

## مبانی هوش محاسباتی

### فصل دو - شبکه عصبی مصنوعی چندلایه



# فهرست مطالب

- معرفی شبکه‌های چندلایه
- الگوریتم آموزش شبکه‌های چندلایه
- نحوه استخراج قانون پسانتشار خطا
- نکات تکمیلی درباره شبکه‌های چندلایه
- کاربردهای شبکه چندلایه



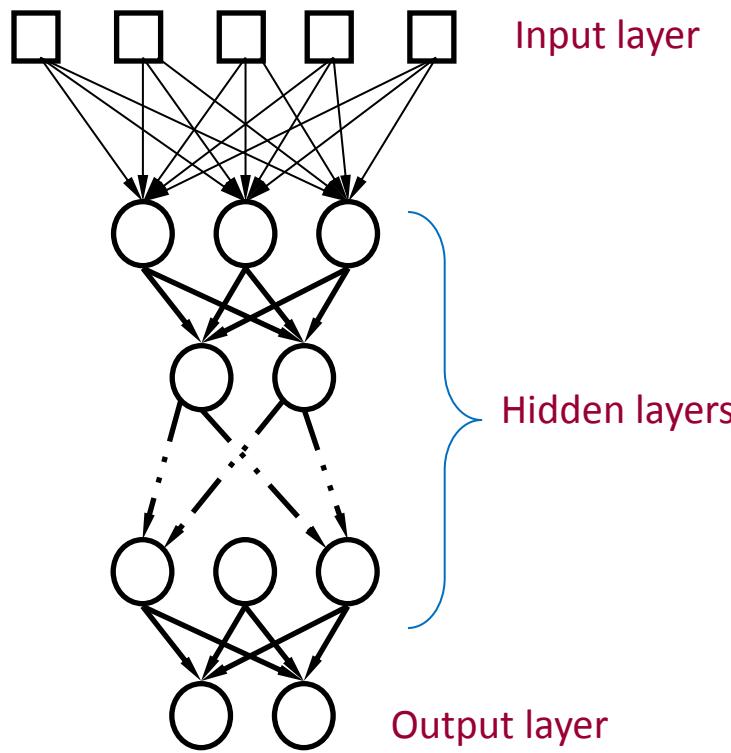
مبانی هوش محاسباتی

فصل دو - شبکه عصبی مصنوعی چندلایه

# شبکه چندلایه و ویژگی‌های آن



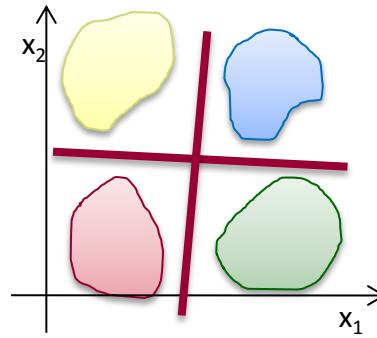
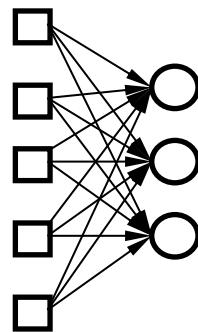
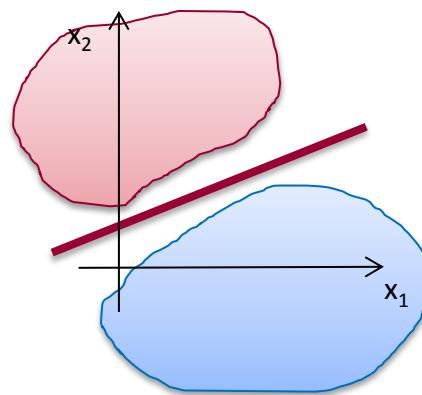
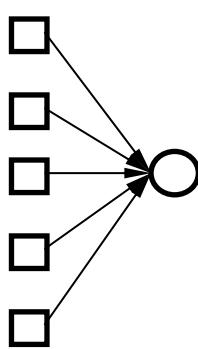
# شبکه عصبی چندلایه



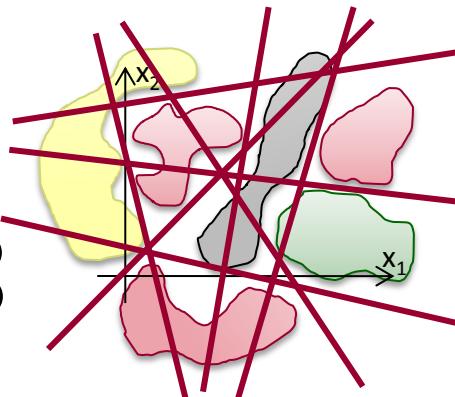
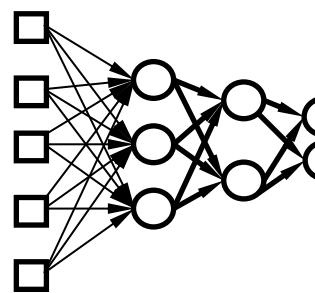
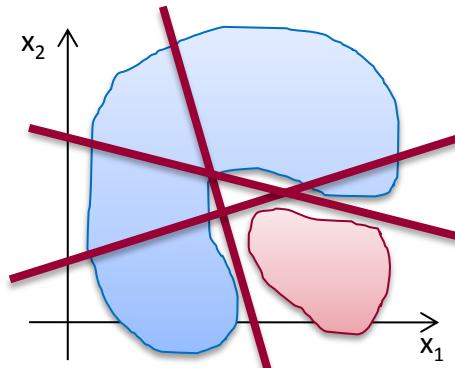
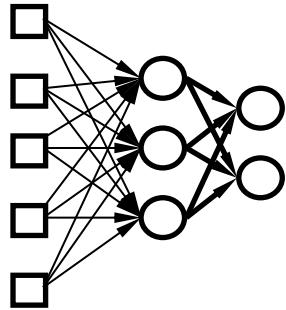
- شبکه عصبی چندلایه پیشخور (feed-forward multilayer NN)
  - تفکیک سلول‌های عصبی به زیرمجموعه‌هایی به نام لایه.
  - هر لایه، ورودی خود را فقط از لایه قبل دریافت می‌کند.
  - تحويل الگوهای ورودی به لایه اول.
  - دریافت خروجی‌ها از خروجی لایه نهایی.
- لایه میانی (مخفي): مجموعه سلول‌هایی که در لایه ورودی یا خروجی قرار ندارند.
- وظیفه سلول‌های هر لایه: جمع‌بندی نتایج سلول‌های لایه قبل.

# تأثیر لایه‌های مخفی

- بررسی تأثیر تعداد نرون‌ها در یک لایه
  - یک سلول عصبی
- مساله دو کلاسه تفکیک‌پذیر خطی
- شبکه یک لایه
- مسایل چند کلاسه تفکیک‌پذیر خطی



# تأثیر لایه‌های مخفی



- شبکه با یک لایه مخفی

- وظیفه لایه مخفی: تقسیم فضای ورودی به کلاس‌های تفکیک شده خطی

- وظیفه لایه خروجی (شبکه): تجمع کلاس‌های به وجود آمده از لایه میانی. و حل مسایل غیرتفکیک پذیر خطی.

- شبکه با دو لایه مخفی

- لایه مخفی اول: تشکیل خطها و کلاس‌های خطی

- لایه مخفی دوم: تشکیل چندضلعی‌ها و کلاس‌های غیرخطی.

- لایه خروجی (شبکه): مجموعه‌ای از کلاس‌های غیرخطی؛ مسایل خیلی پیچیده



# یک قضیه مهم

- سوال اساسی: آیا افزایش لایه‌ها و پیچیده کردن ساختار شبکه، الزاماً به معنای توانمند کردن آن است؟
- قضیه Hecht-Nielsen
  - فرض کنیم  $f$  یک تابع پیوسته به شکل  $f: I^n \rightarrow R^m$  که در آن  $I$  بازه بسته  $[1, 0]$  است. در این صورت، تابع  $f$  می‌تواند توسط یک شبکه پیش‌خور با  $n$  ورودی،  $1 + 2n$  سلول مخفی و  $m$  خروجی، مدل شود. (تقریب زده شود).
  - توابع فعالیت سلولهای مخفی، مستقل از  $f$  و صعودی هستند و توابع فعالیت سلولهای خروجی، به  $f$  وابسته هستند.
  - این قضیه بر اساس قضایای Kolmogorov و Sprecher بیان شده است.
- نتیجه مستقیم قضیه:
  - هر مساله دسته‌بندی، تقریب و تخمین، بهینه‌سازی، ... توسط یک شبکه عصبی با فقط یک لایه مخفی، قابل حل است!





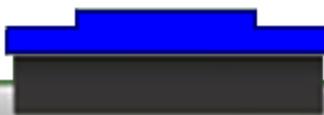
# میزان پیچیدگی شبکه

- یک شبکه با یک لایه مخفی (دولایه) و تعداد مناسب سلول مخفی، به صورت تئوری، قادر به حل تمامی مسایل دسته‌بندی است.
- در بعضی از مسایل، یک شبکه با دولاویه مخفی (سه لایه)، حجم محاسبات (تعداد نرون‌های لازم) را کاهش خواهد داد.
- مانند شکل اسلاید قبل، هنگامی که اعضای یک کلاس در گروه‌های پراکنده‌ای قرار گرفته‌اند، استفاده از یک لایه مخفی اضافی، به سرعت و کاهش حجم محاسبات کمک می‌کند. در صورت استفاده از فقط یک لایه مخفی، ممکن است مجبور شویم تعداد زیادی نرون در لایه مخفی قرار دهیم تا شبکه بتواند به درستی، دسته‌بندی را انجام دهد.
- افزایش پارامترها و پیچیدگی زیاد مدل؛ کمبود داده آموزشی



# نام‌گذاری گوناگون

- شبکه‌های عصبی چندلایه پیش‌خور  
(Feed-forward multi-layer - FF)
- شبکه پرسپترون چندلایه  
(Multi-Layer Perceptron - MLP)
- شبکه عصبی با پساننتشار خطا  
(Back-Propagation Net - BP)



مبانی هوش محاسباتی

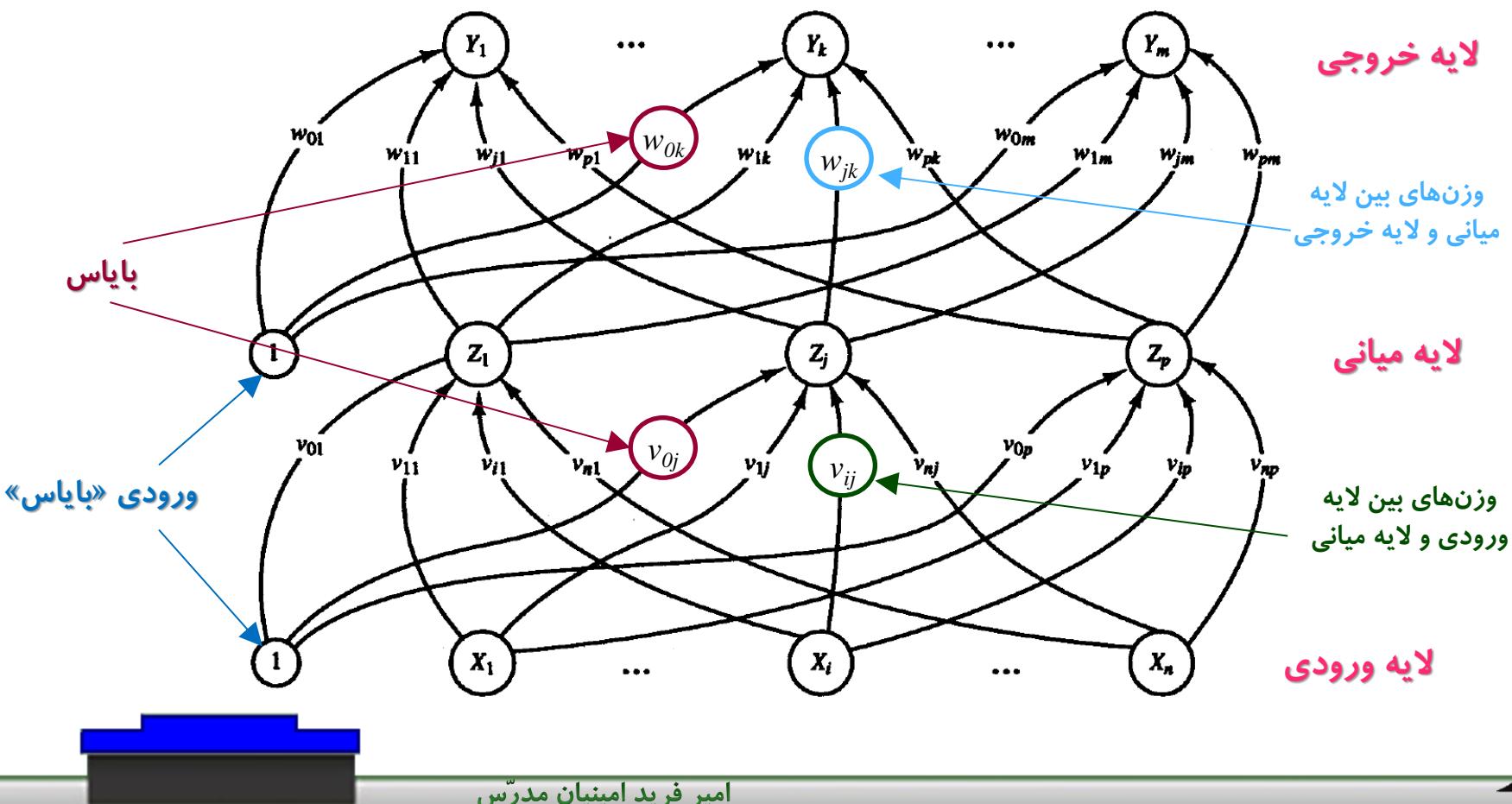
فصل دو - شبکه عصبی مصنوعی چندلایه

# آموزش شبکه‌های عصبی چندلایه



# ساختار شبکه چندلایه

- ساختار شبکه چندلایه و نحوه نام‌گذاری پارامترها



# معرفی قانون آموزش

- قانون آموزش پس انتشار خطا (back propagation of error) یا به اختصار، پس انتشار (back propagation) (Generalized Delta-Rule)
  - نام دیگر؛ قانون دلتای تعمیم یافته (Generalized Delta-Rule)
  - معرفی توسط Bryson & Ho در سال ۱۹۶۹
  - انتشار توسط Rumelhart & Hinton & Williams در سال ۱۹۸۶.
- ایده: روش گرادیان نزولی (Gradient Descent)، با هدف کاهش مجموع مجدول خطای شبکه توسط تغییرات مناسب پارامترها (وزن‌های) آن.
  - یادآوری: تعریف بردار گرادیان (بردار مشتقات نسبی در فضای چند بعدی):

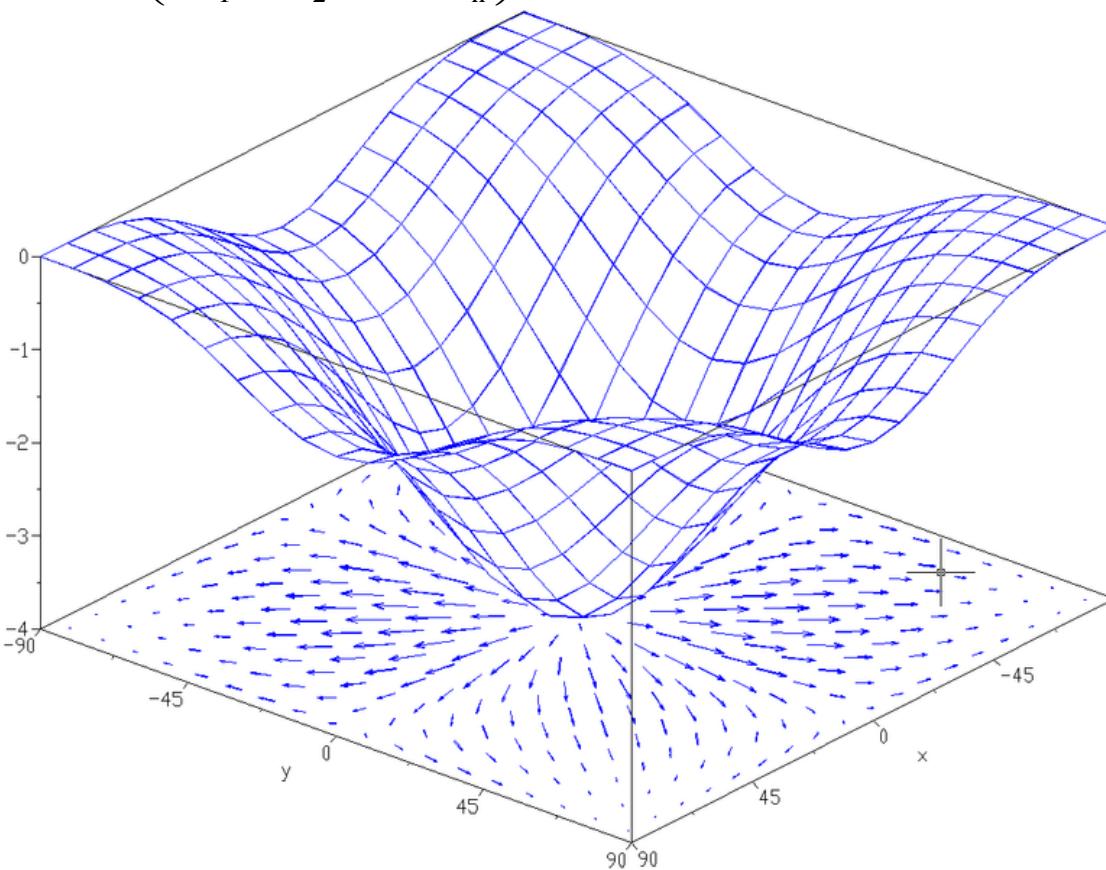
$$f(x_1, x_2, \dots, x_n) \quad \nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$



# روش کاهش گرادیان (گرادیان نزولی)

$$f(x_1, x_2, \dots, x_n)$$

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$



- بردار گرادیان

- تعبیر هندسی: نشان‌دهنده جهت و اندازه بیشترین تغییرات (افزایش) یک تابع چندپارامتری.

- روش گرادیان نزولی:

- اگر خلاف جهت بردار گرادیان حرکت کنیم، به سمت نقطه مینیمم تابع در حرکت هستیم!

- اندازه بردار گرادیان، نشان‌دهنده میزان دوری یا نزدیکی به نقطه اکستریم است، پس طول گام را می‌توان متناسب با آن انتخاب کرد.

# استخراج قانون آموزش پس انتشار خطا

- فرض اولیه: تابع خطا فقط برای یک الگو نوشته شده است.

$$E = \frac{1}{2} \sum_k (t_k - y_k)^2$$

$$y_k = f(y_{in_k})$$

$$y_{in_k} = \sum_j z_j w_{jk}$$

- برای وزن‌های بین لایه میانی و لایه خروجی، داریم:

$$\begin{aligned} \frac{\partial E}{\partial w_{JK}} &= \frac{\partial}{\partial w_{JK}} \left( \frac{1}{2} \sum_k (t_k - y_k)^2 \right) \\ &= (t_K - y_K) \cdot \left( -\frac{\partial}{\partial w_{JK}} f(y_{in_K}) \right) \\ &= -(t_K - y_K) \cdot f'(y_{in_K}) \cdot \frac{\partial}{\partial w_{JK}} \left( \sum_j z_j w_{jk} \right) \end{aligned}$$

$$\frac{\partial E}{\partial w_{JK}} = -(t_K - y_K) \cdot f'(y_{in_K}) \cdot z_J$$

$$\Delta w_{JK} = -\alpha \cdot \frac{\partial E}{\partial w_{JK}} = \alpha \cdot \delta_K \cdot z_J$$

$$\delta_K = (t_K - y_K) \cdot f'(y_{in_K})$$

# استخراج قانون آموزش پس انتشار خطا

- برای وزن‌های بین لایه ورودی و لایه میانی، داریم:

$$E = \frac{1}{2} \sum_k (t_k - y_k)^2$$

$$y_k = f(y_{in_k})$$

$$y_{in_k} = \sum_j z_j w_{jk}$$

$$z_j = f(z_{in_j})$$

$$z_{in_j} = \sum_i x_i v_{ij}$$

$$\begin{aligned} \frac{\partial E}{\partial v_{IJ}} &= \frac{\partial}{\partial v_{IJ}} \left( \frac{1}{2} \sum_k (t_k - y_k)^2 \right) = \sum_k \left[ (t_k - y_k) \cdot \left( -\frac{\partial}{\partial v_{IJ}} f(y_{in_k}) \right) \right] \\ &= -\sum_k \left[ (t_k - y_k) \cdot f'(y_{in_k}) \cdot \frac{\partial}{\partial v_{IJ}} \left( \sum_j z_j w_{jk} \right) \right] \\ &= -\sum_k \left[ \delta_k \cdot w_{Jk} \cdot \frac{\partial}{\partial v_{IJ}} z_j \right] = -\sum_k (\delta_k \cdot w_{Jk}) \cdot f'(z_{in_J}) \cdot \frac{\partial}{\partial v_{IJ}} \sum_i x_i v_{ij} \end{aligned}$$

$$\frac{\partial E}{\partial v_{IJ}} = -\sum_k (\delta_k \cdot w_{Jk}) \cdot f'(z_{in_J}) \cdot x_I$$

$$\Delta v_{IJ} = -\alpha \cdot \frac{\partial E}{\partial v_{IJ}} = \alpha \cdot \delta_J \cdot x_I$$

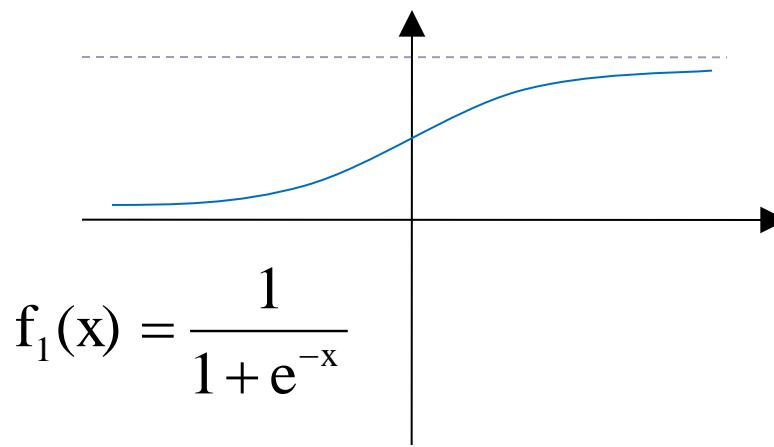
$$\delta_J = \sum_k (\delta_k \cdot w_{Jk}) \cdot f'(z_{in_J})$$

## تابع فعالیت

- ویژگی‌های الزامی برای تابع فعالیت شبکه در آموزش با پس انتشار خطا (حالت استاندارد)
  - پیوسته
  - مشتق‌پذیر در تمام دامنه
  - اکیداً صعودی
- ویژگی غیرالزامی
  - سادگی محاسبه
- مثال:
  - تابع سیگموئید باینری و بایپولار



# نمونه تابع فعالیت



- تابع سیگموئید باینری

- علاوه بر داشتن سه ویژگی اصلی:

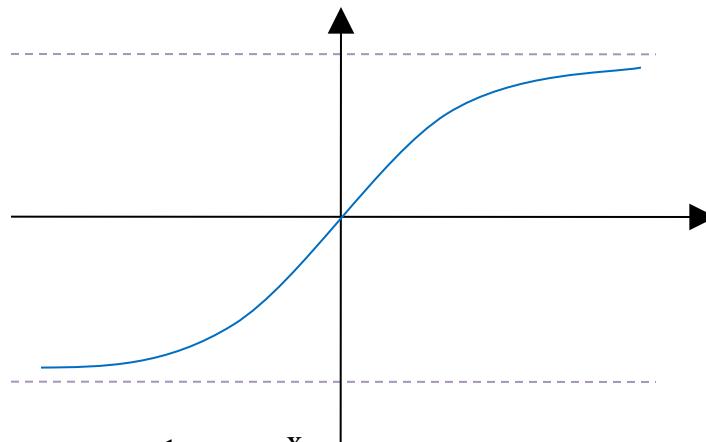
- پیوستگی، مشتق‌پذیری، اکیداً صعودی

مزیت آن این است که مشتق آن، بسیار ساده و به کمک خود تابع محاسبه می‌شود:

$$f_1'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f_1(x) \cdot (1 - f_1(x))$$



# نمونه تابع فعالیت



$$f_2(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

- تابع سیگموئید بایپولار

- مناسب برای خروجی‌های بایپولار

- مزایای پیوستگی، مشتق‌پذیری، اکیداً صعودی و راحتی محاسبه را همانند سیگموئید باینری دارد.

$$f_2'(x) = \frac{2e^{-x}}{(1 + e^{-x})^2} = \frac{1}{2} (1 + f_2(x)) \cdot (1 - f_2(x))$$



# الگوریتم آموزش با پس انتشار خطا

Step 0.

Initialize weights.

(Set to small random values).

Step 1.

While stopping condition is false, do Steps 2–9.

Step 2. For each training pair, do Steps 3–8.

*Feedforward:*

Step 3.

Each input unit ( $X_i, i = 1, \dots, n$ ) receives input signal  $x_i$  and broadcasts this signal to all units in the layer above (the hidden units).

Step 4.

Each hidden unit ( $Z_j, j = 1, \dots, p$ ) sums its weighted input signals,

$$z\_in_j = v_{0j} + \sum_{i=1}^n x_i v_{ij},$$

applies its activation function to compute its output signal,

$$z_j = f(z\_in_j),$$

and sends this signal to all units in the layer above (output units).

Step 5.

Each output unit ( $Y_k, k = 1, \dots, m$ ) sums its weighted input signals,

$$y\_in_k = w_{0k} + \sum_{j=1}^p z_j w_{jk}$$

and applies its activation function to compute its output signal,

$$y_k = f(y\_in_k).$$

فاز یکم: محاسبه خروجی شبکه  
از لایه ورودی به سمت جلو

قرار گرفتن ورودی در ورودی شبکه

محاسبه خروجی لایه میانی (مخفي)

محاسبه خروجی لایه خروجی

# ادامه الگوریتم

## Backpropagation of error:

### Step 6.

Each output unit ( $Y_k, k = 1, \dots, m$ ) receives a target pattern corresponding to the input training pattern, computes its error information term,

$$\delta_k = (t_k - y_k)f'(y_{in_k}),$$

calculates its weight correction term (used to update  $w_{jk}$  later),

$$\Delta w_{jk} = \alpha \delta_k z_j,$$

calculates its bias correction term (used to update  $w_{0k}$  later),

$$\Delta w_{0k} = \alpha \delta_k,$$

and sends  $\delta_k$  to units in the layer below.

Step 7.

Each hidden unit ( $Z_j, j = 1, \dots, p$ ) sums its delta inputs (from units in the layer above),

$$\delta_{in_j} = \sum_{k=1}^m \delta_k w_{jk},$$

multiplies by the derivative of its activation function to calculate its error information term,

$$\delta_j = \delta_{in_j} f'(z_{in_j}),$$

calculates its weight correction term (used to update  $v_{ij}$  later),

$$\Delta v_{ij} = \alpha \delta_j x_i,$$

and calculates its bias correction term (used to update  $v_{0j}$  later),

$$\Delta v_{0j} = \alpha \delta_j.$$

فاز دوم: پس انتشار خطا  
از لایه خروجی به سمت عقب

محاسبه فاکتور خطا در لایه خروجی

محاسبه تغییرات وزن و بایاس بین  
لایه میانی و خروجی

محاسبه فاکتور خطا در لایه میانی

محاسبه تغییرات وزن و بایاس بین  
لایه ورودی و میانی

# ادامه الگوریتم

فاز سوم: اعمال و به روز آوری  
تغییرات وزن و بایاس.

*Update weights and biases:*

*Step 8.* Each output unit ( $Y_k, k = 1, \dots, m$ ) updates its bias and weights ( $j = 0, \dots, p$ ):

$$w_{jk}(\text{new}) = w_{jk}(\text{old}) + \Delta w_{jk}.$$

Each hidden unit ( $Z_j, j = 1, \dots, p$ ) updates its bias and weights ( $i = 0, \dots, n$ ):

$$v_{ij}(\text{new}) = v_{ij}(\text{old}) + \Delta v_{ij}.$$

*Step 9.* Test stopping condition.

اعمال تمامی تغییر وزنها (و بایاس)  
که در فاز قبل، محاسبه شده بود.

- الگوریتم آموزش، در هر تکرار و برای هر الگو، سه مرحله دارد:

- محاسبه خروجی به صورت مستقیم (feed forward)
- محاسبه و انتشار خطا به صورت عقبگرد (back propagation)
- تغییر و تنظیم وزنها (weight adjustment)



# الگوریتم استفاده (کاربرد)

Step 0.

Initialize weights (from training algorithm).

Step 1.

For each input vector, do Steps 2–4.

Step 2. For  $i = 1, \dots, n$ : set activation of input unit

$$x_i;$$

Step 3. For  $j = 1, \dots, p$ :

$$z\_in_j = v_{0j} + \sum_{i=1}^n x_i v_{ij};$$

$$z_j = f(z\_in_j).$$

Step 4. For  $k = 1, \dots, m$ :

$$y\_in_k = w_{0k} + \sum_{j=1}^p z_j w_{jk};$$

$$y_k = f(y\_in_k).$$

- الگوریتم استفاده، فقط یک مرحله دارد:

## — محاسبه خروجی

— پس؛ ممکن است سرعت آموزش شبکه کم باشد، اما وقتی آموزش انجام گرفت، سرعت شبکه در پاسخ به ورودی، مناسب و معقول است.

• توجه مهم: خروجی نهایی شبکه، حاصل تابع فعالیت (مثلًا سیگموئید) است و تقریباً همیشه تفاوت اندکی با مقدار جواب مورد نظر دارد.



# بررسی الگوریتم آموزش

- نکاتی درباره الگوریتم آموزش
  - مثل شبکه‌های تک‌لایه، یک الگوی ورودی ممکن است بارها و بارها به شبکه آموزش داده شود.
- تا هنگام از بین رفتن یا کاهش خطای نرخ قابل تحمل.
- بسیار مهم است که تغییر وزنها، بعد از محاسبه تمامی فاکتورهای خطای در شبکه اعمال شود.
- دلیل: محاسبه خطای لایه قبل، وابسته به وزن‌های بین دو لایه است.
- ایده اصلی، کم کردن خطای روش گرادیان نزولی است.
- تابع گرادیان، میزان خطای متغیر، وزن‌های شبکه است.
- علامت گرادیان، نشان‌دهنده جهتی است که خطای حال زیاد شدن است، پس اگر در خلاف جهت گرادیان حرکت کنیم، خطای کاهش می‌یابد.



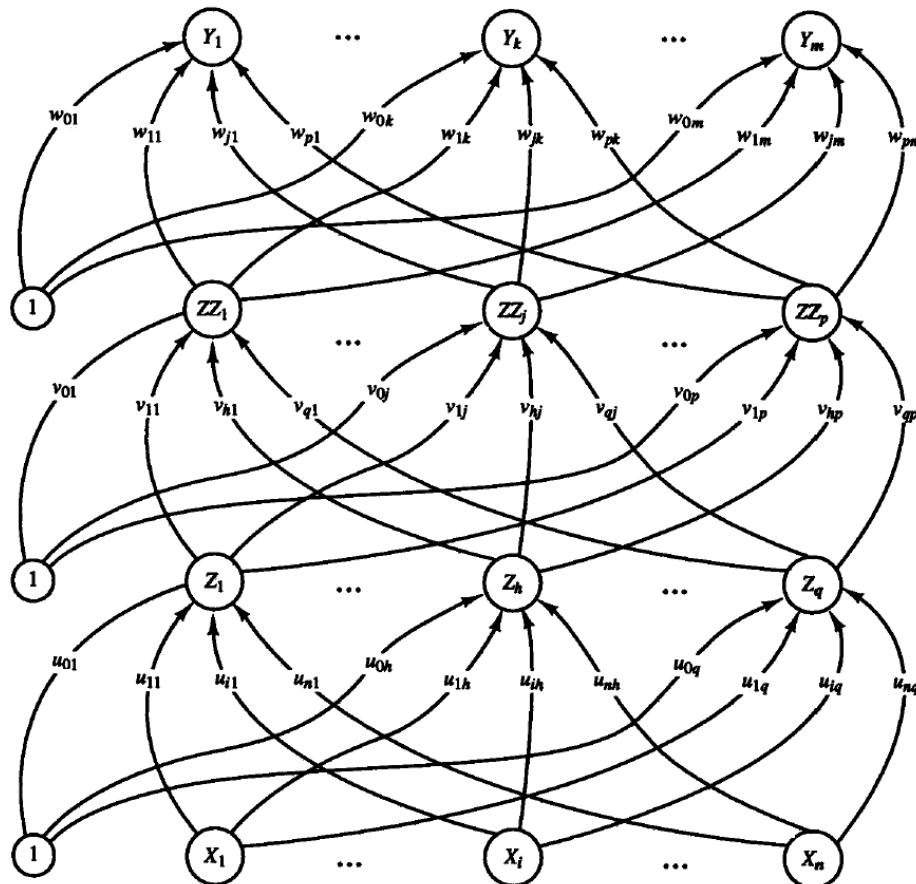
مبانی هوش محاسباتی

فصل دو - شبکه عصبی مصنوعی چندلایه

# نکات تکمیلی درباره شبکه چندلایه با پس انتشار خطا



# شبکه با پیش از یک لایه مخفی



- در صورت افزایش لایه های مخفی شبکه؛ تمامی اعمال برای هر یک از لایه های مخفی تکرار می شود.
  - در فاز ۱، محاسبه فعالیت نرون ها از لایه ورودی به سمت جلو.
  - در فاز ۲، محاسبه فاکتور خطای لایه خروجی به سمت عقب.
  - در فاز ۳، تغییر وزن تمامی لایه ها.

**Feedforward.**

Each input unit ( $X_i, i = 1, \dots, n$ ):

broadcasts input signal to hidden units.

Each hidden unit ( $Z_h, h = 1, \dots, q$ ):

computes input signal

$$z\_in_h = u_{0h} + \sum_{i=1}^n x_i u_{ih},$$

applies activation function to compute output signal

$$z_h = f(z\_in_h),$$

and sends its output signal to the units in the second hidden layer.

Each hidden unit ( $ZZ_j, j = 1, \dots, p$ ):

computes input signal

$$zz\_in_j = v_{0j} + \sum_{h=1}^q z_h v_{hj},$$

applies activation function to compute output signal

$$zz_j = f(zz\_in_j).$$

and sends its output signal to output units.

Each output unit ( $Y_k, k = 1, \dots, m$ ):

sums weighted input signal

$$y\_in_k = w_{0k} + \sum_{j=1}^p zz_j w_{jk}$$

and applies activation function to compute its output signal

$$y_k = f(y\_in_k).$$

# شبکه با دو لایه مخفی

- فاز ۱: پیش خور  
(feedforward)

- محاسبه فعالیت سلول‌ها با شروع از لایه ورودی، به صورت مستقیم، تا لایه خروجی.



## Backpropagation of error.

Each output unit ( $Y_k, k = 1, \dots, m$ ):  
calculates its error

$$e_k = (t_k - y_k),$$

for the current training pattern, multiplies by derivative of activation function (expressed in terms of  $y_k$ ) to get

$$\delta_k = e_k f'(y_{in_k}),$$

calculates weight correction term (used to update  $w_{jk}$  later)

$$\Delta w_{jk} = \alpha \delta_k z_j,$$

calculates bias correction term (used to update  $w_{0k}$  later)

$$\Delta w_{0k} = \alpha \delta_k,$$

and sends  $\delta_k$  to hidden units ( $Z_j, j = 1, \dots, p$ ).

Each hidden unit ( $Z_j, j = 1, \dots, p$ ):  
sums weighted input from units in layer above to get

$$\delta_{in_j} = \sum_{k=1}^m \delta_k w_{jk},$$

multiplies by derivative of its activation function (expressed in terms of  $z_j$ ) to get

$$\delta_j = \delta_{in_j} f'(z_{in_j}),$$

calculates weight correction term (used to update  $v_{hj}$  later)

$$\Delta v_{nj} = \alpha \delta_j Z_n$$

calculates bias correction term (used to update  $v_{0j}$  later)

$$\Delta v_{0j} = \alpha \delta_j,$$

and sends  $\delta_j$  to hidden units ( $Z_h, h = 1, \dots, q$ ).

# شبکه با دو لایه مخفی

## فاز ۲: پس انتشار خطای (backpropagation of error)

- محاسبه فاکتور خطای با شروع از لایه خروجی، به صورت معکوس، تا لایه ورودی.

Each hidden unit ( $Z_h, h = 1, \dots, q$ ):

sums weighted input from units in layer above to get

$$\delta_{in_h} = \sum_{j=1}^p \delta_j v_{hj},$$

multiplies by derivative of its activation function (expressed in terms of  $z_h$ ) to get

$$\delta_h = \delta_{in_h} f'(z_{in_h}),$$

calculates weight correction term (used to update  $v_{ij}$  later)

$$\Delta v_{ih} = \alpha \delta_h x_i,$$

and calculates bias correction term (used to update  $v_{0j}$  later)

$$\Delta v_{0j} = \alpha \delta_j.$$



# شبکه با دو لایه مخفی

## Update Weights and Biases.

For each output unit ( $j = 0, \dots, p; k = 1, \dots, m$ ):

$$w_{jk}(\text{new}) = w_{jk}(\text{old}) + \Delta w_{jk}.$$

For each hidden unit  $Z_j (h = 0, \dots, q; j = 1, \dots, p)$ :

$$v_{hj}(\text{new}) = v_{hj}(\text{old}) + \Delta v_{hj}.$$

For each hidden unit  $Z_h (i = 0, \dots, n; h = 1, \dots, q)$ :

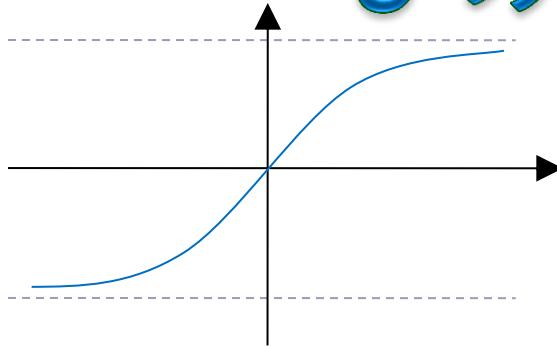
$$u_{ih}(\text{new}) = u_{ih}(\text{old}) + \Delta u_{ih}.$$

- فاز ۳: اعمال تغییرات اوزان و بایاس (update weights & biases)
  - اعمال تمامی تغییرات وزن و بایاس که در فاز قبل، محاسبه شده بود...

- اگر بیش از دو لایه مخفی وجود داشت؛ تمامی مراحل ذکر شده برای لایه‌های مخفی اضافی تکرار می‌شود.
  - در عمل، بیش از دو لایه مخفی استفاده نمی‌شود.



# انتخاب‌های الگوریتم آموزش



$$\Delta w_{JK} = \alpha \cdot \delta_K \cdot z_J = \alpha \cdot (t_K - y_K) \cdot f'(y_{in_K}) \cdot z_J$$

- انتخاب وزن‌های اولیه

- تاثیر مستقیم در همگرایی و سرعت آن.

- وزن‌های اولیه صفر:

- سرعت همگرایی بسیار کم.

- تقویت احتمال گیرافتادن در مینیمم‌های محلی.

- وزن‌های اولیه خیلی کوچک:

- فعالیت سلول‌ها به صفر نزدیک شده و بنابراین سرعت همگرایی کاهش می‌یابد.

- وزن‌های اولیه خیلی بزرگ:

- مقدار مشتق تابع فعالیت سیگموئید، به صفر نزدیک شده و بنابراین سرعت همگرایی کاهش می‌یابد.

- انتخاب مناسب: وزن‌های تصادفی بین [۰/۵ و ۰/۵]



# انتخاب‌های الگوریتم آموزش

- شرط خاتمه الگوریتم
  - چون هدف الگوریتم، کاهش خطای شبکه است، در نگاه اول به نظر می‌رسد شرط خاتمه مناسب، رسیدن خطای کلی شبکه به مقدار قابل تحمل است.
- هدف کلی: برقراری تعادل (balance) بین قدرت به‌یادسپاری (memorization) و قابلیت تعمیم (generalization).
  - به‌یادسپاری؛ یعنی پاسخ درست به ورودی‌های آموزش دیده.
  - تعمیم یا عمومیت بخشی؛ یعنی پاسخ مناسب به ورودی‌های مشابه ورودی‌های آموزش دیده.
- اشکال عمدۀ این روش، این است که ممکن است شبکه در ازای کاهش خط، قابلیت جامع‌نگری و تعمیم را از دست بدهد.

## – روش Hecht-Nielsen برای تعیین شرط خاتمه

- تفکیک الگوها به دو دسته‌جزای آموزشی (training) و اعتبارسنجی (validation)
- روند آموزش و تغییر اوزان به کمک الگوهای آموزشی طی می‌شود.
- در فاصله‌های زمانی مشخص، خطای شبکه برای الگوهای اعتبارسنجی محاسبه می‌شود.
- هر گاه خطای الگوهای آزمایشی رو به افزایش گذاشت، روند آموزش پایان می‌یابد.
- در این نقطه، شبکه قابلیت تعمیم را فدای قابلیت به‌یادسپاری می‌کند.

# انتخاب‌های الگوریتم آموزش

- تعداد الگوهای آموزشی مورد نیاز
  - یک شبکه با یک ساختار مشخص، برای رسیدن به میزان خطای مطلوب، چه تعداد الگوی آموزشی نیاز دارد؟
  - چه رابطه‌ای بین تعداد پارامترهای شبکه، تعداد الگوهای مورد نیاز و خطای شبکه وجود دارد؟
- اگر الگوهای آموزشی به اندازه کافی باشد، شبکه قادر خواهد بود قابلیت تعمیم را به اندازه مورد نظر به دست بیاورد: (Baum & Haussler)
  - رابطه بین تعداد الگوهای آموزشی مورد نیاز، تعداد پارامترها و خطای شبکه
- »  $P$  تعداد وزنهای شبکه،  $W$  تعداد الگوهای آموزشی مورد نیاز،  $e$  خطای مورد انتظار در آموزش

$$P \propto \frac{W}{e}$$

# ورودی-خروجی شبکه

- نوع ورودی و خروجی شبکه
- داده‌های باینری و بایپولار

- «صفر» بودن ورودی یا خروجی، باعث می‌شود یک سلول آموزش نبیند.
- به همین دلیل، توصیه می‌شود از داده‌های بایپولار برای ورودی و خروجی استفاده شود.
- سرعت همگرایی برای الگوهای باینری، تقریباً یکدهم سرعت برای الگوهای بایپولار است.

## داده‌های بایپولار تغییر یافته

- با توجه به ویژگی تابع فعالیت، مشاهده می‌شود دلیل بخشی از خطای نهایی، فاصله خروجی مطلوب و خروجی واقعی است.
- مقادیر خروجی سلول‌ها هیچ‌گاه به مقادیر نهایی (خط مجانب ۱ و -۱) نمی‌رسند.
- انتخاب خروجی مطلوب با توجه به واقعیت فوق.
- به عنوان مثال: ۰/۹ و -۰/۹ - یا ۰/۸ یا -۰/۸



# ورودی-خروجی شبکه

- مثال: بررسی تاثیر نوع ورودی-خروجی و همچنین روش Nguyen-Widrow (تصحیح وزن‌های اولیه) در سرعت همگرایی شبکه
  - مساله XOR
  - دو سلول ورودی، چهار سلول میانی، یک سلول خروجی.
  - شبکه با یک لایه مخفی
  - وزن‌ها و بایاس‌های اولیه، مقادیر تصادفی بین ۰/۵ و -۰/۵

Iterations	RANDOM	NGUYEN-WIDROW
Binary XOR	2891	1935
Bipolar XOR	387	224
Modified bipolar XOR Targets: +0.8 , -0.8	264	127



$w_{jk}$	$y_1$
$z_1$	0.4919
$z_2$	-0.2913
$z_3$	-0.3979
$z_4$	0.3581
'1'	-0.1401

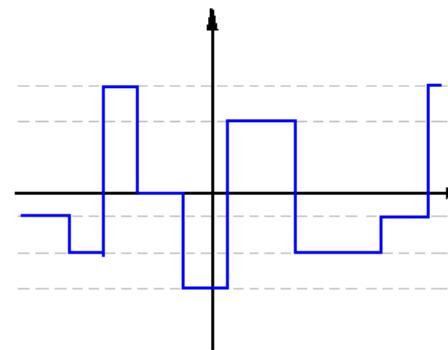
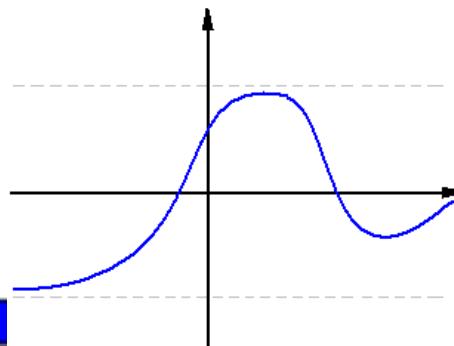
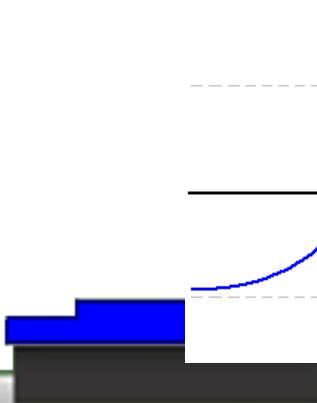
$v_{ij}$	$z_1$	$z_2$	$z_3$	$z_4$
$x_1$	0.1970	0.3191	-0.1448	0.3394
$x_2$	0.3099	0.1904	-0.0347	-0.4861
'1'	-0.3378	0.2771	0.2859	-0.3329



# ورودی-خروجی شبکه

- ورودی-خروجی‌های پیوسته و چندمقداره

- ورودی‌ها و خروجی‌ها می‌توانند مقادیر پیوسته باشند.
- مانند: درجه حرارت یک اتاق که بر حسب درجه سانتیگراد بیان شده است.
- ورودی‌ها یا خروجی‌های شبکه می‌توانند مقادیر جداگانه در یک محدوده باشند.
- مانند درجه حرارت یک اتاق که با یکی از کلمات: [انجماد، سرد، خنک، ولرم، گرم، داغ، جوش] بیان شده باشد.
- آموختش الگوهای پیوسته از الگوهای چندمقداره دشوارتر است.
- الگوهای پیوسته را می‌توان به کمک چندی‌سازی (quantization) به الگوهای چندمقداره تبدیل کرد، در این صورت جواب شبکه در مراتب‌های تفکیک، دقیق نخواهد بود.



# روش‌های تغییر و تصحیح اوزان

- به روز آوری دسته‌ای (batch-updating) یا بعد از آموزش (iteration) یک تکرار (iteration) یا بعد از آموزش چند الگو اعمال می‌شود.
- در این صورت، تغییر وزن‌ها، حاصل جمع تغییر وزن الگوهای آموزش داده شده است.
- مزیت: منحنی تغییرات خطا را نرم کرده و باعث سرعت بیشتر الگوریتم آموزش می‌شود.
- عیب: ممکن است در بعضی از مسایل، باعث همگرایی در مینیمم محلی شود.



# روش‌های تغییر و تصحیح اوزان

- به روز آوری به روش شتاب (گشتاور) (momentum)
  - تغییر وزن‌ها به گرادیان در لحظه جاری و گرادیان در لحظه قبل وابسته است.
- $\mu \in [0,1]$

$$\Delta w_{jk}(t+1) = \alpha \delta_k z_j + \mu \cdot \Delta w_{jk}(t)$$

$$\Delta v_{ij}(t+1) = \alpha \delta_j x_i + \mu \cdot \Delta v_{ij}(t)$$

- اثر لحظه قبل، باقی می‌ماند.
- اگر در چند لحظه متوالی، جهت تغییرات در یک راستا بود، تغییر وزن‌ها بزرگتر می‌شود.
- اگر در چند لحظه متوالی، جهت تغییرات در یک راستا نبود، تغییر وزن‌ها کوچکتر می‌شود.
- مزیت، در مواقعی که ورودی‌های خیلی دور (out-lying) از ورودی‌های درست داریم. این ورودیها در حالت عادی باعث آشفتگی در تغییر وزن‌ها و عدم همگرایی می‌شوند.
- این روش ممکن است شبکه را از گیرافتادن در مینیمم‌های محلی، نجات بخشد.



# روش‌های تغییر و تصحیح اوزان

- نرخ آموزش متغیر

- ایده: افزایش نرخ آموزش برای الگوهای دسته‌ای که تعداد اعضای آن، خیلی کمتر از اعضای دسته‌های دیگر باشد.

- ایده: در هنگام دور بودن از جواب، نرخ آموزش زیاد و در هنگام نزدیکی به جواب، کم باشد.

- ایده: نرخ آموزش هر وزن، مخصوص به خودش بوده و با توجه به میزان تغییرات آن وزن، تغییر کند.

- روش delta-bar-delta، ترکیب دو ایده فوق.



# تابع فعالیت

- با توجه به عملکرد مورد انتظار شبکه، می‌توان تابع فعالیت آن را تغییر داد.

– تابع سیگموئید با برد معین

- محدوده بُرد تابع سیگموئید نرون‌های لایه خروجی، نوع خروجی شبکه را معین می‌کند.

– سیگموئید باینری و سیگموئید باپولار، برای داده‌های باینری و باپولار.

- تابع سیگموئید برای داده‌های محدوده پیوسته  $[a, b]$

$$m = b - a$$

$$g(x) = m \cdot f(x) - n$$

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$g'(x) = \frac{1}{m} [n + g(x)] [m - n - g(x)]$$



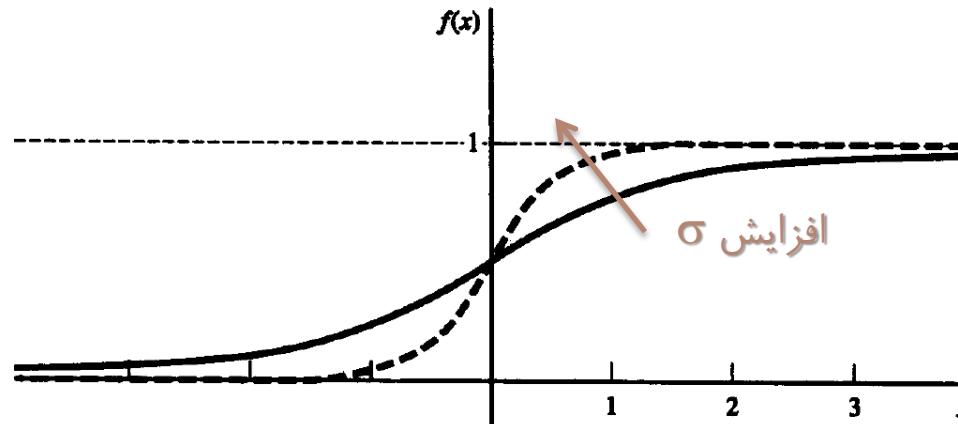
# توابع فعالیت

– تابع سیگموئید با شیب متغیر

- میتوان شیب یا تندری (steepness) یک تابع سیگموئید را به کمک پارامتر  $\sigma$  در فرمول زیر، تعیین کرد:
- $$f(x) = \frac{1}{1 + e^{-\sigma \cdot x}}$$

$$f'(x) = \sigma \cdot f(x) \cdot [1 - f(x)]$$

– نکته: گاهی شیب تابع فعالیت نیز در الگوریتم آموزش، متغیر است.



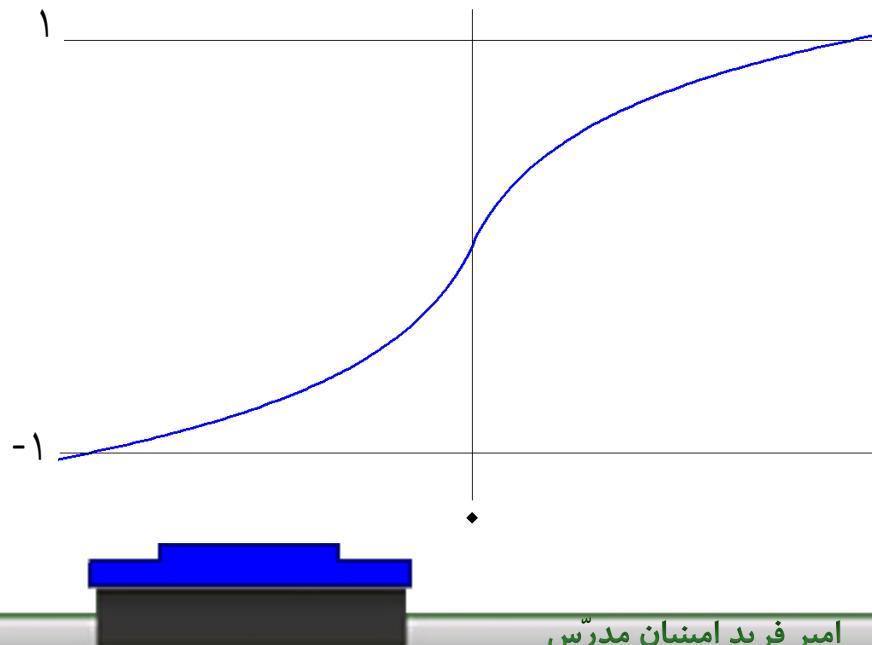
Binary sigmoid with  $\sigma = 1$  and  $\sigma = 3$

# توابع فعالیت

– توابع فعالیت غیراشباعی (non-saturating)

- توابعی که خط مجانب ندارند و به طور معمول، به مقدار نهایی مطلوب می‌رسند.

– تابع لگاریتمی (logarithmic)



$$f(x) = \begin{cases} \log(1+x) & x > 0 \\ -\log(1-x) & x < 0 \end{cases}$$

# توابع فعالیت

## – توابع فعالیت غیرسیگموئید

- توابع حلقوی (radial-basis)

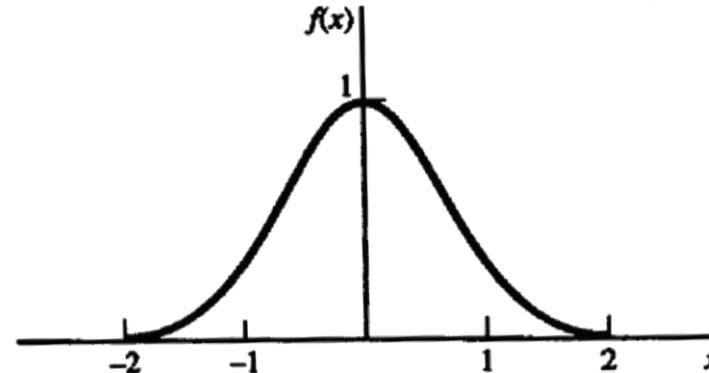
– توابعی که در کل دامنه، مقدار غیرمنفی دارند و فعالیت آنها، حول یک نقطه به اوج خود می‌رسد. هرچه نقطه ورودی از مرکز این تابع دور باشد، فعالیت تابع ضعیفتر می‌شود.

« مثال: تابع گوسی (Gaussian)

- استفاده در لایه مخفی شبکه عصبی با تابع با سطح مقطع حلقوی (Radial-Basis Function Neural Network - RBFNN)

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

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



شبکه‌های عصبی مصنوعی – فصل سوم

# کاربردهای شبکه‌های چندلایه



# کاربردهای شبکه‌های عصبی

- دسته‌بندی classification
- بهینه‌سازی optimization
- مدل‌سازی modeling
- تخمین و تقریب توابع function approximation & estimation
- درون‌یابی و برون‌یابی interpolation & extrapolation
- پیش‌بینی prediction
- فشرده‌سازی compression



# مثال ۱ - شبیه سازی تابع سینوسی

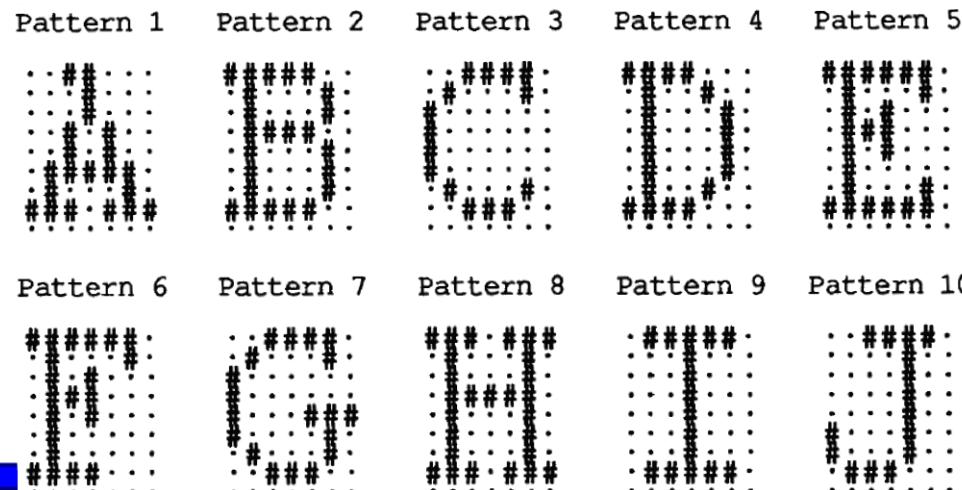
۱.۰	0.00	0.00	0.00	0.00	0.00	0.00
	-0.01	-0.05	-0.12	-0.09	-0.01	0.00
۰.۸	0.00	-0.90	-0.56	0.56	0.90	0.00
	-0.03	-0.91	-0.55	0.53	0.97	0.02
۰.۶	0.00	-0.56	-0.36	0.36	0.56	0.00
	-0.00	-0.59	-0.32	0.37	0.51	0.03
۰.۴	0.00	0.56	0.36	-0.36	-0.56	0.00
	0.01	0.57	0.33	-0.35	-0.55	0.00
۰.۲	0.00	0.90	0.56	-0.56	-0.90	0.00
	0.02	0.84	0.57	-0.57	-0.89	-0.01
۰.۰	0.00	0.00	0.00	0.00	0.00	0.00
	-0.02	-0.02	-0.02	-0.02	-0.03	-0.02
$x_2$	۰.۰	۰.۲	۰.۴	۰.۶	۰.۸	۱.۰

- مساله: تخصیص خروجی طبق فرمول زیر، به هر زوج  $(x_1, x_2)$ 

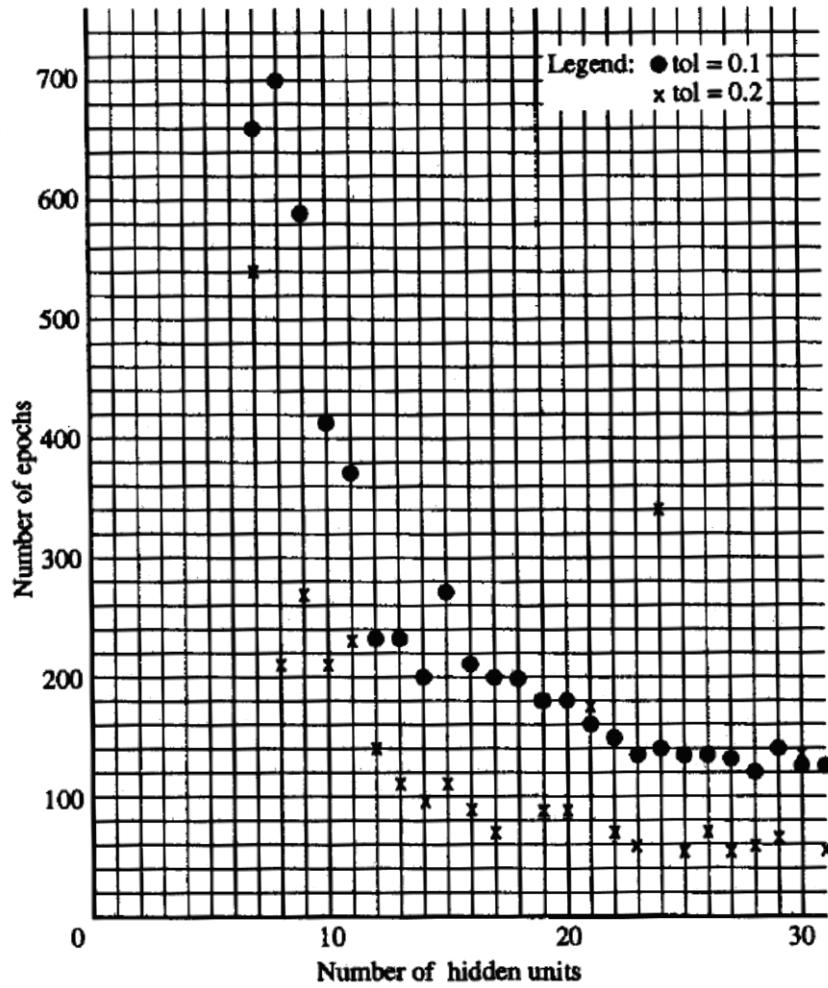
$$y = \sin(2\pi x_1) \cdot \sin(2\pi x_2)$$
- ورودی: اعداد حقیقی، در محدوده  $[0, 1]$  با فاصله  $\frac{1}{2}$ .
- شبکه با یک لایه مخفی شامل ۱۰ سلول.
- تابع فعالیت لایه مخفی، لگاریتمی؛ و لایه خروجی: تابع همانی.
- با نرخ آموزش  $0.05$  بعد از ۵۰۰۰ تکرار، به نرخ خطای مجدور  $0.024$  رسیده‌ایم.
  - سطر اول، هدف (target) و سطر دوم، خروجی شبکه (output) بعد از آموزش.
- تخمین تابع، تقریب تابع، درون‌یابی و برون‌یابی، مدل‌سازی، بهینه‌سازی.

## مثال ۲ - فشرده‌سازی کاراکترها

- کاهش ابعاد فضای ویژگی، به کمک شبکه عصبی.
- هدف: به جای ۶۴ ویژگی، تعداد کمتری ویژگی برای نمایش و تشخیص کاراکتر داشته باشیم.
- شبکه عصبی با یک لایه مخفی، ۶۳ ورودی، ۶۳ سلول خروجی.
- ورودی و خروجی شبکه، یکسان است. (یک کاراکتر به عنوان ورودی و خروجی یک الگو داده می‌شود).
- سوال: تعداد سلول‌های میانی چه تعداد باشد؟



## ادامه مثال ۲ - فشرده‌سازی کاراکترها



- سلول‌های لایه میانی، کمتر از ۶۳ انتخاب می‌شوند تا هدف فشرده‌سازی محقق گردد.
  - در این صورت، پس از آموزش شبکه عصبی، (فعالیت) سلول‌های میانی می‌توانند نمایش خلاصه و فشرده‌ای از کاراکتر ورودی باشد.
  - حداقل سلول‌های لایه میانی قابل محاسبه است.
- اگر  $N$  الگوی دوبه‌دو برهم عمود داشته باشیم، حداقل به  $\log_2 N$  سلول میانی احتیاج داریم.
- فشرده‌سازی، مدل‌سازی، تخمین تابع