Hopfield 网络

张轩，复旦大学核科学与技术系

一、Hopfield 网络

Hopfield 网络是一种针对记忆功能的神经网络建模，在 1982 年由 Hopfield 提出。Hopfield 神经网络的记忆功能的本质是通过训练调节系统的参数，从而产生不动点。这些不动点就被视为所谓的“记忆”。

1.1 记忆和动力系统的简单讨论

我们可以考虑如下单个神经元的动力系统

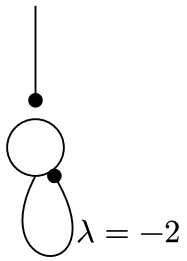


图 1　单个神经元的记忆模型

其动力学方程为

$$\tau_n \frac{\mathrm{d}v}{\mathrm{d}t} = -v + F(h(t) + \lambda v)$$

其中 F 是激活函数，我们可以取为 $F(x) = \tanh(x)$。

不难画出这个系统的相图和分岔图，如图2所示

当 $\lambda < 1$ 的时候，系统只有 $v = 0$ 一个不动点，而当 $\lambda > 1$ 时，存在三个不动点。从 $\lambda < 1$ 到 $\lambda > 1$ 这是一个典型的超临界叉形分叉。

当 $\lambda > 1$ 时，系统的两个不动点 (近似取为 $v = \pm 1$)，可以在外界刺激下相互转换，而没有外界刺激时会保持不变，如图3。这就是单个神经元的记忆模型。

这部分内容有一些简单的代码对应，参考”single neuron and memory.ipynb” 文件。

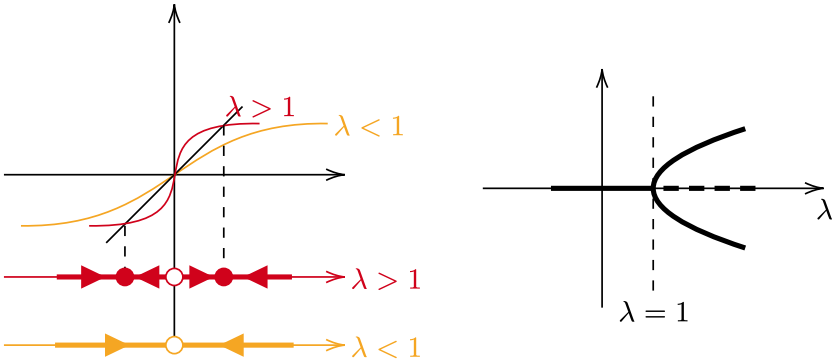


图 2　单个神经元记忆模型的相图

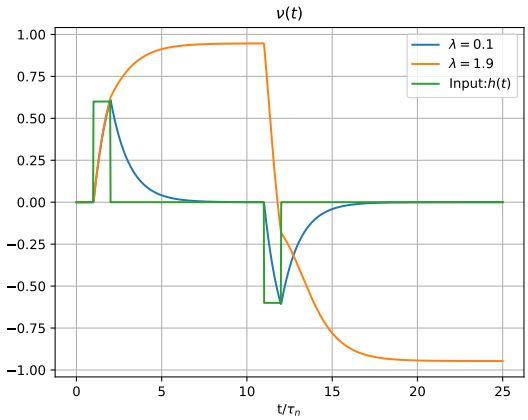


图 3　单个神经元记忆模型在外界刺激下的动力学行为

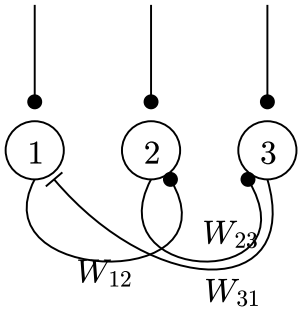


图 4　多个神经元记忆模型

1.2　Hopfield 网络

在一般的神经网络中，不会出现神经的自相互作用，所以上面的谈到的模型只是一个简单到不真实的模型。Hopfield 网络是更接近实际记忆模型，它不关注神经元是否自相互作用。

神经元之间的相互作用以权重矩阵 W_{ij} 描述，无自相互作用意味着 $W_{ii}=0$

$$\begin{cases}\tau_1\frac{\mathrm{d}v_1}{\mathrm{d}t}=-v_1+F\left(h_1(t)+\sum\limits_j W_{1j}v_j\right)\\ \vdots\\ \tau_N\frac{\mathrm{d}v_N}{\mathrm{d}t}=-v_N+F\left(h_N(t)+\sum\limits_j W_{Nj}v_j\right)\end{cases}$$

我们考虑一种简化的模型。取步长为 τ_i 的欧拉折线法近似，取激活函数为包含阈值的二值函数。

$$F(x)=\begin{cases}1 & x\geqslant 0\\ -1 & x<0\end{cases}$$

每个一神经元都只有 $+1,-1$ 两种状态（激活和不激活），它们的动力学演化规律为

$$v_i^{(n+1)}=\begin{cases}1 & \sum\limits_j W_{1j}v_j^{(n)}\geqslant\theta_i\\ -1 & \sum\limits_j W_{1j}v_j^{(n)}<\theta_i\end{cases}\tag{1}$$

其中 θ_i 是 $h_i(t)$ 保留而来的，我们设想演化过程中 $h_i(t)$ 不轻易随时间改变，于是它们发挥着改变阈值的作用。

Hopfield 是一个二值的离散的动力系统，通常的分析工具不再适用。但是巧妙的是这个系统存在一个能量函数（Lyapunov 函数）

$$E(v)=-\frac{1}{2}\sum_{i,j}v_iW_{ij}v_j-\sum_i\theta_iv_i=-\frac{1}{2}\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{v}+\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{v}\tag{2}$$

$$\Delta E^{(n)}=-\sum_i\Delta v_i^{(n+1)}\left(\sum_j W_{ij}v_j^{(n)}-\theta_i\right)\leqslant 0$$

不难看出，每一次演化过程中，能量函数只能降低而不能升高，借助这个能量函数，我们可以分析 Hopfield 网络的动力学行为。

1.3 Hopfield 网络的训练方法

在相关文献[1] [2]中，提到了一种 Hebb 学习规则，通过巧妙地设置权重矩阵，可以让 Hopfield 网络“记住”一些状态。例如，我们想让 Hopfield 网络记住 $\boldsymbol{v}$，就可以将权重矩阵 W 设置为 $\boldsymbol{v}$ 的外积

$$W=\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}},W_{ij}=\sum_{i,j}v_iv_j$$

这样，能量函数就是（阈值为零时）

$$E(\boldsymbol{u})=-\frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}\boldsymbol{u}=-\frac{1}{2}|\boldsymbol{v}\cdot\boldsymbol{u}|^2$$

可以发现，只有当 $\boldsymbol{u}$ 和 $\boldsymbol{v}$，平行的时候，能量达到最小值并稳定。考虑到状态的每一个分量只有 ± 1 可以选，所以被“记住”的状态 $\boldsymbol{u}$ 就是吸引子。

如果需要记住多个状态，可以将权重矩阵设置为这些状态的外积的平均值

$$W = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{v}_k^{\mathrm{T}}\boldsymbol{v}_k, W_{ij} = \frac{1}{N}\sum_{k=1}^{N}\sum_{i,j} v_{k,i}v_{k,j}$$

当然这个里面存在一个小问题，即 Hopfield 网络是要求没有自相互作用的，即权重矩阵的对角元全为 0。这其实很简单

$$\begin{aligned} E &= -\frac{1}{2}\sum_{i,j} v_i W_{ij} v_j - \sum_i \theta_i v_i \\ &= -\frac{1}{2}\sum_{i\neq j} v_i W_{ij} v_j - \sum_i \theta_i v_i - \frac{1}{2}\sum_i W_{ii} v_i^2 \\ &= -\frac{1}{2}\sum_{i\neq j} v_i W_{ij} v_j - \sum_i \theta_i v_i - \frac{n}{2} \end{aligned}$$

因为按照前面的方法训练得到的权重矩阵，对角元都必然是 1。于是对角元的贡献总是一个常值。在能量函数上增加一个常值对系统没有任何影响。

1.4 Hopfeild 网络的进一步讨论

从能量函数(2)中，我们可以对 Hopfield 网络做更进一步的讨论。

这个能量从能量函数中不难发现，当阈值 θ_i 取 0，即没有外界刺激的的时候。系统具有一个 $\mathbb{Z}_2$ 对称性。我们将状态中的 -1 替换为 $+1$，$+1$ 替换为 -1，系统的能量不变。

这意味着如果我们希望 Hopfield “记住”某个状态 $\boldsymbol{v}$，则这个状态的“反状态” $-\boldsymbol{v}$ 也会被记住。即记忆的吸引子一定是成对出现的，如果不考虑外界刺激，Hopfield 网络不可能存在单一吸引子，也即 Hopfield 不可能只记住一个状态。这将在后面的图片记忆中看到，Hopfield 网络记住的某个图片，那么这个的反色图片也会被记住。

Hopfield 网络中，可能会存在一些“假吸引子”，这主要是为了构造记忆的吸引子，能量函数会在这些吸引子上达到最小值，于是在其他地方可能就存在小的极小值（如图5），是的状态被吸引到小的极小值上去。产生了“假吸引子”，从而让 Hopfield “回忆”出错误的结果。这也将在后面看到。

1.5 Hopfield 网络的物理学对应

从 Hopfield 的能量函数(2)，以及 ± 1 的状态设置，不难发现这在物理学上，对应着伊辛模型（Ising Model）。

伊辛模型是统计物理学的常用的模型，其实际对应是一种“经典粒子的自旋”模型。“经典粒子”说明这些例子是可区分的，而且位置固定的分布在物质中，“自旋”是

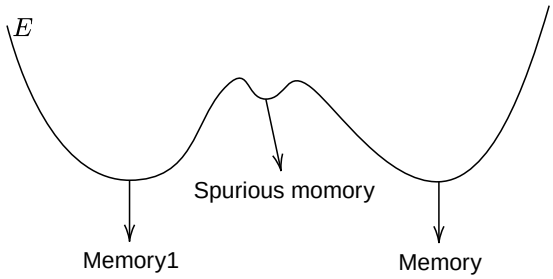


图 5　假吸引子

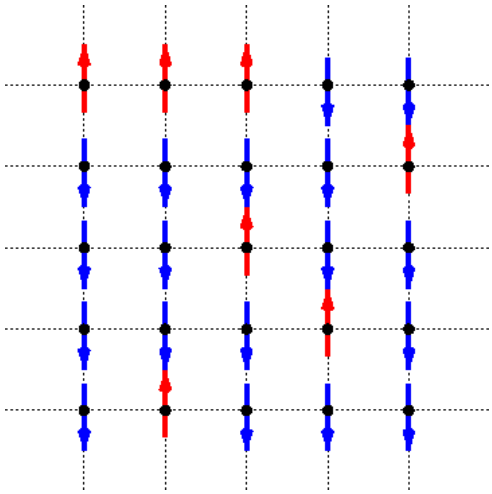


图 6　伊辛模型

这些粒子的唯一的自由度。粒子之间的自旋相互作用和自旋磁矩在外界磁场的能量构成了系统的能量，其表达式和(2)非常相似。

$$H(s_1, \ldots, s_N) = -\sum_{<i,j>} J_{ij} s_i s_j - \sum_i h_i s_i$$

这里 $h_i = \mu B_i$，μ 是自旋磁矩，B_i 是外磁场磁感应强度。求和号 $< i, j >$ 表示伊辛网络中毗邻的项，例如，在一维伊辛链中，可以写作 $\sum_{|i-j|\leqslant 1}$。

如果模型中的 J_{ij}, h_i 都是和下标无关的常数，统计物理学的做法是采用平均场近似处理，这种近似需要网络具有足够好的对称性。对于一维和二维的伊辛模型，已经存在解析解。对于更高维度的伊辛模型，解析求解的似乎希望渺茫。更进一步研究方法可以采用 Monte Carlo 方法。

当然，统计物理的研究对象是平衡态（热力学）。其方法几乎和动力学无关部分，Hopfield 网络确实纯粹的动力学模型，或许 Hopfield 网络也可以给伊辛模型的研究提供新的思路。

二、Hopfield 网络的搭建

我们考虑为 Hopfield 网络专门构建一个类，这个类在初始化的时候，需要确定节点的个数，并且将权重矩阵 W_{ij} 和阈值 θ_j 全部设为 0。

这个类需要具有一些功能，例如对于给定的状态进行训练以及训练完成后的动力学演化。实现了这个两个功能 Hopfield 网络就算搭建完成了。

值得一提的是根据(1)的公式，有两种演化放方法。分别是同步更新方法和异步更新方法。同步更新方法指的是一次将状态整个更新，从 n 到 $n+1$ 所有节点 j 都更新。而异步更新方法指的是将状态逐个更新，从 n 到 $n+1$ 只改动一个节点，从而这个节点的改动会影响到每一个更新周期中其他节点的状态。

有趣的是，我们发现异步更新方法下吸引子更强烈，而同步更新方法更容易产生“震荡”现象。

这些在我的代码文件“Hopfield_network.ipynb”中具有对应，除此之外余下的功能都是为了可视化以及进一步研究而建立的。

三、小规模 Hopfield 网络的动力学探究

N 个节点的 Hopfield 网络就会产生 2^N 个状态，这种数量巨大的离散状态分析是非常复杂的。我们从较简单的情况入手。

3.1 2 节点 Hopfield 网络

考虑构建一个 2 节点的 Hopfield 网络，并且训练它记住 $[-1,1]$ 这个状态。于是其权重矩阵是

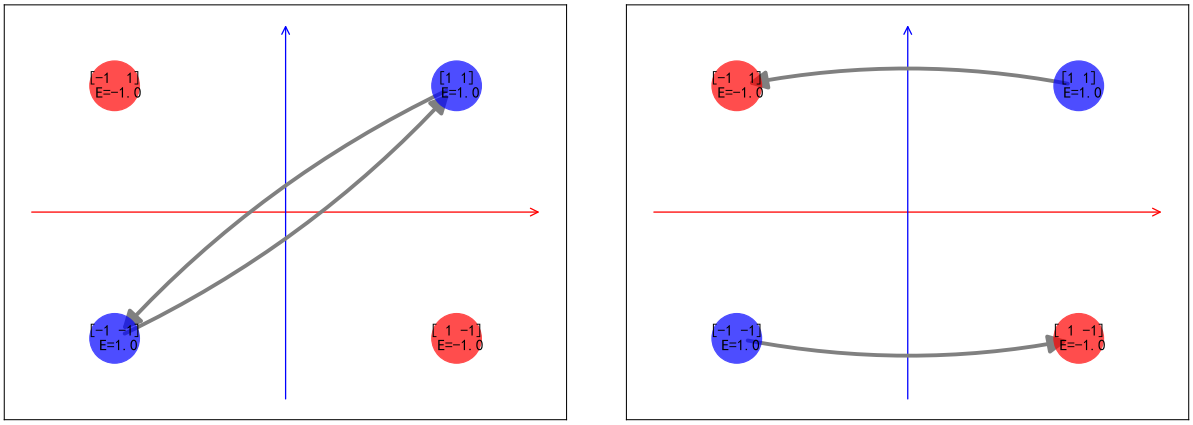
$$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

分别绘制其同步更新策略和异步更新策列的相图（状态转换图），如图7

可以看到在两种更新策略下，所存储的记忆 [1,-1] 以及其“反记忆” [-1,1] 都是不动点。而在同步更新策略下，余下的两个能量为 0 的节点相互震荡，即从它们出发使用同步更新策略，永远无法无法“回忆”出存储的状态。而异步更新策略解决了这个问题，它可以破除这个震荡，让它们都回到稳定不动点上去。

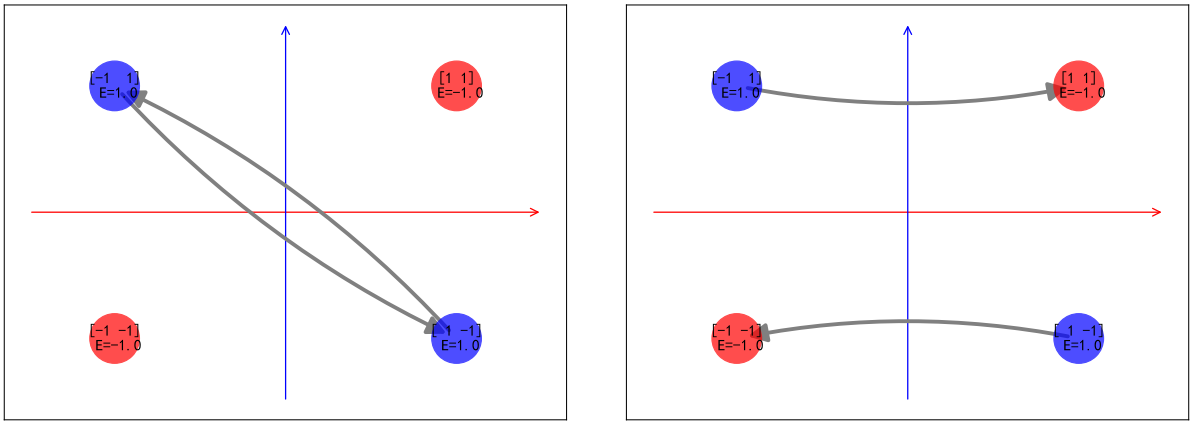
如果考虑让两节点 Hopfield 网络记忆状态 [1,1]，其权重矩阵为，其相图为图8。可以看到，这个 Hopfield 网络表现出和之前的网络的一种对称性和相似性。

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



(a) 同步更新　　(b) 异步更新

图 7　2 节点记忆 [-1,1]Hopfield 网络，图中红色的节点表示稳定状态



(a) 同步更新　　(b) 异步更新

图 8　2 节点记忆 [1,1]Hopfield 网络，图中红色的节点表示稳定状态

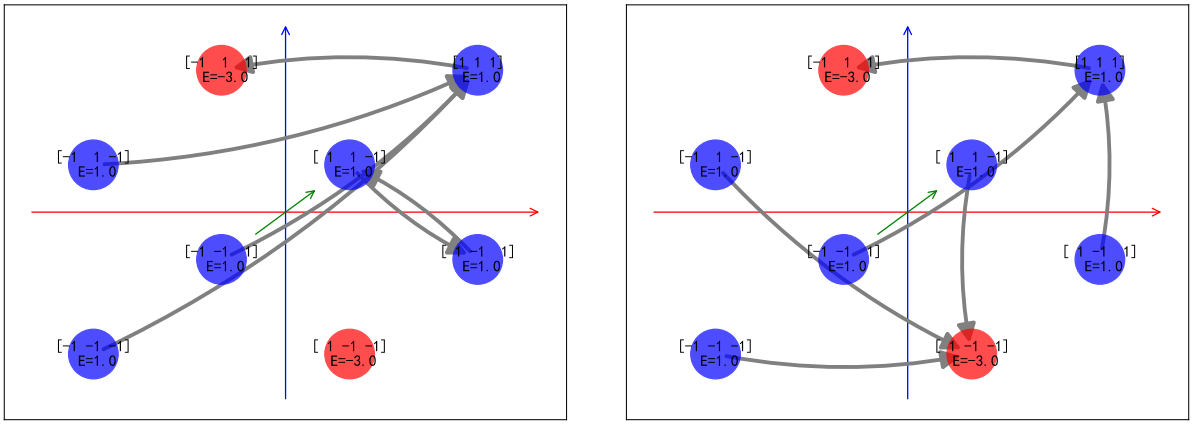
3.2　3 节点 Hopfield 网络

考虑构建一个 3 节点的 Hopfield 网络，并且训练它记住 [-1,1,1] 这个状态。于是其权重矩阵是

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

我们仍然用相图和能量方法分析这个 Hopfield 网络 可以看到，希望被记忆的状态和其反状态都成为了不动点，在同步更新策略下，相图出现了不对称性，而且出现了 [1,1,1] 和 [1,-1,1] 之间的震荡。不对称性实际上来源于我们选择恰好在阈值处的临界状态下让节点被激活，震荡也是同步更新策略的缺点。而在异步更新策略下，震荡又被解决了。

如果考虑 3 节点的 Hopfield 网络并训练它记住 [-1,1,1] 这个状态，结果在我的代码



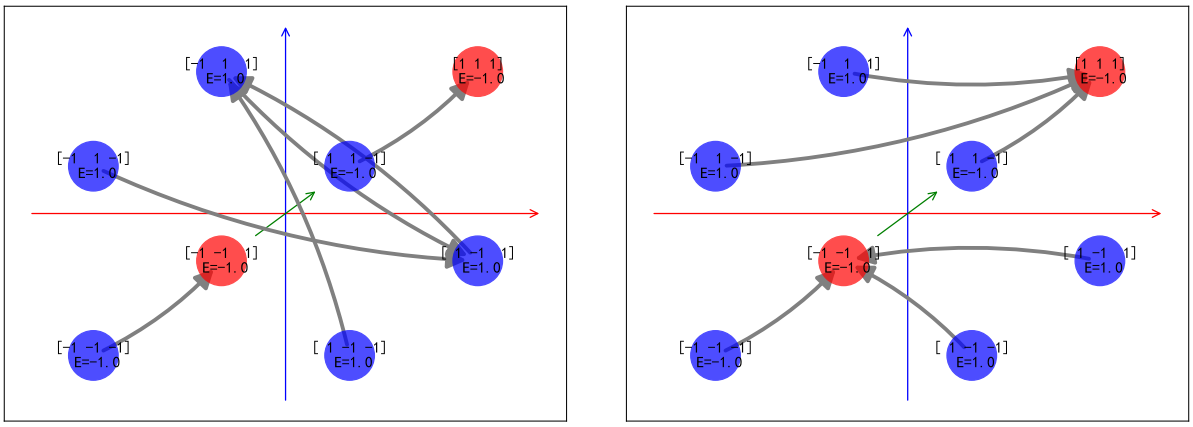
(a) 同步更新　　(b) 异步更新

图 9　3 节点记忆 [-1,1,1]Hopfield 网络，图中红色的节点表示稳定状态

文件中有展示。3 节点网络记住单个状态的其他情况都是上面着两种情况的轮换对称和“$\mathbb{Z}_2$”对称性。

接着我们考虑让 3 节点的 Hopfield 网络记住两个状态 [1,1,1],[1,1,-1]，其权重矩阵和相图（图10）为

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



(a) 同步更新　　(b) 异步更新

图 10　2 节点记忆 [1,1]Hopfield 网络，图中红色的节点表示稳定状态

可以看到此时不动点违背了正反成对出现的对称性，实际被记住的状态是 [1,1,1] 和 [-1,-1,1]，这也是阈值临界状态的处理造成的。它们的反状态同样具有最低能量，却并不是稳定的，这也是合理的，因为能量最小只是稳定的必要条件，而不是充分条件。

我的代码文件中，简单探索了 4 节点，10 节点以及 100 节点的 Hopfield 网络。并没有更有趣的结果了，所以这里不再详细讨论了。实际上，Hopfield 网络的节点数目越高，

记忆能力越好。

四、Hopfield 网络的图片记忆功能

Hopfield 网络的功能就是记忆并修复破损图片的能力，所以这里尝试了让 Hopfield 网络去记忆了一两张图片。验证了其功能。我们挑选了两张的图片，分别是当前十分流行的一个卡通形象以及我在一个研究所参观时所拍摄的照片。

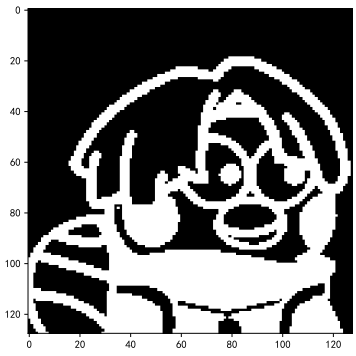
(a) 测试图片 1

(b) 测试图片 2

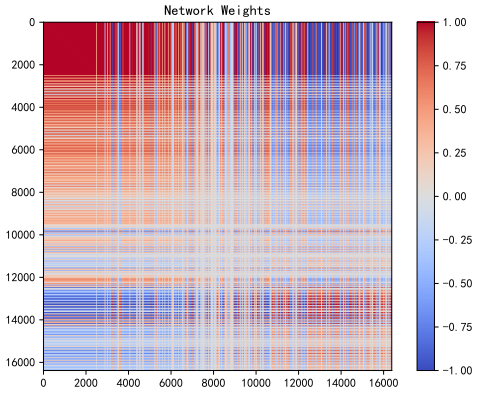
图 11　测试图片

4.1　记忆单张图片

考虑到本地环境和 Hopfield 网络本书的限制，我将测试图片 1 的像素调节为了 128×128，并且将其调整为黑白两色如图12a。



(a) 测试图片 1

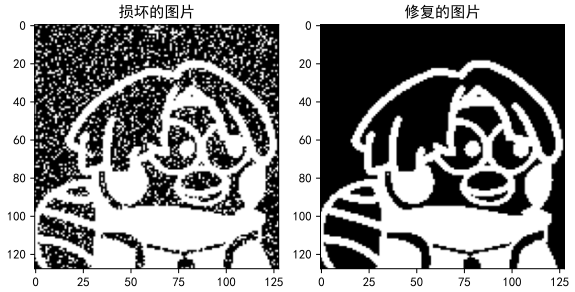


(b) 权重矩阵图

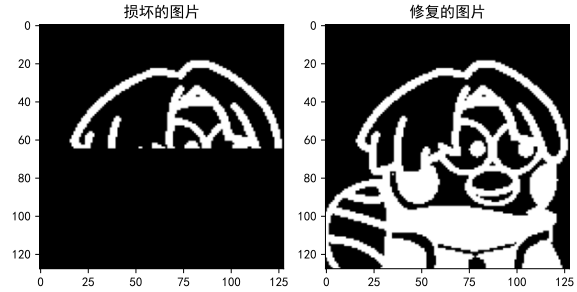
图 12　测试图片 1 产生 Hopfield 网络

存储该图片后，Hopfield 网络的权重矩阵则如图12b所示。

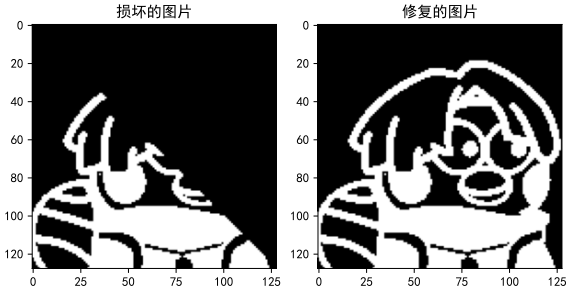
存储单张图片的回复性能较好，对于以下几种损坏方法都能修复，如图13



(a) 随机损坏



(b) 部分覆盖 1



(c) 部分覆盖 2

图 13　单张图片的 Hopfield 网络的恢复能力

考虑到 Hopfield 网络具有“$\mathbb{Z}_2$ 对称性”，如果将图片反色处理，仍然是稳定的演化状态。如图14

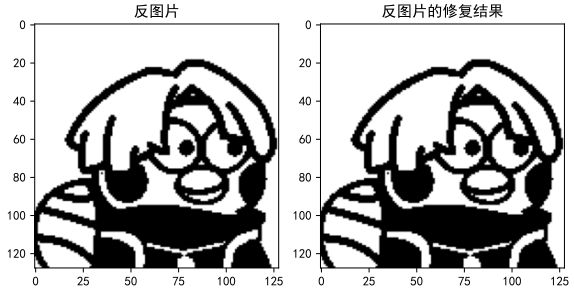


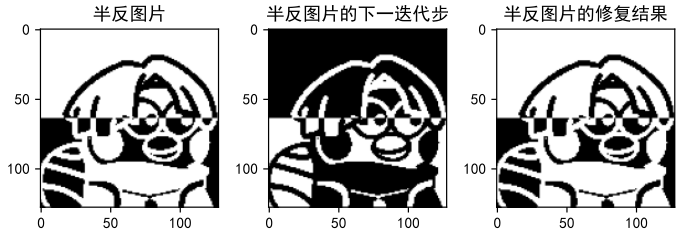
图 14　全反色处理

如图15，考虑到 Hopfield 网络具有“$\mathbb{Z}_2$ 对称性”，将图片中的一部分修改为反色。这种情况下，如果采用同步更新策略，Hopfield 网络的无法演化到稳定值，似乎会陷入到震荡的情况。如果采用异步更新策略，Hopfield 网络的就能克服这些困难。

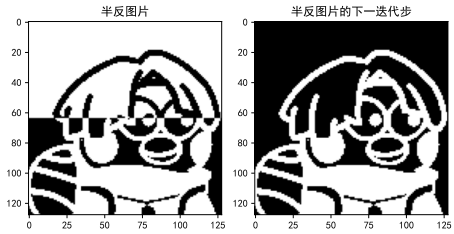
4.2　记忆多张图片

考虑记忆多张图片，为了提高代码的效率，将图片的像素调整为 64×64，所选择的两张图片如图16a所示

我们对图片做不同处理，图片的修复情况如下17

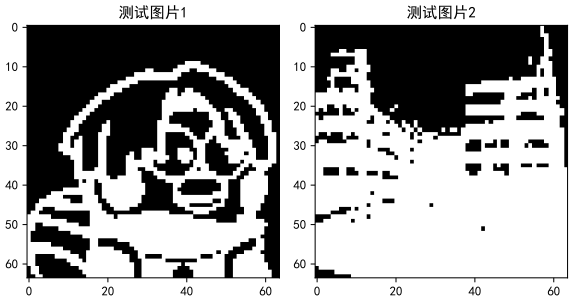


(a) 同步更新策略（迭代不稳定）

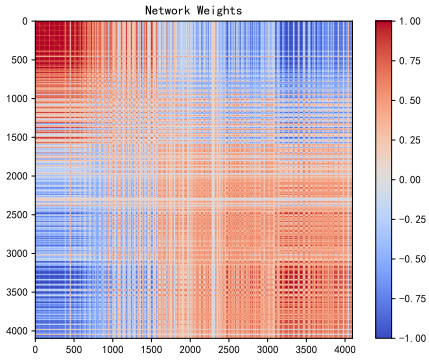


(b) 异步更新策略（修复成功）

图 15　半反色处理



(a) 记忆两张测试图片



(b) 该 Hopfield 网络的权重矩阵

图 16　记忆两张测试图片的 Hopfield 网络

可以看出，将测试图片 2 进行大部分遮挡时，图片会被修复称测试图片 1，这某种程度上展示了 Hopfield 网络的局限性。

五、结论

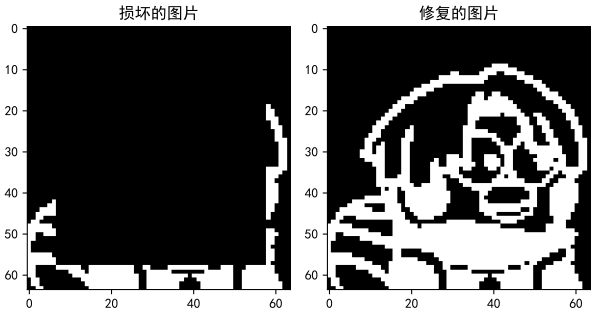
我探究了小规模 Hopfield 网络的相图，发现 Hopfield 网络的确可以使用吸引子产生“记忆”，Hopfield 网络具有一定程度的”$\mathbb{Z}_2$” 对称性，但是由于临界阈值的处理，对称性不是很完美。当然我并没有探究阈值的影响，如果改变阈值，应该会直接破坏掉对称性，这也和伊辛模型中外磁场造成自发对称性破缺有关。

在 Hopfield 网络的迭代方法中，如果采用同步更新策略，很容易陷入到震荡的现象中。如果采用异步更新测类，总能解决这个问题。

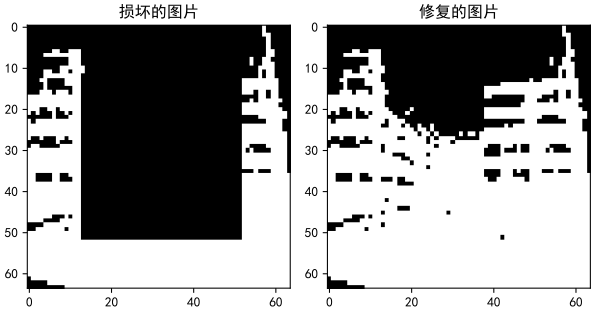
最后，我们验证了 Hopfield 网络记忆图片的功能，也展示了 Hopfield 网络的局限性。

六、参考文献

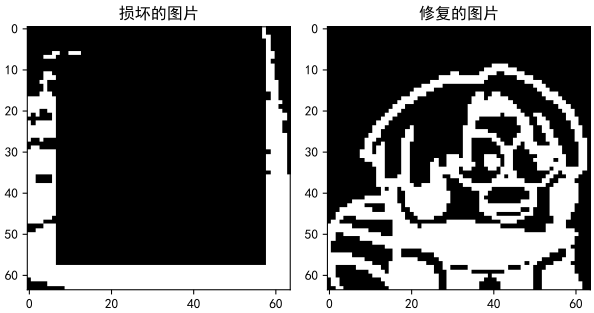
[1] HOPFIELD J J. Neural networks and physical systems with emergent collective computational abilities.[J/OL]. Proceedings of the National Academy of Sciences, 1982, 79(8): 2554-2558. https://www.pnas.org/doi/abs/10.1073/pnas.79.8.2554.



(a) 部分覆盖 1



(b) 部分覆盖 2



(c) 部分覆盖 3

图 17　半反色处理

[2] Wikipedia contributors. Hopfield network[EB/OL]. 2023. https://en.wikipedia.org/w/index.php?title=Hopfield_network&oldid=1186576971.

[3] Yosuke Katada. Hopfield network[EB/OL]. 2014. https://github.com/yosukekatada/Hopfield_network.

[4] takyamamoto. Hopfield-network[EB/OL]. 2018. https://github.com/takyamamoto/Hopfield-Network.

[5] OPENCOURSEWARE M. Introduction to neural computation[EB/OL]. Massachusetts Institute of Technology: MIT OpenCouseWare, 2018. https://ocw.mit.edu/courses/9-40-introduction-to-neural-computation-spring-2018/resources/20/.