شبکه‌های عصبی مصنوعی

۴۳-۱۱-۷۱۳-۰۱

بخش سوم

RBF

دانشگاه شهید بهشتی

دانشکده‌ی مهندسی و علوم کامپیوتر

زمستان ۱۳۹۴

احمد محمودی ازناوه

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

فهرست مطالب

پیش‌گفتار

- در اینجا به طراحی شبکه‌ی عصبی به صورت یک مسأله‌ی «برازش منحنی» (curve fitting) و «درون‌یابی» نیز نگریسته می‌شود.
  - با این نگاه مسأله معادل یافتن یک رویه در فضایی چند بعدی است که به بهترین نحو با داده‌های آموزشی تطابق داشته باشد.
- همچنین برای دسته‌بندی از شبکه‌ی RBF استفاده می‌شود.
- قضیه‌ی cover:
  - در یک مسأله‌ی پیچیده‌ی بازشناخت الگو، با نگاشت به فضای high dimensional با احتمال بیشتری جدایی‌پذیر خطی خواهد

Cover, T. M. (1965). "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition." Electronic Computers, IEEE Transactions on EC-14(3): 326-334.

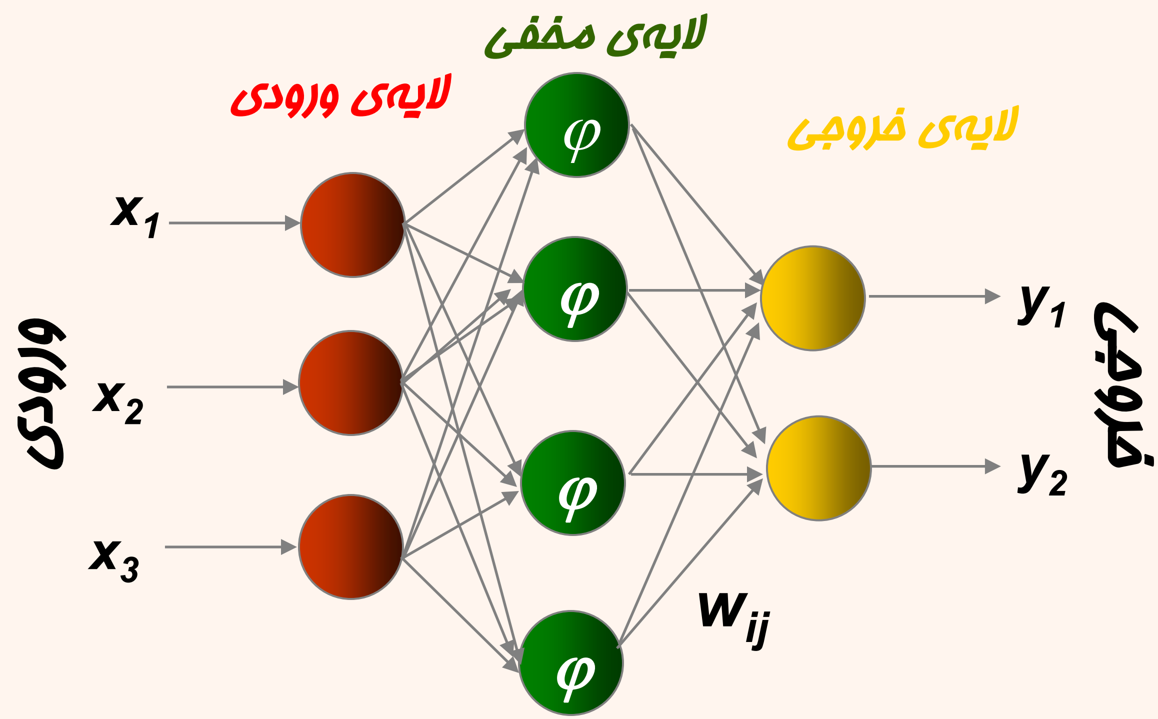
Radial Basis Function

کلاس‌های جدایی‌ناپذیر خطی → Transform to "higher"-dimensional vector space → کلاس‌های جدایی‌پذیر خطی

- در شبکه‌های RBF سه نوع لایه وجود دارد:
- لایه ورودی(sensory unit)
  - گره‌های منبع شبکه و رابط شبکه با دنیای بیرون
- لایه مخفی(hidden layer)
  - هر گره، تابعی شعاعی که مرکز و شعاعی مختص به خود را داراست، به ورودی‌ها اعمال می‌کند.
- لایه خروجی(output layer)
  - ترکیبی خطی از توابع لایه‌های مخفی

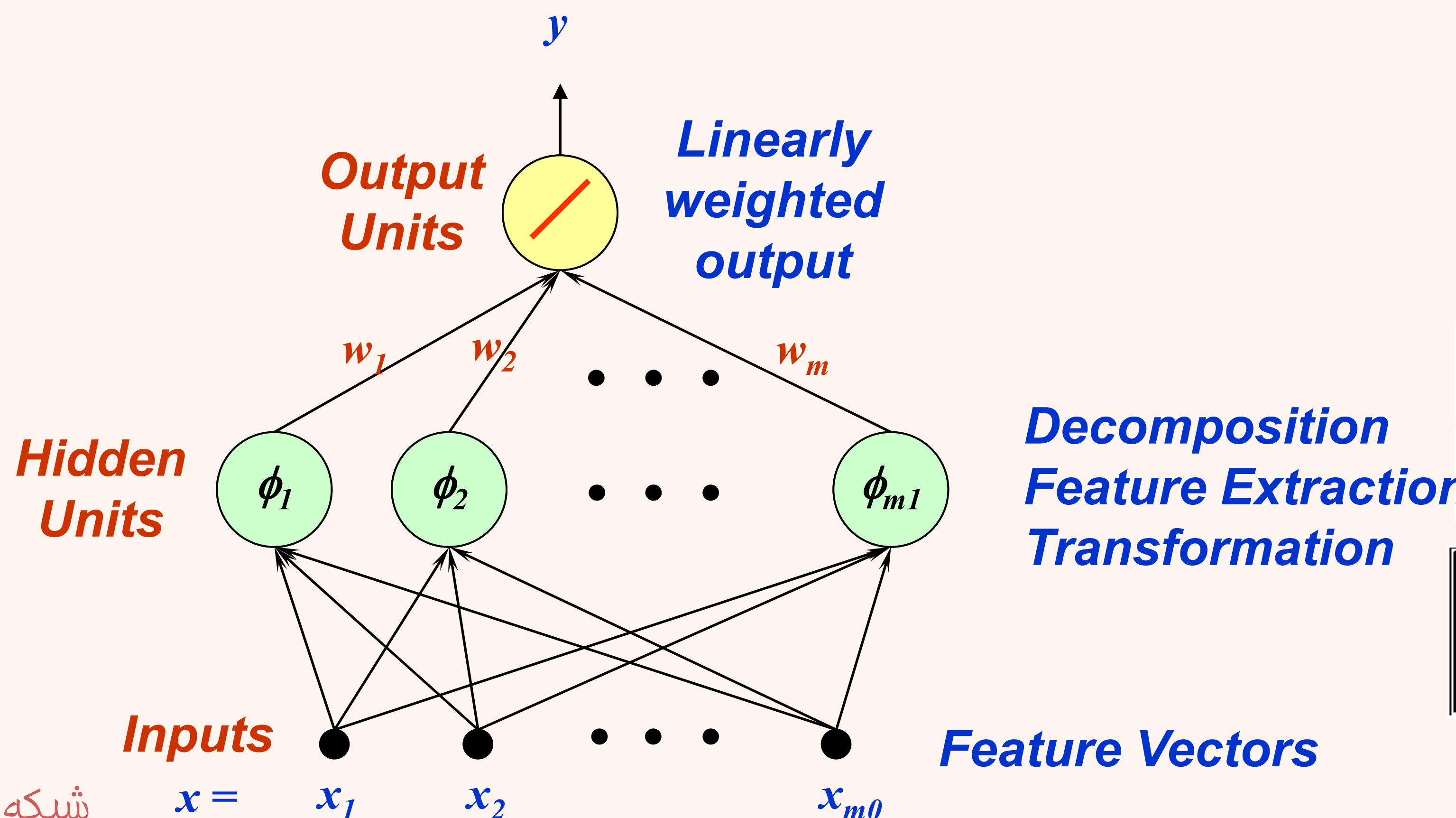
RBF

- ساختار یک شبکه‌ی RBF همانند MLP است؛ بدین‌ترتیب می‌تواند برای «دسته‌بندی» و یا «تقریب تابع» استفاده شود.



Radial Basis Function

$$f(\mathbf{x}) = \sum_{i=1}^{m_1} w_i \phi_i(\mathbf{x})$$



شبکه عصبی

http://aimm02.cse.ttu.edu.tw/class_2008_1/nn.html

RBF

- همان گونه که گفته شد شبکه‌های RBF بر تعریف تابعی وابسته به فاصله از مرکز استوار است.

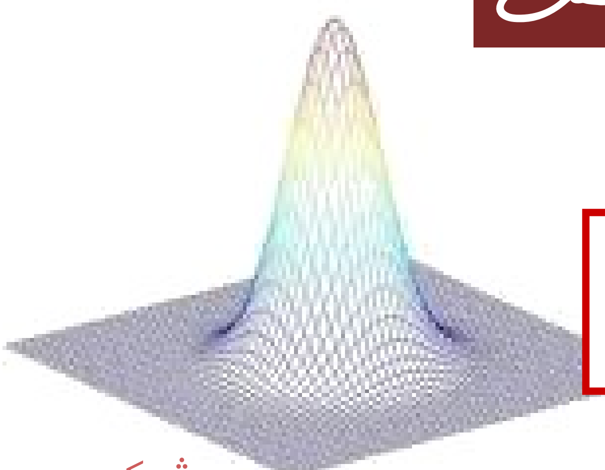
$$\phi_i(x)=\phi\,(\|x-x_i\|)$$

شکل

فاصله

مرکز

$$\|x-y\|=\sqrt{(x-y)^t(x-y)}=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$



چند نمونه تابع فاصله‌ای

$$r = \left\| x - x_j \right\|$$

- گاوسی

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \quad \sigma > 0 \quad \text{and} \quad r \in \Re$$

- **Hardy Multiquadratic**

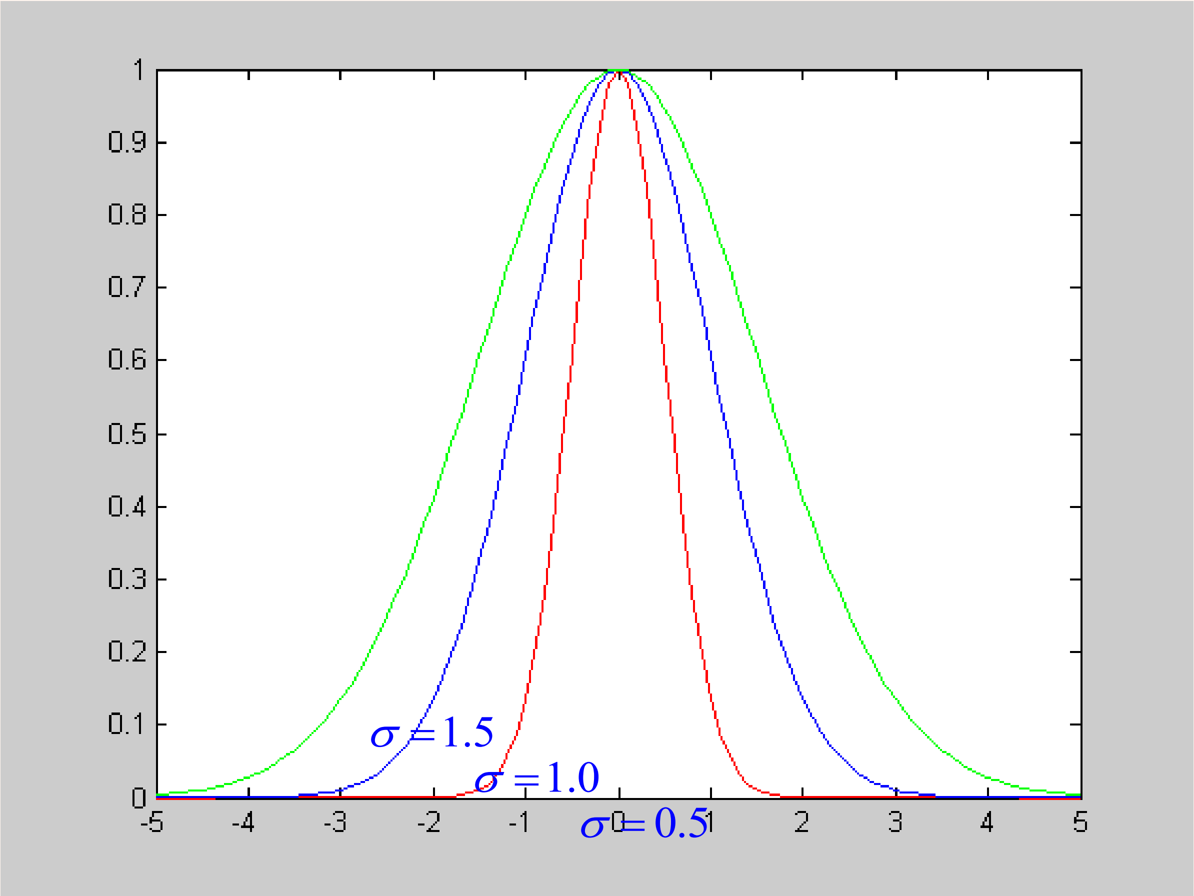
$$\phi(r) = \sqrt{r^2 + c^2} / c \quad c > 0 \quad \text{and} \quad r \in \Re$$

- **Inverse Multiquadratic**

$$\phi(r) = c / \sqrt{r^2 + c^2} \quad c > 0 \quad \text{and} \quad r \in \Re$$

تابع گاوسی

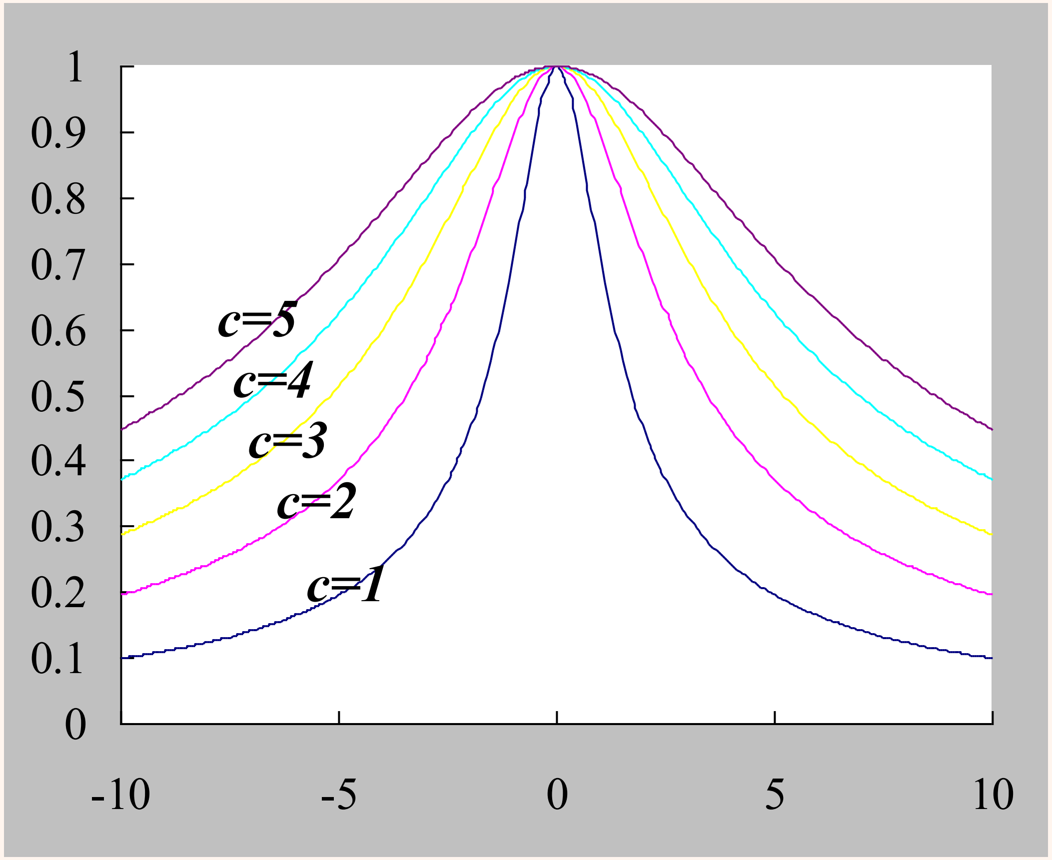
$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \qquad \sigma > 0 \quad \text{and} \quad r \in \Re$$



http://aimm02.cse.ttu.edu.tw/class_2008_1/nn.html

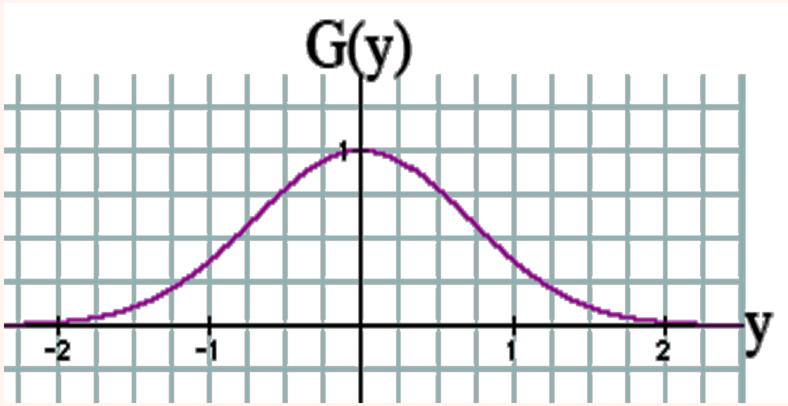
Inverse Multiquadratic

$$\phi(r) = c \Big/ \sqrt{r^2 + c^2} \qquad c > 0 \quad \text{and} \quad r \in \Re$$



RBF

- تابع گاوسی:



$$G(y) = exp(-y^2/2\sigma^2)$$

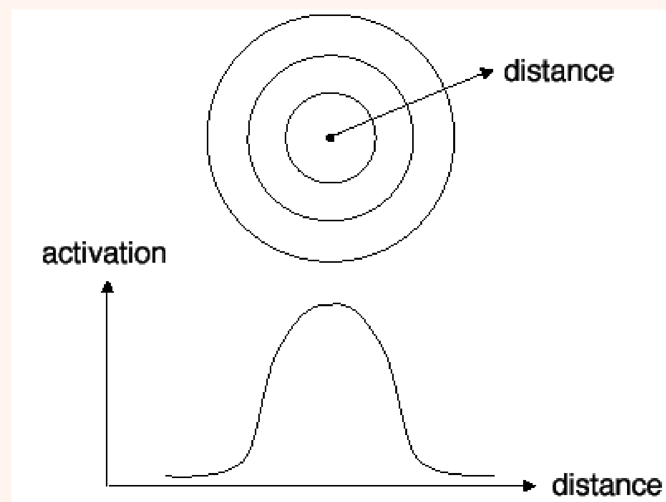
واریانس

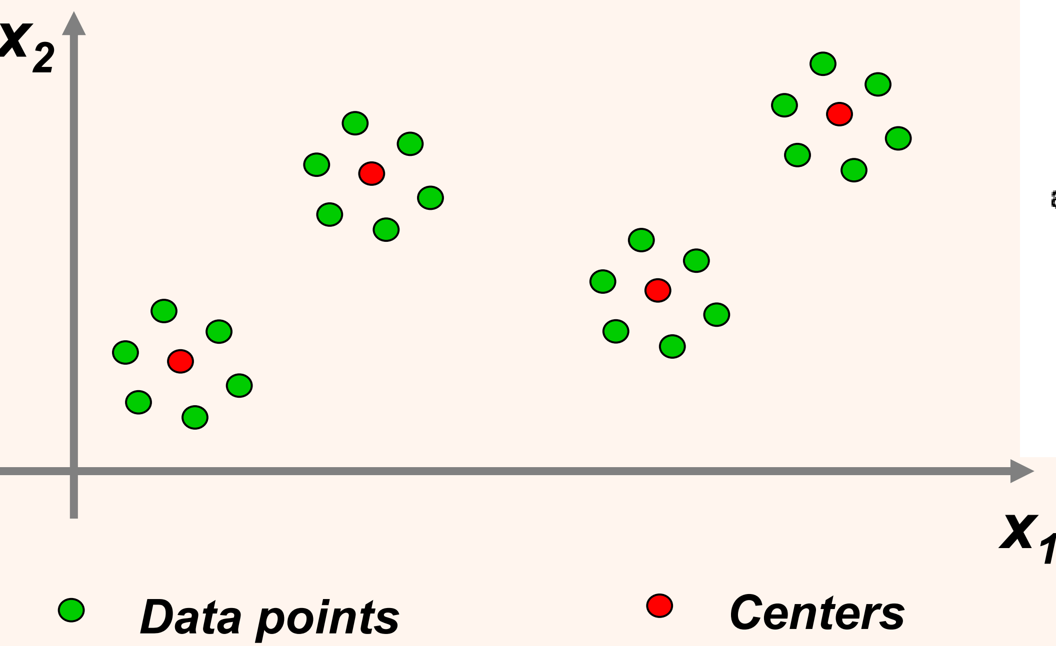
- هر چه واریانس کم‌تر باشد انتخاب‌های محدودتر و هر چه واریانس بالا رود انتخاب‌ها گسترده‌تر است.

به وسیله‌ی سیگما میزان پراکندگی داده را مشخص می‌کنیم.

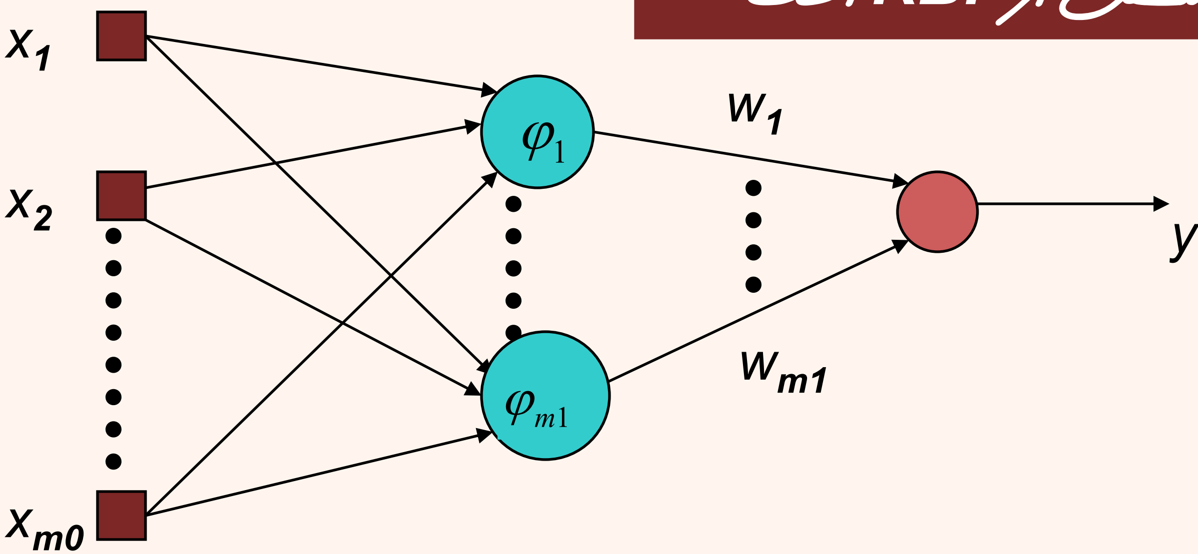
توابع پایه

- خاصیت تقارن از مرکز برای فضای n بعدی وجود دارد (nتعداد مراکز است)
- هرچه از مرکز دورتر شویم ورودی اهمیت کم‌تری پیدا کرده در نتیجه پاسخ کاهشی خواهد بود.





لایه‌ی مخفی متشکل از RBF است



$\varphi_1 \ldots \varphi_{m_1}$

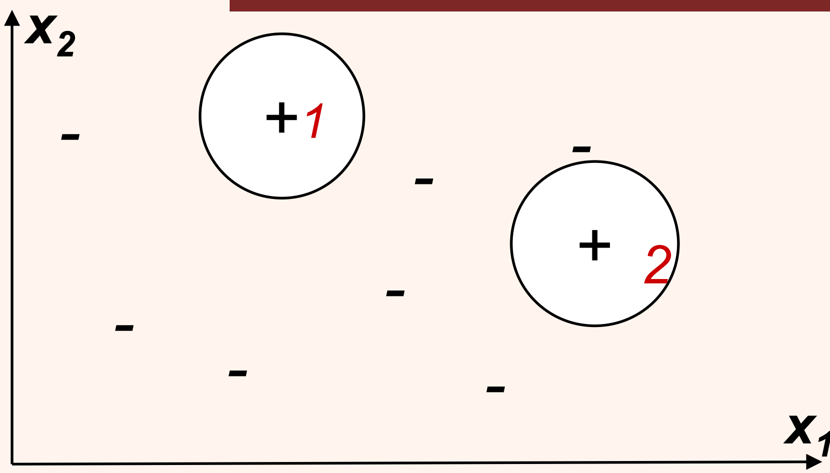
لایه‌ی خروجی متشکل از تابع جدایی‌پذیر خطی است

$$y = w_1 \varphi_1(\| x - t_1 \|) + \ldots + w_{m_1} \varphi_{m_1}(\| x - t_{m_1} \|)$$

$$x = (x_1, \ldots, x_{m_0})$$

مثال

در این مثال هدف جداسازی دو کلاس است: + و -



$<x_1,x_2> \longrightarrow$ input space

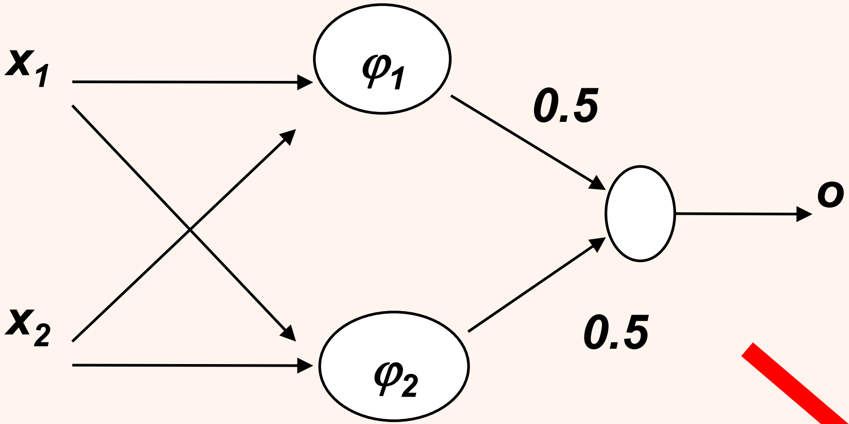
$$\varphi_1(\|x-t_1\|)=\begin{cases}1 \text{ if } \|x-t_1\| <= r_1\\ 0 \text{ if } \|x-t_1\| > r_1\end{cases}$$

$$\varphi_2(\|x-t_2\|)=\begin{cases}1 \text{ if } \|x-t_2\| <= r_2\\ 0 \text{ if } \|x-t_2\| > r_2\end{cases}$$

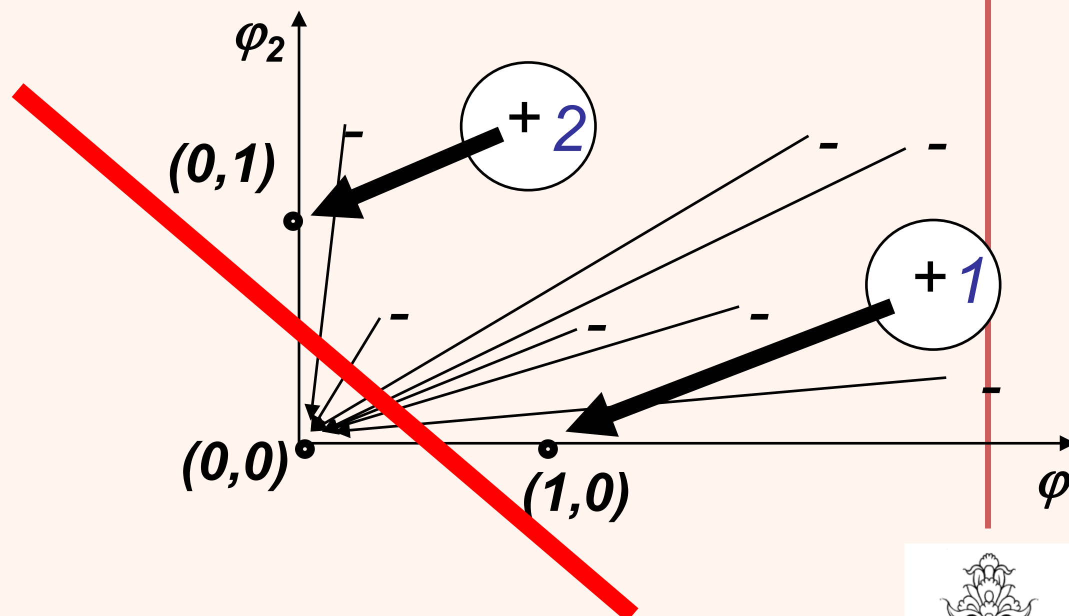
t_1 و t_2 مراکز دو دایره بالا هستند

http://www.ptcdb.edu.ps/~agarad/courses/ANN/Courseoutline.html

ادامه‌ی مثال



$\langle \varphi_1, \varphi_2 \rangle \longrightarrow$ **feature space**



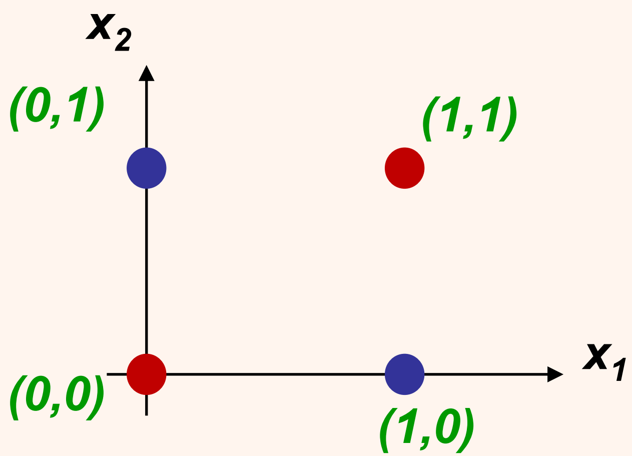
$$\varphi\,(\| x-t \|) = \begin{cases} 1 \text{ if } \| x-t \| <= c \\ 0 \text{ if } \| x-t \| > c \end{cases}$$

جدا شونده‌ی خطی

مثال

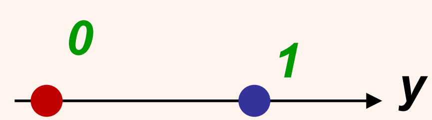
- حل مساله‌ی XOR با شبکه‌ی RBF

ورودی‌ها



$$t_1 = (1,1) \text{ and } t_2 = (0,0)$$

خروجی‌ها



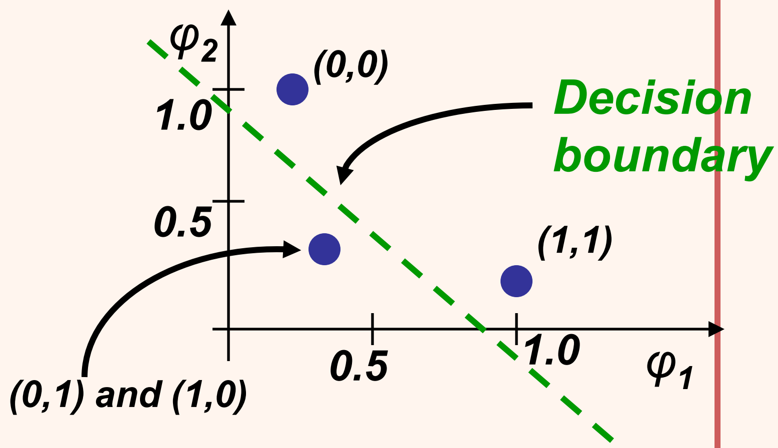
(0,0) و (1,1) نگاشت به صفر شده در یک گروه

(0,1) و (1,0) نگاشت به یک شده در گروه دیگر قرار می‌گیرند

مثال

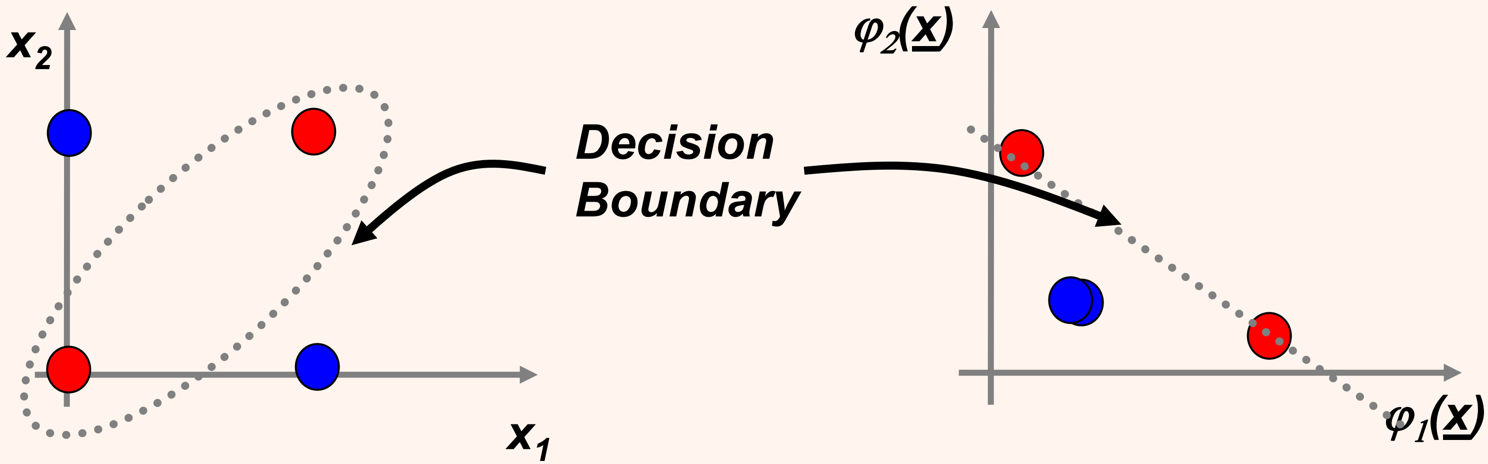
$$\varphi_1(\| x - t_1 \|) = e^{-\|x-t_1\|^2}$$

$$\varphi_2(\| x - t_2 \|) = e^{-\|x-t_2\|^2}$$

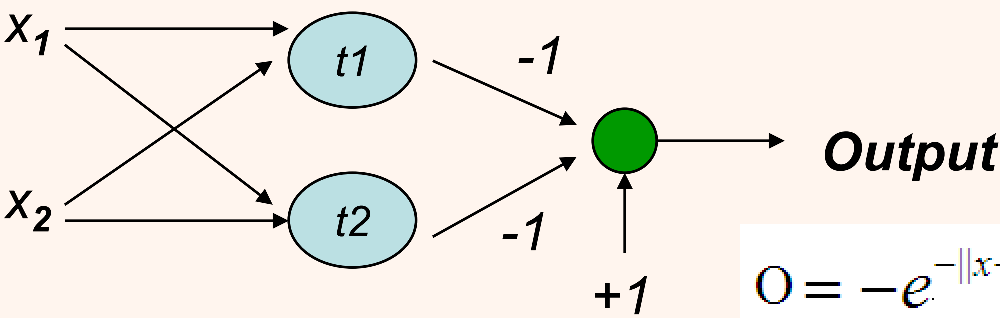


x_1	x_2	o
0	0	0
0	1	1
1	0	1
1	1	0

$\varphi_1(\underline{x})$	$\varphi_2(\underline{x})$	o'
0.13	1	0
0.36	0.36	1
0.36	0.36	1
1	0.13	0



مثال

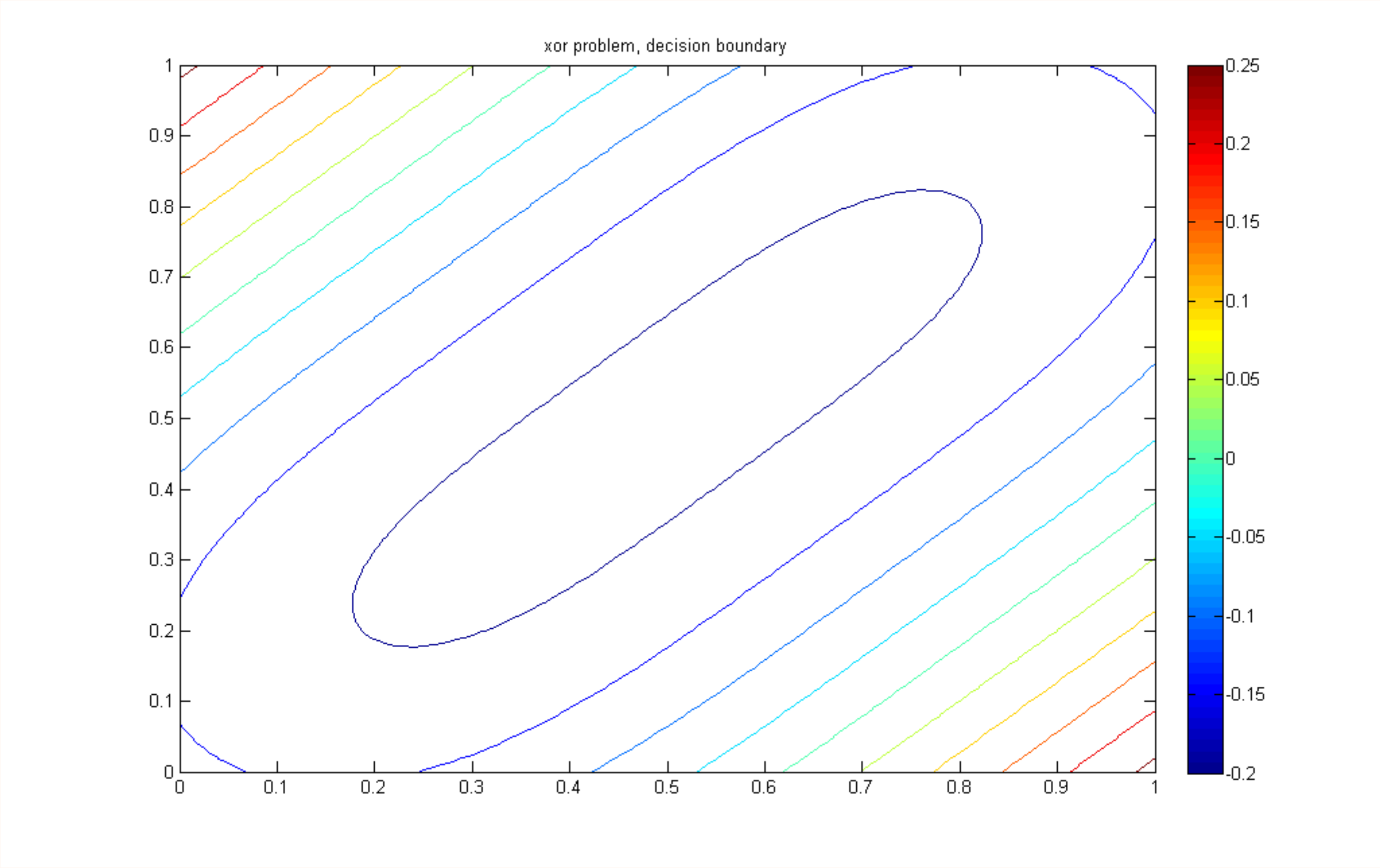


$$O = -e^{-\|x-t_1\|^2} - e^{-\|x-t_2\|^2} + 1$$

If $O > 0$ then class 1 otherwise class 0

```
x1=[0:.01:1];
x2=[0:.01:1];
[X1,X2]=meshgrid(x1,x2);
f1=exp(-((X1-1).^2+(X2-1).^2));
f2=exp(-((X1).^2+(X2).^2));
z=-f1-f2+1;
contour(x1,x2,z);
title('xor problem, decision boundary');
```

مرز تصمیم‌گیری



دسته‌بندی خطی

- یک دسته‌بند پیچیده‌ی غیرخطی در فضای m_0 بعدی را می‌توان با نگاشت به یک فضای بزرگ‌تر به صورت دسته‌بند خطی معادل ساخت.
- فرض کنید N الگوی x_1 x_2 ...... x_N با اندازه‌ی $1\times m_0$ را بخواهیم در دو کلاس A_1 و A_2 طبقه‌بندی کنیم.
- برای هر الگوی آموزشی یک بردار m_1 عضوی جدید که $m_0 \leq m_1$ است تعریف می‌کنیم به گونه‌ای که

$$\varphi(x) = \left[\varphi_1(x), \ldots\ldots, \varphi_{m_1}(x) \right]$$

- پس توسط تابع ϕ می‌خواهیم فرایندی داشته باشیم که:

$$\varphi : R^{m_0} \rightarrow R^{m_1}$$

دسته‌بندی کلاس‌ها

- مجموع تمامی توابع مخفی را به صورت زیر نشان می‌دهیم:

$$\{\varphi_i(x)\}_{i=1}^{m_1}$$

- حال اگر داشته باشیم:

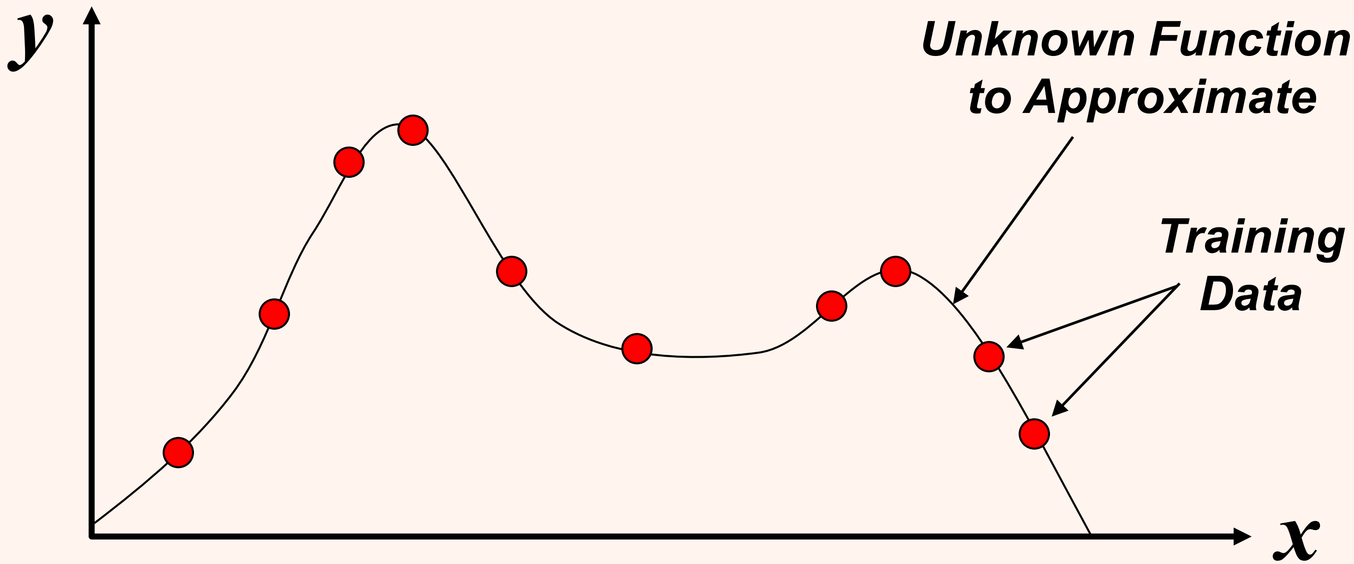
$$W^T\varphi(x) > 0 \Rightarrow x \in A_1$$

$$W^T\varphi(x) < 0 \Rightarrow x \in A_2$$

بردارهای x_i در فضای m_0 بعدی تبدیل به بردارهای جدید m_1 با خاصیت جدایی پذیری خطی شده‌اند.

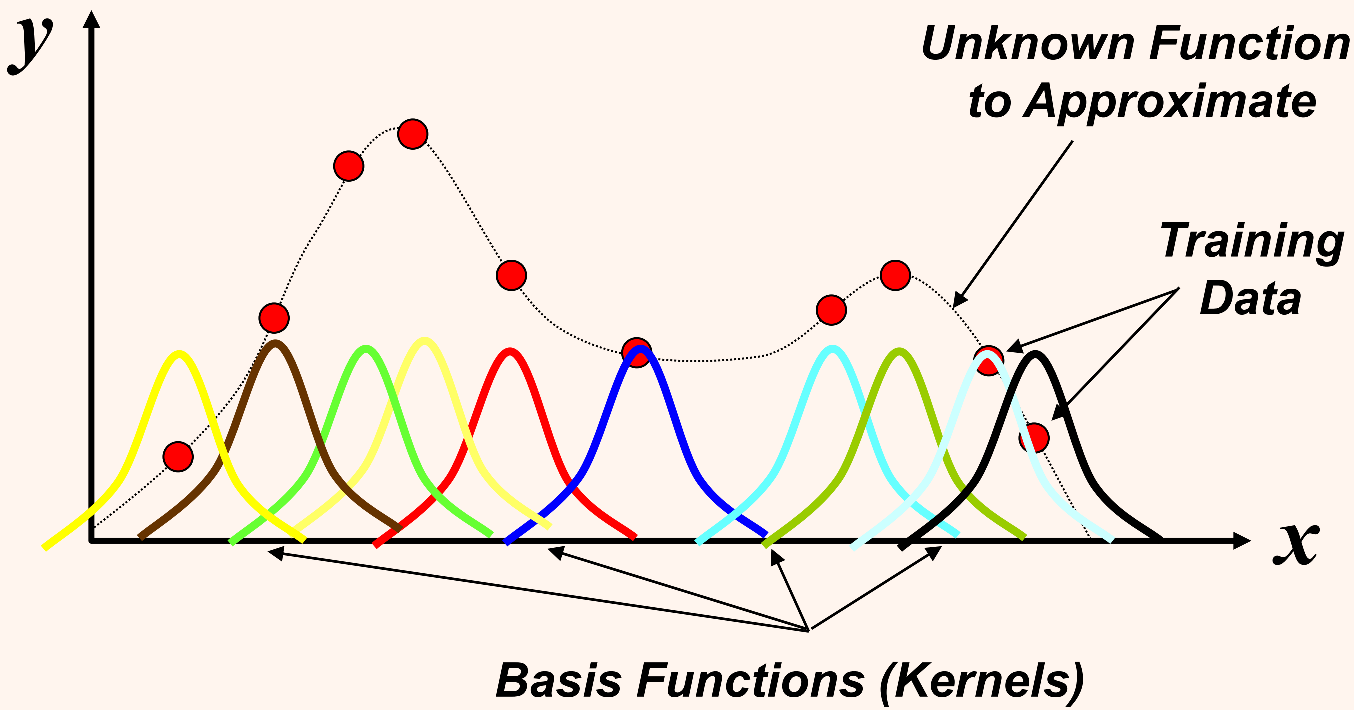
$W^T\varphi(x) = 0$ « نشان‌گر رویه‌ی جداکننده است.

تقریب تابع



http://aimm02.cse.ttu.edu.tw/class_2008_1/nn.html

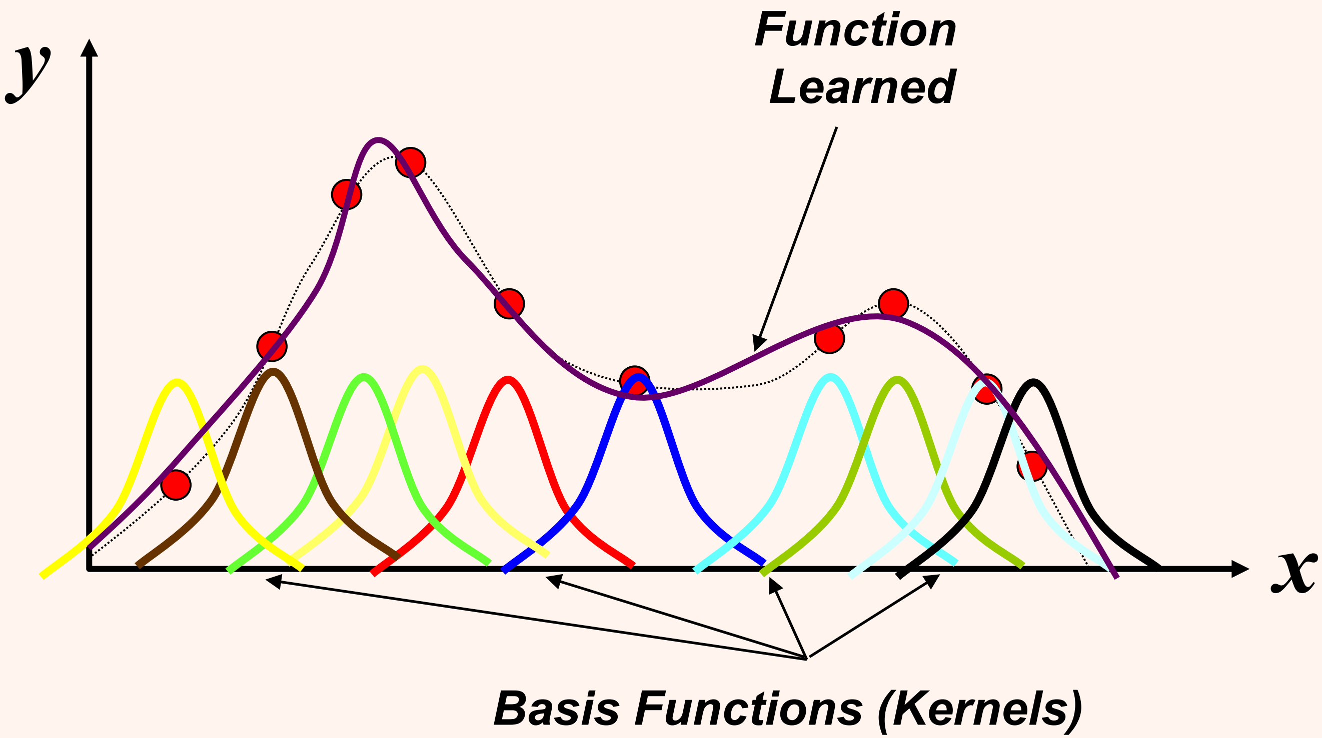
تقریب تابع (ادامه...)



http://aimm02.cse.ttu.edu.tw/class_2008_1/nn.html

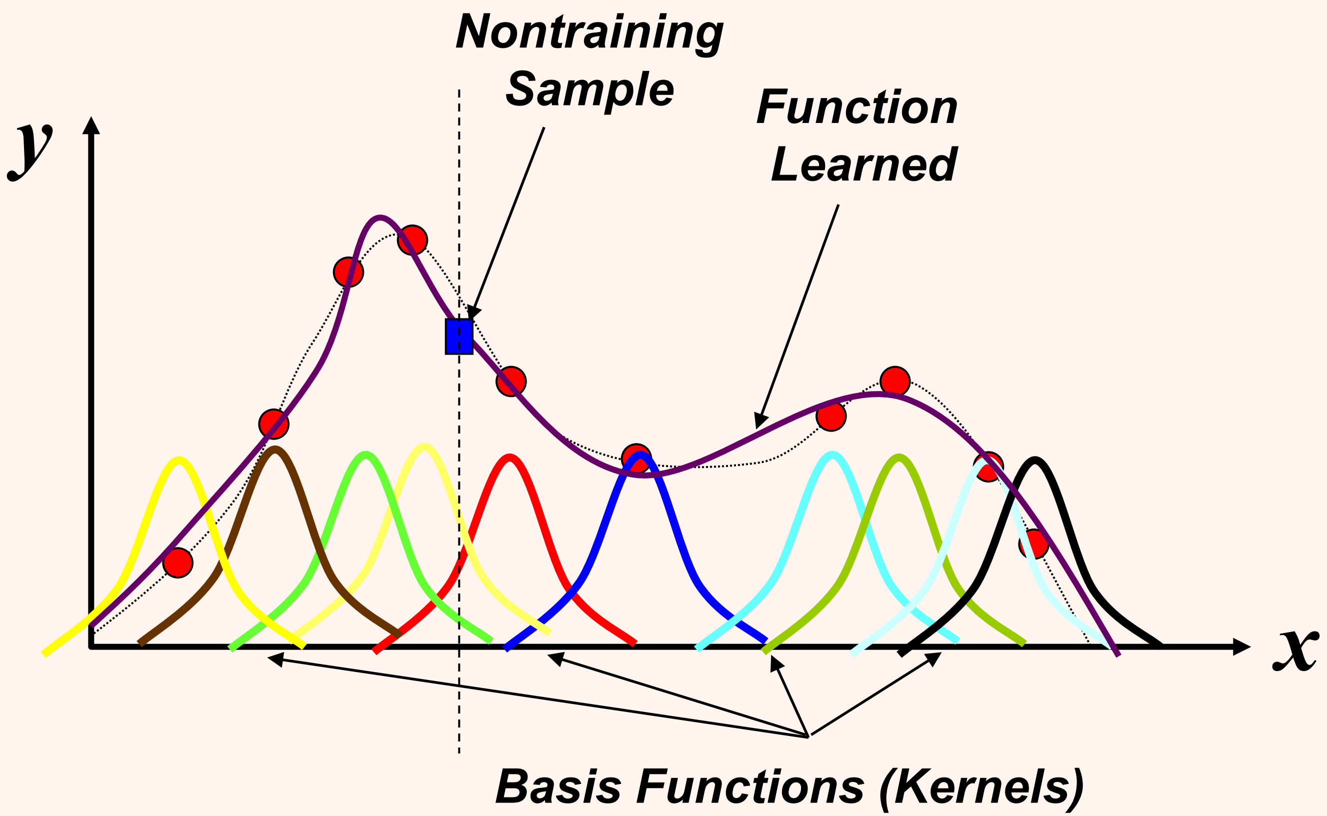
تقریب تابع (ادامه...)

$$y = f(\mathbf{x}) = \sum_{i=1}^{m_1} w_i \phi_i(\mathbf{x})$$



تقریب تابع (ادامه...)

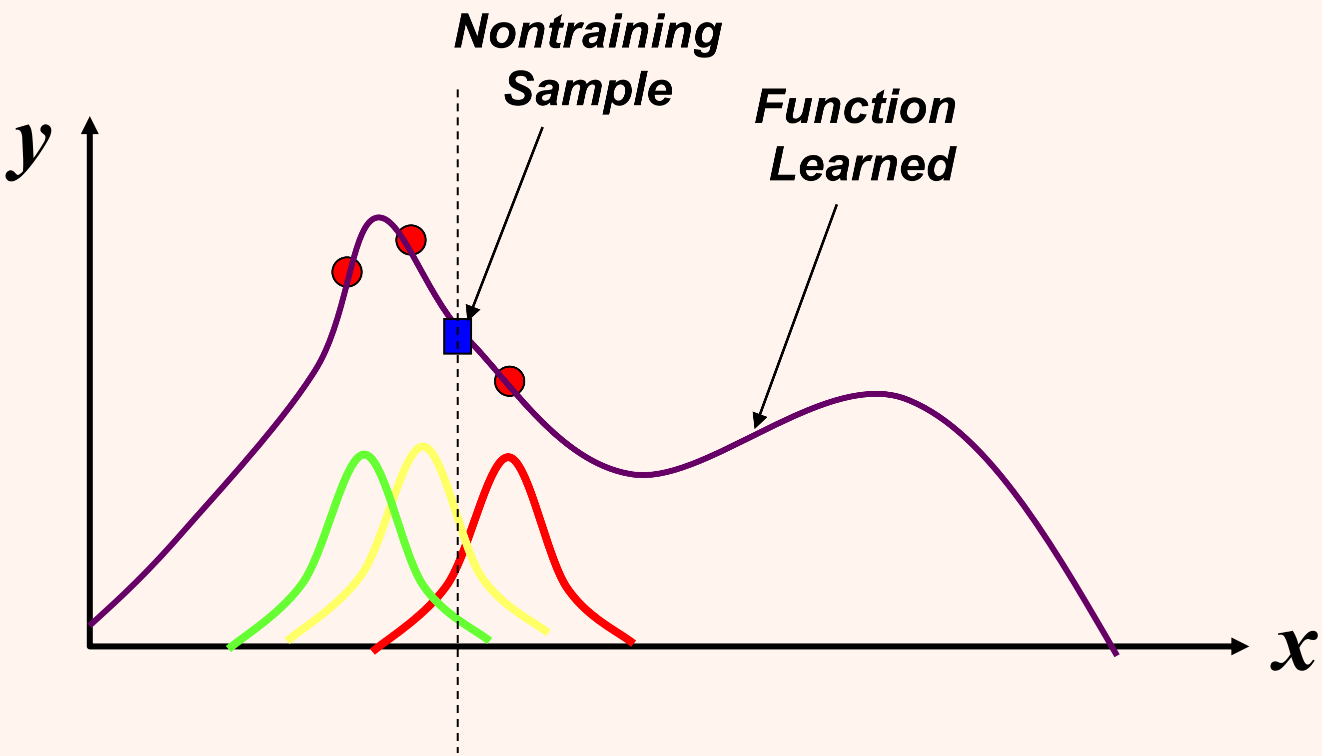
$$y = f(\mathbf{x}) = \sum_{i=1}^{m_1} w_i \phi_i(\mathbf{x})$$



http://aimm02.cse.ttu.edu.tw/class_2008_1/nn.html

تقریب تابع (ادامه...)

$$y = f(\mathbf{x}) = \sum_{i=1}^{m_1} w_i \phi_i(\mathbf{x})$$



http://aimm02.cse.ttu.edu.tw/class_2008_1/nn.html

روش‌های یادگیری

- دقیق
  - الگوهای آموزشی برابر با تعداد واحدهای مخفی
- درون‌یابی تقریبی
  - الگوهای آموزشی بیشتر از تعداد واحدهای مخفی
- مجهول‌ها
  - مراکز t
  - واریانس تابع
  - وزن‌ها

تعداد واحدهای مخفی برابر با تعداد بردارهای ورودی

- حل مسأله در حالت کلی:
  - N بردار متفاوت در فضای m_0 بعدی :

$$\left\{\mathbf{x_i} \in R^{m_0}\right\}_{i=1}^{N}$$

  - N عدد حقیقی

$$\left\{d_i \in R^{1}\right\}_{i=1}^{N}$$

$$F : R^{m_0} \to R^{1}$$

  - مطلوب است یافتن F

$$F(\mathbf{x_i}) = d_i$$

آموزش دقیق

- شبکه‌ی RBF

$$F(\mathbf{x}) = \sum_{j=1}^{N} w_j \varphi\Big(\underbrace{\left\| \mathbf{x} - \mathbf{x}_j \right\|}_{\text{تابع فاصله است}}\Big)$$

$$\left\| \mathbf{x} - \mathbf{x_j} \right\| = \sqrt{\sum_{j=1}^{m_0} (x_k - x_{kj})^2}$$

- یعنی انتخاب F به گونه‌ای که:

$$F(x_i) = \sum_{j=1}^{N} w_j \varphi\left(\left\| \mathbf{x}_i - \mathbf{x}_j \right\|\right) = d_i$$

$$w_1\varphi_1(\| \mathbf{x}_i - \mathbf{x}_1 \|) + w_2\varphi_2(\| \mathbf{x}_i - \mathbf{x}_2 \|) + \ldots + w_N\varphi_N(\| \mathbf{x}_i - \mathbf{x}_N \|) = d_i$$

آموزش دقیق (ادامه...)

• برای سادگی از اندیس‌های توابع شعاعی صرفنظر می‌کنیم:

$$\begin{bmatrix} \varphi(\|x_1-t_1\|) & \varphi(\|x_1-t_2\|)\ldots\ldots\varphi(\|x_1-t_N\|) \\ \varphi(\|x_2-t_1\|) & \varphi(\|x_2-t_2\|)\ldots\ldots\varphi(\|x_2-t_N\|) \\ \ldots & \\ \varphi(\|x_N-t_N\|) & \varphi(\|x_N-t_2\|)\ldots\ldots\varphi(\|x_N-t_N\|) \end{bmatrix} [w_1\ldots w_N]^T = [d_1\ldots d_N]^T$$

$$\Phi\begin{bmatrix} w_1 \\ \ldots \\ w_{m1} \end{bmatrix} = \begin{bmatrix} d_1 \\ \ldots \\ d_N \end{bmatrix} \Rightarrow \mathbf{\Phi.W = D} \Rightarrow W = \Phi^{-1}.D$$

به شرط معکوس‌پذیری جواب دارد

معکوس‌پذیری

- اگر تابع ϕ یکی از توابع زیر باشد و نمونه‌ها تکراری نباشد در این حالت ماتریس درون‌یابی (nonsingular) است و معکوس‌پذیر:

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \quad \sigma > 0 \quad \text{and} \quad r \in \Re$$

$$\phi(r) = \sqrt{r^2 + c^2} / c \quad c > 0 \quad \text{and} \quad r \in \Re$$

$$\phi(r) = c / \sqrt{r^2 + c^2} \quad c > 0 \quad \text{and} \quad r \in \Re$$

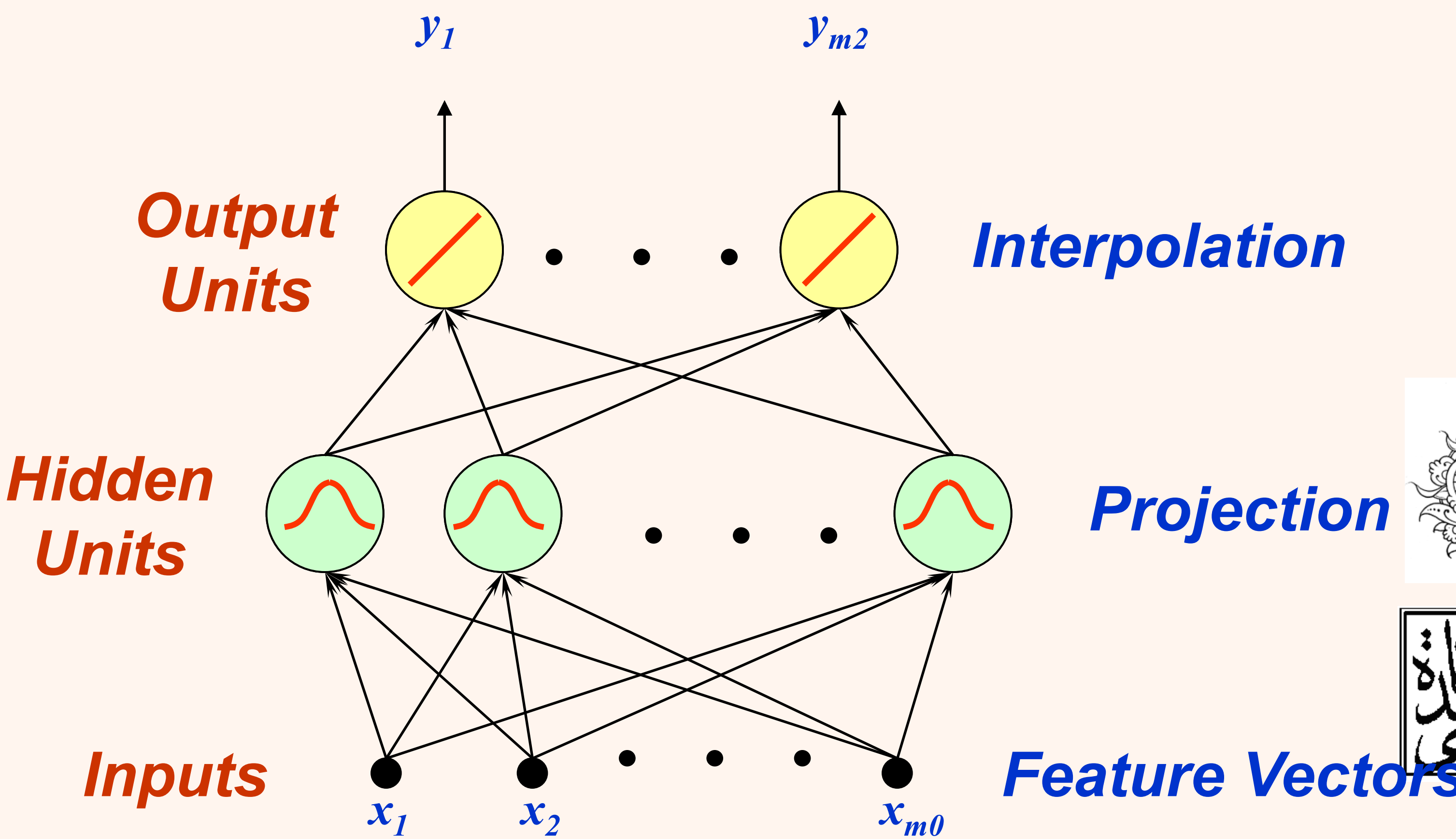
Micchelli's theorem

Micchelli, C. (1986). "Interpolation of scattered data: Distance matrices and conditionally positive definite functions." Constructive Approximation 2(1): 11-22.

ساختار شبکه

- جهت تخمین توابع می‌توان از ساختاری همانند زیر بهره گرفت:

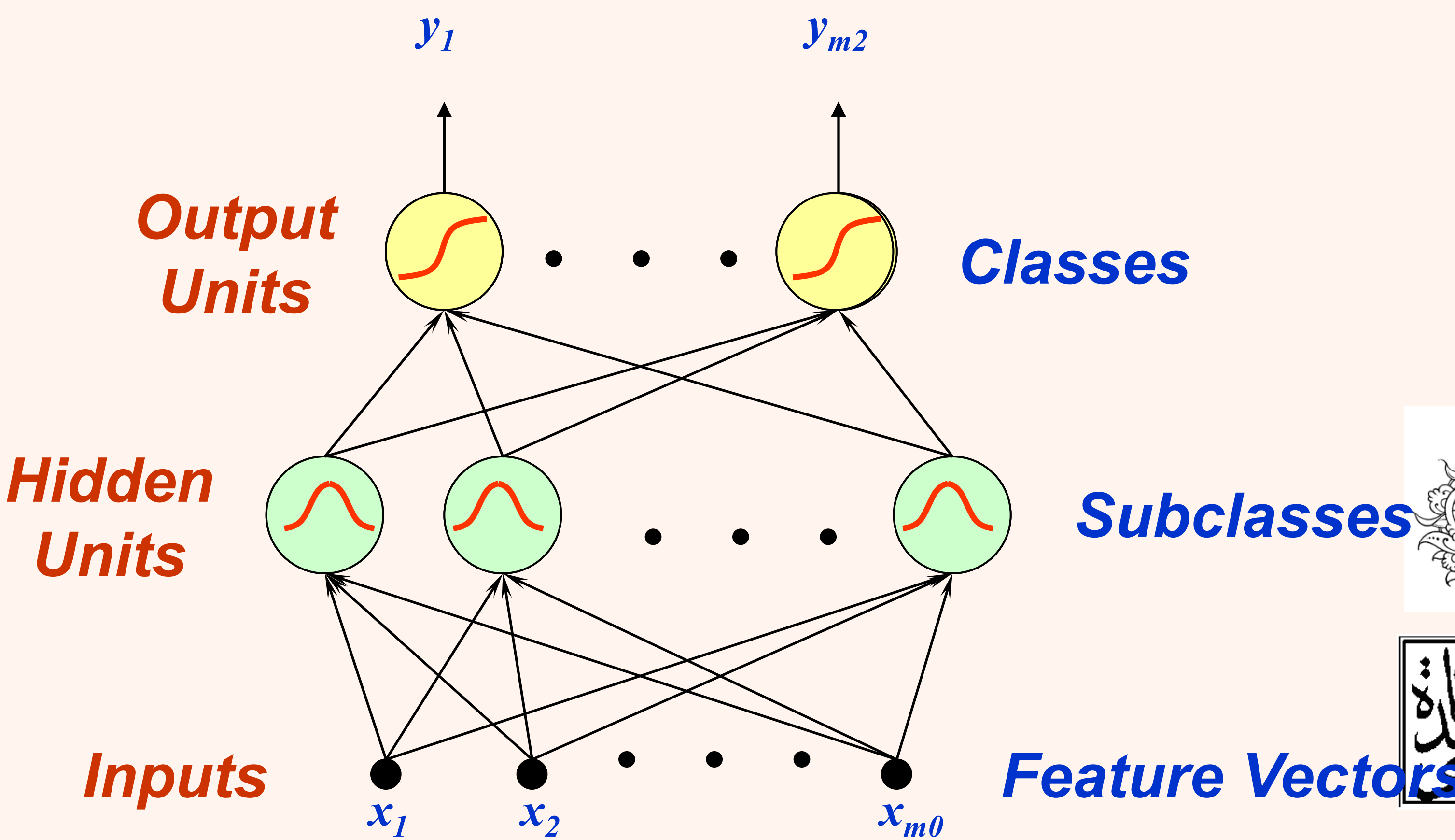
$$y = f(x)$$



http://aimm02.cse.ttu.edu.tw/class_2008_1/nn.html

ساختار شبکه

- هنگامی که از RBF به عنوان Classifier استفاده می‌شود:

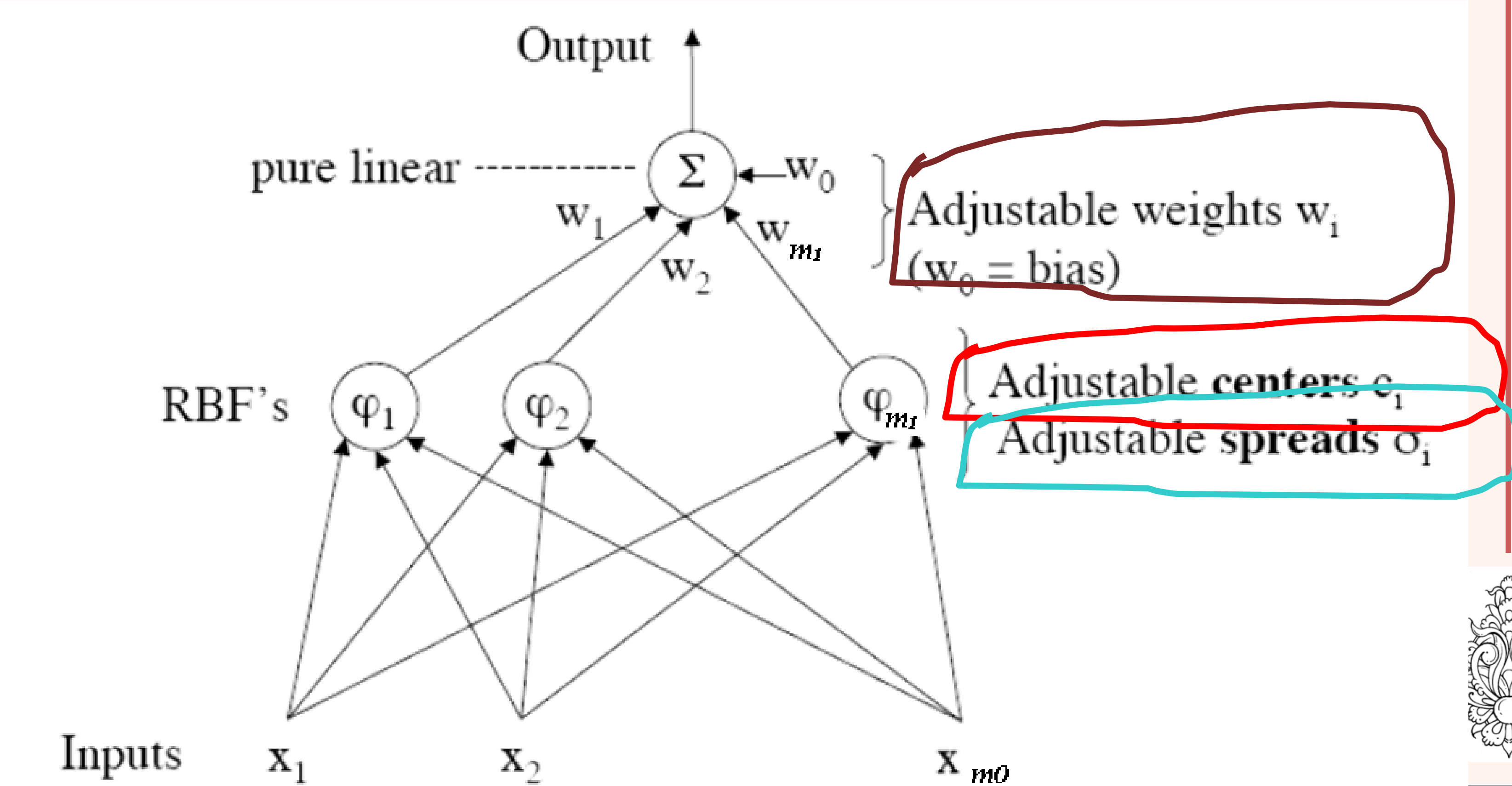


http://aimm02.cse.ttu.edu.tw/class_2008_1/nn.html

اندازه‌ی ماتریس

- اگر N تعداد الگوهای آموزشی باشد، تعداد واحدهای لایه‌ی مخفی را برابر با N در نظر می‌گیریم.
- در این حالت هرچه N بزرگ‌تر شود تعداد گره‌های لایه‌ی مخفی هم بیشتر شده و مشکل معکوس نمودن یک ماتریس بزرگ را در پی خواهد داشت.
- راه حل
  – تعداد واحدهای لایه‌ی مخفی را کم‌تر از N در نظر می‌گیریم ($M<N$)
  – دراین حالت تعداد الگوهای آموزشی برابر با N و تعداد واحدهای لایه‌ی مخفی M است.

پارامترهای آزاد



ماتریس درون‌یابی

تعداد واحدهای مخفی کم‌تر از تعداد بردارهای ورودی

- در این حالت خواهیم داشت:

$$\tilde{F}'(\mathbf{x}) = \sum_{j=0}^{M} w_j \varphi_j(\|\mathbf{x} - \mathbf{x_j}\|) \qquad \varphi_0 = 1$$

بایاس

$$\{t_i = x_i \;,\; M \le N\}$$

- t_i همان مراکز هستند که می‌توانند برابر با xها باشند و یا نباشند.

$$\tilde{F}'(\mathbf{x}) \approx F(\mathbf{x}) \qquad E(\tilde{F}') = \sum_{j=1} (d_j - \tilde{F}'(x_j))^2$$

interpolation matrix

ماتریس درون‌یابی

$$\begin{bmatrix} \varphi_1(\| x_1 - t_1 \|) & \varphi_2(\| x_1 - t_2 \|) \ldots\ldots \varphi_M(\| x_1 - t_M \|) \\ \varphi_1(\| x_2 - t_1 \|) & \varphi_2(\| x_2 - t_2 \|) \ldots\ldots \varphi_M(\| x_2 - t_M \|) \\ \ldots & \\ \varphi_1(\| x_N - t_1 \|) & \varphi_2(\| x_N - t_2 \|) \ldots\ldots \varphi_M(\| x_N - t_M \|) \end{bmatrix} [w_1 \ldots w_M]^T = [d_1 \ldots d_N]^T$$

$$\mathbf{\Phi}_{N\times M} \mathbf{W}_{M\times 1} = \mathbf{D}_{N\times 1}$$

- چون ماتریس نتیجه شده مربعی نیست از طریق محاسبه‌ی ماتریس معکوس نمی‌توان عمل نمود.
  - تعداد داده‌ها از پارامترهای آزاد بیشتر است.

Overdetermined problem

ماتریس درون‌یابی

- هنگامی جواب بهینه است که حداقل خطا را داشته باشیم:

$$E(\tilde{F}) = \sum_{j=1} (d_j - \tilde{F}(x_j))^2$$

$$\mathbf{E} = \|\mathbf{D} - \mathbf{\Phi W}\|^2 = (\mathbf{D} - \mathbf{\Phi W})^T (\mathbf{D} - \mathbf{\Phi W})$$

$$= \mathbf{D}^T\mathbf{D} - \mathbf{W}^T\mathbf{\Phi}^T\mathbf{D} - \mathbf{D}^T\mathbf{\Phi W} + \mathbf{W}^T\mathbf{\Phi}^T\mathbf{\Phi W}$$

$$= \mathbf{D}^T\mathbf{D} - 2\mathbf{W}^T\mathbf{\Phi}^T\mathbf{D} + \mathbf{W}^T\mathbf{\Phi}^T\mathbf{\Phi W}$$

ماتریس درون‌یابی

- برای حداقل کردن خطا

$$\nabla_{\mathbf{w}}\mathbf{E} = \mathbf{0}$$

$$\mathbf{E} = \mathbf{D}\mathbf{D}^T - 2\mathbf{W}^T\mathbf{\Phi}^T\mathbf{D} + \mathbf{W}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{W}$$

$$\nabla_{\mathbf{w}}\mathbf{E} = -2\Phi^T D + 2\Phi^T\Phi W$$

$$= \underbrace{-2\,\underbrace{\underbrace{\mathbf{\Phi}^T}_{M\times N}\;\underbrace{\mathbf{D}}_{N\times 1}}_{M\times 1} + 2\,\underbrace{\underbrace{\underbrace{\mathbf{\Phi}^T}_{M\times N}\;\underbrace{\mathbf{\Phi}}_{N\times M}}_{M\times M}\;\underbrace{\mathbf{W}}_{M\times 1}}_{M\times 1}}_{M\times 1}$$

$$\mathbf{W} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{D}$$

$$\boxed{\mathbf{W} = \mathbf{\Phi}^{+}\mathbf{D}}$$

Pseudo Inverse

$$\boxed{M = N \rightarrow W = \Phi^{-1}D}$$

مالت ایده‌آل

مثال(XOR)

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

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

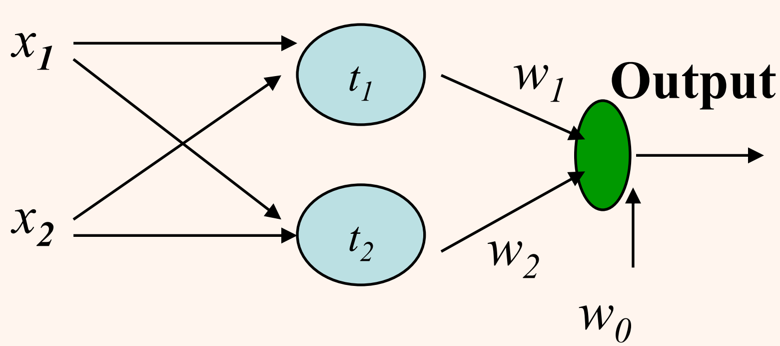
$$y(x) = \sum_{j=0}^{2} w_j \varphi_j(\|x - t_j\|) + w_0$$

$$\varphi(x - t_j) = \exp(-\|x - t_j\|^2)$$

$$\boxed{t_1 = [1\ 1]^T \qquad t_2 = [0\ \ 0]^T}$$

$$\mathbf{\Phi W = D}$$

$$\begin{bmatrix} 1 & 1 & 0.13 \\ 1 & 0.36 & 0.36 \\ 1 & 0.13 & 1 \\ 1 & 0.36 & 0.36 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$



$$\mathbf{W} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{D} = \begin{bmatrix} -2.5 \\ -2.5 \\ 2.8 \end{bmatrix}$$

مثال (ادامه...)

$$\mathbf{W} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{D}$$

- در صورت در نظر گرفتن دو ورودی دیگر به عنوان مرکز

$$\boxed{\mathbf{t}_1 = [1\ 0]^{\mathrm{T}} \qquad \mathbf{t}_2 = [0\ \ 1]^{\mathrm{T}}}$$

$$\mathbf{W} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{D} = \begin{bmatrix} 1.9 \\ 1.9 \\ -1.4 \end{bmatrix}$$

استراتژی‌های یادگیری

Fixed Center Selected at Random

الگوریتم یک

– مراکز به صورت ثابت از میان الگوهای آموزشی انتخاب می‌شوند.

– انحراف معیار از رابطه‌ی زیر محاسبه می‌شود:

$$\boxed{\sigma = \frac{\text{Maximum distance between any 2 centers}}{\sqrt{\text{number of centers}}} = \frac{d_{\max}}{\sqrt{2m_1}}}$$

$$\varphi(\|x-t_i\|^2) = \exp\left(-\frac{m_1}{d_{\max}^2}\|x-t_i\|^2\right) \qquad i = 1, 2, \ldots\ldots, m_1$$

– با توجه به مقادیر فوق تابع مورد نظر انتخاب می‌شود.

– وزن‌ها با توجه به رابطه‌ی زیر محاسبه می‌شود:

$$\boxed{\mathbf{W} = \mathbf{\Phi}^{+}\mathbf{D}}$$

محاسبه‌ی ماتریس شبه‌معکوس

- اگر G یک ماتریس $N \times M$ باشد، ماتریس‌های متعامد U و V وجود دارند به صورتی که:

$$\mathbf{U} = \{u_1, u_2, \ldots\ldots\ldots, u_N\} \qquad \mathbf{V} = \{v_1, v_2, \ldots\ldots\ldots, v_M\}$$

- به صورتی که:

$$U^T G V = diag\{\sigma_1, \sigma_2, \ldots\ldots\ldots, \sigma_k\} \qquad k = \min\{M, N\}$$

- در این صورت برای محاسبه‌ی G^+ داریم:

$$\mathbf{G}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T$$

- که در آن:

$$\mathbf{\Sigma}^+ = diag\{1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_k, 0, 0\ldots, 0\}$$

- با محاسبه‌ی ماتریس فوق وزن‌ها قابل محاسبه است.

$$\text{where } \Sigma^+ is\ M \times N \text{ matrix}$$

Self Organizing Selection of Centers

الگوریتم دو

- M واحد مخفی انتخاب می‌شود.

1. مقداردهی اولیه: یک سری متغیر تصادفی برای t_kها (مراکز) در نظر گرفته می‌شود. $t_k(0)$
2. یک ورودی x انتخاب و به شبکه اعمال می‌شود.
3. فاصله‌ی بردار x را از تمامی مراکز به دست می‌آوریم و آن مرکز که نزدیک‌ترین است را به روز می‌نماییم:

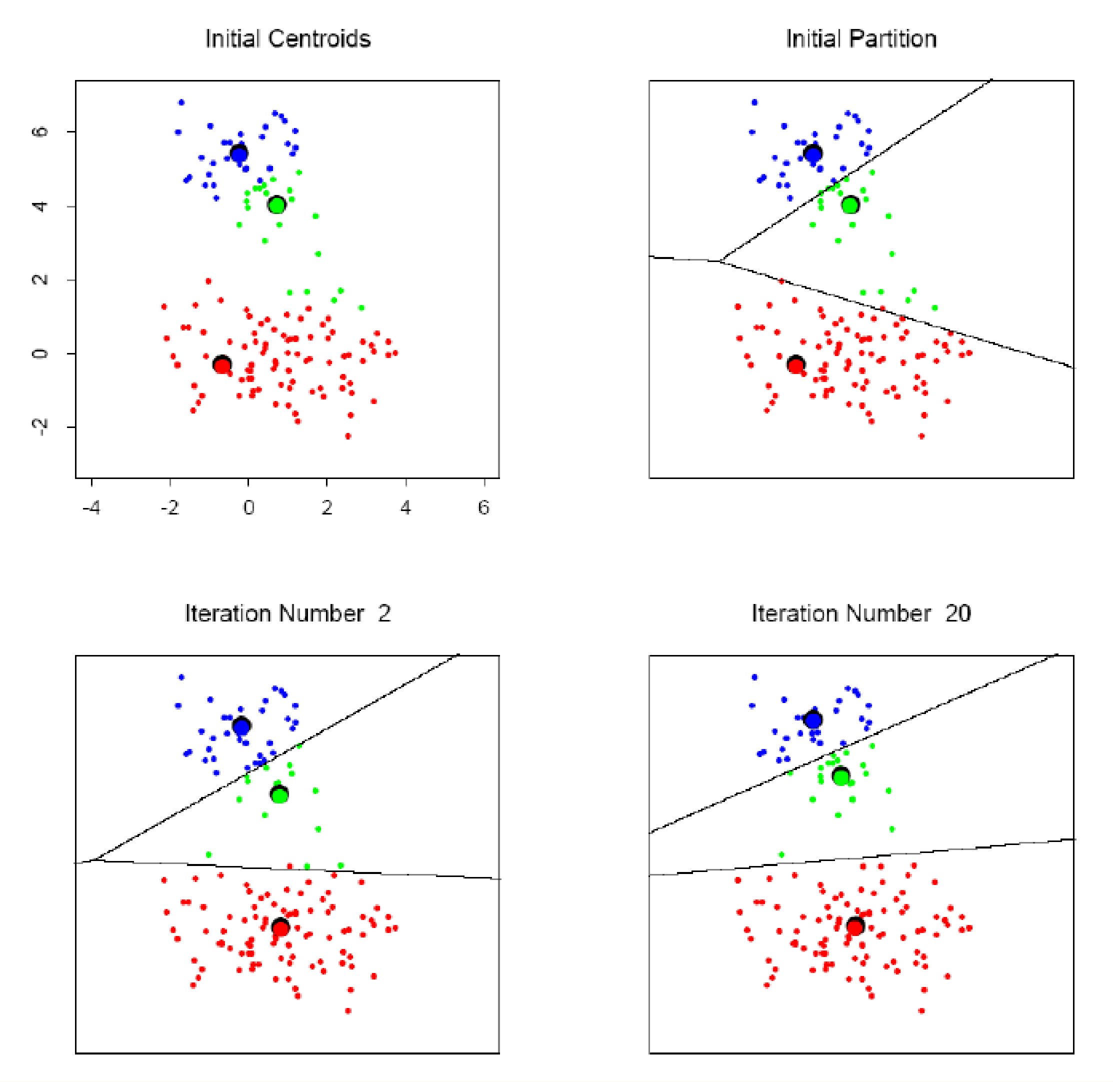
$$h(x) = \arg \min_{k} \left\| x - t_k(n) \right\|^2 \qquad k = 1, 2, \ldots\ldots, m_1$$

$$t_k(n+1) = \begin{cases} t_k(n) + \eta\left[x(n) - t_k(n)\right] & \text{if } k = h(x) \\ t_k(n) & \text{otherwise} \end{cases}$$

4. بازگشت به مرحله‌ی دوم تا زمانی که مراکز تغییر چندانی نداشته باشند.

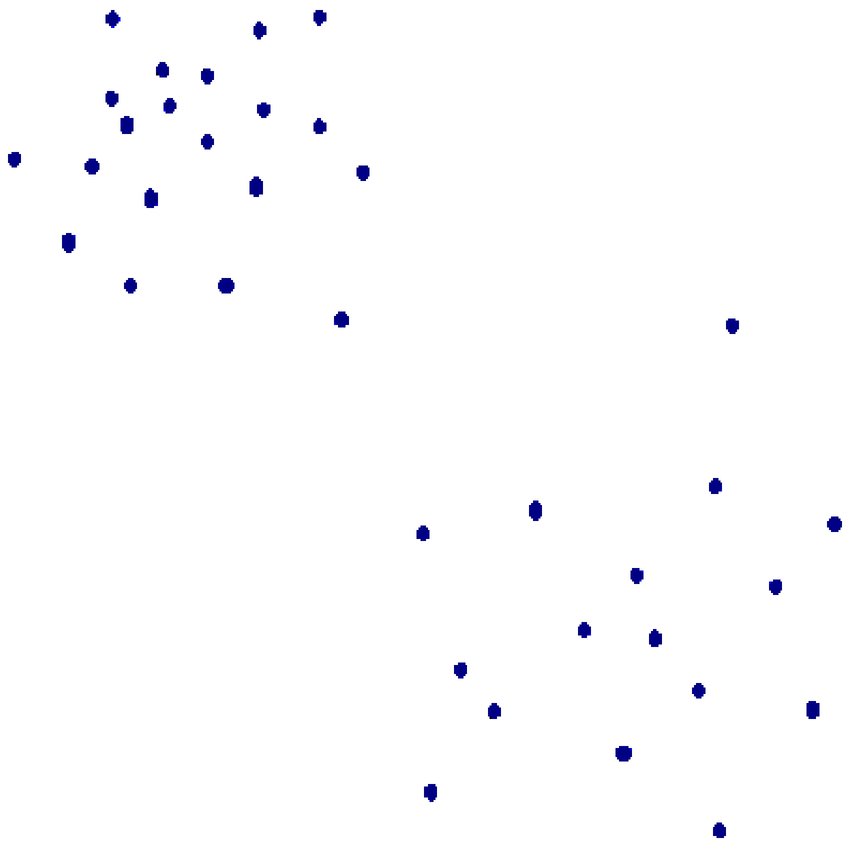
http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

K-means clustering example



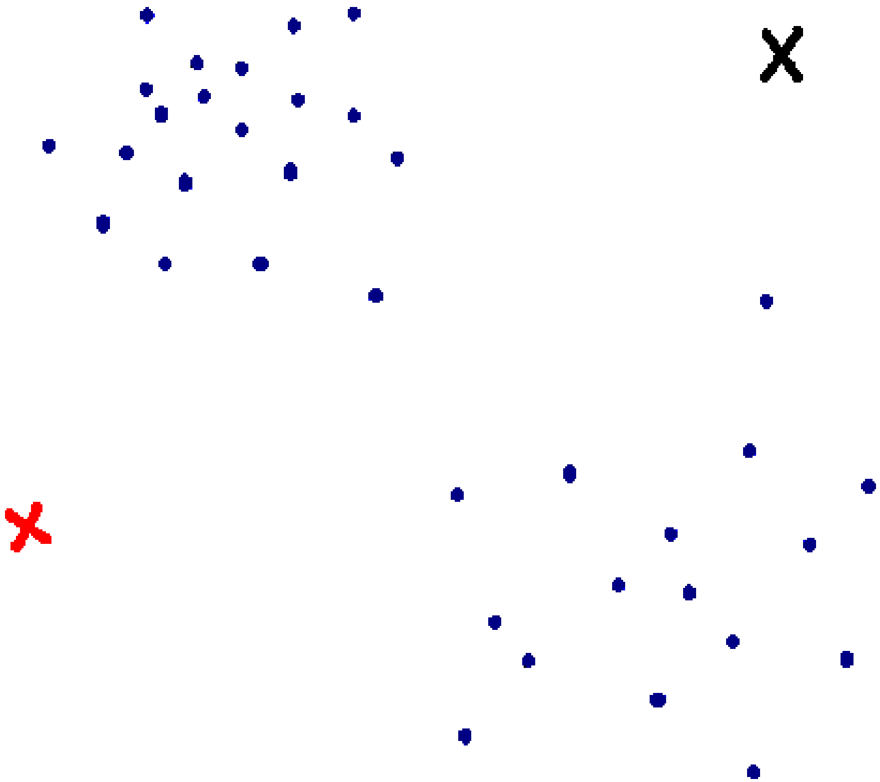
K-means clustering example

مثال



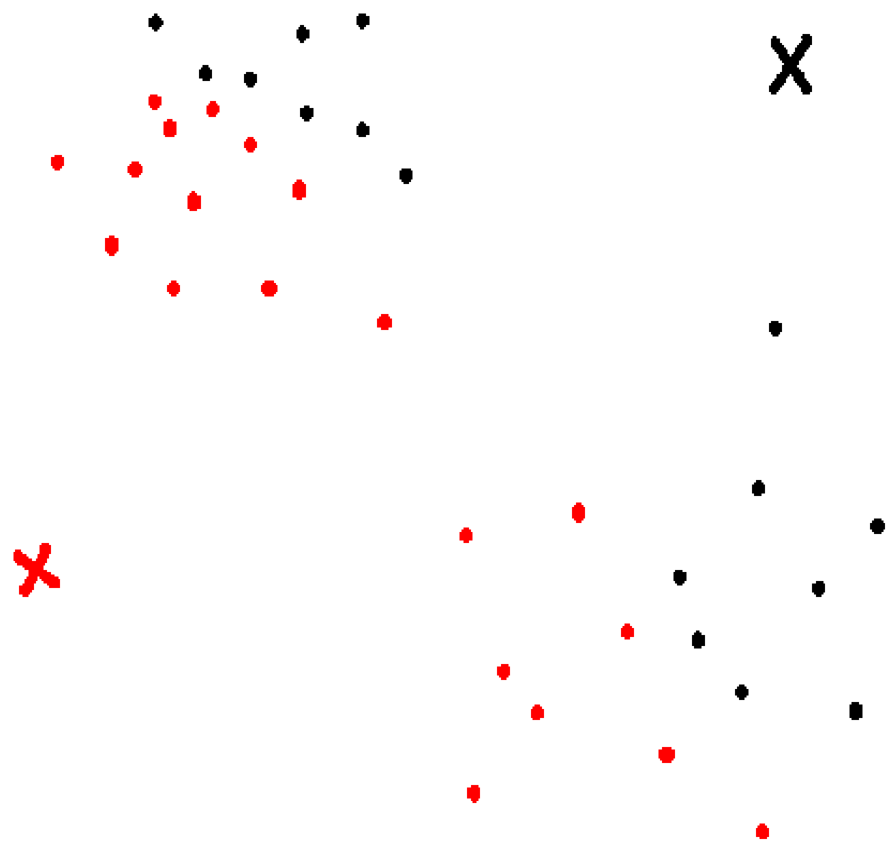
K-means clustering example

مثال



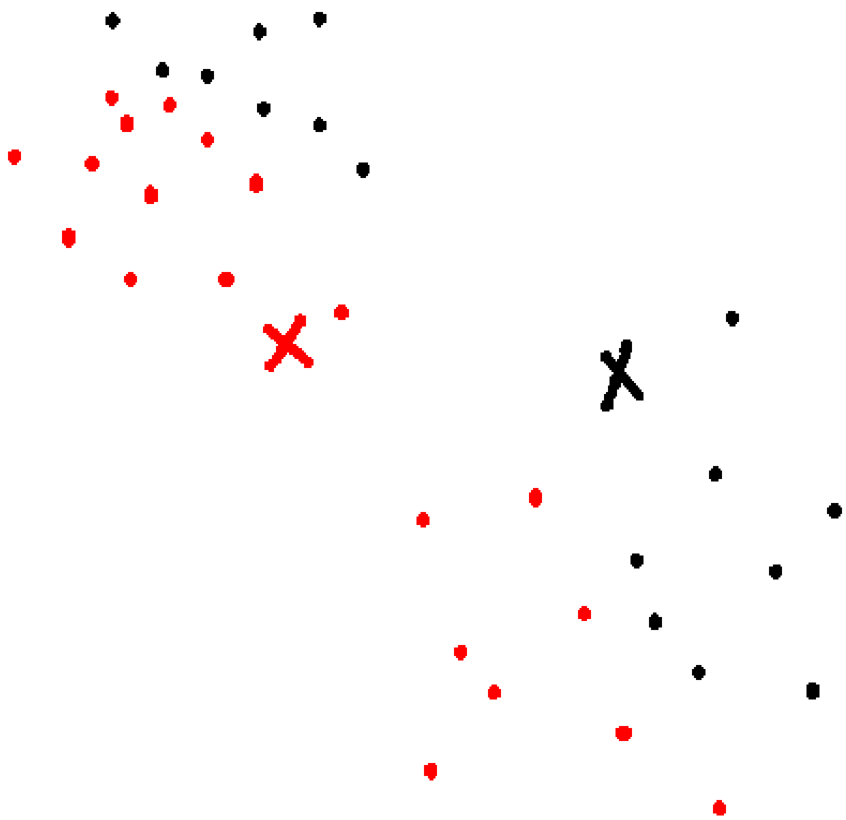
K-means clustering example

مثال



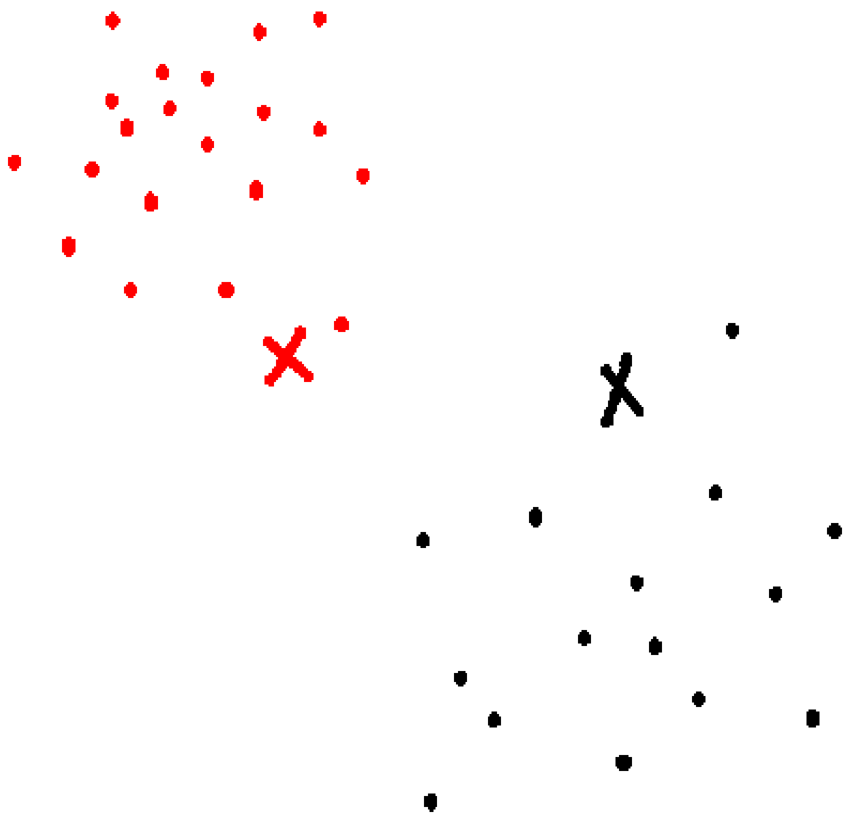
K-means clustering example

مثال



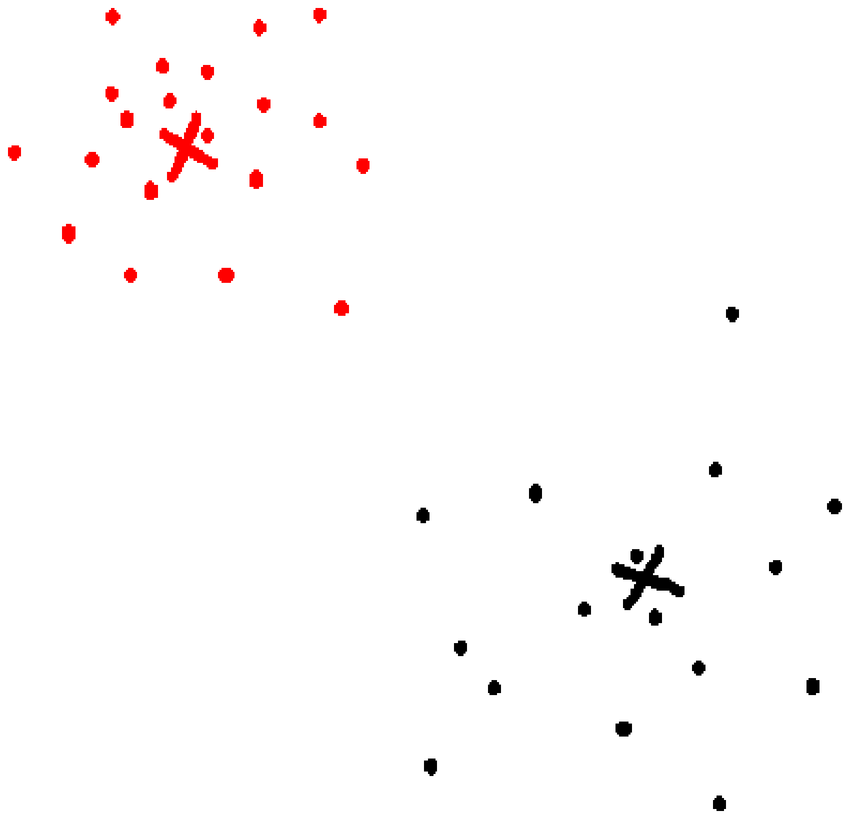
K-means clustering example

مثال



K-means clustering example

مثال



Self Organizing

- مراکز از طریق فرآیند خوشه‌بندی (Clustering) مشخص گردید.
- انحراف معیار:

$$\sigma = \frac{\text{Maximum distance between any 2 centers}}{\sqrt{\text{number of centers}}} = \frac{d_{\max}}{\sqrt{2m_1}}$$

- وزن‌ها از طریق الگوریتم LMS محاسبه می‌گردد.

سیستم Supervised

به روزرسانی وزن‌ها

- به روز رسانی وزن‌ها برپایه‌ی .G.D است.
- ابتدا تابع معیار خطا تعریف می‌شود.

$$\tilde{E}(\tilde{F}) = \sum_{j=1}^{N} (d_j - \tilde{F}(x_j))^2$$

$$= \sum_{j=1}^{N} \left[d_j - \sum_{i=1}^{M} w_i \varphi(\|x_j - t_i\|) \right]^2$$

$$\frac{\partial E(n)}{\partial w_i(n)} = \sum_{j=1}^{N} (-2e_j)\varphi(\|x_j - t_{i(n)}\|)$$

$$w_i(n+1) = w_i(n) - \eta_1 \frac{\partial E(n)}{\partial w_i(n)} \quad 1 \le i \le M$$

مراکز

به روزرسانی مراکز

- برای به روز رسانی مراکز می‌باید از تابع معیار خطا برحسب t_i مشتق گرفته شود:

$$E(\tilde{F}) = \sum_{j=1}^{N} (d_j - \tilde{F}(x_j))^2 = \sum_{j=1}^{N} \left[d_j - \sum_{i=1}^{M} w_i \varphi(\|x_j - t_i\|) \right]^2$$

$$\frac{\partial E(n)}{\partial t_i(n)} = \sum_{j=1}^{N} (2e_j)(-w_i) \frac{\partial \varphi(\|x_j - t_i\|)}{\partial t_i}$$

$$= \sum_{j=1}^{N} (-2e_j)(w_i) \frac{\partial \varphi(\|x_j - t_i\|)}{\partial \|x_j - t_i\|^2} \times \frac{\partial \|x_j - t_i\|^2}{\partial t_i}$$

$$= \sum_{j=1}^{N} (-2e_j)(w_i) \varphi'(\|x_j - t_i\|)(-2(x_j - t_i))$$

$$t_i(n+1) = t_i(n) - \eta_2 \frac{\partial E(n)}{\partial t_i(n)}$$

الگوریتم سه

$$\frac{\partial \mathrm{E(n)}}{\partial \sigma_i(n)} = \sum_{j=1}^{N} (2e_j)(-w_i)\frac{\partial \varphi(\|x_j - t_i\|)}{\partial \sigma_i}$$

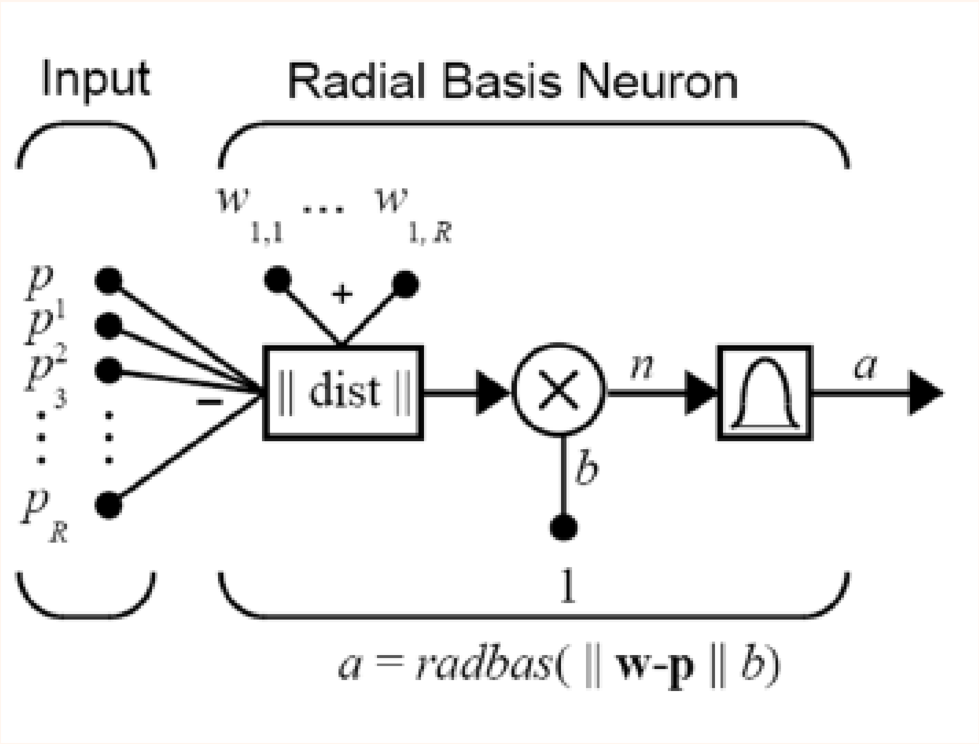
$$A = \frac{\partial \exp[\frac{-r^2}{2\sigma_i^2}]}{\partial \sigma_i} = \frac{4r^2\sigma_i(n)}{4\sigma_{i(n)}^4} \times \exp[\frac{-r^2}{2\sigma_i^2}]$$

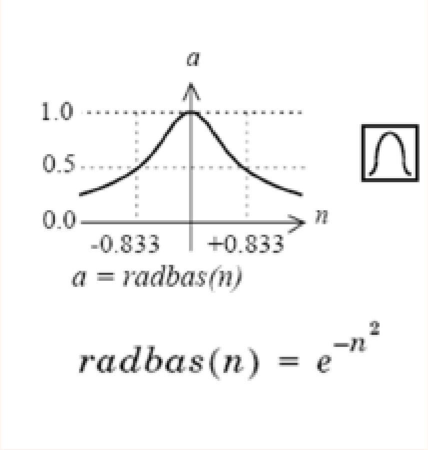
$$\sigma_i(n+1) = \sigma_i(n) - \eta_3 \frac{\partial E(n)}{\partial \sigma_i(n)}$$

مقایسه میان RBF و MLP

- در MLP می‌توانیم چند لایه‌ی مخفی داشته باشیم حال آن که در RBF یک تنها لایه‌ی مخفی داریم.
- معمولاً در MLP اگر برای pattern classification استفاده شود تمامی توابع غیر خطی‌اند.
- آرگومان توابع در RBF فاصله‌ی اقلیدسی است حال آن که در MLP این آرگومان ضرب داخلی بردار ورودی لایه در وزن هاست.
- در MLP یک تقریب کلی از رابطه‌ی ورودی-خروجی به دست می‌آورد در صورتی که در RBF این تخمین به صورت محلی محاسبه می‌شود.
- سرعت یادگیری RBF بیشتر است، در عین حال تعداد پارامترهای آزاد آن هم بیشتر است.

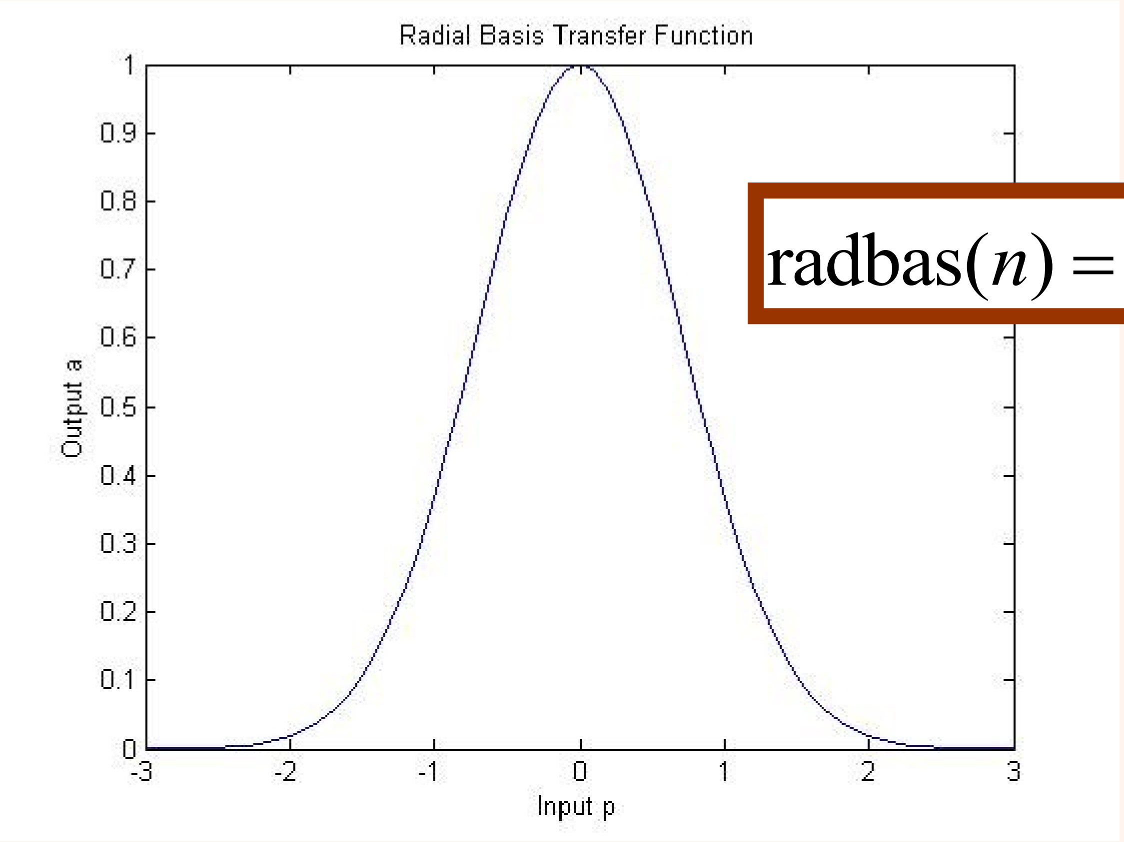
شبکه‌های RBF در Matlab





مثال-تقریب تابع(ادامه...)

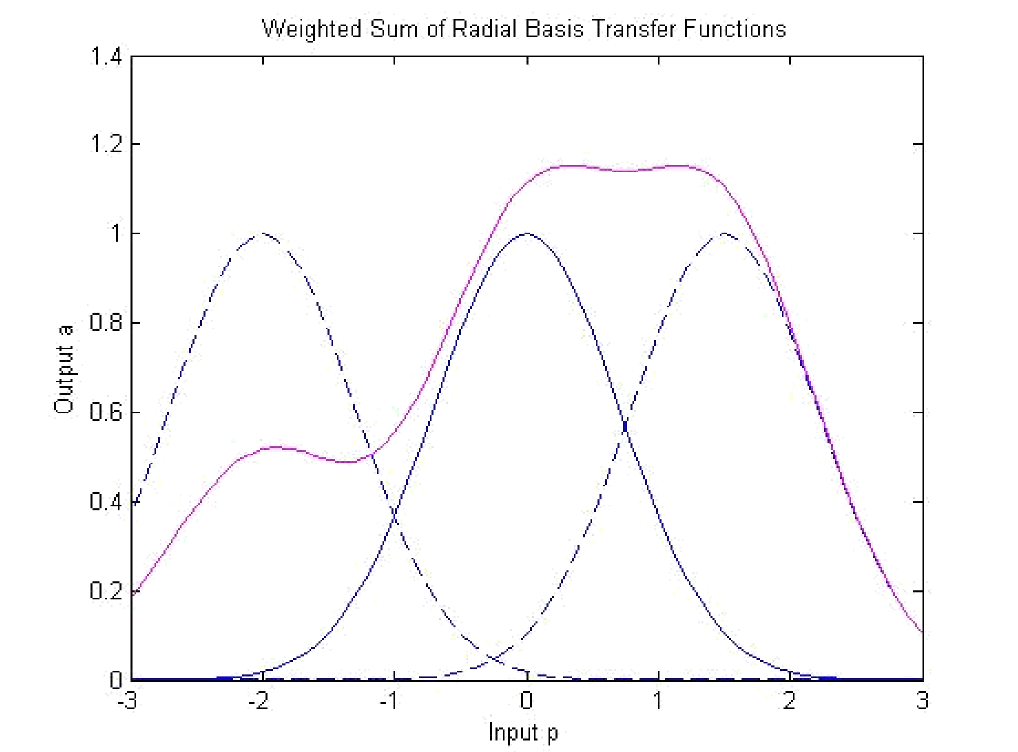
```
p = -3:.1:3;
a = radbas(p);
plot(p,a)
title('Radial Basis Transfer Function');
xlabel('Input p');
ylabel('Output a');
```



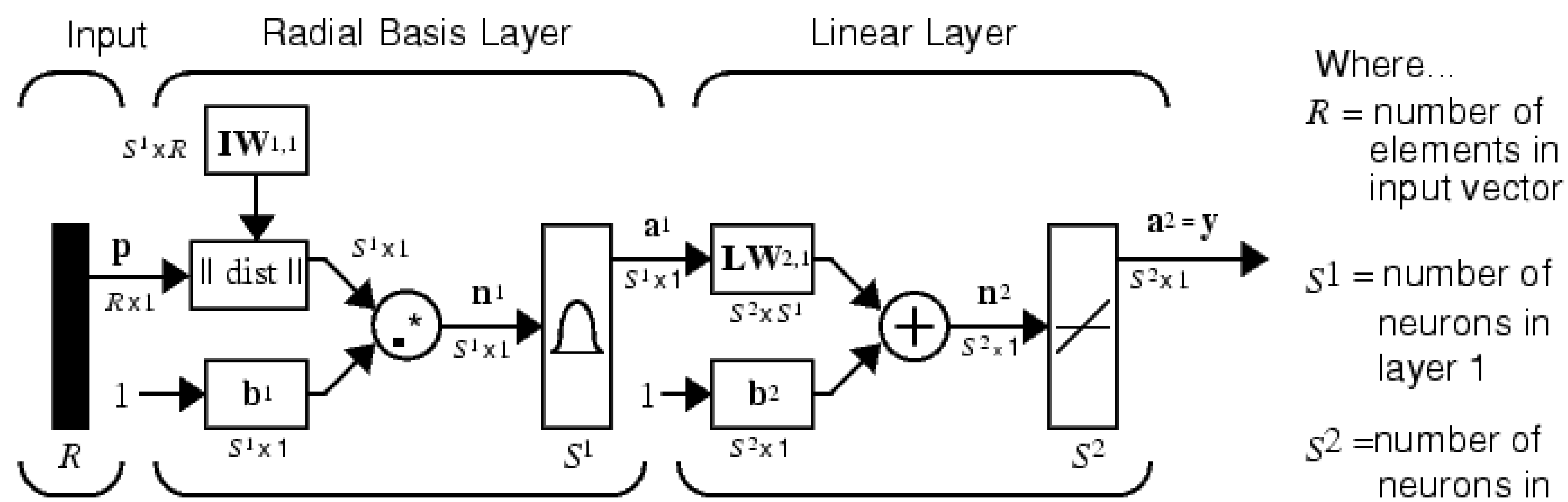
$$\mathrm{radbas}(n) = e^{-n^2}$$

مثال-تقریب تابع(ادامه...)

```
p = -3:.1:3;
a = radbas(p);
a2 = radbas(p-1.5);
a3 = radbas(p+2);
a4 = a + a2*1 + a3*0.5;
plot(p,a,'b-',p,a2,'b--',p,a3,'b--',p,a4,'m-')
title('Weighted Sum of Radial Basis Transfer Functions');
xlabel('Input p');
ylabel('Output a');
```



شبکه‌های RBF در Matlab (ادامه...)



$a_i^1 = radbas(\| {}_i\mathbf{IW}^{1,1} - \mathbf{p} \| b_i^1)$ $\mathbf{a}^2 = purelin(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2)$

a_i^1 is ith element of $\mathbf{a}^1$ where ${}_i\mathbf{IW}^{1,1}$ is a vector made of the ith row of $\mathbf{IW}^{1,1}$

ایجاد شبکه‌ی RBF– شیوه‌ی دقیق

```
net = newrbe(P,T,SPREAD)
P - input vectors.
T - target class vectors.
SPREAD - of radial basis functions, default = 1.0.
```

- در این حالت خطا برای داده‌های آموزشی صفر است.
- تعداد نرون‌های لایه‌ی مخفی برابر با داده‌های آموزشی خواهد بود.
- انتخاب SPREAD باید به دقت انجام شود.

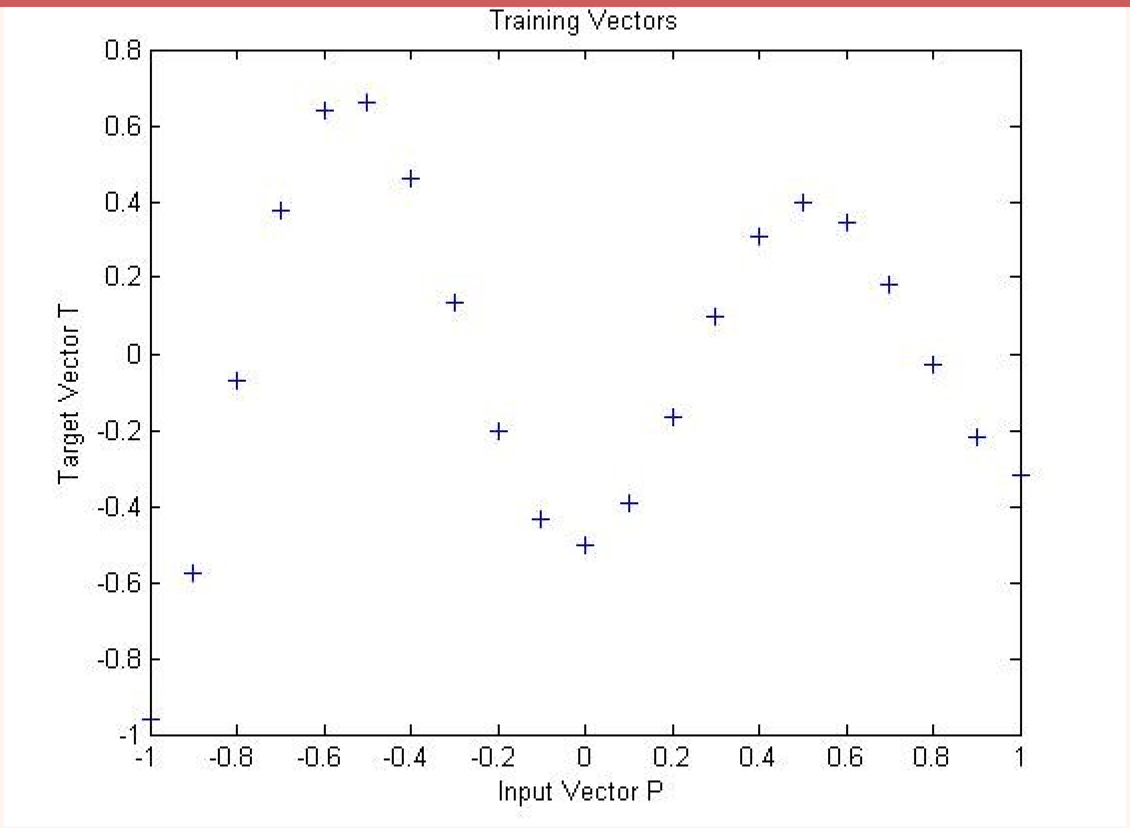
ایجاد شبکه‌ی RBF

```
[net] = newrb(P,T,goal,spread,MN,DF)
P - input vectors; R-by-Q matrix of Q input vectors
T - target class vectors.
SPREAD - of radial basis functions, default = 1.0.
GOAL-Mean squared error goal (default = 0.0)
MN- Maximum number of neurons (default is Q)
DF- Number of neurons to add between displays (default = 25)
```

- در این حالت، به تعداد نرون‌های لایه‌ی مخفی افزوده می‌شود، تا زمانی که میزان خطا از حد تعیین شده کم‌تر شود و یا تعداد نرون‌های لایه‌ی مخفی به تعداد ورودی‌ها برسد.

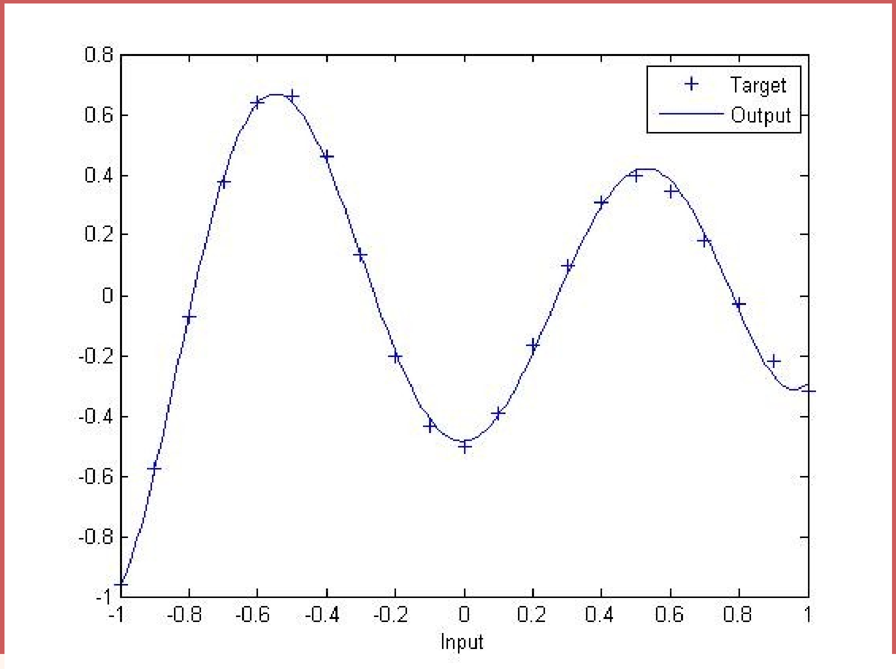
مثال-تقریب تابع

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609 ...
      .1336 -.2013 -.4344 -.5000 -.3930 -.1647  .0988 ...
      .3072  .3960  .3449  .1816 -.0312 -.2189 -.3201];
plot(P,T,'+');
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');
```

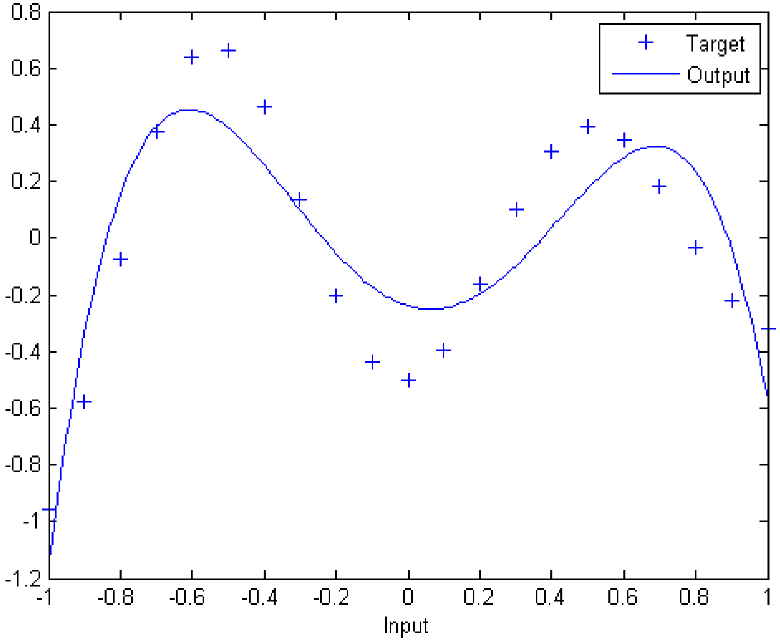


مثال-تقریب تابع(ادامه...)

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609 ...
      .1336 -.2013 -.4344 -.5000 -.3930 -.1647  .0988 ...
      .3072  .3960  .3449  .1816 -.0312 -.2189 -.3201];
eg = 0.02; % sum-squared error goal
sc = 1;    % spread constant
net = newrb(P,T,eg,sc);
plot(P,T,'+');
xlabel('Input');

X = -1:.01:1;
Y = sim(net,X);

hold on;
plot(X,Y);
hold off;
legend({'Target','Output'})
```

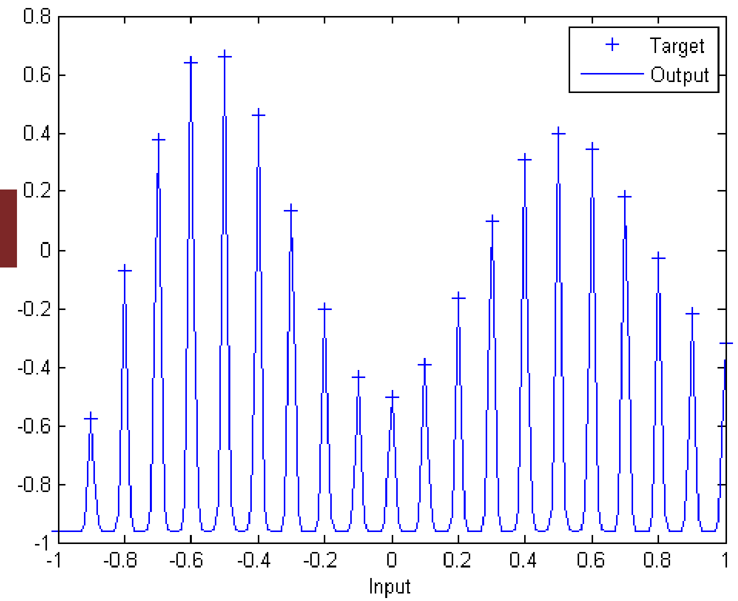


انتخاب انحراف معیار



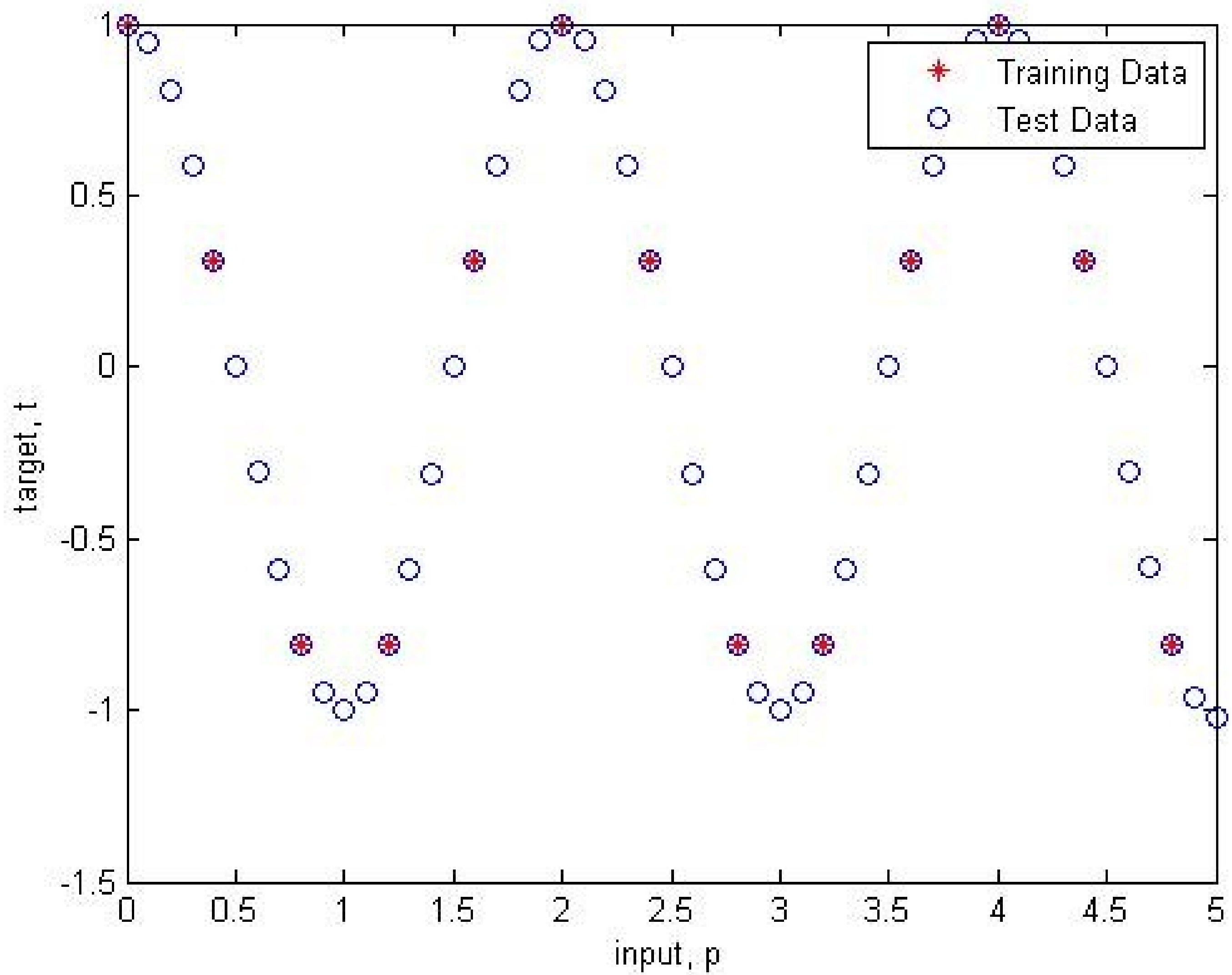
SPRED=100

SPRED=0.01

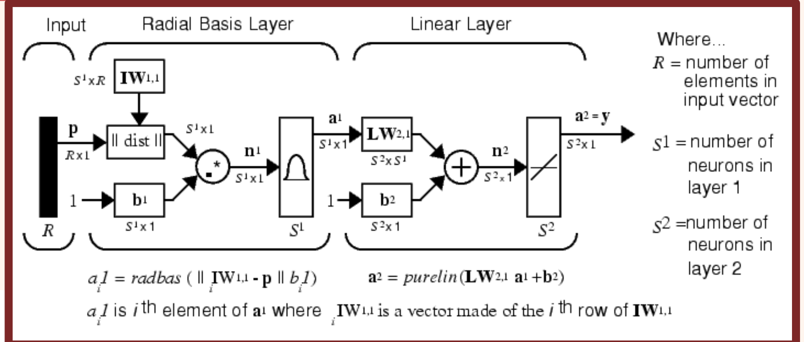


مثال

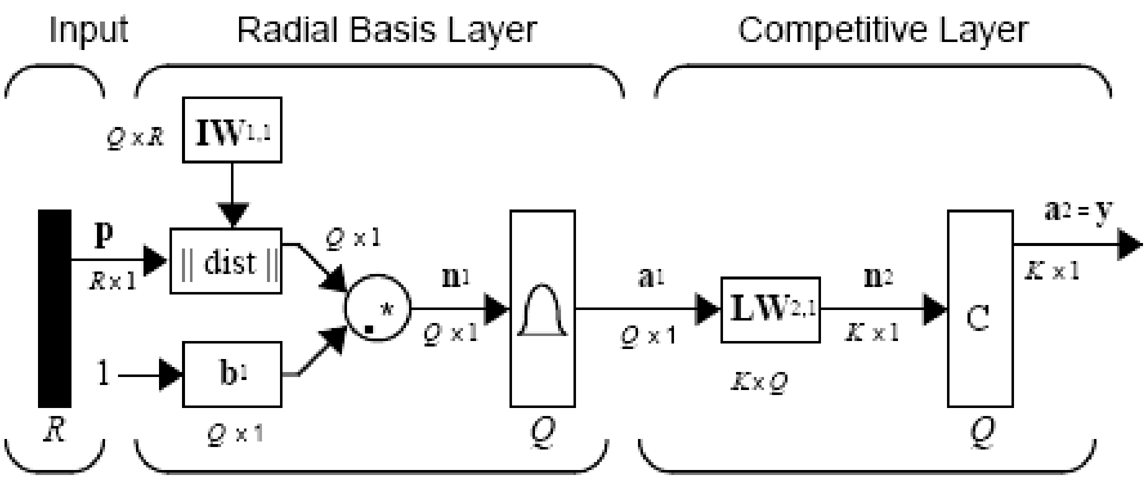
```
%generate training data (input and target)
p = [0:0.4:5];
t = cos(p*pi);
%Define and train RBF Network
net = newrb(p,t);
plot(p,t,'*r');hold;
%generate test data
p1 = [0:0.1:5];
%test network
y = sim(net,p1);
plot(p1,y,'ob');

legend('Training Data','Test Data');
xlabel('input, p');
ylabel('target, t‘)
```


Training Data
Test Data
target, t
input, p
1
0.5
0
-0.5
-1
-1.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Probabilistic Neural Networks



Probabilistic Neural Network Architecture



Where...

R = number of elements in input vector

$a_i^1 = radbas(\| {}_i\mathbf{IW}^{1,1} - \mathbf{p} \| \, b_i^1)$ $\qquad$ $\mathbf{a}^2 = compet(\mathbf{LW}^{2,1}\mathbf{a}^1)$

a_i^1 is i th element of $\mathbf{a}^1$ where ${}_i\mathbf{IW}^{1,1}$ is a vector made of the i th row of $\mathbf{IW}^{1,1}$

Q = number of input/target pairs = number of neurons in layer 1
K = number of classes of input data = number of neurons in layer 2

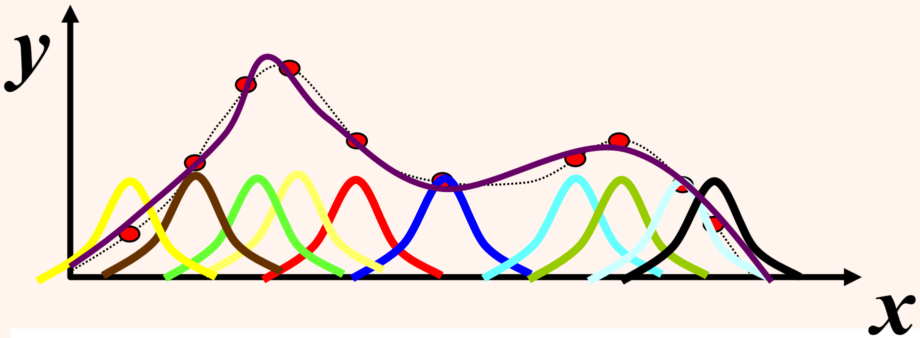
Probabilistic Neural Networks

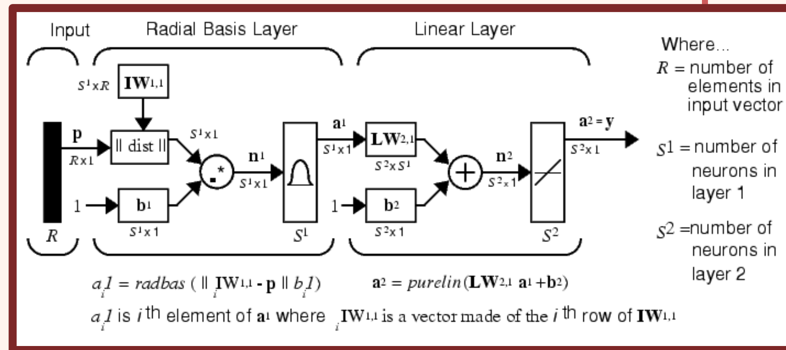
- نوعی شبکه‌ی RBF است که برای استفاده در دسته‌بندی مناسب است.
- اگر spread را نزدیک صفر در نظر بگیریم، در عمل 1-NN خواهد بود.

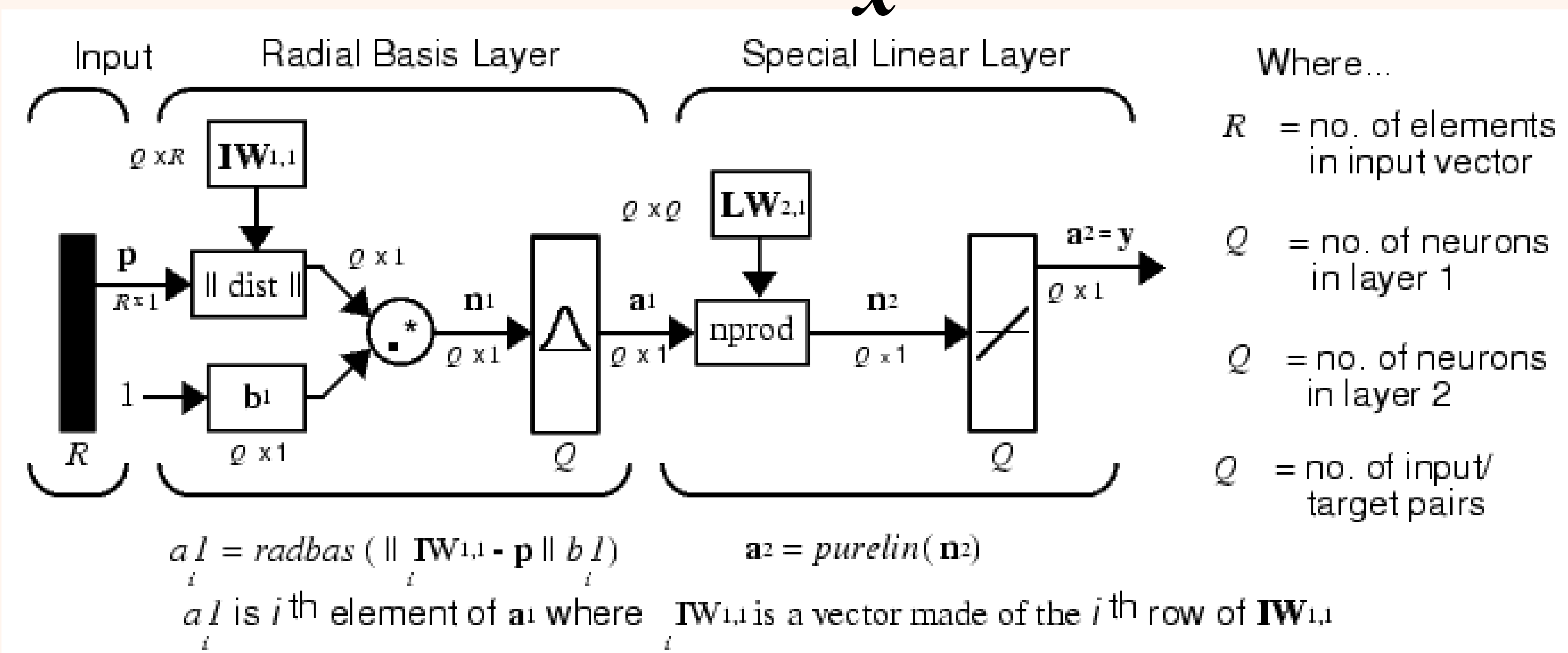
```
P = [1 2 3 4 5 6 7];
Tc = [1 2 3 2 2 3 1];
T = ind2vec(Tc)
net = newpnn(P,T,0.001);
Y = sim(net,P)
Yc = vec2ind(Y)
```

```
Yc =

    1    2    3    2    2    3    1
```

Generalized Regression Networks







Generalized Regression Networks

- نوعی شبکه‌ی RBF است که برای استفاده در رگرسیون مناسب است.
- لایه‌ی دوم این شبکه، با RBF معمولی متفاوت است.
- در این حالت، وزن متصل از لایه‌ی مخفی به خروجی متناسب با خروجی مطلوب در نظر گرفته می‌شود.
- مزیت آن در این است که به محاسبه‌ی ماتریس معکوس نیازی ندارد.