شبکه‌های عصبی مصنوعی

درس ۱۷

شبکه‌های توابع پایه‌ی شعاعی

Radial Basis Function Networks

کاظم فولادی قلعه

دانشکده مهندسی، پردیس فارابی

دانشگاه تهران

http://courses.fouladi.ir/nn

Radial Basis Networks

شبکه‌های توابع پایه‌ی شعاعی

۱

شبکه‌ی توابع پایه‌ی شعاعی

شبکه‌ی توابع پایه‌ی شعاعی

RADIAL BASIS FUNCTION NETWORK (RBF)

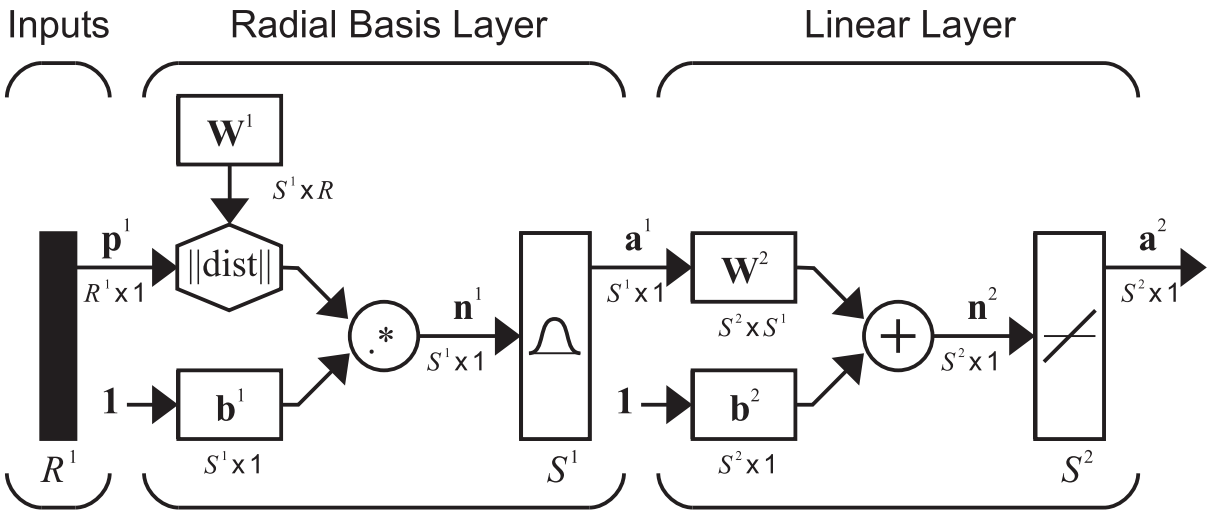
شبکه‌ی RBF: به‌عنوان یک شبکه‌ی تقریب‌زننده‌ی جهانی

دارای کاربردهای متعدد همچون MLP

مقاله‌ی اصلی RBF: پاول، 1987 ←
استفاده از توابع پایه‌ی شعاعی برای درون‌یابی دقیق در یک فضای چندبعدی

اولین شبکه‌ی RBF: برومهد، 1988 ←
تولید تابع درون‌یابی هموار:
تلاش نمی‌شود شبکه مجبور شود دقیقاً به خروجی‌های تارگت پاسخ دهد،
بلکه تاکید بر شبکه‌ای است که بتواند به‌خوبی موقعیت‌های جدید را نیز تعمیم دهد.

Radial Basis Network



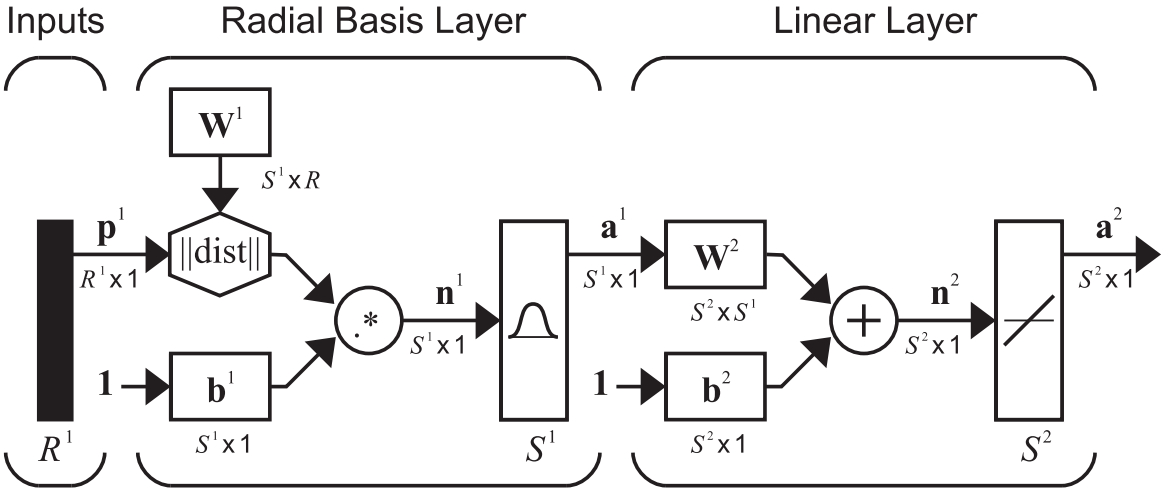
$$n_i^1 = \left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\| b_i^1 \qquad b = 1/\left(\sigma\sqrt{2}\right) \qquad a = f(n) = e^{-n^2}$$

The first layer weight vectors ${}_i\mathbf{w}^1$ are called “centers” of the basis functions.

شبکه‌ی توابع پایه‌ی شعاعی

شبکه‌ی دو لایه

RADIAL BASIS FUNCTION NETWORK (RBF)



$$n_i^1 = \left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\| b_i^1 \qquad b = 1/\left(\sigma\sqrt{2}\right) \qquad a = f(n) = e^{-n^2}$$

The first layer weight vectors ${}_i\mathbf{w}^1$ are called “centers” of the basis functions.

بردارهای وزن لایه‌ی اول ${}_i\mathbf{w}^1$، «مرکزهای» توابع پایه نامیده می‌شوند.

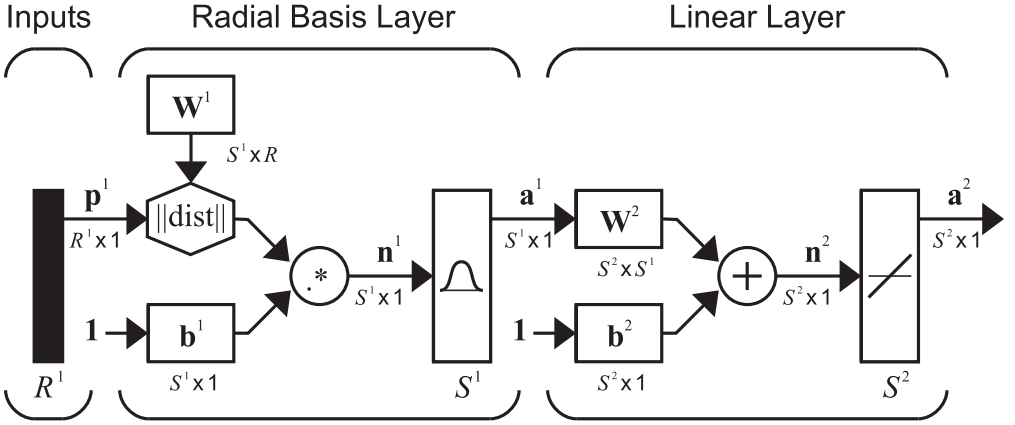
شبکه‌ی توابع پایه‌ی شعاعی

شبکه‌ی دو لایه

RADIAL BASIS FUNCTION NETWORK (RBF)

RBF یک شبکه‌ی دو لایه است؛ لایه‌ی دوم یک لایه استاندارد خطی است.
تفاوت لایه‌ی یک آن با لایه‌ی یک MLP:

۱) محاسبه‌ی فاصله‌ی ورودی با وزن مربوطه (به‌جای ضرب داخلی)
۲) ضرب کردن در بایاس (به‌جای جمع کردن با بایاس)



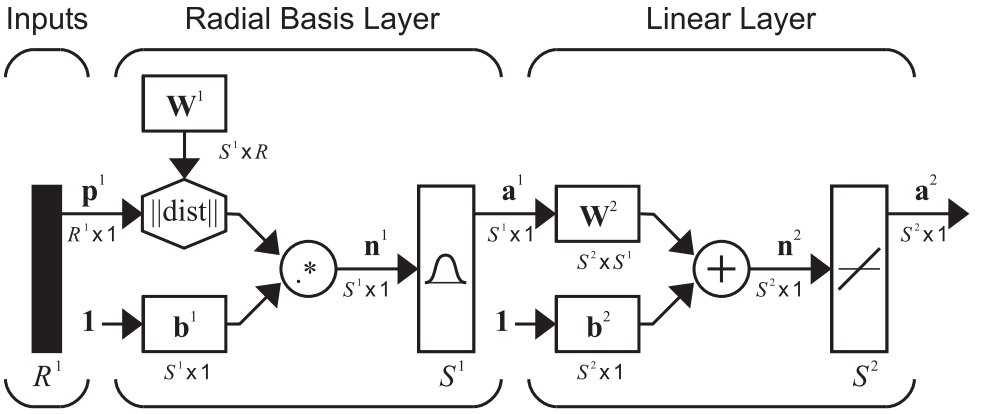
$$a^1_i = radbas(\|_i\mathbf{w}^1 - \mathbf{p}\|b^1_i) \qquad \mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2$$

$$n^1_i = \left\|\mathbf{p} - {}_i\mathbf{w}^1\right\| b^1_i \qquad b = 1/\left(\sigma\sqrt{2}\right) \qquad a = f(n) = e^{-n^2}$$

شبکه‌ی توابع پایه‌ی شعاعی

مرکز، بایاس و تابع انتقال گاوسی

RADIAL BASIS FUNCTION NETWORK (RBF)



$$a^1_i = radbas(\|_i\mathbf{w}^1 - \mathbf{p}\|b^1_i) \qquad \mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2$$

$$n^1_i = \left\|\mathbf{p} - {}_i\mathbf{w}^1\right\| b^1_i \qquad b = 1/\left(\sigma\sqrt{2}\right) \qquad a = f(n) = e^{-n^2}$$

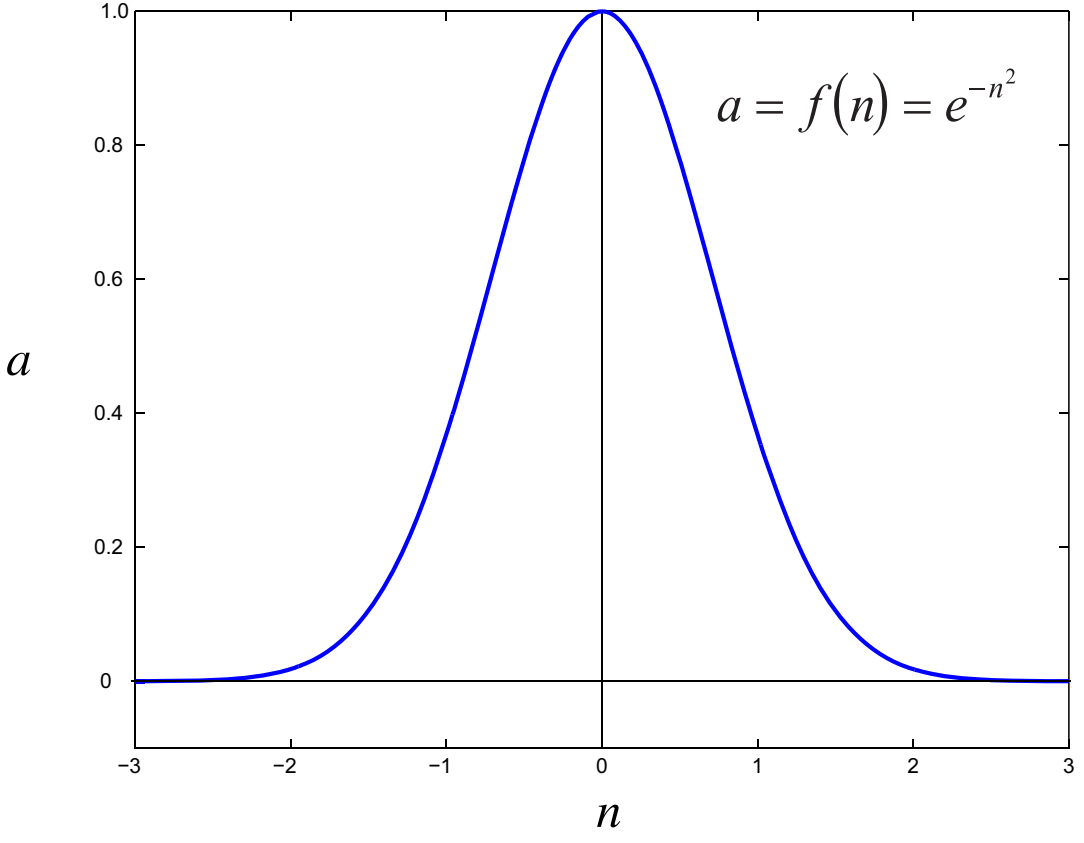
مرکز (center): ${}_i\mathbf{w}^1$،
نقطه‌ای که ورودی خالص نرون در آن صفر می‌شود.

وقتی از تابع انتقال گاوسی استفاده می‌شود، بایاس وابسته به واریانس است.

تابع انتقال گاوسی

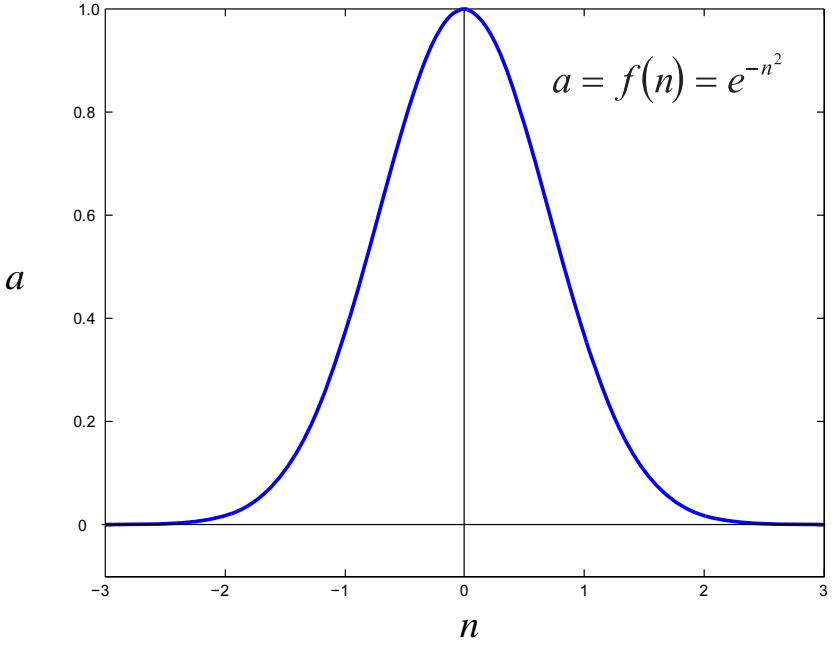
بایاس (bias): b^1_i،
میزان مقیاس روی تابع انتقال (پایه) که موجب فشردگی یا کشیدگی آن می‌شود.
[واریانس، انحراف استاندارد، ثابت پراکندگی (spread constant)]

Gaussian Transfer Function (Local)



تابع انتقال گاوسی (محلی)

GAUSSIAN TRANSFER FUNCTION (LOCAL)

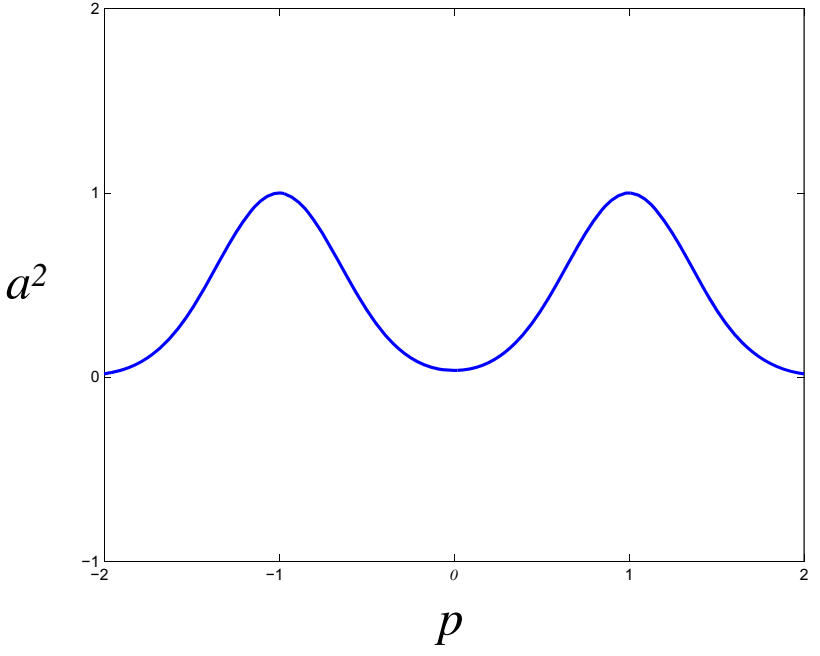


تابع انتقال گاوسی، یک تابع محلی (local) است:
خروجی به صفر نزدیک می‌شود اگر در هر جهت از مرکز دور شویم.
برخلاف **تابع سیگموئید** که یک تابع سراسری (global) است:
خروجی نزدیک یک می‌ماند هرگاه ورودی خالص به سمت بی‌نهایت برود.

Example Network Function

$$w^1_{1,1} = -1,\ w^1_{2,1} = 1,\ b^1_1 = 2,\ b^1_2 = 2$$

$$w^2_{1,1} = 1,\ w^2_{1,2} = 1,\ b^2 = 0$$



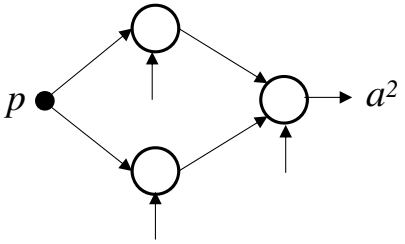
شبکه‌ی توابع پایه‌ی شعاعی

مثال از تابع شبکه: (۱ از ۲)

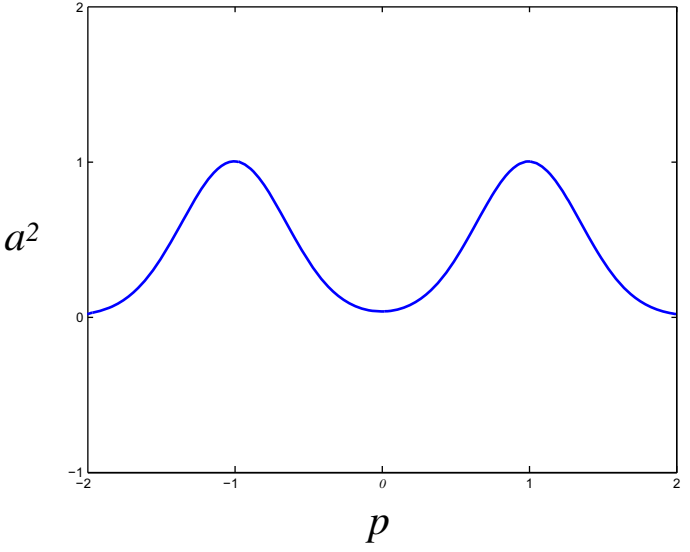
EXAMPLE NETWORK FUNCTION

$$w^1_{1,1} = -1,\ w^1_{2,1} = 1,\ b^1_1 = 2,\ b^1_2 = 2$$

$$w^2_{1,1} = 1,\ w^2_{1,2} = 1,\ b^2 = 0$$

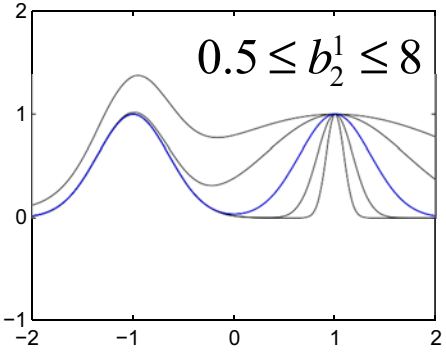


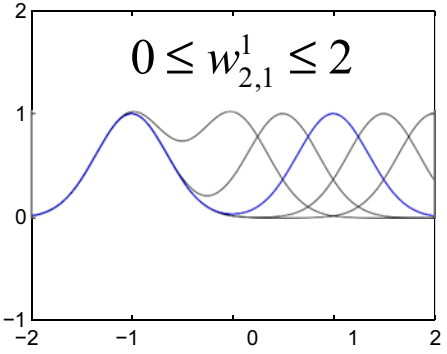
در پاسخ:
برای هر گاوسی
[هر نرون گاوسی (تابع پایه)
در لایه‌ی اول]
یک تپه داریم
⇓
با تنظیم پارامترهای شبکه،
شکل و مکان هر تپه تغییر می‌کند.

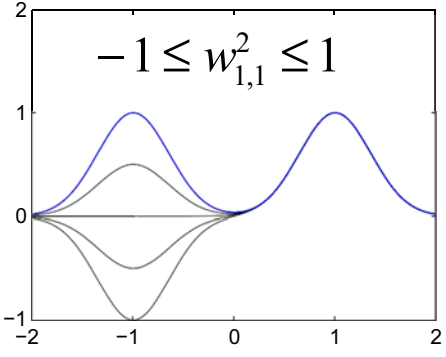


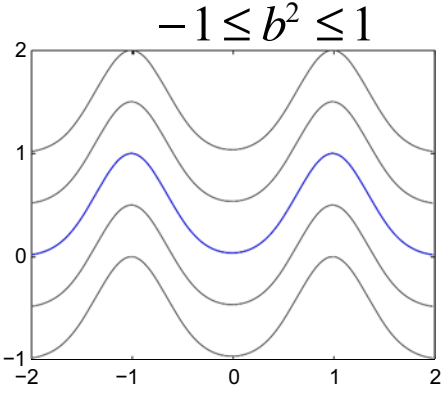
تأثیر وزن‌ها و بایاس‌های لایه‌ی یک در RBF با MLP متفاوت است:
در MLP هر تابع سیگموئید یک پله ایجاد می‌کند: وزن‌ها شیب پله را تغییر می‌دهند و بایاس‌ها مکان پله را تغییر می‌دهند.

Parameter Variations







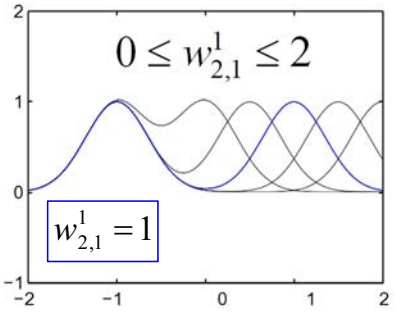


شبکه‌ی توابع پایه‌ی شعاعی

مثال از تابع شبکه: (۲ از ۲): اثر تغییرات پارامتر

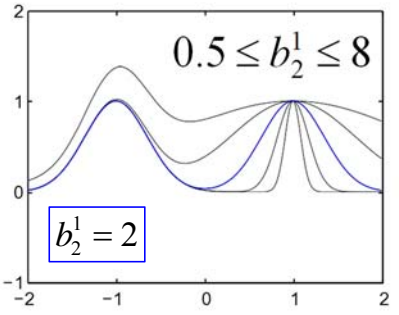
PARAMETER VARIATIONS

برای ورودی چندبعدی به مرکز هر سطر ماتریس وزن یک تپه وجود دارد **(مرکز نرون)**

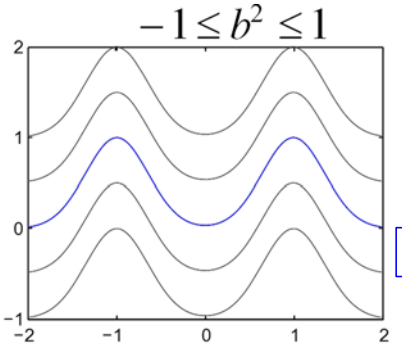


وزن لایه‌ی یک برای تعیین محل تپه‌ها: در هر وزن لایه‌ی یک، یک تپه به مرکز آن وجود دارد.

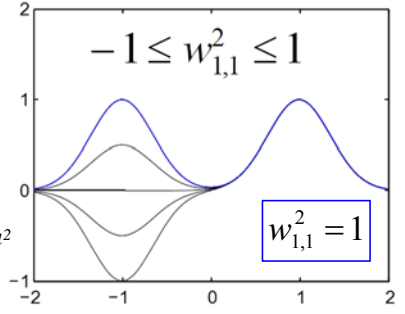
بایاس لایه‌ی یک برای تغییر عرض تپه‌ها استفاده می‌شود:
بایاس بزرگ‌تر
⇓
تپه‌ی باریک‌تر

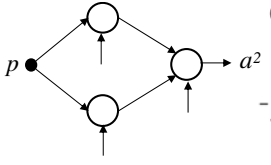


بایاس لایه‌ی دو، خروجی کل شبکه را به سمت بالا یا پایین جابه‌جا می‌کند.

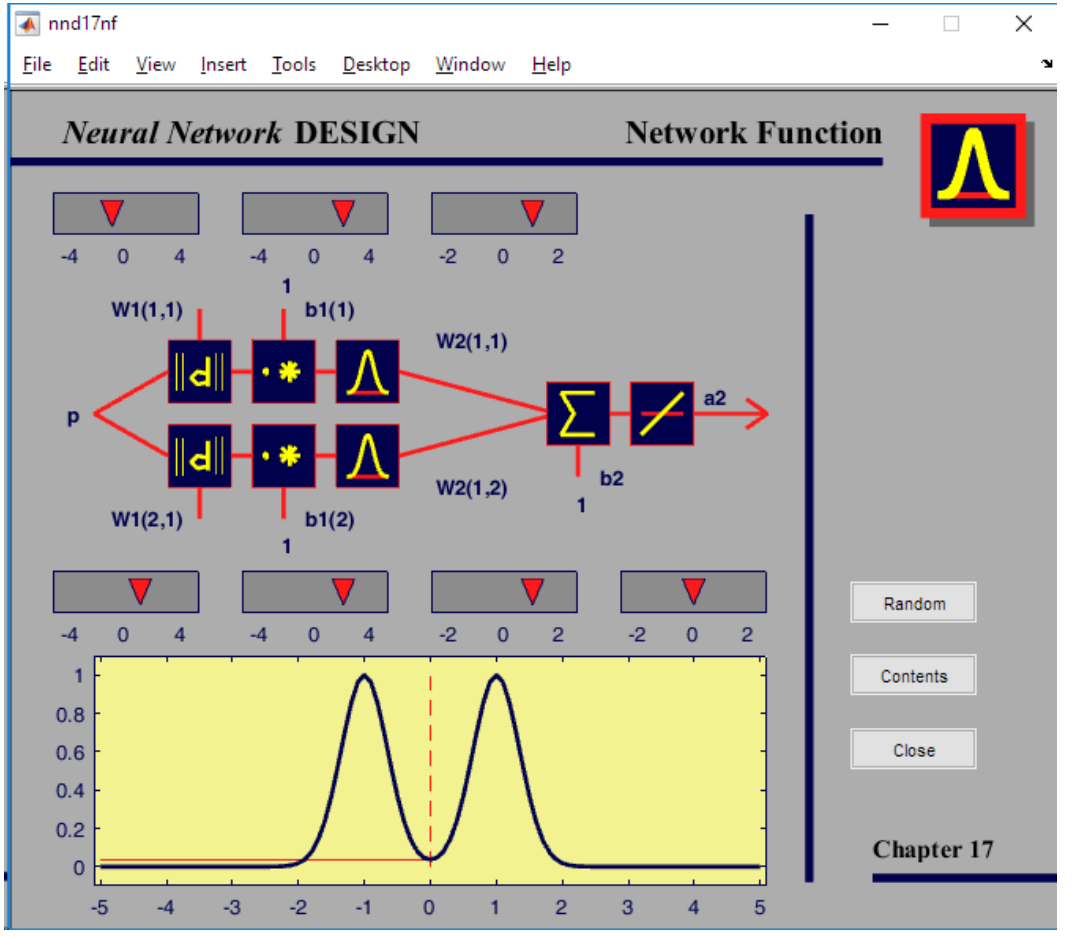


وزن‌های لایه‌ی دو، ارتفاع تپه‌ها را کم و زیاد (scale) می‌کند.





کار لایه‌ی دو در RBF مشابه لایه‌ی خطی در MLP است: ایجاد جمع وزن‌دار از خروجی نرون‌های لایه‌ی یک.


nnd17nf
File Edit View Insert Tools Desktop Window Help
Neural Network DESIGN
Network Function
W1(1,1)
b1(1)
W2(1,1)
p
a2
W2(1,2)
b2
W1(2,1)
b1(2)
Random
Contents
Close
Chapter 17

```
>> nnd17nf
```

شبکه‌ی توابع پایه‌ی شعاعی

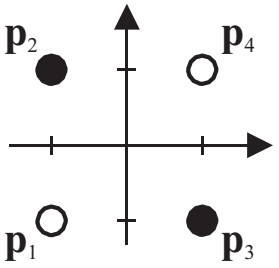
تقریب تابع

* اگر تعداد کافی نرون در لایه‌ی اول RBF داشته باشیم، می‌توانیم هر تابع دلخواهی را تقریب بزنیم (با روشی متفاوت از MLP).

برای شبکه‌ی RBF هر تابع انتقال فقط بر روی ناحیه‌ی کوچکی از فضای ورودی فعال است (پاسخ محلی دارد).
اگر ورودی از مرکز داده شده دور شود، خروجی نرون متناظر به صفر نزدیک می‌شود.

این موضوع پیامدهایی برای طراحی شبکه‌های RBF دارد:
باید تعداد مراکز کافی توزیع شده در سرتاسر محدوده‌ی ورودی‌های شبکه داشته باشیم
و باید بایاس‌ها را به‌گونه‌ای انتخاب کنیم که توابع پایه به‌شکل چشمگیری با هم همپوشانی داشته باشند.
(بایاس، عرض تابع پایه را تغییر می‌دهد.)

Pattern Recognition Problem

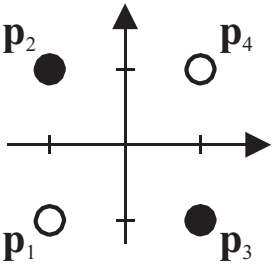


$$\text{Category 1}: \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\} \quad \text{Category 2}: \left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

مسئله‌ی بازشناسی الگو

مسئله‌ی XOR

PATTERN RECOGNITION PROBLEM



$$\text{Category 1}: \left\{\mathbf{p}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\} \quad \text{Category 2}: \left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\}$$

RBF برای این مسئله جواب‌های متعددی دارد.

یک راه‌حل:

وقتی الگوی ورودی نزدیک به $\mathbf{p}_2$ یا $\mathbf{p}_3$ می‌شود،
شبکه خروجی بزرگ‌تر از صفر تولید می‌کند و برای سایر ورودی‌ها خروجی کوچک‌تر از صفر.

مسئله: دو ورودی + یک خروجی ⇐ دو نرون در لایه‌ی اول (دو تابع پایه‌ی شعاعی) +
سطرهای ماتریس وزن لایه‌ی اول= مراکز توابع پایه ⇐ تنظیم حداکثر خروجی شبکه برای ورودی این الگوها

بایاس لایه‌ی اول: وابسته به عرض مورد نظر برای توابع پایه:
می‌خواهیم دو قله‌ی مجزا در $\mathbf{p}_2$ و $\mathbf{p}_3$ داشته باشیم، پس نمی‌خواهیم توابع با هم زیاد همپوشانی داشته باشند.

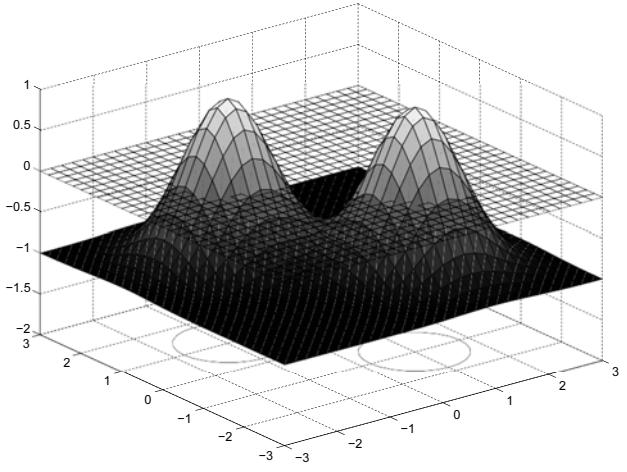
Radial Basis Solution

Choose centers at $\mathbf{p}_2$ and $\mathbf{p}_3$:

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

Choose bias to be 1:

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



This will cause the following reduction in the basis functions where they meet:

$$a = e^{-n^2} = e^{-\left(1\cdot\sqrt{2}\right)^2} = e^{-2} = 0.1353$$

Choose the second layer bias to produce negative outputs, unless we are near $\mathbf{p}_2$ and $\mathbf{p}_3$. Choose second layer weights so that output moves above 0 near $\mathbf{p}_2$ and $\mathbf{p}_3$.

$$\mathbf{W}^2 = \begin{bmatrix} 2 & 2 \end{bmatrix}, b^2 = \begin{bmatrix} -1 \end{bmatrix}$$

مسئله‌ی بازشناسی الگو

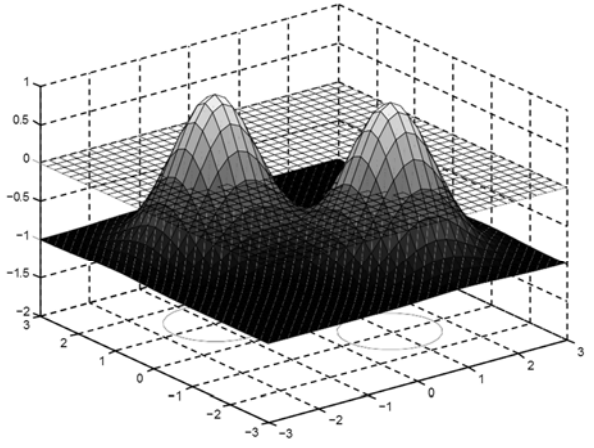
مسئله‌ی XOR: راه‌حل با پایه‌های شعاعی

RADIAL BASIS SOLUTION

۱) مرکزها را بر روی $\mathbf{p}_2$ و $\mathbf{p}_3$ انتخاب می‌کنیم:

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_2 \\ \mathbf{p}_3^T \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

فاصله‌ی مراکز از مبدأ = $\sqrt{2}$



۲) بایاس را برابر با ۱ انتخاب می‌کنیم:

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

۳) این باعث می‌شود که این مقدار کاهش در توابع پایه اتفاق بیفتد:

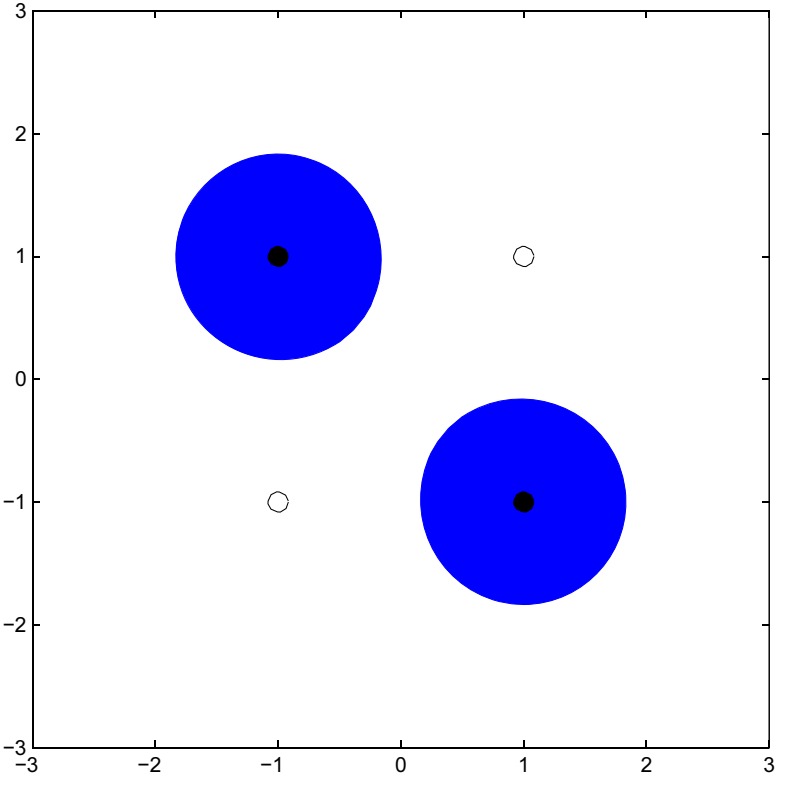
$$a = e^{-n^2} = e^{-(1\cdot\sqrt{2})^2} = e^{-2} = 0.1353$$

۴) بایاس لایه‌ی دو به‌گونه‌ای انتخاب می‌شود که خروجی‌ها منفی شوند، مگر اینکه ورودی نزدیک به $\mathbf{p}_2$ یا $\mathbf{p}_3$ باشد.

۵) وزن لایه‌ی دو به‌گونه‌ای انتخاب می‌شود که نزدیک $\mathbf{p}_2$ و $\mathbf{p}_3$ خروجی به بالای صفر حرکت کند:

$$\mathbf{W}^2 = [2 \quad 2], b^2 = [-1]$$

Final Decision Regions



مسئله‌ی بازشناسی الگو

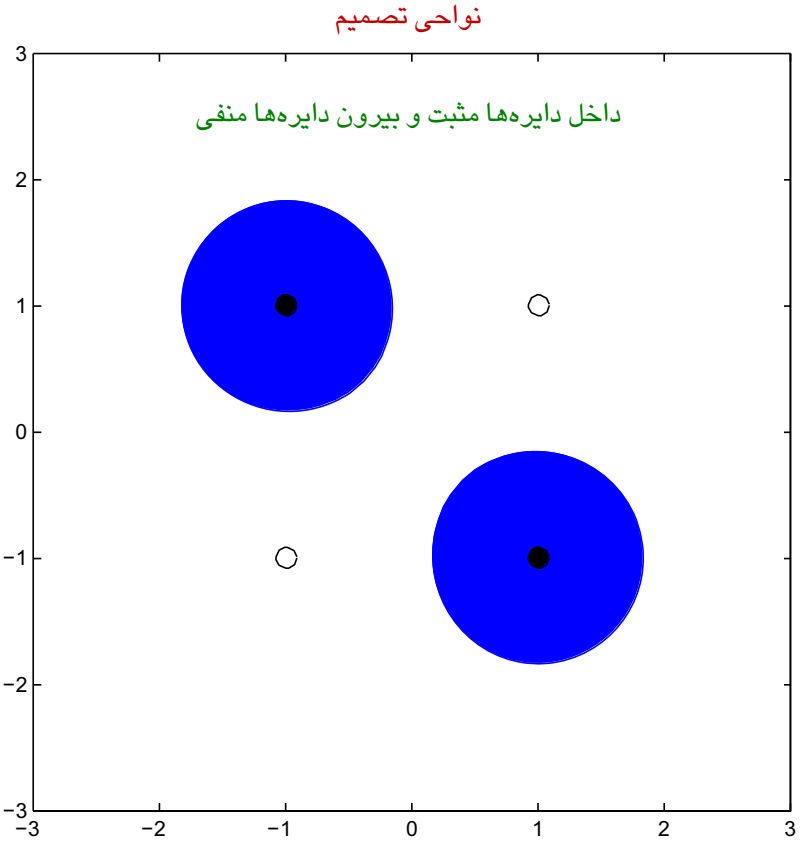
مسئله‌ی XOR: راه‌حل با پایه‌های شعاعی: نواحی تصمیم نهایی

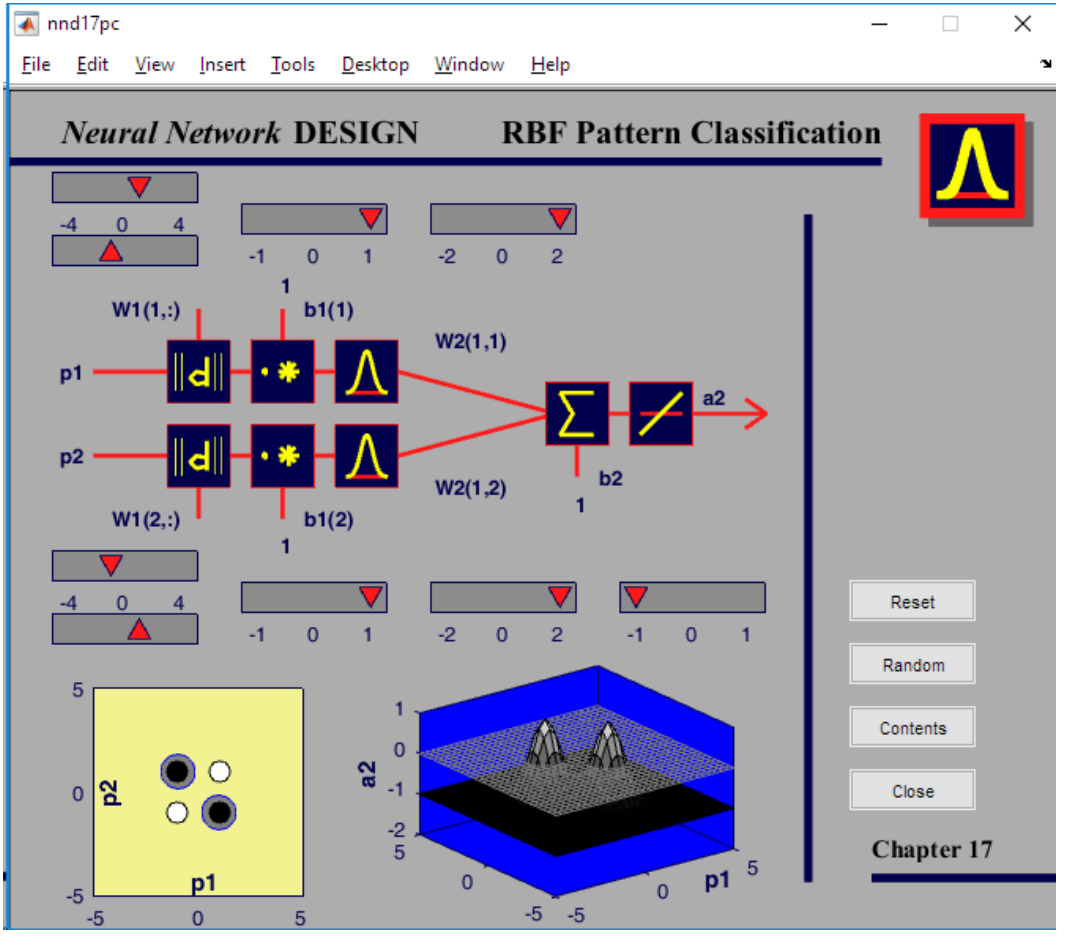
FINAL DECISION REGIONS

این راه‌حل بهترین راه‌حل نیست:
زیرا همیشه الگوهای ورودی را به‌نزدیک‌ترین بردار پروتوتایپ نسبت نمی‌دهد (برخلاف MLP).

- **مرزهای تصمیم** در RBF دایره‌ای است.
- **مرزهای تصمیم** در پرسپترون خطی است.
- **نواحی تصمیم** در RBF ترکیبی از نواحی دایره‌ای است.
- **نواحی تصمیم** در MLP ترکیبی از چند خط است.

وقتی تعداد زیادی نرون استفاده می‌شود (که مراکز نزدیک به هم دارند)،
مرزهای ابتدایی RBF دیگر دایره‌ای خالص نیست
(مرزهای ابتدایی MLP هم دیگر خطی خالص نیست).
با این وجود:
نسبت دادن مرزهای دایره‌ای به شبکه‌های RBF و مرزهای خطی به شبکه‌های MLP
در درک عملیات آنها به‌عنوان طبقه‌بندی کننده‌ی الگو کمک می‌کند.





MATLAB

```
>> nnd17pc
```

Global Versus Local

- Multilayer networks create a distributed representation.
  - All sigmoid or linear transfer functions overlap in their activity.
- Radial basis networks create local representations.
  - Each basis function is only active over a small region.
- The global approach requires fewer neurons. The local approach is susceptible to the "curse of dimensionality."
- The local approach leads to faster training and is suitable for adaptive methods.

«سراسری» در برابر «محلی»

مزایا و معایب توابع انتقال محلی در RBF نسبت به توابع انتقال سراسری در MLP

GLOBAL VERSUS LOCAL

- Multilayer networks create a distributed representation.
  - All sigmoid or linear transfer functions overlap in their activity.

- شبکه‌های MLP یک بازنمایی توزیع‌شده ایجاد می‌کنند.
  - همه‌ی توابع انتقال سیگموئید یا خطی در فعالیت‌های خود دارای همپوشانی هستند (نرون‌های متعددی فعال هستند و خروجی قابل توجه دارند).

- Radial basis networks create local representations.
  - Each basis function is only active over a small region.

- شبکه‌های RBF یک بازنمایی محلی ایجاد می‌کنند.
  - هر تابع پایه فقط بر روی ناحیه‌ی محدودی از ورودی فعال است (نرون‌های محدودی فعال هستند).

- The global approach requires fewer neurons.
  The local approach is susceptible to the “curse of dimensionality.”

- روی‌کرد سراسری تمایل به تعداد نرون کمتری در لایه‌ی پنهان دارد (زیرا هر نرون در پاسخ به بخش بزرگی از فضای ورودی نقش دارد).
  روی‌کرد محلی مشکوک به «بیچارگی بُعدیت» است (زیرا برای ایجاد یک تقریب دقیق، مراکز پایه‌ها باید در سرتاسر فضای ورودی پراکنده شوند. وقتی تعداد نرون‌ها زیاد باشد، پارامترهای بیشتری داریم و احتمال بیش‌برازش شبکه بر روی داده‌های آموزشی و شکست در تعمیم بیشتر می‌شود).

- The local approach leads to faster training and is suitable for adaptive methods.

- روی‌کرد محلی منجر به آموزش سریع‌تر می‌شود و برای روش‌های تطبیقی مناسب است (به‌خصوص الگوریتم‌های دومرحله‌ای).

«سراسری» در برابر «محلی»

مزایا و معایب توابع انتقال محلی در RBF نسبت به توابع انتقال سراسری در MLP : ملاحظات آموزش

GLOBAL VERSUS LOCAL

- اگر در یک بازه‌ی زمانی، داده‌های آموزشی تنها در ناحیه‌ی خاصی از فضای ورودی ظاهر شوند، آن‌گاه یک بازنمایی سراسری تمایل دارد دقت آن را در آن نواحی با هزینه کردن آن بازنمایی در سایر نواحی بهبود دهد. بازنمایی‌های محلی این مشکل را با همان گستردگی نخواهند داشت.

- از آنجا که هر نرون تنها در ناحیه‌ی کوچکی از فضای ورودی فعال است، وزن‌های آن تنظیم نمی‌شود، اگر ورودی خارج از آن ناحیه بیفتد.

شبکه‌های عصبی مصنوعی

شبکه‌های توابع پایه‌ی شعاعی

۲

آموزش شبکه‌های RBF

آموزش شبکه‌های RBF

RADIAL BASIS TRAINING

آموزش RBF با روش‌های متعددی (علاوه بر روش‌های مبتنی بر گرادیان) ممکن است.

- به‌دلیل طبیعت محلی تابع انتقال و عملکرد وزن‌ها و بایاس‌های لایه‌ی یک، در اینجا تمایل وجود دارد که می‌نیمم‌های محلی ارضانشدنی بسیاری در رویه‌های خطای RBF (نسبت به MLP) وجود داشته باشد.
- به همین دلیل، الگوریتم‌های مبتنی بر گرادیان برای آموزش کامل شبکه‌های RBF راضی‌کننده نیستند.
- الگوریتم‌های مبتنی بر گرادیان بنابر موقعیت، برای تنظیم دقیق (fine-tune) کردن شبکه پس از آنکه ابتدائاً توسط سایر روش آموزش داده شد، استفاده می‌شود.

متداول‌ترین الگوریتم آموزش RBF از دو مرحله تشکیل می‌شود
و در هر مرحله با یک لایه به‌طور جداگانه کار می‌شود.

Radial Basis Training

- Radial basis network training generally consists of two stages.
- During the first stage, the weights and biases in the first layer are set. This can involve unsupervised training or even random selection of the weights.
- The weights and biases in the second layer are found during the second stage. This usually involves linear least squares, or LMS for adaptive training.
- Backpropagation (gradient-based) algorithms can also be used for radial basis networks.

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای

RADIAL BASIS TRAINING

متداول‌ترین الگوریتم آموزش RBF از دو مرحله تشکیل می‌شود
و در هر مرحله با یک لایه به‌طور جداگانه کار می‌شود:

- در **مرحله‌ی اول**، وزن‌ها و بایاس‌ها در لایه‌ی اول تنظیم می‌شوند:
  - این کار می‌تواند با آموزش بدون نظارت یا حتی انتخاب تصادفی وزن‌ها انجام شود.
- در **مرحله‌ی دوم**، وزن‌ها و بایاس‌ها در مرحله‌ی دوم تعیین می‌شوند:
  - این کارشامل روش حداقل مربعات خطی یا LMS برای آموزش وفقی است.

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای: ساده‌ترین الگوریتم

RADIAL BASIS TRAINING

وزن‌های لایه‌ی یک (مراکز) در یک الگوی توری (grid) در سرتاسر بازه‌ی ورودی چینش می‌شوند و یک بایاس ثابت انتخاب می‌شود تا توابع پایه درجه‌ای از همپوشانی را داشته باشند.

- این روال بهینه نیست، زیرا دقیق‌ترین روش تقریب کارآمد، تعداد توابع پایه‌ی بیشتری را در ناحیه‌هایی که تابع مورد تقریب در آن پیچیده‌تر است، قرار می‌دهد.
- به‌علاوه در بسیاری از مسائل کاربردی، کل ناحیه‌ی فضای ورودی استفاده نمی‌شود و در نتیجه، بسیاری از توابع پایه هدر می‌روند.
- وقتی مراکز بر روی یک توری انتخاب می‌شوند، از «بیچارگی بُعدیت» رنج می‌برند: یعنی با افزایش بعد فضای ورودی، تعداد توابع پایه‌ی لازم به‌طور هندسی زیاد می‌شود (برای $d = 1$، n نرون، برای $d = 2$، n^2 نرون و ...).

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای: الگوریتم ۲

RADIAL BASIS TRAINING

وزن‌های لایه‌ی یک (مراکز) به‌صورت تصادفی در قالب یک زیرمجموعه از بردارهای ورودی در مجموعه‌ی آموزشی انتخاب می‌شوند
و یک بایاس ثابت انتخاب می‌شود تا توابع پایه درجه‌ای از همپوشانی را داشته باشند.

- این کار تضمین می‌کند که مراکز پایه‌ها در ناحیه‌هایی قرار گیرند که برای شبکه مفید است.
- اما به دلیل تصادفی بودن انتخاب، این روال بهینه نیست.

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای: الگوریتم ۳

RADIAL BASIS TRAINING

وزن‌های لایه‌ی یک (مراکز) با آموزش بدون ناظر تعیین می‌شوند:
از روشی شبیه به لایه‌ی رقابتی کوهونن برای خوشه‌بندی فضای ورودی استفاده می‌کنیم
و مراکز خوشه‌ها را به‌عنوان مراکز توابع پایه در نظر می‌گیریم؛
و یک بایاس ثابت انتخاب می‌شود تا توابع پایه درجه‌ای از همپوشانی را داشته باشند.

- این کار تضمین می‌کند که مراکز پایه‌ها در ناحیه‌هایی با فعالیت قابل توجه قرار گیرند.

Assume Fixed First Layer

We begin with the case where the first layer weights (centers) are fixed. Assume they are set on a grid, or randomly set. For random weights, the bias can be

$$b_i^1 = \frac{\sqrt{S^1}}{d_{\max}}$$

The training data is given by

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

With first layer weights and biases fixed, the first layer output can be computed:

$$n_{i,q}^1 = \left\| p_q - {}_i w^1 \right\| b_i^1 \qquad \mathbf{a}_q^1 = \mathbf{radbas}\left(\mathbf{n}_q^1\right)$$

This provides a training set for the second layer:

$$\{\mathbf{a}_1^1, \mathbf{t}_1\}, \{\mathbf{a}_2^1, \mathbf{t}_2\}, \ldots, \{\mathbf{a}_Q^1, \mathbf{t}_Q\}$$

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای: با فرض ثابت بودن لایه‌ی یک (مرحله‌ی اول)

ASSUME FIXED FIRST LAYER

با حالتی شروع می‌کنیم که در آن وزن‌های لایه‌ی اول (مرکزها) ثابت هستند.
فرض می‌کنیم مرکزها بر روی یک توری قرار گرفته‌اند یا به‌صورت تصادفی تنظیم شده باشد.
برای وزن‌های تصادفی، بایاس می‌تواند به‌صورت زیر باشد:

$d_{\max}$: حداکثر فاصله بین مراکز مجاور
S^1: تعداد نرون لایه‌ی اول

$$b_i^1 = \frac{\sqrt{S^1}}{d_{\max}}$$

برای اطمینان از درجه‌ی دلخواه همپوشانی میان توابع پایه

مجموعه داده‌های آموزشی به‌صورت زیر داده می‌شود:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

با ثابت بودن وزن‌ها و بایاس‌های لایه‌ی یک،
خروجی لایه‌ی یک می‌تواند به‌صورت زیر محاسبه شود:

$$n_{i,q}^1 = \left\| p_q - {}_i w^1 \right\| b_i^1 \qquad \mathbf{a}_q^1 = \mathbf{radbas}\left(\mathbf{n}_q^1\right)$$

که این یک مجموعه‌ی آموزشی برای لایه‌ی دو فراهم می‌کند:

$$\{\mathbf{a}_1^1, \mathbf{t}_1\}, \{\mathbf{a}_2^1, \mathbf{t}_2\}, \ldots, \{\mathbf{a}_Q^1, \mathbf{t}_Q\}$$

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای: با فرض ثابت بودن لایه‌ی یک (مرحله‌ی دوم)

ASSUME FIXED FIRST LAYER

روال نهایی:
حداقل مربعات متعامد

بر اساس یک روش عمومی برای ایجاد مدل‌های خطی به‌نام «انتخاب زیرمجموعه»

این روش با تعداد زیادی از مراکز ممکن (معمولاً همه‌ی بردارهای ورودی از داده‌های آموزشی)، شروع می‌کند.
در هر مرحله از روال، یک مرکز را انتخاب می‌کند تا به وزن لایه‌ی یک اضافه کند.
این انتخاب بر این اساس است که نرون جدید چه میزان مجموع مربعات خطا را کاهش می‌دهد.
نرون‌ها اضافه می‌شوند تا اینکه برخی ضوابط برآورده شوند
(مثلاً ضابطه‌ی ماکزیمم‌کردن تعمیم‌پذیری شبکه).

Linear Least Squares (2nd Layer)

$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2 \qquad F(\mathbf{x}) = \sum_{q=1}^{Q}\left(\mathbf{t}_q - \mathbf{a}_q^2\right)^T\left(\mathbf{t}_q - \mathbf{a}_q^2\right)$$

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w}^2 \\ b^2 \end{bmatrix} \qquad \mathbf{z}_q = \begin{bmatrix} \mathbf{a}_q^1 \\ 1 \end{bmatrix}$$

$$a_q^2 = \left({}_1w^2\right)^T a_q^1 + b^2 = \mathbf{x}^T\mathbf{z}_q$$

$$F(\mathbf{x}) = \sum_{q=1}^{Q}\left(\mathbf{t}_q - \mathbf{x}^T\mathbf{z}_q\right)^T\left(\mathbf{t}_q - \mathbf{x}^T\mathbf{z}_q\right)$$

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای: با فرض ثابت بودن لایه‌ی یک (مرحله‌ی دوم): حداقل مربعات خطی (لایه‌ی دو)

LINEAR LEAST SQUARES (2ND LAYER)

$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2 \qquad F(\mathbf{x}) = \sum_{q=1}^{Q}\left(\mathbf{t}_q - \mathbf{a}_q^2\right)^T\left(\mathbf{t}_q - \mathbf{a}_q^2\right)$$

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w}^2 \\ b^2 \end{bmatrix} \qquad \mathbf{z}_q = \begin{bmatrix} \mathbf{a}_q^1 \\ 1 \end{bmatrix}$$

$$a_q^2 = \left({}_1\mathbf{w}^2\right)^T a_q^1 + b^2 = \mathbf{x}^T\mathbf{z}_q$$

$$F(\mathbf{x}) = \sum_{q=1}^{Q}\left(\mathbf{t}_q - \mathbf{x}^T\mathbf{z}_q\right)^T\left(\mathbf{t}_q - \mathbf{x}^T\mathbf{z}_q\right)$$

Matrix Form

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_Q \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} {}_1\mathbf{u}^T \\ {}_2\mathbf{u}^T \\ \vdots \\ {}_Q\mathbf{u}^T \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_Q^T \end{bmatrix} \qquad \mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_Q \end{bmatrix}$$

$$\mathbf{e} = \mathbf{t} - \mathbf{U}\mathbf{x} \qquad F(\mathbf{x}) = (\mathbf{t} - \mathbf{U}\mathbf{x})^T(\mathbf{t} - \mathbf{U}\mathbf{x})$$

$$F(\mathbf{x}) = (\mathbf{t} - \mathbf{U}\mathbf{x})^T(\mathbf{t} - \mathbf{U}\mathbf{x}) + \sum_{i=1}^{n} x_i^2 = (\mathbf{t} - \mathbf{U}\mathbf{x})^T(\mathbf{t} - \mathbf{U}\mathbf{x}) + \rho\mathbf{x}^T\mathbf{x}$$

$$= \mathbf{t}^T\mathbf{t} - 2\mathbf{t}^T\mathbf{U}\mathbf{x} + \mathbf{x}^T\mathbf{U}^T\mathbf{U}\mathbf{x} + \rho\mathbf{x}^T\mathbf{x}$$

$$= \mathbf{t}^T\mathbf{t} - 2\mathbf{t}^T\mathbf{U}\mathbf{x} + \mathbf{x}^T\left[\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}\right]\mathbf{x}$$

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای: با فرض ثابت بودن لایه‌ی یک (مرحله‌ی دوم): حداقل مربعات خطی (لایه‌ی دو): فرم ماتریسی

MATRIX FORM

$$\mathbf{t}=\begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_Q \end{bmatrix} \qquad \mathbf{U}=\begin{bmatrix} {}_1\mathbf{u}^T \\ {}_2\mathbf{u}^T \\ \vdots \\ {}_Q\mathbf{u}^T \end{bmatrix}=\begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_Q^T \end{bmatrix} \qquad \mathbf{e}=\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_Q \end{bmatrix}$$

$$\mathbf{e}=\mathbf{t}-\mathbf{U}\mathbf{x} \qquad F(\mathbf{x})=(\mathbf{t}-\mathbf{U}\mathbf{x})^T(\mathbf{t}-\mathbf{U}\mathbf{x})$$

برای رگولاریزاسیون به‌منظور جلوگیری از بیش‌برازش

$$F(\mathbf{x})=(\mathbf{t}-\mathbf{U}\mathbf{x})^T(\mathbf{t}-\mathbf{U}\mathbf{x})+\sum_{i=1}^{n}x_i^2=(\mathbf{t}-\mathbf{U}\mathbf{x})^T(\mathbf{t}-\mathbf{U}\mathbf{x})+\rho\mathbf{x}^T\mathbf{x}$$

$$=\mathbf{t}^T\mathbf{t}-2\mathbf{t}^T\mathbf{U}\mathbf{x}+\mathbf{x}^T\mathbf{U}^T\mathbf{U}\mathbf{x}+\rho\mathbf{x}^T\mathbf{x}$$

$$=\mathbf{t}^T\mathbf{t}-2\mathbf{t}^T\mathbf{U}\mathbf{x}+\mathbf{x}^T\left[\mathbf{U}^T\mathbf{U}+\rho\mathbf{I}\right]\mathbf{x}$$

Linear Least Squares Solution

$$F(\mathbf{x}) = \mathbf{t}^T\mathbf{t} - 2\mathbf{t}^T\mathbf{U}\mathbf{x} + \mathbf{x}^T\left[\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}\right]\mathbf{x}$$

$$= c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} \qquad \text{(Quadratic Function)}$$

$$\nabla F(\mathbf{x}) = \nabla\left(c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}\right) = \mathbf{d} + \mathbf{A}\mathbf{x}$$

$$= -2\mathbf{U}^T\mathbf{t} + 2\left[\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}\right]\mathbf{x} = 0$$

$$\left[\mathbf{U}^T\mathbf{U} + \ \mathbf{I}\right]\mathbf{x}^* = \mathbf{U}^T\mathbf{t}$$

آموزش شبکه‌های RBF

الگوریتم دومرحله‌ای: با فرض ثابت بودن لایه‌ی یک (مرحله‌ی دوم): حداقل مربعات خطی (لایه‌ی دو): راه‌حل

LINEAR LEAST SQUARES SOLUTION

$$F(\mathbf{x}) = \mathbf{t}^T\mathbf{t} - 2\mathbf{t}^T\mathbf{U}\mathbf{x} + \mathbf{x}^T\left[\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}\right]\mathbf{x}$$

$$= c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} \qquad \text{(Quadratic Function)}$$

$$\nabla F(\mathbf{x}) = \nabla\left(c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}\right) = \mathbf{d} + \mathbf{A}\mathbf{x}$$

$$= -2\mathbf{U}^T\mathbf{t} + 2\left[\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}\right]\mathbf{x} = 0$$

← یافتن نقاط ایستان

$$\left[\mathbf{U}^T\mathbf{U} + \quad \mathbf{I}\right]\mathbf{x}^* = \mathbf{U}^T\mathbf{t}$$

$$\mathbf{x}^* = [\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}]^{-1}\mathbf{U}^T\mathbf{t}$$

← اگر هسی معین مثبت باشد

Example (1)

$$g(p) = 1 + \sin\left(\frac{\pi}{4} p\right) \text{for} -2 \le p \le 2$$

$$p = \{-2, -1.2, -0.4, 0.4, 1.2, 2\}$$

$$t = \{0, 0.19, 0.69, 1.3, 1.8, 2\}$$

$$\mathbf{W}^1 = \begin{bmatrix} -2 \\ 0 \\ 2 \end{bmatrix}, \mathbf{b}^1 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

آموزش شبکه‌های RBF

مثال (۱ از ۴)

EXAMPLE

یک RBF با ۳ نرون در لایه‌ی یک برای تقریب زدن تابع g:

$$g(p) = 1 + \sin\left(\frac{\pi}{4} p\right) \text{for} -2 \le p \le 2$$

۶ نمونه‌ی آموزشی:

$$p = \{-2, -1.2, -0.4, 0.4, 1.2, 2\}$$

$$t = \{0, 0.19, 0.69, 1.3, 1.8, 2\}$$

$$\mathbf{W}^1 = \begin{bmatrix} -2 \\ 0 \\ 2 \end{bmatrix}, \mathbf{b}^1 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

مراکز توابع پایه در محدوده‌ی ورودی با فاصله‌ی مساوی انتخاب می‌شوند: -2، 0، $+2$
بایاس هم در میانه‌ی فضای بین نقاط انتخاب می‌شود.

Example (2)

$$n_{i,q}^1 = \left\| p_q -_i w^1 \right\| b_i^1 \qquad \mathbf{a}_q^1 = \mathbf{radbas}\left(\mathbf{n}_q^1\right)$$

$$\mathbf{a}^1 = \left\{ \begin{bmatrix} 1 \\ 0.368 \\ 0.018 \end{bmatrix}, \begin{bmatrix} 0.852 \\ 0.698 \\ 0.077 \end{bmatrix}, \begin{bmatrix} 0.527 \\ 0.961 \\ 0.237 \end{bmatrix}, \begin{bmatrix} 0.237 \\ 0.961 \\ 0.527 \end{bmatrix}, \begin{bmatrix} 0.077 \\ 0.698 \\ 0.852 \end{bmatrix}, \begin{bmatrix} 0.018 \\ 0.368 \\ 1 \end{bmatrix} \right\}$$

$$\mathbf{U}^T = \begin{bmatrix} 1 & 0.852 & 0.527 & 0.237 & 0.077 & 0.018 \\ 0.368 & 0.698 & 0.961 & 0.961 & 0.698 & 0.368 \\ 0.018 & 0.077 & 0.237 & 0.527 & 0.852 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{t}^T = \begin{bmatrix} 0 & 0.19 & 0.69 & 1.3 & 1.8 & 2 \end{bmatrix}$$

آموزش شبکه‌های RBF

مثال (۲ از ۴)

EXAMPLE

خروجی لایه‌ی اول را محاسبه می‌کنیم:

$$n_{i,q}^1 = \left\| p_q - {}_i w^1 \right\| b_i^1 \qquad \mathbf{a}_q^1 = \mathbf{radbas}\left(\mathbf{n}_q^1\right)$$

$$\mathbf{a}^1 = \left\{ \begin{bmatrix} 1 \\ 0.368 \\ 0.018 \end{bmatrix}, \begin{bmatrix} 0.852 \\ 0.698 \\ 0.077 \end{bmatrix}, \begin{bmatrix} 0.527 \\ 0.961 \\ 0.237 \end{bmatrix}, \begin{bmatrix} 0.237 \\ 0.961 \\ 0.527 \end{bmatrix}, \begin{bmatrix} 0.077 \\ 0.698 \\ 0.852 \end{bmatrix}, \begin{bmatrix} 0.018 \\ 0.368 \\ 1 \end{bmatrix} \right\}$$

محاسبه‌ی $\mathbf{U}$ و $\mathbf{t}$ از روی $\mathbf{a}^1$:

$$\mathbf{U}^T = \begin{bmatrix} 1 & 0.852 & 0.527 & 0.237 & 0.077 & 0.018 \\ 0.368 & 0.698 & 0.961 & 0.961 & 0.698 & 0.368 \\ 0.018 & 0.077 & 0.237 & 0.527 & 0.852 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{t}^T = \begin{bmatrix} 0 & 0.19 & 0.69 & 1.3 & 1.8 & 2 \end{bmatrix}$$

Example (3)

$$\mathbf{x}^* = \left[\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}\right]^{-1}\mathbf{U}^T\mathbf{t}$$

$$\mathbf{x}^* = \begin{bmatrix} 2.07 & 1.76 & 0.42 & 2.71 \\ 1.76 & 3.09 & 1.76 & 4.05 \\ 0.42 & 1.76 & 2.07 & 2.71 \\ 2.71 & 4.05 & 2.71 & 6 \end{bmatrix}^{-1} \begin{bmatrix} 1.01 \\ 4.05 \\ 4.41 \\ 6 \end{bmatrix} = \begin{bmatrix} -1.03 \\ 0 \\ 1.03 \\ 1 \end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix} -1.03 & 0 & 1.03 \end{bmatrix} \qquad \mathbf{b}^2 = \begin{bmatrix} 1 \end{bmatrix}$$

آموزش شبکه‌های RBF

مثال (۳ از ۴)

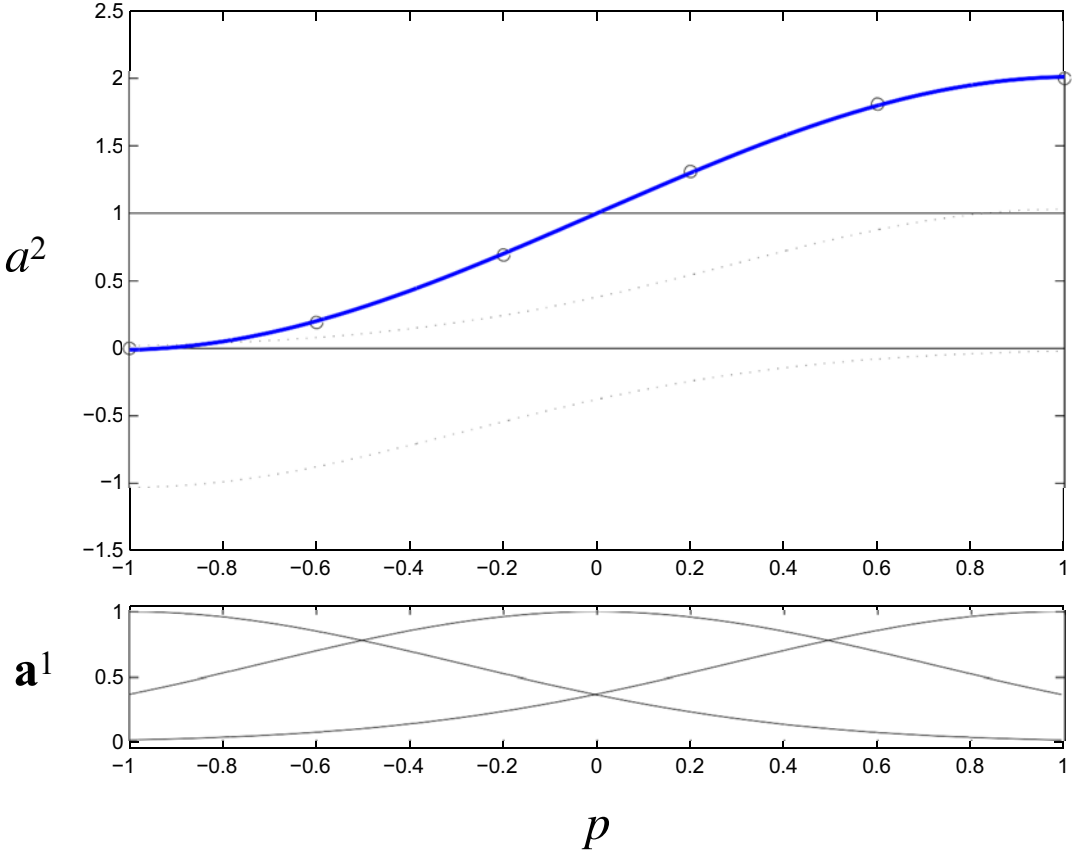
EXAMPLE

$$\mathbf{x}^* = \left[\mathbf{U}^T\mathbf{U} + \rho\mathbf{I}\right]^{-1}\mathbf{U}^T\mathbf{t} \qquad \rho = 0$$

$$\mathbf{x}^* = \begin{bmatrix} 2.07 & 1.76 & 0.42 & 2.71 \\ 1.76 & 3.09 & 1.76 & 4.05 \\ 0.42 & 1.76 & 2.07 & 2.71 \\ 2.71 & 4.05 & 2.71 & 6 \end{bmatrix}^{-1} \begin{bmatrix} 1.01 \\ 4.05 \\ 4.41 \\ 6 \end{bmatrix} = \begin{bmatrix} -1.03 \\ 0 \\ 1.03 \\ 1 \end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix} -1.03 & 0 & 1.03 \end{bmatrix} \qquad \mathbf{b}^2 = \begin{bmatrix} 1 \end{bmatrix}$$

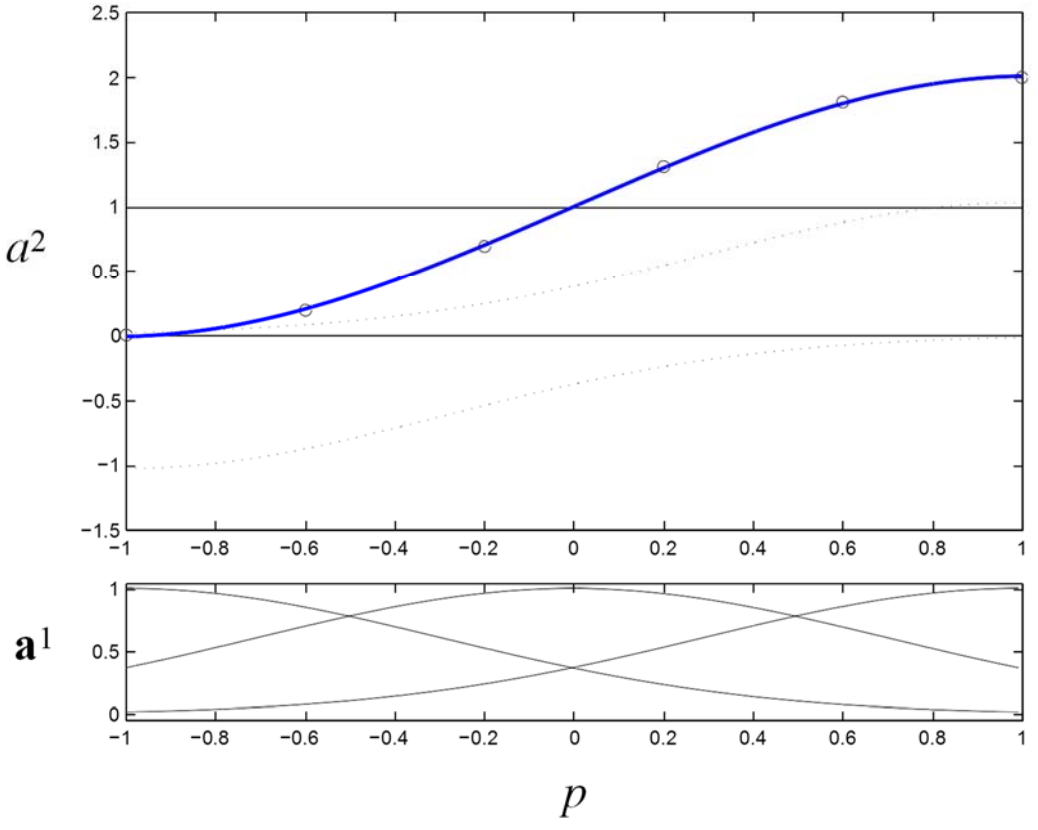
Example (4)



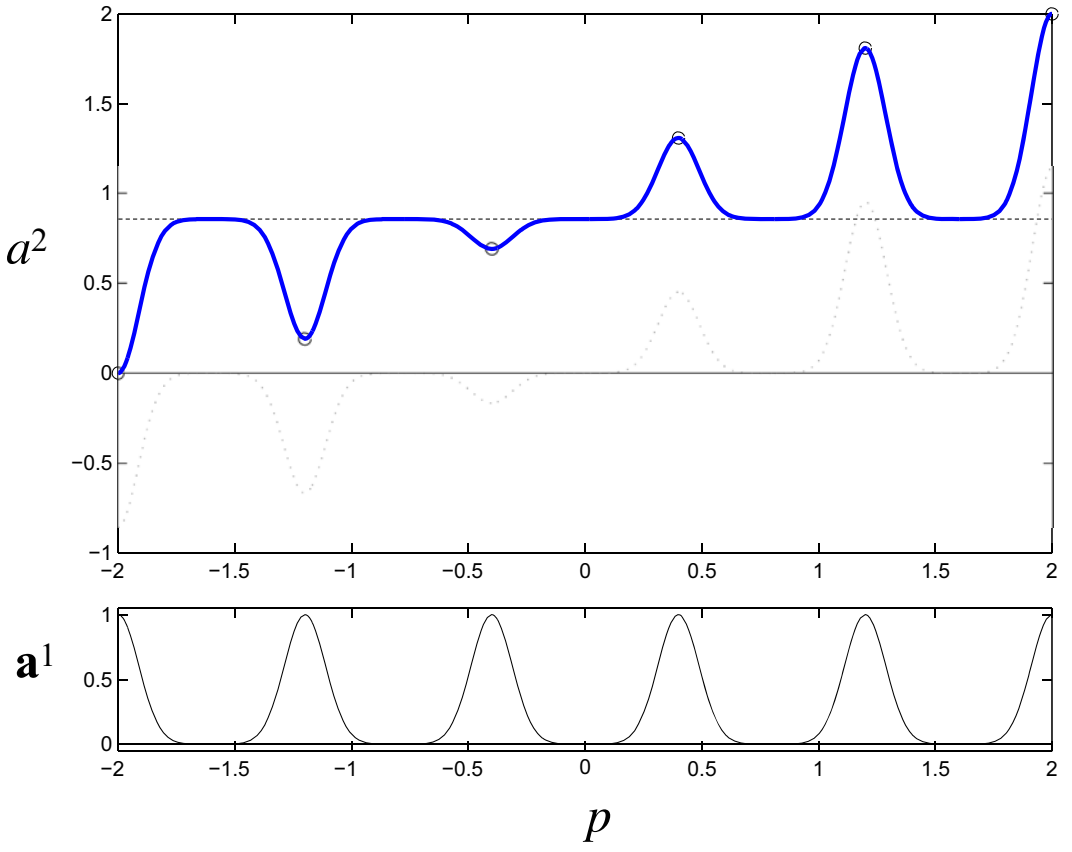
آموزش شبکه‌های RBF

مثال (۴ از ۴)

EXAMPLE



Bias Too Large



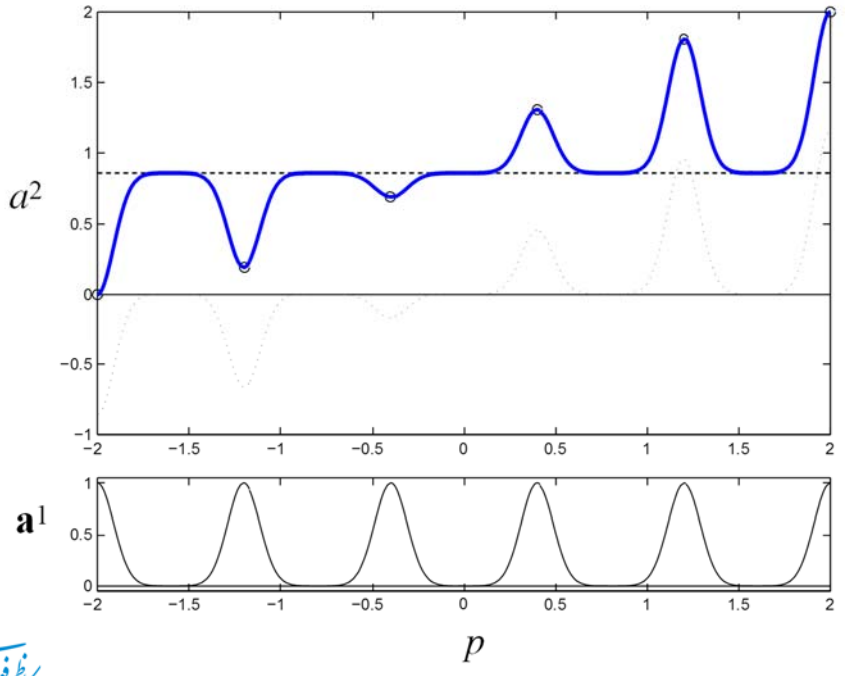
$$\mathbf{b}^1 = \begin{bmatrix} 8 \\ 8 \\ 8 \end{bmatrix}$$

آموزش شبکه‌های RBF

بسیار بزرگ بودن بایاس

BIAS TOO LARGE

$$\mathbf{b}^1 = \begin{bmatrix} 8 \\ 8 \\ 8 \end{bmatrix}$$

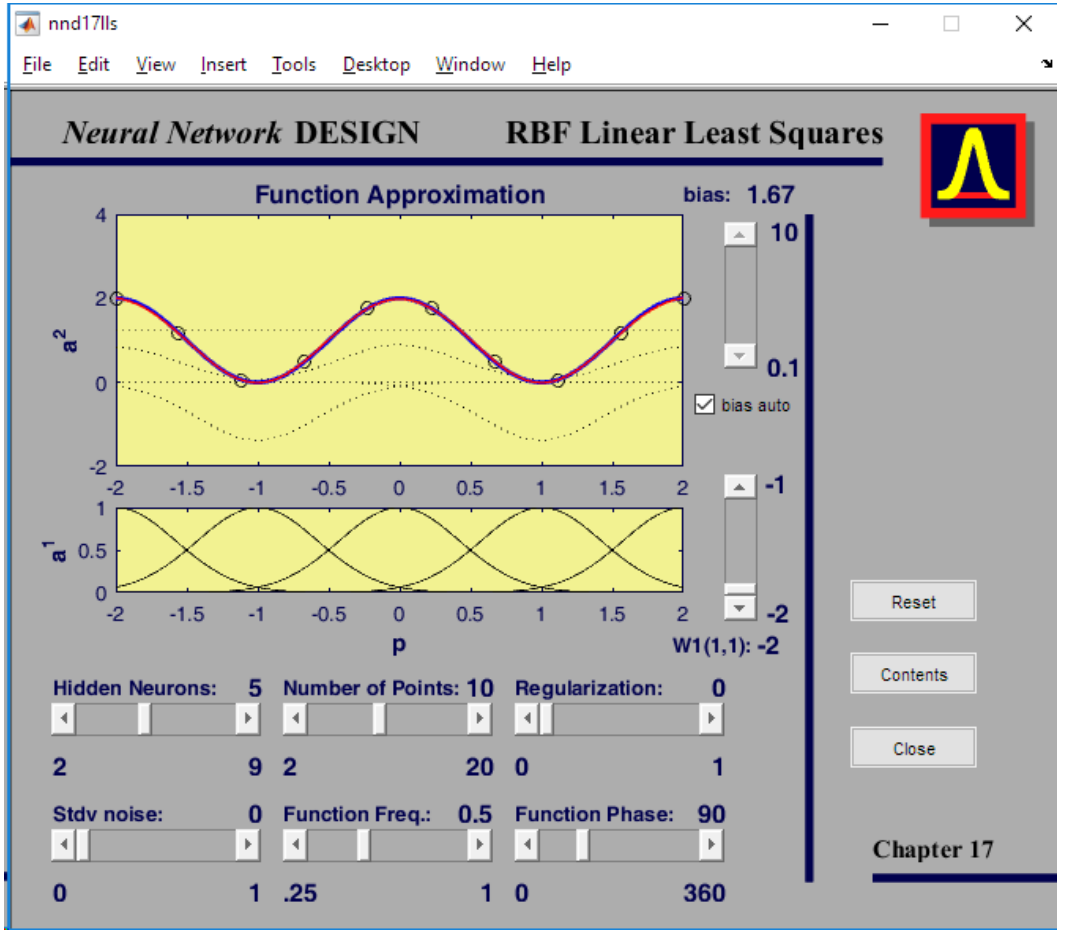


طراحی شبکه‌ی RBF به انتخاب مکان مرکزها و بایاس حساس است.

برای مثال:
۶ مرکز (۶ تابع پایه) و ۶ نقطه داده
بایاس لایه‌ی یک = 8.
در این مثال،
اگرچه همه‌ی نقاط داده به‌خوبی تطبیق داده شدند، اما به‌خاطر طبیعت محلی بودن RBF تقریب تابع واقعی بین نقاط داده دقیق نیست.

گسترش (پراکندگی) تابع پایه با معکوس بایاس کاهش می‌یابد.

وقتی بایاس بزرگ باشد، همپوشانی کافی در توابع پایه برای ایجاد یک تقریب هموار وجود ندارد.


nnd17lls
File Edit View Insert Tools Desktop Window Help
Neural Network DESIGN
RBF Linear Least Squares
Function Approximation
bias: 1.67
10
0.1
bias auto
-1
-2
W1(1,1): -2
Reset
Contents
Close
Chapter 17
Hidden Neurons: 5
2 9
Number of Points: 10
2 20
Regularization: 0
0 1
Stdv noise: 0
0 1
Function Freq.: 0.5
.25 1
Function Phase: 90
0 360

```
>> nnd17lls
```

روش‌های انتخاب مرکزها

روش‌های متفاوتی برای انتخاب مرکزها **(وزن‌های لایه‌ی یک)** وجود دارد:

فرض می‌کنیم تعدادی مرکز بالقوه وجود دارد.

این مرکزها می‌توانند:
* شامل کل بردارهای ورودی در مجموعه‌ی آموزشی باشند.
* شامل بردارهایی باشند که از یک الگوی توری می‌آیند.
* شامل بردارهایی باشند که با هر روش دیگری به‌دست می‌آیند.

هر بار یک بردار از این مجموعه از مراکز بالقوه انتخاب می‌شود
تا اینکه کارآیی شبکه به حد راضی کننده برسد.

هر یک از روش‌های «**انتخاب زیرمجموعه**» می‌تواند برای انتخاب مرکزهای RBF استفاده شود.

Subset Selection

- Given a set of potential first layer weights (centers), which combination should we use?
- An exhaustive search is too expensive.
- Forward selection begins with an empty set and adds centers one at a time.
- Backward elimination begins by using all of the potential centers and then removes them one at a time.
- There are other combinations of the forward and backward methods.
- We will concentrate on one forward selection method, called Orthogonal Least Squares.

روش‌های انتخاب مرکزها

انتخاب زیرمجموعه

SUBSET SELECTION

ایده‌ی اصلی این روش از آمار می‌آید: انتخاب زیرمجموعه

هدف عمومی: انتخاب یک زیرمجموعه‌ی مناسب از متغیرهای مستقل برای کارآمدترین پیش‌بینی یک متغیر وابسته‌ی تارگت.

جستجوی جامع *Exhaustive Search*	بررسی همه‌ی زیرمجموعه‌های ممکن از مرکزها و یافتن کوچک‌ترین آنها که کارآیی لازم را برآورده می‌سازند (بسیار گران است: پیچیدگی زمانی نمایی)

روش‌های زیربهینه با هزینه‌های بسیار کمتر از جستجوی جامع:

انتخاب پیش‌رو *Forward Selection*	با یک مجموعه‌ی تهی شروع می‌کند و هر بار یک مرکز را اضافه می‌کند. (در هر مرحله یک متغیر مستقل که بزرگ‌ترین کاهش در مربع خطا را ایجاد می‌کند اضافه می‌شود. اضافه کردن متغیرها وقتی متوقف می‌شود که به کارآیی لازم برسیم.)
حذف پس‌رو *Backward Elimination*	با به‌کارگیری همه‌ی مرکزهای بالقوه شروع می‌کند و هر بار یک مرکز را حذف می‌کند. (در هر مرحله یک متغیر مستقل که کوچک‌ترین افزایش در مربع خطا را ایجاد می‌کند حذف می‌شود. حذف کردن متغیرها وقتی متوقف می‌شود که کارآیی ناکافی شود.)
سایر روش‌ها *Other Methods*	روش‌های دیگری وجود دارند که ترکیبی از روش‌های پیش‌رو و پس‌رو هستند.

بر روی یک روش انتخاب پیش‌رو به‌نام «حداقل مربعات متعامد: OLS» تمرکز می‌کنیم.

Forward Selection

$$\mathbf{t} = \mathbf{U}\mathbf{x} + \mathbf{e}$$

$$\mathbf{U} = \begin{bmatrix} {}_1\mathbf{u}^T \\ {}_2\mathbf{u}^T \\ \vdots \\ {}_Q\mathbf{u}^T \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_Q^T \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \end{bmatrix} \qquad n = S^1 + 1$$

- There will be one row of **U** for each input/target pair.
- If we consider all input vectors as potential centers, there will be one first-layer neuron for each input vector: $n = Q + 1$.
- In this case, the columns of **U** represent the potential centers.
- We will start with zero centers selected, and at each step we will add the center (or column of **U**) which produces the largest reduction in squared error.

روش‌های انتخاب مرکزها

انتخاب پیش‌رو با متعامدسازی حداقل مربعات

FORWARD SELECTION

$$\mathbf{t} = \mathbf{U}\mathbf{x} + \mathbf{e}$$

$$\mathbf{U} = \begin{bmatrix} {}_1\mathbf{u}^T \\ {}_2\mathbf{u}^T \\ \vdots \\ {}_Q\mathbf{u}^T \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_Q^T \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \end{bmatrix} \qquad n = S^1 + 1$$

برای بایاس

- برای هر زوج ورودی/ تارگت، یک سطر در $\mathbf{U}$ وجود دارد.
- اگر همه‌ی بردارهای ورودی را به‌عنوان مرکزهای بالقوه در نظر بگیریم، آن‌گاه برای هر بردار ورودی یک نرون در لایه اول خواهیم داشت: $n = Q + 1$.
- در این حالت، ستون‌های $\mathbf{U}$ مرکزهای بالقوه را بازنمایی می‌کنند.
- با انتخاب صفر مرکز شروع می‌کنیم و در هر گام یک مرکز (یک ستون $\mathbf{U}$) را اضافه می‌کنیم تا بیشترین کاهش در مربع خطا به‌وجود آید.

روش‌های انتخاب مرکزها

انتخاب پیش‌رو با متعامدسازی حداقل مربعات: مدل رگرسیون خطی

FORWARD SELECTION

$$\mathbf{t} = \mathbf{U}\mathbf{x} + \mathbf{e} \quad (*)$$

$$\mathbf{U} = \begin{bmatrix} {}_1\mathbf{u}^T \\ {}_2\mathbf{u}^T \\ \vdots \\ {}_Q\mathbf{u}^T \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_Q^T \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \end{bmatrix} \qquad n = S^1 + 1$$

برای بایاس

- معادله‌ی (*) در شکل استاندارد **مدل رگرسیون خطی** است.
- به ماتریس **U** **ماتریس رگرسیون** و به ستون‌های **U** **بردارهای رگرسور (regressor)** می‌گویند.

هدف OLS: تعیین اینکه چند ستون (نرون / بردار مرکز / تابع پایه) باید استفاده شود.

- **گام اول:** هر ستون بالقوه به چه میزان مربع خطا را کاهش می‌دهد؟ (باید محاسبه شود)
- **مشکل:** ستون‌ها عموماً با هم همبستگی دارند و تعیین اینکه هر ستون چه میزان خطا را کاهش می‌دهد دشوار است. ⇐

لازم است ابتدا ستون‌ها را متعامد کنیم (تا همبستگی آنها صفر شود).

* برای متعامد کردن ستون‌های **U** باید آن را به‌صورت **U = MR** تجزیه کنیم.

Orthogonalize the Columns

$$\mathbf{U} = \mathbf{MR}$$

$$\mathbf{R} = \begin{bmatrix} 1 & r_{1,2} & r_{1,3} & \cdots & r_{1,n} \\ 0 & 1 & r_{2,3} & \cdots & r_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_{n-1,n} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$\mathbf{M}^T\mathbf{M} = \mathbf{V} = \begin{bmatrix} v_{1,1} & 0 & \cdots & 0 \\ 0 & v_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & v_{n,n} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^T\mathbf{m}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{m}_2^T\mathbf{m}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{m}_n^T\mathbf{m}_n \end{bmatrix}$$

روش‌های انتخاب مرکزها

انتخاب پیش‌رو با متعامدسازی حداقل مربعات: متعامدسازی ستون‌ها

ORTHOGONALIZE THE COLUMNS

$\mathbf{R}$: ماتریس بالامثلثی با قطر تمام یک
$\mathbf{M}$: ماتریسی با ستون‌های متعامد $\mathbf{m}_i$

$$\mathbf{U} = \mathbf{MR}$$

$$\mathbf{R} = \begin{bmatrix} 1 & r_{1,2} & r_{1,3} & \cdots & r_{1,n} \\ 0 & 1 & r_{2,3} & \cdots & r_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_{n-1,n} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$\mathbf{M}^T\mathbf{M} = \mathbf{V} = \begin{bmatrix} v_{1,1} & 0 & \cdots & 0 \\ 0 & v_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & v_{n,n} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^T\mathbf{m}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{m}_2^T\mathbf{m}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{m}_n^T\mathbf{m}_n \end{bmatrix}$$

Orthogonalized Least Squares

$$\mathbf{t} = \mathbf{MRx} + \mathbf{e} = \mathbf{Mh} + \mathbf{e}$$

$$\mathbf{h} = \mathbf{Rx}$$

$$\mathbf{h}^* = \left[\mathbf{M}^T\mathbf{M}\right]^{-1}\mathbf{M}^T\mathbf{t} = \mathbf{V}^{-1}\mathbf{M}^T\mathbf{t}$$

$$h_i^* = \frac{\mathbf{m}_i^T\mathbf{t}}{v_{i,i}} = \frac{\mathbf{m}_i^T\mathbf{t}}{\mathbf{m}_i^T\mathbf{m}_i}$$

روش‌های انتخاب مرکزها

انتخاب پیش‌رو با متعامدسازی حداقل مربعات: روش حداقل مربعات متعامد (OLS)

ORTHOGONALIZED LEAST SQUARES

$$\mathbf{t} = \mathbf{U}\mathbf{x} + \mathbf{e} \Rightarrow \qquad \mathbf{t} = \mathbf{M}\mathbf{R}\mathbf{x} + \mathbf{e} = \mathbf{M}\mathbf{h} + \mathbf{e} \qquad (*)$$

$$\mathbf{U} = \mathbf{M}\mathbf{R}$$

$$\mathbf{h} = \mathbf{R}\mathbf{x}$$

راه‌حل حداقل مربعات برای (*):

$$\mathbf{h}^* = \left[\mathbf{M}^T\mathbf{M}\right]^{-1}\mathbf{M}^T\mathbf{t} = \mathbf{V}^{-1}\mathbf{M}^T\mathbf{t}$$

چون $\mathbf{V}$ قطری است، عناصر $\mathbf{h}^*$ به‌صورت زیر محاسبه می‌شوند:

$$h_i^* = \frac{\mathbf{m}_i^T\mathbf{t}}{v_{i,i}} = \frac{\mathbf{m}_i^T\mathbf{t}}{\mathbf{m}_i^T\mathbf{m}_i}$$

از روی $\mathbf{h}^*$ می‌توان $\mathbf{x}^*$ را به‌صورت $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}^*$ محاسبه کرد.

برای یافتن بردارهای متعامد $\mathbf{m}_i$ روش‌های متعددی وجود دارد که یکی از آنها روش متعامدسازی گرام–اشمیت است.

Gram-Schmidt Orthogonalization

$$\mathbf{m}_1 = \mathbf{u}_1$$

$$\mathbf{m}_k = \mathbf{u}_k - \sum_{i=1}^{k-1} r_{i,k}\mathbf{m}_i$$

$$r_{i,k} = \frac{\mathbf{m}_i^T \mathbf{u}_k}{\mathbf{m}_i^T \mathbf{m}_i}, \quad i = 1,\ldots,k-1$$

روش‌های انتخاب مرکزها

انتخاب پیش‌رو با متعامدسازی حداقل مربعات: روش حداقل مربعات متعامد (OLS): متعامدسازی گرام–اشمیت

GRAM-SCHMIDT ORTHOGONALIZATION

$$\mathbf{m}_1 = \mathbf{u}_1$$

$$\mathbf{m}_k = \mathbf{u}_k - \sum_{i=1}^{k-1} r_{i,k}\mathbf{m}_i$$

فصل ۵

$$r_{i,k} = \frac{\mathbf{m}_i^T \mathbf{u}_k}{\mathbf{m}_i^T \mathbf{m}_i}, \quad i = 1,\ldots,k-1$$

با متعامدسازی ستون‌های $\mathbf{U}$ قادر می‌شویم نقش هر بردار پایه در مربع خطا را به‌طور کارآمد محاسبه کنیم.

Incremental Error

The total squared value is:

$$\mathbf{t}^T\mathbf{t} = \left[\mathbf{Mh}+\mathbf{e}\right]^T\left[\mathbf{Mh}+\mathbf{e}\right] = \mathbf{h}^T\mathbf{M}^T\mathbf{Mh}+\mathbf{e}^T\mathbf{Mh}+\mathbf{h}^T\mathbf{M}^T\mathbf{e}+\mathbf{e}^T\mathbf{e}$$

$$\mathbf{e}^T\mathbf{Mh} = \left[\mathbf{t}-\mathbf{Mh}\right]^T\mathbf{Mh} = \mathbf{t}^T\mathbf{Mh}-\mathbf{h}^T\mathbf{M}^T\mathbf{Mh}$$

$$\mathbf{h}^* = \mathbf{V}^{-1}\mathbf{M}^T\mathbf{t} \quad \Longrightarrow \quad \mathbf{e}^T\mathbf{Mh}^* = \mathbf{t}^T\mathbf{Mh}^* - \mathbf{t}^T\mathbf{MV}^{-1}\mathbf{M}^T\mathbf{Mh}^* = \mathbf{0}$$

$$\mathbf{t}^T\mathbf{t} = \mathbf{h}^T\mathbf{M}^T\mathbf{Mh}+\mathbf{e}^T\mathbf{e} = \sum_{i=1}^{n} h_i^2\mathbf{m}_i^T\mathbf{m}_i + \mathbf{e}^T\mathbf{e}$$

Therefore, basis function *i* contributes the following to the squared value:

$$h_i^2\mathbf{m}_i^T\mathbf{m}_i$$

Normalized error contribution: $o_i = \dfrac{h_i^2\mathbf{m}_i^T\mathbf{m}_i}{\mathbf{t}^T\mathbf{t}}$

روش‌های انتخاب مرکزها

انتخاب پیش‌رو با متعامدسازی حداقل مربعات: روش حداقل مربعات متعامد (OLS): محاسبه‌ی خطای افزایشی

INCREMENTAL ERROR

The total squared value is: مجموع مربعات تارگت‌ها به‌صورت زیر محاسبه می‌شود:

$$\mathbf{t}^T\mathbf{t} = \left[\mathbf{Mh}+\mathbf{e}\right]^T\left[\mathbf{Mh}+\mathbf{e}\right] = \mathbf{h}^T\mathbf{M}^T\mathbf{Mh} + \mathbf{e}^T\mathbf{Mh} + \mathbf{h}^T\mathbf{M}^T\mathbf{e} + \mathbf{e}^T\mathbf{e}$$

$$\mathbf{e}^T\mathbf{Mh} = \left[\mathbf{t}-\mathbf{Mh}\right]^T\mathbf{Mh} = \mathbf{t}^T\mathbf{Mh} - \mathbf{h}^T\mathbf{M}^T\mathbf{Mh}$$

$$\mathbf{h}^* = \mathbf{V}^{-1}\mathbf{M}^T\mathbf{t} \quad \Longrightarrow \quad \mathbf{e}^T\mathbf{Mh}^* = \mathbf{t}^T\mathbf{Mh}^* - \mathbf{t}^T\mathbf{MV}^{-1}\mathbf{M}^T\mathbf{Mh}^* = \mathbf{0}$$

$$\mathbf{t}^T\mathbf{t} = \mathbf{h}^T\mathbf{M}^T\mathbf{Mh} + \mathbf{e}^T\mathbf{e} = \sum_{i=1}^{n} h_i^2\mathbf{m}_i^T\mathbf{m}_i + \mathbf{e}^T\mathbf{e}$$

سهم در مربع خطا وابسته به بردارهای رگرسور

مابقی مربع خطا عیر قابل توضیح با رگرسورها

Therefore, basis function i contributes the following to the squared value:

$$h_i^2\mathbf{m}_i^T\mathbf{m}_i$$ سهم تابع پایه‌ی i در مقدار مربع خطا:

Normalized error contribution: $$o_i = \frac{h_i^2\mathbf{m}_i^T\mathbf{m}_i}{\mathbf{t}^T\mathbf{t}}$$ سهم نرمال‌شده: (مقداری بین صفر و یک)

انتخاب پیش‌رو با متعامدسازی حداقل مربعات: روش حداقل مربعات متعامد (OLS): الگوریتم OLS

OLS ALGORITHM

الگوریتم OLS

با همه‌ی توابع بالقوه‌ی موجود در ماتریس رگرسیون $\mathbf{U}$ شروع می‌کنیم

(اگر همه‌ی بردارهای ورودی در مجموعه‌ی آموزشی به‌عنوان مراکز بالقوه‌ی توابع پایه در نظر گرفته شوند، آن‌گاه اندازه‌ی ماتریس $\mathbf{U}$ می‌شود: $Q \times (Q+1)$)

- **مرحله‌ی صفر:** هنوز هیچ تابع پایه‌ای به شبکه اضافه نشده است.
- **مرحله‌ی یکم:** انتخاب تابع پایه‌ای که بیشترین کاهش در خطا را ایجاد می‌کند.
- **مرحله‌ی kام:** متعامدسازی
- تکرار تا رسیدن به شرط توقف

OLS Algorithm

First Step ($k = 1$):

$$\mathbf{m}_1^{(i)} = \mathbf{u}_i, \quad i = 1,\ldots,Q \qquad h_1^{(i)} = \frac{\mathbf{m}_1^{(i)T}\mathbf{t}}{\mathbf{m}_1^{(i)T}\mathbf{m}_1^{(i)}}$$

$$o_1^i = \frac{\left(h_1^{(i)}\right)^2 \mathbf{m}_1^{(i)T}\mathbf{m}_1^{(i)}}{\mathbf{t}^T\mathbf{t}} \qquad o_1 = o_1^{(i_1)} = \max\left\{o_1^{(i)}\right\} \qquad \mathbf{m}_1 = \mathbf{m}_1^{(i_1)} = \mathbf{u}_{i_1}$$

For $i = 1,\ldots,Q, \quad i \neq i_1, i \neq i_2, \ldots, i \neq i_{k-1}$

$$r_{j,k}^{(i)} = \frac{\mathbf{m}_j^T\mathbf{u}_i}{\mathbf{m}_j^T\mathbf{m}_j}, \quad j = 1,\ldots,k-1 \qquad \mathbf{m}_k^{(i)} = \mathbf{u}_i - \sum_{j=1}^{k-1} r_{j,k}^{(i)}\mathbf{m}_j$$

$$h_k^{(i)} = \frac{\mathbf{m}_k^{(i)T}\mathbf{t}}{\mathbf{m}_k^{(i)T}\mathbf{m}_k^{(i)}} \qquad o_k^i = \frac{\left(h_k^{(i)}\right)^2 \mathbf{m}_k^{(i)T}\mathbf{m}_k^{(i)}}{\mathbf{t}^T\mathbf{t}} \qquad o_k = o_k^{(i_k)} = \max\left\{o_k^{(i)}\right\}$$

$$r_{j,k} = r_{j,k}^{(i_k)}, \quad j = 1,\ldots,k-1 \qquad \mathbf{m}_k = \mathbf{m}_k^{(i_k)}$$

روش‌های انتخاب مرکزها

انتخاب پیش‌رو با متعامدسازی حداقل مربعات: روش حداقل مربعات متعامد (OLS): الگوریتم OLS

OLS ALGORITHM

مرحله‌ی یکم:
برای همه‌ی نرون‌ها

First Step ($k=1$):

$$\mathbf{m}_1^{(i)} = \mathbf{u}_i, \quad i = 1,\ldots,Q \qquad h_1^{(i)} = \frac{\mathbf{m}_1^{(i)T}\mathbf{t}}{\mathbf{m}_1^{(i)T}\mathbf{m}_1^{(i)}}$$

$$o_1^i = \frac{\left(h_1^{(i)}\right)^2 \mathbf{m}_1^{(i)T}\mathbf{m}_1^{(i)}}{\mathbf{t}^T\mathbf{t}} \qquad o_1 = o_1^{(i_1)} = \max\left\{o_1^{(i)}\right\} \qquad \mathbf{m}_1 = \mathbf{m}_1^{(i_1)} = \mathbf{u}_{i_1}$$

انتخاب تابع پایه‌ای که بیشترین کاهش در مربع خطا ایجاد می‌کند.

مرحله‌ی kام:
برای نرون‌های بعدی

For $i = 1,\ldots,Q, \quad i \neq i_1, i \neq i_2, \ldots, i \neq i_{k-1}$

$$r_{j,k}^{(i)} = \frac{\mathbf{m}_j^T\mathbf{u}_i}{\mathbf{m}_j^T\mathbf{m}_j}, \quad j = 1,\ldots,k-1 \qquad \mathbf{m}_k^{(i)} = \mathbf{u}_i - \sum_{j=1}^{k-1} r_{j,k}^{(i)}\mathbf{m}_j$$

$$h_k^{(i)} = \frac{\mathbf{m}_k^{(i)T}\mathbf{t}}{\mathbf{m}_k^{(i)T}\mathbf{m}_k^{(i)}} \qquad o_k^i = \frac{\left(h_k^{(i)}\right)^2 \mathbf{m}_k^{(i)T}\mathbf{m}_k^{(i)}}{\mathbf{t}^T\mathbf{t}} \qquad o_k = o_k^{(i_k)} = \max\left\{o_k^{(i)}\right\}$$

$$r_{j,k} = r_{j,k}^{(i_k)}, \quad j = 1,\ldots,k-1 \qquad \mathbf{m}_k = \mathbf{m}_k^{(i_k)}$$

Stopping Criteria

$$1 - \sum_{j=1}^{k} o_j < \delta$$

To convert to original weights:

$$x_n = h_n, \quad x_k = h_k - \sum_{j=k+1}^{n} r_{j,k} x_j$$

روش‌های انتخاب مرکزها

انتخاب پیش‌رو با متعامدسازی حداقل مربعات: روش حداقل مربعات متعامد (OLS): الگوریتم OLS: ضابطه‌ی توقف

STOPPING CRITERIA

تکرارها ادامه می‌یابد تا به شرط مطرح در ضابطه‌ی توقف برسیم، مثلاً:

$$1-\sum_{j=1}^{k} o_j < \delta$$

δ یک مقدار کوچک است.

اگر δ خیلی کوچک باشد، بیش‌برازش داریم، زیرا شبکه بسیار پیچیده می‌شود.

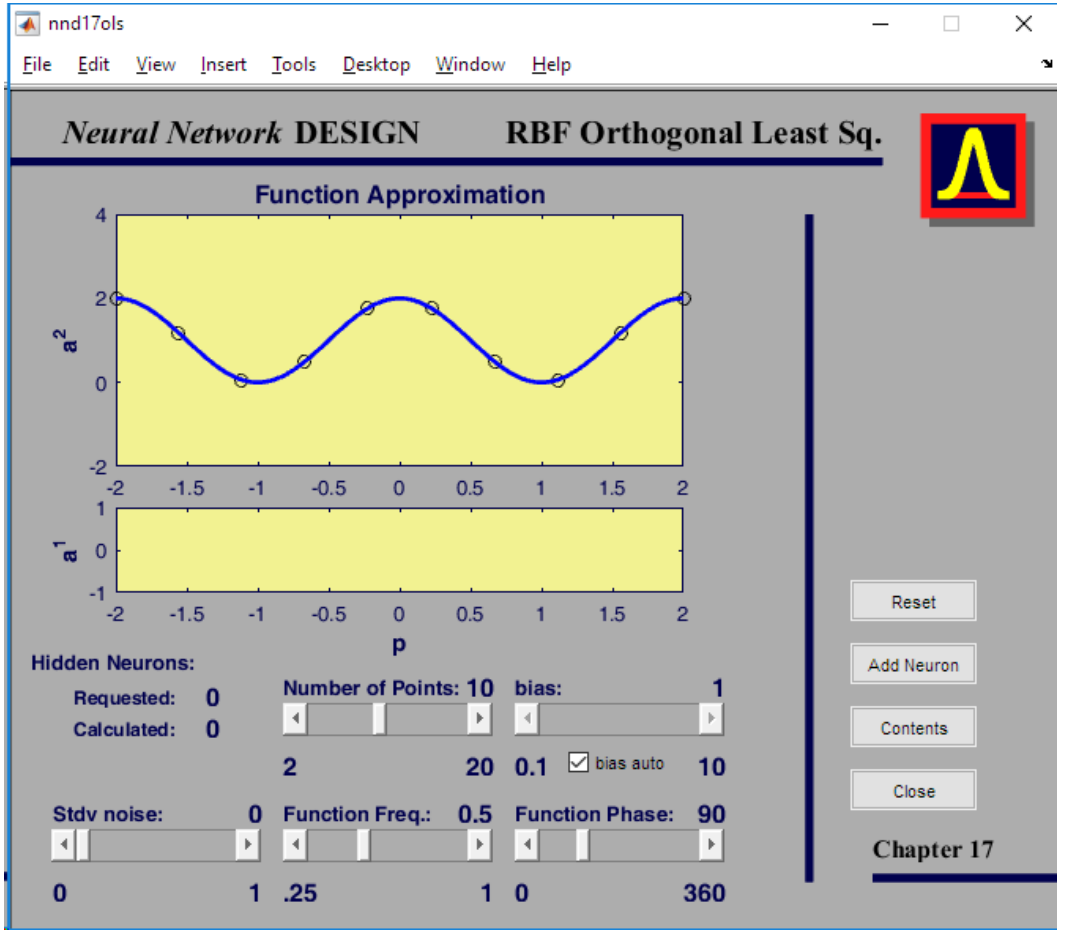
راه‌حل: استفاده از یک مجموعه‌ی اعتبارسنجی (شرط توقف: زمانی که خطا روی مجموعه‌ی اعتبارسنجی افزایش یابد.)

To convert to original weights:

پس از همگرایی الگوریتم، مقادیر اصلی x_k با جایگزینی معکوس (از x_n تا x_1) به‌دست می‌آید:

$$x_n = h_n, \quad x_k = h_k - \sum_{j=k+1}^{n} r_{j,k} x_j$$

n تعداد نهایی وزن‌ها و بایاس‌های لایه‌ی دوم است.


nnd17ols
File Edit View Insert Tools Desktop Window Help
Neural Network DESIGN
RBF Orthogonal Least Sq.
Function Approximation
a2
a1
p
Hidden Neurons:
Requested: 0
Calculated: 0
Number of Points: 10
2 20
bias: 1
0.1 10
bias auto
Stdv noise: 0
0 1
Function Freq.: 0.5
.25 1
Function Phase: 90
0 360
Reset
Add Neuron
Contents
Close
Chapter 17

```
>> nnd17ols
```

روش‌های انتخاب مرکزها

خوشه‌بندی

CLUSTERING

خوشه‌بندی: یک روی‌کرد دیگر برای انتخاب وزن‌ها و بایاس‌ها در لایه‌ی اول شبکه‌ی RBF.
(مانند استفاده از شبکه‌های رقابتی مثل کوهونن و ...)

خوشه‌بندی فضای ورودی
⇓
پس از آموزش، سطرهای شبکه‌ی رقابتی حاوی پروتوتایپ‌ها (مراکز خوشه‌ها) هستند.
از مراکز خوشه‌ها به‌عنوان مراکز توابع پایه استفاده می‌کنیم.

نقطه‌ضعف بالقوه در روش خوشه‌بندی برای آموزش لایه‌ی اول RBF:
این روش فقط بردارهای ورودی را در نظر می‌گیرد و تارگت‌ها را کنار می‌گذارد.
⇐ ممکن است تابع مورد نظر برای تقریب‌زنی، در ناحیه‌هایی که نرون کمتری داریم پیچیده‌تر باشد و در این حالت خوشه‌بندی مراکز را به‌طور مناسب توزیع نمی‌کند.
از طرفی، می‌توان امیدوار بود که داده‌های آموزشی در نواحی پر استفاده‌ی شبکه قرار گرفته‌اند و بنابراین، تقریب تابع در این نواحی دقیق‌تر است.

* می‌توان از سایر روش‌های خوشه‌بندی (مانند K-means) هم استفاده کرد.
پس از این مرحله، برای وزن‌های لایه‌ی دوم و بایاس‌های آن از روش LMS استفاده می‌شود.

Competitive Learning for First Layer

- Cluster the input space using a competitive layer (or Feature Map).
- Use the cluster centers as basis function centers.
- The bias can be computed from the variation in each cluster:

$$dist_i = \frac{1}{n_c}\left(\sum_{j=1}^{n_c}\left\|\mathbf{p}^i_j - {}_i\mathbf{w}^1\right\|^2\right)^{\frac{1}{2}}$$

$$b_i^1 = \frac{1}{\sqrt{2}dist_i}$$

روش‌های انتخاب مرکزها

خوشه‌بندی: یادگیری رقابتی برای لایه‌ی اول

COMPETITIVE LEARNING FOR FIRST LAYER

- Cluster the input space using a competitive layer (or Feature Map).
  - فضای ورودی را با استفاده از یک لایه‌ی رقابتی (یا نقشه‌ی ویژگی) خوشه‌بندی کنید.
- Use the cluster centers as basis function centers.
  - از مرکزهای خوشه‌ها به‌عنوان مراکز توابع پایه استفاده کنید.
- The bias can be computed from the variation in each cluster:
  - بایاس می‌توان از روی واریانس هر خوشه محاسبه شود.

$$dist_i = \frac{1}{n_c}\left(\sum_{j=1}^{n_c}\left\|\mathbf{p}_j^i - {}_i\mathbf{w}^1\right\|^2\right)^{\frac{1}{2}}$$

متوسط فاصله میان مرکز و همسایه‌های آن

n_c = تعداد نرون‌های لایه‌ی یک
(تعداد بردارهای نزدیک به بردار مرکز)

$$b_i^1 = \frac{1}{\sqrt{2}dist_i}$$

بایاس نرون‌ها با هم متفاوت خواهد بود.

Backpropagation

$$n_i^1 = \left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\| b_i^1 = b_i^1 \sqrt{\sum_{j=1}^{S^1} \left(p_j - w_{i,j}^1 \right)^2}$$

$$\frac{\partial n_i^1}{\partial w_{i,j}^1} = \frac{b_i^1 \frac{1}{2}}{\sqrt{\sum_{j=1}^{S^1} \left(p_j - w_{i,j}^1 \right)^2}} 2\left(p_j - w_{i,j}^1 \right)(-1) = \frac{b_i^1 \left(w_{i,j}^1 - p_j \right)}{\left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\|}$$

$$\frac{\partial n_i^1}{\partial b_i^1} = \left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\|$$

$$\frac{\partial \hat{F}}{\partial w_{i,j}^1} = s_i^1 \frac{b_i^1 \left(w_{i,j}^1 - p_j \right)}{\left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\|} \qquad \frac{\partial \hat{F}}{\partial b_i^1} = s_i^1 \left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\|$$

پس‌انتشار

BACKPROPAGATION

بهینه‌سازی غیرخطی: آموزش RBF مشابه MLP.

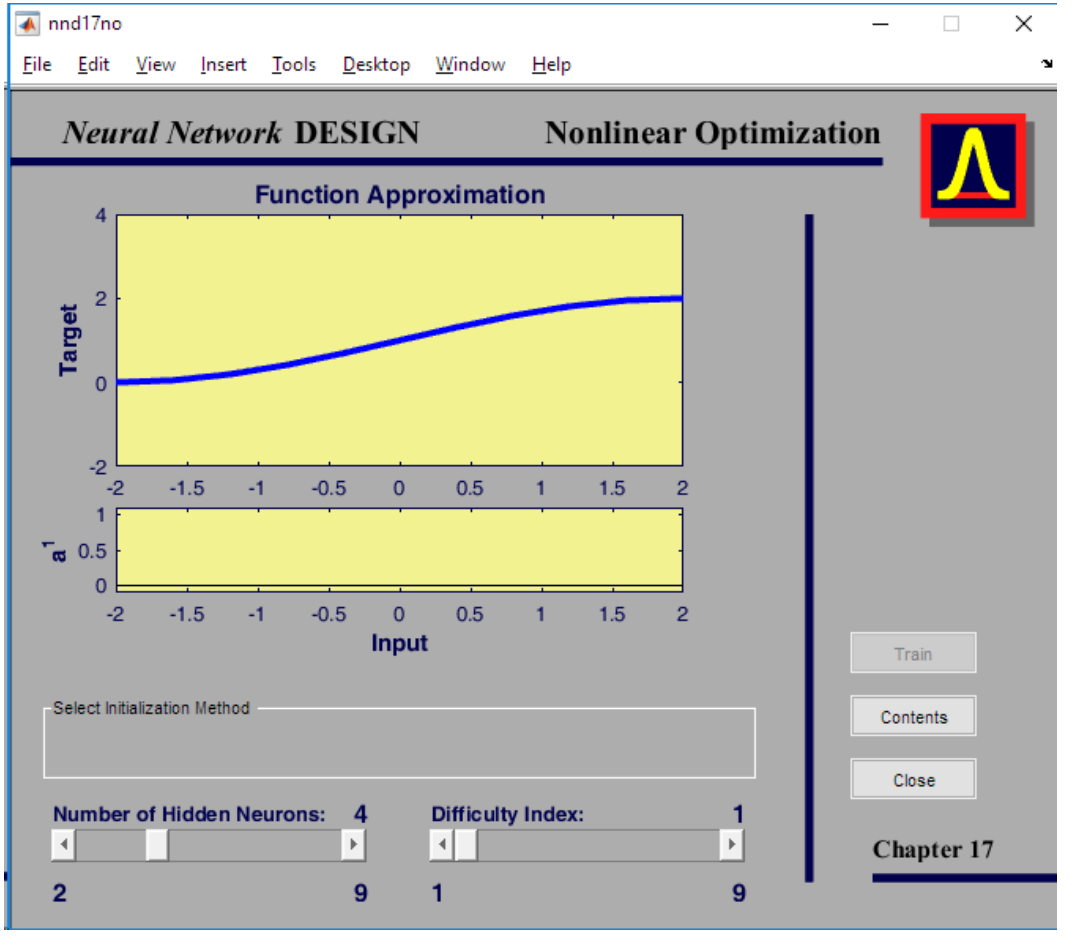
برای آموزش کامل شبکه‌ی RBF استفاده نمی‌شود، اما برای fine-tune کردن شبکه پس از آموزش اولیه مناسب است.

$$n_i^1 = \left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\| b_i^1 = b_i^1 \sqrt{\sum_{j=1}^{S^1} \left(p_j - w_{i,j}^1 \right)^2}$$

$$\frac{\partial n_i^1}{\partial w_{i,j}^1} = \frac{b_i^1 \frac{1}{2}}{\sqrt{\sum_{j=1}^{S^1} \left(p_j - w_{i,j}^1 \right)^2}} 2\left(p_j - w_{i,j}^1 \right)(-1) = \frac{b_i^1 \left(w_{i,j}^1 - p_j \right)}{\left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\|}$$

$$\frac{\partial n_i^1}{\partial b_i^1} = \left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\|$$

$$\frac{\partial \hat{F}}{\partial w_{i,j}^1} = s_i^1 \frac{b_i^1 \left(w_{i,j}^1 - p_j \right)}{\left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\|} \qquad \frac{\partial \hat{F}}{\partial b_i^1} = s_i^1 \left\| \mathbf{p} - {}_i\mathbf{w}^1 \right\|$$


nnd17no
File Edit View Insert Tools Desktop Window Help
Neural Network DESIGN
Nonlinear Optimization
Function Approximation
Target
4
2
0
-2
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2
a1
1
0.5
0
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2
Input
Select Initialization Method
Number of Hidden Neurons: 4
2 9
Difficulty Index: 1
1 9
Train
Contents
Close
Chapter 17

```
>> nnd17no
```

سایر تکنیک‌های آموزش RBF

- گسترش OLS برای چند خروجی
- استفاده از معیار کارآیی رگولاریزه
- استفاده از الگوریتم ژنتیک برای انتخاب وزن‌ها و بایاس لایه‌ی یک
- استفاده از الگوریتم ماکزیمم‌سازی امید (EM) برای انتخاب وزن‌ها و بایاس لایه‌ی یک
- استفاده از درخت رگرسیون برای انتخاب وزن‌ها و بایاس لایه‌ی یک

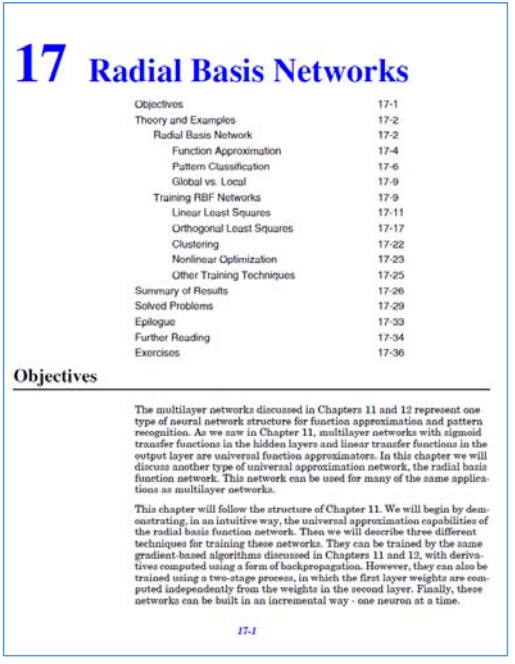
معماری RBF به‌خودی خود روی‌کردهای متعددی برای آموزش را پیشنهاد می‌دهد.

شبکه‌های توابع پایه‌ی شعاعی

۳

منابع

منبع اصلی



17 Radial Basis Networks

Objectives

The multilayer networks discussed in Chapters 11 and 12 represent one type of neural network structure for function approximation and pattern recognition. As we saw in Chapter 11, multilayer networks with sigmoid transfer functions in the hidden layers and linear transfer functions in the output layer are universal function approximators. In this chapter we will discuss another type of universal approximation network, the radial basis function network. This network can be used for many of the same applications as multilayer networks.

This chapter will follow the structure of Chapter 11. We will begin by demonstrating, in an intuitive way, the universal approximation capabilities of the radial basis function network. Then we will describe three different techniques for training these networks. They can be trained by the same gradient-based algorithms discussed in Chapters 11 and 12, with derivatives computed using a form of backpropagation. However, they can also be trained using a two-stage process, in which the first layer weights are computed independently from the weights in the second layer. Finally, these networks can be built in an incremental way - one neuron at a time.

17-1

Martin T. Hagan, Howard B. Demuth, Mark H. Beale, Orlando De Jesus,
Neural Network Design,
2nd Edition, Martin Hagan, 2014.
Chapter 17

Online version can be downloaded from: http://hagan.okstate.edu/nnd.html