HOPFIELD NETWORK

一个完全连接的对称加权网络，其中每个节点同时充当输入和输出节点。

Used for associated memories and combinatorial optimization。

不同的形式：离散和连续。我们将重点介绍 discrete Hopfield model，因为它的数学描述更直接。

在离散模型中，每个神经元的输出为 1 或 -1。

在最简单的形式中，输出函数是符号函数 (sign function)，它为大于 0 的参数生成 1，否则生成 -1。

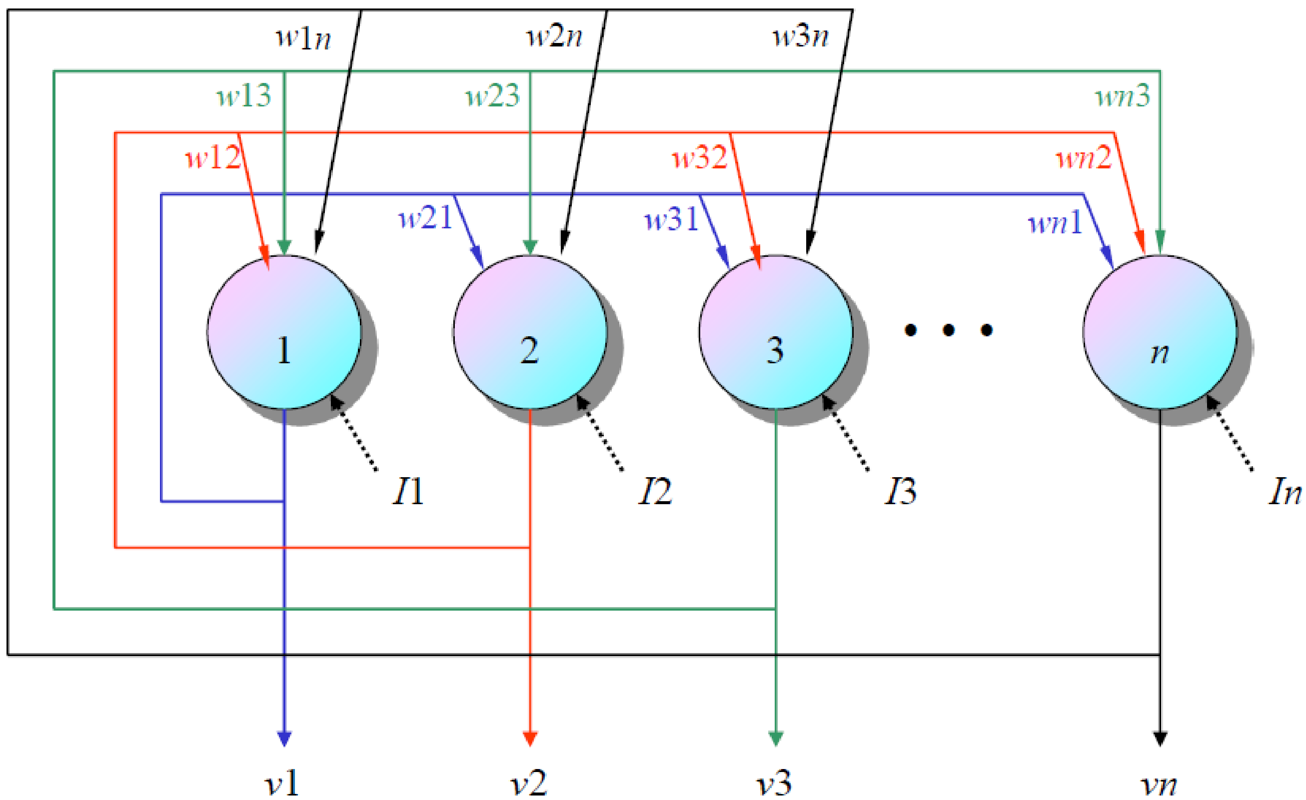
Discrete Hopfield Network

Architecture:

- single layer（unit 同时用作输入和输出）
- nodes are threshold units (binary or bipolar)
- 权重：全连接、对称和对角线为 0 (对角线为 0 意味着 unit 连接自身的权重为 0)

$$w_{ij} = w_{ji}, \quad w_{ii} = 0$$

- x_i 是外部输入，可能是瞬态的 (transient)，也可能是永久性的 (permanent)



注：上图的 I_i 就是输入，也是 x_i。

根据以下等式执行存储：

$$w_{ij} = \frac{1}{N}\sum_{p=1}^{P} x_i^p x_j^p$$

注：每个神经元中，输入和输出的积的总和，i 是 input，j 是 output。

权重矩阵是对称的，即 $w_{ij} = w_{ji}$。

约束条件 $w_{ii} = 0$ 对于 network behavior 很重要。从数学上可以证明，在这些条件下，网络将在无限次迭代中达到稳定的激活状态 (stable activation state)。

在模型的离散版本中，输入或输出向量的每个分量只能假定值为 1 或 −1。然后根据以下公式计算神经元 i 在时间 t 处的输出：

$$v_i(t) = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} v_j(t-1)\right)$$

注：sgn 是激活函数。

Stable state

网络将输入模式与自身相关联，这意味着在每次迭代中，激活模式将绘制到其中一个模式。

收敛后，网络很可能会呈现它初始化时使用的模式之一。

因此，Hopfield network 可以用来恢复不完整或嘈杂的输入模式。

Update rule

使用输入向量 recall 存储的向量。

每次，随机选择一个 unit 进行更新。

定期检查收敛（stable state）。

异步模式更新规则 (Asynchronous mode update rule)：

$$H_i(t+1) = \sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} v_j(t) + I_i$$

$$v_i(t+1) = \operatorname{sgn}\left[H_i(t+1)\right] = \begin{cases} 1 & H_i(t+1) \geq 0 \\ -1 & H_i(t+1) < 0 \end{cases}$$

Update Example

- A 4 node network, stores 2 patterns (1 1 1 1) and (-1 -1 -1 -1)
- Weights: $w_{\ell,j} = 1, \text{ for } \ell \neq j, \text{ and } w_{j,j} = 0 \text{ for all } j$

- Corrupted input pattern: (1 1 1 -1)

注：W 为 1，输入为 (1 1 1 -1)

Node selection		**output pattern**
node 2:	$w_{2,1}x_1 + w_{2,3}x_3 + w_{2,4}x_4 + I_2 = 1 + 1 - 1 + 1 = 2$	(1 1 1 -1)
node 4:	$1+1+1-1 = 2$	(1 1 1 1)

No more change of state will occur, the correct pattern is recovered

注：选择 node 2 和 4 进行更新。根据上面的规则，node 4 的结果为 2，大于 0，因此将 node 4 的输出变为 1。

- Equal distance: (1 1 -1 -1)

node 2:	net = 0, no change	(1 1 -1 -1)
node 3:	net = 0, change state from -1 to 1	(1 1 1 -1)
node 4:	net = 0, change state from -1 to 1	(1 1 1 1)

No more change of state will occur, the correct pattern is recovered

如果使用不同的节点选择顺序，则存储的模式 (-1 -1 -1 -1) 可能被 recall。

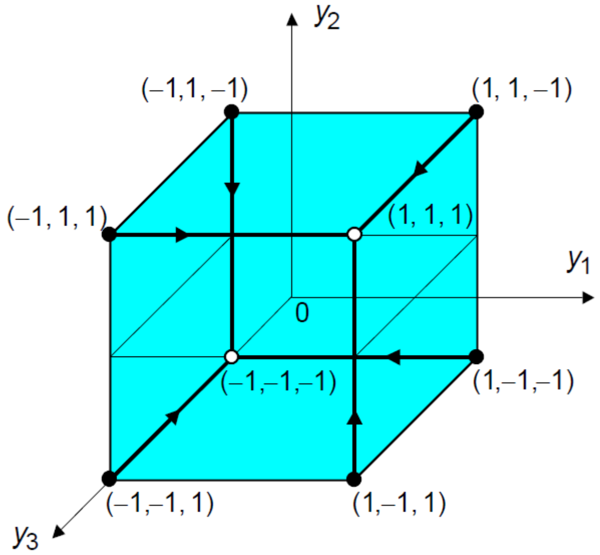
- Missing input element: (1 0 -1 -1)

Node selection		**output pattern**
node 2:	$w_{12}x_1 + w_{32}x_3 + w_{42}x_4 + I_2 = 1 - 1 - 1 + 0 < 0$	(1 -1 -1 -1)
node 1:	net = -3, change state to -1	(-1 -1 -1 -1)

No more change of state will occur, the correct pattern is recovered

Discrete Hopfield Network

三神经元的 Hopfield network 可能有的 8 种状态：



注：其中黑点代表不稳定的状态，白点代表稳定的状态 (也叫 fundamental memories)，不稳定的状态会在几次迭代后到底稳定的状态。

稳定状态由权重矩阵 W、当前的输入向量 X 和阈值矩阵 q 决定。如果输入向量部分不正确或不完整，则初始状态将在几次迭代后收敛到稳定状态。

例如，假设网络需要记住两个相反的状态，(1, 1, 1) 和 (-1, -1, -1)。

因此：

$$Y_1 = \begin{bmatrix}1\\1\\1\end{bmatrix} \quad Y_2 = \begin{bmatrix}-1\\-1\\-1\end{bmatrix} \quad \text{or} \quad Y_1^\top = \begin{bmatrix}1 & 1 & 1\end{bmatrix} \quad Y_2^T = \begin{bmatrix}-1 & -1 & -1\end{bmatrix}$$

构建权重矩阵：

$$\mathbf{W} = \sum_{k=1}^{p} \mathbf{x}^k (\mathbf{x}^k)^T - p\mathbf{I} \qquad w_{ij} = \begin{cases} \sum_{k=1}^{p} x_i^k x_j^k & i \neq j \\ 0 & i = j \end{cases}$$

确定权重矩阵，如下所示：

$$\mathbf{W} = \begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}1 & 1 & 1\end{bmatrix} + \begin{bmatrix}-1\\-1\\-1\end{bmatrix}\begin{bmatrix}-1 & -1 & -1\end{bmatrix} - 2\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0 & 2 & 2\\2 & 0 & 2\\2 & 2 & 0\end{bmatrix}$$

接下来，通过输入向量序列 X1 和 X2 测试网络，它们分别等于输出（或目标）向量 Y1 和 Y2。

通过使用输入向量 X 激活 Hopfield 网络并计算实际输出向量 Y：

$$Y_1 = sign\left\{\begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right\} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

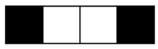
$$Y_2 = sign\left\{\begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right\} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

现在回到上面的图，基本记忆（稳定状态）(1, 1, 1) 吸引不稳定状态 (-1, 1, 1), (1, -1, 1) 和 (1, 1, -1)。与基本记忆 (1, 1, 1) 相比，这些不稳定状态中的每一个都代表一个错误。

因此，Hopfield 网络可以充当纠错网络。

Example 1: Weights Matrix

$$\mathbf{x}^1 = (1, -1, -1, 1)^T$$



$$\mathbf{x}^2 = (-1, 1, -1, 1)^T$$

$$\mathbf{x}^1(\mathbf{x}^1)^T + \mathbf{x}^2(\mathbf{x}^2)^T = \begin{bmatrix} 2 & -2 & 0 & 0 \\ -2 & 2 & 0 & 0 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & -2 & 2 \end{bmatrix}$$

$$\mathbf{x}^1(\mathbf{x}^1)^T = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$\mathbf{x}^2(\mathbf{x}^2)^T = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$

$$\mathbf{W} = \sum_{k=1}^{p} \mathbf{x}^k (\mathbf{x}^k)^T - p\mathbf{I}$$

$$\mathbf{W} = \begin{bmatrix} 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \end{bmatrix}$$

Example 2: Spurious State

- Use a 4-node Hopfield network to store 3 patterns:

  (1 1 -1 -1), (1 1 1 1) and (-1 -1 1 1)

  - weights:

$$\begin{pmatrix} 0 & 1 & -1/3 & -1/3 \\ 1 & 0 & -1/3 & -1/3 \\ -1/3 & -1/3 & 0 & 1 \\ -1/3 & -1/3 & 1 & 0 \end{pmatrix}$$

- Corrupted input pattern: (-1 -1 -1 -1)
  - if node 4 is randomly selected:

  $(-1/3\ -1/3\ 1\ 0)\ (-1\ -1\ -1\ -1)^T + (-1) = 1/3 + 1/3 - 1 - 0 - 1 = -4/3$
  - no change of state for node 4
  - same for all other nodes, net stabilized at (-1 -1 -1 -1)
  - a spurious state is recalled

注：spurious state 就是一个稳定的，但是不在 memory 中的状态。

- For input pattern (-1 -1 -1 0)
  - if node 4 is selected first

  $(-1/3\ -1/3\ 1\ 0)\ (-1\ -1\ -1\ 0)^T + (0) = 1/3 + 1/3 - 1 - 0 - 0 = -1/3$
  - node 4 changes state to -1: (-1 -1 -1 -1)
  - network stabilizes at (-1 -1 -1 -1)
  - however, if the node selection sequence is 1>2>3>4, the net stabilizes at state (-1 -1 1 1): a correct pattern

$$W = \begin{pmatrix} 0 & 1 & -1/3 & -1/3 \\ 1 & 0 & -1/3 & -1/3 \\ -1/3 & -1/3 & 0 & 1 \\ -1/3 & -1/3 & 1 & 0 \end{pmatrix}$$

-1	-1	-1	0
-1	-1	-1	0
-1	-1	-1	0
-1	-1	1	0
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1

Limitations of Hopfield Network

可以存储和准确调用的模式数量受到严重限制

- net 可能收敛到一种新的 spurious pattern

如果示例模式与另一个示例模式共享许多共同的位，则该示例模式将不稳定（共同点太多，会变成其他的模式）。