Little-Hopfield Memory Storage with Minimum Probability Flow

Chris Hillar
(Kilian Koepsell, Jascha Sohl-Dickstein, Ngoc Tran)

Redwood Center for Theoretical Neuroscience
(HWNI UC Berkeley)
Outline

• Motivation: Distributed Memory Storage in Networks

• Necessary Properties of Sensor Networks

• Background:
  • Little-Hopfield Networks
  • Learning with Minimum Probability Flow (MPF)

• Demonstrations:
  • Exact Storage from Noisy / Corrupted Signals
  • Robust to Signal Corruption and Neural Damage
  • Learns Exponential Structures from Few Examples
  • Discovery of Neural Assemblies from Spike Trains
Modeling Sensor/Memory Networks

- Brain receives impoverished sensory input
- Yet we perceive detailed and accurate signal

Training

World

Sensor

Novel sensor signal

Cleanup

Computation

Percept

network trains on noisy sensor signals

synaptic changes

iterative dynamics
Desired Properties of a Neural Memory/Sensor Model

- **Signal Recovery:**
  - **Fast Dynamics** (\( \sim \log N \) updates for \( N \) neurons)
  - Robust to High Levels of Noise / Signal Loss
  - Robust to Neural Damage / Failure

- **Learning:**
  - Unsupervised (No Labels on Input Signals)
  - Learns Patterns from Noisy Signals
  - Synaptic Coupling Updates are Local in Nature
  - Efficient (e.g. Convex Objective)

- **Bonus:**
  - Potential for Robust Exponential Storage [HTK’12]
  - Learns Exponential Structures from Few Examples
• **Little-Hopfield Computational Network**  
  [Little ’74], [Hopfield ’82], [Cohen, Grossberg ‘83]

• **States:** binary \( \{0, 1\} \) vectors \( x = (x_1, ..., x_N) \)

• **Energy** of \( x \):

\[
E_x(J, \theta) := -\frac{1}{2} x^\top J x + \theta^\top x = -\sum_{i < j} x_i x_j J_{ij} + \sum_{i=1}^{n} \theta_i x_i
\]

• **Dynamics:** state \( x \) updates to state \( x' \) such that \( x' \) has lower (or equal) energy

• **Boltzmann machine:** stochastic dynamics  
  [Hinton, Sejnowski ’85]
3-Node Hopfield Network

\[
J = \begin{bmatrix}
0 & -1 & 1 \\
-1 & 0 & 2 \\
1 & 2 & 0 \\
\end{bmatrix}
\]

\[
\begin{array}{cccccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]
3-Node Hopfield Network

\[ J = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix} \]

\[ \begin{array}{c|cccccccccc} \hline x_1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ x_2 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ x_3 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} \]

e.g. \[ x'_1 : \quad x_2 \cdot (-1) + x_3 \cdot (1) > 0 \]
3-Node Hopfield Network

\[ J = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix} \]

\[ E_x \]

| \( x_1 \) | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| \( x_2 \) | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| \( x_3 \) | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

e.g.  \( x_1' : x_2 \cdot (-1) + x_3 \cdot (1) > 0 ? \)
3-Node Hopfield Network

\[ J = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix} \]

\[ E_x \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_3 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E.g. \( x_1' : x_2 \cdot (-1) + x_3 \cdot (1) > 0 \) ?
3-Node Hopfield Network

\[ J = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix} \]

\[ E_x \]

| \( x_1 \) | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| \( x_2 \) | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| \( x_3 \) | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

e.g. \( x'_1 : x_2 \cdot (-1) + x_3 \cdot (1) > 0 \) ?
3-Node Hopfield Network

\[ J = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix} \]

\[ x_1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \]

\[ x_2 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \]

\[ x_3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \]

e.g. \[ x'_1 : \quad x_2 \cdot (-1) + x_3 \cdot (1) > 0 \quad ? \]
3-Node Hopfield Network

\[ J = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix} \]

\[
\begin{array}{cccccccc}
 x_1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
 x_2 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
 x_3 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[ x'_1 : \quad x_2 \cdot (-1) + x_3 \cdot (1) > 0 \]
Inverse Problem: Efficiently store the optimal number of patterns possible as fixed-points (ie. Energy minima) in Little-Hopfield networks

\[ E_x \]

\[ E_{x'} \]

Stored Patterns
Relationship of dynamics in Little-Hopfield networks to theory of computation

- **Image Segmentation** [Shi-Malik, 2000]

- **Minimize quadratic energy** $E$ over **discrete set**
  - Minimum Cut, Normalized Cut, Modularity, etc
  - **NP-Hard** Discrete Optimization Problems
Learning in Little-Hopfield Networks

- [Hopfield ’82] **Outer Product Rule** (OPR)

\[ J = \sum_{x \in D} xx^\top \]

- [Wallace ’86, Jinwen ’93, ...] **Perceptron Rule** (PER)
Learning in Little-Hopfield Networks

- **Minimum Probability Flow (MPF)**
  [Sohl-Dickstein, Battaglino, DeWeese, PRL ’11]

- tractable convex objective function

- leads to neurologically plausible learning rule to train network

\[
\begin{array}{c|c|c}
\Delta J_{ij} & x_i = 0 & x_i = 1 \\
\hline
x_j = 0 & 0 & -e^{\frac{1}{2}(J_j x - \theta_j)} \\
x_j = 1 & -e^{\frac{1}{2}(J_i x - \theta_i)} & e^{-\frac{1}{2}(J_i x - \theta_i)} + e^{-\frac{1}{2}(J_j x - \theta_j)}
\end{array}
\]

**Theorem** [HSK’11]: If a set of patterns can be stored in a Little-Hopfield network then MPF stores these patterns
Algorithm

• Given a collection $D$ of binary patterns to store

**Minimize MPF Objective Function:**

$$K_D(J) := \sum_{x \in D} \sum_{x' \in N(x)} \exp \left( \frac{E_x - E_{x'}}{2} \right).$$

• Use $J$ as coupling matrix of a Hopfield net

**Theorem:** If $D$ can be stored in a Hopfield network then the above algorithm stores $D$

Algorithm

• (1) Given a collection $D$ of binary patterns to store

Minimize (over $J$) MPF Objective Function:

$$K_D(J) := \sum_{x \in D} \sum_{x' \in \mathcal{N}(x)} \exp \left( \frac{E_x - E_{x'}}{2} \right).$$

• (2) Use $J$ as coupling matrix of a Hopfield net
Algorithm

• (1) Given a collection $D$ of binary patterns to store

Minimize (over $J$) MPF Objective Function:

$$ K_D(J) := \sum_{x \in D} \sum_{x' \in \mathcal{N}(x)} \exp \left( \frac{E_x - E_{x'}}{2} \right). $$

• (2) Use $J$ as coupling matrix of a Hopfield net
## Little-Hopfield Fitting Algorithms (for N node network)

<table>
<thead>
<tr>
<th>Patterns Storable</th>
<th>Outer Product Rule (OPR)</th>
<th>Perceptron (PER)</th>
<th>Minimum Probability Flow (MPF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>N/log(N)</td>
<td>1.6N</td>
<td>1.6N</td>
</tr>
<tr>
<td>YES</td>
<td></td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimal</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>YES</td>
<td></td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tractable</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td></td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>YES</td>
<td></td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Local Learning Rule</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td></td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learns from Noisy Patterns</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td></td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cleanup / Recovery</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Poor</td>
<td></td>
<td>Better</td>
<td>Best</td>
</tr>
</tbody>
</table>
Optimal storage of binary patterns in a Hopfield Network

Theorem [H,S-D,K]: MPF stores the optimal number of patterns, and this number is at least \( N \) (asymptotically)
Optimal storage of binary patterns in a Hopfield Network

Theorem [H,S-D,K]: MPF stores the optimal number of patterns, and this number is at least \( N \) (asymptotically).
More efficient than Perceptron

Max PER iterations

dimension of binary vectors = 64
MPF fitting of Hopfield networks outperforms Perceptron fitting in corruption tolerance.

Perceptron fit network

MPF Fit network

dimension of binary vectors = 128
Average number of Hopfield iterations until convergence compared to $\log(N)$.
Noisy pattern storage

- Dimension = 64
- Bit corruption during learning = 20 bits flipped (31%)
Noisy Fingerprints

80 fingerprints
(DB2_B database)
4096 = 64 x 64 pixel binary images
another example binary fingerprint
ex. sample of a binary fingerprint shown during training (30% corrupted bits)
ex. sample of a binary fingerprint shown during training (30% corrupted bits)

new sample shown to trained network (40% corruption)
ex. sample of a binary fingerprint shown during training (30% corrupted bits)

one Hopfield dynamics update

new sample shown to trained network (40% corruption)
new sample shown to trained network (40% corruption)

ex. sample of a binary fingerprint shown during training (30% corrupted bits)

one Hopfield dynamics update

converged pattern (avg. ~ 5 dynamics updates)
A sample of a binary 4096-bit fingerprint shown during learning (30% corrupted bits) converges to a pattern (avg. ~ 5 dynamics updates) after one dynamics update. A new sample shown to a trained network (40% corruption) achieves this converged pattern.

**Experiment:**
Stored 80 fingerprints (DB2_B database)
4096 = 64 x 64 pixel binary images

The energy landscape illustrates the process, highlighting spurious minima and the path to the converged pattern.
More Noisy Fingerprints

- train
- new
- 1-hop
- converge
Exponential Storage

**Theorem:** There exist Little-Hopfield networks with robust exponential storage

### Exponential Storage in Little-Hopfield Nets

<table>
<thead>
<tr>
<th>Hopfield</th>
<th># of memories</th>
<th>robust</th>
<th>prescribed</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/log(N)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>~1.6 N</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>exponential</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>exponential</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>exponential</td>
<td>no</td>
<td>no</td>
<td>(yes)</td>
</tr>
<tr>
<td>exponential</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

- **Hopfield**: Little '74, Hopfield '82
- **MPF**: HSK '11
- **random**
- **equal entries**
- **non-binary**
- **clique net**

References:
- [Little, '74](#)
- [Hopfield, '82](#)
- [HSK, '11](#)
- [Tanaka et al, '80](#)
- [Gross et al, '84](#)
- [McEliece et al, '85](#)
- [Fulan, '88](#)
- [Kumar et al, 2011](#)
- [HTK, '12](#)
Learning Structure from few observations

\[
\begin{pmatrix}
12 & 1 \\
13 & 0 \\
14 & 1 \\
23 & 0 \\
24 & 1 \\
34 & 0
\end{pmatrix}
\]
Learning Structure from few observations

\[
\begin{array}{ccc}
12 & \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\
13 & 1 \\
14 & 1 \\
23 & 0 \\
24 & 1 \\
34 & 0 \\
\end{array}
\]
Learning Structure from few observations

Example computational result:

Network learns all 15-cliques of a 30-node graph from 360 (randomly generated) training 15-cliques.

There are 155,117,520 15-cliques on 30 nodes.

e.g. 4-clique in 5-node graph
Each square represents a 15-clique in a 30-node graph.

Network learns all 15-cliques of a 30-node graph from 360 (randomly generated) training 15-cliques.

There are 155,117,520 15-cliques on 30 nodes.

Training examples used to learn all other squares.
A network with $v = 100$ vertices robustly stores $\binom{100}{50} \approx 10^{29}$ memories (i.e., all 50-cliques in a 100-node graph) using binary vectors of length 4950, each having $\binom{50}{2} = 1225$ nonzero coordinates. Figure shows that a 50-clique memory represented with 4950 bits is recovered by the dynamics after flipping $\frac{1}{2} \cdot 1225 = 612$ at random.
Detecting Neural Assemblies
PCA Components of Spike Trains
Redwood Center for Theoretical Neuroscience (U.C. Berkeley)
Mathematical Sciences Research Institute
National Science Foundation

Ngoc Tran
Kilian Koepsell
Jascha Sohl-Dickstein