho'(1)x_l(x_0)+O(x_l^2(x_0))$. This implies

Theorem 27.7: Given (λ,ρ); $p=x_0$.
If $p\lambda'(0)\rho'(1)>1$ then $\lim x_l(x_0)$ is bounded away from 0 for all $x_0 \in (0,1)$
(fixed point at 0 is unstable)
If $p\lambda'(0)\rho'(1)<1$ then $\lim x_l(x_0)=0$ for all x_0 in a (finite) small neighborhood of $x_0=0$.

This implies that $p^*(\lambda,\rho)\leq 1/(\lambda'(0)\rho'(1))$

Capacity-achieving distributions

Definition 27.2: Let $p^*(\lambda,\rho)$ be the threshold of the degree distribution (λ,ρ), let $R=R(\lambda,\rho)=1-\int\rho/\int\lambda$ be the design rate. The quantity $\delta(\lambda,\rho)$ is called *gap to capacity* of the degree dist. if

$$R(\lambda,\rho)=(1-\delta(\lambda,\rho))(1-p^*(\lambda,\rho))$$

Main Definition 27.3: A sequence $\{(\lambda^{(n)},\rho^{(n)})\}_{n\geq 1}$ of degree distributions is said to achieve capacity of BEC(p) if

$$\lim_{n\to\infty} R(\lambda^{(n)},\rho^{(n)})=1-p, \lim_{n\to\infty}\delta(\lambda^{(n)},\rho^{(n)})=0$$

Example: Let M be a natural number, $\alpha=1/M$. Let $\lambda_\alpha^{(n)}$ be the (normalized) nth partial sum of the Taylor series around 0 of

$$\lambda_\alpha(x)=1-(1-x)^\alpha$$

and let $\rho_\alpha^{(n)}=x^M$, $n=1,2\ldots$.

The sequence $\{(\lambda_\alpha^{(n)},\rho_\alpha^{(n)})\}$ is capacity-achieving.

Proof (of achieving capacity):

$$\lambda_\alpha^n(x) = \sum_{i=1}^{n-1} (-1)^{i-1} \binom{\alpha}{i} x^i$$ (since $\alpha<1$, the coefficients $\lambda_i^{(n)} \geq 0$)

In the next 2 formulas we use

$$\binom{n-1}{m} = \left(\binom{n}{m} - \binom{n-1}{m-1}\right) = \left(\binom{n}{m} - \binom{n}{m-1} + \binom{n-1}{m-2}\right) = \cdots = \sum_{i=0}^{m} (-1)^{m-i} \binom{n}{i}$$

$$\lambda_\alpha^{(n)}(1) = \sum_{i=1}^{n-1} (-1)^{i-1} \binom{\alpha}{i} = 1 - (-1)^{n-1} \frac{n}{\alpha} \binom{\alpha}{n}$$

$$\int_0^1 \lambda_\alpha^{(n)}(x) dx = \sum_{i=1}^{n-1} (-1)^{i-1} \binom{\alpha}{i} \int_0^1 x^i dx = \sum_{i=1}^{n-1} \frac{(-1)^{i-1}}{i+1} \binom{\alpha}{i} = \frac{\alpha - (-1)^{n-1} \binom{\alpha}{n}}{\alpha + 1}$$

$$R(\lambda_\alpha^{(n)}, \rho_\alpha^{(n)}) = 1 - \frac{\int \rho^{(n)}}{\int \lambda_\alpha^{(n)}} = 1 - \left(1 - \frac{(-1)^{n-1} n}{\alpha} \binom{\alpha}{n}\right) \frac{\alpha}{\alpha - (-1)^{n-1} \binom{\alpha}{n}}$$

$$= \frac{D(1 - 1/n)}{1 - D/n}$$ where $D = (-1)^{n-1} \frac{n}{\alpha} \binom{\alpha}{n}$

Likewise, $\delta(\lambda_\alpha^{(n)},\rho_\alpha^{(n)})\leq(1-D)/(n-D)$

We have

$\ln D=\ln(1-\alpha)+G(\alpha)-G(0)$, where $G(x)=\sum_{i=2}^{n-1}\ln(i-x)$

$G'(0)=-\sum_{i=2}^{n-1}(1/i)\approx 1-\ln(n-1)$

For small α we have $G(\alpha)-G(0)=\alpha\, G'(0)+\frac{1}{2}\alpha^2 G''(0)+o(\alpha^2)$

$$\leq \alpha - \alpha \ln(n-1) + O(\alpha^2)$$

$\ln D=(-\alpha-\frac{1}{2}\alpha^2-\ldots) + \alpha - \alpha\ln(n-1) + O(\alpha^2)=-\alpha\ln(n-1)+O(\alpha^2)$

Hence by choosing $\alpha=(\ln(1-p))/(-\ln(n-1))$ we can make $D\approx 1-p$.

This gives $R(\lambda_\alpha^{(n)},\rho_\alpha^{(n)})=(1-p)(1-1/n)/(1-(1-p)/n)\to 1-p$

$\delta(\lambda_\alpha^{(n)},\rho_\alpha^{(n)})\to 0$ ▲

Another example ("tornado codes") : $\lambda_\alpha(x)=-(1/\alpha)\ln(1-x)$, $\rho_\alpha^{(n)}(x)=e^{\alpha(x-1)}$

More books on coding:

General:

W.C. Huffman and V. Pless, Fundamentals of error-correcting codes, 2002

R. Blahut, Algebraic codes for data transmission, 1999

Mathematical:

J. van Lint, Introduction to error-correcting codes, 3rd ed., 1999.

Engineering:

D. Costello and S. Lin, Error control coding, 2nd ed., 2003

RS codes and algebraic geometry codes:

R. Blahut, Algebraic codes on lines, planes, curves, 2007

LDPC codes:

T. Richardson and R. Urbanke, Modern coding theory, 2007