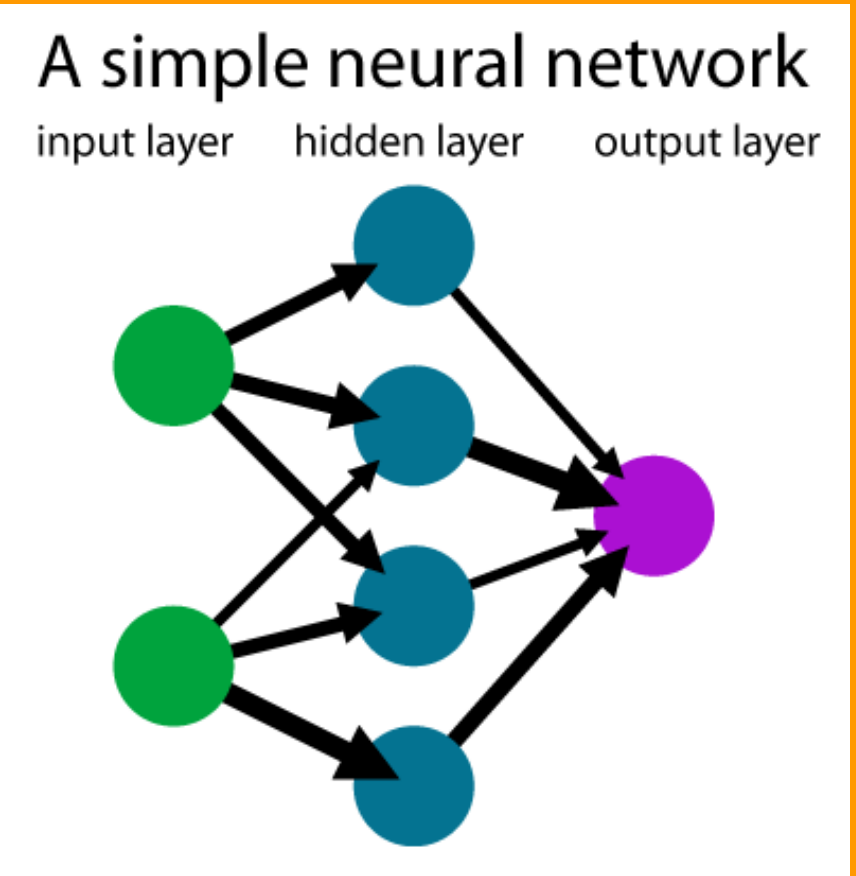
به نام خدا

دانش اگر در ثریا هم باشد مردانی از سرزمین پارس بدان دست خواهند یافت. رسول اکرم (ص)



دانشگاه شهید چمران اهواز
دانشکده مهندسی
گروه برق

شبکه های عصبی مصنوعی
Artificial Neural Networks (ANN)



استاد: دکتر محمود جورابیان

دانشیار دانشگاه شهید چمران اهواز

مراجع:

کتاب درس:

"Neural Networks" ; Simon Haykin

1)Understanding Neural Networks and Fuzzy logic

S.V.Kartalopoulos ترجمه : جورابیان –هوشمند

2)Artifical Neural Networks ;Rbert J.Schalaff

ترجمه : جورابیان

3) Neural Networks Fundamentals (N.K. Bace,P.Liang)

4) مقالات ومجلات علمی در زمینه شبکه های عصبی وکاربردها

فهرست مطالب

تاریخچه شبکه های عصبی مصنوعی

مدلسازی نرونها و شبکه های عصبی مصنوعی

آشنایی با شبکه های پرسپترون ، هاپفیلد،کوهونن RBF، BAM ، ART و...

یاد گیری در شبکه های عصبی وروشهای مختلف

شبکه های عصبی اتو کاستیک ،ماشین بولتزمن sinulated annealing

بهینه سازی به کمک شبکه های عصبی

کاربرد شبکه های عصبی در کنترل

کاربرد شبکه های عصبی در قدرت

ارزشیابی

امتحان پایان ترم 50 %

امتحان میان ترم 20 %

پروژه تحقیقی 20 %

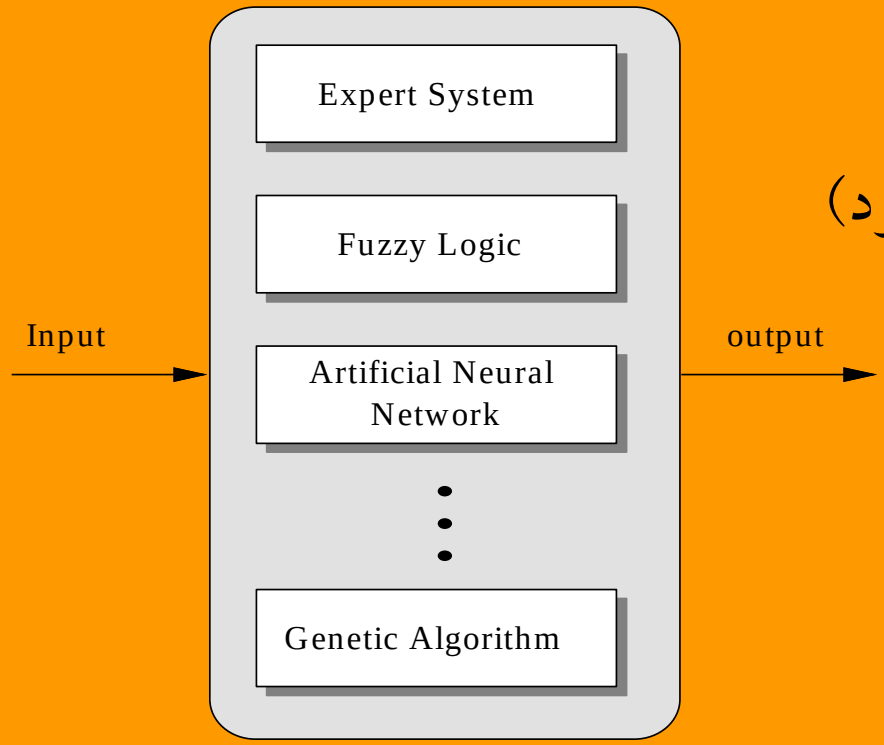
تمرین 10 %

AI Techniques

-منطق فازی: از دیدگاه انسان نسبت به صفات وتعاریف، نشات می گیرد

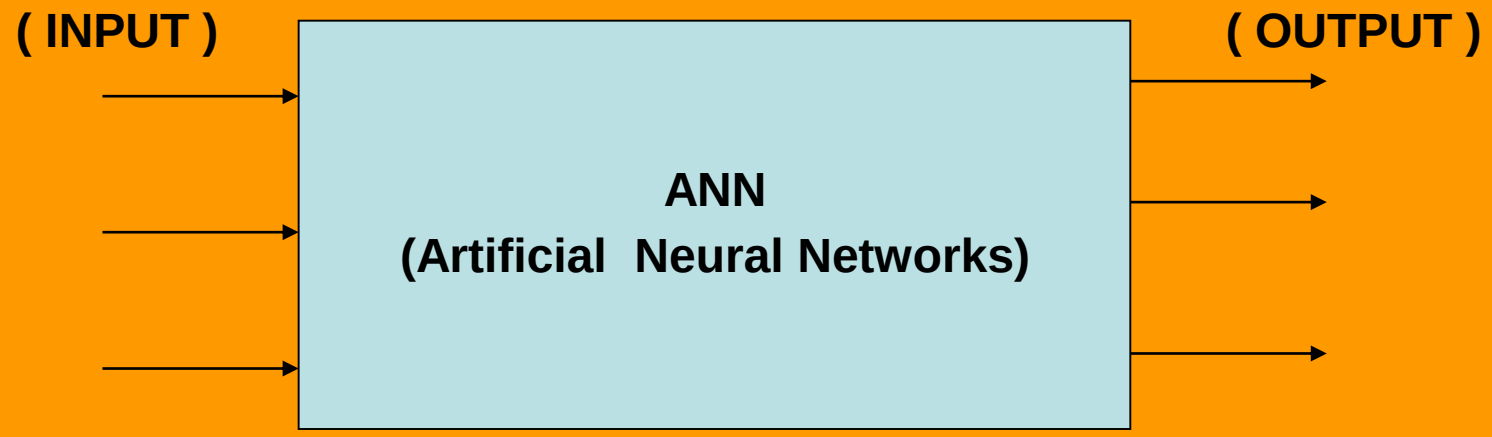
-الگوریتم ژنتیک : از طبیعت انسان نشات می گیرد
(در بحث هایی مانند تولید مثل عنوان می شود)

-شبکه های عصبی:
ازچگونگی عملکرد سلولها وعصبها نشات می گیرد



-هرسلول (**Neuron**): به عنوان یک تابع ریاضی مطرح می شود.
یعنی خروجی عصب تابعی است از ورودی آن.

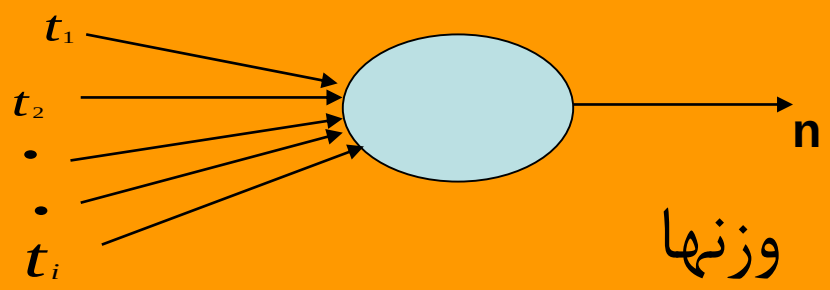
-اطلاعات درشبکه های عصبی در ضرایب یا وزنهای ارتباطی وپارامترهای توابع عصبی ذخیره خواهد شد.



- قبل از بکارگیری (**ANN**)باید آنرا آموزش داد که این کار توسط الگوریتم های آموزشی (**EX.**) انجام می شود.یعنی اینکه برای محاسبه $Y = X_1 + X_2$ تمام حالات ممکن را در اختیارآن قرار می دهیم.

تاریخچه (ANN):

۱۹۴۳ - توسط Mc Cullogh &Pits مدل سازی نرون



وزنها

آ ستانه
(Threshold)

$$n_i(t+1) = f\left(\sum_j w_{ij} n_j(t) - \theta_i\right)$$

خروجی دوقطبی $n_i = \pm 1$

$= 0,1$

خروجی

وزن از نرون j ام به نرون i ام= W_{ij}

تابع پله واحد=

f تابع تبدیل

۱۹٤۸ - توسط Wiener مطالعه ومقایسه بین دستگاه عصبی خودکار که مرکب از مغز واعصاب می باشد –با دستگاه الکتریکی ومکانیکی مرتب شده است.

۱۹٤۹ - توسط Hebb بررسی رفتارهای مغز .

۱۹۵۲ - توسط Ashby طراحی مغز .

۱۹۵٤ - توسط Minsky مربوط به تزدکتری(شبکه های عصبی) .

۱۹٦۰ - توسط Frank Rosonblat (چگونگی بدست آوردن وزنها)
پرسپترون (Perseptron)
شبکه های پیشخورد (Feed-Forward Neural Networks)

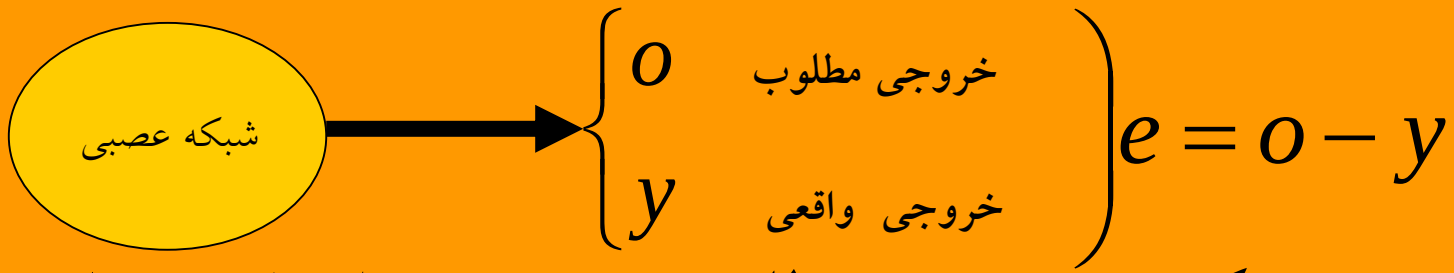
۱۹۶۸- کتاب پرسپترون Minsky &Popert از شبکه های دولایه نمی توان برای تحقق همه توابع logic (NADN OR, AND , XOR و...) استفاده کرد.

۱۹۸۰- ۱۹۶۰ –Dark adge سالهایی که توقفی درعلم شبکه های عصبی حاصل شد.

۱۹۷۰- توسط Kohonen برای آموزش سیستمهای رقابتی استفاده می شود.

۱۹۸۰ - توسط Gross berg
ارائه دهنده مدل ART (Adaptive Resonance Theyo)

۱۹۸۲- شبکه هاپفیلد (Hopfield Net) بر مبنای اثر متقابل نرونها .



اگر خروجی واقعی نزدیک به خروجی مطلوب شود یعنی اختلاف میان آنها به یک مقداری که ما تعیین کردیم(دقت محاسبات)برسد که error (e=o-y) در صورت بزرگتر بودن از مقدار دقت محاسبات که ما تعیین کردیم به صورت یک فیدبک به وزنها اعمال می شود تا به یک حالت تعادل برسیم.

۱۹۸۲- توسط Kohenen

ارائه متد :شبکه خود ساز(Self Organizing Networks)

در اینجا هم از یک سری سیستمهای رقابتی استفاده شده است .

۱۹۸۶- توسط Rumelhart
در مورد پردازش موازی گسترده
الگوریتم پس انتشار خطا(Error Back Propagation)

۱۹۸۷- توسط Winter-Widrow -Boxter
تکنیک های جدید ومتفاوتی برای مسئله تطبیقی بودن شبکه های عصبی

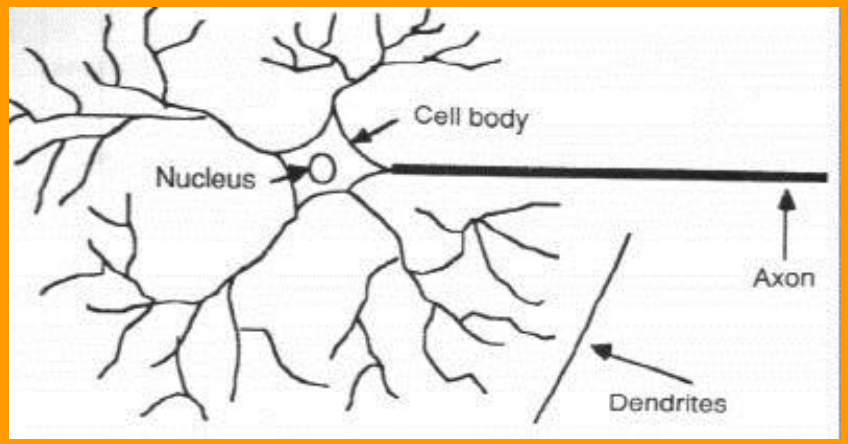
۱۹۸۸- ارائه مدل های *MadlineI , II*

معرفی سلول عصبی:

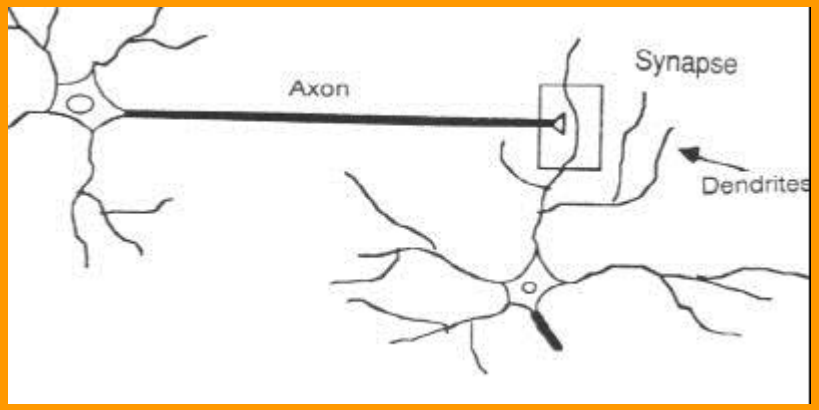
شبکه‌های عصبی را می‌توان با اغماض زیاد، مدل‌های الکترونیکی از ساختار عصبی مغز انسان نامید وپایه مدلهای تحلیل عصبی بر شبیه‌سازی فعالیتهای یک سلول عصبی (نرون) استوار است.

بافت‌هایی که عصب نامیده می‌شوند، اجتماعی از نرونها می‌باشند.

بیشتر نرون‌ها از سه قسمت اساسی تشکیل شده‌اند:
۱–بدنه سلول: که شامل هسته و قسمتهای حفاظتی دیگر می‌باشد.
۲–دندریت
۳–آکسون



محل تلاقي يك اكسون از يك سلول به دندريتهاي سلول ديگر را سيناپس گويند.



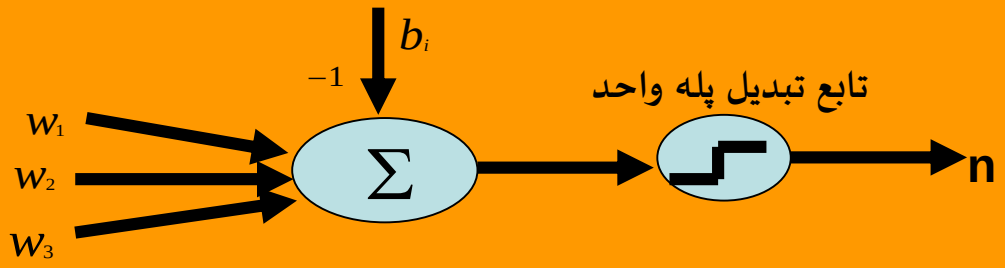
نرون‌ها براساس ساختارهايي كه بين آنها پيامها هدايت مي‌شود به‌سه دسته تقسيم مي‌گردند:

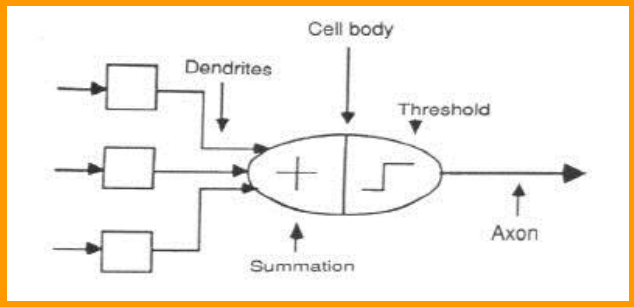
۱-نرونهاي حسي كه اطلاعات را از ارگانهاي حسي به مغز و نخاع مي‌فرستند

۲-نرون‌هاي محرك كه سيگنال‌هاي فرمان را از مغز و نخاع به ماهيچه و غدد هدايت مي‌كنند.

۳-نرون‌ها ارتباطي كه نرونها را به هم متصل مي‌كنند.

مدل پايه براي يك سلول i ام در ارتباط با سلولهاي ديگر:





نرون مصنوعي -مدل Mc Cullogh &Pits

bias

$$n_i(t+1) = f\left(\sum_j w_{ij}\, n_j(t) - b_i\right)$$

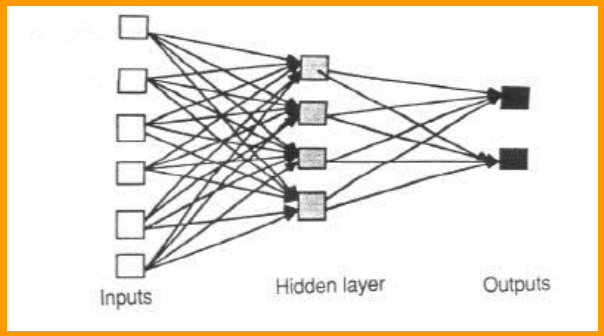
f تابع پله واحد=



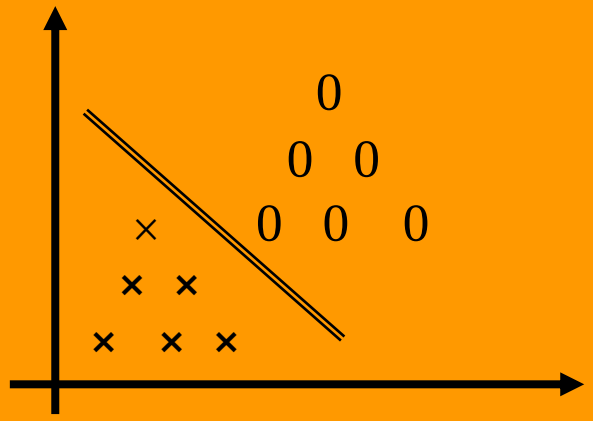
b_i : آستانه نرون i

قدرت سيناپسي ميان سلول i ام و سلول j

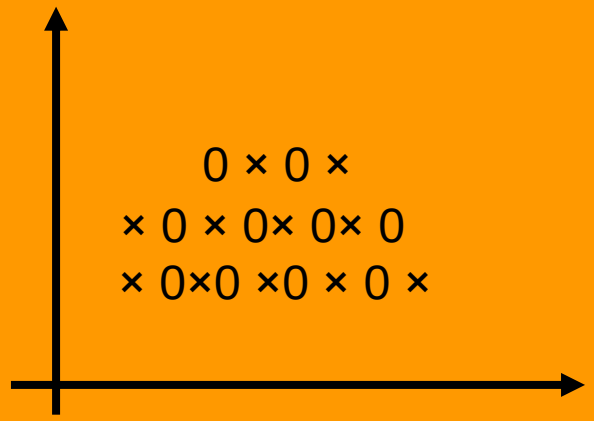
هر لايه بايد تمام سلولهايش يك ساختار داشته باشد ولي ساختار لايه ها ممكن است با يكديگرتفاوت داشته باشد (يك لايه نسبت به لايه ديگر)



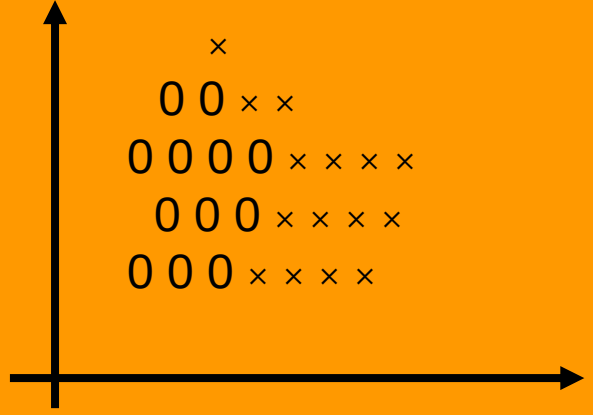
بدليل ۰و۱ بودن خروجي(خروجي باينري) به عنوان يك سيستم دسته بندي مثلا"A و B(بودن يا نبودن) در نظر گرفت. اين رابطه مي تواند الگوها را بصورت خطي تفكيك كند مثلا":



قابليت تفكيك دارد

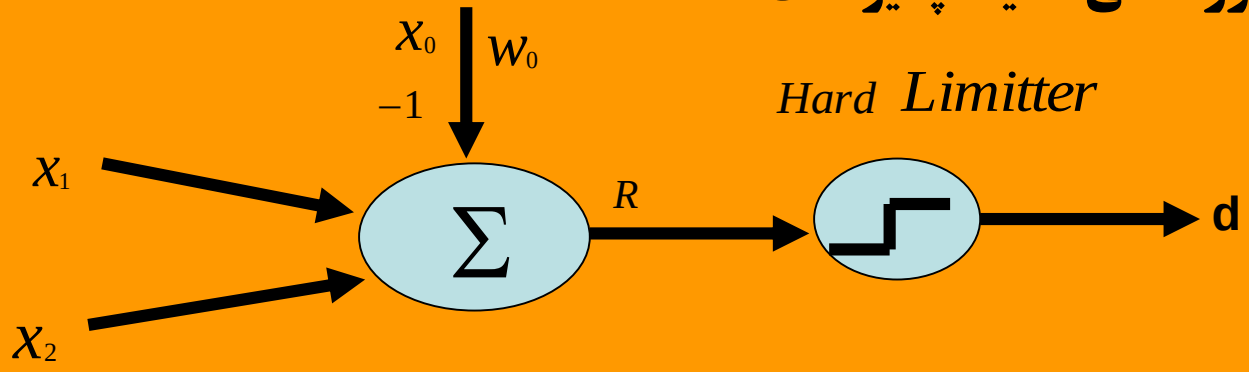


قابليت تفكيك ندارد

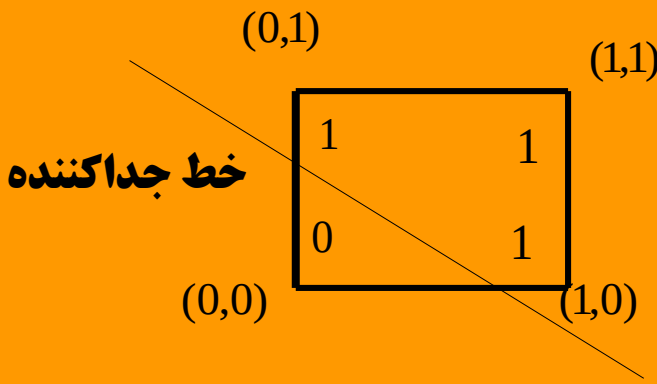


اگر همچنین الگو به شکل زیر باشد
قابلیت تفکیک پذیری را دارد ولی نه با مدل پایه
وساده نرون (بطور خطی تفکیک پذیر نیست)

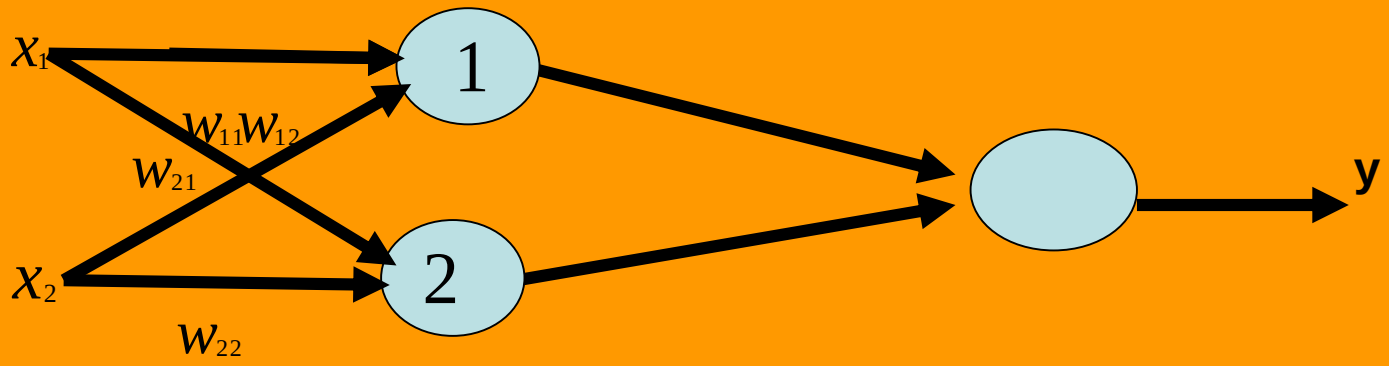
بعنوان مثال تابع منطقی OR بطور خطی تفکیک پذیر است.

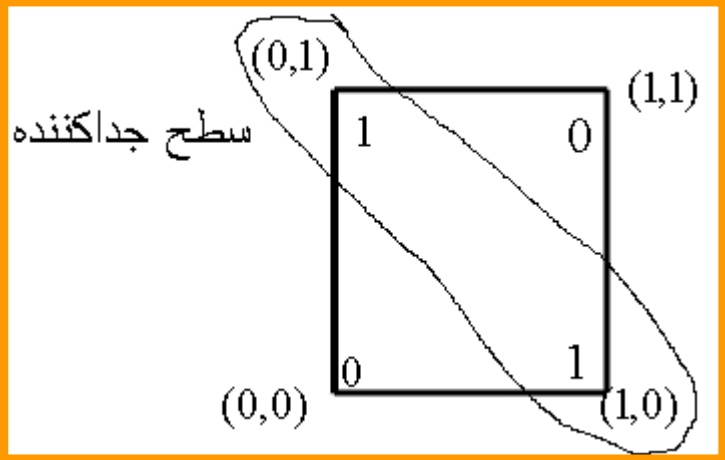


x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



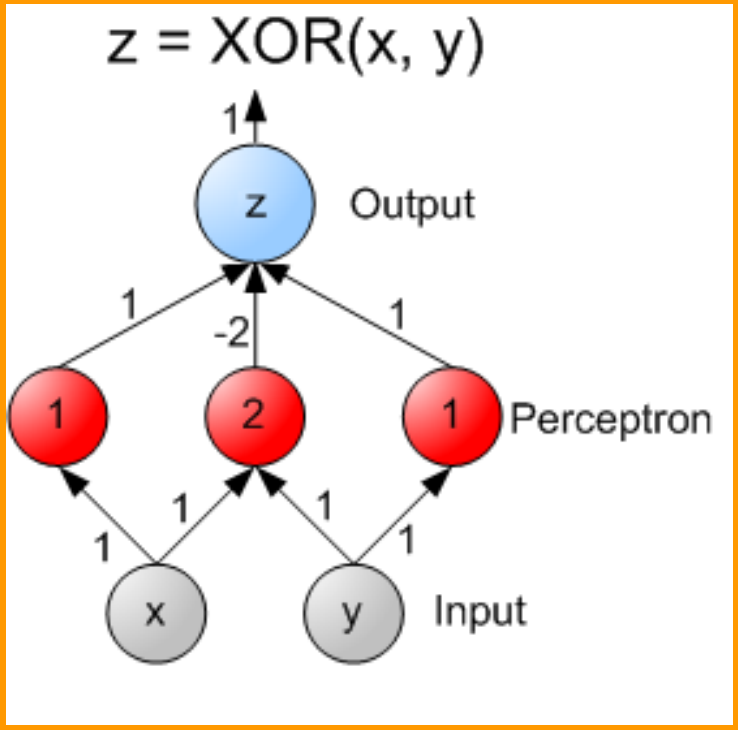
مثال: تابع منطقی XOR بطور خطی تفکیک پذیر است.





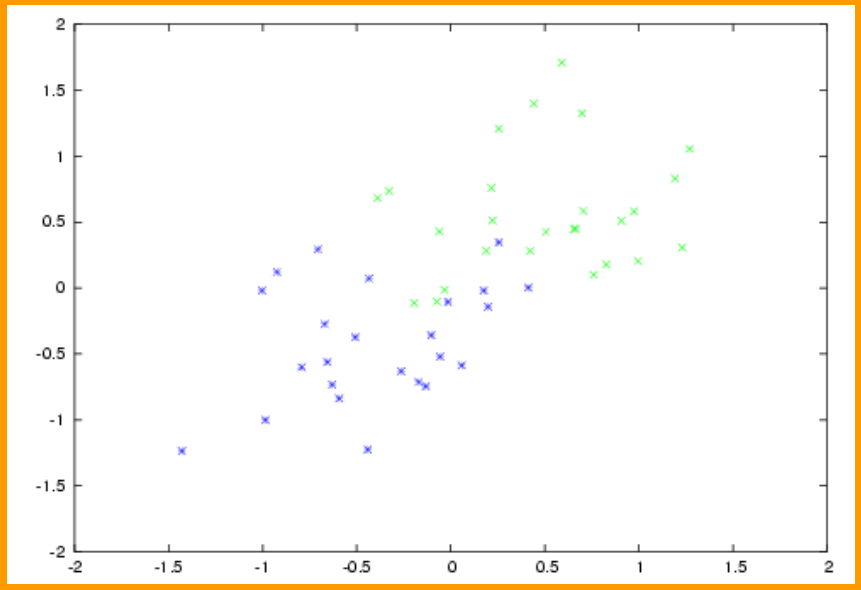
A two-layer neural network capable of calculating XOR. The numbers within the neurons represent each neuron's explicit threshold (which can .(ʅbe factored out so that all neurons have the same threshold, usually .stupni eht fo thgiew eht tneserper sworra etatonna taht srebmun ehT si (ʅ- ton) orez ,dehcaer ton si dlohserht eht fi taht semussa ten sihT a deredisnoc syawla ton si stupni fo reyal mottob eht taht etoN .tuptuo

layer krowten laruen laer

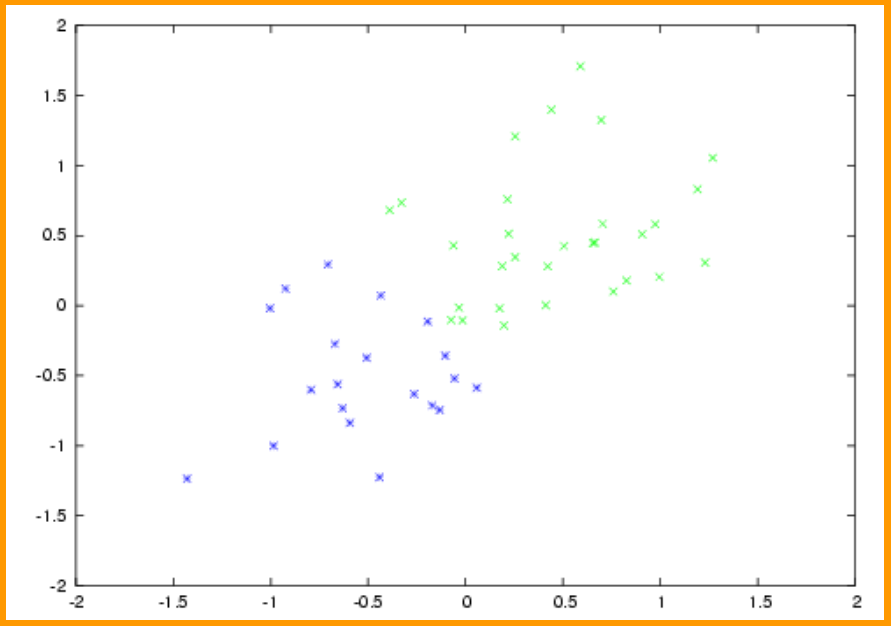


example, consider the case of having to classify data into two classes. Here is a small such data set, consisting of two points coming from two

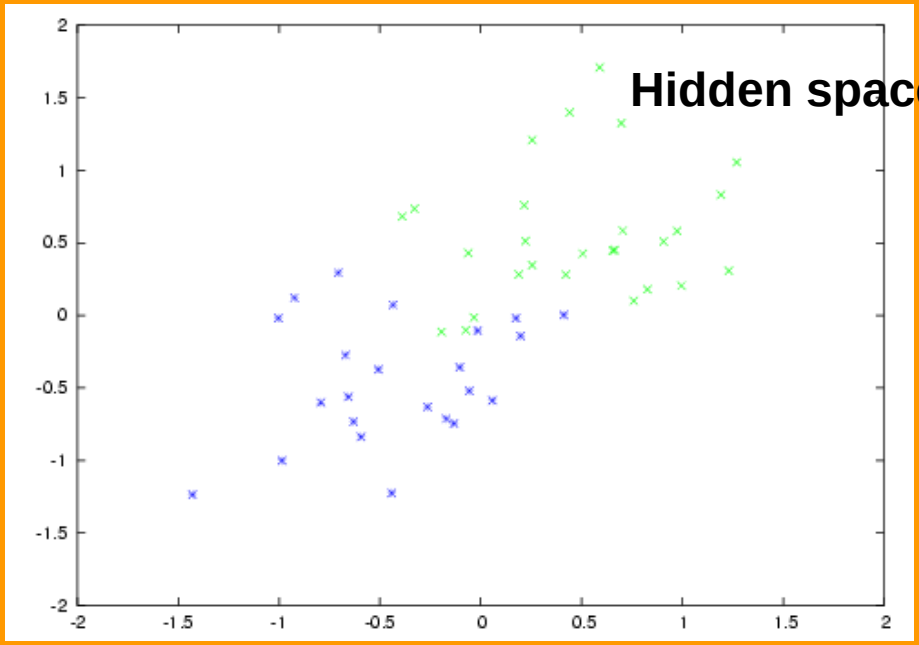
Gaussian distributions.



Two class Gaussian data



Linear classifier on Gaussian data



Hidden space linear classifier on Gaussian data

A linear classifier can only separate things with a hyperplane, so it's not possible to perfectly classify all the examples
A linear classifier can then separate the data, as shown in the third figure. However the data may still not be completely separable in this

space, in which the perceptron algorithm would not converge

قابلیتهای مغز:

مغز به عنوان یک سیستم پردازش اطلاعات با ساختار موازی و کاملا پیچیده که ۲درصد وزن بدن را تشکیل می‌دهد و بیش از ۲۰ درصد از کل اکسیژن مصرفی بدن را مصرف می‌کند برای خواندن، تنفس، حرکت، تفکرو تفحص و کلیه اعمال آگاهانه وبسیاری از اعمال ناآگاهانه استفاده می‌شود. اینکه مغز چگونه این اعمال را انجام می‌دهد از زمانی شروع شد که دریافتند مغز برای محاسبات خود اساساً از ساختاری کاملاً مغایر با ساختار کامپیوترهای متداول برخوردارمی‌باشد. تلاش برای فهم موضوع فوق خصوصاً از زمانی مطرح شد که برای نخستین بار در سال ۱۹۱۱ شخصی به نام سگال (Segal) اعلام نمود که مغز از عناصر ساختاری به نام نرون (Neuron)تشکیل یافته است.



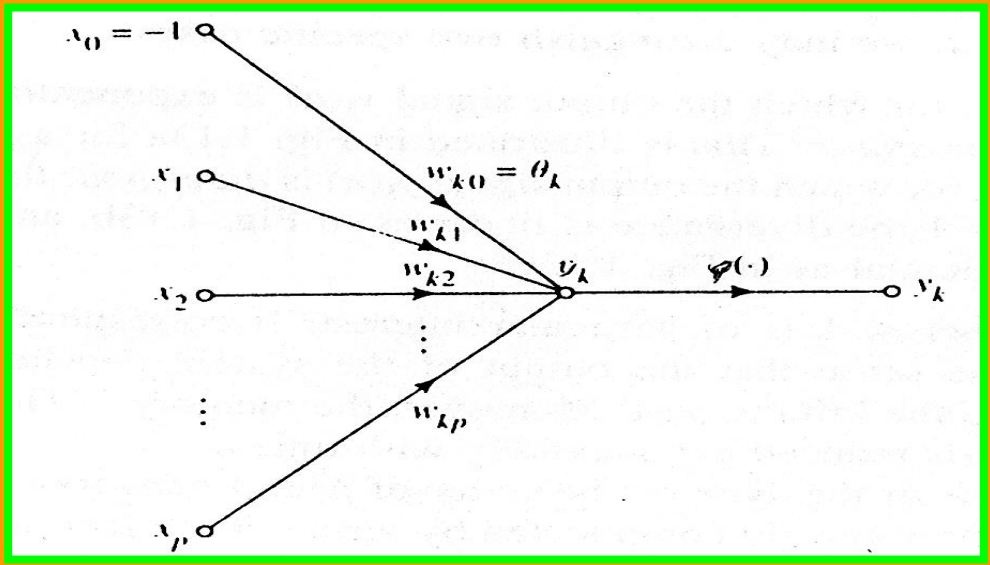
نمودار گذر سیگنال

(signal-flow-graph)

نمودار گذر سیگنال شامل تعدادي شاخه وگره مي باشد

هر شاخه داراي يك تابع تبديل است

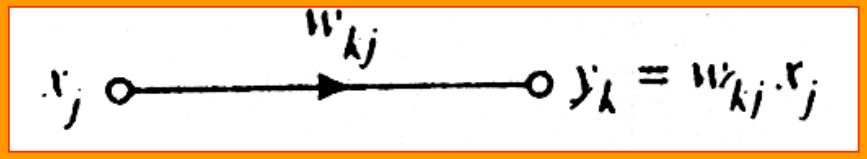
هر گره داراي يك سيگنال خروجي مي باشد



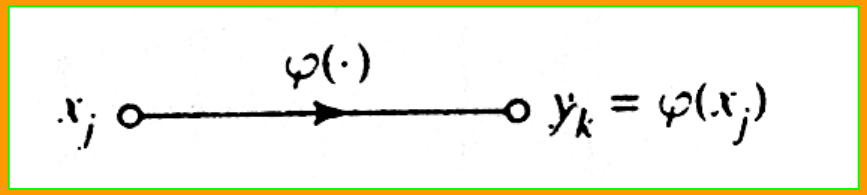
قاعده اول: مسير عبور سيگنال در يك شاخه در جهت فلاش مي باشد

دو نوع شاخه وجود دارد

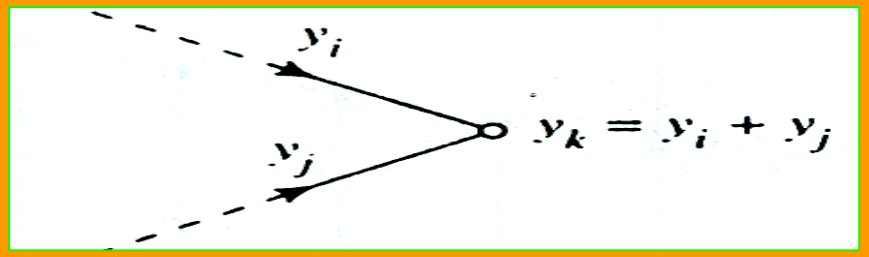
شاخه اتصالي كه نسبت خروجي و ورودي آن خطي است



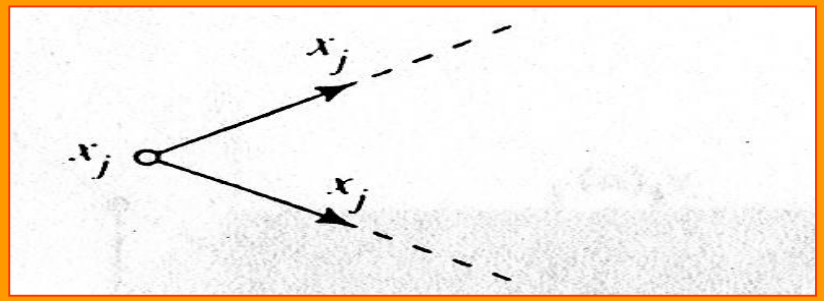
شاخه كنش داركه نسبت خروجي و ورودي آن غير خطي است



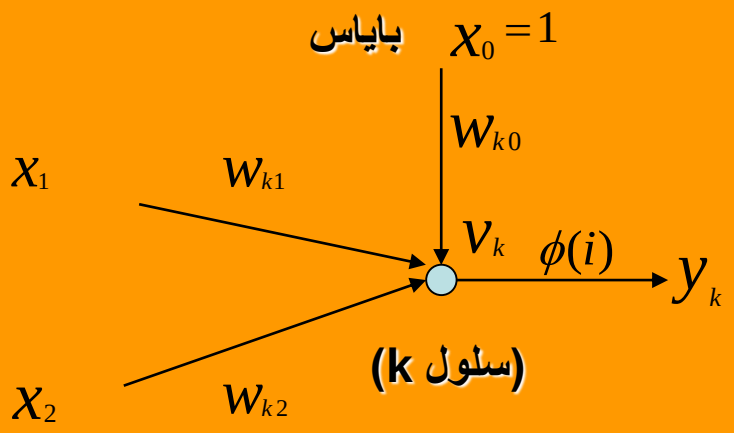
قاعده دوم :سيگنال خروجي يك گره برابر با جمع جبري تمام سيگنالهاي وارد به آن گره مي باشد



قاعده سوم :سيگنال خروجي ازيك گره به تما م شاخه هاي متصل به آن گره بطور يكسان ومستقل ازيكديگرانتقال مي يابد.



به عنوان مثال نمودار گذر سيگنال يك سلول)نرون(عصبي:



در روش آموزش ' از روش تکرار استفاده مي شود

$w_{kj}(n) =$ مقدار وزن در لحظه n
با تکرار nام

$\Delta w_{kj}(n) =$ مقدار تغییرات دروزن در
تکرار nام

$w_{kj}(n+1) =$ مقدار وزن جدید یا
مقدار وزن در لحظه n+1

ما دراين محاسبات يك خروجي واقعي داريم كه همان خروجي در فرآيند محاسبات است . ولي ما يك خروجي مطلوب مي خواهيم كه اين كار از طريق تفاوت خروجي واقعي ومطلوب كه يا صفر است ويا مقدار در نظر مي گيريم. اگر به اين وضعيت نرسيديم بايستي مقدار وزنها را تغيير داد تا به اين خروجي مطلوب دست يافت.

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

دوسلول $$y_k = \phi\left(\sum_j w_{kj} x_j\right)$$

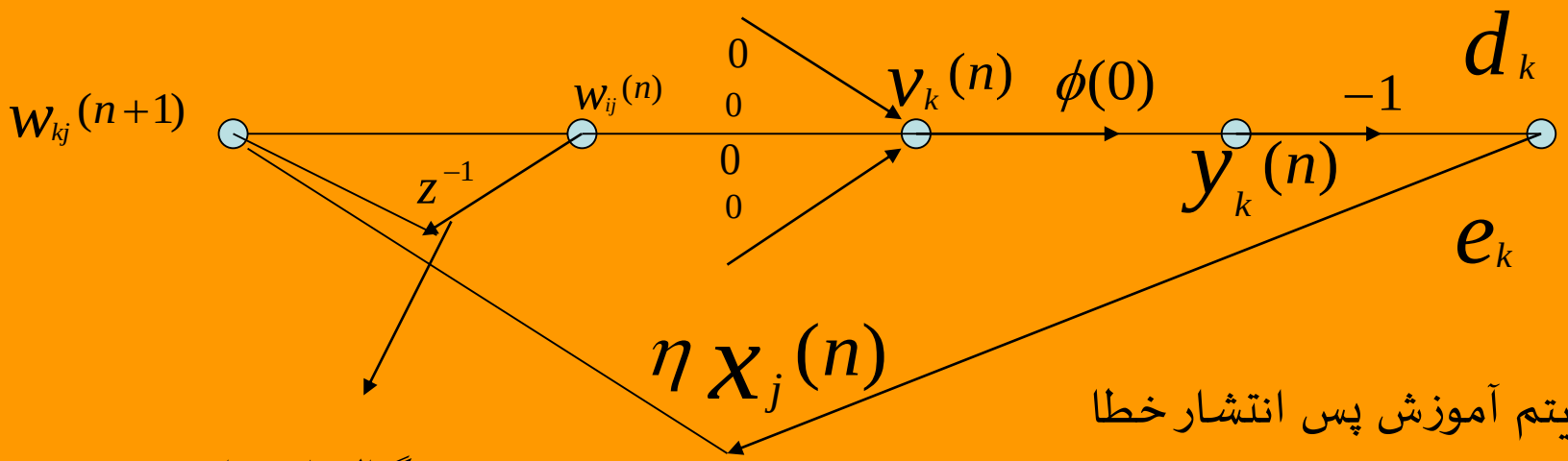
$$\Delta w_{kj}(n) = \eta \cdot e_k(n) \cdot x_j(n) \quad (\eta < 1)$$

سيگنال خطا (error-signal) $$e_k(n) = d - o$$

d: خروجي مطلوب

o: خروجي واقعي

تنها ارتباط يك ورودي را به سلولk در نظر گرفتيم.



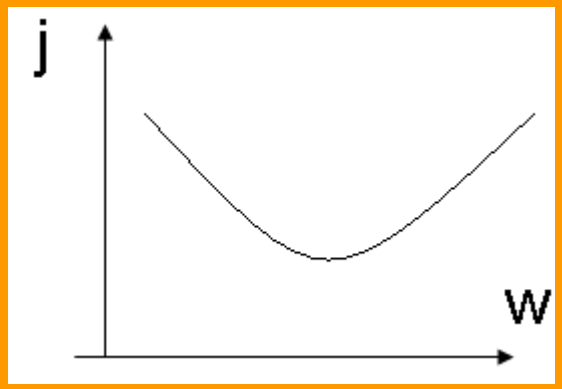
الگوريتم آموزش پس انتشار خطا

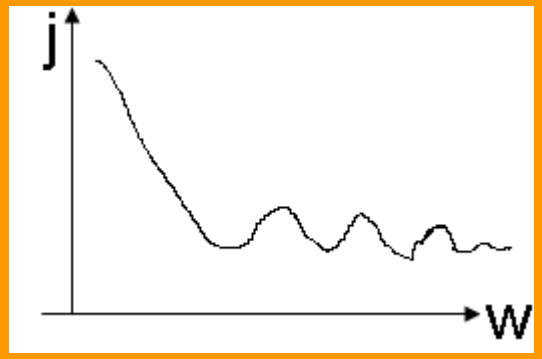
سيگنال واحد تاخيري

$$z^{-1}.[w_{kj}(n)] = w_{kj}(n+1)$$

استفاده از تابع هزينه:

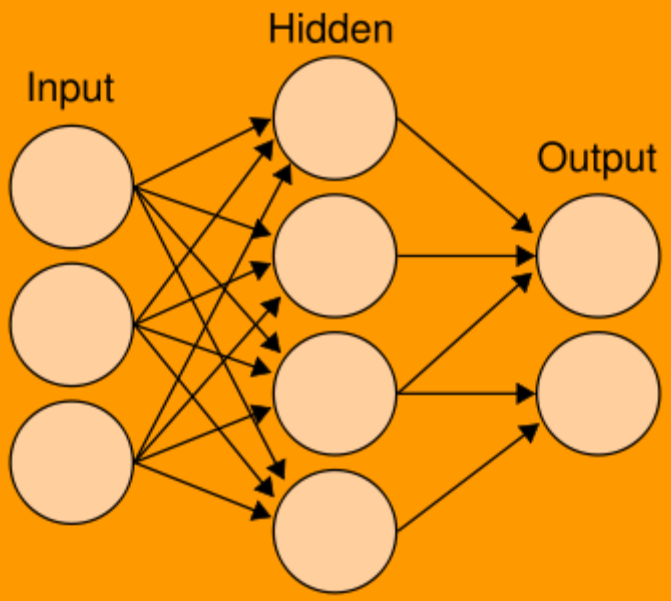
براي آموزش





local min حداقل محلي

Global min حداقل مطلق


Input
Hidden
Output

فصل دوم

فرایند یادگیری

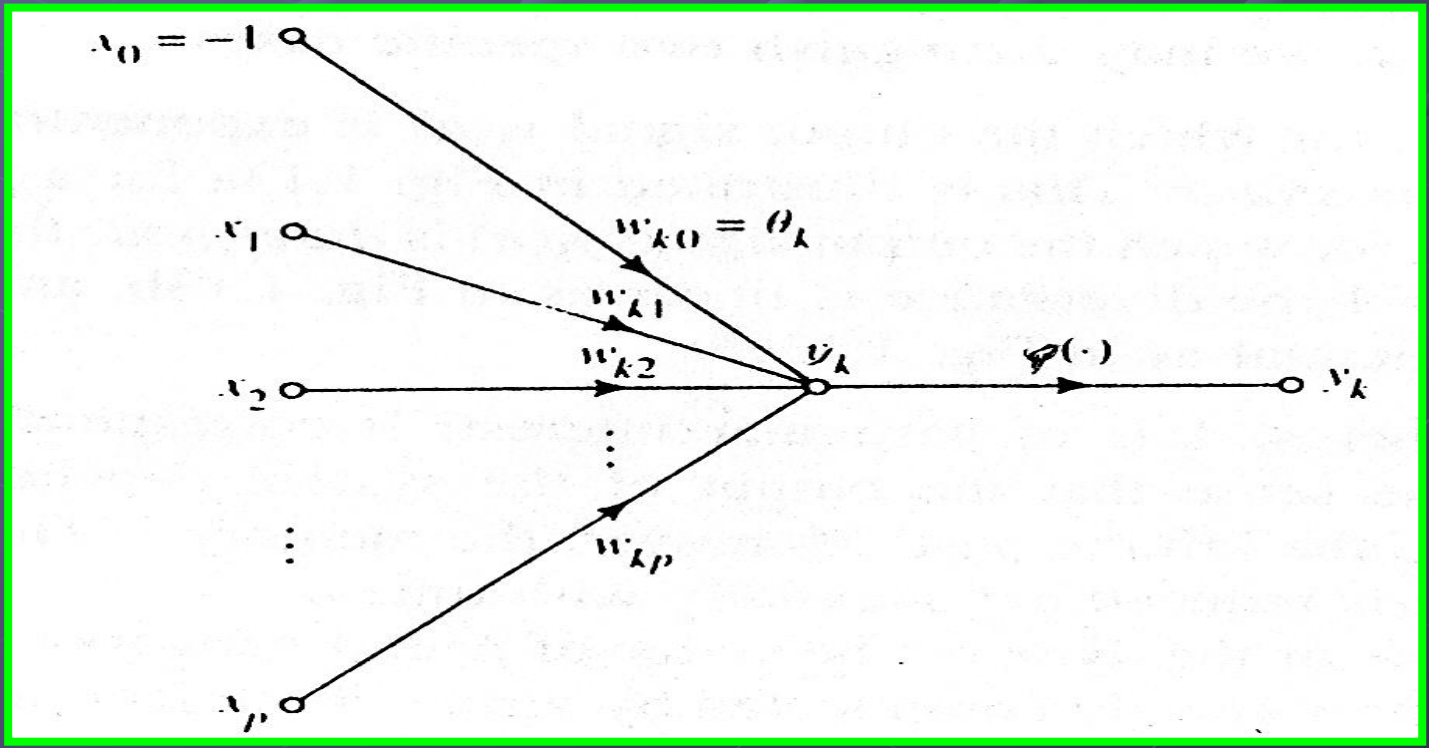
در این فصل بصورت خلاصه انواع روشهای یادگیری شبکه های عصبی مورد بررسی قرار می گیرد و مطالب تکمیلی در فصل های آینده ارایه می گردد

نمودار گذر سیگنال

نمودار گذر سیگنال شامل تعدادی شاخه وگره می باشد

هر شاخه دارای یک تابع تبدیل است

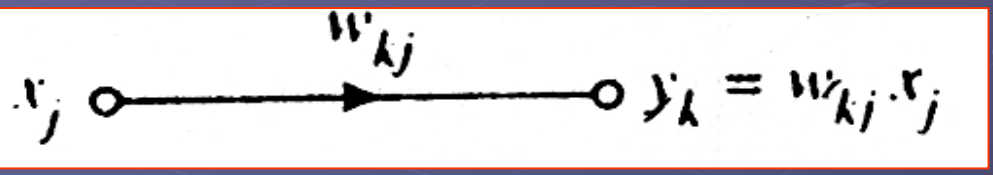
هر گره دارای یک سیگنال خروجی می باشد



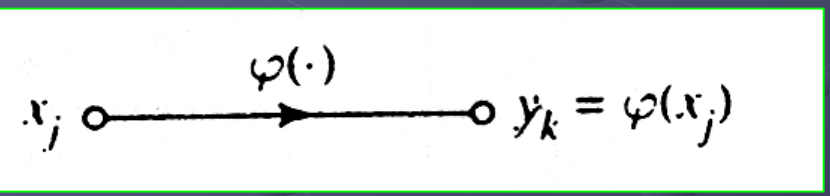
قاعده اول: مسير عبور سيگنال در يك شاخه در جهت فلاش مي باشد

دو نوع شاخه وجود دارد

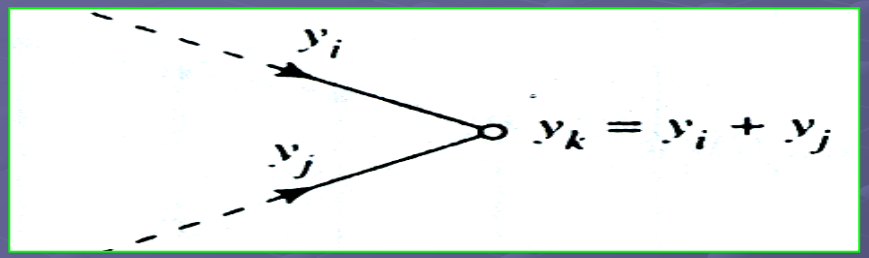
شاخه اتصالي كه نسبت خروجي و ورودي آن خطي است



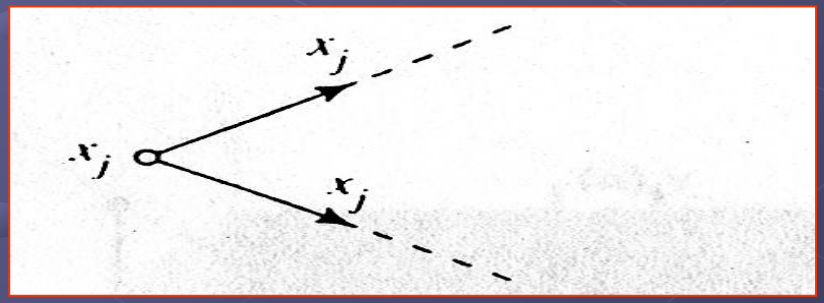
شاخه كنش داركه نسبت خروجي و ورودي آن غير خطي است



قاعده دوم :سيگنال خروجي يك گره برابر با جمع جبري تمام سيگنالهاي وارد به آن گره مي باشد



قاعده سوم :سيگنال خروجي از يك گره به تما م شاخه هاي متصل به آن گره بطور يكسان ومستقل از يكديگرانتقال مي يابد



فرایند یادگیری

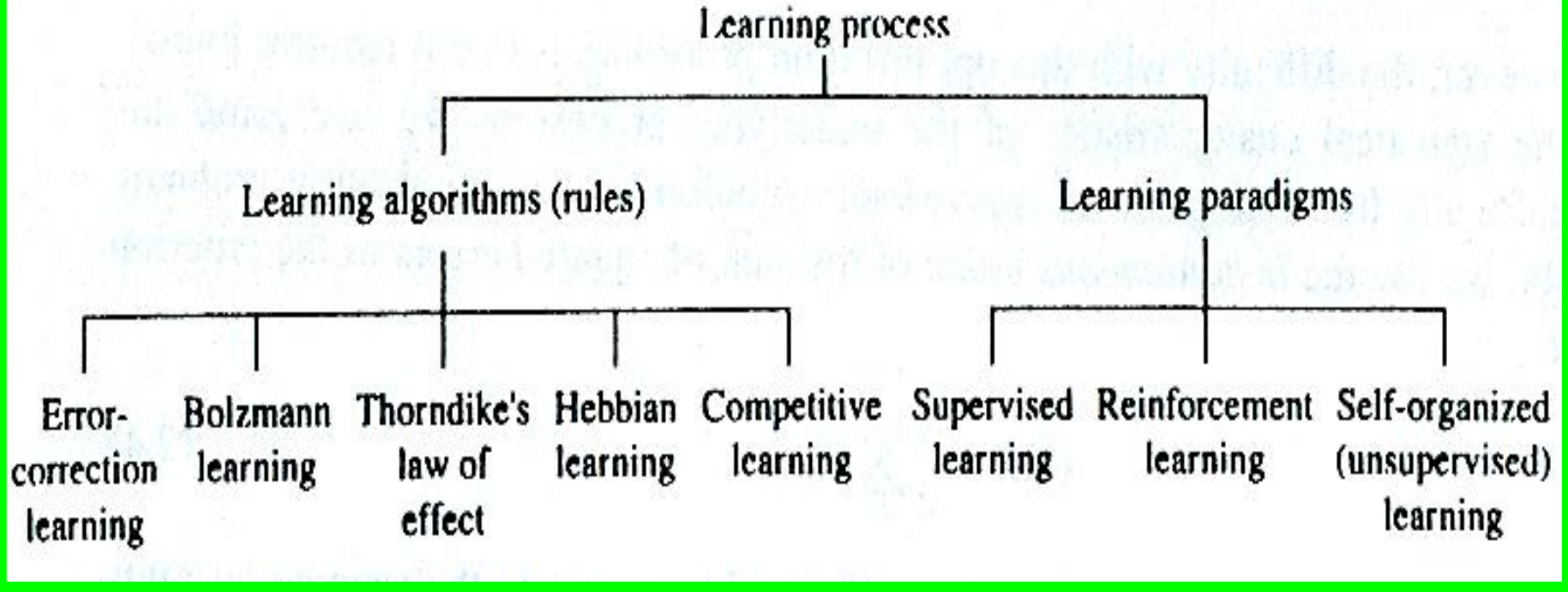
یاد گیری از محیط و بهبود بخشیدن به اطلاعات

یادگیری از طریق وزنها و مقادیر آستانه

انواع یادگیری و الگوریتم های تعلیم

به صورت کلی فراگیری به معنای میزان کردن مقادیر ضرایب وزنی می باشد

یک نمونه طبقه بندی از فرایند یادگیری



یادگیری تصحیح خطا

خروجي مطلوب

پاسخ شبکه

$$e_k(n) = d_k(n) - y_k(n)$$

میانگین مربعات خطا

تابع هزینه

$$J = E\left[\frac{1}{2}\sum_t e_k^2(n)\right]$$

مقدار تغییر وزن در لحظه n

مقدار وزن در لحظه n

مقدار تغییر وزن در لحظه n+1

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

ضریب یادگیری

$$\mathscr{E}(n) = \frac{1}{2}\sum_{i} e_i^2(n)$$

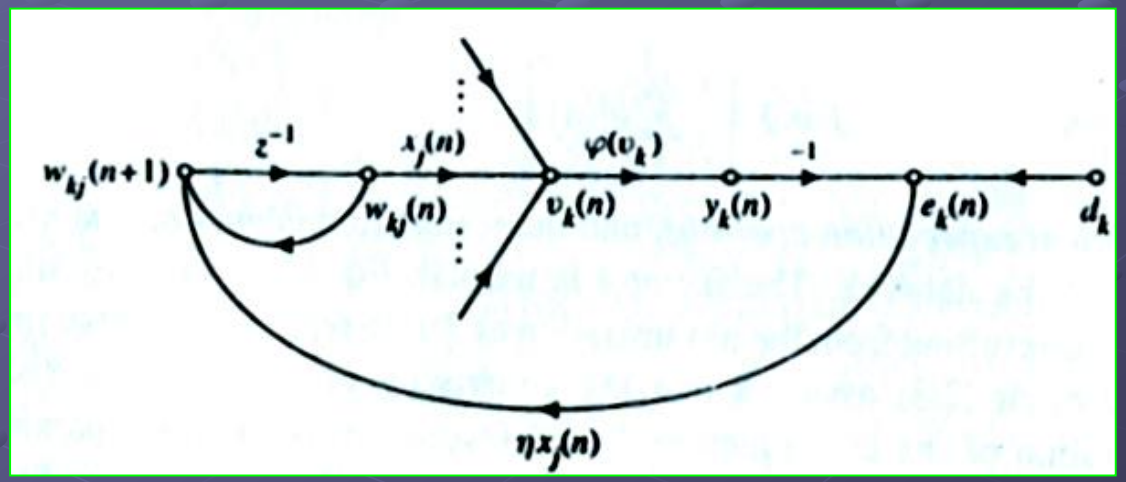
$$\Delta w_{ij}(n) = \eta e_i(n) x_j(n)$$

$$z^{-1}[w_{ij}(n+1)] = w_{ij}(n)$$

خروجی سلول j در اثر ورودی X(n)

$$v_k(n) = \sum_j x_j(n) w_{kj}(n)$$

$$y_k(n) = \varphi(v_k(n))$$

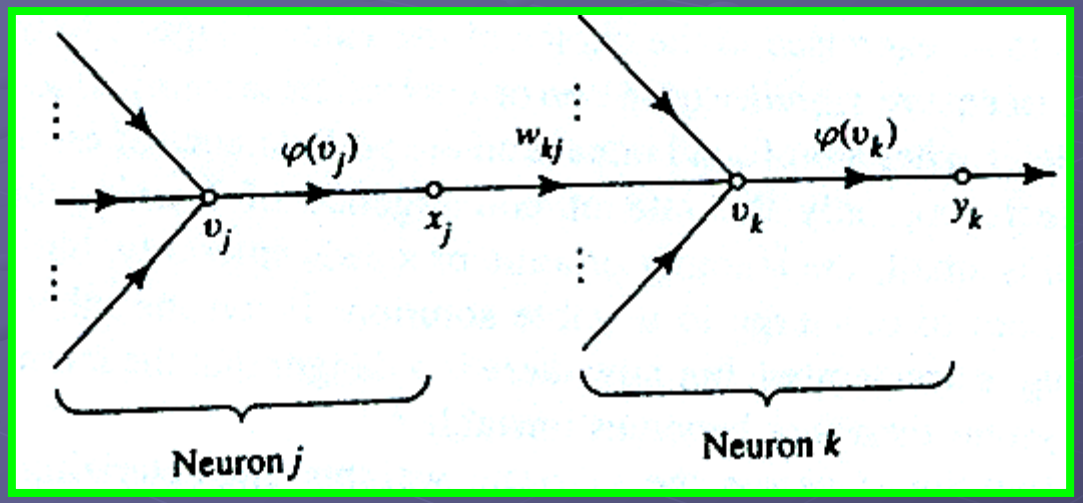


تعیین مقدار ضریب یادگیری برای تخمین پایداری در روشهای یادگیری از اهمیت خاصی برخوردار است

فراگیری هب

- ۱- اگر دو نرون در دو طرف یک وزن را به طورهمزمان تحریک نماییم در این صورت استحکام آن وزن تقویت می گردد.
- ۲- اگردو نرون در دو طرف یک وزن را به طورغیرهمزمان تحریک نماییم در این صورت استحکام آن وزن کاهش می یابد ویا آن وزن حذف می شود.

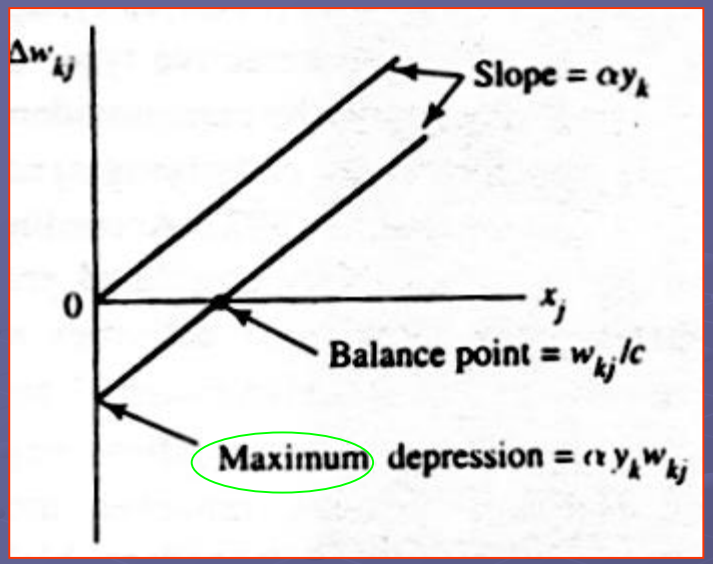
مدل ریاضی هب تغییریافته



$$\Delta w_{kj}(n) = F(y_k(n), x_j(n))$$

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$$

با مقداری تغییر، فرم تغییر یافته روش فراگیری هب بدست می آید



$$\Delta w_{kj}(n) = \eta \operatorname{cov}[y_k(n), x_j(n)]$$
$$= \eta E[(y_k(n) - \bar{y}_k)(x_j(n) - \bar{x}_j)]$$

$$\Delta w_{kj}(n) = \eta\{E[y_k(n)x_j(n)] - \bar{y}_k\bar{x}_j\}$$

محدودیتها

وزنها نمی توانند به طوردلخواه بزرگ انتخاب شوند

تغییرات در وزنها ازیک دسته ورودی به دسته دیگر بصورت آهسته صورت می گیرد

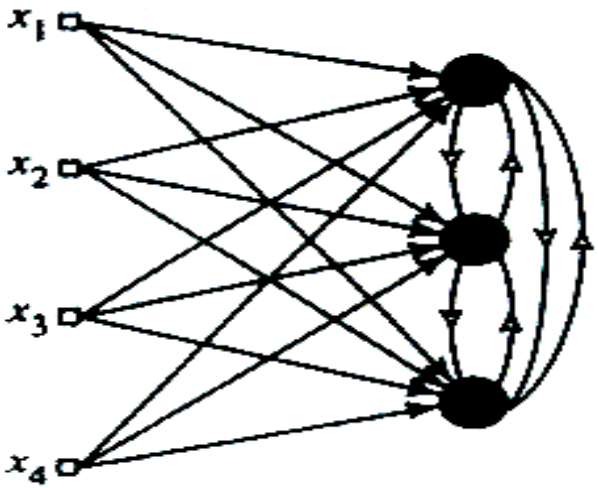
تغییر در وزن ها بصورت لایه به لایه انجام می شود

فراگیری رقابتی

خروجی های سلولهای عصبی با یکدیگربه رقابت می پردازند تا یکی از آنها که دارای امتیاز بیشتری است برنده وازسایر سلولها متمایز شود

این فرایند در لایه دوم صورت می گیرد

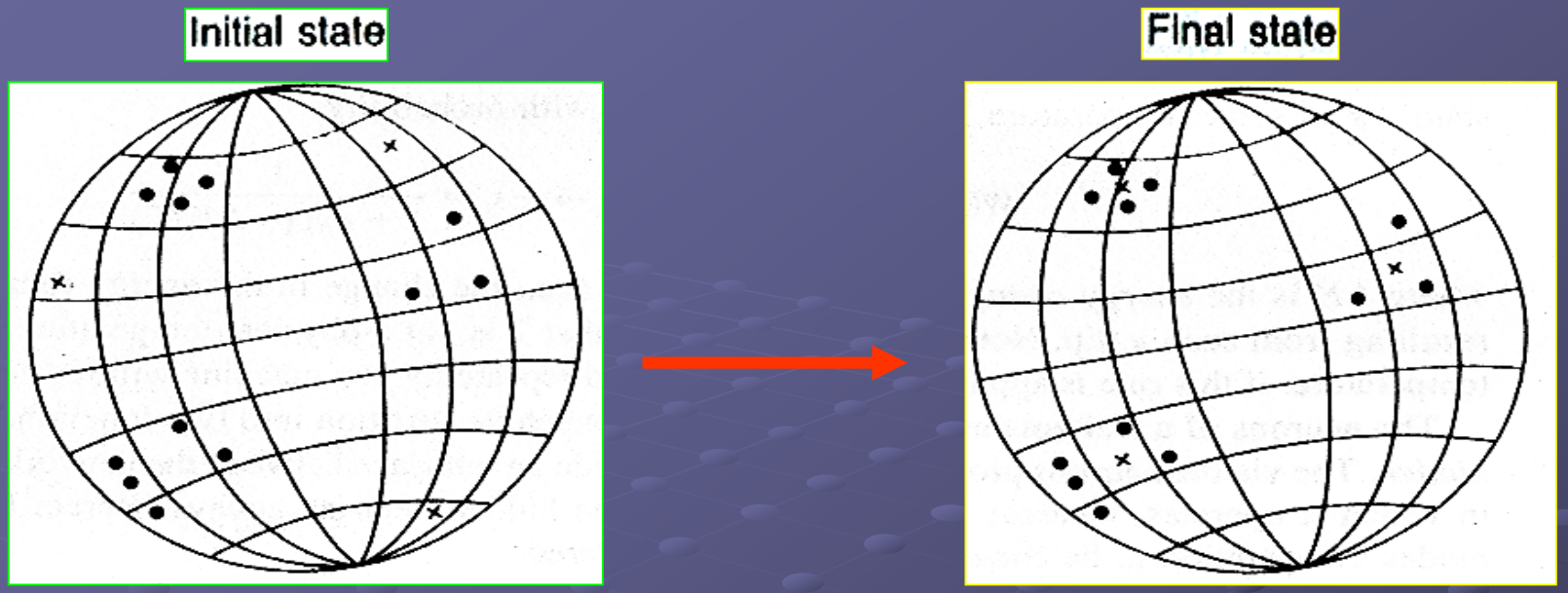
لایه دوم ، لایه رقابت بین سلولها وتعیین سلول برنده می باشد



$$\sum_i w_{ji} = 1 \qquad \text{for all } j$$

$$\Delta w_{ji} = \begin{cases} \eta(x_i - w_{ji}) & \text{if neuron } j \text{ wins the competition} \\ 0 & \text{if neuron } j \text{ loses the competition} \end{cases}$$

$$\sum_j w_{ji}^2 = 1 \qquad \text{for all } j$$



هر قانون یادگیری دارای سه مؤلفه زیر است

مجموعه سلولهایی که به یک گروه از ورودیها پاسخهای مختلف می دهند این ویژگی ناشی از توزیع تصادفی وزنهای سلول می باشد

مکانیسمی که فرصت رقابت را در اختیار سلولها قرار می دهد. این فرصت تمایز یک سلول (سلول برنده) از دیگر سلولها را فراهم می کند

وجود یک حد اشباع روی خروجی هر سلول

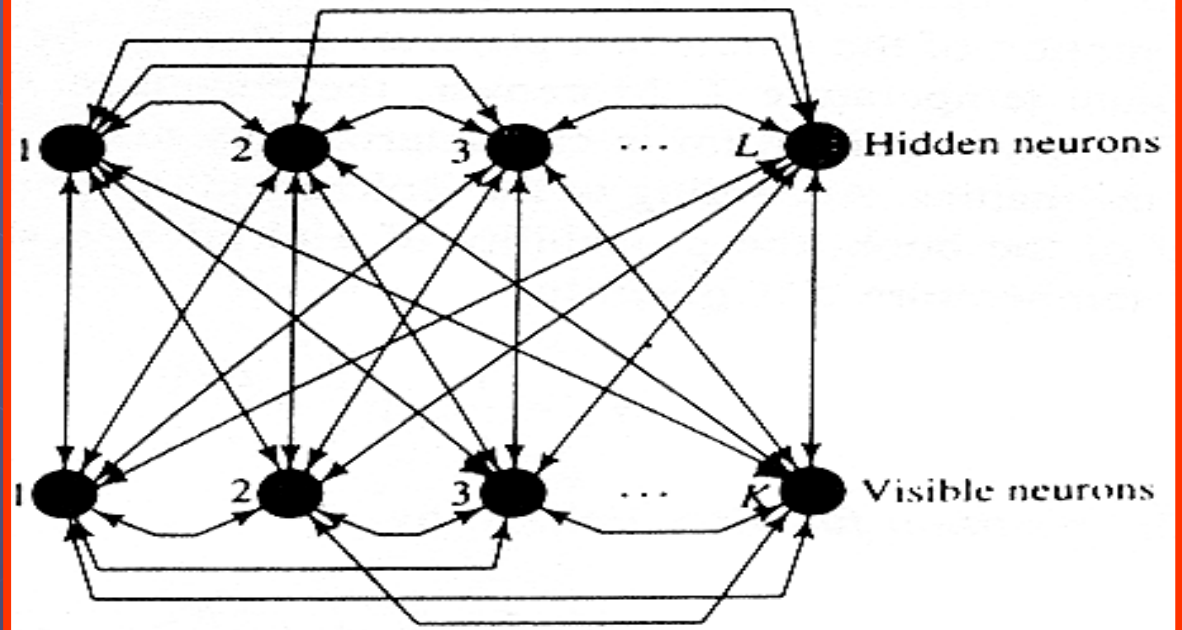
فراگیری بولتزمن

اهداف:

کلاسه بندی الگوهای ورودی بر طبق تئوری بولتزمن

تعیین یک شبکه با حالت های اتفاقی

رسیدن به تعادل حرارتی



حالتها

تابع انرژي

$$E = -\frac{1}{2}\sum_{i}\sum_{j(\neq i)} w_{ji}s_j s_i$$

تغييرات انرژي ماشين

$$W(s_j \rightarrow -s_j) = \frac{1}{1 + \exp(-\Delta E_j/T)}$$

احتمال تغيير حالت سلول

درجه حرارت كاذب

هرکدام از بردارهای ورودی به مدت کافی به شبکه اعمال می شوند تا به حالت تعادل حرارتی برسند

بردارهای ورودی و خروجی بصورت اتفاقی به شبکه اعمال می شوند

حالت سلول

احتمال مشترك در وضعيت مفيد

$$\rho_{ji}^{+} = \sum_{\alpha}\sum_{\beta} P_{\alpha\beta}^{+} S_{j|\alpha\beta} S_{i|\alpha\beta}$$

$$\rho_{ji}^{-} = \sum_{\alpha}\sum_{\beta} P_{\alpha\beta}^{-} S_{j|\alpha\beta} S_{i|\alpha\beta}$$

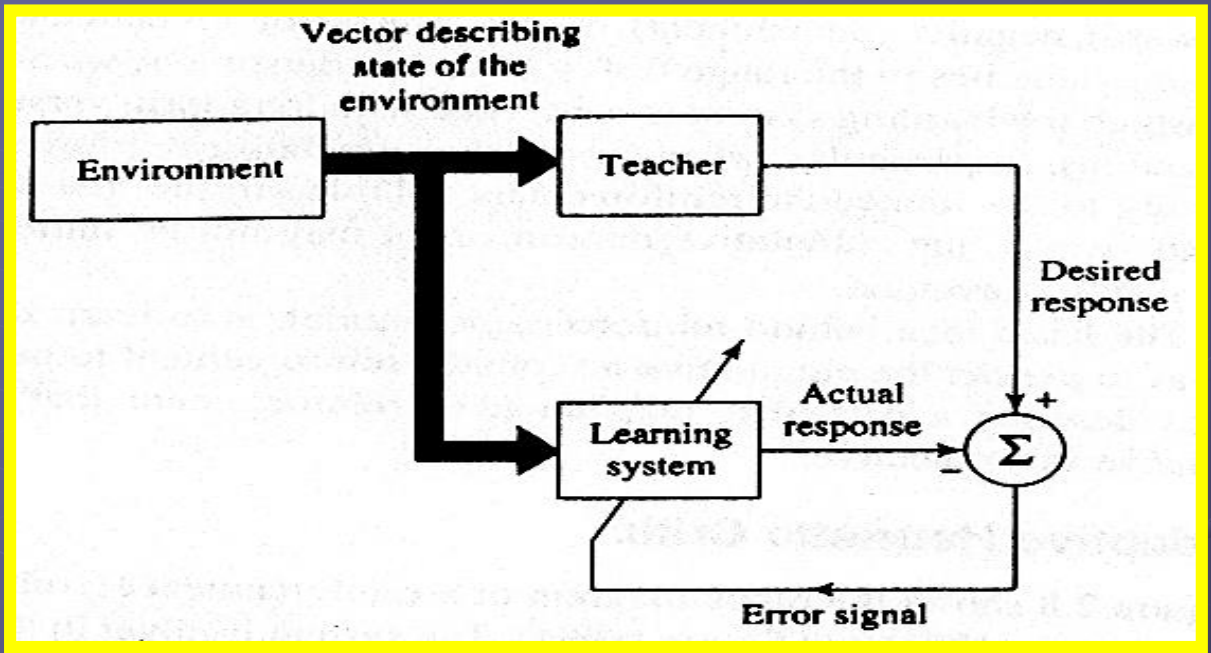
همبستگي بدون شرط

احتمال مشترك در وضعيت آزاد

همبستگي بين دو حالت، زماني كه سلول هاي آشكار در وضعيت مفيد باشند

$$\Delta w_{ji} = \eta(\rho_{ji}^{+} - \rho_{ji}^{-}), \qquad i \neq j$$

فراگیری با ناظر



- معلم سیستمی است که بر محیط وقوف دارد
- محیط برای شبکه عصبی مجهول است
- جواب مطلوب توسط معلم به شبکه داده می شود
- شبکه عصبی توسط دو سیگنال ورودی و خطا تنظیم می شود

در یادگیری با ناظر فرض براین که در هر مرحله تکرار الگوریتم یادگیری ، جواب مطلوب سیستم یادگیرنده از قبل آماده است

همواره خطای یادگیری جهت بهبود رفتار سیستم یادگیرنده در دسترس می باشد

یادگیری توسط الگوریتم های برگشتی و با معادلات دیفرانسیل بیان می شود

یادگیری تقویتی Reinforcement

یک یادگیری آن لاین از یک نگاشت ورودی – خروجی است

یادگیری از طریق یک پروسه سعی وخطا انجام می گردد

اگر شبکه عصبی پارامترهایش را به گونه ای تغییر دهد که منجر به یک حالت مساعد شود انگاه تمایل سیستم یادگیرنده جهت تولید آن عمل خاص تقویت می گردد

یاد گیری تقویتی مانند یادگیری با ناظر نیست و در سیستم های کنترلی کار برد دارد

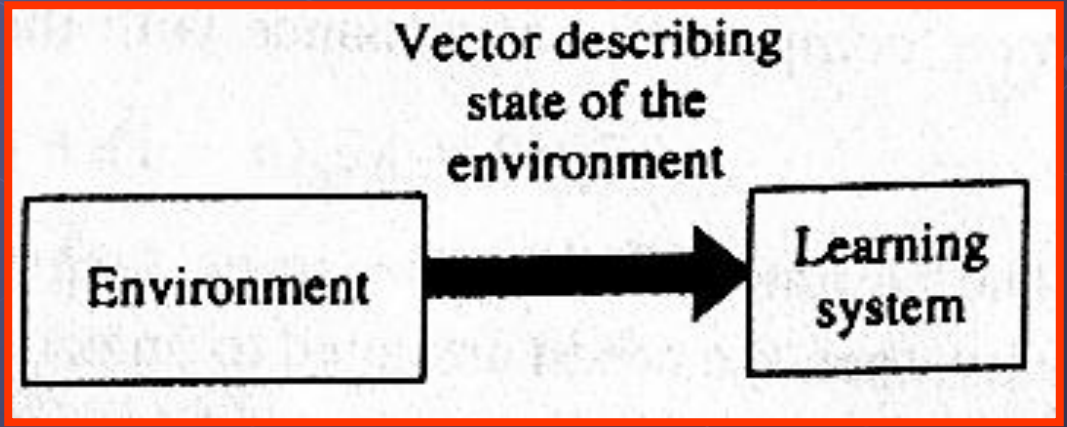
مشکل یادگیری با ناظرجهت تصمیم در مواضع جدید را حل می کند

فراگیری بدون ناظر یا خود سازمانده

پارامترهای شبکه عصبی تنها توسط پاسخ سیستم تنظیم می شود

تنها اطلاعات دریافتی از محیط به شبکه بردارهای ورودی راتشکیل می دهند

به شبکه عصبی هیچ نمونه ای از تابعی که قرار است بیاموزد داده نمی شود



انواع وظايف يادگيری

1- تقريب زدن(RBF) :

فرض كنيم يك تابع غير خطي(x)g=d باشد
ورودي و خروجي بصورت زير مي باشد
(X1,d2),(X2,d2),…,(Xn,dn)

2- همبستگي (Hopfiled):

الف-همبستگي اتوماتيك

الگوها را در شبكه ذخيره مي كنيم يكي از الگوها را به صورت ناقص ويانويزي به شبكه داده وانتظار داريم كه شبكه آن را بازسازي كند

ب- همبستگي غير اتوماتيك

خروجي مور نظر به شبكه اعمال مي شود

3-طبقه بندي الگو(MLP):

يك الگو را به شبكه داده ومي خواهيم آنراطبقه بندي كنيم

4- پيش گويي :

m مقدار گذشته سيگنال را به شبكه داده و مي خواهيم مقدار كنوني را بدست آوريم



5- كنترل عصبي:

يك سيستم ديناميكي غيرخطي ناشناخته را
مي خواهيم كنترل كنيم (مقايسه)

$$Plant=\{u(t),y(t)\}$$

$$Reftence\ Model=\{r(t),d(t)\}$$

هدف بدست آوردن $u(t)$ براي $t\geq 0$

$$\lim_{t\to\infty}|d(t)-y(t)|=0$$

فصل چهارم

پرسپترون

- پرسپترون به معنی سلول عصبی دارای ادراک می باشد و از perception(درک کردن) / perceptrop (ادراک گر) می آید.

- رزنبلات (Rosenblatt) ثابت کرد که اگر الگوهای ورودی ازدو گروه (کلاس) مختلف انتخاب شوند بطوریکه این دو کلاس به طور خطی قابل جدا کردن باشند ، در این صورت الگوریتم پرسپترون همگرا خواهد بود و جداسازی این دو کلاس توسط یک صفحه اجرا خواهد شد.

Single layer perceptron
شبکه تک لایه

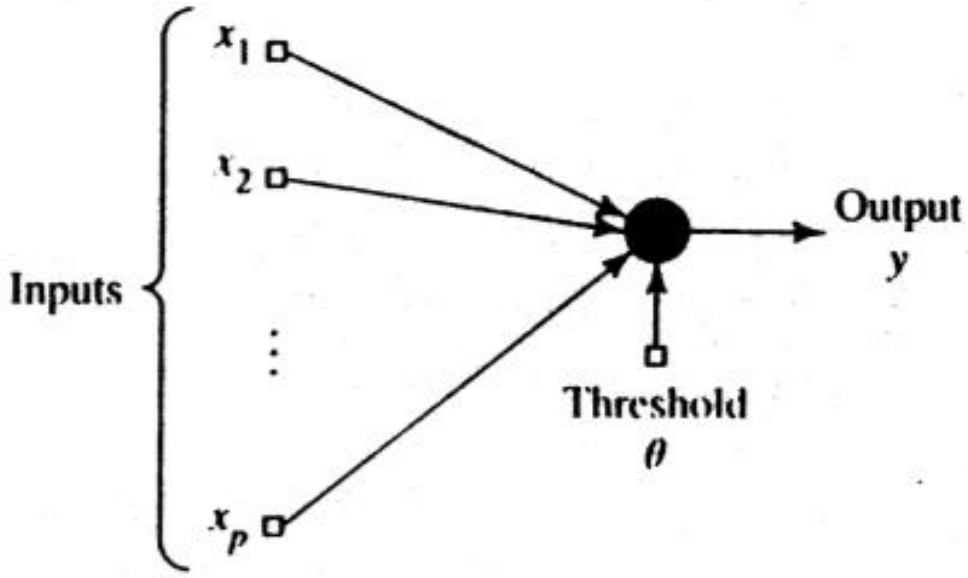


FIGURE 4.1 Single-layer perceptron.

Single flow graph perceptron

نمودار گذر سیگنال

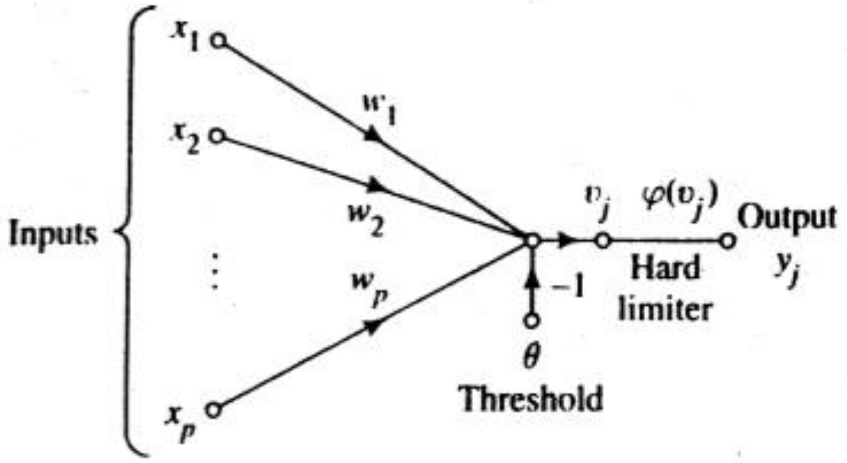


FIGURE 4.2 Signal-flow graph of the perceptron.

$$v = \sum_{i=1}^{p} w_i x_i - \theta$$

- اگر V_j مثبت باشد مقدار Y_j برابر ۱ است .
- اگر V_j منفی باشد مقدار Y_j برابر ۱- است.
- وظیفه این سلول جدا کردن سیگنال های ورودی به دو کلاس C_1 و C_2 است.
- اگر Y_j برابر ۱ باشد آنگاه X متعلق به کلاس C_1 است.
- اگر Y_j برابر ۱- باشد آنگاه X متعلق به کلاس C_2 است.

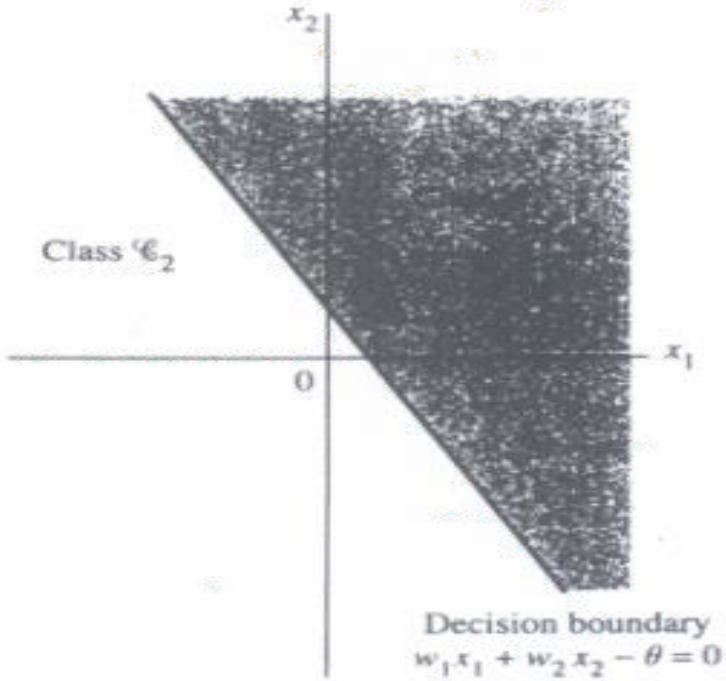


FIGURE 4.3 Illustrating linear reparability for a two-dimensional, two-class pattern-classification problem.

- جداکردن این دو کلاس توسط یک سطح با معادله

$$\sum_{i=1}^{p} w_i x_i - \theta = 0$$

صورت می پذیرد که به این سطح مرز تصمیم گیری گویند. (decision boundary)

- وزن های w_1 تا w_p را می توان بر اساس قاعده تصحیح خطا تنظیم کرد. به این قاعده الگوریتم همگرایی در پرسپترون می گویند.
- ابتدا تغییرات زیر در نمودار گذر سیگنال را در نظر می گیریم.

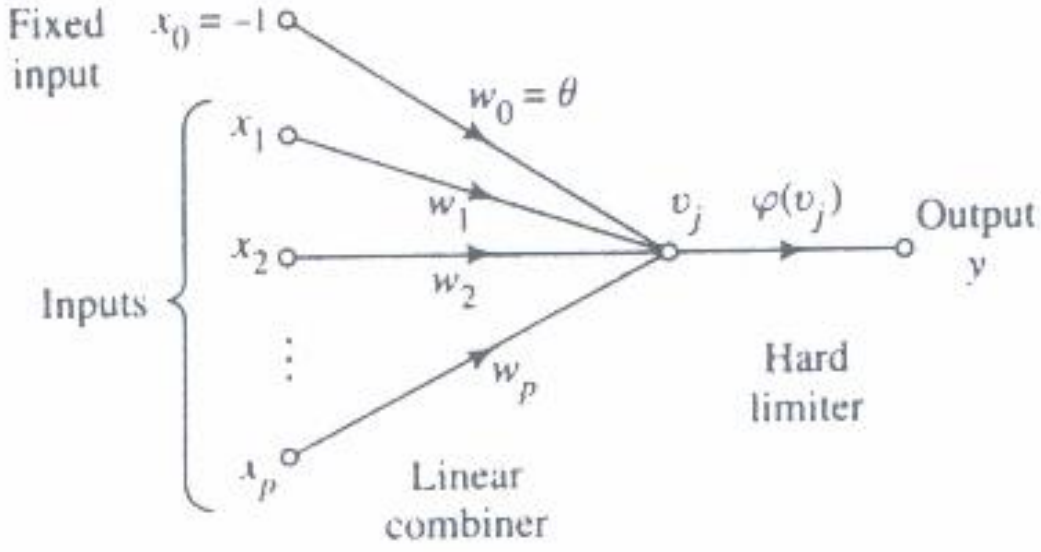


FIGURE 4.4 Equivalent signal-flow graph of the perceptron.

بنابراین

بردارهای ورودی

$$\mathbf{x}(n) = [-1, x_1(n), x_2(n), \ldots, x_p(n)]^T$$

وزن ها

$$\mathbf{w}(n) = [\theta(n), w_1(n), w_2(n), \ldots, w_p(n)]^T$$

$$v(n) = \mathbf{w}^T(n)\mathbf{x}(n)$$

برای مقدار ثابت n (یک لحظه معین) معادله

$$W^T X = 0$$

نشان دهنده معادله سطح خواهد بود که در مختصات p بعدی رسم شده است.

این سطح فضای ورودی را به دو قسمت تقسیم می کند:

۱- $W^T X > 0$ ⟵ بردار ورودی متعلق به کلاس C_1 است.

۲- $W^T X < 0$ ⟵ بردار ورودی متعلق به کلاس C_2 است.

بنابراین با داشتن دو دسته بردار ورودی X_1 و X_2 که به طور خطی قابل تفکیک هستند ، مسئله ما پیدا کردن بردار وزن W است بطوریکه شرایط ۱ و ۲ در بالا را برآورده سازد.

برای پیدا کردن بردار W :

۱- چنانچه در لحظه n بردار ورودی به طور صحیح کلاسه بندی شده باشد ، در این صورت هیچ گونه تغییر وزنی در W ایجاد نخواهد شد، بعبارت دیگر :

$$\mathbf{w}(n+1) = \mathbf{w}(n) \quad \text{if } \mathbf{w}^T(n)\mathbf{x}(n) \geq 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1$$

and

$$\mathbf{w}(n+1) = \mathbf{w}(n) \quad \text{if } \mathbf{w}^T(n)\mathbf{x}(n) < 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2$$

۲- در غیر این صورت :

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n) \quad \text{if } \mathbf{w}^T(n)\mathbf{x}(n) \geq 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2$$

and

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n) \quad \text{if } \mathbf{w}^T(n)\mathbf{x}(n) < 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1$$

$\eta(n)$: ضریب یادگیری (learning rate parameter) : مقدار **تغییرات در وزن در هر مرحله را کنترل می کند.**

چنانچه $\eta(n) = \eta > 0$ ⟵ قاعده افزایش ثابت.

اکنون ثابت می کنیم که قاعده بالا همگراست و تعداد دفعات تکرار ثابت است.

اثبات

فرض می کنیم:

۱- $W(1) = 0$

۲- $W^T(n)\, X(n) < 0$ در حالیکه $X(n)$ متعلق به کلاس C_1 (زیر مجموعه X_1) باشد.

۳- $\eta = 1$

بنابراین

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mathbf{x}(n)$$

$$\mathbf{w}(n+1) = \mathbf{x}(1) + \mathbf{x}(2) + \cdots + \mathbf{x}(n)$$

از آنجاییکه فرض شده است دو کلاس C_1 و C_2 به طور خطی قابل جدا کردن هستند ، بنابراین همواره جوابی مثل W_0 وجود دارد بطوریکه

$$W^T_0 X > 0$$

اکنون کوچکترین حاصلضرب $W^T_0 X$ را برابر ضریب ثابت α قرار می دهیم

$$\alpha = \min_{\mathbf{x}(n) \in X_1} \mathbf{w}_0^T \mathbf{x}(n)$$

از ضرب کردن دو طرف رابطه دوم در W^T_0 داریم:

$$\mathbf{w}_0^T \mathbf{w}(n+1) = \mathbf{w}_0^T \mathbf{x}(1) + \mathbf{w}_0^T \mathbf{x}(2) + \cdots + \mathbf{w}_0^T \mathbf{x}(n)$$

$$\mathbf{w}_0^T\mathbf{w}(n+1) \geq n\alpha$$

$$\|\mathbf{w}_0\|^2\|\mathbf{w}(n+1)\|^2 \geq [\mathbf{w}_0^T\mathbf{w}(n+1)]^2$$

با استفاده از نا مساوی کوشی – شوارتز

$$\|\mathbf{w}_0\|^2\|\mathbf{w}(n+1)\|^2 \geq n^2\alpha^2$$

$$\|\mathbf{w}(n+1)\|^2 \geq \frac{n^2\alpha^2}{\|\mathbf{w}_0\|^2} \qquad (\times)$$

بدست آوردن $||W(n+1)||^2$ از راه دیگر

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}(k) \qquad \text{for } k = 1, \ldots, n \quad \text{and} \quad \mathbf{x}(k) \in X_1$$

$$\|\mathbf{w}(k+1)\|^2 = \|\mathbf{w}(k)\|^2 + \|\mathbf{x}(k)\|^2 + 2\mathbf{w}^T(k)\mathbf{x}(k)$$

از آنجایی که بردار ورودی $X(k)$ بطور غلط کلاسه بندی شده است، بنابراین

$$\mathbf{w}^T(k)\mathbf{x}(k) < 0$$

$$\|\mathbf{w}(k+1)\|^2 \leq \|\mathbf{w}(k)\|^2 + \|\mathbf{x}(k)\|^2$$

$$\|\mathbf{w}(k+1)\|^2 - \|\mathbf{w}(k)\|^2 \leq \|\mathbf{x}(k)\|^2, \qquad k = 1, \ldots, n$$

$K=1 \quad ||W(2)||^2 - ||W(1)||^2 \le ||X(1)||^2$

$K=2 \quad ||W(3)||^2 - ||W(2)||^2 \le ||X(2)||^2$

.

.

$K=n \quad ||W(n+1)||^2 - ||W(n)||^2 \le ||X(n)||^2$

با فرض اینکه $W(0)=0$ و جمع روابط بالا

$$\|\mathbf{w}(n+1)\|^2 \le \sum_{k=1}^{n} \|\mathbf{x}(k)\|^2$$

چنانچه فرض کنیم

$$\beta = \max_{\mathbf{x}(k) \in X_1} \|\mathbf{x}(k)\|^2$$

بنابراین

$$||W(n+1)||^2 \le n\beta \quad (\#)$$

دو معادله (×) و (#) برای مقادیر بزرگ n در تناقض هستند مگر اینکه حالت تساوی را برقرار کنیم که در این حالت n مقدار حداکثر خود را خواهد داشت.

$$\frac{n_{max}^2 \alpha^2}{\|w_0\|^2} = n_{max}\beta$$

$$n_{max} = \frac{\beta \|w_0\|^2}{\alpha^2}$$

$$\alpha = \min_{x(n) \in X_1} w_0^T x(n)$$

$$\beta = \max_{x(k) \in X_1} \|x(k)\|^2$$

همچنین با توجه به مقادیر α وβ ، جواب مسئله برای W_0 واحد نخواهد بود.

$$w(n+1) = w(n) + \eta[d(n) - y(n)]x(n)$$

$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases}$$

معیار کارآیی:

به دنبال یک معیاری هستیم که توسط آن بتوانیم بهبود در یادگیری شبکه را بسنجیم.
برای حالت ساده یک پرسپترون رابطه زیر پیشنهاد شده است.

$$J = -E[e(n)v(n)]$$

$$\hat{J}(n) = -e(n)v(n)$$
$$= -[d(n) - y(n)]v(n)$$

گرادیان لحظه ای این تابع نسبت به بردار W(n)

$$\nabla_{\mathbf{w}} \hat{J}(n) = \frac{\partial J(n)}{\partial \mathbf{w}(n)}$$

$$\nabla_{\mathbf{w}} \hat{J}(n) = -[d(n) - y(n)] \frac{\partial v(n)}{\partial \mathbf{w}(n)}$$

$$\frac{\partial v(n)}{\partial \mathbf{w}(n)} = \mathbf{x}(n)$$

$$\nabla_{\mathbf{w}} \hat{J}(n) = -[d(n) - y(n)]\mathbf{x}(n)$$

که تغییرات در وزن جزیی از $\nabla_{\mathbf{w}} \hat{J}(n)$ خواهد بود

$$\begin{aligned}\Delta \mathbf{w}(n) &= -\eta \nabla_{\mathbf{w}} \hat{J}(n) \\ &= \eta[d(n) - y(n)]\mathbf{x}(n)\end{aligned}$$

فصل پنجم

الگوریتم کمترین میانگین مربعات

Least-Mean-Square Algorithm

معادلات Wiener-Hopf برای فیلترهای بهینه خطی

- فرض کنید که P حسگر در فضای محیط قرار داده شده اند. این P حسگر تولید P سیگنال $(X_1,\ldots,X_p)$ را نموده اند. آنها را به شبکه اعمال می کنیم. این شبکه را می توان به عنوان یک فیلتر در نظر گرفت.

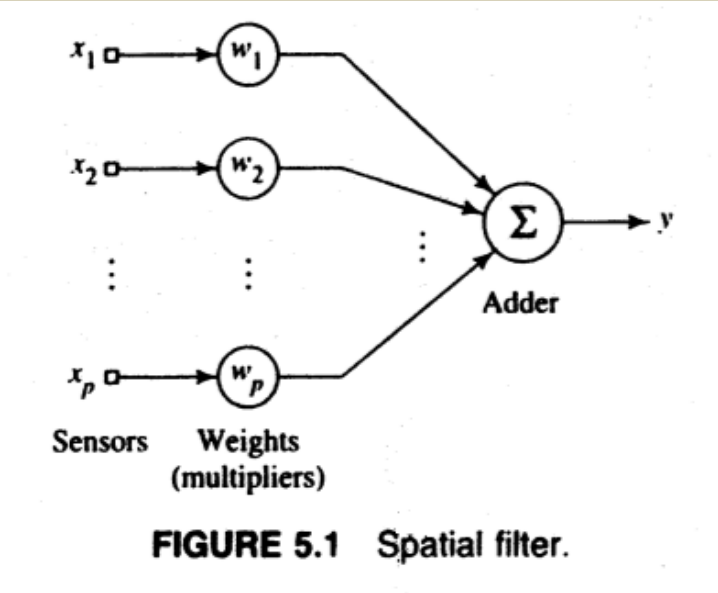


FIGURE 5.1 Spatial filter.

$$y = \sum_{k=1}^{p} w_k x_k$$

در اینجا هدف پیدا کردن وزنهای بهینه ای برای فیلتر فوق است بطوری که تفاضل بین خروجی فیلتر(y) و مقدار دلخواه (d) به مفهوم **میانگین مربعات** ، حداقل گردد.

$$e = d - y$$

$$J = \frac{1}{2} E[e^2]$$

برای معیار کارآیی:

بنابراین می خواهیم مقدار J (میانگین مربعات خطا) را با یافتن مقدار بهینه ای برای وزنها به حداقل برسانیم.

با قرار دادن دو معادله اول در معادله سوم:

$$J = \frac{1}{2} E[d^2] - E\left[\sum_{k=1}^{p} w_k x_k d\right] + \frac{1}{2} E\left[\sum_{j=1}^{p} \sum_{k=1}^{p} w_j w_k x_j x_k\right]$$

$$J = \frac{1}{2}E[d^2] - \sum_{k=1}^{p} w_k E[x_k d] + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k E[x_j x_k]$$

یعنی اینکه وزنها بصورت مقادیر ثابت در نظر گرفته شده اند.برای اختصار در رابطه بالا:

۱. میانگین مربعات پاسخ دلخواه

$$r_d = E[d^2]$$

۲. تابع همبستگی متقابل بین x و d

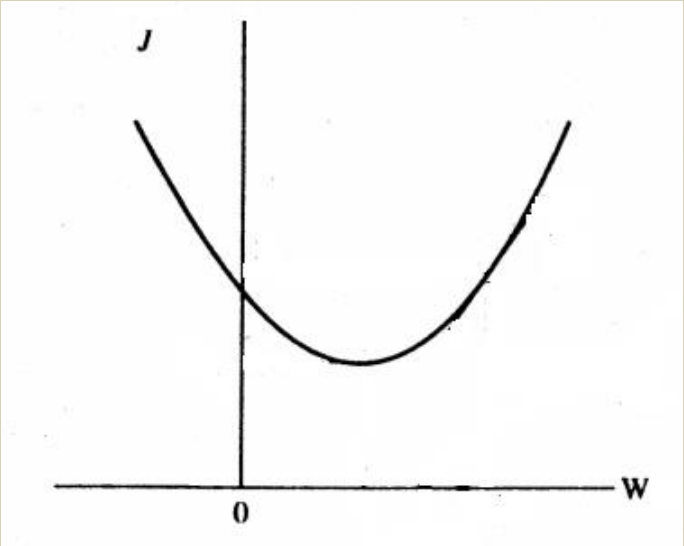
$$r_{dx}(k) = E[dx_k], \qquad k = 1, 2, \ldots, p$$

۳. تابع همبستگی خودی بین سیگنالهای ورودی

$$r_x(j,k) = E[x_j x_k], \qquad j, k = 1, 2, \ldots, p$$

$$J = \frac{1}{2}r_d - \sum_{k=1}^{p} w_k r_{dx}(k) + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k r_x(j,k)$$

با رسم J بر حسب تغییرات وزنها در شبکه ، یک سطح سهمی بدست می آید که به آن سطح کار آیی خطای فیلتر می گویند.



- واضح است که یک چنین سطحی فقط یک مقدار حداقل دارد که در آن نقطه مقدار بهینه برای وزنها حاصل شده است. برای این کار گرادیان J نسبت به وزن مورد نظر را پیدا کرده و آن را برابر صفر قرار می دهیم.

$$\nabla_{w_k} J = \frac{\partial J}{\partial w_k}, \qquad k = 1, 2, \ldots, p$$

$$\nabla_{w_k} J = -r_{dx}(k) + \sum_{j=1}^{p} w_j r_x(j,k)$$

شرایط بهینه فیلتر به صورت زیر تعریف می گردد:

$$\nabla_{w_k} J = 0, \qquad k = 1, 2, \ldots, p$$

چنانچه مقدار بهینه وزن W_k را با W_{k0} نشان دهیم خواهیم داشت:

$$\sum_{j=1}^{p} w_{oj} r_x(j,k) = r_{xd}(k), \qquad k = 1, 2, \ldots, p$$

دسته معادلات فوق را معادلات وینر-هوف و فیلترهایی که شرایط این معادله را برآورده کنند را فیلتر وینر (Wiener filter) گویند.

روش نزولی ترین
Method of Steepest Descent

دسته معادلات وینر-هوف را می توان به صورت ماتریسی نوشت:

$$\begin{bmatrix} r_x(1,1) & \cdots & r_x(1,p) \\ \vdots & & \\ r_x(p,1) & \cdots & r_x(p,p) \end{bmatrix} \begin{bmatrix} W_{01} \\ \vdots \\ W_{0p} \end{bmatrix} = \begin{bmatrix} r_{dx}(1) \\ \vdots \\ r_{dx}(p) \end{bmatrix}$$

$$R_x W_0 = r_{dx}$$

بنابراین یافتن وزنهای بهینه برای فیلتر تطبیق خطی مستلزم پیدا کردن معکوس ماتریس R_x است.

به جای پیدا کردن معکوس ماتریس می توان از روش دیگری استفاده کرد که آن را روش نزولی ترین می نامند.

بر طبق این روش وزنهای فیلتر بطور پیاپی بر روی سطح کارآیی خطا تنظیم شده تا مقدار بهینه برای J بدست آید.

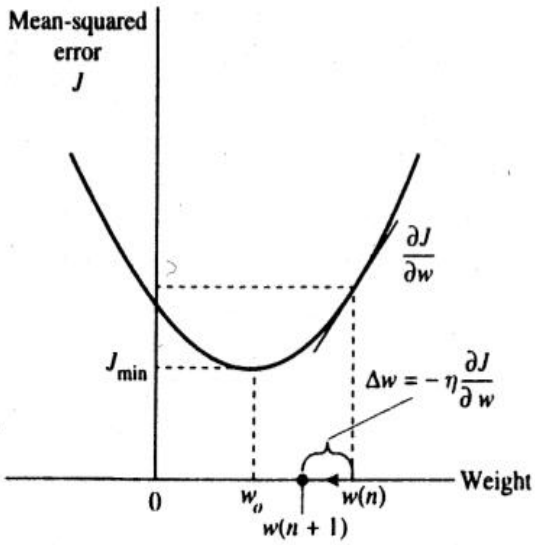


FIGURE 5.2 Illustrating the mean-squared error criterion for a single weight.

$$\nabla_{w_k} J(n) = -r_{dx}(k) + \sum_{j=1}^{p} w_j(n) r_x(j,k)$$

$$\Delta w_k(n) = -\eta \nabla_{w_k} J(n), \qquad k = 1, 2, \ldots, p$$

$$\begin{aligned} w_k(n+1) &= w_k(n) + \Delta w_k(n) \\ &= w_k(n) - \eta \nabla_{w_k} J(n), \qquad k = 1, 2, \ldots, p \end{aligned}$$

$$w_k(n+1) = w_k(n) + \eta \left[r_{dx}(k) - \sum_{j=1}^{M} w_j(n) r_x(j,k) \right], \qquad k = 1, 2, \ldots, p$$

محاسبات بالا بر اساس میانگین مربعات خطا انجام گرفت:

$$J(n) = \frac{1}{2} E[e^2(n)]$$

ولی می توان روش نزولی ترین را بر اساس مجموع مربعات خطا نیز انجام داد:

$$\begin{aligned} \mathscr{E}_{\text{total}}(n) &= \sum_{i=1}^{n} \mathscr{E}(i) \\ &= \frac{1}{2} \sum_{i=1}^{n} e^2(i) \end{aligned}$$

در این روش نیاز به محاسبه توابع همبستگی داریم و چنانچه محیطی که در آن فیلتر وینر کار می کند محیطی نا آشنا باشد مجبور خواهیم بود از مقادیر تخمینی به جای مقادیر واقعی استفاده کنیم که به آن روش کمترین میانگین مربعات می گویند.

الگوریتم کمترین میانگین مربعات

Least-Mean-Square Algorithm

این روش مبتنی بر تخمین لحظه ای تابع $r_x(j,k)$و تابع $r_{dx}(k)$ بصورت زیر است:

$$\hat{r}_x(j,k;n) = x_j(n)x_k(n)$$

$$\hat{r}_{dx}(k;n) = x_k(n)d(n)$$

$$\begin{aligned}\hat{w}_k(n+1) &= \hat{w}_k(n) + \eta\left[x_k(n)\,d(n) - \sum_{j=1}^{p}\hat{w}_j(n)x_j(n)x_k(n)\right] \\ &= \hat{w}_k(n) + \eta\left[d(n) - \sum_{j=1}^{p}\hat{w}_j(n)x_j(n)\right]x_k(n) \\ &= \hat{w}_k(n) + \eta[d(n) - y(n)]x_k(n), \qquad k = 1, 2, \ldots, p\end{aligned}$$

$$y(n) = \sum_{j=1}^{p}\hat{w}_j(n)x_j(n)$$

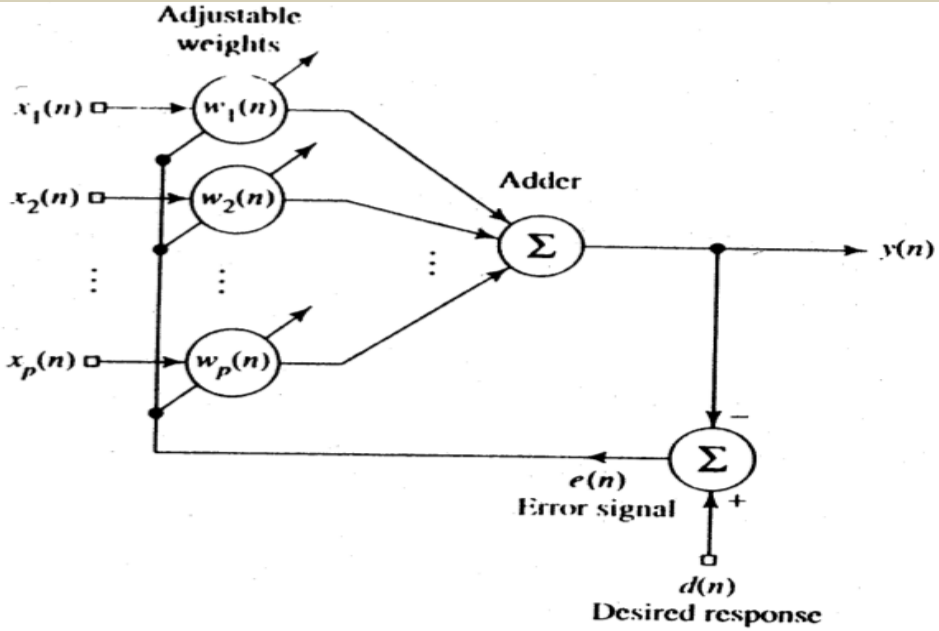


FIGURE 5.3 Adaptive spatial filter.

TABLE 5.1 Summary of the LMS Algorithm

1. Initialization. Set

$$\hat{w}_k(1) = 0 \qquad \text{for } k = 1, 2, \ldots, p$$

2. Filtering. For time $n = 1, 2, \ldots$, compute

$$y(n) = \sum_{j=1}^{p} \hat{w}_j(n) x_j(n)$$

$$e(n) = d(n) - y(n)$$

$$\hat{w}_k(n+1) = w_k(n) + \eta e(n) x_k(n) \qquad \text{for } k = 1, 2, \ldots, p$$

نمودارگذر سیگنال الگوریتم LMS

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta[d(n) - \mathbf{x}^T(n)\hat{\mathbf{w}}(n)]\mathbf{x}(n)$$

$$\hat{\mathbf{w}}(n) = [\hat{w}_1(n), \hat{w}_2(n), \ldots, \hat{w}_p]^T$$

$$\hat{\mathbf{w}}(n+1) = [\mathbf{I} - \eta\mathbf{x}(n)\mathbf{x}^T(n)]\hat{\mathbf{w}}(n) + \eta\mathbf{x}(n)\, d(n)$$

$$\hat{\mathbf{w}}(n) = z^{-1}[\hat{\mathbf{w}}(n+1)]$$

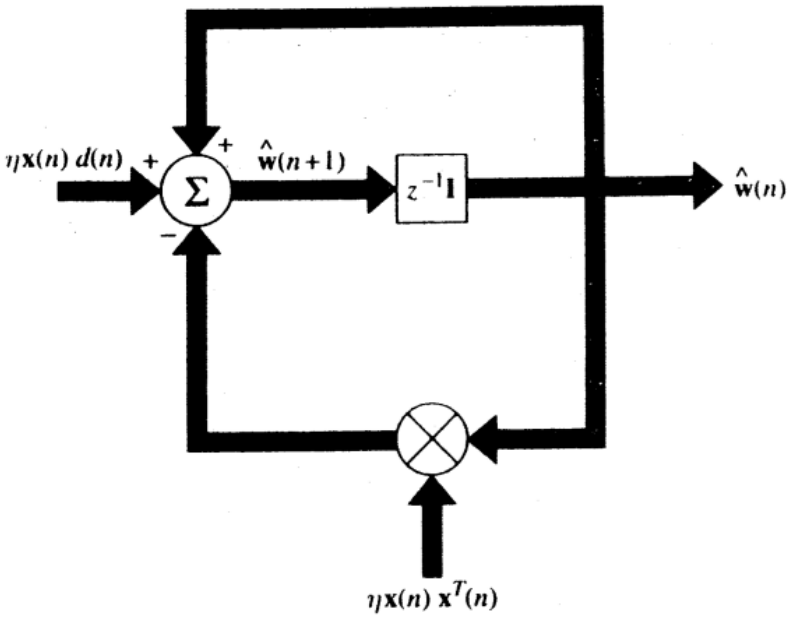


FIGURE 5.4 Signal-flow graph representation of the LMS algorithm.

- بنابراین روش **LMS** نشان دهنده یک سیستم اتفاقی با پسخورد است.
- وجود پسخورد در همگرا بودن یا واگرا بودن الگوریتم اثر مستقیم دارد.
- حلقه پسخورد مهم حلقه پایینی است که تابع تبدیل آن بستگی به **η** و بردار ورودی **X(n)** دارد.

همگرایی در روش LMS

۱. همگرایی در میانگین

$$E[\hat{\mathbf{w}}(n)] \rightarrow \mathbf{w}_o \qquad \text{as } n \rightarrow \infty$$

۲. همگرایی در میانگین مربعات

$$E[e^2(n)] \rightarrow \text{constant} \qquad \text{as } n \rightarrow \infty$$

همگرایی در میانگین

$$E[\hat{\mathbf{w}}(n)] \rightarrow \mathbf{w}_o \qquad \text{as } n \rightarrow \infty$$

با افزایش تعداد دفعات تکرار و نزدیک شدن آن به بینهایت ، مقدار میانگین وزنها به مقدار بهینه نزدیک می شود. بنابراین:

$$E[\hat{\mathbf{w}}(n)\mathbf{x}(n)] = \mathbf{0}$$

$$E[\hat{\mathbf{w}}(n+1)] = (\mathbf{I} - \eta\mathbf{R}_x)E[\hat{\mathbf{w}}(n)] + \eta\mathbf{r}_{dx}$$

$$\mathbf{r}_{dx} = E[\mathbf{x}(n)d(n)]$$

$$\mathbf{R}_x = E[\mathbf{x}(n)\mathbf{x}^T(n)]$$

ماتریس R_x را که یک ماتریس متقارن است می توان با استفاده از انتقال متعامد مشابه بدست آورد(Λماتریس قطری شامل مقادیر ویژه R_x است و Qماتریس متعامد شامل مقادیر ویژه R_x است):

$$\mathbf{Q}^T\mathbf{R}_x\mathbf{Q} = \boldsymbol{\Lambda}$$

$$\mathbf{R}_x = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$$

$$\mathbf{Q}^{-1} = \mathbf{Q}^T$$

$$\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$$

$$\mathbf{R}_x\mathbf{w}_o = \mathbf{r}_{dx}$$

$$\mathbf{Q}^T E[\hat{\mathbf{w}}(n+1)] = (\mathbf{I} - \eta\boldsymbol{\Lambda})\mathbf{Q}^T E[\hat{\mathbf{w}}(n)] + \eta\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}_o$$

$$\mathbf{v}(n) = \mathbf{Q}^T(E[\hat{\mathbf{w}}(n)] - \mathbf{w}_o)$$

$$E[\mathbf{w}(n)] = \mathbf{Q}\mathbf{v}(n) + \mathbf{w}_o$$

$$\mathbf{v}(n+1) = (\mathbf{I} - \eta\mathbf{\Lambda})\mathbf{v}(n)$$

این معادله نشان دهنده P معادله تفاضلی درجه اول است که از یکدیگر مجزا شده اند.

$$v_k(n+1) = (1 - \eta\lambda_k)v_k(n), \qquad k = 1, 2, \ldots, p$$

$$v_k(n) = (1 - \eta\lambda_k)^n v_k(0), \qquad k = 1, 2, \ldots, p$$

برای اینکه مقدار $V_k(n)$ در معادله بالا همگرا شود، باید

$$|1 - \eta\lambda_k| < 1 \qquad \text{for } k = 1, 2, \ldots, p$$

$$v_k(n) \to 0 \qquad \text{as } n \to \infty$$

$$0 < \eta < \frac{2}{\lambda_{max}}$$

همگرایی در میانگین مربعات

- شرایط اثبات در این حالت بسیار سخت تر از حالت قبل است، بنابراین η باید این شرط را داشته باشد:

$$0 < \eta < \frac{2}{\mathrm{tr}[\mathbf{R}_x]}$$

$$\mathrm{tr}[\mathbf{R}_x] = \sum_{k=1}^{p} \lambda_k \geq \lambda_{max}$$

- شرط همگرایی در میانگین مربعات شرطی است که شرط همگرایی در میانگین را در بر می گیرد. پس:

$$0 < \eta < \frac{2}{\text{total input power}}$$

شبکه آدلاین (Adaline)

عنصر تطبیقی خطی (Adaptive Linear Element) که به اختصار Adaline گفته می شود از الگوریتم LMS استفاده می کند.

یک دسته ورودی به شبکه اعمال می کنیم و با فرض خروجی مشخص ، وزنهای بهینه را برای حداقل شدن e_a پیدا می کنیم.

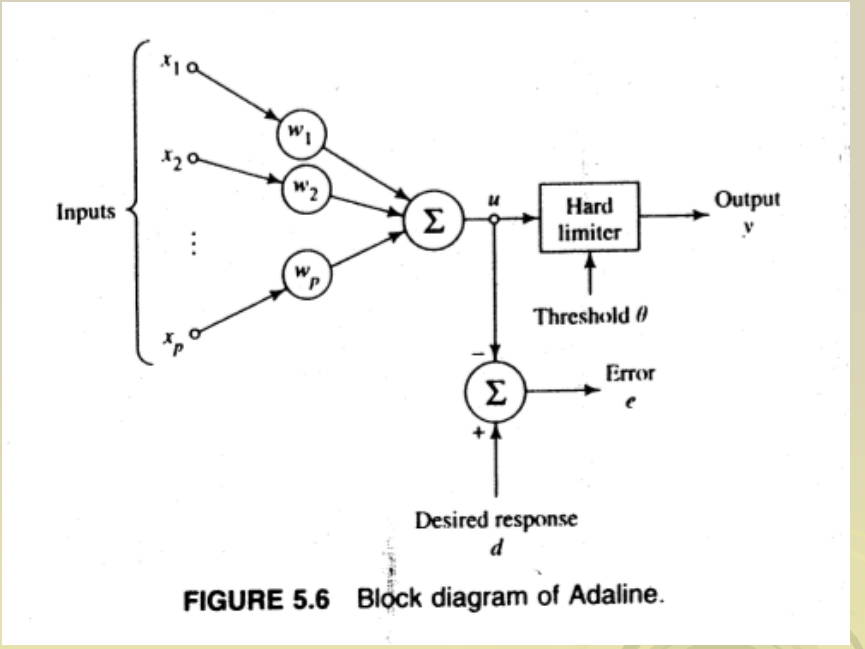


FIGURE 5.6 Block diagram of Adaline.

$$y = \begin{cases} +1 & \text{if } u \geq \theta \\ -1 & \text{if } u < \theta \end{cases}$$

Chapter 6

Multilayer Perceptron(Part1)

شبکه‌های پیش‌خورد چند لایه

جهت سیگنال ورودی بصورت لایه به لایه از ورودی به خروجی است، سلول‌های عصبی چند لایه را (MLP) نیز می‌نامند.

- نوع یادگیری این شبکه‌ها از نوع تحت نظارت است.
- الگوریتم یادگیری این شبکه‌ها از نوع تصحیح خطا است که به آن در اینجا الگوریتم پس انتشار خطا (BP) نیز می‌گویند.
- که در واقع فرم تعمیم یافته الگوریتم کمترین میانگین مربعات (LMS) است.

الگوریتم پس انتشار خطا شامل دو بخش متمایز از هم است:

۱- عبور مستقیم:

بردار سیگنال به لایه ورودی اعمال می‌شود، خروجی لایه به لایه بدست می‌آید، در حالیکه وزن‌ها ثابت‌اند.

۲- عبور معکوس:

خطای خروجی محاسبه شده و لایه به لایه به سمت ورودی برمی‌گردد. وزن‌ها اصلاح می‌شود .

شبکه پیش خورد دارای سه مشخصه بارز است:

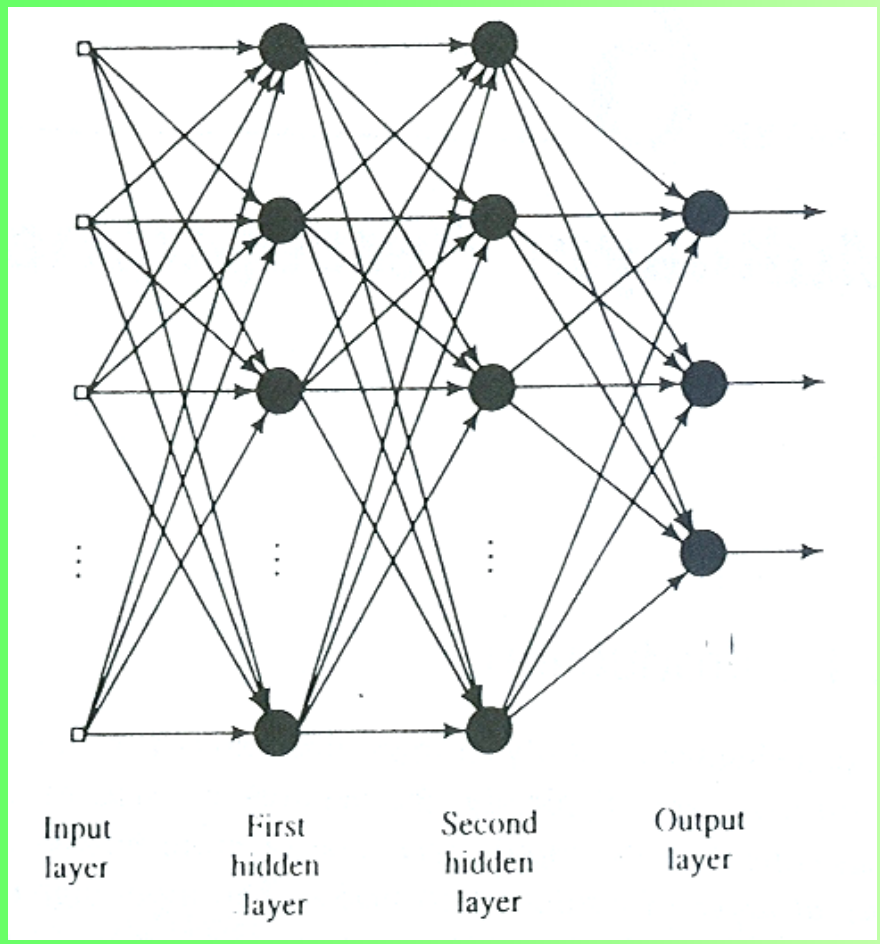
۱- هر سلول دارای یک تابع S شکل مشتق‌پذیر در خروجی است مثل Sigmoid

$$y_j = \frac{1}{1+e^{-v_j}}$$

۲- دارای یک یا چند لایه مخفی است که هدف آن یادگیری خصوصیات ویژه الگوی ورودی است.

۳- ازدحام وزن‌ها بسیار بالاست بطوریکه شبکه توانایی پردازش موازی را فراهم می‌کند.

برخی نمادهای آورده شده روی شکل عبارتند از:



۱- n: مرحله تکرار الگوریتم تعلیم

۲- $\varepsilon(n)$ جمع لحظه‌ای مربعات خطا در مرحله nام و ε_{av} میانگین همه این مقادیر

۳- اندیس‌های i و j و k برای سه لایه به ترتیب از چپ به راست.

۴- $y_k(n)$, $d_k(n)$, $e_k(n)$ به ترتیب خطای خروجی، خروجی مطلوب و خروجی واقعی سلول k در مرحله n هستند.

۵- $W_{ji}(n)$ وزن ارتباطی سلول‌های i و j در مرحله n و تغییرات این وزن است.

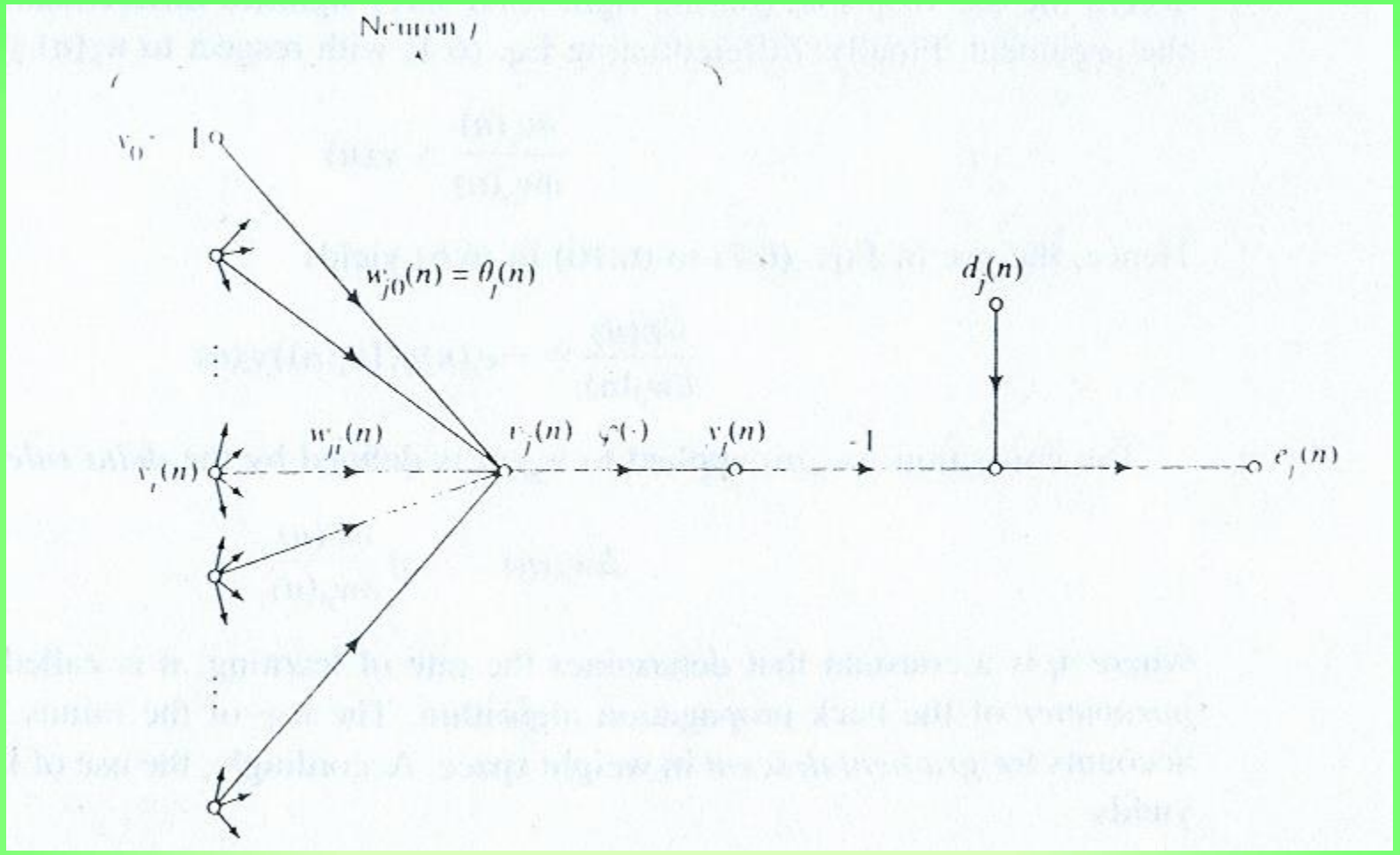
۶- جمع تمامی ورودی‌های وزن‌دار شده سلولی j را $V_j(n)$ گویند.

۷- $\varphi(.)$ رابطه غیرخطی ورودی‌ها و خروجی سلول j است.

۸- مقدار آستانه سلول j برابر θ_j است که آنرا با W_{j0}ورودی ثابت y_0= -1 نشان می‌دهیم.

۹- $X_j(n)$ ورودی j ام درمرحله nام است.

۱۰- η ضریب یادگیری است.



$$e_k(n) = d_k(n) - y_k(n)$$

$$\varepsilon(n) = \frac{1}{2}\sum_{l=1}^{m} e_l^2(n)$$

می‌خواهیم مقدار این خطا را به حداقل برسانیم که این به کمک تنظیم مناسب وزن‌ها صورت می‌گیرد:

$$V_k(n) = \sum_{j=1}^{p} W_{kj}(n) y_j(n)$$

P: تعداد الگوها

$$y_k(n) = \varphi_k\left[V_k(n)\right]$$

$$\frac{\partial \varepsilon(n)}{\partial w_{kj}(n)} = \frac{\partial \varepsilon(n)}{\partial e_k(n)} \cdot \frac{\partial e_k(n)}{\partial y_k(n)} \cdot \frac{\partial y_k(n)}{\partial V_k(n)} \cdot \frac{\partial V_k(n)}{\partial w_{kj}(n)}$$

$$-e_k(n)\varphi_k'\left[V_k(n)\right] y_j(n) \quad \text{کاهش گرادیان=}$$

$$\Delta w_{kj}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{kj}(n)}$$

$$\delta_k(n) = e_k(n) \times \varphi_k'(n)$$

$$\Delta w_{kj}(n) = \eta e_k \varphi_k' y_j(n)$$

$$\Delta w_{kj}(n) = \eta \delta_k y_j(n)$$

دومرحله متمایز در یادگیری:

۱- مرحله مستقیم:

$$V_k(n) = \sum_{j=1}^{m} W_{kj}(n) y_j(n)$$

$$y_k(n) = \varphi_k\left[V_k(n)\right]$$

$$e_k(n) = d_k(n) - y_k(n)$$

۲- مرحله معکوس:

$$\delta_k(n) = \varphi'_k\left[V_k(n)\right] e_k(n)$$

$$\Delta w_{kj}(n) = \eta \delta_k y_j(n)$$

$$\Delta w_{ji}(n) = \eta \delta_j y_i(n)$$

$$\delta_j(n) = \varphi'_j\left[V_j(n)\right] \sum_{k=1}^{m} \delta_k(n) w_{kj}(n)$$

تغییرات در وزن‌ها در الگوریتم فوق براساس اعمال یک الگوی ورودی و محاسبه وزن‌های جدید است (مربعات خطا). در حالیکه در LMS تغییرات در وزنها بر اساس میانگین مربعات خطا بوده است .

در اینجا دو سبک مختلف را برای تغییرات در وزن‌ها براساس روش پس انتشار خطا می‌گوئیم:

۱- سبک الگویی (Pattern Mode) یک الگو بعنوان ورودی اعمال می‌شود و وزن‌ها تنظیم می‌گردد سپس الگوی بعدی. [X(1) , d(1)] چنانچه پس از آخرین الگو خطا هنوز در حال کاهش باشد، یک دوره دیگر N نمونه را اعمال می‌کنیم.

۲- سبک دسته‌ای (Batch Mode)

کل الگوها اعمال می‌شود و سپس وزن‌ها اصلاح می‌گردد. (epoch)

به جای خطای لحظه‌ای E(n) خطای میانگین E_{av} کاهش داده می‌شود.

$$\varepsilon_{av} = \frac{1}{N}\sum_{n=1}^{N}\varepsilon(n) = \frac{1}{2N}\sum_{n=1}^{N}\sum_{k=1}^{m}e_k^2(n)$$

برای مقایسه : در سبک الگوئی :

$$\Delta \hat{w}_{ji} = \frac{1}{N}\sum_{n=1}^{N} \Delta w_{ji}(n)$$

$$\Delta \hat{w}_{ji}(n) = -\frac{\eta}{N}\sum_{n=1}^{N} \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$$

$$\Delta \hat{w}_{ji}(n) = -\frac{\eta}{N}\sum_{n=1}^{N} e_j(n)\frac{\partial e_j(n)}{\partial w_{ji}(n)}$$

تغییرات در وزنها بعد از هر الگو انجام می گیرد .

ولی در سبک دسته ای :

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon_{av}(n)}{\partial w_{ji}(n)} = -\frac{\eta}{N}\sum_{n=1}^{N} e_j(n)\frac{\partial e_j(n)}{\partial w_{ji}(n)}$$

تغییرات در وزنها پس از اعمال کامل الگوها در یک Epoch صورت می گیرد.

ضابطه توقف الگوریتم پس انتشار خطا: Stopping Criteria

فرض کنید بردار W شامل تمام وزن‌ها و W* بردار وزن برای حالتی که خطای شبکه به حداقل (محلی – مطلق) رسیده باشد است:

۱- چنانچه گرادیان g(W*) به اندازه کافی کوچک باشد:

$$\left\| g(w^*) \right\| \le \varepsilon$$

الف – احتمال طولانی شدن زمان تعلیم

ب – احتیاج به محاسبه بردار g(W)

۲- چنانچه میانگین خطا در یک دوره ε_{av} به اندازه کافی کوچک باشد.

$$\varepsilon_{av}(w^*) \leq \tau$$

مشکل اینجاست که ممکن است ε_{av} هیچگاه به مقدار از پیش تعیین شده نرسد.

۳- می‌توان معادلات دو حالت قبل را همزمان اعمال کرد:

- ضریب یادگیری را می‌توان تغییر داد.
- حالت اولیه وزن‌ها را می‌توان تغییر داد.

$$\left\| g(w^*) \right\| \leq \varepsilon$$

$$\varepsilon_{av}(w^*) \leq \tau$$

انتخاب تابع غیرخطی سلول‌ها:

چون برای خطا نیاز به محاسبه مشتق تابع داریم، پس این تابع باید مشتق‌پذیر و ترجیحاً S شکل باشد.

۱- توابع استدلالی

$$y_j = \varphi_j\left[V_j(n)\right] = \frac{1}{1+e^{-\beta v_j}}$$

$$-\infty < V_j(n) < \infty \qquad 0 < y_j(n) < 1$$

الف) سلول‌های لایه خروجی

$$\delta_k(n) = \varphi'_k\left[V_k(n)\right]e_k(n)$$

$$\varphi'_k\left[V_k(n)\right] = \beta y_k(n)\left[1 - y_k(n)\right]$$

$$\boxed{\delta_k(n) = \beta\left[d_k(n) - y_k(n)\right]y_k(n)\left[1 - y_k(n)\right]}$$

ب) سلول‌های لایه مخفی

$$\delta_j(n) = \varphi'\left[V_j(n)\right]\sum_{k=1}^{m}\delta_k(n)w_{kj}(n)$$

$$\boxed{\delta_j(n) = \beta y_j(n)\left[1 - y_j(n)\right]\sum_{k=1}^{m}\delta_k(n)w_{kj}(n)}$$

در صورت استفاده از تانژانت هايپربوليك:

$$\delta_k(n) = \beta\left[1 - y^2{}_k(n)\right]\left[d_k(n) - y_k(n)\right]$$

$$\delta_j(n) = \beta\left[1 - y^2{}_j(n)\right]\sum_{k=1}^{m}\delta_k(n)w_{kj}(n)$$

$$\varphi\left[v(n)\right] = \tanh(\beta v(n))$$

با داشتن β شيب تعيين مي‌گردد.

ضریب یادگیری:

ضریب یادگیری کوچک:

تغییرات در وزن‌ها در هر مرحله کمتر خواهد شد.

الف) ممکن است زمان تعلیم زیاد شود.

ب) ممکن است تعداد مراحل تکرار زیاد شود.

ضریب یادگیری بزرگ:

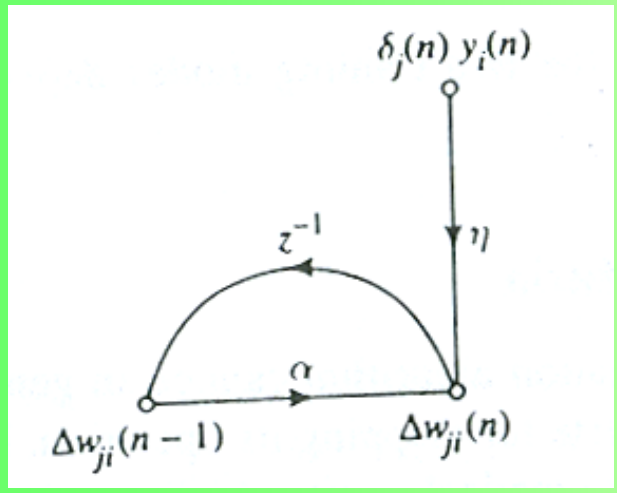
الف) تسریع در همگرایی

ب) امکان واگرا یا نوسانی شدن سیستم

راه حل افزایش سرعت و جلوگیری از عدم پایداری، اضافه کردن عبارت جنبش لحظه‌ای (Momentum Term) است.

$$\Delta w_{ji}(n) = \eta \delta_j y_i(n) + \alpha \Delta w_{ji}(n-1)$$

بنابراین ضریب α حلقه پسخورد برای تغییرات وزن را کنترل می کند .



معادله اخیر را می توان به صورت یک سری زمانی از t=0 تا t=n نشان داد .

$$\Delta w_{ji}(n) = \eta \sum_{t=i}^{n} \alpha^{n-t} \delta_j y_i(n)$$

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -\delta_j(t) y_i(t)$$

$$\Delta w_{ji}(n) = -\eta \sum_{t=i}^{n} \alpha^{n-t} \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$$

برای اینکه سری زمانی همگرا شود:

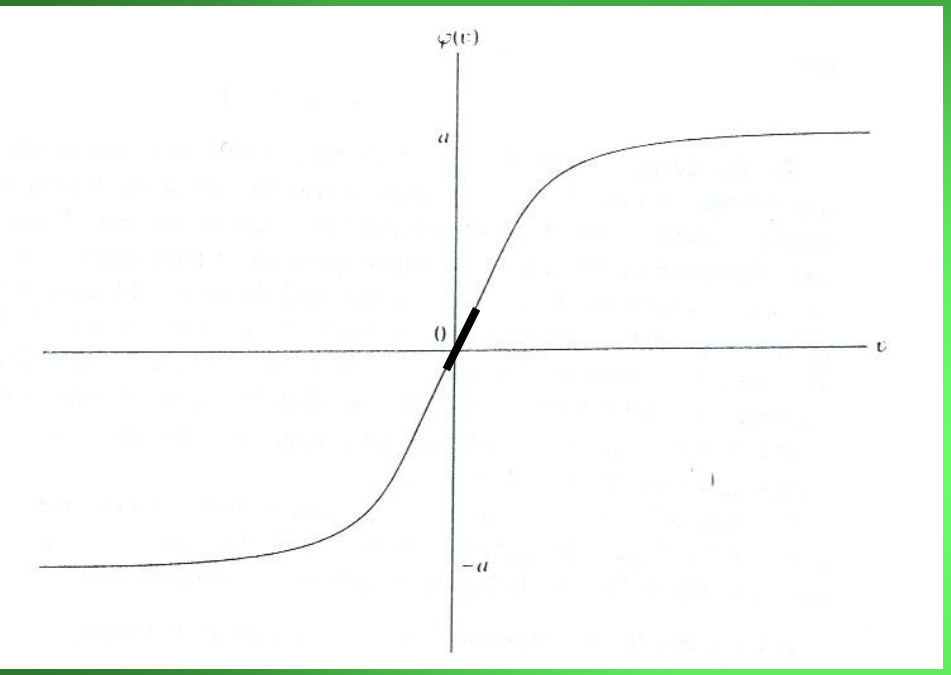
۱- تغییرات در وزن‌ها یک سری نمایی است و برای همگرا شدن آن باید $0 < |\alpha| < 1$

۲- علامت مشتق جزئی $\frac{\partial \varepsilon(t)}{\partial w_{ji}(t)}$ در تمام مراحل:

الف) علامت‌های یکسان: همگرایی شتاب می‌گیرد

ب) علامت‌ها متقارن: امکان نوسانی یا واگرا شدن

مقداردهی اولیه وزن‌ها (Initialization)



۱- اشباع خروجی سلول‌ها:

اگر تعداد ورودی‌ها زیاد باشد و وزن‌ها کوچک اختیار نشود، خروجی به ماکزیمم خود رسیده و مشتق تابع خروجی نسبت به متغیرها تقریباً صفر می‌شود، در نتیجه تغییری در وزن‌ها صورت نمی‌گیرد در صورتی که خطا E_n زیاد است.

۲- اشباع زودرس: مقدار خطا برای مدتی ثابت می‌ماند ولی هنوز به حداقل مطلق نرسیده است.

۳- اشباع نمایی: خروجی شبکه نزدیک 1+ و خروجی دلخواه نزدیک 1- است.

در این حالت خطای شبکه بسیار زیاد اما تغییرات آن بسیار کم است. خروج از این حالت نیازمند زمان زیادی است.

برای پرهیز از مسائل فوق، نکات زیر را باید رعایت کرد:

۱- انتخاب مقادیر اولیه وزن‌ها باید اتفاقی و در یک محدوده کوچک صورت پذیرد هرچه تعداد سلول‌های لایه قبل بیشتر باشد ، محدوده باید کوچکتر شود.

محدوده پیشنهادی $(-\frac{2.4}{F_i},+\frac{2.4}{F_i})$

که Fi تعداد ورودیهای به نرون i است

۲- تعداد سلول‌های لایه مخفی را باید تا حد امکان کم اختیار کرد. بهتر است با تعداد کمی شروع کنیم و سپس به تدریج آنها را افزایش دهیم.

۳- چنانچه تابعه خروجی سلول‌ها تابعی غیر متقارن (تابع فرد) باشد عملکرد کلی شبکه بهبود خواهد یافت. مثل تابع تانژانت هایپربولیک.

$$\varphi(v) = a\tanh(\beta v)$$

۴- مقیاس و جابجایی حدود ورودی و خروجی:

چنانچه محدوده تغییرات ورودی‌ها و خروجی‌ها مشخص باشد یا بتوان آنرا حدس زد، می‌توان آنرا بصورت زیر مقیاس و جابجایی کرد:

$$\bar{x}_i(n) = S_i x_i(n) + O_i$$

$$S_i = \frac{Hi - Low}{Max - Min}$$

$$O_i = \frac{MaxLow - MinHi}{Max - Min}$$

$$\bar{y}_k(n) = S_k y_k(n) + O_k$$

MAX :حداکثر سیگنال واقعی

MIN : حداقل سیگنال واقعی

Hi حداکثر دلخواه

Low حداقل دلخواه

۴- بهتر است ضریب یادگیری برای سلول‌های یک لایه ثابت باشد و لایه به لایه به سمت خروجی کمتر شود تا اثر خطای خروجی در لایه‌های نزدیک به ورودی کمتر و کمتر شود.

۵- برای حالت تعلیم online بهتر است از روش الگویی استفاده شود.

۶- الگوهای ورودی در حالت تعلیم offline بهتر است بصورت تصادفی به شبکه اعمال شود.

۷- الگوهای ورودی باید تا حد امکان غنی باشد یعنی تمام زوایا و مشخصات غیرخطی تابع f(.) را که می‌خواهیم تقریب بزنیم داشته باشد.

۸- انتخاب مقادیر η و α :

افزایش باید با کاهش همراه باشد و بالعکس. انتخاب یکی از دو حالت زیر به افزایش سرعت یادگیری کمک می‌کند.

$$\begin{cases}\eta \to 0\\ \alpha \to 1\end{cases} \qquad \begin{cases}\eta \to 1\\ \alpha \to 0\end{cases}$$

- در روش پس انتشار خطا با اعمال الگوهای تعلیم، شبکه وزن‌ها را تنظیم می‌کند و امیدواریم که شبکه تابع را بطور عمومی تقریب بزند.

منظور از " به طور عمومی" این است که شبکه بتواند برای الگوهای سنجش نیز به خوبی درون‌یابی کند.

- چنانچه شبکه بیش از حد تعلیم ببیند، تقریب عمومی آن از ورودی غیرخطی رضایت بخش نخواهد بود. این مسئله معمولاً وقتی اتفاق می‌افتد که تعداد سلول‌های لایه مخفی نسبت به ابعاد ورودی‌ها بزرگ باشد.

بنابراین می‌توان در انتخاب شبکه دو حالت زیر را در نظر گرفت:

۱- ساختمان شبکه ثابت است و مسئله تعیین نمونه‌های اعمالی به شبکه است.

۲- تعداد الگوهای اعمالی به شبکه ثابت است و مسئله یافتن شبکه مناسب برای کار است. (معمولاً حالت دوم اعمال می‌شود)

- برای حالت اول رابطه زیر پیشنهاد شده است:

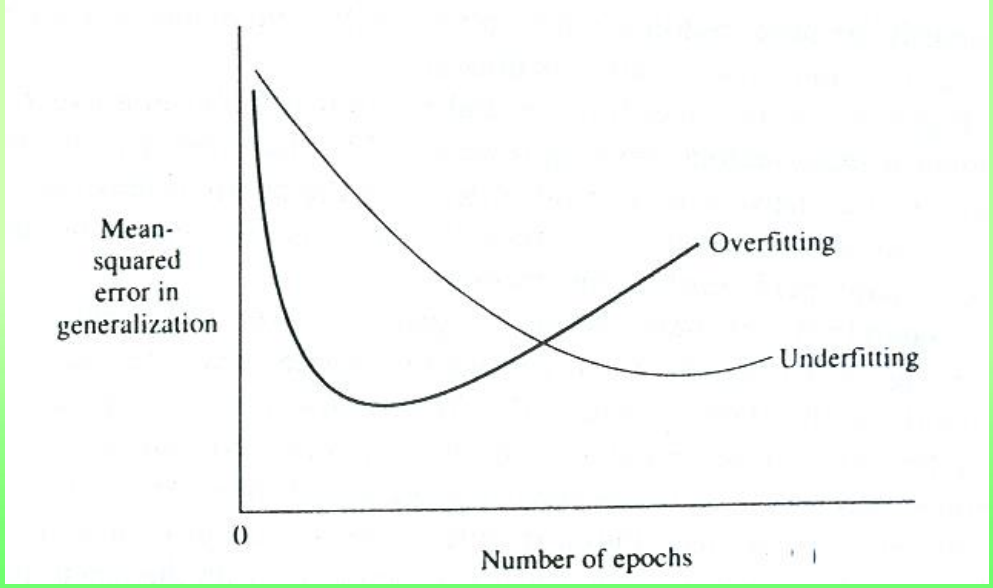
$$N \geq \frac{32w}{\varepsilon} Ln(\frac{32M}{\varepsilon})$$

- M: تعداد سلول‌های مخفی W : تعداد کل وزن‌ها
- a = حداکثر خطای مجاز شبکه به هنگام استفاده
- E خطا هنگام تعلیم
- این رابطه برای یک لایه مخفی و خروجی باینری معتبر است.

رابطه اخیر معمولاً برای بدترین حالت پیشنهاد می‌شود ولی درعمل تعداد کمتری نمونه لازم است. بطور سرانگشتی:

$$N \geq \frac{W}{\varepsilon}$$

برای تعیین اینکه شبکه تقریب عمومی خوبی داشته است یا بد می‌توان به روند تغییرات خطا نیز توجه کرد:



تئوری تقریب عمومی:

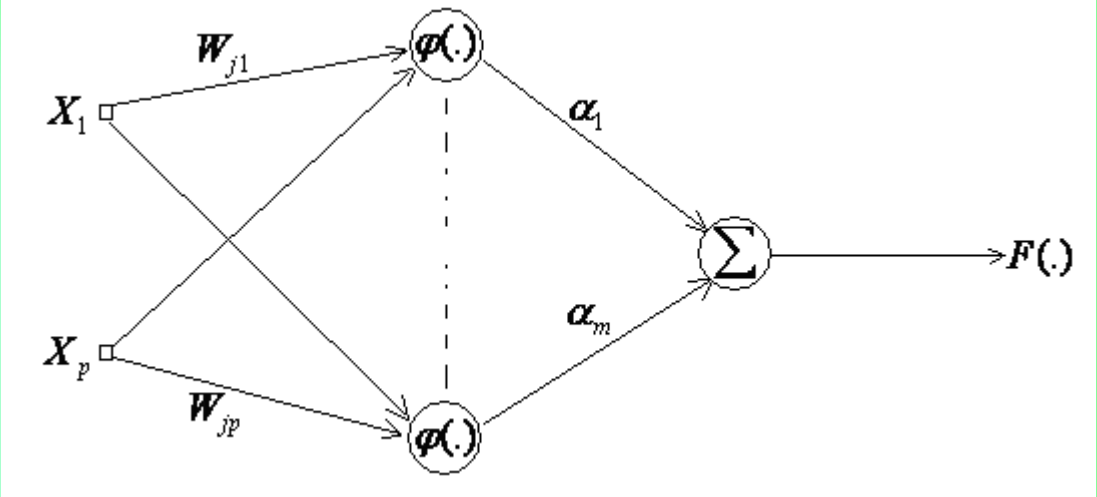
چنانچه y(.) یک تابع پیوسته، غیرثابت، محدود و افزایشی بطوریکنواخت باشد و I_p نشان دهنده یک مکعب واحد با ابعاد p باشد $[0\ ,\ 1]^p$، و نیز چنانچه فضای پیوسته توابعی روی I_p توسط $C(I_p)$ نشان داده شود، در این صورت برای هر تابع دلخواه $f \in C(I_p)$ با یک عدد صحیح m و یک سری ضرائب α_i و w_{ji} و θ_i وجود دارد بطوریکه:

$$F(x_1,...,x_p) = \sum_{j=1}^{m} \alpha_j \varphi(\sum_{i=1}^{p} w_{ji} x_i - \theta_j)$$

تقریب تابع F(.) خواهد بود یعنی این که:

$$\left|F(x_1,...,x_p) - f(x_1,...,x_p)\right| < \varepsilon$$

تئوری بالا را می‌توان توسط یک شبکه پیشخورد اجرا کرد:



۱- شبکه دارای p ورودی و یک لایه مخفی با m سلول است.

۲- سلول مخفی j دارای وزن‌های و تعداد آستانه است.

۳- خروجی شبکه، ترکیب خطی از خروجی سلول‌های مخفی است.

در اینجا ضرایب این ترکیب خطی هستند.

نکته مهم:

برطبق تئوری تقریب عمومی، یک لایه مخفی برای تقریب هر تابع خطی با دقت دلخواه کافی است. ولی این تئوری راهی برای بدست آوردن ساختار بهینه شبکه بدست نمی‌دهد.

تطبیق ضریب یادگیری در روش پس انتشار خطا:

نکته ۱: هر پارامتر قابل تغییر در شبکه باید دارای ضریب یادگیری مخصوص به خودش باشد.

نکته۲: هر ضریب یادگیری باید بتواند از مرحله‌ای به مرحله بعد تغییر کند.

نکته ۳: هنگامیکه مشتق تابع هزینه نسبت به یک متغیر در چندین مرحله پیاپی دارای علامت یکسان باشد در اینصورت ضریب یادگیری باید افزایش یابد.

نکته ۴: هنگامیکه مشتق تابع هزینه نسبت به یک متغیر در چندین مرحله پیاپی تغییر علامت بدهد، ضریب یادگیری باید کاهش یابد.

بنابراین :

$$\begin{cases} W_{ji}(n+1) = W_{ji}(n) + \Delta W_{ji}(n) \\ \eta_{kj}(n+1) = \eta_{kj}(n) + \Delta \eta_{kj}(n) \end{cases}$$

$$\varepsilon(n) = \frac{1}{2}\sum_{l=1}^{M} e_l^2(n)$$

$$V_k(n) = \sum_{j=0}^{q} W_{kj}(n) y_j(n)$$

$$w_{kj}(n) = w_{kj}(n-1) - \eta_{kj}(n)\frac{\partial \varepsilon(n-1)}{\partial w_{kj}(n-1)}$$

$$\frac{\partial \varepsilon(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi_k'(V_k(n))y_j(n)$$

اکنون می‌خواهیم تغییرات تابع هزینه نسبت به متغیر جدید را در هر مرحله پیدا کرده و آنرا تغییر دهیم :

$$\eta_{kj}(n+1) = \eta_{kj}(n) + \gamma \frac{\partial \varepsilon(n)}{\partial \eta_{kj}(n)}$$

$$\frac{\partial \varepsilon(n)}{\partial \eta_{kj}(n)} = \frac{\partial \varepsilon(n)}{\partial y_k(n)} \cdot \frac{\partial y_k(n)}{\partial V_k(n)} \cdot \frac{\partial V_k(n)}{\partial \eta_{kj}(n)}$$

$$V_k(n) = \sum_{j=0}^{q} y_j(n) \left[w_{kj}(n-1) - \eta_{kj}(n) \frac{\partial \varepsilon(n-1)}{\partial w_{kj}(n-1)} \right]$$

$$\frac{\partial V(n)}{\partial \eta_{kj}(n)} = -y_j(n)\frac{\partial \varepsilon(n-1)}{\partial w_{kj}(n-1)}$$

$$\frac{\partial \varepsilon(n)}{\partial \eta_{kj}(n)} = -e_k(n)\varphi_k^{'}(V_k(n))y_j(n)\left[-\frac{\partial \varepsilon(n-1)}{\partial w_{kj}(n-1)}\right]$$

$$\frac{\partial \varepsilon(n)}{\partial \eta_{kj}(n)} = -\frac{\partial \varepsilon(n)}{\partial w_{kj}(n)}\cdot\frac{\partial \varepsilon(n-1)}{\partial w_{kj}(n-1)}$$

$$\Delta \eta_{kj}(n) = -\gamma\frac{\partial \varepsilon(n)}{\partial \eta_{kj}(n)} = \gamma\frac{\partial \varepsilon(n)}{\partial w_{kj}(n)}\cdot\frac{\partial \varepsilon(n-1)}{\partial w_{kj}(n-1)}$$

از معادله اخیر:

۱- تغییر علامت در دو مرحله پیاپی منجر به منفی شدن $\Delta\eta_{kj}$ و در نتیجه کاهش ضریب یادگیری می‌شود.

۲- یکسان بودن علامت مشتق در دو مرحله پیاپی منجر به مثبت شدن $\Delta\eta_{kj}$ و در نتیجه افزایش ضریب یادگیری می‌شود.

به این روش گاهی روش دلتا – دلتا نیز می‌گویند.

معایب روش دلتا – دلتا:

۱- یافتن مقدار مناسب γ

۲- احتیاج به حافظه بیشتر نسبت به روش پس انتشار

۳- امکان منفی شدن ضریب یادگیری

برای رفع مشکلات اول و سوم:
علامت و مقدار تغییرات در تمامی مراحل قبلی را برای تغییرات در مرحله کنونی دخالت می‌دهیم.

$$\Delta\eta_{kj}(n+1)=\begin{cases}\lambda & S_{kj}(n)D_{kj}(n)>0\\ -\beta\eta_{kj}(n) & S_{kj}(n)D_{kj}(n)<0\\ 0 & \text{Otherwise}\end{cases}$$

$$D_{kj}(n)=\frac{\partial\varepsilon(n)}{\partial w_{kj}(n)}$$

$$S_{kj}(n)=(1-\xi)D_{kj}(n-1)+\xi S_{kj}(n-1)$$

$$S_{kj}(n)=\sum_{p=0}^{n-2}\xi^{p}(1-\xi)D_{kj}(n-p-1)+\xi^{n-1}S_{kj}(1)$$

روش اخیر بنام دلتا – بار – دلتا معروف است.

۱- چنانچه $\beta = \lambda = 0$ ، روش پس انتشار خطا را خواهیم داشت.
۲- افزایش ضریب یادگیری خطی است.
۳- کاهش در ضریب یادگیری بطور نمائی است.
۴- ضریب یادگیری همواره مثبت است.

- در روش دلتا – بار – دلتا احتیاج به حافظه بیشتری داریم.

پایان



Chapter 6

Multilayer Perceptron(Part2)

تطبیق ضریب یادگیری در روش پس انتشار خطا:

نکته ۱: هر پارامتر قابل تغییر در شبکه باید دارای ضریب یادگیری مخصوص به خودش باشد.

نکته۲: هر ضریب یادگیری باید بتواند از مرحله‌ای به مرحله بعد تغییر کند.

نکته ۳: هنگامیکه مشتق تابع هزینه نسبت به یک متغیر در چندین مرحله پیاپی دارای علامت یکسان باشد در اینصورت ضریب یادگیری باید افزایش یابد.

نکته ۴: هنگامیکه مشتق تابع هزینه نسبت به یک متغیر در چندین مرحله پیاپی تغییر علامت بدهد، ضریب یادگیری باید کاهش یابد.

روش دلتا - بار– دلتا

- ۱ . چنانچه $\lambda = \beta = 0$ آنگاه پس انتشار خواهيم داشت
- ۲ . افزايش در ضريب يادگيري خطي است
- ۳ . كاهش در ضريب يادگيري نمايي است
- ۴ . ضريب يادگيري همواره مثبت است
- در اين روش تابع هزينه به صورت جمع لحظه اي مربعات خطا ميباشد

- در اين روش تابع هزينه به صورت جمع لحظه اي مربعات خطا مي باشد

فرض بر اين است كه فضاي پارامترهاي وابسته به شامل ضرايب يادگيري متفاوتي است

$$E(n) = \frac{1}{2}\sum_j e_j^2(n) = \frac{1}{2}\sum_j [d_j(n) - y_j(n)]^2$$

$$\frac{\partial \upsilon_j(n)}{\partial \eta_{ji}(n)} = -y_i(n)\frac{\partial \mathcal{E}(n-1)}{\partial w_{ji}(n-1)}$$

$$\frac{\partial E(n)}{\partial \eta_{ji}(n)} = \frac{\partial E(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial \upsilon_j(n)}\frac{\partial \upsilon_j(n)}{\partial \eta_{ji}(n)}$$

$$\frac{\partial y_j(n)}{\partial \upsilon_j(n)} = \varphi_j'(\upsilon_j(n))$$

$$\upsilon_j(n) = \sum_i w_{ji}(n)\, y_i(n)$$

$$\frac{\partial E(n)}{\partial y_j(n)} = -[d_j - y_j(n)] = -e_j(n)$$

$$w_{ji}(n) = w_{ji}(n-1) - \eta_{ji}(n)\frac{\partial \mathcal{E}(n-1)}{\partial w_{ji}(n-1)}$$

$$\upsilon_j(n) = \sum_i y_i(n)\left[w_{ji}(n-1) - \eta_{ji}(n)\frac{\partial \mathcal{E}_i(n-1)}{\partial w_{ji}(n-1)}\right]$$

از روابط بالا دو نتیجه مهم میتوان گرفت :

- ۱ . وقتي مشتق سطح خطا نسبت به وزن ها در دو تکرار متوالي علامت يکساني دارد تغيير مقداري مثبت دارد . بنابراين، فرآيند تطبيق پارامترنرخ آموزش را براي وزن افزايش مي دهد و فرآيند پس انتشار در راستاي آن وزن تسريع ميشود.
- ۲ . وقتي مشتق خطا نسبت به وزن در دو تکرار متوالي نوسان ميکند مقدار منفي دارد و فرآيند پس انتشار در راستاي آن وزن کند خواهد شد .

در نتیجه:

$$\Delta\eta_{ji}(n+1) = \begin{cases} \kappa & \text{if } S_{ji}(n-1)D_{ji}(n) > 0 \\ -\beta\eta_{ji}(n) & \text{if } S_{ji}(n-1)D_{ji}(n) < 0 \\ 0 & \text{otherwise} \end{cases}$$

$$D_{ji}(n) = \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$$

$$S_{ji}(n) = (1-\xi)D_{ji}(n-1) + \xi S_{ji}(n-1)$$

که در آن ξ یک ضریب مثبت است و D_{ij} مقدار مشتق جزيي است
مقدار وزن دهي شده و نمايي مشتقات كنوني و قبلي نسبت به وزن ها است

از مباحث بالا به نتايج زير مي رسيم:

۱ . اگر مشتق در تكرار متوالي علامت يكسان داشته باشند نرخ يادگيري مقداري ثابت خواهد داشت و اگر مقادير مثبت و منفی داشته باشند نرخ يادگيري با ضريب كاهش خواهد يافت.

۲ . افزايش خطي و كاهش آن نمايي است.

معايب :

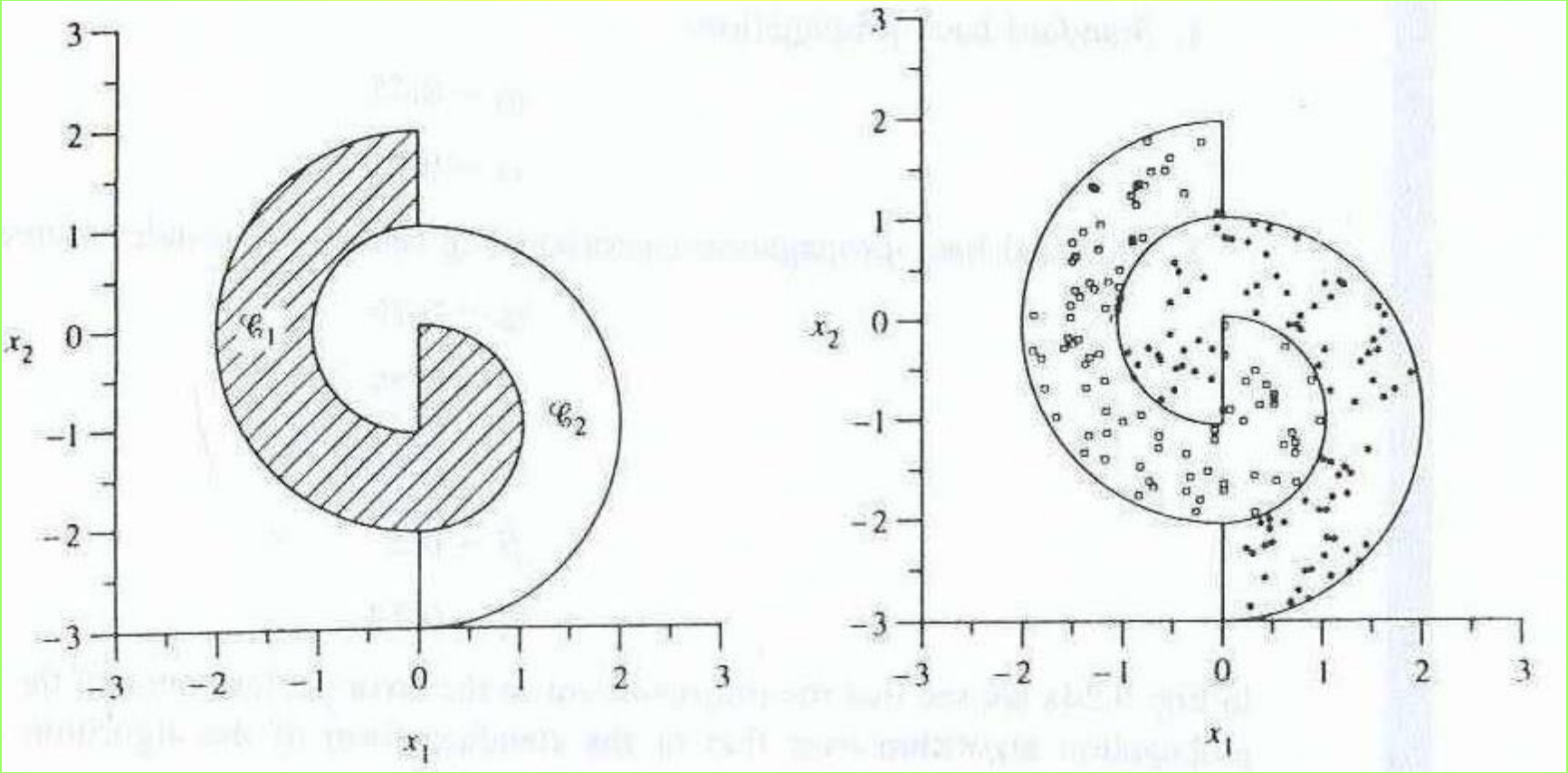
۱ . انتخاب مقدار مناسب

۲ . احتياج به حافظه بيشتر

۳ . امكان منفي شدن ضريب يادگيري

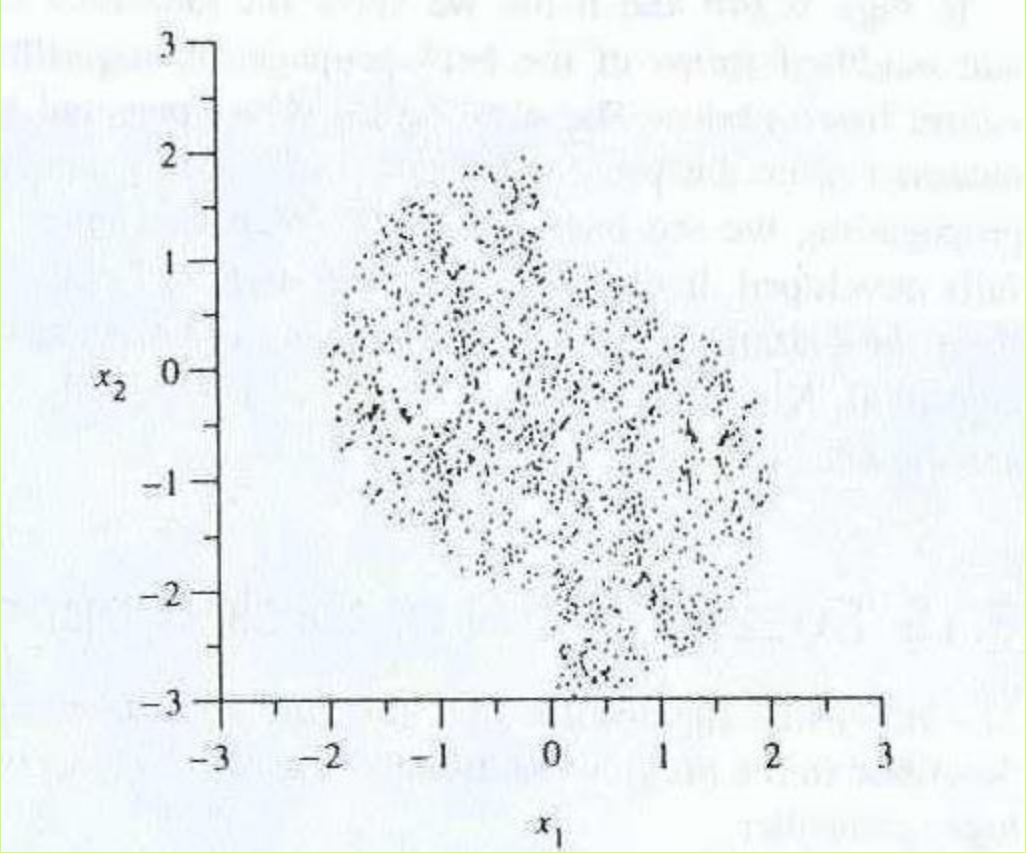
تجربه كامپيوتري :

در اينجا يک مثال از دسته بندي دو بعدي را مطرح مي كنيم. توزيع دو كلاس در شكل نشان داده شده است ℓ_1, ℓ_2

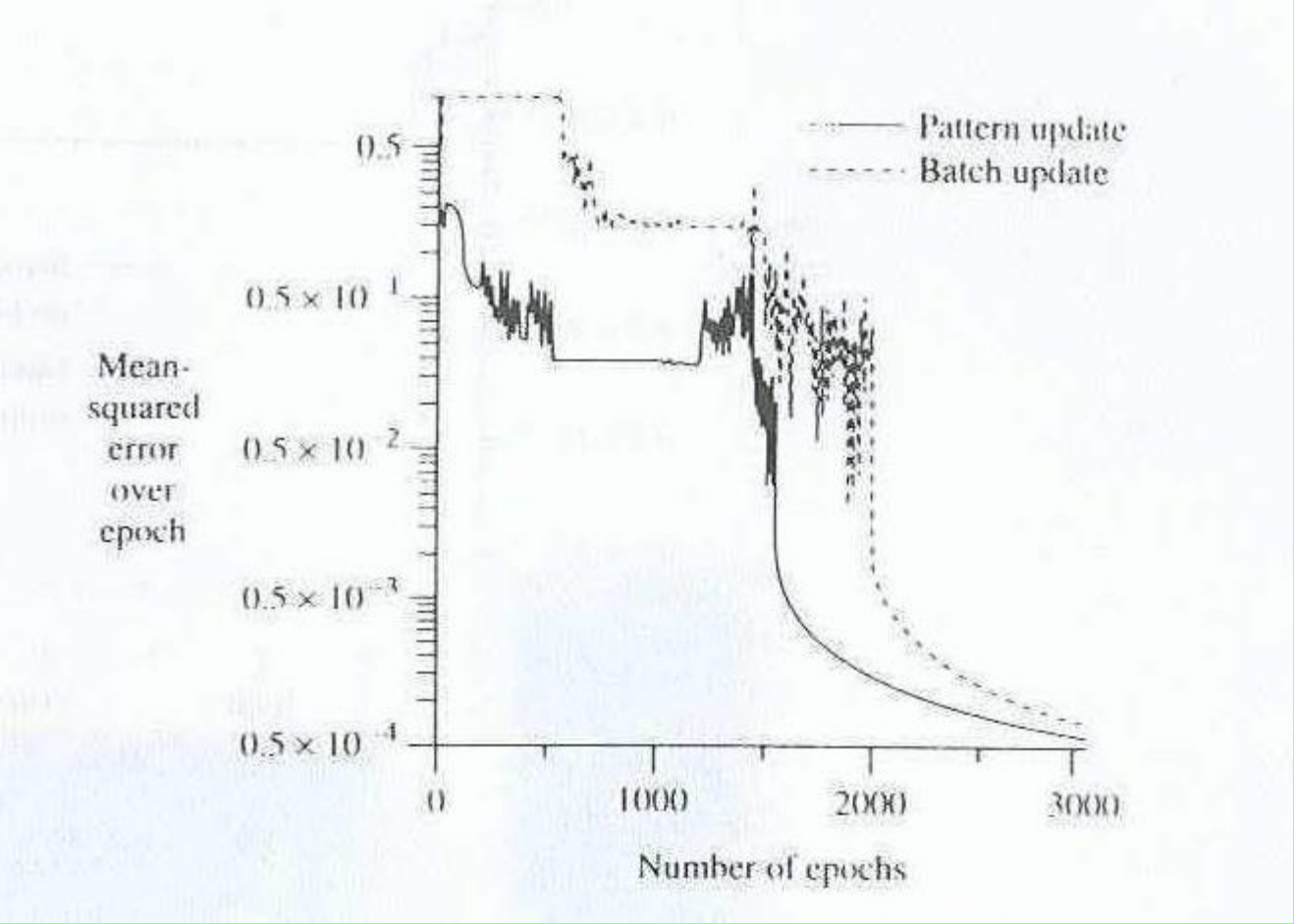


محاسبات مربوط به شبکه عصبي حاوي اطلاعات زير است:

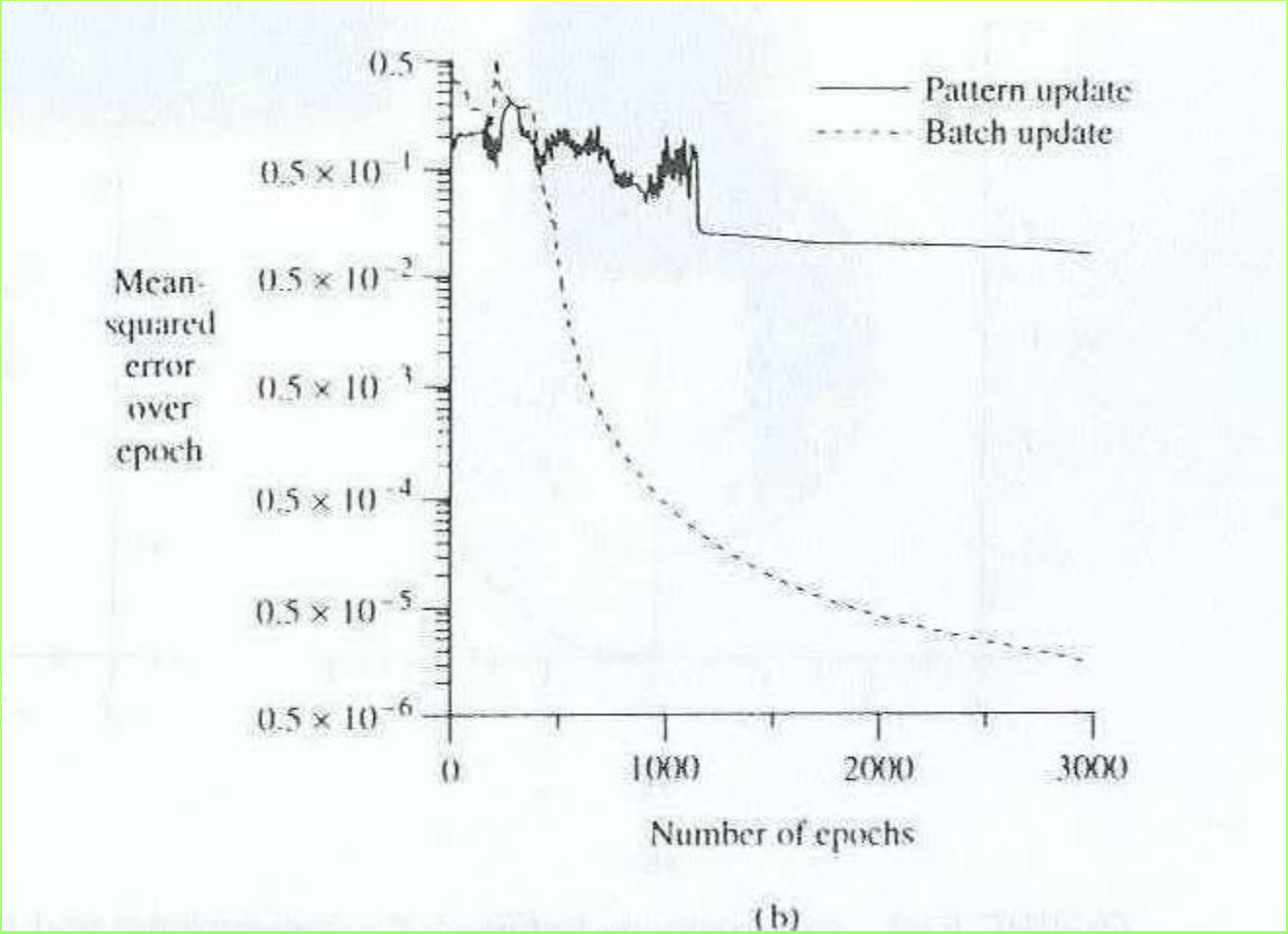
- تعداد ورودي ها : ۲
- تعداد نرون هاي اولين لايه مخفي :۱۲
- تعداد نرون هاي دومين لايه مخفي : ۴
- تعداد نرون هاي لايه خروجي : ۲
- درآموزش ۱۰۰ نمونه از الگوي اول و ۱۰۰ نمونه از الگوي دوم ، در تست ۴۸۵ نمونه از الگوي اول و ۵۱۸ نمونه از الگوي دوم استفاده شده است


3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1

در شکل مقایسه بین روش آموزش دسته اي Batch و روش تطبیق الگو را مشاهده مي کنید



در شکل مقایسه بین روش معمول (Batch) و شکل بهبود یافته آن را مشاهده مي کنید



نظريه فازي

- منطق فازي : از ديدگاه انسان نسبت به صفات و تعاريف (هدف : بيان عدم قطعيت در موضوع)
- مجموعه كلاسيک _منطق دو ارزشي (منطق قطعي):
- در اين مجموعه هر عضو يا متعلق به مجموعه است يا نيست
- از مشاهداتي كه معلوم ومشخص نيست صرفنظر مي شود
- مجموعه فازي:
- كلاسي از اجزا با درجه عضويت پيوسته
- توابع عضويت وابسته است كه به هر شي يک درجه عضويت نسبت مي دهد
- در اين صورت كليه مشاهدات غير دقيق را مدل سازي مي كند

تصميم گيري در فضاي فازي

- محاسبات تصميم گيري در فضاي فازي را ميتوان به سه مرحله تقسيم كرد
- Fuzzification : فازي كردن اطلاعات ورودي براي دست يابي به پايگاه اطلاعاتي (data base)
- Rule base (استنتاج) بهره گيري از قوانين تصميم گيري به صورت پايگاه قوانين
- Defuzzification غير فازي سازي اطلاعات خروجي

- **پايگاه اطلاعاتي :**
- براي دستيابي به يک تصميم گيري معقول بهتر است پردازش اطلاعات متناسب با دانش تصميم گيري باشد. در اين راستا جمع آوري اطلاعات متناسب با تصميم گيري در فايلي به نام پايگاه اطلاعات

(data base) اهميت كليدي دارد

پايگاه اطلاعاتي بايد به اين نكته توجه شود كه اعداد حقيقي به بخش هايي يا طيف هايي از داده تقسيم مي شوند.

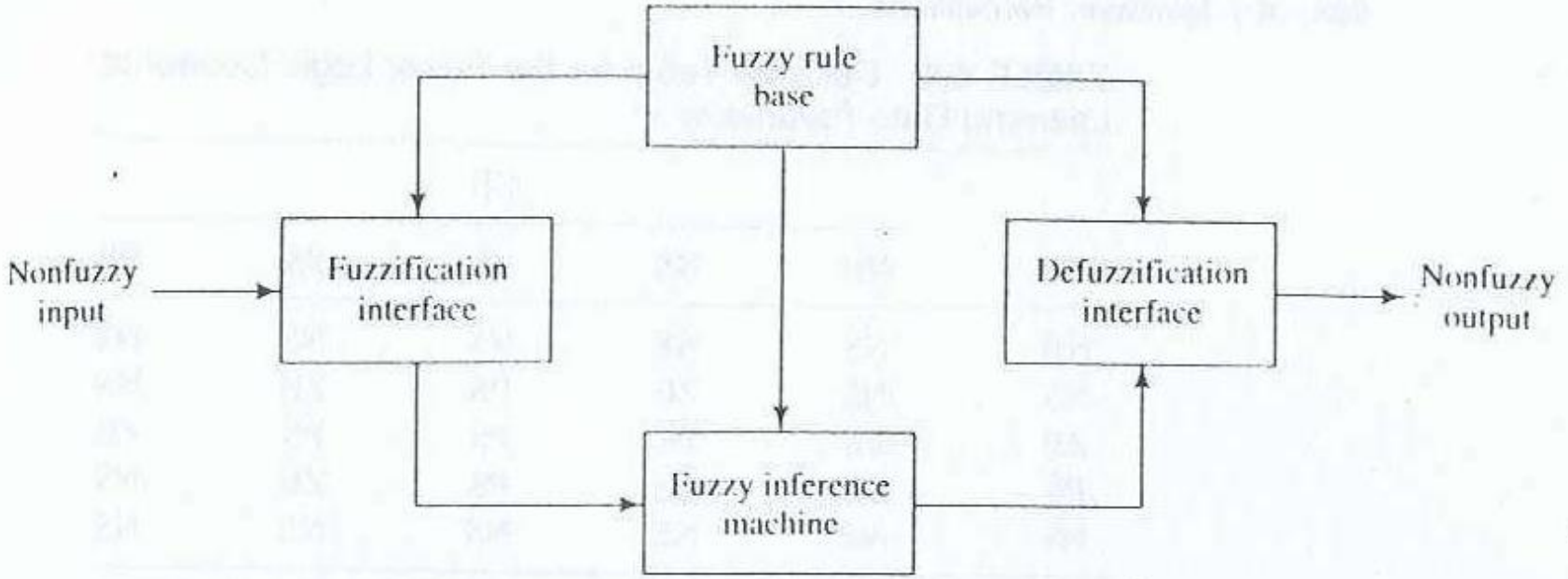


FIGURE 6.26 Block diagram of a fuzzy system.

متغيرهاي كلامي در قالب زير مجموعه هاي فازي:

بسياري از كلمات ، جملات و اصطلاحاتي كه ما در زندگي روز مره بكار مي بريم مفاهيمي غير دقيق و مبهم هستند . يک متغير كلامي يک عدد فازي است .

معمولا فضاي فازي اطلاعات به هفت بخش زير تقسيم ميشود :

۱ . منفي بزرگ : Negative Big (NB)

۲ . منفي متوسط : Negative Medium (NM)

۳ . منفي كوچک : Negative Small (NS)

۴ . صفر : Zero (Z)

۵ . مثبت كوچک : Positive Small (PS)

۶ . مثبت متوسط : Positive Medium (PM)

۷ . مثبت بزرگ positive Big (PB)

تسريع در همگرايي BP

- فرض کنيد که گراديان سطح خطا تقريبا برابر تغييرات خطا باشد (خطا در اينجا جمع مربعات لحظه اي است) و آن را با CE نشان ميدهيم . همچنين فرض کنيد که گراديان دوم سطح خطا تقريبا برابر تغييرات در تغييرات خطا باشد و آن را با CCE نشان دهيم :
- ۱ . اگر CE کوچک باشد و هيچ تغيير علامتي در چندين مرحله تکرار متوالي نداشته باشد در اين صورت ضريب يادگيري را بايد افزايش داد.
- ۲ . اگر تغيير علامت در CE در چندين مرحله تکرار متوالي وجود داشته باشد در اين صورت ضريب ياد گيري را بايد بدون توجه به مقدار CCE کاهش داد
- ۳ . اگر هر دو CE و CCE کوچک باشند و هيچ تغيير علامتي در چندين μ مرحله تکرار متوالي نداشته باشد در اين صورت هر دو ضريب يادگيري وجنبش آني را افزايش α داد

قوانين فازي براي آموزش پس انتشار در جدول هاي زير آورده شده است

TABLE 6.6 Decision Table for the Fuzzy Logic Control of Learning-Rate Parameter η[a]

	CE				
CCE	NB	NS	ZE	PS	PB
NB	NS	NS	NS	NS	NS
NS	NS	ZE	PS	ZE	NS
ZE	ZE	PS	PS	PS	ZE
PS	NS	ZE	PS	ZE	NS
PB	NS	NS	NS	NS	NS

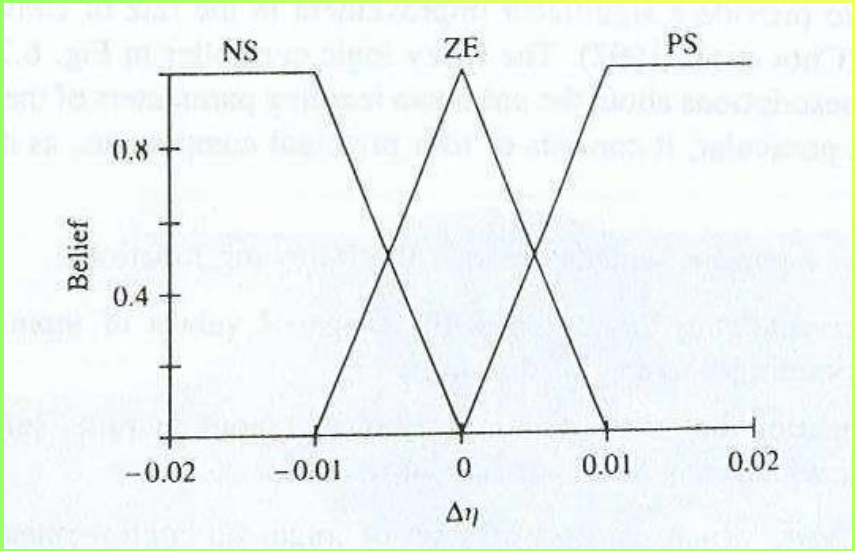
[a] Definitions: NB, negative big; NS, negative small; ZE, zero; PS, positive small; PB, positive big.

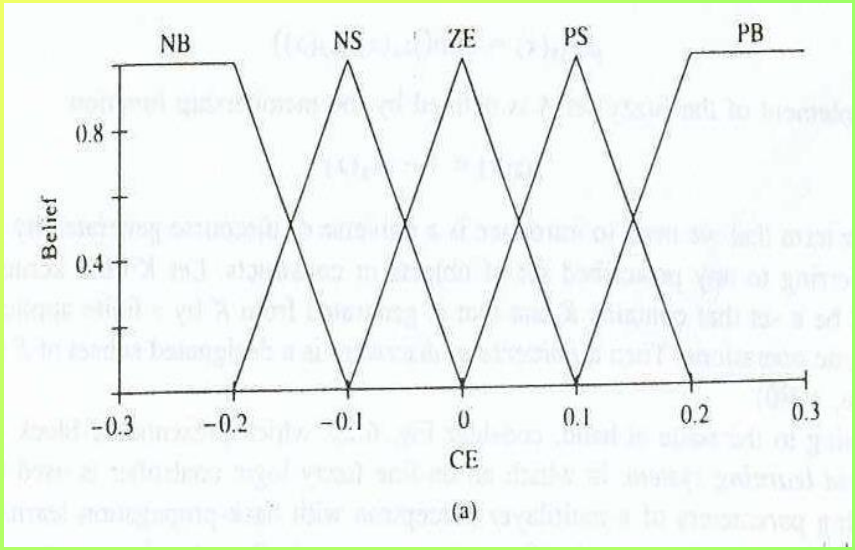
TABLE 6.7 Decision Table for the Fuzzy Control of Momentum Constant α[a]

	CE				
CCE	NB	NS	ZE	PS	PB
NB	NS	NS	ZE	ZE	ZE
NS	NS	ZE	ZE	ZE	ZE
ZE	ZE	PS	PS	PS	ZE
PS	ZE	ZE	ZE	ZE	NS
PB	ZE	ZE	ZE	NS	NS

[a] Definitions: NB, negative big; NS, negative small; ZE, zero; PS, positive small; PB, positive big.

توابع عضویت به شرح زیر است





سیستم آموزش پیوندي (Hybrid) براي کنترل آموزش پس انتشار خطا :

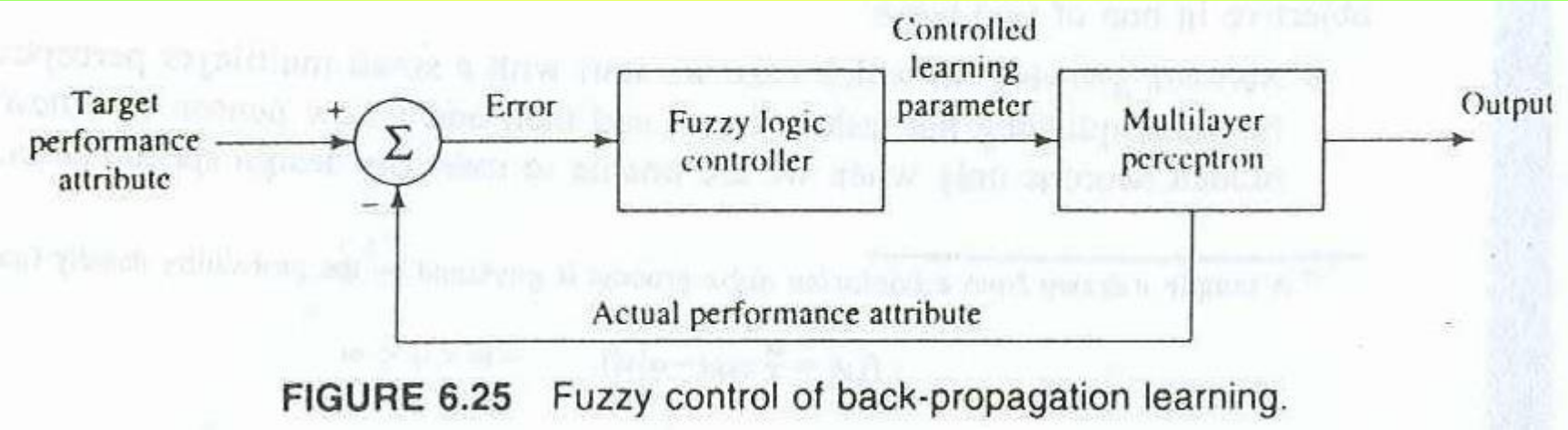


FIGURE 6.25 Fuzzy control of back-propagation learning.

شبکه هاي عصبي – فازي (Neuro-Fuzzy):

۱ . تطبيق

۲ . آموزش

۳ . پردازش

FNN به دو صورت و ساختار معرفي مي شود:

۱ . ساختار اول بدون فيدبک است که شکل آنرا درزير مي بينيد:

۲ . ساختار دوم با فيدبک است که در آن ANN تعيين کننده مکانيزم استنتاج است

-شبکه هاي عصبي ذاتا فازي

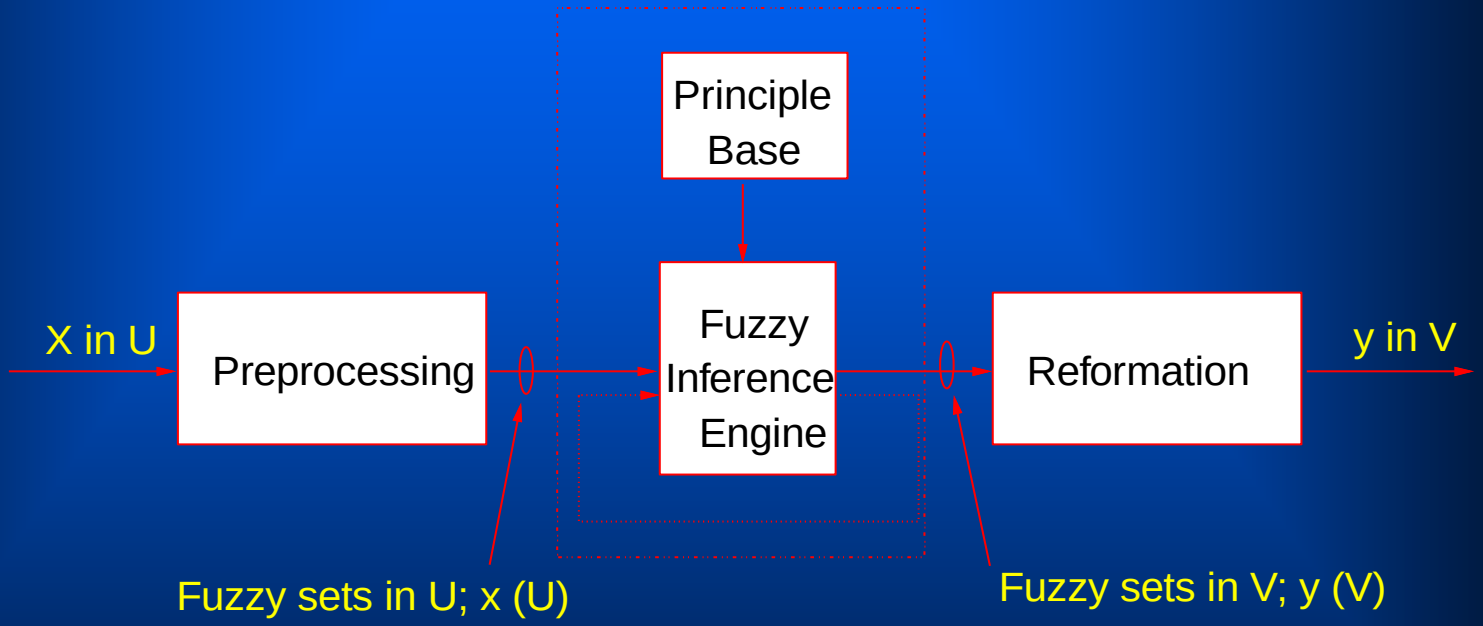
تبديل سلول عصبي به سلول فازي :

۱ . شبکه عصبي يقيني (Deterministic) : پارامترهاي Crisp توابع فعاليت يقيني در ايجاد شبکه استفاده خواهد شد. شبکه هاي MLP از نوع مذکور هستند .

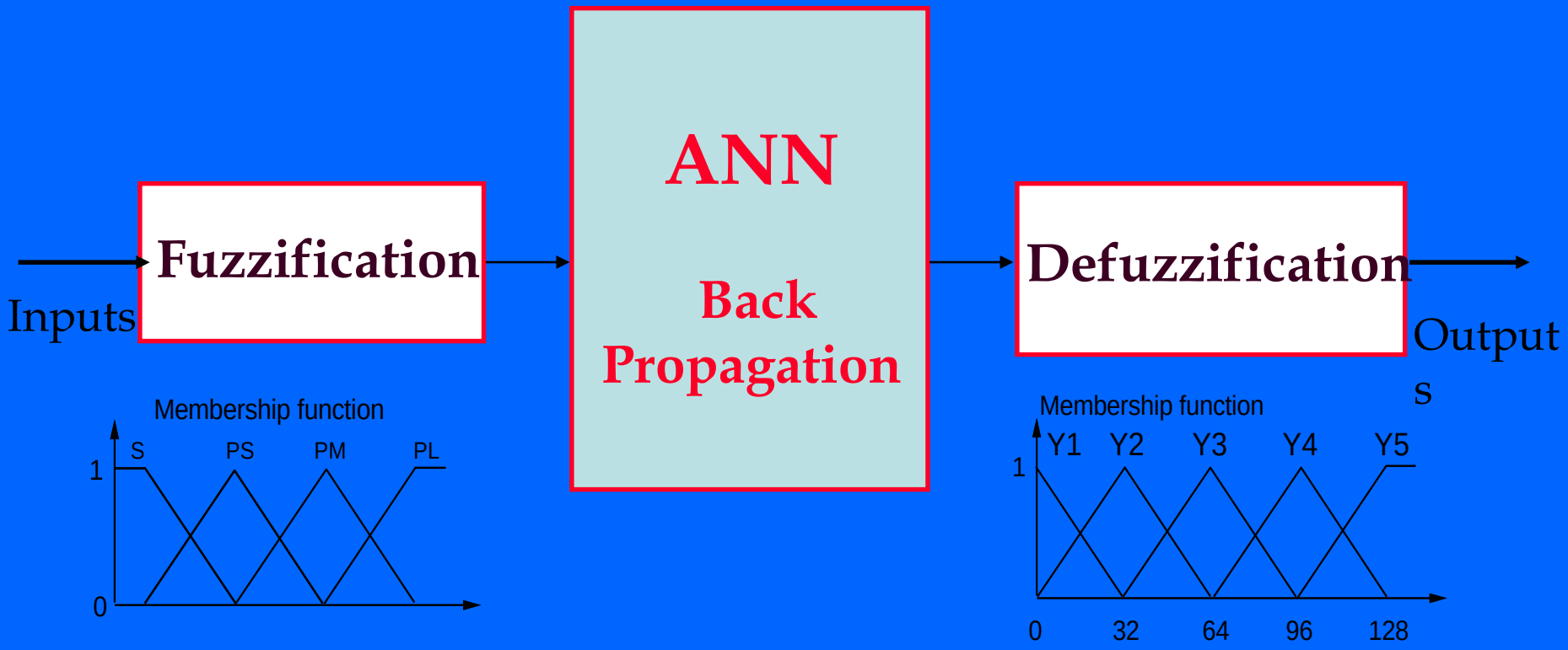
۲ . شبکه هاي عصبي اتفاقي (stochastic) : دراين نوع شبکه ها با فرآيند هاي اتفاقي سروکار داريم. Boltzman Machine داراي توابع اتفاقي (احتمال فعال شدن نرون) هستند.

درشبکه هاي عصبي ذاتا فازي سيگنال فازي درشبکه بکاربرده ميشود

FUZZY NEURAL NETWORK ARCHITECTURE



FUZZY NEURAL NETWORK ARCHITECTURE



Fuzzy Neural Network Structure

شبکه با ابعاد بهينه :

براي بدست آوردن شبکه با ابعاد بهينه از دو روش زير استفاده مي کنيم

۱ . رشد دادن (Network Growing) : که با يک شبکه چند لايه کوچک شروع مي کنيم و در صورت جواب نگرفتن به نرون ها يا لايه ها اضافه مي کنيم.

Cascade – Correlation Learning مثالي از روش رشد دادن است ، در اين روش ابتدا با روش آموزشي مانند LMS و تعدادي نرون و بدون لايه مخفي آموزش شروع ميشود. سپس در حالي که وزن هاي لايه ورودي بدون تغيير باقي مي ماند. به تعداد نرون ها و لايه هاي مخفي تا رسيدن به جواب مطلوب افزوده ميشود.

۲ . هرس کردن **(Network-Pruning)** : با يک شبکه بزرگ شروع مي کنيم وکم کم وزنهاي خاصي را حذف مي کنيم.

در اين بخش دو روش هرس کردن را بررسي مي کنيم:

۱ . تنظيم در ترکيب شبکه (Complexity Regularization)

در اين روش يک معيار براي انتخاب ترکيب مدل (يعني اينکه کدام پارامترها بطور مستقل تنظيم ميشوند) مشخص مي شود.

دو روش مناسب براي انتخاب ترکيب عبارتند از:

الف . روش کمترين طول هندسي (minimum description length)

ب . سنجش بر اساس تئوري اطلاعات (an information theoretic criterion)

هر دو اين روش ها داراي بيان مشترک به صورت زير هستند:

- تفاوت در روش هاي مختلف مربوط به تعيين ضرايب پنالتي است.
- در تمام روش ها به دنبال يافتن روشي هستيم که ضريب ريسک را کاهش مي دهد
- که در آن $\xi_s(w)$ اندازه عملکرد استاندارد است و در پس انتشار MSE (mean square error) مي باشد

$$\begin{pmatrix}\text{Model-complexity}\\ \text{criterion}\end{pmatrix} = \begin{pmatrix}\text{log-likelihood}\\ \text{function}\end{pmatrix} + \begin{pmatrix}\text{model-complexity}\\ \text{penalty}\end{pmatrix}$$

$$R(\mathbf{w}) = \mathscr{E}_r(\mathbf{w}) + \lambda \mathscr{E}_c(\mathbf{w})$$

- $\xi_c(w)$ ضريب پنالتي تركيب است كه فقط به طراحي شبكه و نحوه ارتباط سلول ها با يكديگر وابسته است
- λ پارامتر تنظيم است كه بيان كننده ميزان اهميت نسبي اندازه عملكرد و ضريب پنالتي در تعيين ميزان ريسك شبكه ميباشد
- وقتي λ صفر است به اين معني است كه تركيب شبكه با مثال هايي كه به آن آموزش ميدهند مشخص مي شود وقتي λ بزرگ است به اين معني است كه مثال هاي آموزش دهنده قابل اطمينان نيستند و ضرايب پنالتي تعيين كننده محدوديت ها در تركيب و طراحي شبكه است .

کاهش وزنها (Weight Decaying)

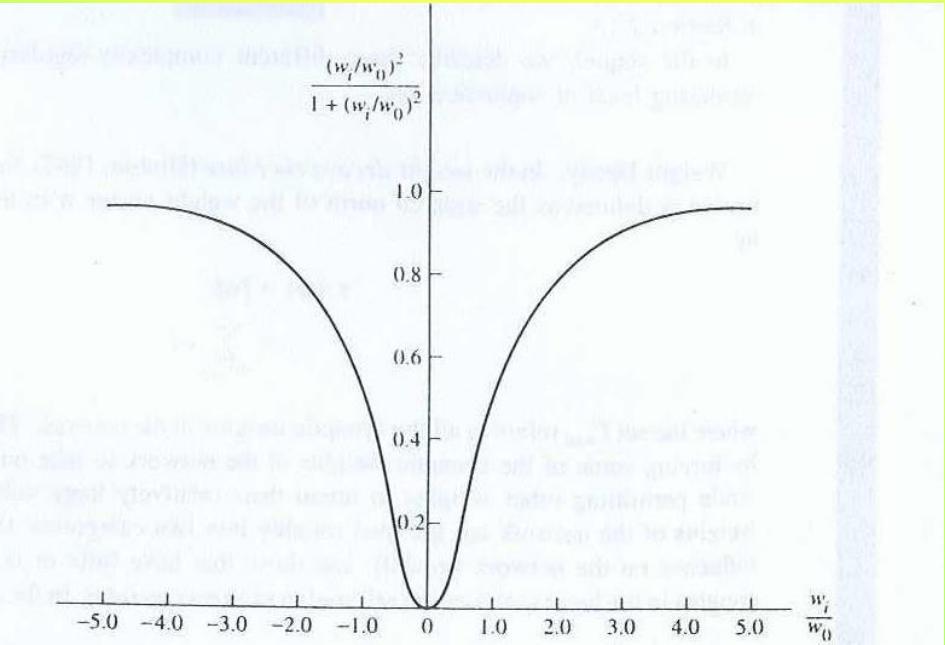
- در این روش تنظیم وزن ها با تعیین نرم کل وزن ها مشخص میشود
- در این روش وزن ها به دو گروه موثر و غیر موثر تقسیم میشوند درغیاب روش تنظیم ترکیب این وزن هاي غیر موثر باعث over fit میشوند . با روش تنظیم ترکیب شبکه این وزن ها کاهش داده میشوند تا شبکه مدل را براي حالت کلي پیدا کند

$$\mathcal{E}_r(\mathbf{w}) = \|\mathbf{w}\|^2 = \sum_{i \in C_{total}} w_i^2$$

- **حذ ف وزن ها**
- **(weight elimination)**
- در این بخش روش تعیین ضرایب پنالتي بیان مي شود

$$\mathcal{E}_r(\mathbf{w}) = \sum_{i \in C_{total}} \frac{(w_i/w_0)^2}{1 + (w_i/w_0)^2}$$

تغییرات ضرایب پنالتي بر اساس $\frac{w_i}{w_0}$ در شکل نشان داده شده است



۱ . هنگامي كه $|\omega_i| << \omega_0$ باشد ضريب پنالتي صفر ميشود. و در مورد آن داده آموزش دهنده بايد وزن حذف شود.

۲ . هنگامي كه $|\omega_i| >> \omega_0$ است آن وزن بايد تقويت شود

اگر n نشان دهنده تكراري باشد كه انجام ميشود.

$R(n)$: ريسک كل

$R(n-1)$: ريسک تكرار قبلي

D: ريسک مطلوب

$$R_{av}(n) = \gamma R_{av}(n-1) + (1-\gamma)R(n)$$

كه در آن $0 \le \gamma \le 1$ است

يكي از عملكرد هاي زير انتخاب ميشود

۱ . افزايش پارامتر با يک مقدار كوچک و مناسب و افزايش پنالتي تركيب كه اين عمل تحت شرايط زير انجام ميشود و در حدود است 10^{-6}

$$R(n) \geq R(n-1), \quad R(n) < R_{av}(n), \quad \text{and} \quad R(n) \gtrless D$$

۲ . كاهش با يک مقدار متعادل كه باعث افزايش عبارت اندازه گيري عملكرد ميشود اين كاهش هنگامي اعمال ميشود كه ريسک كمي افزايش مي يابد ولي در طول آموزش مقداري مناسب دارد.

$$R(n) \geq R(n-1), \quad R(n) \geq R_{av}(n), \quad \text{and} \quad R(n) \geq D$$

۳ . كاهش ناگهاني پارامتر در حدود ۱۰ درصد . اين عمل هنگامي انجام ميشود كه ريسک و مقدار ميانگين آن در حال افزايش باشد.

۲ . انحناي تابع تقريب زننده (curvature – driven smoothing) :

- در اين روش تنظيم تركيب وابسته به خصوصيات تابع تقريب زننده دارد . هر چه اين منحني صاف تر باشد ، كمتر ضريب تنظيم را تحت تاثير قرار ميدهد .

$$\mathscr{E}_c(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{q}\int\left(\frac{\partial^2 y_j(\mathbf{w})}{\partial x_i^2}\right)^2 dx_1 \cdots dx_p$$

- اين روش براي كليت بخشي به شبكه و كاهش حساسيت به اطلاعات ورودي مفيد است.

ماتریس هیسین برای سطح خطا

- هدف این بخش یافتن مصالحه بین ترکیب شبکه و کاهش سطح خطا است. فرض کنید اغتشاش در وزن w_i با تغییر δw_i معادل درتابع هزینه برابر است با

$$\delta \mathscr{E} = \sum_i g_i \, \delta w_i(n) + \frac{1}{2} \sum_i \sum_j h_{ji} \, \delta w_i \, \delta w_j + \text{higher-order terms}$$

- که در آن g_i امین بردار گرادیان و h_{ji} امین مولفه ماتریس هیسین است

$$g_i = \frac{\partial \mathscr{E}_i}{\partial w_i}$$

$$h_{ji} = \frac{\partial^2 \mathscr{E}_i}{\partial w_j \, \partial w_i}$$

۱ . تقريب كرانه اي

فرض بر اين است كه پارامترها به ازاي هر مينيمم محلي و مطلق يک مقدار مشخص دارند.
بنابراين ميتوان فرض كرد كه صفر است

۲ . تقريب درجه دوم

فرض بر اين است كه حداكثر درجه موجود برابر دو است

$$\delta\mathscr{E} \simeq \frac{1}{2}\sum_i \sum_j h_{ji}\,\delta w_i\,\delta w_j$$
$$= \frac{1}{2}\,\delta \mathbf{w}^T\,\mathbf{H}\,\delta \mathbf{w}$$

δw_i اغتشاش در وزن ها و H ماتريس هيسين حاوي تمام مشتقات درجه دوم است.

حذف وزن به شرط زير است:

$$\mathbf{1}_i^T\,\delta \mathbf{w} + w_i = 0$$

براي حل مسئله بهينه سازي بالا معادله لاگرانژ زير را حل مي كنيم

$$\delta \mathbf{w} = -\frac{w_i}{[\mathbf{H}^{-1}]_{ii}}\,\mathbf{H}^{-1}\mathbf{1}_i$$

آموزش با نظارت برای تشخیص مدل غیر خطی

این روش راه حل مسئله است که هدف آن مینیمم کردن یک تابع هدف است . تابع هزینه به عنوان عبارتي از انحراف خروجي از خروجي مطلوب در نظر گرفته میشود . پارامتر مطلوب وزن ها هستند و هدف این است که خروجي شبکه به مقدار مطلوب نزدیک شود.

روش معمول پس انتشار در حالت بلادرنگ استفاده از گرادیان خطا نسبت به پارامتر مورد نظر است .

به علت کندي ذاتي LMS از روش RLS استفاده مي شود که شکل خاصي از فیلتر کالمن است .

مزاياي RLS

۱. سرعت همگرايي زياد
۲. نرخ يادگيري داخلي
۳. حساس نبودن به شرايط عددي اطلاعات ورودي که به نسبت بزگترين مقدار ويژه به کمترين مقدار ويژه ماتريس کوواريانس اطلاعات ورودي وابسته است.

آموزش کالمن گسترش یافته با استفاده از BP

فرض کنید که در یک شبکه پارامتر آزاد همه نرون هاي شبکه وزن هاي بین آنها باشد تابع هزینه به صورت زیر است:

$$\mathscr{E}_{av}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\sum_{j\in C}[d_j(n) - y_j(n)]^2$$

حال با خطي سازي تابع هزینه در هر نقطه کار الگوریتم فیلتر خطي کالمن را پیاده مي کنیم رفتار یک سلول در طول آموزش مانند یک سیستم غیر خطي دینامیک است که معادلات آن طبق روش کالمن به فرم زیر است

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n)$$

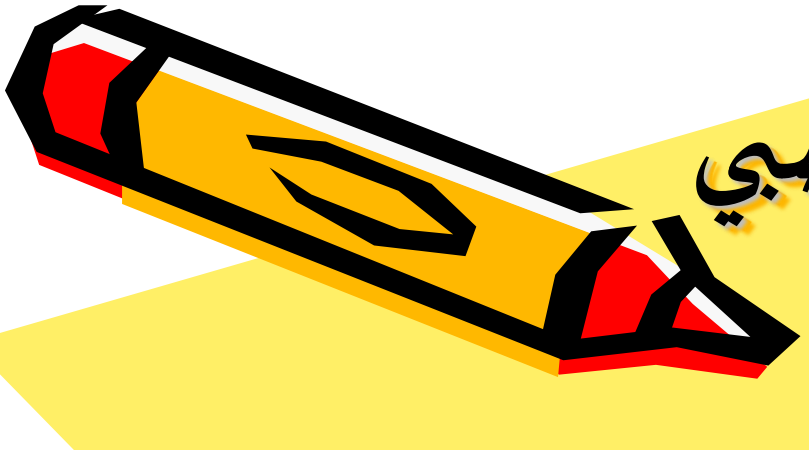
$$d_i(n) = \varphi(\mathbf{x}_i^T(n)\mathbf{w}_i(n)) + e_i(n)$$

کاربرد شبکه های عصبی مصنوعی در سیستمهای قدرت

- *پیش بینی و کنترل*
- – پایداری گذرا
- – پایداری حالت مانا
- – پیش بینی بار
- – مدلسازی بارهای دینامیک
- – کنترل توان راکتیو
- *شناسایی*
- – حفاظت
- – تشخیص خطا
- *دسته بندی*
- – پروسه آلارمها
- – بررسی شرایط اضطراری و دقت خطا
- – *بهینه سازی*
- – پخش بار بهینه
- – برنامه ریزی تولید
- – پخش بار اقتصادی
- – در مدار قرار گرفتن نیروگاهها

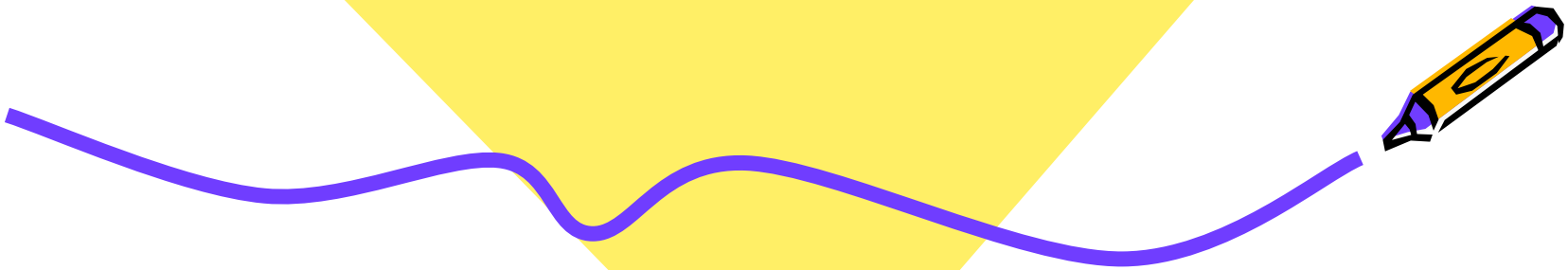


شبکه هاي عصبي



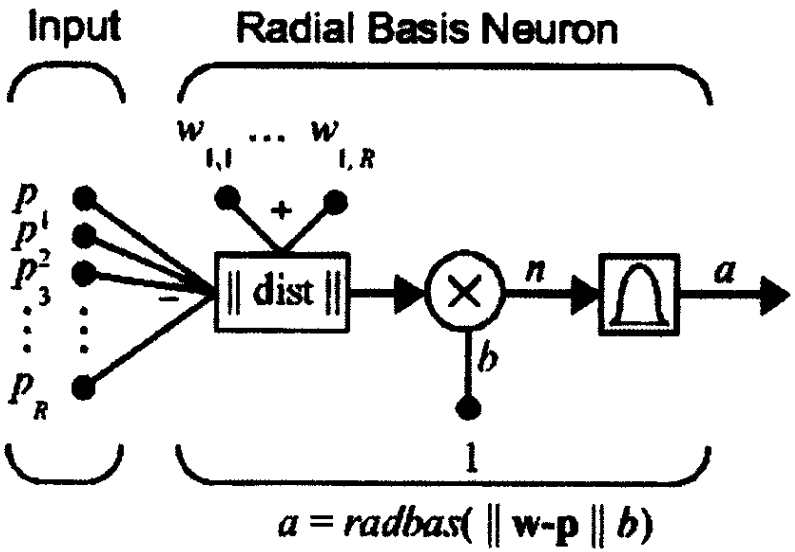
شبکه هاي RBF

دکتر جورابيان



شبکه های با پایه شعاعی
Radial Basis Function

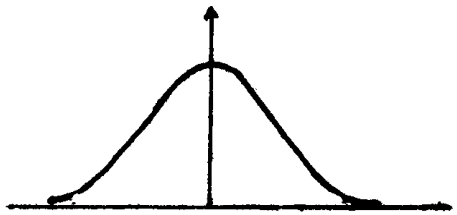
- تعليم بدون نظارت (ورودي)
- تعليم با نظارت (تنظيم وزنها)



شبکه های با پایه شعاعی

شبکه های RBF علاوه بر لایه ورودی که متشکل از گره های منبع است شامل دو لایه دیگر نیز هستند

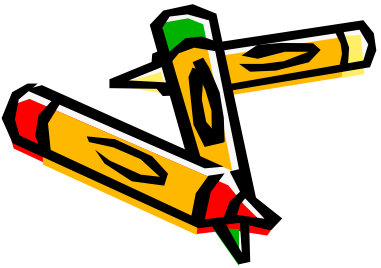
- یک لایه پنهان که شامل S1 سلول است و از توابع غیر خطی استفاده می کند
- لایه دیگر که لایه خروجی خطی شامل S2سلول (نرون) است که S2 بیان کننده تعداد خروجی های شبکه آموزش یافته می باشد
- تابع تبدیل استفاده شده جهت سلول بصورت زیر می باشد :
- rad bas (n) = e^{n2}



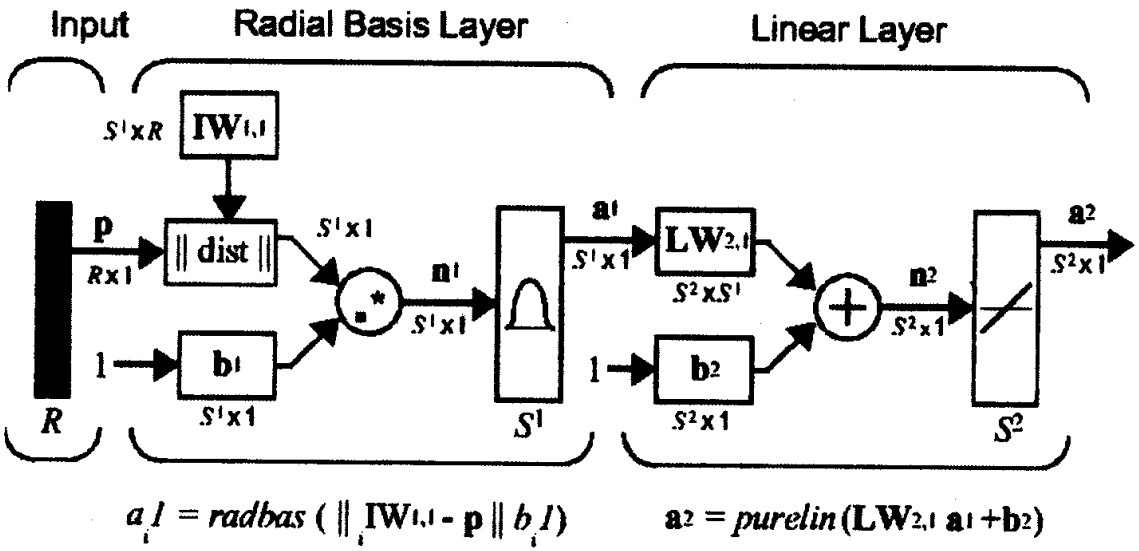


ورودی آن یعنی تفاضل (w - p) در حداقل مقدار خود صفر و در حداکثر مقدار خود 1 می شود و با افزایش این تفاضل میزان تابع کاهش می یابد.

- این نوع شبکه ممکن است در لایه مخفی به تعداد بیشتری نرون (سلول) نسبت به شبکه های پیشخورد نیاز داشته باشد ولی معمولاً در مدت زمان بسیار کمتری می تواند به خطای دلخواه رسیده و همگرا شود
- با استفاده از خروجی های محاسبه شده در لایه مخفی در پاسخ به یک بردار ورودی و با استفاده از یک پاسخ مطلوب در خروجی ، وزنها با روش آموزش بانظارت معین می شوند.



ساختار کلی شبکه RBF



Where...

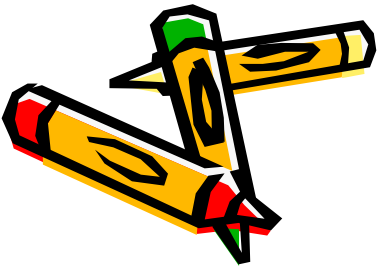
R = # elements in input vector

S^1 = # Neurons in layer 1

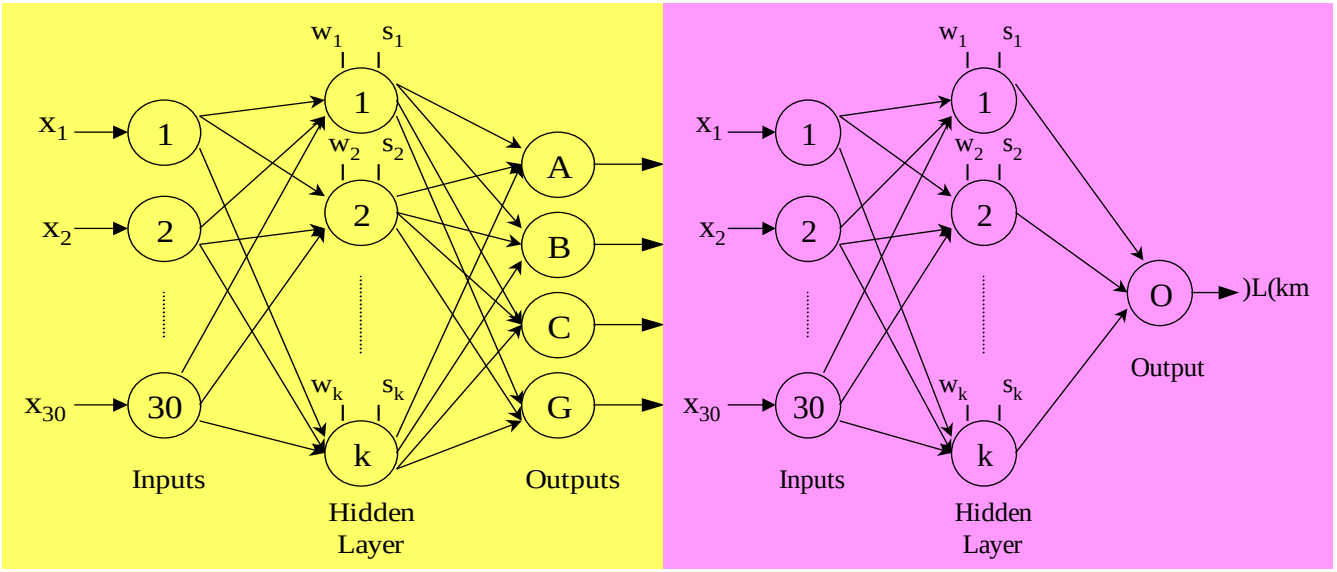
S^2 = # Neurons in layer 2

فرآیند نگاشت ورودی- خروجی در شبکه RBF در دو مرحله انجام می شود :

- الف - یک تبدیل غیر خطی که فضای ورودی مختلط را به فضای واسطه حقیقی نگاشت می کند.
- ب - یک تبدیل خطی که فضای واسطه حقیقی را به فضای خروجی مختلط نگاشت می کند.



شبكه هاي RBF



x_1
x_2
x_{30}
1
2
30
Inputs
w_1 s_1
w_2 s_2
w_k s_k
1
2
k
Hidden Layer
A
B
C
G
Outputs
x_1
x_2
x_{30}
1
2
30
Inputs
w_1 s_1
w_2 s_2
w_k s_k
1
2
k
Hidden Layer
O
Output
)L(km

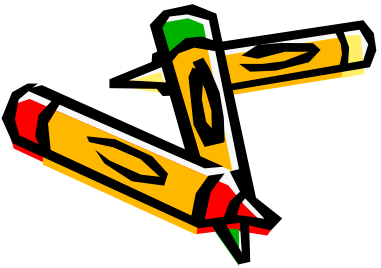
مقایسه شبکه های عصبی پیشخورد (پرسپترون) و شبکه های عصبی مبتنی بر توابع پایه شعاعی

- شبکه های عصبی مبتنی بر توابع پایه شعاعی (RBF) تنها یک لایه مخفی دارد، در حالیکه شبکه های پیشخورد می تواند دارای یک یا چند لایه مخفی باشد
- در شبکه های RBF توابع انتقال بین لایه ورودی و لایه مخفی غیر خطی اند و توابع انتقال بین لایه مخفی و لایه خروجی خطی هستند ولی در شبکه های پرسپترون چند لایه، نگاشت بین هر لایه مخفی و لایه قبلی اش غیر خطی است و توابع انتقال لایه خروجی ممکن است خطی یا غیر خطی باشند
- سلولهای مخفی در شبکه های RBF در واقع فاصله (اندازه هندسی یا اندازه اقلیدسی) بین بردار ورودی و وزنهای مذکور آنرا پیدا می کند، در حالیکه در شبکه های پیشخورد سلول مخفی، حاصلضرب داخلی بردار ورودی و بردار وزن آن سلول را مشخص می کند

مقایسه شبکه های عصبی پیشخورد (پرسپترون) و شبکه های عصبی مبتنی بر توابع پایه شعاعی

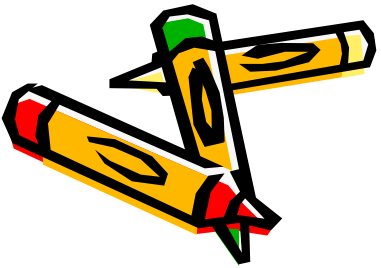


- شبکه پیشخورد تقریب عمومی برای نگاشت غیرخطی ورودی – خروجی پیدا می کند، این تقریب حتی برای قسمتهایی از فضای ورودی صادق است که هیچگونه اطلاعاتی از آن قسمت شبکه داده نشده باشد
- شبکه RBF، تقریب های غیرخطی مکانی برای نگاشت ورودی – خروجی پیدا می کند، در نتیجه سرعت یادگیری شبکه نسبت به شبکه پیشخورد بیشتر خواهد بود و همچنین شبکه RBF نسبت به ترتیب ارائه ورودیها کمتر حساس است ولی برای بدست آوردن یک تابع هموار ممکن است تعداد سلولهای لایه مخفی زیادتر شود
- شبکه های عصبی RBF و شبکه های پرسپترون چند لایه دارای این خاصیت مشترک هستند که هر دو یک تقریب زن پیشخورد هستند و هردو نگاشت ورودی – خروجی ارائه می دهند



مقایسه شبکه های عصبی پیشخورد (پرسپترون) و شبکه های عصبی مبتنی بر توابع پایه شعاعی

- شبکه های RBF در حل مسائل کلاسه بندی الگوهای پیچیده استفاده و کاربرد بسیاری دارند
- گروههایی که بطور تحقیقاتی بر روی شبکه های RBF مطالعاتی انجام داده اند معمولاً روند یادگیری شبکه را به دو مرحله تقسیم کرده اند
- الف – یادگیری مراکز و پهنای توابع شعاعی در لایه مخفی
- ب – یادگیری وزنهای متصل از لایه میانی به لایه خروجی
- نحوه انتخاب نرونها در این نوع شبکه نیز معمولاً بصورت آزمایش و خطاست بدین ترتیب که ابتدا الگوریتم یادگیری با تعداد زیاد نرون در لایه مخفی شروع شده و سپس تا جائی که بتوان از این مقدار نرونها کاسته می شود. این کاستن تعداد نرونها با کمینه کردن میزان خطای محاسباتی همراه است و در خطای مطلوب که با تست کردن ورودیهای آموزش نیافته بدست می آید الگوریتم آموزش به پایان می رسد



Consider a family of surfaces where each naturally divides an input space into two regions. Let $\mathcal{X}$ denote a set of N patterns (vectors) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, each of which is assigned to one of two classes $\mathcal{X}_1$ and $\mathcal{X}_2$. This *dichotomy* (binary partition) of the points is said to be separable with respect to the family of surfaces if a surface exists in the family that separates the points in the class $\mathcal{X}_1$ from those in the class $\mathcal{X}_2$. For each pattern $\mathbf{x} \in \mathcal{X}$, define a vector made up of a set of real-valued functions $\{\varphi_i(\mathbf{x}) | i = 1, 2, \ldots, m_1\}$, as shown by

$$\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T \tag{5.1}$$

Suppose that the pattern $\mathbf{x}$ is a vector in an m_0-dimensional input space. The vector $\boldsymbol{\varphi}(\mathbf{x})$ then maps points in m_0-dimensional input space into corresponding points in a new space of dimension m_1. We refer to $\varphi_i(\mathbf{x})$ as a *hidden function*, because it plays a role similar to that of a hidden unit in a feedforward neural network. Correspondingly, the space spanned by the set of hidden functions $\{\varphi_i(\mathbf{x})\}_{i=1}^{m_1}$ is referred to as the *hidden space* or *feature space*.





A dichotomy $\{\mathcal{X}_1, \mathcal{X}_2\}$ of $\mathcal{X}$ is said to be φ-*separable* if there exists an m_1-dimensional vector $\mathbf{w}$ such that we may write (Cover, 1965)

$$\begin{aligned} \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) &> 0, \qquad \mathbf{x} \in \mathcal{X}_1 \\ \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) &< 0, \qquad \mathbf{x} \in \mathcal{X}_2 \end{aligned} \tag{5.2}$$

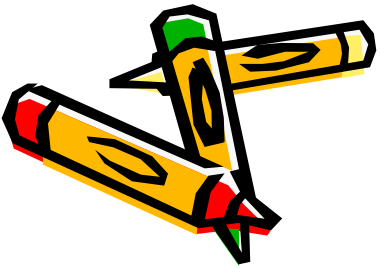
The hyperplane defined by the equation

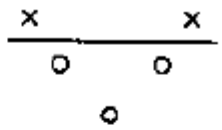
$$\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) = 0$$

describes the separating surface in the φ-space (i.e., hidden space). The inverse image of this hyperplane, that is,

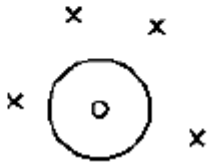
$$\mathbf{x}: \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) = 0 \tag{5.3}$$

defines the *separating surface* in the input space.

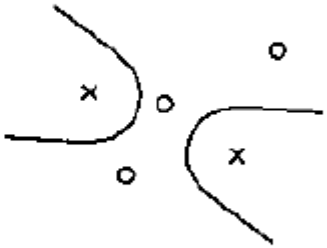




(a)



(b)



(c)

FIGURE 5.1 Three examples of φ-separable dichotomies of different sets of five points in two dimensions: (a) linearly separable dichotomy; (b) spherically separable dichotomy; (c) quadrically separable dichotomy.

XOR-Problem

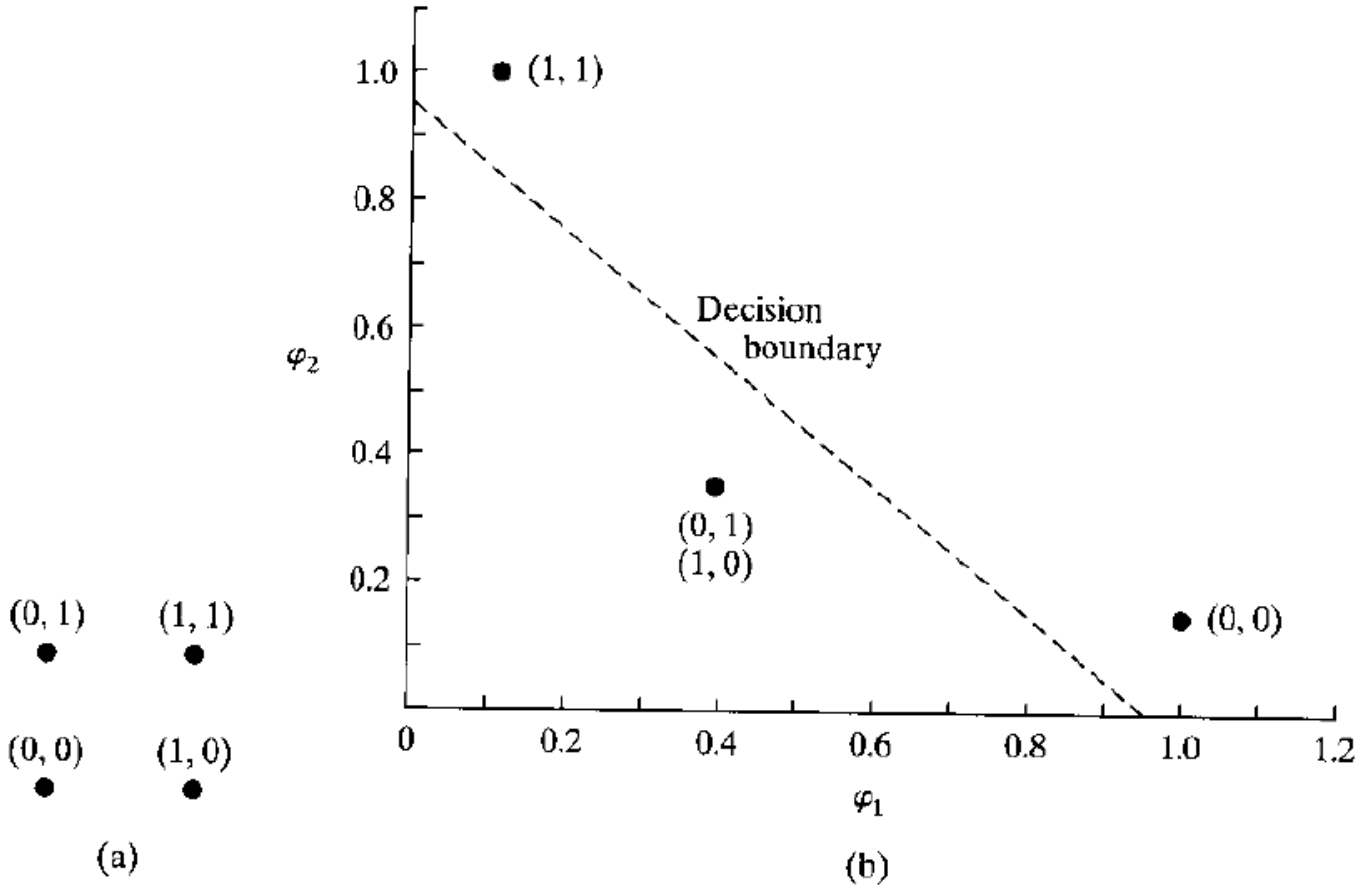


FIGURE 5.2 (a) The four patterns of the XOR problem; (b) Decision-making diagram.

Define a pair of Gaussian hidden functions as follows:

$$\varphi_1(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_1\|^2}, \qquad \mathbf{t}_1 = [1, 1]^T$$

$$\varphi_2(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_2\|^2}, \qquad \mathbf{t}_2 = [0, 0]^T$$

TABLE 5.1 Specification of the Hidden Functions for the XOR Problem of Example 5.1

Input Pattern, $\mathbf{x}$	First Hidden Function, $\varphi_1(\mathbf{x})$	Second Hidden Function, $\varphi_2(\mathbf{x})$
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678

We may then construct the results summarized in Table 5.1 for the four different input patterns of interest. The input patterns are mapped onto the $\varphi_1 - \varphi_2$ plane as shown in Fig. 5.2b. Here we now see that the input patterns (0, 1) and (1, 0) are linearly separable from the remaining input patterns (1, 1) and (0, 0). Thereafter, the XOR problem may be readily solved by using the functions $\varphi_1(\mathbf{x})$ and $\varphi_2(\mathbf{x})$ as the inputs to a linear classifier such as the perceptron.

■

Given a set of N different points $\{\mathbf{x}_i \in \mathbb{R}^{m_0} | i = 1, 2, \ldots, N\}$ *and a corresponding set of N real numbers* $\{d_i \in \mathbb{R}^1 | i = 1, 2, \ldots, N\}$, *find a function* $F: \mathbb{R}^N \to \mathbb{R}^1$ *that satisfies the interpolation condition:*

$$F(\mathbf{x}_i) = d_i, \qquad i = 1, 2, \ldots, N \tag{5.10}$$

The *radial-basis functions* (RBF) technique consists of choosing a function F that has the following form (Powell, 1988):

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{5.11}$$

where $\{\varphi(\|\mathbf{x} - \mathbf{x}_i\|) | i = 1, 2, \ldots, N\}$ is a set of N arbitrary (generally nonlinear) functions, known as *radial-basis functions*, and $\|\cdot\|$ denotes a *norm* that is usually Euclidean. The known data points $\mathbf{x}_i \in \mathbb{R}^{m_0}$, $i = 1, 2, \ldots, N$ are taken to be the *centers* of the radial-basis functions.

Inserting the interpolation conditions of Eq. (5.10) in (5.11), we obtain the following set of simultaneous linear equations for the unknown coefficients (weights) of the expansion $\{w_i\}$:

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \tag{5.12}$$

where

$$\varphi_{ji} = \varphi(\|\mathbf{x}_j - \mathbf{x}_i\|), \quad (j, i) = 1, 2, \ldots, N \tag{5.13}$$

Let

$$\mathbf{d} = [d_1, d_2, \ldots, d_N]^T$$
$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T$$

The N-by-1 vectors $\mathbf{d}$ and $\mathbf{w}$ represent the *desired response vector* and *linear weight vector,* respectively, where N is the *size of the training sample.* Let $\mathbf{\Phi}$ denote an N-by-N matrix with elements φ_{ji}:

$$\mathbf{\Phi} = \{\varphi_{ji} | (j, i) = 1, 2, \ldots, N\} \tag{5.14}$$

We call this matrix the *interpolation matrix.* We may then rewrite Eq. (5.12) in the compact form

$$\mathbf{\Phi w} = \mathbf{x} \tag{5.15}$$

Assuming that $\mathbf{\Phi}$ is nonsingular and therefore that the inverse matrix $\mathbf{\Phi}^{-1}$ exists, we may go on to solve Eq. (5.15) for the weight vector $\mathbf{w}$ as shown by

$$\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{x} \tag{5.16}$$

The vital question is: How can we be sure that the interpolation matrix $\mathbf{\Phi}$ is nonsingular? It turns out that for a large class of radial-basis functions and under certain conditions, the answer to this question is given in the following important theorem.

$$F(x)=\sum_{i=1}^{N} W_i\,\varphi(\|\,x-x_i\,\|) \qquad W=\Phi^{-1}d$$

تئوري تنظيم كننده :

جواب مساله با شرايط نامطلوب را به كمک چند مقدار غير منفي پايدار نمود.

مثال :

فرض كنيد داده هاي ورودي و خروجي به صورت زير باشد:

سيگنال ورودي $\quad x_i \in R^p \quad i=1,\ldots,N$

پاسخ دلخواه $\quad d_i \in R^1 \quad i=1,\ldots,N$

تابع تقريب زننده $\mathbf{F(x)}$ با به حداقل رساندن يک تابع هزينه به دست مي آيد كه داراي ۲ جمله است. جمله خطاي استاندارد و جمله تنظيم كننده.

$$\varepsilon_s(F)=\frac{1}{2}\sum_{i=1}^{N}(d_i-y_i)^2=\frac{1}{2}\sum_{i=1}^{N}\big(d_i-F(x_i)\big)^2$$

جمله دوم به خصوصيات هندسي تابع تقريب بستگي دارد :

$$\varepsilon_C(F) = \frac{1}{2} \| PF \|^2$$

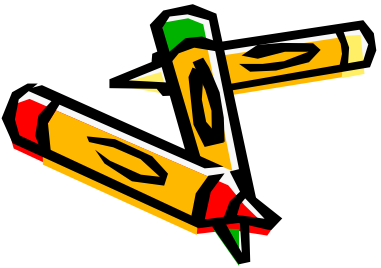
P اپراتور خطي مشتقگير است.

تابع هزينه كل براي به حداقل رساندن جمله زير استفاده مي شود:

$$\varepsilon(F) = \varepsilon_S(F) + \lambda \varepsilon_C(F) = \frac{1}{2} \sum_{i=1}^{N} \left(d_i - F(x_i)\right)^2 + \frac{1}{2} \lambda \| PF \|^2$$

مساله :

مطلوب است تعيين تابع **F(x)** به طوري كه تابع هزينه كل حداقل شود



براي گرفتن مشتق از $\varepsilon(F)$ از قاعده مشتق **Frechet** استفاده مي كنيم.

$$\varepsilon'(F)\,dF \overset{\Delta}{=} d\varepsilon(F,dF) = \lim_{\beta\to 0}\frac{\varepsilon(F+\beta h)-\varepsilon(F)}{\beta} = \frac{d\varepsilon(F+\beta h)}{d\beta}\Big|_{\beta=0}$$

بنابراين $$d\varepsilon(F,\beta h) = d\varepsilon_S(F,h) + \lambda d\varepsilon_C(F,\beta h)$$

$$d\varepsilon_S(F,h) = \left(\frac{d}{d\beta}\varepsilon_S(F,\beta h)\right)_{\beta=0}$$

$$= \left\{\frac{1}{2}\frac{d}{d\beta}\sum_{i=1}^{N}\left[di - F(x_i) - \beta h(x_i)\right]^2\right\}_{\beta=0}$$

$$= -\sum_{i=1}^{N}\left[di - F(x_i) - \beta h(x_i)\right]h(x_i)\Big|_{\beta=0}$$

$$d\varepsilon_S(F,h) = -\sum_{i=1}^{N}\left[di - F(x_i)\right]h(x_i)$$

رابطه بالا را مي توان بصورت ضرب داخلي در فضاي **هيلبرت** نوشت.

بنا به تعريف فضاي هيلبرت فضاي *كامل* ضرب داخلي است. يعني هر اندازه (نرم) اي در داخل آن فضا قرار دارد.

براي اين مساله ضرب داخلي به اين صورت تعريف مي شود:

$$\big(h(x), F(x)\big)_H = \int_{R^P} h(x)\, F(x)\, dx$$

براي حالت بالا:

$$\left(h(x), \sum_{i=1}^{N} \big(di - F(x)\big)\delta(x - x_i) \right)_H$$

$$= \int_{R^P} h(x) \sum_{i=1}^{N} \big(di - F(x)\big)\delta(x - x_i)\, dx$$

$$= \sum_{i=1}^{N} h(x_i)\big(di - F(x_i)\big)$$

$$d\varepsilon_S(F,h) = -\left(h(x), \sum_{i=1}^{N} (di - F(x))\delta(x - x_i) \right)$$

$$d\varepsilon_S(F,h) = \left[\frac{d}{d\beta} \varepsilon_C \left(F(x) - \beta h(x) \right) \right]_{\beta=0}$$

$$= \frac{1}{F} \frac{d}{d\beta} \int_R \left[P\left(F(x) + \beta h(x) \right) \right]^2 dx \,|_{\beta=0}$$

$$d\varepsilon_C(F,h) = \int_R P\left[F(x) + \beta h(x) \right] Ph(x) dx \,|_{\beta=0}$$

$$\int_R PF(x) Ph(x) dx \,|_{\beta=0} = \left(PF(x), Ph(x) \right)_H$$

برای جمله تنظِم کننده:

برای نرم (اندازه) **PF** از انتگرال استفاده می کنیم

با توجه به تعریف اپراتور الحاقی **(joint operator)**

$$d\varepsilon(F,h) = \left(h(x), P^+ PF(x) \right)_H$$

برای اپراتور **P** می‌توان یک اپراتور الحاقی **P^+** تعریف کرد که ویژگی زیر را داشته باشد:

$$(Ph, F)_H = (H, P^+ F)_H \qquad (۱)$$

مثال : **P** یک ماتریس است

$$(Px,Z)_H=(Ax,Z)_H$$

$$(Px,Z)_H=Z^T(Ax)=(A^TZ^T)x=(x,A^TZ) \qquad (۲)$$

از (۱) و (۲) داریم

$$P^+Z=A^TZ$$

مثال : **P** یک مشتقگیر است

$$Pf=f'(x)$$

$$(Pf,y)_H=\int_a^b f'(x)y(x)dx=f(x)g(x)\Big|_a^b-\int_a^b f(x)g'(x)dx$$

$$(Pf,y)_H=-\int_a^b f(x)y'(x)dx=\big(f,F(y)\big)_H$$

بنابراین $P^+y=-y'(x)$ پس $P^+=-P$

$$d\varepsilon_S(F,h)+\lambda d\varepsilon_C(F,h)=0$$

$$-(h,\sum_{i=1}^{N}\big(di-F(x_i)\big)\delta(x-x_i)+\lambda(h+P^+PF)_H=0$$

$$\left[h,\lambda P^+PF(x)-\sum_{i=1}^{N}\big(di-F(x_i)\big)\delta(x-x_i)\right]_H=0$$

براي اينكه ضرب داخلي معادله آخر برابر صفر شود، بايد جمله دوم نيز صفر شود.

$$\lambda P^{+}PF(x)-\sum_{i=1}^{N}\big(di-F(x_i)\big)\delta(x-x_i)=0$$

معادله اولر لاگرانژ

$$P^{+}PF(x)=\frac{1}{\lambda}\sum_{i=1}^{N}\big(di-F(x_i)\big)\delta(x-x_i)$$

اپراتور $P^{+}P$ را اپراتور خودالحاقي مشتقگير گويند.

براي حل معادله ديفرانسيل فوق از تابع گرين استفاده مي شود. تابع $G(x,x_i)$ را بعنوان تابع گرين كه مركز آن x_i است را در نظر مي گيريم. بر طبق تعريف چنانچه اپراتور مشتقگير بر روي تابع گرين $P^{+}P$ اعمال شود، در اين صورت

$$P^{+}PG(x,x_i)=\delta(x-x_i)$$

و در نتيجه

$$P^{+}PF(x)=\frac{1}{\lambda}\sum_{i=1}^{N}\big(di-F(x)\big)\delta(x-x_i)=\frac{1}{\lambda}\sum_{i=1}^{N}\big(di-F(x_i)\big)P^{+}PG(x,x_i)$$

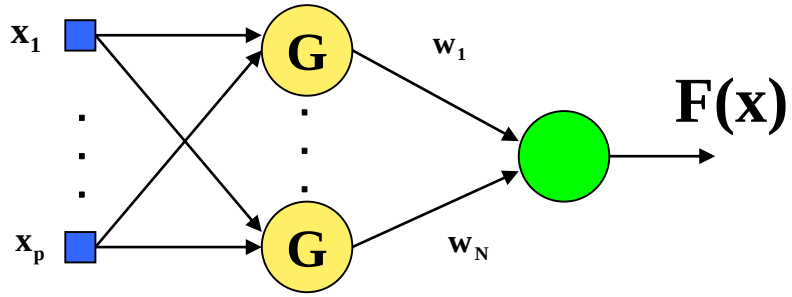
$$F(x)=\frac{1}{\lambda}\sum_{i=1}^{N}\big(di-F(x_i)\big)G(x,x_i)$$

بنا بر اين جواب مساله تنظيم كننده، **F(x)** است كه عبارتست از جمع آثار **N** تابع گرين ضرب در ضريب $\frac{1}{\lambda}(di - F(x_i))$

در مورد استفاده از شبكه **RBF** جهت يافتن **F(x)** (تابع درونيابي براي **N** نقطه داده شده) را مي توان به صورت زير در نظر گرفت

$$w_i = \frac{1}{\lambda}\left(di - F(x_i)\right) \quad , i = 1,\ldots, N$$

$$F(x) = \sum_{i=1}^{N} w_i G(x, x_i)$$



$$F(x_j)=\sum_{i=1}^{N} w_i G(x_j,x_i) \quad j=1,\ldots,N$$

$$F=\left[F(x_1),\ldots,F(x_N)\right]^T \quad ,d=\left[d_1,\ldots,d_N\right]^T$$

یافتن w_i ها

$$G=\begin{bmatrix} G(x_1,x_1) & \cdots & G(x_1,x_N) \\ \vdots & & \vdots \\ G(x_N,x_1) & \cdots & G(x_N,x_N) \end{bmatrix} \quad ,w=\left[w_1,\cdots,w_N\right]^T$$

$$w=\frac{1}{\lambda}(d-F) \quad ,F=Gw \quad \Rightarrow w=\frac{1}{\lambda}(1-Gw) \quad \Rightarrow Gw+\lambda w=d$$

$$w=\left(G+\lambda I\right)^{-1}d$$

هر چه مقادير روي قطب بزرگتر باشد ماتريس معكوس بهتري حاصل مي شود. در اينجا λI مقدار دلخواهي است كه به تابع اضافه مي شود و مي تواند بزرگ انتخاب شود تا معكوس پذيري بهتر شود.

ماتريس **G** يک ماتريس متقارن است و براي يک سري از توابع گرين مثبت معين است (به شرطي كه نقاط $x_1, x_2, \ldots, x_n$ هم مجزا باشند).

با مقايسه با رابطه قبلي براي تعيين **W**، $W = \varphi^{-1} d$ در اينجا اين امكان وجود دارد كه با انتخاب به مقدار كافي بزرگ همواره ماتريس $G + \lambda I$ مثبت معين باشد. بنابراين قابل معكوس كردن است.

انتخاب تابع گرين معمولا بستگي به اپراتور **P** دارد.

مثال :

$$\| PF \|^2 = \sum_{k=0}^{N} a_k \| D^k F(x) \|^2$$

$$\| D^k F(x) \|^2 = \sum_{\alpha=k} \int_{R^P} | \delta^\alpha F(x) |^2$$

انتخاب تابع گرين مناسب براي اپراتور D^k بالا

$$G(x, x_i) = e^{-\frac{1}{2\delta_i^2}\| x - x_i \|^2}$$

$$\| x - x_i \|^2 = \sum_{i=1}^{p} (x_k - x_{ik})^2$$

محاسبه تابع **F(x)** و بردار **w** در اين حالت برابر است با در نظر گرفتن **N** تابع گرين و معكوس كردن ماتريس N*N. در صورتيكه تعداد نمونه هاي ورودي (N) خيلي زياد باشد، بدست آوردن جواب دقيق مشكل خواهد بود. براي ساده كردن محاسبات مي توان از تابع در لايه مخفي استفاده كرد.

بنابراین مقدار تقریبی تابع درونیابی مورد نظر برابر است با

$$\hat{F}(x) = \sum_{i=1}^{M} w_i G(x, t_i)$$

که در آن مراکز t_i (یا $i = 1, \cdots, M$) تعیین شوند.

چنانچه تابع گرین بستگی به اندازه فاصله بین X و t_i داشته باشد، $G(\| x - t_i \|)$ را این صورت تابع هزینه :

$$\varepsilon(\hat{F}) = \frac{1}{F}\sum_{i=1}^{N}\left(di - \sum_{j=1}^{M} \nu_j G(\| x_i - x_j \|) \right)^2 + \frac{1}{2}\lambda \| P\hat{F} \|^2$$

$$\| P\hat{F} \|^2 = (P\hat{F}, P\hat{F})_H = (\hat{F}, P^+ P\hat{F})_H$$

$$G = \begin{bmatrix} G(x_1,t_1) & \cdots & G(x_1,t_M) \\ \vdots & \ddots & \vdots \\ G(x_N,t_1) & \cdots & G(x_N,t_M) \end{bmatrix}_{N\times M|}$$

$$\| P\hat{F} \|^2 = \left[\sum_{i=1}^{M} w_i G(x,t_i)\ P^+ P \sum_{i=1}^{M} w_i G(x,t_i) \right]_H$$

$$= \left[\sum_{i=1}^{M} w_i G(x,t_i)\ , \sum_{i=1}^{M} w_i \delta(x-t_i) \right]_H$$

$$= \sum_{i=1}^{M}\sum_{j=1}^{M} w_j w_i G(t_j,t_i) = W^T G_0$$

$$G = \begin{bmatrix} G(t_1,t_1) & \cdots & G(t_1,t_M) \\ \vdots & \ddots & \vdots \\ G(t_M,t_1) & \cdots & G(x_M,t_M) \end{bmatrix}$$

در نتیجه با اعمال مشتق Frichet بر روي $\varepsilon(w)$ داریم:

$$\varepsilon(w) = \frac{1}{2} \| d - G(\| w \|^2) + \frac{1}{2} \lambda w^T G_0 w \|$$

$$d\varepsilon(w,h) = -h^T G^T (d - Gw) + \lambda h^T G_0 w = 0$$

$$\left(G^T G + \lambda G_0\right) w = G^T d$$

$$w = \left(G^T G - \lambda G_0\right)^{-1} G^T d \xrightarrow{\lambda=0} w = \left(G^T G\right)^{-1} G^+ d$$

$$Ax = y \rightarrow A^T Ax = A^T y \rightarrow \left(A^T A\right)^{-1}\left(A^T A\right)x = \left(A^T A\right)^{-1} A^T y$$

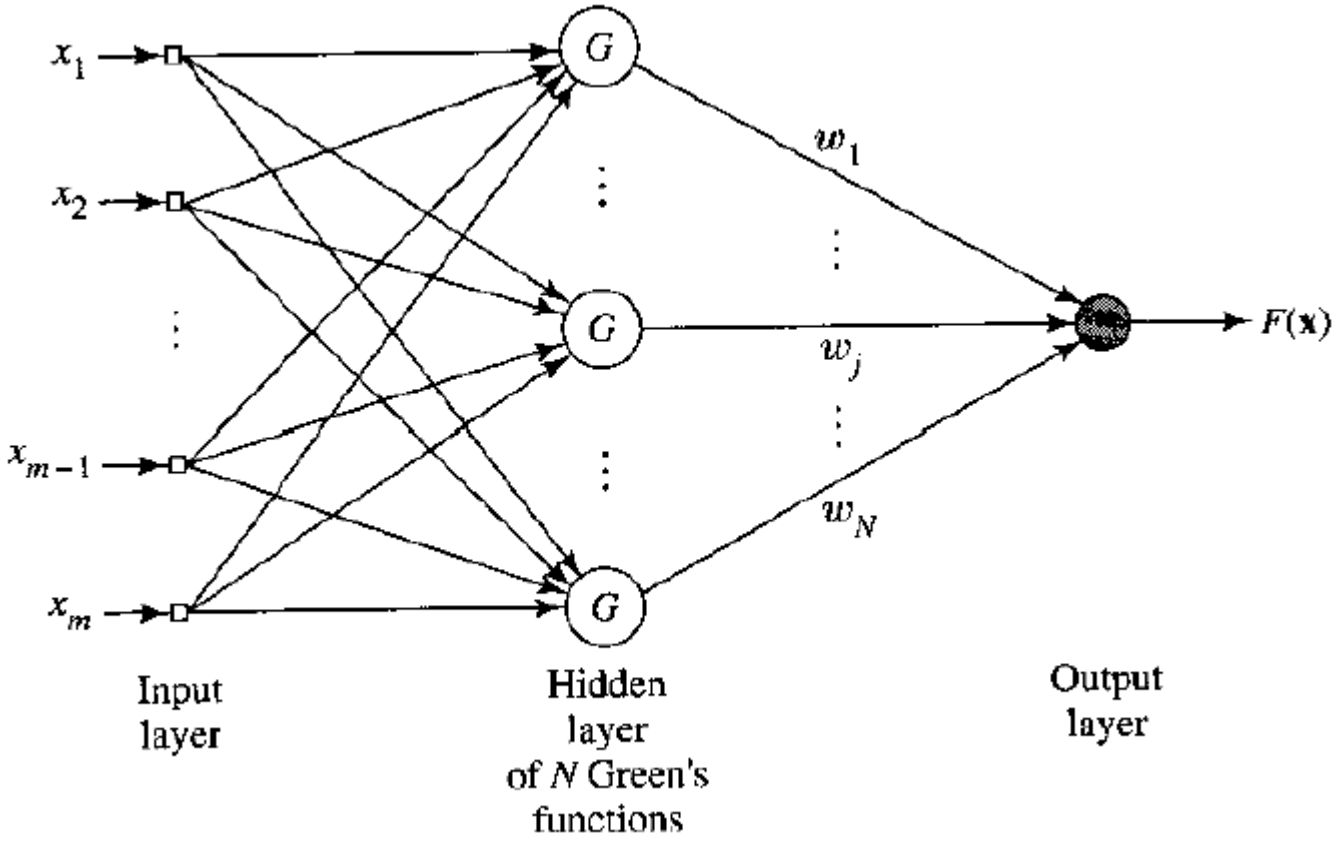
$$\Rightarrow x = \left(A^T A\right)^{-1} A^T y$$

این فرم Least Square است که براي $\lambda = 0$ به دست مي آید.

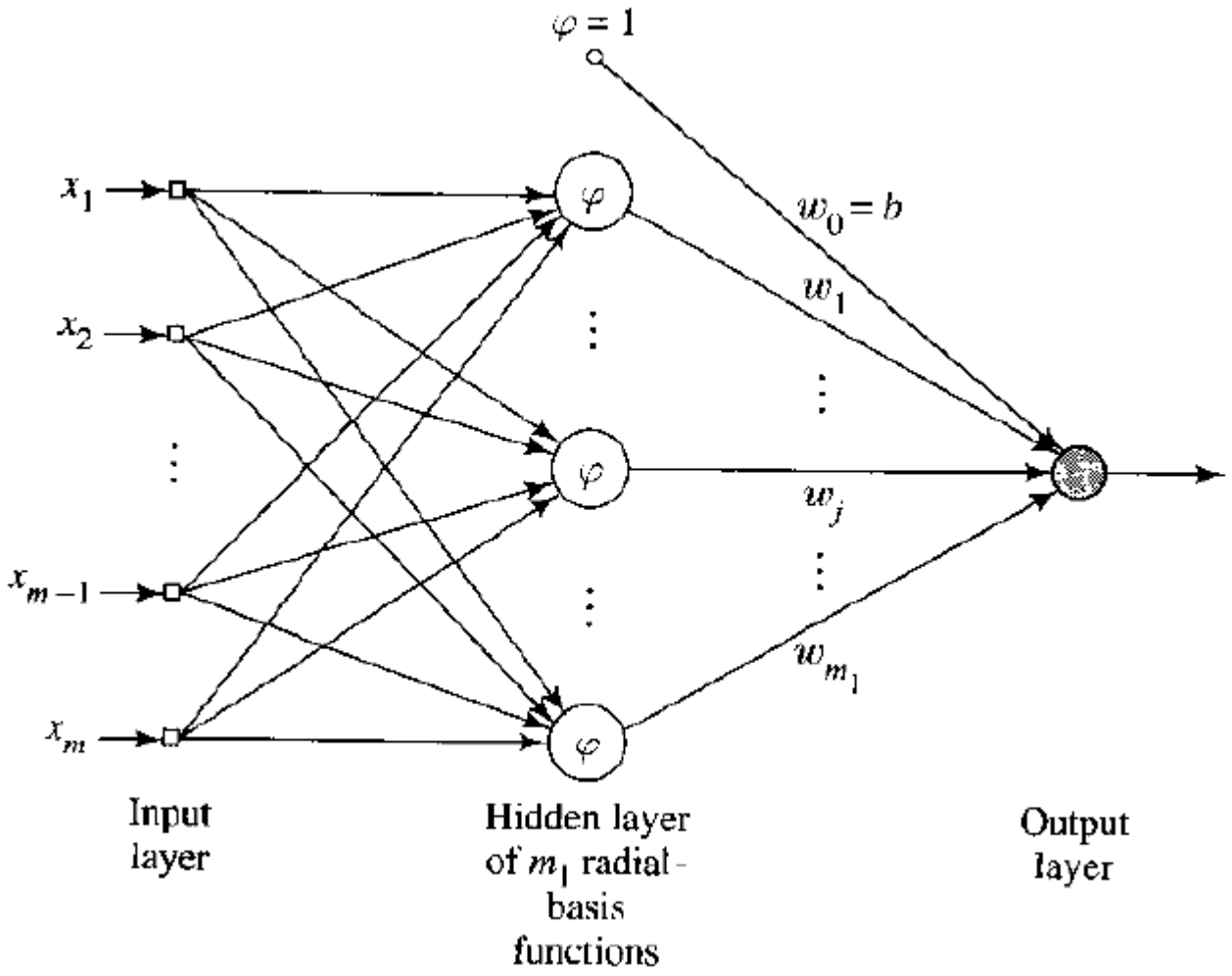
$$w = G^+ d \qquad G^+ = \left(G^T G\right)^{-1} G^T$$

G^+ معکوس جعلي (Pseudo inverse) است.

REGULARIZATION NETWORKS



RBF NETWORK


$\varphi = 1$
x_1
x_2
x_{m-1}
x_m
φ
φ
φ
$w_0 = b$
w_1
w_j
w_{m_1}
Input layer
Hidden layer of m_1 radial-basis functions
Output layer

XOR PROBLEM (REVISITED)

$$G(\|\mathbf{x} - \mathbf{t}_i\|) = \exp(-\|\mathbf{x} - \mathbf{t}_i\|^2), \quad i = 1, 2 \tag{5.85}$$

where the centers $\mathbf{t}_1$ and $\mathbf{t}_2$ are

$$\mathbf{t}_1 = [1, 1]^T$$

$$\mathbf{t}_2 = [0, 0]^T$$

For the characterization of the output unit, we assume the following:

1. The output unit uses *weight-sharing*, which is justified by virtue of the symmetry of the problem; this is a form of prior information being built into the design of the network. With only two hidden units, we therefore only have a single weight w to be determined.
2. The output unit includes a bias b (i.e., data-independent variable). The significance of this bias is that the desired output values of the XOR function have nonzero mean.

Thus the structure of the RBF network proposed for solving the XOR problem is as shown in Fig. 5.6. The input–output relation of the network is defined by

$$y(\mathbf{x}) = \sum_{i=1}^{2} w G(\|\mathbf{x} - \mathbf{t}_i\|) + b \tag{5.86}$$

To fit the training data of Table 5.2, we require that

$$y(\mathbf{x}_j) = d_j, \qquad j = 1, 2, 3, 4 \tag{5.87}$$

where $\mathbf{x}_j$ is an input vector and d_j is the corresponding value of the desired output. Let

$$g_{ji} = G(\|\mathbf{x}_j - \mathbf{t}_i\|), \qquad j = 1, 2, 3, 4; \; i = 1, 2 \tag{5.88}$$

Then, using the values of Table 5.2 in Eq. (5.88), we get the following set of equations written in matrix form:

$$\mathbf{Gw} = \mathbf{d} \tag{5.89}$$

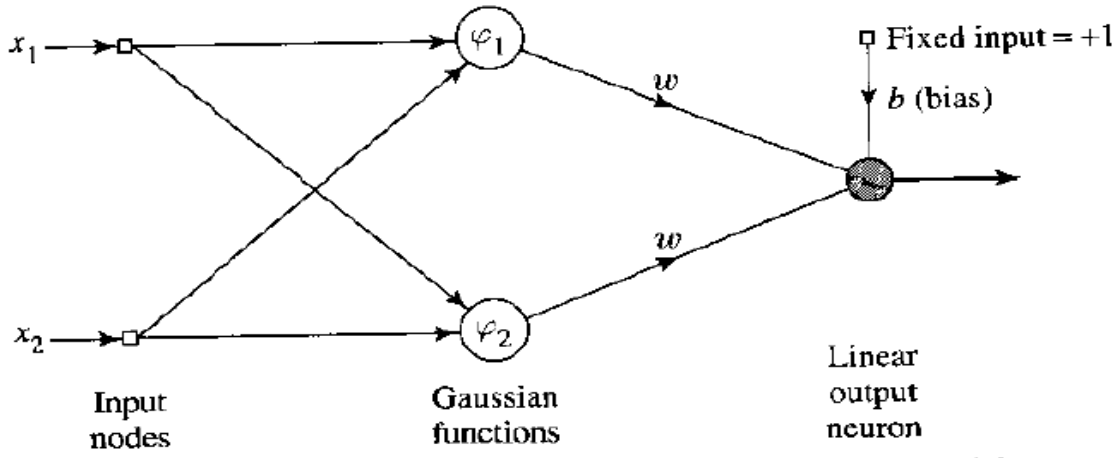


FIGURE 5.6 RBF network for solving the XOR problem.

TABLE 5.2 Input–Output Transformation Computed for XOR Problem

Data Point, j	Input Pattern, $\mathbf{x}_j$	Desired Output, d_j
1	(1, 1)	0
2	(0, 1)	1
3	(0, 0)	0
4	(1, 0)	1

where

$$\mathbf{G} = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1 & 1 \\ 0.3678 & 0.3678 & 1 \end{bmatrix} \tag{5.90}$$

$$\mathbf{d} = [0 \quad 1 \quad 0 \quad 1]^T \tag{5.91}$$

$$\mathbf{w} = [w \quad w \quad b]^T \tag{5.92}$$

The problem described here is *overdetermined in the sense that we have more data points than free parameters*. This explains why the matrix $\mathbf{G}$ is not square. Consequently, no unique inverse exists for the matrix $\mathbf{G}$. To overcome this difficulty, we use the *minimum-norm* solution of Eq. (5.78), and so write

$$\begin{aligned} \mathbf{w} &= \mathbf{G}^{+}\mathbf{d} \\ &= (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{d} \end{aligned} \tag{5.93}$$

Note that $\mathbf{G}^T\mathbf{G}$ is a square matrix with a unique inverse of its own. Substituting Eq. (5.90) in (5.93),we get

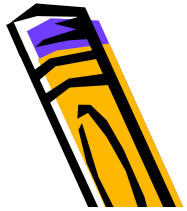
$$\mathbf{G}^{+} = \begin{bmatrix} 1.8292 & -1.2509 & 0.6727 & -1.2509 \\ 0.6727 & -1.2509 & 1.8292 & -1.2509 \\ -0.9202 & 1.4202 & -0.9202 & 1.4202 \end{bmatrix} \tag{5.94}$$

Finally, substituting Eqs. (5.91) and (5.94) in (5.93), we get

$$\mathbf{w} = \begin{bmatrix} -2.5018 \\ -2.5018 \\ +2.8404 \end{bmatrix}$$

which completes the specification of the RBF network.

COMPUTER EXPERIMENT: PATTERN CLASSIFICATION



In this section we use a computer experiment to illustrate the design of a regularized RBF network based on the use of strict interpolation. The computer experiment involves a binary classification problem based on data drawn from two equiprobable overlapping two-dimensional Gaussian distributions corresponding to classes $\mathscr{C}_1$ and $\mathscr{C}_2$. Details of the Gaussian distributions are the same as those described in Section 4.8. Class $\mathscr{C}_1$ is characterized by mean vetor $[0, 0]^T$ and common variance 1, whereas class $\mathscr{C}_2$ is characterized by mean vector $[0, 2]^T$ and common variance 4. The experiment described in this section may thus be viewed as the regularization RBF counterpart to the back-propagation learning experiment of Section 4.8.

With two classes $\mathscr{C}_1$ and $\mathscr{C}_2$, the regularized RBF network is constructed to have two output functions, one for each class. Also, binary-valued class indicator outputs are used as the desired output values, as shown by

$$d_k^{(p)} = \begin{cases} 1 & \text{if pattern } p \text{ belongs to class } \mathscr{C}_k \\ 0 & \text{otherwise} \end{cases}$$

where $k = 1, 2$.

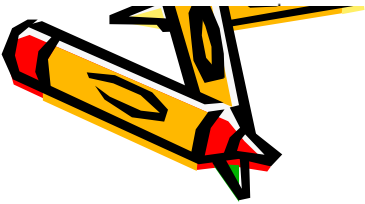


TABLE 5.5 Size of Hidden Layer $m_1 = 20$ Centers: Details of Probability of Correct Classification, $P_c(\%)$ for varying Regularization Parameter

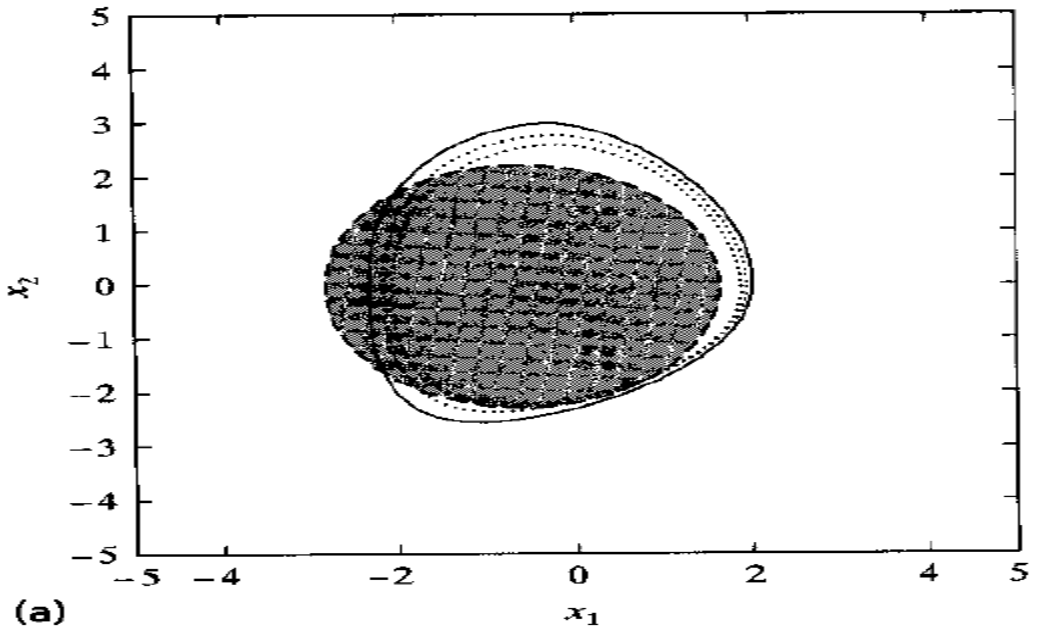
Ensemble Statistic	Regularization Parameter, λ					
	0	0.1	1	10	100	1000
Mean	57.49	72.42	74.42	73.80	72.46	72.14
Std. dev.[a]	7.47	4.11	3.51	4.17	4.98	5.09
Minimum	44.20	61.60	65.80	63.10	60.90	60.50
Maximum	72.70	78.30	78.90	79.20	79.40	79.40

[a]Std. dev.: Standard deviation.

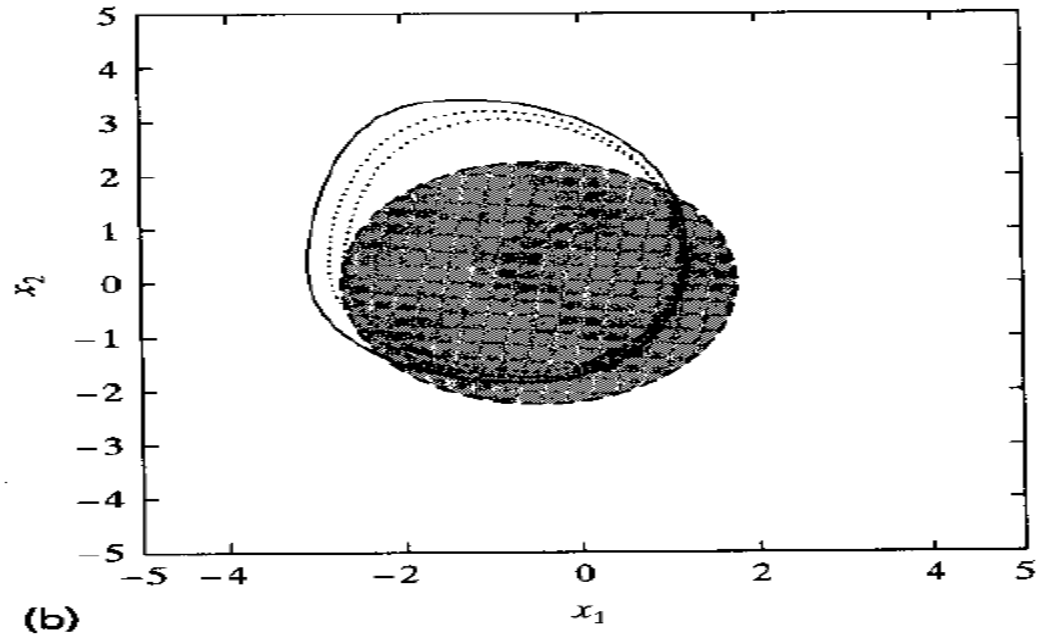
TABLE 5.6 Size of Hidden Layer $m_1 = 100$ Centers: Details of Probability of Correct Classification, $P_c(\%)$ for Varying Regularization Parameter

Ensemble Statistic	Regularization Parameter, λ					
	0	0.1	1	10	100	1000
Mean	50.58	77.03	77.72	77.87	76.47	75.33
Std. dev.[a]	4.70	1.45	0.94	0.91	1.62	2.25
Minimum	41.00	70.60	75.10	75.10	72.10	70.10
Maximum	61.30	79.20	79.80	79.40	78.70	78.20

[a]Std. dev.: Standard deviation.

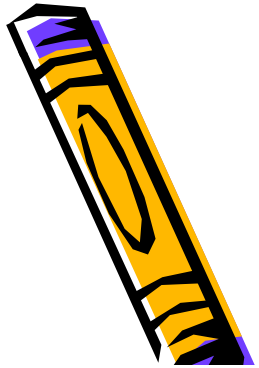


(a)



(b)

FIGURE 5.7 Results of experiment on pattern classification using strict interpolation with regularized RBF networks. (a) Best solution. (b) Worst solution. Dashed (shaded) circle represents the optimum Bayes solution.



Comparing Tables 5.5 and 5.6, we make the following observations:

1. For both $m_1 = 20$ and $m_1 = 100$ centers, the classification performance of the network for $\lambda = 0$ is relatively poor.
2. The use of regularization has a dramatic influence on the classification performance of the RBF network.
3. For $\lambda \geq 0.1$, the classification performance of the network is somewhat insensitive to an increase in the regularization parameter λ. For $m_1 = 20$ centers, best performance is obtained at $\lambda = 1$, and for $m_1 = 100$ centers, it is obtained at $\lambda = 10$.
4. Increasing the number of centers from $m_1 = 20$ to $m_1 = 100$ improves the classification performance by about 4.5 percent.

