به نام خداوند بخشنده مهربان

# مبانی هوش محاسباتی

## فصل سه – معرفی چند شبکه عصبی کاربردی

# فهرست مطالب

- تخصیص الگو؛ شبکه Hopfield
- آموزش غیرنظارتی و رقابت در شبکه عصبی؛ شبکه SOM
- دسته‌بندی با مفهوم توابع هسته؛ شبکه RBF
- مفاهیم جدید در شبکه‌های عصبی: عمیـق و پیچشـی CNN و DNN

مبانی هوش محاسباتی

فصل سه – شبکه‌های عصبی مصنوعی کاربردی

# کاربرد شبکه عصبی در تخصیص و ذخیره سازی الگو – شبکه هاپفیلد گسسته

# به‌یادسپاری الگوها

- یادگیری و به‌یادسپاری در ذهن انسان
  - نظریه آریستول (Aristotle)
    - ذهن انسان موضوعات و رویدادهایی که شبیه یا متضاد هم هستند یا با هم رخ‌داده‌اند را به یکدیگر مرتبط کرده و از این طریق، یادگیری و به‌یادسپاری حاصل می‌شود.
  - به‌یادسپاری یک الگو (به‌حافظه‌سپردن)
    - ارتباط یک الگو با خودش!

# شبکه‌های حافظه انجمنی (تخصیص‌دهنده الگو)

- شبکه‌های عصبی [حافظه] انجمنی (تخصیص‌دهنده) (Associative Memory Neural Net.)
  - معمولاً شبکه‌های تک‌لایه هستند.
  - حفظ و نگهداری الگوها به کمک تغییر و به‌روزآوری وزن‌ها.
    - به همین دلیل، بر خلاف حافظه‌های معمول، جای مشخصی را نمی‌توان برای نگهداری الگو مشخص نمود.
    - اثر حفظ یک الگو، در تمام شبکه پخش می‌گردد.
  - ویژگی مهم شبکه‌های عصبی حافظه انجمنی
    - بازیابی با ارایه الگویی شبیه الگوی آموزش داده‌شده نیز ممکن است.

# انواع شبکه‌های حافظه انجمنی

- الگوهای ورودی، در قالب زوج‌بردارهای آموزشی s:t بیان می‌شوند.
  - اگر s=t، شبکه را حافظه خودانجمنی (Autoassociative memory) و در غیراینصورت، حافظه دیگرانجمنی (Heteroassociative memory) گوییم.
- شبکه‌های حافظه انجمنی از لحاظ نوع ساختار
  - پیشخور (feedforward): انتقال اطلاعات و محاسبات همیشه از لایه ورودی به سمت لایه خروجی است.
  - بازگشتی (recurrent) یا تکراری (iterative): اتصال‌هایی بین نرون‌ها وجود دارد که باعث ایجاد حلقه بسته (فیدبک) در شبکه می‌شود.
- قوانین آموزش مورد استفاده در شبکه‌های حافظه انجمنی
  - قانون هب – قانون دلتا – قانون دلتای تعمیم یافته
  - دقیقاً همانند مطالب ذکر شده درباره شبکه‌های یک لایه

# قانون هب برای حافظه انجمنی

- با فرض اینکه وزنهای اولیه جملگی صفر باشند؛ قانون هب را می‌توان به کمک ضرب خارجی دو بردار ورودی و خروجی پیاده سازی کرد.
  - فرض کنیم الگوی s:t باید آموزش داده شود:

$$\mathbf{s} = (s_1, \ldots, s_i, \ldots, s_n)$$

$$\mathbf{t} = (t_1, \ldots, t_j, \ldots, t_m)$$

$$\mathbf{s}^T.\mathbf{t} = \begin{bmatrix} s_1 \\ \vdots \\ s_i \\ \vdots \\ s_n \end{bmatrix} . \begin{bmatrix} t_1 & \cdots & t_j & \cdots & t_m \end{bmatrix} = \begin{bmatrix} s_1t_1 & \cdots & s_1t_j & \cdots & s_1t_m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ s_it_1 & \cdots & s_it_j & \cdots & s_it_m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ s_nt_1 & \cdots & s_nt_j & \cdots & s_nt_m \end{bmatrix}$$

- این دقیقاً همان ماتریس وزنی است که بعد از آموزش این الگو توسط الگوریتم آموزش هب، به‌دست می‌آید!

*Step 0.* Initialize all weights ($i = 1, \ldots, n; j = 1, \ldots, m$):

$w_{ij} = 0.$

*Step 1.* For each input training–target output vector pair **s:t**, do Steps 2–4.

*Step 2.* Set activations for input units to current training input ($i = 1, \ldots, n$):

$x_i = s_i$

*Step 3.* Set activations for output units to current target output ($j = 1, \ldots, m$):

$y_j = t_j.$

*Step 4.* Adjust the weights ($i = 1, \ldots, n; j = 1, \ldots, m$):

$w_{ij}(\text{new}) = w_{ij}(\text{old}) + x_i y_j.$

# قانون هب برای حافظه انجمنی

– استفاده از ضرب خارجی برای آموزش بیش از یک الگو

- اگر P الگو به شکل زیر داشته باشیم:

$$\mathbf{s}_{(p)} = (s_{1(p)}, \ldots, s_{i(p)}, \ldots, s_{n(p)}) \quad , \quad \mathbf{t}_{(p)} = (t_{1(p)}, \ldots, t_{j(p)}, \ldots, t_{m(p)})$$

- از آنجا که وزنهای حاصل از آموزش هب به شکل زیر خواهند بود:

$$w_{ij} = \sum_{p=1}^{P} s_{i(p)} . t_{j(p)}$$

- می‌توان صورت ماتریسی را به کمک ضرب خـارجی بردارهـای ورودی و خروجی به صورت زیر نوشت:

$$\mathbf{W} = \sum_{p=1}^{P} \mathbf{s}_{(p)}^{T} . \mathbf{t}_{(p)}$$

# قانون دلتا برای حافظه انجمنی

- مزیت: مناسب برای الگوهایی که مستقل خطی هستند، اما ناهمبسته (عمود) نیستند.
- قانون دلتا (شکل اولیه):
  - تابع فعالیت خروجی؛ همانی:

$$y_j = \sum_{i=1}^{n} x_i w_{ij}$$

  - تغییر وزن قانون دلتا:

$$\Delta w_{ij} = \alpha(t_j - y_j)x_i$$

- در این صورت، مجذور خطای ورودی سلول (net input) و خروجی مطلوب، کمینه خواهد شد.

# قانون دلتا برای حافظه انجمنی

- قانون دلتای گسترش یافته ( extended or generalized Delta rule)
  - تابع فعالیت: تابع پیوسته مشتق‌پذیر.
  - کاهش خطای خروجی شبکه و خروجی مطلوب.
  - نحوه استخراج قانون دلتای گسترش یافته:

$$E = \sum_{j=1}^{m} (t_j - y_j)^2$$

$$\frac{\partial E}{\partial w_{IJ}} = \frac{\partial}{\partial w_{IJ}} \sum_{j=1}^{m} (t_j - y_j)^2 = \frac{\partial}{\partial w_{IJ}} (t_J - y_J)^2$$

- با توجه به روابط روبرو، داریم:

$$y_{in_J} = \sum_i x_i w_{iJ} \quad , \quad y_J = f(y_{in_J})$$

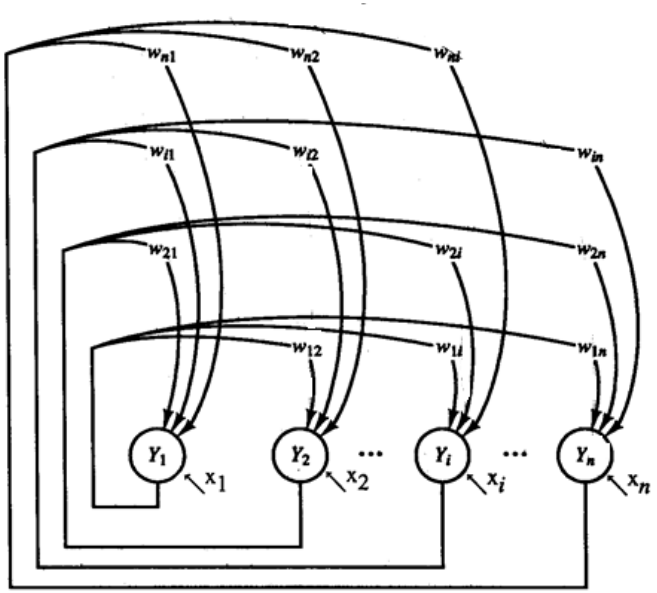
$$\frac{\partial E}{\partial w_{IJ}} = -2(t_J - y_J)\frac{\partial y_J}{\partial w_{IJ}} = -2(t_J - y_J) \cdot x_I \cdot f'(y_{in_J})$$

- و بنابراین، تغییر وزنها چنین خواهد بود:

$$\Delta w_{IJ} = \alpha \cdot (t_J - y_J) \cdot x_I \cdot f'(y_{in_J})$$

# Hopfield Net

- انواع شبکه هاپفیلد
  - گسسته
    - معمولاً باینری؛ کاربرد: حافظه با دسترسی محتوایی (Content Addressable Memory)
  - پیوسته
    - کاربرد در اختصاص الگو، بهینه سازی.




- هاپفیلد گسسته (Discrete Hopfield Net)
  - توپولوژی یک شبکه کاملاً متصل (fully interconnected)
    - ماتریس وزنها متقارن، قطر اصلی صفر.
  - دو نکته مهم در شیوه عملکرد هاپفیلد گسسته:
    - در هر لحظه، تنها یک سلول فعالیتش را تغییر می‌دهد (به روز می‌کند).
    - هر سلول، علاوه بر فعالیت سلول‌های دیگر، یک ورودی مستقل دارد. یعنی بردار ورودی همیشه در شبکه تاثیر دارد.

# آموزش و استفاده از هاپفیلد گسسته

- آموزش الگوها
  - الگوهای ورودی باینری یا بایپولار
  - آموزش هب، به صورت بایپولار (حتی برای داده‌های باینری).
  - وزن‌های قطر اصلی، صفر.
- الگوریتم استفاده

***Application Algorithm for the Discrete Hopfield Net***

*Step 0.* Initialize weights to store patterns.
(Use Hebb rule.)

While activations of the net are not converged, do Steps 1–7.

*Step 1.* For each input vector **x**, do Steps 2–6.

*Step 2.* Set initial activations of net equal to the external input vector **x**:

$$y_i = x_i, (i = 1, \ldots n)$$

*Step 3.* Do Steps 4–6 for each unit $Y_i$.
(Units should be updated in random order.)

*Step 4.* Compute net input:

$$y\_in_i = x_i + \sum_j y_j w_{ji}.$$

*Step 5.* Determine activation (output signal):

$$y_i = \begin{cases} 1 & \text{if } y\_in_i > \theta_i \\ y_i & \text{if } y\_in_i = \theta_i \\ 0 & \text{if } y\_in_i < \theta_i. \end{cases}$$

*Step 6.* Broadcast the value of $y_i$ to all other units.
(This updates the activation vector.)

*Step 7.* Test for convergence.

# یک مثال ساده از عملکرد هاپفیلد

- ذخیره سازی یک الگو؛ بازیابی الگوی تخریب شده
  - الگوی باینری: [۰ ۱ ۱ ۱] یا معادل بایپولار آن: [۱- ۱ ۱ ۱]
  - ماتریس وزن‌ها با قطر اصلی صفر:

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

  - انتخاب ترتیب تصادفی برای به‌روز رسانی فعالیت سلولها:

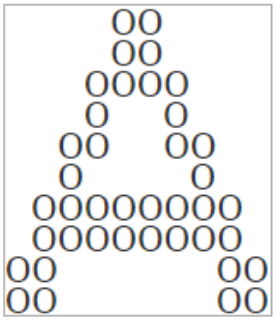
$$Y_1, Y_4, Y_3, Y_2$$

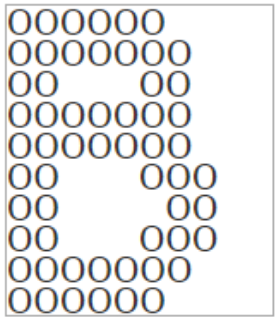
- یک دوره از الگوریتم استفاده هاپفیلد
  - توجه به شیوه تغییر فعالیت سلولها.
  - چون فعالیتها تغییر کرده، حداقل یک دوره دیگر ادامه دارد.

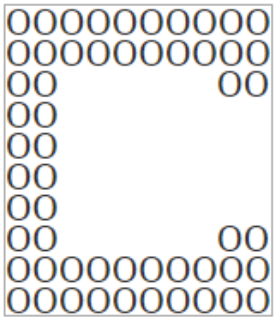
*Step 1.* The input vector is $\mathbf{x} = (0, 0, 1, 0)$. For this vector,
*Step 2.* $\mathbf{y} = (0, 0, 1, 0)$.
*Step 3.* Choose unit $Y_1$ to update its activation:
*Step 4.* $y\_in_1 = x_1 + \sum_j y_j w_{j1} = 0 + 1.$
*Step 5.* $y\_in_1 > 0 \rightarrow y_1 = 1.$
*Step 6.* $\mathbf{y} = (1, 0, 1, 0)$.
*Step 3.* Choose unit $Y_4$ to update its activation:
*Step 4.* $y\_in_4 = x_4 + \sum_j y_j w_{j4} = 0 + (-2).$
*Step 5.* $y\_in_4 < 0 \rightarrow y_4 = 0.$
*Step 6.* $\mathbf{y} = (1, 0, 1, 0)$.
*Step 3.* Choose unit $Y_3$ to update its activation:
*Step 4.* $y\_in_3 = x_3 + \sum_j y_j w_{j3} = 1 + 1.$
*Step 5.* $y\_in_3 > 0 \rightarrow y_3 = 1.$
*Step 6.* $\mathbf{y} = (1, 0, 1, 0)$.
*Step 3.* Choose unit $Y_2$ to update its activation:
*Step 4.* $y\_in_2 = x_2 + \sum_j y_j w_{j2} = 0 + 2.$
*Step 5.* $y\_in_2 > 0 = y_2 = 1.$
*Step 6.* $\mathbf{y} = (1, 1, 1, 0)$.
*Step 7.* Test for convergence.

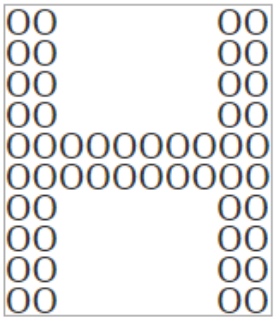
# مثال هاپفیلد

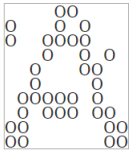
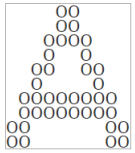
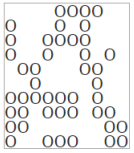
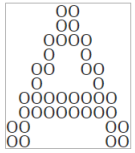
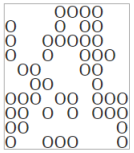
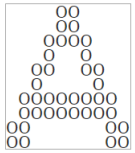
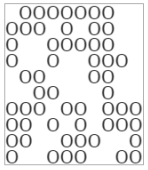
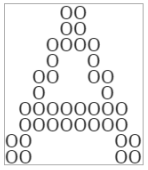
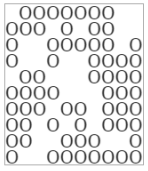
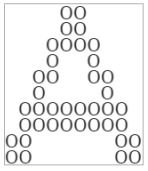
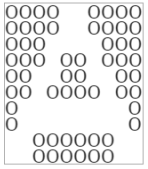
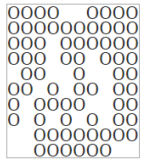
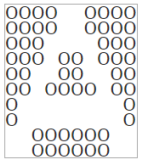
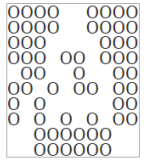
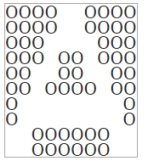
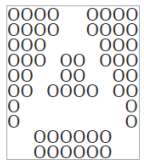
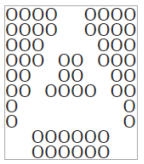
## Initial patterns





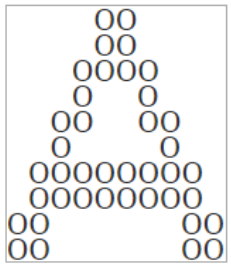


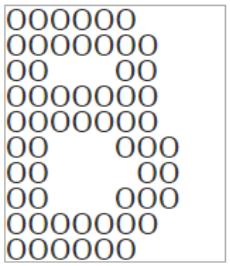


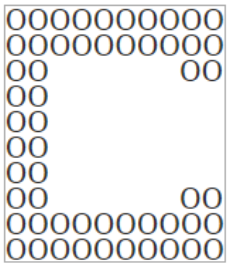
| Noise percternentage | Type of a distorted pattern | The result of recognition |
|---|---|---|
| 10% |  |  |
| 20% |  |  |
| 30% |  |  |
| 40% |  |  |
| 50% |  |  |
| 60% |  |  |
| 70% |  |  |
| 80% |  |  |
| 90% |  |  |
| 100% |  |  |

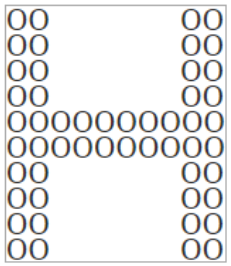
# مثال هاپفیلد

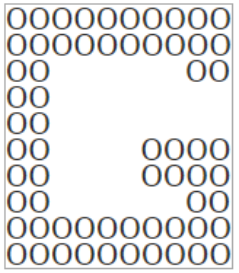
**Initial patterns**



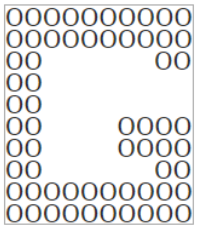


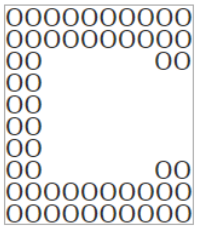




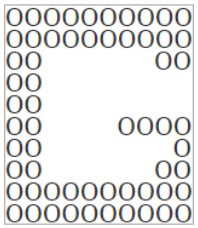


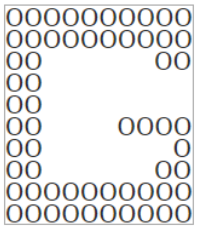
**The test pattern**





**The result of recognition**



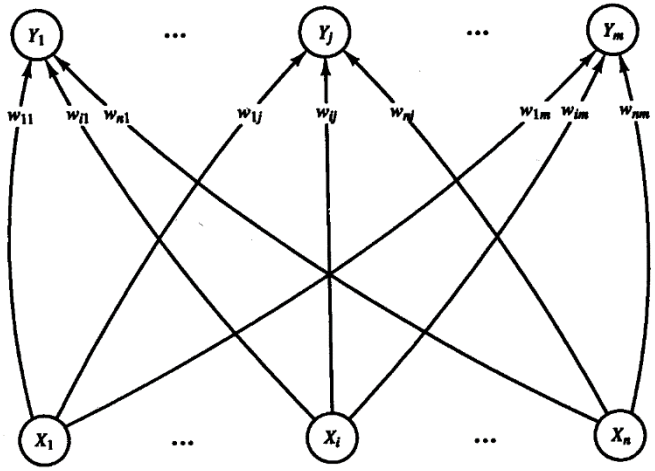


# کاربرد شبکه عصبی در خوشه‌بندی الگو – شبکه «نقشه خودسازمانده» (آموزش غیر نظارتی و رقابت در شبکه عصبی)

# Kohonen Self-Organizing Maps

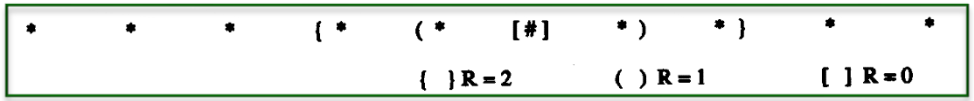
- نکات شاخص و متفاوت در شبکه Kohonen SOM
  - آموزش بدون ناظر (Unsupervised Learning)
    - برچسب دسته مشخص نیست، وظیفه شبکه یافتن مشابهت در الگوهای ورودی و دسته بندی الگوها بر اساس آن است.
    - یافتن مرکز تجمع داده‌های ورودی، به کمک شباهت بین آنها.
  - شبکه رقابتی
    - سلولها در یک ساختار یک یا دو بعدی، قرار گرفته اند.
    - هر سلول، با توجه به تعریف، تعدادی همسایه دارد.
    - نرون‌ها با همسایگان در حال رقابت هستند.
    - در هر دور آموزش، یک سلول (با توجه به میزان مشابهت با الگوی ورودی) برنده می‌شود.
    - فقط سلول و همسایه‌های آن، حق آموزش (تغییر اوزان) را دارند.
  - معمولاً به جای ضرب وزن‌ها در الگوی ورودی، فاصله آنها سنجیده می‌شود.
    - در الگوها با اندازه یک، از نظر ریاضی هیچ تفاوتی ندارد.

# Kohonen SOM

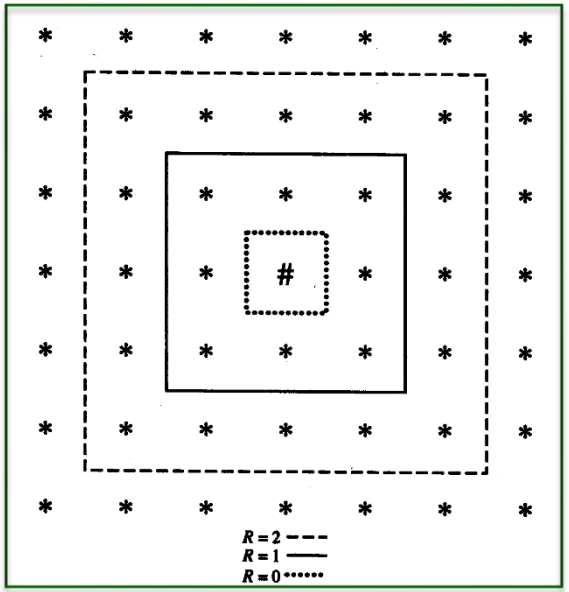



- شبکه تک لایه:
- مفهوم همسایگی و شعاع همسایگی

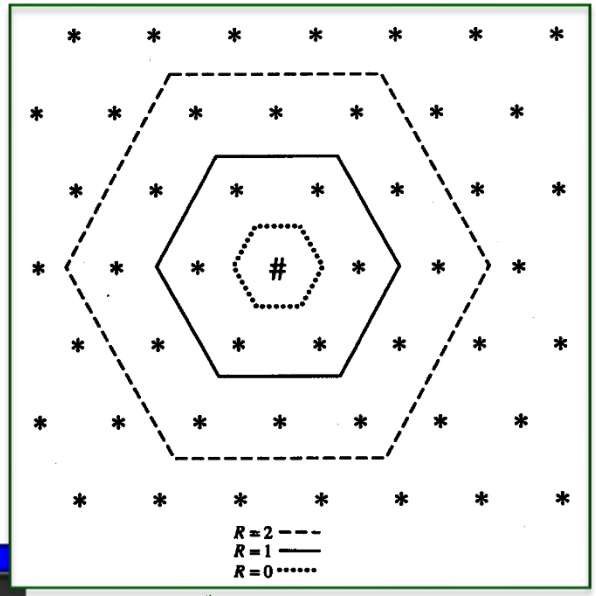
یک بُعدی




دو بُعدی
مربعی




دو بُعدی
شش ضلعی



# SOM

- الگوریتم آموزش و استفاده:

*Step 0.* Initialize weights $w_{ij}$. (Possible choices are discussed below.)
Set topological neighborhood parameters.
Set learning rate parameters.

*Step 1.* While stopping condition is false, do Steps 2–8.

*Step 2.* For each input vector x, do Steps 3–5.

*Step 3.* For each $j$, compute:

$$D(j) = \sum_i (w_{ij} - x_i)^2.$$

محاسبه فاصله بردار ورودی تا بردارهای نمونه

*Step 4.* Find index $J$ such that $D(J)$ is a minimum.

*Step 5.* For all units $j$ within a specified neighborhood of $J$, and for all $i$:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})].$$

یافتن سلول برنده؛ تغییر وزن برنده و همسایگان

*Step 6.* Update learning rate.

کاهش نرخ آموزش به آهستگی، با تصاعد هندسی یا حسابی؛

*Step 7.* Reduce radius of topological neighborhood at specified times.

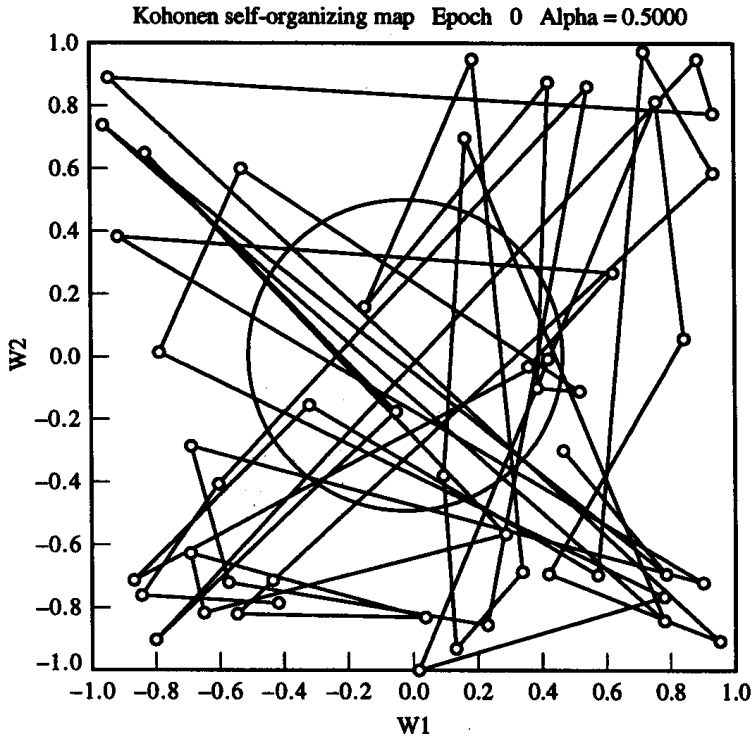
کاهش شعاع همسایگی

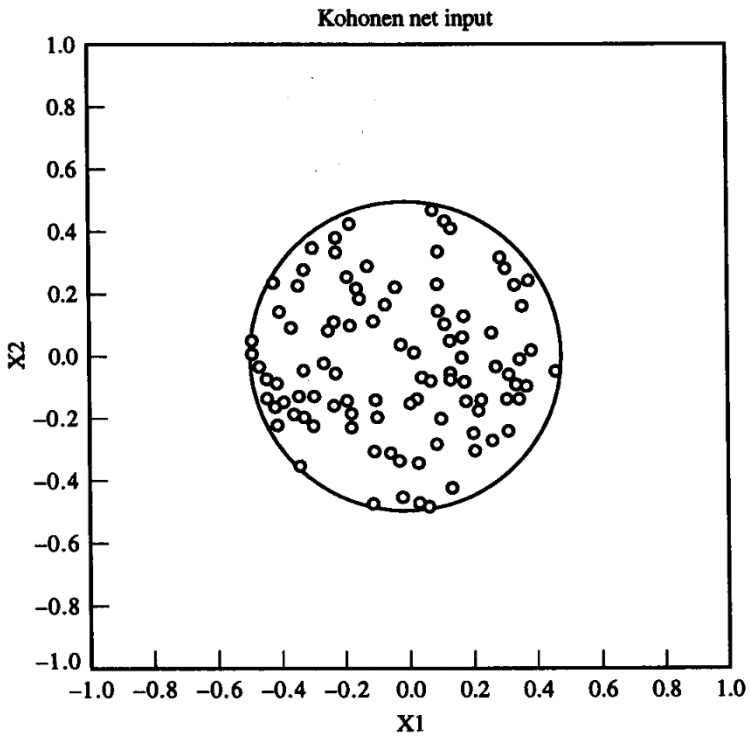
*Step 8.* Test stopping condition.

شرط خاتمه: تغییرات وزن از حد خاصی فراتر نرود. یا صفر شدن شعاع همسایگی و نرخ آموزش.

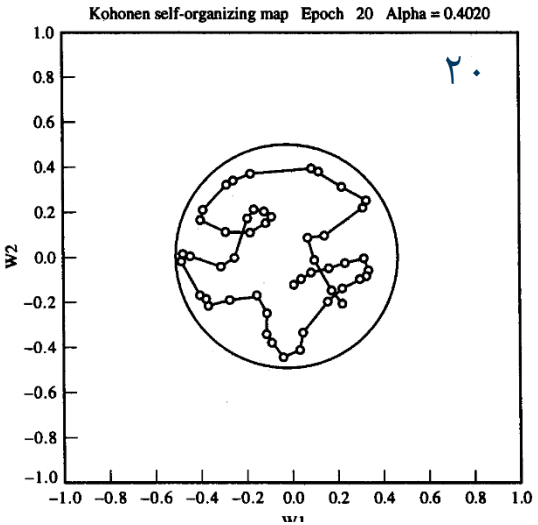
# مثال هندسی دوبعدی SOM (۱)

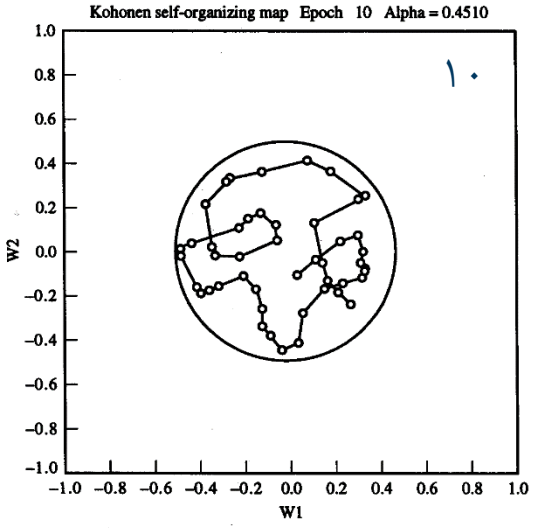
- داده‌های ورودی:
  - ۱۰۰داده تصادفی در دایره.
- وزن‌های اولیه:
  - ۵۰ سلول با همسایگی خطی ۱.

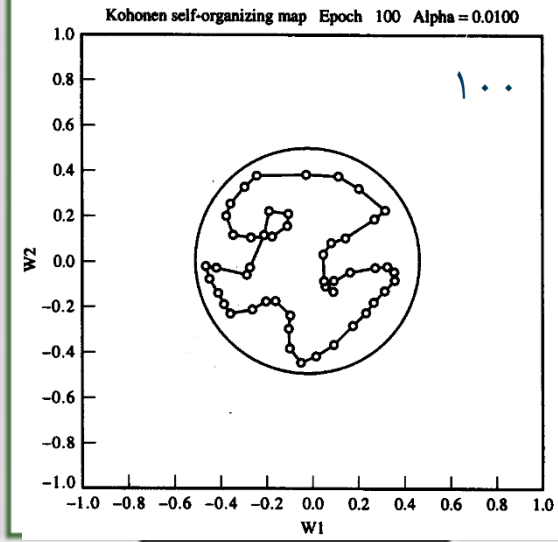


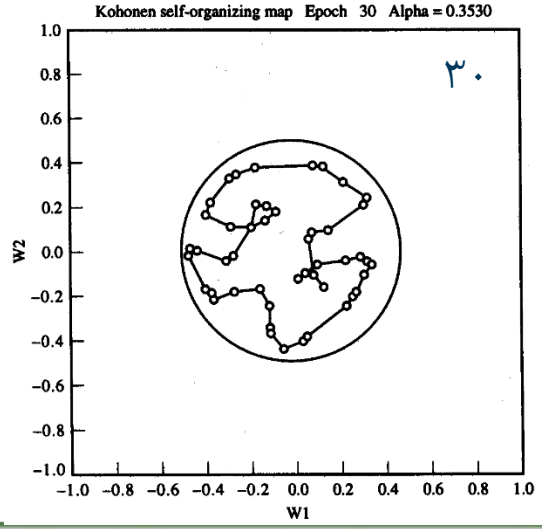



# مثال هندسی دوبعدی SOM (۲)



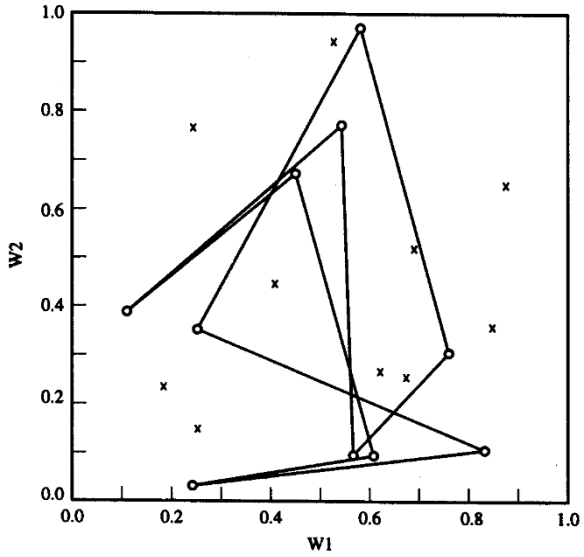



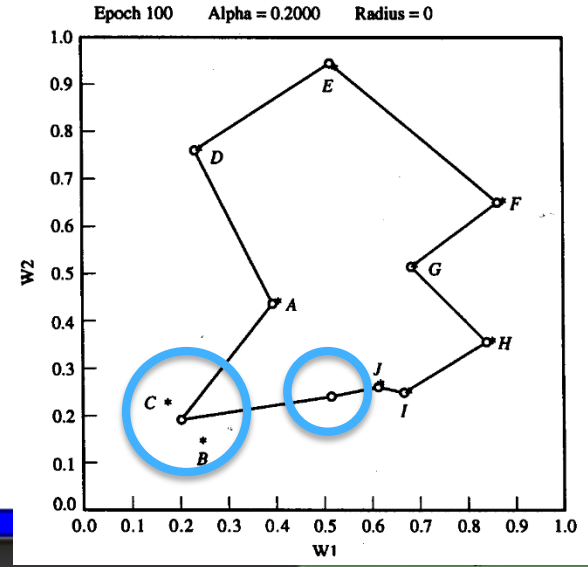




- نتایج بعد از:
  - ۱۰ تکرار
  - ۲۰ تکرار
  - ۳۰ تکرار
  - ۱۰۰ تکرار
- با افزایش تکرارها، علاوه بر تجمع وزنها در دایره مرکزی، سلولهای همسایه چینش نرم‌تری پیدا می‌کنند.
- با انتخاب شعاع همسایگی بیشتر در ابتدا، ترتیب از این هم صاف‌تر می‌شود.

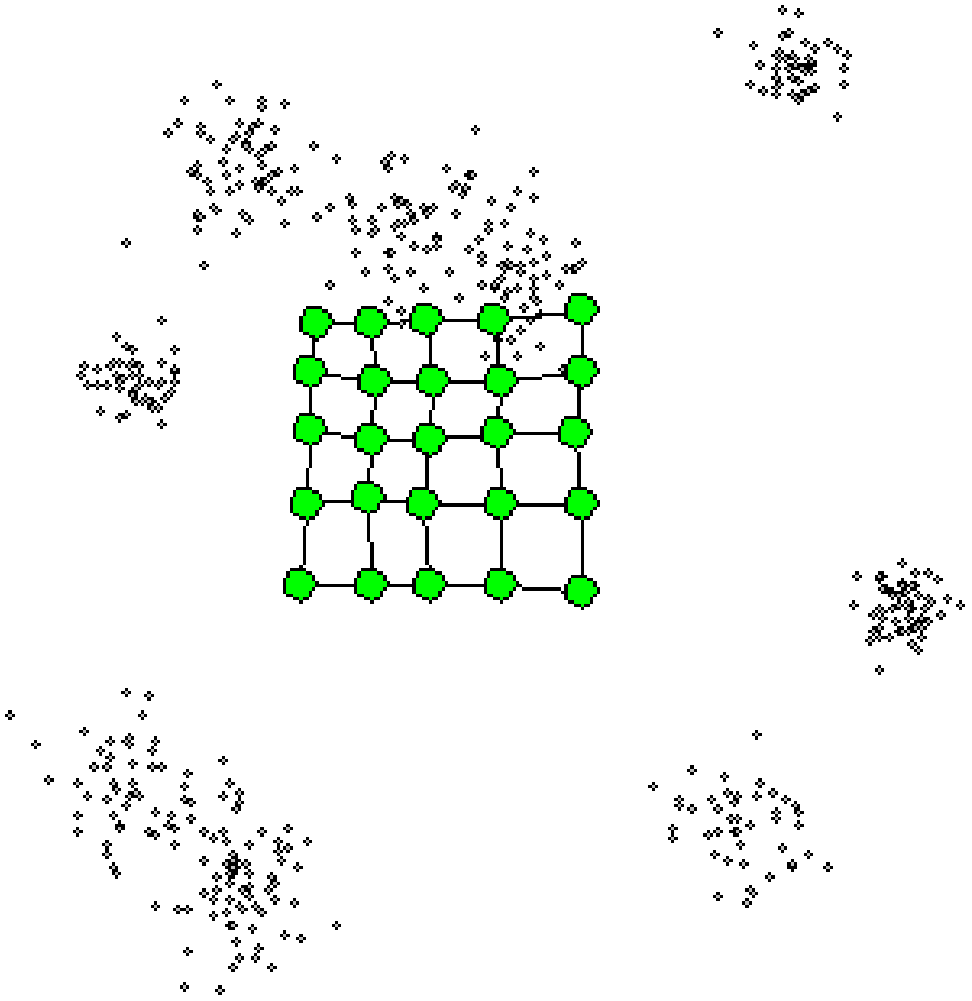
# مثال: حل TSP توسط SOM



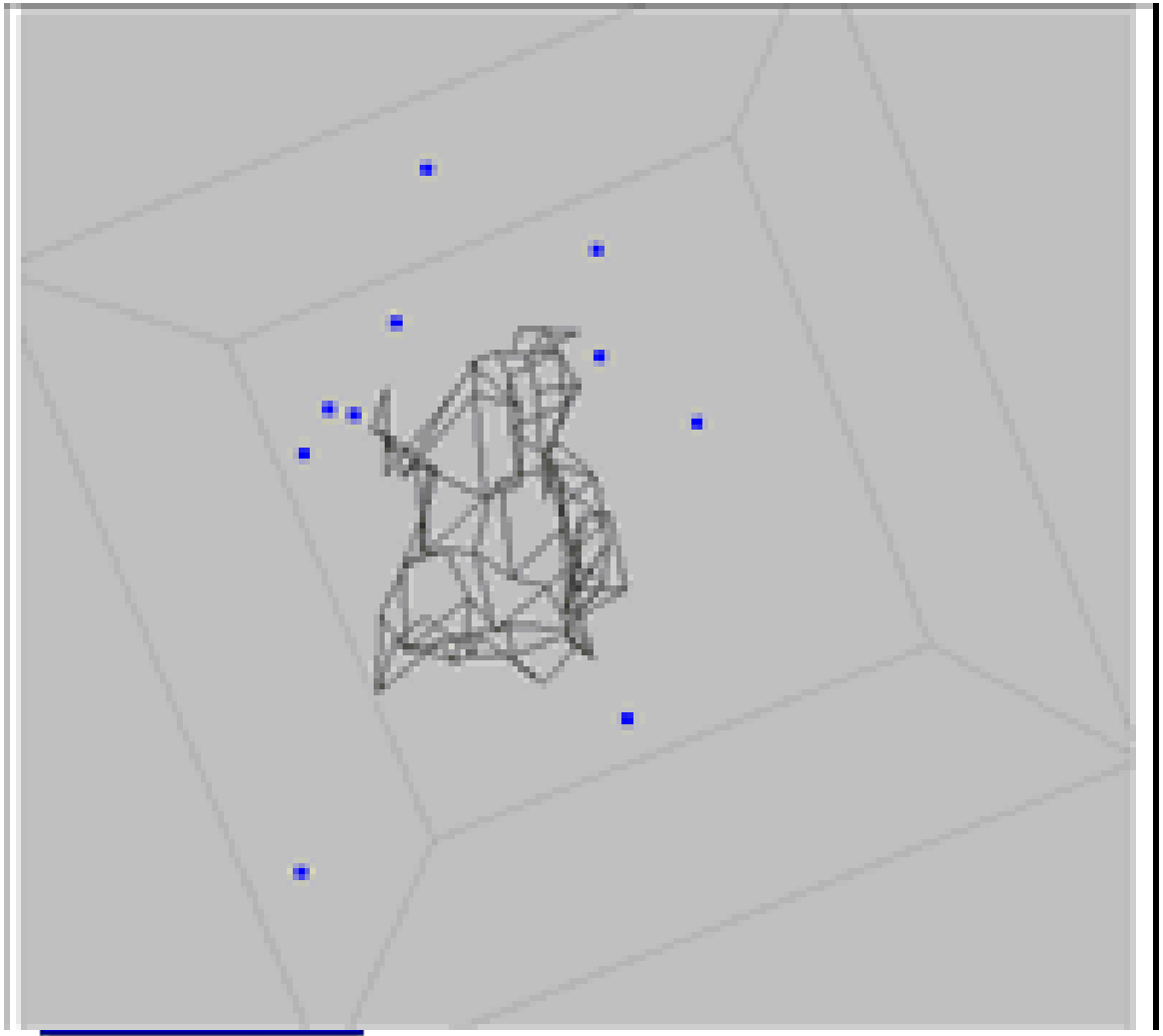

- داده‌های ورودی: مختصات شهرها
- ۱۰ سلول در لایه خروجی.
  - همسایگی: نشان‌دهنده شهرهای مجاور در تور انتخاب شده.
  - حالت مطلوب: هر شهر، یک سلول را به عنوان نماینده انتخاب می‌کند.
  - در عمل، ممکن است بعضی سلولها بین دو یا چند شهر، (اصطلاحاً) گیر بیافتند.

# مثال SOM



- کاربرد در:
  - مدل‌سازی.
  - تقریب توابع.
  - بهینه سازی.
  - خوشه بندی.
  - چندی سازی.
  - کـاهش ابعـاد داده‌ها.

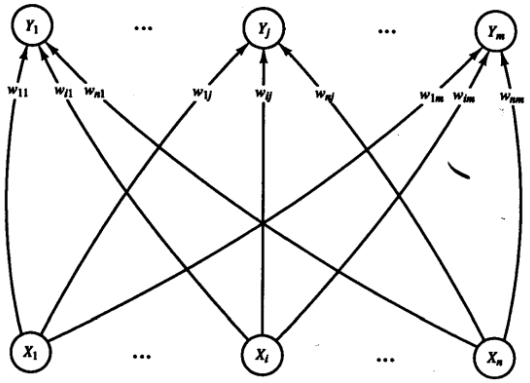
# مثال SOM



- کاربرد در:
  - مدل‌سازی.
  - تقریب توابع.
  - بهینه سازی.
  - خوشه بندی.
  - چندی سازی.
  - کـاهش ابعـاد داده‌ها.

# مدرج‌سازی برداری با یادگیری (LVQ)

- Learning Vector Quantization - LVQ
  - ترجمه Quantization: مدرج‌سازی - کمّی‌سازی - چندی‌سازی
    - معنا: گسسته‌سازی مقادیر پیوسته.
- LVQ شبکه با نظارت (Supervised) است.
  - بر خلاف SOM که بدون نظارت است.
  - بردارهای مرجع، به عنوان نماینده کلاس، در الگوریتم آموزش تنظیم می‌شوند.
  - پس از آموزش، LVQ بردارهای ورودی را به نزدیک‌ترین بردار مرجع، نسبت می‌دهد.
- توپولوژی شبکه
  - همانند SOM؛ با این تفاوت که همسایگی وجود ندارد.



# آموزش LVQ

*Step 0.* Initialize reference vectors (several strategies are discussed shortly); initialize learning rate, $\alpha(0)$.

*Step 1.* While stopping condition is false, do Steps 2–6.

*Step 2.* For each training input vector $\mathbf{x}$, do Steps 3–4.

*Step 3.* Find $J$ so that $\|\mathbf{x} - \mathbf{w}_J\|$ is a minimum.

*Step 4.* Update $\mathbf{w}_J$ as follows:

if $T = C_J$, then

$$\mathbf{w}_J(\text{new}) = \mathbf{w}_J(\text{old}) + \alpha[\mathbf{x} - \mathbf{w}_J(\text{old})];$$

if $T \neq C_J$, then

$$\mathbf{w}_J(\text{new}) = \mathbf{w}_J(\text{old}) - \alpha[\mathbf{x} - \mathbf{w}_J(\text{old})].$$

*Step 5.* Reduce learning rate.

*Step 6.* Test stopping condition:
The condition may specify a fixed number of iterations (i.e., executions of Step 1) or the learning rate reaching a sufficiently small value.

- نکات آموزش
  - نزدیک‌ترین بردار مرجع به بردار ورودی (سلول با شبیه‌ترین بردار وزن) برنده می‌شود.
  - اگر سلول برنده با بردار ورودی هم‌کلاس بود، بردار مرجع به بردار ورودی نزدیک می‌شود.
  - اگر نبود، دور می‌شود.

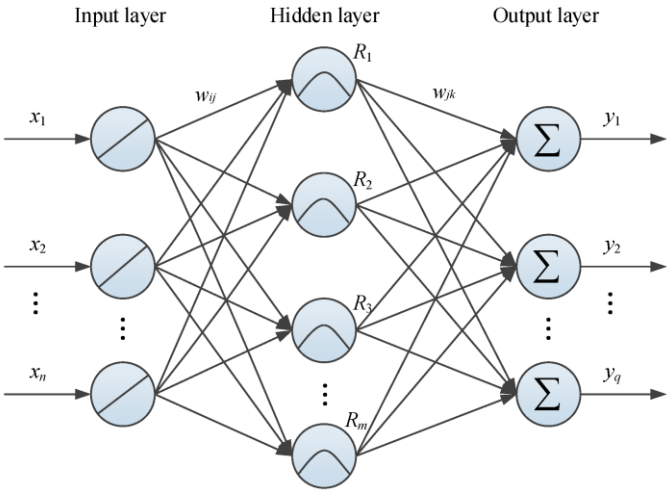
- نکات آموزش
  - کاهش نرخ آموزش، باعث ایجاد همگرایی است.
  - شرط توقف، کم‌شدن بیش از حد نرخ آموزش یا رسیدن به تعداد تکرار مشخص است.

# نکاتی درباره LVQ

- شیوه انتخاب وزن‌ها (بردارهای مرجع) اولیه
  - انتخاب تصادفی
  - انتخاب k (تعداد کلاس) بردار اولیه آموزشی به عنوان k بردار مرجع.
  - استفاده از الگوریتم‌های خوشه‌بندی برای ایجاد یک تقریب اولیه از بردارهای مرجع.
    - مثال؛ استفاده از k-means یا SOM!!

# دسته‌بندی با مفهوم توابع هسته؛ شبکه RBF

# ساختار شبکه RBFNN



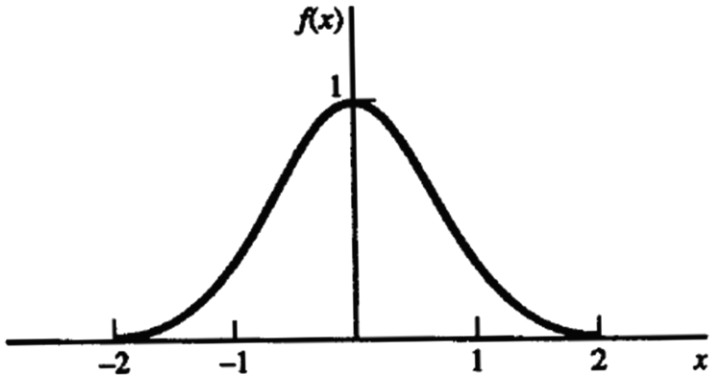

- شبکه عصبی با توابع با سطح مقطع حلقوی( Radial-Basis Function Neural Network - RBFNN)
  - ساختار شبکه: دولایه، با یک لایه مخفی
  - تفاوت با MLP در توابع فعالیت نرونهاست
    - لایه میانی: توابع حلقوی [پایه شعاعی]
    - لایه خروجی: تابع همانی

# تابع فعالیت لایه میانی

- توابع حلقوی (radial-basis)
  - توابعی که در کل دامنه، مقدار غیرمنفی دارند و فعالیت آنها، حول یک نقطه به اوج خود می‌رسد. هرچه نقطه ورودی از مرکز این توابع دور باشد، فعالیت تابع ضعیف‌تر می‌شود.
    - « مثال: تابع گوسی (Gaussian)

$$f(x) = e^{-x^2}$$

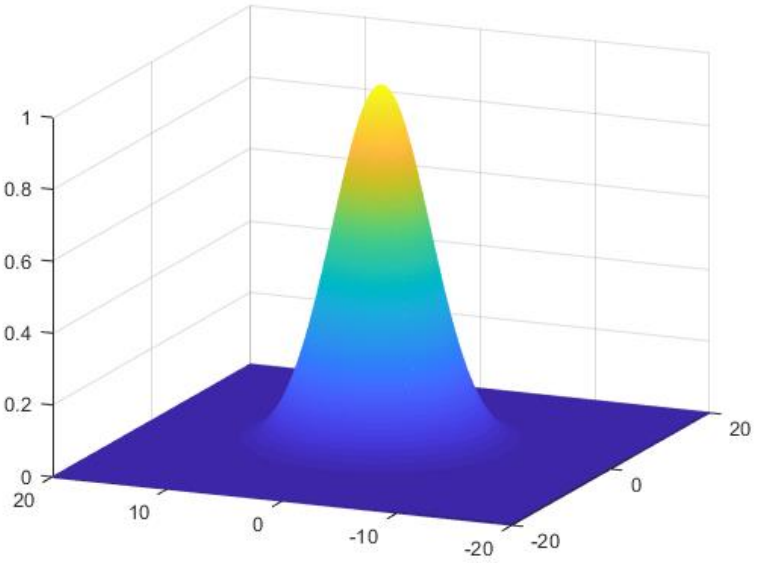
$$f'(x) = -2x \cdot f(x)$$



# توابع حلقوی

$$p(\boldsymbol{x}) = \frac{1}{\left|\sqrt{2\pi\boldsymbol{\Sigma}}\right|} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

- Mean: $\boldsymbol{\mu}$ (vector 2x1)
- Covariance: $\boldsymbol{\Sigma}$ (matrix 2x2)

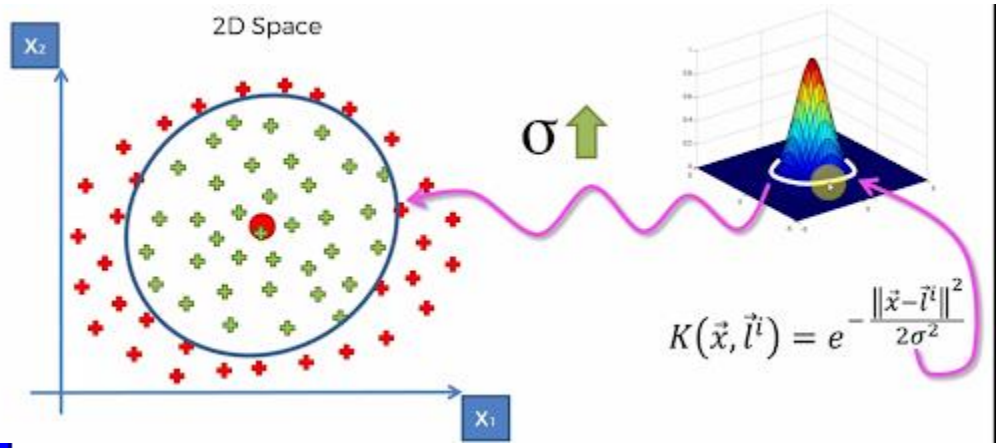



$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}}$$

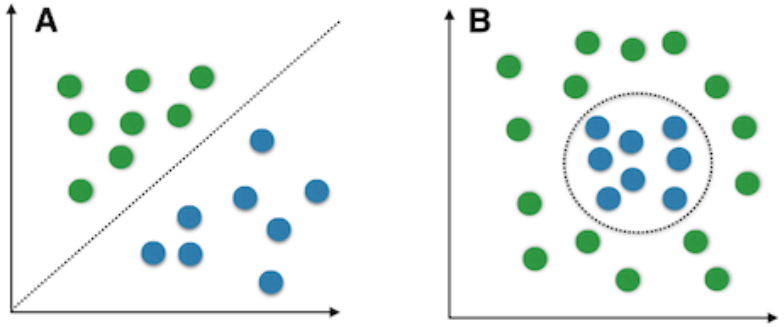
- توابع گاوسی – توزیع نرمال
  - شکل عمومی تابع توزیع نرمال چندمتغیره
  - تجسم تصویری تابع دو متغیره
- دلیل نام‌گذاری
  - سطح مقطع (تقاطع با صفحه عمود بر محور Z)، به شکل دایره است.

# دسته بندی توسط RBFNN

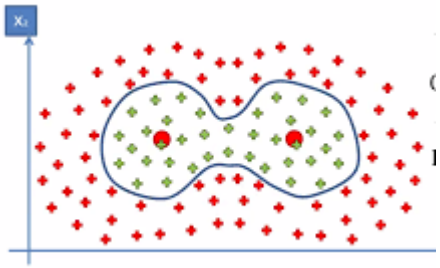
- RBF چگونه عمل می‌کند؟
  - مرکز تابع حلقوی توسط وزن‌های آن تعیین می‌شود. واریانس تابع حلقوی پارامتر سلول است و وزنی معادل آن تعیین نشده است!
  - با توجه به فاصله نقاط داده تا مرکز تـابع و واریـانس آن، داده در داخل یا خارج دسته قرار می‌گیرد.

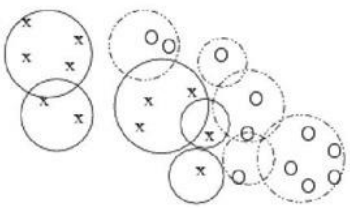


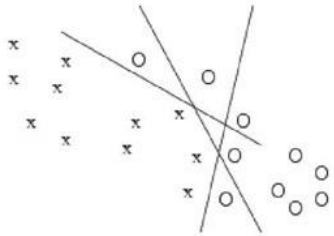
# اثر دسته بندی توابع RBF




- گاهی اوقات، دسته‌بندی توسط RBFNN سریعتر است یا به ساختار ساده تری احتیاج دارد، به خاطر شکل مساله.

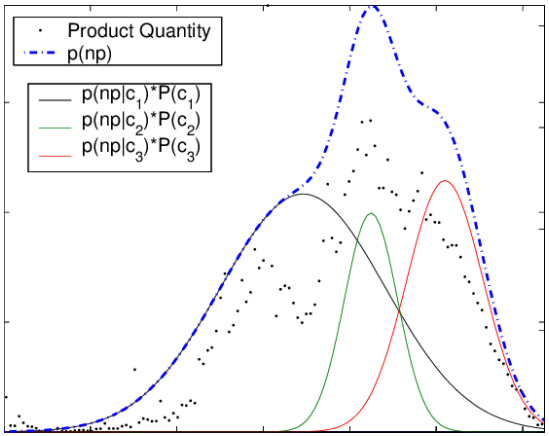


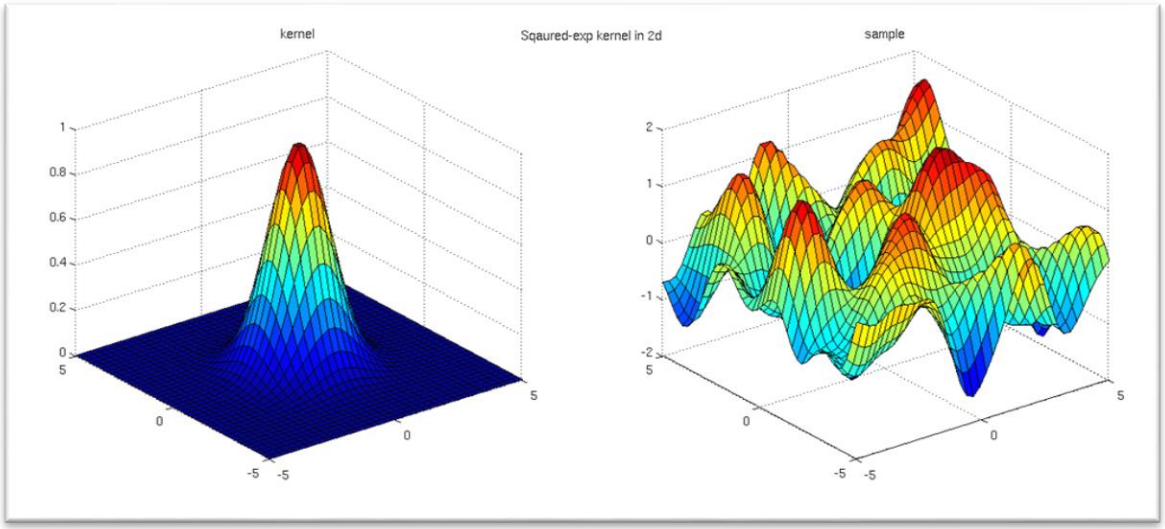




# شبکه RBF و مدلسازی

- استفاده از ترکیب توابع گوسی ( mixture of gaussians - MOG) در مدلسازی توابع پیچیده بسیار متداول است.
  - هر تابع پیوسته‌ای را می‌توان به کمک تعدادی تابع گوسی، با ضرایب، مراکز و واریانس‌های متفاوت، با دقت خوبی تقریب زد.



مبانی هوش محاسباتی

فصل سه – شبکه‌های عصبی مصنوعی پرکاربرد

# مفاهیم جدید در شبکه‌های عصبی: عمیق و پیچشی CNN و DNN