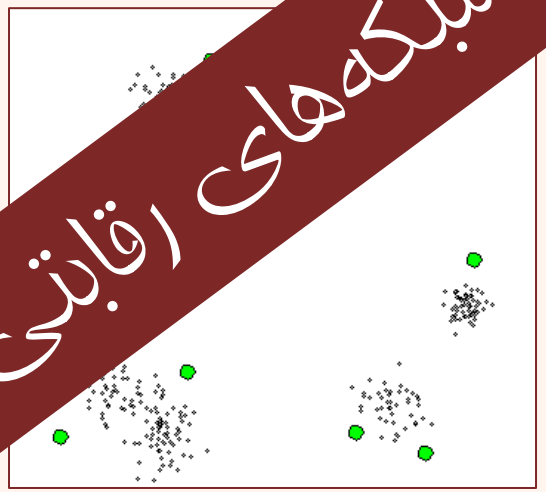


شبکه‌های عصبی مصنوعی
۰۱-۷۱۳-۱۱-۱۳
بخش پنجم

دانشگاه شهید بهشتی
دانشکده‌ی مهندسی و علوم کامپیوتر
زمستان ۱۳۹۴
احمد محمودی ازناوه

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

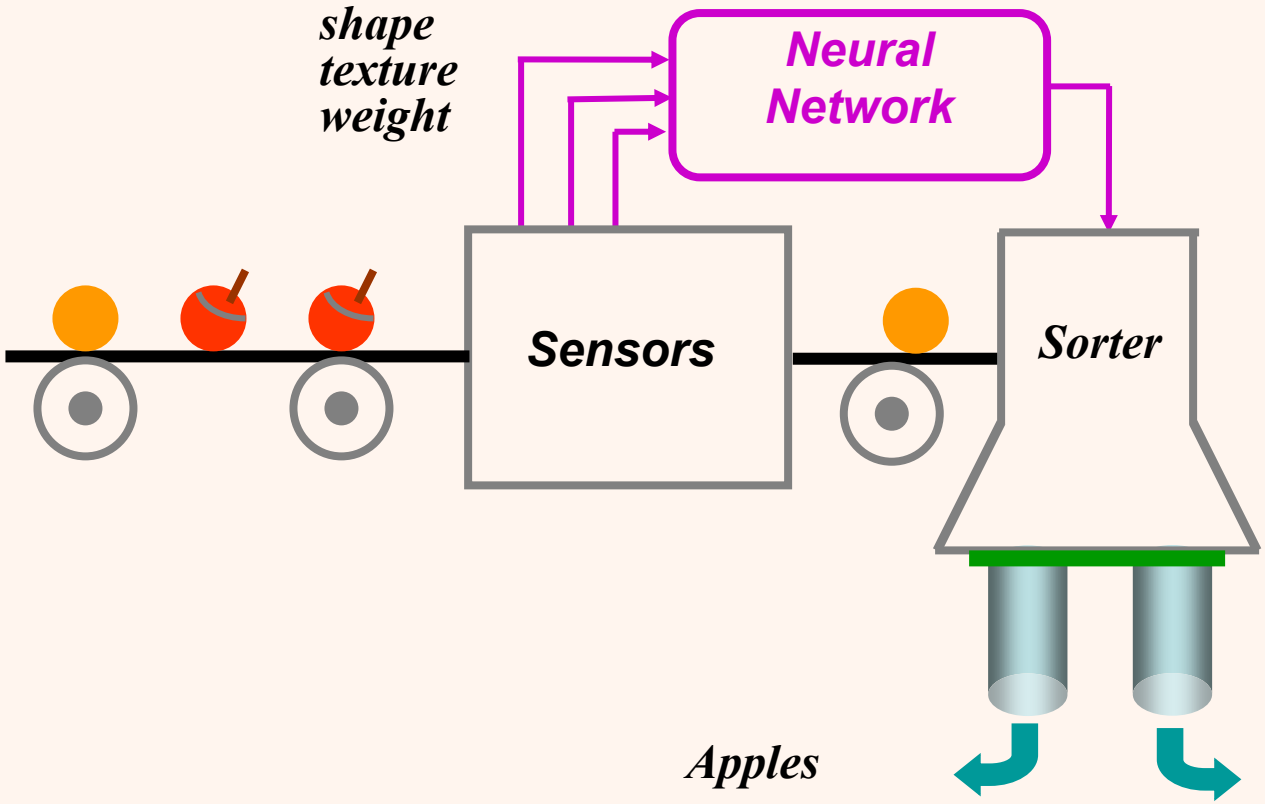
فهرست مطالب

- شبکه‌های رقابتی
  - شبکه‌ی همینگ
    - مقایسه گُذرای شبکه‌ی همینگ و پرسپترون
    - لایه‌ی feedforward
    - لایه‌ی رقابتی- recurrent
  - آموزش رقابتی
  - مشکلات آموزش رقابتی
- شبکه‌های خودسازمان‌ده (SOM)
- LVQ

پیش‌گفتار

- برای حل یک مسأله‌ی «بازشناخت الگو» از راه‌حل‌های متفاوتی می‌توان استفاده کرد:
  - Perceptron
  - **Hamming network**
  - Hopfield network
- تاکنون با برخی از روش‌ها آشنا شدیم. در این بخش با «یادگیری رقابتی» آشنا خواهیم شد.

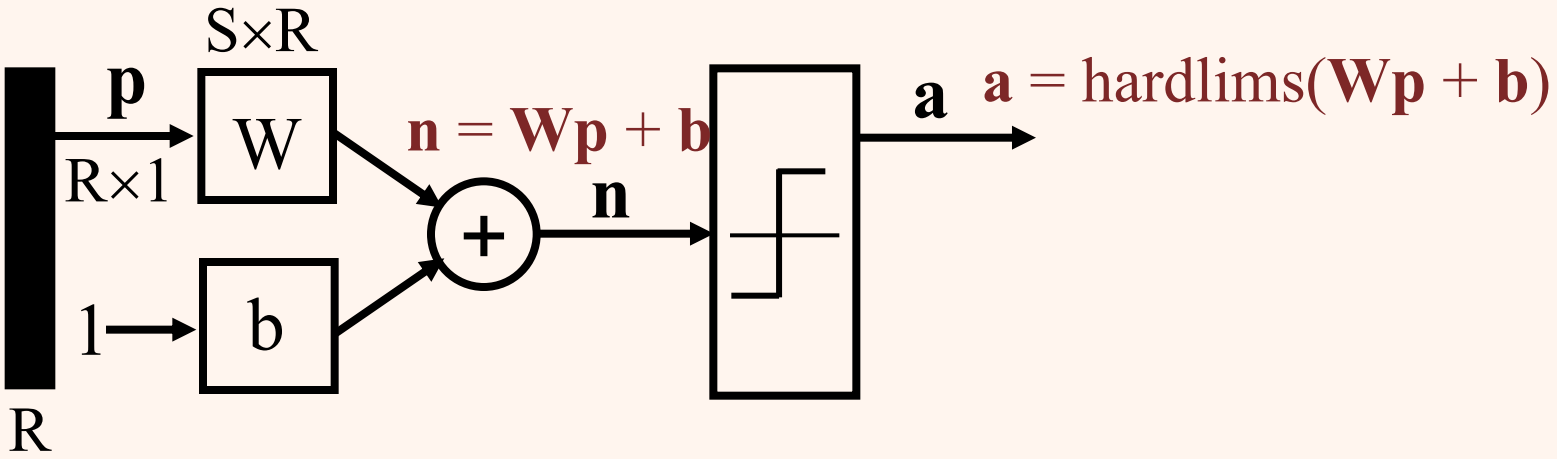
مثال



$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \Rightarrow \mathbf{p}(apple) = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{p}(orange) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

Shape sensor: 1 -- round, –1 -- elliptical.
Texture sensor: 1 -- smooth, –1 -- rough.
Weight sensor: 1 -- > 1 pound, –1 -- < 1 pound.

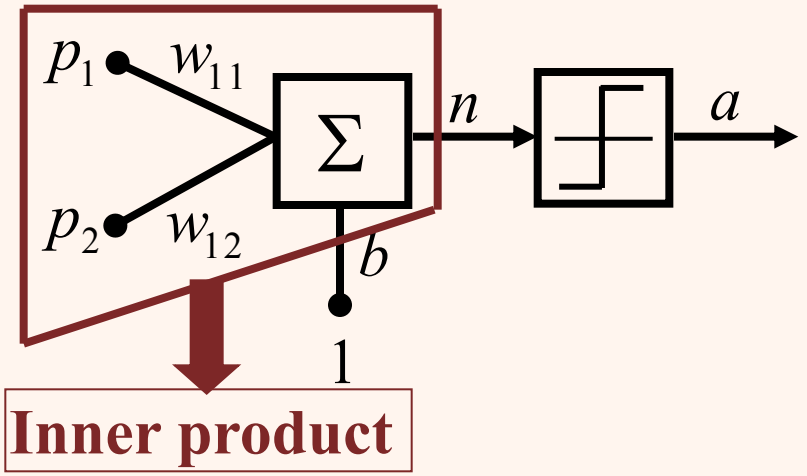
Single-layer Perceptron/ Instar



Symmetrical Hard Limit

- **a = -1, if n < 0**
- **a = +1, if n ≥ 1**
- **MATLAB function: hardlims**

Two-Input / Single-Neuron Perceptron



$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}, \mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \end{bmatrix}$$

$$\Rightarrow n = \mathbf{Wp} + b$$

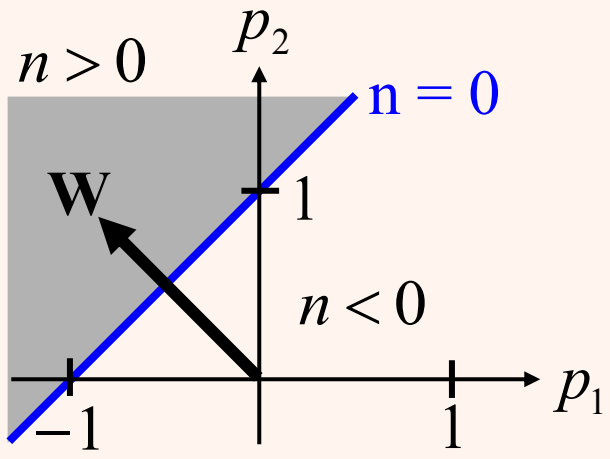
$$a = \text{hardlims}(n)$$

- دیدیم یک نورون می‌تواند یک بردار ورودی را به دو گروه دسته‌بندی کند.

If $\mathbf{Wp} \geq -b$,
then
a = **+1**;
otherwise,
a = **-1**.

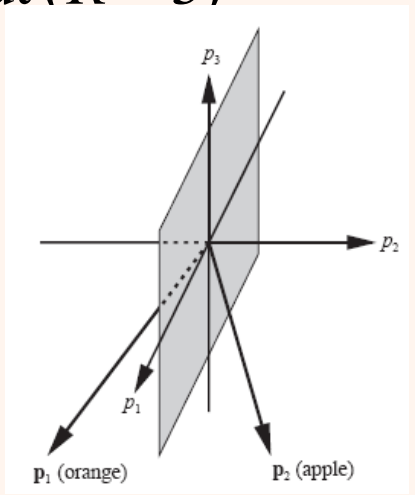
Two-Input / Single-Neuron Perceptron

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \end{bmatrix} = \begin{bmatrix} -1 & 1 \end{bmatrix}, b = -1$$
$$\Rightarrow n = \mathbf{Wp} + b = -p_1 + p_2 - 1$$
$$\Rightarrow a = \text{hardlims}(n)$$



$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \Rightarrow \text{three-dimensional input } (R = 3)$$

$$n = \mathbf{Wp} + b, \; a = \text{hardlims}(n)$$



شناسایی الگوها

$$\mathbf{p}(apple)=\begin{bmatrix}1\\1\\-1\end{bmatrix}\quad \mathbf{p}(orange)=\begin{bmatrix}1\\-1\\-1\end{bmatrix}\quad \Rightarrow \mathbf{W}=\begin{bmatrix}0 & 1 & 0\end{bmatrix},\ b=0$$

$$\textbf{Orange}: a=\text{hardlims}\left(\begin{bmatrix}0 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\-1\\-1\end{bmatrix}+0\right)=-1$$

$$\textbf{Apple}: a=\text{hardlims}\left(\begin{bmatrix}0 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix}+0\right)=1$$

شناسایی الگوها

- حال اگر ورودی به صورتی باشد که تصمیم دقیق میسر نگردد راه حل چیست؟

$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \Rightarrow \quad \mathbf{p}(orange) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

$$a = \text{hardlims}\left(\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1 \Rightarrow \textbf{orange}$$

Hamming Network

شبکه‌ی همینگ ساده‌ترین شبکه‌ی رقابتی است.

Hamming Network

- شبکه‌ی Hamming برای دسته‌بندی داده‌های دودویی مطرح شد.
- در شبکه‌ی همینگ دو لایه وجود دارد:
  - لایه‌ی یک (FeedForward)
  - لایه‌ی دو (Recurrent)
- لایه‌ی یک
  - تصمیم می‌گیرد ورودی به کدام بردار شناسه نزدیک‌تر است.
- لایه‌ی دو
  - وقتی لایه‌ی دوم یا recurrent پایان می‌پذیرد، تنها یک خروجی غیر صفر باقی می‌ماند، در واقع این لایه، لایه‌ی رقابتی است.

Recurrent Network:

شبکه‌ای است که در آن feedback وجود دارد؛ در آن برخی خروجی‌ها به لایه‌ی ورودی متصل هستند. خروجی مرحله‌ی قبل به عنوان ورودی مرحله‌ی فعلی استفاده می‌شود.

Hamming Network

- در طبقه‌بندی همینگ هدف دسته‌بندی ورودی‌ها در S کلاس متفاوت است.
- در شبکه‌ی همینگ کلاس‌های S معلوم هستند (مرکز هر کلاس مشخص است).
- فرض می‌شود S بردار شناسه(Prototype) وجود دارد.

بردارهای شناسه

$$P_{1(R\times1)}, P_{2(R\times1)}, \ldots\ldots\ldots, P_{s(R\times1)}$$

- ورودی: برداری به اندازه‌ی R×1 است.

$$X = [x_1, \ldots\ldots, x_R]^T$$

فاصله

- در خوشه‌بندی و شبکه‌های رقابتی باید معیاری که بیان‌گر فاصله‌ی ورودی از بردار شناسه باشد، تعریف شود. سه معیار متداول به ترتیب زیر هستند:

– فاصله‌ی اقلیدسی: $$d(\mathbf{X},\mathbf{P})=\sum_{k}[\mathbf{X}(k)-\mathbf{P}(k)]^2$$

– ضرب داخلی: بزرگ‌تر بودن حاصل‌ضرب داخلی به معنای کم‌تر بودن زاویه‌ی دوبردار است.

– فاصله‌ی همینگ: برابر با تعداد مؤلفه‌هایی است که با هم تفاوت دارند.

Hamming Network

هدف

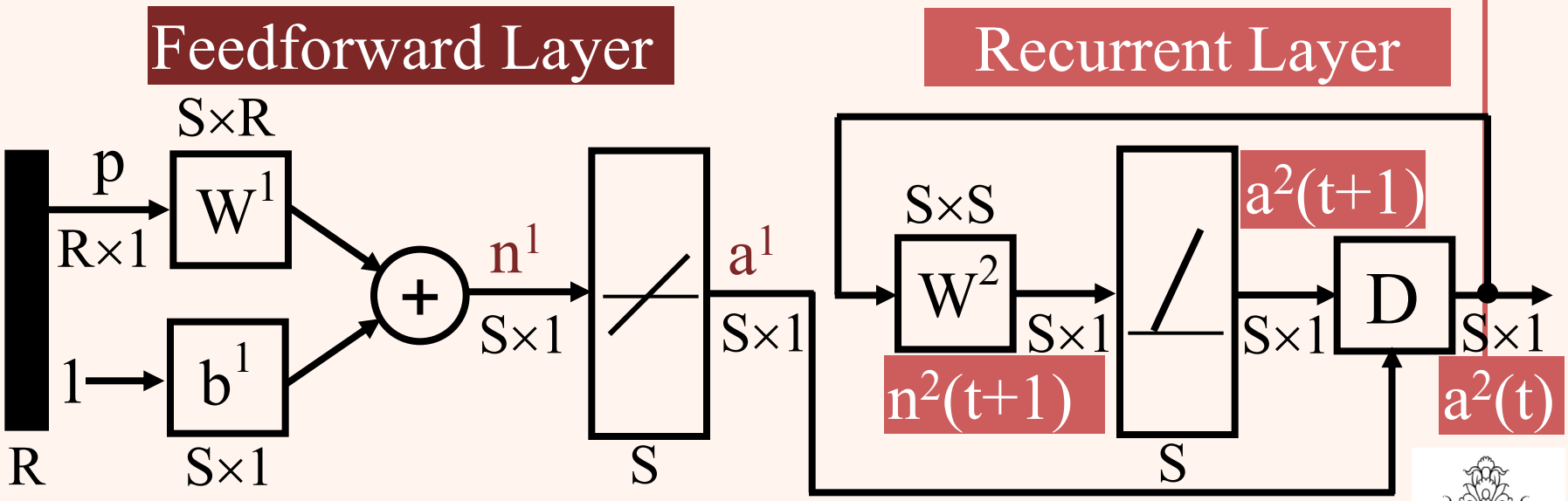
- دسته‌بندی نمودن ورودی‌ها در S کلاس مختلف با S بردار شناسه
  - بردارهای شناسه هم‌اندازه‌ی بردار ورودی است.
- در صورتی که هم‌اندازه نباشد آن‌ها را هم‌اندازه (Normalize) می‌کنیم.

$$X_i = \frac{X_i}{\|X_i\|}\|P_j\|$$

- پس از نرمالایز کردن، می‌توان داده‌ها را به صورت نقاط روی محیط یک دایره(ابرکره) در نظر گرفت.

$$Y_{S\times 1} = W_{S\times R}.X_{R\times 1} + B^T_{S\times 1}$$

Hamming Network



$$\mathbf{n}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1$$

$$\mathbf{a}^1 = \underline{\mathbf{pure}}\mathrm{lin}(\mathbf{n}^1)$$

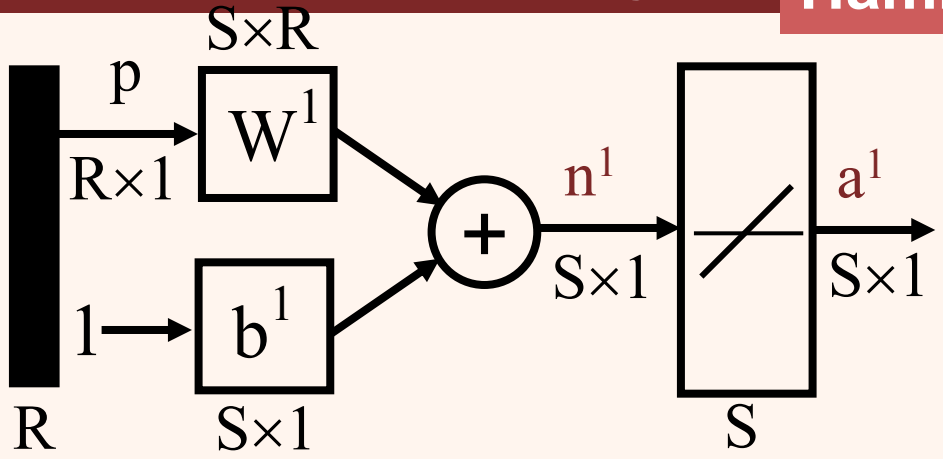
$$\mathbf{n}^2(t+1) = \mathbf{W}^2\mathbf{a}^2(t)$$

$$\mathbf{a}^2(t+1) = \underline{\mathbf{pos}}\mathrm{lin}[\mathbf{n}^2(t+1)]$$

$$\mathbf{a}^2(0) = \mathbf{a}^1$$

Feedforward Layer

Hamming Network



- لایه‌ی پک

$$n^1 = W^1 p + b^1$$
$$a^1 = \underline{\text{pure}}\text{lin}(n^1)$$

- همبستگی بین بردارهای شناسه و ورودی محاسبه می‌شوند.
- ماتریس W^1 با استفاده از بردارهای شناسه مقداردهی می‌شود.

connection matrix W^1

- بایاس را به اندازه‌ی مثبت (R) در نظر می‌گیریم که خروجی منفی نگردد.

در صورتی که ورودی‌ها همه دومقداری(۱ و ۱-) باشند، بین ضرب داخلی و فاصله‌ی همینگ چه ارتباطی وجود خواهد داشت؟

Hamming Network

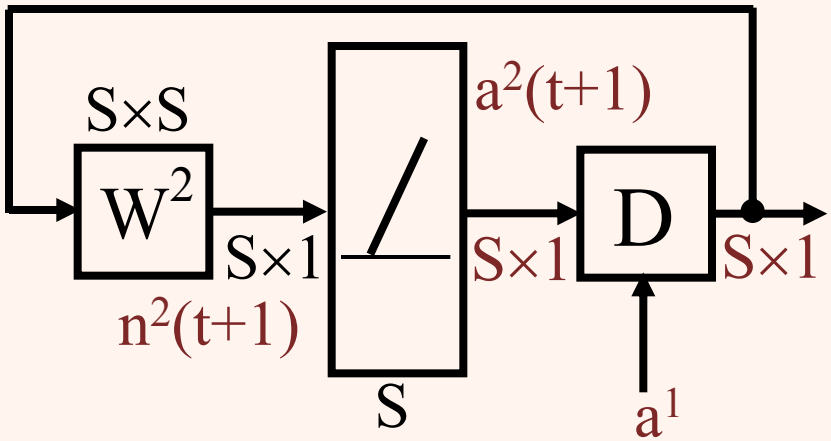
$$Y_{S\times 1} = W_{S\times R}.X_{R\times 1} + B^{T}_{S\times 1}$$

$$Y = \begin{bmatrix} P_1^T \\ . \\ . \\ P_S^T \end{bmatrix}.\begin{bmatrix} x_1 \\ . \\ . \\ x_R \end{bmatrix} + \begin{bmatrix} R \\ . \\ . \\ R \end{bmatrix} \qquad \theta_k = \measuredangle$$

$$Y = \begin{bmatrix} P_1^T .\mathbf{x} + R \\ . \\ . \\ P_S^T .\mathbf{x} + R \end{bmatrix} = \begin{bmatrix} \|P_1\|.\|\mathbf{x}\|\cos\theta_0 + R \\ . \\ . \\ \|P_s\|.\|\mathbf{x}\|\cos\theta_s + R \end{bmatrix}$$

Recurrent Layer (MaxNet)

Hamming Network



$$\mathbf{n}^2(t+1) = \mathbf{W}^2\mathbf{a}^2(t)$$
$$\mathbf{a}^2(t+1) = \underline{\mathbf{pos}}\mathrm{lin}[\mathbf{n}^2(t+1)]$$
$$\mathbf{a}^2(0) = \mathbf{a}^1$$

- همان لایه‌ی رقابتی "Competitive" است که در این لایه نورون‌ها برای تعیین برنده با هم به رقابت می‌پردازند.
- برنده نورونی است که بیشترین مقدار را دارد. تنها یک نورون برنده خواهد شد.

Biologically plusibility

فرض کنیم خروجی k از همه بزرگ‌تر است چون بقیه از k کم‌ترند اثر تضعیفی بر بقیه بیشتر است و در انتها تنها همین عنصر k غیر صفر باقی خواهد ماند.

$$O_{s\times 1}(0) = y$$

$$O_{s\times 1}(n+1) = f\left(w_2 . O(n)\right)$$

$$f(x) = \begin{cases} x & x \geq 0 \\ 0 & OW \end{cases}$$

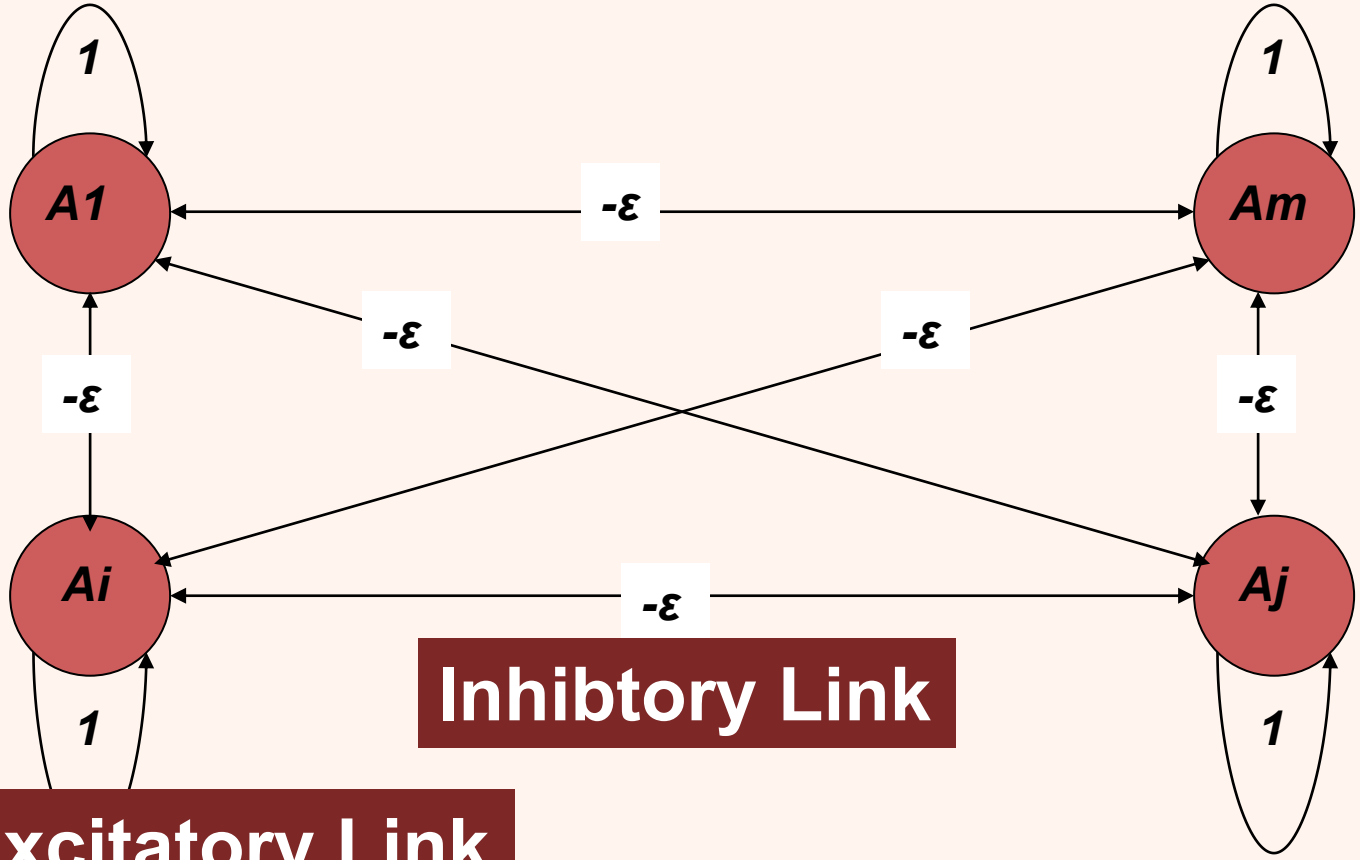
$$W^2_{(i,j)} = \begin{cases} 1 & i = j \\ -\varepsilon & i \neq j \end{cases} \qquad \varepsilon < \frac{1}{s-1}$$

Lateral Inhibition: excites itself and inhibits all the other neurons

$$W^2_{(i,i)} = \begin{bmatrix} 1 & . & . & -\varepsilon \\ . & 1 & . & . \\ . & . & 1 & . \\ -\varepsilon & . & . & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & . & . & -\varepsilon \\ . & 1 & . & . \\ . & . & 1 & . \\ -\varepsilon & . & . & 1 \end{bmatrix} \begin{bmatrix} O_1(n) \\ . \\ . \\ O_s(n) \end{bmatrix} = \begin{bmatrix} O_1(n) - \varepsilon \sum_{i=2}^{s} O_i(n) \\ . \\ . \\ O_s(n) - \varepsilon \sum_{i=1}^{s-1} O_i(n) \end{bmatrix}$$

MAXNET



Lippmann, R. P. (1987). "An introduction to computing with neural nets." ASSP Magazine, IEEE 4(2): 4-22.

Recurrent Layer

Hamming Network

تعداد نورون‌ها

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}, \quad \mathbf{a}^2(t) = \begin{bmatrix} a_1^2(t) \\ a_2^2(t) \end{bmatrix}, \quad \text{and} \quad \varepsilon < \frac{1}{S-1}$$

$$\mathbf{a}^2(t+1) = \text{poslin}\left(\mathbf{W}^2\mathbf{a}^2(t)\right) = \text{poslin}\left(\begin{bmatrix} a_1^2(t) - \varepsilon a_2^2(t) \\ a_2^2(t) - \varepsilon a_1^2(t) \end{bmatrix}\right)$$

- تفاوت میان دو عنصری که بزرگ و کوچک هستند در این صورت افزایش می‌یابد.

$$\text{If } \mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow \mathbf{a}^2(0) = \mathbf{a}^1 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \text{ let } \varepsilon = 0.5 \Rightarrow \mathbf{W}^2 = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

$$\mathbf{a}^2(1) = \text{poslin}\left(\begin{bmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{bmatrix}\begin{bmatrix} 4 \\ 2 \end{bmatrix}\right) = \text{poslin}\left(\begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

$$\mathbf{a}^2(2) = \text{poslin}\left(\begin{bmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) = \text{poslin}\left(\begin{bmatrix} 3 \\ -1.5 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

مثال

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix}, \mathbf{b}^1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix}\mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p} + 3 \\ \mathbf{p}_2^T\mathbf{p} + 3 \end{bmatrix}$$

ضرب داخلی دو بردار به علاوه‌ی بایاس

$$\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow \mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

فاصله‌ی همینگ میان دو بردار دودویی برابر است با تعداد عناصری که در مقایسه با یکدیگر متفاوتند

Output = 2 × (R – Hamming distance)

$$\mathbf{a}^1_1 = 2 \times (3 - 1) = 4, \mathbf{a}^1_2 = 2 \times (3 - 2) = 2$$

هنگامی‌که نتیجه‌ی دو تکرار یکسان شود گوییم الگوریتم به همگرایی رسیده است.

مثال

- در یک فضای سه بعدی بردارهای شناسه مانند زیر است ورودی اگر x باشد:

$$p_1 = [1 \quad -1 \quad -1]^T \qquad p_2 = [1 \quad 1 \quad -1]^T$$

$$x = [2 \quad 3 \quad -6]^T$$

- ابتدا بایاس را در نظر می‌گیریم:

$$\mathbf{b}^1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

- بردار ورودی را نرمالیزه می‌کنیم:

$$\|p_1\|^2 = \|p_2\|^2 = 3 \qquad \|x\|^2 = 49$$

$$\|x_{new}\|^2 = \frac{49}{a} = 3 \quad a \approx 16 \qquad \|x_{new}\|^2 = \frac{4}{a} + \frac{9}{a} + \frac{36}{a}$$

ادامه‌ی مثال

$$\|x_{new}\|^2 = \frac{4}{a} + \frac{9}{a} + \frac{36}{a}$$

$$x_{new} = \left[\frac{2}{\sqrt{a}}, \frac{3}{\sqrt{a}}, \frac{-6}{\sqrt{a}}\right] \qquad x_{new} = [0.5, 0.71, -1.5]$$

$$x = \begin{bmatrix} 0.5 \\ 0.71 \\ -1.5 \end{bmatrix} \Rightarrow \mathbf{a}^1 = \mathbf{W}^1 x + \mathbf{b}^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.71 \\ -1.5 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 1.29 \\ 2.71 \end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}, \quad \varepsilon < \frac{1}{S-1}$$

$$O_{s\times 1}(0) = y$$

$$O_{s\times 1}(n+1) = f(w_2 . O(n))$$

ادامه‌ی مثال

$$O_{s\times 1}(0) = y$$

$$O_{s\times 1}(n+1) = f\,(w_2 . O(n))$$

$$O_{s\times 1}(1) = f\,(w_2 . O(0)) = f\,(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix} 1.29 \\ 2.71 \end{bmatrix}) = f\,(\begin{bmatrix} <0 \\ 2.07 \end{bmatrix})$$

- اگر ε کوچک در نظر گرفته شود سرعت همگرایی پایین می‌آید.

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}, \quad \varepsilon < \frac{1}{S-1}$$

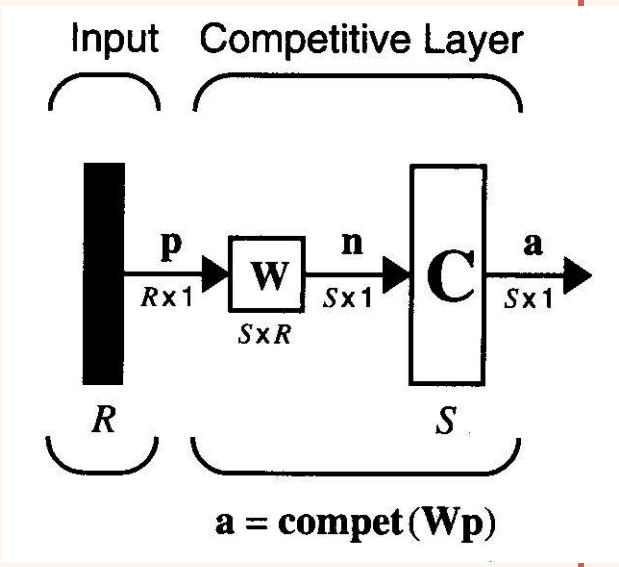
A recurrent competitive layer

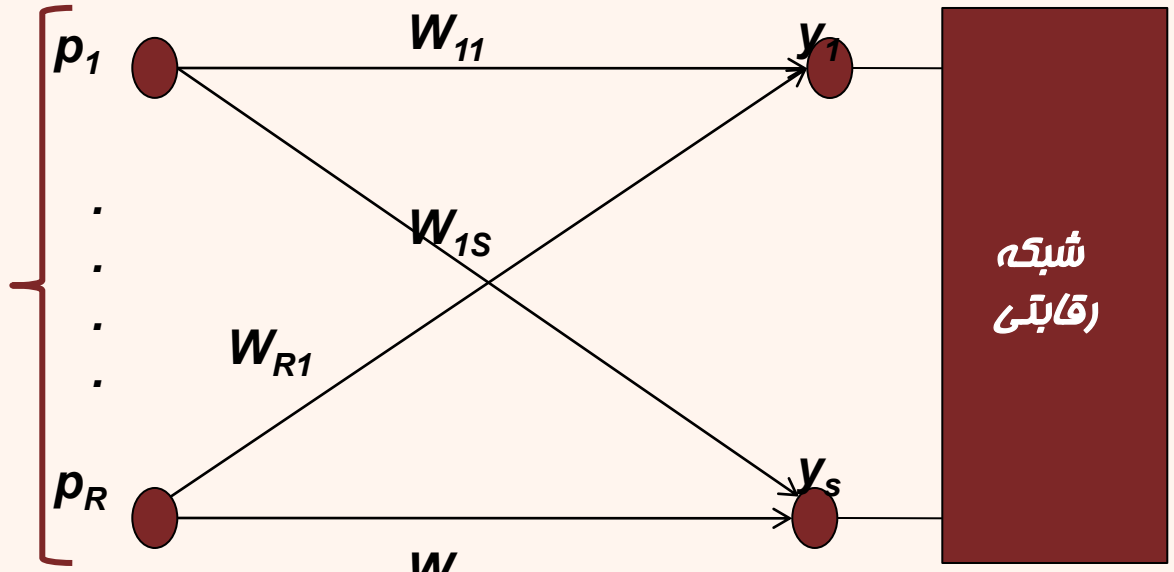
لایه‌ی رقابتی

$$\mathbf{n} = \mathbf{W}\mathbf{p} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix} \mathbf{p} = \begin{bmatrix} L^2 \cos\theta_1 \\ L^2 \cos\theta_2 \\ \vdots \\ L^2 \cos\theta_S \end{bmatrix}$$

$$n_{i^*} = \max\{n_1, n_2, ..., n_S\}$$

$$\mathbf{a} = \mathbf{compet}(\mathbf{n}) \qquad a_i = \begin{cases} 1, i = i^* \\ 0, i \neq i^* \end{cases}$$





Competitive Learning

آموزش رقابتی

- تنها تعداد دسته‌ها مشخص است.
- بردارهای شناسه ناشناخته‌اند.
  - ماتریس وزن‌ها تصادفی انتخاب می‌شود.
- هدف:
  - یافتن ماتریس وزن(بردارهای شناسه) بهینه
- می‌توان هر سطر ماتریس W به صورت تصادفی انتخاب کرد.
  - ورودی‌ها به اندازه‌ی $R \times 1$
  - تعداد گروه‌ها به اندازه‌ی S

آموزش رقابتی (ادامه...)

مراحل الگوریتم:

۱) W_i ها به طور تصادفی انتخاب می‌شوند.

۲) همگی به یک مقدار نرمالیزه می‌شوند.

۳) در هر تکرار یک ورودی به شبکه اعمال می‌شود.

$$P_{(n)} = \left[P_1, ..., P_R\right]^T$$

۴) در خروجی رابطه‌ی زیر محاسبه می‌شود:

$$y_{i(n)} = P_{(n)}^T w_{i(n)} = \left\|P_{(n)}\right\| \left\|w_{i(n)}\right\| . \cos\theta_{i(n)}$$

$$i = 1, 2, ..., S$$

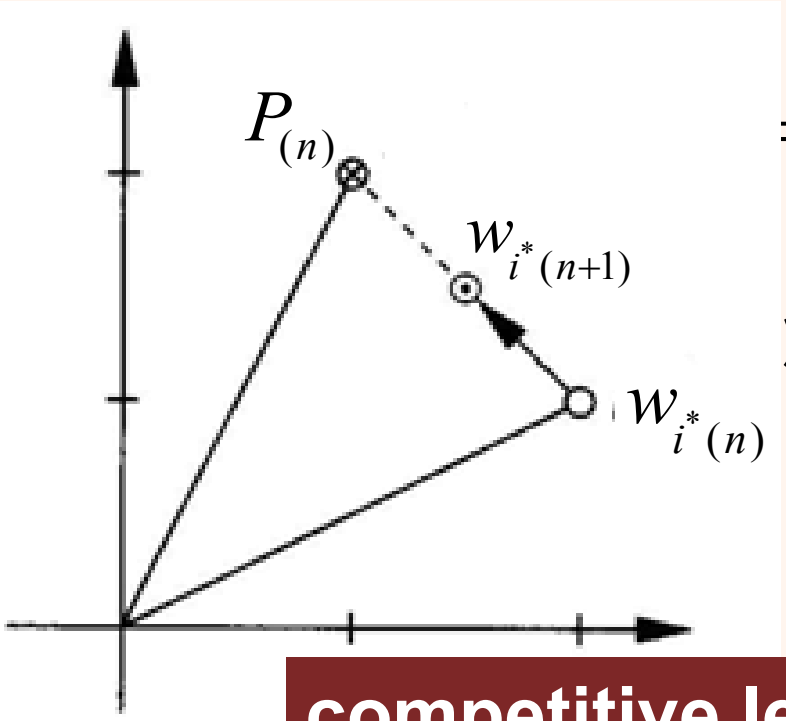
آموزش رقابتی (ادامه...)

۵) واحدی که دارای بیشترین مقدار خروجی باشد به عنوان «برنده» انتخاب می‌شود و با نام واحد شماره‌ی i^* خوانده می‌شود.

$$y_{i^*} > y_i$$

۶) مرحله‌ی تطبیق وزن‌ها:

$$= w_{i^*(n)} + \eta[P_{(n)} - w_{i^*(n)}]$$



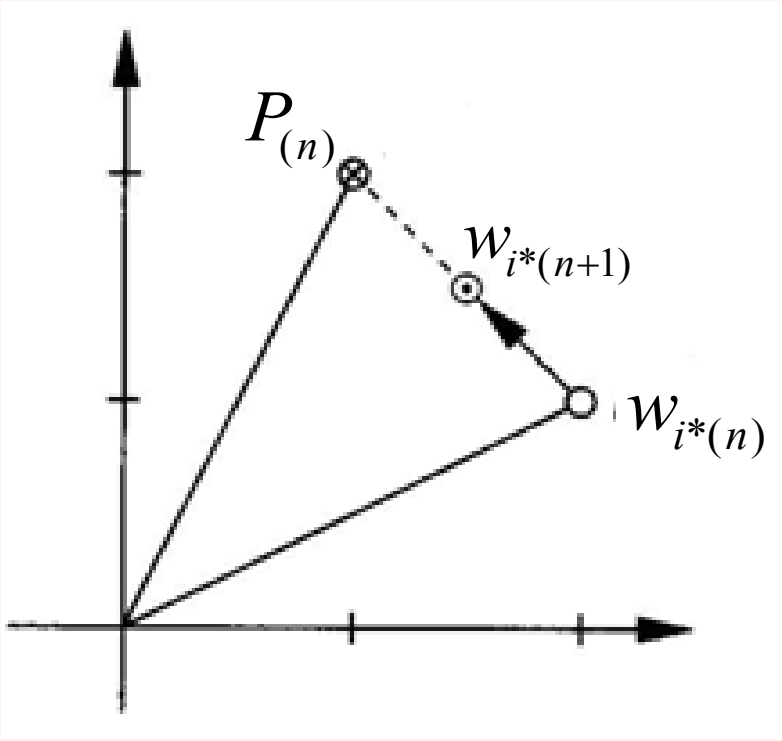
۷) اعمال ورودی بعدی و تکر
مرحله ۳ به بعد

competitive learning rule(Kohonen rule)

آموزش رقابتی (ادامه...)

$$w_{i^*(n+1)} = w_{i^*(n)} + \eta[P_{(n)} - w_{i^*(n)}]$$

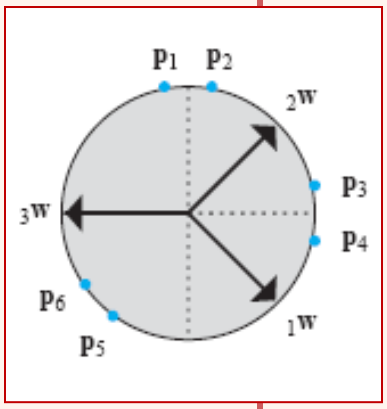
$$w_{i^*(n+1)} = (1-\eta)w_{i^*(n)} + \eta P_{(n)}$$



مثال

$$P_1 = \begin{bmatrix} -0.19 \\ 0.98 \end{bmatrix} \quad P_3 = \begin{bmatrix} 0.98 \\ 0.19 \end{bmatrix} \quad P_5 = \begin{bmatrix} -0.58 \\ 0.81 \end{bmatrix}$$

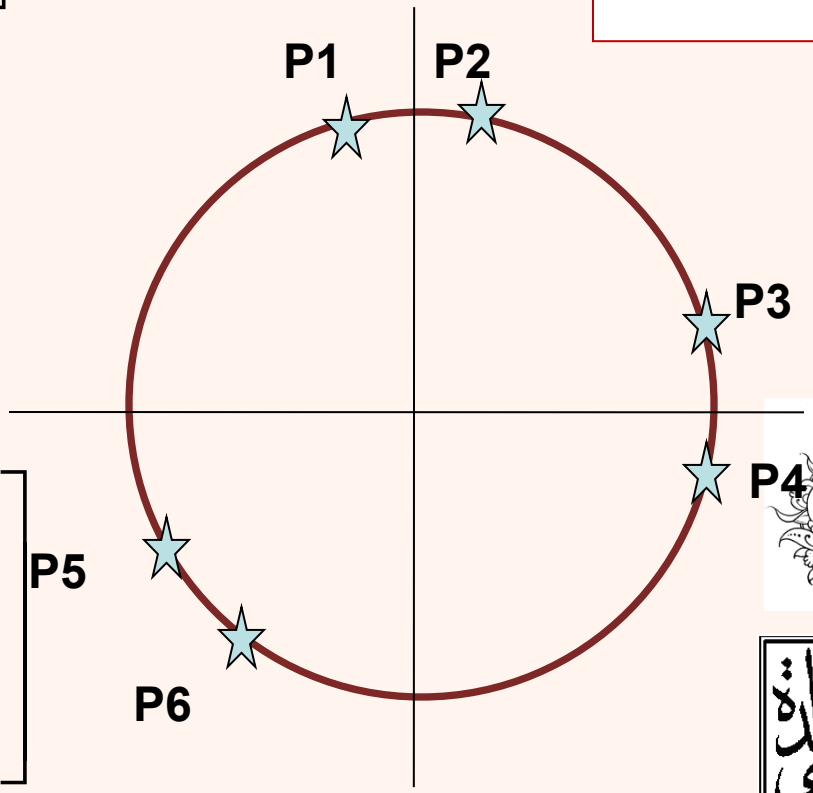
$$P_2 = \begin{bmatrix} 0.19 \\ 0.98 \end{bmatrix} \quad P_4 = \begin{bmatrix} 0.98 \\ -0.19 \end{bmatrix} \quad P_6 = \begin{bmatrix} -0.81 \\ -0.58 \end{bmatrix}$$



$$W_1 = \begin{bmatrix} 0.7 \\ -0.7 \end{bmatrix} \quad W = \begin{bmatrix} W_1^T \\ W_2^T \\ W_3^T \end{bmatrix}$$

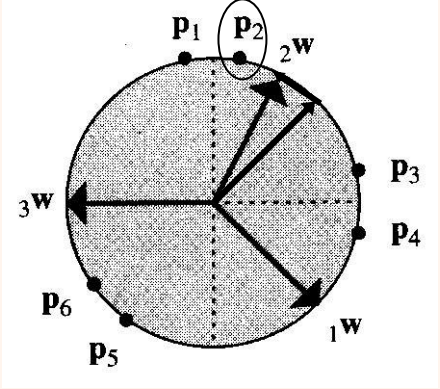
$$W_2 = \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix}$$

$$W_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad W = \begin{bmatrix} 0.7 & -0.7 \\ 0.7 & 0.7 \\ -1 & 0 \end{bmatrix}$$



مشخص است که سه گروه داریم وزن ها را تصادفی انتخاب می کنیم

ادامه‌ی مثال



- ورودی P_2 را به شبکه اعمال می کنیم.

$$W.P_2 = \begin{bmatrix} 0.7 & -0.7 \\ 0.7 & 0.7 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0.19 \\ 0.98 \end{bmatrix} = \begin{bmatrix} -0.55 \\ 0.8 \\ -0.1 \end{bmatrix}$$

واحد برنده شماره‌ی ۲ است

- وزن واحد برنده را به روز می‌نماییم:

$$w_{2(n+1)} = w_{2(n)} + \eta[P_2 - w_{2(n)}]$$

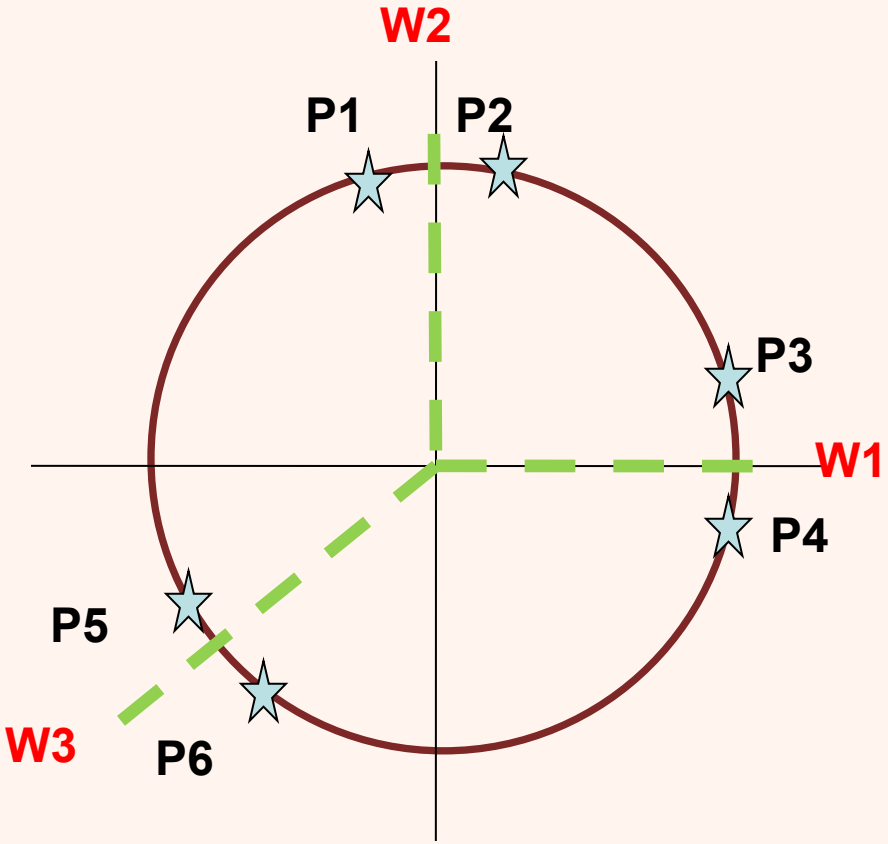
$$= \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} + 0.5\left[\begin{bmatrix} 0.19 \\ 0.98 \end{bmatrix} - \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix}\right] = \begin{bmatrix} 0.45 \\ 0.84 \end{bmatrix}$$

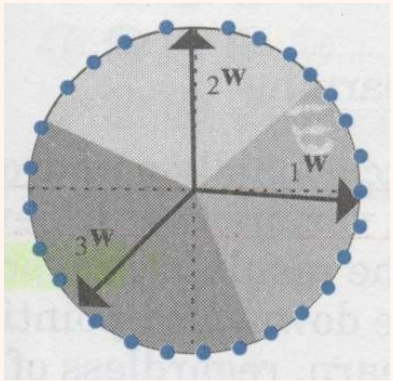
$$\begin{bmatrix} 0.7 & -0.7 \\ 0.45 & 0.84 \\ -1 & 0 \end{bmatrix}$$

ماتریس وزن به‌روز شده

ادامه‌ی مثال

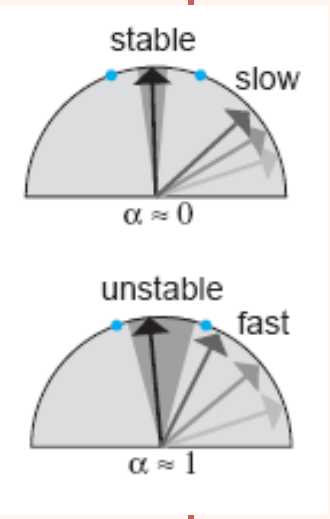
- ورودی بعدی را اعمال کرده بر حسب واحد برنده وزن‌ها را به روز می‌کنیم.
- در نهایت بردارهای شناسه همانند شکل زیر خواهد شد:





مشکلات یادگیری رقابتی

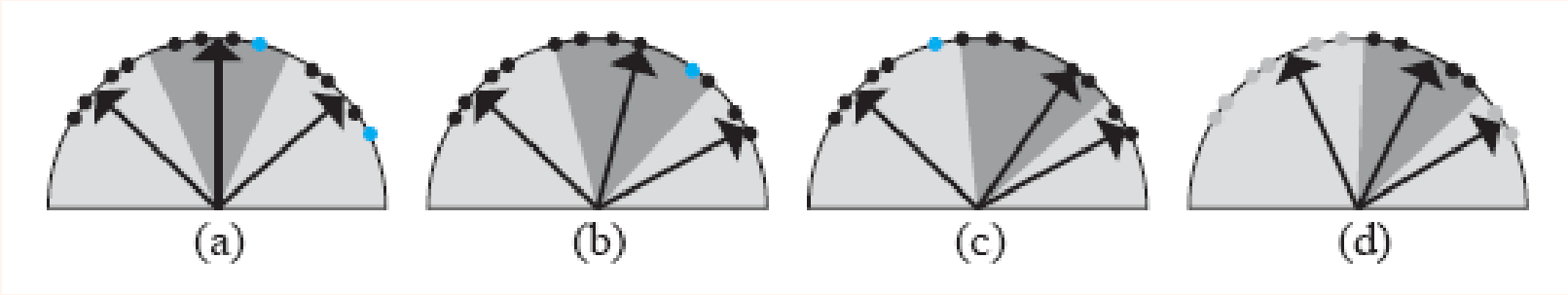
- انتخاب ضریب آموزش η



- $\eta \to 0$ روند همگرایی را کند می‌کند ولی پس از رسیدن به یک بردار شناسه، در آن حالت پایدار می‌ماند.
- η بزرگ انتخاب شود، همگرایی سریع ولی ناپایداری دارد.
- معمولاً برای رسیدن به حدود اولیه، ابتدا η را بزرگ در نظر می‌گیرند، ولی برای افزایش دقت آن‌را کاهش می‌دهند.
  - در صورتی که نیاز باشد، بردارهای ورودی همواره به روز شوند، این شیوه کارایی نخواهد داشت.

مشکلات یادگیری رقابتی (ادامه...)

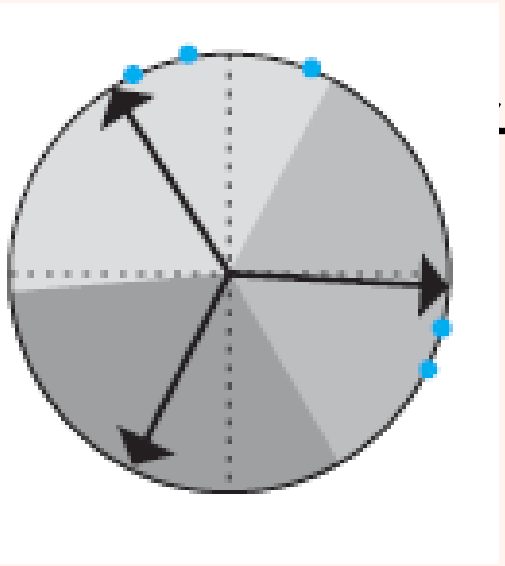
- در صورت نزدیک بودن بردارهای شناسه به یکدیگر، جداسازی امکان‌پذیر نخواهد بود.
  - ممکن است بردارهای شناسه به حریم یکدیگر وارد شوند.
  - در این حالت فرآیند آموزش ناپایدار خواهد بود.



Dead neuron

نورون مرده

- نورون مرده
  - اگر نورونی خیلی دور باشد هیچ‌گاه برنده نخواهد شد و در نتیجه آموزش نخواهد دید. چنین نورونی را نورون مرده می‌گویند.
  - یگ راه برای غلبه بر این مشکل این است که سازوکاری در پیش گرفته شود که شانس نورون‌های برنده برای برنده شدن دوباره کم‌تر شود.
  - این کار با افزودن مقدار منفی به بایاس آن‌ها امکان‌پذیر است. این شیوه «وجدان» خوانده می‌شود.



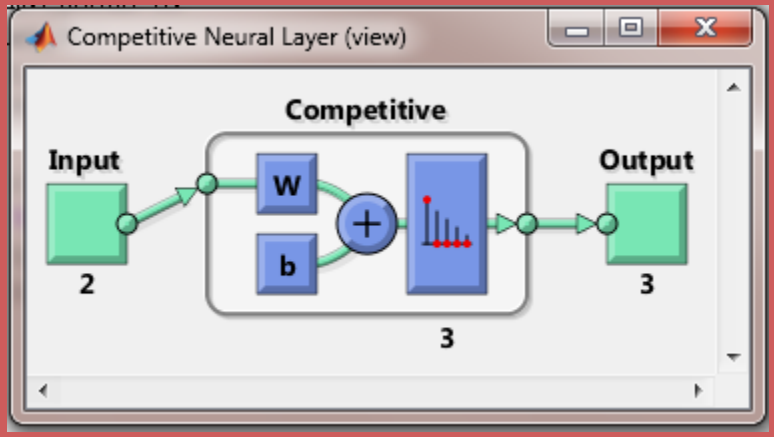
conscience

مشکلات یادگیری رقابتی (ادامه...)

- در صورتی که تعداد کلاس‌ها مشخص نباشد، این شیوه به کار نخواهد آمد.
- این شیوه قابلیت جداسازی کلاس‌های غیر محدب یا گسسته (شامل نواحی گسسته) را ندارد.

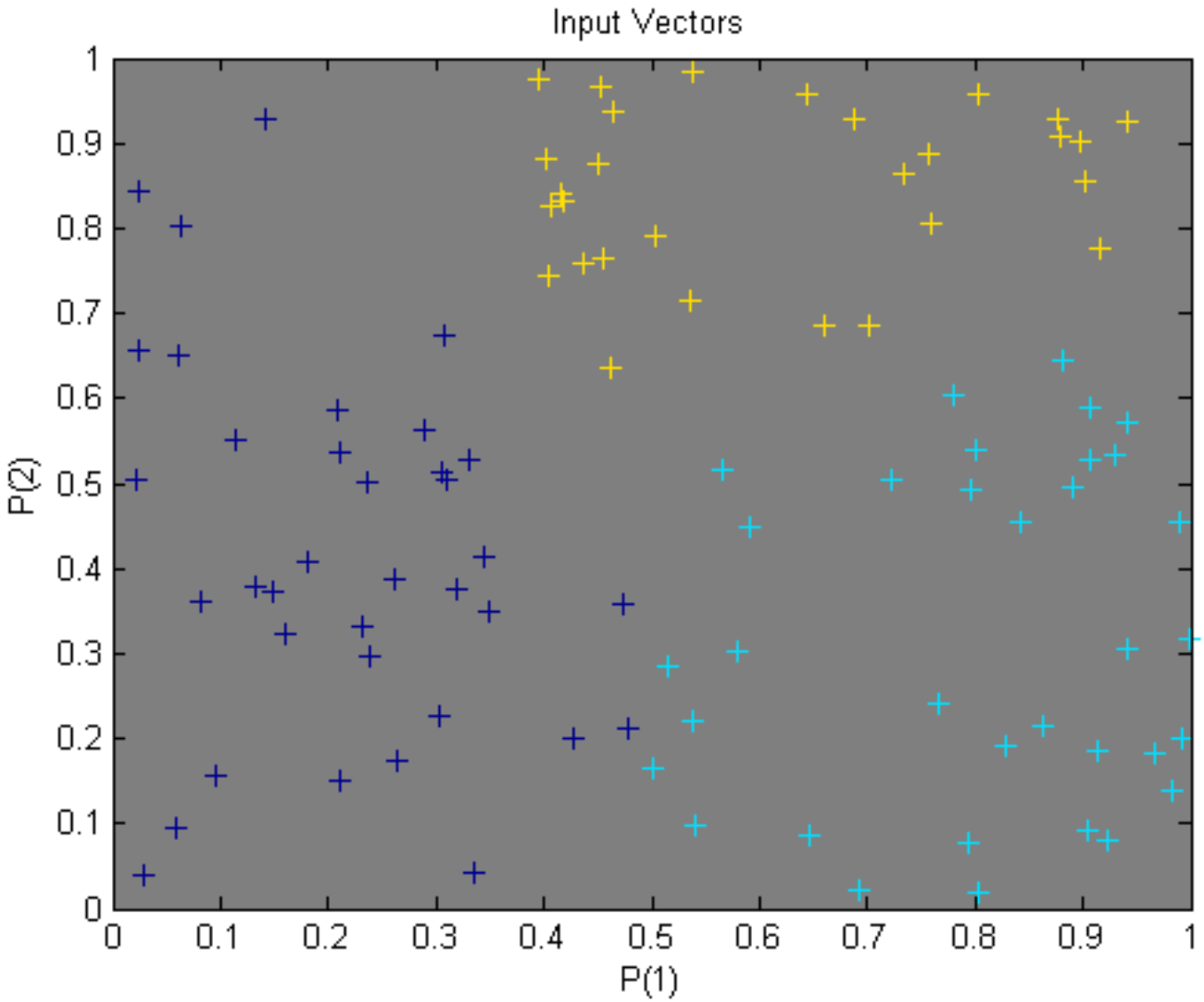
شبکه‌ی رقابتی (مثال)

```
clear all;
P=rand([2 100]);
net=competlayer(3);
% net = newc([0 1;0 1],3);  obsoleted in R2010b NNET 7.0.

net.trainParam.epochs=500;
net = train(net,P);
Y = sim(net,P);
Yc = vec2ind(Y);

plotvec(P,Yc);
title('Input Vectors');
xlabel('P(1)');
ylabel('P(2)');
view(net);
```



`competlayer(numClasses,kohonenLR,conscienceLR);`

شبکه‌ی رقابتی(ادامه‌ی مثال)

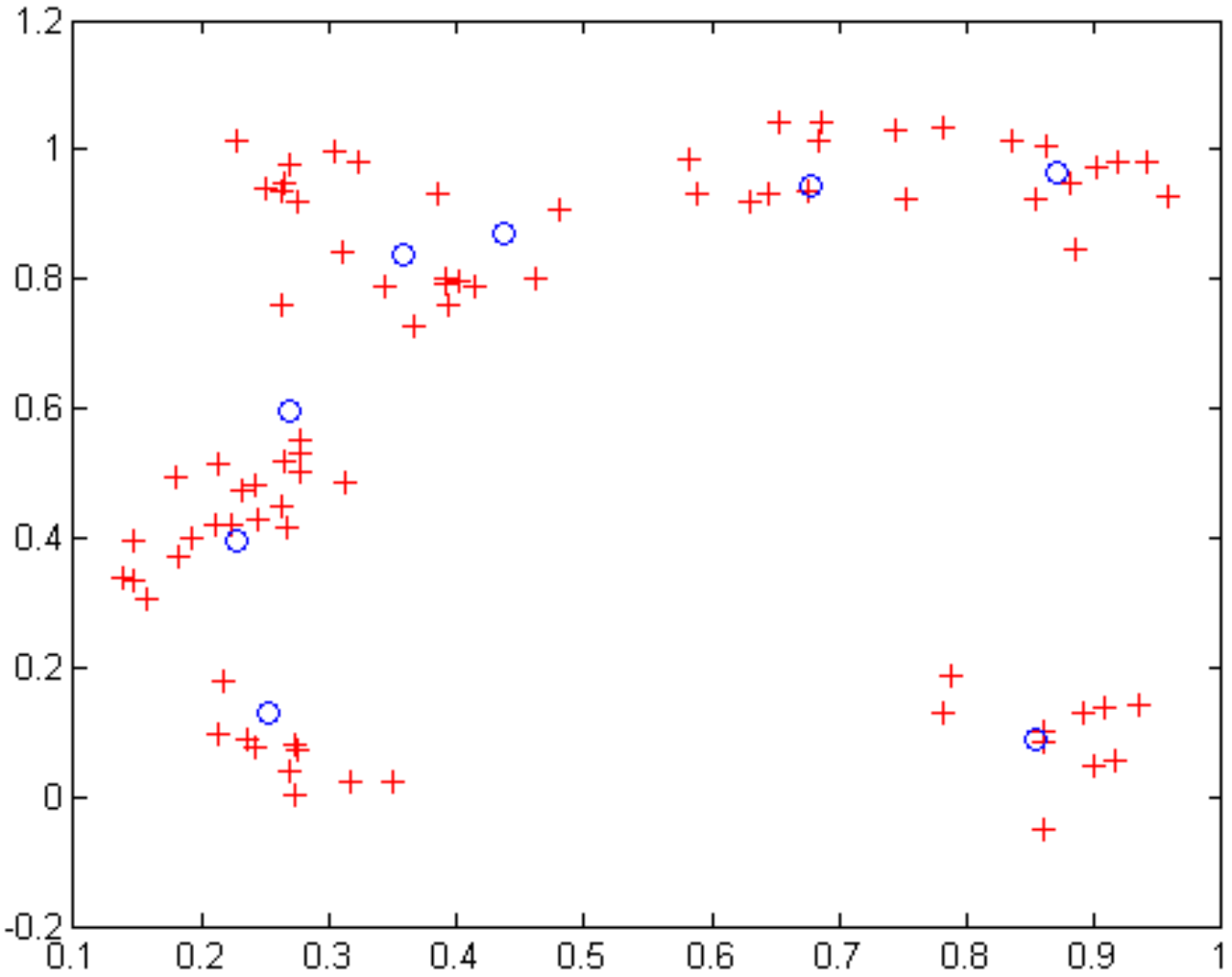


شبکه‌ی رقابتی(مثال)

```
% Create P.
X = [0 1; 0 1];    % Cluster centers to be in these bounds.
clusters = 8;      % This many clusters.
points = 10;       % Number of points in each cluster.
std_dev = 0.05;    % Standard deviation of each cluster.
P = nngenc(X,clusters,points,std_dev);

% Plot P.
plot(P(1,:),P(2,:),'+r');
title('Input Vectors');
xlabel('p(1)');
ylabel('p(2)');
net = newc([0 1;0 1],8,.1);
w = net.IW{1};
plot(P(1,:),P(2,:),'+r');
hold on;
circles = plot(w(:,1),w(:,2),'ob');
```

```
net.trainParam.epochs =10;
net = train(net,P);
w = net.IW{1};
delete(circles);
plot(w(:,1),w(:,2),'ob');

p = [0; 0.2];
a = sim(net,p)
```

شبکه‌ی رقابتی(ادامه‌ی مثال)

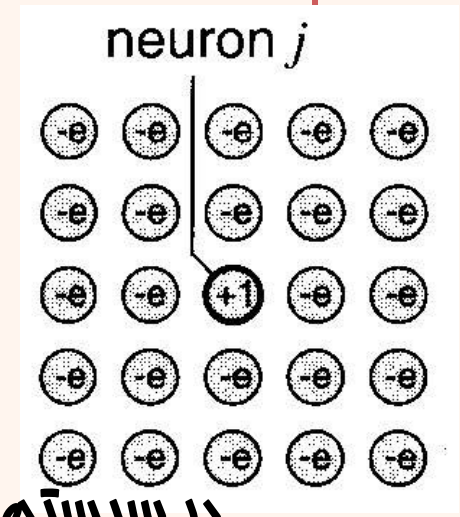


SOM

Compet. Layers in Biology

در مورد ساختار نورون‌ها صحبتی نشد، وزن‌های لایه‌ی دوم شبکه‌ی hamming به صورت زیر است:

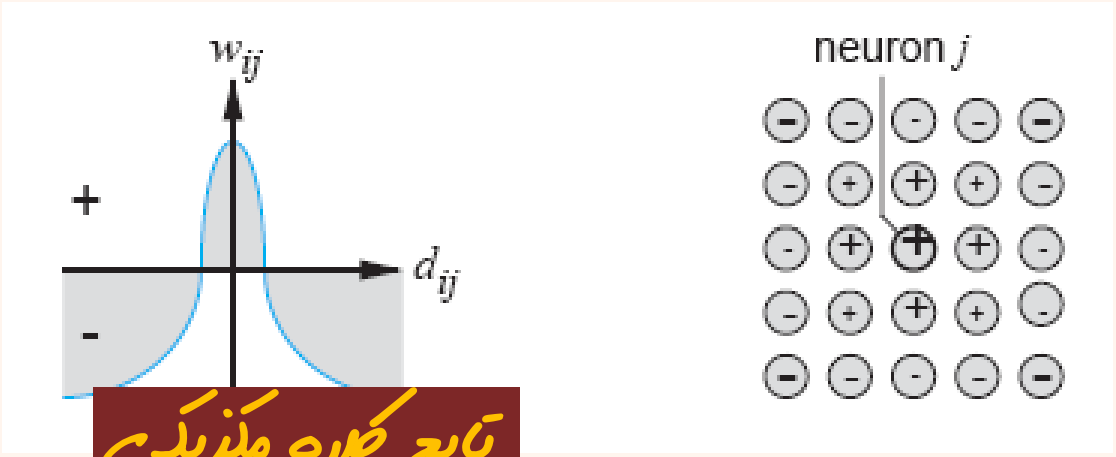
$$w_{ij} = \begin{cases} 1, & \text{if } i = j \\ -\varepsilon, & \text{if } i \neq j \end{cases} \qquad w_{ij} = \begin{cases} 1, & \text{if } d_{ij} = 0 \\ -\varepsilon, & \text{if } d_{ij} \neq 0 \end{cases}$$



Each neuron reinforces itself (center), while inhibiting all other neurons (surround).

On-center/off-surround

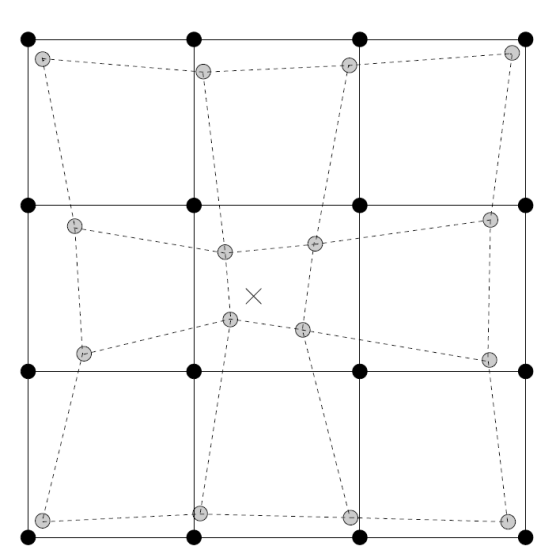
در سیستم‌های بیولوژیکی، بین ناحیه‌ی تحریک و تضعیف مرز دقیقی وجود ندارد.



تابع کلاه مکزیکی

SOM (SOFM)

- به نوعی می‌توان گفت تقلید بهتر از شبکه‌ی عصبی انسان، منجر به ارائه‌ی SOM توسط kohonen شد.
- در این شبکه افزون بر نورون برنده، نورون‌های همسایه‌ی آن هم آموزش می‌بینند.



Marc M. Van Hulle

SOM (SOFM)

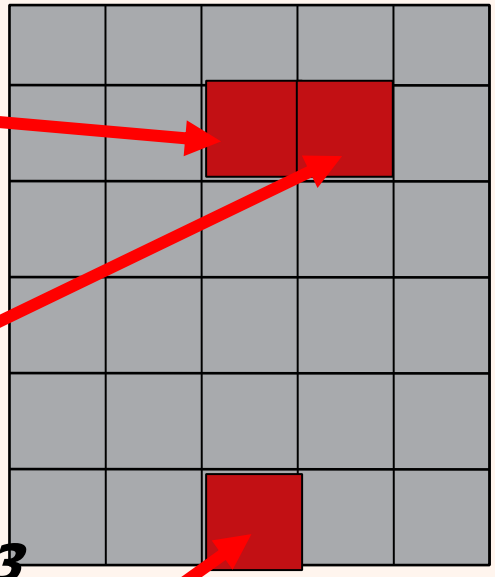
- تعریف همسایگی و به تبع آن توپولوژی شبکه به طرق مختلف امکان پذیر است.
  - یک بعدی
  - دوبعدی (مشبک دو بعدی)
  - سه بعدی
- این شبکه از انواع شبکه‌های بی‌نظارت است.

unsupervised

SOM (SOFM)

- به نوعی هدف، یافتن مفاهیم مشترک است.

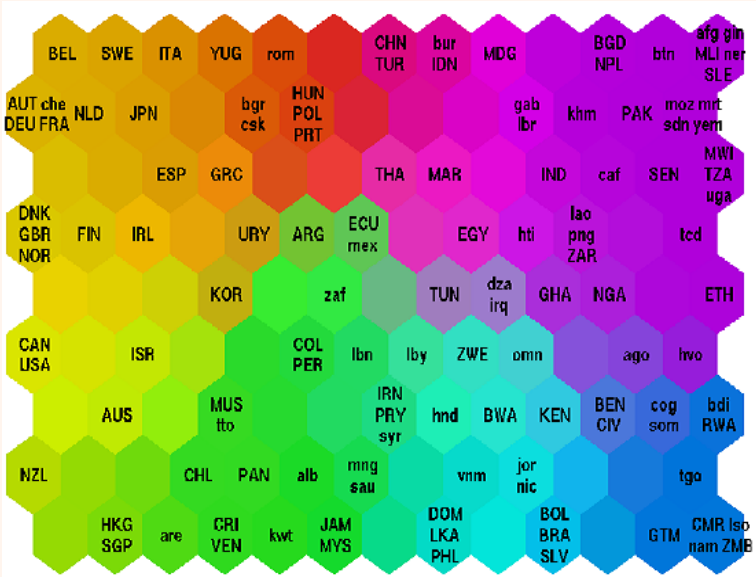
Input Pattern 2

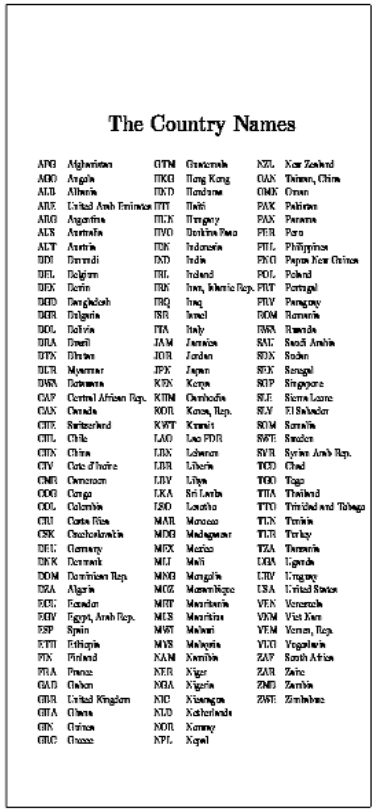


Input Pattern 3

Jing Li Mail: jing.li@lijing.de

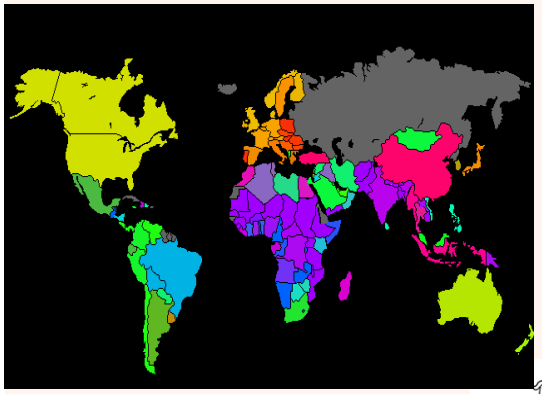
SOM – Result Example





The Country Names

AFG	Afghanistan	GTM	Guatemala	NZL	New Zealand
AGO	Angola	HKG	Hong Kong	OAN	Taiwan, China
ALB	Albania	HND	Honduras	OMN	Oman
ARE	United Arab Emirates	HTI	Haiti	PAK	Pakistan
ARG	Argentina	HUN	Hungary	PAN	Panama
AUS	Australia	IDN	Burkina Faso	PER	Peru
AUT	Austria	IDN	Indonesia	PHL	Philippines
BDI	Burundi	IND	India	PNG	Papua New Guinea
BEL	Belgium	IRL	Ireland	POL	Poland
BEN	Benin	IRN	Iran, Islamic Rep.	PRT	Portugal
BGD	Bangladesh	IRQ	Iraq	PRY	Paraguay
BGR	Bulgaria	ISR	Israel	ROM	Romania
BOL	Bolivia	ITA	Italy	RWA	Rwanda
BRA	Brazil	JAM	Jamaica	SAU	Saudi Arabia
BTN	Bhutan	JOR	Jordan	SDN	Sudan
BUR	Myanmar	JPN	Japan	SEN	Senegal
BWA	Botswana	KEN	Kenya	SGP	Singapore
CAF	Central African Rep.	KHM	Cambodia	SLE	Sierra Leone
CAN	Canada	KOR	Korea, Rep.	SLV	El Salvador
CHE	Switzerland	KWT	Kuwait	SOM	Somalia
CHL	Chile	LAO	Lao PDR	SWE	Sweden
CHN	China	LBN	Lebanon	SYR	Syrian Arab Rep.
CIV	Cote d'Ivoire	LBR	Liberia	TCD	Chad
CMR	Cameroon	LBY	Libya	TGO	Togo
COG	Congo	LKA	Sri Lanka	THA	Thailand
COL	Colombia	LSO	Lesotho	TTO	Trinidad and Tobago
CRI	Costa Rica	MAR	Morocco	TUN	Tunisia
CSK	Czechoslovakia	MDG	Madagascar	TUR	Turkey
DEU	Germany	MEX	Mexico	TZA	Tanzania
DNK	Denmark	MLI	Mali	UGA	Uganda
DOM	Dominican Rep.	MNG	Mongolia	URY	Uruguay
DZA	Algeria	MOZ	Mozambique	USA	United States
ECU	Ecuador	MRT	Mauritania	VEN	Venezuela
EGY	Egypt, Arab Rep.	MUS	Mauritius	VNM	Viet Nam
ESP	Spain	MWI	Malawi	YEM	Yemen, Rep.
ETH	Ethiopia	MYS	Malaysia	YUG	Yugoslavia
FIN	Finland	NAM	Namibia	ZAF	South Africa
FRA	France	NER	Niger	ZAR	Zaire
GAB	Gabon	NGA	Nigeria	ZMB	Zambia
GBR	United Kingdom	NIC	Nicaragua	ZWE	Zimbabwe
GHA	Ghana	NLD	Netherlands		
GIN	Guinea	NOR	Norway		
GRC	Greece	NPL	Nepal		



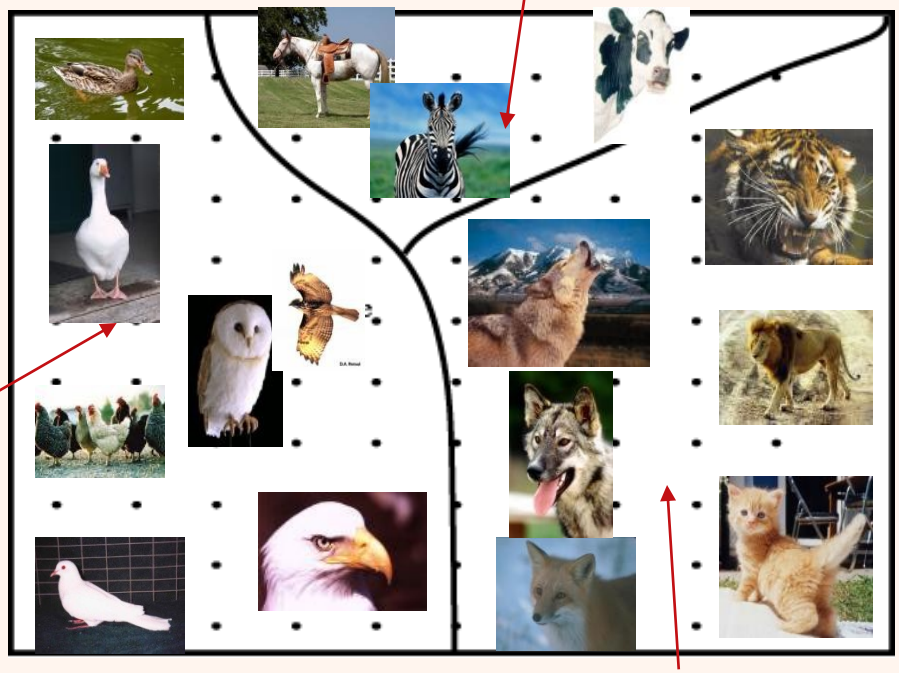
‘Poverty map’ based on 39 indicators from World Bank statistics (1992)

Jing Li Mail: jing.li@lijing.de

SOM – Result Example

Animal names and their attributes

		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
is	Small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	Medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	Big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	Mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	Feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes to	Hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	Run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	Fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	Swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

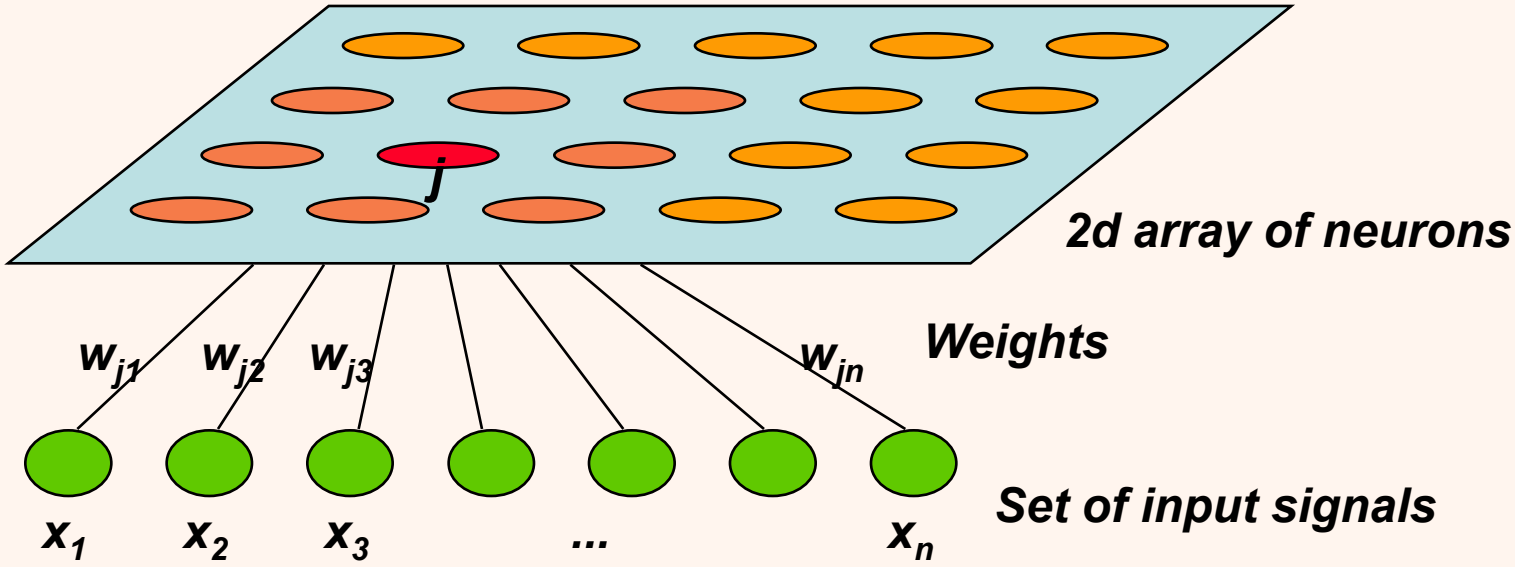


[Teuvo Kohonen 2001] Self-Organizing Maps; Springer;

Jing Li Mail: jing.li@lijing.de

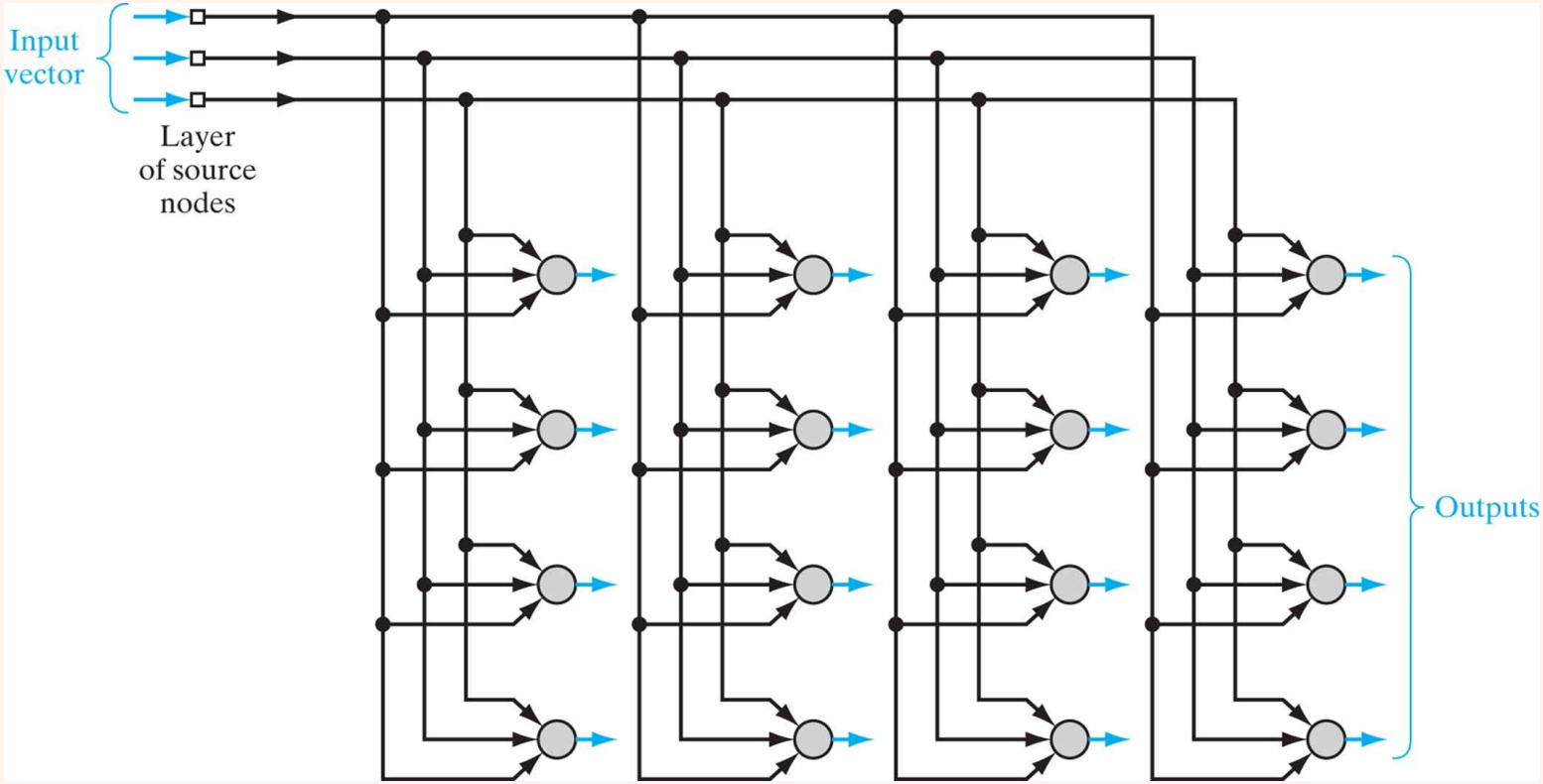
توپولوژی دوبعدی

- در شبکه‌ی SOM با توپولوژی دوبعدی فرض بر این است که مشبکی (Lattice) از نورون‌ها به سیگنال‌های ورودی به شبکه پاسخ می‌دهند.



Jing Li Mail: jing.li@lijing.de

توپولوژی دوبعدی (ادامه...)

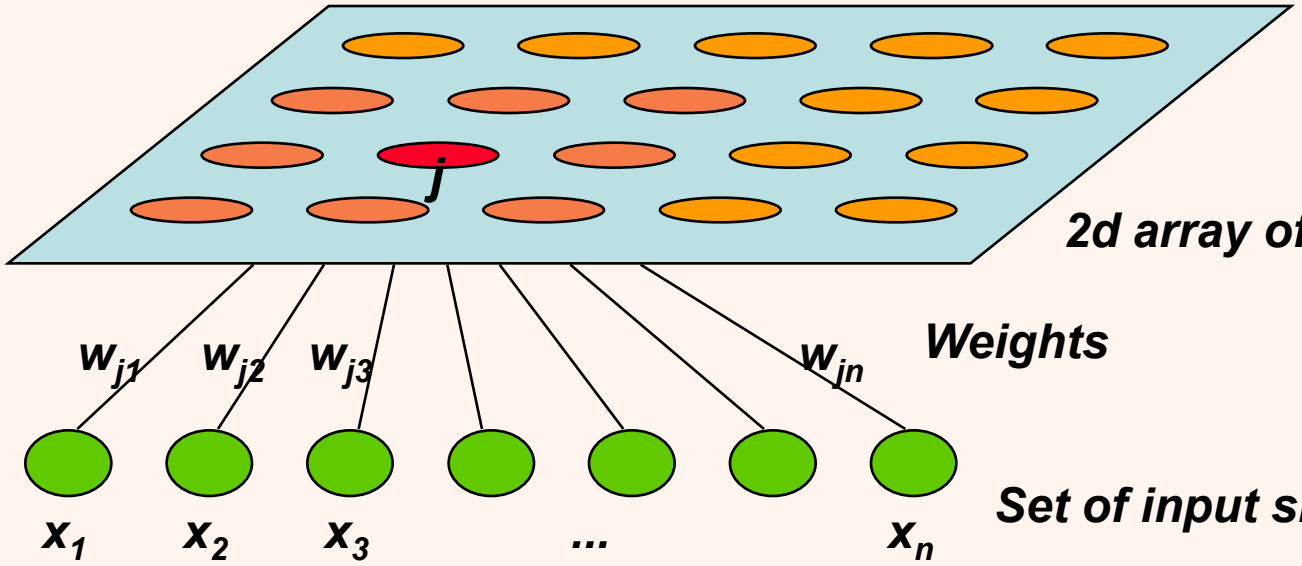


گام‌های مورد استفاده در SOM

- پس از انتخاب اولیه‌ی وزن‌ها سه گام صورت می‌پذیرد:
  - رقابت (competition)
  - همکاری (cooperation)
  - به روزرسانی وزن‌ها (synaptic adaptation)

توپولوژی دوبعدی

- بردار برنده از مشبک انتخاب می‌شود.
- نورون انتخاب شده به همراه همسایگانش با ضریب معینی فعال می‌گردد.
- هر چه به نورون برنده نزدیک‌تر باشیم میزان ضریب آموزش بالاتر خواهد بود.



Jing Li Mail: jing.li@lijing.de

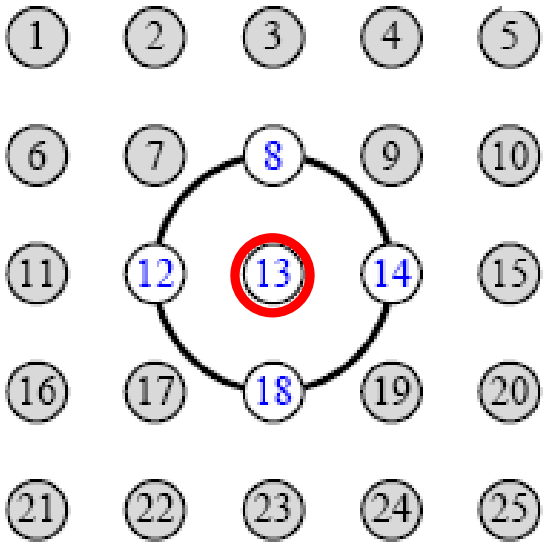
همسایگی

- اگر همسایگی را به وسیله‌ی عبارت زیر نشان‌دهیم، تمامی همسایه‌های یک نورون در فاصله‌ای کم‌تر یا مساوی d در نظر گرفته می‌شوند.

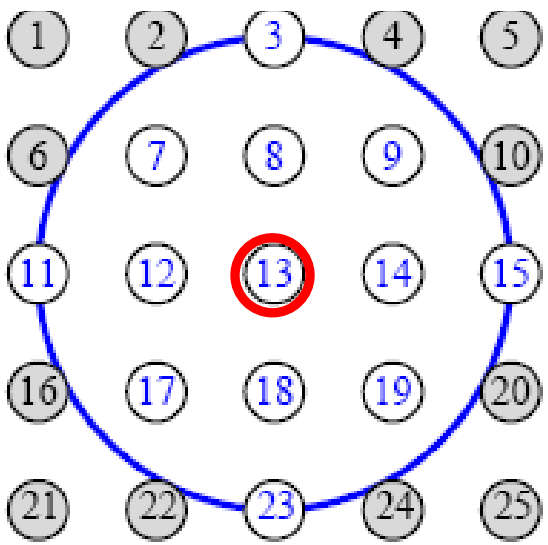
$$N_i(d) = \{j, d_{ij} \leq d\}$$

$N_{13}(1) = \{8, 12, 13, 14, 18\}$ and

$N_{13}(2) = \{3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, 23\}$



$N_{13}(1)$



$N_{13}(2)$

SOM (SOFM)

- در چنین شبکه‌ای دو لایه بیشتر نداریم:
  - لایه‌ی ورودی
  - لایه‌ی خروجی
- هر واحد خروجی خود را با توجه به وزن‌هایی که بدان متصل است نشان می‌دهد.
  - می‌توان گفت شاخص هر خروجی در این صورت وزن‌هایش است.
- ابتدا وزن‌ها را به صورت تصادفی انتخاب می‌کنیم.

SOM (SOFM)

الگوریتم یادگیری

- وزن‌های شاخه‌ها به صورت تصادفی انتخاب می‌شوند.

تعداد گروه‌ها S

$$W_{i,j(n)} \quad 1 \le i \le R \qquad 1 \le j \le S$$

وزن شاخه‌ی ورودی از i به j

- برای هر واحد خروجی یک شعاع همسایگی N در نظر گرفته می‌شود.
- معیار فاصله را می‌توان ضرب داخلی دو بردار و یا فاصله‌ی اقلیدسی در نظر گرفت.

XW_j^T **Max**

$$d_j = \sum_{i=1}^{R} (X_{i(n)} - W_{i,j(n)})^2$$ **Min**

- واحد برنده انتخاب می‌شود: j^*

الگوریتم یادگیری

- به روز رسانی وزن‌های واحد برنده و همسایگانش

$$w_{ij(n+1)} = w_{ij(n)} + \eta[x_{i(n)} - w_{ij(n)}]$$

$$\text{for } j \in N_{j(n)}$$

شعاع همسایگی معمولاً در هر دوره متغیر است

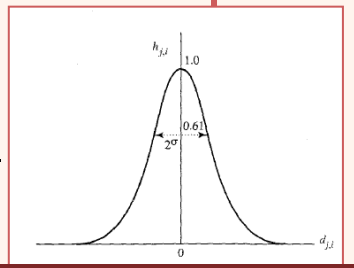
- نرخ آموزش η نیز در این حالت متغیر است برای همسایه‌های نزدیک‌ترها به برنده، مقدار بیشتری دارد.
- و البته نرخ آموزش به تدریج کاهش می‌یابد.

$$0 < \eta(n) \le \eta(n-1) \le 1$$

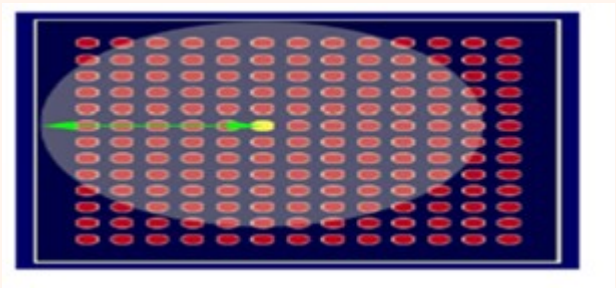
تابع همسایگی

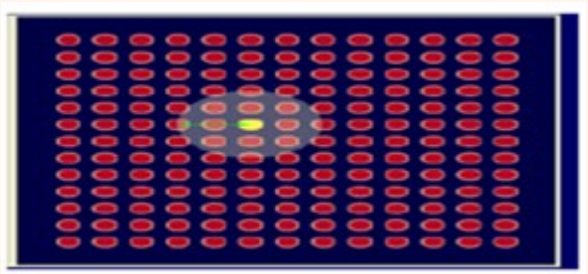
- یکی از مهمترین توابع همسایگی گاوسی است.

$$h_{j,i(x)} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right)$$



- $d_{i,j}$ فاصله‌ی دو نورون را نشان می‌دهد اگر lattice یک بعدی باشد:
  - $|j - i|$
- اگر lattice دو بعدی باشد:
  - $\| r_j - r_i \|$

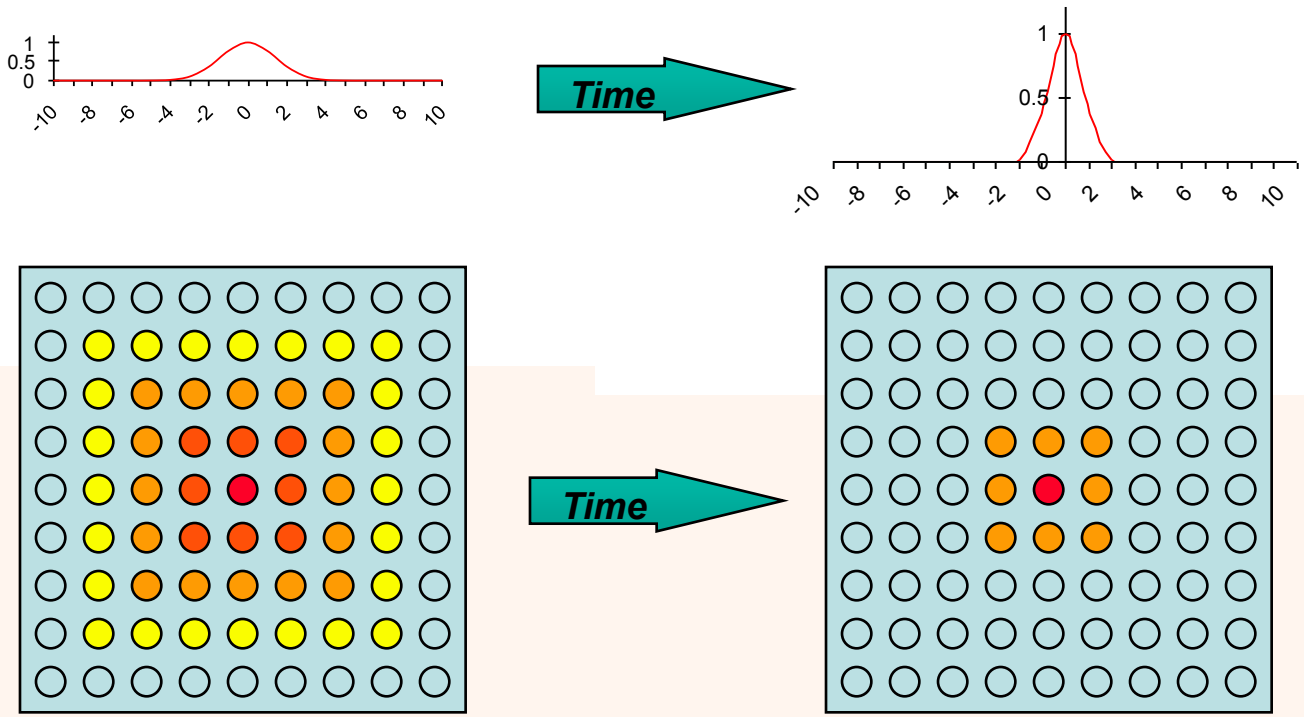




تابع همسایگی

- تابع همسایگی از شعاع همسایگی بزرگ شروع کند و به تدریج تمرکز تنها روی واحد برنده باقی می‌ماند.

$$h_{j,i(x)} = \exp\left(-{d_{j,i}^{2}}/{2\sigma^{2}}\right)$$



تابع همسایگی

- تعداد همسایگان هم‌روند با زمان تغییر می‌کند.
- اگر نورون i برنده باشد d_{ij} همان فاصله‌ی نورون j از نورون برنده(i) است.

$$h_{j,i}(x) = \exp\left(-d_{j,i}^2 / 2\sigma^2\right)$$

- برای این منظور، انحراف معیار تابع گاوسی را متغیر در نظر گرفته می‌شود.

$$\sigma(n) = \sigma_0 \exp\left(-n/\tau_1\right) \quad n=0,1,2,\ldots$$

$$h_{j,i(x)}(n) = \exp\left(-d_{j,i}^2 / 2\sigma^2(n)\right) \quad n = 0,1,2,\ldots$$

به روزرسانی وزن‌ها

- بر طبق روابط به دست آمده خواهیم داشت:

$$w_j(n+1) = w_j(n) + \eta(n)\, h_{ij(x)}(n)\, \left(x - w_j(n)\right)$$

- نرخ یادگیری را نیز به صورت نمایی متغیر در نظر گرفته می‌شود.

$$\eta(n) = \eta_0 \exp\left(-n/\tau_2\right)$$

به روزرسانی وزن‌ها(ادامه...)

$$w_j(n+1) = w_j(n) + \eta(n)\, h_{ij(x)}(n)\, \left(x - w_j(n)\right)$$

$$h_{j,i(x)}(n) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(n)}\right) \quad n = 0,1,2,\ldots$$

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right) \quad n = 0,1,2,\ldots$$

$$\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_2}\right) \quad n = 0,1,2,\ldots$$

به روزرسانی وزن‌ها(ادامه...)

مقادیر η_0، σ_0 ، τ_1 و τ_2 به صورت تجربی در بازه‌های زیر خواهند بود:

$$\eta_0 = 0.1 \qquad \tau_2 = 1000$$

این مقدار تدریجی کاهش می‌یابد اما نهایتاً بالاتر از ۰.۰۱ خواهد بود

- مقداری که برای σ_0 در نظر گرفته می‌شود برابر با شعاع مشبک است.
- بنابراین این مقدار وابسته به چیدمان نورون‌هاست.
- برای τ_1 مقدار زیر را در نظر می‌گیرند:

$$\tau_1 = \frac{1000}{\log \sigma_0}$$

فازهای یادگیری

- در SOM هر بردار که برنده باشد وزن‌های خود و همسایگانش به نسبت همسایگی، تنظیم می‌شوند.
  - هدف
    - مکان جغرافیایی واحدها مشخص می‌شود.
    - شعاع همسایگی به اندازه‌ی تمامی واحدها شروع و سپس رو به کاهش می‌گذارد.
    - تعداد تکرارها حدود ۱۰۰۰ در نظر گرفته می‌شود.
    - در طی تکرارهای متوالی η کوچک و کوچک‌تر می‌شود.

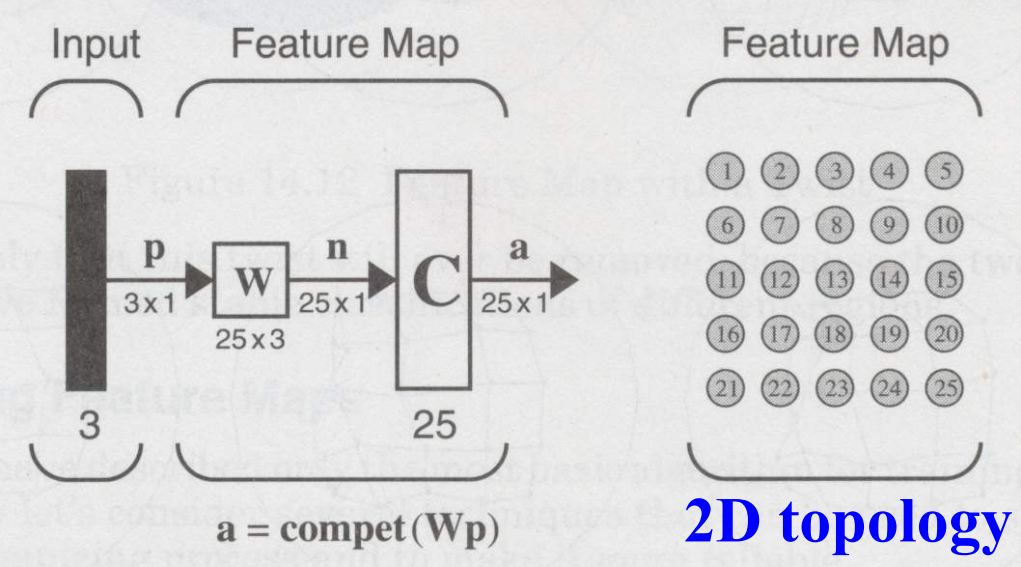
Ordering وارد جزییات نمی‌شود. به‌گونه‌ای فرم زمخت یا Coarse را نشان می‌دهد.

فازهای یادگیری(ادامه...)

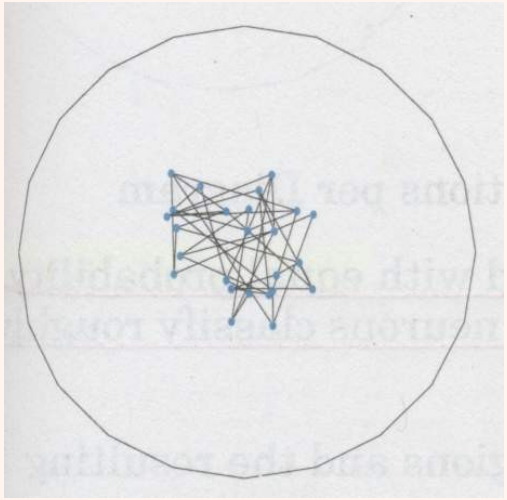
Convergence(tuning)

- هدف
  - جزییات تقسیم‌بندی گروه‌ها مشخص می‌شود.
  - شعاع همسایگی به خود واحد برنده محدود خواهد شد.
  - معمولا تعداد تکرارها ۵۰۰ برابر تعداد واحدهای خروجی است.
    - تعداد گروه‌ها اهمیت دارد.
  - نرخ آموزش کم و در حد ۰.۰۱ در نظر گرفته می‌شود.
  - در این حالت جزییات هر واحد و نه همسایه‌ها اهمیت دارد.
- نکته:
  - بردارهای وزن اولیه مقادیر کوچکی انتخاب می‌شوند.
  - اگر خیلی پراکنده باشند، مسأله‌ی واحد مرده پیش می‌آید.

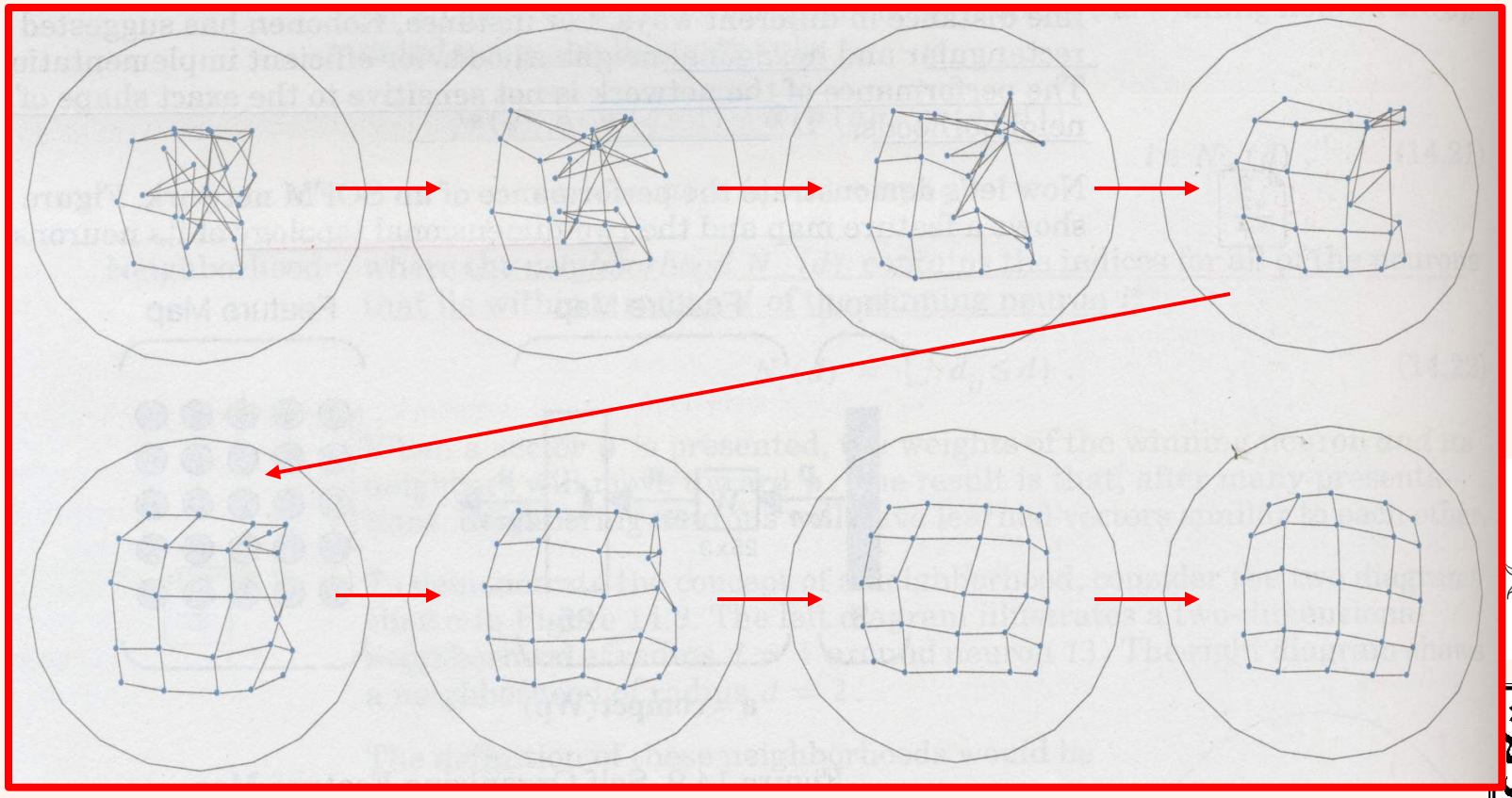
SOM (SOFM)



initial weight vectors



SOM (SOFM)



250 iterations per diagram

Renormalized SOM

- گفته شد که در ابتدا شعاع همسایگی بزرگ در نظر گرفته می‌شود. در این صورت با توجه به پارامترهای زیادی که درگیر هستند، آموزش کند خواهد شد.
- در این شیوه پیشنهاد می‌شود، شعاع همسایگی ثابت در نظر گرفته شود، اما نورون‌ها به تدریج در فرآیند آموزش دخیل شوند.

Luttrell, S. (1989). "Hierarchical vector quantisation (image compression)." Communications, Speech and Vision, IEE Proceedings I 136(6): 405-413.

جمع‌بندی SOM

- گام‌های زیر می‌باید انجام شود:

۱) مقداردهی اولیه (initialization)

- وزن‌ها به صورت تصادفی انتخاب می‌شوند، تنها محدودیت این است که وزن‌ها باید متفاوت باشند. مناسب‌تر مقدارهای انتخاب شده اندازه‌ی کوچکی داشته باشند،
- همچنین می‌توان از ورودی‌ها به عنوان وزن اولیه بهره جست.

۲) نمونه‌برداری (sampling)

- یک الگوی ورودی به شبکه اعمال می‌شود.

جمع‌بندی SOM (ادامه...)

۳) تطبیق شباهت (similarity matching)

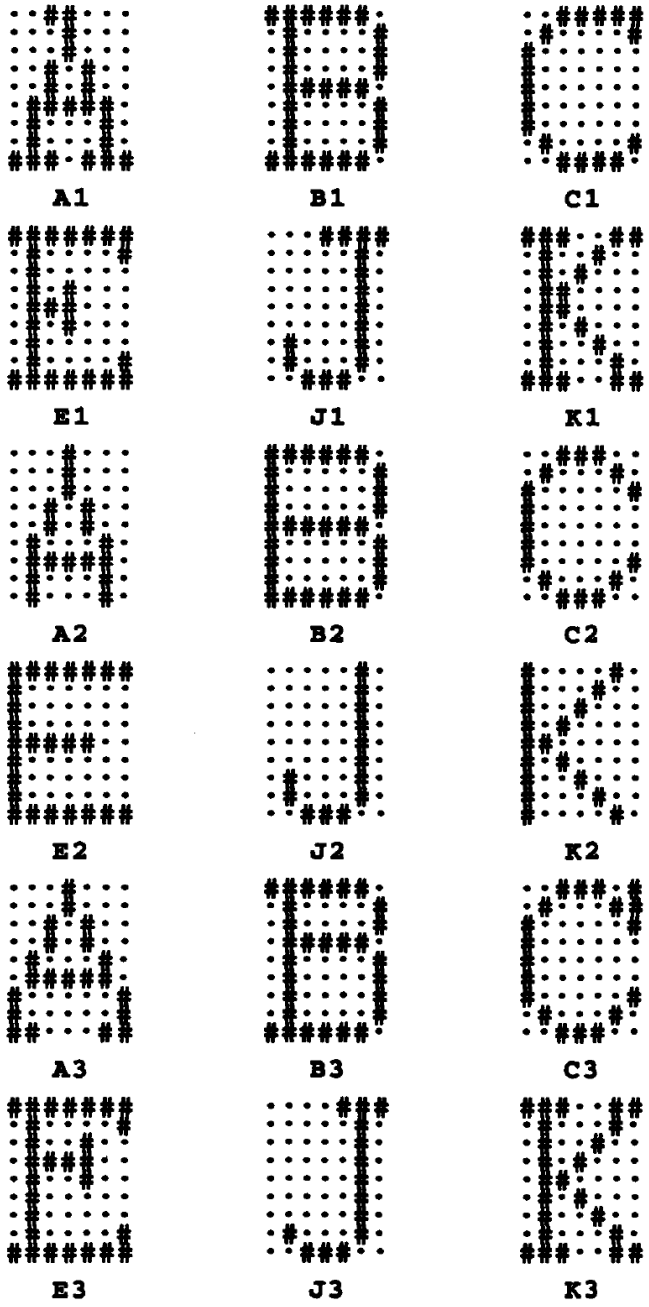
- نورون برنده انتخاب شود.

۴) به‌روزرسانی (updating)

- طبق قانون آموزش مطرح شده، نورون برنده و همسایه‌هایش آموزش می‌بینند.

۵) از گام دو مراحل از سر گرفته شوند، تا شبکه به پایداری برسد.

خوشه‌بندی کاراکتر



raining input patterns for character recognition examples.

- ۲۱ الگوی ورودی و
- ۲۵ خوشه در نظر گرفته شده است.

- کاهش نرخ یادگیری به صورت خطی(از ۰.۶ تا ۰.۰۱)

- استفاده از سه توپولوژی مختلف

خوشه‌بندی بدون ساختار

no topological structure

UNIT	PATTERNS
3	C1, C2, C3
13	B1, B3, D1, D3, E1, K1, K3, E3
16	A1, A2, A3
18	J1, J2, J3
24	B2, D2, E2, K2

خوشه‌بندی اشتباه

ساختار خطی

linear structure (R=1)

UNIT	PATTERNS	UNIT	PATTERNS
6	K2	20	C1, C2, C3
10	J1, J2, J3	22	D2
14	E1, E3	23	B2, E2
16	K1, K3	25	A1, A2, A3
18	B1, B3, D1, D3		

خوشبندی اشتباه

ساختار لوزی

diamond structure

i\j	1	2	3	4	5
1		J1, J2, J3		D2	
2	C1, C2, C3		D1, D3		B2, E2
3		B1		K2	
4			E1, E3, B3		A3
5		K1, K3		A1, A2	

خوشبندی اشتباه

Neural Networks. Vol. 1, pp. 289–293, 1988
Printed in the USA.

0893-6080/88 $3.00 + .00

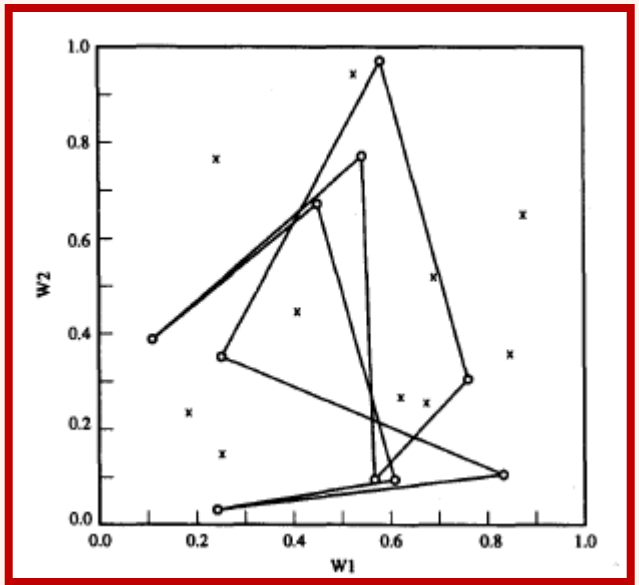
ORIGINAL CONTRIBUTION

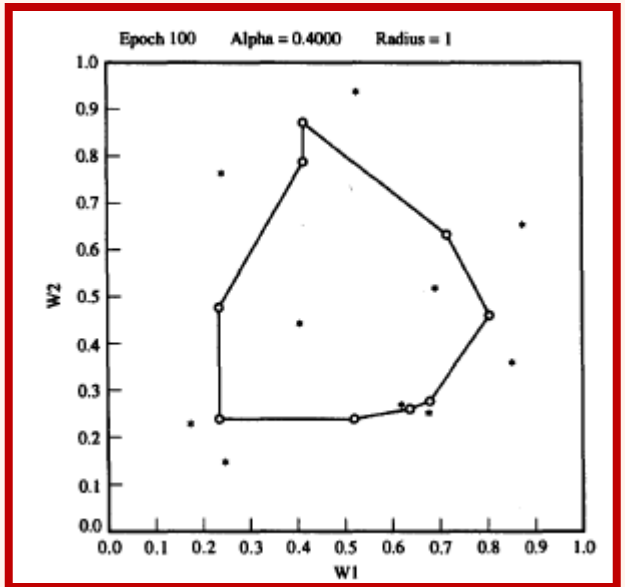
Self-Organizing Feature Maps and the Travelling Salesman Problem

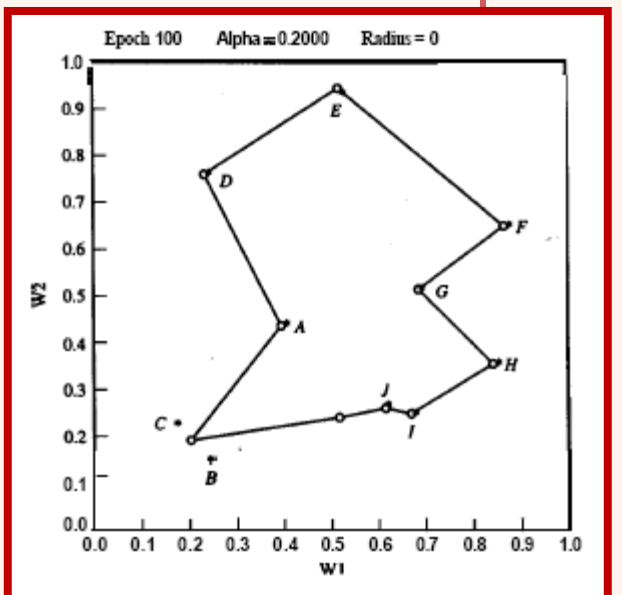
BERNARD ANGÉNIOL, GAËL DE LA CROIX VAUBOIS AND JEAN-YVES LE TEXIER

Thomson CSF/DSE

(*Received March* 1988; *revised and accepted* May 1988)







فروشنده‌ی دوره‌گرد

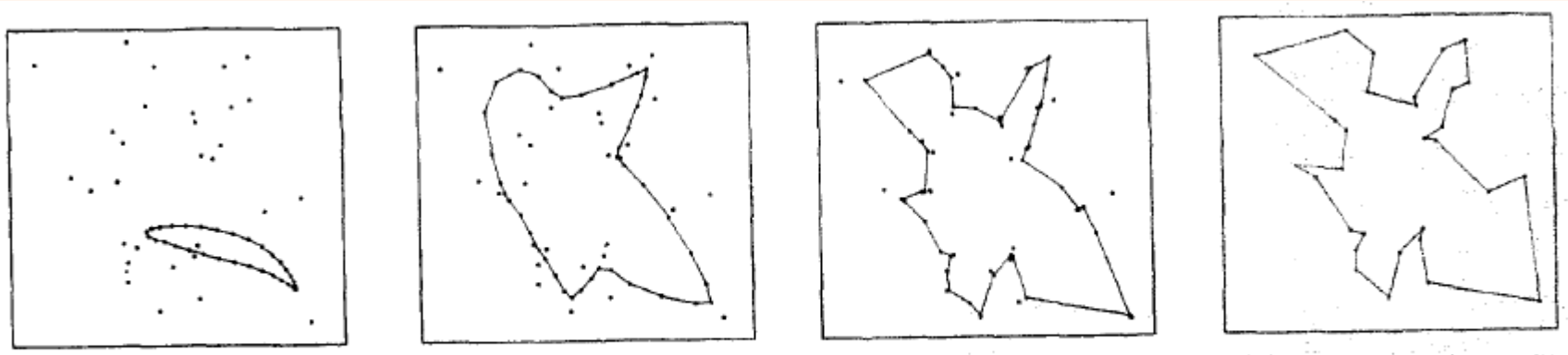


FIGURE 1. Evolution of the ring on a set of 30 cities used by Tank and Hopfield (1985). The solution shown here is the same as Lin–Kernighan's, and happens with probability 1.5×10^{-3} for $\alpha = 0.2$.

توپولوژی ring

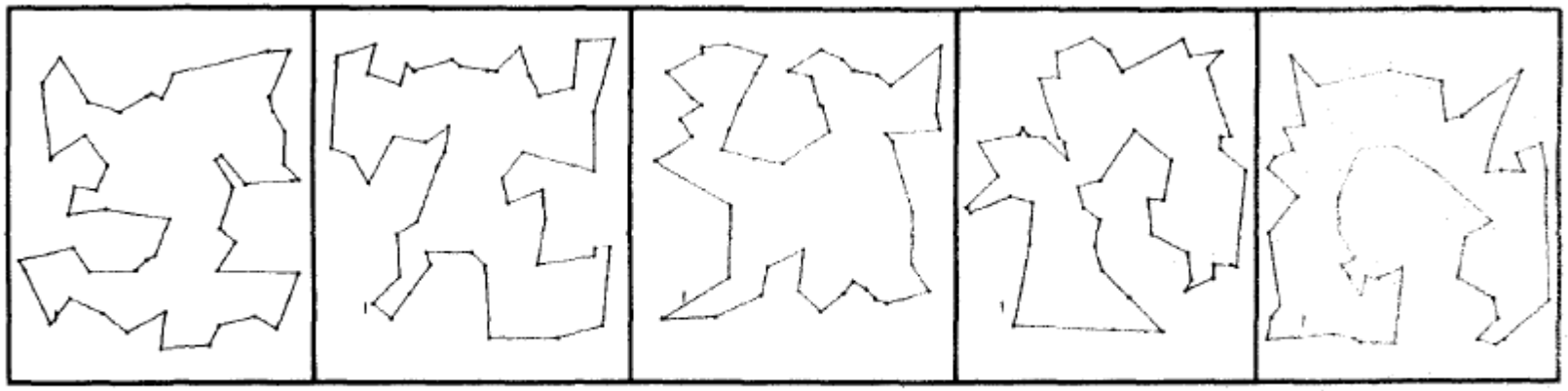
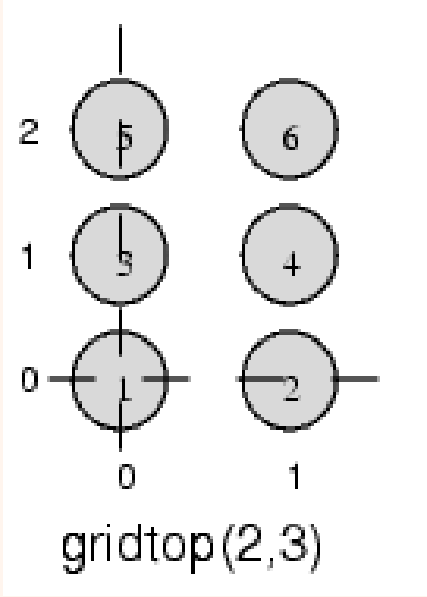


FIGURE 3. The best solution found on the five sets of 50 cities (Table 1c).

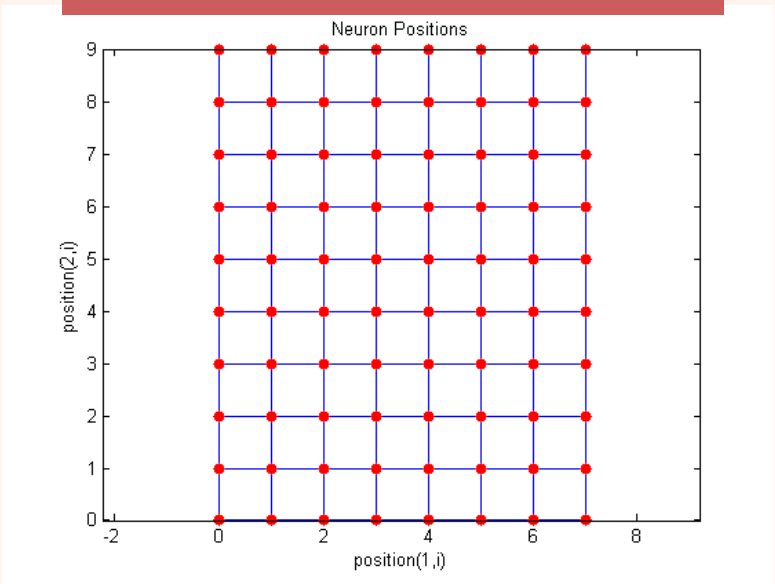
SOM در Matlab

- توپولوژی‌های دو بعدی مورد استفاده عبارتند از:
- مستطیلی (Gridtop)
- شش‌ضلعی (hextop)
- تصادفی (randtop)

```
pos = gridtop(2,3);
```

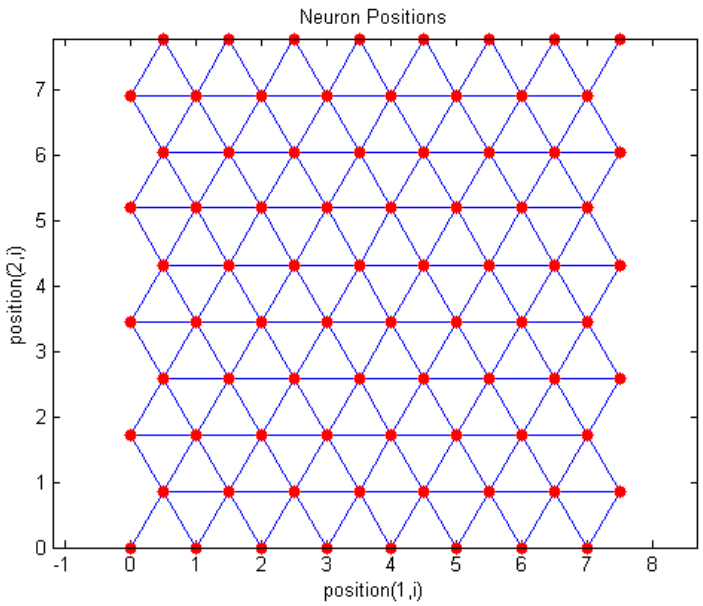



```
pos = gridtop(16,12);
plotsom(pos);
```

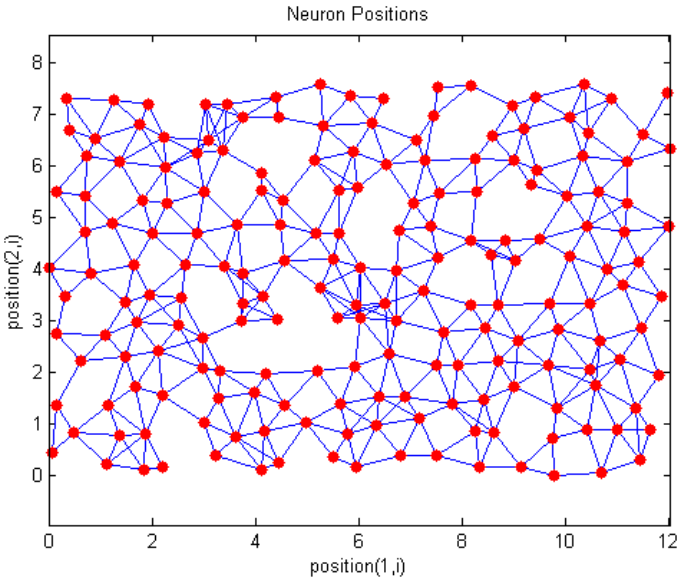


توپولوژیهای دوبعدی مورد استفاده

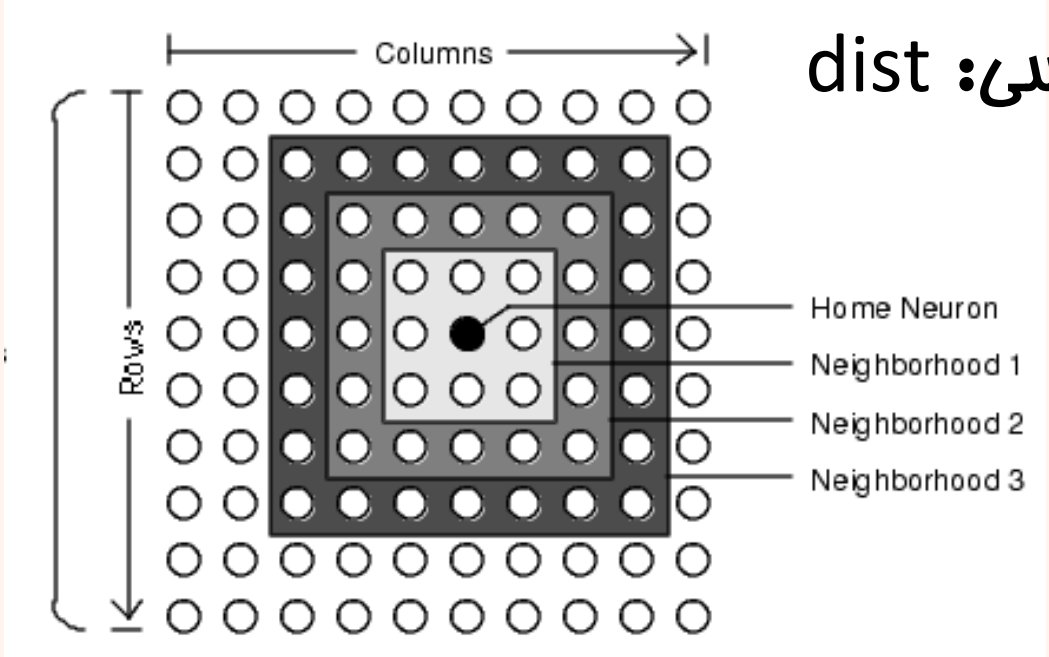
```
pos = hextop(8,10);
plotsom(pos);
```



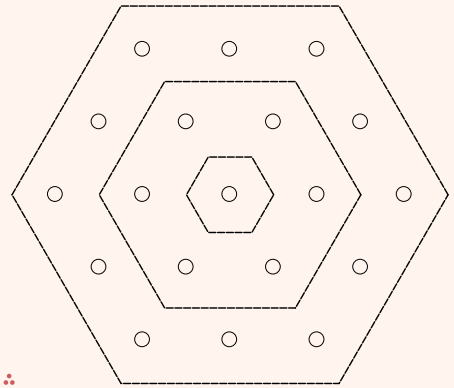

```
pos = randtop(16,12);
 plotsom(pos);
```



توابع فاصله



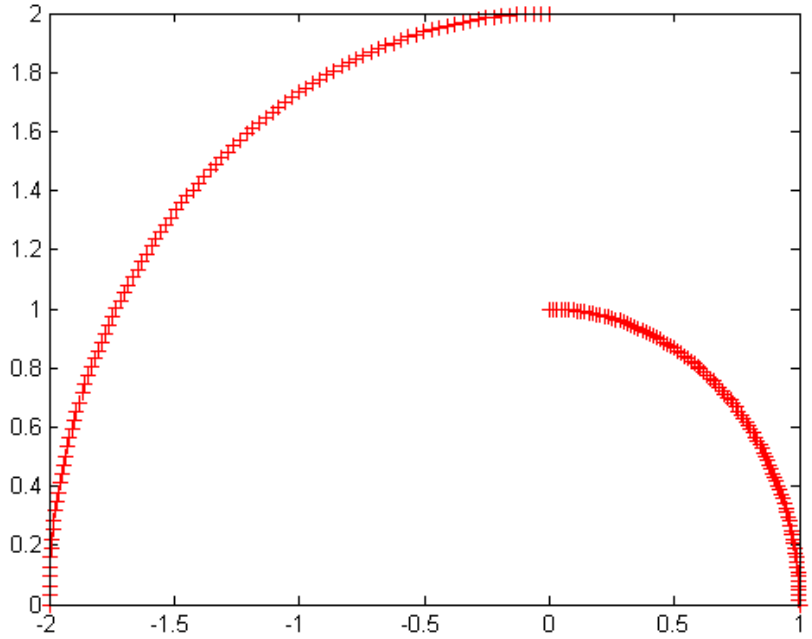
- فاصله‌ی اقلیدسی: dist
- Boxdist
- linkdist
- mandist: sum(abs(x-y))

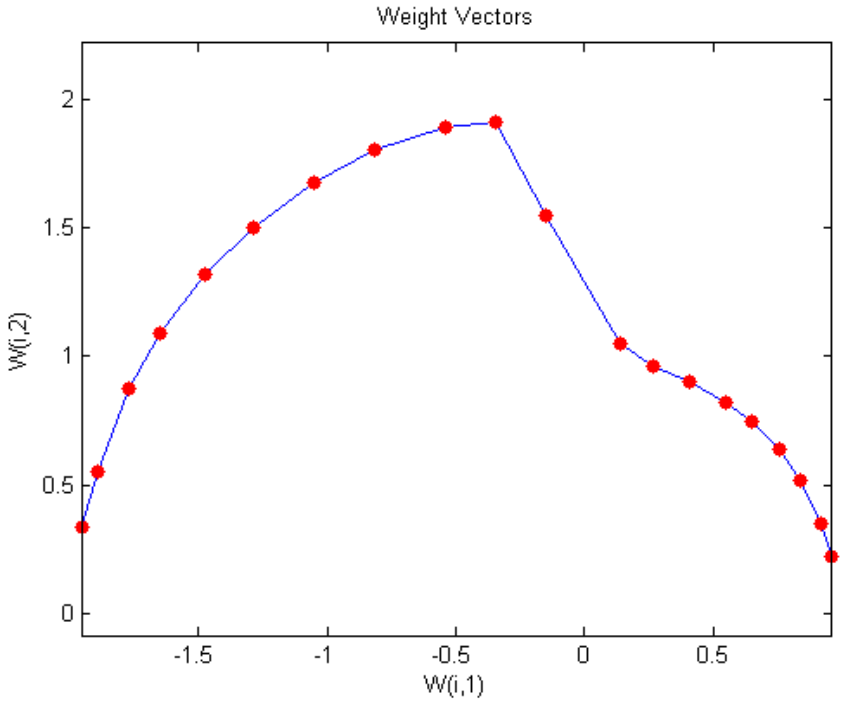


مثال

```
angles = 0:0.5*pi/99:0.5*pi;
P1 = [sin(angles); cos(angles)];
P2 = [-sin(angles); cos(angles)];
P=[P1,2.*P2]
plot(P(1,:),P(2,:),'+r')

net=selforgmap([20]);
%net = newsom([0 1;0 1],[20]);
net.trainParam.epochs = 10;
net = train(net,P);
figure;
plotsom(net.iw{1,1},net.layers{1}.distances)
p = [1;0];
a = sim(net,p)
```

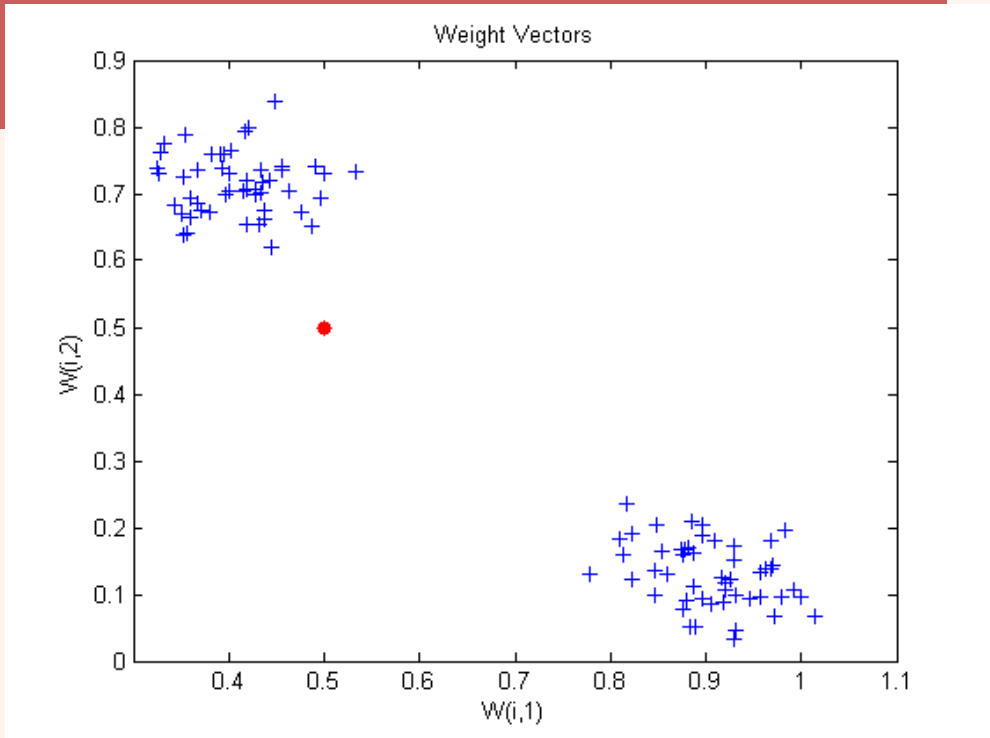
selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn)


2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-2
-1.5
-1
-0.5
0
0.5
1

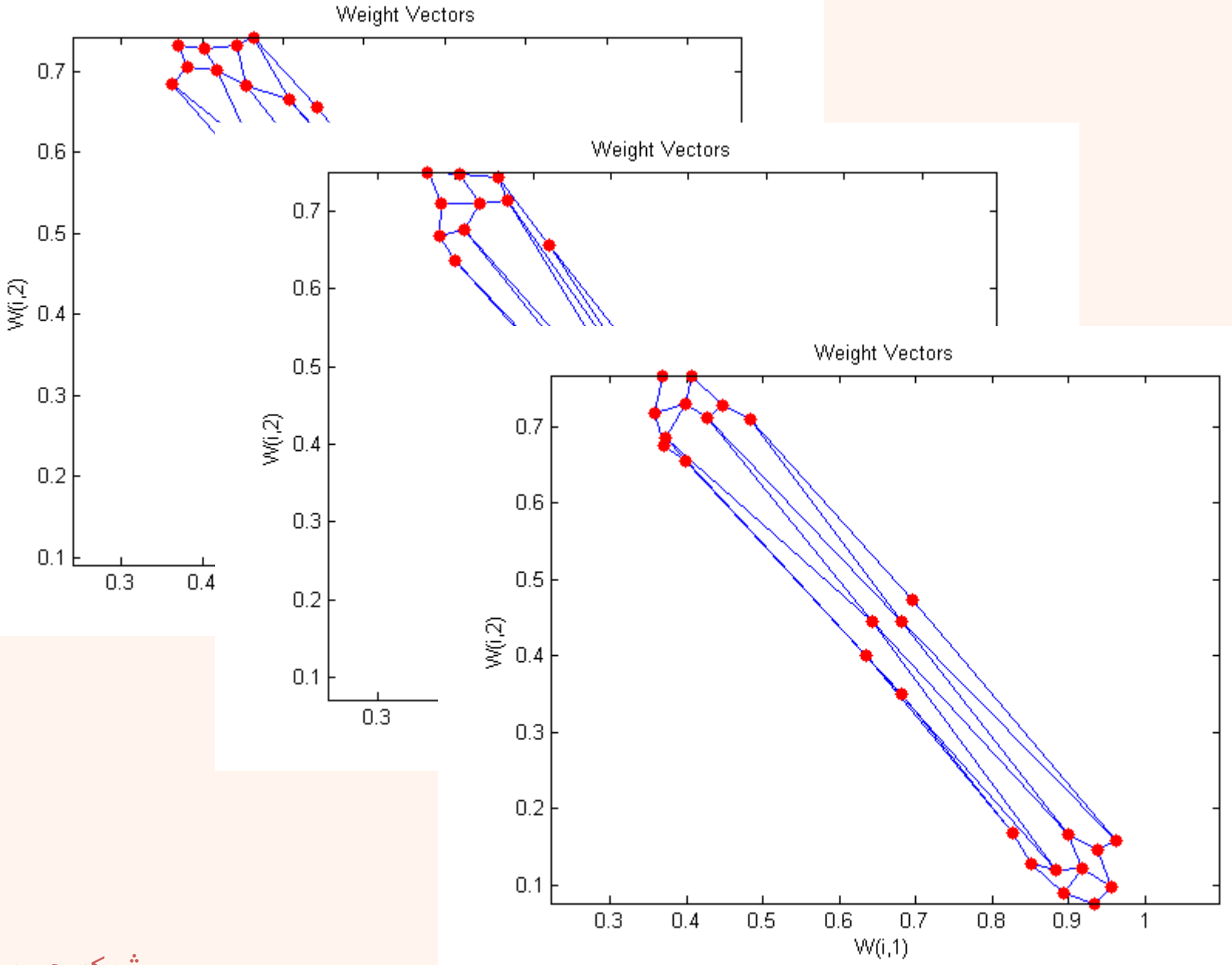

Weight Vectors
W(i,2)
W(i,1)
2
1.5
1
0.5
0
-1.5
-1
-0.5
0
0.5

مثال

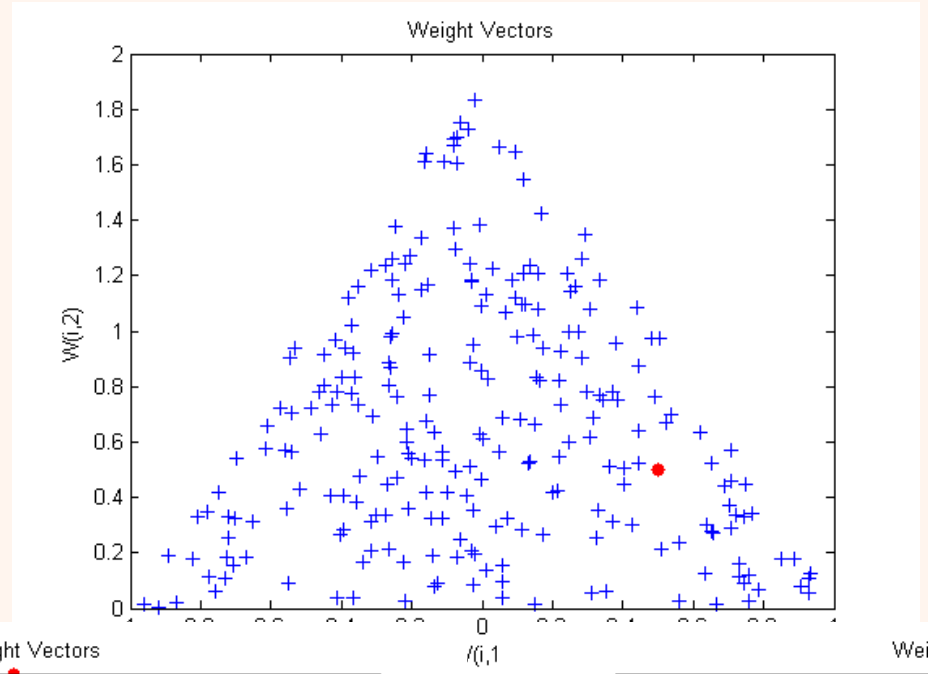
```
figure;
plot(P(1,:),P(2,:),'+b')
net = newsom([0 1; 0 1],[5 5],'gridtop');
hold on;
plotsom(net.iw{1,1},net.layers{1}.distances)
net.trainParam.epochs = 10;
net = train(net,P);
figure;
plotsom(net.iw{1,1},net.layers{1}.distances)
p = [0.5;0.3];
a = sim(net,p)
```

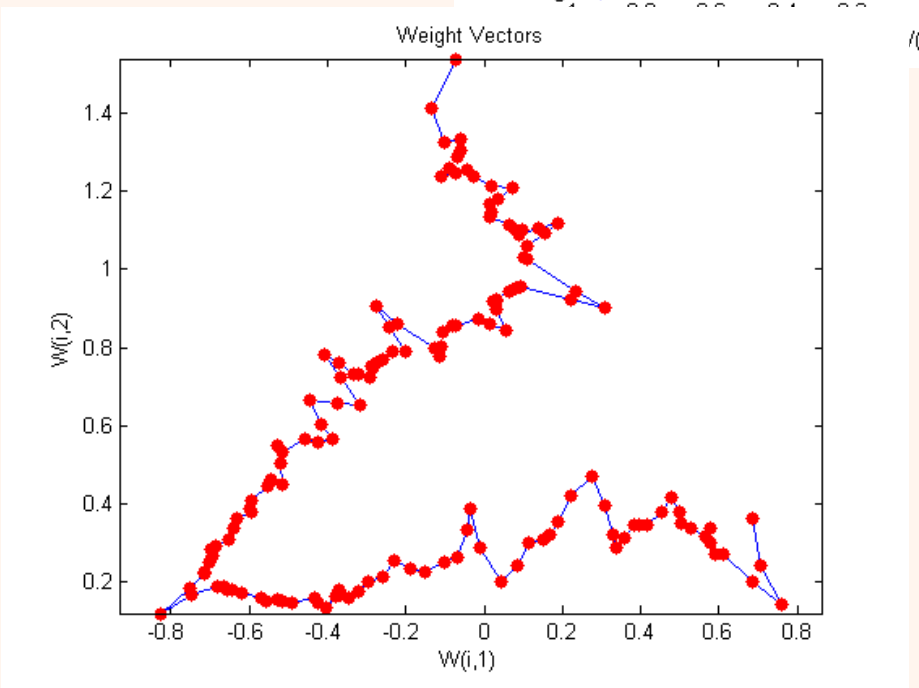


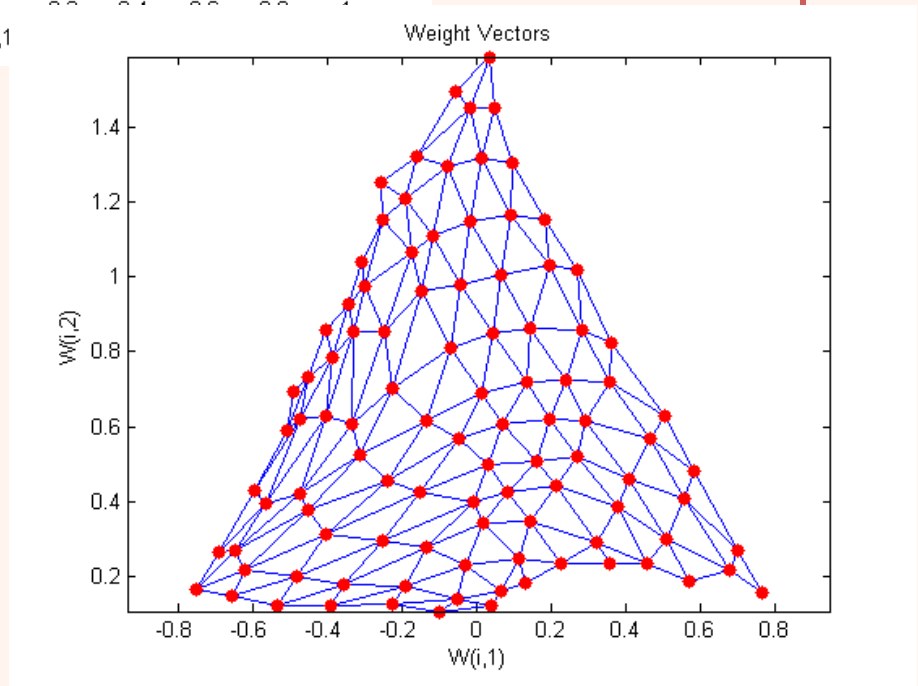
ادامه‌ی مثال

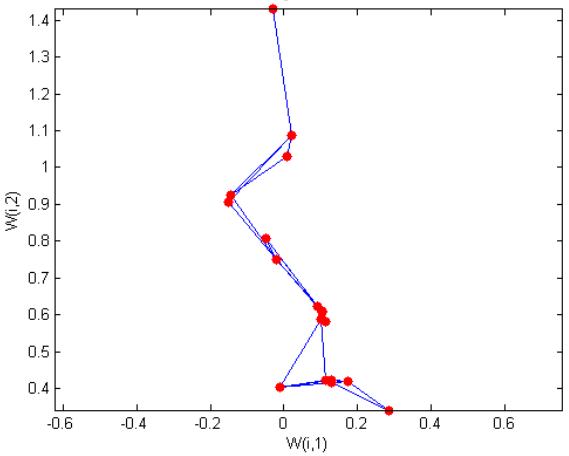


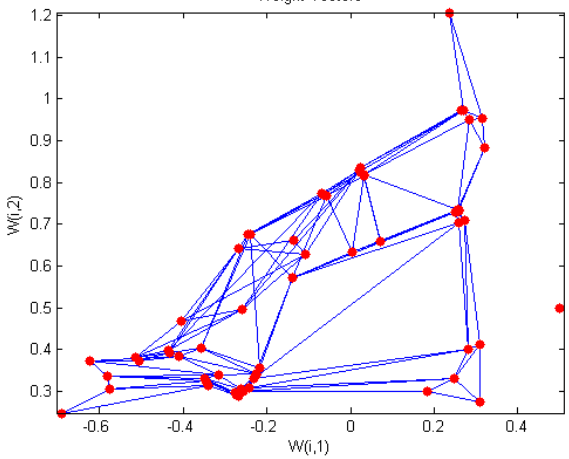
مثال

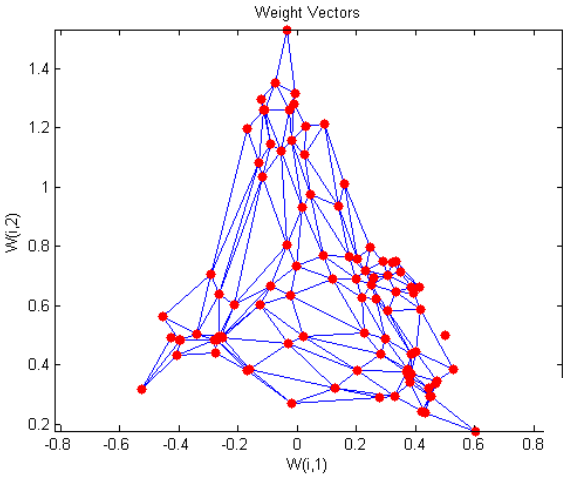

Weight Vectors
W(i,2)
/(i,1

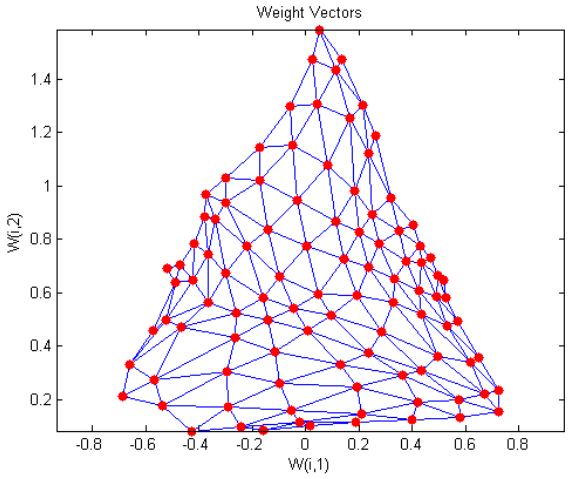

Weight Vectors
W(i,2)
W(i,1)

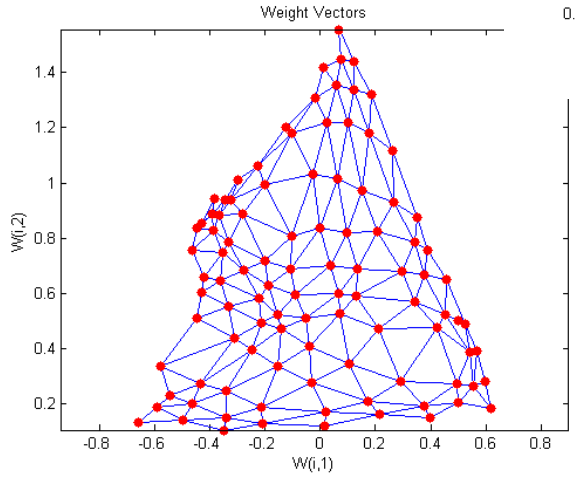

Weight Vectors
W(i,2)
W(i,1)


Weight Vectors
W(i,2)
W(i,1)


Weight Vectors
W(i,2)
W(i,1)

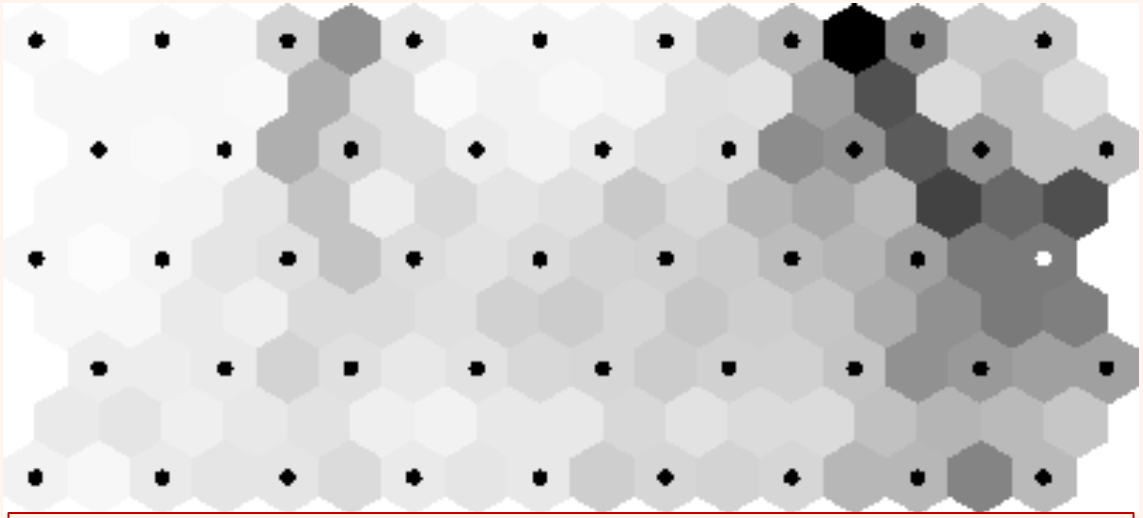

Weight Vectors
W(i,2)
W(i,1)


Weight Vectors
W(i,2)
W(i,1)


Weight Vectors
W(i,2)
W(i,1)

ویژگی‌های SOM

- پس از این آموزش به پایان برسد، وزن‌های به دست آمده به نوعی حاوی خواص آماری داده‌های ورودی خواهند بود.
- به نوعی «چگالی توزیع» را نشان می‌دهد.

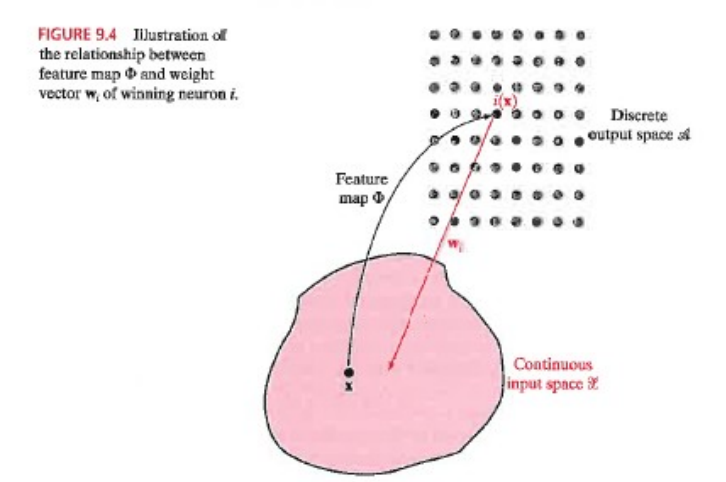


U-matrix representation of the Self-Organizing Map

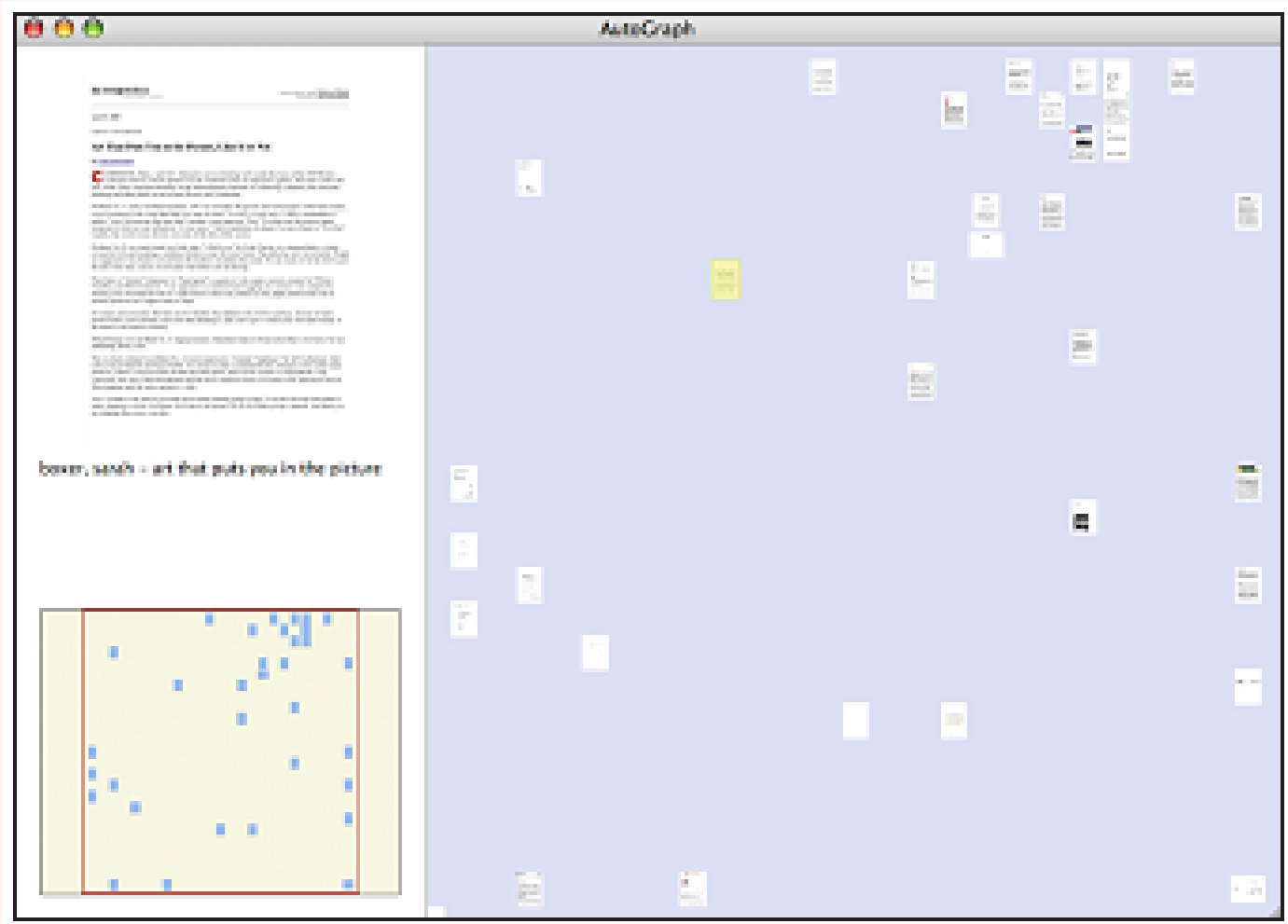
ویژگی‌های SOM (ادامه...)

- برای انتخاب «بردار خصیصه» نیز قابل استفاده خواهد بود.

$$\Phi : H \rightarrow A$$



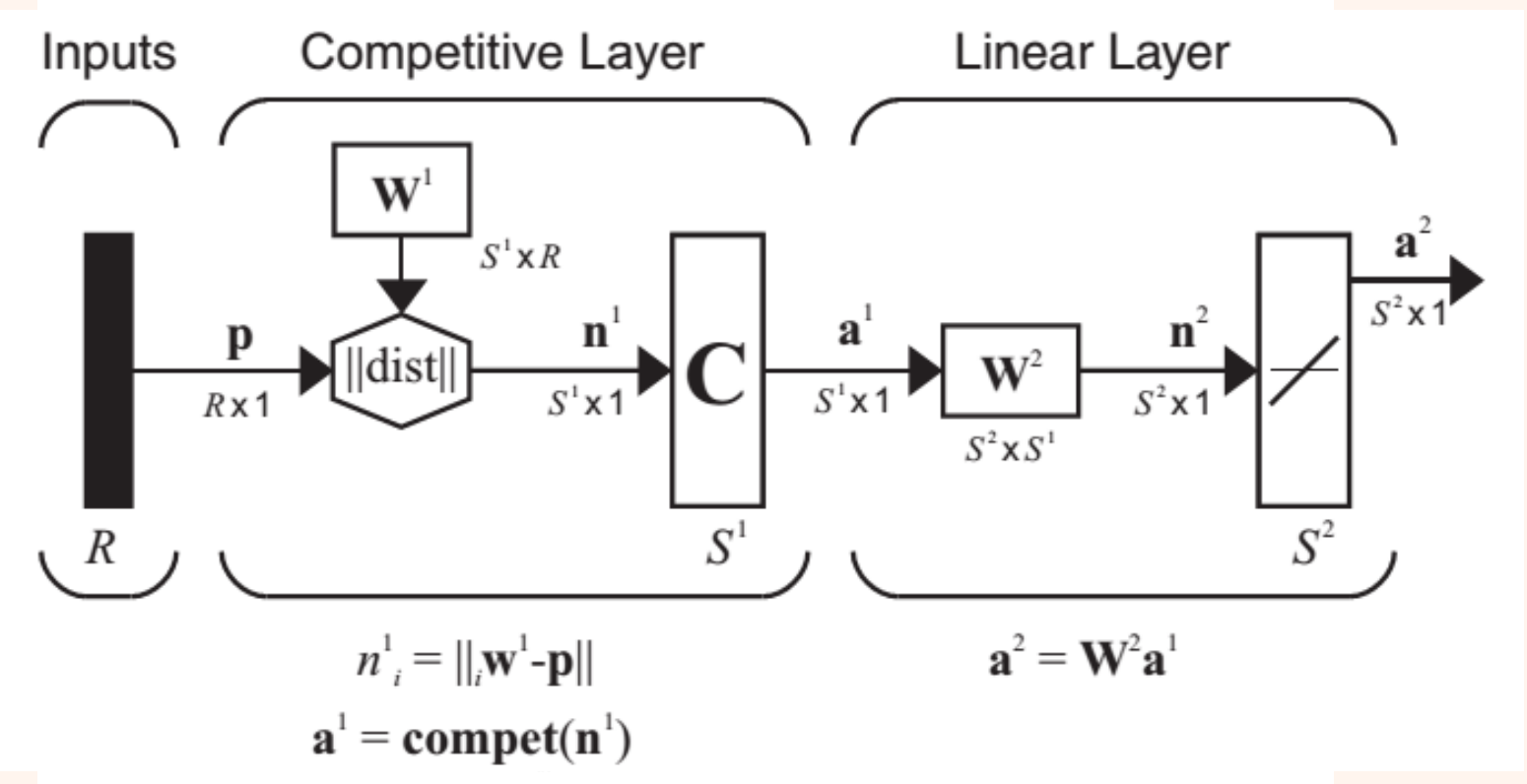
The Self-Organizing Desktop: An Unsupervised Content Classification System for Efficient Document Browsing



LVQ

Learning Vector Quantization

- این شبکه یک شبکه‌ی ترکیبی است که به هر دو شیوه‌ی supervised و unsupervised آموزش می‌بیند.



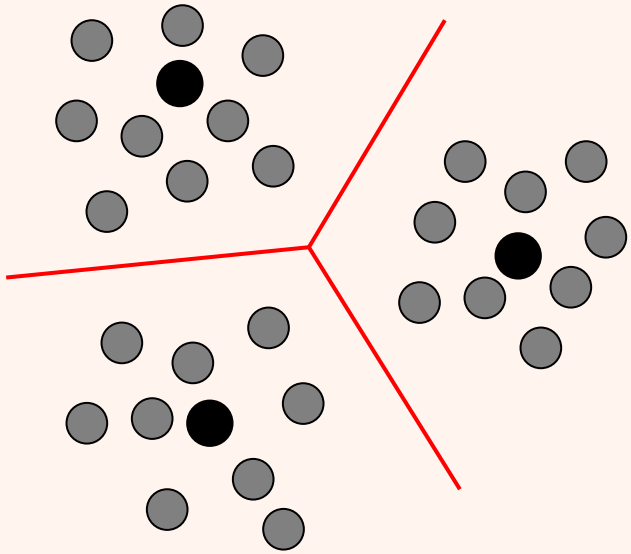
لایه‌ی رقابتی

- در نخستین لایه، هر کلاس به یک یا چند نورون واگذار می‌شود.
- تعداد نورون‌ها در این لایه باید حداقل به تعداد کلاس‌ها باشد.
- در لایه‌ی دوم، به ازای هر کلاس، یک نورون وجود دارد.
- تعداد نورون‌ها با تعداد کلاس‌ها برابرست.

$$n_i^1 = -\left\| {}_i\mathbf{w}^1 - \mathbf{p} \right\| \Rightarrow \mathbf{n}^1 = -\begin{bmatrix} \left\| {}_1\mathbf{w}^1 - \mathbf{p} \right\| \\ \left\| {}_2\mathbf{w}^1 - \mathbf{p} \right\| \\ \vdots \\ \left\| {}_{S^1}\mathbf{w}^1 - \mathbf{p} \right\| \end{bmatrix} \Rightarrow \mathbf{a}^1 = \mathbf{compet}(\mathbf{n}^1)$$

زیر کلاس‌ها

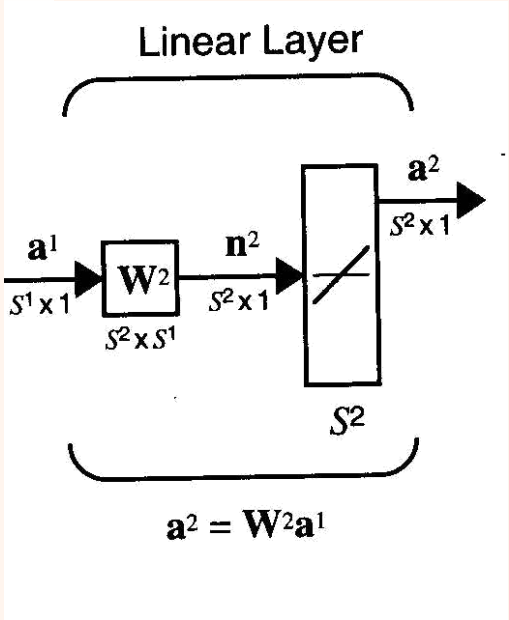
- بدین‌ترتیب در لایه‌ی اول هر کلاس به یک‌سری «زیرکلاس» تقسیم می‌شود.
- برنده‌ی لایه‌ی اول مشخص می‌کند که خروجی به کدام زیرکلاس تعلق دارد.



زیر کلاس‌ها (ادامه...)

- در لایه‌ی بعدی این که زیر کلاس‌ها برنده شده به کدام کلاس تعلق دارد، بررسی می‌شود.
- بدین ترتیب و برخلاف شبکه‌ی رقابتی در این حالت مرزهای کلاس‌ها پیچیده‌تر خواهند شد و قابلیت جداسازی کلاس‌های غیرمحدب را خواهد داشت.

لایه‌ی خطی



- سطرهای ماتریس بیانگر کلاس‌ها و ستون‌ها بیان‌گر زیرکلاس‌ها هستند.
- در هر ستون فقط یک عنصر یک خواهیم داشت.

$$w_{ki}^{2}=1 \Rightarrow$$ زیر کلاس i بخشی از کلاس k است.

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

Subclasses 1 & 3 belong to class 1
Subclasses 2 & 5 belong to class 2
Subclass 4 belongs to class 3

شیوه‌ی آموزش

- آموزش رقابتی با ناظر همراه می‌شود.
- وزن‌های $\mathbf{W}^2$ مشخص می‌شوند، معمولا به هر کلاس تعداد زیر کلاس یکسانی نسبت داده می‌شود.
- وزن‌های لایه‌ی اول به صورت زیر آموزش می‌بینند:
  - اگر نورون لایه‌ی اول به درستی برنده شده باشد:
    - به سمت بردار ورودی ($\mathbf{p}$) حرکت داده می‌شود.

$$ {}_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) + \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)\right) \quad \text{if} \quad a^2_{k^*} = t_{k^*} = 1 $$

  - وگرنه:
    - از بردار ورودی دور می‌شود.

$$ {}_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) - \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)\right) \quad \text{if} \quad a^2_{k^*} \neq t_{k^*} $$

شیوه‌ی آموزش (ادامه...)

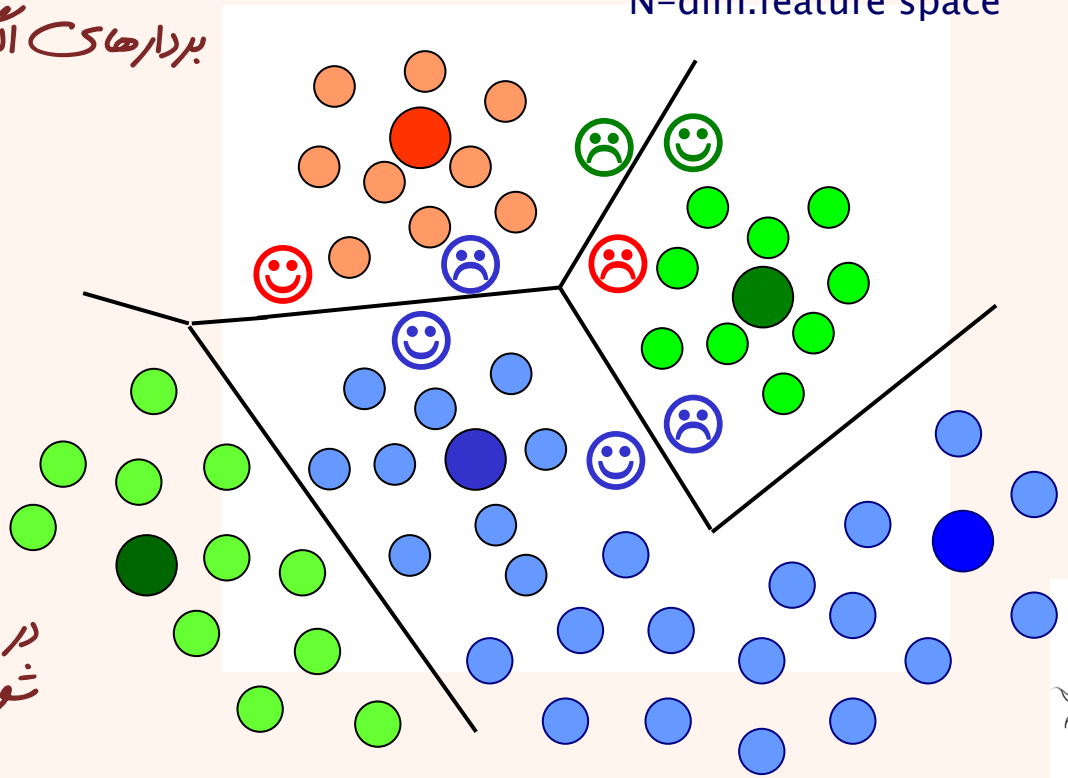
بردارهای الگو مقداردهی اولیه می‌شوند

یک ورودی اعمال می‌شود

نزدیک‌ترین بردار الگو انتخاب شود

در صورت درست بودن الگو به بردار ورودی نزدیک شود

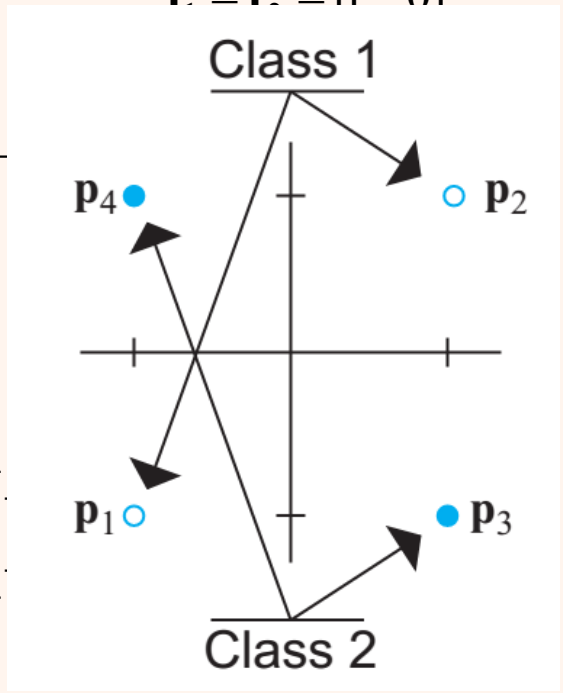
در صورتی که درست نیست از ورودی دور شود



$$w^*(t+1) = w^*(t) \pm \eta_w \left(\xi^{(t)} - w^*(t)\right)$$

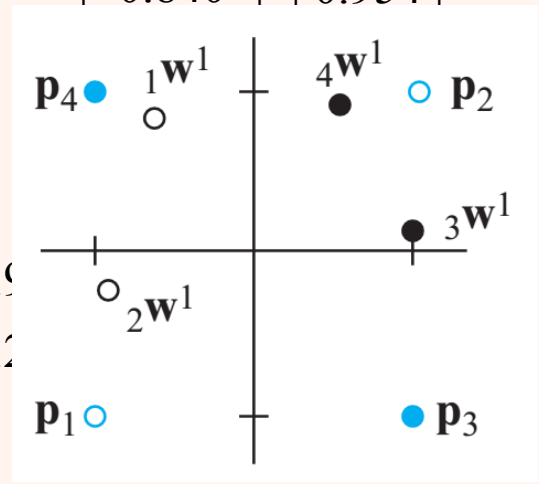
مثال

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



$$\mathbf{t}_3 = \mathbf{t}_4 = [0 \quad 1]^T$$

$$\begin{bmatrix} -0.543 \\ 0.840 \end{bmatrix} \quad \begin{bmatrix} 0.456 \\ 0.954 \end{bmatrix}$$



$$\begin{bmatrix} 0.997 \\ 0.094 \end{bmatrix}$$

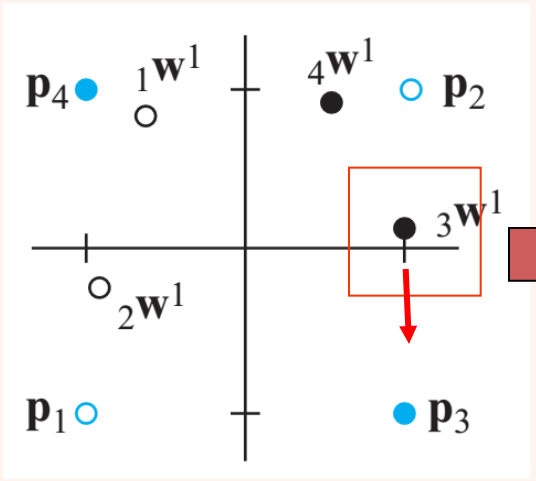
مثال-فرآیند آموزش

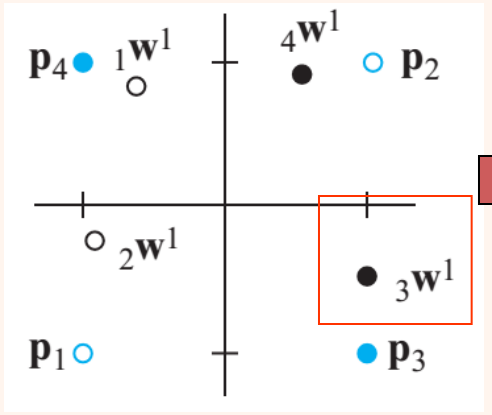
$$\mathbf{a}^1 = \mathbf{compet}(\mathbf{n}^1) = \mathbf{compet}\left(-\begin{bmatrix}\|_1\mathbf{w}^1 - \mathbf{p}\| \\ \|_2\mathbf{w}^1 - \mathbf{p}\| \\ \|_3\mathbf{w}^1 - \mathbf{p}\| \\ \|_4\mathbf{w}^1 - \mathbf{p}\|\end{bmatrix}\right) = \mathbf{compet}\left(\begin{bmatrix}-2.40 \\ -2.11 \\ -1.09 \\ -2.03\end{bmatrix}\right) = \begin{bmatrix}0 \\ 0 \\ 1 \\ 0\end{bmatrix}$$

$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 = \begin{bmatrix}1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1\end{bmatrix}\begin{bmatrix}0 \\ 0 \\ 1 \\ 0\end{bmatrix} = \begin{bmatrix}0 \\ 1\end{bmatrix} = \mathbf{t}_3$$

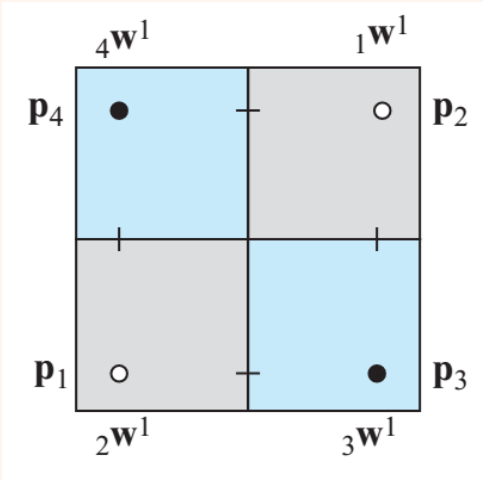
$${}_3\mathbf{w}^1(1) = {}_3\mathbf{w}^1(0) + 0.5\left(\mathbf{p}_3 - {}_3\mathbf{w}^1(0)\right) = \begin{bmatrix}0.998 \\ -0.453\end{bmatrix}$$

مثال-فرآیند آموزش





بعد از یک تکرار



بعد از تکرار زیاد

برخی مشکلات و موانع

- با این که LVQ در موارد بسیاری از عهده‌ی حل مسائل بر می‌آید، با محدودیت‌هایی مواجه خواهد بود:
  - ممکن است در این حالت نیز «نورون مرده» داشته باشیم، در این جا نیز می‌توان از «مکانیزم وجدان» استفاده نمود.
  - در برخی موارد لازم است یک نورون به ناحیه‌ای دیگر حرکت کند و در این بین ممکن است مجبور به گذر از ناحیه‌ای باشد که به کلاس او تعلق ندارد. با توجه به دفع توسط آن کلاس امکان عبور نخواهد یافت و در نتیجه به درستی آموزش نخواهد دید.

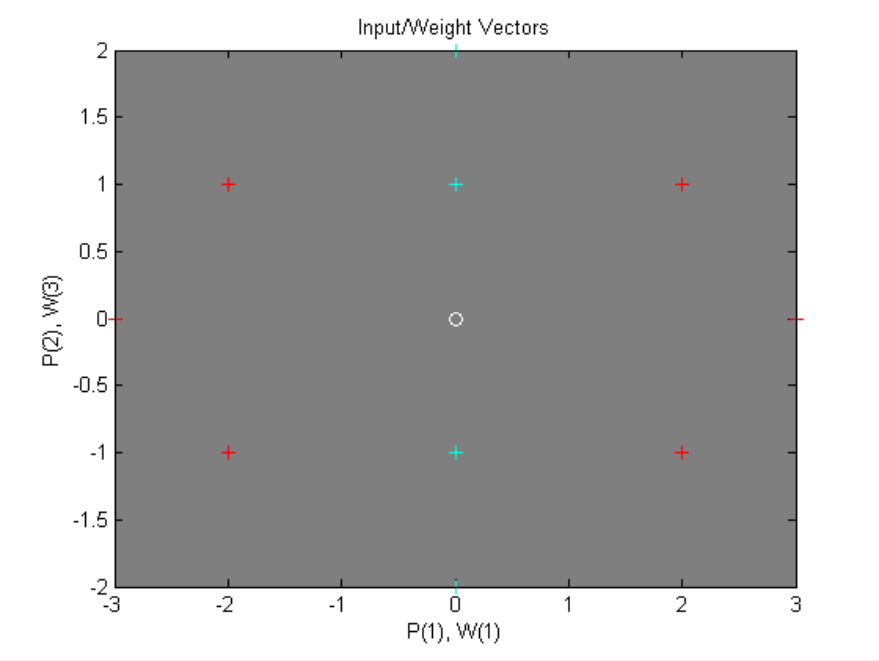
برخی مشکلات و موانع

- برای رفع این مشکل قانون آموزش kohonen به شیوه‌ی زیر بهبود می‌یابد:
- در صورتی که کلاس‌بندی به درستی انجام شده باشد، آموزش مانند قبل انجام می‌شود.
- و گرنه، زیرکلاس انتخاب شده از نمونه‌ی آموزشی دور شده و نزدیک‌ترین کلاس به نمونه‌ی آموزشی به آن نزدیک خواهد شد. الگوریتم حاصل LVQ2 نام دارد.

مثال

```
P = [-3 -2 -2  0  0  0  0 +2 +2 +3;
      0 +1 -1 +2 +1 -1 -2 +1 -1  0];
C = [1 1 1 2 2 2 2 1 1 1];
T = ind2vec(C);

colormap(hsv);
plotvec(P,C)
title('Input Vectors');
xlabel('P(1)');
ylabel('P(2)');

net = newlvq(minmax(P),4,[.6 .4],0.1);

hold on
W1 = net.IW{1};
plot(W1(1,1),W1(1,2),'ow')
title('Input/Weight Vectors')
xlabel('P(1), W(1)');
ylabel('P(2), W(3)');
```

lvqnet(hiddenSize,lvqLR,lvqLF)



ادامه‌ی

```
net.trainParam.epochs=150;
net=train(net,P,T);

cla;
plotvec(P,C);
hold on;
plotvec(net.IW{1}',vec2ind(net.LW{2}),'o');
W1 = net.IW{1};
hold on;
voronoi(W1(:,1),W1(:,2))


p = [0.2; 1];
a = vec2ind(sim(net,
```