united nations
educational, scientific
and cultural
organization

international atomic
energy agency

the
abdus salam
international centre for theoretical physics

SMR.1347 - 3

WORKSHOP ON STATISTICAL PHYSICS AND CAPACITY-APPROACHING CODES

CONNECTIONS BETWEEN STATISTICAL PHYSICS, INFERENCE AND ERROR-CORRECTING CODES

J. YEDIDIA
Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge, MA 02139, U.S.A.

Tutorial

Connections between Statistical Physics, Inference and Error-Correcting Codes

Jonathan Yedidia

Mitsubishi Electric Research Laboratories

Thanks to Bill Freeman, Yair Weiss, and Jean-Philippe Bouchaud

Three Apologies

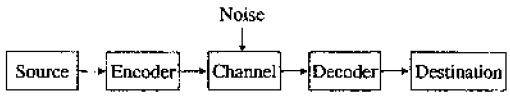
- Very basic material for an advanced audience
  - But, by the end I'll be discussing "Replicas combined with GBP to compute thresholds of codes."
- No citations to the literature
  - But I'll try to fix that in a written version
- Misleading: I emphasize codes with tiny blocklengths or infinite blocklengths, but what about intermediate blocklengths?
  - But see my research paper.

Outline

- Basics of Error-Correcting Codes
- Decoding codes: Belief Propagation
- Free energies: Bethe, Kikuchi, etc.
- Analyzing codes: Density Evolution, Replicas
- Using Replicas + BP to compute properties of codes

Errir-Correcting Cofes

01100111001000011101010011111 0 ⟶ ???



A Parity check

Parity check bit

010 1

Information bits

Problem: can only *detect* errors

1011 ⟶ ????

Code-words
0000
0011
0101
0110
1001
1010
1100
1111

Parity check code

Combine multiple parity checks to make a code

Information bits { 0 1 0

1st parity check: bits 1,2: 0+1=1 ⟶ 1

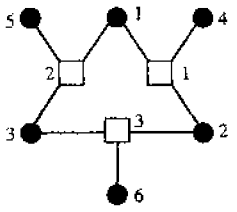
2nd parity check: bits 1,3: 0+0=0 ⟶ 0

3rd parity check: bits 2,3: 1+0=1 ⟶ 1

Parity Check Matrix

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

Tanner Graph



Codewords:
000000
001011
010101
011110
100110
101101
110011
111000

Blocklength: $N=6$

# of codewords: $2^k = 8$

Rate: $R=k/N = 3/6 = 1/2$

Minimum distance: $d=3$

Can now correct single errors

Codewords:
000000
001011
010101
011110
100110
101101
110011
111000

000011 ⟶ 001011

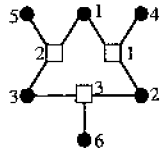
$d=3$, so each word has some nearest codeword

In general, we want a code with the maximal d, given N and k.

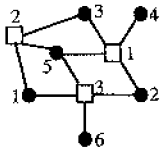
Different Representations of the Same Code

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

Information bits | parity bits



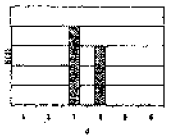
$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$



Parity check codes are *linear*

Codewords:

000000 001011
010101 011110
100110 101101
110011 111000

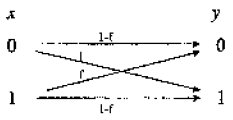


- Add any two codewords--get another codeword.
- All codewords are typical; they have the same N(d) histogram.
- Can always assume all-zeros codeword is transmitted.
- Refer to other codewords in terms of "weight".

Three Noise Channels

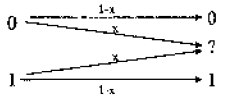
Binary Symmetric Channel (BSC)

010 ⟶ 000



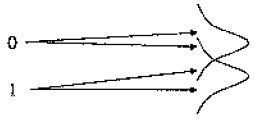
Binary Erasure Channel (BEC)

010 ⟶ 0?0



Additive White Gaussian Noise Channel (AWGNC)

010 ⟶ (1.1,0.2,.8)



Maximum Likelihood Decoding

Receive 000011 *in BSC*

Decode to ⟶

likelihood (codeword) = $\prod_i p(x_i|y_i)$

Codeword	likelihood
000000	$f^2(1-f)^4$
001011	$f(1-f)^5$
010101	$f^3(1-f)^3$
011110	$f^4(1-f)^2$
100110	$f^3(1-f)^3$
101101	$f^4(1-f)^2$
110011	$f^2(1-f)^4$
111000	$f^3(1-f)$

Obviously intractable when the number of codewords is exponentially large.

Bit-by-bit Bayesian Decoding

Receive 000011 *in BSC*

$$p(x_1 = 0) = \frac{1}{Z}\left[f(1-f)^5 + f^2(1-f)^4 + f^3(1-f)^3 + f^4(1-f)^2\right]$$

codeword	*probability*
000000	$f^3(1-f)^3/Z$
001011	$f(1-f)^5/Z$
010101	$f^3(1-f)^3/Z$
011110	$f^4(1-f)^2/Z$
100110	$f^3(1-f)^3/Z$
101101	$f^4(1-f)^2/Z$
110011	$f^2(1-f)^4/Z$
111000	$f^3(1-f)^3/Z$

$$Z = \sum_{codeword} likelihood(codeword)$$

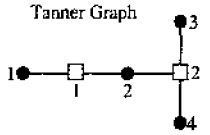
$$Z = \sum_{word} likelihood(word)$$

Inference notation

Parity check matrix

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

Tanner Graph



Probability function

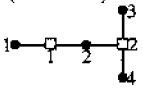
$$p(x_1, x_2, x_3, x_4 \mid y_1, y_2, y_3, y_4) = \frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{234}(x_2, x_3, x_4)p(x_1 \mid y_1)p(x_2 \mid y_2)p(x_3 \mid y_3)p(x_4 \mid y_4)$$

In general: $$p(\{x\} \mid \{y\}) = \prod_{\mu} \psi_{\mu}(\{x_{\mu}\}) \prod_{i} p(x_i \mid y_i)$$

Decoding: compute belief (marginal probability) at each node, and choose bit value with highest belief.

Physics notation $S_i = (-1)^{x_i}$ *"spins"* ± 1

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



$$p(S_1, S_2, S_3, S_4) = \lim_{T \to 0} e^{-H}$$

$$H = -\frac{1}{T}S_1S_2 - \frac{1}{T}S_2S_3S_4 - h_1S_1 - h_2S_2 - h_3S_3 - h_4S_4$$

In General: $$p(\{S\}) = \lim_{T \to 0} e^{-H}$$

$$H = -\frac{1}{T}\sum_{\mu}\prod_{i_{\mu}} S_{i_{\mu}} - \sum_i h_i S_i$$

"local magnetic fields"

Parity check codes as magnetic spin systems

- Parity checks are "hard" multi-spin interactions.
- ~~Transmitted~~ Received block gives random, biased magnetic fields:

  For BSC: $$p(h_i) = (1-f)\delta(h_i - h_0) + f\delta(h_i + h_0)$$

  $$h_0 = \ln\left(\sqrt{(1-f)/f}\right)$$

- Different quenched magnetic field for each block (sample).
- Ferromagnetic state for strong bias in field (low noise).
- Paramagnetic state for weak bias in field (high noise).
- *Decoding*: compute local magnetizations, and threshold.

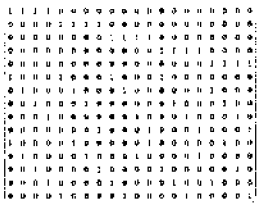
"Hidden" nodes

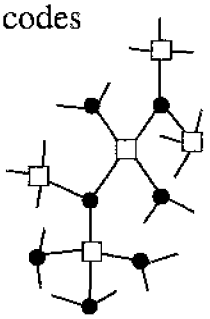
- Many ~~very~~ good codes (Kanter-Saad codes, turbocodes, Mackay-Neal codes, Repeat-Accumulate codes) have some nodes that are not transmitted across the channel, but help define the codewords.
- These nodes can be considered to have "no local evidence" or "zero local magnetic field."

Gallager 1962

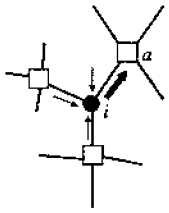
- Gallager Codes
- Decoding algorithms, including "probabilistic" decoding algorithm equivalent to belief propagation
- Analysis methods, including the density evolution approach

Gallager codes

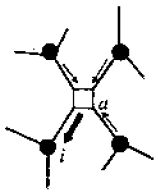




Belief Propagation Decoding in BEC



m_{ia} If any incoming message to variable node i is a '0' or '1', send "*I am a 0,*" or "*I am a 1,*" otherwise send "*I don't know.*"



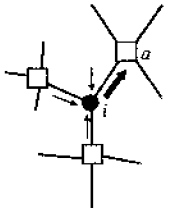
m_{ai} If *all* incoming messages to check node a are '0' or '1', send "*You are (sum modulo 2)*" otherwise send "*Don't know.*"

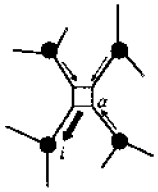
BP decoding for BSC or AWGNC

- Unlike BEC, never certain about x_i
- Therefore, messages are probabilities: "*I am a '0' with 72% probability.*" or "*You are a '0' with 58% probability.*"
- Use log-likelihood ratios:

$$m_{ia} \propto \begin{pmatrix} e^{v_{ia}/2} \\ e^{-v_{ia}/2} \end{pmatrix} \leftarrow \textit{Likelihood I am a '0'} \qquad m_{ai} \propto \begin{pmatrix} e^{u_{ai}/2} \\ e^{-u_{ai}/2} \end{pmatrix} \leftarrow \textit{Likelihood you are a '0'}$$

BP decoding equations





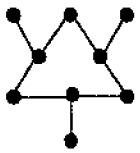
$$v_{in} = u_i^o + \sum_{b \neq n} u_{bi} \qquad \tanh(u_{ai}) = \prod_{j \neq i} \tanh(v_{ja})$$

$u_i^o = 0$ *for hidden nodes*

Pairwise Markov Random Fields

$$p(\{x\}) = \frac{1}{Z} \prod_{(ij)} \psi_{ij}(x_i, x_j) \prod_i \psi_i(x_i)$$

$$H = -\sum_{\langle ij \rangle} J_{ij}(x_i, x_j) - \sum_i h_i(x_i)$$



Trick: Parity check codes can be written as a pairwise MRF:

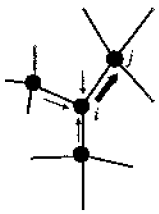
Introduce nodes x_a that have e.g. 8 states like

000, 001, 010, 011, 100, 101, 110, 111

and "hard" local evidence terms ψ_a that pick out even parity states.

BP for Markov Random Fields

All messages are like: *"You should be in your states with the following probabilities: (70%, 20%, 5%, 5%)."*



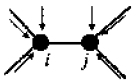
$$m_{ij}(x_j) = \sum_{x_i} \psi_{ij}(x_i, x_j) \psi_i(x_i) \prod_{k \neq j} m_{ki}(x_i)$$

This "sum-product" rule is mathematically equivalent to the u_{ai} and v_{ia} rules.

Belief equations

Iterate message equations to convergence, and then:

$$b(x_i) \propto \psi_i(x_i) \prod_k m_{ki}(x_i)$$



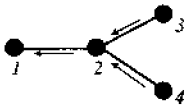
$$b(x_i, x_j) \propto \psi_i(x_i)\psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k \neq i} m_{ki}(x_i) \prod_{l \neq j} m_{lj}(x_j)$$

Note that message equations come from

$$b(x_i) = \sum_{x_j} b(x_i, x_j)$$

But how do we justify the belief equations?

Belief equations exact on a tree



$$b(x_1) \propto \psi_1(x_1) m_{21}(x_1)$$

$$b(x_1) \propto \psi_1(x_1) \sum_{x_2} \psi_2(x_2)\psi_{12}(x_1, x_2) m_{32}(x_2) m_{42}(x_2)$$

$$b(x_1) \propto \psi_1(x_1) \sum_{x_2} \psi_2(x_2)\psi_{12}(x_1, x_2) \sum_{x_3} \psi_3(x_3)\psi_{23}(x_2, x_3) \sum_{x_4} \psi_4(x_4)\psi_{24}(x_2, x_4)$$

$$b(x_1) = \sum_{x_2, x_3, x_4} p(x_1, x_2, x_3, x_4)$$

"Dynamic programming"

"Transfer matrix"

Gibbs Free energy

- The Gibbs free energy $G[p(\{x\})] = U[p(\{x\})] - S[p(\{x\})]$ is a function which has its minimum at the equilibrium probability distribution: $p(\{x\}) = e^{-H(\{x\})} / Z$
- The value of the Gibbs free energy at its minimum is equal to the Helmholz free energy $F = -\ln Z$

Gibbs free energy for 2 nodes

$$H(x_1, x_2) = -J_{12}(x_1, x_2) - h_1(x_1) - h_2(x_2)$$

Introduce

$$G[p(x_1, x_2)] \equiv U[p(x_1, x_2)] - S[p(x_1, x_2)]$$

where

$$U[p(x_1, x_2)] \equiv \sum_{x_1, x_2} p(x_1, x_2) H(x_1, x_2)$$

$$S[p(x_1, x_2)] \equiv -\sum_{x_1, x_2} p(x_1, x_2) \ln p(x_1, x_2)$$

Minimize $G[p(x_1, x_2)]$ with respect to $p(x_1, x_2)$ subject to the constraint $\sum_{x_1, x_2} p(x_1, x_2) = 1$ to recover

$$p(x_1, x_2) = \frac{1}{Z} e^{-H(x_1, x_2)} \quad \text{where} \quad Z \equiv \sum_{x_1, x_2} e^{-H(x_1, x_2)}$$

Approximations

- Instead of using the "full" $G[p\{x\}]$, we only constrain part of $p\{x\}$:
  - $G[b_i(x_i)]$: Mean field, TAP, ...
  - $G[b_i(x_i), b_{ij}(x_i, x_j)]$: Bethe
  - $G[b_i(x_i), b_{ij}(x_i, x_j), b_{ijk}(x_i, x_j, x_k), \ldots]$: Kikuchi
- Various justifications possible for different approximations:
  - *Variational arguments*
  - *Taylor expansions*
  - *Exact limits*

Constraining One-site Beliefs

$$H = -\sum_{\langle ij \rangle} J_{ij} S_i S_j - \sum_i h_i S_i \quad , \quad S_i = \pm 1 \quad , \quad m_i \equiv b(S_i = 1) - b(S_i = -1)$$

Expand in powers of J:

$$G[\{m_i\}] = \sum_i \left[\frac{1+m_i}{2} \ln \frac{1+m_i}{2} + \frac{1-m_i}{2} \ln \frac{1-m_i}{2} \right] - \sum_{\langle ij \rangle} J_{ij} m_i m_j$$

Mean Field

$$- \frac{1}{2} \sum_{\langle ij \rangle} J_{ij}^2 (1 - m_i^2)(1 - m_j^2)$$

TAP correction

$$- \frac{2}{3} \sum_{\langle ij \rangle} J_{ij}^3 m_i (1 - m_i^2) m_j (1 - m_j^2)$$

$$- \sum_{\langle ijk \rangle} J_{ij} J_{ik} J_{jk} (1 - m_i^2)(1 - m_j^2)(1 - m_k^2)$$

$$- \ldots$$

More corrections...

Constraining one and two-site beliefs: the Bethe approximation

For pairwise MRF: $p(\{x\}) = \frac{1}{Z}\prod_{\langle ij \rangle}\psi_{ij}(x_i,x_j)\prod_i \psi_i(x_i)$

$G_{Bethe}[\{b_i(x_i), b_{ij}(x_i,x_j)\}] =$

$$\sum_{ij}\sum_{x_i,x_j} b_{ij}(x_i,x_j)\ln\frac{b_{ij}(x_i,x_j)}{\psi_{ij}(x_i,x_j)\psi_i(x_i)\psi_j(x_j)} - \sum_i (q_i - 1)\sum_{x_i} b_i(x_i)\ln\frac{b_i(x_i)}{\psi_i(x_i)}$$

Derived from exact result on a tree: $p(x_1, x_2, \ldots, x_N) = \prod_{\langle ij \rangle} b_{ij}(x_i,x_j)\prod_i [b_i(x_i)]^{1-q_i}$

Minimizing Bethe Free Energy Gives Belief Propagation

Lagrange multipliers $\lambda_{ij}(x_j)$

enforce the constraints $b_j(x_j) = \sum_{x_i} b_{ij}(x_i,x_j)$

Bethe stationary conditions = message update rules

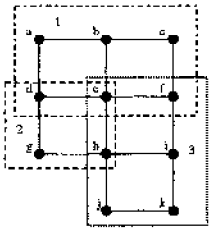
with $\lambda_{ij}(x_j) = \ln\prod_{k\neq i} m_{kj}(x_j)$

New directions for decoding

- Minimize Bethe free energy directly to arrive at BP results with guaranteed convergence. (Possible problem: could still end up in local minima which aren't codewords).
- Use better (Kikuchi) free energies and corresponding (generalized) belief propagation algorithms.

Kikuchi Approximations

Break lattice up into overlapping clusters



Bethe is special case where clusters are linked pairs of nodes.

Kikuchi Free Energy

Over-counting number

$$F_K = \sum_{r\in R} c_r\left(-\sum_{x_r} b_r(x_r)\ln\psi_r(x_r) + \sum_{x_r} b_r(x_r)\ln b_r(x_r)\right)$$

Average energy — Entropy

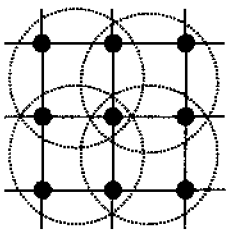
Approximation improves as basic clusters get bigger

Intuition: the approximation is improved because we account for troublesome tight loops exactly.

Generalized Belief Propagation

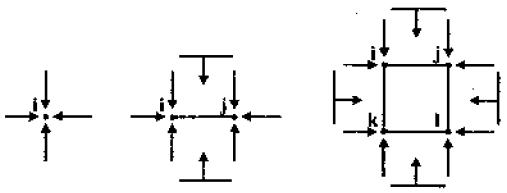
- Provably equivalent to Kikuchi approximation (messages are complicated combinations of Lagrange multipliers)
- Empirically, more likely to converge than ordinary BP, but not guaranteed.
- Can be nearly as fast as ordinary BP, but much more accurate. Complexity is exponential in cluster size.
- Works with arbitrary network topology and arbitrary cluster approximations.

Example: Square Lattice



Clusters of four nodes around a square

Region beliefs under Generalized Belief Propagation

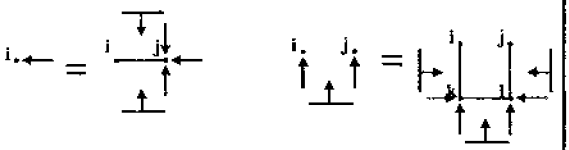


$$b_r(x_r) \propto \psi_r(x_r) \prod_{m_{s'r} \in M(r)} m_{s'r}(x_{s'})$$

GBP message-update rules

Use marginalization condition $b_s(x_s) = \sum_{x_{r \setminus s}} b_r(x_r)$

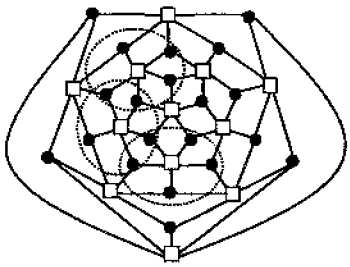
Groups of nodes send messages to other groups of nodes.



Update for messages ←

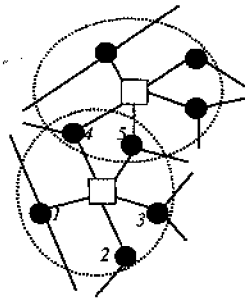
Update for messages ↑

GBP Decoding of Parity Check Codes

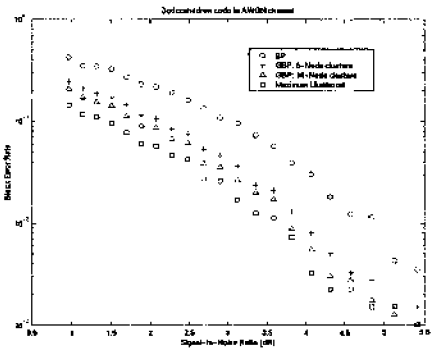


Clusters: all the nodes in a parity check

New messages



$$m_{123 \to 45}(x_4, x_5)$$



BP vs. GBP decoding

- BP decoding works well when the graph for the code has no tight loops.
- GBP is a big improvement over BP when there *are* tight loops.
- Many of the best codes have been designed for the BP decoder; hence no tight loops.
- Do best codes for long blocklengths have tight loops?

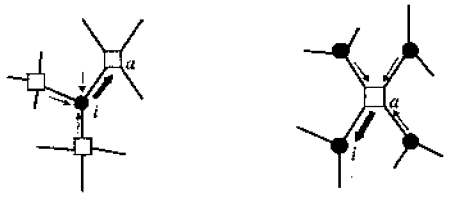
Analysis of Codes

"Averaging over the disorder" of the different blocks ("random magnetic fields") to compute average performance of code.

Methods:

- Density Evolution
- Replicas
- Renormalization Group (I.S. Yedidia and J.-P. Bouchaud)
- typical set method

Density Evolution in BEC



p_{ia} is the probability that i sends a a "?" message.

$$p_{ia} = p_e \prod_{b \neq a} q_{bi}$$

q_{ai} is the probability that a sends i a "?" message.

$$1 - q_{ai} = \prod_{b \neq a} (1 - p_{ia})$$

Density Evolution

- Neglecting correlations is reasonable when loops are large. Predictions normally become correct in infinite blocklength limit.
- Possible to handle other channels--e.g. for AWGN channel, one solves equations for probability distributions of messages, which can be approximated as Gaussians.
- One finds threshold behavior: e.g. for (3,5) Regular Gallager code in BEC:

$$p = p_e q^3$$

$$q = 1 - (1-p)^4$$

Below $p_e \approx .51757$ $p=q=0$ (no errors)

- Optimizing the threshold predicted by density evolution for the infinite blocklength limit is a popular way of designing codes.

Replica method

- The replica method is an approach to dealing with an average over disorder, e.g. the average over possible received blocks.
- *Basic idea*:
  - Stage 1: convert the problem with an average over disorder into an equivalent problem with no disorder, but with n replicas.
  - Stage 2: Solve replicated problem for any n, then take limit $n \to 0$.
- The replica method itself is exact, but it is often combined with approximations needed to solve the resulting "replicated" problem.
- I will first demonstrate the replica method for a tiny trivial problem, but replicas can also be used for larger problems when exact methods are not available. (Also, I will use the BSC, but other channels can also be handled).
- I will derive a replicated Hamiltonian in excruciating detail, but once you've done this once ("eaten your spinach"), you can usually jump to the replicated Hamiltonian very quickly.

A *really* trivial "code"

- In BSC

Codewords: 0 1

Assume that the "all-zeros codeword" is always transmitted.

Two possible received words: 0 with probability $1-f$

1 with probability f

When we receive a 0, we assign a belief $1-f$ in 0, f in 1

When we receive a 1, we assign a belief $1-f$ in 1, f in 0

Average belief in 0: $\overline{b(x=0)} = (1-f)^2 + f^2$

Average belief in 1: $\overline{b(x=1)} = 2f(1-f)$

The point of replicas

- For this example it was easy to average over disorder, but when N is large, it's not so simple.
- We want to avoid having to do Monte Carlo averaging over noise.

Replica approach: step-by-step

± 1 Spins: $S = (-1)^x$

Hamiltonian: $H = -hS$

Probability distribution for magnetic field: $p(h) = (1-f)\delta(h-h_0) + f\delta(h+h_0)$

$$h_0 \equiv \frac{1}{2}\ln\frac{1-f}{f}$$

Partition function for a given received block: $Z_h = e^{-h} + e^{h}$

Average Helmholz free energy: $\overline{F} = -\int_{-\infty}^{\infty} dh\; p(h)\ln Z_h$

Goal: $\overline{F} = -\int_{-\infty}^{\infty} dh\; p(h)\ln Z_h \longrightarrow \overline{F} = -\ln\sum_{\vec{H}}\exp(-\vec{H})$

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\left(1 + n\int_{-\infty}^{\infty} dh\; p(h)\ln Z_h\right) \quad \textit{using} \quad x = \lim_{n\to 0}\frac{1}{n}\ln(1+nx)$$

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\int_{-\infty}^{\infty} dh\; p(h)(1 + n\ln Z_h) \quad \textit{using} \quad \int dh\; p(h) = 1$$

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\int_{-\infty}^{\infty} dh\; p(h) Z_h^n \quad \textit{using} \quad \ln Z = \lim_{n\to 0}\frac{Z^n - 1}{n}$$

We've gotten closer to the form we want, but still need some more tricks to deal with the $p(h)$.

"Replica Identity"

So far:

$$p(h) = (1-f)\delta(h-h_0) + f\delta(h+h_0)$$

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\int_{-\infty}^{\infty} dh\; p(h) Z_h^n \qquad h_0 \equiv \frac{1}{2}\ln\frac{1-f}{f} \qquad Z_h = \sum_{S=\pm 1}\exp(-hS)$$

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\sum_{\tau=\pm 1}\left[\frac{1}{2} + \left(\frac{1}{2} - f\right)\tau\right]\left(\sum_{S=\pm 1}\exp(h_0\tau S)\right)^n$$

$$\textit{using} \quad p(h) = \sum_{\tau=\pm 1}\left[\frac{1}{2} + \left(\frac{1}{2} - f\right)\tau\right]\delta(h - \tau h_0)$$

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\sum_{\tau=\pm 1}\exp\left(h_0\tau + \ln\sqrt{f(1-f)}\right)\left(\sum_{S=\pm 1}\exp(h_0\tau S)\right)^n$$

$$\textit{using} \quad \sum_{\tau=\pm 1}\left[\frac{1}{2} + \left(\frac{1}{2} - f\right)\tau\right] = \sum_{\tau=\pm 1}\exp\left(h_0\tau + \ln\sqrt{f(1-f)}\right)$$

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\sum_{\tau=\pm 1}\exp\left(h_0\tau + \ln\sqrt{f(1-f)}\right)\left(\sum_{S=\pm 1}\exp(h_0\tau S)\right)^n$$

Now the critical (hard to justify) step: instead of treating n as a small real number, treat it as a positive integer (but then solve the resulting problem for any n, and take $n=0$ at the end):

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\sum_{\tau=\pm 1}\sum_{S_1=\pm 1}\cdots\sum_{S_n=\pm 1}\exp\left(h_0\tau + h_0\tau\sum_{m=1}^{n} S_m + \underbrace{\ln\sqrt{f(1-f)}}_{\text{negligeable constant}}\right)$$

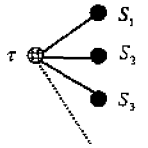
$$\textit{using} \quad \left(\sum_{S=\pm 1}\exp(h_0\tau S)\right)^n = \sum_{S_1=\pm 1}\cdots\sum_{S_n=\pm 1}\exp\left(h_0\tau\sum_{m=1}^{n} S_m\right)$$

(true for positive integers)

Stage 1 Goal Achieved

$$\overline{F} = -\lim_{n\to 0}\frac{1}{n}\ln\sum_{\tau=\pm 1}\sum_{S_1=\pm 1}\cdots\sum_{S_n=\pm 1}\exp\left(-\tilde{H}\right)$$

$$\tilde{H} \equiv -h_0\tau - h_0\tau\sum_{m=1}^{n} S_m + const.$$



$$\psi(\tau, S) = \begin{pmatrix} e^{h_0} & e^{-h_0} \\ e^{-h_0} & e^{h_0} \end{pmatrix}$$

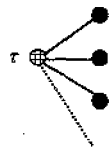
$$\psi(\tau) = \begin{pmatrix} e^{h_0} \\ e^{-h_0} \end{pmatrix}$$

Stage 2: Solve Replicated Model

We will use belief propagation, and assume

$m_{\tau S_1}(S_1) = m_{\tau S_2}(S_2) = \cdots = m_{\tau S_n}(S_n)$

$m_{S_1 \tau}(\tau) = m_{S_2 \tau}(\tau) = \cdots = m_{S_n \tau}(\tau)$ *"Replica symmetry"*



$$m_{\tau S}(S) = \sum_{\tau'=\pm 1} \psi_{\tau S}(\tau', S) \psi_\tau(\tau') (m_{S\tau}(\tau'))^{n-1}$$

$$m_{S\tau}(\tau) = \sum_{S'=\pm 1} \psi_{\tau S}(\tau, S')$$

$$\overline{b(x)} = \lim_{n\to\infty} \frac{1}{n} \sum_{n=1}^{n} b_n(S_n) = m_{\tau S}(S)$$

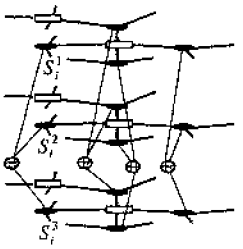
BP equations

Solving, we finally find

$$\overline{b(x=0)} = (1-f)^2 + f^2 \qquad \overline{b(x=1)} = 2f(1-f)$$

Replicated (3,4) Regular Gallager code

Effective model:

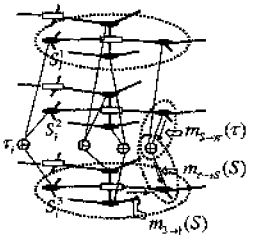


$$\psi(\tau_i, S_i^a) = \begin{pmatrix} e^{h_0} & e^{-h_0} \\ e^{-h_0} & e^{h_0} \end{pmatrix}$$

$$\psi(\tau_i) = \begin{pmatrix} e^{h_0} \\ e^{-h_0} \end{pmatrix}$$

Notice that we now have loops, so BP will not be exact.

Use simple "GBP" clusters



Solving the equations, we find a threshold, with $f_c \cong .180$

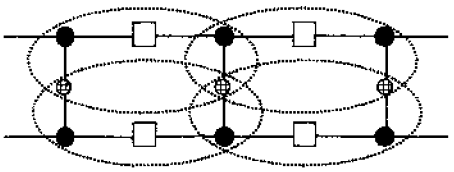
compared to exact result from density evolution: $f_c \cong .167$

$$m_{S\to\tau}(\tau) = \sum_{S=\pm 1} \psi(S,\tau) (m_{S\to 1}(S))^3$$

$$m_{\tau\to S}(S) = \sum_{\tau=\pm 1} \psi(S,\tau) \psi(\tau) (m_{S'\to\tau}(\tau))^{n-1}$$

$$m_{3\to 1}(S) = \sum_{\substack{S_1, S_2, S_3 \\ \Delta S_1 + S_2 + S_3 = 0}} m_{2\to 1}^2(S_1) m_{3\to 1}^2(S_2) m_{1\to 2}^2(S_3) m_{1\to 3}(S_2) m_{1\to 3}(S_3) m_{2\to 3}(S_1)$$

Better GBP clusters



Gives a slightly better prediction: $f_c = .177$