



بخش اول

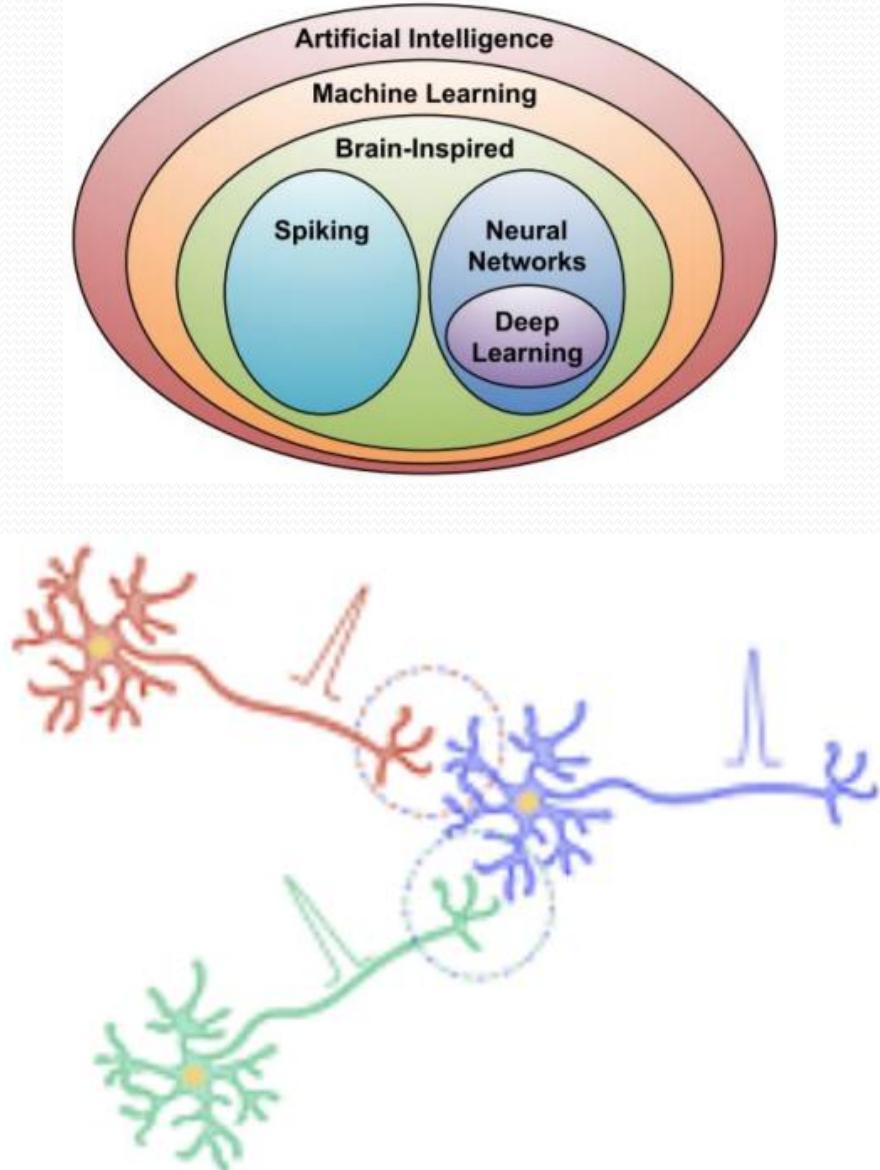
شبکه های عصبی مصنوعی

(Artificial Neural Networks)

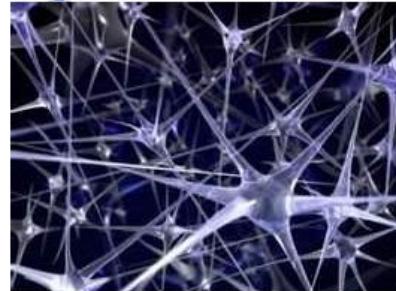
S. H. Klidbary

# موضوعات

- مقدمه
- معنای شبکه های عصبی
- مدل ریاضی نورون
- شبکه عصبی پرسپترون
- شبکه های عصبی چند لایه
- شبکه های عصبی RBF
- ✓ شبکه های عصبی PNN
- شبکه SVM
- شبکه های عصبی RNN,CNN و ...
- شبکه های عصبی اسپایکی



# مغز انسان



✓ ذخیره اطلاعات در اتصالات سیناپسی (معادل وزن ها)

- مغز انسان

- ✓ دارای  $10^{11}$  نورون

- ✓ دارای  $10^{15}$  اتصال سیناپسی

- نورون (Neuron) = عصب طبیعی (سلول مغزی)

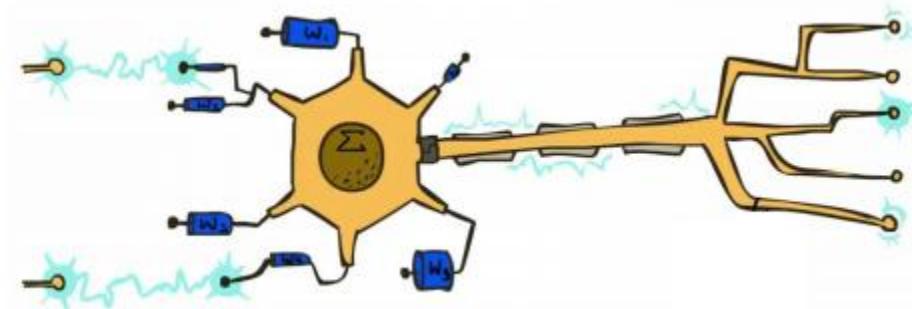
- ✓ عنصر پردازشگر تشکیل دهنده یک شبکه عصبی مصنوعی

- سه جز تشکیل دهنده یک نورون طبیعی

- ✓ دندrit ها

- ✓ سوما

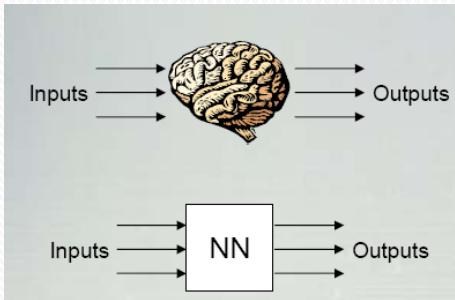
- ✓ آکسون



# شبکه های عصبی زیستی

- مجموعه ای بسیار عظیم از پردازشگرهای موازی به نام نورون اند و توسط سیناپس ها اطلاعات را منتقل می کنند.
- دریافت سیگنال از سایر نورون ها توسط دندریت ها
- عبور سیگنال ها با یک فرآیند شیمیایی از فاصله سیناپسی (Synaptic Gap)
- عمل شیمیایی انتقال دهنده، سیگنال ورودی را تغییر می دهد(تضعیف/تقویت سیگنال)
- سوما سیگنال های ورودی به سلول را جمع می بندد.
- زمانی که یک سلول به اندازه کافی ورودی دریافت نماید(تغییر غلظت یون ها)، برانگیخته می شود و سیگنالی را از آکسون خود به سلول های دیگر می فرستد.
- یادگیری در این سیستم ها به صورت تطبیقی صورت می گیرد، یعنی با استفاده از مشاهدات، وزن سیناپس ها به گونه ای تغییر می کند که در صورت دادن ورودی های جدید، سیستم پاسخ درستی تولید کند.

# شبکه های عصبی مصنوعی



- ایده اصلی این شبکه ها مبتنی «شبکه های عصبی زیستی» است.
- شبکه عصبی مصنوعی

- ✓ یک سیستم پردازش اطلاعات با ویژگی های مشترکی با شبکه های عصبی طبیعی
- ✓ تعمیم یافته مدل های ریاضی تشخیص انسان بر اساس زیست شناسی عصبی

- فرضیات پایه شبکه عصبی مصنوعی
  - ✓ پردازش اطلاعات در اجزای ساده ای با تعداد فراوان، به نام نورون صورت می گیرد.
  - ✓ سیگنال ها در بین شبکه از طریق پیوندها یا اتصالات آن ها منتقل می شوند.
  - ✓ هر پیوند، وزن مربوط به خود را دارد که در شبکه های عصبی رایج در سیگنال های انتقال یافته از آن پیوند ضرب می شود.
  - ✓ هر نورون یک تابع فعال سازی را بر روی ورودی های خود را اعمال می کند تا سیگنال خروجی خود را تولید نماید.

گره ها → نرون ها

وزن ها → سیناپس ها

# شبکه های عصبی مصنوعی

## ● یادگیری

✓ هدف از آموزش شبکه، رسیدن به شرایطی است که شبکه قادر به پاسخگویی صحیح به داده‌های ارائه شده در آموزش شبکه (به خاطر سپردن) و همچنین داده‌های مشابه و متفاوت از ورودی هایی که از آنها برای آموزش شبکه استفاده شده است(تعییم دادن)، باشد.

□ منظور از یادگیری در ANN، تنظیم وزن‌ها و بایاس‌های شبکه می‌باشد.

✓ برتری عمده شبکه‌های عصبی آموزش داده شده بر محاسبات کلاسیک این است که نتایج مورد نیاز با تلاش کمتر و در زمان کمتری قابل حصول است. در نتیجه این مزایا خصوصاً برای مسائلی که مستلزم محاسبات طولانی هستند بسیار مفید و موثر واقع گردد.

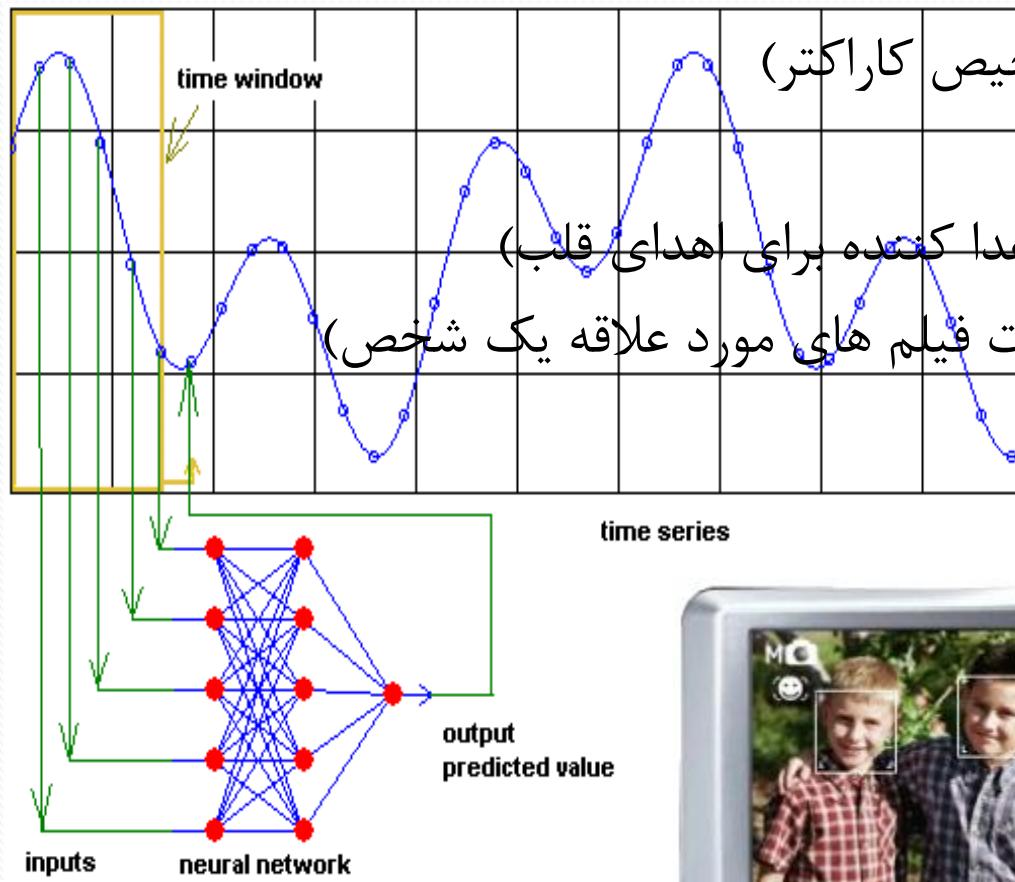
✓ دو نوع آموزش شبکه به شکل زیر است:

الف - آموزش با معلم

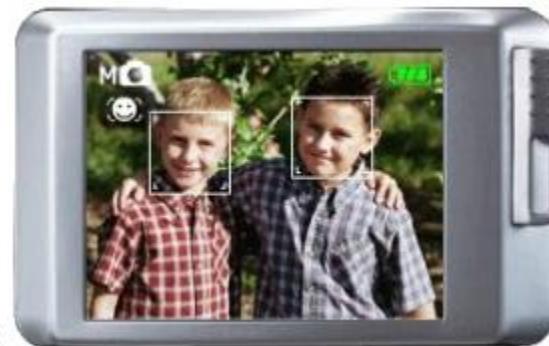
ب - آموزش بدون معلم

# شبکه های عصبی مصنوعی

## ● قابلیت های شبکه عصبی مصنوعی:



- ✓ یادگیری (تشخیص چهره / تشخیص کاراکتر)
- ✓ ذخیره سازی اطلاعات
- ✓ تصمیم گیری (یافتن بهترین اهدای کنده برای اهدای قلب)
- ✓ پیش بینی (قیمت سهام / لیست فیلم های مورد علاقه یک شخص)
- ✓ محاسبه
- ✓ تقریب تابع (مدل سازی تابع)
- ✓ پردازش سیگنال
- ✓ رگرسیون

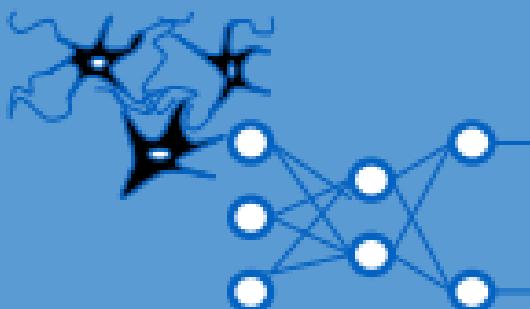


# شبکه های عصبی مصنوعی

## • ویژگی های شبکه های عصبی:

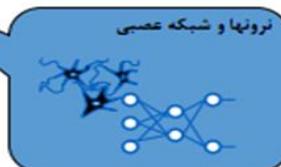
- پردازش موازی (سرعت بالا)
- تعمیم پذیری
- محاسبات غیرخطی
- برقراری ارتباط یک سری ورودی و یک سری خروجی
- بازیابی اطلاعات ✓
- توانایی تطبیق
- پاسخ به داده های نویزی
- تحمل پذیری خطأ
- یادگیری

نرونها و شبکه عصبی



# شبکه های عصبی مصنوعی

- یک شبکه عصبی باید خصوصیات زیر را داشته باشد:
  - ✓ بتواند الگوها را طبقه بندی کند.
  - ✓ به اندازه کافی کوچک باشد تا از نظر فیزیکی واقع گرایانه باشد.
  - ✓ با بکار گیری آموزش، قدرت یادگیری داشته باشد.
  - ✓ توانایی تنظیم پارامترهای شبکه (وزن ها)، را داشته باشد.
  - ✓ اگر شبکه برای یک وضعیت خاص آموزش دید و تغییر کوچکی در شرایط محیطی شبکه رخ داد، شبکه بتواند با آموزش مختصر، برای شرایط جدید نیز کارآمد باشد.
  - ✓ توانایی تعمیم را با استفاده از مثال های ارائه شده در فرآیند آموزش، داشته باشد.



# شبکه های عصبی مصنوعی

## • نیازمندی های شبکه های عصبی

- ✓ جمع آوری و آنالیز مناسب داده
- ✓ طرح، آموزش و تست

Feedback یا Feed forward یا ساختار شبکه (چندلایه و از نوع (Recurrent))

- کدینگ اطلاعات
- الگوریتم یادگیری (با ناظر یا بدون ناظر)
- ❖ انتخاب وزن ها به صورت تصادفی
- ❖ اعمال مجموعه آموزشی

$$S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

$$x_i = (x_{i1}, x_{i2}, \dots, x_{iD})^T$$

- اعمال هر ورودی به شبکه و به دست آوردن خروجی
- مقایسه خروجی مطلوب و واقعی
- آموزش شبکه به صورت تغییر وزن ها و در جهت نزدیک شدن خروجی مطلوب و واقعی

# شبکه های عصبی مصنوعی مک کلاچ- پیترز

- نورون مک کلاچ- پیترز

- ✓ اولین شبکه های عصبی مصنوعی

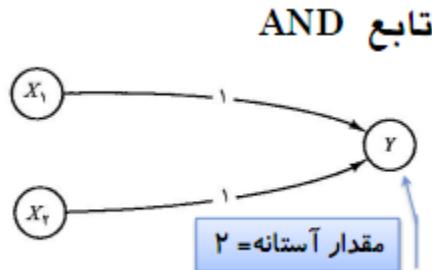
- ✓ ارائه شده در سال ۱۹۴۳ و توسعه آن در سال ۱۹۴۷

- ✓ تنظیم وزن های نورون ها برای ایجاد نقش یک واحد منطقی ساده

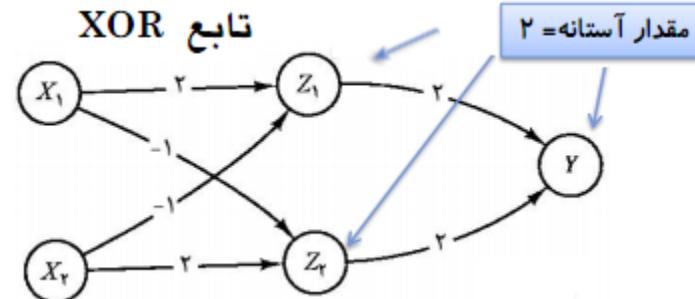
- ❑ وزن مثبت روی اتصال = مسیر تحریکی

- ❑ وزن منفی روی اتصال = مسیر بازدارنده

- ✓ هر نورون دارای آستانه ثابتی است، اگر ورودی شبکه به آن نورون، بزرگ تر از مقدار آستانه باشد، نورون برانگیخته می شود.

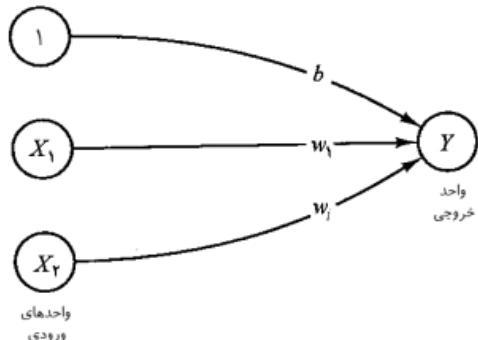


$x_1$	$x_2$	$\rightarrow$	$y$
1	1		1
1	0		0
0	1		0
0	0		0



$x_1$	$x_2$	$\rightarrow$	$y$
1	1		0
1	0		1
0	1		1
0	0		0

# شبکه های عصبی مصنوعی هب



- یادگیری هب

✓ دونالد هب یکی از روانشناسان دانشگاه McGill

✓ اولین (و ساده‌ترین) قانون یادگیری برای شبکه های عصبی در سال ۱۹۴۹

✓ در صورت فعال شدن هم زمان دو نورون، تقویت ضرایب بین آن‌ها (افزایش استحکام اتصال بین آن‌ها)

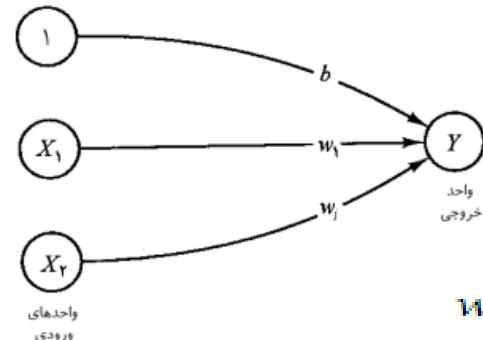
□ بیان دیگر یادگیری هب: در این روش وزن مربوط به ورودی یک نورون، زمانی افزایش می‌یابد که سیگنال ورودی و خروجی هر دو فعال باشند.

✓ هب درباره نورون‌هایی که به طور هم زمان برانگیخته نمی‌شوند، چیزی نمی‌گوید.

✓ فرضیه مطرح شده توسط هب، در حال حاضر بر روی تحقیقات عصب‌شناسی موثر است.

## شبکه های عصبی مصنوعی هب

- یادگیری هب (شبکه هب یک لایه است)



مرحله ۰ - به تمام وزن ها مقدار اولیه صفر بدهید  $w_i = 0 \quad (i = 1, \dots, n)$

مرحله ۱ - برای هر بردار آموزش ورودی و خروجی هدف،  $s: t$ ، مراحل ۲ تا ۴ را انجام بده

مرحله ۲ - فعال سازی های واحد های ورودی را تعیین کن  $x_i = s_i \quad (i = 1, \dots, n)$

مرحله ۳ - برای واحد خروجی فعال سازی را تعیین کن  $y = t$

مرحله ۴ - وزن ها و بایاس را بروز کن

$$w_i(\text{new}) = w_i(\text{old}) + x_i y \quad (i = 1, \dots, n)$$

$$b(\text{new}) = b(\text{old}) + y$$

$$w(\text{new}) = w(\text{old}) + x \cdot y \quad \Rightarrow \quad w(\text{new}) = w(\text{old}) + \Delta w$$

$\Delta w = x \cdot y$

داده های آموزشی فقط یک بار به شبکه نشان داده شده و آموزش به اتمام می رسد

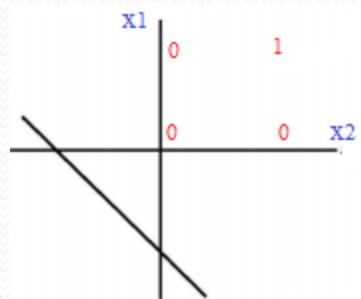
## شبکه های عصبی مصنوعی هب

INPUT	TARGET
( $x_1$ $x_2$ 1)	1
(1    1    1)	1
(1    0    1)	0
(0    1    1)	0
(0    0    1)	0

- مثال: تابع AND با ورودی ها و هدف های دودویی

برای ورودی ها

INPUT	TARGET	WEIGHT CHANGES	WEIGHTS
( $x_1$ $x_2$ 1)	$t$	( $\Delta w_1$ $\Delta w_2$ $\Delta b$ )	( $w_1$ $w_2$ $b$ )
			(0    0    0) ← مقادیر اولیه ها
داده اول	(1    1    1)	1    1    1	(1    1    1)
داده دوم	(1    0    1)	0    0    0	(1    1    1)
داده سوم	(0    1    1)	0    0    0	(1    1    1)
داده چهارم	(0    0    1)	0    0    0	(1    1    1)



$$x_2 = -x_1 - 1$$

پاسخ  
نادرست

## شبکه های عصبی مصنوعی هب

INPUT	TARGET
$(x_1 \ x_2 \ 1)$	$t$
(1 1 1)	1
(1 0 1)	-1
(0 1 1)	-1
(0 0 1)	-1

- مثال: تابع AND با ورودی های دودویی و هدف های دوقطبی

برای ورودی ها

INPUT	TARGET	WEIGHT CHANGES	WEIGHTS
-------	--------	----------------	---------

	$(x_1 \ x_2 \ 1)$	$t$	$(\Delta w_1 \ \Delta w_2 \ \Delta b)$	$(w_1 \ w_2 \ b)$	مقادیر اولیه ها
داده اول	(1 1 1)	1	(1 1 1)	(1 1 1)	
داده دوم	(1 0 1)	-1	(-1 0 -1)	(0 1 0)	
داده سوم	(0 1 1)	-1	(0 -1 -1)	(0 0 -1)	
داده چهارم	(0 0 1)	-1	(0 0 -1)	(0 0 -2)	

پاسخ  
نادرست

نکته: انتخاب الگوهای آموزش و نحوه نمایش آن ها نقش مهمی در تعیین قابل حل بودن مسائل با استفاده از قانون هب دارد.

# شبکه های عصبی مصنوعی هب

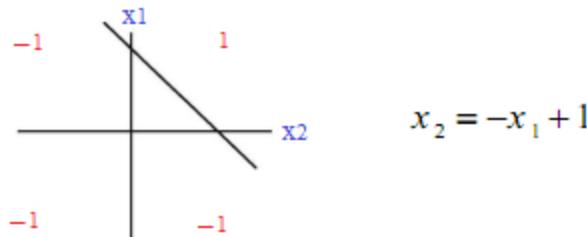
INPUT	TARGET
$(x_1 \ x_2 \ l)$	$t$
(1 1 1)	1
(1 -1 1)	-1
(-1 1 1)	-1
(-1 -1 1)	-1

- مثال: تابع AND با ورودی های و هدف های دو نقطی

برای ورودی ها

INPUT	TARGET	WEIGHT CHANGES	WEIGHTS
$(x_1 \ x_2 \ l)$	$t$	$(\Delta w_1 \ \Delta w_2 \ \Delta b)$	$(w_1 \ w_2 \ b)$
			<span style="border: 1px solid blue; padding: 2px;">(0 0 0)</span>
داده اول	(1 1 1)	1	(1 1 1)
داده دوم	(1 -1 1)	-1	(0 2 0)
داده سوم	(-1 1 1)	-1	(1 1 -1)
داده چهارم	(-1 -1 1)	-1	(2 2 -2)

مقادیر اولیه ها



پاسخ  
درست

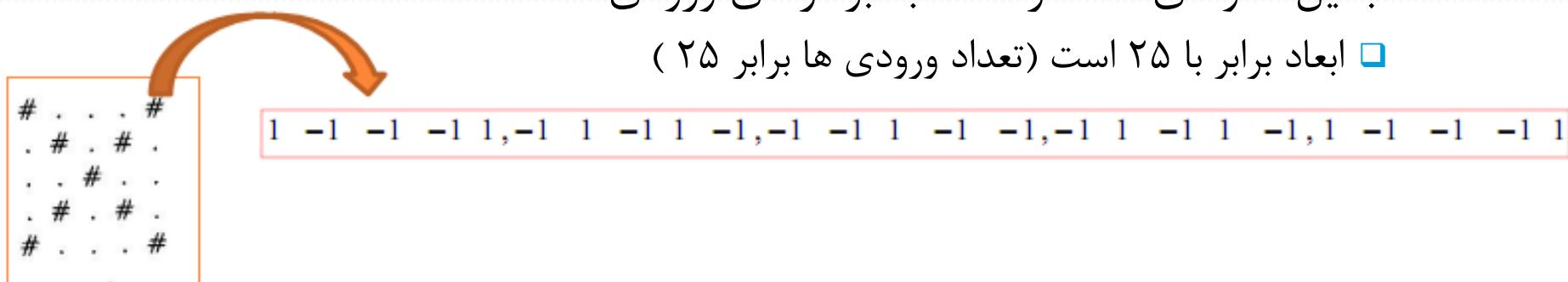
## شبکه های عصبی مصنوعی هب

- مثال: یک شبکه هب برای تشخیص الگوی «X» از «O»

# . . . # . # . # . . . # . . . # . # . # . . . #	and	. # # # . # . . . # # . . . # # . . . # . # # # .
الگوی ۱		الگوی ۲

✓ تبدیل الگوهای «X» از «O» به بردارهای ورودی

□ ابعاد برابر با ۲۵ است (تعداد ورودی ها برابر ۲۵)



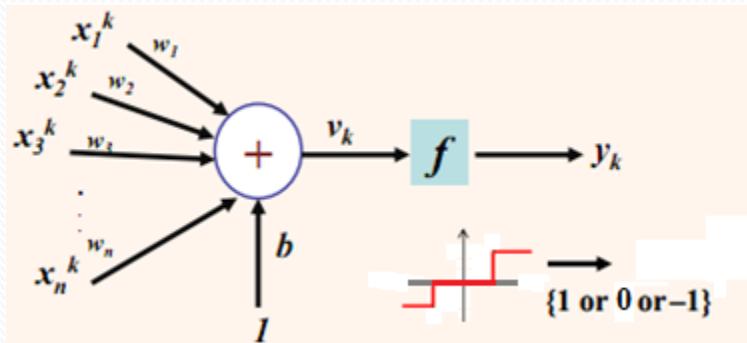
✓ بررسی اینکه الگوی ورودی در کدام دسته قرار دارد (پاسخ بله (+1) و خیر(-1))

$$y\_in = b + \sum_i x_i w_i \rightarrow b + \sum_i x_i w_i = 0 \text{ (Decision Boundary)}$$



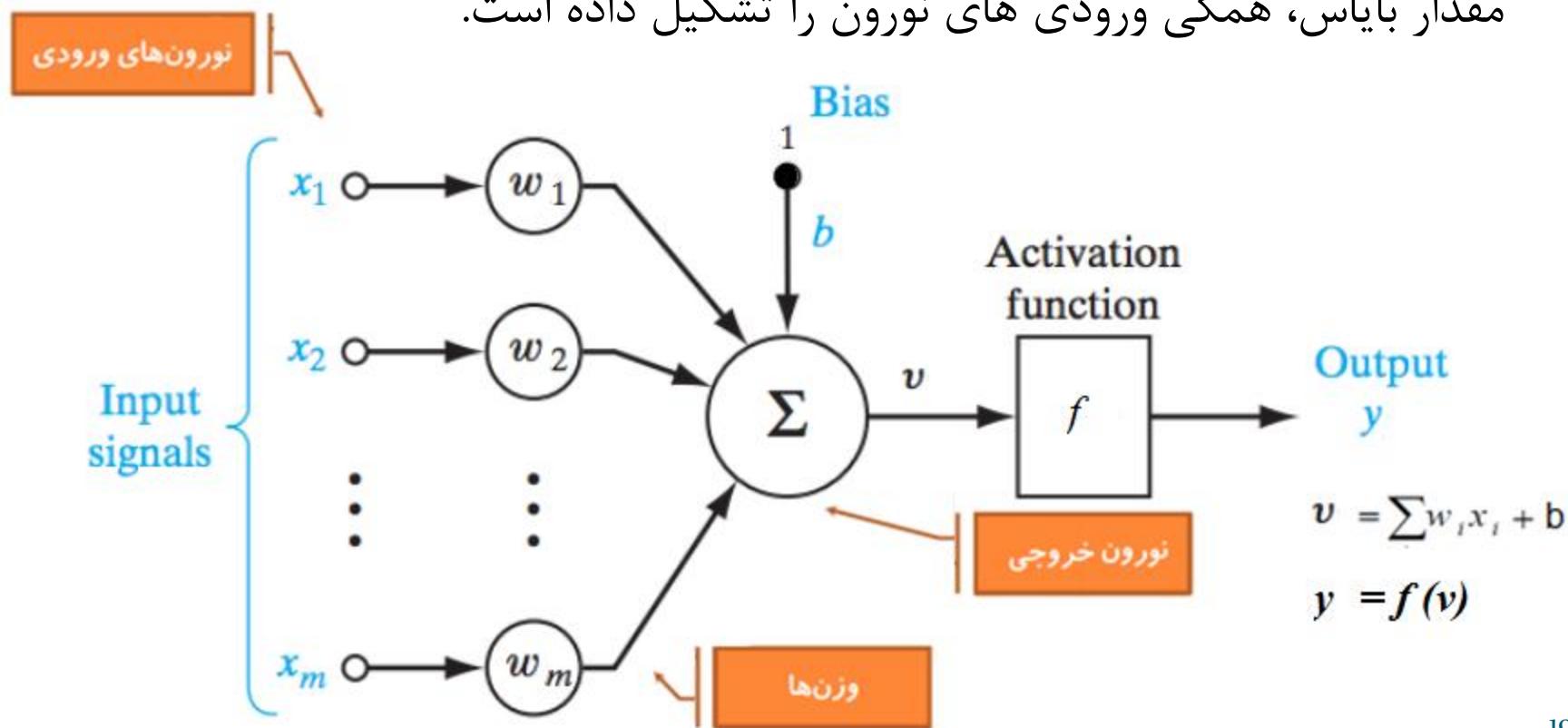
## شبکه های تک لایه (شبکه های عصبی پرسپترون)

- شبکه های تک لایه، با توابع فعال سازی آستانه ای، توسط روزنبلات در سال ۱۹۵۸ بنیان گذاری شدند که این نوع شبکه ها، پرسپترون نامیده شدند.
- الهام گرفته از شبکیه چشم
- یک شبکه پرسپترون یک بردار ورودی را گرفته، ترکیبی خطی از آن ها را محاسبه نموده، خروجی را فراهم می آورد. اگر خروجی از میزان آستانه ای بالاتر بود یک و در غیر این صورت صفر(منهای یک) باز می گرداند.
- پرسپترون توانایی جداسازی داده های دوستحی را دارد(می توان آن را به صورت یک جدا کننده دوستحی در نظر گرفت).



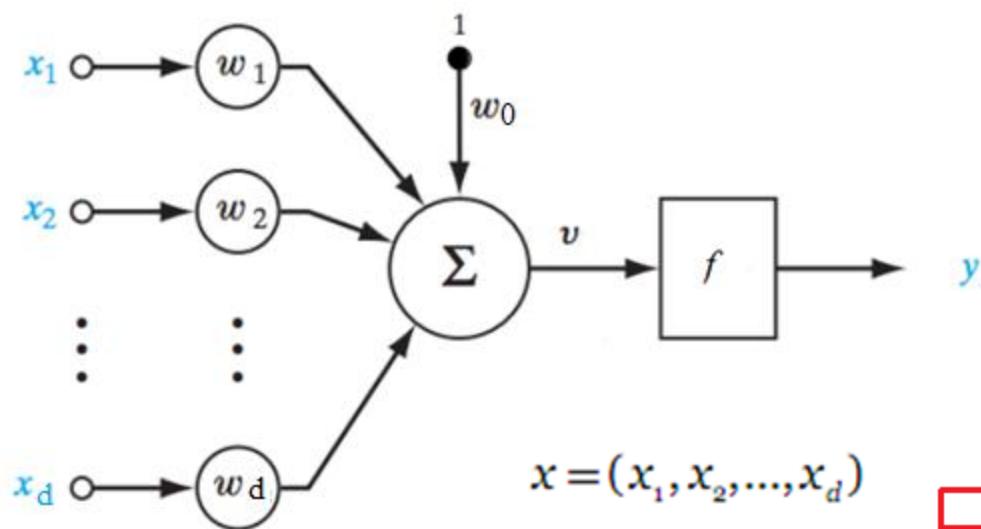
# مدل ریاضی نورون

- سیگنال های ورودی  $X_1$  تا  $X_n$  معادل سیگنال های عصبی ورودی و وزن های  $W_1$  تا  $W_n$  معادل مقادیر اتصالات سیناپسی ورودی های نورون می باشند که در کنار مقدار بایاس، همگی ورودی های نورون را تشکیل داده است.



## مدل ریاضی نورون

- فرم ماتریسی



$$x = (1, x_1, x_2, \dots, x_d)$$

$$w = (w_0, w_1, w_2, \dots, w_d)$$

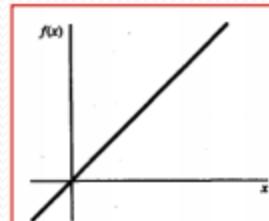
$$v(x) = w^t x$$

اگر بردار ویژگی  $x$  متعلق به کلاس  $C_1$  باشد  $w^t x > 0$

اگر بردار ویژگی  $x$  متعلق به کلاس  $C_2$  باشد  $w^t x < 0$

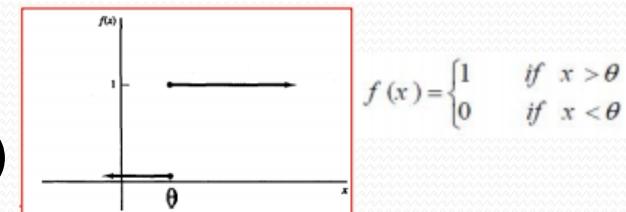
# انواع توابع تحریک(انگیزش)

- Linear function



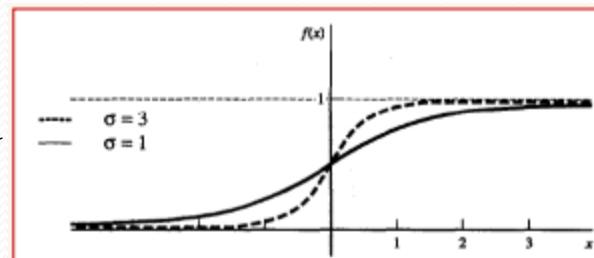
$$f(x) = x$$

- Binary step function (with threshold  $\Theta$ )



$$f(x) = \begin{cases} 1 & \text{if } x > \theta \\ 0 & \text{if } x < \theta \end{cases}$$

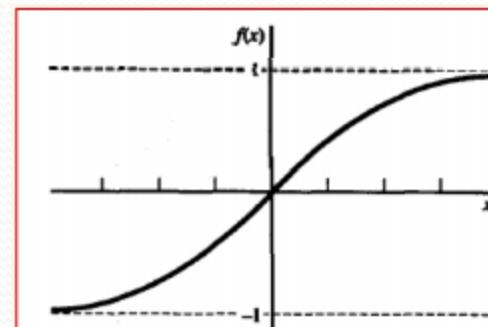
- Binary sigmoid



$$f(x) = \frac{1}{1 + \exp(-\sigma x)}$$

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

- Bipolar sigmoid

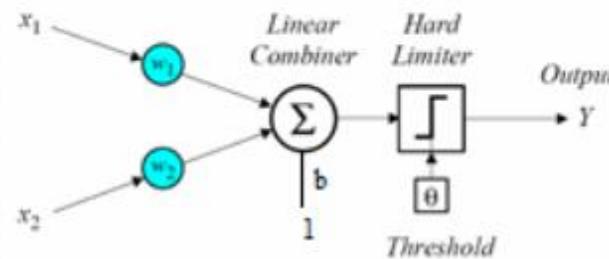


$$g(x) = 2f(x) - 1 = \frac{2}{1 + \exp(-\sigma x)} - 1 = \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)}$$

# شبکه های عصبی پرسپترون

Single-layer two-input Single-layer two-input perceptron

Inputs



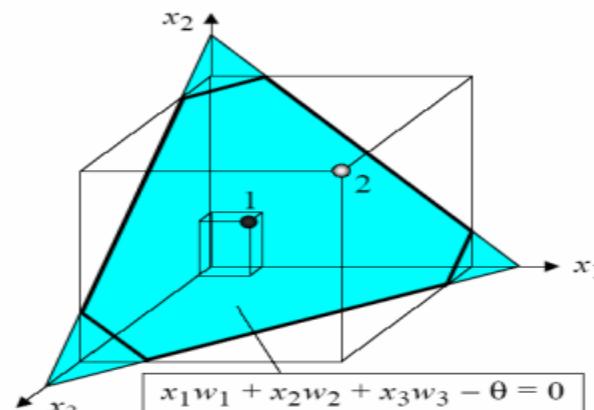
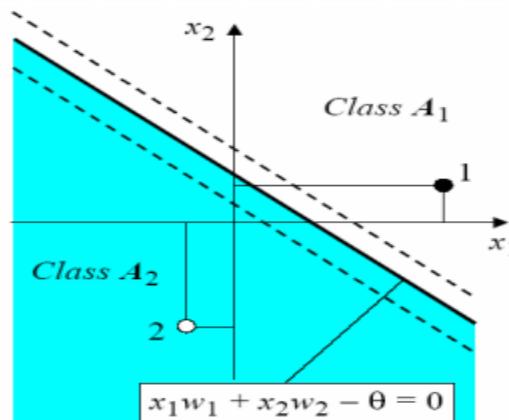
- یادگیری پرسپترون

✓ عبارت است از پیدا کردن مقادیر مناسب برای  $W$

- مرز جدا کننده

✓ خط مرزی همواره عمود بر بردار وزن بوده و محل قرار گرفتن مرز توسط بردار

بایاس تعیین می شود.

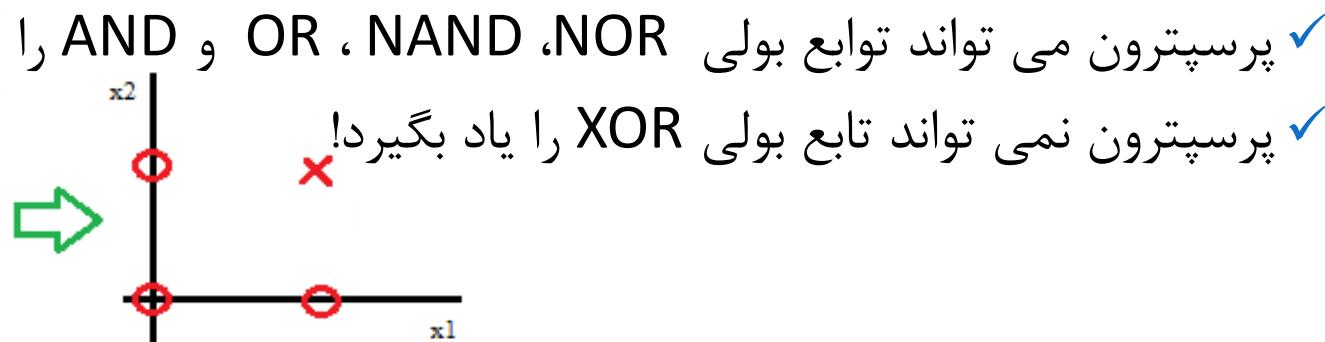


# شبکه های عصبی پرسپترون

## ● توابع بولی و پرسپترون

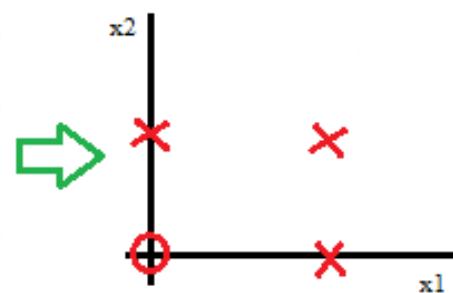
**AND**

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1



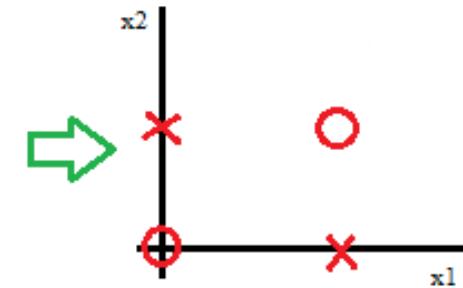
**OR**

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1



**XOR**

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0



## AND

$x_1$	$x_2$	$z$
0	0	-1
0	1	-1
1	0	-1
1	1	1

# شبکه های عصبی پرسپترون

مثال:

تابع بولی ✓ AND

برای الگوی 1

$$y = \text{Sign}(w x + b) = \text{Sign}([0.5 \ 0.5] \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.8) = \text{Sign}(-0.8) = -1$$

برای الگوی 2

$$y_2 = \text{Sign}(w x + b) = \text{Sign}([0.5 \ 0.5] \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 0.8) = \text{Sign}(-0.3) = -1$$

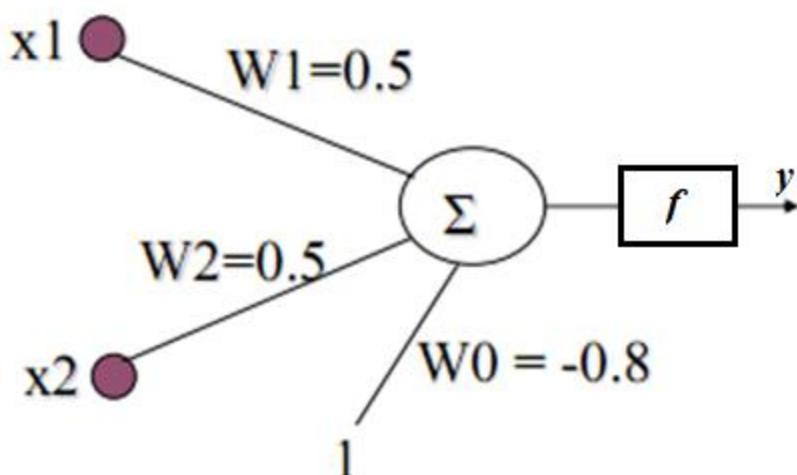
برای الگوی 3

$$y_3 = \text{Sign}(w x + b) = \text{Sign}([0.5 \ 0.5] \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 0.8) = \text{Sign}(-0.3) = -1$$

برای الگوی 4

$$y_4 = \text{Sign}(w x + b) = \text{Sign}([0.5 \ 0.5] \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0.8) = \text{Sign}(0.2) = 1$$

همانطور که در بالا مشاهده شد شبکه قادر به شناسایی تمامی الگوهای ورودی است



# شبکه های عصبی پرسپترون

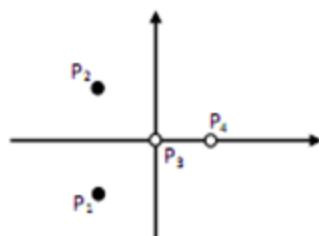
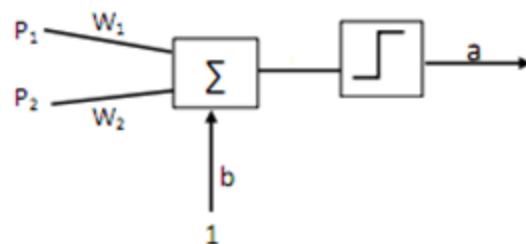
مثال:

✓ شبکه پرسپترون را برای ورودی های زیر بررسی کنید.

$$\{P_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_1 = 1\} \quad \{P_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_3 = -1\}$$

$$\{P_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_2 = 1\} \quad \{P_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_4 = -1\}$$

$$w_1 = -1 \quad w_2 = 0 \quad b = -0.5$$



برای الگوی 1

$$a = \text{Sign}(w P_1 + b) = \text{Sign}\left(\begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} - 0.5\right) = \text{Sign}(0.5) = 1$$

برای الگوی 2

$$a = \text{Sign}(w P_2 + b) = \text{Sign}\left(\begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} - 0.5\right) = \text{Sign}(0.5) = 1$$

برای الگوی 3

$$a = \text{Sign}(w P_3 + b) = \text{Sign}\left(\begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.5\right) = \text{Sign}(-0.5) = -1$$

برای الگوی 4

$$a = \text{Sign}(w P_4 + b) = \text{Sign}\left(\begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 0.5\right) = \text{Sign}(-1.5) = -1$$

همانطور که در بالا مشاهده شد شبکه قادر به شناسایی تمامی الگوهای ورودی است

# شبکه های عصبی پرسپترون

## یادگیری پرسپترون

- مرحله ۰ - مقداردهی اولیه به وزنها و بایاس (مقدار صفر)

تعیین نرخ یادگیری  $\alpha \leq 0$  (مقدار ۱)

- مرحله ۱ - تا زمانی که شرایط توقف برقرار نیست، مراحل ۲ تا ۶ را انجام دهید
- مرحله ۲ - انجام مراحل ۳ تا ۵ برای هر جفت داده آموزش  $s:t$
- مرحله ۳ - فعال سازی های واحد های ورودی را مشخص کنید:  $x_i = s_i$
- مرحله ۴ - پاسخ واحد خروجی را محاسبه کنید:

$$y_{in} = b + \sum_i x_i w_i \quad \Rightarrow \quad y = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

- مرحله ۵ - اگر خطایی رخ داده است، وزنها و بایاس را به روز کنید.

اگر  $y \neq t$  باشد، آنگاه:

$$w_i^{(new)} = w_i^{(old)} + \alpha x_i t$$

$$b^{(new)} = b^{(old)} + \alpha t$$

$$w_i^{(new)} = w_i^{(old)}$$

$$b^{(new)} = b^{(old)}$$

در غیر این صورت:

- مرحله ۶ - شرایط توقف را آزمایش کنید:

اگر در مرحله ۲ هیچ وزنی تغییر نکرد، الگوریتم را متوقف کنید، در غیر این صورت ادامه دهید

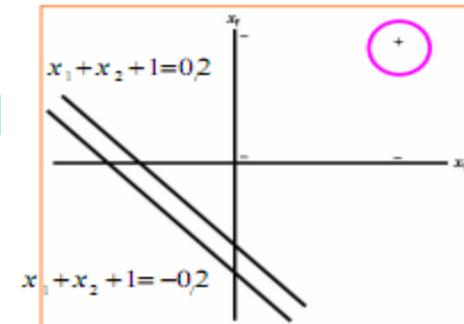
# شبکه های عصبی پرسپترون

INPUT	TARGET
( $x_1 \ x_2 \ 1$ )	$t$
(1 1 1)	1
(1 0 1)	-1
(0 1 1)	-1
(0 0 1)	-1

- مثال: تابع AND با ورودی های دودویی و هدف های دو قطبی

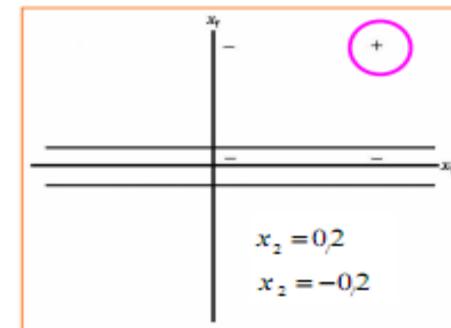
• ارائه ورودی اول

INPUT	NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS
				$(\Delta w_1 \ \Delta w_2 \ \Delta b)$	$(w_1 \ w_2 \ b)$		
( $x_1 \ x_2 \ 1$ )	$y\_in$	$y$	$t$	(1 1 1)	(0 0 0)		
(1 1 1)	0	0	1	(1 1 1)	(1 1 1)		



• ارائه دومین ورودی

INPUT	NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS
				$(\Delta w_1 \ \Delta w_2 \ \Delta b)$	$(w_1 \ w_2 \ b)$		
( $x_1 \ x_2 \ 1$ )	$y\_in$	$y$	$t$	(1 1 1)	(0 1 0)		
(1 0 1)	2	1	-1	(-1 0 -1)	(0 1 0)		



# شبکه های عصبی پرسپترون

INPUT	TARGET
( $x_1 \ x_2 \ b$ )	$t$
(1 1 1)	1
(1 0 1)	-1
(0 1 1)	-1
(0 0 1)	-1

- مثال: تابع AND با ورودی های دودویی و هدف های دو قطبی

• ارائه سومین ورودی

INPUT	NET	OUT	TARGET	WEIGHT		
				CHANGES	WEIGHTS	
( $x_1 \ x_2 \ b$ )	$y\_in$	$y$	$t$	( $\Delta w_1 \ \Delta w_2 \ \Delta b$ )	( $w_1 \ w_2 \ b$ )	
(0 1 1)	1	1	-1	(0 -1 -1)	(0 0 -1)	

• ارائه چهارمین ورودی

○ با توجه به برابر بودن پاسخ شبکه و مقدار هدف، وزن ها تغییری نمی کنند

INPUT	NET	OUT	TARGET	WEIGHT		
				CHANGES	WEIGHTS	
( $x_1 \ x_2 \ b$ )	$y\_in$	$y$	$t$	( $\Delta w_1 \ \Delta w_2 \ \Delta b$ )	( $w_1 \ w_2 \ b$ )	
(0 0 1)	-1	-1	-1	(0 0 0)	(0 0 -1)	

کامل شدن اولین دور  
(Epoch ) آموزش

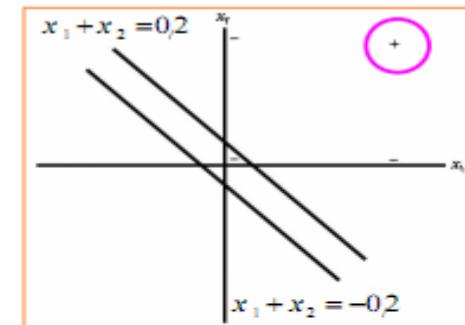
# شبکه های عصبی پرسپترون

INPUT	TARGET
( $x_1$ $x_2$ 1)	$t$
(1   1   1)	1
(1   0   1)	-1
(0   1   1)	-1
(0   0   1)	-1

- مثال: تابع AND با ورودی های دودویی و هدف های دو قطبی

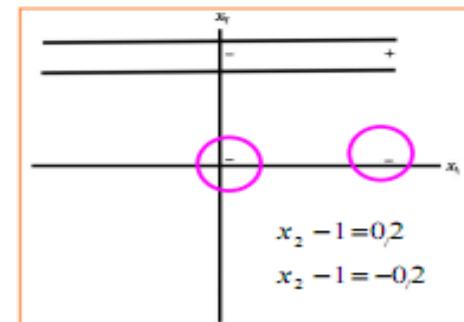
- دومین دور آموزش - ارائه اولین ورودی

INPUT	NET	OUT	TARGET	WEIGHT CHANGES	WEIGHTS
( $x_1$ $x_2$ 1)	$y_{in}$	$y$	$t$	$(\Delta w_1 \quad \Delta w_2 \quad \Delta b)$	$(w_1 \quad w_2 \quad b)$
(1   1   1)	-1	-1	1	(1   1   1)	(0   0   -1)



- دومین دور آموزش - ارائه دومین ورودی

INPUT	NET	OUT	TARGET	WEIGHT CHANGES	WEIGHTS
( $x_1$ $x_2$ 1)	$y_{in}$	$y$	$t$	$(\Delta w_1 \quad \Delta w_2 \quad \Delta b)$	$(w_1 \quad w_2 \quad b)$
(1   0   1)	1	1	-1	(-1   0   -1)	(0   1   -1)



# شبکه های عصبی پرسپترون

INPUT	TARGET
( $x_1 \ x_2 \ b$ )	$t$
(1 1 1)	1
(1 0 1)	-1
(0 1 1)	-1
(0 0 1)	-1

- مثال: تابع AND با ورودی های دودویی و هدف های دو قطبی

- دومین دور آموزش - ارائه سومین ورودی

پاسخ برای تمام ورودی ها منفی

INPUT	NET	OUT	TARGET	WEIGHT		
				CHANGES	WEIGHTS	
( $x_1 \ x_2 \ b$ )	$y\_in$	$y$	$t$	( $\Delta w_1 \ \Delta w_2 \ \Delta b$ )	( $w_1 \ w_2 \ b$ )	
(0 1 1)	0	0	-1	(0 -1 -1)	(0 1 -1)	(0 0 -2)

- دومین دور آموزش - ارائه چهارمین ورودی

پاسخ برای تمام ورودی ها منفی

INPUT	NET	OUT	TARGET	WEIGHT		
				CHANGES	WEIGHTS	
( $x_1 \ x_2 \ b$ )	$y\_in$	$y$	$t$	( $\Delta w_1 \ \Delta w_2 \ \Delta b$ )	( $w_1 \ w_2 \ b$ )	
(0 0 1)	-2	-1	-1	(0 0 0)	(0 0 -2)	

کامل شدن دومین دور  
آموزش (Epoch)

# شبکه های عصبی پرسپترون

INPUT	TARGET
$(x_1 \ x_2 \ 1)$	$t$
(1 1 1)	1
(1 0 1)	-1
(0 1 1)	-1
(0 0 1)	-1

- مثال: تابع AND با ورودی های دودویی و هدف های دو قطبی

• پنجمین، ششمین، .... دور آموزش

• نهمین دور آموزش

• دهمین دور آموزش

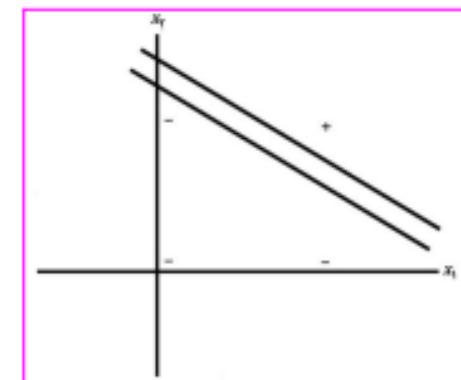
◦ عدم تغییر وزنها = توقف الگوریتم

◦ همگرایی وزنها

$$\begin{array}{ccccccc} (1 & 1 & 1) & 0 & 0 & 1 & (1 & 1 & 1) & (3 & 3 & -3) \\ (1 & 0 & 1) & 0 & 0 & -1 & (-1 & 0 & -1) & (2 & 3 & -4) \\ (0 & 1 & 1) & -1 & -1 & -1 & (0 & 0 & 0) & (2 & 3 & -4) \\ (0 & 0 & 1) & -4 & -1 & -1 & (0 & 0 & 0) & (2 & 3 & -4) \end{array}$$

$$\begin{array}{ccccccc} (1 & 1 & 1) & 1 & 1 & 1 & (0 & 0 & 0) & (2 & 3 & -4) \\ (1 & 0 & 1) & -2 & -1 & -1 & (0 & 0 & 0) & (2 & 3 & -4) \\ (0 & 1 & 1) & -1 & -1 & -1 & (0 & 0 & 0) & (2 & 3 & -4) \\ (0 & 0 & 1) & -4 & -1 & -1 & (0 & 0 & 0) & (2 & 3 & -4) \end{array}$$

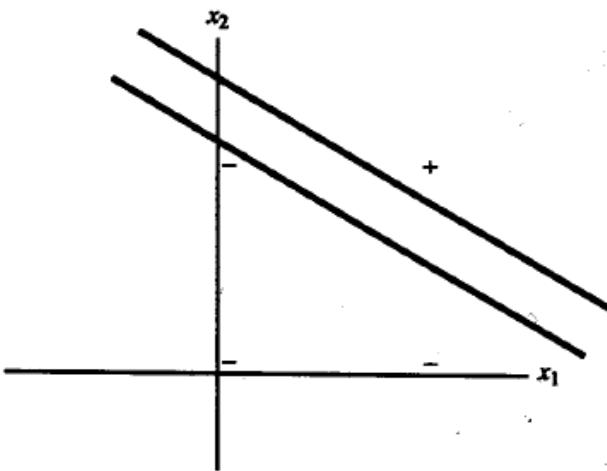
$$\begin{cases} 2x_1 + 3x_2 - 4 > 0,2 \Rightarrow x_2 = -\frac{2}{3}x_1 + \frac{7}{5} \\ 2x_1 + 3x_2 - 4 < -0,2 \Rightarrow x_2 = -\frac{2}{3}x_1 + \frac{19}{15} \end{cases}$$



# شبکه های عصبی پرسپترون

## • همگرایی پرسپترون

- ✓ در صورتی که مجموعه وزن های  $W^*$  وجود داشته باشد که قابلیت جداسازی یک مجموعه محدود (جدایی پذیر خطی) را داشته باشد، قانون آموزش پرسپترون به یک پاسخ همگرا خواهد شد.
- ✓ بنابراین قانون یادگیری پرسپترون به بردار وزنی نزدیک می شود که برای تمامی الگوهای آموزش پاسخ صحیحی می دهد و این کار در مراحلی با تعداد متناهی انجام می شود.

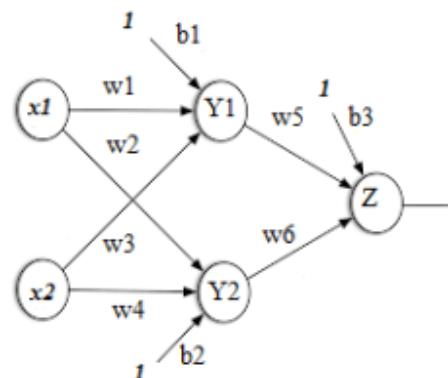


# شبکه های عصبی پرسپترون

## • حل XOR با اتصال چند تا نورون پرسپترون به هم

سه بار اجرای الگوریتم پرسپترون:

- یک بار برای نورون and (نورون Y1)
- یک بار برای نورون or (نورون Y2)
- یک بار برای نورون خروجی (نورون Z)



جدول:					
x1	x2	Y1	Y2	Z	
0	0	0	0	0	
0	1	0	1	1	
1	0	0	1	1	
1	1	1	1	0	

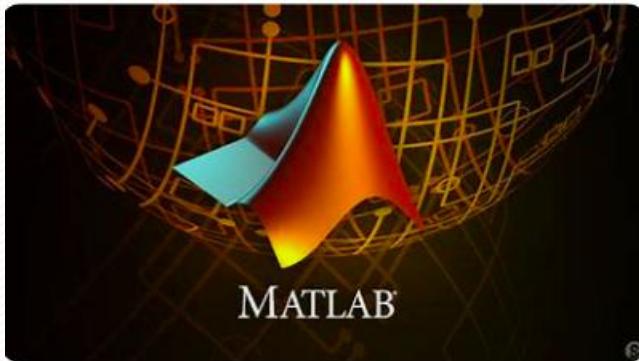
پارامترها:

وزن اولیه = 0	$\theta = 0$	$\alpha = 1$	کدینگ باپلار / باینری
---------------	--------------	--------------	-----------------------

# شبیه سازی

## • متلب

- جعبه ابزار شبکه عصبی که یکی از پر کاربردی و مفیدترین جعبه ابزار Neural Network Toolbox نرم افزار MATLAB می باشد.



## • پایتون



- زبان برنامه نویسی پایتون.
- سرویس کولب Google Colab
- محیط شبیه Jupyter Notebook
- ❖ اجرای آنلاین روی سرور گوگل
- ❖ دسترسی به GPU و TPU رایگان
- ❖ اتصال به Google Drive

# گسترش پرسپترون

## ● آدالاین (Adaptive Linear Neuron=ADALINE)

✓ توسط ویدرو- هاف در سال ۱۹۶۰

✓ دارای قانون یادگیری متفاوت با هب و پرسپترون

✓ قانون یادگیری با نام قانون (Widrow-Hoff Rule) یا میانگین مربعات کمینه (LMS=Least Mean Square). (Delta Rule) و یا قانون دلتا

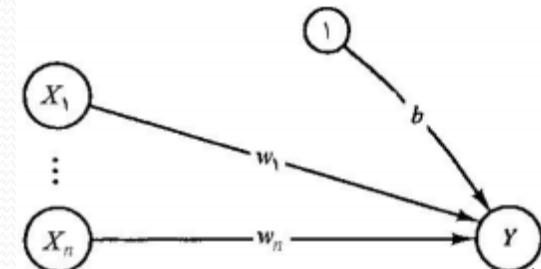
## ● تفاوت قانون پرسپترون و قانون دلتا

✓ در پرسپترون برای هر واحدی که پاسخ نادرست دارد، وزن های اتصال آن واحد تنظیم می شود، اما قانون دلتا وزن ها را طوری تنظیم می کند تا اختلاف بین خروجی شبکه و خروجی مطلوب کمینه شود.

✓ قانون دلتا منجر به افزایش قابلیت تعمیم (Generalization) شبکه می شود.

✓ قانون دلتا مبنای قانون پس انتشار (Back Propagation) برای یادگیری شبکه های چند لایه است.

# آدالاین



- ساختاری مشابه با شبکه های عصبی پرسپترون
  - ✓ این روش همانند پرسپترون می تواند مسائل خطی جدایی پذیر را حل کند.
  - ✓ تابع فعال سازی در زمان آموزش یک تابع خطی است و در فاز استفاده پله ای است.

- قانون ویدرو - هاف:
  - ✓ برای تولید وزن های جدید از تاثیر خطا استفاده می شود.
  - ✓ میزان بروز رسانی متناسب با میزان خطا، نتیجه همگرایی سریع تر در پی خواهد داشت.
  - ✓ این قانون وزن ها و بایاس را به گونه های تغییر می دهد که میانگین مربعات خطا (بین خروجی مطلوب و خروجی واقعی) سیستم را به حداقل برساند.

# آدالاین

## ● الگوریتم آموزش

- مرحله ۰ - مقداردهی اولیه به وزن‌ها (مقادیر تصادفی کوچک)
- مرحله ۱ - تا زمانی که شرایط توقف برقرار نیست، مراحل ۲ تا ۶ را انجام دهید.
- مرحله ۲ - برای هر جفت آموزش دوقطبی  $s_t : t$ ، مراحل ۳ تا ۵ را انجام دهید.
- مرحله ۳ - فعال‌سازی‌های واحدهای ورودی را مشخص کنید:
$$x_i = s_i \quad i = 1, \dots, n$$
- مرحله ۴ - مقدار ورودی شبکه را به واحد خروجی محاسبه کنید:
$$y\_in = b + \sum_i x_i w_i$$
- مرحله ۵ - مقادیر وزن‌ها و بایاس را به روز کنید:
$$\begin{cases} b(new) = b(old) + \alpha \cdot (t - y\_in) \\ w_i(new) = w_i(old) + \alpha \cdot (t - y\_in) \cdot x_i \end{cases}$$
- مرحله ۶ - شرایط توقف را آزمایش کنید:  
اگر بزرگ‌ترین تغییر وزنی که در مرحله ۲ رخ داده است از یک مقدار کوچک کم‌تر باشد، الگوریتم را متوقف کنید، و غرنه ادامه دهید.

## آدالاین

- مثال: تابع AND با ورودی های دودویی و هدف هایی دو قطبی

همانطور که در تعریف قانون دلتا بیان شد، آدالاین به گونه ای طراحی می شود که وزن هایی را نتیجه دهد که خطای کل را به حداقل می رساند.

$$e = E \{(\hat{t} - t)^2\} = \sum_{p=1}^4 [ \{x_1(p)w_1 + x_2(p)w_2 + w_0\} - t(p) ]^2$$

شبکه بعد از آموزش ✓

$$w_1 = 1 \quad w_2 = 1 \quad w_0 = -\frac{3}{2}$$

$$x_1 + x_2 - \frac{3}{2} = 0$$

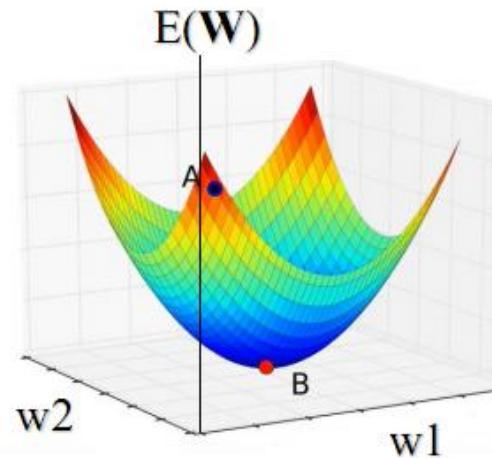
# آدالاین

## ● قانون دلتا

- کمینه کردن خطای بین خروجی شبکه و مقدار هدف متناظر

$$E = (t - y_{in})^2$$

$$y_{in} = \sum_{i=1}^n x_i w_i$$



- گرادیان تابع خطأ = مشتقهای جزئی خطأ نسبت به هر یک از وزن‌ها
- گرادیان بیانگر جهت سریع‌ترین رشد خطأ
- جهت مخالف گرادیان = سریع‌ترین کاهش خطأ

$$\frac{\partial E}{\partial w_i} = -2(t - y_{in}) \frac{\partial y_{in}}{\partial w_i} = -2(t - y_{in})x_i$$

$$\Delta w_i = \alpha(t - y_{in})x_i$$

## آدالاین

### ● تنظیم نرخ یادگیری (آموزش)

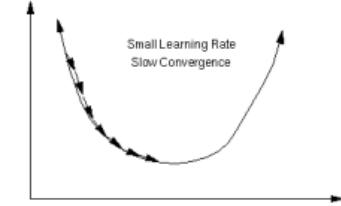
- ✓ تاثیر بر سرعت و روند همگرایی الگوریتم
- ✓ نیاز به اختصاص مقدار مناسب
- ✓ دارای کران بالا از نظر تئوری

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

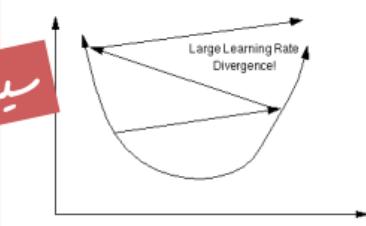
✓ نرخ آموزش بزرگ: همگرایی تند است ولی روند حرکت با تغییرات زیاد (موجب ناپایداری)

❑ روش مناسب تطبیقی: ابتدا با گام های بزرگتر حرکت صورت بگیرد و در نزدیکی جواب مقادیر کوچکتر شوند.

همگرایی کند است.

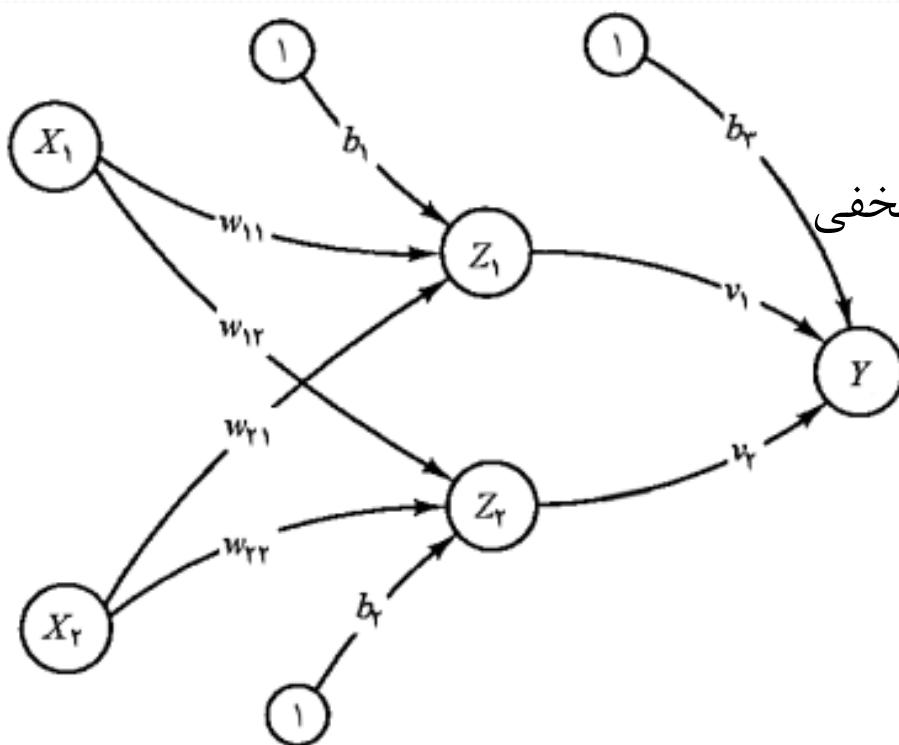


سیستم ناپایدار است.



# مادلاین (Multilayer ADALINE)

- حالت چند لایه شبکه آدالاین (شکل توسعه یافته آدالاین)
  - ✓ ترکیب چندین واحد آدالاین در یک شبکه با هم
  - ✓ افزایش قابلیت های محاسباتی شبکه = حل مسائل پیچیده تر

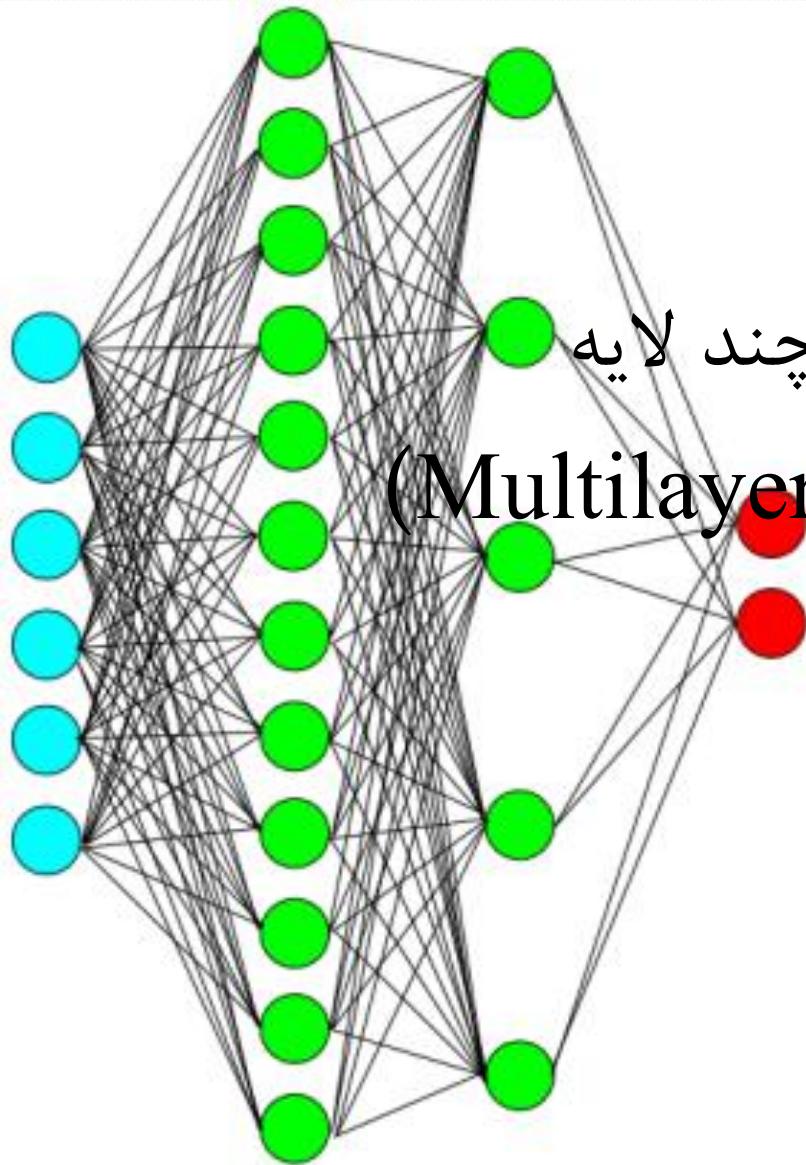


- ساختار
  - ✓ شامل یک لایه مخفی با دو واحد آدالاین مخفی
  - ✓ شامل یک واحد خروجی آدالاین

- الگوریتم آموزش
  - MRI ✓
  - MRII ✓

## پرسپترون چند لایه

- محدودیت پرسپترون های یک لایه در سال ۱۹۶۹ توسط مینسکی و پاپرت در کتاب آنها به چاپ رسید. نتایج مطالعه این کتاب باعث شد که شبکه های عصبی به مدت دو دهه، کمتر مورد توجه قرار گیرند.
- ✓ عدم موفقیت پرسپترون های یک لایه در حل مسائلی ساده ای (مانند تابع XOR)
- ✓ اواخر دهه ۶۰ و اوایل دهه ۷۰ - سال های خاموش
- الگوریتم SOM (Kohonen Self-Organizing Map) در سال ۱۹۷۲
- الگوریتم ART(Adaptive Resonance Theory) در سال ۱۹۷۶
- با کشف الگوریتم پس انتشار توسط روملهات، هینتن و ویلیامز در سال ۱۹۸۶ مطالعات جدید بر روی شبکه های عصبی مجدداً شروع شد. اهمیت ویژه این الگوریتم این بود که شبکه های عصبی چند لایه توسط آن می توانستند آموزش داده شوند.
- ✓ دهه ۸۰ - شکوفایی شبکه های عصبی



شبکه های عصبی چند لایه

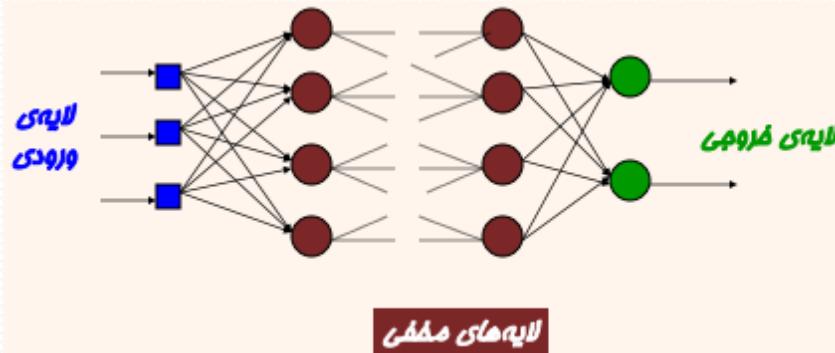
(Multilayer Neural Networks)

## شبکه های عصبی چند لایه

- بر خلاف پرسپترون ها، شبکه های چند لایه می توانند برای یادگیری مسائل غیرخطی و همچنین مسائلی با تصمیم گیری های متعدد بکار روند.
- شبکه های چند لایه از یک لایه ورودی، یک لایه خروجی و یک یا چند لایه بین آنها (لایه پنهان) که مستقیماً به داده های ورودی و نتایج خروجی متصل نیستند تشکیل یافته اند.
- واحدهای لایه ورودی صرفاً وظیفه توزیع مقادیر ورودی را به لایه بعد برعهده دارند و هیچ گونه تاثیری بر روی سیگنال های ورودی ندارند. به همین دلیل در شمارش تعداد لایه ها به حساب نیامده اند. شبکه شامل یک لایه خروجی است که پاسخ سیگنال های ورودی را ارائه می دهد. که تعداد نورون ها در لایه ورودی و لایه خروجی برابر با تعداد ورودی ها و خروجی ها می باشد و لایه یا لایه های پنهان وظیفه ارتباط دادن لایه ورودی به لایه خروجی را بر عهده دارند. شبکه با داشتن این لایه های پنهان قادر می گردد که روابط غیر خطی را از داده های ارائه شده به شبکه استخراج کند.

# شبکه های عصبی چند لایه

## ● Multilayer Perceptron (MLP)

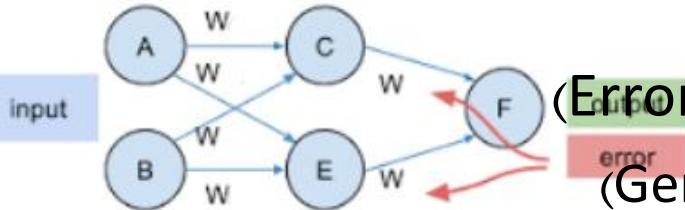


- ✓ ورودی ها به صورت مستقیم به خروجی متصل نیستند.
- ✓ هر واحد از لایه قبلی به تمامی واحدهای لایه بعدی متصل است(وزن صفر مجاز است).
- ✓ تعداد واحدهای مخفی مشخص است.
- ✓ تمام اتصالات روبه جلو است .
- ✓ تابع فعال سازی تابعی غیر خطی است.

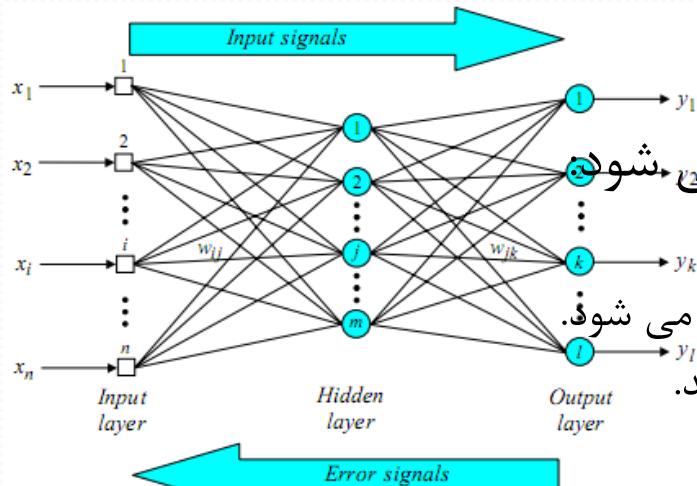
**Feed Forward**

# شبکه های عصبی چند لایه

- آموزش با پس انتشار خطا (Error Back Propagation) (Generalized Delta Rule)
  - ✓ قانون دلتای تعمیم یافته
  - ✓ کاهش گرادیان برای به حداقل رساندن کل مربعات خطای خروجی (مبنای ریاضی الگوریتم پس انتشار)
- هدف آموزش شبکه با پس انتشار، رسیدن به تعادل بین قابلیت یادگیری و تعمیم است
  - ✓ قابلیت یادگیری
  - پاسخ گویی صحیح به الگوهای ورودی به کار رفته برای آموزش
  - ✓ تعمیم
  - پاسخ دهنده منطقی (خوب) به ورودی های شبیه اما نه دقیقاً یکسان به ورودی های آموزش



# شبکه های عصبی چند لایه



- آموزش شامل سه مرحله است و به طریقه زیر عمل می شود
  - ✓ روبره جلو (Forward)

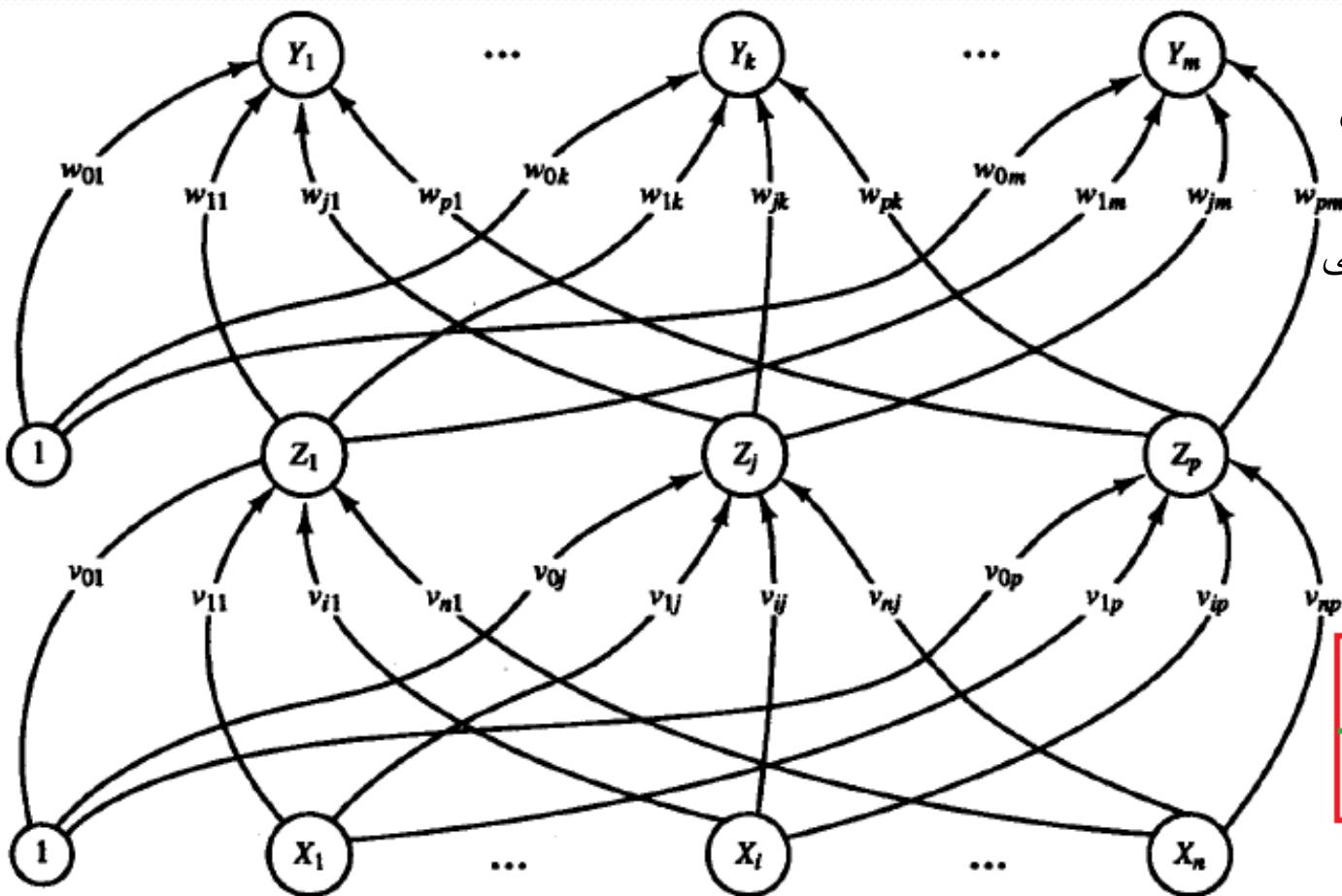
- بردار ورودی به شبکه اعمال شده و خروجی واقعی محاسبه می شود.
- در این مسیر، پارامترهای شبکه ثابت و بدون تغییر می مانند.
- ✓ روبره عقب (Backward)

- خطا (خروجی واقعی - خروجی مطلوب) محاسبه شده و بر حسب تابع معیار، سیگنالی متناسب با خطای تولید می شود. این سیگنال لایه لایه حرکت کرده و پس انتشار می شود.
  - ✓ تنظیم وزن ها
  - وزن ها با توجه به خطای خروجی تا لایه ورودی اصلاح می شود.

- پس از اصلاح وزن ها (بعد از ارائه تمامی الگوهای ورودی) می گوییم یک Epoch صورت گرفته است.
- وزن ها به گونه ای اصلاح می شوند که میانگین مجموع مربعات خطای کمینه گردد.

# شبکه های عصبی چند لایه

- الگوریتم یادگیری پس انتشار خطأ



- شبکه سه لایه
- یک لایه ورودی
- یک لایه مخفی
- یک لایه خروجی

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

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

# شبکه های عصبی چند لایه

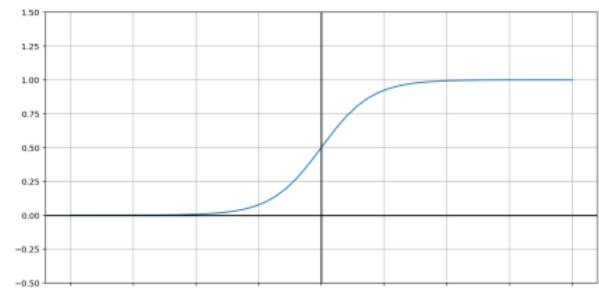
- الگوریتم یادگیری پس انتشار خطأ

- تابع فعال سازی ✓

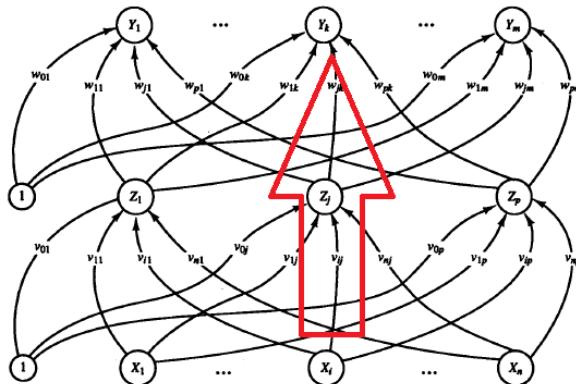
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\begin{aligned}\sigma'(z) &= \frac{0 - (-e^{-z})}{(1 + e^{-z})^2} = \frac{e^{-z}}{(1 + e^{-z})^2} \\ &= \frac{1 + e^{-z} - 1}{(1 + e^{-z})^2} = \frac{\cancel{1 + e^{-z}}}{\cancel{(1 + e^{-z})^2}} - \frac{1}{(1 + e^{-z})^2}\end{aligned}$$

$$= \frac{1}{1 + e^{-z}} - \frac{1}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \left( 1 - \frac{1}{1 + e^{-z}} \right)$$



$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \quad \text{This will be helpful!}$$



## شبکه های عصبی چند لایه

- الگوریتم یادگیری پس انتشار خطاط

- مرحله ۰ - به وزن‌ها مقدار اولیه بدهید (مقادیر تصادفی کوچک را انتخاب کنید).
- مرحله ۱ - تا زمانی که شرایط توقف برقرار نیست، مراحل ۲ تا ۹ را انجام دهید.
- مرحله ۲ - برای هر جفت آموزش (مقادیر ورودی و هدف)، مراحل ۳ تا ۸ را انجام دهید.

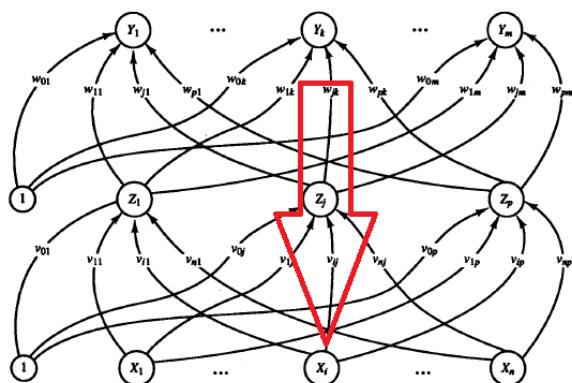
### پیش‌خور

- مرحله ۳ - ارسال سیگنال ورودی  $x_i$  به تمام واحدها در لایه بعدی (واحدهای مخفی)
- مرحله ۴ - محاسبه ورودی واحدهای مخفی و اعمال تابع فعال‌سازی

$$z\_in_j = v_{0j} + \sum_{i=1}^n x_i v_{ij} \quad z_j = f(z\_in_j)$$

- مرحله ۵ - محاسبه ورودی واحدهای خروجی و اعمال تابع فعال‌سازی

$$y\_in_k = w_{0k} + \sum_{j=1}^p z_j w_{jk} \quad y_k = f(y\_in_k)$$



## شبکه های عصبی چند لایه

- الگوریتم یادگیری پس انتشار خطا

### پس انتشار خطا

- مرحله ۶- محاسبه خطا برای واحدهای خروجی (استفاده از الگوی هدف)

$$\delta_k = (t_k - y_k) f'(y\_in_k)$$

محاسبه پارامتر تصحیح وزن (بعداً در بهروز کردن به کار می رود)

محاسبه پارامتر تصحیح بایاس (بعداً در بهروز کردن به کار می رود)

ارسال  $\delta_k$  (مقادیر دلتا) به واحدهای لایه قبلی (لایه مخفی)

- مرحله ۷- دریافت ورودی های دلتا توسط واحدهای مخفی از واحدهای خروجی

$$\delta\_in_j = \sum_{k=1}^m \delta_k w_{jk}$$

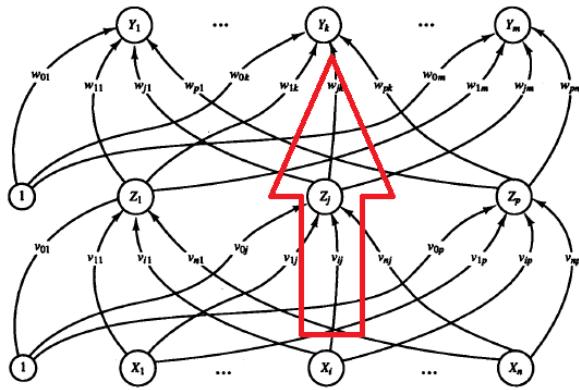
ضرب در مشتق تابع فعال سازی جهت محاسبه پارامتر مربوط به اطلاعات خطا

$$\delta_j = \delta\_in_j f'(z\_in_j)$$

محاسبه مقدار تصحیح وزن و بایاس (استفاده در بهروز کردن )

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

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



## شبکه های عصبی چند لایه

- الگوریتم یادگیری پس انتشار خطاطا

### بعد از آموزش

- فقط مرحله پیشخور مورد نیاز است

مرحله ۰: مقادیر وزن‌های شبکه را با استفاده از الگوریتم آموزش تعیین کنید.

مرحله ۱: برای هر بردار ورودی، مراحل ۲ تا ۴ را انجام دهید.

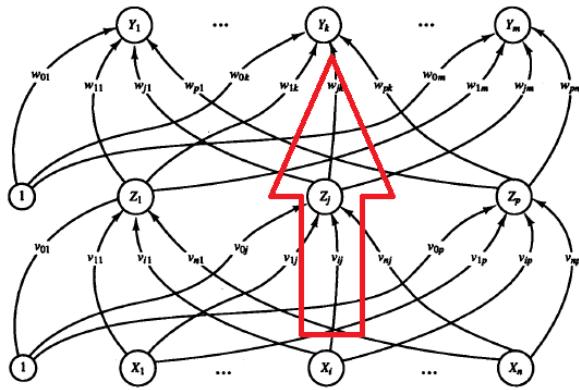
مرحله ۲: برای تمام نرون‌های ورودی، فعال‌سازی واحد ورودی را تعیین کنید.

مرحله ۳: برای واحدهای مخفی:

$$z\_in_j = v_{0j} + \sum_{i=1}^n x_i v_{ij} \Rightarrow z_j = f(z\_in_j)$$

مرحله ۴: برای واحدهای خروجی:

$$y\_in_k = w_{0k} + \sum_{j=1}^p z_j w_{jk} \Rightarrow y_k = f(y\_in_k)$$



## شبکه های عصبی چند لایه

- اثبات (Error Back Propagation)

- قانون دلتا...

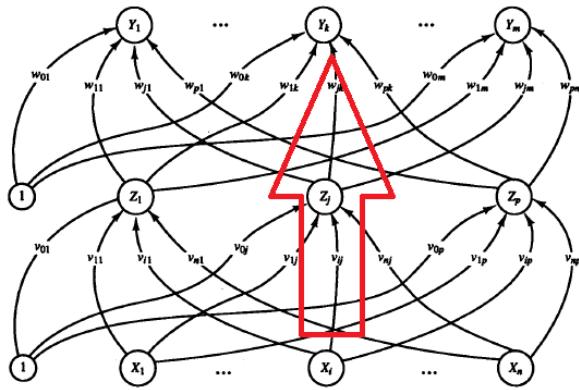
$$E = (t - y_{in})^2 \quad / \quad E = \frac{1}{2} \sum (t_i - y_{in})^2$$

$$y_{in} = \sum_{i=1}^n x_i w_i$$

$$\frac{\partial E}{\partial w_i} = -2(t - y_{in}) \frac{\partial y_{in}}{\partial w_i} = -2(t - y_{in})x_i$$

$$\Delta w_i = \alpha(t - y_{in})x_i$$

$$W_{i\ new} = W_{i\ old} + \alpha(t - y_{in})x_i$$



# شبکه های عصبی چند لایه

## اثبات (Error Back Propagation) •

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

$$\begin{aligned} \frac{\partial E}{\partial w_{jk}} &= \frac{1}{2} \times 2 \times (t_k - y_k) \times \frac{\partial}{\partial w_{jk}} \times (t_k - y_k) \\ &= (t_k - y_k) \times \frac{-\partial y_k}{w_{jk}} = -(t_k - y_k) \frac{\partial f(y_{-ink})}{\partial w_{jk}} \\ &= -(t_k - y_k) f'(y_{-ink}) \frac{\partial y_{-ink}}{\partial w_{jk}} \\ &= -(t_k - y_k) f'(y_{-ink}) z_j \end{aligned}$$

$$\Rightarrow \frac{\partial f(y_{-ink})}{\partial w_{jk}} = \frac{\partial f(w_{ok} + \sum(z_j w_{jk}))}{\partial w_{jk}}$$

یادآوری:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial x}$$

$$\frac{\partial f(u)}{\partial x} = f'(u) \cdot \frac{\partial u}{\partial x}$$

$$\frac{\partial f(g(x))}{\partial x} = f'(x) \cdot g'(x)$$

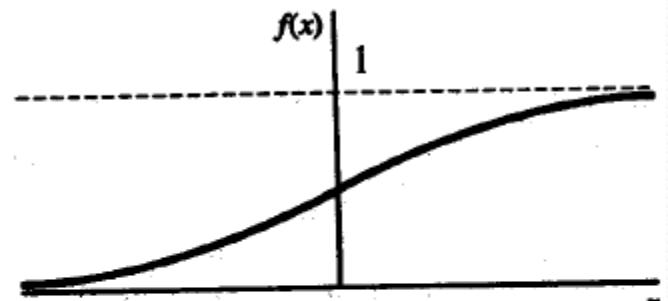
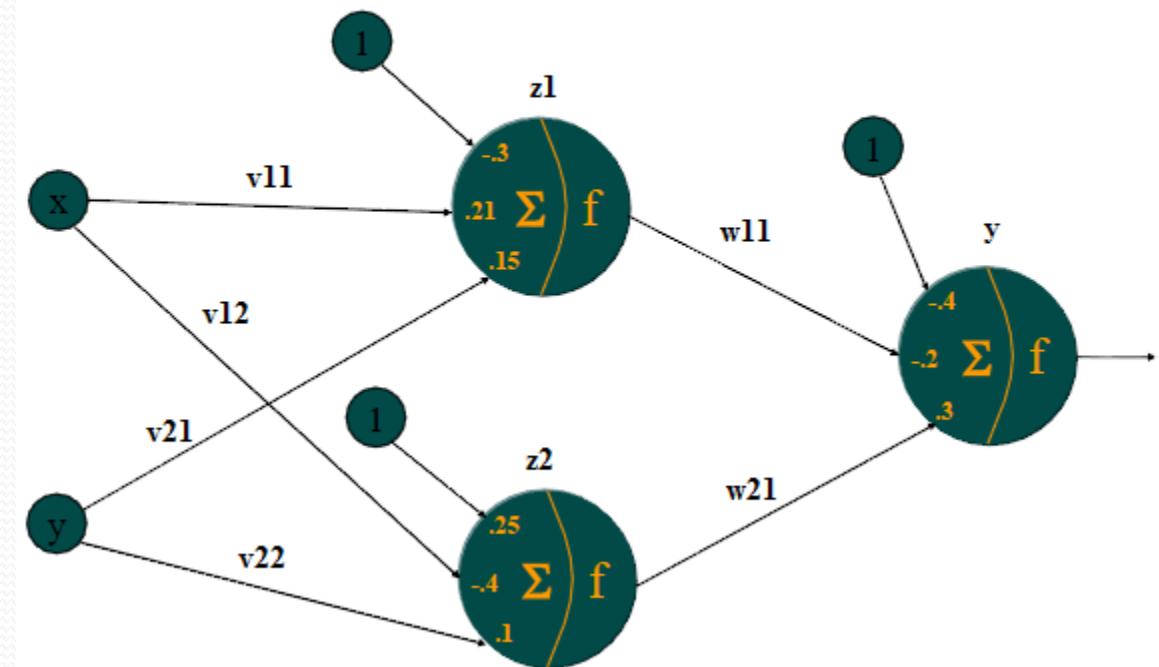
# شبکه های عصبی چند لایه

x	y	t
0	0	0
0	1	1
1	0	1
1	1	0

- الگوریتم یادگیری پس انتشار خطأ

مثال (تابع XOR ✓)

- مقدار دهی اولیه (تصادفی)



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

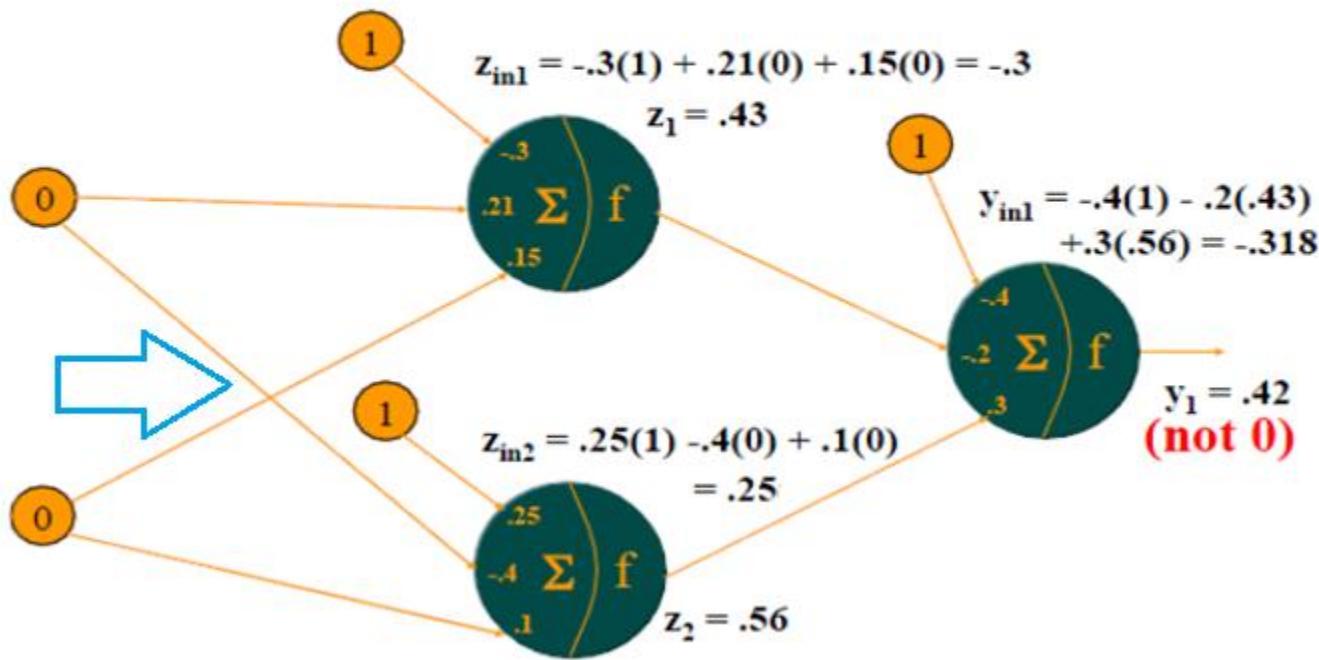
$$f'(x) = f(x)[1 - f(x)]$$

## شبکه های عصبی چند لایه

x	y	t
0	0	0
0	1	1
1	0	1
1	1	0

- الگوریتم یادگیری پس انتشار خطای مثال (تابع XOR ✓)

x	y	t	پیشخور کردن ورودی
0	0	0	



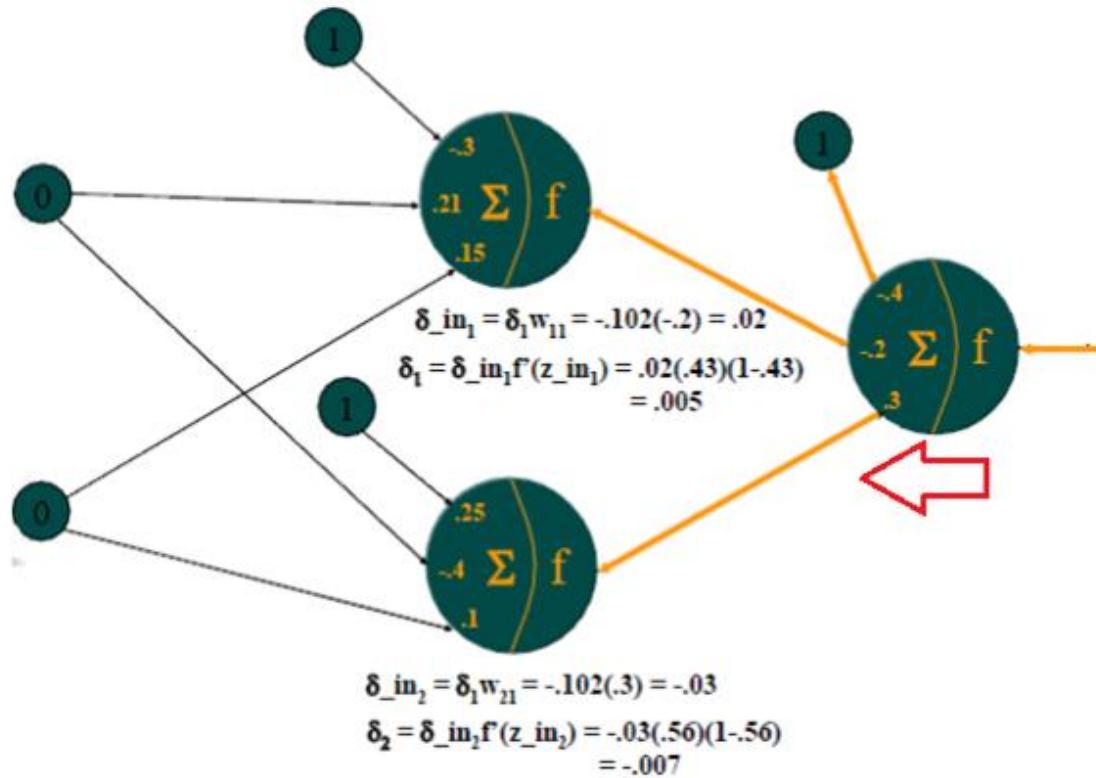
## شبکه های عصبی چند لایه

- الگوریتم یادگیری پس انتشار خطا

مثال (تابع XOR ✓

پس انتشار خطا •

x	y	t
0	0	0
0	1	1
1	0	1
1	1	0



$$\begin{aligned}\delta_1 &= (t_1 - y_1)f'(y_{in_1}) \\ &= (t_1 - y_1)[f(y_{in_1})][1-f(y_{in_1})] \\ &= (0 - .42).42[1-.42] = -.102\end{aligned}$$

# شبکه های عصبی چند لایه

$x$	$y$	$t$
0	0	0
0	1	1
1	0	1
1	1	0

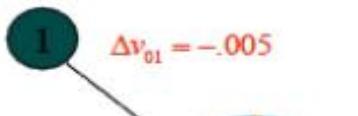
- الگوریتم یادگیری پس انتشار خطأ

مثال (تابع XOR ✓)

- محاسبه وزنها

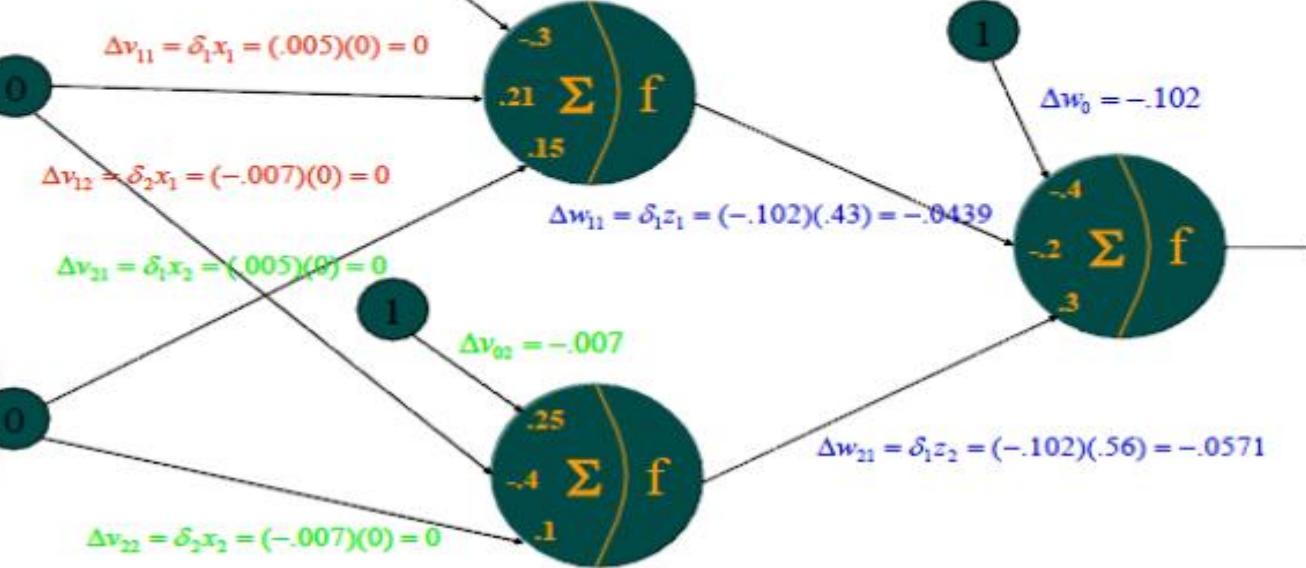
$$\Delta v_{ij} = \alpha \delta_j x_i \quad j = 1, 2$$

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



$$\Delta w_{j1} = \alpha \delta_1 z_j \quad j = 1, 2$$

$$\Delta w_0 = \alpha \delta_1$$



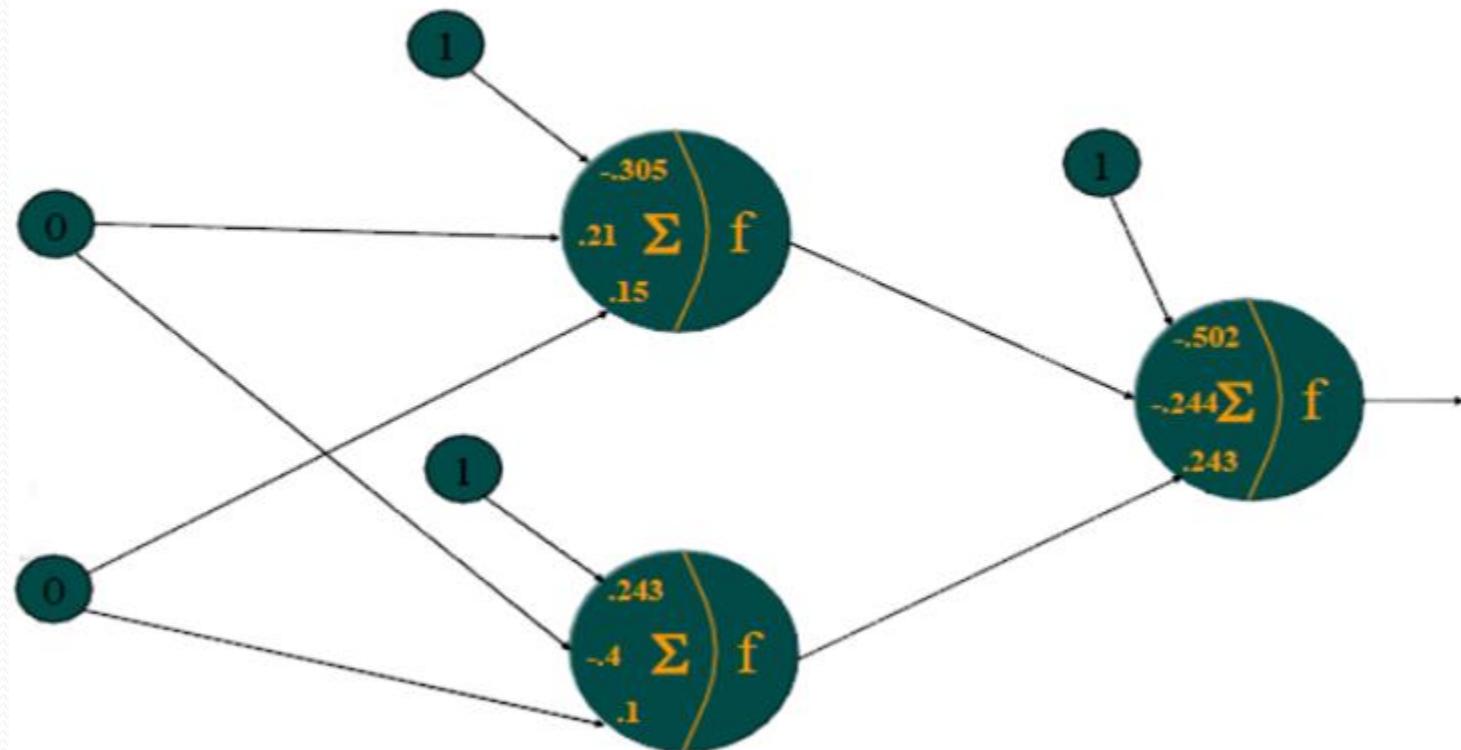
## شبکه های عصبی چند لایه

- الگوریتم یادگیری پس انتشار خطأ

مثال (تابع XOR ✓

بروز کردن وزنها

x	y	t
0	0	0
0	1	1
1	0	1
1	1	0

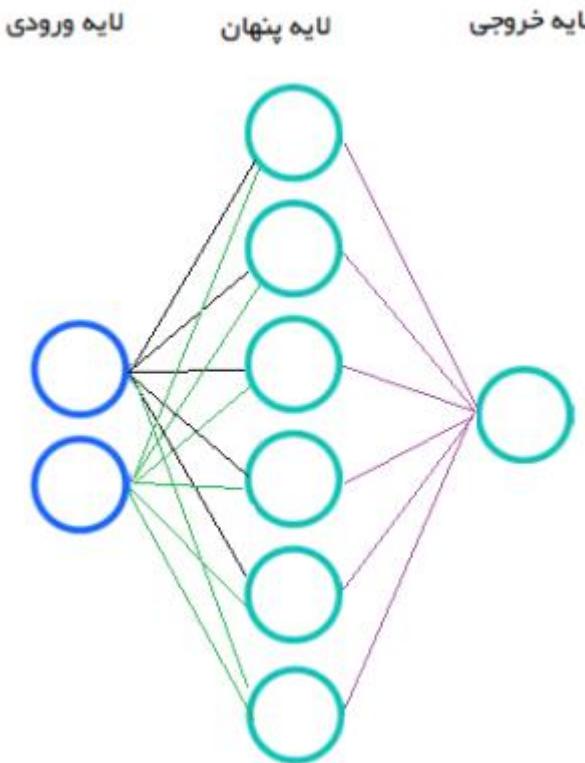


# شبکه های عصبی چند لایه

## HW 2

- از توپولوژی داده شده ، بهره گرفته و طبق قانون پس انتشار خطأ، وزن های شبکه را برای مجموعه داده های ارائه شده در فایل pdf بدست آورید.

- شبکه با یک لایه نورون مخفی
- مجموعه داده Jain



## شبکه های عصبی چند لایه

### ● شرط خاتمه الگوریتم BP

✓ معمولاً الگوریتم BP پیش از خاتمه هزاران بار با استفاده همان داده های آموزشی تکرار می گردد شروط مختلفی را می توان برای خاتمه الگوریتم بکار برد:

- توقف بعد از تکرار به دفعات معین
- توقف وقتی که خطا از یک مقدار تعیین شده کمتر شود.
- ❖ عیب این روش این است که فرآیند آموزش ممکن است طولانی شود.
- توقف وقتی که خطا در مثال های مجموعه اعتبار سنجی از قاعده خاصی پیروی نماید.

✓ اگر دفعات تکرار کم باشد خطا خواهیم داشت (مشکل Underfitting) و اگر زیاد باشد مشکل Overfitting رخ خواهد داد.

# شبکه های عصبی چند لایه

- الگوریتم یادگیری پس انتشار خطأ

- انواع شیوه های آموزش

## شیوه ترتیبی (Sample/Online Mode) ✓

- در این روش، نمونه‌ها یکی‌یکی به شبکه داده می‌شوند و اصلاح وزن‌ها صورت می‌پذیرد.

- مزیت: سریع واکنش نشون می‌دهد و می‌تواند در داده‌های بزرگ‌تر بهتر تعمیم پیدا کند.

- عیب: نویز (نوسانات) در بهروزرسانی وزن‌ها بیشتره.

## شیوه دسته‌ای (Batch Mode) ✓

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

- این روش همان Batch Gradient Descent است.

- مزیت: مسیر بهینه‌سازی یکنواخت‌تر.

- عیب: نیاز به حافظه زیاد و سرعت کمتر برای داده‌های بزرگ.

# شبکه های عصبی چند لایه

- الگوریتم یادگیری پس انتشار خطأ
- انواع شیوه های آموزش

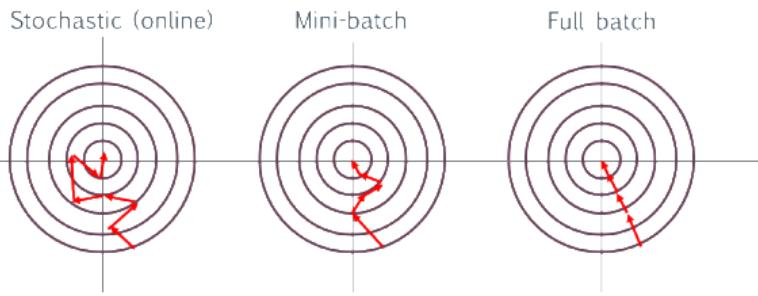
## شیوه Mini-Batch Mode ✓

در عمل، بیشتر پژوهش‌ها از Mini-Batch Gradient Descent استفاده می‌کنند.

- داده‌ها به دسته‌های کوچک (مثلًاً ۳۲ یا ۶۴ نمونه) تقسیم می‌شوند.
- بعد از پردازش هر دسته کوچک، وزن‌ها به روزرسانی می‌شوند.
- تعادل بین سرعت و دقیق.

- یک دوره کامل ارائه نمونه های آموزشی در فرآیند آموزش را epoch می‌نامند.
- یک بار ارائه یک batch را یک iteration می‌نامند.

اگر ۱۰۰۰ داده داشته باشیم و در هر مرحله ۱۰۰ داده به شبکه بدیم Mini-Batch = 100 در این صورت یک Epoch شامل ۱۰ بار به روزرسانی وزن‌ها خواهد بود.



# شبکه های عصبی چند لایه

## الگوریتم یادگیری پس انتشار خطأ

انتخاب مقادیر اولیه وزن ها ✓

□ تاثیر بر سرعت همگرایی شبکه

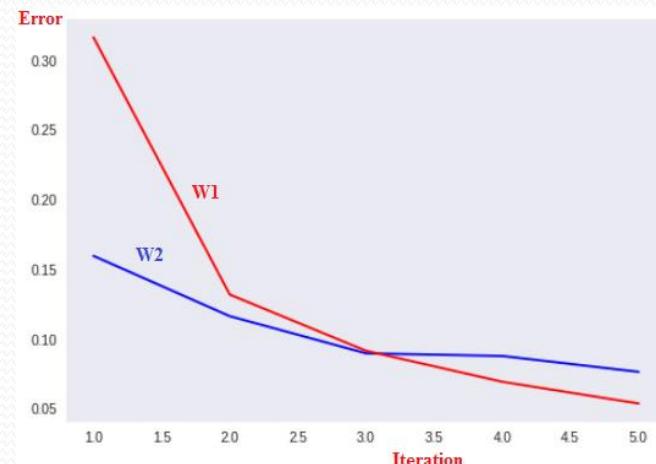
□ مقادیر می توانند مثبت و یا منفی باشند.

□ مقادیر وزن های اولیه نباید خیلی بزرگ و یا خیلی کوچک باشند.

❖ بازه متداول برای مقادیر وزن ها و بایاس ها بین 0.5 و -0.5 (یا بین -1 و +1)

❖ اگر وزن های اولیه خیلی کوچک باشند، ورودی شبکه به واحد مخفی یا به واحد خروجی به صفر نزدیک خواهد بود که موجب کند شدن یادگیری می شود.

❖ اگر وزن های اولیه خیلی بزرگ باشند، ورودی شبکه به واحد مخفی یا به واحد خروجی به ناحیه اشباع نزدیک خواهد بود که در آن مشتق تابع سیگموید مقدار بسیار کوچکی دارد.



# شبکه های عصبی چند لایه

## ورودی ها

الگوریتم های شبکه عصبی به پراکنده‌گی داده ها و تفاوت مقیاس داده ها حساسیت نشان داده و نتایج مناسبی ارائه نخواهند داد.

- Linear scaling to the interval [0,1]

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

- Standardization (making variable approx. std. normal)

- Linear scaling to the interval [-1,1]

$$x_i = 2 \left( \frac{x_i - \bar{x}}{x_{max} - x_{min}} \right) - 1$$

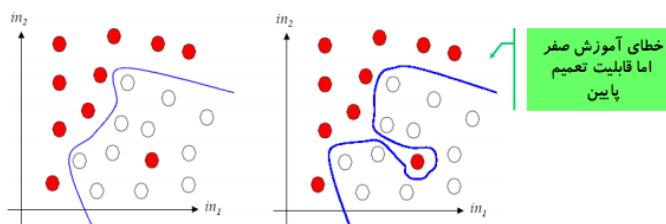
$$x_i = \frac{x_i - \bar{x}}{\sigma}; \quad \sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$$

# شبکه های عصبی چند لایه

- قدرت تعمیم و Overfitting
- هدف آموزش شبکه
  - ✓ تعادل بین یادگیری و تعمیم
  - پاسخ صحیح به الگوهای آموزش داده شده به شبکه و تولید پاسخ مناسب به ازای الگوهای جدید
  - شبکه قوانین حاکم بر داده ها را یاد بگیرد و نه فقط نمونه های آموزش
  - ادامه آموزش شبکه زمانی که مقدار مربعات خطا واقعاً حداقل شده، الزاماً مفید نمی باشد.  
Overfitting!)

## Overfitting •

- ✓ ناشی از تنظیم وزن ها برای در نظر گرفتن مثال های نادری است که ممکن است با توزیع کلی داده ها مطابقت نداشته باشند.

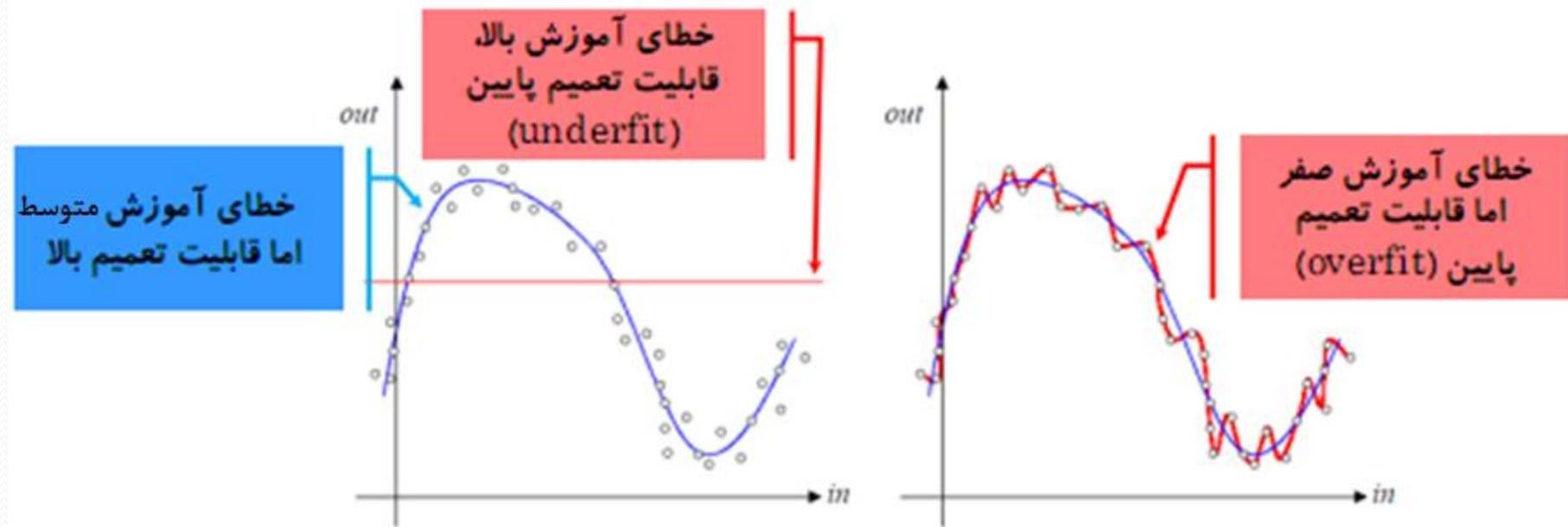


# شبکه های عصبی چند لایه

- قدرت تعمیم و Overfitting

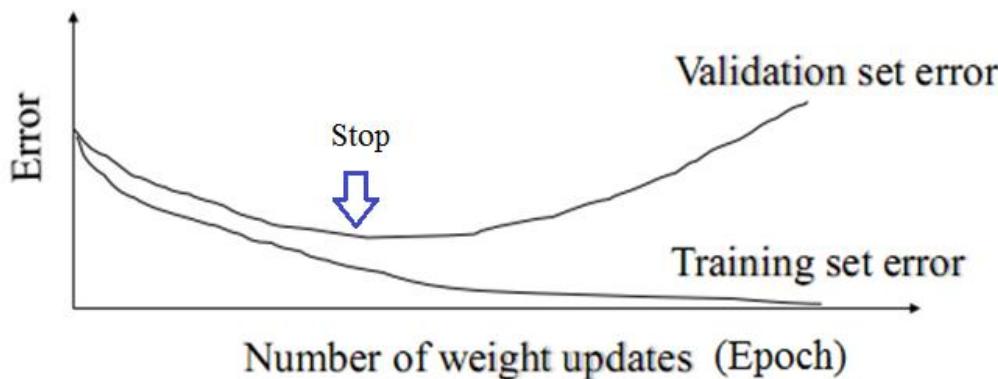
- قابلیت تعمیم در تقریب یک تابع فرضی

- بررسی Underfitting و Overfitting به ازای سه روند آموزش متفاوت



# شبکه های عصبی چند لایه

- قدرت تعمیم و Overfitting
- ✓ برای حل مشکل Overfitting
- ❑ یک مجموعه برای آموزش الگوها و یک مجموعه برای اعتبار سنجی (Validation)
- ❑ استفاده از روش K-fold Cross validation
- ❖ تقسیم داده آموزش به  $k$  زیر مجموعه و هر بار یکی از این زیر مجموعه ها برای اعتبارسنجی بهره گرفته شود.



# شبکه های عصبی چند لایه

- بهبود عملکرد الگوریتم پس انتشار

✓ روش بروز رسانی وزن ها (پس انتشار با گشتاور (Momentum))

□ تغییر وزن ترکیبی از گرادیان (شیب) فعلی و گرادیان قبلی است.

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

قانون تغییر وزن  
عبارت ممتنم

✓ نرخ یادگیری وفقی (Adaptive Learning Rate)

□ روش آموزش دسته ای

□ روش آموزش ترتیبی

✓ الگوریتم دلتا-بار-دلتا (Delta-Bar-Delta)

□ فراهم کردن نرخ یادگیری مخصوص برای هر وزن (نرخ یادگیری وابسته به وزن)

□ تغییر نرخ های یادگیری با پیش روی آموزش

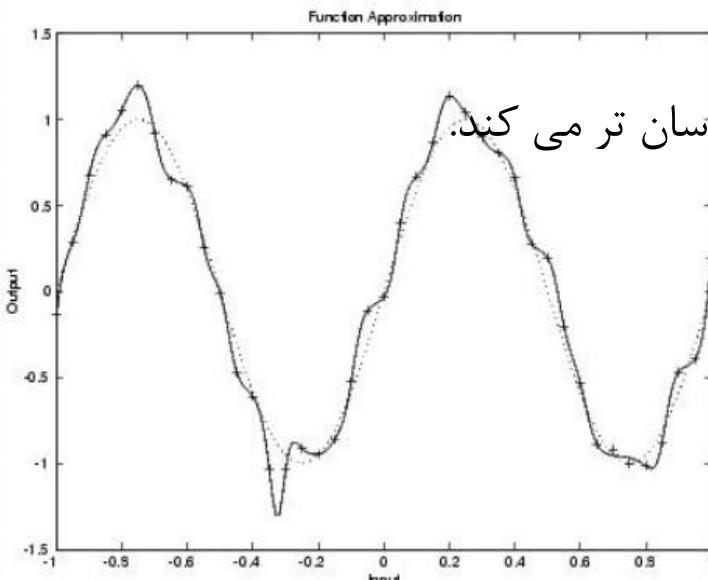
# شبکه های عصبی چند لایه

## • تقریب زننده های جهانی (Universal Approximator)

- ✓ تقریب زدن تابع پیوسته به عنوان یکی از کاربردهای شبکه های عصبی محسوب می شود.
- ✓ سوال: شبکه چند لایه تقریب تابع را با چه کیفیتی انجام می دهد؟

□ پاسخ: قضیه عصبی کولموفگروف (Kolmogorov)

- ❖ یک شبکه عصبی پیش خور با دو لایه نورون (واحدهای مخفی و واحدهای خروجی) می تواند هر تابع پیوسته ای را به صورت دقیق نمایش دهد.



✓

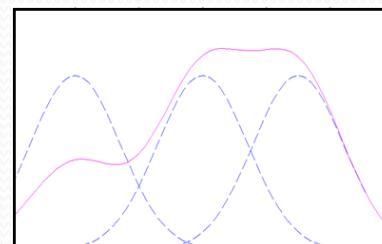
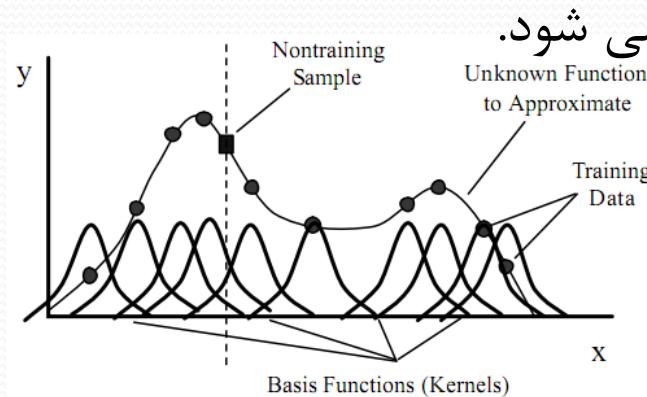
●

شبکه های عصبی مصنوعی شعاع مبنا

RBF(Radial Basis  
Function)

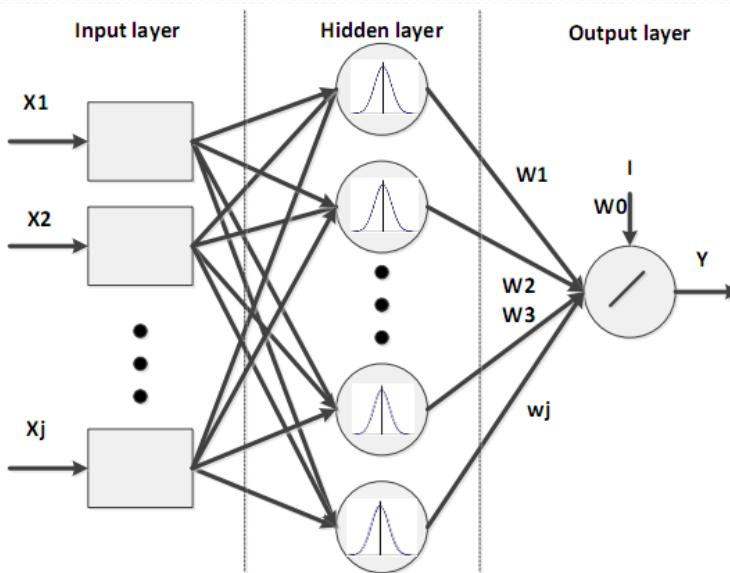
# Radial Basis Function

- ارائه شده توسط Powel در سال ۱۹۸۵
- با هدف حل مسائل درون یابی در توابع چند متغیره
- ساختاری شبیه به MLP دارند ولی با استراتژی متفاوتی عمل می کنند.
- تفاوت در توابع فعال سازی
- تفاوت در الگوریتم آموزش
- آموزش در این نوع شبکه ها بصورت Supervised است.
- از این شبکه برای دسته بندی و تقریب توابع بهره گرفته می شود.



# Radial Basis Function

- ساختار پایه دارای ۳ لایه است (Feed Forward).
- لایه نهان: تبدیل غیر خطی از فضای ورودی به فضای نهان را انجام می‌دهد.
- لایه خروجی: تبدیل خطی از فضای نهان به فضای خروجی اعمال می‌کند.



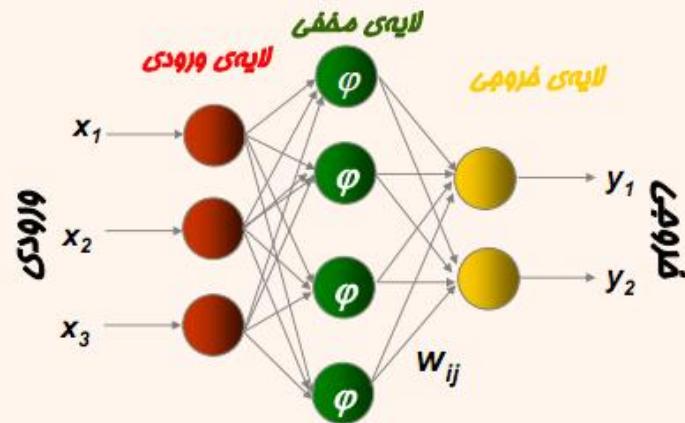
$$y = \sum_{i=1}^m w_i \Phi_i = w^T \Phi(x)$$

# Radial Basis Function

- لایه نهان یک تبدیل غیر خطی از فضای ورودی انجام می دهد. که معمولاً این تبدیل منجر به این می شود که فضای نهان دارای ابعاد بیشتری از فضای ورودی باشد.
- ابعاد فضای لایه نهان به دلیل دست یابی به فضای خطی تفکیک پذیر (طبق نظریه Cover) از ابعاد فضای ورودی معمولاً بیشتر است.
- ✓ احتمال اینکه یک مساله پیچیده طبقه بندی الگوها در فضا با بعد بیشتر خطی تفکیک پذیر باشد بیشتر است از حالتی که در فضای با بعد کمتر باشد ( قضیه Cover).
- لایه خروجی مانند شبکه های معمولی یک رگرسیون خطی بر روی خروجی های لایه نهان، به منظور دست یابی به خروجی مطلوب انجام می دهد.

# Radial Basis Function

- نکته: تابع فعال سازی نورون در لایه نهان غیرخطی و به شکل گاوسی است و به همین دلیل به آن ها RBF می گویند.
- شبکه های RBF بر تعریف تابعی وابسته به فاصله از مرکز استوار است.



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

مکان  
فاصله  
مرکز

- نکته: بنابراین در لایه مخفی، هر گره، تابعی شعاعی که مرکز و شعاعی مختص به خود را دارد، به ورودی ها اعمال می کند.

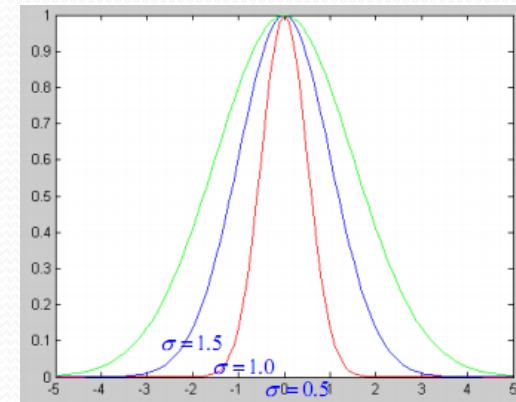
# Radial Basis Function

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

• چند نمونه از توابع فاصله:

- Gaussian:

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$$



- Multi-Quadratic:

$$\varphi(r) = (r^2 + c^2)^{\frac{1}{2}}$$

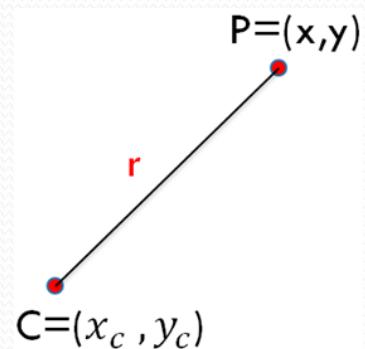
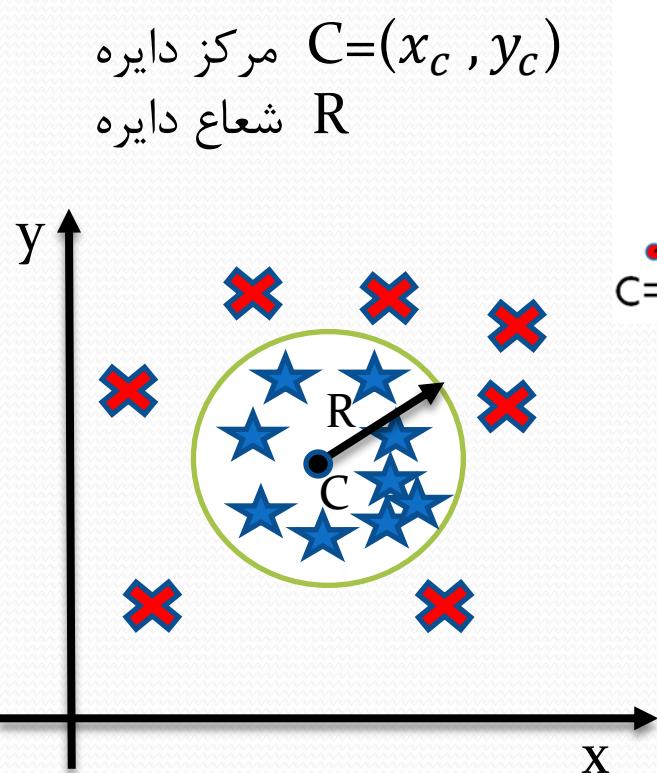
- Inverse Multi-Quadratic:

$$\varphi(r) = \frac{1}{(r^2 + c^2)^{\frac{1}{2}}}$$

$$r = \|\mathbf{x} - \mathbf{x}_i\|$$

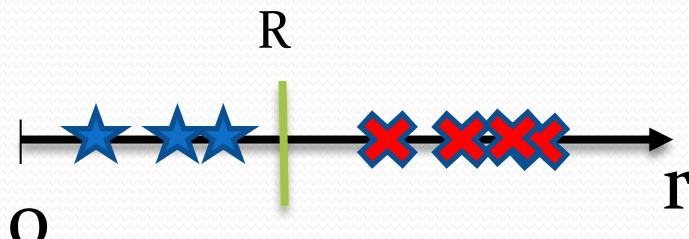
• نکته: تابع فعال سازی نورون در لایه خروجی خطی فرض می شود.

# Radial Basis Function



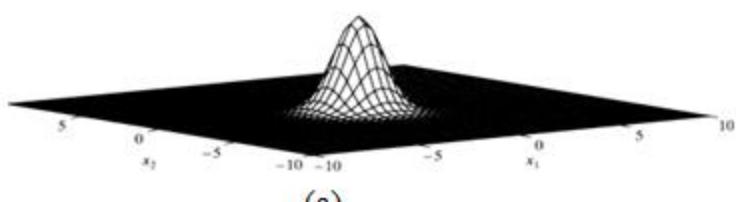
• مجموعه داده با دو ویژگی ✓

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$$

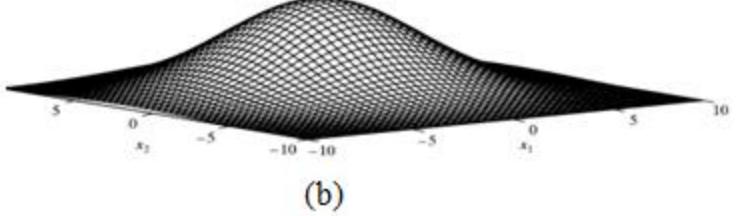


# Radial Basis Function

- در فرآیند آموزش، پارامترهایی که باید آموزش ببینند عبارتند از (مجھول ها):



(a)



(b)

- مراکز توابع گاووسی ✓
- پراکندگی توابع گاووسی ✓
- وزن های شبکه در لایه نهان و لایه خروجی ✓

- روش های یادگیری:

- دقیق ✓

□ تعداد واحدهای مخفی برابر با تعداد الگوهای آموزشی (بردارهای ورودی)

✓ درون یابی تقریبی

□ تعداد واحدهای مخفی کمتر از تعداد الگوهای آموزشی (بردارهای ورودی)

# Radial Basis Function

- استراتژی های یادگیری (الگوریتم آموزش):
  - Fixed Center Selected at Random (**Algorithm 1**)

✓ مراکز توابع شعاعی (نورون ها) در لایه میانی به صورت تصادفی از داده های ورودی که برای آموزش به کار می روند استفاده می شوند.  
✓ پراکندگی ها نیز با نرمالیز شدن از طریق فرمول زیر انتخاب می شوند:

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

✓ وزن ها با استفاده از حل دستگاه  $\Phi W = Y$  بدست خواهد آمد.

# Radial Basis Function

- استراتژی های یادگیری (الگوریتم آموزش):
- Fixed Center Selected at Random (**Algorithm 1**)

$$\begin{array}{ll} \{x_1, x_2, \dots, x_N\} & \text{مجموعه ورودی} \\ \{y_1, y_2, \dots, y_N\} & \text{مجموعه خروجی} \end{array}$$

حل دستگاه ✓

$$\tilde{y}_i = \sum_{j=1}^N w_j \underbrace{\Phi_j(||x_i - x_j||)}_{\Phi_{ij}} \cong y_i$$

تعداد الگوها - تعداد واحدهای مخفی - **N**

$$\begin{bmatrix} \Phi_{11} & \cdots & \Phi_{1N} \\ \vdots & \ddots & \vdots \\ \Phi_{N1} & \cdots & \Phi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad \Phi W = Y \Rightarrow W = \Phi^{-1}Y$$

# Radial Basis Function

- استراتژی های یادگیری (الگوریتم آموزش):
- Fixed Center Selected at Random (Algorithm 1)

✓ pseudo-inverse method

$$\begin{array}{ll} \{x_1, x_2, \dots, x_N\} & \text{مجموعه ورودی} \\ \{y_1, y_2, \dots, y_M\} & \text{مجموعه خروجی} \end{array}$$

$$\tilde{y}_i = \sum_{j=1}^M w_j \underbrace{\Phi_j(||x_i - x_j||)}_{\Phi_{ij}} \cong y_i$$

تعداد الگوهای  $N$   $\Rightarrow$  تعداد واحدهای مخفی  $M$

$$\begin{bmatrix} \Phi_{11} & \cdots & \Phi_{1M} \\ \vdots & \ddots & \vdots \\ \Phi_{N1} & \cdots & \Phi_{MM} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

$$\Phi W = Y \Rightarrow W = (\Phi^T \Phi)^{-1} \Phi^T Y$$

$\downarrow$   
Pseudo Inverse

$$W = \Phi^+ Y$$

# Radial Basis Function

- استراتژی های یادگیری (الگوریتم آموزش):

- Self Organizing Selection of Centers (**Algorithm 2**)

مراکز نورون ها از طریق روش آموزش بدون ناظر و صرفا با استفاده از داده های ورودی بdst می آید (مسئله خوشه بندی). ✓

پراکندگی ها می توانند مثل روش قبل از طریق نرمالیزه کردن (در واقع واریانس بین خوشه ها) به دست می آید و یا از روشی موسوم به K-nearest method استفاده شود. ✓

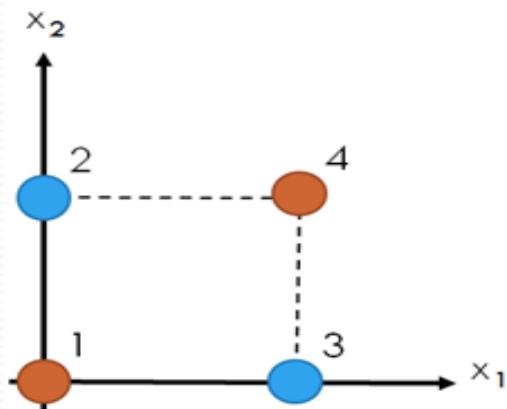
$$\sigma_i = \sqrt{\frac{1}{K} \sum_{k=1}^K \|\vec{c}_k - \vec{c}_i\|^2}$$

**k-th nearest neighbor of  $c_i$**

در نهایت پیدا کردن وزن ها مرحله آموزش با ناظر است که از الگوریتم LMS استفاده می شود. ✓

# Radial Basis Function

مثال) بررسی تابع بولین XOR



$$\varphi_1(x_1, x_2) = e^{-\|x-t_1\|^2}$$

$$\varphi_2(x_1, x_2) = e^{-\|x-t_2\|^2}$$

مراکز

$$t_1 = [0, 0]^T$$

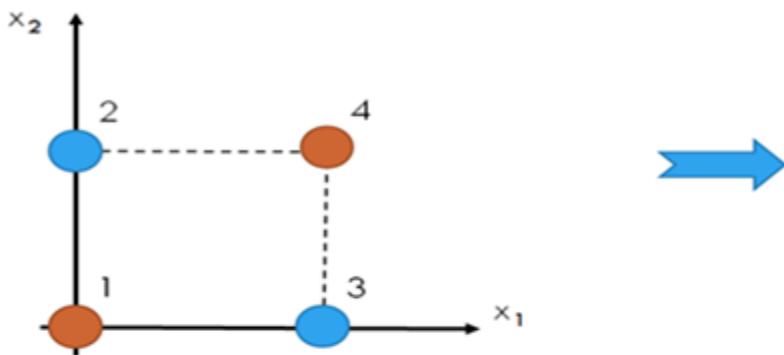
$$t_2 = [1, 1]^T$$

P	x <sub>1</sub>	x <sub>2</sub>	Target
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

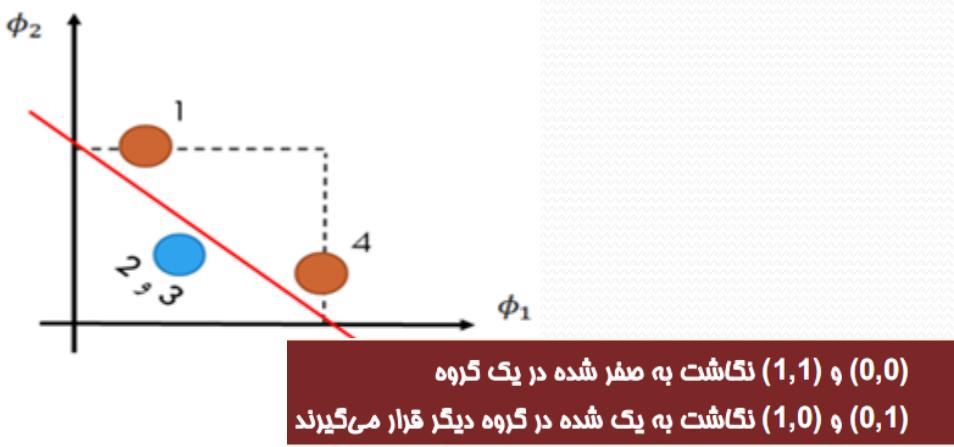
P	x <sub>1</sub>	x <sub>2</sub>	φ <sub>1</sub>	φ <sub>2</sub>
1	0	0	1.0000	0.1353
2	0	1	0.3678	0.3678
3	1	0	0.3678	0.3678
4	1	1	0.1353	1.0000

# Radial Basis Function

مثال) بررسی تابع بولین XOR



$$\text{Output } y = w_1\phi_1 + w_2\phi_2 + \theta$$



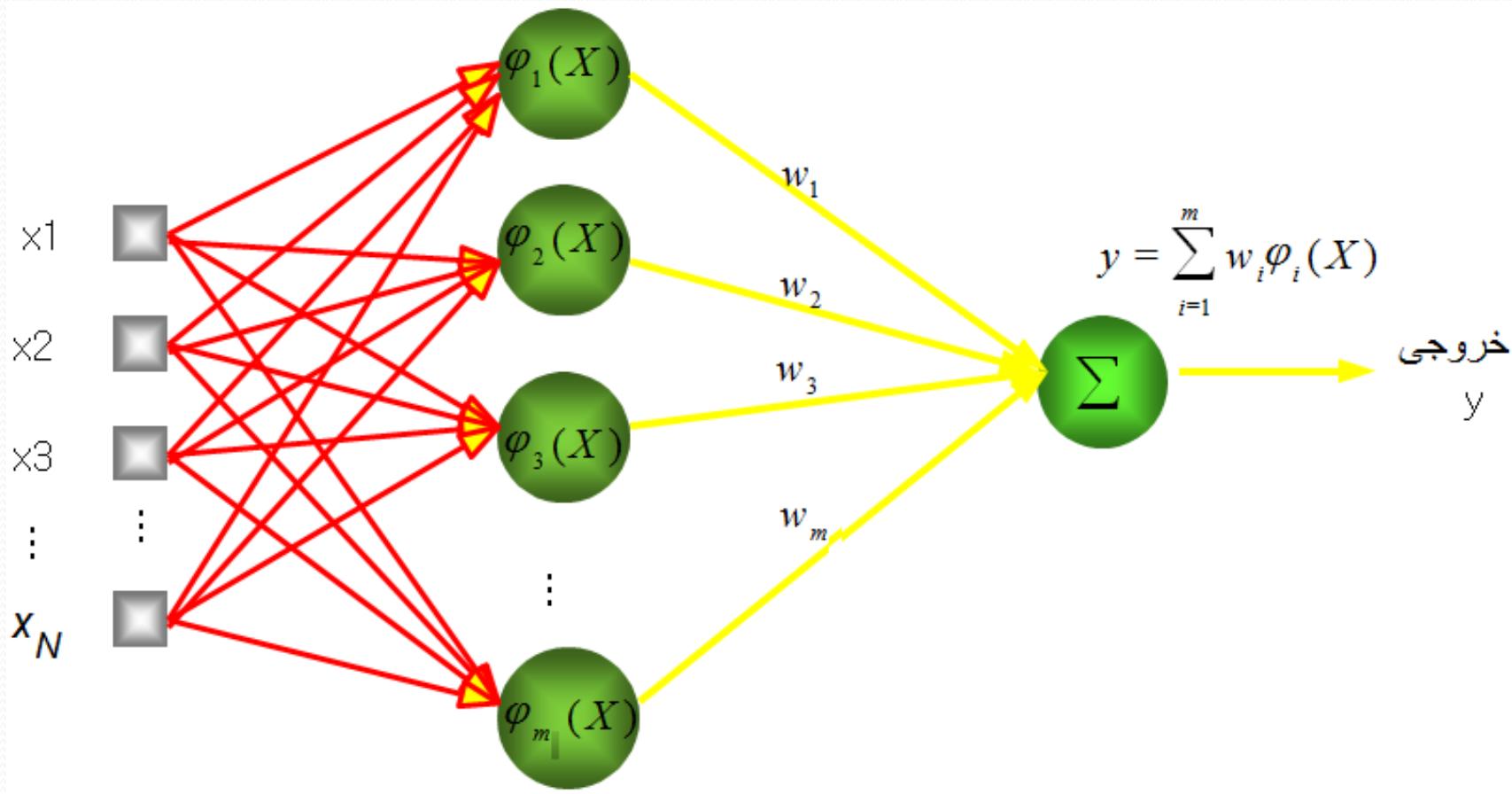
$$\Phi = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.1353 & 1 & 1 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \theta \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\Phi^+ = (\Phi^T \Phi)^{-1} \Phi^T$$

$$w = \Phi^+ y \quad \rightarrow \quad \mathbf{w} = [2.5031 \quad 2.5031 \quad -1.848]^T$$

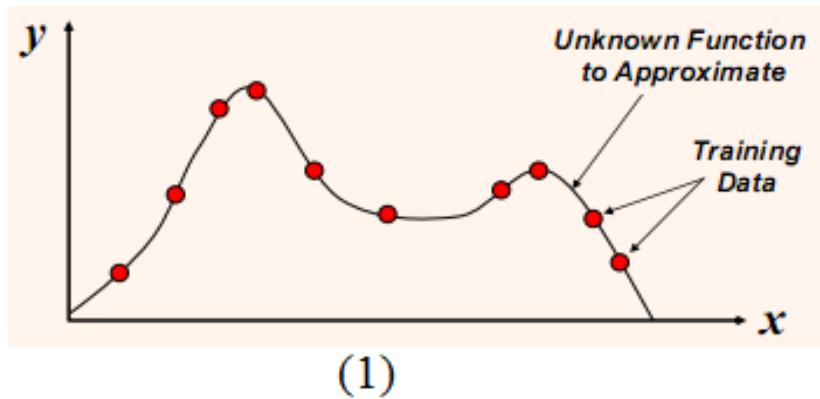
# Radial Basis Function

تقريب تابع

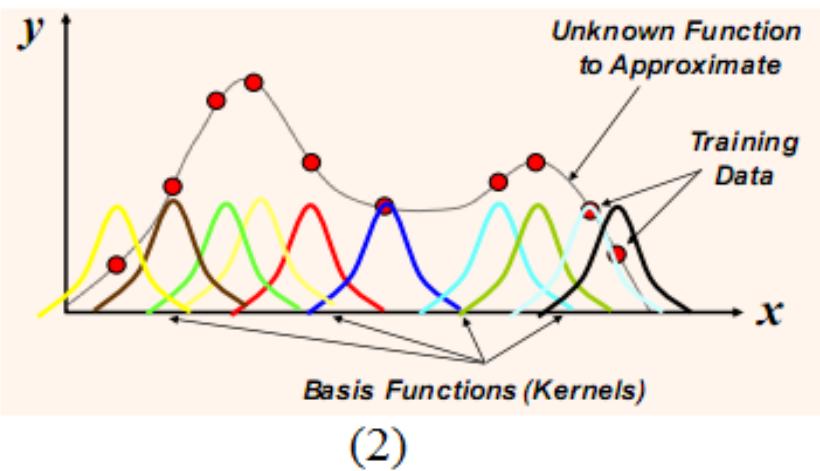


# Radial Basis Function

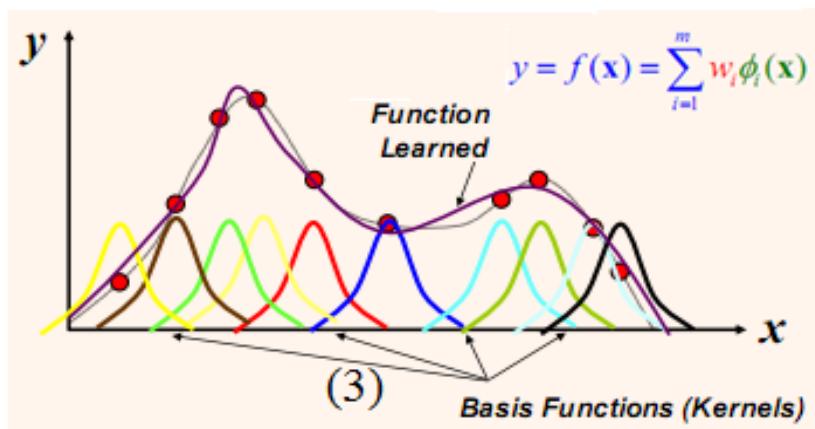
مثال) تقریب تابع



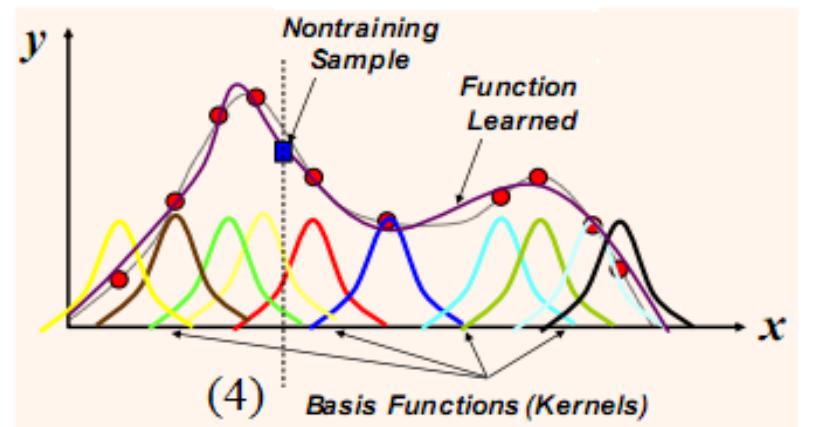
(1)



(2)



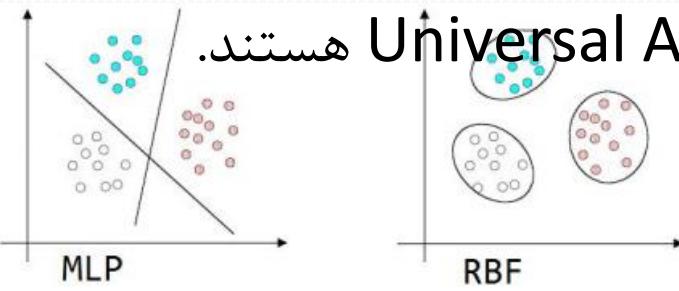
(3)



(4)

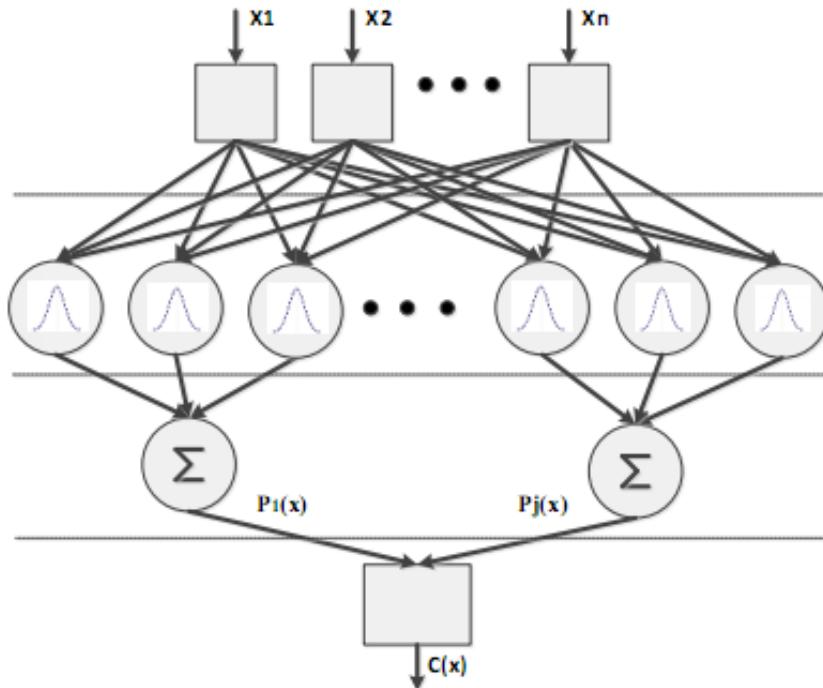
## مقایسه شبکه های MLP و RBF

- در MLP می توانیم چند لایه مخفی داشته باشیم حال آن که در RBF تنها یک لایه مخفی داریم.
- معمولا در MLP اگر برای Pattern Classification استفاده شود تمامی توابع غیر خطی هستند.
- آرگومان توابع در RBF فاصله اقلیدسی است حال آن که در MLP این آرگومان ضرب داخلی بردار ورودی لایه در وزن هاست.
- در MLP یک تقریب کلی از رابطه ورودی- خروجی به دست می آورد در صورتی که در RBF این تخمین به صورت محلی محاسبه می شود.
- سرعت یادگیری RBF بیشتر است، در عین حال تعداد پارامترهای آزاد آن هم بیشتر است(بعد از آموزش در هنگام استفاده MLP سریع تر است).
- هر دو روش MLP و RBF از نوع Universal Approximator هستند.



# PNN (Probabilistic Neural Network)

- نوعی شبکه RBF است که برای استفاده در دسته بندی مناسب است.
- در این شبکه مشابه روش تخمین گر پنجره پارزن، تابع چگالی احتمال برای هر کلاس تخمین زده می شود و مشابه کلاس بند بیز یادگیری الگوها صورت می گیرد.



$$y_j(x) = \frac{1}{n_j} \sum_{l=1}^{n_j} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(||x_{j,l} - x||)^2}{2\sigma^2}\right), j \in [1, 2, \dots, M]$$

# PNN (Probabilistic Neural Network)

- مثال: مجموعه داده برای دو کلاس با داده های زیر داده شده است.

$$x_{1,1} = 2,$$

$$x_{1,2} = 2.5,$$

class 1

$$x_{1,3} = 3,$$

$$x_{1,4} = 1,$$

$$x_{1,5} = 6,$$

and

$$x_{2,1} = 6,$$

class 2

$$x_{2,2} = 6.5,$$

$$x_{2,3} = 7,$$

$$y_1(x) = \frac{1}{5} \sum_{i=1}^5 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_{1,i} - x)^2}{2}\right)$$

$$y_2(x) = \frac{1}{3} \sum_{i=1}^3 \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_{2,i} - x)^2}{2}\right)$$

Using the Gaussian window function with  $\sigma = 1$ , the Parzen pdf for class 1 and class 2 at  $x$  are respectively. The PNN classifies a new  $x$  by comparing the values of  $y_1(x)$  and  $y_2(x)$ . If  $y_1(x) > y_2(x)$ , then  $x$  is assigned to class 1; Otherwise class 2.

# PNN (Probabilistic Neural Network)

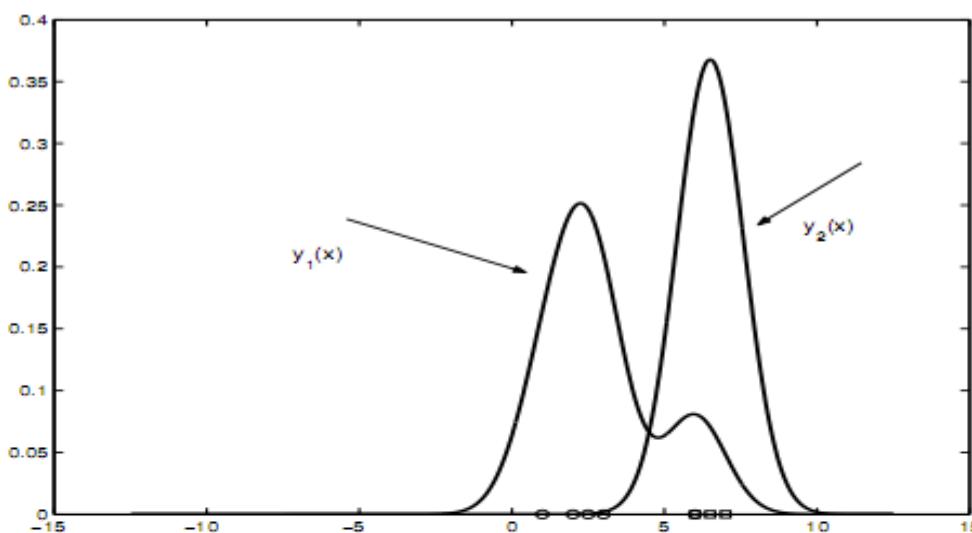


Figure above, Parzen window pdf for two classes.

$$\begin{aligned}y_2(3) &= \frac{1}{3\sqrt{2\pi}} \left\{ \exp\left(-\frac{(6-3)^2}{2}\right) + \exp\left(-\frac{(6.5-3)^2}{2}\right)\right. \\&\quad \left. + \exp\left(-\frac{(7-3)^2}{2}\right) \right\} = 0.0011 < 0.2103 = y_1(x)\end{aligned}$$

so the sample  $x = 3$  will be classified as class 1 using PNN.

ادامه مثال:

# HW 3

- بررسی مجموعه داده های زیر:

ردیف	نام مجموعه داده	تعداد الگوها	تعداد ویژگی ها	تعداد کلاس ها
۱	Flame	240	2	2
۲	Banana	5300	2	2
۳	Aggregation	788	3	7
۴	Iris	150	4	3
۵	Wine	178	13	3

- الگوریتم ها:

MLP

PNN

RBF