

یادگیری عمیق بخش دوم

MLP
Error Back
propagation



دانشگاه شهید بهشتی
زمستان ۱۴۰۰
احمد محمودی ازناوه

فهرست مطالب

- شبکه‌ی عصبی چندلایه

- آموزش

- الگوریتم پس‌انتشار خطا

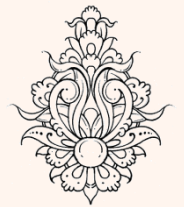
- Momentum

- انواع آموزش

- توقف آموزش

- نکاتی برای تسریع آموزش

- بردارهای ورودی



فهرست مطالب (ادامه...)

– تابع انگیزش

– نرخ یادگیری

– تابع هزینه

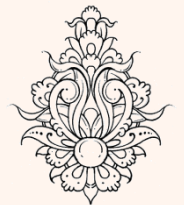
• الگوریتم‌های بهینه‌سازی

– Gradient descent

• روش نیوتن

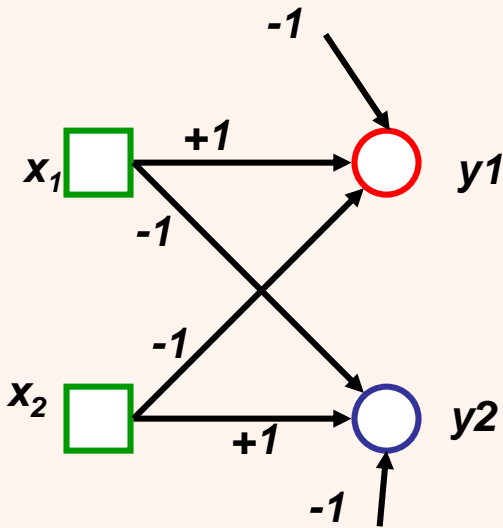
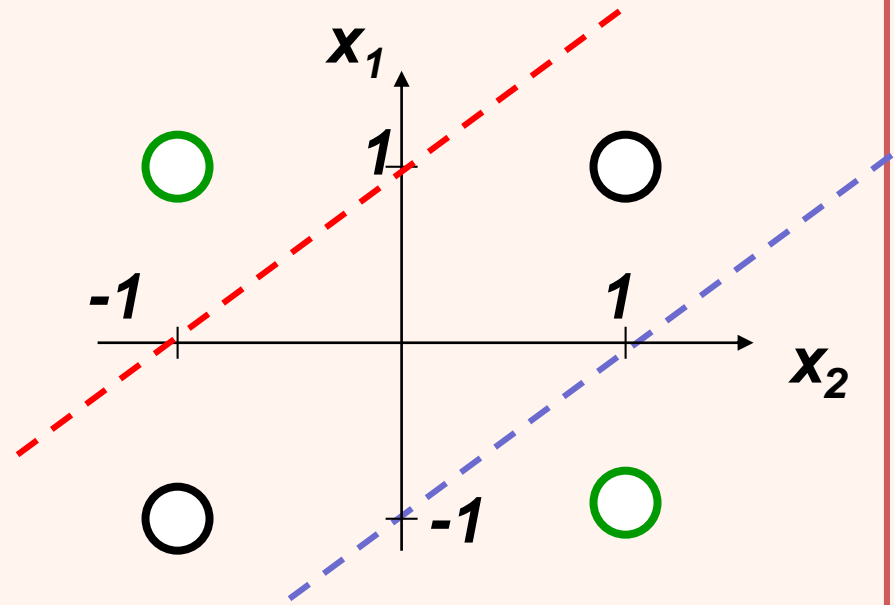
• Levenberg-Marqualt

• Conjuagte gradient



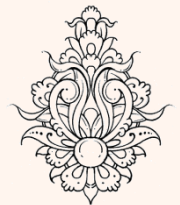
مثال

x_1	x_2	$x_1 \text{ XOR } x_2$
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1



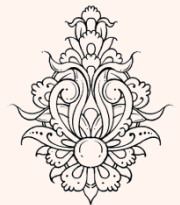
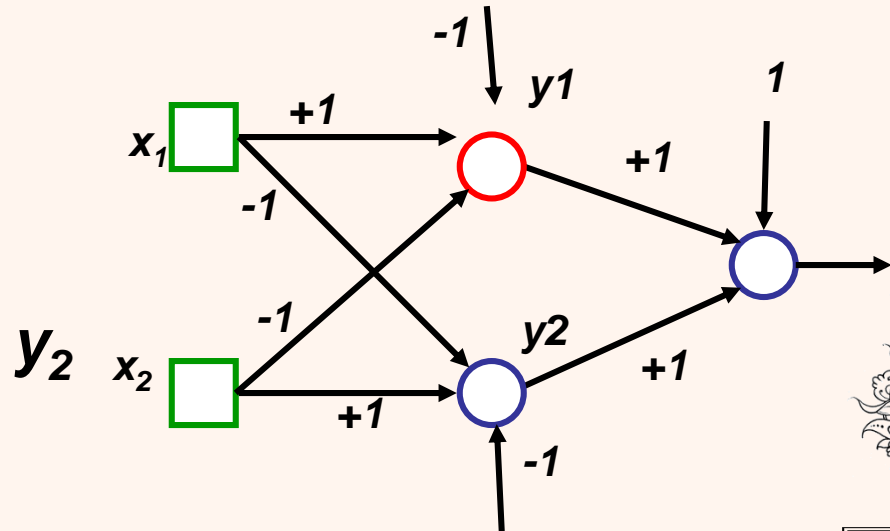
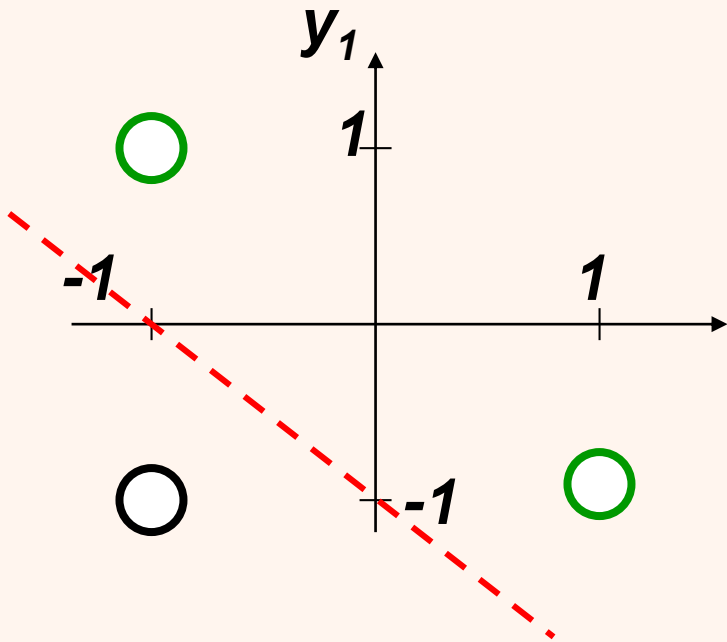
$$\varphi(v) = \begin{cases} 1 & \text{if } v > 0 \\ -1 & \text{if } v \leq 0 \end{cases}$$

φ is the sign function.



مثال (ادامه...)

x_1	x_2	y_1	y_2	$x_1 \text{ XOR } x_2$
-1	-1	-1	-1	-1
-1	1	-1	1	1
1	-1	1	-1	1
1	1	-1	-1	-1

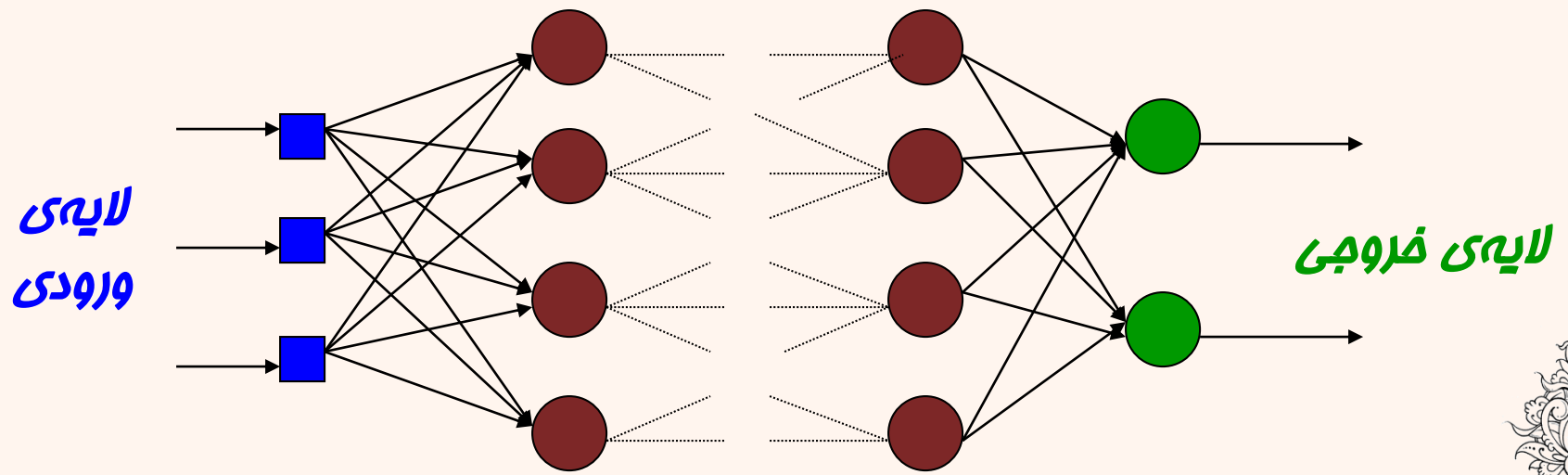


دگیری عمیق Minsky & Papert (1969) offered solution to XOR problem by combining perceptron unit responses using a second layer of units

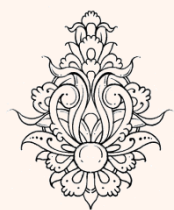
Multilayer Neural Network

شبکه عصبی چند لایه

Multilayer Perceptron (MLP)



Lایه های پنهان

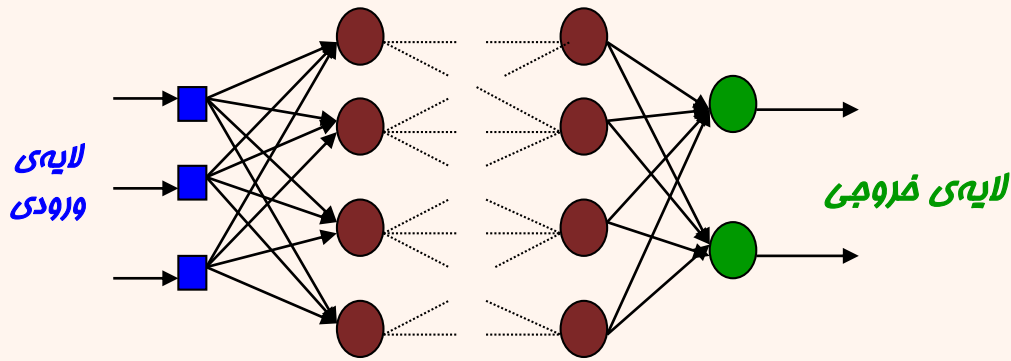


شبکه‌ی عصبی چند لایه (ادامه...)

- ورودی‌ها به صورت مستقیم به خروجی متصل نیستند.
- هر واحد از لایه‌ی قبلی به تمامی واحدهای لایه‌ی بعدی متصل است. (وزن صفر مجاز است)
- تعداد لایه‌ها و همچنین تعداد واحدهای مخفی در هر لایه باید از پیش مشخص باشد.
- تمام اتصالات **رو به جلو** است.
- تابع فعال‌سازی نورون‌ها (جز لایه خروجی) باید تابعی **غیرخطی** باشد.

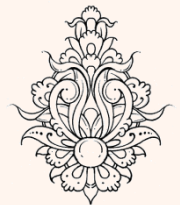
Fully connected

Feed Forward



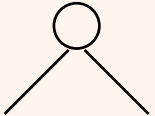
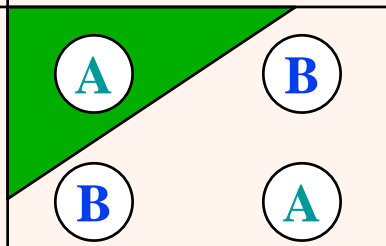
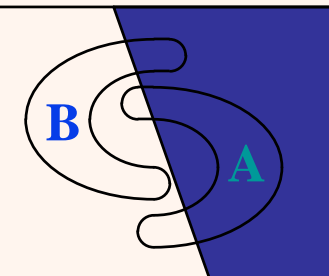
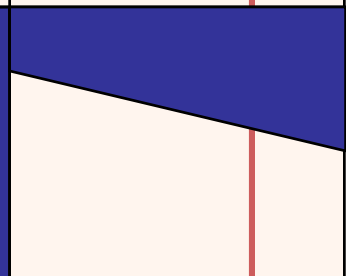
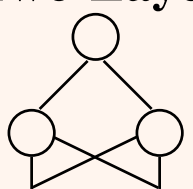
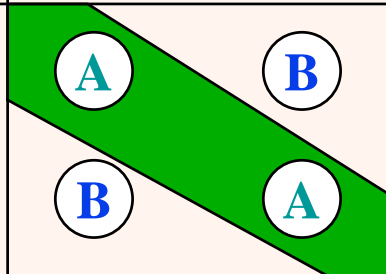
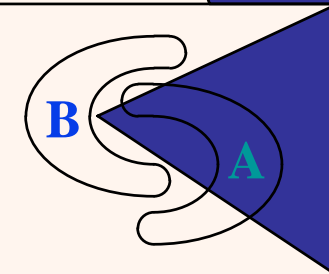
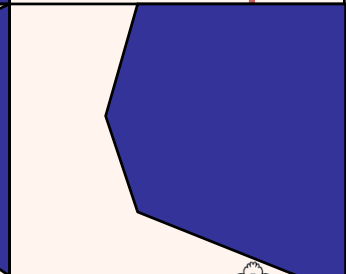
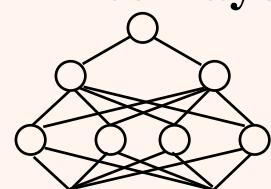
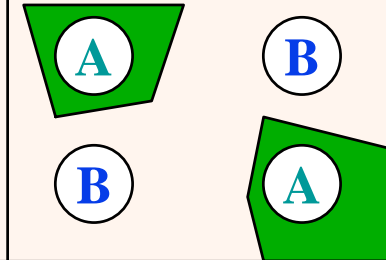
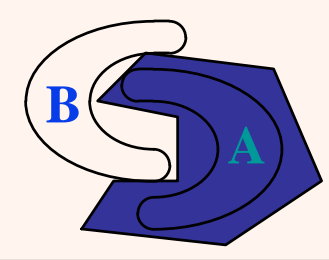
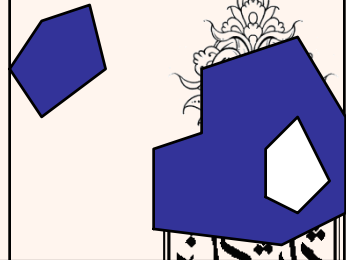
یادگیری عمیق

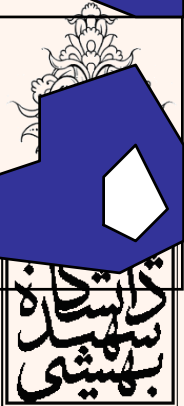
لایه‌های مخفی



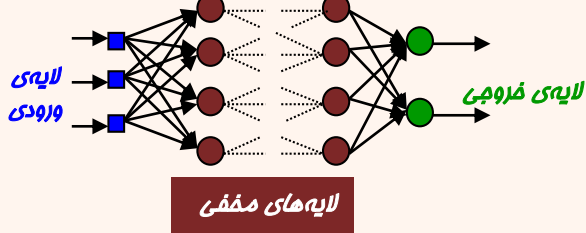
شبکہ عصبی چند لایہ

An introduction to computing with neural nets (Lippmann, R. P. 1987)

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			



یادگیری



- برای آموزش این شبکه‌ها از الگوریتم «پس‌انتشار خطا» استفاده می‌شود.

Back Propagation

- برای آموزش این دست شبکه‌ها به طریقه‌ی زیر عمل می‌شود:

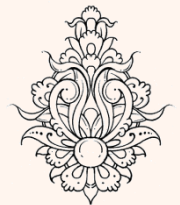
– رو به جلو (Forward)

- بردار ورودی به شبکه اعمال شده و خروجی واقعی محاسبه می‌شود.

– رو به عقب (Backward)

- خطا (خروجی واقعی - خروجی مطلوب) محاسبه شده و بر حسب تابع معیار، سیگنالی متناسب با خطا تولید می‌شود. این سیگنال لایه لایه حرکت کرده و وزن‌ها را تا لایه‌ی ورودی

sequential mode

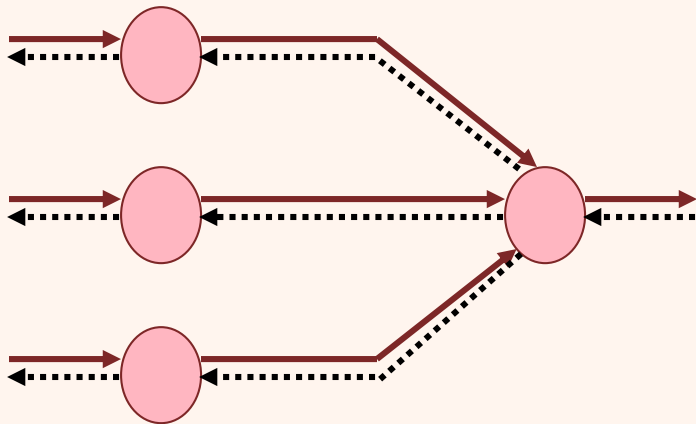


پس از اصلاح وزن‌ها می‌نویسیم یک iteration صورت گرفته است.

عملکرد شبکه

• فرض شبکه

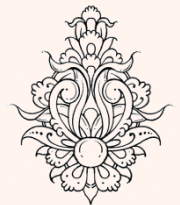
– m لایه بدون در نظر گرفتن لایه ورودی



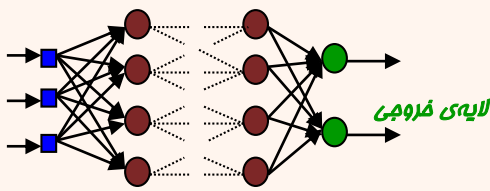
→ **Function signals**
Forward Step

← **Error signals**
Backward Step

وزن‌ها به گونهای اصلاح می‌شوند که میانجین
مجموع مربعات خطا کمینه گردد.



لایه‌ی ورودی



لایه‌ی خروجی

میزان خطا

sequential mode

- میزان خطا و میانگین مربعات خطا برای نورون زام خروجی در تکرار n ام (به ازای ورودی n -ام) به شیوه‌ی زیر محاسبه می‌شود:

$$e_j(n) = d_j(n) - y_j(n)$$

خطای خروجی نورون زام
(گره‌ی خروجی)

sequential mode

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

instantaneous error energy

$$E_{AV} = \frac{1}{N} \sum_{n=1}^N E(n)$$

averaged squared error energy

Batch mode

دانشگاه شهید بهشتی

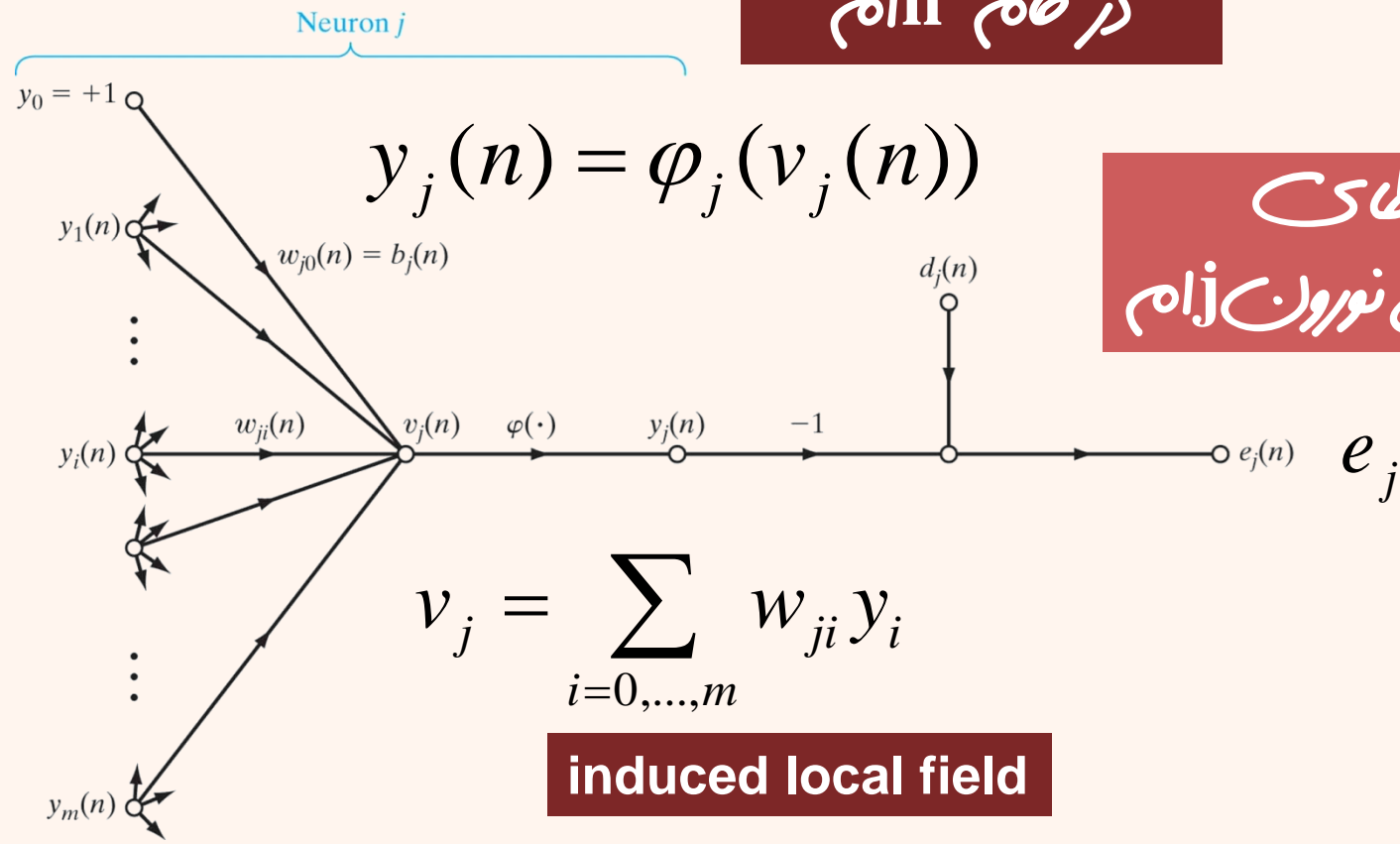
هدف آموزش
کمینه نمودن
 E_{AV}

آموزش شبکه‌های چندلایه

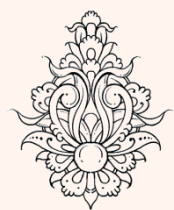
- به روز نمودن وزن‌ها همانند LMS است:

خروجی نورون زام
در گام nام

خطای
خروجی نورون زام

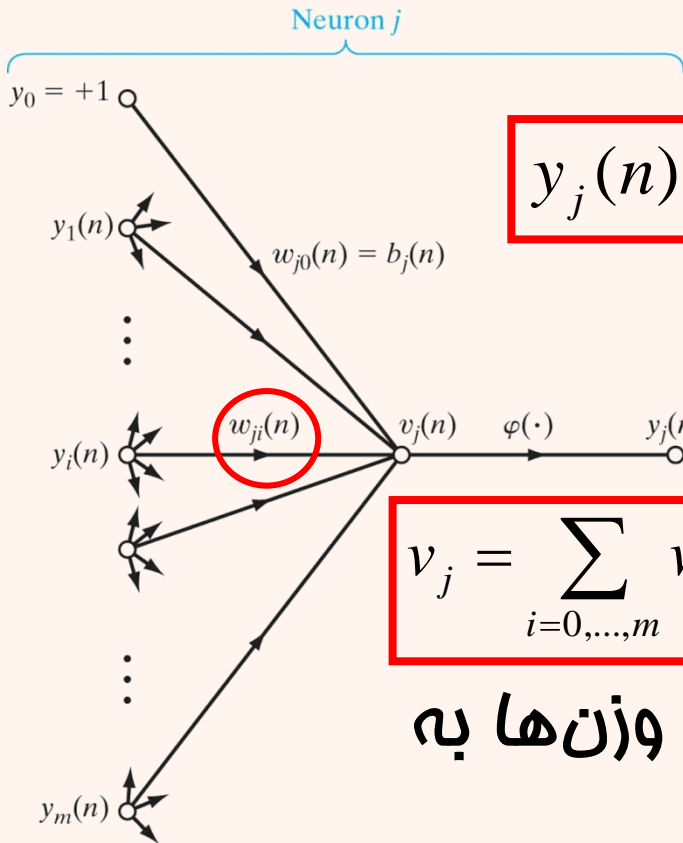


induced local field



آموزش لایه‌ی خروجی

sequential mode



$$y_j(n) = \varphi_j(v_j(n))$$

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

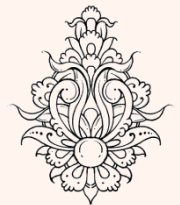
$$v_j = \sum_{i=0, \dots, m} w_{ji} y_i$$

$$e_j(n) = d_j - y_j(n)$$

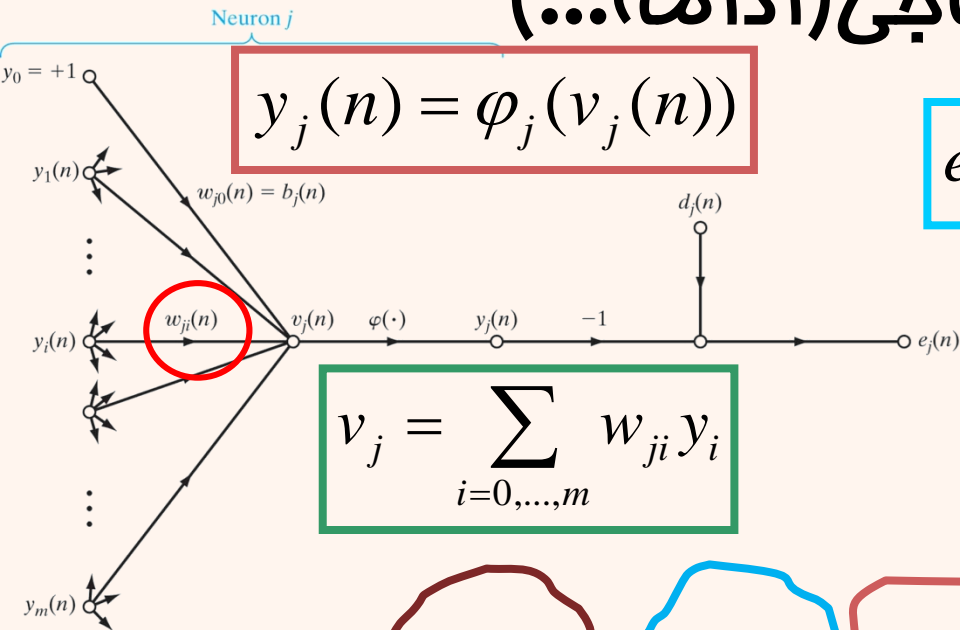
- همانند LMS برای به روز کردن وزن‌ها به شیوه‌ی زیر عمل می‌کنیم:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

sensitivity factor



آموزش لایه خروجی (ادامه...)



$$y_j(n) = \varphi_j(v_j(n))$$

$$e_j(n) = d_j - y_j(n)$$

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

$$v_j = \sum_{i=0, \dots, m} w_{ji} y_i$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$e_j(n)$ -1 $\phi'_j(v_j(n))$ $y_i(n)$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) \phi'_j(v_j(n)) y_i(n)$$



آموزش لایه خروجی (ادامه...)

• از قانون **دلتا** داشتیم:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}$$

Step in direction opposite to the gradient

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) \phi_j'(v_j(n)) y_i(n)$$

$$\Delta w_{ji}(n) = \eta e_j(n) \phi_j'(v_j(n)) y_i(n)$$

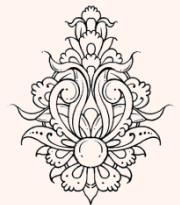
$$\delta_j(n)$$

گرادین مملی

گرادین مملی برای یک نورون به فضای ایجاد شده برای آن نورون مشتق تابع مرتباً آن بستگی دارد

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)}$$

$$= -\frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n) \phi_j'(v_j(n))$$



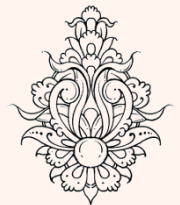
آموزش لایه خروجی (ادامه...) $e_j = d_j - y_j$

$$\Delta w_{ji}(n) = \eta e_j(n) \phi'_j(v_j(n)) y_i(n)$$

$$\delta_j(n)$$

$$\delta_j = (d_j - y_j) \phi'(v_j)$$

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



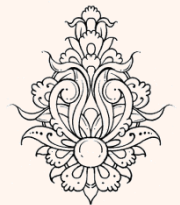
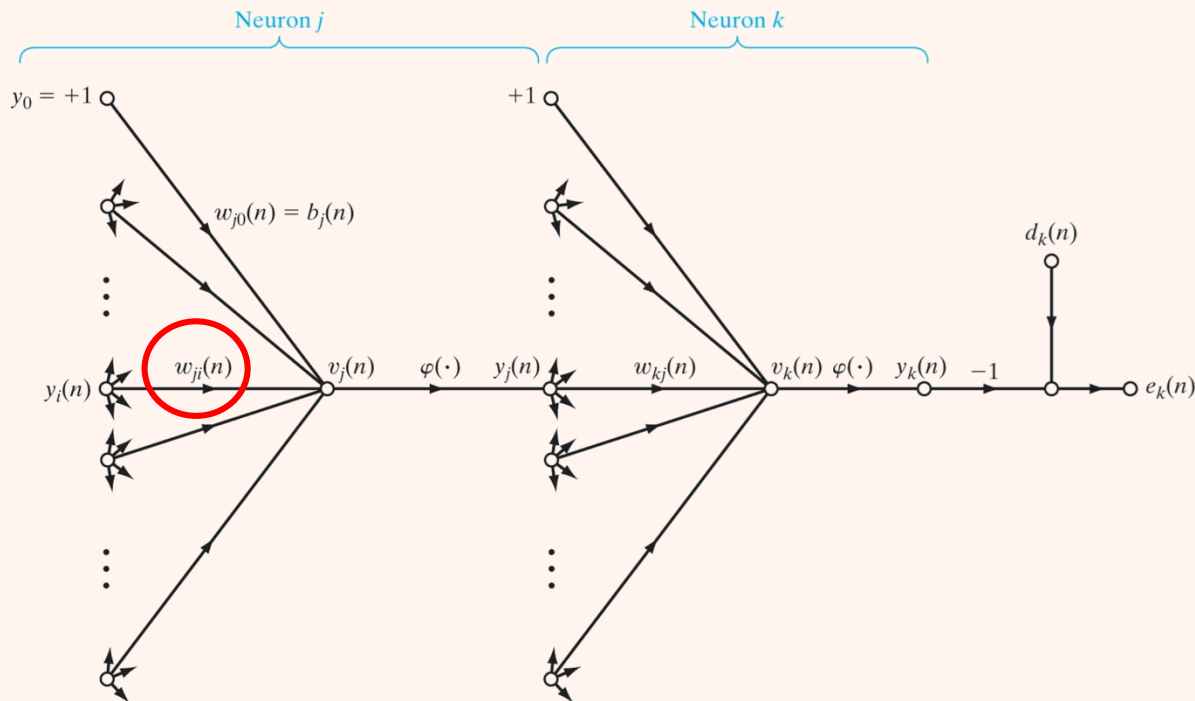
دانشگاه

در ادامه به شیوه آموزش لایه مخفی خواهیم پرداخت

آموزش لایه مخفی

- اگر نرون‌ها را از لایه مخفی در نظر بگیریم برای محاسبه‌ی خطا:

– برای محاسبه‌ی گرادیان محلی نرون‌ها در نظر می‌باید تمامی گرادیان‌های محلی نرون‌هایی که با نرون‌ها در ارتباط هستند را لحاظ نماییم.

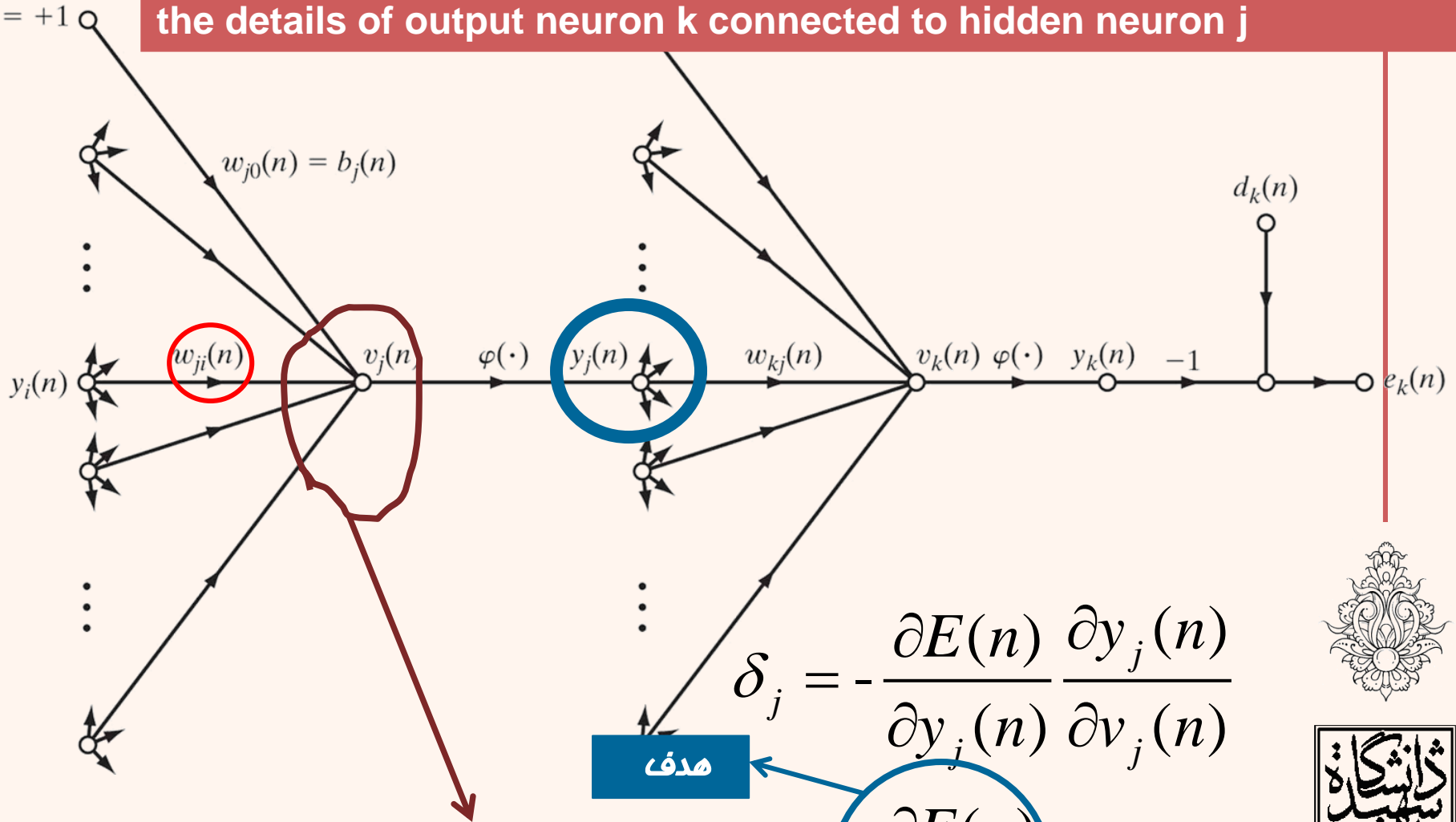


آموزش لایه مخفی (ادامه...)

Neuron j

Neuron k

the details of output neuron k connected to hidden neuron j



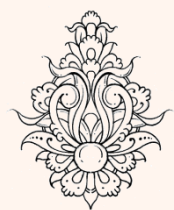
$$\delta_j = - \frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)}$$

$$= - \frac{\partial E(n)}{\partial y_j(n)} \varphi'_j(v_j(n))$$

هدف

نورونز متعلق به لایه مخفی است

یادگیری عمیق



آموزش لایه مخفی (ادامه...)

$$\delta_j = - \frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)}$$

$$= - \frac{\partial E(n)}{\partial y_j(n)} \varphi_j'(v_j(n))$$

$$E(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n)$$

نورون کایک
node خروجی
است

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)}$$

$$v_k(n) = \sum_{j=0}^m w_{kj}(n) y_j(n)$$

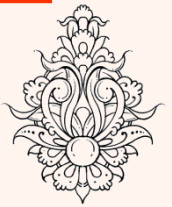
$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \left(\frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \right)$$

$$w_{kj}(n)$$

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

$$-\varphi_k'(v_k(n))$$

$$\text{پادگیری عمیق} = d_k(n) - \varphi_k(v_k(n))$$



آموزش لایه مخفی (ادامه...)

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \rightarrow w_{kj}(n)$$

$$-\phi'_k(v_k(n))$$

$$\frac{\partial E(n)}{\partial y_j(n)} = - \sum_k e_k(n) \phi'_k(v_k(n)) w_{kj}(n)$$

$$= - \sum_k \delta_k(n) w_{kj}(n)$$



$$\delta_j = \phi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

$$\delta_j = - \frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)}$$

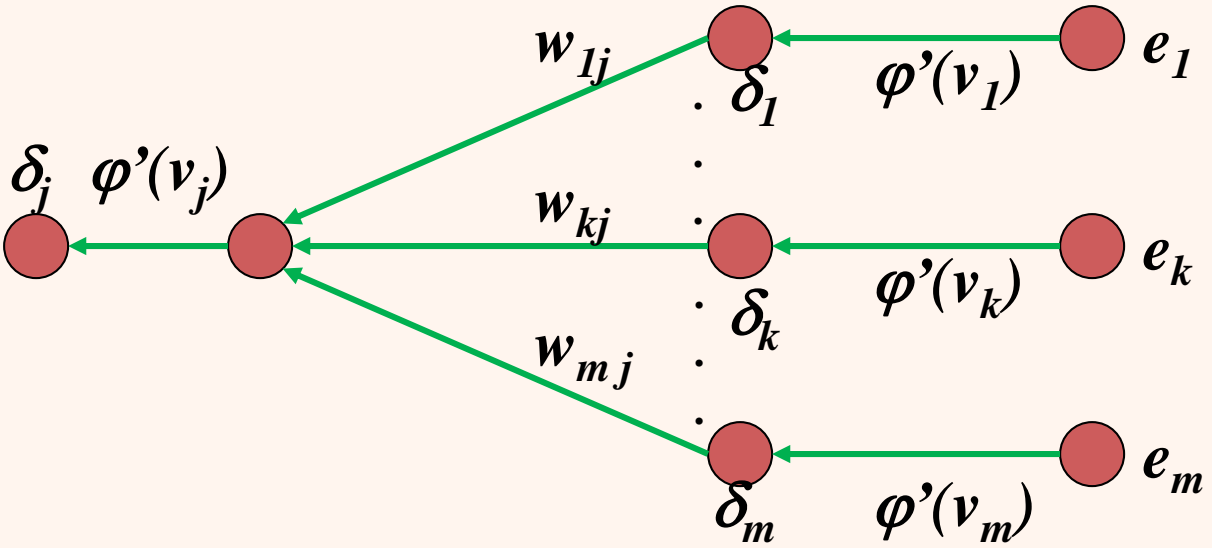
$$= - \frac{\partial E(n)}{\partial y_j(n)} \phi'_j(v_j(n))$$

آموزش لایه مخفی (Local Gradient of hidden neuron j)

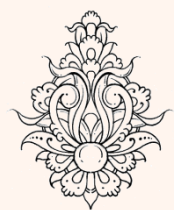
$$\delta_j = \varphi'_j(v_j) \sum_{k \in C} \delta_k w_{kj}$$

Iteration شماره n

Signal-flow graph of back-propagation error signals to neuron j



$$\begin{pmatrix} \text{Weight} \\ \text{Correction} \\ \Delta w_{ij}(n) \end{pmatrix} = \begin{pmatrix} \text{learning} \\ \text{rate} \\ \eta \end{pmatrix} \cdot \begin{pmatrix} \text{local} \\ \text{gradient} \\ \delta_j(n) \end{pmatrix} \cdot \begin{pmatrix} \text{input} \\ \text{of neuron } j \\ y_i(n) \end{pmatrix}$$

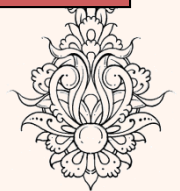


آموزش لایه‌ی مخفی (ادامه...)

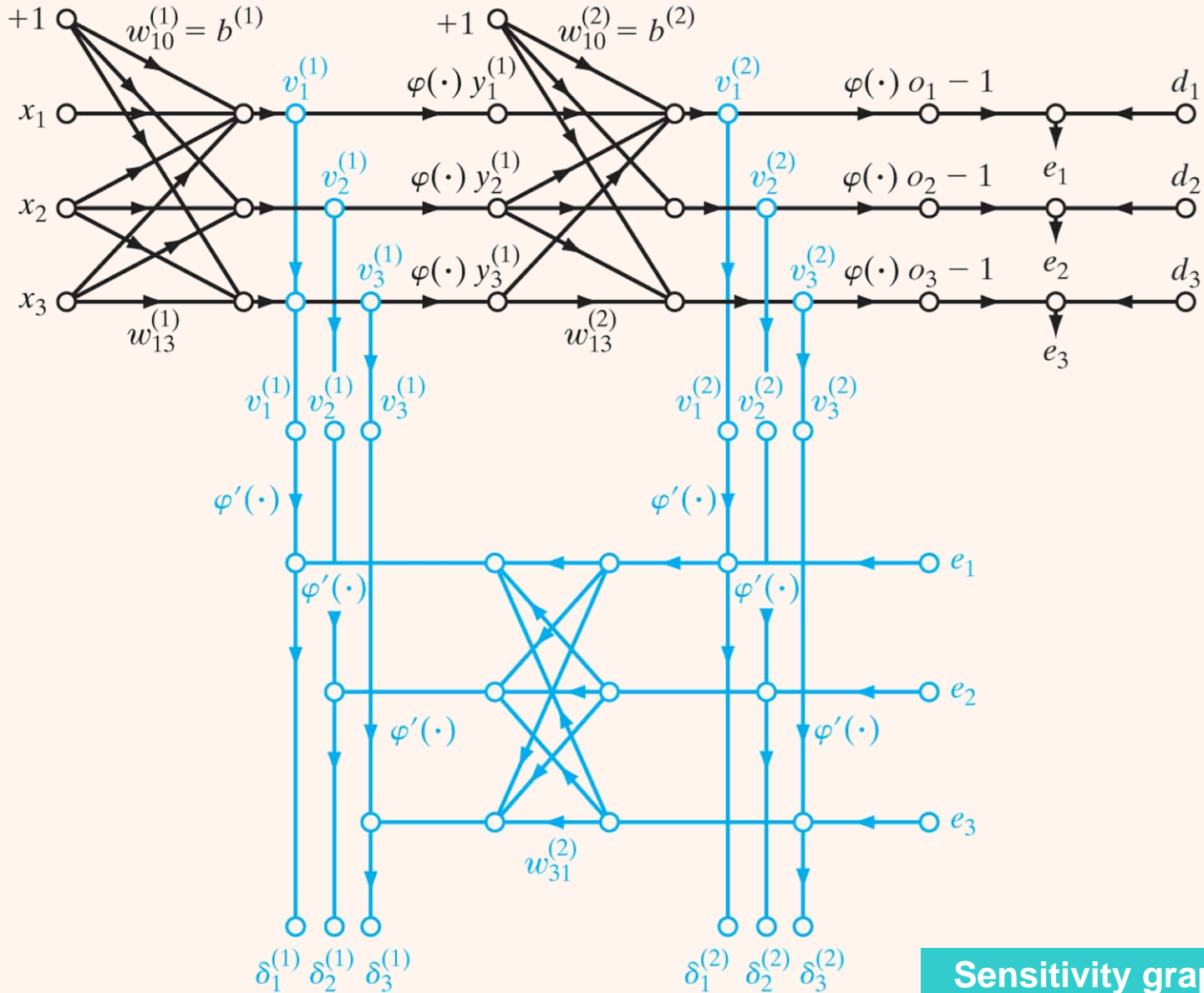
- Delta rule $\Delta w_{ji} = \eta \delta_j y_i$

$$\delta_j = \begin{cases} \varphi'(v_j)(d_j - y_j) & \text{IF } j \text{ output node} \\ \varphi'(v_j) \sum_{k \in C} \delta_k w_{kj} & \text{IF } j \text{ hidden node} \end{cases}$$

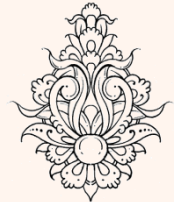
C: Set of neurons in the layer following the one containing j



آموزش لایه‌ی مخفی (ادامه...)



Sensitivity graph



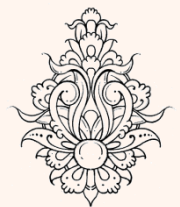
در این شیوه تمام نمونه‌های آموزشی اعمال شده، سپس اصلاح وزن‌ها صورت می‌پذیرد.

$$E_{AV} = \frac{1}{N} \sum_{n=1}^N E(n)$$

$$\Delta w_{ji} = -\eta \frac{\partial E_{AV}}{\partial w_{ji}}$$

$$\frac{\partial E_{AV}}{\partial w_{ji}} = \frac{\partial E_{AV}}{\partial E} \cdot \frac{\partial E}{\partial w_{ji}}$$

$$= \frac{1}{N} \sum_{n=1}^N \frac{\partial E(n)}{\partial w_{ji}}$$



انواع شیوه‌های آموزش

Sequential Mode

online

pattern or stochastic mode

• شیوهی ترتیبی:

– در این شیوه نمونه‌ها تک‌تک برای اصلاح وزن‌ها به کار می‌روند.

هر بار که یک ورودی به شبکه اعمال شده و اصلاح وزن‌ها را یک **iteration** می‌نامند.

Batch Mode

• شیوهی دسته‌ای:

یک دوره کامل ارائه‌ی نمونه‌های آموزشی در فرآیند آموزش را **epoch** می‌نامند.

$$E_{AV} = \frac{1}{2N} \sum_{n=1}^N \sum_{j \in C} e_j^2(n)$$

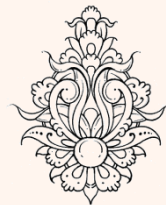
$$\Delta w_{ji} = -\frac{\eta}{N} \sum_{n=1}^N \sum_{j \in C} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}$$

مماسی به این بخش به همان شیوه‌ای که پیش از این گفته شد، انجام می‌شود



انواع شیوه‌های آموزش (ادامه...)

- شیوهی ترتیبی به مافضهی کمتری احتیاج دارد.
- در صورتی که نمونه‌ها به صورت ترتیبی و تصادفی اعمال شود، احتمال این که الگوریتم در دام مینیمم محلی بیفتند، کمتر خواهد بود.
- هر چند این تصادفی بودن، تحلیل نظری شرایط همگرایی را دشوارتر می‌کند، در حالی که با استفاده از شیوهی دسته‌ای تقریب بهتری از بردار گرادیان به دست می‌آید و همگرایی به سوی مینیمم محلی تضمین شده است.
- وقتی که داده‌های تکراری داشته باشیم، روش ترتیبی به صورت مؤثرتری از داده‌های تکراری استفاده می‌کند.
- استفاده از پردازش موازی در شیوهی دسته‌ای به مراتب ساده‌است.



انواع شیوه‌های آموزش (ادامه...)

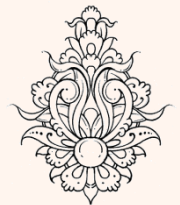
علیرغم، معایب روش تدریجی، این روش در عمل ترجیح داده می‌شود، از این جهت که پیاده‌سازی آن ساده‌تر است. برای مسائل دشوار و بزرگ راه حل موثری است.

- برای بهره‌برداری از مزایای هر دو شیوه، آموزش به صورت «minibatch» نیز معمول است که به‌روزرسانی وزن‌ها، بعد از هر n ($n > 1$) گام صورت می‌گیرد.

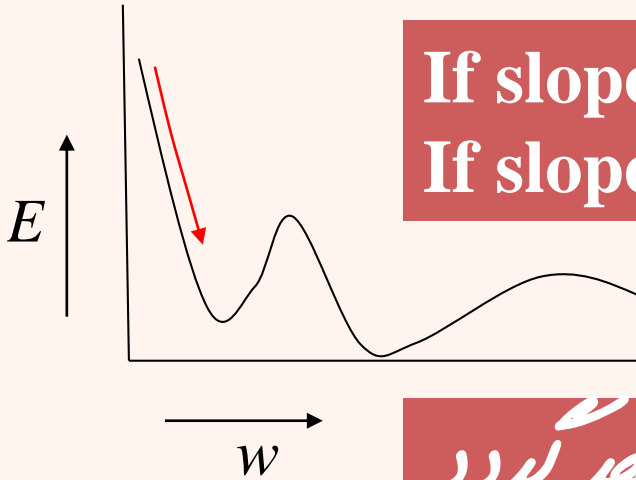
Stochastic Gradient Descent(SGD)

stochastic approximation of gradient descent

هنگامی برای بهینه‌سازی به جای بردار گرادیان تابع از تقریب آن استفاده شود (به جای استفاده از همه داده از یک زیرمجموعه که به صورت تصادفی انتخاب شده است استفاده شود)

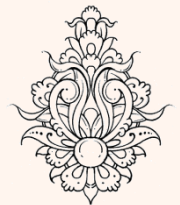
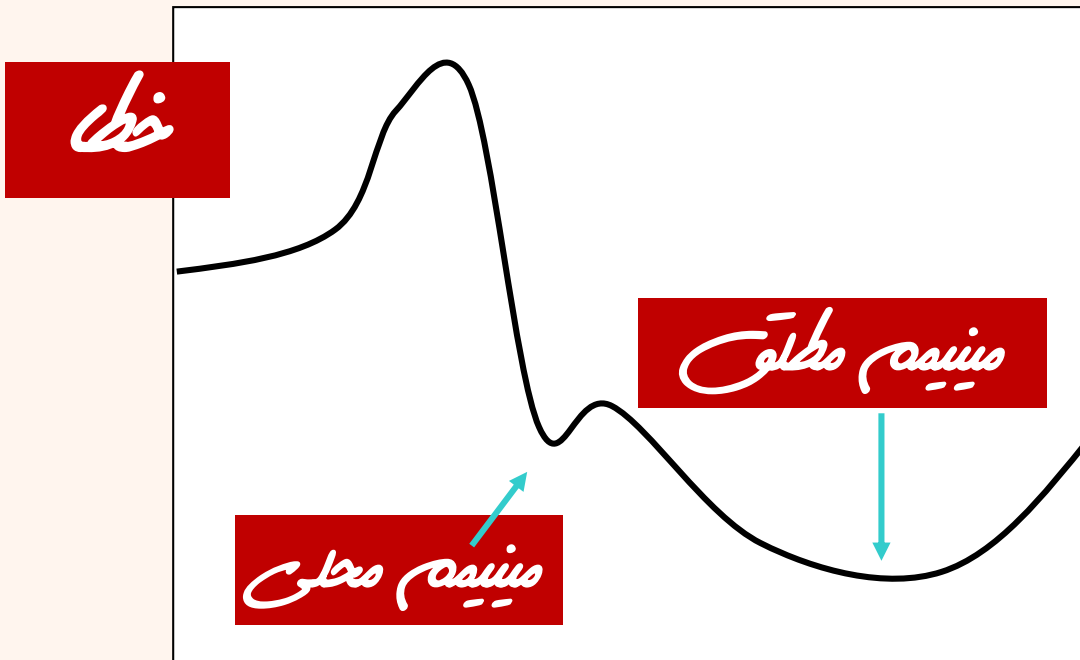


Gradient Descent



If slope is negative \rightarrow increase w
If slope is positive \rightarrow decrease w

مینیمم محلی جایی است که مشتق صفر گردد



معیارهای توقف آموزش

• در عمل، به راحتی نمی‌توان مشخص نمود که آیا فرآیند آموزش به نقطه‌ی مورد نظر رسیده است. اما ضوابطی برای پایان دادن به آموزش مطرح شده است.

– آموزش تا جایی پیش رود که اندازه‌ی بردار گرادیان از یک حد آستانه کمتر شود.

• عیب این روش این است که فرآیند آموزش ممکن است طولانی شود.

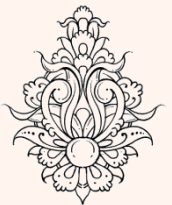
• همچنین لازم است اندازه بردار گرادیان مناسبه شود.

– آموزش زمانی متوقف می‌شود که اختلاف خطای میانگین به دست آمده در دو دوره (epoch) متوالی به اندازه‌ی کافی کوچک باشد.

• این شیوه ممکن است به توقف زودهنگام فرآیند آموزش منجر شود.

– شیوه‌ی مناسب‌تر این است که پس از هر فاز آموزش،

شبکه با داده‌هایی غیر از داده‌های آموزشی بررسی شود و قدرت «تعمیم‌پذیری» آن ملاکی برای توقف آموزش



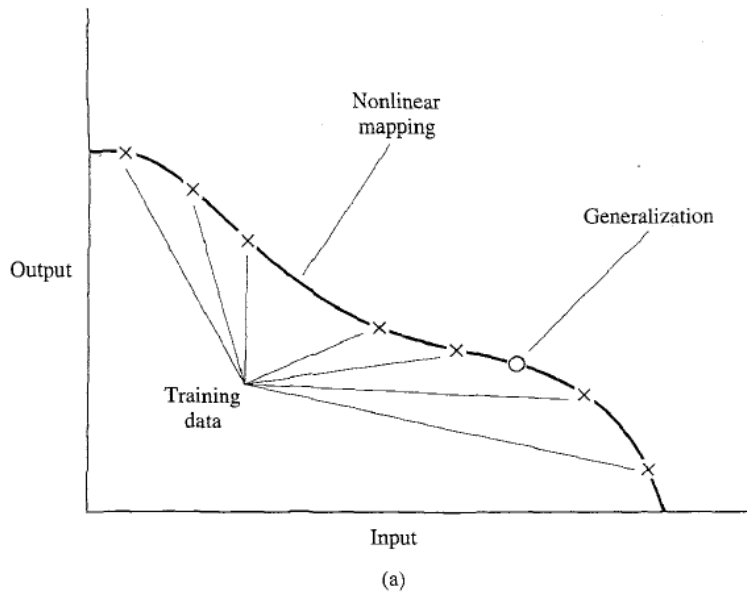
Early stopping

Validation set

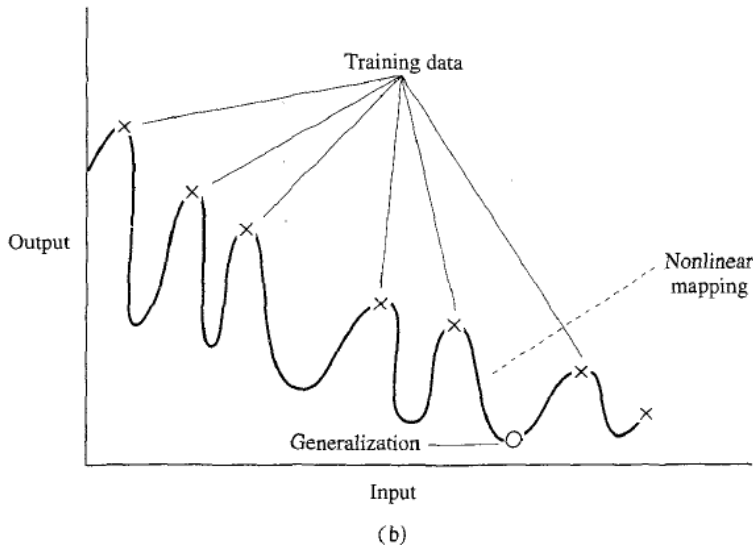
سپهر
بهشتی

یکی از شیوه‌های کم‌هزینه‌ی پیش‌گیری از بیش‌براش توقف زودهنگام فرآیند آموزش است.

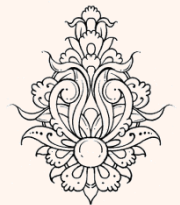
- آموزش بیش از حد
- استفاده از لایه‌های مخفی
- بیش از حد نیاز



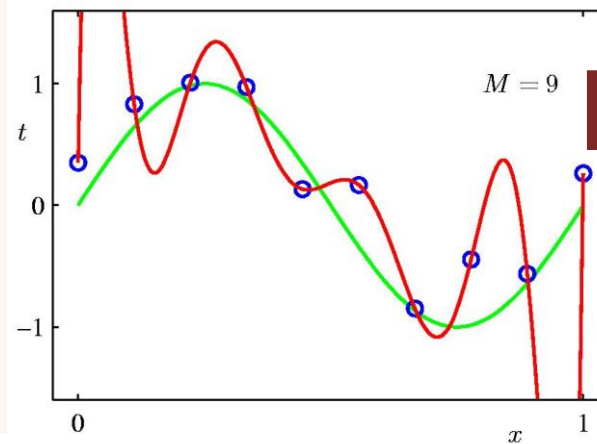
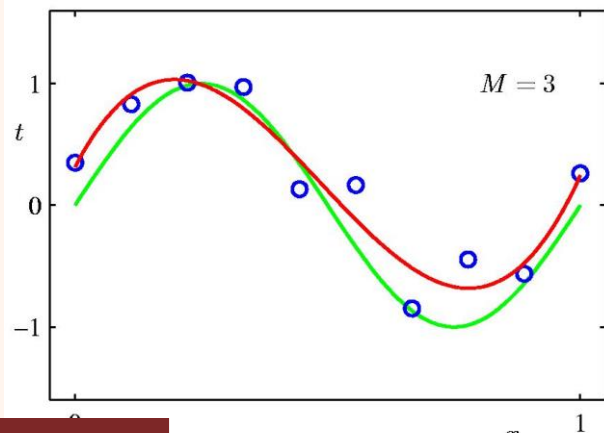
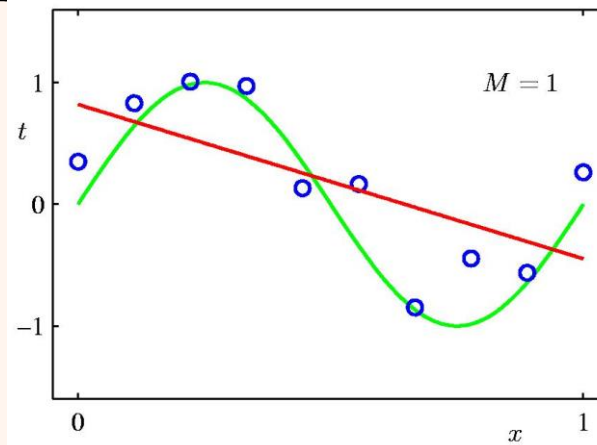
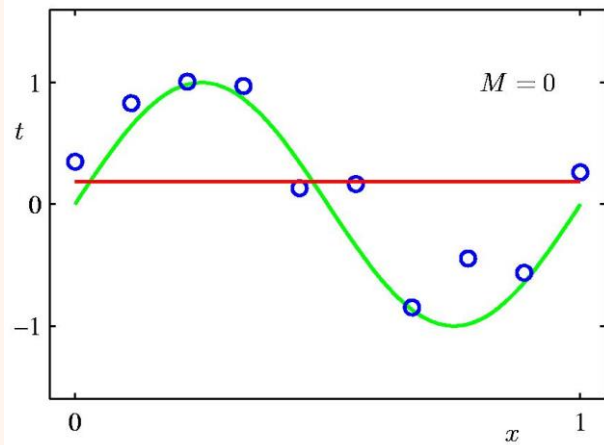
properly fitted data
(good generalization)



Overfitted data
(poor generalization)

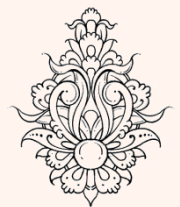
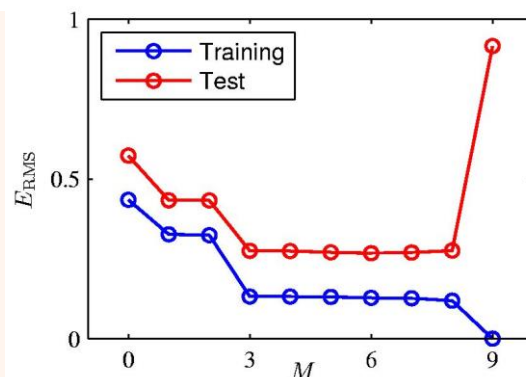


چند عملیاتی با درجات بالاتر



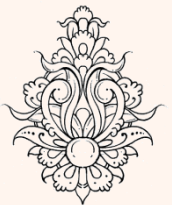
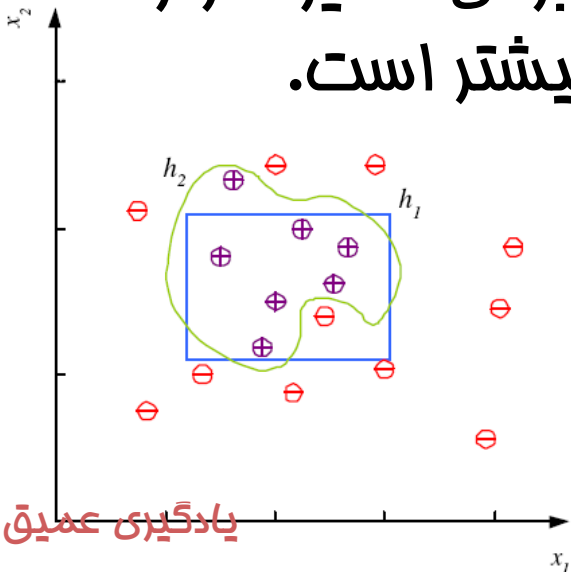
Over fitting

Best fit





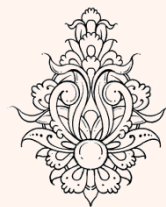
تیغ Occam اصلی منسوب به William of Ockham منطق‌دان و فیلسوف انگلیسی است. در قرن ۱۴ میلادی ویلیام اوکام اصلی را مطرح کرد که به نام اصل «تیغ Occam» شناخته شد. طبق این اصل، هر گاه دربارهٔ علت بروز پدیده‌ای دو توضیح مختلف ارائه شود، در آن توضیحی که **پیچیده‌تر** باشد احتمال بروز اشتباه بیشتر است و بنابراین، در شرایط مساوی بودن سایر موارد، توضیح **ساده‌تر**، احتمال صحتش بیشتر است.



تعمیم در شبکه‌های عصبی (ادامه...)

- سه فاکتور در تعمیم‌پذیری مؤثر هستند:
 - حجم مجموعه‌ی آموزشی و توزیع آن (تا چه حد گویاست)
 - ساختار شبکه‌ی عصبی (پیچیدگی مدل)
 - پیچیدگی مسئله
- تعداد نمونه‌های آموزشی برای تعمیم‌پذیری مناسب

$$N = O\left(\frac{W}{\varepsilon}\right)$$

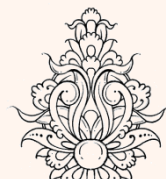


- برای بررسی تصمیم‌پذیری، بخشی از داده‌ها را در آموزش مورد استفاده قرار نمی‌دهیم (validation set)، و تنها برای بررسی تصمیم‌پذیری از آن‌ها استفاده می‌شود.
- در نتیجه، فرضیه‌ای که با داده‌های validation بهترین پاسخ را دارند، به عنوان فرضیه‌ی مناسب انتخاب می‌شود.
- بعد از آموزش، برای مقایسه روش مورد استفاده، داده‌های آزمایش که باید متفاوت از داده‌های آموزشی و داده‌های validation هستند، مورد استفاده قرار گیرد.

Test set (publication set)

stratification

در تقسیم داده‌ها به دو قسمت باید توجه داشت که توزیع داده‌ها حفظ شود. بدین ترتیب دانش پیشین در مورد یک کلاس دستخوش تغییر نخواهد شد.



K-fold cross validation

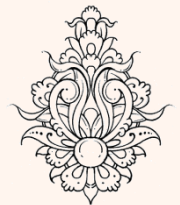
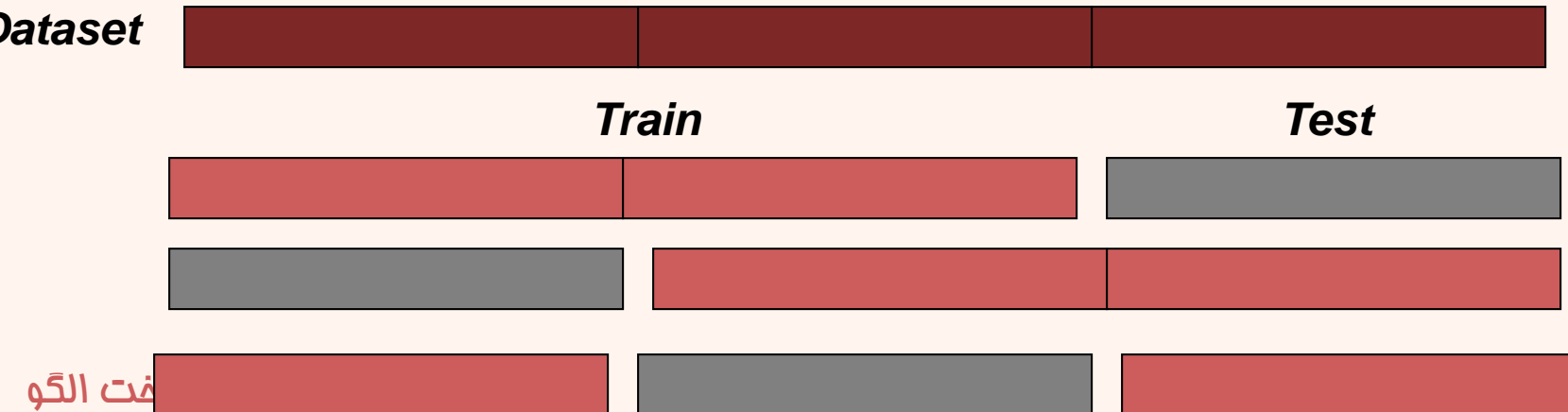
• در این شیوه مجموعه داده‌ها به k قسمت مساوی تقسیم می‌شود. هر بار یک قسمت برای اعتبارسنجی و مابقی به عنوان مجموعه آموزشی به کار می‌رود.

$$\mathcal{V}_1 = \mathcal{X}_1 \quad \mathcal{T}_1 = \mathcal{X}_2 \cup \mathcal{X}_3 \cup \dots \cup \mathcal{X}_k$$

$$\mathcal{V}_2 = \mathcal{X}_2 \quad \mathcal{T}_2 = \mathcal{X}_1 \cup \mathcal{X}_3 \cup \dots \cup \mathcal{X}_k$$

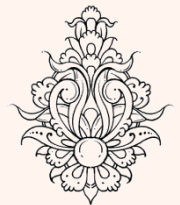
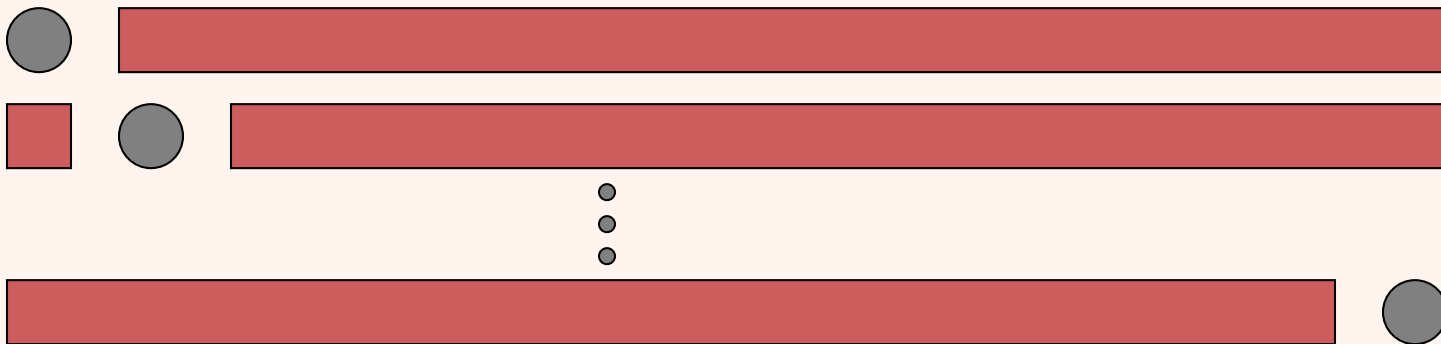
⋮

$$\mathcal{V}_k = \mathcal{X}_k \quad \mathcal{T}_k = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \dots \cup \mathcal{X}_{k-1}$$



Leave-one-out (N-fold cross validation)

- یک حالت خاص k-fold است ($k=N$).
- این شیوه معمولاً در مواردی مورد استفاده قرار می‌گیرد که تهیهی دادهی برچسب خورده دشوار باشد. مانند تشخیص‌های پزشکی



5×2 Cross-Validation

$$\mathcal{T}_1 = \mathcal{X}_1^{(1)} \quad \mathcal{V}_1 = \mathcal{X}_1^{(2)}$$

$$\mathcal{T}_2 = \mathcal{X}_1^{(2)} \quad \mathcal{V}_2 = \mathcal{X}_1^{(1)}$$

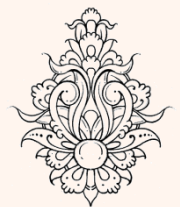
$$\mathcal{T}_3 = \mathcal{X}_2^{(1)} \quad \mathcal{V}_3 = \mathcal{X}_2^{(2)}$$

$$\mathcal{T}_4 = \mathcal{X}_2^{(2)} \quad \mathcal{V}_4 = \mathcal{X}_2^{(1)}$$

⋮

$$\mathcal{T}_9 = \mathcal{X}_5^{(1)} \quad \mathcal{V}_9 = \mathcal{X}_5^{(2)}$$

$$\mathcal{T}_{10} = \mathcal{X}_5^{(2)} \quad \mathcal{V}_{10} = \mathcal{X}_5^{(1)}$$

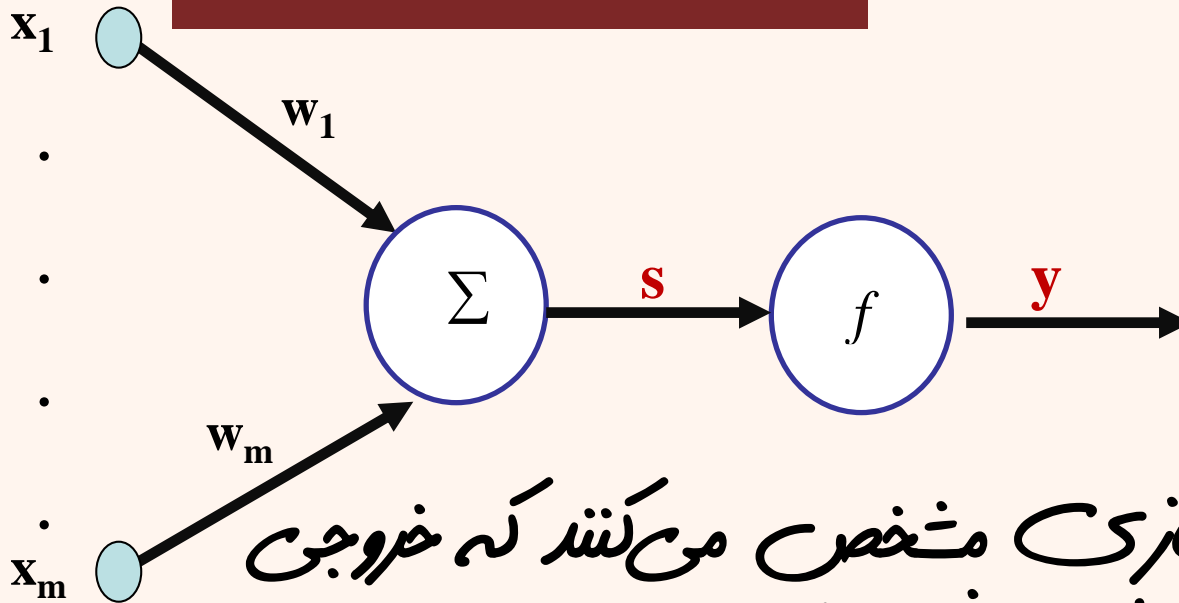


5 times 2 fold cross-validation (Dietterich, 1998)

activation functions

توابع فعال ساز

transfer functions

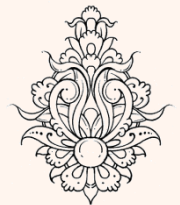


توابع فعال سازی مشخص می کنند که خروجی یک نورون فعال باشد یا نه

$$w_k(n+1) = w_k(n) + (d - y(n)) x_k$$

بسته به تابع f متفاوت است

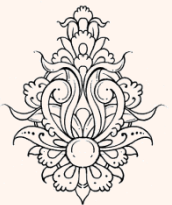
$$-ef'(s)x_k$$



توابع فعال‌ساز

- چنانچه پیش از این مطرح شد، در شبکه‌های عصبی چندلایه، لازم است توابع فعال‌سازی نورون‌های لایه‌های پنهان **غیرخطی** باشند.
 - در صورت خطی بودن نورون‌های لایه مخفی کل شبکه به یک سیستم خطی تبدیل می‌شود.
 - معمولاً برای لایه‌های مخفی یک نوع تابع فعال‌ساز در نظر گرفته می‌شود.
- نوع تابع فعال‌ساز لایه آخر عملکرد شبکه را مشخص می‌کند.
- از طرفی برای محاسبه‌ی گرادیان محلی هر نورون، به مشتق تابع فعال‌سازی آن نیاز است. از این جهت لازم است **مشتق‌پذیر** باشند.

$$\delta_j = \begin{cases} \varphi'(v_j)(d_j - y_j) & \text{IF } j \text{ output node} \\ \varphi'(v_j) \sum_{k \in C} \delta_k w_{kj} & \text{IF } j \text{ hidden node} \end{cases}$$

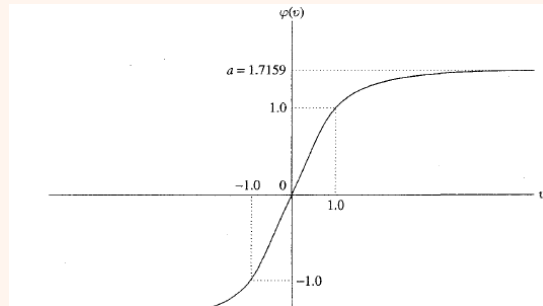


- تابع فعال سازی:

- انتخاب تابع فعال سازی مناسب در کارایی موثر است.

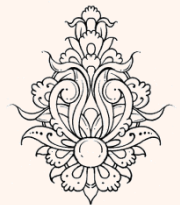
- مقدار خروجی مطلوب:

- این مقدار باید در محدوده‌ی برد تابع فعال سازی باشد، وگرنه پارامترهای آزاد به سمت بی‌نهایت می‌روند.



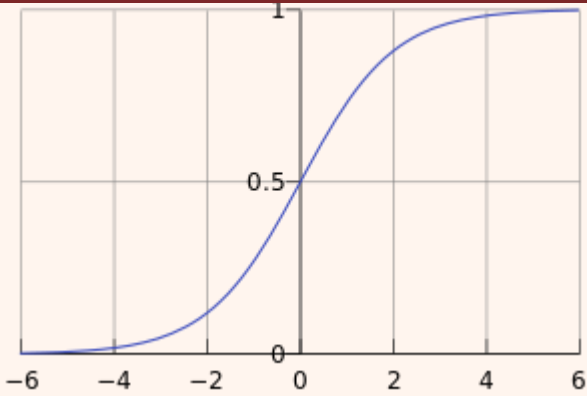
Vanishing Gradients(Exploding Gradients)

هنگامی که عمق شبکه افزایش یابد، گرادینت وزن‌ها لایه ابتدایی (نزدیک به ورودی) کوچک (بزرگ) می‌شوند.



Sigmoid function (logistic)

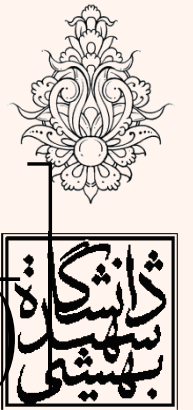
تابع سیگموئید



$$y_j(n) = \varphi_j(v_j(n)) = \frac{1}{1 + \exp(-av_j(n))} \quad a > 0$$

$$\varphi'_j(v_j(n)) = \frac{a \exp(-av_j(n))}{\left[1 + \exp(-av_j(n))\right]^2}$$

$$\varphi'_j(v_j(n)) = a \times \frac{1}{1 + \exp(-av_j(n))} \times \left[1 - \frac{1}{1 + \exp(-av_j(n))}\right]$$



تاشکانه
بهبهشتی

یادگیری عمیق $\varphi'_j(v_j(n)) = ay_j(n) \times [1 - y_j(n)]$

توابع فعال سازی (ادامه...)

Sigmoid function

- اگر نورون در لایه‌ی خروجی باشد:

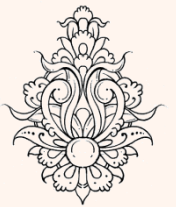
$$\delta_j(n) = e_j(n) \phi'_j(v_j(n))$$

$$\delta_j(n) = a [d_j(n) - o_j(n)] o_j(n) \times [1 - o_j(n)]$$

- و اگر در لایه‌ی مخفی باشد:

$$\delta_j = \alpha y_j(n) \times [1 - y_j(n)] \sum_k \delta_k(n) w_{kj}(n)$$

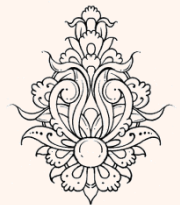
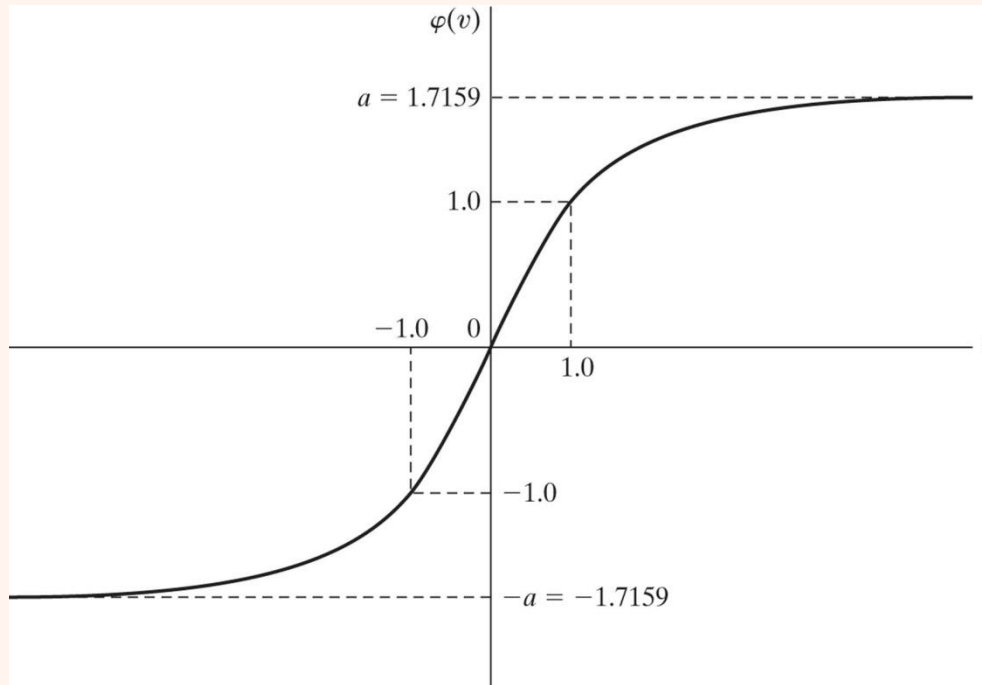
بدین ترتیب گرادیان مملى بدون نیاز به محاسبه صریح مشتق تابع فعال سازی قابل محاسبه می‌باشد.



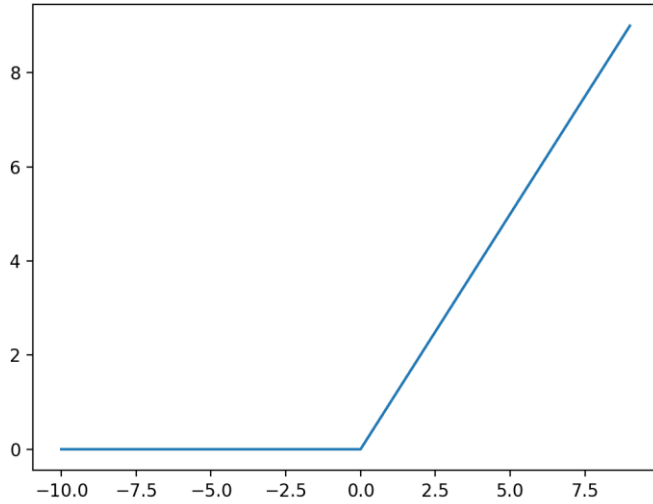
توابع فعال سازی (ادامه...) *hyperbolic tangent*

$$\varphi_j(v_j(n)) = a \tanh(bv_j(n)) \quad (a, b) > 0$$

$$\varphi'_j(v_j(n)) = \frac{b}{a} [a - y_j(n)][a + y_j(n)]$$



ReLU(rectified linear activation function)

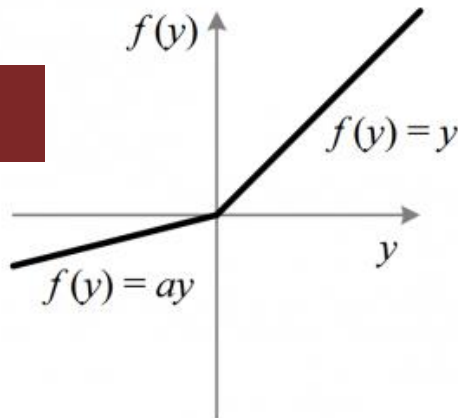


dead neurons

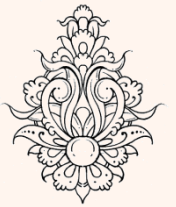
محاسبات ساده
بازنمایی تنگ
شیبه به تابع خطی
عدم اشباع

چنانچه مقدار ورودی نورون منفی باشد، وزن‌های نورون آموزش نمی‌بینند. این شرایط می‌تواند منجر به ایجاد **نورون مرده** شود، نورونی که هیچ‌گاه فعال نمی‌شود.

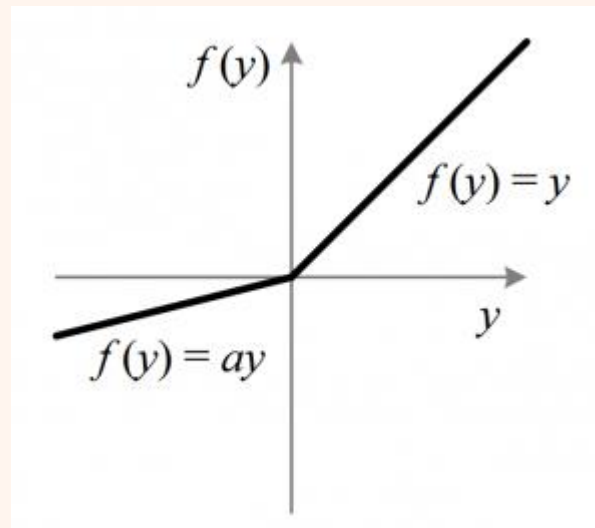
Leaky ReLU



یادگیری عمیق

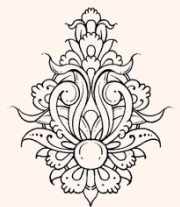
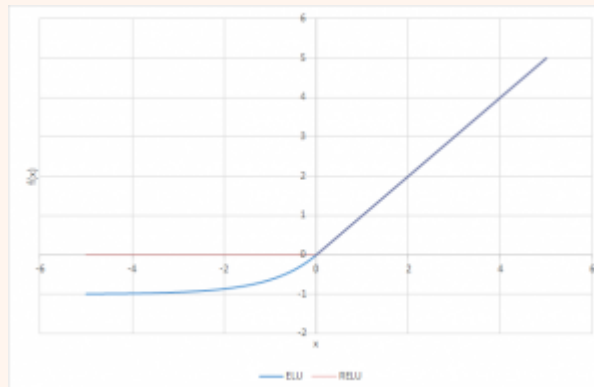


Parameterised ReLU



در این شیوه پارامتر a نیز به پارامترهای قابل آموزش اضافه می‌شود.

Exponential Linear Unit(ELU)



تأثیر نرخ آموزش

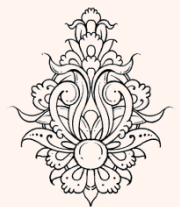
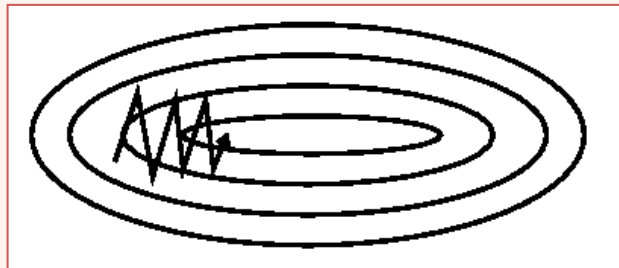
- چنانچه پیش از این مطرح شد، نرخ آموزش بالا موجب ناپایداری می‌شود، در حالی که نرخ آموزش پایین فرآیند آموزش را طولانی خواهد کرد.

نرخ آموزش کوچک: همگرایی کند است ولی روند حرکت بدون تغییرات زیاد

نرخ آموزش بزرگ: همگرایی تند است ولی روند حرکت با تغییرات زیاد

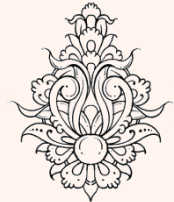
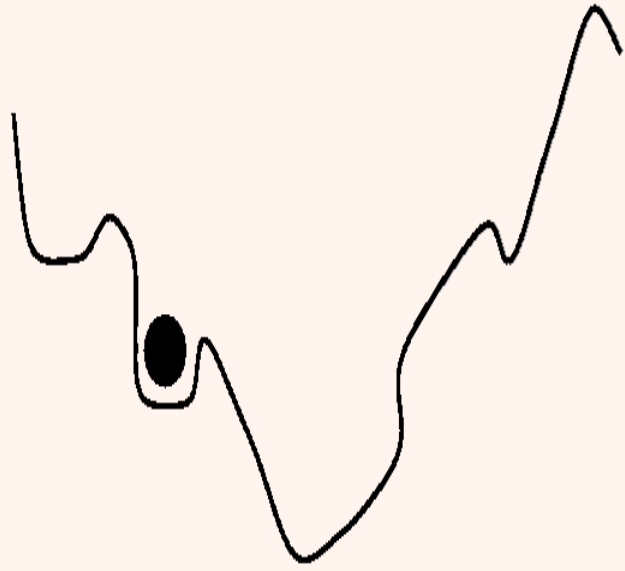
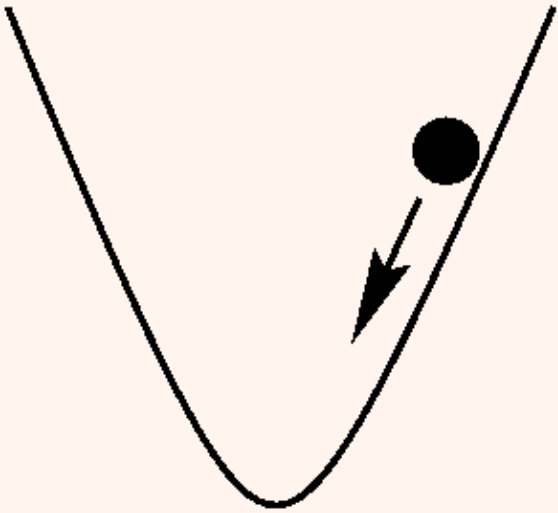
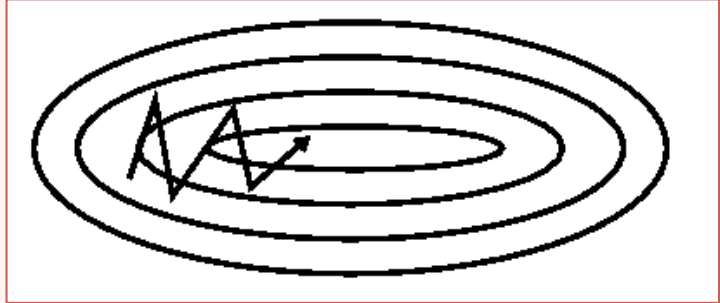
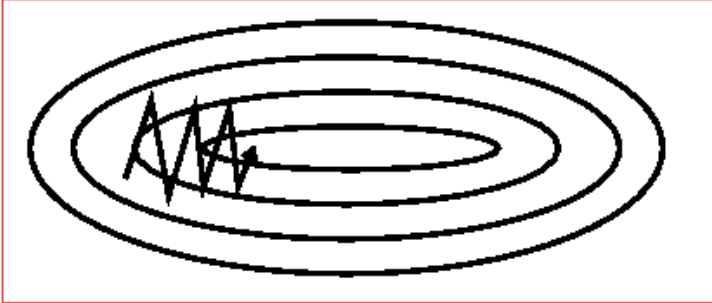
- همچنین می‌توان نرخ آموزش را برای وزن‌های مختلف **متغیر** در نظر گرفت.

Connection dependent



Momentum and Learning Rate Adaptation

Local Minima



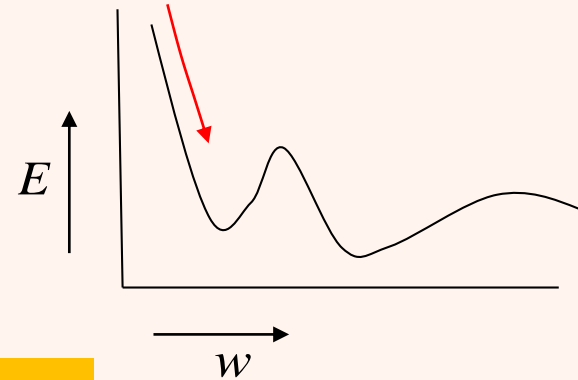
روش استفاده از momentum

$$w_{new} = w_{old} + \Delta w$$

• داشتیم:

generalized delta rule

$$\Delta w_{ji}^l(n) = \alpha \Delta w_{ji}^l(n-1) - \eta \frac{\partial E(n)}{\partial w_{ji}^l(n)}$$



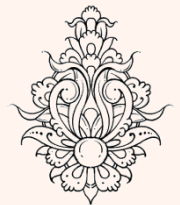
ثابت Momentum

$$0 \leq \alpha < 1$$

تخیرات وزن در مرحله پیشین

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

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



به روز رسانی وزن ها

روش استفاده از momentum

تغییرات وزن در مرحله پیشین

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

برای iteration اول بررسی می کنیم:

$$\Delta w_{ji}^l(0) = \eta \delta_j^l(0) \cdot y_i^{l-1}(0)$$

$$\Delta w_{ji}^l(1) = \alpha \eta \delta_j^l(0) \cdot y_i^{l-1}(0) + \eta \delta_j^l(1) \cdot y_i^{l-1}(1)$$

$$\Delta w_{ji}^l(2) = \alpha^2 \eta \delta_j^l(0) \cdot y_i^{l-1}(0) + \alpha \eta \delta_j^l(1) \cdot y_i^{l-1}(1) + \eta \delta_j^l(2) \cdot y_i^{l-1}(2)$$

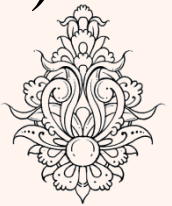
تاثیر Momentum

$$\Delta w_{ji}^l(n) = \eta \sum_{h=0}^n \alpha^{n-h} \delta_j^l(h) \cdot y_i^{l-1}(h)$$

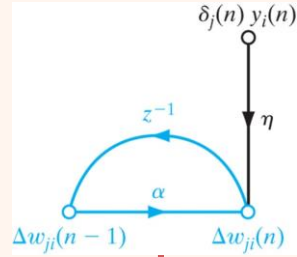
نرخ آموزش

یادگیری عمیق

ثابت Momentum



روش استفاده از momentum



• اگر ثابت ممنتوم صفر باشد به روز رسانی عادی است.
هر اندازه این ثابت به سمت یک میل کند به روز رسانی بیشتر از الگوی قبلی تبعیت می‌کند.

- در صورتی که که تخریفات هم‌جهت باشند، در نظر گرفتن momentum باعث **تسریع** آموزش می‌شود.

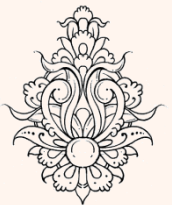
accelerating effect

- در صورت تخریفات یکسان نباشند، موجب **پایداری** فرآیند آموزش می‌شود.

stabilizing effect

• واضح است که ثابت کمتر از صفر و بیشتر از یک ناکارآمد است.

• با تخریب میزان نرخ آموزش می‌توان پاسخی بهینه دریافت کرد.



ادامه...

$$\Delta w_{ij}^l(n) = -\eta \sum_{h=0}^n \alpha^{n-h} \frac{\partial E(n)}{\partial w_{ji}(n)}$$

ثابت Momentum	iteration
α	1
α^2	2
.	.
α^{n-h}	n

هرچه iteration بالاتر رود ضریب کوچکتر خواهد شد بدین ترتیب از مقدار بهینه کمتر فاصله خواهیم گرفت



روش استفاده از momentum

$$\Delta w_{ji}^l(n) = -\eta \sum_{h=0}^n \alpha^{n-h} \delta_j^l(h) \cdot y_i^{l-1}(h)$$

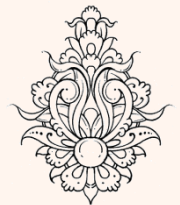
نرخ آموزش

ثابت Momentum

$$\Delta w_{ji}^l(n) = -\eta \sum_{h=0}^n \alpha^{n-h} \frac{\partial E(n)}{\partial w_{ji}(n)}$$

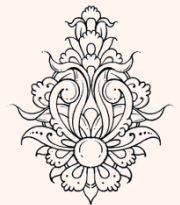
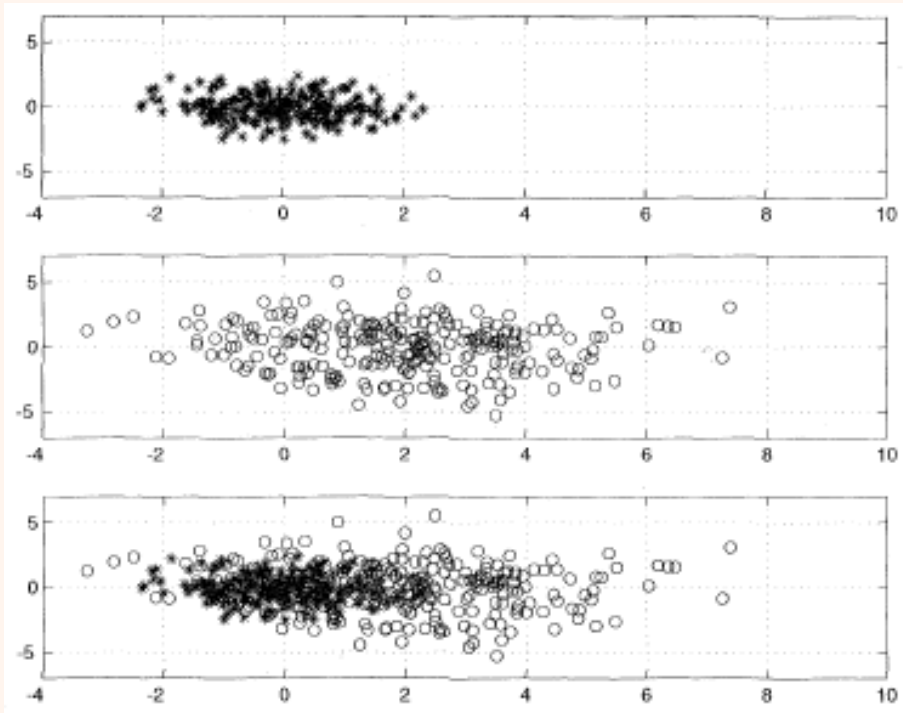
- میانگین زمان یادگیری برای یک شبکه بدون/با استفاده از Momentum

momentum	Training time
۰	۲۱۷
۰.۹	۹۵



مثال

- هدف طراحی شبکه‌ای است که دو دسته داده زیر را از هم جدا کند.



مثال

Class C_1 :
$$f_{\mathbf{x}}(\mathbf{x}|C_1) = \frac{1}{2\pi\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2} \|\mathbf{x} - \boldsymbol{\mu}_1\|^2\right)$$

$$\boldsymbol{\mu}_1 = \text{mean vector} = [0, 0]^T$$

$$\sigma_1^2 = \text{variance} = 1$$

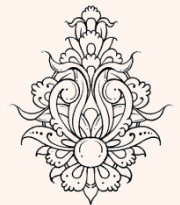
Class C_2 :
$$f_{\mathbf{x}}(\mathbf{x}|C_2) = \frac{1}{2\pi\sigma_2^2} \exp\left(-\frac{1}{2\sigma_2^2} \|\mathbf{x} - \boldsymbol{\mu}_2\|^2\right)$$

$$\boldsymbol{\mu}_2 = \text{mean vector} = [2, 0]^T$$

$$\sigma_2^2 = \text{variance} = 1$$

Likelihood ratio

$$\Lambda(\mathbf{x}) = \frac{f_{\mathbf{x}}(\mathbf{x}|C_1)}{f_{\mathbf{x}}(\mathbf{x}|C_2)} \leq 1$$



ادامہ ...

$$\Lambda(\mathbf{x}) = \frac{\sigma_2^2}{\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2} \|\mathbf{x} - \boldsymbol{\mu}_1\|^2 + \frac{1}{2\sigma_2^2} \|\mathbf{x} - \boldsymbol{\mu}_2\|^2\right)$$

$$\frac{\sigma_2^2}{\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2} \|\mathbf{x} - \boldsymbol{\mu}_1\|^2 + \frac{1}{2\sigma_2^2} \|\mathbf{x} - \boldsymbol{\mu}_2\|^2\right) = 1$$

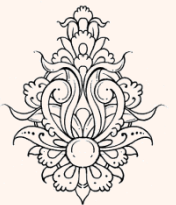
$$\frac{1}{2\sigma_2^2} \|\mathbf{x} - \boldsymbol{\mu}_2\|^2 - \frac{1}{2\sigma_1^2} \|\mathbf{x} - \boldsymbol{\mu}_1\|^2 = 4 \log \frac{\sigma_2^2}{\sigma_1^2}$$

$$\|\mathbf{x} - \mathbf{x}_c\|^2 = r^2$$

Optimum decision boundary

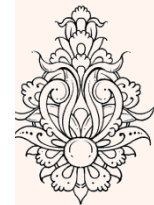
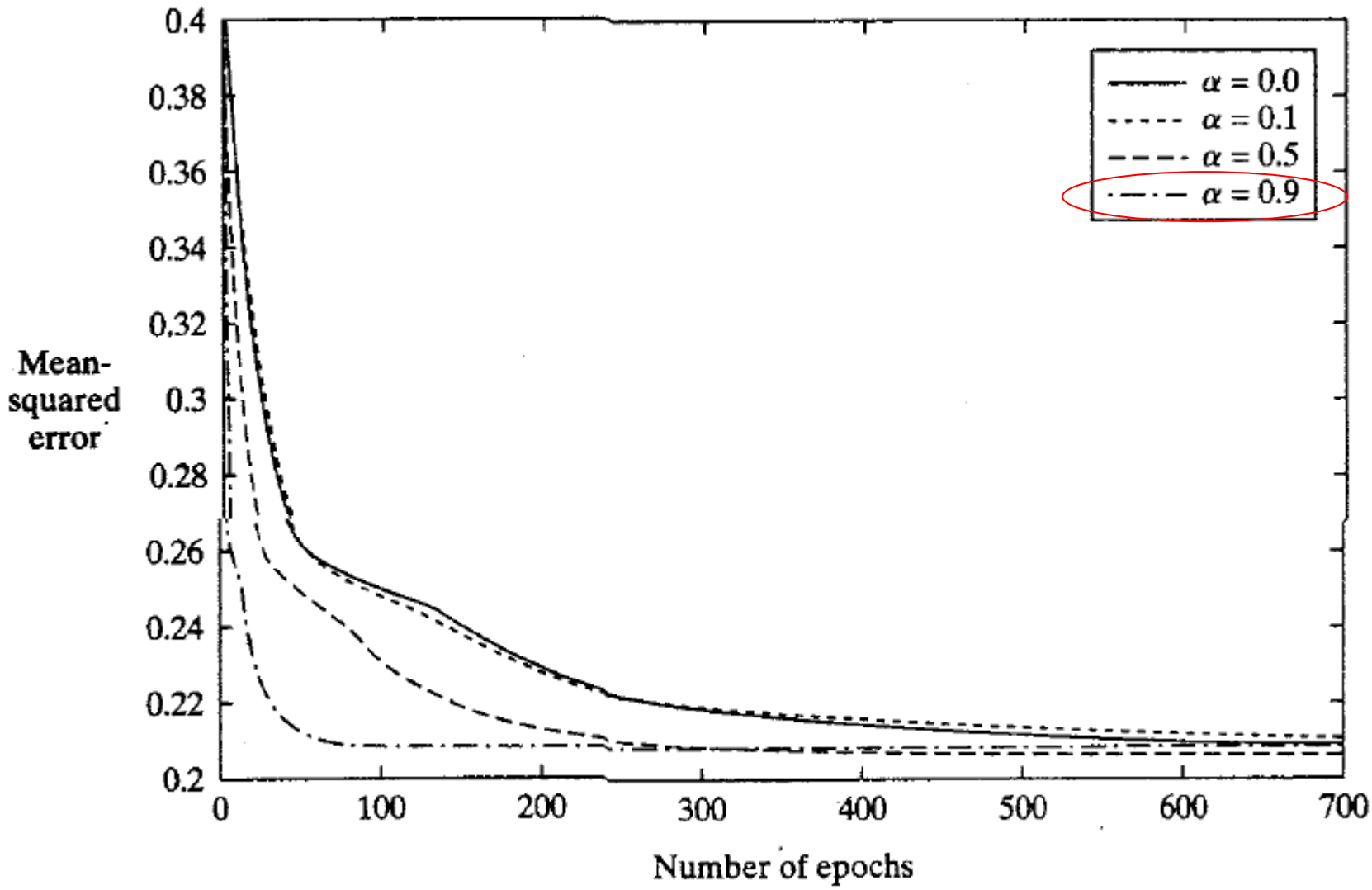
$$P_c = 0.8151$$

Probability of correct classification



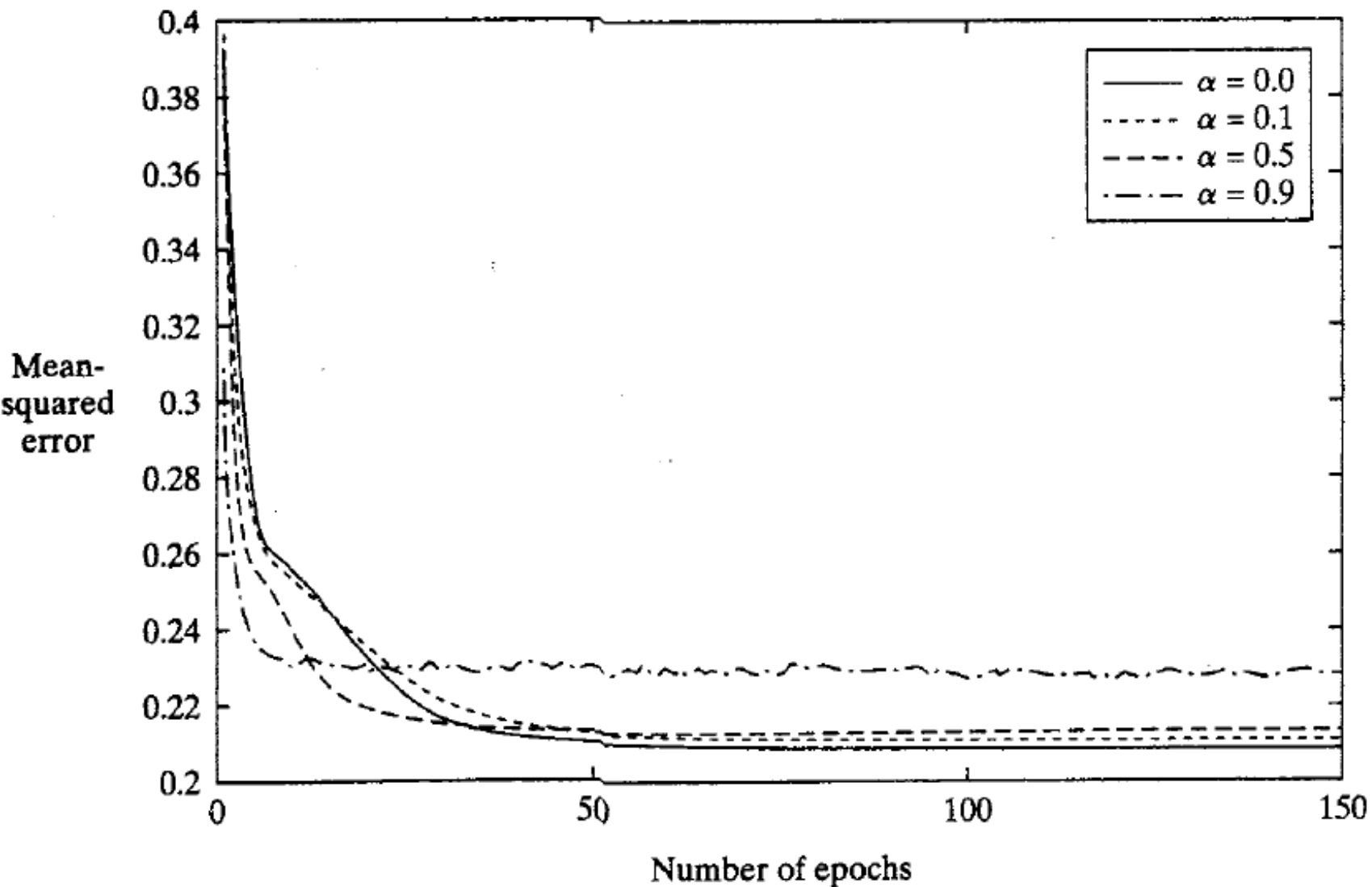
$\eta=0.01$

مثال



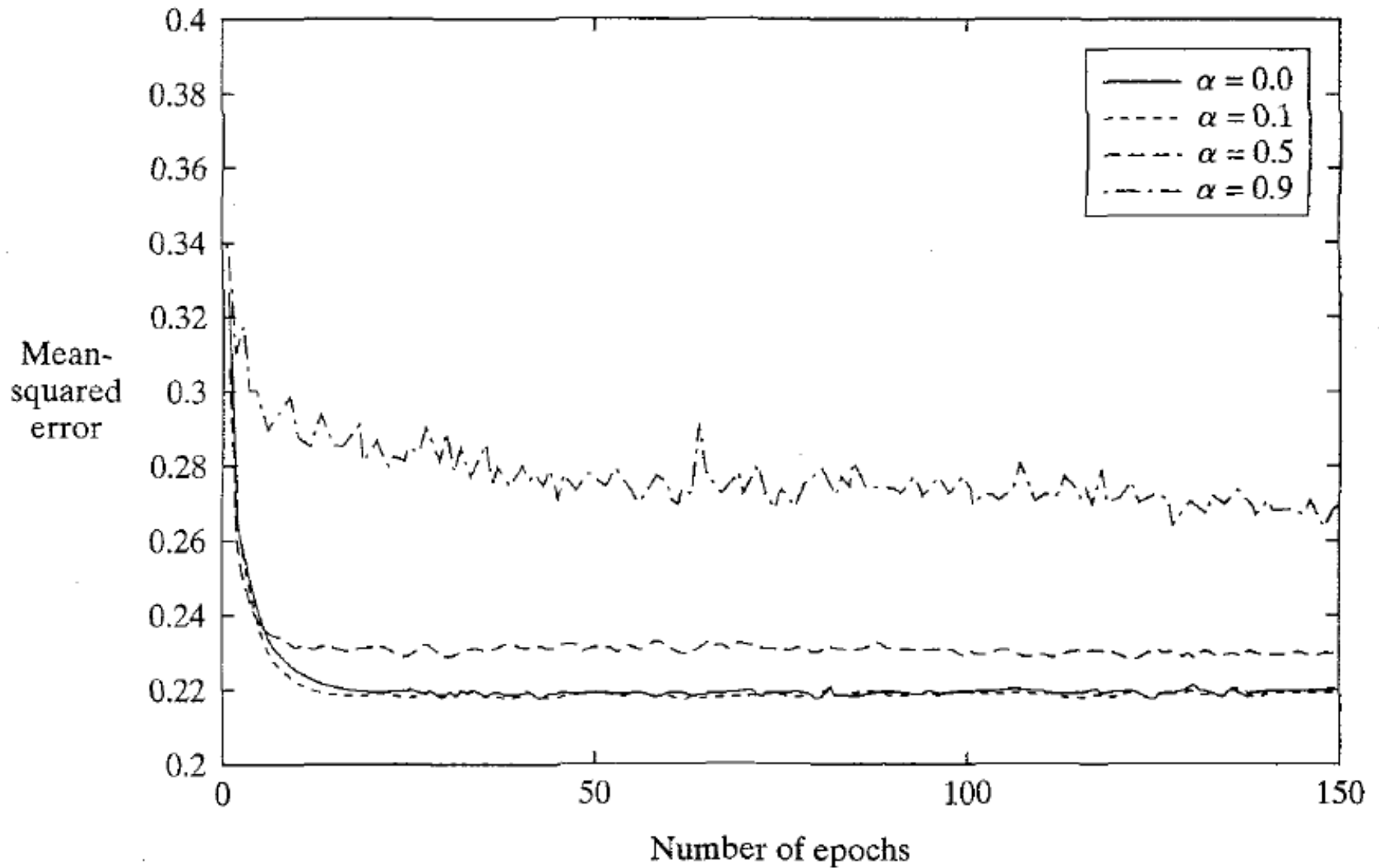
دانشگاه
تهران
پیشرو

$\eta = 0.1$



آزاد
بهرین

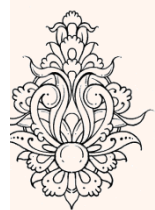
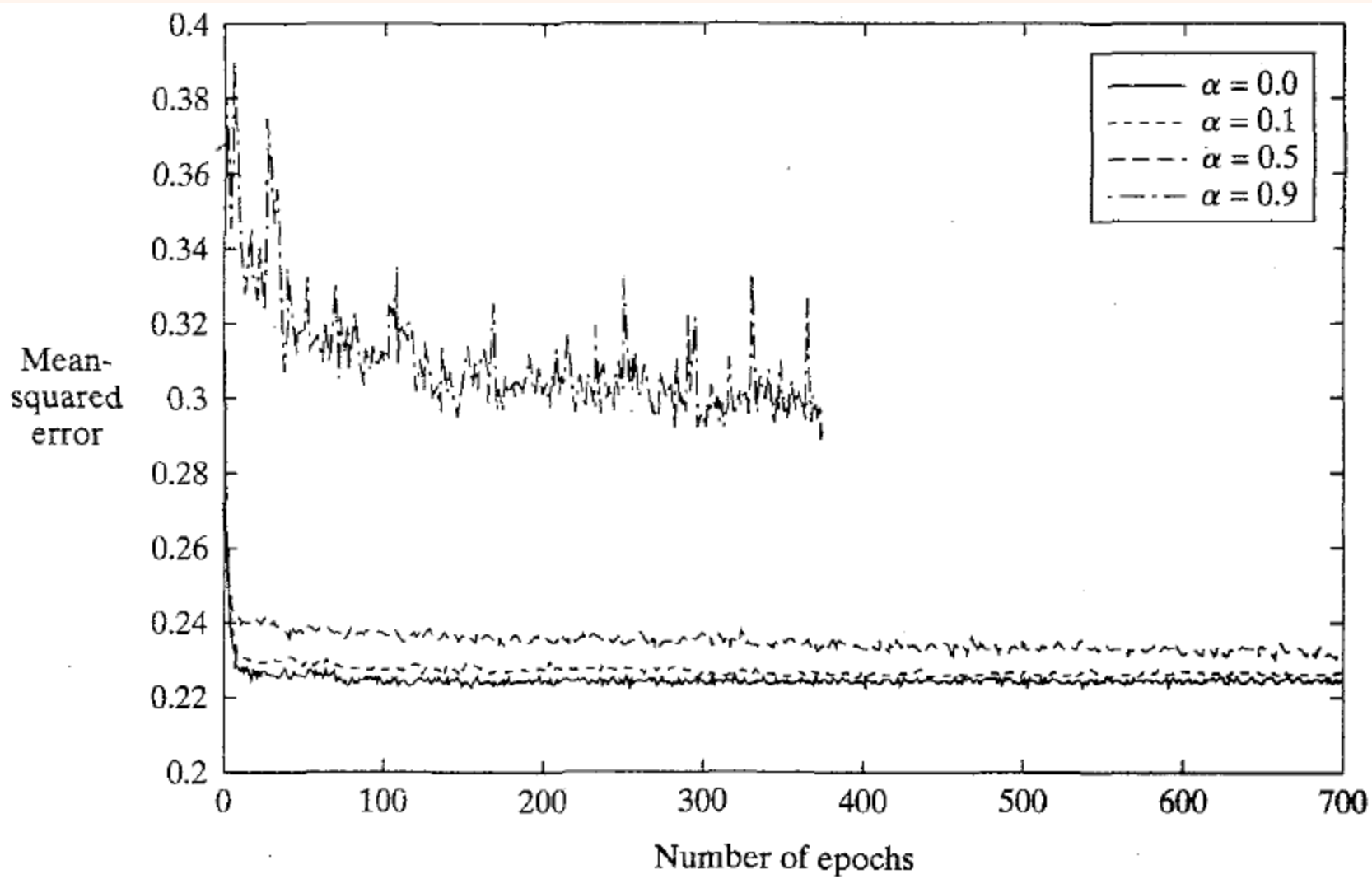
$\eta = 0.5$



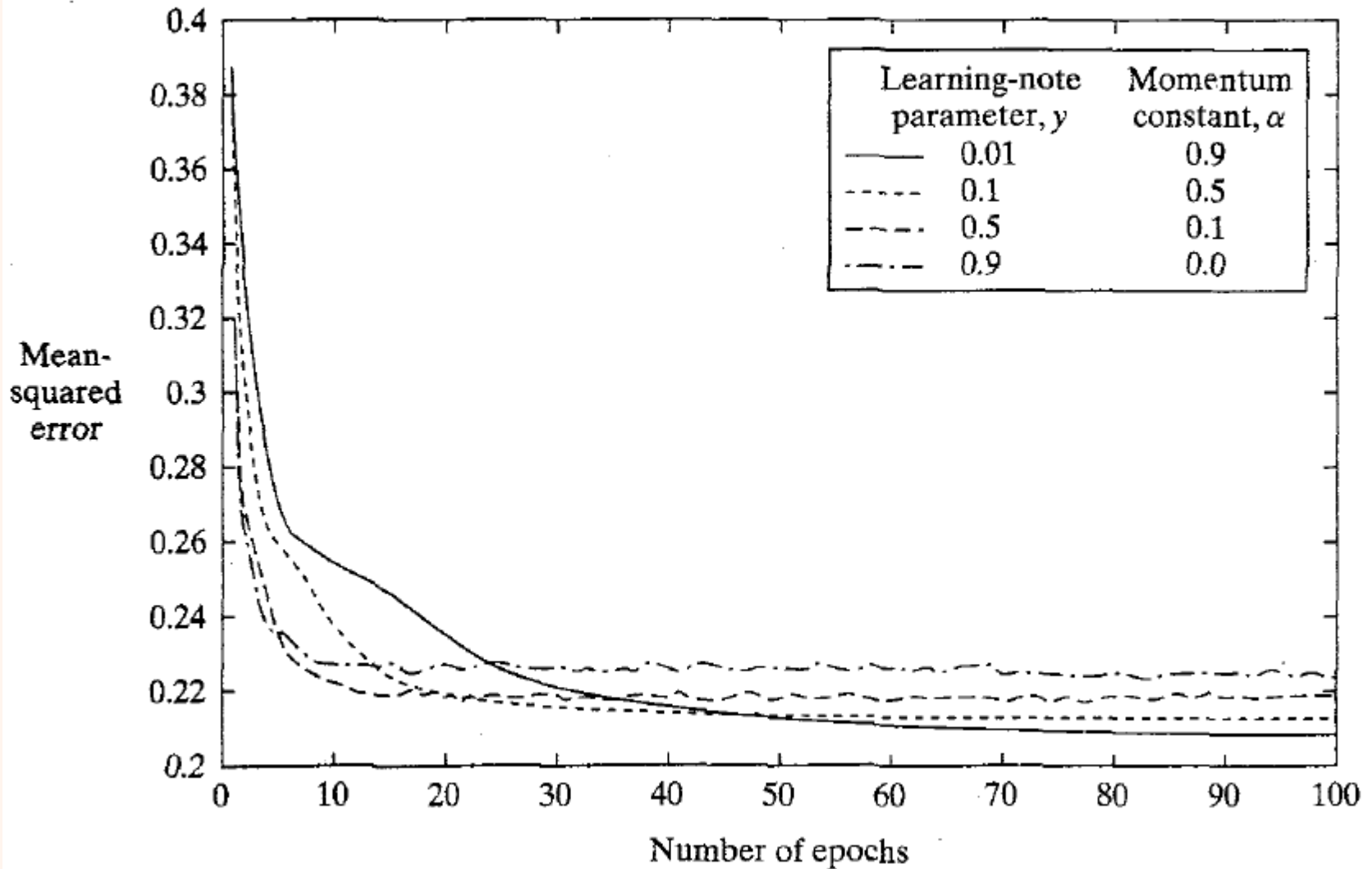
موسسه

پژوهش

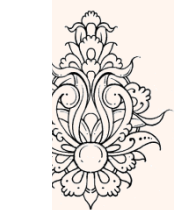
$\eta = 0.9$



دانشگاه
تهران
پیشرو

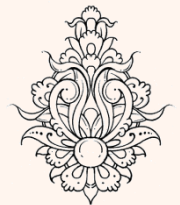
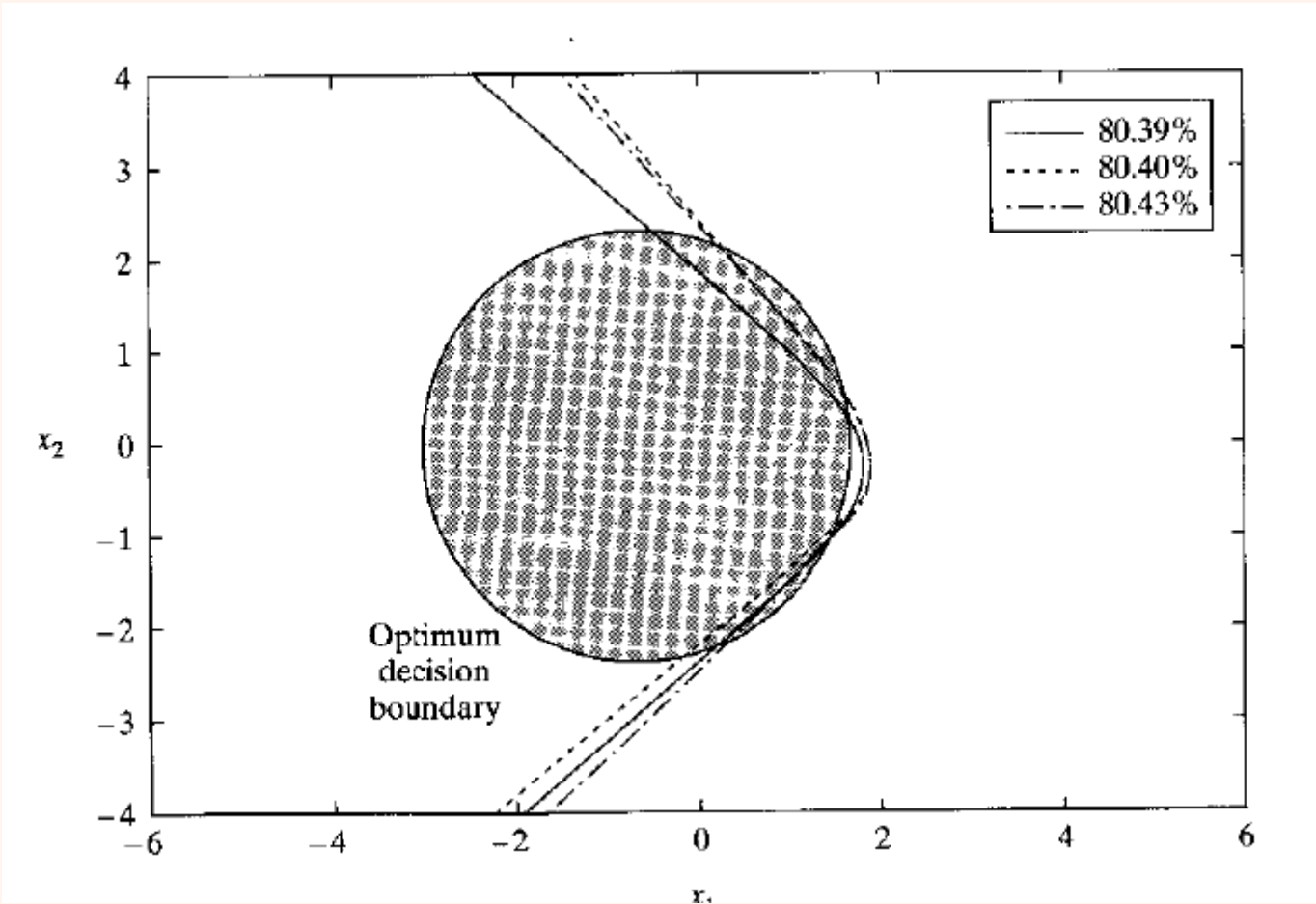


Best learning curves selected from the four parts



دانشگاه شهید بهشتی

مثال



نرخ یادگیری و ضریب گشتاور

- هر اندازه نرخ یادگیری کوچکتر باشد تخخیرات نمودار کندتر است، اما کمینهی محلی بهتری را می‌تواند به دست آورد. در این شرایط فضای بیشتری از رویه‌ی خطا مورد بررسی قرار می‌گیرد.

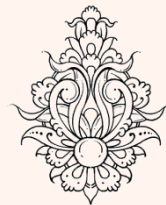
For $\eta \rightarrow 0$, $\alpha \rightarrow 1$

باعث افزایش سرعت همگرایی

For $\eta \rightarrow 1$, $\alpha \rightarrow 0$

باعث ثبات یادگیری

- در صورتی که برای نرخ یادگیری و ضریب گشتاور اعداد ثابت و بزرگی در نظر گرفته شود، باعث خواهد شد میزان خطا به صورت نوسانی تخخیر کند و یا به مقدار بالاتری همگرا شود.



روش نرخ یادگیری متغیر (وفاقی)

Adaptive Learning Rate

Bold Driver

ضریب momentum را صفر می‌گذاریم

if $E(k+1) > (1 + \xi)E(k) \longrightarrow w(k+1) = w(k)$

$$\xi > 0.01$$

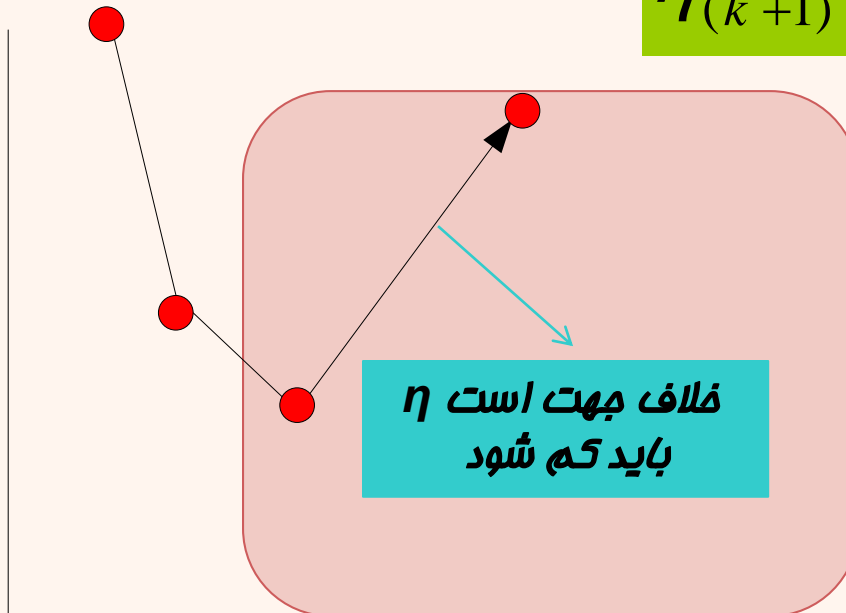
در این حالت جهت حرکت نامناسب است

$$\eta_{(k+1)} = \eta_{(k)} \cdot \rho$$

$$0 < \rho < 1$$

معمولا ۰.۵

SSE



روش نرخ یادگیری متغیر (وفاقی)

ضریب momentum به مقدار اصلی برمی‌گردد

$$\text{if } E(k+1) < E(k) \longrightarrow w(k+1) = w(k) + \Delta w(k)$$

در این حالت جهت حرکت مناسب است

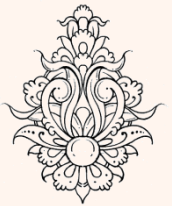
$$\eta_{(k+1)} = \eta_{(k)} \cdot d$$

$$1 < d < 2$$

معمولا ۱.۰۱ تا ۱.۰۵

SSE

η بیشتر شود



روش نرخ یادگیری متغیر (وفاقی)

$$\text{if } E(k+1) > (1 + \beta)E(k)$$

$$\beta < \xi$$

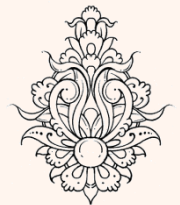
$$w(k+1) = w(k) + \Delta w(k)$$

$$\eta_{(k+1)} = \eta_{(k)}$$

چنانچه شرط قبلی برقرار نباشد

وزن های جدید را قبول کرده
ولی η ثابت می ماند و α به مقدار
اولیه تغییر داده می شود.

این شیوه تنها برای آموزش دسته ای مفید است، در
صورت استفاده در روش ترتیبی منجر به واگرایی می شود.

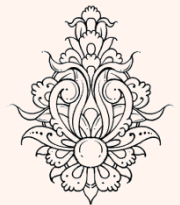


- در روش ترتیبی، استفاده از Bold driver منجر به واگرایی الگوریتم می‌شود، از این رو به جای در نظر گرفتن نرخ وفقی، نرخ آموزش به صورت نزولی در نظر گرفته می‌شود.

$$\eta_{(k)} = \frac{\eta_{(0)}}{1 + \frac{k}{T}}$$

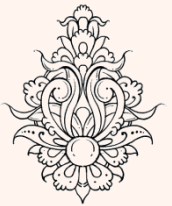
- بدین ترتیب، در گام‌های اولیه نرخ آموزش تقریباً ثابت است. بعد از رسیدن به محدوده می‌نیم، نرخ آموزش کاهش می‌یابد.

- در این حالت یک هایپرپارامتر آزاد به مجموعه پارامترها افزوده خواهد شد.



روش‌های بهبود کارایی

- غالباً گفته می‌شود که طراحی شبکه‌ی عصبی بیش از آن که «علم» باشد، نوعی «هنر» است، چرا که تنظیم پارامترهای زیادی که در طراحی دخیل هستند، تا حد زیادی به تجربه‌ی شخص بستگی دارد.
- با این وجود روش‌هایی برای بهبود کارایی شبکه وجود دارد:
- روش‌های ترتیبی در مقابل دسته‌ای:
 - جاهایی که مجموعه‌ی آموزشی بزرگی در اختیار داریم و افزودن در داده‌ها بالاست روش ترتیبی مناسب‌تر است.
 - مواردی که مجموعه‌ی آموزشی ثابت نیست، داده‌های جدید به مجموعه اضافه شود.



روش‌های بهبود کارایی (ادامه...)

Maximizing information content

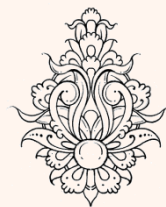
• استفاده از حداکثر اطلاعات در آموزش:

– مجموعه‌ی آموزشی نقش بسیار مهمی در همگرایی و تصمیم‌پذیری دارد.

• بهتر است از نمونه‌هایی استفاده شود که بیشترین خطا در آموزش را ایجاد می‌کند.

• نمونه‌هایی که با هم متفاوت هستند. این کار باعث می‌شود گستره‌ی وسیع‌تری از وزن‌ها بررسی شوند.

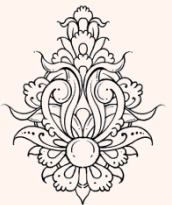
– در بازشناسی الگو در شیوه‌ی ترتیبی، اعمال الگوها به صورت تصادفی موجب می‌شود تا داده‌های یک کلاس به صورت پشت سرهم انتخاب نشوند.



روش‌های بهبود کارایی (ادامه...)

emphasizing scheme

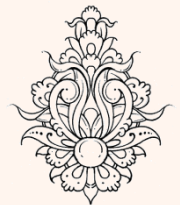
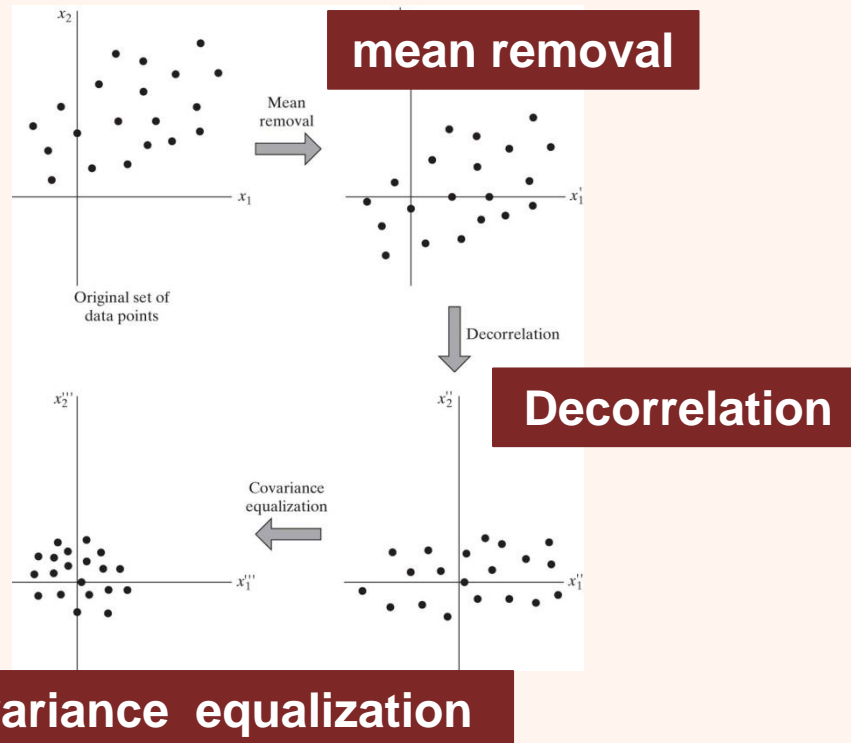
- نمونه‌های دشوارتر به شبکه بیشتر اعمال شوند.
این شیوه البته با مشکلاتی هم روبرو است.
 - ترتیب اعمال نمونه‌های آموزشی به هم می‌ریزد.
 - یا داده‌های بیرون‌هسته (outlier) در مجموعه‌ی آموزشی وجود داشته باشد. چنین داده‌های «تعمیم‌پذیری» را با چالش مواجه می‌کند.



روش‌های بهبود کارایی (ادامه...)

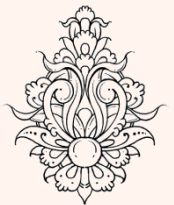
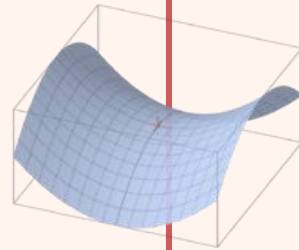
- بهنجارسازی ورودی‌ها:

– بر روی داده‌های ورودی باید پیش‌پردازش‌های صورت پذیرد. بهتر است میانگین ورودی‌ها صفر شود. در غیر این صورت به صورت زیگزاگ به سمت میانی حرکت می‌کند.



روش‌های بهبود کارایی (ادامه...)

- مقداردهی اولیهی وزن‌ها:
 - مقادیر بزرگ ← رفتن به نامیهی اشباع ← کند شدن آموزش
 - مقادیر کوچک ← نزدیکی مبدأ ← مبدأ به صورت «نقطه‌ی زینی» است.
 - مقدار مناسب مقداری بین این دو است.
- ثابت می‌شود در صورتی که میانگین مقدار اولیهی وزن‌ها صفر و انحراف معیار آن $\sigma_w = m^{-1/2}$ در نظر گرفته شود، (m تعداد اتصالات به نرون) نتیجه‌ی بهتری به دست می‌آید (زمانی که تابع انگیزش tanh باشد).



روش‌های بهبود کارایی (ادامه...)

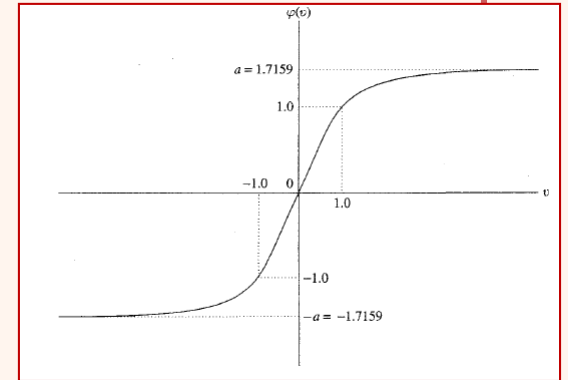
مفروضات

$$v_j = \sum_{i=1}^m w_{ji} y_i$$

$$\mu_y = E[y_i] = 0 \quad \text{for all } i$$

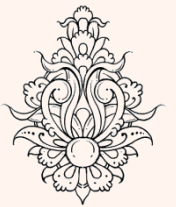
$$\sigma_y^2 = E[(y_i - \mu_i)^2] = E[y_i^2] = 1 \quad \text{for all } i$$

$$E[y_i y_k] = \begin{cases} 1 & \text{for } k = i \\ 0 & \text{for } k \neq i \end{cases}$$



$$\mu_w = E[w_{ji}] = 0 \quad \text{for all } (j, i) \text{ pairs}$$

$$\sigma_w^2 = E[(w_{ji} - \mu_w)^2] = E[w_{ji}^2] \quad \text{for all } (i, j) \text{ pairs}$$

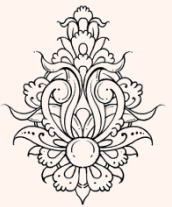
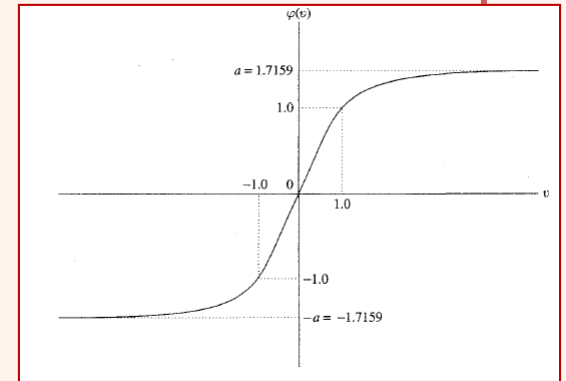


روش‌های بهبود کارایی (ادامه...)

انتخاب

$$\mu_v = E[v_j] = E\left[\sum_{i=1}^m w_{ij} y_i\right] = \sum_{i=1}^m E[w_{ji}] E[y_i] = 0$$

$$\begin{aligned}\sigma_v^2 &= E\left[(v_j - \mu_v)^2\right] = E[v_j^2] \\ &= E\left[\sum_{i=1}^m \sum_{k=1}^m w_{ji} w_{jk} y_i y_k\right] \\ &= \sum_{i=1}^m \sum_{k=1}^m E[w_{ji} w_{jk}] E[y_i y_k] \\ &= \sum_{i=1}^m E[w_{ji}^2] \\ &= m\sigma_w^2\end{aligned}$$



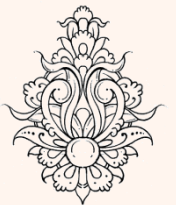
در صورتی که وزن‌ها به گونه‌ای در نظر گرفته شوند که انحراف آن‌ها $m^{-1/2}$ باشد، مقدار v واریانس یک خواهد داشت.



روش‌های بهبود کارایی (ادامه...)

• نرخ آموزش:

- در مورد نرخ آموزش وقتی صحبت شد.
- همه‌ی نرون‌ها باید با یک سرعت آموزش ببینند. نورون‌لایه‌ی آخر معمولاً گرادیان بزرگ‌تری دارد. بنابراین بهتر است، نرخ آموزش لایه‌های آخر، کمتر در نظر گرفته شود.
- توصیه می‌شود نرخ آموزش بر اساس تعداد اتصالات نیز تنظیم شود.



روش‌های سرعت بخشیدن به همگرایی

- برای تصحیح وزن w_{ji} ابتدا خطا را محاسبه می‌کردیم:

$$E(k) = \frac{1}{2} \sum_{j \in C} e_j^2(k) \quad e_j(k) = d_j(k) - y_j(k)$$

$$y_j = \varphi(v_j)$$

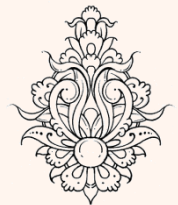
$$\Delta w_{ji}(k) = -\frac{\partial E(k)}{\partial w_{ji}(k)} = -\frac{1}{2} \frac{\partial e_j^2(k)}{\partial w_{ji}(k)}$$

$$= -e_j(k) \cdot \frac{\partial e_j(k)}{\partial w_{ji}(k)}$$

$$= e_j(k) \cdot \frac{\partial y_j(k)}{\partial w_{ji}(k)}$$

واقعاً

مطلوب



روش‌های سرعت بخشیدن به همگرایی

$$\Delta w_{ji}(k) = e_j(k) \cdot \frac{\partial y_j(k)}{\partial v_j(k)} \cdot \frac{\partial v_j(k)}{\partial w_{ji}(k)}$$

اگر f تابع sigmoid باشد

$$y_j = \varphi(v_j)$$

ورودی به شافنی زا

$$\Delta w_{ji}(k) = e_j(k) y_j(k) (1 - y_j(k)) \cdot \frac{\partial v_j(k)}{\partial w_{ji}(k)}$$

مناسب

$$e_j \ll$$

$$y_j \approx 0$$

بزرگ ولی

$$e_j$$

$$y_j \approx 1$$

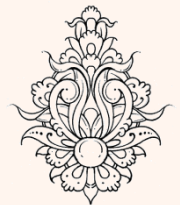
بزرگ ولی

$$e_j$$

$$\Delta w_{ji} \ll$$

نامناسب

کند شدن همگرایی



• تعویض تابع معیار خطا

– در روش عادی B.P تابع معیار خطا

$$E(k) = \sum_{h=1}^M e_h^2(k)$$

– روش (V-O) Van Oyen, Nienhuis

$$E(k) = - \sum_{h=1}^M [d_h \ln(y_h(k)) + (1 - d_h) \ln(1 - y_h(k))]$$

فروبی مطلوب

فروبی واقعی

Binary cross entropy



Improving the convergence of the back-propagation algorithm

Van Ooyen, A., and Nienhuis, B. (1992). *Neural Networks* 5: 465-471

$$E(k) = - \sum_{h=1}^M [d_h \ln(y_h(k)) + (1 - d_h) \ln(1 - y_h(k))]$$

$$\Delta w_{ji}(k) = - \frac{\partial E(k)}{\partial w_{ji}(k)} = - \frac{\partial E(k)}{\partial y_j(k)} \cdot \frac{\partial y_j(k)}{\partial v_j(k)} \cdot \frac{\partial v_j(k)}{\partial w_{ji}(k)}$$

$$= \left[d_j \times \frac{1}{y_j(k)} - (1 - d_j) \times \frac{1}{(1 - y_j(k))} \right] \cdot y_j(k) (1 - y_j(k)) \cdot y_i$$

$$= e_j \cdot \frac{\partial v_j(k)}{\partial w_{ji}(k)}$$

e_j

فرومی لایه ما قبل آخر



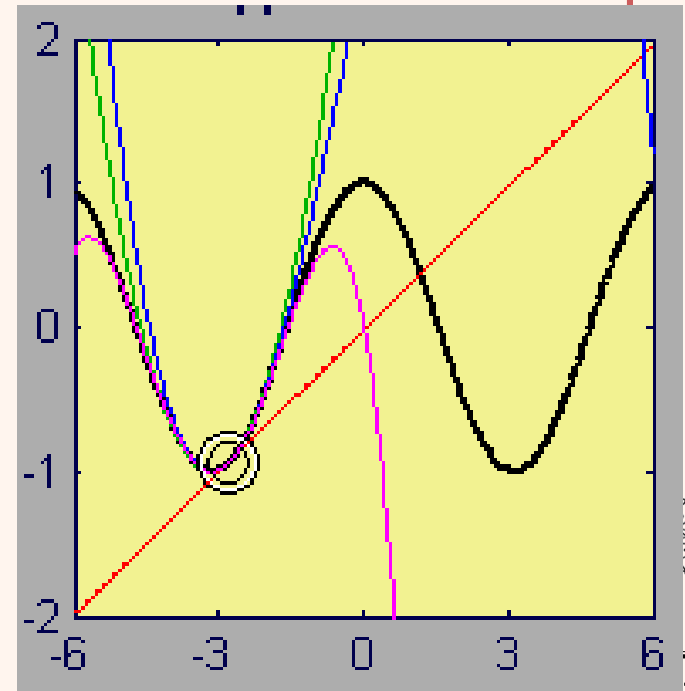
بسط تیلور

- توابع تحلیلی قابل تقریب زدن با چندجمله‌ای‌ها هستند.

$$F(x) = F(x^*) + \frac{d}{dx}F(x) \Big|_{x=x^*} (x - x^*)$$

$$+ \frac{1}{2} \frac{d^2}{dx^2} F(x) \Big|_{x=x^*} (x - x^*)^2 +$$

$$+ \frac{1}{n!} \frac{d^n}{dx^n} F(x) \Big|_{x=x^*} (x - x^*)^n +$$



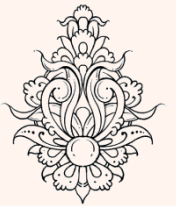
nnd8ts1



حالت برداری

$$F(\mathbf{X}) = F(x_1, x_2, \dots, x_n)$$

$$\begin{aligned} F(\mathbf{X}) = & F(\mathbf{X}^*) + \frac{\partial}{\partial x_1} F(\mathbf{X}) \Big|_{\mathbf{X}=\mathbf{X}^*} (x_1 - x_1^*) + \frac{\partial}{\partial x_2} F(\mathbf{X}) \Big|_{\mathbf{X}=\mathbf{X}^*} (x_2 - x_2^*) \\ & + \dots + \frac{\partial}{\partial x_n} F(\mathbf{X}) \Big|_{\mathbf{X}=\mathbf{X}^*} (x_n - x_n^*) + \frac{1}{2} \frac{\partial^2}{\partial x_1^2} F(\mathbf{X}) \Big|_{\mathbf{X}=\mathbf{X}^*} (x_1 - x_1^*)^2 \\ & + \frac{1}{2} \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{X}) \Big|_{\mathbf{X}=\mathbf{X}^*} (x_1 - x_1^*) (x_2 - x_2^*) + \dots \end{aligned}$$



فرم ماتریسی

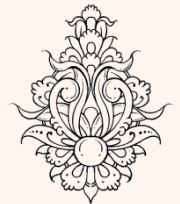
$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \dots$$

Gradient

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n} F(\mathbf{x}) \end{bmatrix}$$

Hessian

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_1 \partial x_n} F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_2 \partial x_n} F(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_n \partial x_2} F(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_n^2} F(\mathbf{x}) \end{bmatrix}$$



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

فرم کلی یک تابع درجه ۲

الگوریتم‌های بهینه‌سازی

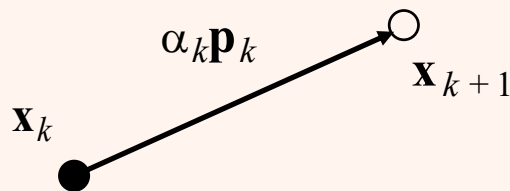
برای یافتن نقطه‌ی مینیمم خطی بر روی رویه‌ی کارایی (performance(error) surface) شیوه‌های متفاوتی وجود دارد.

هدف یافتن نقطه‌ی مینیمم تابع خطی ($F(X)$) می‌باشد با استفاده از الگوریتم‌های تکرار شونده است. از یک حدس اولیه شروع خواهیم کرد.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

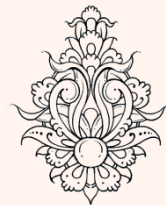
۱

$$\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$$



\mathbf{p}_k - Search Direction

α_k - Learning Rate



- روند به‌روزرسانی می‌باید به گونه‌ای باشد که

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$$

داریم:

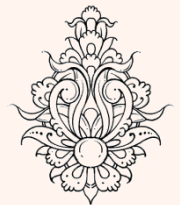
$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k$$

سری تیلور مرتبه‌ی ۱

- که در آن \mathbf{g}_k از رابطه‌ی زیر به دست می‌آید:

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k}$$

گرادیان



الگوریتم‌های بهینه‌سازی

- برای این $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$ بخش آخر رابطه‌ی زیر باید کوچک‌تر از صفر باشد.

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta \mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta \mathbf{x}_k$$

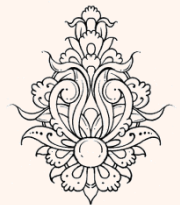
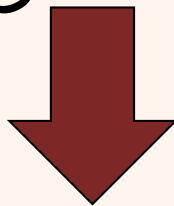
- پس داریم:

$$\mathbf{g}_k^T \Delta \mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0$$

- اگر α بین صفر و یک باشد، خواهیم داشت:

$$\mathbf{g}_k^T \mathbf{p}_k < 0$$

- به هر بردار \mathbf{p}_k که شرط بالا صدق کند، «descent direction» می‌گویند.



الگوریتم‌های بهینه‌سازی

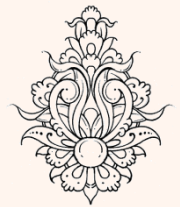
$$g_k^T p_k < 0$$

- در رابطه‌ی فوق می‌باید ضرب داخلی دو بردار گرادیان و بردار جهت نزولی (descent direction) منفی باشد، هر چه عبارت فوق منفی‌تر باشد، سریع‌تر به نقطه‌ی مزبور نزدیک می‌شویم.
- چگونه می‌توان به سریع‌ترین کاهش دست یافت؟

$$p_k = -g_k$$

- برای متد steepest decent داریم:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$$



• Steepest decent را برای مسالهی زیر اعمال

$$F(x) = x_1^2 + 25x_2^2$$

کنید:

• با در نظر گرفتن $x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$ خواهیم داشت:

$$\nabla F(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(x) \\ \frac{\partial}{\partial x_2} F(x) \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix} \rightarrow \nabla F(x) \Big|_{x=x_0} = \begin{bmatrix} 1 \\ 25 \end{bmatrix}$$



• با در نظر گرفتن $\alpha=0.01$ خواهیم داشت:

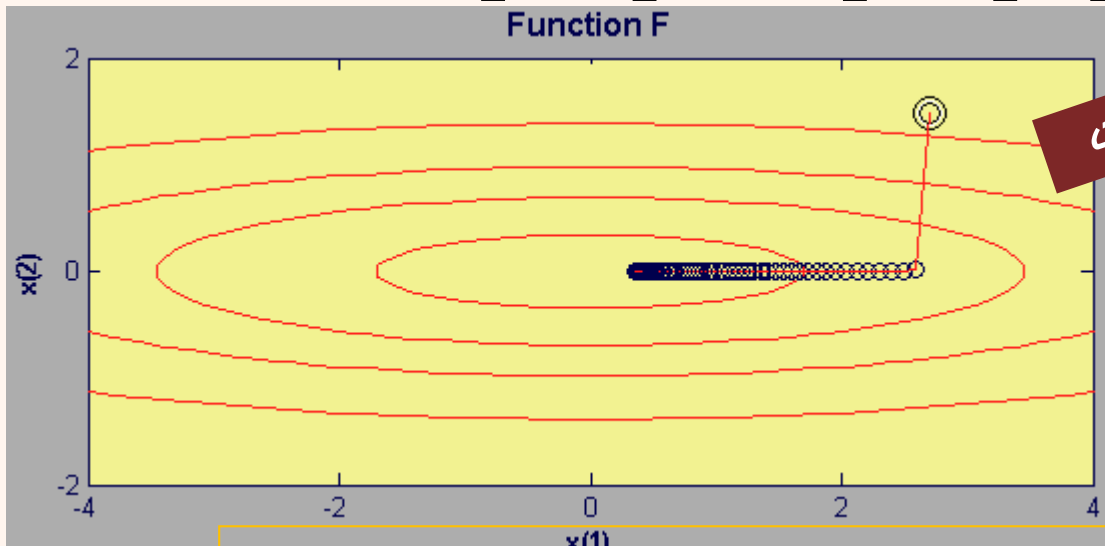
First iteration

$$x_1 = x_0 - \alpha g_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.01 \begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} 0.49 \\ 0.25 \end{bmatrix}$$

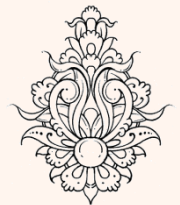
Second iteration

$$x_2 = x_1 - \alpha g_1 = \begin{bmatrix} 0.49 \\ 0.25 \end{bmatrix} - 0.01 \begin{bmatrix} 0.98 \\ 12.5 \end{bmatrix} = \begin{bmatrix} 0.4802 \\ 0.125 \end{bmatrix}$$

در صورت ادامه‌ی روند خواهیم داشت

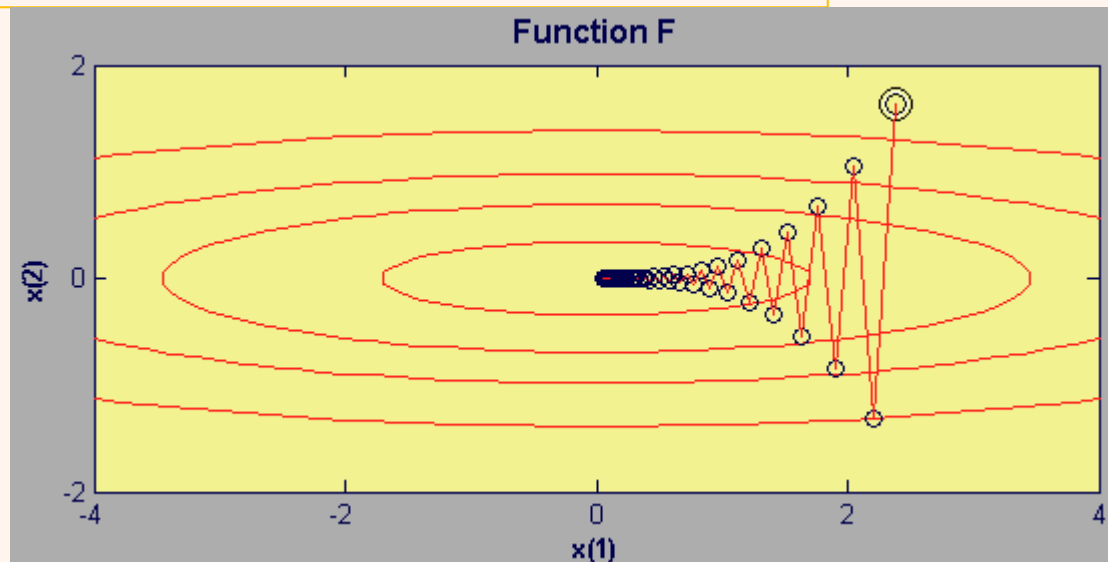


Trajectory for Steepest Descent with $\alpha = 0.01$

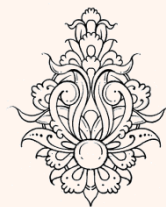


- اگر نرخ آموزش را بالا ببریم چیزی شبیه به شکل زیر خواهیم داشت:
- در این صورت نوسان بیشتری خواهیم داشت و ناپایداری بیشتری خواهد شد.

Trajectory for Steepest Descent with $\alpha = 0.035$



همواره جهت تخییرات بر مسیر عمود است
و این به دلیل استفاده از گرادیان است.



مثال

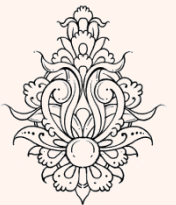
$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad \alpha = 0.1$$

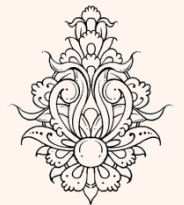
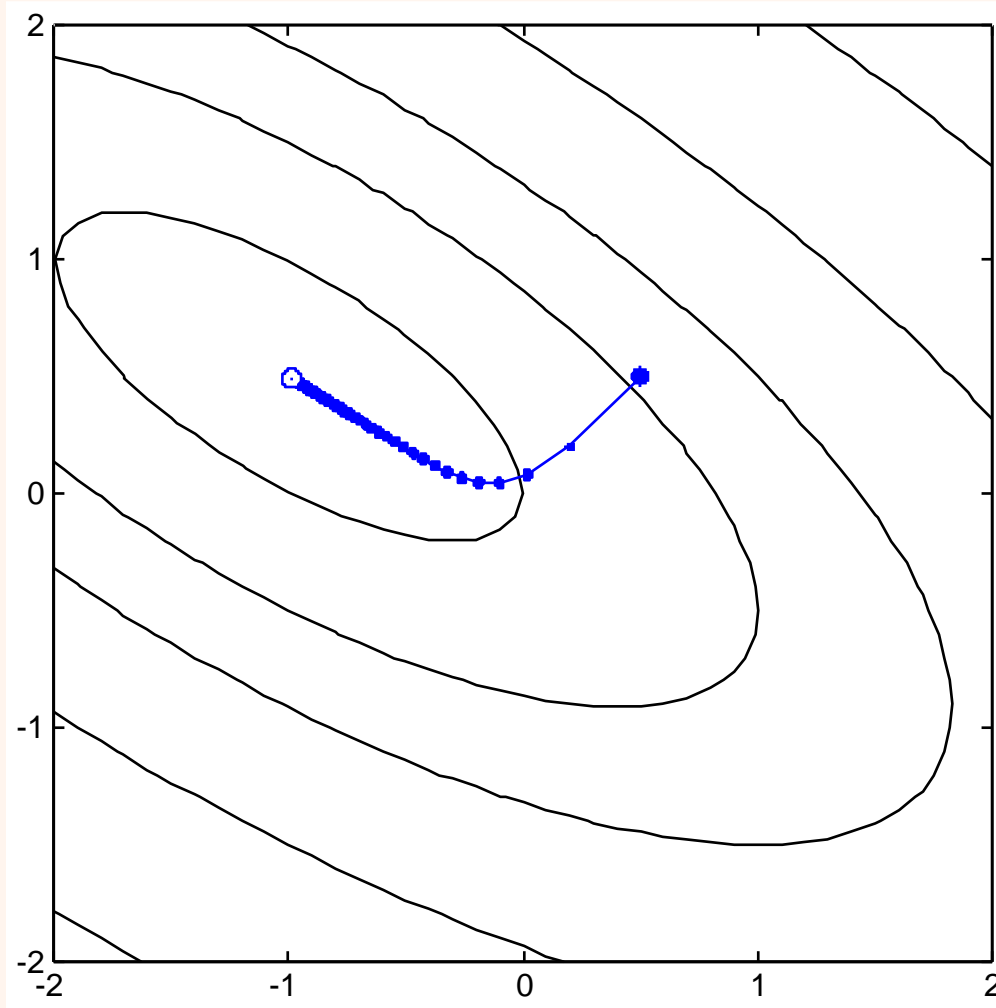
$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \quad \mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.1 \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} - 0.1 \begin{bmatrix} 1.8 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.08 \end{bmatrix}$$



نمودار



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

$$\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k = \mathbf{x}_k - \alpha (\mathbf{A} \mathbf{x}_k + \mathbf{d}) \longrightarrow \mathbf{x}_{k+1} = \underbrace{[\mathbf{I} - \alpha \mathbf{A}]}_{\text{linear dynamic system}} \mathbf{x}_k - \alpha \mathbf{d}$$

پایداری وابسته به مقادیر ویژهی این ماتریس است

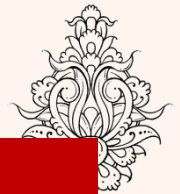
$(\lambda_i$ - eigenvalue of $\mathbf{A})$

$$[\mathbf{I} - \alpha \mathbf{A}] \mathbf{z}_i = \mathbf{z}_i - \alpha \mathbf{A} \mathbf{z}_i = \mathbf{z}_i - \alpha \lambda_i \mathbf{z}_i = \underbrace{(1 - \alpha \lambda_i)}_{\text{Eigenvalues of } [\mathbf{I} - \alpha \mathbf{A}]} \mathbf{z}_i$$

Eigenvalues of $[\mathbf{I} - \alpha \mathbf{A}]$.

با فرض آن که دارای که مینم مطلوب با

$$|(1 - \alpha \lambda_i)| < 1 \longrightarrow \alpha < \frac{2}{\lambda_i} \longrightarrow \alpha < \frac{2}{\lambda_{max}}$$



مثال

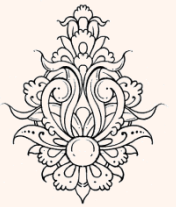
- با اعمال این مساله بر مثال قبلی برآئیه
بیشترین میزان نرخ آموزش مجاز را محاسبه کنید

$$F(x) = x_1^2 + 25x_2^2$$

- همان‌گونه که مشاهده می‌شود مثالی درجه دو
است پس Hessian Matrix به صورت زیر خواهد
بود:

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}$$

- پس برای مقادیر ویژه داریم:
 $\{(\lambda_1 = 2), \left(z_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)\}, \{(\lambda_2 = 50), \left(z_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right)\}$



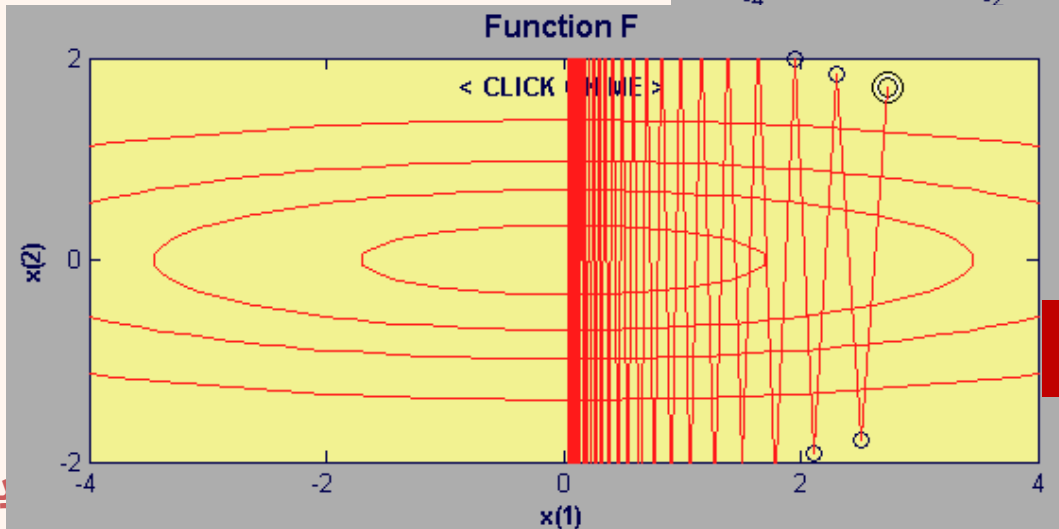
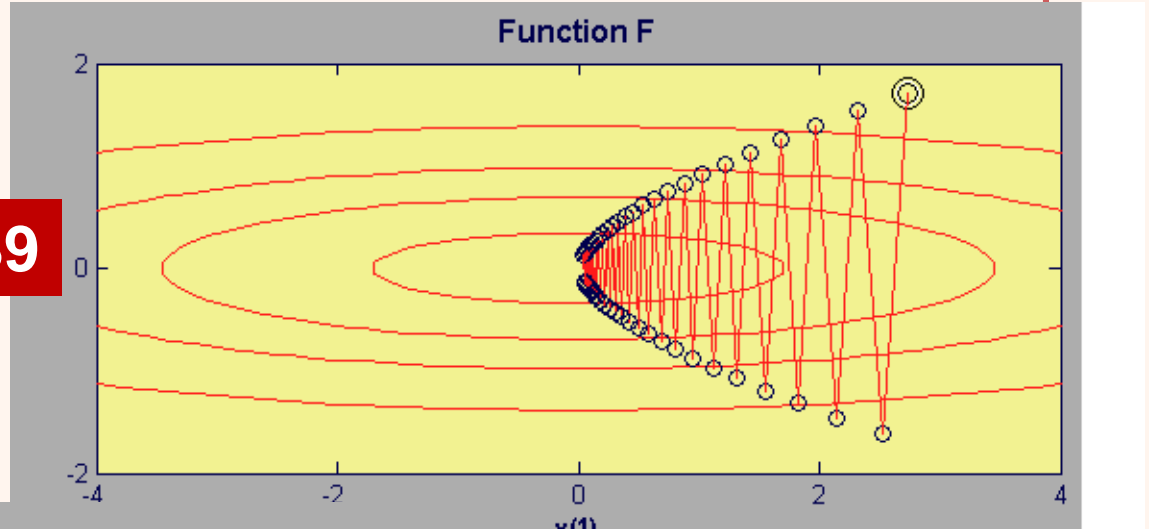
مثال-ادامه

• پس بیشترین میزان نرخ آموزش از رابطه‌ی زیر به

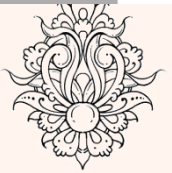
$$\alpha < \frac{2}{\lambda_{\max}} = \frac{2}{50} = 0.04$$

دست می‌آید:

$\alpha=0.039$



$\alpha=0.041$



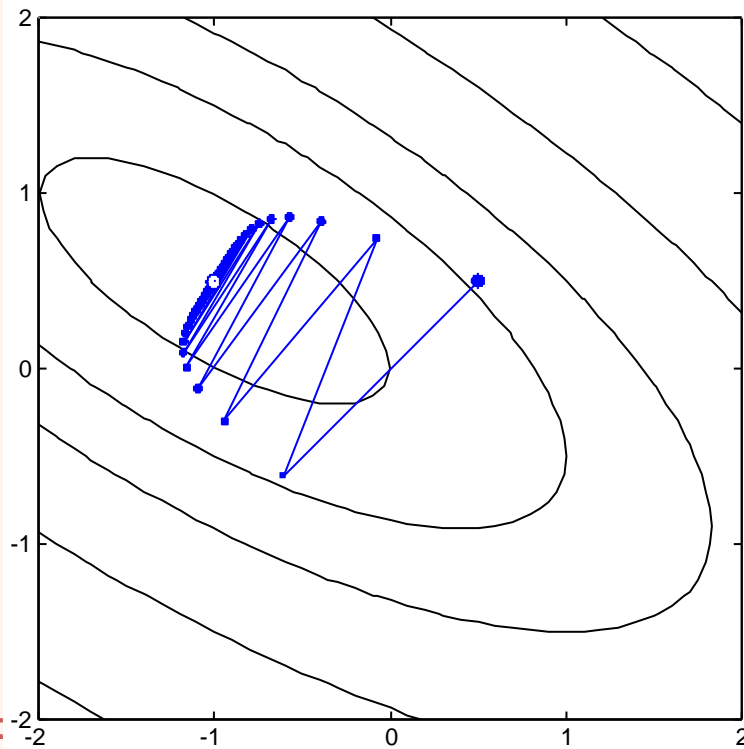
مثال

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

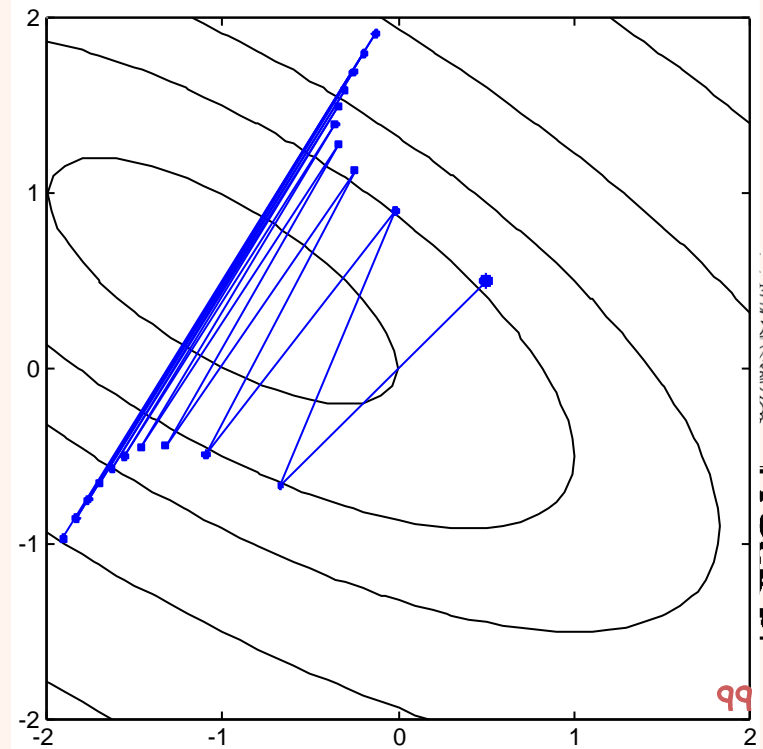
$$\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \quad \left\{ (\lambda_1 = 0.764), \left(\mathbf{z}_1 = \begin{bmatrix} 0.851 \\ -0.526 \end{bmatrix} \right) \right\}, \left\{ \lambda_2 = 5.24, \left(\mathbf{z}_2 = \begin{bmatrix} 0.526 \\ 0.851 \end{bmatrix} \right) \right\}$$

$$\alpha < \frac{2}{\lambda_{max}} = \frac{2}{5.24} = 0.38$$

$\alpha = 0.37$



$\alpha = 0.39$



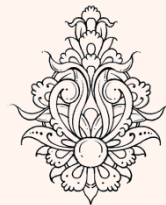
تنظیم نرخ یادگیری *Minimizing Along a Line*

- در مورد انتخاب نرخ آموزش به صورت افقی پیش از این صحبت شد.
- راه دیگر انتخاب نرخ آموزش به گونه‌ای است که عبارت زیر مینیمم شود:

$$F(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$$

- برای این منظور لازم است در راستای \mathbf{p}_k جستجویی صورت پذیرد.
- برای توابع درجه‌ی دوم می‌توان راه حلی تحلیلی ارائه نمود:

$$\frac{d}{d\alpha_k}(F(\mathbf{x}_k + \alpha_k \mathbf{p}_k)) = \nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k + \alpha_k \mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k$$



تنظیم نرخ یادگیری (ادامه...)

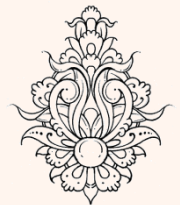
$$\frac{d}{d\alpha_k}(F(\mathbf{x}_k + \alpha_k \mathbf{p}_k)) = \nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k + \alpha_k \mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k$$

در نتیجه

$$\alpha_k = - \frac{\nabla F(\mathbf{x})^T \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k}{\mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k} = - \frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}$$

که در آن

$$\mathbf{A}_k \equiv \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_k}$$



مثال

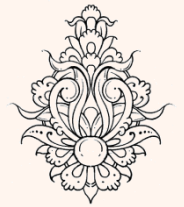
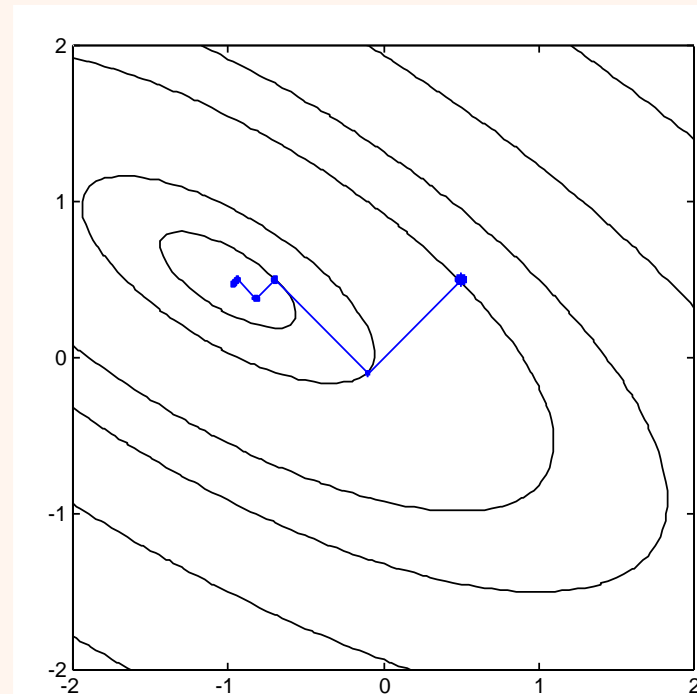
$$F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x} + [1 \ 0] \mathbf{x} \quad \mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix}$$

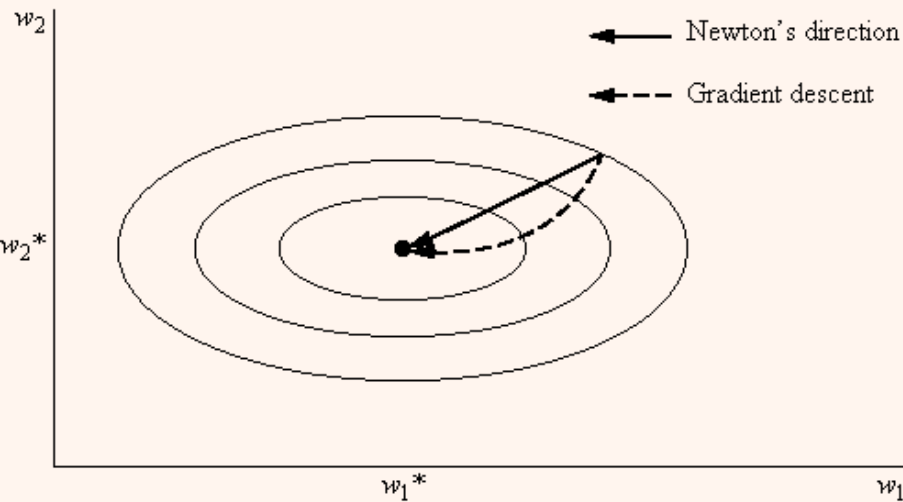
$$\mathbf{p}_0 = -\mathbf{g}_0 = -\nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$$

$$\alpha_0 = -\frac{\begin{bmatrix} 3 & 3 \end{bmatrix} \begin{bmatrix} -3 \\ -3 \end{bmatrix}}{\begin{bmatrix} -3 & -3 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} -3 \\ -3 \end{bmatrix}} = 0.2$$

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.2 \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} -0.1 \\ -0.1 \end{bmatrix}$$



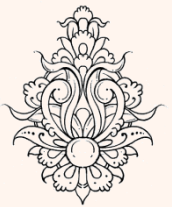
روش نیوتن



در ریاضیات از روش نیوتن جهت یافتن ریشه عبارت ریاضی به وسیله الگوریتمی تکراری استفاده می‌شود.

در مسأله‌ی بهینه‌سازی از این الگوریتم برای یافتن نقاط مانا (Stationary Point) به گونه‌ای که مشتق را صفر کند، استفاده می‌شود.

در این حالت با شروع از نقطه‌ی x_0 به دنبال نقطه‌ی x^* هستیم به گونه‌ای که $f'(x^*)=0$



روش نیوتن

- در بسط سری تیلور با استفاده از روابط زیر خواهیم داشت:

$$\Delta x = x_{k+1} - x_k$$

$$f_T(x_k + \Delta x) \approx f(x_k) + f'(x_k)\Delta x + \frac{1}{2}f''(x_k)\Delta x^2$$

- با مشتق گرفتن از رابطه‌ی فوق و قرار دادن آن برابر با صفر داریم:

$$f'(x_k) + f''(x_k)\Delta x = 0 \quad \rightarrow \quad \Delta x = -\frac{f'(x_k)}{f''(x_k)}$$

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \quad k = 0, 1, \dots$$

روش نیوتن

$$f_T(x_k + \Delta x) = f(x_k) + f'(x_k)\Delta x + \frac{1}{2}f''(x_k)\Delta x^2$$

• در نمایش برداری داریم:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta \mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta \mathbf{x}_k + \frac{1}{2} \Delta \mathbf{x}_k^T \mathbf{A}_k \Delta \mathbf{x}_k$$

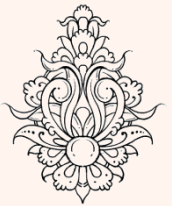
سری تیلور مرتبه ۲

با مشتق گرفتن از این تقریب و قرار دادن رابطه برابر با صفر، نقطه‌ی
مانا را بیابیم:

$$\mathbf{g}_k + \mathbf{A}_k \Delta \mathbf{x}_k = \mathbf{0}$$

$$\Delta \mathbf{x}_k = -\mathbf{A}_k^{-1} \mathbf{g}_k$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k$$

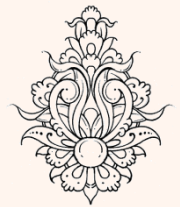
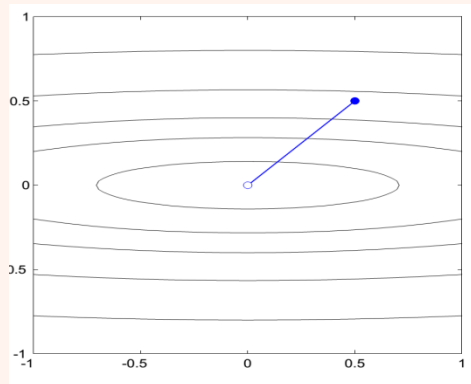


$$F(x) = x_1^2 + 25x_2^2$$

• با در نظر گرفتن $x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$ خواهیم داشت:

$$\nabla F(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(x) \\ \frac{\partial}{\partial x_2} F(x) \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix} \quad \nabla^2 F(x) = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = 0$$



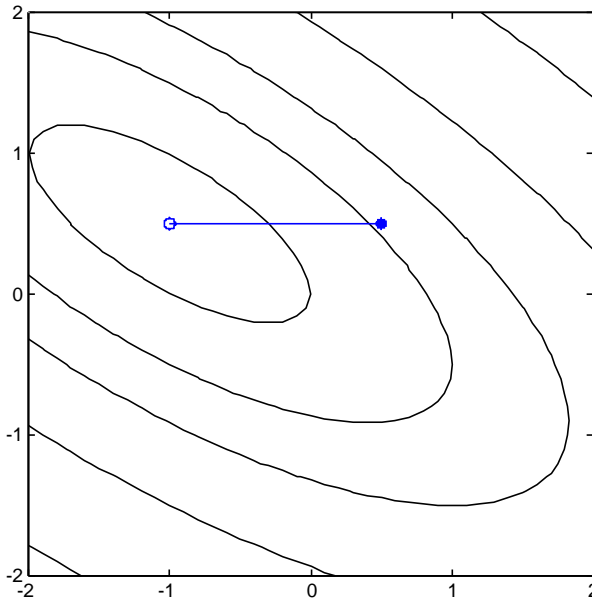
مثال ۲

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

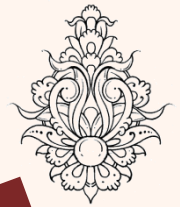
$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \quad \mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\mathbf{x}_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 1 & -0.5 \\ -0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$$

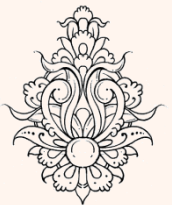


Quadratic Termination



مشکلات روش نیوتن

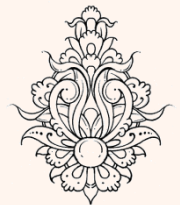
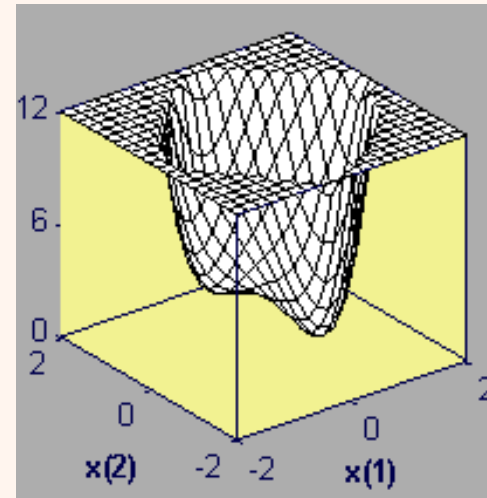
- این روش نیاز به محاسبه ماتریس **معکوس** Hessian دارد.
- ممکن است در شرایطی این ماتریس **معکوس پذیر** نباشد.
- هنگامی که تابع خطا (کارایی)، درجهی دو نباشد، تضمینی مبنی بر **همگرایی** وجود نخواهد داشت.



همگرایی

- در صورتی که تابع هزینه (کارایی) درجه‌ی دوم نباشد، با استفاده از روش نیوتن نمی‌توان همگرایی روش را تضمین کرد.
- در این صورت، همگرایی وابسته به تابع هزینه و حتی حدس اولیه است.

$$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$$

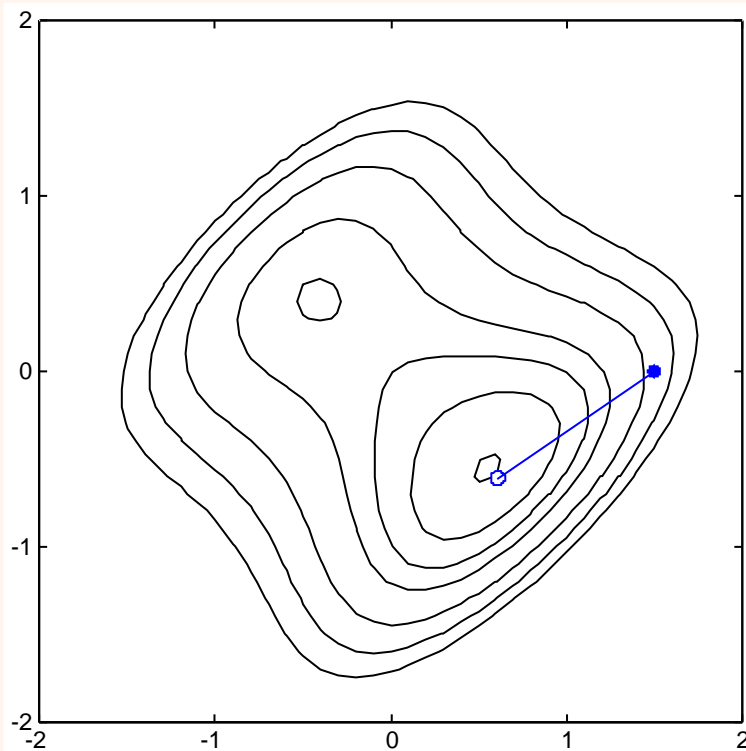


شرایط اولیه متفاوت

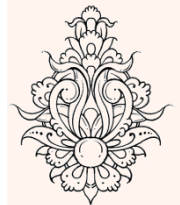
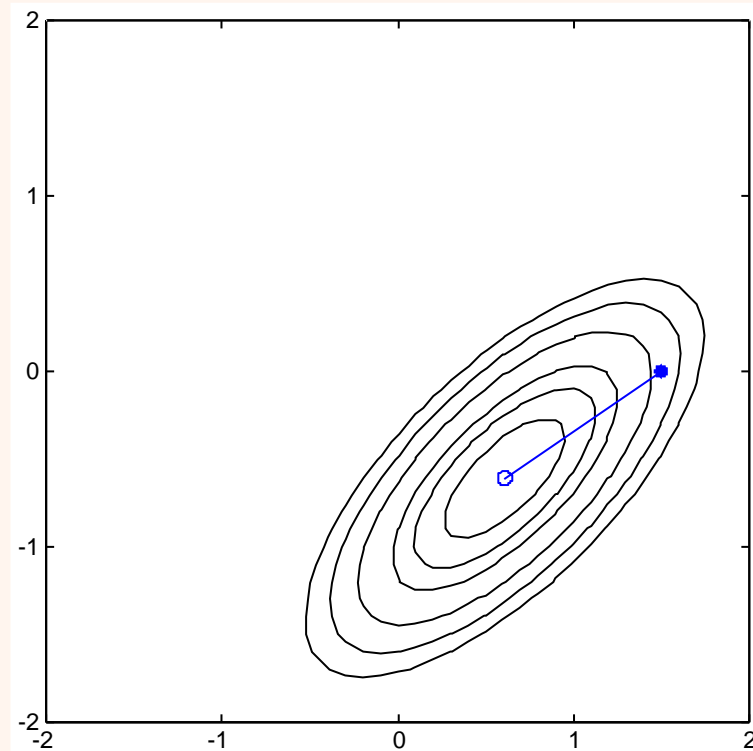
$$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$$

Stationary Points: $\mathbf{x}^1 = \begin{bmatrix} -0.42 \\ 0.42 \end{bmatrix}$ $\mathbf{x}^2 = \begin{bmatrix} -0.13 \\ 0.13 \end{bmatrix}$ $\mathbf{x}^3 = \begin{bmatrix} 0.55 \\ -0.55 \end{bmatrix}$

$F(\mathbf{x})$

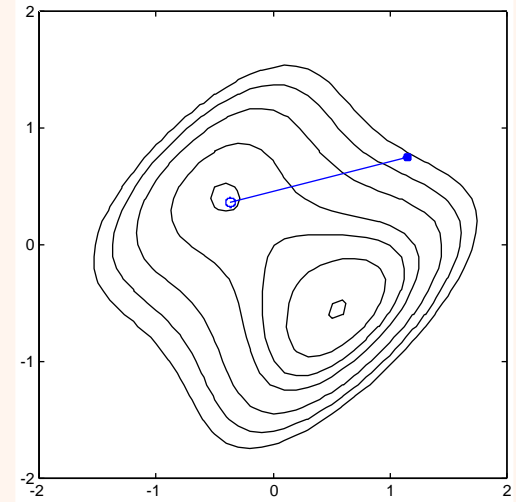
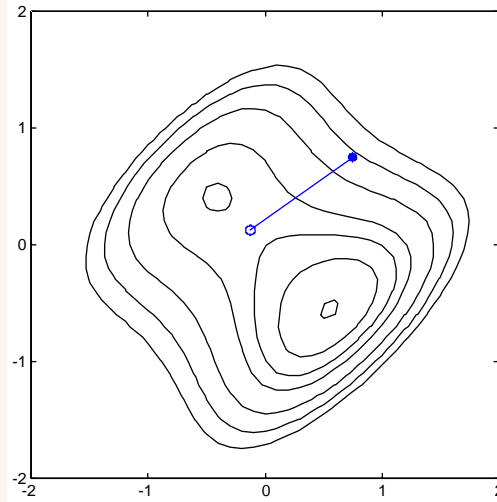
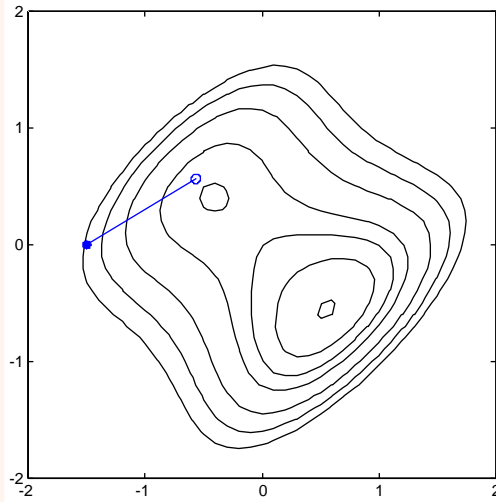


$F_2(\mathbf{x})$

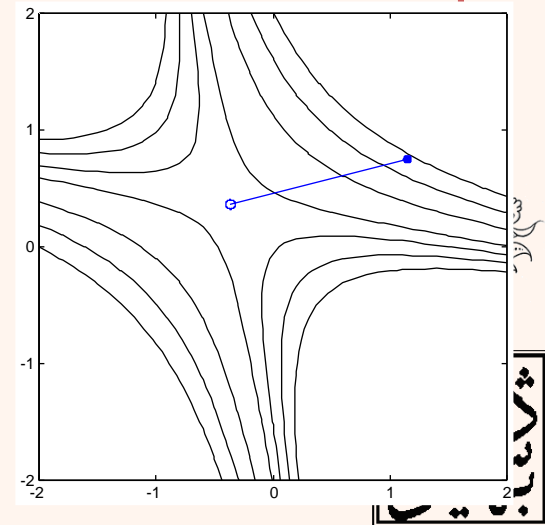
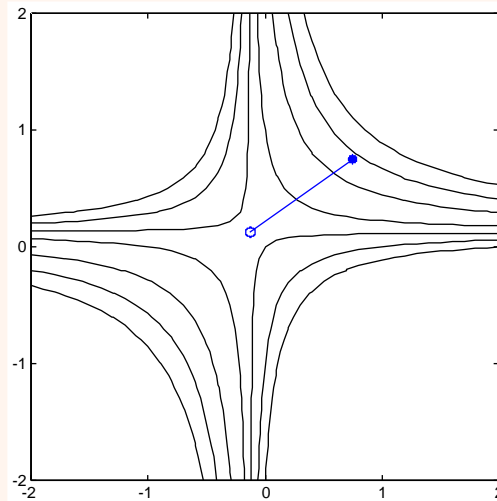
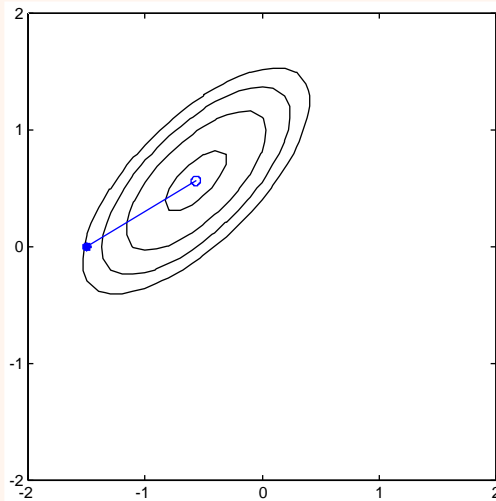


شرایط اولیه متفاوت

$F(\mathbf{x})$



$F_2(\mathbf{x})$



$X = \{w \text{ and } b \text{ of layer 1, } w \text{ and } b \text{ of layer 2, } \dots\}$

• در روش عادی B.P داشتهیم:

$$X_{k+1} = X_k - \mu \nabla F(X_k)$$

• در روش نیوتن داریم:

$$X_{k+1} = X_k - A_k^{-1} \nabla F(X_k)$$

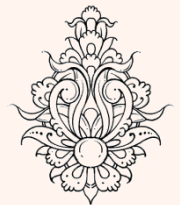
تابع معیار خطا

$$F(X) = \sum_{i=1}^M e_i^2(X)$$

• که در آن

$$g_k = \nabla F(X) \Big|_{X=X_k}$$

$$A_k = \nabla^2 F(X) \Big|_{X=X_k}$$



$X = \{w \text{ and } b \text{ of layer 1, } w \text{ and } b \text{ of layer 2, } \dots\}$

تابع معیار خطا

$$F(X) = \sum_{i=1}^M e_i^2(X) = E^T(X)E(X)$$

$$E(X) = [e_1 \ e_2 \ \dots \ e_M]^T$$

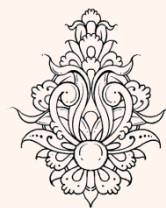
$$[\nabla F(X)]_{x_j} = \frac{\partial F(X)}{\partial x_j} = 2 \sum_{i=1}^M e_i(X) \cdot \frac{\partial e_i(X)}{\partial x_j}$$

$$\nabla F(X) = 2J^T(X)E(X)$$

ماتریس ژاکوبی

$J(X) =$

$$\begin{bmatrix} \frac{\partial e_1(X)}{\partial x_1} & \dots & \frac{\partial e_1(X)}{\partial x_p} \\ \vdots & & \vdots \\ \frac{\partial e_M(X)}{\partial x_1} & \dots & \frac{\partial e_M(X)}{\partial x_p} \end{bmatrix}$$



محاسبه مشتق دوم

$$\frac{\partial F(X)}{\partial x_j} = 2 \sum_{i=1}^M e_i(X) \cdot \frac{\partial e_i(X)}{\partial x_j}$$

$$\left[\nabla^2 F(X) \right]_{x_k, j} = \frac{\partial^2 F(X)}{\partial x_k \partial x_j} = \frac{\partial}{\partial x_k} \left[\frac{\partial F(X)}{\partial x_j} \right]$$

$$= 2 \sum_{i=1}^M \left[\frac{\partial e_i(X)}{\partial x_k} \cdot \frac{\partial e_i(X)}{\partial x_j} + e_i(X) \cdot \frac{\partial^2 e_i(X)}{\partial x_k \partial x_j} \right]$$

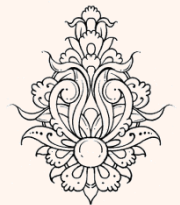
$$\nabla^2 F(X) = 2J^T(X)J(X) + 2S(X)$$

$$S(X) = \sum_{i=1}^M e_i(X) \cdot \frac{\partial^2 e_i(X)}{\partial x_k \partial x_j}$$

$S(X)$ معمولاً بسیار کوچک است به همین دلیل در محاسبات می‌توان از آن صرف‌نظر کرد

$$\nabla^2 F(X) \approx 2J^T(X)J(X)$$

در این صورت متد را **Gauss-Newton** گویند



• در روش نیوتن داشتیم

$$X_{k+1} = X_k - A_k^{-1} g_k$$

$$g_k = \nabla F(X) \Big|_{X=X_k} \longrightarrow \nabla F(X) = 2J^T(x)E(X)$$

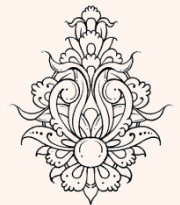
$$A_k = \nabla^2 F(X) \Big|_{X=X_k} \longrightarrow \nabla^2 F(X) = 2J^T(X)J(X)$$

برای مناسبی رابطه‌ی مذکور تنها از مشتق اول استفاده می‌شود

Gauss–Newton

$$X_{k+1} = X_k - \left[2J^T(X) \cdot J(X) \right]^{-1} \cdot 2J^T(x)E(X)$$

آیا این ماتریس همواره معکوس پذیر است؟



Levenberg–Marquardt

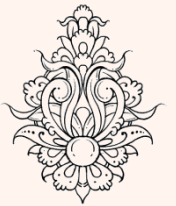
$$X_{k+1} = X_k - \underbrace{\left[2J^T(X_k) J(X_k) \right]^{-1}}_{H_k} \cdot 2J^T(x) E(X)$$

- در این روش به جای استفاده از H_k از ماتریس G_k استفاده می‌شود.

$$G_k = H_k + \mu_k I$$

بسیار کوچک

- اگر $\mu = 0$ باشد روش گاوس-نیوتن است.



Levenberg–Marquardt

مقدار ویژه

یادآوری

$$Aq_i = \lambda_i q_i$$

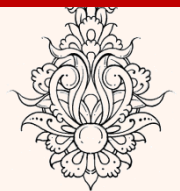
بردار ویژه

$$\begin{aligned} Gq_i &= [H + \mu I]q_i \\ &= Hq_i + \mu q_i \\ &= \lambda_i q_i + \mu q_i \\ &= (\lambda_i + \mu)q_i \end{aligned}$$

فرض می‌کنیم H دارای مقادیر ویژه λ_i و بردارهای ویژه q_i باشد

G دارای مقادیر ویژه $\lambda_i + \mu$ و بردارهای ویژه q_i خواهد بود.

$$Gq_i = (\lambda_i + \mu)q_i$$



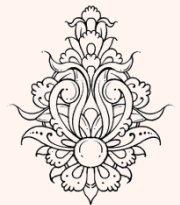
Levenberg–Marquardt

G دارای مقادیر ویژه $\mu + \lambda_i$ و بردارهای ویژه q_i است

$$Gq_i = (\lambda_i + \mu)q_i$$

- برای محکوس‌پذیری کافی است مقادیر ویژه ماتریس مثبت باشد.
- می‌توان آنقدر μ را تخییر داد تا مقادیر ویژه مثبت گردد.
- در روش گاوس نیوتن این مقدار ثابت و غیرقابل تخییر بود.

$$X_{k+1} = X_k - \left[2J^T(X_k) \cdot J(X_k) + \mu_k I \right]^{-1} \cdot 2J^T(X_k) E(X_k)$$



Levenberg–Marquardt

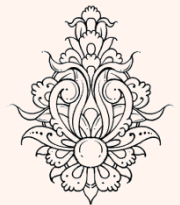
$$X_{k+1} = X_k - \left[2J^T(X_k) \cdot J(X_k) + \mu_k I \right]^{-1} \cdot 2J^T(X_k) E(X_k)$$

• می‌توان نشان داد، با افزایش میزان μ رابطه همانند زیر می‌شود:

$$X_{k+1} = X_k - \frac{1}{\mu_k} J^T(X_k) E(X_k) = X_k - \frac{1}{2\mu_k} \nabla F(X_k)$$

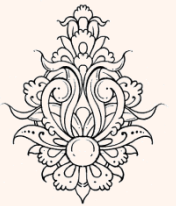
• معمولاً الگوریتم را با μ کوچک در حدود 0.01 شروع کرده در صورت کاهش خطا (موفقیت) با ضریب θ (در حدود ۱۰) کاهش می‌دهند.

• در صورت عدم موفقیت μ را با ضریب θ افزایش می‌دهند.



این روش برای شبکه‌های کوچک تعداد تکرارهای بسیار کمتری از BP دارد اما حجم محاسباتی آن بالا و زمان‌گیر است

- در روش نیوتن احتیاج به محاسبه‌ی ماتریس Hessian داریم؛ در واقع محاسبه و ذخیره‌سازی مشتق دومی لازم است.
- هدف یافتن روشی است که **بدون نیاز به محاسبه‌ی مشتق دومی** همگرایی را افزایش دهد.
- از طرفی استفاده از steepest descent باعث حرکت زیزایی به سمت مینیمم می‌شود.
- در صورتی که در راستای بردارهای ویژه‌ی ماتریس Hessian حرکت کنیم، می‌توان انتظار داشت که سرعت همگرایی افزایش یابد.



Conjugate gradient

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

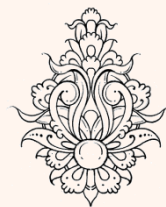
دو بردار نسبت به ماتریس \mathbf{A} (که positive definite است)، conjugate نامیده می‌شوند:

$$\mathbf{p}_k^T \mathbf{A} \mathbf{p}_j = 0 \quad k \neq j$$

ثابت می‌شود در صورتی که جستجو در راستای مجموعه بردارهای conjugate باشد، می‌توان طی یک دور جستجو در راستای این بردارها به مینیمم مطلق رسید (برای توابع درجه‌ی دو).
یک مجموعه از این بردارها، بردارهای ویژه‌ی ماتریس است.

$$\mathbf{z}_k^T \mathbf{A} \mathbf{z}_j = \lambda_j \mathbf{z}_k^T \mathbf{z}_j = 0 \quad k \neq j$$

در صورتی که ماتریس متقارن باشد، بردارهای ویژه‌ی آن متعامد است.



Conjugate gradient

$$\nabla F(\mathbf{x}) = \mathbf{Ax} + \mathbf{d}$$

$$\nabla^2 F(\mathbf{x}) = \mathbf{A}$$

تغییرات گرادیان در گام k -ام

$$\Delta \mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k = (\mathbf{Ax}_{k+1} + \mathbf{d}) - (\mathbf{Ax}_k + \mathbf{d}) = \mathbf{A}\Delta \mathbf{x}_k$$

که

$$\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$$

$$\alpha_k \mathbf{p}_k^T \mathbf{A} \mathbf{p}_j = \Delta \mathbf{x}_k^T \mathbf{A} \mathbf{p}_j = \Delta \mathbf{g}_k^T \mathbf{p}_j = 0 \quad k \neq j$$

بدون نیاز به محاسبه بردار Hessian بردار conjugate مناسبه شد.
توجه داشته باشید که به یک مجموعه بردار conjugate نیاز است، در
واقع در گام k -ام به جهتی نیاز است که بر تمام جهتهای زیر عمود
باشد:

Conjugate gradient

- مرحله‌ی اول مشابه steepest decent است.

F تابع معیار فضا است

$$g_0 = \nabla F \Big|_{x=x(0)}$$

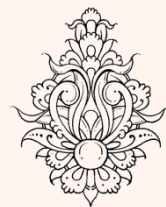
- Direction را به گونه‌ای انتخاب می‌کنیم که بر خلاف مشتق باشد.

$$p_0 = -g_0$$

- در هر iteration بردار p را به گونه‌ای محاسبه می‌کنیم که عمود بر تخییرات گرادیان واقع شود.

$$p_k = -g_k + \beta_k p_{k-1}$$

اثر گرادیان‌های قبلی



Gram-Schmidt Orthogonalization

Conjugate gradient

$$p_k = -g_k + \beta_k p_{k-1}$$

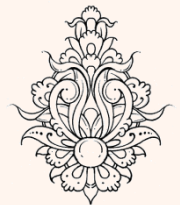
$$p_{k-1} \mathbf{A} p_k = -p_{k-1} \mathbf{A} g_k + \beta_k p_{k-1} \mathbf{A} p_{k-1}$$

$$\beta_k p_{k-1} \mathbf{A} p_{k-1} = p_{k-1} \mathbf{A} g_k$$

$$\beta_k = \frac{p_{k-1} \mathbf{A} g_k}{p_{k-1} \mathbf{A} p_{k-1}}$$

با توجه به نیاز به ماتریس Hessian از این شیوه
نمی‌توان استفاده کرد

صفر



Conjugate gradient

- ضریب β_k می‌تواند از یکی از روش‌های زیر محاسبه گردد:

$$\Delta \mathbf{g}_{k-1}^T = (\mathbf{g}_k - \mathbf{g}_{k-1})^T$$

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\Delta \mathbf{g}_{k-1}^T \mathbf{p}_{k-1}}$$

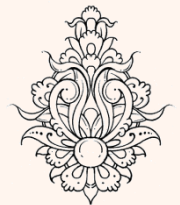
Hestenes and Steifel

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$$

Fletcher and Reeves

Polak and Ribiere

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$$



Conjugate gradient

• الگوریتم conjugate gradient به صورت خلاصه:

– جهت جستجوی اولیه را به گونه‌ای که بر جهت مشتق عمود

باشد در نظر می‌گیریم
 $p_0 = -g_0$

– مقدار α را که همان نرخ یادگیری است محاسبه می‌نماییم تا

رابطه‌ی زیر به دست آید، با توجه به این که این شیوه به نرخ

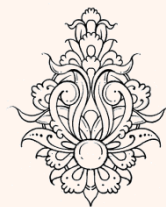
آموزش حساس است و نرخ آموزش نیز به ماتریس Hessian

وابسته است، برای تخمین این میزان از روش‌های تکرارشونده

برای محاسبه‌ی نرخ آموزش استفاده می‌شود.

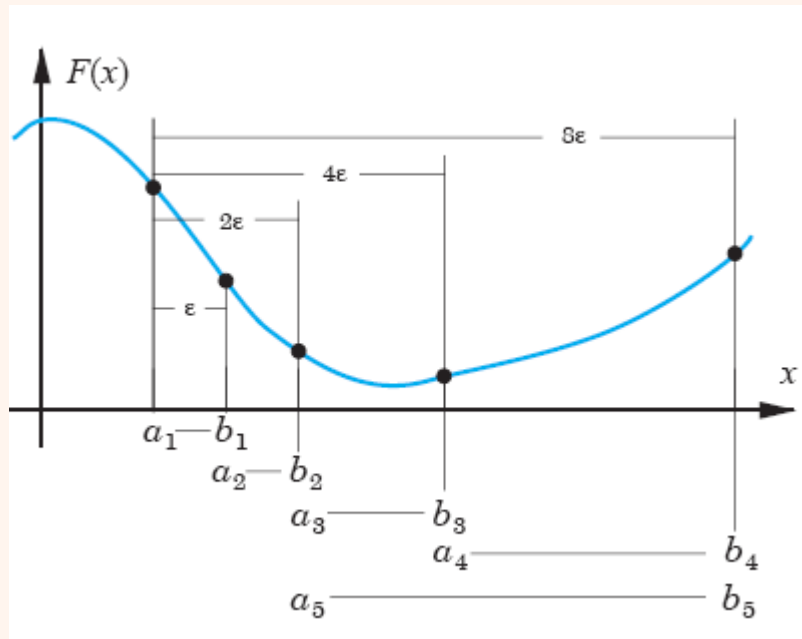
(For quadratic functions.)

$$x_{k+1} = x_k + \alpha_k p_k \quad \alpha_k = - \frac{\nabla F(\mathbf{x})^T \big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k}{\mathbf{p}_k^T \nabla^2 F(\mathbf{x}) \big|_{\mathbf{x} = \mathbf{x}_k} \mathbf{p}_k} = - \frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}$$

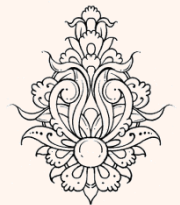


• تخمین نرخ آموزش در دو مرحله صورت می‌گیرد:

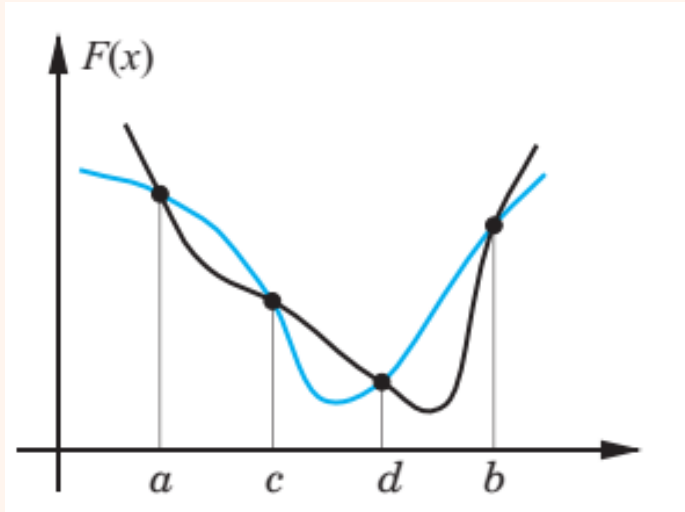
- Interval location
- Interval reduction



Interval location



Conjugate gradient

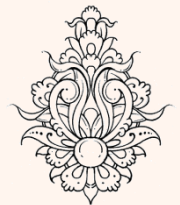


Interval reduction

- با محاسبه β مقدار زیر را که همان جهت بهینه‌ی جستجو است محاسبه می‌شود.

$$p_k = -g_k + \beta_k p_{k-1}$$

- اگر الگوریتم به همگرایی نرسید از گام دو دوباره شروع می‌کنیم.

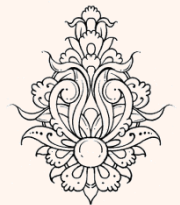


مثال

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x} + [1 \ 0] \mathbf{x} \quad \mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \quad \mathbf{p}_0 = -\mathbf{g}_0 = -\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$$

$$\alpha_0 = -\frac{[3 \ 3] \begin{bmatrix} -3 \\ -3 \end{bmatrix}}{[-3 \ -3] \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} -3 \\ -3 \end{bmatrix}} = 0.2 \quad \mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.2 \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} -0.1 \\ -0.1 \end{bmatrix}$$



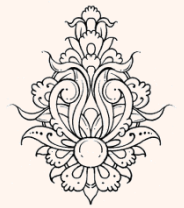
مثال-ادامہ

$$\mathbf{g}_1 = \nabla F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_1} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} -0.1 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.6 \end{bmatrix}$$

$$\beta_1 = \frac{\mathbf{g}_1^T \mathbf{g}_1}{\mathbf{g}_0^T \mathbf{g}_0} = \frac{\begin{bmatrix} 0.6 & -0.6 \end{bmatrix} \begin{bmatrix} 0.6 \\ -0.6 \end{bmatrix}}{\begin{bmatrix} 3 & 3 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix}} = \frac{0.72}{18} = 0.04$$

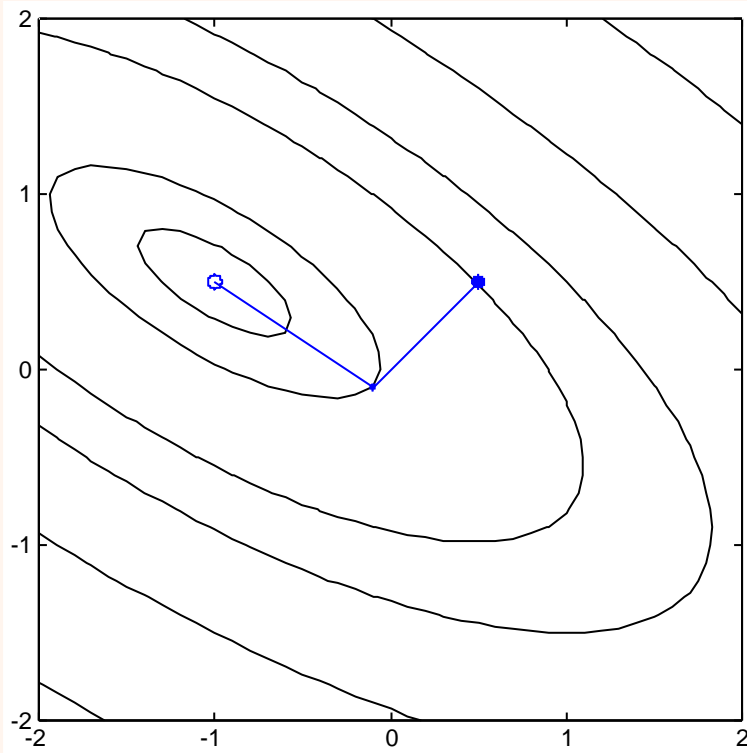
$$\mathbf{p}_1 = -\mathbf{g}_1 + \beta_1 \mathbf{p}_0 = \begin{bmatrix} -0.6 \\ 0.6 \end{bmatrix} + 0.04 \begin{bmatrix} -3 \\ -3 \end{bmatrix} = \begin{bmatrix} -0.72 \\ 0.48 \end{bmatrix}$$

$$\alpha_1 = -\frac{\begin{bmatrix} 0.6 & -0.6 \end{bmatrix} \begin{bmatrix} -0.72 \\ 0.48 \end{bmatrix}}{\begin{bmatrix} -0.72 & 0.48 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} -0.72 \\ 0.48 \end{bmatrix}} = -\frac{-0.72}{0.576} = 1.25$$



$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1 = \begin{bmatrix} -0.1 \\ -0.1 \end{bmatrix} + 1.25 \begin{bmatrix} -0.72 \\ 0.48 \end{bmatrix} = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$$

Conjugate Gradient



Steepest Descent

