5. Density evolution
Probabilistic analysis of message passing algorithms

variable nodes  factor nodes

\[ x_1 \quad \psi_a(x_i, x_j, x_k) \]
\[ x_2 \]
\[ x_3 \]
\[ x_4 \]

- Consider factor graph model \( G = (V, F, E) \) and

\[
\mu(x) = \frac{1}{Z} \prod_{a \in F} \psi_a(x_{\partial a}) \prod_{i \in V} \psi_i(x_i)
\]

- Sum-product algorithm and max-product algorithms are instances of message-passing algorithms
  - Discrete \( x_i \in X \)
  - Two sets of messages \( \{\nu_{i \rightarrow a}(x_i)\} \) and \( \{\tilde{\nu}_{a \rightarrow i}(x_i)\} \)
  - Update:

\[
\nu_{i \rightarrow a}^{(t+1)} = F_{i \rightarrow a}(\{\tilde{\nu}_{b \rightarrow i}^{(t)} : b \in \partial i \setminus a\})
\]
\[
\tilde{\nu}_{a \rightarrow i}^{(t)} = G_{a \rightarrow i}(\{\nu_{j \rightarrow a}^{(t)} : j \in \partial a \setminus i\})
\]

Density evolution
assumptions for probabilistic analysis

- a *random graph* is a graph $G = (V, F, E)$ where $E$ is drawn randomly from a set of possible graphs
  e.g., Erdös-Renyi graph, random regular graph
- asymptotic analysis: in the limit $n \to \infty$

*density evolution* is used in

- analyzing channel codes
- analyzing solution space of XORSAT
- analyzing a message-passing algorithm for crowdsourcing
- analyzing belief propagation for community detection
- etc.
Example: channel coding

- sending messages through a noisy \textit{channel}

\[
x \xrightarrow{\text{Channel}} y
\]

- channel is defined by \( P_{Y|X}(y|x) \)
- Binary Erasure Channel (BEC)
  - input \( x_i \in \{0, 1\} \), output \( y_i \in \{0, 1, *\} \)

\[
\begin{array}{c}
0 \\
1
\end{array} \xrightarrow{1-\epsilon} \begin{array}{c}
1-\epsilon \\
\epsilon
\end{array} \xrightarrow{1} \begin{array}{c}
0 \\
? \\
1
\end{array}
\]

- goal: estimate \( \hat{x}_1, \ldots, \hat{x}_n \) given \( y_1, \ldots, y_n \)
- performance metric: average bit error probability

\[
P_{\text{error}} \equiv \frac{1}{n} \sum_{i=1}^{n} P(x_i \neq \hat{x}_i)
\]
Message length vs. block length

- no coding: \((01001) \Rightarrow (01 \ast 0\ast)\)
  - message \(k = 5\) bits, block length \(n = 5\)
    \(\Rightarrow\) rate of this code \(r \triangleq k/n = 1\), delay is one
  - \(P_{\text{error}} = \epsilon/2\)
- repetition code: \((00011100000000111) \Rightarrow (0 \ast \ast 1 \ast 10 \ast 0 \ast \ast \ast 111)\)
  - \(k = 5\), \(n = 15\)
  - rate \(r = 1/3\) and \(P_{\text{error}} = \epsilon^3/2\), delay is 3
  - in general, \(P_{\text{error}} = \epsilon^{1/r}/2 > 0\) (unless rate is zero)
- information theory
  - capacity of a BEC is \(1 - \epsilon\)
  - there exists a code such that \(\lim_{n \to \infty} P_{\text{error}} = 0\) with rate \(r < 1 - \epsilon\)
  - using the BEC \(n\) times, one can reliably send \(k = (1 - \epsilon)n\) bits of messages
Modern coding theory

- modern codes = iterative decoding (belief propagation)
  - Turbo code
  - Low-Density Parity Check (LDPC) code
  - Polar code
  - etc.

- LDPC code is defined by a factor graph model

```
<table>
<thead>
<tr>
<th>variable nodes</th>
<th>factor nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( b )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>( a )</td>
</tr>
<tr>
<td>( x_3 )</td>
<td></td>
</tr>
<tr>
<td>( x_4 )</td>
<td></td>
</tr>
</tbody>
</table>
```

\[ \psi_a(x_i, x_j, x_k) = \mathbb{I}(x_i \oplus x_j \oplus x_k = 0) \]

- block length \( n = 4 \)
- number of factors \( m = 2 \)
- allowed messages = \[\{0000, 0111, 1010, 1101\}\]
- message size \( k \triangleq \log_2(\text{# of allowed messages}) = 2 \) \( (k = n - m) \)
- rate \( r \triangleq k/n = 1/2 \)
- received \( y = (0 \ast 1\ast) \), then \( \hat{x} = (0111) \)
- received \( y = (0 \ast \ast\ast) \), then ?
Peeling decoder is equivalent to sum-product for BEC without loss of generality, suppose all 0’s sent
Peeling decoder is equivalent to sum-product for BEC
without loss of generality, suppose all 0’s sent
Peeling decoder is equivalent to sum-product for BEC
without loss of generality, suppose all 0’s sent

Density evolution
Peeling decoder is equivalent to sum-product for BEC
without loss of generality, suppose all 0’s sent
Peeling decoder is equivalent to sum-product for BEC
without loss of generality, suppose all 0’s sent
Peeling decoder is equivalent to sum-product for BEC

without loss of generality, suppose all 0’s sent
Peeling decoder is equivalent to sum-product for BEC
without loss of generality, suppose all 0’s sent
Peeling decoder is equivalent to sum-product for BEC
without loss of generality, suppose all 0’s sent
Peeling decoder is equivalent to sum-product for BEC
without loss of generality, suppose all 0’s sent

Density evolution
Modern coding theory

- decoding using belief propagation

\[ \mu_y(x) = \frac{1}{Z} \prod_{i \in V} \mathbb{P}_{Y|X}(y_i|x_i) \prod_{a \in F} \mathbb{I}(\bigoplus x_a = 0) \]

- use (parallel) sum-product algorithm to find \( \mu(x_i) \) and let

\[ \hat{x}_i = \arg \max \mu(x_i) \]

- minimizes bit error rate
Decoding by sum-product algorithm

- Directly applying parallel sum-product algorithm

\[
\nu_{i \rightarrow a}^{(t+1)}(x_i) = \mathbb{P}(y_i | x_i) \prod_{b \in \partial_i \setminus \{a\}} \hat{\nu}_{b \rightarrow i}^{(t)}(x_i)
\]

\[
\hat{\nu}_{a \rightarrow i}^{(t+1)} = \sum_{x_{\partial a \setminus \{i\}}} \prod_{j \in \partial a \setminus \{i\}} \nu_{j \rightarrow a}(x_j) \mathbb{I}(\oplus x_{\partial a} = 0)
\]

- Notice that all \(\nu, \hat{\nu}\)'s can take only one of the following three values:

\[
\nu_{i \rightarrow a}(x_i) \in \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \right\}
\]

hence, we will map these vectors to symbols \(\{0, 1, *\}\)

- because (proof by induction)

  - initially,

\[
\hat{\nu}_{a \rightarrow i}^{(0)}(x_i) = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, \quad \nu_{i \rightarrow a}^{(1)}(x_i) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{if } y_i = 0 \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if } y_i = 1 \\ \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} & \text{if } y_i = * \end{cases}
\]
recursively, assuming the input messages up to \( t \) are one of the three types,

\[
\begin{align*}
\tilde{v}^{(t+1)}_{a \rightarrow i} (x_i) &= \begin{cases} 
1 \\
0 \\
0 \\
1/2 \\
1/2 
\end{cases} 
& \text{if all other bits are determined and add up to 0} \\
\begin{cases} 
0 \\
1 \\
1/2 \\
1/2 
\end{cases} 
& \text{if all other bits are determined and add up to 1} \\
\begin{cases} 
1/2 \\
1/2 
\end{cases} 
& \text{if there is at least one bit that is not determined}
\end{align*}
\]

\[
\begin{align*}
\nu^{(t+1)}_{i \rightarrow a} (x_i) &= \begin{cases} 
1 \\
0 \\
0 \\
1 \\
1/2 \\
1/2 
\end{cases} 
& \text{if at least one of the input message is} \\
\begin{cases} 
1 \\
0 \\
0 \\
1 \\
1/2 \\
1/2 
\end{cases} 
& \text{if all input messages are}
\end{align*}
\]

consequently, the messages only take those three values

we will denote those three types of messages as 0, 1, and *, meaning determined to be 0 or 1, or not determined.
(simplified) Parallel sum-product for BEC

- \( \nu_{i \to a}^{(t)} \in \{0, 1, *\} \) our belief about \( x_i \)
- \( \tilde{\nu}_{a \to i}^{(t)} \in \{0, 1, *\} \) our belief about \( x_i \)
- at iteration 0: \( \nu_{i \to a}^{(0)} = y_i \)
- at iteration \( t \):

\[
\tilde{\nu}_{a \to i}^{(t)} = \begin{cases} 
* & \text{if any of the incoming messages is a *} \\
\oplus x_{\partial a \setminus i} & \text{otherwise}
\end{cases}
\]

\[
\nu_{i \to a}^{(t)} = \begin{cases} 
* & \text{if all of the incoming messages are *} \\
x_{b \to i} & \text{otherwise}
\end{cases}
\]

- this is equivalent to the peeling decoder
Probabilistic analysis: density evolution

- an LDPC code is defined by a graph $G$
- probabilistic analysis: we want to predict the performance of a given LDPC code $G$
- to this end, we use density evolution on the computation tree
- if $G$ is locally tree like up to depth $k$, and if we run sum-product algorithm for $k$ iterations, then the resulting message $\nu_{i \rightarrow a}^{(k)}$ is fully described by the computation tree for the message $\nu_{i \rightarrow a}^{(k)}$.
however, it is not always possible to apply density evolution

a few assumptions

- sparse random graph construction
  (e.g. random \((\ell, r)\)-regular graph from the configuration model)
- asymptotic analysis:
  in the limit \( n \to \infty \) but finite number of iterations \( t \)

why do we need these assumptions?

- it is difficult to analyze one particular graph, so we resort to the expected performance where the expectation also take into account the randomness in the graph generation
- random sparse graphs are locally tree-like
  - if we consider random \((d, d')\)-regular graphs, the expected number of 2-cycles is \((\frac{1}{n} + \cdots + \frac{d-1}{n}) \times n\), which is small compared to the number of edges
Probabilistic analysis: density evolution

- locally-tree like structure ensures that the incoming messages are independent
- formally, as \( n \to \infty \) local neighborhood of a node converges in probability to a random tree

\[
P(\lim_{n \to \infty} \text{depth } k \text{ neighborhood of a random } i\text{ is a tree}) = 1
\]

- **density evolution** for \((\ell, r)\)-regular graph
  - \( z_t \in [0, 1] \) be the probability a randomly chosen message from \( \{\nu_{i \to a}^{(t)}\} \) is an erasure
  - \( w_t \in [0, 1] \) be the probability a randomly chosen message from \( \{\tilde{\nu}_{a \to i}^{(t)}\} \) is an erasure
  - in the limit \( n \to \infty \), they satisfy the **density evolution equations**

\[
\begin{align*}
  w_t &= 1 - (1 - z_{t-1})^{r-1} \\
  z_t &= \epsilon w_t^{\ell-1}
\end{align*}
\]
\[ z_t = \epsilon(1 - (1 - z_{t-1})^{r-1})^{\ell-1} \]

with initial condition \( z_0 = \epsilon \)

- density evolution for (3,6) code with \( \epsilon = 0.4 \) (left) and 0.45 (right)

- rate of this code = 0.5, threshold \( \epsilon^* \approx 0.4xxx \),
- this simple code achieves rate less than the capacity = 1 - \( \epsilon \)
- \( P_{\text{error}}(t) = \lim_{n \to \infty} P_{\text{error}}(n, t) \)
- analyze \( \lim_{t \to \infty} \lim_{n \to \infty} P_{\text{error}}(n, t) \), is this what we want?
for a given value of $\epsilon$, we can **numerically** run the density evolution, since it is an evolution of a scalar value, which gives

bit error rate of (3, 6)-codes

how do we find $\epsilon^*$?
\[ z_t = \epsilon (1 - (1 - z_{t-1})^{r-1})^{\ell-1} \]

Let's change the equation to

\[ \left( \frac{z_t}{\epsilon} \right)^{1/(\ell-1)} = 1 - (1 - z_{t-1})^{r-1} \]

For a given \( \epsilon \), if there is no overlap, then achieve zero error probability

\[ \int_0^1 \epsilon y^{\ell-1} \, dy = \frac{\epsilon}{\ell}, \quad \int_0^1 (1 - (1 - x)^{r-1}) \, dx = 1 - \frac{1}{r} \]

Rate of the code = \( 1 - \frac{\ell}{r} \) vs. capacity = \( 1 - \epsilon \)

Extend this analysis to construct capacity achieving tornado codes.
Density evolution for general message passing algorithms

Consider factor graph model $G = (V, F, E)$ and

$$
\mu(x) = \frac{1}{Z} \prod_{a \in F} \psi_a(x_{\partial a}) \prod_{i \in V} \psi_i(x_i)
$$

▶ update:

$$
\nu_{i \rightarrow a}^{(t+1)} = F_{i \rightarrow a}(\{\hat{\nu}_{b \rightarrow i}^{(t)} : b \in \partial i \setminus a\})
$$

$$
\hat{\nu}_{a \rightarrow i}^{(t)} = G_{a \rightarrow i}(\{\nu_{j \rightarrow a}^{(t)} : j \in \partial a \setminus i\})
$$

▶ density evolution equation

$$
\zeta^{(t+1)} = F(\nu_1^{(t)}, \ldots, \nu_{\ell-1}^{(t)})
$$

$$
w^{(t)} = G(\zeta_1^{(t)}, \ldots, \zeta_{k-1}^{(t)})
$$
formally, as \( n \to \infty \) a randomly chosen message from \( \nu_{i \to a}^{(t)} \) converge in probability to \( z^{(t)} \)

who cares about random graphs?

who cares about asymptotics?

<table>
<thead>
<tr>
<th>alphabet ( x_i \in \mathcal{X} )</th>
<th>messages ( \nu_{i \to a} \in \mathcal{Y} )</th>
<th>density ( \mathcal{Z} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>discrete ( {0, 1} )</td>
<td>discrete ( {0, 1, *} )</td>
<td>continuous ( \mathbb{R} )</td>
</tr>
<tr>
<td>discrete</td>
<td>continuous ( \mathbb{R}^{\mathcal{X}</td>
<td>^{-1}} )</td>
</tr>
<tr>
<td>continuous ( \mathbb{R} )</td>
<td>distribution over ( \mathbb{R} )</td>
<td>dist. over dist. over ( \mathbb{R} )</td>
</tr>
</tbody>
</table>

how do we compute evolution of distributions?

- quantization
- Gaussian approximation
- \textit{population dynamics}: represent the density using ‘samples’