Nearly-optimal associative memories based on distributed constant weight codes

**Goals**
- Design a nearly-optimal associative memory in terms of efficiency,
  - Associative memory: device able to retrieve previously learned messages from part of their content,
  - Efficiency: ratio of the amount of bits learned to the amount of bits used,
- Use for that an architecture based on distributed constant weight codes.

**Binary constant weight codes**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Constant weight</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (n)</td>
<td>(\sum_{i=1}^{n} m_i = w)</td>
<td>({m} \subset {0;1}^n)</td>
</tr>
<tr>
<td>Weight (w)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overlapping (r)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Thrifty code**

A particular constant weight code with weight 1

Code containing only binary words with a single "1": 

![Binary code example](image)

**Drawback**

\(d_{\text{min}} = 2\) 

**Pros**

- Easy to decode and minimise the energy: 
  - Winner-take-all

**Clique code**

Another binary constant weight code: the clique code

- Cliques of constant size,
- Idea = use connections instead of vertices,
  - Example: \(\{3; 4; 5; 6; 7; 8\}\) characterizes a clique,
  - Other representation: \(0 0 1 1 1 1 1 1 0 0\) = constant weight code.

**Static parameters**

- \(n\) neurons,
- \(c\) clusters,
- \(l = \frac{n}{c}\) neurons per cluster,
- Memory effect \(\gamma\),
- \(W\) the binary adjacency matrix.

**Dynamic parameters**

\(v_i^{t+1}\) the value of neuron \(i\) of cluster \(c_i\) at iteration \(t\).

**Retrieving equations**

\[
v_{i}^{t+1} = \begin{cases} 1 & \text{if } \sum_{j \neq i} v_{j}^{t} + W(c_{i}, i) = \gamma + c - 1 \\ \max_{j \neq i} v_{j}^{t} & \text{otherwise} \\ 0 & \text{otherwise} \end{cases}
\]

**Density**

![Density graph](image)

**Retrieving (\(c, l = 8, 256\))**

![Retrieving graph](image)

**Efficiency**

![Efficiency graph](image)

**References**