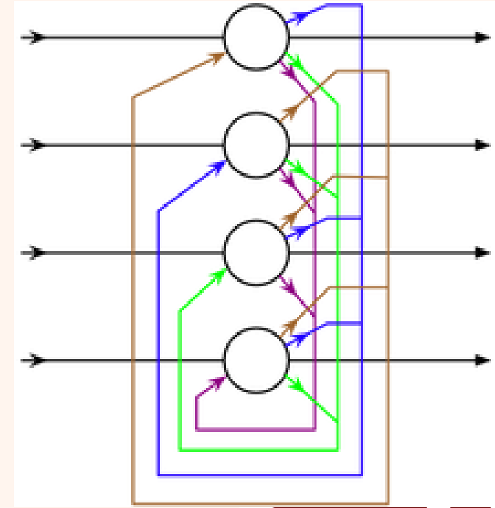
شبکه‌های عصبی مصنوعی

۱۳-۱۱-۷۱۳-۰۱

بخش هفتم



Hopfield

دانشگاه شهید بهشتی

دانشکده‌ی علوم و مهندسی کامپیوتر

بهار ۱۳۹۵

احمد محمودی ازناوه

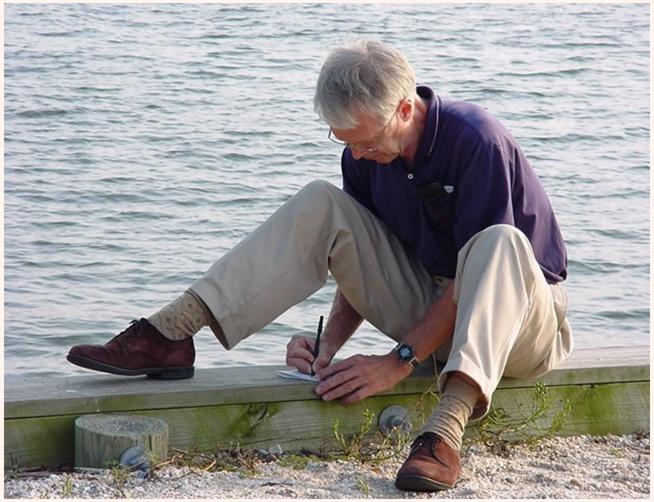
http://faculties.sbu.ac.ir/~a_mahmoudi/

فهرست مطالب

- یک مثال
- حافظه‌های تداعی‌گر
- شبکه‌ی Hopfiled دودویی
- یادگیری(به خاطر سپردن)
- همگرایی
  - تابع انرژی
- ظرفیت
- چند مثال

پیش‌گفتار

- در واقع هاپفیلد نشان داد که برخی مسائل محاسباتی، توسط مدل سیستم‌های فیزیکی قابل حل هستند.



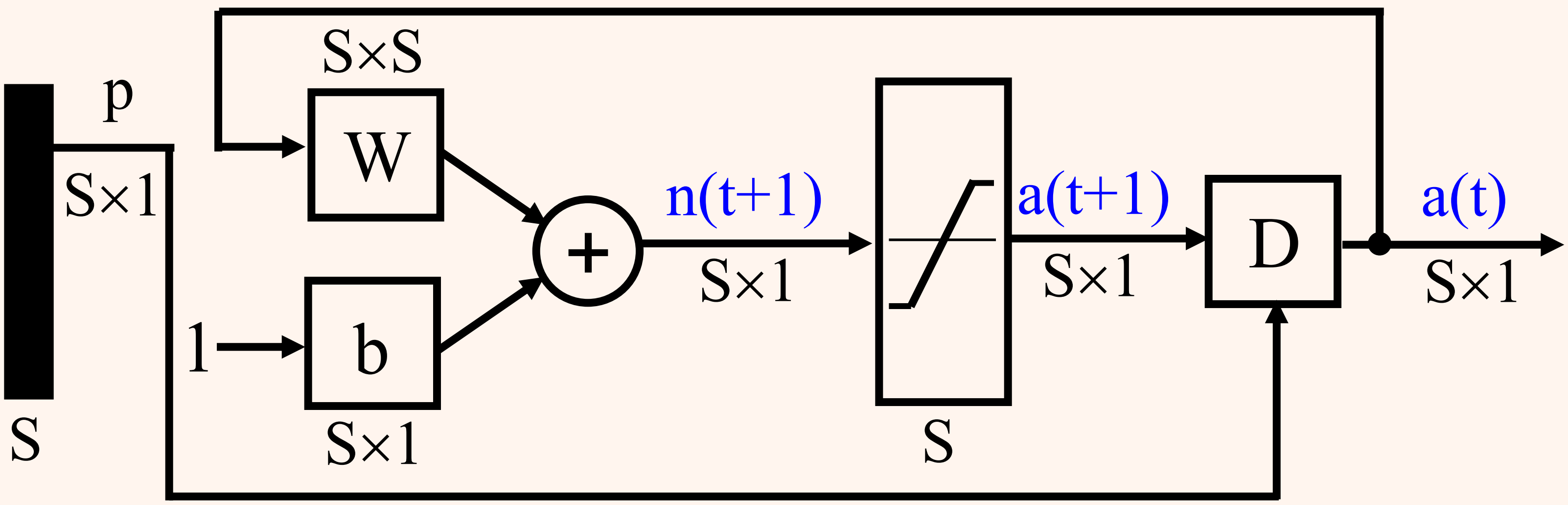
John Hopfield

Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities." <u>Proceedings of the national academy of sciences 79(8): 2554-2558.</u>

http://www.genomics.princeton.edu/hopfield/Index.html

Hopfield Network

معرفی



$\mathbf{n}(t+1) = \mathbf{W}\mathbf{a}(t) + \mathbf{b}, \quad \mathbf{a}(t+1) = \text{satlins}[\mathbf{n}(t+1)], \; \mathbf{a}(0) = \mathbf{p}$

- در شبکه‌ی Hamming مقدار «**ناصفر**» در خروجی کلاس را مشخص می‌کند، در حالی که در شبکه‌ی Hopfiled بردار الگوی کلاس به عنوان خروجی داده می‌شود.

مثال سیب و پرتقال!

$$\mathbf{W}=\begin{bmatrix}0.2 & 0 & 0\\ 0 & 1.2 & 0\\ 0 & 0 & 0.2\end{bmatrix},\ \mathbf{b}=\begin{bmatrix}0.9\\ 0\\ -0.9\end{bmatrix}$$

$$\mathbf{a}(\boldsymbol{t}+1)=\text{satlins}\left(\mathbf{Wa}(\boldsymbol{t})+\mathbf{b}\right),$$

$$\mathbf{a}(t+1)=\text{satlins}\left(\begin{bmatrix}0.2 & 0 & 0\\ 0 & 1.2 & 0\\ 0 & 0 & 0.2\end{bmatrix}\mathbf{a}(t)+\begin{bmatrix}0.9\\ 0\\ -0.9\end{bmatrix}\right)$$

$$\mathbf{a}(0)=\begin{bmatrix}-1\\ -1\\ -1\end{bmatrix}\Rightarrow \mathbf{a}(1)=\begin{bmatrix}0.7\\ -1\\ -1\end{bmatrix}\Rightarrow \mathbf{a}(2)=\begin{bmatrix}1\\ -1\\ -1\end{bmatrix}\Rightarrow \mathbf{a}(3)=\begin{bmatrix}1\\ -1\\ -1\end{bmatrix}$$

Hebb 1949

- فرضیه‌ی مطرح شده توسط Hebb در حال حاضر بر روی تحقیقات عصب‌شناسی مؤثر است.
- این فرضیه پیشتر نیز به بیان‌های مختلف مطرح شده بود.

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased

شبکه‌های تداعی‌گر

Associative network

Content Addressable Memory

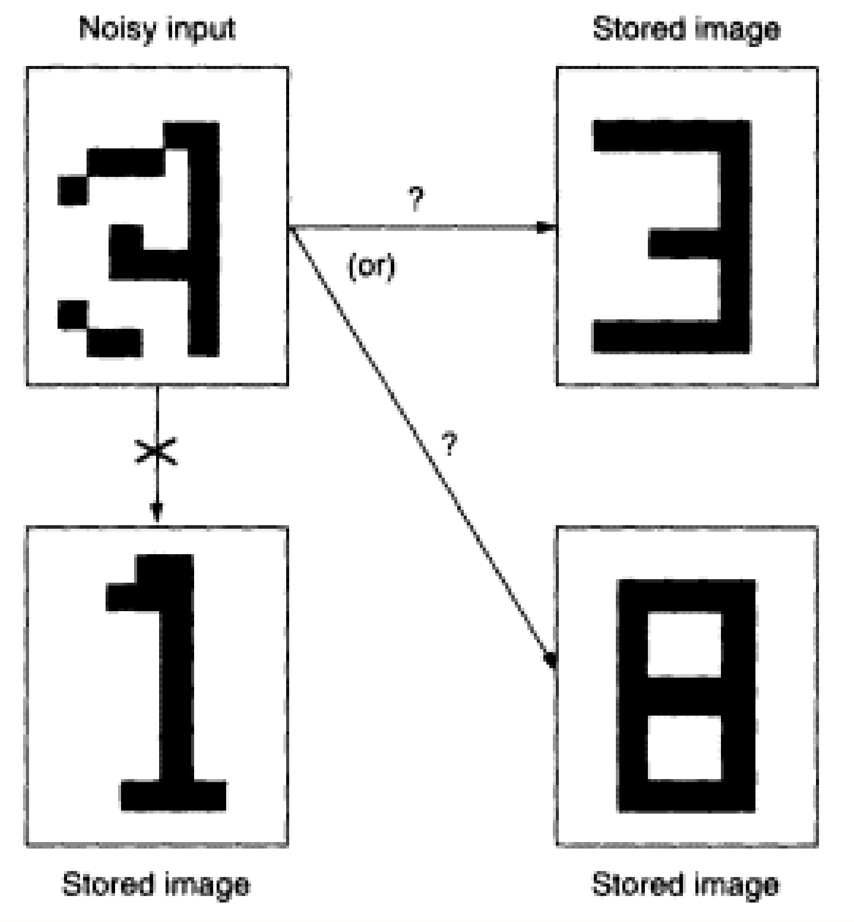
- این شبکه‌ها قابلیت به خاطر سپردن یک سری «**الگو**» را دارا هستند.
- این الگوها در اتصالات بین نورون‌ها ذخیره می‌شود، درست مانند آنچه در مغز انسان رخ

Hetero-association

- بردارهای ورودی و خروجی در دو فضای متفاوت هستند. (ترجمه یک لغت از یک زبان به زبان دیگر)
- بردارهای ورودی و خروجی هر دو در یک فضا هستند. (کاربرد: شناسایی کاراکتر)

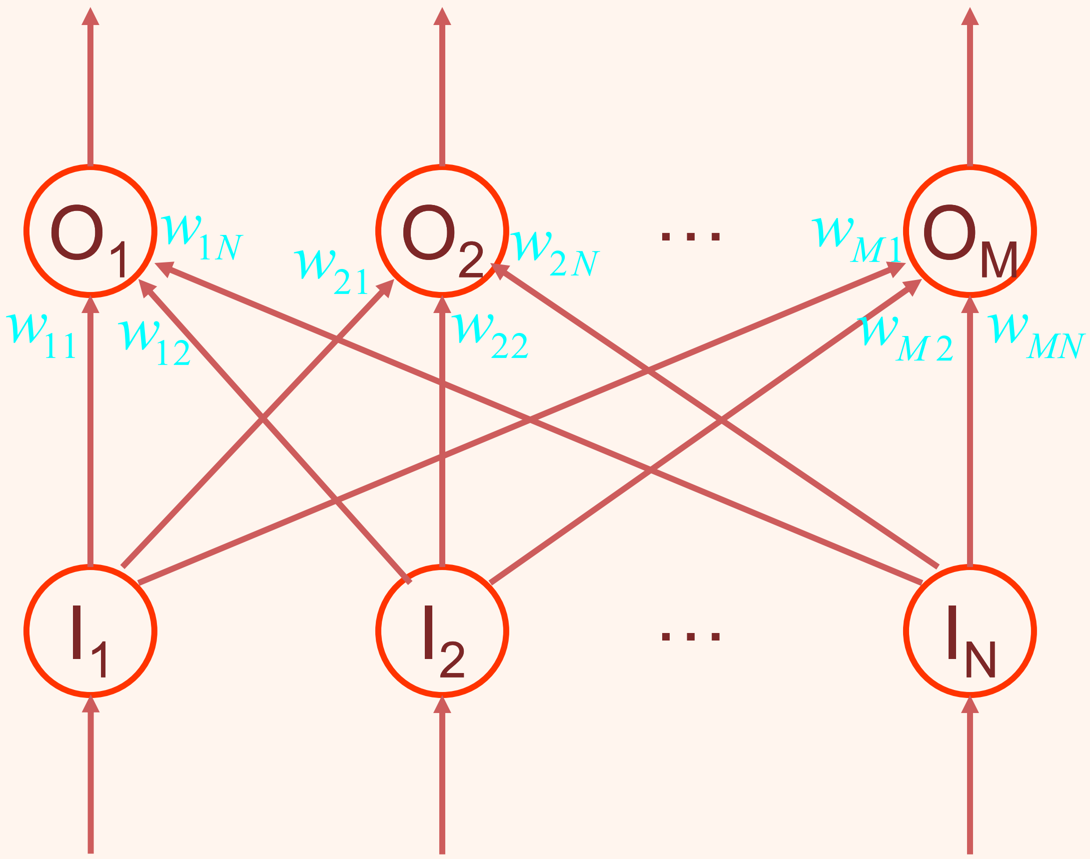
Auto-association

Auto-association



Interpolative Associative Memory

Non-iterative (One shot)



Memory Attractor

$$o_m = \sum_{n=1}^{N} w_{mn} i_n \quad \text{for} \quad m = 1, \ldots, M$$

Interpolative Associative Memory

- در صورتی که بردارهای ورودی «متعامد یکه» باشند، شبکه‌ی تک لایه‌ی معرفی شده به راحتی و حتی بدون آموزش می‌تواند به خوبی از عهده‌ی پیاده‌سازی نقش یک حافظه‌ی تداعی‌گر برآید.
- چنین شبکه‌ای «interpolative associative memory» نامیده می‌شود.

$$o_m = \sum_{n=1}^{N} w_{mn} i_n \quad \text{for} \quad m = 1, \ldots, M$$

محدودیتی که وجود دارد، یک بردار N بعدی نمی‌تواند بیش از N بردار متعامد یکه و در نتیجه N حافظه داشته باشد.

Interpolative Associative Memory

$$o_m = \sum_{n=1}^{N} w_{mn} i_n \quad \text{for} \quad m = 1, \ldots, M$$

$$\begin{bmatrix} o_1 \\ o_2 \\ \ldots \\ o_M \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1N} \\ w_{21} & w_{22} & & w_{2N} \\ \ldots & & \ldots & \ldots \\ w_{M1} & w_{M2} & \ldots & w_{MN} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ \ldots \\ i_N \end{bmatrix} \quad \text{or} \quad o = W\, \boldsymbol{i}$$

$$\mathbf{o}_m = \sum_{n=1}^{N} w_{mn} i_n \quad \text{for} \quad m = 1, \ldots, M$$

$$\mathbf{W} = \sum_{p=1}^{P} \mathbf{y}_p \mathbf{x}_p^t$$

matrix associative memory

مثال

$$\left\{ \left(\begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} \right), \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \right), \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 5 \\ 2 \\ 8 \end{bmatrix} \right) \right\}$$

$$\boxed{W = \sum_{p=1}^{P} y_p x_p^t}$$

$$W = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 5 \\ 2 \\ 8 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$W = \begin{bmatrix} 0 & -3 & 0 \\ 0 & -3 & 0 \\ 0 & -3 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 0 & 0 & 2 \\ 0 & 0 & 8 \end{bmatrix} = \begin{bmatrix} 1 & -3 & 5 \\ 2 & -3 & 2 \\ 3 & -3 & 8 \end{bmatrix}$$

مثال

$$\left\{ \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right), \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right) \right\}$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} =$$

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

در این مثال داده‌ها دودویی هستند، تنها شامل {1,-1}، از این رو تابع فعالیت را می‌توان تابع علامت در نظر گرفت.

W = (matrix whose columns are output vectors) × (matrix whose rows are input vectors)

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} -2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

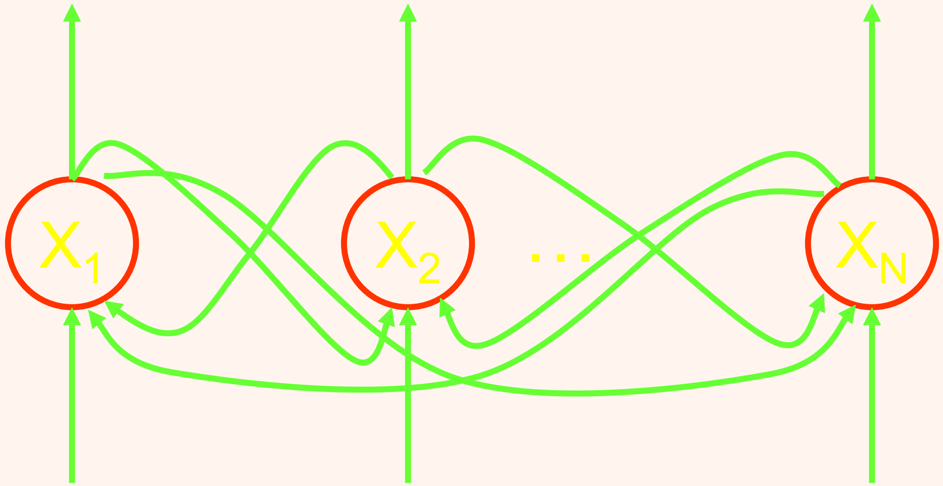
$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} \Rightarrow \begin{bmatrix} -2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Interpolative Associative Memory

- در صورتی که هدف ساختن یک تابع خطی $R^N \rightarrow R^M$ باشد و بردار نمونه‌ی متعامد یکه در اختیار داشته باشیم، این شیوه بهترین راه حل است.
- به آموزش احتیاج نداشته، تطبیق خوبی دارد و برای بردارهای جدید درون‌یابی دقیقی(خطی) خواهد داشت.
- برای کاربردهایی نظیر دسته‌بندی، در برابر خطا تحمل‌پذیر نیست.

شبکه‌ی Hopfield

- شبکه‌ی Hopfield یک شبکه‌ی بازگشتی تک‌لایه است.
- مانند شبکه‌ی قبلی، به جای آموزش وزن‌ها مقداردهی می‌شوند.



شبکه‌ی Hopfield گسسته

- در این شبکه خروجی‌ها تنها دو مقدار ۱ و ۱- را می‌پذیرند.
- در این حالت تابع انگیزش تابع علامت در نظر گرفته می‌شود.
- برای ورودی-خروجی‌های مشخص، شبیه آن‌چه در حافظه‌ی تداعی‌گر گفته شد، می‌توان وزن‌ها را تعیین نمود:

$$\mathbf{W} = \sum_{p=1}^{P} y_p x_p^t \qquad w_{ij} = \sum_{p=1}^{P} y_p^{(i)} x_p^{(j)}$$

نحوه‌ی تعیین وزن‌ها در حافظه‌ی تداعی‌گر

شبکه‌ی Hopfield گسسته

- ورودی‌ها و خروجی‌ها تنها شامل ۱ و ۱– خواهند بود.
- خروجی در زمان t از روی خروجی زمان t-1 به دست می‌آید و این رابطه‌ی بازگشتی ادامه می‌یابد.

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

- با توجه به این که ورودی‌ها متعامد یکه نیستند، تضمینی وجود ندارد که به ازای ورودی x_p خروجی متناظر y_p به دست آید.

شبکه‌ی Hopfield گسسته (ادامه...)

auto-associators

- در صورتی که هر ورودی خودش را تداعی کند:
- $(x_1, x_1), (x_2, x_2), \ldots, (x_P, x_P)$
- مقدار اولیه‌ی وزن‌ها به صورت زیر به دست می‌آید:

$$w_{ij} \propto \sum_{p=1}^{P} x_p^{(i)} x_p^{(j)}$$

- در این حالت ماتریس وزن‌ها متقارن خواهد شد.
- حال اگر «المان‌های روی قطر اصلی» صفر در نظر گرفته شوند، به خاصیت جالبی خواهیم رسید:
  - طی حالت‌های محدود، می‌توان به پایداری رسید.

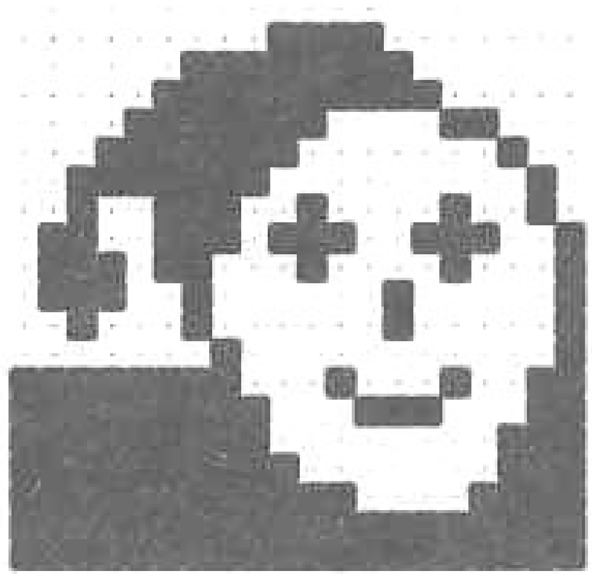
Self-excitation term

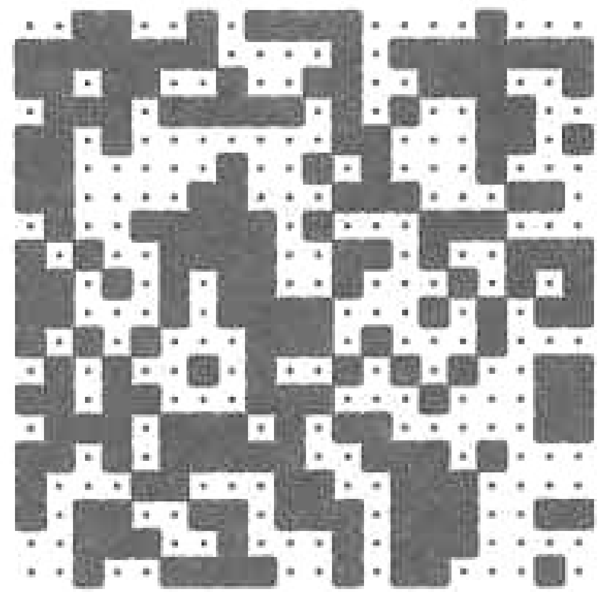
شبکه‌ی Hopfield گسسته (ادامه...)

- در این حالت شبکه الگوهای ورودی را تداعی می‌کند؛ به این معنا که در هر تکرار به سمت یکی از الگوها نزدیک‌تر خواهیم شد.
- در پایان در شبیه‌ترین حالت به ورودی اولیه متوقف خواهیم شد.
- این شبکه برای بازیابی داده‌های ناقص و یا همراه با نویز به کار می‌رود.

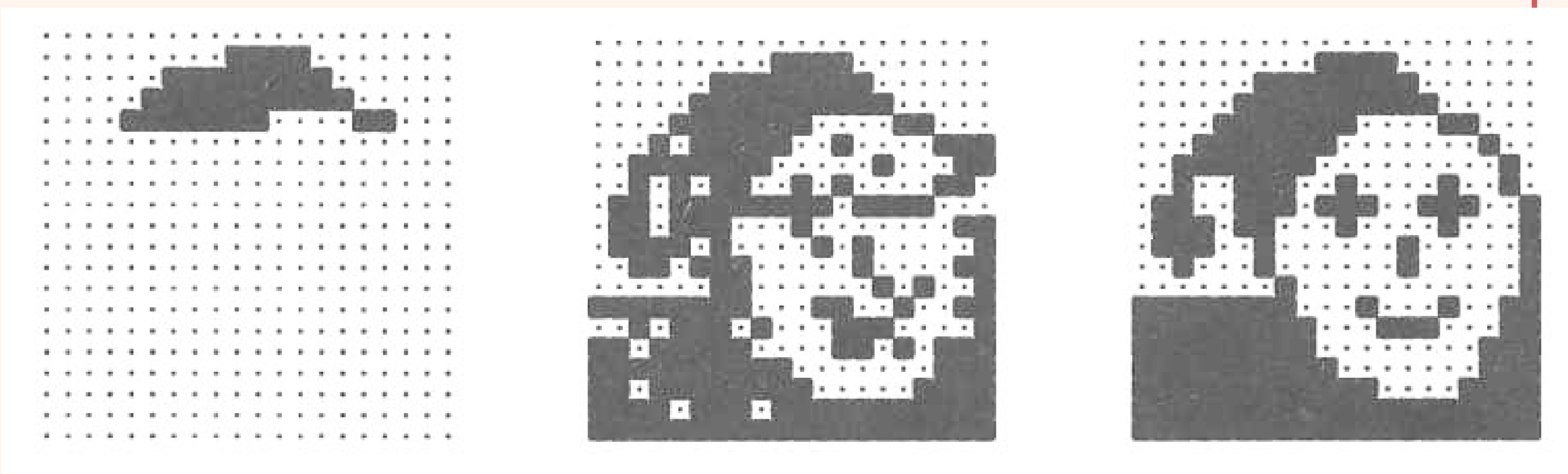
مثال

- یک شبکه‌ی با تصاویر ۲۰×۲۰ آموزش می‌بیند.
  - آموزش با بیست تصویر انجام می‌شود:

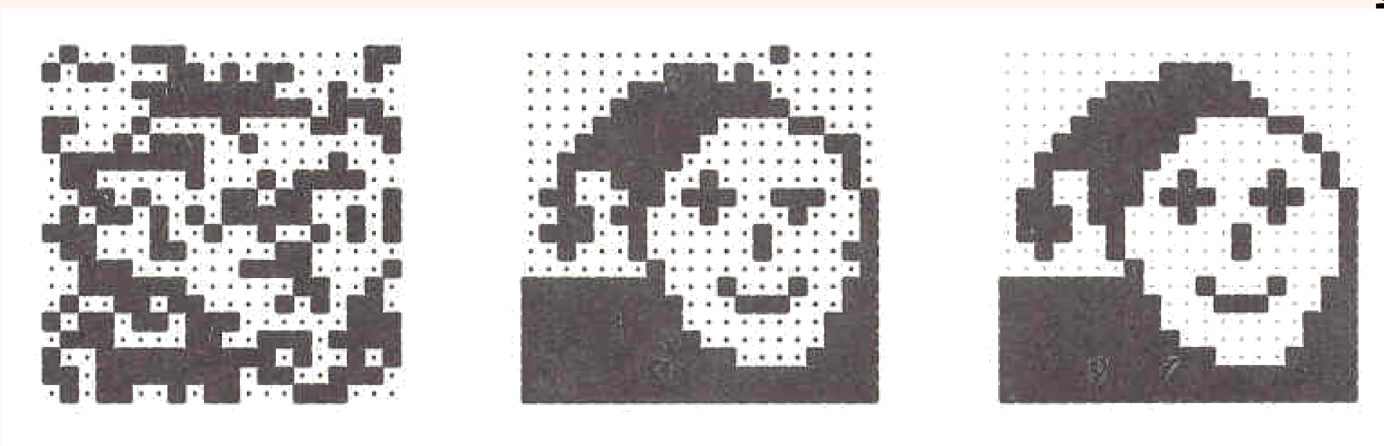




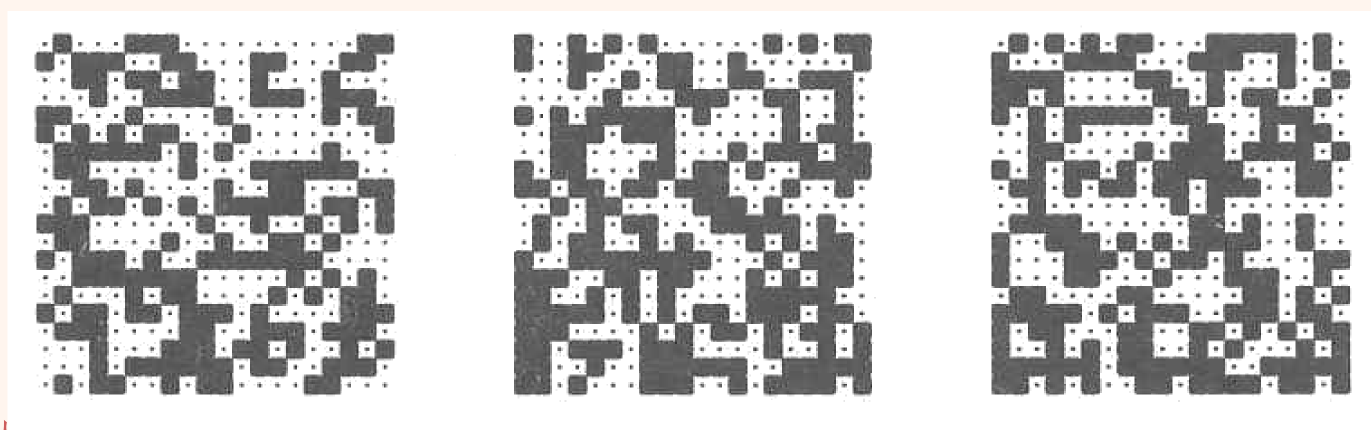
- با اعمال ورودی که شامل یک چهارم تصویر است، طی دو تکرار شبکه به خوبی از عهده‌ی بازسازی برخواهد آمد.



- با افزودن نویز باز هم بازیابی به خوبی انجام می‌ش



- با افزایش بیشتر نویز



مشکلات

- در صورتی که نمونه شیفت یافته باشد، این شبکه قادر به شناسایی نمونه نخواهد بود.
- تنها تعداد کمی داده‌ی بسیار متفاوت قابل حفظ است.

به‌روزرسانی گره‌ها

- به روزرسانی گره‌ها از قانون زیر پیروی می‌کند:

net input

$$x_{p,k}(t+1) = \operatorname{sgn}\left(\qquad \right)$$

$$I_{p,k}(t) = \begin{cases} \text{initial input, if } t = 0 \\ 0, \qquad\qquad \text{otherwise} \end{cases}$$

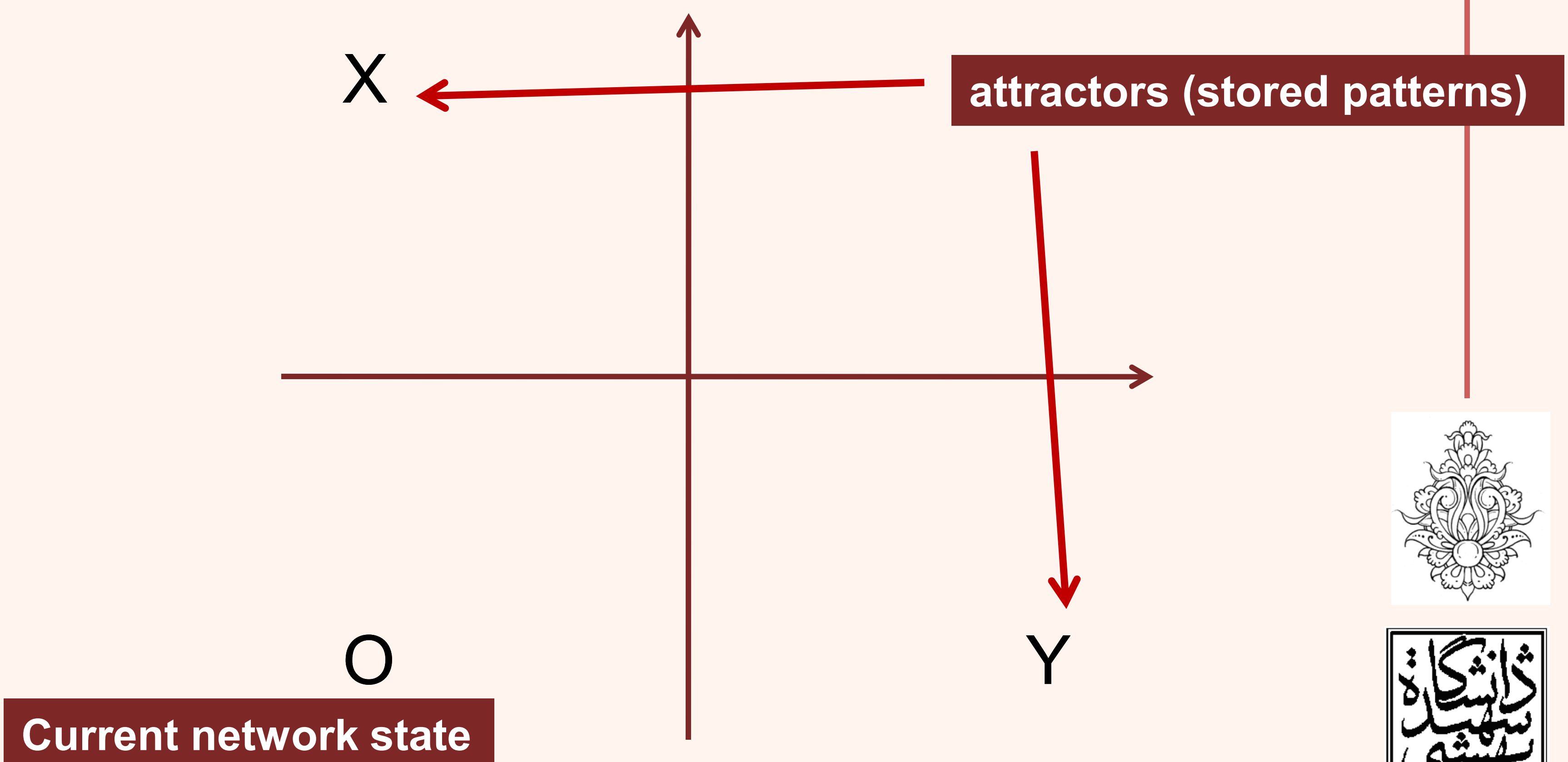
به‌روزرسانی گره‌ها

- به روزرسانی گره‌ها به دو شیوه‌ی انجام می‌شود:
  - همگام: **Synchronous**
    تمام گره‌ها به صورت همزمان به روز می‌شوند.
  - در شیوه‌ی همگام ممکن است هیچ‌گاه به پایداری نرسیم.
  - ناهمگام: **Asynchronous**
    در هر لحظه تنها یک گره به روز می‌شود.

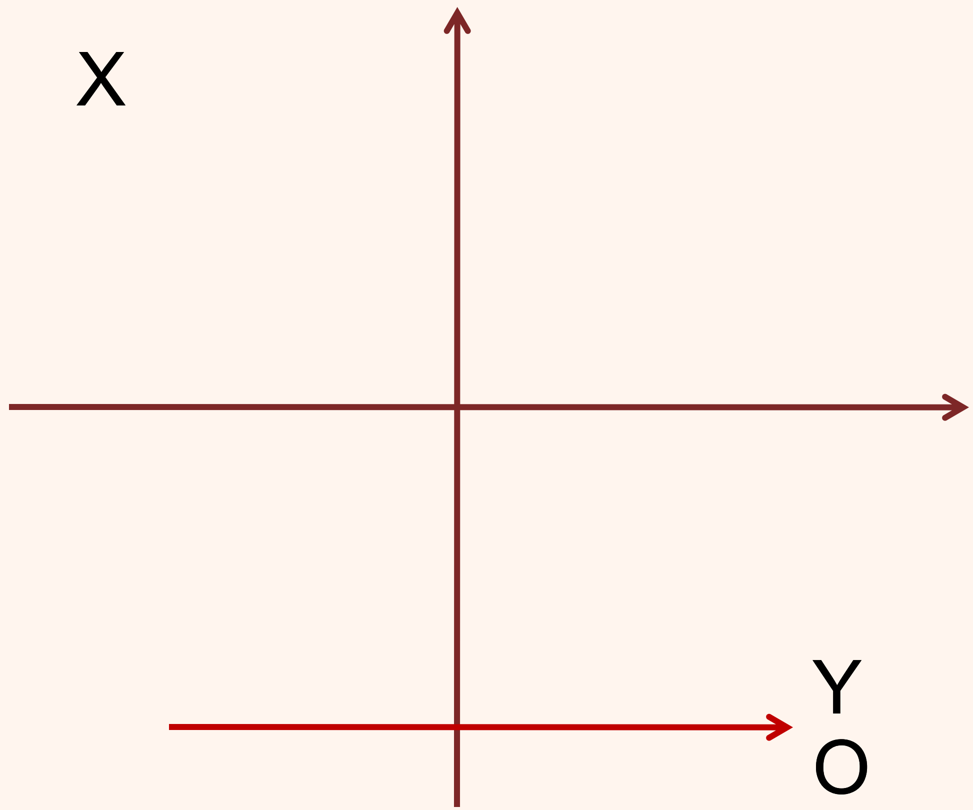
$$x_{p,l}(t+1)=\begin{cases} x_{p,l}(t+1) & \text{if } l\neq k \\ \operatorname{sgn}\left(\sum_{j=1}^{n} w_{l,j}x_{p,j}(t)+I_{p,l}(t)\right) & \text{if } l=k \end{cases}$$

  - باید برای همه‌ی گره‌های شانس مساوی در تغییر حالت در نظر گرفت. **Fairness**

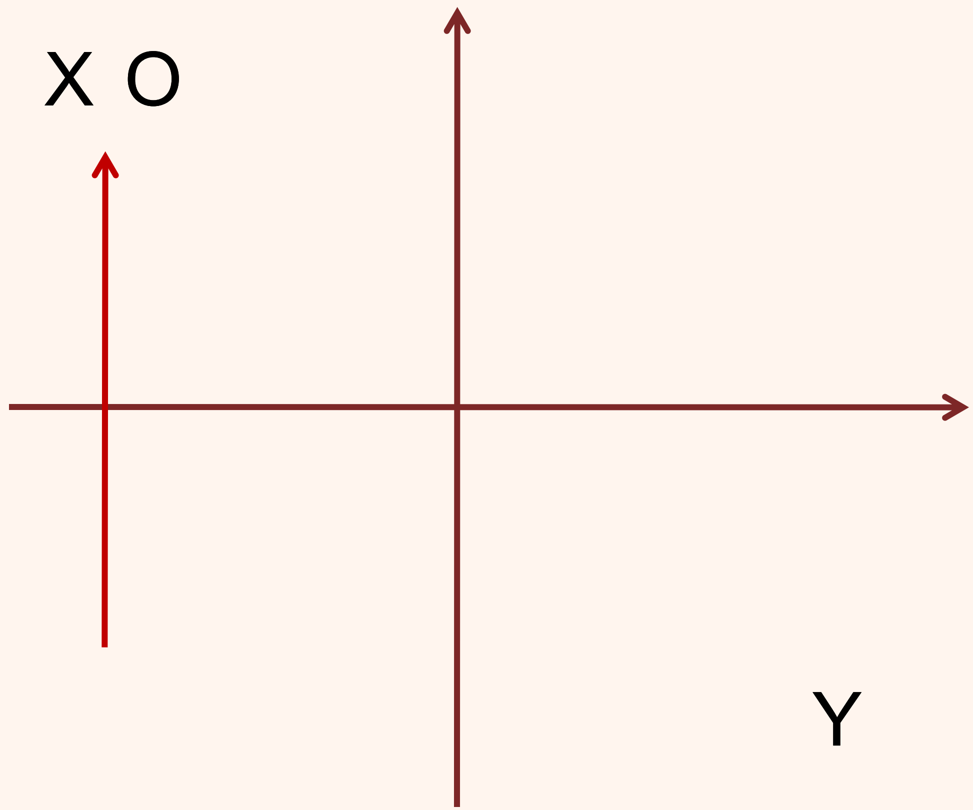
به‌روزرسانی به شیوه‌ی ناهمگام



به‌روزرسانی به شیوه‌ی ناهمگام

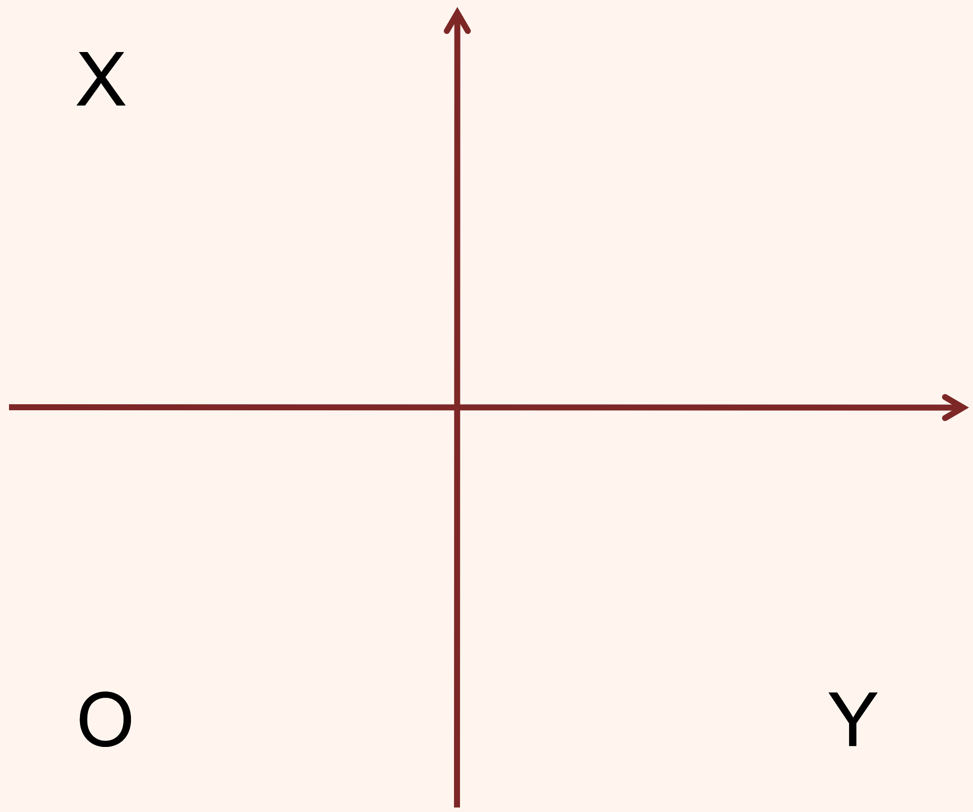


بەروزرسانی بە شیوەی ناهمگام



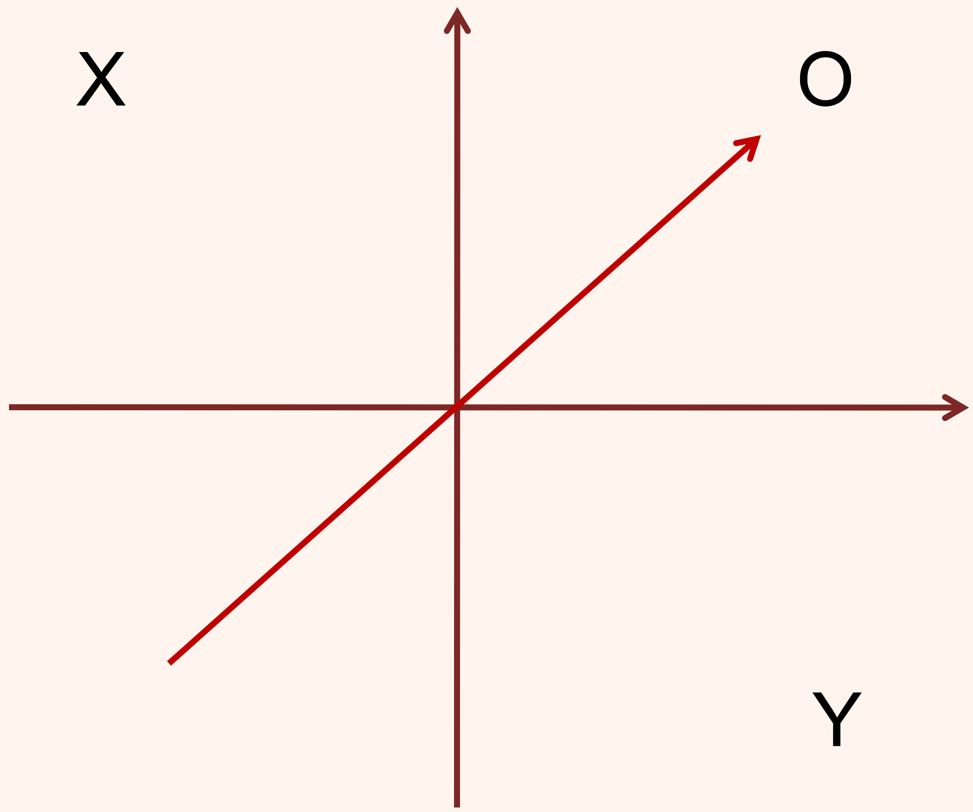
به‌روزرسانی به شیوه‌ی همگام

در این صورت در گام بعدی حالت شبکه چه خواهد بود؟



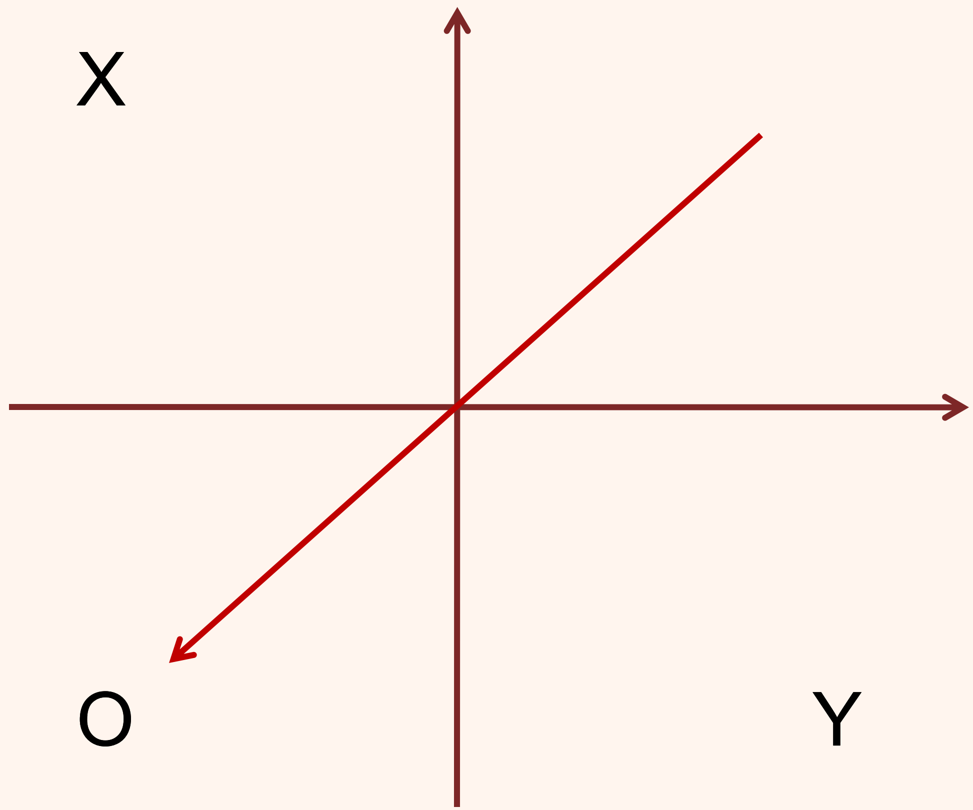
به‌روزرسانی به شیوه‌ی همگام

گام بعدی؟



به‌روزرسانی به شیوه‌ی همگام

ممکن است شبکه بین این دو حالت نوسان کند!



مثال

- در یک شبکه Hopfiled دو حالت داریم:

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

- در مورد وزن‌ها چه می‌توان گفت؟

$$w_{l,j} = \begin{cases} 1 & l \neq j \\ 0 & l = j \end{cases}$$

- در صورتی که ورودی‌هایی همراه با نویز و یا ناشناخته اعمال شود، چه؟

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

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

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

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

- استفاده از هاپفیلد برای ذخیره‌سازی یک سری بردار
- لازم است بردارهای مورد نظر به عنوان نقاط تعادل شبکه تعریف شوند که در این نقاط شبکه به کمینه‌ی انرژی خود می‌رسد.
- فرض کنید چند نقطه‌ی تعادل داریم به عنوان مثال می‌خواهیم P^s نقطه‌ی تعادل جدید شبکه باشد.

$$P^S = [P_1^s, P_2^s, \ldots\ldots, P_N^s]$$

- تابع را به ازای P^s بازنویس می‌کنیم.

اضافه کردن نقطه‌ی تعادل

- بناست تابع در نقطه‌ی Ps به تعادل برسد.

w_{ij}

وزن قدیم که Ps جزیی از نقاط تعادل نبود

$\overline{w_{ij}}$

وزن جدید که Ps جزیی از نقاط تعادل است

$$\overline{w_{ij}} = w_{ij} + w_{ij}^{S}$$

وزن تحمیلی به ازای وارد شدن Ps

- اگر $P_i^s = \pm 1$ در نظر گرفته شود برای ماکزیمم شدن:

$$w_{ij}^s = P_i^s P_j^s$$

- برای ذخیره سازی K الگو باید

$$\begin{cases} w_{ij} = \sum_{s=1}^{k} P_i^s P_j^s, & i \neq j \\ w_{ii} = 0, & i = j \end{cases}$$

مثال

- می‌خواهیم الگوی p را ذخیره کنیم و پس از آن پایداری شبکه را بررسی نماییم.

$$p = [1 \quad -1 \quad 1 \quad -1]^T$$

$$w = p^T . p = \begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{bmatrix}$$

مثال

$$p = \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}^T$$

$$p \rightarrow wp^T = \begin{bmatrix} 3 & -3 & 3 & -3 \end{bmatrix}^T \xrightarrow{sign} \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}^T$$

پایدار

$$X(0) = p_1 = \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}^T$$

$$p_1 \rightarrow wp_1^T = \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix}^T \xrightarrow{sign} \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix}^T = O(0)$$

$$X(1) \rightarrow w.X(1)^T = \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}^T \xrightarrow{sign} \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}^T = O(1) = X(0)$$

نوسانی

$$X(0) = p_2 = \begin{bmatrix}1 & -1 & -1 & -1\end{bmatrix}^T$$

$$X(0) \rightarrow w.X(0)^T = \begin{bmatrix}1 & -1 & 3 & -1\end{bmatrix}^T \xrightarrow{sign} \begin{bmatrix}1 & -1 & 1 & -1\end{bmatrix}^T$$

به P همگرا شده است

$$X(0) = p_3 = \begin{bmatrix}-1 & 1 & -1 & 1\end{bmatrix}^T$$

$$X(0) \rightarrow w.X(0)^T = \begin{bmatrix}-3 & 3 & -3 & 3\end{bmatrix}^T \xrightarrow{sign} \begin{bmatrix}-1 & 1 & -1 & 1\end{bmatrix}^T$$

پایدار

تابع انرژی (هزینه)

- دیدیم که وزن‌ها در حالتی که P عنصر حافظه داشته باشیم، به صورت زیر تعیین می‌شوند:

$$w_{l,j} \propto \sum_{p=1}^{P} (i_{p,l} i_{p,j})$$

- به بیانی دیگر، اگر دو واحد اغلب موارد فعال(۱) یا غیر فعال(۱-) باشند، انتظار می‌رود با وزن بزرگ و مثبت به یکدیگر وصل شوند. در صورتی که با هم همخوانی نداشته باشند، وزن منفی با اندازه‌ی بزرگ انتظار می‌رود.

تابع انرژی (هزینه)

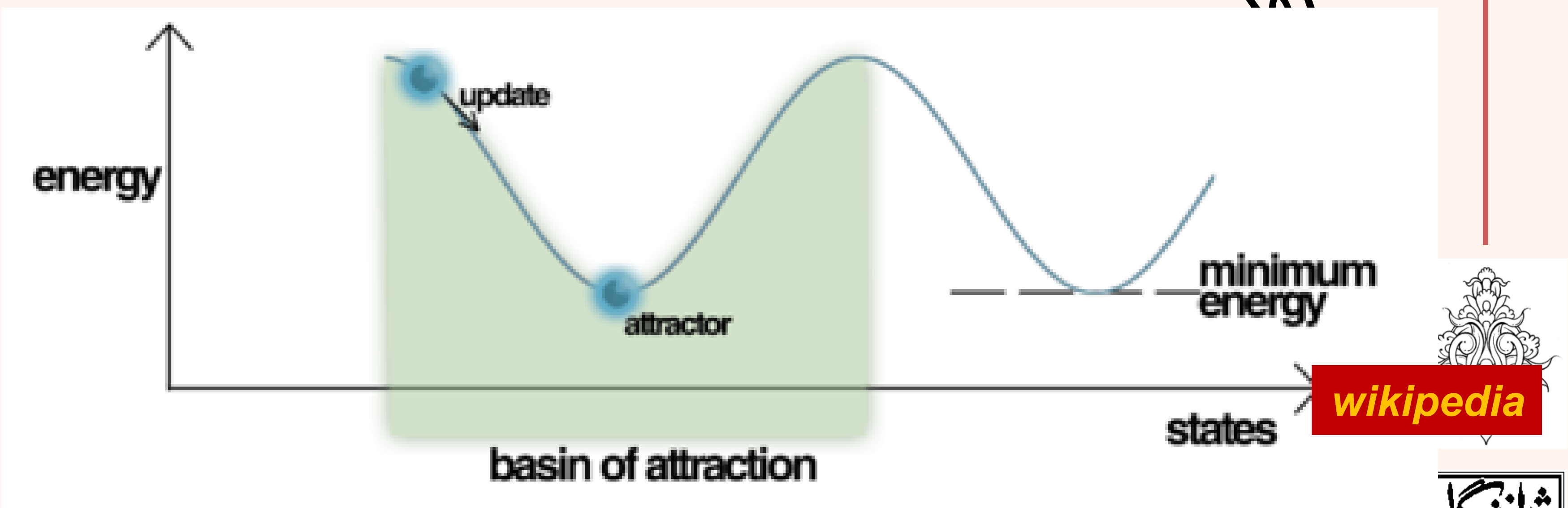
- با توجه به این که وزن‌ها بر اساس رابطه‌ی داده شده محاسبه می‌شود، انتظار می‌رود، تابع زیر به ازای نمونه‌های ذخیره شده؛ مقدار مثبت و بزرگی داشته باشد.

$$\sum_{l}\sum_{j} w_{l,j}\, i_{p,l}\, i_{p,j}$$

- همچنین برای ورودی‌های شبیه نیز مقداری بزرگ و مثبت خواهد شد.
- هر چه شباهت کم‌تر باشد، این مقدار نیز کم‌تر خواهد شد.

تابع انرژی (هزینه)

- چنان‌چه در طی به روز شدن مقدار گره‌ها، تابع انرژی تعریف شده کمینه شود، به حالتی پایدار خواهیم رسید، که این حالت یک attractor خواهد بود.



$$-\sum_{l}\sum_{j} w_{l,j}\, i_{p,l}\, i_{p,j}$$

تابع انرژی (هزینه)

- البته هدف این نیست که تنها به یک attractor برسیم، بلکه می‌خواهیم به حالتی برسیم که شبیه‌ترین حالت به ورودی اولیه باشد.
- برای رسیدن به این هدف، تابع انرژی به صورت زیر اصلاح می‌شود:

$$E = -a\sum_{l}\sum_{j} w_{l,j}\, x_l\, x_j - b\sum_{l} I_l\, x_l$$

Lyapunov function

- در واقع جزیی به تابع انرژی می‌افزاییم که در صورت دور شدن از ورودی اولیه انرژی افزایش یابد.

تابع انرژی (هزینه)

- تابع هزینه‌ی به دست آمده در طی فرآیند به‌روزرسانی به صورت زیر تغییر می‌کند:

$$\Delta E(t) = E(t+1) - E(t)$$

$$\Delta E(t) = -a\sum_{l}\sum_{j\neq l} w_{l,j}[x_l(t+1)x_j(t+1) - x_l(t)x_j(t)] - b\sum_{l} I_l[x_l(t+1) - x_l(t)]$$

- در صورتی که تنها واحد kام به روز شود:

$x_j(t+1)=x_j(t)$ for every node $j\neq k$:

$$\Delta E(t) = -a\sum_{j\neq k}[(w_{k,j}+w_{j,k})(x_k(t+1)-x_k(t))x_j(t)] - bI_k[x_k(t+1)-x_k(t)]$$

$$\Delta E(t) = -\left[a\sum_{j\neq k}(w_{k,j}+w_{j,k})x_j(t)] + bI_k\right](x_k(t+1)-x_k(t))$$

تابع انرژی (هزینه)

- با توجه به این که وزن‌ها متقارن است:

$$\Delta E(t) = -\left[\sum_{j \neq k} w_{j,k} x_j(t) + I_k\right](x_k(t+1) - x_k(t))$$

a = 0.5 and b = 1

$$\Delta E(t) = -\text{net}_k(t)\Delta x_k(t)$$

- یعنی حالت گره‌ی kام در صورتی عوض خواهد شد:

$$\text{net}_k(t)\Delta x_k(t) > 0$$

- و وزن‌ها هم از رابطه‌ی زیر به دست خواهد آمد:

$$w_{j,l} = \frac{1}{P}\sum_{p=1}^{P}(i_{p,l}i_{p,j})$$

مثال

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

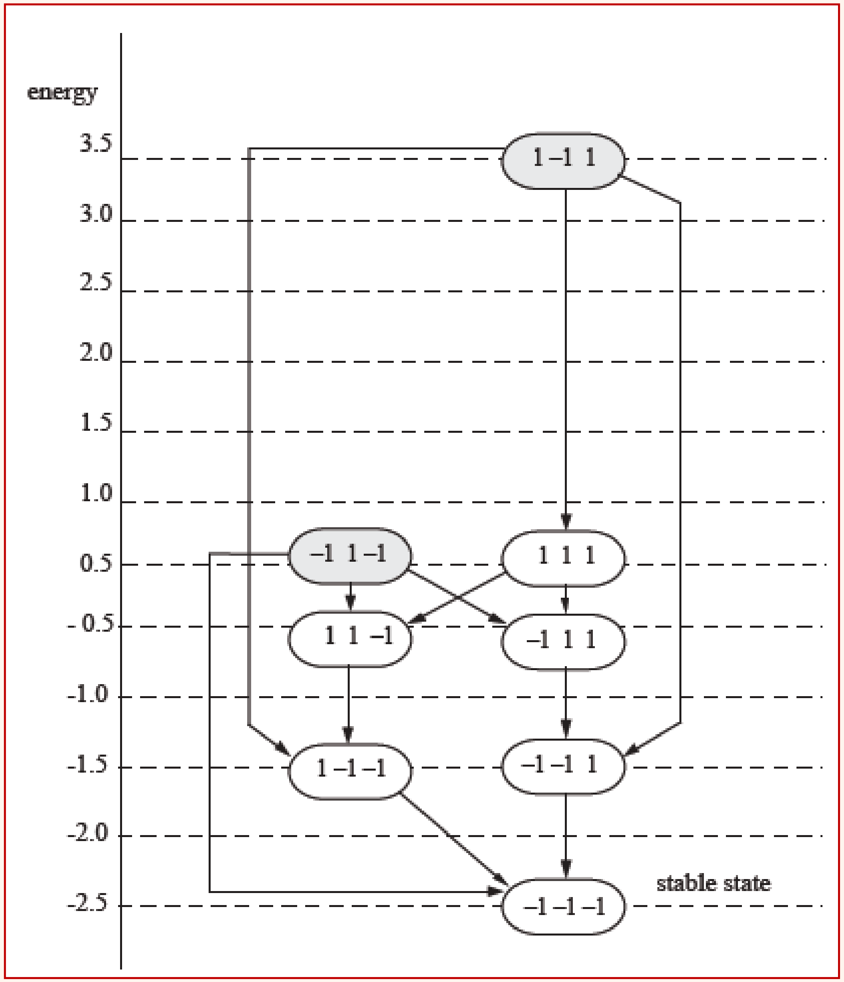
$$\frac{1}{3}\left(\begin{bmatrix}1 & 1 & -1\\ 1 & 1 & -1\\ -1 & 1 & 1\\ -1 & 1 & 1\end{bmatrix}\times\begin{bmatrix}1 & 1 & -1 & -1\\ 1 & 1 & 1 & 1\\ -1 & -1 & 1 & 1\end{bmatrix}\right) \xrightarrow{\text{با صفر قرار دادن عناصر قطر اصلی}} \begin{bmatrix}0 & 1 & -1/3 & -1/3\\ 1 & 0 & -1/3 & -1/3\\ -1/3 & -1/3 & 0 & 1\\ -1/3 & -1/3 & 1 & 0\end{bmatrix}$$

- در این مثال چنان‌چه مشاهده می‌شود:
- دو گره‌ی اول همبستگی کاملی با یکدیگر دارند، از این جهت $w_{1,2}=1$ در نظر گرفته می‌شود.
- در مورد گره‌ی سوم و چهارم هم می‌توان چنین گفت: $w_{3,4}=1$
- در مورد گره‌ی اول و سوم چه می‌توان گفت؟
  - در یک مورد هماهنگ و در دو مورد ناهماهنگ هستند:

$$w_{1,3}=\frac{(\text{number of agreements})-(\text{number of disagreements})}{(\text{number of patterns})}=\frac{1-2}{3}=-1/3$$

پایداری و تابع انرژی

- بدین‌ترتیب با اعمال یک ورودی شبکه تاجایی پیش می‌رود که انرژی آن مینیمم شود.



http://page.mi.fu-berlin.de/rojas/neural/

ظرفیت شبکه‌ی هاپفلید

- ظرفیت (حافظه) بستگی به تعداد گره‌های خروجی دارد. یک تقریب برای حد بالای ظرفیت در صورتی که تعداد گره‌های خروجی n باشد:

$$\boxed{\frac{n}{4\ln n}}$$

Amit, D. J. (1992). Modeling Brain Function: The World of Attractor Neural Networks, Cambridge University Press.

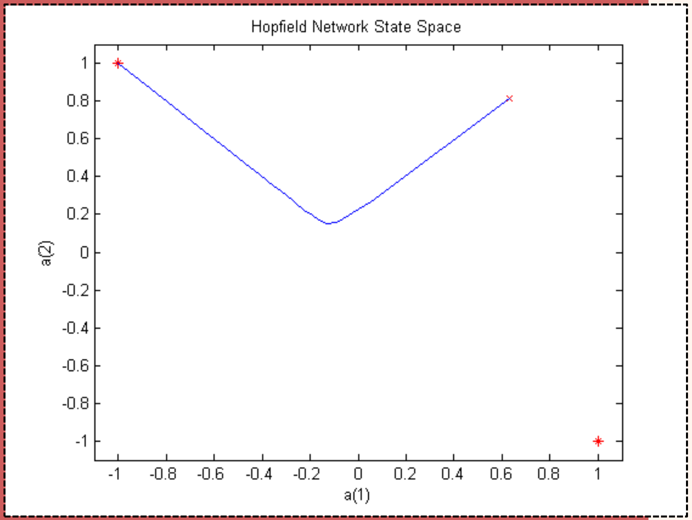
- حد بالای به دست آمده در بالا، براساس احتمال بازیابی درست تک‌تک بیت‌ها به دست آمده است و خاصیت اصلاح تدریجی الگو را لحاظ نکرده است، تقریب واقعی‌تر به صورت زیر می‌باشد.

$$\boxed{.138n}$$

Hertz, J., A. Krogh, et al. (1991). Introduction to the Theory of Neural Computation, Addison-Wesley.

مثال ۱

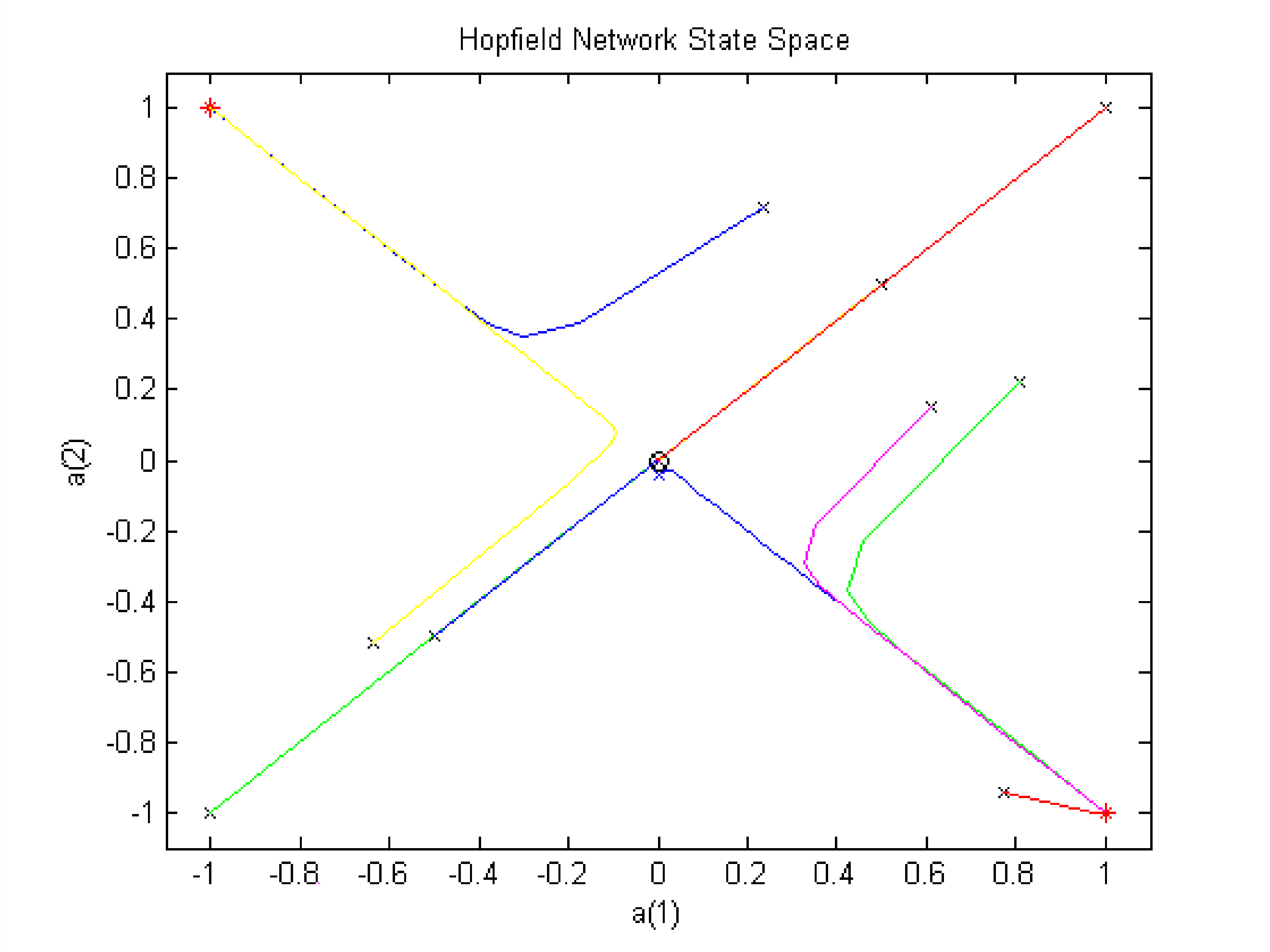
```
 T = [+1 -1; ...
      -1 +1];
plot(T(1,:),T(2,:),'r*')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield Network State Space')
xlabel('a(1)');
ylabel('a(2)');
net = newhop(T);
Y = sim(net,2,[],T)

a = {rands(2,1)};
y = sim(net,{1 20},{},a);

record = [cell2mat(a) cell2mat(y)];
start = cell2mat(a);
hold on
plot(start(1,1),start(2,1),'bx',record(1,:),record(2,:))
```



مثال ۱۱(ادامه...)

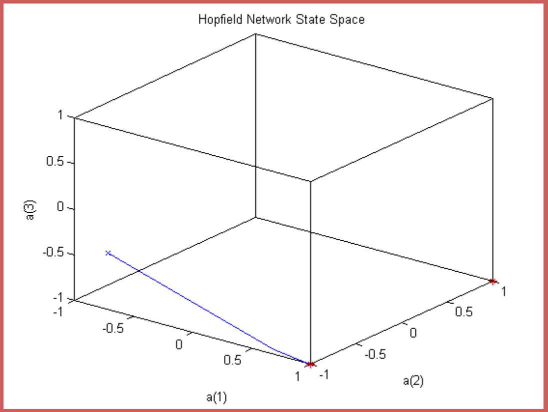
```
color = 'rgbmy';
for i=1:5
   a = {rands(2,1)};
   [y] = sim(net,{1 20},{},a);
   record=[cell2mat(a) cell2mat(y)];
   start=cell2mat(a);
plot(start(1,1),start(2,1),'kx',record(1,:),record(2,:),color(rem(i,5)+1))
end
plot(0,0,'ko');
P = [-1.0 -0.5 0.0 +0.5 +1.0;
     -1.0 -0.5 0.0 +0.5 +1.0];
color = 'rgbmy';
for i=1:5
   a = {P(:,i)};
   [y] = sim(net,{1 50},{},a);
   record=[cell2mat(a) cell2mat(y)];
   start = cell2mat(a);
   plot(start(1,1),start(2,1),'kx',record(1,:),record(2,:),color(rem(i,5)+1))
   drawnow
end
```

مثال ۱ (ادامه...)



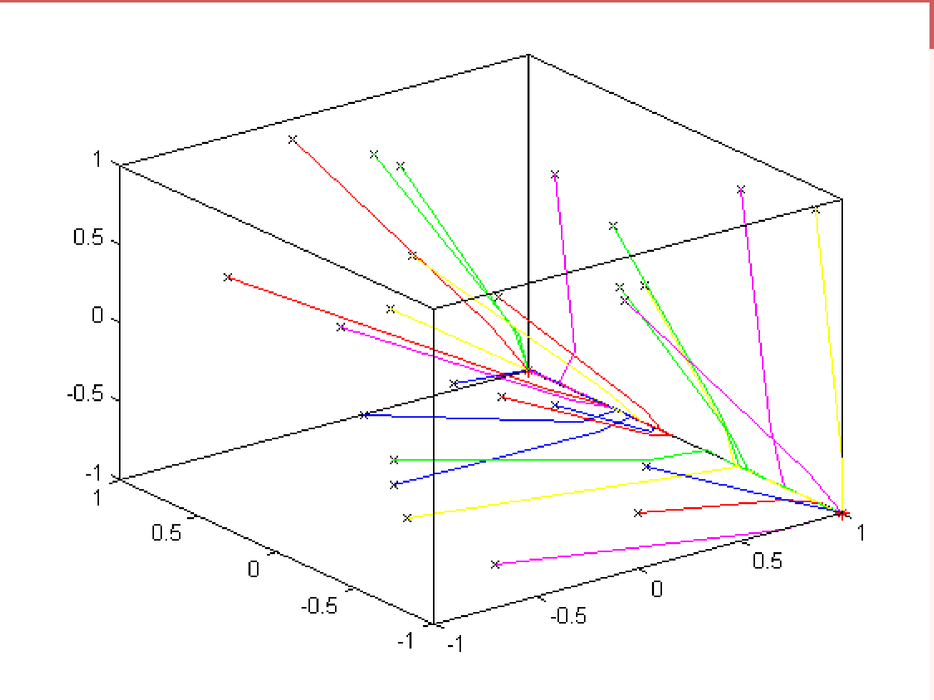
مثال ۲

```
T = [+1 +1; ...
      -1 +1; ...
      -1 -1];
axis([-1 1 -1 1 -1 1])
set(gca,'box','on'); axis manual;  hold on;
plot3(T(1,:),T(2,:),T(3,:),'r*')
title('Hopfield Network State Space')
xlabel('a(1)');
ylabel('a(2)');
zlabel('a(3)');
view([37.5 30]);
net = newhop(T);
a = {rands(3,1)};
[y,Pf,Af] = net({1 10},{},a);
record = [cell2mat(a) cell2mat(y)];
start = cell2mat(a);
hold on
plot3(start(1,1),start(2,1),start(3,1),'bx', ...
   record(1,:),record(2,:),record(3,:))
```



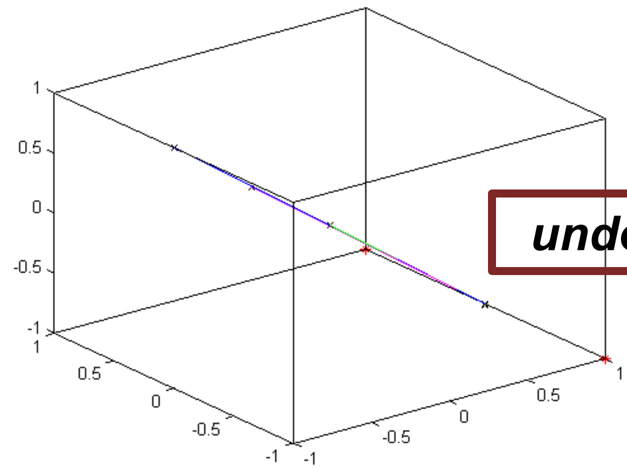
مثال ۲(ادامه...)

```
color = 'rgbmy';
for i=1:25
    a = {rands(3,1)};
    [y,Pf,Af] = net({1 10},{},a);
    record=[cell2mat(a) cell2mat(y)];
    start=cell2mat(a);
    plot3(start(1,1),start(2,1),start(3,1),'kx', …
        record(1,:),record(2,:),record(3,:),color(rem(i,5)+1))
end
```



مثال ۲(ادامه...)

```
P = [ 1.0  -1.0  -0.5  1.00  1.00  0.0; ...
      0.0   0.0   0.0  0.00  0.00 -0.0; ...
     -1.0   1.0   0.5 -1.01 -1.00  0.0];
plot3(T(1,:),T(2,:),T(3,:),'r*')
color = 'rgbmy';
for i=1:6
   a = {P(:,i)};
   [y,Pf,Af] = net({1 10},{},a);
   record=[cell2mat(a) cell2mat(y)];
   start=cell2mat(a);
   plot3(start(1,1),start(2,1),start(3,1),'kx', …
      record(1,:),record(2,:),record(3,:),color(rem(i,5)+1))
end
```



undesired stable point

```
close all;
clear all;
% Design of a Hopfield network which
stores 4 vectors
vectors=[-1 1 -1 -1 1 -1 -1 1 -1;
         -1 -1 -1 1 1 1 -1 -1 -1;
         -1 -1 1 -1 1 -1 1 -1 -1;
          1 -1 -1 -1 1 -1 -1 -1 1]';
net=newhop(vectors);
result=sim(net,4,[],vectors);
disp('Stored vectors:'); disp(vectors);
disp('Fixed points:'); disp(result);
% Dest data
test={[0.1; 0.8; -1; -0.7; 0.5; -1; -0.9;
0.85; -1]};
result=sim(net,{1,5},{},test);
% Network state after each iteration
for i=1:5,
    disp(sprintf('Network state after %d
    iterations:',i));
    disp(result{i});
end
```

شبکه عصبی

```
Network state after 1 iterations:
   -0.4930
    0.8601
   -1.0000
   -1.0000
    0.9661
   -1.0000
   -1.0000
    0.8712
   -0.7384

Network state after 2 iterations:
   -0.7045
    0.9879
   -1.0000
   -1.0000
    1.0000
   -1.0000
   -1.0000
    0.9904
   -0.7593

Network state after 3 iterations:
   -0.8625
    1.0000
   -1.0000
   -1.0000
    1.0000
   -1.0000
   -1.0000
    1.0000
   -0.8748

Network state after 4 iterations:
   -0.9966
    1.0000
   -1.0000
   -1.0000
    1.0000
   -1.0000
   -1.0000
    1.0000
   -0.9993

Network state after 5 iterations:
    -1
     1
    -1
    -1
     1
    -1
    -1
     1
    -1

>>
```

```
vectors =[[1,-1,-1,-1,1,1,1,-1,-1,1,1,-1,1,-1,1,1,-1,-1,1,1,1,-1,-1,-1,1];
          [1,1,1,1,1,1,-1,-1,-1,-1,1,1,1,1,1,1,-1,-1,-1,-1,1,1,1,1,1];
          [1,1,1,1,1,1,-1,-1,-1,1,1,-1,-1,-1,1,1,-1,-1,-1,1,1,1,1,1,1];
          [-1,1,1,1,-1,1,-1,-1,-1,1,1,-1,-1,-1,1,1,-1,-1,-1,1,-1,1,1,1,-1];]';
 subplot(2,4,1);imshow( vecto2D(vectors(:,1)),[]);title('1th vect');
 subplot(2,4,2);imshow(vecto2D(vectors(:,2)),[]);title('2th vect');
 subplot(2,4,3);imshow(vecto2D(vectors(:,3)),[]);title('3th vect');
 subplot(2,4,4);imshow(vecto2D(vectors(:,4)),[]);title('4th vect');

net=newhop(vectors);
result=sim(net,4,[],vectors);
disp('Stored vectors:'); disp(vectors);
disp('Fixed points:'); disp(result);
% Dest data

test={[1,1,1,1,1,1,-1,-1,-1,-1,1,1,1,1,1,1,-1,-1,1,1,-1,1,1,1,1]'};
subplot(2,4,5);imshow(vecto2D(test{1,1}),[]);title('test pattern');
result=sim(net,{1 8},{},test);

subplot(2,4,8);imshow(vecto2D(result{1,8}),[]);title('8th iteration
result');
 figure;
 % Network state after each iteration
 for i=1:8
     subplot(2,4,i);imshow(vecto2D(result{1,i}),[]);
 end
```

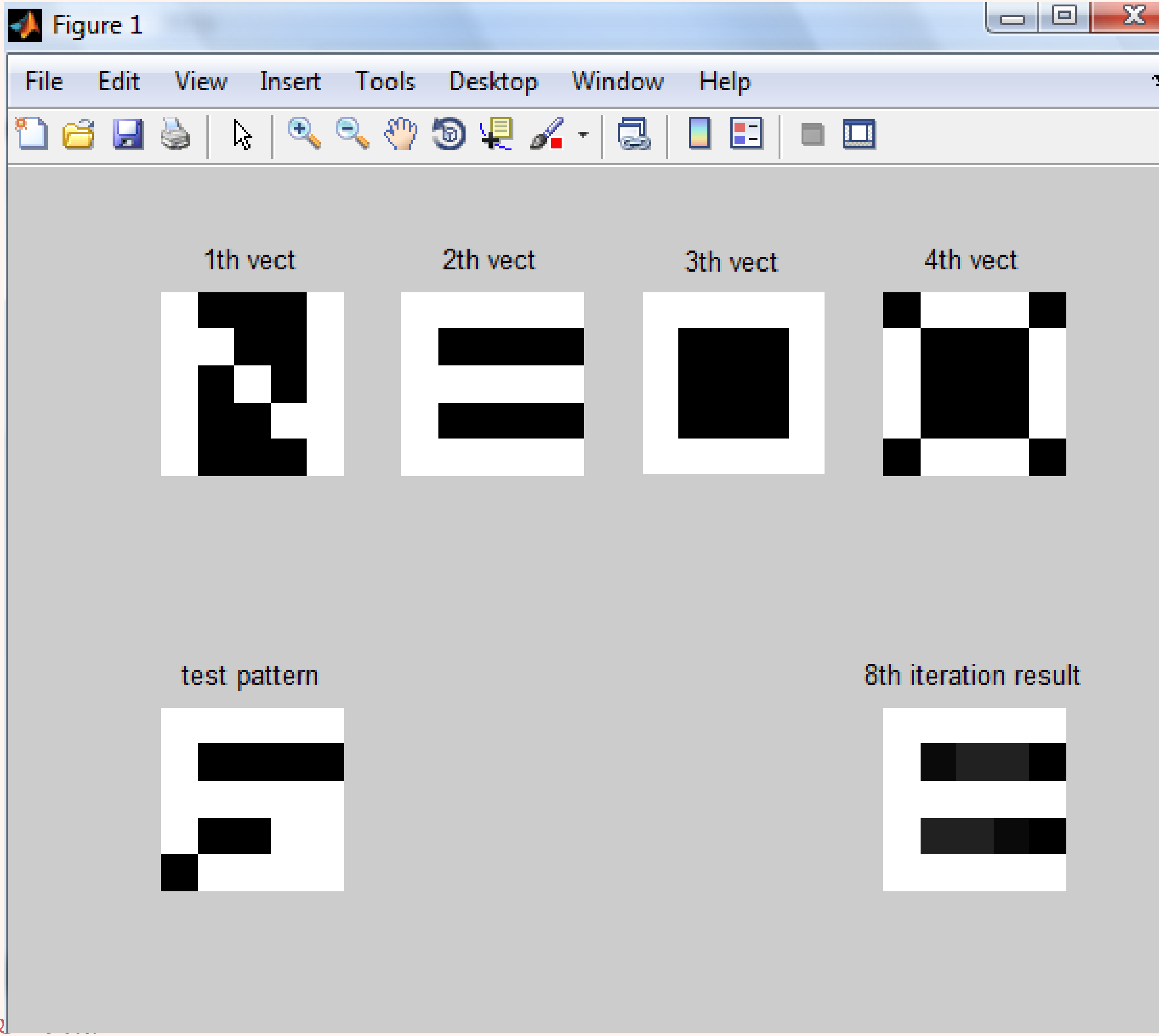
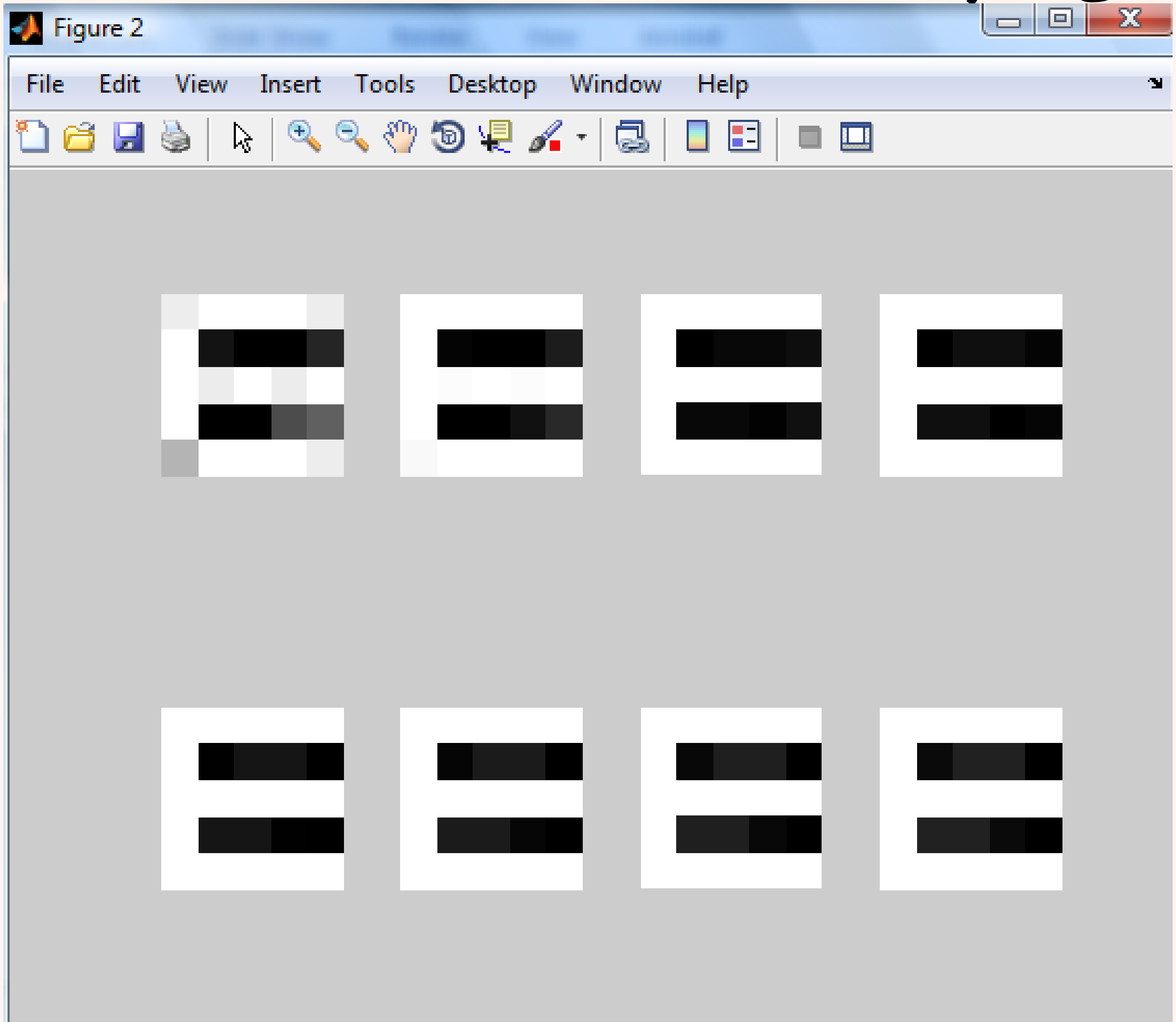
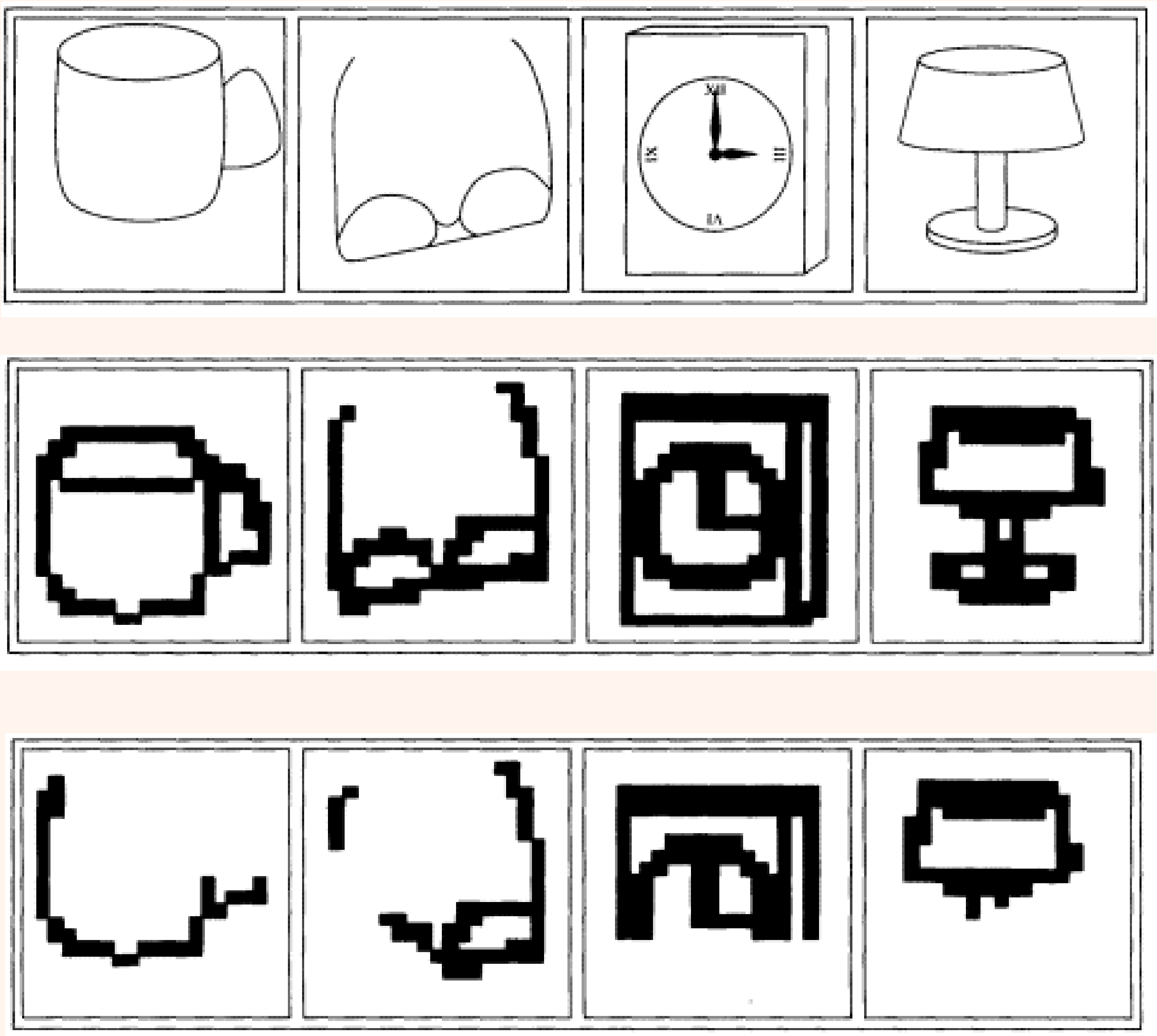


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
1th vect
2th vect
3th vect
4th vect
test pattern
8th iteration result

مراحل کار

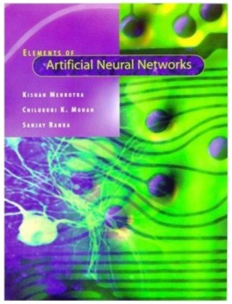


مثال



منابع مورد استفاده

- عمده‌ی مطالب این بخش برگرفته از بخش ششم کتاب زیر است:
  - Elements of Artificial Neural Network Kishan Mehrotra, Chilukuri K. Mohan and Sanjay Rank



- مستندات Matlab و مثال‌های آن مورد استفاده قرار گرفته است.
- در تهیه‌ی این مطالب، از اسلایدهای زیر نیز استفاده شده است.
  - http://www.cs.umb.edu/~marc/cs672/