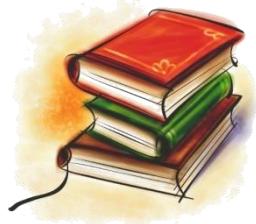


# شبکه‌های عصبی مصنوعی

هادی ویسی

[h.veisi@ut.ac.ir](mailto:h.veisi@ut.ac.ir)

دانشگاه تهران - دانشکده علوم و فنون نوین



## فهرست

- شبکه عصبی مصنوعی
  - مقدمه و معرفی + تاریخچه
- شبکه عصبی پرسپترون
  - آموزش + مثال
- شبکه عصبی آدالاین
  - آموزش + مثال
- شبکه عصبی پرسپترون چندلایه (MLP)
  - آموزش + مثال + نکات تکمیلی
- یادگیری عمیق
  - شبکه باور عمیق (DBN)
- شبکه‌های عصبی بازگشتی
  - حافظه کوتاه مدت ماندگار (LSTM)

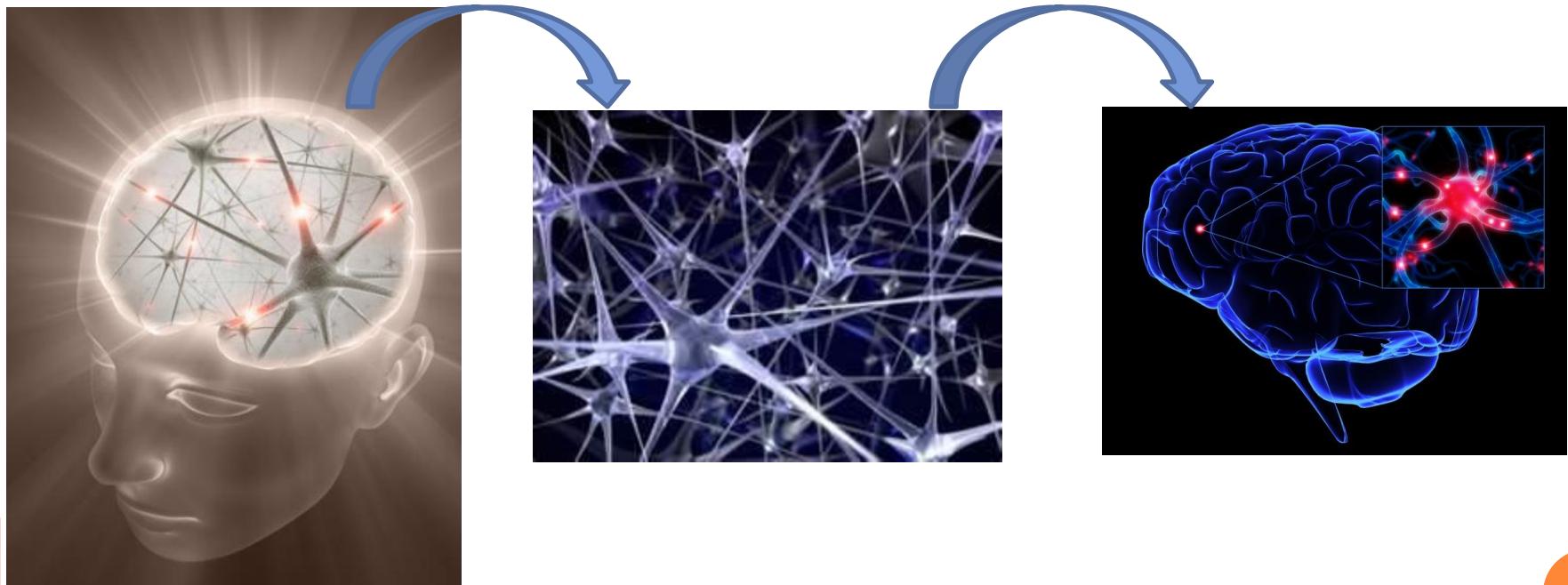
## شبکه عصبی؟

○ مغز = شبکه‌ای بسیار بزرگ از عصب‌ها (نرون‌ها)

- ۱۰۰.۰۰۰.۰۰۰.۰۰۰ نرون

- ۱۰.۰۰۰ اتصال برای هر نرون

○ شبکه عصبی مصنوعی = شبیه‌سازی شبکه عصبی طبیعی



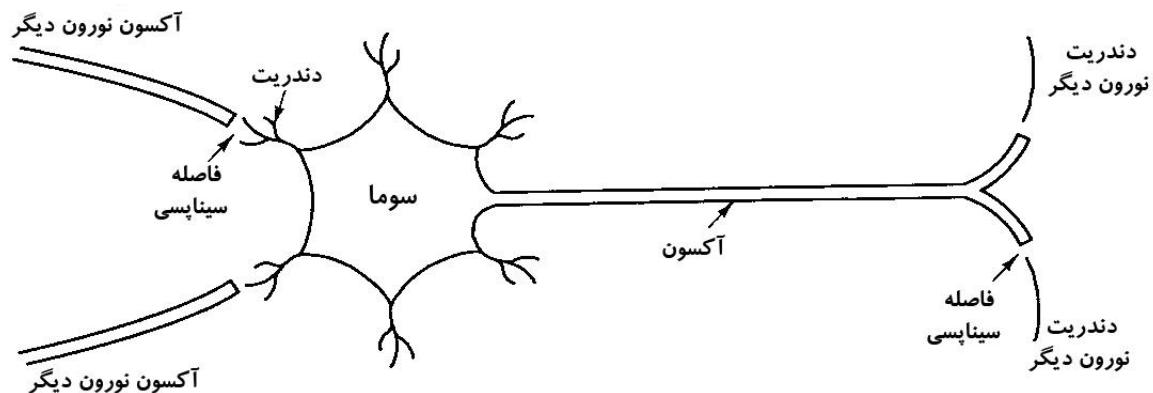
## شبکه عصبی طبیعی ...

### ○ عنصر پردازشگر تشکیل دهنده یک شبکه عصبی مصنوعی

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

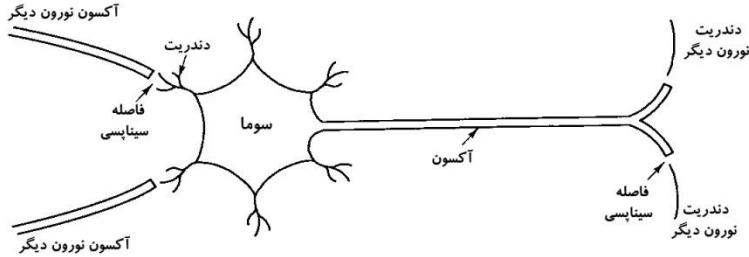
### ○ سه جزء تشکیل دهنده یک نرون طبیعی

- دندrit ها (Dendrite): دریافت سیگنال از سایر نرونها
- سوما (Soma) = بدن سلول: سیگنال های ورودی به سلول را جمع می‌بندد
- آکسون (Axon): ارسال سیگنال به نرون(های) دیگر



# شبکه عصبی طبیعی . . .

## ○ عملکرد نرون طبیعی



- دریافت سیگنال از سایر نرون‌ها توسط دندریت‌ها
- عبور سیگنال‌ها با یک فرآیند شیمیایی از فاصله سیناپسی (Synaptic Gap)
- عمل شیمیایی انتقال دهنده، سیگنال ورودی را تغییر می‌دهند (تضییف/تقویت سیگنال)
- سوما سیگنال‌های ورودی به سلول را جمع می‌بندد
- زمانی که یک سلول به اندازه کافی ورودی دریافت نماید، برانگیخته می‌شود و سیگنالی را از آکسون خود به سلول‌های دیگر می‌فرستد.
  
- انتقال سیگنال از یک نرون خاص نتیجه غلظت‌های مختلف یون‌ها در اطراف پوشش آکسون نرون («ماده سفید» مغز) می‌باشد.
- یون‌ها = پتاسیم، سدیم و کلرید
- سیگنال‌ها به صورت ضربه‌های الکتریکی هستند



## شبکه عصبی مصنوعی . . .

### ◦ شبکه عصبی مصنوعی [Artificial Neural Network]

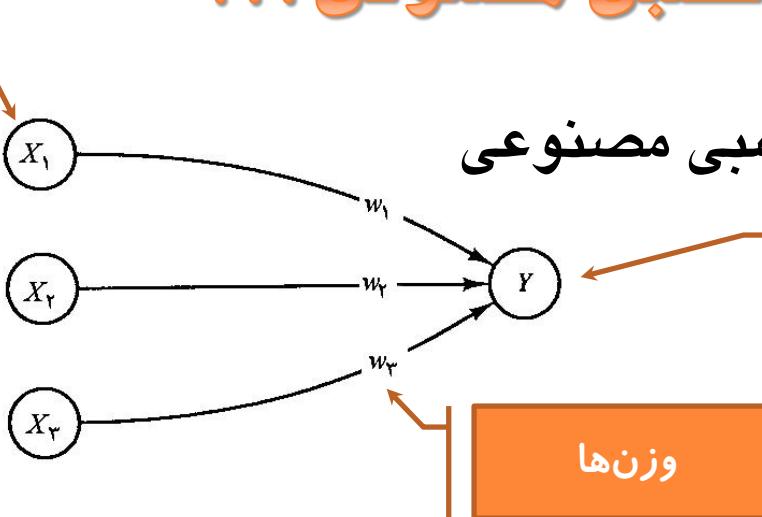
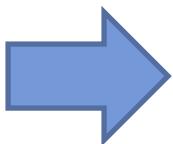
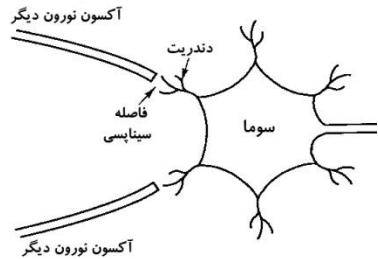
- یک سیستم پردازش اطلاعات با ویژگی‌های مشترکی با شبکه‌های عصبی طبیعی
- تعمیم یافتهٔ مدل‌های ریاضی تشخیص انسان بر اساس زیست‌شناسی عصبی

### ◦ فرضیات پایه شبکه عصبی مصنوعی

- پردازش اطلاعات در اجزای ساده‌ای با تعداد فراوان، به نام نرون‌ها صورت می‌گیرد.
- سیگنال‌ها در بین نرون‌های شبکه از طریق پیوندها یا اتصالات (Connections) آنها منتقل می‌شوند.
- هر پیوند، وزن (Weight) مربوط به خود را دارد که در شبکه‌های عصبی رایج در سیگنال‌های انتقال یافته از آن پیوند ضرب می‌شود.
- هر نرون یک تابع فعال‌سازی (Activation Function) را بر روی ورودی‌های خود اعمال می‌کند تا سیگنال خروجی خود را تولید نماید.
- تابع معمولاً غیرخطی است

# شبکه عصبی مصنوعی . . .

نرون‌های ورودی



نرون خروجی

وزن ها

- فعالسازی‌ها یا سیگنال‌های خروجی نرون‌های ورودی به ترتیب  $x_1, x_2$  و  $x_3$  هستند
- ورودی شبکه به نرون  $Y$ ، حاصل جمع وزن‌دار سیگنال‌های ورودی و وزن‌هاست:

$$y\_in = w_1 x_1 + w_2 x_2 + w_3 x_3 = \sum_i w_i x_i$$

- فعالسازی نرون  $Y$  با اعمال تابع فعالسازی  $f$  روی ورودی آن به دست می‌آید

$$y = f(y\_in)$$

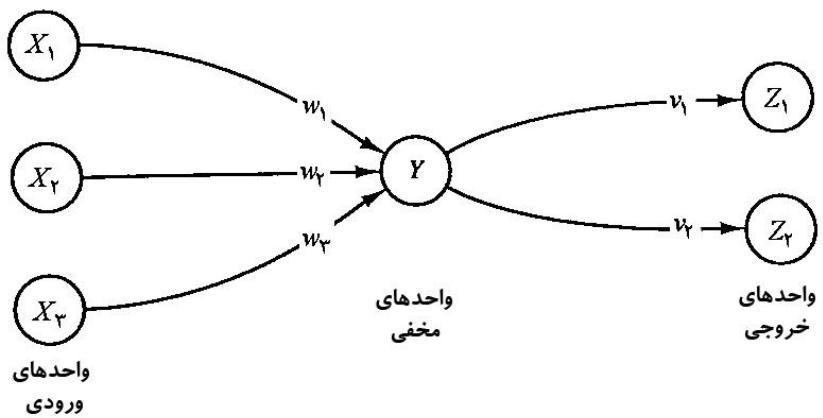
$f(x) = \begin{cases} 1 & \text{if } x > \theta \\ 0 & \text{if } x < \theta \end{cases}$	$f(x) = \frac{1}{1 + \exp(-x)}$
-------------------------------------------------------------------------------------------	---------------------------------

تابع پله

تابع سیگموئید (Sigmoid)

# شبکه عصبی مصنوعی . . .

## ○ یک شبکه عصبی مصنوعی



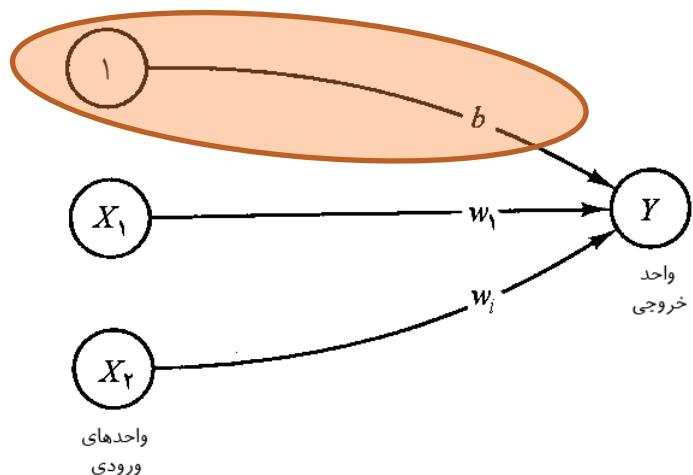
- در یک شبکه یک نرون می‌تواند ورودی‌های مختلفی را از چند نرون دریافت کند

- سه لایه: ورودی، مخفی و خروجی
- دو دسته وزن:  $W$ ها و  $V$ ها

# شبکه عصبی مصنوعی . . .

## ● بایاس

- در ورودی شبکه عصبی، علاوه بر ورودی‌های موردنظر، یک ورودی ثابت با مقدار ۱ نیز داشته باشیم.



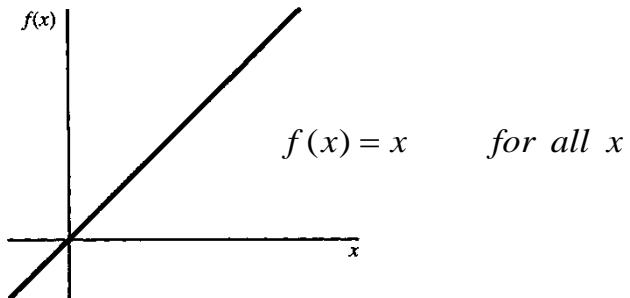
$$y_{in} = b + w_1 x_1 + w_2 x_2 = b + \sum_i w_i x_i$$

# شبکه عصبی مصنوعی . . .

## ○ توابع فعال‌سازی متداول . . .

### • تابع همانی (Identity Function)

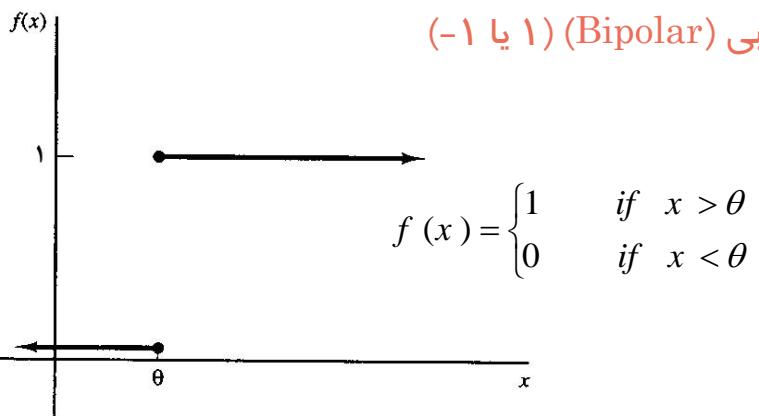
◦ برای واحدهای ورودی



### • تابع پله‌ای دودویی (Step Function)

◦ تابع آستانه (Threshold Function) یا تابع هویساید (Heaviside Function)

◦ خروجی = سیگنال دودویی (۱ یا -۱) یا دوقطبی (Bipolar) (۱ یا -۱)



## شبکه عصبی مصنوعی . . .

### ○ توابع فعال‌سازی متداول . . .

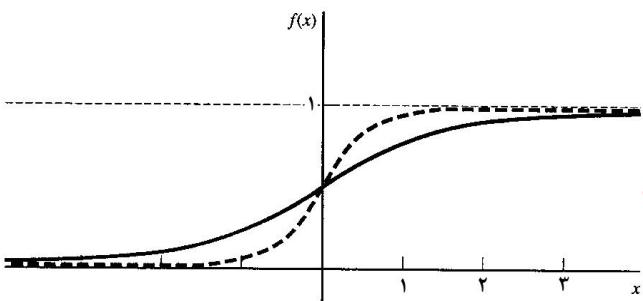
- توابع سیگموید (Sigmoid Functions)

- منحنی‌هایی به شکل S

○ استفاده در شبکه‌های عصبی پساننتشار (نیاز به مشتق‌گیری)

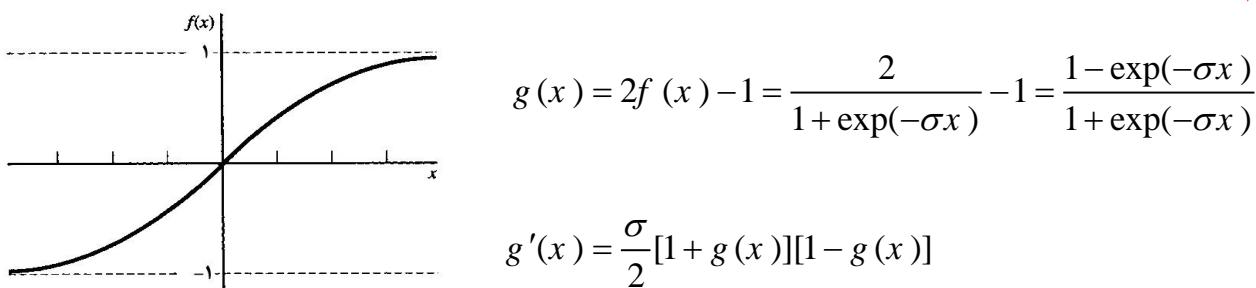
○ سیگموید دودویی - تابع لجستیک (Logistic Function)

○ دامنه ۰ تا ۱، مقادیر مطلوب خروجی یا دودویی است و یا بین ۰ و ۱ است



○ سیگموید دوقطبی - شبیه به تابع تانژانت هیپربولیک (Hyperbolic Tangent Function)

○ دامنه -1 تا 1



# شبکه عصبی مصنوعی . . .

## ◦ توابع فعال‌سازی متداول . . .

ReLU: Rectified Linear Unit •

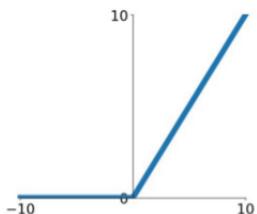
◦ سرعت محاسبه بالاتر نسبت به  $\exp$

◦ رفع مشکل اشباع gradient (تا حدودی)

◦ پر کاربرد در یادگیری عمیق

**ReLU**

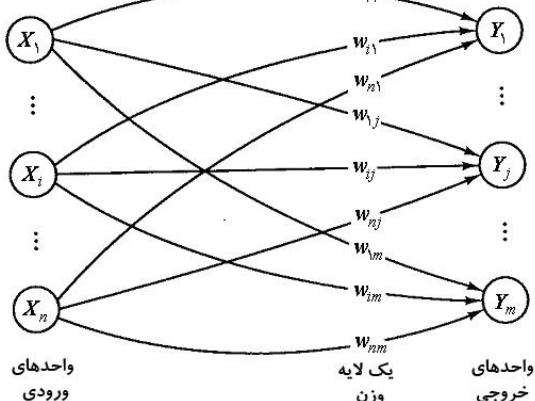
$\max(0, x)$



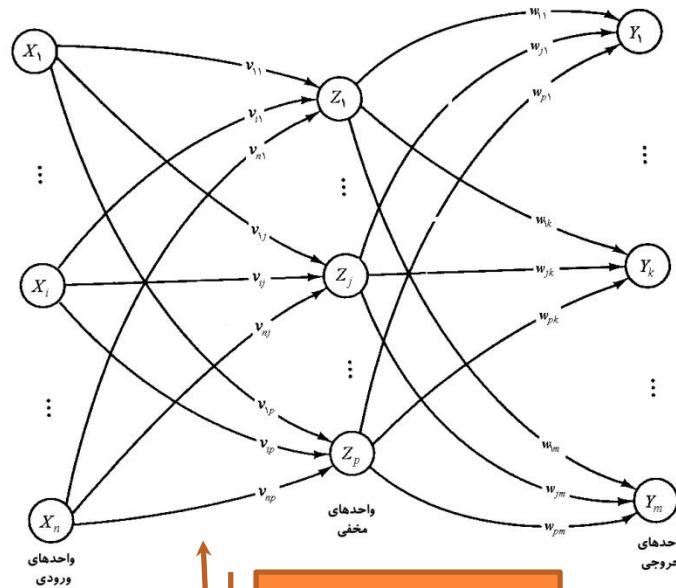
# شبکه عصبی مصنوعی . . .

## ○ ساختارهای رایج . . .

- ساختار یا معماری: آرایش نرون‌ها در لایه‌ها و الگوهای ارتباط داخل و بین لایه‌ها
- شبکه‌های پیش‌خور (Feedforward) - سیگنال‌ها در یک جهت و از سمت واحدهای ورودی به سمت واحدهای خروجی (به سمت جلو) می‌روند



شبکه یک لایه

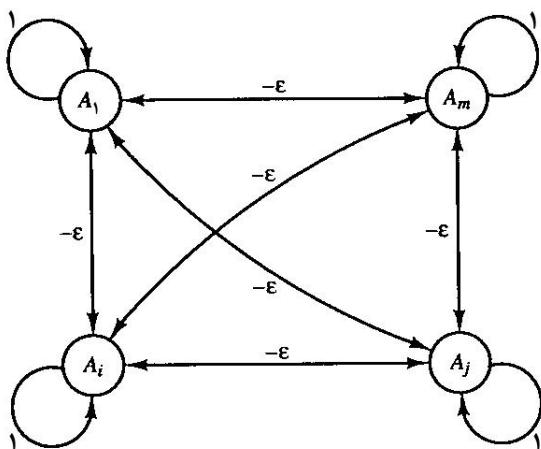


شبکه دولایه

# شبکه عصبی مصنوعی . . .

## ○ ساختارهای رایج

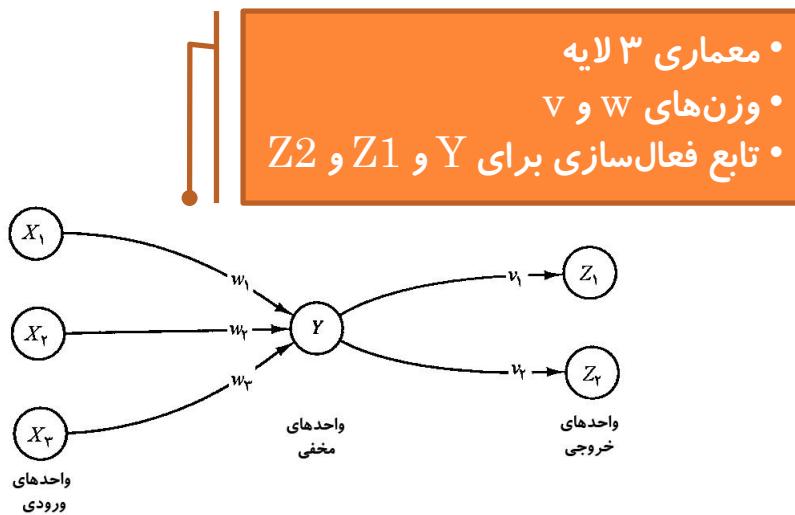
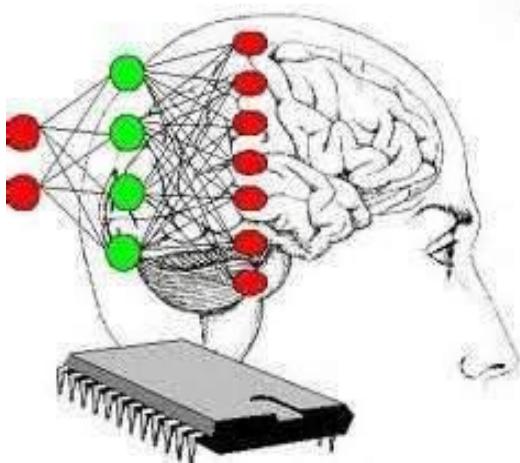
- شبکه بازگشته (Recurrent)، مسیرهای بسته سیگنال از یک واحد به خودش وجود دارد
- شبکه رقابتی: واحدهای آن کاملاً به هم مرتبطاند



# شبکه عصبی مصنوعی . . .

## ویژگی‌های مشخص کننده یک شبکه عصبی مصنوعی

- ساختار یا معماری شبکه (Architecture): الگوی پیوندهای بین نرون‌های مختلف
- الگوریتم آموزش یا یادگیری (Training or Learning Algorithm): روش تعیین وزن‌های روی پیوندهای شبکه
- تابع فعال‌سازی شبکه (Activation Function) که هر نرون روی ورودی‌های خود اعمال می‌کند





## شبکه‌های عصبی مصنوعی: تاریخچه ...

### ○ دهه ۴۰ - اولین شبکه‌های عصبی مصنوعی

- ۱۹۴۳ - معرفی نرون مک‌کلاچ-پیترز (اولین شبکه عصبی مصنوعی)
- ۱۹۴۹ - شبکه هب: توسط دونالد هب (روانشناس) - اولین قانون یادگیری

### ○ دهه ۵۰ - پرسپترون

- ۱۹۵۸ - توسط فرانک روزنبلات - قانون یادگیری قوی‌تر از هب (مفهوم خطا و تکرار)

### ○ دهه ۶۰ - گسترش پرسپترون + آدالاین

- ۱۹۶۰ - شبکه آدالاین توسط ویدرو و هاف - قانون یادگیری دلتا (مبتنی بر کاهش خطا)

### ○ دهه ۷۰ - سال‌های خاموش

- ۱۹۷۲ - اولین کار کوهونن (از هلسینکی)، روی شبکه‌های عصبی حافظهٔ پیوندی

# شبکه‌های عصبی مصنوعی: تاریخچه ...

## ۸۰- دهه شکوفایی شبکه‌های عصبی

- ۱۹۸۲ - شبکه‌های هاپفیلد (جزء شبکه‌های حافظه انجمنی)
- ۱۹۸۲ - نگاشت‌های خودسازمانده کوهون (SOM)
- ۱۹۸۵ - الگوریتم پساننتشار خطا برای آموزش شبکه‌های چندلایه (MLP)
- ۱۹۸۵ - ماشین بولتزمن: تغییر وزن‌ها براساس تابع چگالی احتمال
- ۱۹۸۷ - شبکه‌های نظریه نوسان وفقی (ART) توسط کارپنز و گراسبرگ
- ۱۹۸۷ - شبکه Neocognitron توسط فوکوشیما برای بازشناسی نویسه‌ها
- مطالعات ریاضیاتی و زیست‌شناختی
- پیاده‌سازی سخت‌افزاری شبکه عصبی



# شبکه‌های عصبی مصنوعی: تاریخچه

## ○ دهه ۹۰ - دهه کاربرد

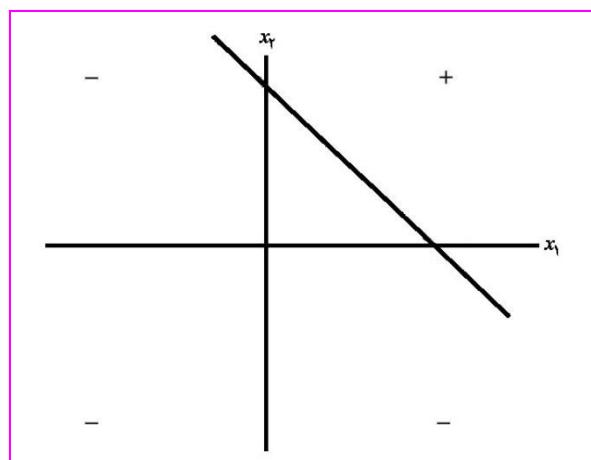
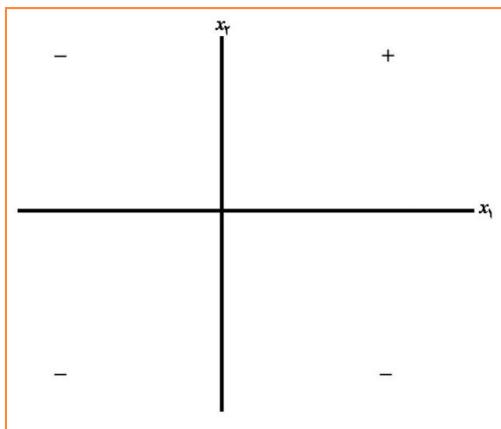
- به کار گیری شبکه‌های عصبی در کاربردهای مختلف
- توسعه شبکه توابع پایه شعاعی (RBF)
- LSTM - شبکه بازگشتی ۱۹۹۷

## ○ ۲۰۰۰ به بعد

- یادگیری عمیق (Deep Learning)
- شبکه‌های بازگشتی

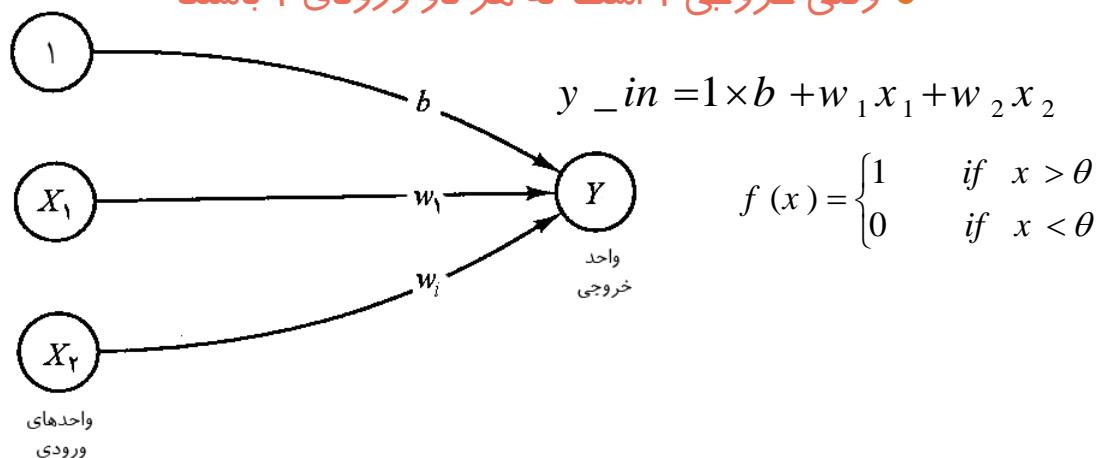
# شبکه عصبی پرسپترون ...

INPUT( $x_1, x_2$ )	OUTPUT
(1, 1)	+1
(1, -1)	-1
(-1, 1)	-1
(-1, -1)	-1



- دو ورودی (دوقطبی) و یک خروجی (دوقطبی)

وقتی خروجی 1 است که هر دو ورودی 1 باشند



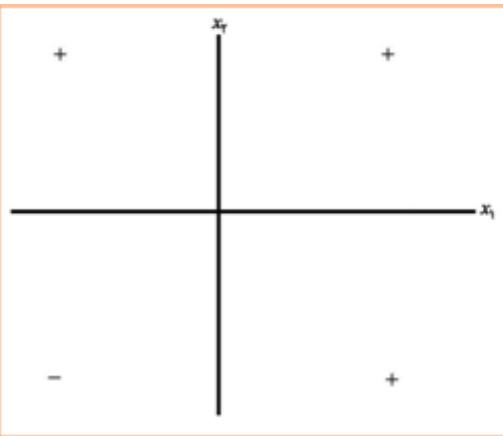
- مرز تصمیم‌گیری

$$b + w_1 x_1 + w_2 x_2 = 0$$

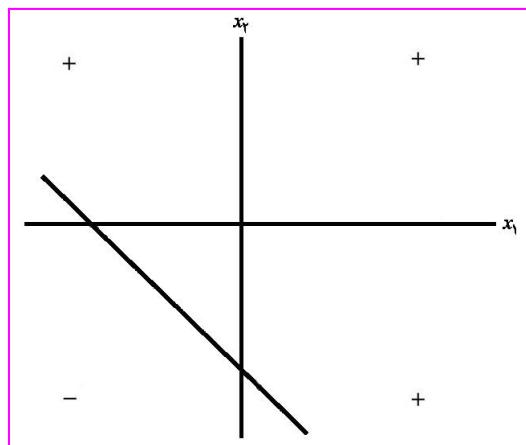
$$x_2 = -x_1 + 1$$

پاسخ

# شبکه عصبی پرسترون ...



INPUT( $x_1, x_2$ )	OUTPUT
(1, 1)	+1
(1, -1)	+1
(-1, 1)	+1
(-1, -1)	-1



- اگر وزن بایاس وجود نداشت، مرز تصمیم‌گیری باید از مبدأ عبور می‌کرد

## ○ مثال: تابع OR

- دو ورودی (دوقطبی) و یک خروجی (دوقطبی)

وقتی خروجی 1 است که حداقل یکی از ورودی‌ها 1 باشد

## ○ مرز تصمیم‌گیری

$$b = 1, w_1 = 1, w_2 = 1$$

$$x_2 = -x_1 - 1$$



## شبکه عصبی پرسپترون . . .

○ مساله: نحوه بدست آوردن وزن‌ها (معادله خط تصمیم‌گیری)؟

- الگوریتم‌های یادگیری شبکه عصبی: هب، پرسپترون، آدلاین و ....

### ○ پرسپترون

- جزو معروف‌ترین شبکه‌های عصبی است
- بیشترین اثرگذاری بر شبکه‌های عصبی اولیه
- روزنبلات در سال ۱۹۶۲ و مینسکی و پاپرت در سال‌های ۱۹۶۹ و ۱۹۸۸

○ ایده قانون یادگیری مبتنی بر قانون یادگیری هب اما با چند بهبود کلیدی

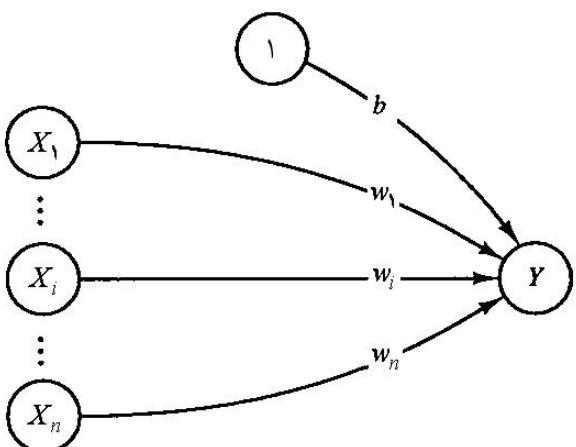
#### ○ یادگیری همراه با تکرار

- در قانون هب، فقط یک بار (بدون تکرار) داده‌های آموزش به شبکه داده می‌شد
- وزن‌ها فقط زمانی تغییر می‌کند که پاسخ شبکه به ازای آن ورودی دارای خطا باشد
- خطا = خروجی محاسبه شده توسط شبکه با مقدار هدف یکی نباشد

# شبکه عصبی پرسترون: ساختار . . .

## ○ ساختار اولیه

- مدل تقریبی شبکیه چشم
- سه لایه نرون (واحدهای حسی، واحدهای پیونددهنده و واحد پاسخ)
  - فقط وزن‌های بین لایه‌های دوم و سوم آموزش داده می‌شود
  - خروجی واحدهای پیونددهنده به واحدهای پاسخ یک بردار دودویی است
  - عملاً شبکه‌ای با یک لایه وزن است



## ○ ساختار برای دسته‌بندی الگو

- دو لایه نرون (یک لایه وزن)
- یک لایه ورودی و یک لایه خروجی
- خروجی دو حالت
  - متعلق بودن به دسته با پاسخ +
  - متعلق نبودن با پاسخ -

# شبکه عصبی پرسپترون: الگوریتم . . .

- مرحله ۰ - مقداردهی اولیه به وزن‌ها و بایاس (مقدار صفر)
  - تعیین نرخ یادگیری  $\alpha \leq 1 < 0$  (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 - فعال‌سازی‌های واحدهای ورودی را مشخص کنید:  $x_i = s_i$
- مرحله ۴ - پاسخ واحد خروجی را محاسبه کنید:

$$y\_in = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 & \text{if } y\_in > \theta \\ 0 & \text{if } -\theta \leq y\_in \leq \theta \\ -1 & \text{if } y\_in < -\theta \end{cases}$$

۰ = ناحیه عدم تصمیم‌گیری (۲θ)

۱ = تعلق به دسته

۱ = تعلق به دسته



# شبکه عصبی پرسپترون: الگوریتم . . .

- مرحله ۵- اگر خطای رخ داده است، وزن‌ها و بایاس را بهروز کنید.

*اگر  $y \neq t$  است، آنگاه:*

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

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

بهروز کردن  
مشروط وزن‌ها

*در غیراین صورت:*

$$w_i(\text{new}) = w_i(\text{old})$$

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

- مرحله ۶- شرایط توقف را آزمایش کنید:

◦ اگر در مرحله ۲ هیچ وزنی تغییر نکرد، الگوریتم را متوقف کنید، در غیراین صورت ادامه دهید.

خطا = برابر نبودن پاسخ شبکه و مقدار هدف

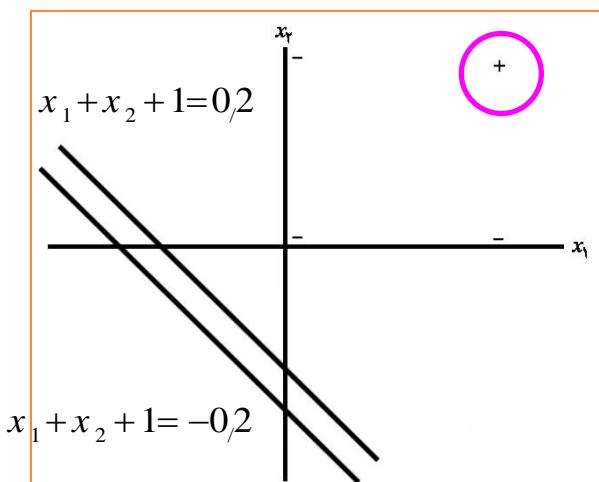
نرخ یادگیری

## شبکه عصبی پرسپترون: مثال . . .

○ تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .

- دودویی: مقادیر صفر و یک – دوقطبی: مقادیر ۱ + و ۱ -
- وزن‌های اولیه و بایاس را صفر؛ نرخ اولیه یادگیری = ۱؛ آستانه = ۰.۲.

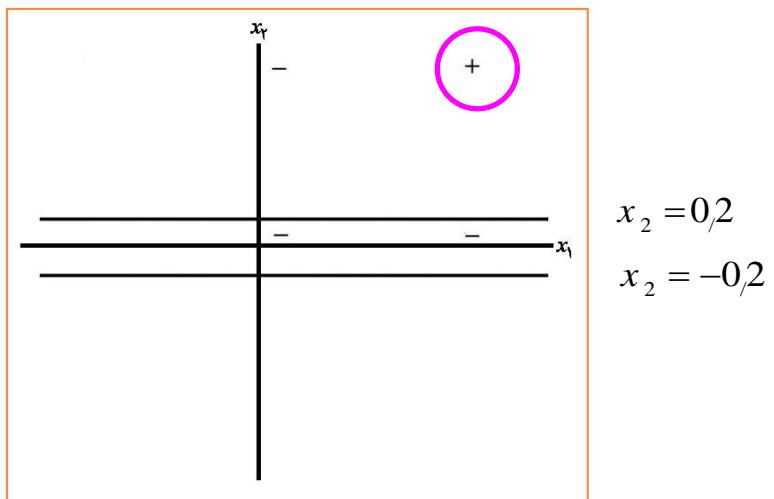
INPUT	NET	OUT	TARGET	WEIGHT	CHANGES	WEIGHTS
$(x_1 \quad x_2 \quad 1)$	$y\_in$	$y$	$t$		$(\Delta w_1 \quad \Delta w_2 \quad \Delta b)$	$(w_1 \quad w_2 \quad b)$
$(1 \quad 1 \quad 1)$	0	0	1		$(1 \quad 1 \quad 1)$	$(0 \quad 0 \quad 0)$



# شبکه عصبی پرسپترون: مثال . . .

- تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .
- ارائه دومین ورودی

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



## شبکه عصبی پرسپترون: مثال . . .

### ○ تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .

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

- ارائه سومین ورودی

- ارائه چهارمین ورودی

○ با توجه به برابر بودن پاسخ شبکه و مقدار هدف، وزن‌ها تغییری نمی‌کنند

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

کامل شدن اولین دور آموزش (Epoch )

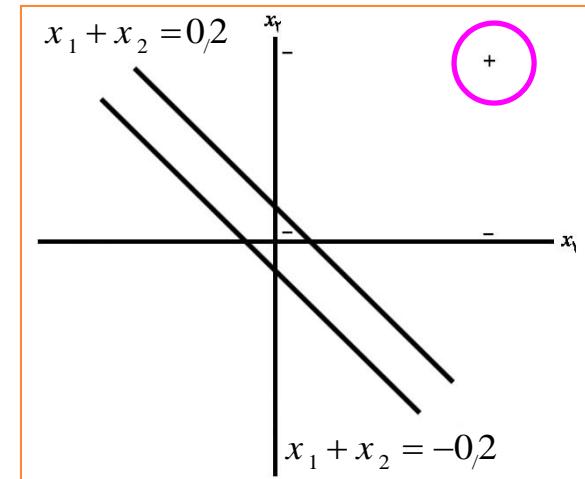
## شبکه عصبی پرسپترون: مثال . . .

○ تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .

- نیاز به تکرار؟؟ صحیح نبودن پاسخ برای اولین الگوی ورودی
- تکراری بودن فرآیند آموزش (Iterative)

• دومین دور آموزش - ارائه اولین ورودی

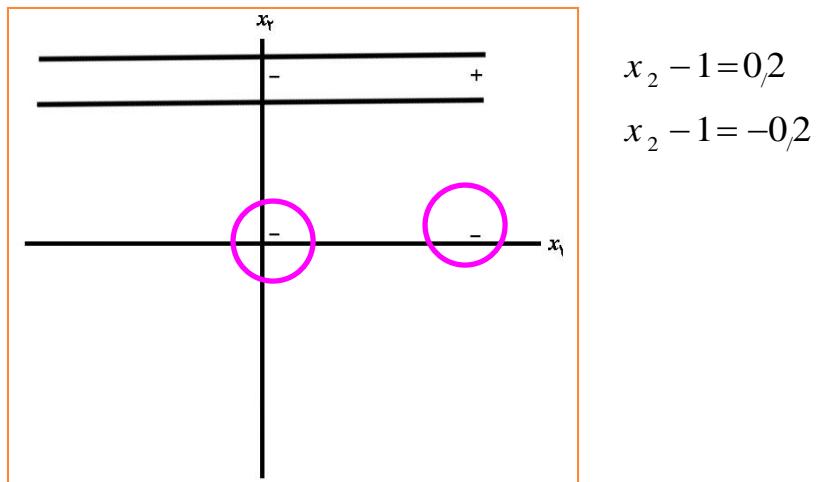
INPUT	NET	OUT	TARGET	WEIGHT	CHANGES	WEIGHTS
$(x_1 \quad x_2 \quad 1)$	$y\_in$	$y$	$t$	$(w_1 \quad w_2 \quad b)$	$(\Delta w_1 \quad \Delta w_2 \quad \Delta b)$	$(w_1 \quad w_2 \quad b)$
$(1 \quad 1 \quad 1)$	-1	-1	1	$(0 \quad 0 \quad -1)$	$(1 \quad 1 \quad 1)$	$(1 \quad 1 \quad 0)$



## شبکه عصبی پرسپترون: مثال . . .

- تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .
- دومین دور آموزش - ارائه دومین ورودی

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



## شبکه عصبی پرسپترون: مثال . . .

○ تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .

- دومین دور آموزش - ارائه سومین ورودی

○ پاسخ برای تمام ورودی‌ها منفی

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

○ دومین دور آموزش - ارائه چهارمین ورودی

○ پاسخ برای تمام ورودی‌ها منفی

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

کامل شدن دومین دور  
آموزش (Epoch)

# شبکه عصبی پرسپترون: مثال . . .

- تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .
- سومین دور آموزش

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

- چهارمین دور آموزش

(1 1 1)	-1	-1	1	(1 1 1)	(1 2 -1)
(1 0 1)	0	0	-1	(-1 0 -1)	(0 2 -2)
(0 1 1)	0	0	-1	(0 -1 -1)	(0 1 -3)
(0 0 1)	-3	-1	-1	(0 0 0)	(0 1 -3)

## شبکه عصبی پر سپترون: مثال . . .

### ○ تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .

- پنجمین، ششمین، .... دور آموزش

$$(1 \ 1 \ 1) \quad 0 \ 0 \ 1 \quad (1 \ 1 \ 1) \quad (3 \ 3 \ -3)$$

$$(1 \ 0 \ 1) \quad 0 \ 0 \ -1 \quad (-1 \ 0 \ -1) \quad (2 \ 3 \ -4)$$

$$(0 \ 1 \ 1) \quad -1 \ -1 \ -1 \quad (0 \ 0 \ 0) \quad (2 \ 3 \ -4)$$

$$(0 \ 0 \ 1) \quad -4 \ -1 \ -1 \quad (0 \ 0 \ 0) \quad (2 \ 3 \ -4)$$

- نهمین دور آموزش

- دهمین دور آموزش

○ عدم تغییر وزن‌ها = توقف الگوریتم

○ همگرایی وزن‌ها

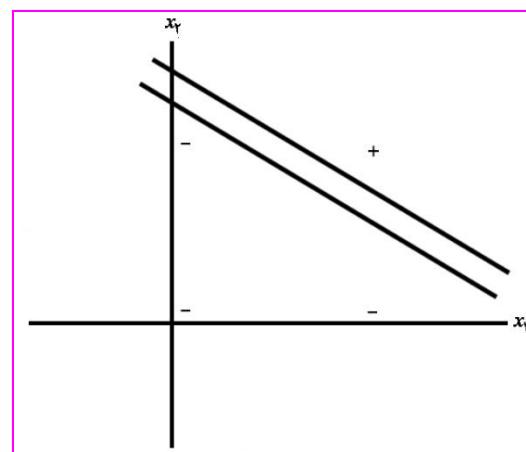
$$(1 \ 1 \ 1) \quad 1 \ 1 \ 1 \quad (0 \ 0 \ 0) \quad (2 \ 3 \ -4)$$

$$(1 \ 0 \ 1) \quad -2 \ -1 \ -1 \quad (0 \ 0 \ 0) \quad (2 \ 3 \ -4)$$

$$(0 \ 1 \ 1) \quad -1 \ -1 \ -1 \quad (0 \ 0 \ 0) \quad (2 \ 3 \ -4)$$

$$(0 \ 0 \ 1) \quad -4 \ -1 \ -1 \quad (0 \ 0 \ 0) \quad (2 \ 3 \ -4)$$

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



# شبکه عصبی پرسپترون: مثال . . .

## ○ تابع AND با ورودی‌ها و هدف‌های دوقطبی

- آستانه، بایاس و وزن‌های اولیه برابر با صفر؛ نرخ یادگیری برابر با ۱

INPUT	NET	OUT	TARGET	WEIGHT		
				CHANGES		WEIGHTS
( $x_1 \ x_2 \ 1$ )	$y\_in$	$y$	$t$	$(\Delta w_1 \ \Delta w_2 \ \Delta b)$	$(w_1 \ w_2 \ b)$	
					$(0 \ 0 \ 0)$	
(1 1 1)	0	0	1	(1 1 1)	(1 1 1)	
(1 -1 1)	1	1	-1	(-1 1 -1)	(0 2 0)	
(-1 1 1)	2	1	-1	(1 -1 -1)	(1 1 -1)	
(-1 -1 1)	-3	-1	-1	(0 0 0)	(1 1 -1)	

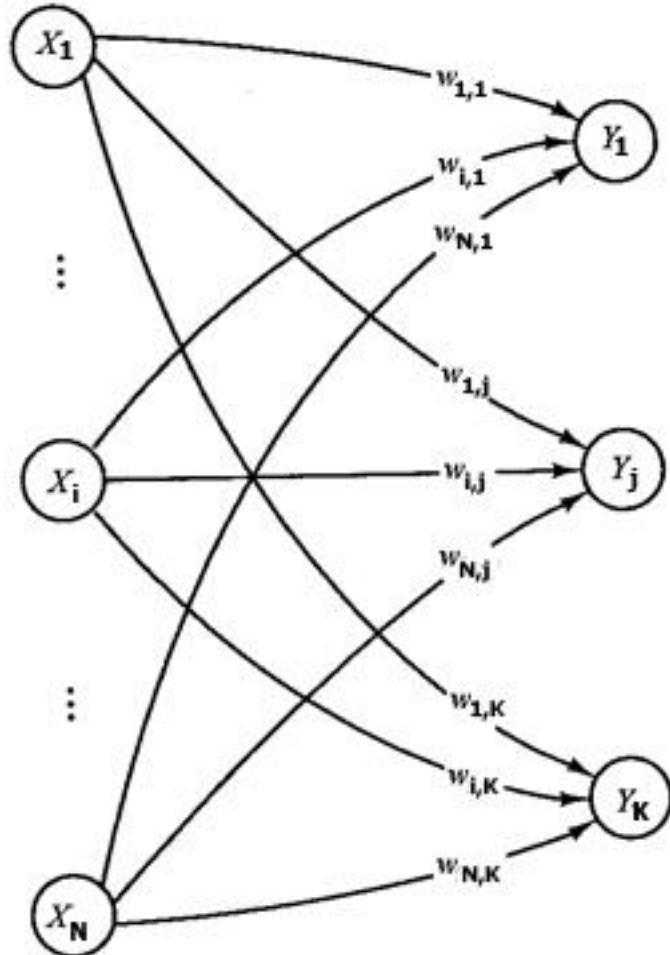
● دور اول آموزش

(1 1 1)	1	1	1	(0 0 0)	(1 1 -1)
(1 -1 1)	-1	-1	-1	(0 0 0)	(1 1 -1)
(-1 1 1)	-1	-1	-1	(0 0 0)	(1 1 -1)
(-1 -1 1)	-3	-1	-1	(0 0 0)	(1 1 -1)

● دور دوم آموزش

● بهبود نتایج با تغییر نمایش دودویی به دوقطبی

# شبکه عصبی پر سپترون: مثال . . .



## ○ تشخیص عنوان (موضوع) متن

- تعداد K دسته (موضوع)
- تعداد K نرون خروجی
- داده = سند متني

○ تبدیل هر سند به یک بردار ویژگی N بعدی

○ تعداد N نرون ورودی

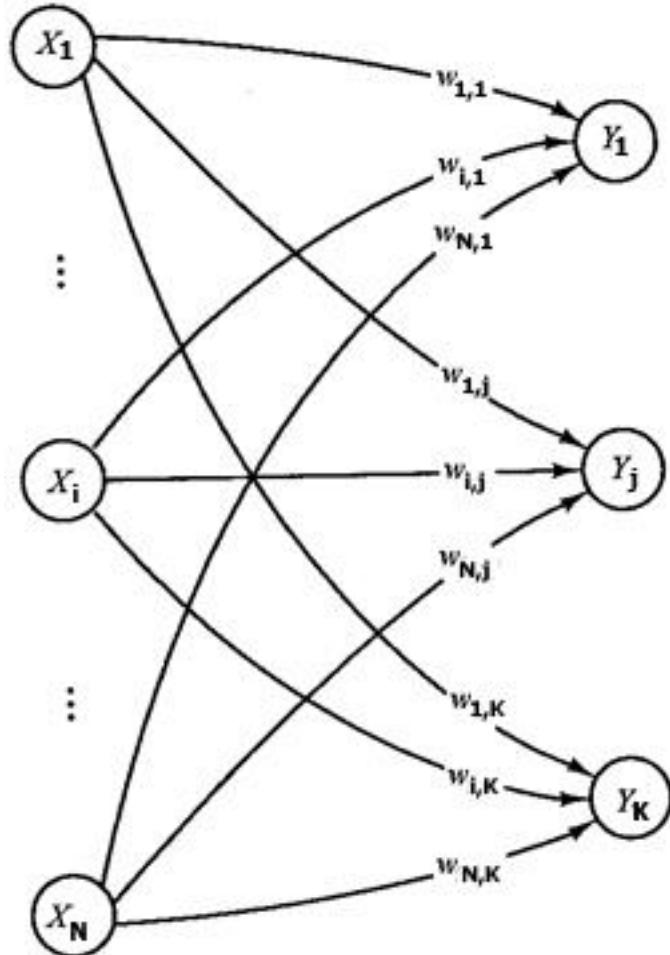
## ○ آموزش

- داده = تعداد M سند دارای برچسب
- خروجی: وزن‌های شبکه = یک ماتریس N\*K (مدل)

## ○ آزمون

- ورودی: یک سند با عنوان نامشخص
- تبدیل سند به یک بردار N بعدی و دادن آن به شبکه
- خروجی: نرونی (دسته‌ای) که مقدار بزرگ‌تر دارد

## شبکه عصبی پر سپترون: مثال . . .



### ○ تشخیص زبان

- تعداد  $K$  دسته (زبان)  $C = \{c_1, c_2, \dots, c_K\}$
- تعداد  $K$  نرون خروجی
- داده = سند متنی یا سند صوتی
- تبدیل هر سند به یک بردار ویژگی  $N$  بعدی
- تعداد  $N$  نرون ورودی

# شبکه عصبی پرسترون: مثال . . .

## ○ تشخیص جنسیت

• تعداد ۲ دسته (زن و مرد)  $C = \{C_1, C_2\}$

○ تعداد یک نرون خروجی (صفر و یک)

○ می‌توان از ۲ نرون هم استفاده کرد

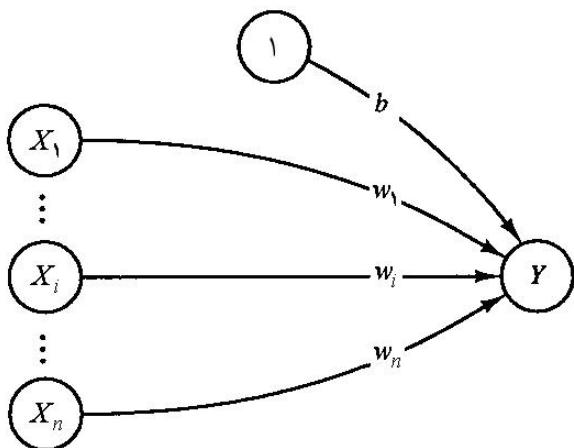
• داده = سند متنی یا سند صوتی

○ تبدیل هر سند به یک بردار ویژگی  $N$  بعدی

○ تعداد  $N$  نرون ورودی

○ ویژگی‌های متنی: تعداد رنگ، صفات، فعل‌ها و ...

○ ویژگی‌های صوتی: انرژی صدا، فرکانس، و ...



# شبکه عصبی پرسترون: مثال . . .

## ◦ تشخیص قطبیت نظرات (موافق/مخالف)

- تعداد ۲ دسته (موافق و مخالف)  $C = \{C_1, C_2\}$

◦ تعداد یک نرون خروجی (صفر و یک)

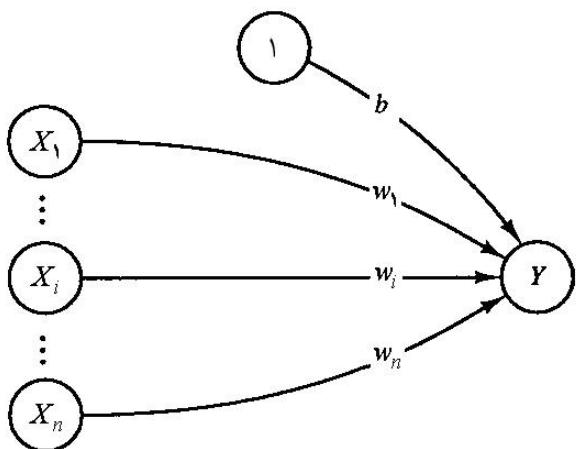
◦ می‌توان از ۲ نرون هم استفاده کرد

◦ داده = سند متنی

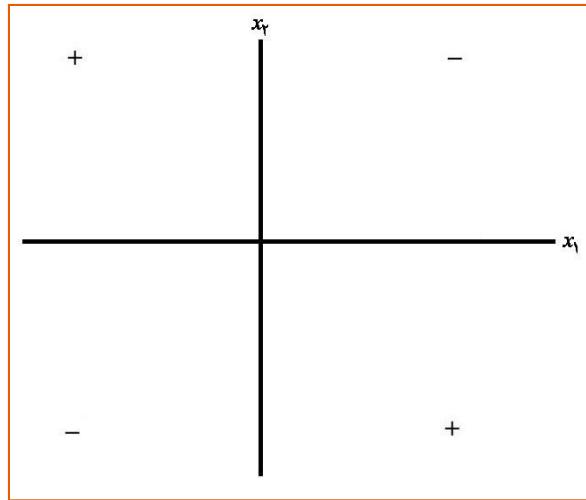
◦ تبدیل هر سند به یک بردار ویژگی  $N$  بعدی

◦ تعداد  $N$  نرون ورودی

◦ ویژگی‌های متنی: صفاتی مثبت/منفی، فعل‌های مثبت/منفی و ...



# شبکه عصبی پرسپترون: مثال



**INPUT( $x_1, x_2$ )**

INPUT( $x_1, x_2$ )	OUTPUT
(1, 1)	-1
(1, -1)	+1
(-1, 1)	+1
(-1, -1)	-1

## ○ مثال: تابع XOR

- دو ورودی (دوقطبی) و یک خروجی (دوقطبی)

- وقتی خروجی 1 است که فقط یکی از ورودی‌ها 1 باشد

- حل؟

- الگوریتم همگرا نمی‌شود (جواب درست نمی‌دهد)

- فضای داده‌های ورودی به صورت خطی جداپذیر (Linearly Separable) نیست.

- هیچ خط مستقیم نمی‌تواند نقاط مثبت و منفی را جدا کند: پرسپترون قادر به یافتن پاسخ نیست

# شبکه عصبی پرسپترون: همگرایی قانون یادگیری

## ○ قضیه

- اگر بردار وزن  $w^*$  وجود داشته باشد به‌طوری‌که برای تمام  $p$ ‌ها داشته باشیم:

$$f(x(p) \cdot w^*) = t(p)$$

آنگاه برای هر بردار اولیه  $w$ ، قانون یادگیری پرسپترون به بردار وزنی نزدیک می‌شود (نه الزاماً منحصر به فرد و نه الزاماً  $w^*$ ) که برای تمام الگوهای آموزش پاسخ صحیحی می‌دهد و این کار در مراحلی با تعداد متناهی انجام می‌شود.

$p$  = تعداد بردارهای ورودی آموزش

$x(p)$  = بردارهای ورودی آموزش

$t(p)$  = مقدار هدف معادل بردارهای ورودی آموزش (دوقطبی)

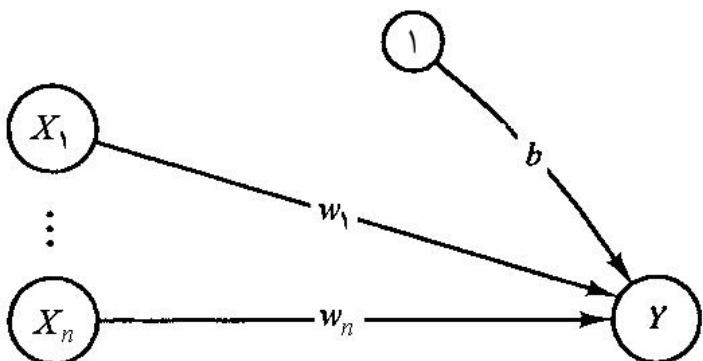
$f$  = تابع فعال‌سازی خروجی

- برقراری این قضیه فقط برای مسائل خطی جدایی‌پذیر (Linearly Separable)

## شبکه آدالاین ...

### آدالاین = نرون خط و فقی (ADaptive LInear Neuron)

- توسط ویدور و هاف در سال ۱۹۶۰
- دارای قانون یادگیری متفاوت با پرسپترون
- قانون یادگیری = قانون دلتا = قانون میانگین مربعات کمینه (LMS) = قانون ویدرو-هاف
- میانگین مربعات خطای بین مقدار خروجی شبکه و مقدار هدف در هر مرحله از آموزش کاهش یابد
- استفاده از فعال‌سازی‌های دوقطبی برای سیگنال‌های ورودی و خروجی
- تابع فعال‌سازی خروجی = تابع همانی



- ساختار مشابه با سایر شبکه‌های قبلی
- چند ورودی
- بایاس = ورودی برابر با ۱

## شبکه آداآین: الگوریتم . . .

- مرحله ۰ - مقداردهی اولیه به وزن‌ها (مقادیر تصادفی کوچک)
- مقداردهی به نرخ یادگیری
- مرحله ۱ - تا زمانی که شرایط توقف برقرار نیست، مراحل ۲ تا ۶ را انجام دهید.
- مرحله ۲ - برای هر جفت آموزش دوقطبی  $t : s$  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 گرنه ادامه دهید.

## شبکه آدالین: الگوریتم

### ○ تفاوت یادگیری آدالین با یادگیری پرسپترون

- تغییر وزن‌ها متناسب با میزان تفاوت پاسخ شبکه به یک ورودی و مقدار هدف متناظر این ورودی است.
- در برگیرنده مفهوم خطای که در یادگیری پرسپترون نیز وجود دارد

### ○ نرخ یادگیری

- تاثیر بر سرعت و روند همگرایی الگوریتم
- روش: ابتدا مقدار را بزرگ فرض کرده (مثلًا ۰.۸) و به مرور مقدار آن را کوچک کنیم
- اگر مقدار خیلی بزرگی باشد، فرآیند یادگیری همگرا نخواهد بود
- اگر مقدار بسیار کوچکی باشد، یادگیری بسیار کند می‌شود

## شبکه آدآژین: مثال

### • تابع AND: ورودی‌های دودویی، هدف‌های دوقطبی

- شبکه بعد از آموزش

$x_1$	$x_2$	$t$	$w_1 = 1$	$w_2 = 1$	$w_0 = -\frac{3}{2}$
1	1	1			
1	0	-1			
0	1	-1			
0	0	-1			

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

- مربعات خطای چهار الگوی آموزش با این وزن‌ها = ۱

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

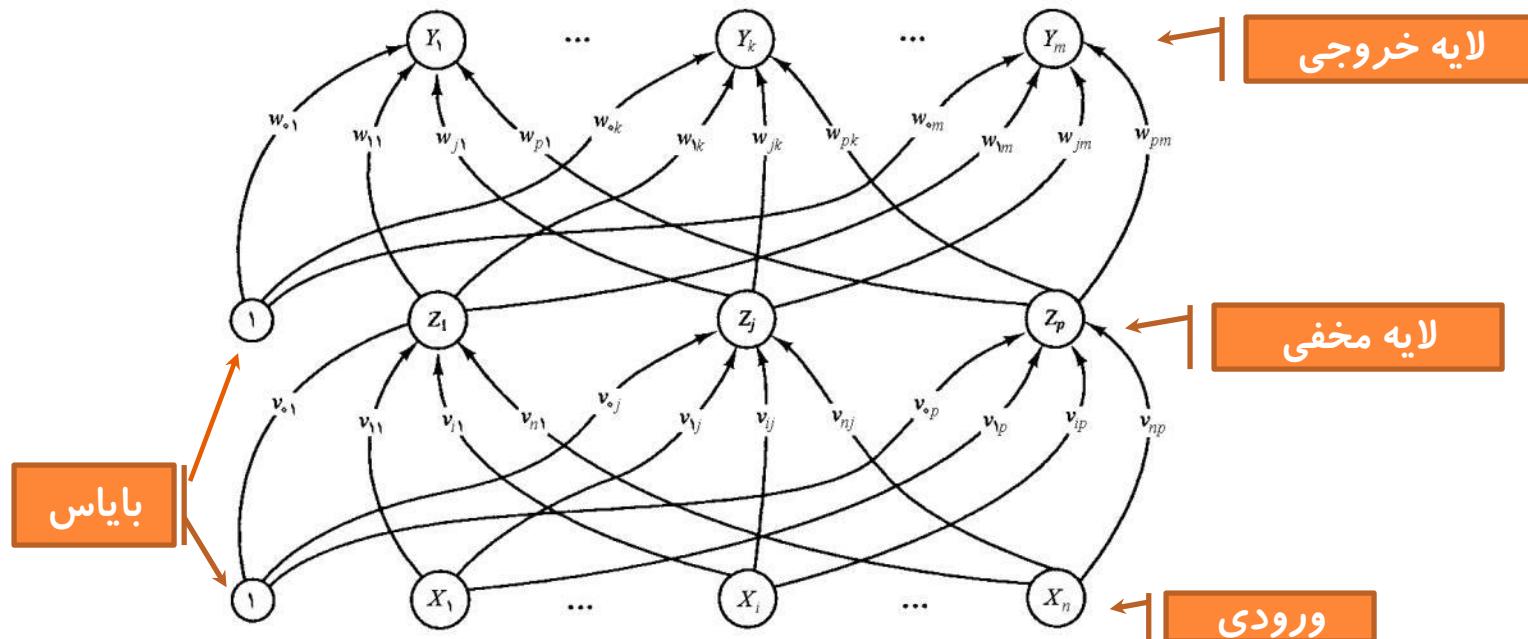
# پرسپترون چندلایه (MLP) ...

- شبکه عصبی پرسپترون چندلایه (MLP: Multi-Layer Perceptron)
  - توسعه شبکه‌های عصبی به حالت چند لایه
  - آموزش با الگوریتم پساننتشار خطا (Error Back-Propagation)
    - قانون دلتای تعمیم‌یافته (Generalized Delta Rule)
      - مبتنی بر قانون دلتای شبکه آدالین
  - روش کاهش گرادیان برای به حداقل رساندن کل مربعات خطای خروجی
  - (از) مهم‌ترین و پرکاربردترین شبکه‌های عصبی

# پر سپترون چند لایه (ساختار) ...

## ○ شبکه سه لایه

- یک لایه ورودی (واحدهای  $X$ )
- یک لایه واحدهای مخفی (واحدهای  $Z$ )
- یک لایه خروجی (واحدهای  $Y$ )



# پرسپترون چندلیه (الگوریتم آموزش) . . .

## ○ مراحل

- پیش خور کردن الگوی آموزش ورودی
- پس انتشار خطای مربوط
- تنظیم وزن‌ها

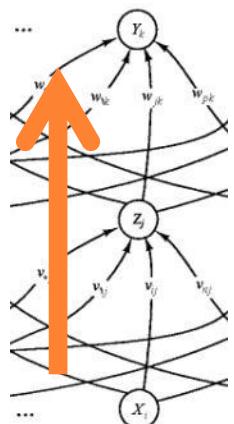
- مبنای ریاضی الگوریتم پس انتشار = بهینه‌سازی کاهش گرادیان (Gradient Descent)
- گرادیان (شیب) یک تابع = نمایانگر جهتی که تابع در آن سریع‌تر افزایش می‌یابد
- شیب با علامت منفی = جهتی نشان دهنده کاهش سریع‌تر آن تابع
- در اینجا تابع مورد نظر = تابع خطای شبکه
- متغیرهای مورد نظر = وزن‌های شبکه

# پرسپترون چندلایه (الگوریتم آموزش) . . .

- 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

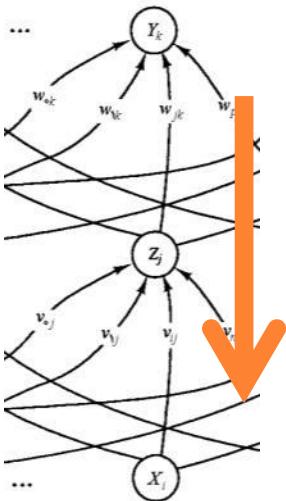
- مرحله ۳ - ارسال سیگنال ورودی  $x_i$  به تمام واحدها در لایه بعدی (واحدهای مخفی)
- مرحله ۴ - محاسبه ورودی واحدهای مخفی و اعمال تابع فعال‌سازی



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

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

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



## پرسپترون چندلایه (الگوریتم آموزش) ...

### ○ 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 به روز کردن به کار می‌رود)

محاسبه پارامتر تصحیح بایاس (بعداً در به روز کردن به کار می‌رود)  
ارسال  $\delta_k$  (مقادیر دلتا) به واحدهای لایه قبلی (لایه مخفی)

- مرحله ۷- دریافت ورودی‌های دلتا توسط واحدهای مخفی از واحدهای خروجی

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

ضرب در مشتق تابع فعال‌سازی جهت محاسبه پارامتر مربوط به اطلاعات خطا

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

محاسبه مقدار تصحیح وزن و بایاس (استفاده در به روز کردن)

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

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

# پرسپترون چندلیه (الگوریتم آموزش) . . .

## ○ به روز کردن وزن‌ها و بایاس‌ها

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

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

- مرحله ۸- به روز کردن وزن‌ها و بایاس‌های واحدهای خروجی

به روز کردن وزن‌ها و بایاس‌های واحدهای مخفی

- مرحله ۹- شرایط توقف را بررسی کنید.

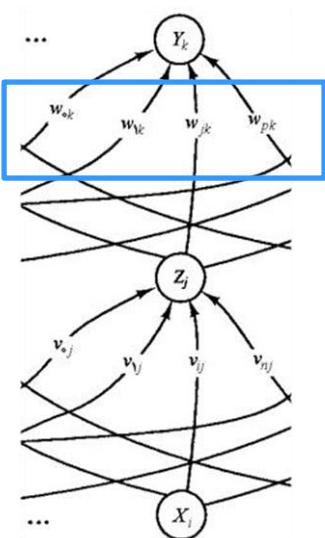
# پرسپترون چندلیه: استخراج قوانین یادگیری . . .

$$y_{in_k} = \sum_j z_j w_{jk} \Rightarrow y_k = f(y_{in_k})$$

برای وزن‌های واحدهای خروجی

$$E = 0.5 \sum_k [t_k - y_k]^2$$

خطا (تابعی از وزن‌ها): باید حداقل شود



$$\begin{aligned} \frac{\partial E}{\partial w_{JK}} &= \frac{\partial}{\partial w_{JK}} 0.5 \sum_k [t_k - y_k]^2 \\ &= \frac{\partial}{\partial w_{JK}} 0.5 [t_k - f(y_{in_k})]^2 \\ &= -[t_k - y_k] \frac{\partial}{\partial w_{JK}} f(y_{in_k}) \\ &= -[t_k - y_k] f'(y_{in_k}) \frac{\partial}{\partial w_{JK}} (y_{in_k}) \\ &= -[t_k - y_k] f'(y_{in_k}) z_J = -\delta_k z_J \end{aligned}$$

$$\begin{aligned} \Delta w_{jk} &= -\alpha \frac{\partial E}{\partial w_{jk}} \\ &= \alpha [t_k - y_k] f'(y_{in_k}) z_j \\ &= \alpha \delta_k z_j \end{aligned}$$

# پرسپترون چندلیه: استخراج قوانین یادگیری

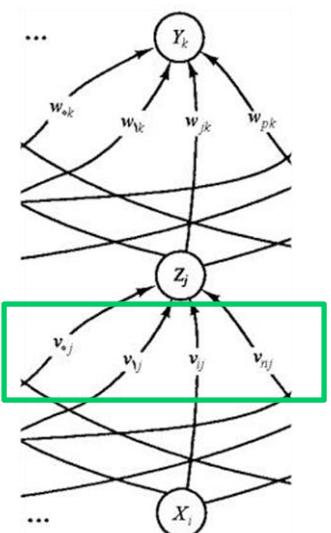
## برای وزن‌های واحدهای مخفی

$$y_{in_k} = \sum_j z_j w_{jk} \Rightarrow y_k = f(y_{in_k})$$

$$E = 0.5 \sum_k [t_k - y_k]^2$$

- داریم (از اسلاید قبلی)

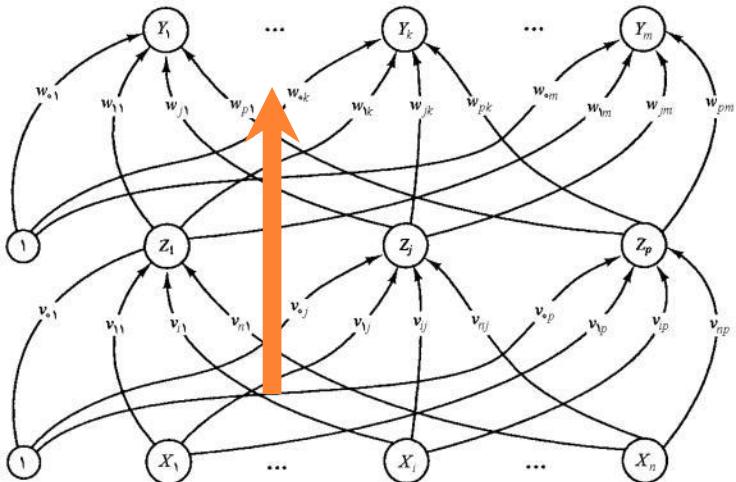
- خطا (تابعی از وزن‌ها): باید حداقل شود



$$\begin{aligned}
 \frac{\partial E}{\partial v_{IJ}} &= -\sum_k [t_k - y_k] \frac{\partial}{\partial v_{IJ}} y_k \\
 &= -\sum_k [t_k - y_k] f'(y_{in_k}) \frac{\partial}{\partial v_{IJ}} y_{in_k} \\
 &= -\sum_k \delta_k \frac{\partial}{\partial v_{IJ}} y_{in_k} \\
 &= -\sum_k \delta_k w_{jk} \frac{\partial}{\partial v_{IJ}} z_J \\
 &= -\boxed{\sum_k \delta_k w_{jk} f'(z_{in_j}) x_i} = -\delta_j x_I
 \end{aligned}$$

$$\begin{aligned}
 \Delta v_{ij} &= -\alpha \frac{\partial E}{\partial v_{ij}} \\
 &= \alpha f'(z_{in_j}) x_i \sum_k \delta_k w_{jk} \\
 &= \alpha \delta_j x_i
 \end{aligned}$$

# پرسپترون چندلایه (کاربرد) . . .



## ○ 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:

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

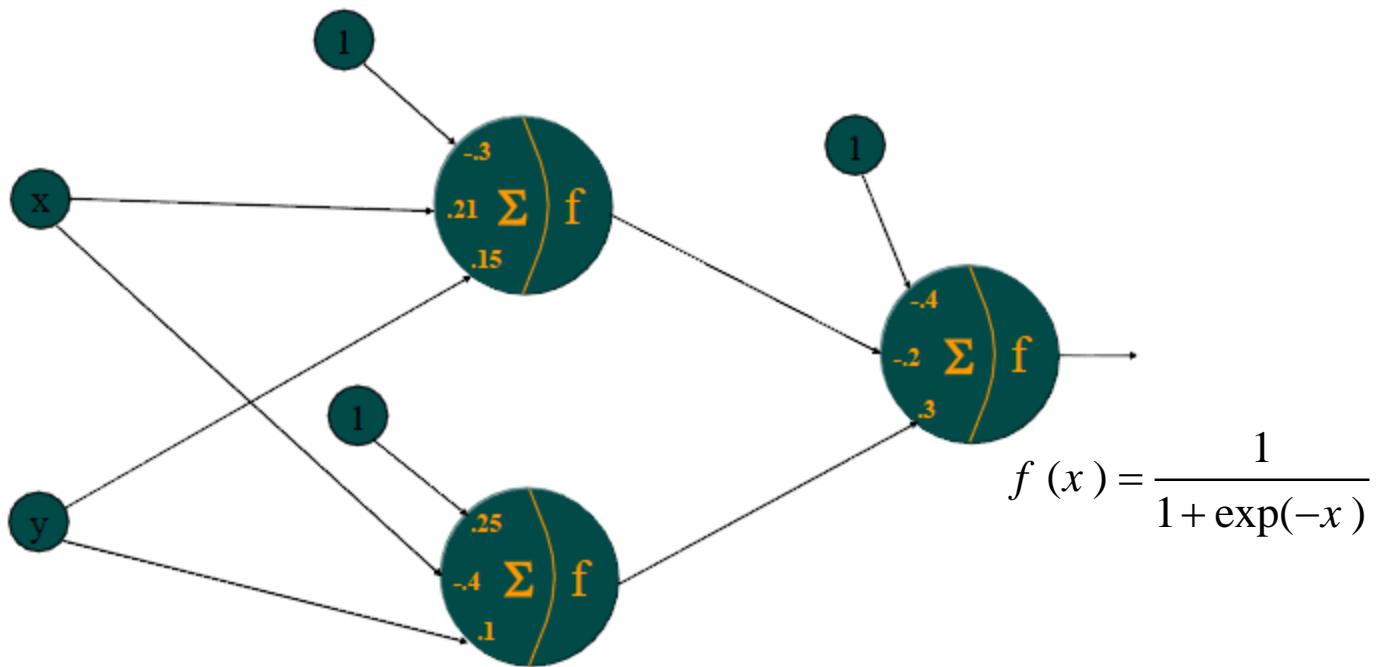
- مرحله ۴: برای واحدهای خروجی:

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

# پرسپترون چندلایه (مثال) . . .

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

- تابع XOR: نمایش دودویی (۱ از ۶)
- مقدار دهی اولیه (تصادفی)

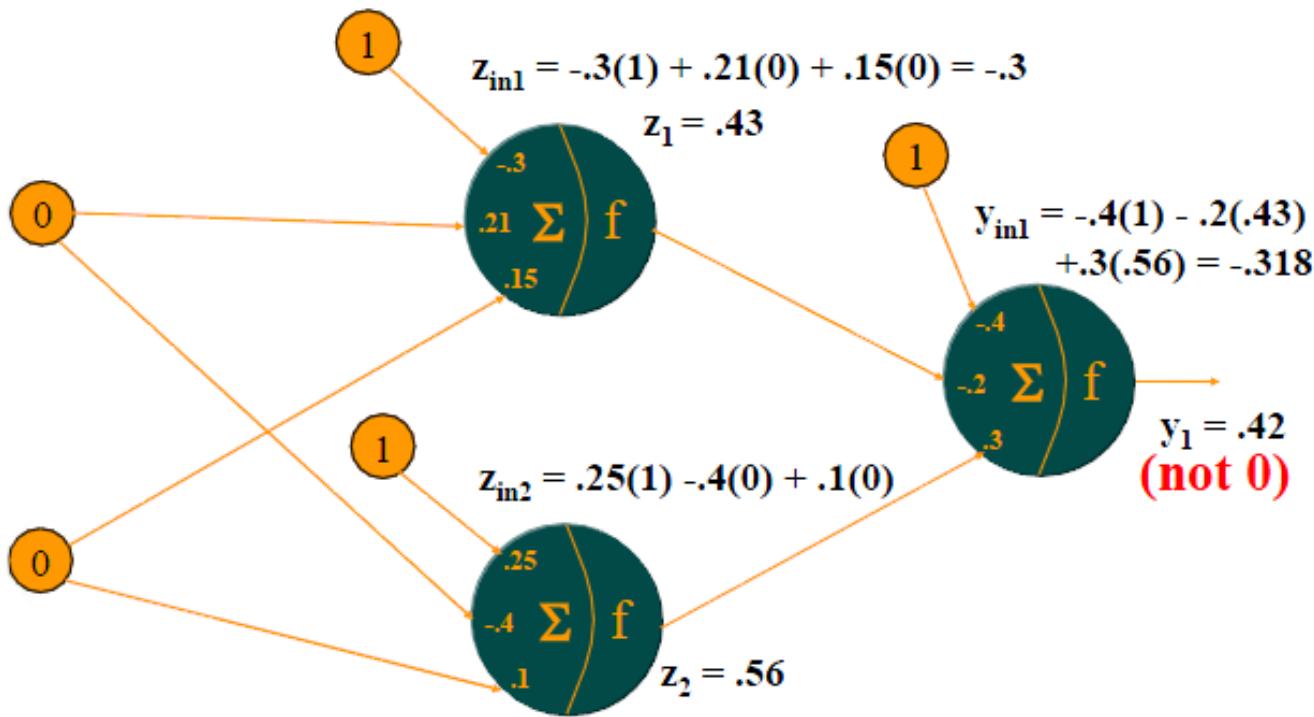


# پرسپترون چندلایه (مثال) . . .

○ تابع XOR نمایش دودویی (۲ از ۶) . . .

$$\begin{array}{ccc} x_1 & x_2 & y \\ 0 & 0 & 0 \end{array}$$

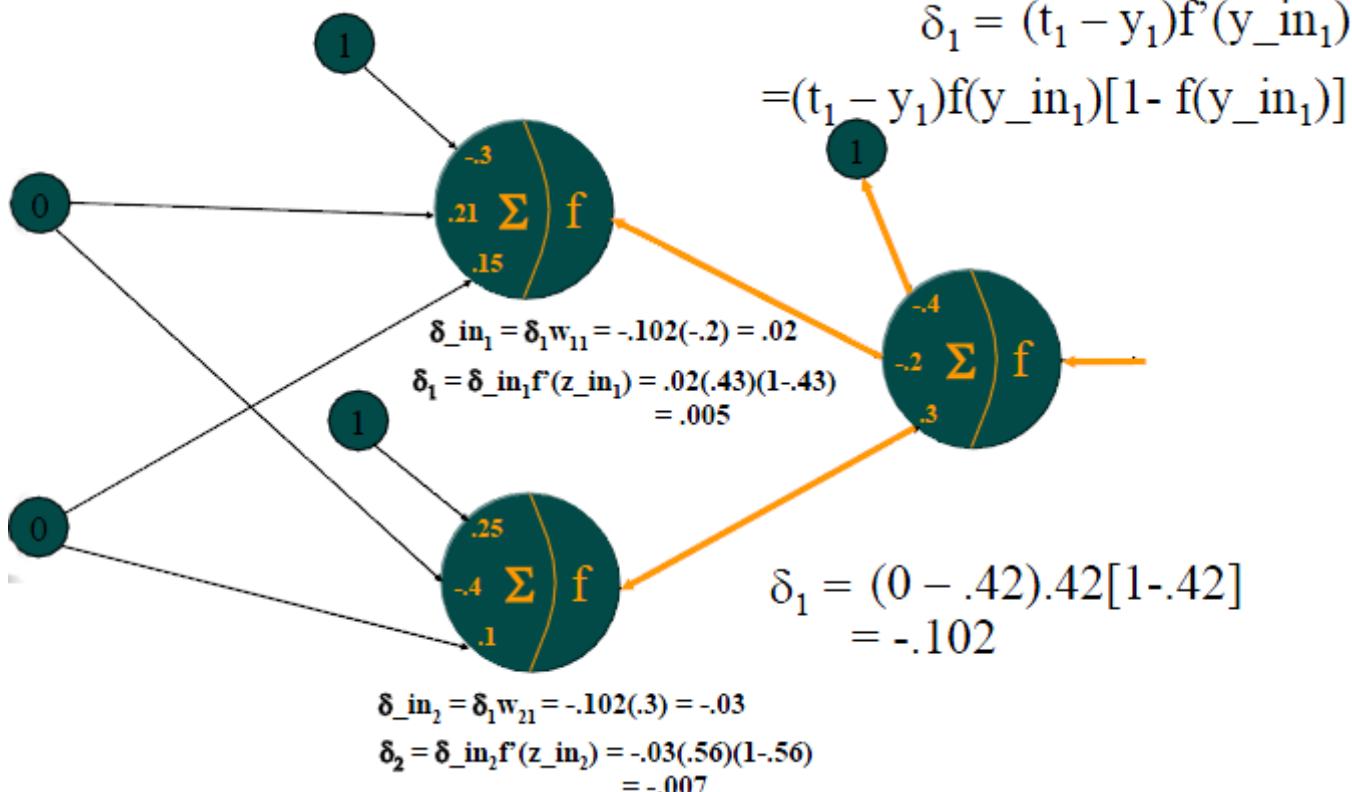
● پیشخور کردن ورودی



# پرسپترون چندلایه (مثال) . . .

○ تابع XOR نمایش دودویی (۳ از ۶) . . .

- پس انتشار خطأ



# پر سپترون چند یه (مثال) . . .

• تابع XOR نمایش دودویی (۴ از ۶) . . .

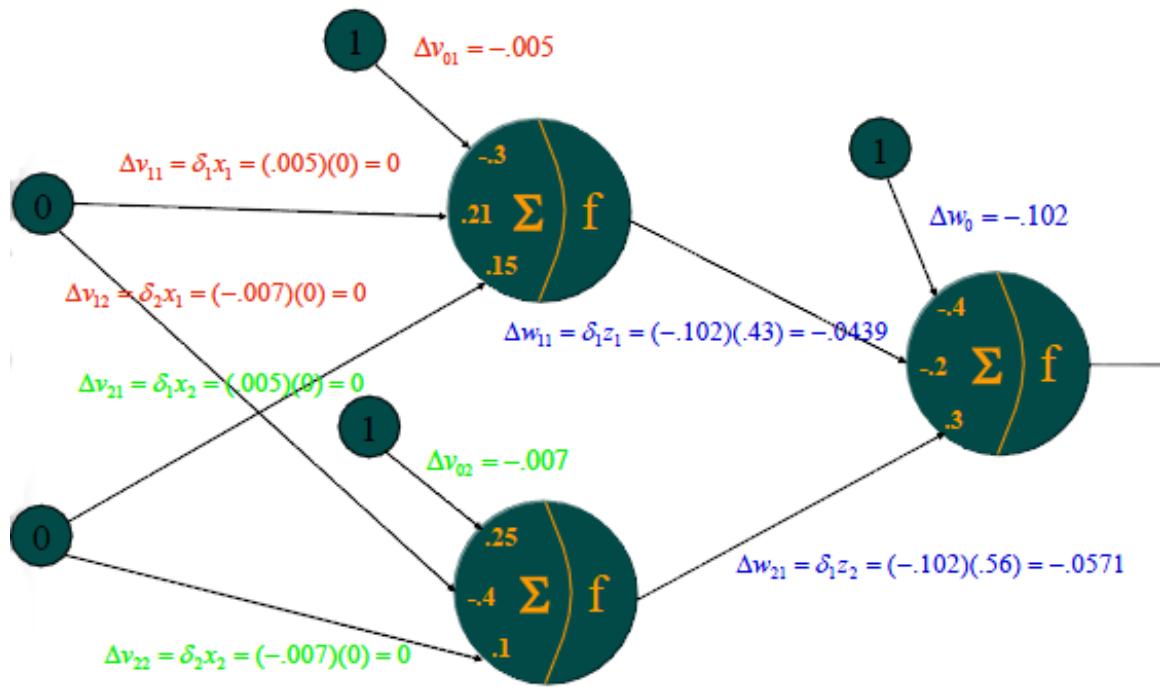
- محاسبه وزن ها

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

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

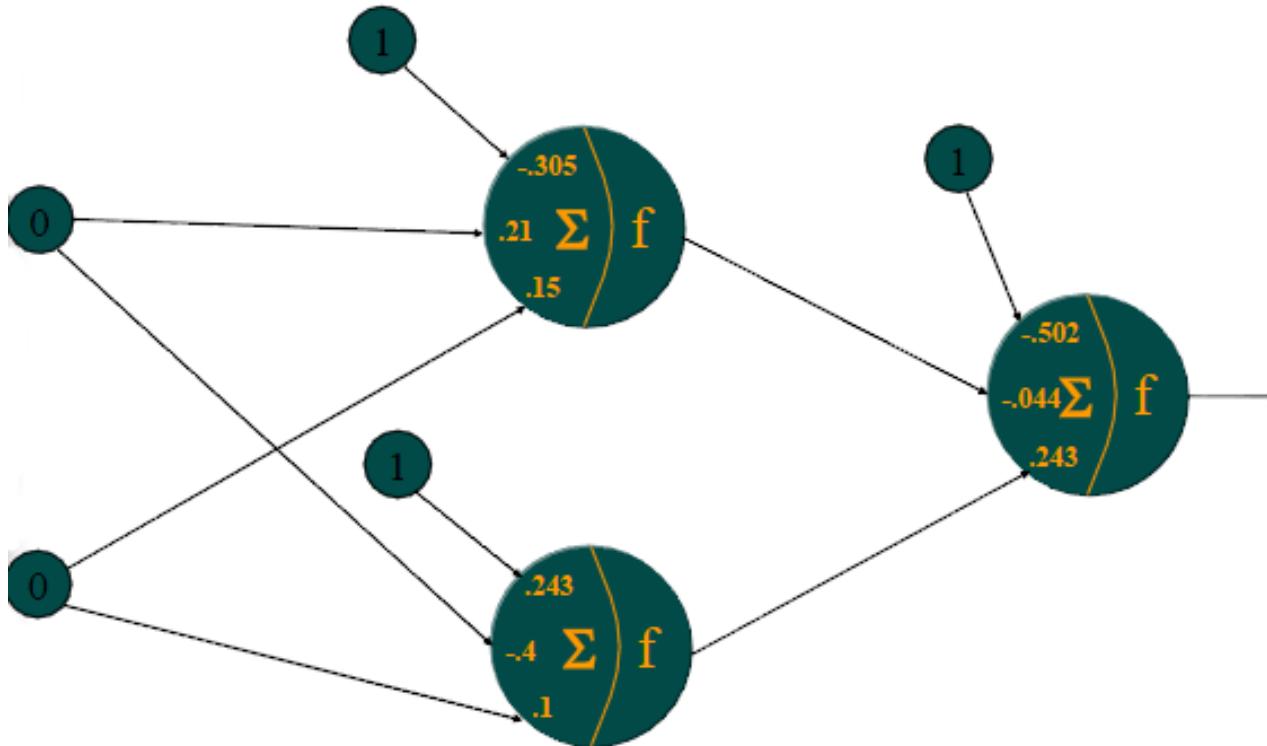
$$\Delta w_{j1} = \alpha \delta_1 z_j \quad j = 1, 2$$

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



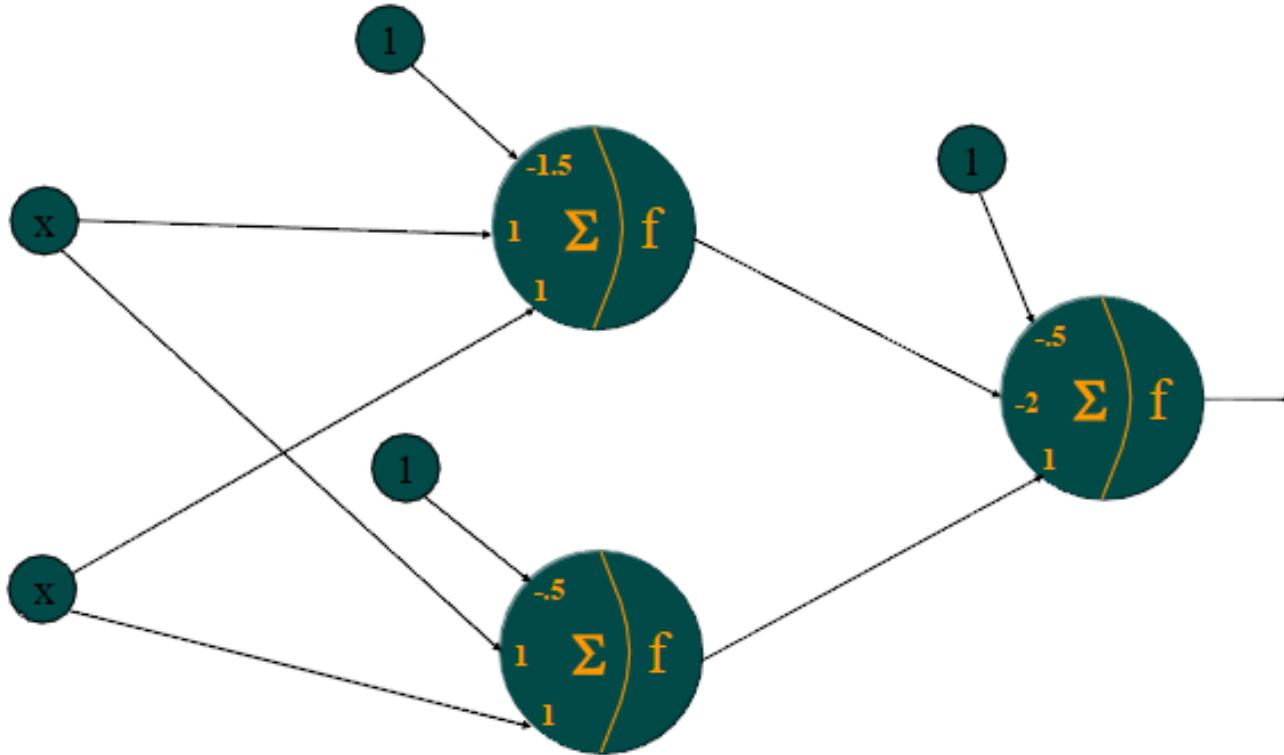
# پرسپترون چندلایه (مثال) . . .

- تابع XOR نمایش دودویی (۵ از ۶) . . .
- بروز کردن وزن‌ها



# پرسپترون چندلایه (مثال) . . .

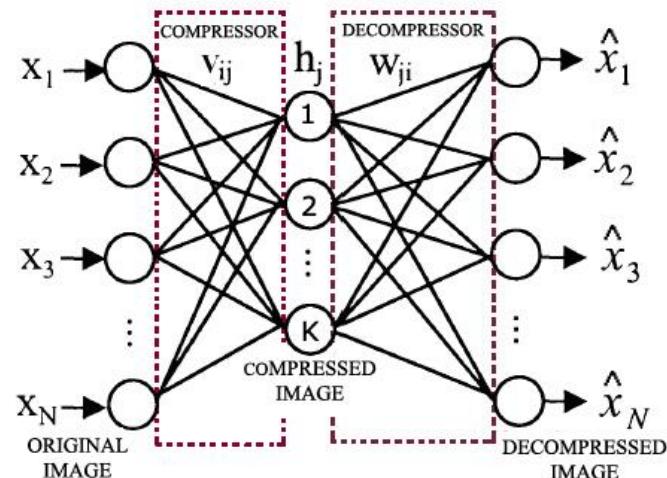
- تابع XOR نمایش دودویی (۶ از ۶)
- وزن‌های نهایی (بعد از ۵۰۰ تکرار)



# شبکه‌های پس انتشار: مثال

## ○ فشرده‌سازی داده‌ها

- فشرده‌سازی با اتلاف تصویر
- شکستن تصویر به بلوک‌های کوچک (مثلا  $8 \times 8 = 64$  نرون ورودی و خروجی)
- مقدار ورودی و خروجی (خروجی تابع فعال‌سازی) پیوسته هستند



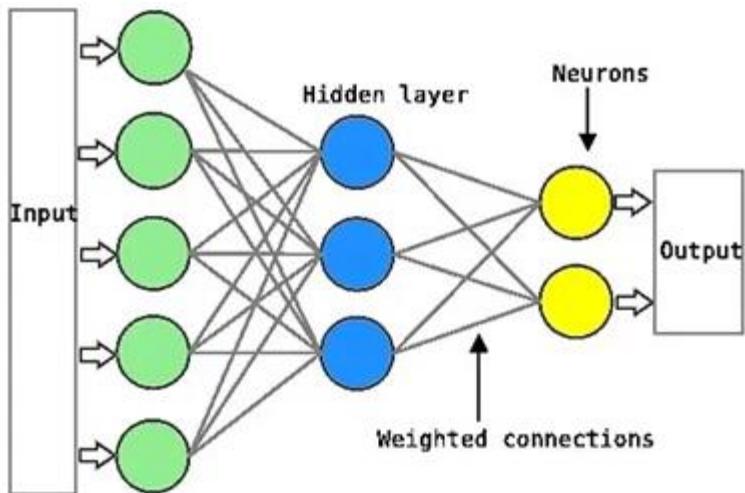
# پرسپترون چندلایه (مثال) . . .

## ○ کاربرد در پردازش گفتار و پردازش زبان

- تشخیص گوینده
- تشخیص جنسیت
- جداسازی گفتار از غیر گفتار

## ○ تعیین نقش دستوری (POS Tagging)

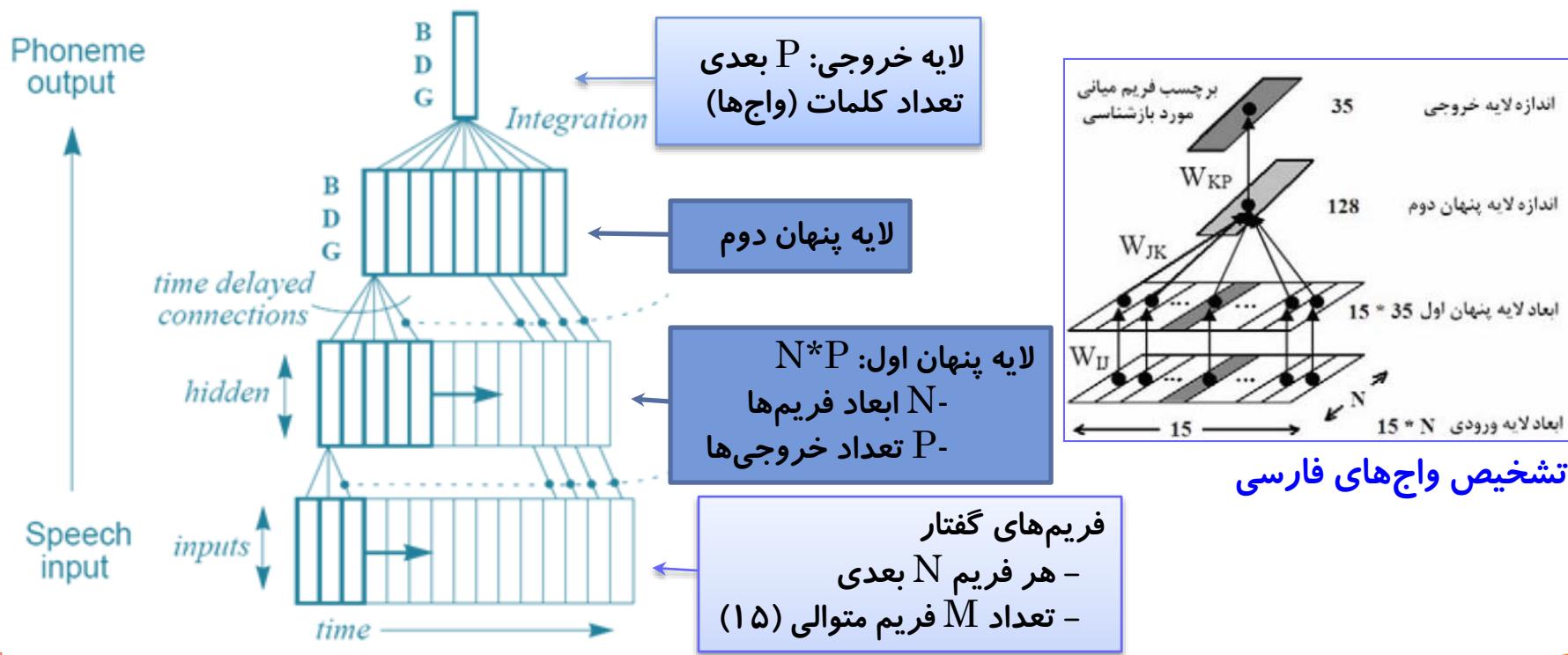
- تعیین شباهت دو متن
- تعیین عنوان متن
- ...



# پرسپترون چندلایه (مثال) . . .

## TDNN: Time-Delay Neural Network با تشخیص گفتار

- ورودی: دنباله متوالی از فریم‌های سیگنال گفتار
- خروجی: واحد موردنظر در تشخیص (کلمه، واژ)

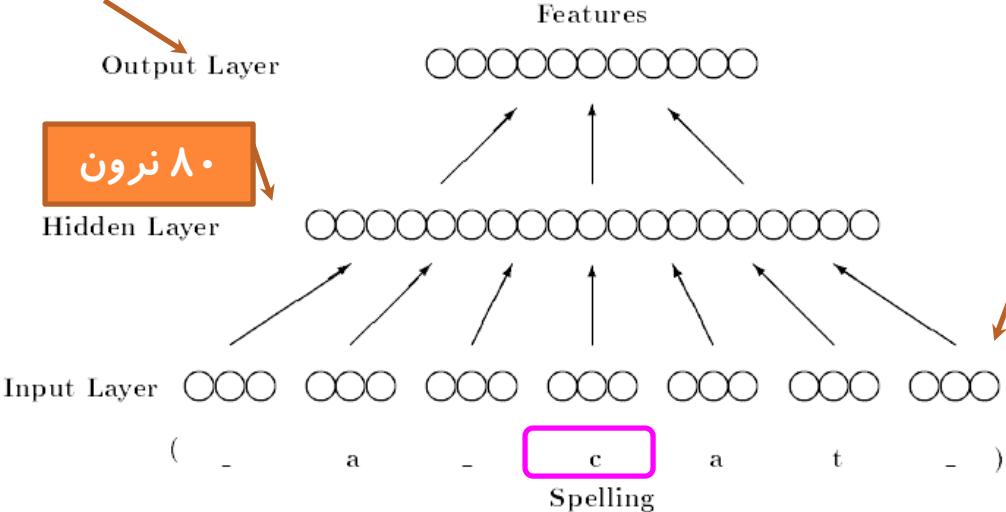
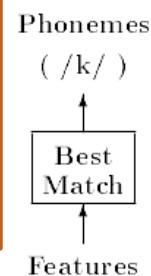


# پرسپترون چندلایه (مثال) . . .

## ○ متن خوان انگلیسی NETtalk

- ورودی: متن نوشته شده و خروجی: واژهایی که صدا را می‌سازند
- وابستگی واژه‌ها به محتوا
- تفاوت تلفظ 'a' در "have", "brave" (کشیده) و "gave" (کوتاه)
- تبدیل متن به صورت واژی
- Phone → f o n (f-on-)

۲۶ نرون = ۲۶ ویژگی  
 ۲۳ ویژگی زبانی (صدادار،  
 بی‌صدا، خیشومی و ...)  
 ۳ ویژگی نوایی (استرس و ...)



۷ = واژ مورد نظر و ۳ واژ در  
 دو طرف آن  
 ۲۹ = بردار دودویی، هر بعد  
 معادل یک واژ

# پر سپترون چند یه (مثال) . . .

## بردار کلمات . . .

Word Embedding , Word2Vec •

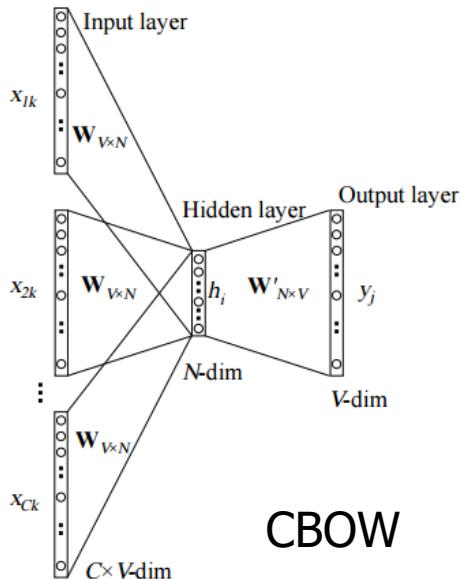
روش‌ها •

○ سبد کلمات پیوسته (CBOW: continuous bag-of-words)

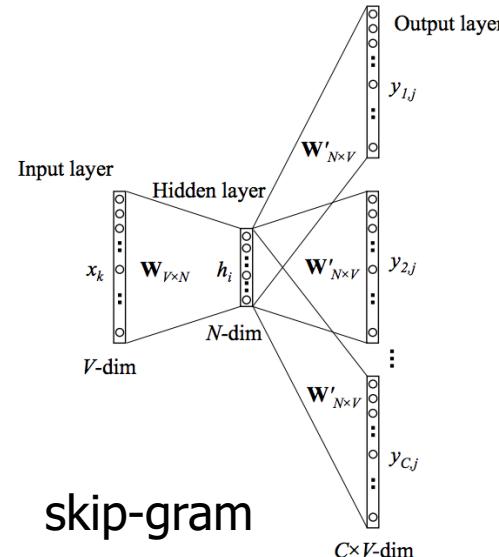
○ پیش بینی کلمه با در نظر دو (چند) کلمه قبل و دو (چند) کلمه بعد

○ پرس چندتایی (skip-gram)

○ پیش بینی کلمات (احتمال همه کلمه‌های بعدی) از روی کلمه فعلی



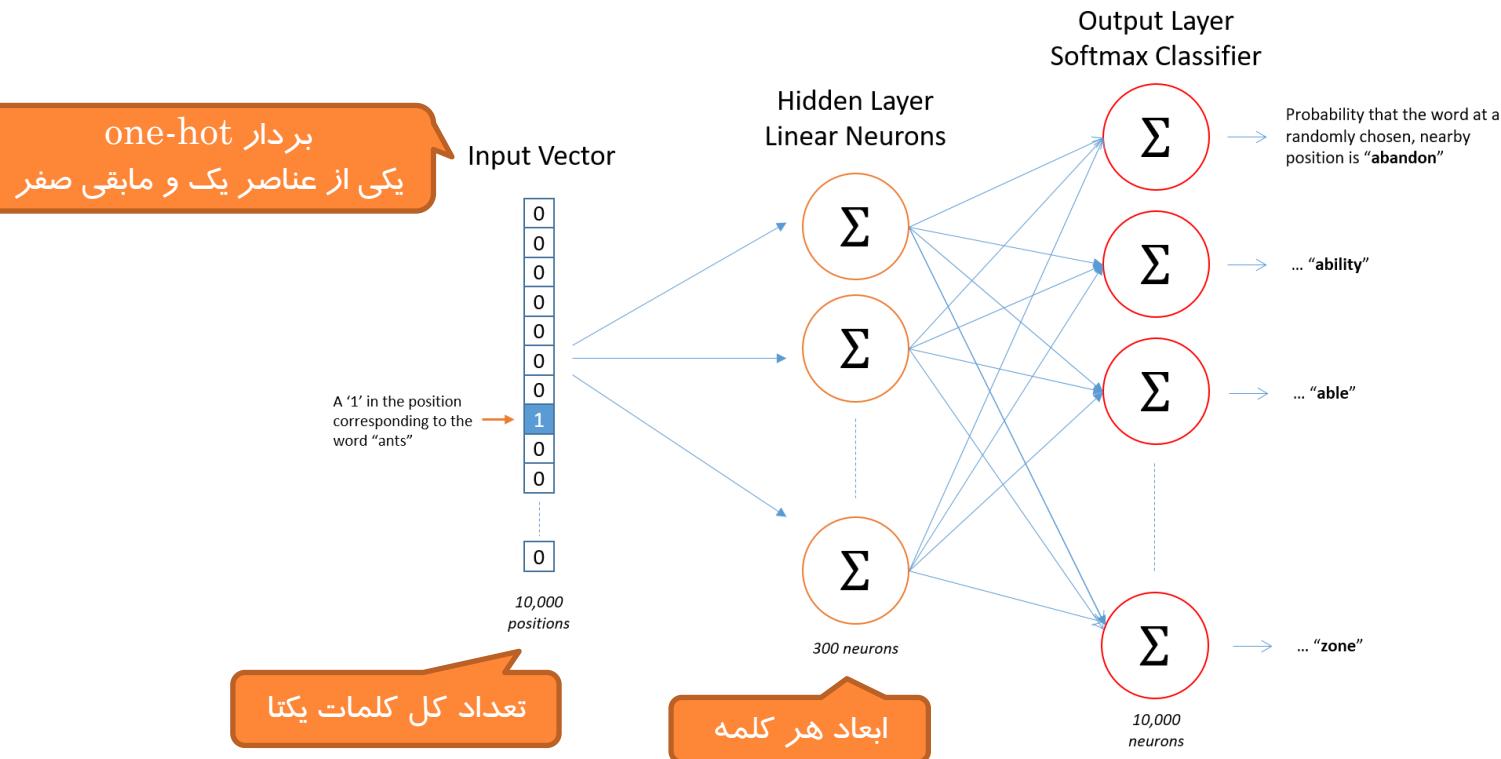
CBOW



skip-gram

# پرسترون چندیه (مثال) . . .

## بردار کلمات: پرس چندتایی



# پرسپترون چندلایه (مثال) . . .

- بردار کلمات: پرس چندتایی
- داده ورودی

Source Text

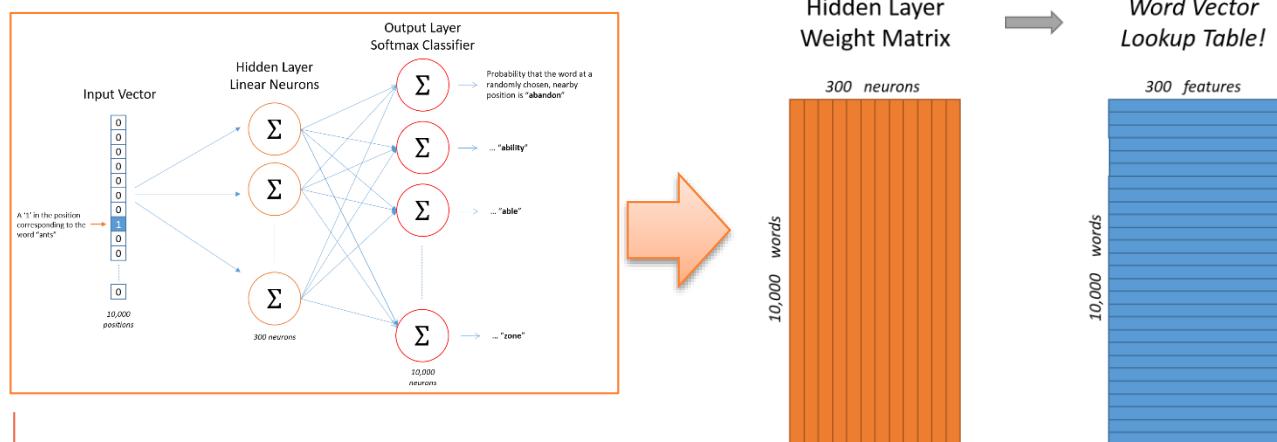
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

- بردار کلمات نهایی

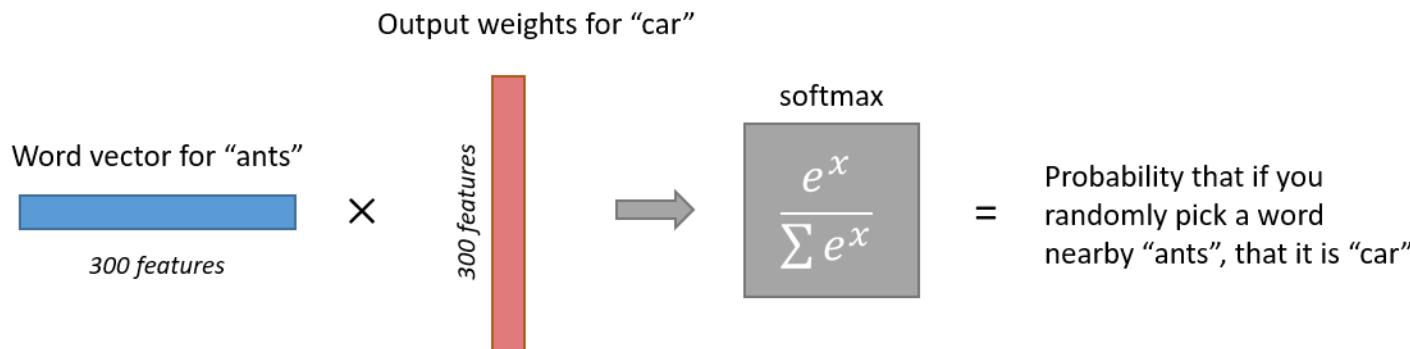


# پرسترون چندی به (مثال) . . .

## ○ بردار کلمات: پرش چندتایی

- لایه خروجی

- تابع فعال سازی SoftMax: تولید خروجی بین صفر و یک و جمع همه خروجی ها برابر با یک
- به ازای هر کلمه یک نرون



# پرسترون چندی به (مثال) . . .

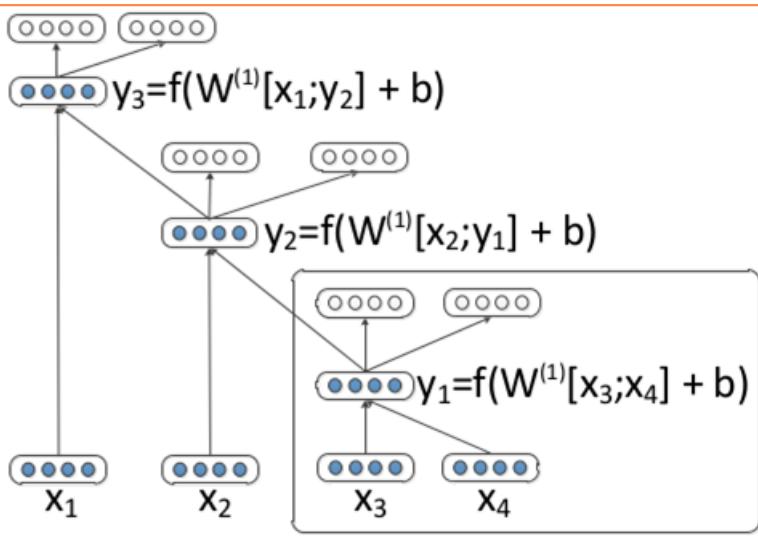
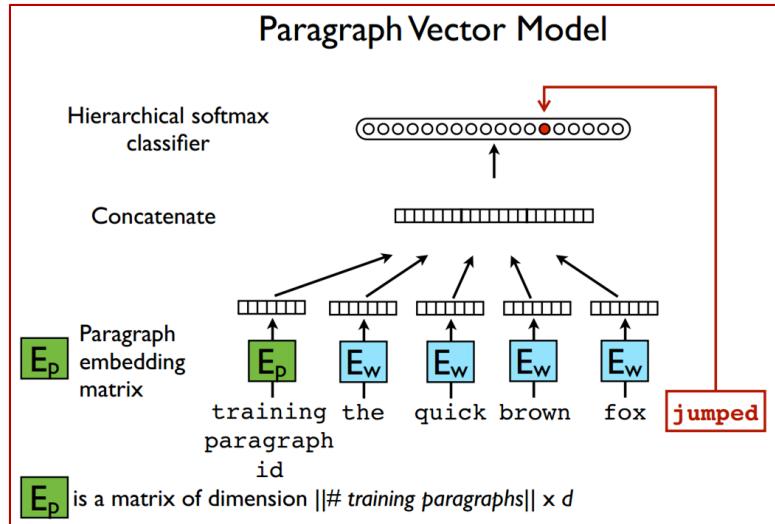
## ○ بردار جمله/پاراگراف/سند

- ترکیب بردار کلمات ساده

○ اصفهان+ (و دخانه) → زاینده (و)

Model	Function
Additive	$P_i = u_i + v_i$
Kintsch	$P_i = u_i + v_i + n_i$
Multiplicative	$P_i = u_i v_i$
Tensor product	$P_{ij} = u_i + v_j$
Circular convolution	$P_i = \sum_j u_j v_{i-j}$
Weighted additive	$P_i = \alpha v_i + \beta u_i$
Dilation	$P_i = v_i \sum_j u_j u_j + (\lambda - 1) u_i \sum_j u_j v_j$
Head only	$P_i = v_i$
Target unit	$P_i = v_i(t_1 t_2)$

# پرسپترون چندلایه (مثال)



*R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in Advances in Neural Information Processing Systems, pp. 801-809, 2011.*

# پرسپترون چندلایه: نکات تکمیلی ...

## ○ انتخاب مقادیر اولیه

- تأثیر مقادیر وزن‌های اولیه بر همگرایی شبکه به حداقل خطای سراسری (Global) یا فقط همگرایی شبکه به حداقل خطای محلی (Local)
- مقادیر اولیهٔ تصادفی (مثبت یا منفی)
- بازه متداول برای مقادیر تصادفی وزن‌ها و بایاس‌ها بین  $-0.5$  و  $0.5$ .

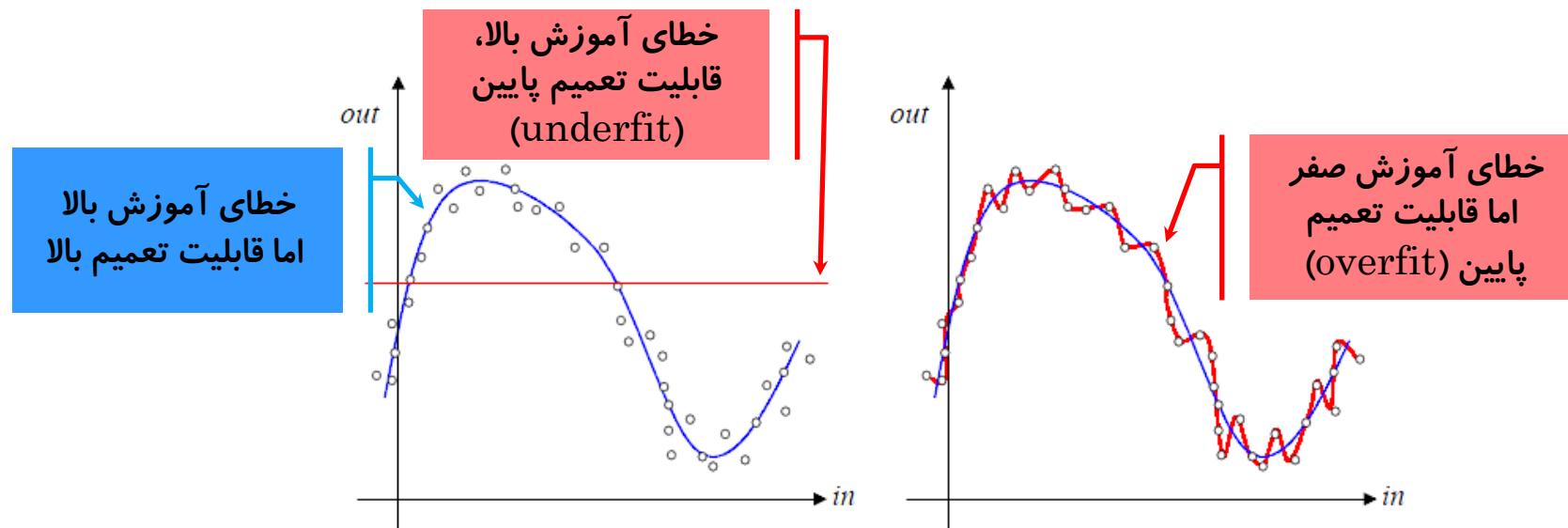
## ○ آموزش شبکه عصبی با بیش از یک لایه مخفی

- مشابه الگوریتم آموزش با یک لایه مخفی
- محاسبه‌ها برای هر لایه مخفی اضافی مشابه لایه مخفی بیان شده در الگوریتم است
- برای هر لایه مخفی، گام ۴ در مرحله پیش‌خور و گام ۷ در مرحله پساننتشار تکرار می‌شود.
- یک لایه مخفی در شبکه پساننتشار برای تقریب زدن هر نگاشت پیوسته‌ای از الگوهای ورودی به الگوهای خروجی با میزان دلخواهی از دقت کافی است.
- در برخی شرایط استفاده از دو لایه مخفی، آموزش شبکه را آسان‌تر می‌کند.

# پرسپترون چندلیه: نکات تکمیلی ...

## ○ تعادل بین یادگیری الگوها و تعمیم

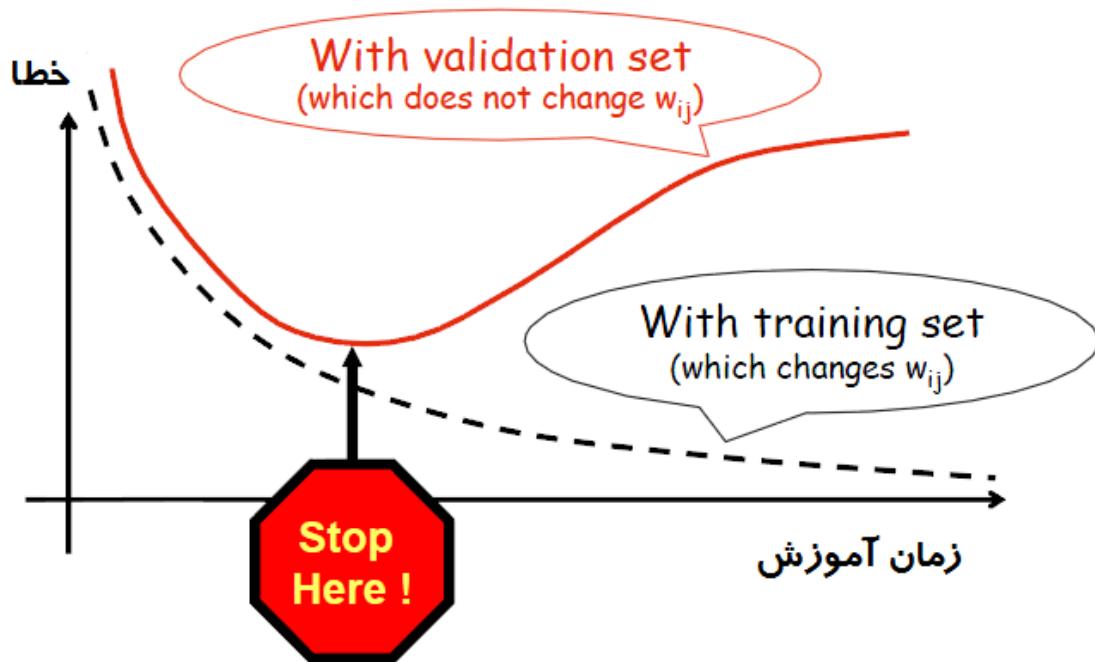
- پاسخ صحیح به الگوهای آموزش داده شده به شبکه + تولید پاسخ مناسب به الگوهای جدید
- شبکه قوانین حاکم بر داده‌ها را یاد بگیرد نه فقط نمونه‌های آموزش
- ادامه آموزش شبکه زمانی که مقدار مربعات خطا واقعاً حداقل شده، الزاماً مفید نمی‌باشد



# پرسپترون چندلایه: نکات تکمیلی ...

○ نکاتی که قابلیت تعمیم را افزایش می‌دهد (جلوگیری از Overfitting)

- تعداد نمونه‌های کمتر در لایه مخفی (پیچیدگی کمتر)
- توقف به موقع شبکه با افزایش خطای مجموعه ارزیابی (تست)
- داده‌های آموزش پوششی از انواع و تنوع نمونه‌ها باشد



# پرسپترون چندلیه: نکات تکمیلی . . .

- به روز کردن وزن با پس انتشار با گشتاور (Momentum)
- تغییر روش کاهش گرادیان: مقدار تغییر وزن ترکیبی از گرادیان (شیب) فعلی و گرادیان قبلی
- به روز شدن وزن‌های زمان  $t+1$  وابسته به وزن‌های زمان‌های قبل تر (مانند  $t$  و  $t-1$ )

$$w_{jk}(t+1) = w_{jk}(t) + \alpha \delta_k z_j + \mu [w_{jk}(t) - w_{jk}(t-1)]$$

$$v_{ij}(t+1) = v_{ij}(t) + \alpha \delta_j x_i + \mu [v_{ij}(t) - v_{ij}(t-1)]$$

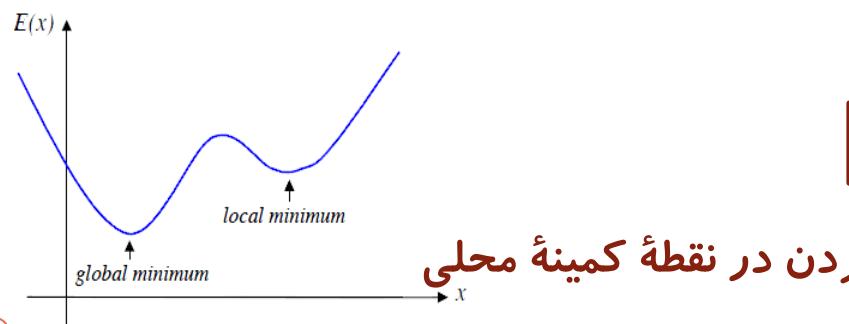
$$\Delta w_{jk}(t+1) = \alpha \delta_k z_j + \mu \Delta w_{jk}(t)$$

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

گرادیان فعلی

گرادیان قبلی

پارامتر ممان (بین ۰ تا ۱)



- همگرایی سریع‌تر + کاهش احتمال گیر کردن در نقطه کمینه محلی

# پرسپترون چندلیه: نکات تکمیلی ...

## ○ تعداد داده‌های آموزش: قاعدهٔ تجربی

- $P$  = تعداد الگوهای آموزش موجود،
- $W$  = تعداد وزن‌های مورد آموزش در شبکه
- $e$  = صحت دسته‌بندی مورد نظر

- آموزش شبکه برای دسته‌بندی صحیح کسری معادل  $(e/2)-1$  از الگوهای آموزشی،
- می‌توان مطمئن بود که شبکه  $e-1$  الگوی آزمایش را نیز به درستی دسته‌بندی کند؟

$$P = \frac{W}{e} \quad \text{یا} \quad \frac{W}{P} = e \quad \text{کافی بودن الگوهای آموزشی :}$$

- مثال: با  $e=0.1$ ، شبکه‌ای با ۸۰ وزن، ۸۰۰ الگوی آموزش لازم خواهد داشت تا از دسته‌بندی صحیح ۹۰٪ الگوهای آزمایش اطمینان حاصل شود، با این فرض که شبکه برای دسته‌بندی صحیح ۹۵٪ الگوهای آموزشی، آموزش دیده باشد.

# پرسپترون چندلیه؛ نکات تکمیلی ...

## ○ به روز کردن دسته‌ای (Batch Updating)

- به جای به روز کردن وزن‌های شبکه بعد از ارائه هر الگوی آموزشی
- ادغام مقدار تصحیح (تغییر) وزن را برای چند الگو یا برای تمام الگوهای در یک دور کامل
- تشکیل یک مقدار تنظیم وزن برای هر وزن، برابر با میانگین عبارات تصحیح وزن‌ها
- آسان‌تر کردن تصحیح وزن‌ها
- مقاوم بودن در برابر داده‌های نویزی
- موازی‌سازی محاسبات
- افزایش احتمال نزدیک شدن به کمینه محلی

# پرسپترون چندلایه: نکات تکمیلی

- شبکه MLP تقریب‌زننده‌های جهانی است: قضیه هج-نیلسون

- هر تابع پیوسته  $f: I^n \rightarrow R^m$  را که در آن  $I$  بازه بسته  $[0,1]$  است، می‌توان دقیقاً با یک شبکه عصبی پیش‌خور با  $n$  واحد ورودی،  $2n+1$  واحد مخفی و  $m$  واحد خروجی نمایش داد.

$$z_j = \sum_{i=1}^n \lambda^i \psi(x_i + \varepsilon j) + j$$

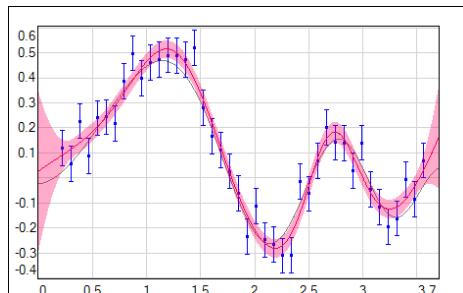
عددی ثابت و  
حقيقي

تابع پیوسته، حقيقی و یکنواهی صعودی،  
مستقل از  $f$  و وابسته به  $n$

- مقدار ثابت  $\varepsilon$  برای فراهم بودن شرایط قضیه اسپرچر است.

$$y_k = \sum_{j=1}^{2n+1} g_k z_j$$

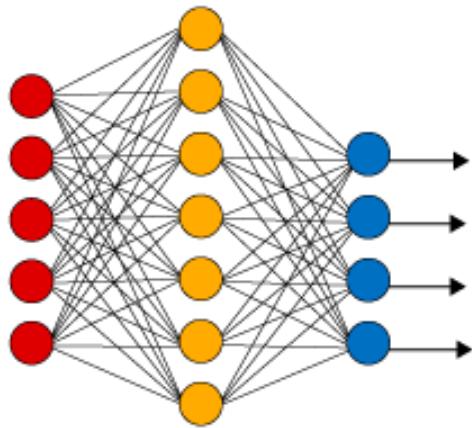
تابع پیوسته، حقيقی و وابسته به  $f$  و  $\varepsilon$



# یادگیری عمیق . . .

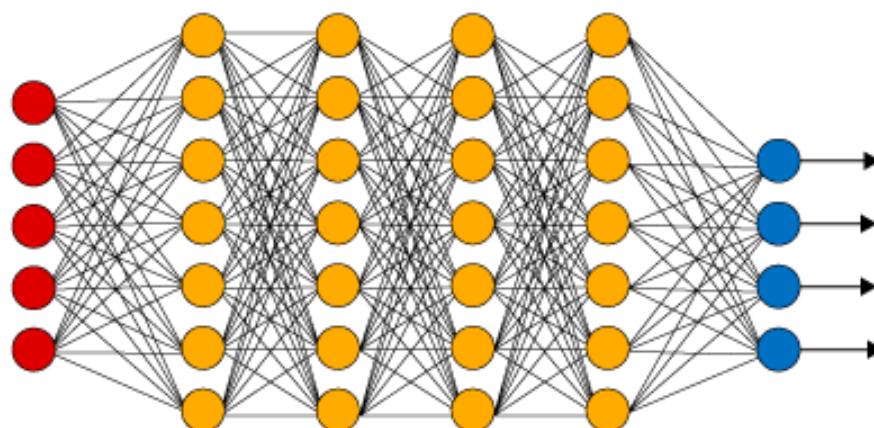
- افزایش تعداد لایه های مخفی

**Simple Neural Network**



● Input Layer

**Deep Learning Neural Network**



● Hidden Layer

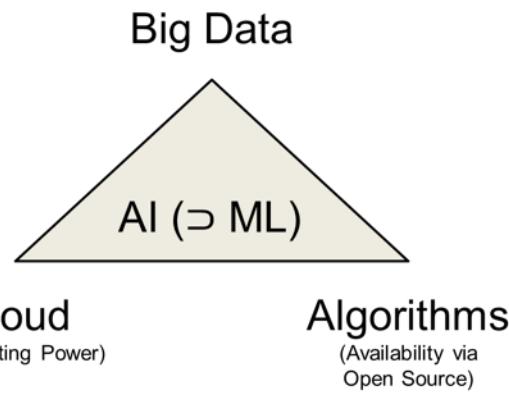
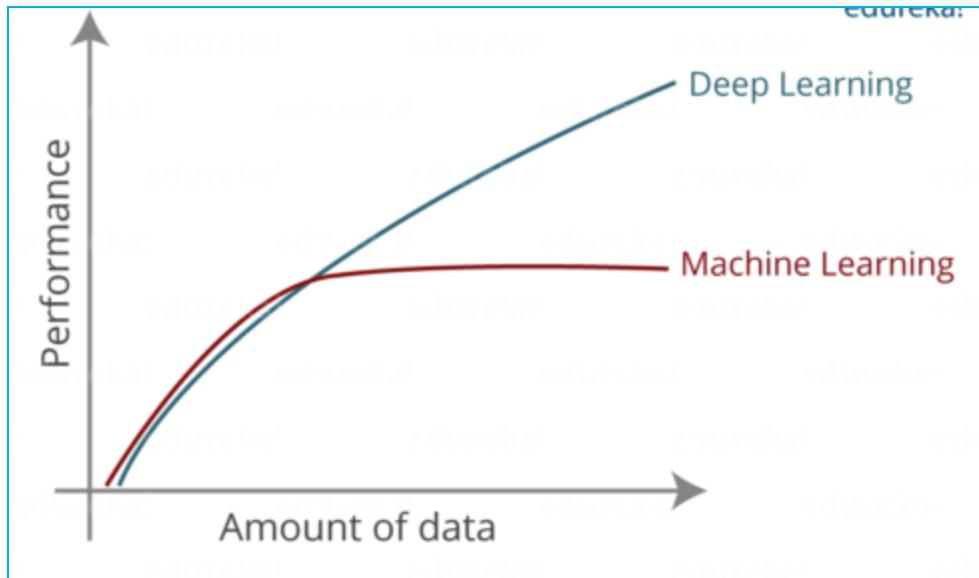
● Output Layer

- مدل پیچیده = تعداد پارامترهای زیاد = توان محاسباتی بالا

# یادگیری عمیق . . .

## ○ سه رکن اصلی

- داده جیم (Big Data)
- توان پردازشی (GPU)
- الگوریتم یادگیری

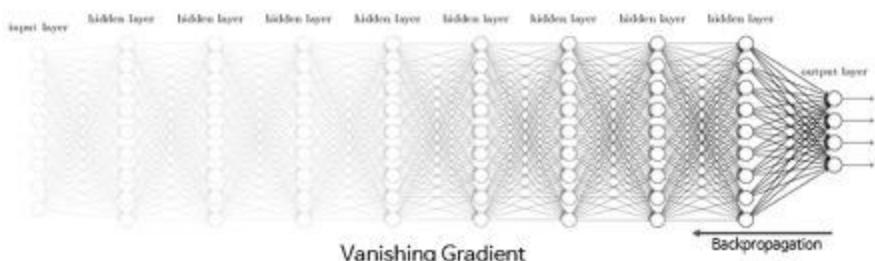
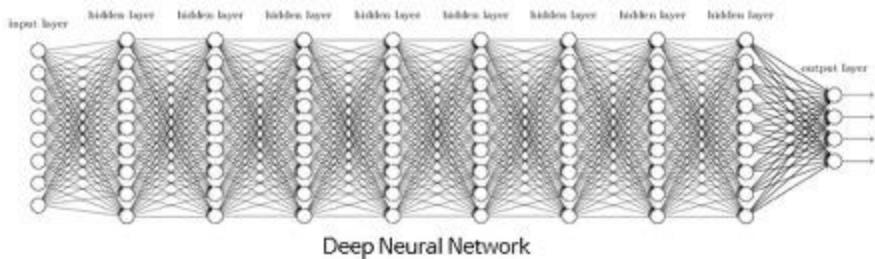


# یادگیری عمیق . . .

## مشکل: یادگیری

### • محو گرادیان (Vanishing Gradient)

- کاهش گرادیان برگشتی از لایه آخر به لایه اول



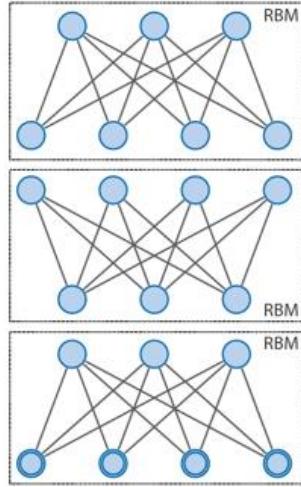
## • راه حل

- پیش آموزش شبکه برای تعیین وزن های اولیه مناسب

- استفاده از ماشین بولتزمن محدود (RBM)

- بهبود وزن ها با پس انتشار خطأ

# یادگیری عمیق؛ شبکه باور عمیق (DBN) ...



## ○ معماری

- از چند لایه ماشین بولتزمن محدود (RBM) ساخته می شود  
RBM: Restricted Boltzman Machine ○

## ○ آموزش

- ابتدا لایه زیرین (ابتدايی) را با داده های ورودی مقداردهی و آموزش می دهیم
  - استفاده از روش واگرایی متقابل (CD: Contrastive Divergence)
  - آموزش بدون نظارت
- با ويژگی های استخراج شده از لایه اول مثل داده ورودی برای لایه دوم برخورد می کنیم
  - در واقع ويژگی ويژگی ها استخراج می شود
  - ادامه اين روند تا رسیدن به لایه آخر



# شبکه باور عمیق (DBN): الگوریتم آموزش و جزئیات

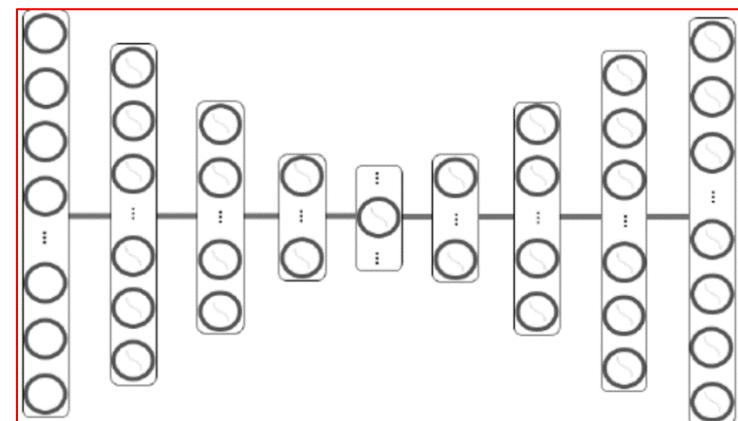
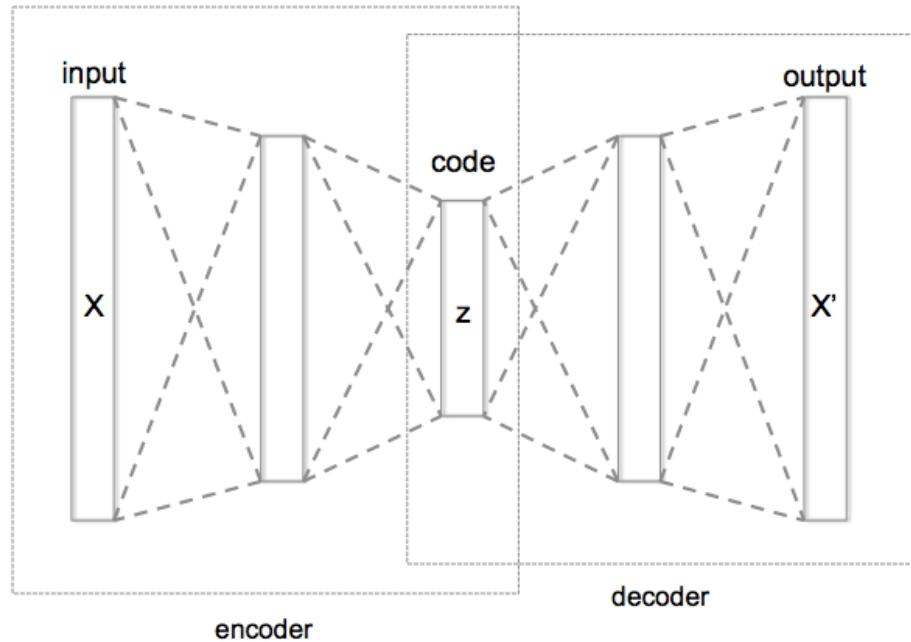
○ مراجعه کنید به

- <http://dsp.ut.ac.ir/en/wp-content/uploads/2018/04/ANN-Lecture4-DBN.pdf>

# شبکه های خودرمزگذار بازگشتی (RAE)

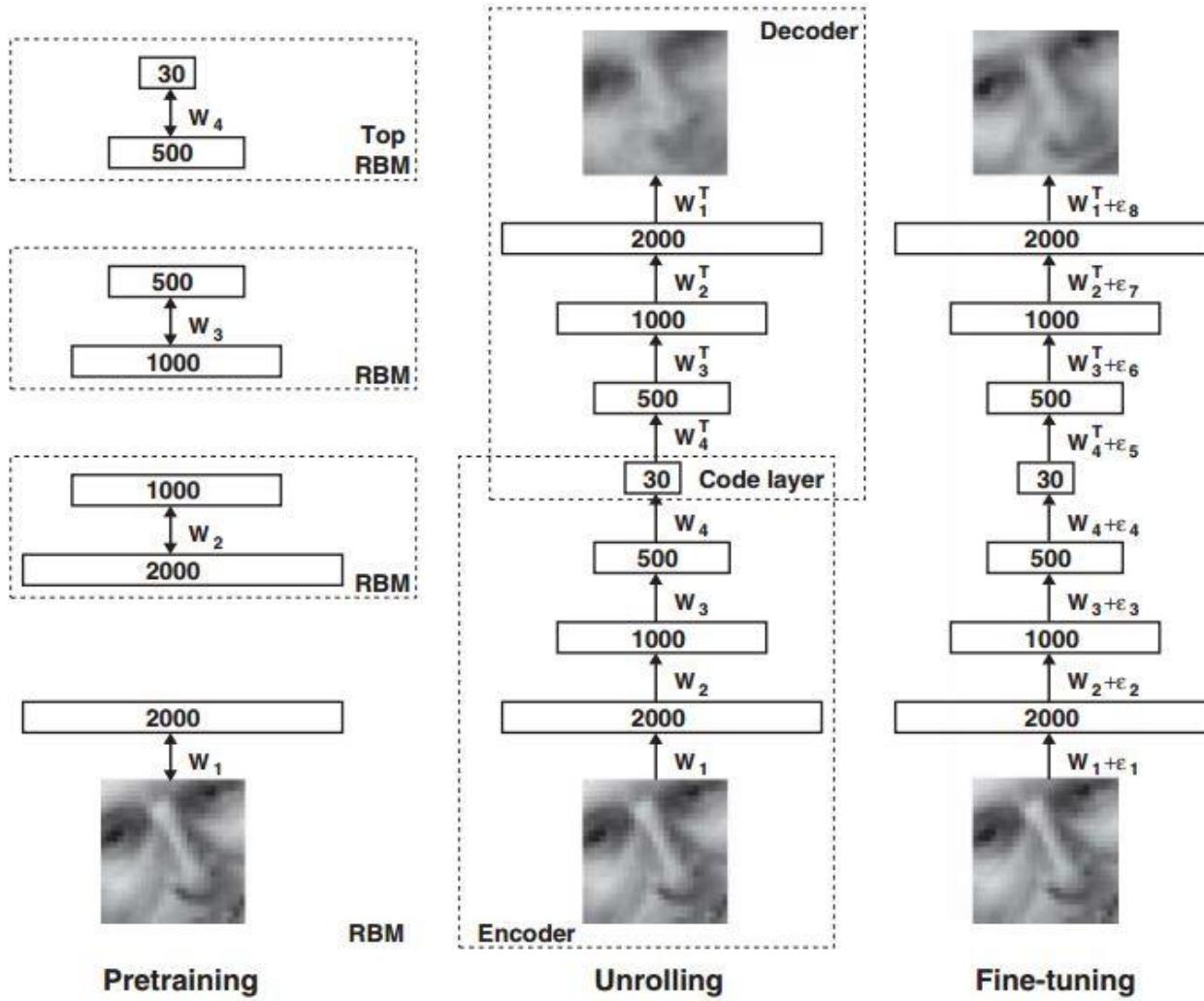
- شبکه های خودرمزنگذار بازگشتی (RAE: Recursive AutoEncoders)

- استفاده از لایه وسط به عنوان ویژگی

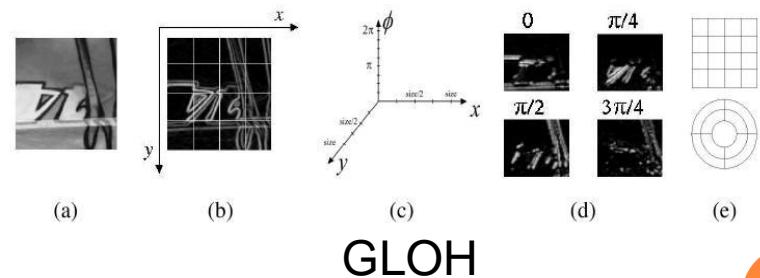
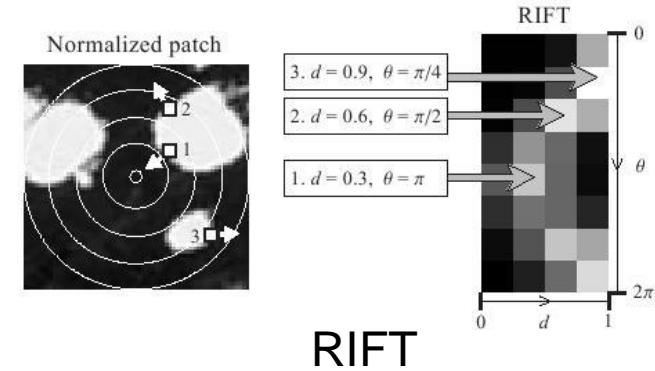
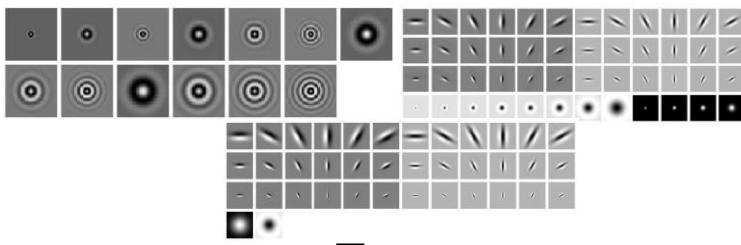
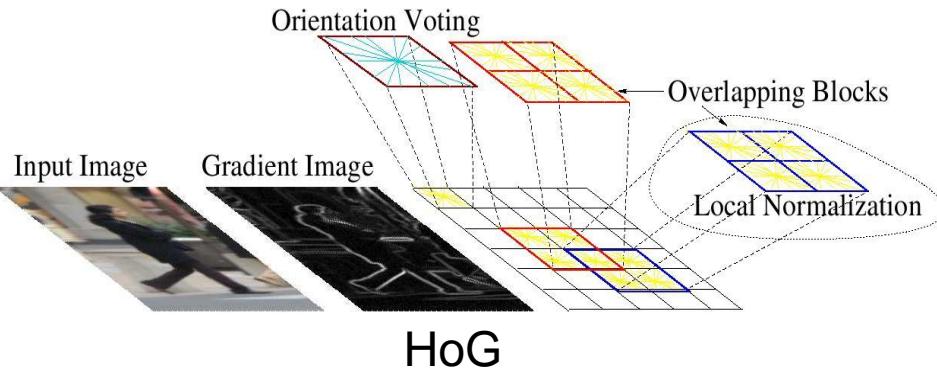
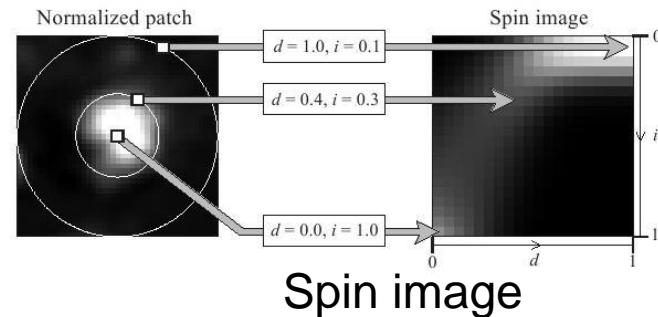
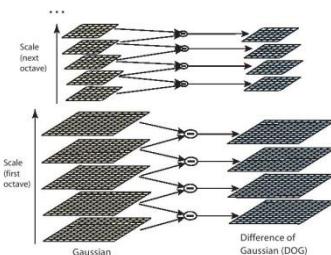
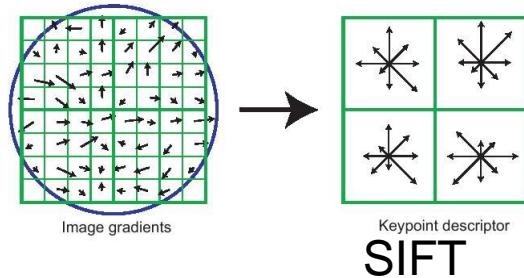


# پادگیری عمیق . . .

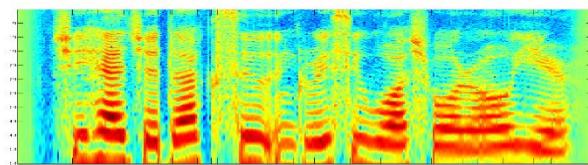
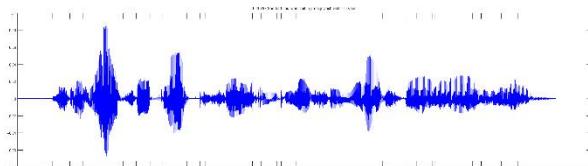
خودر مزگذار



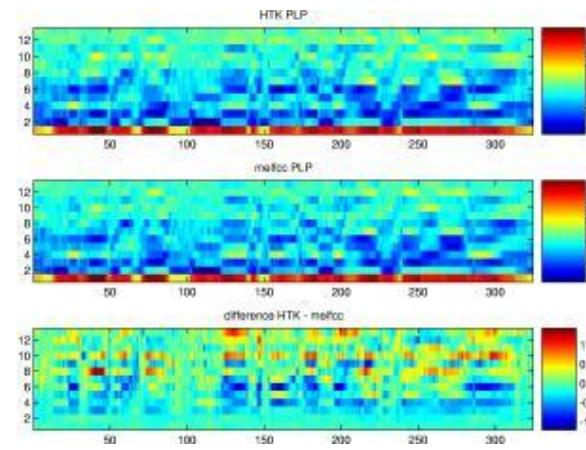
# یادگیری عمیق: استخراج ویژگی (تصویر)



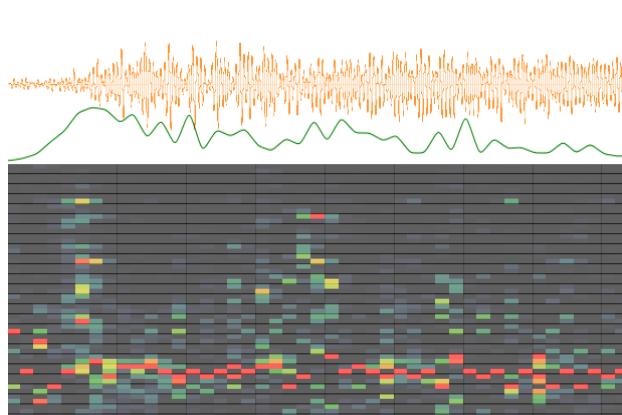
# یادگیری عمیق: استخراج ویژگی (صد)



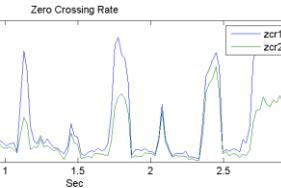
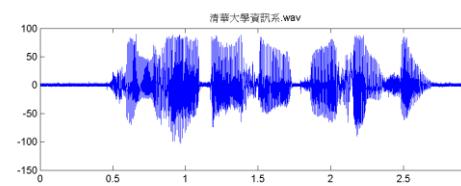
Spectrogram



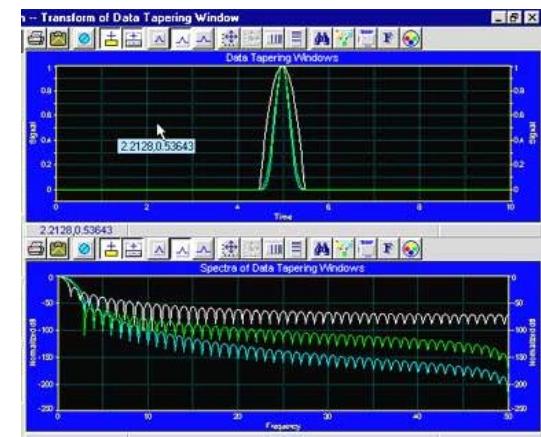
MFCC



Flux



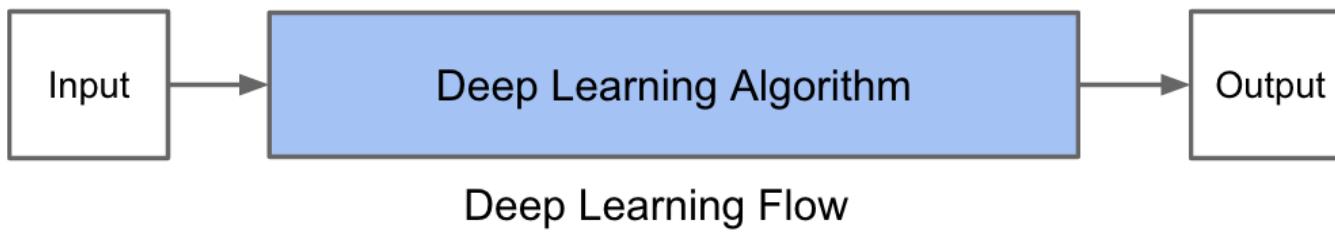
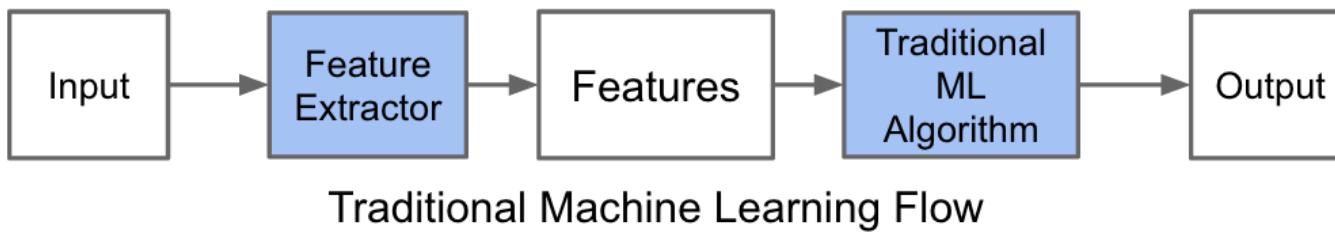
ZCR



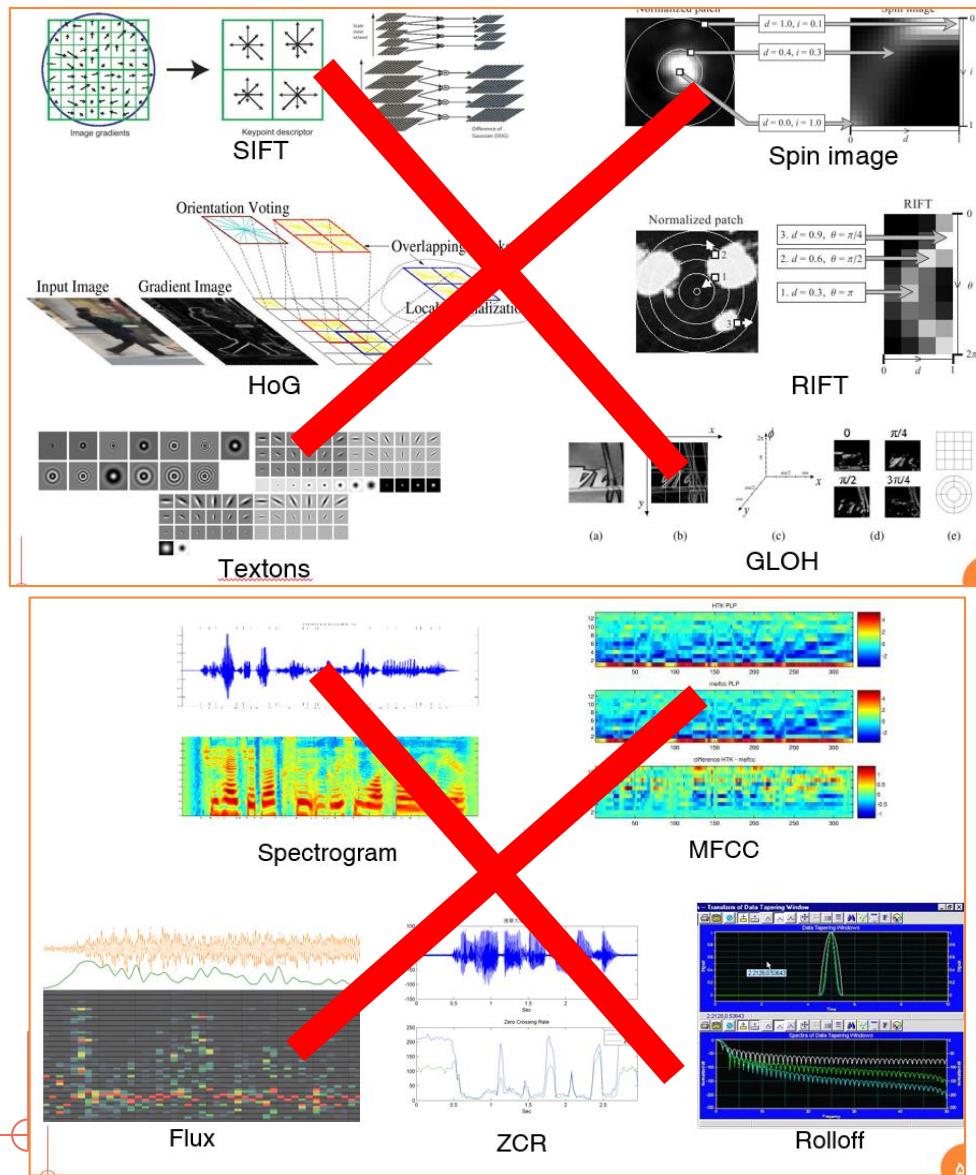
Rolloff

# یادگیری عمیق . . .

## ○ ترکیب استخراج ویژگی و دسته‌بندی



# پادگیری عمیق: استخراج ویژگی و دسته‌بندی



Unlabeled images

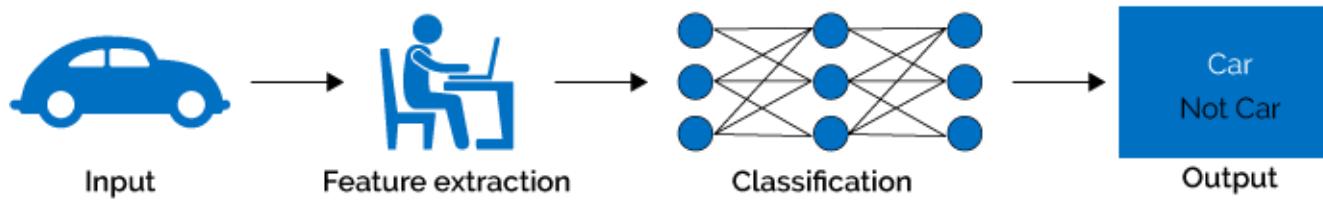
Learning  
algorithm

Feature representation

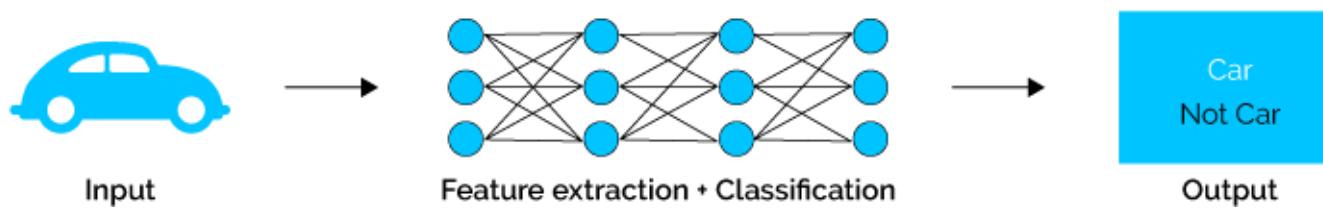
# پادگیری عمیق . . .

## ○ ترکیب استخراج ویژگی و دسته‌بندی

Machine Learning



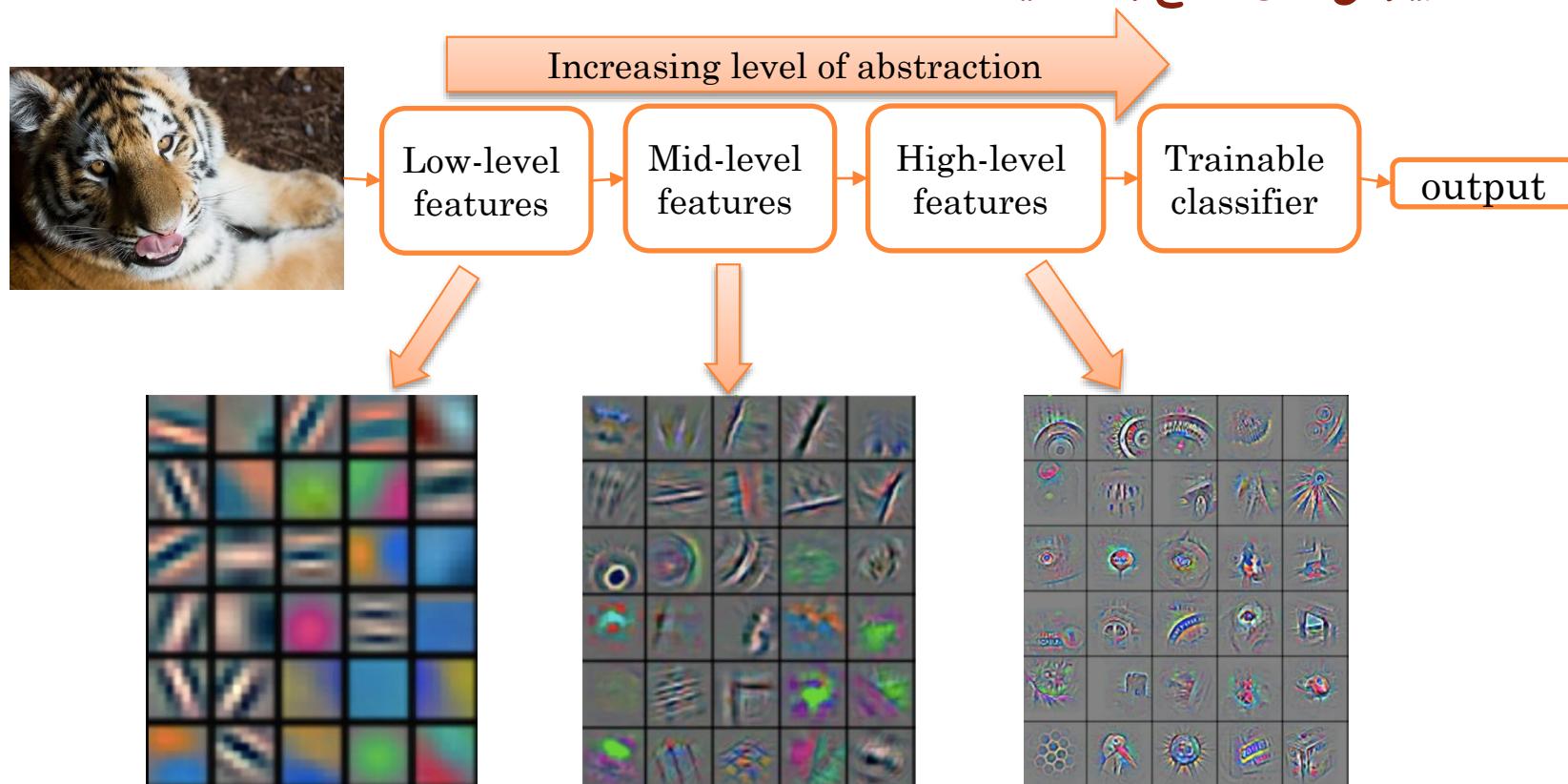
Deep Learning



# یادگیری عمیق: استخراج ویژگی ...

## ○ استخراج ویژگی سلسله مراتبی ...

- از ورودی خام (جزئیات)
- تا ویژگی های سطح بالا (کلیات)





# پادگیری عمیق؛ استخراج ویژگی

## ○ استخراج ویژگی سلسله مراتبی

● تصویر

Pixel → edge → texton → motif → part → object ○

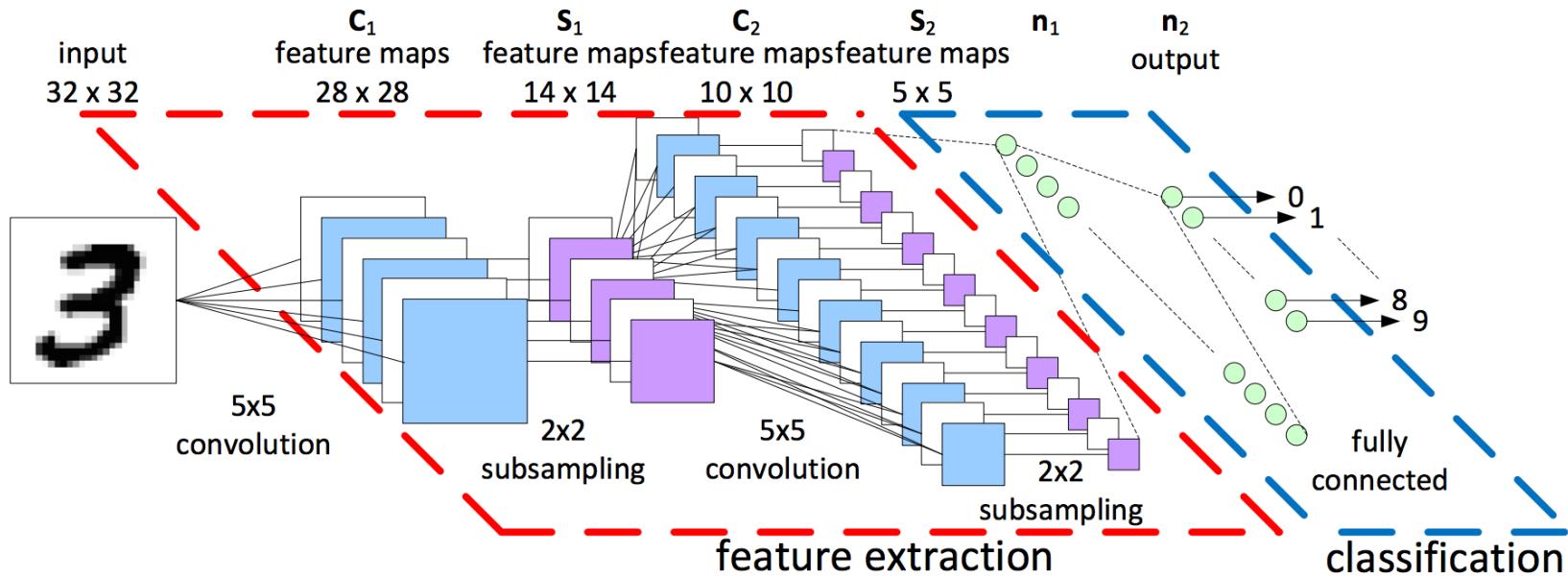
● متن

Character → word → word group → clause → sentence → story ○

# یادگیری عمیق: شبکه عصبی پیچش ...

## ◦ شبکه عصبی پیچشی (CNN: Convolutional Neural Network)

- استخراج ویژگی از تصاویر (و دسته‌بندی)



# یادگیری عمیق: شبکه عصبی پیچش ...

- لایه‌های رایج در این شبکه (به ترتیب)

- ورودی (Input)

● یک تصویر ۳ بعدی: طول، عرض و عمق (مثلاً معادل ۳ برابر با سه مولفه RGB)

- پیچش (Convolution)

● هر نرون بلوکی از پیکسل‌های نزدیک به هم را می‌بیند

● تابع فعال‌سازی ReLU

$$f(x) = \max(0, x)$$

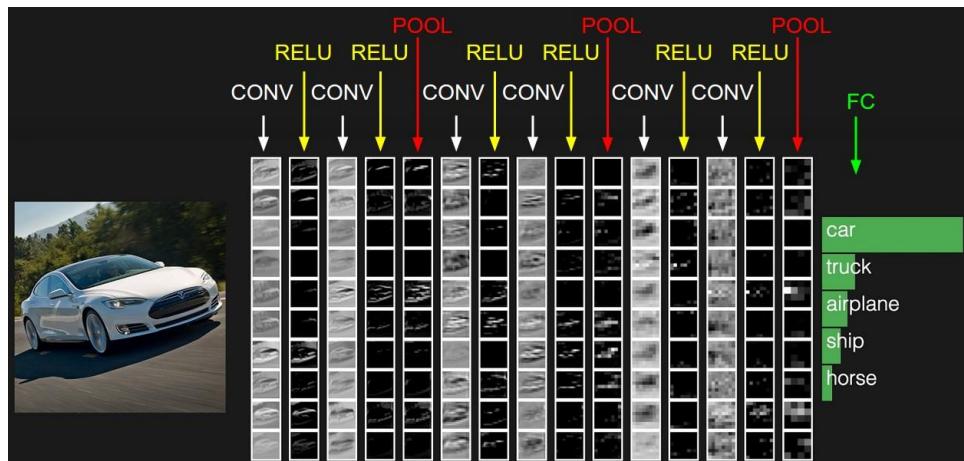
● غیرخطی کردن رفتار شبکه

- نمونه‌برداری (Pooling)

● کاهش اندازه فضای ویژگی‌ها

- تمام متصل (Fully Connected)

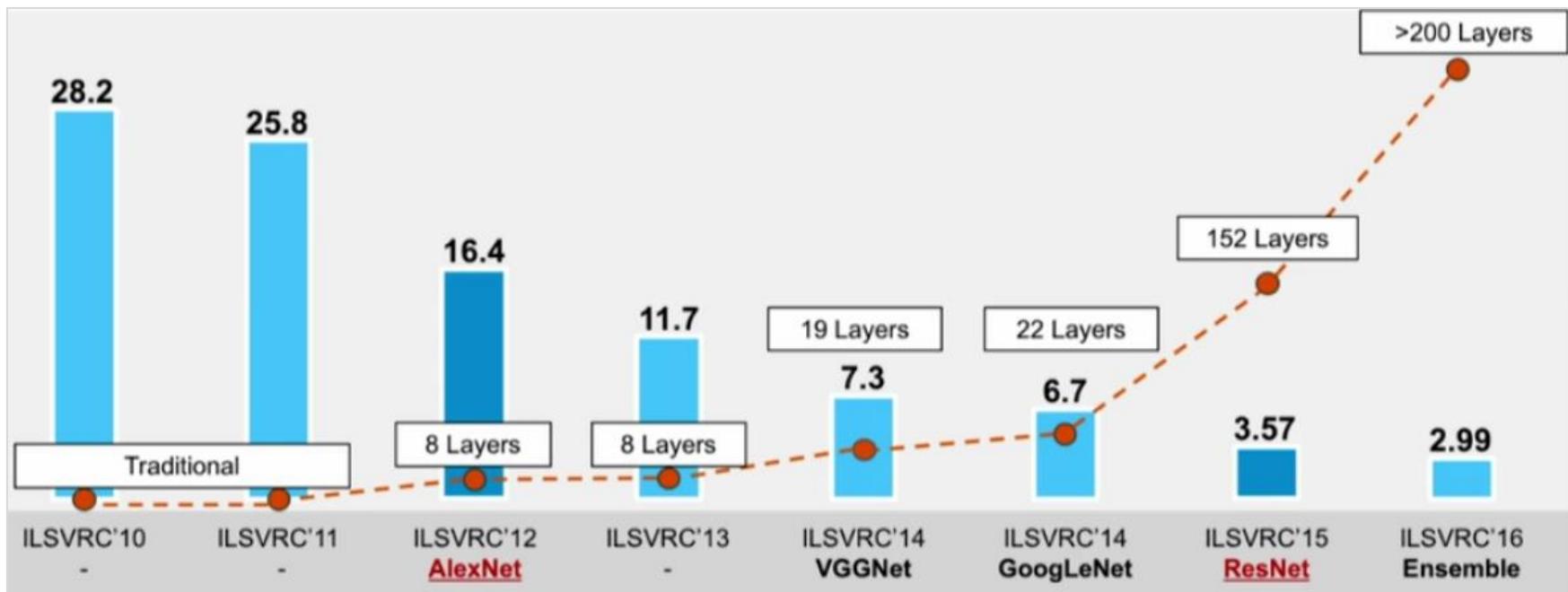
● برای دسته‌بندی، مانند شبکه‌های عصبی MLP



# یادگیری عمیق: شبکه عصبی پیچش ...

## ○ نرخ خطای دسته‌بندی روی ImageNet

- ۱.۲ میلیون تصویر در ۱۰۰۰ دسته
- نزدیک شدن کارایی شبکه پیچشی به عملکرد انسان



# پادگیری عمیق: تنظیم ...

## ○ تنظیم شبکه (Regularization)

- روش‌هایی برای افزایش قدرت تعمیم پذیری شبکه و جلوگیری از بیش برازش



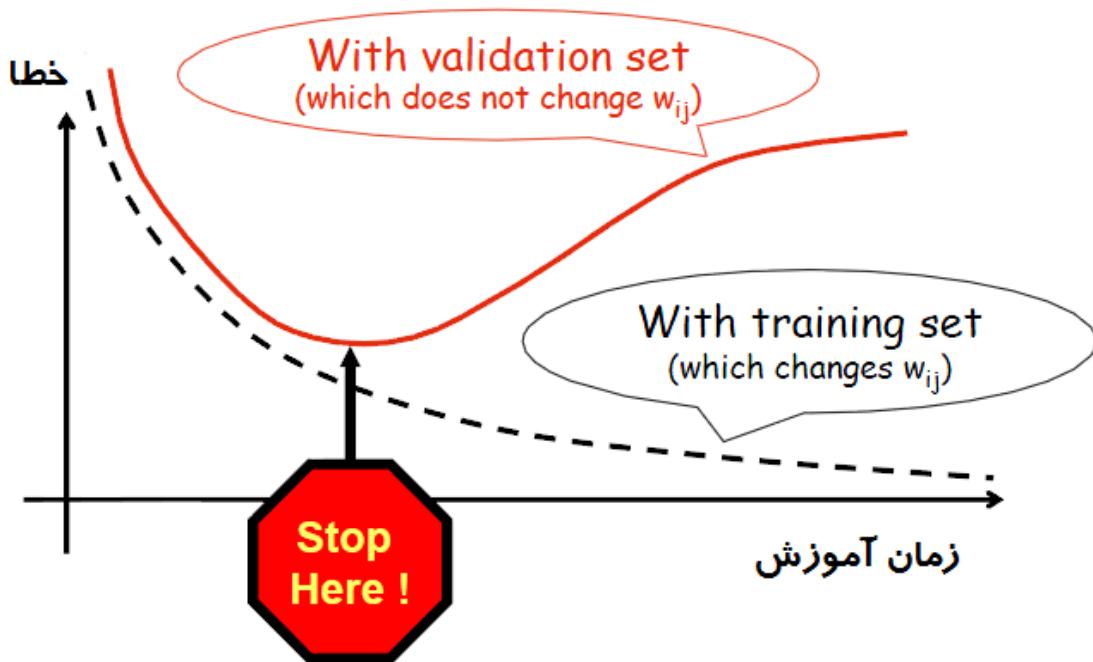
## ○ روش‌ها

- توقف زودهنگام (Early Stopping)
- تغییر تابع هزینه
- حذف تصادفی (Dropout)
- تقویت داده (Data Augmentation)

## پادگیری عمیق: تنظیم ...

### ○ توقف زودهنگام (Early Stopping)

- جلوگیری از تکرار زیاد و وابسته شدن کامل شبکه به داده آموزشی
- استفاده از مجموعه اعتبارسنجی یا اعتبارسنجی متقطع
- توقف در صورت افزایش خطا روی داده اعتبارسنجی



# یادگیری عمیق: تنظیم ...

## ○ تغییر تابع هزینه

- افزودن مقداری به مقدار خطای مورد استفاده در تابع هزینه
- $Cost\ function = Loss + Regularization\ term$

مقدار خطای MSE

- عبارت افزوده سعی در کاهش مقدار وزن‌های شبکه دارد
- وزن‌های کوچکتر = حفظ سادگی ساختار شبکه

ضریب تنظیم

$$Cost\ function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$$

• تنظیم L2

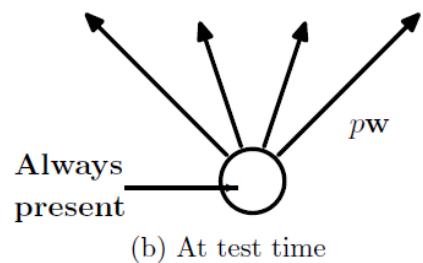
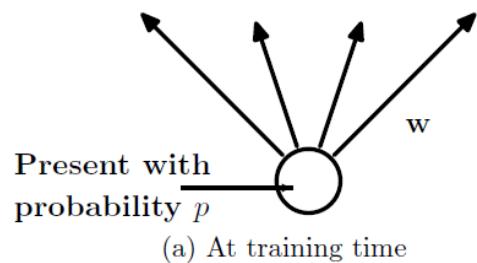
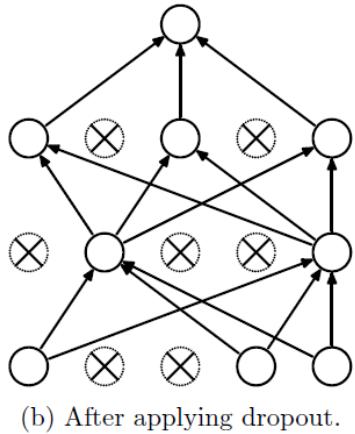
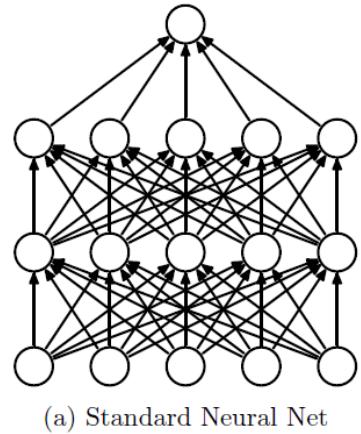
◦ به آن weight decay هم می‌گویند

◦ اثر کاهش وزن به صورت  $W = W - \lambda * \nabla$  به سمت صفر

$$Cost\ function = Loss + \frac{\lambda}{2m} * \sum \|w\|$$

• تنظیم L1

# یادگیری عمیق: تنظیم ...



## ○ حذف تصادفی (Dropout)

- در زمان آموزش
- هر نرون با احتمال  $p$  حذف می شود
- حذف اتصال ورودی و خروجی
- مقدار نوعی  $p=0.5$

## ● در زمان آزمون

- همه نرون ها وجود دارند
- اما وزن آنها در  $p$  ضرب می شود

## ● عملکرد مشابه آموزش گروهی (Ensemble Learning)

# پادگیری عمیق: تنظیم

## ○ تقویت داده (Data Augmentation)

- افزایش تنوع و حجم داده‌ها به صورت مصنوعی
- برای تصویر
  - جابجایی، تغییر مقیاس، معکوس، چرخاندن، برش و ...



- برای صدا
  - تغییر دامنه، افزودن نویز و ...

# یادگیری عمیق: پیش آموزش ...

## ○ ایده پیش آموزش (Pre-Train)

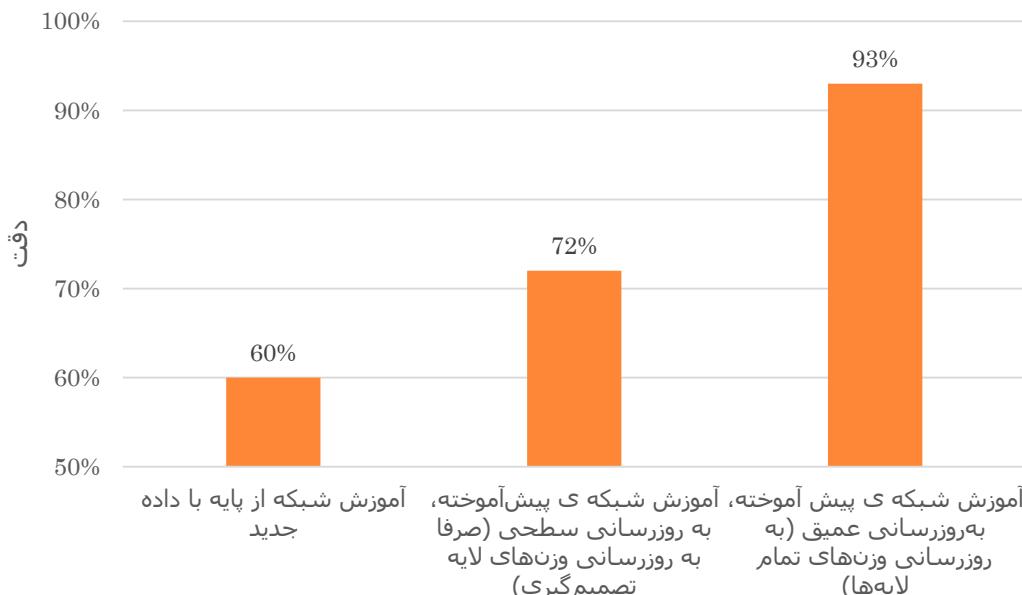
- آموزش شبکه با داده (حجیم) از یک حوزه و انتقال وزن‌های آموزش دیده برای استفاده از حوزه دیگر (به عنوان وزن‌های اولیه)
  - انتقال وزن‌های مربوط به لایه‌های استخراج ویژگی
  - صرفنظر کردن از لایه دسته بندی
- بهینه کردن وزن‌های انتقال یافته با داده حوزه جدید (Fine-tune)

## ◦ مرتبه با مفهوم یادگیری انتقالی (Transfer Learning)

# پادگیری عمیق: پیش آموزش

## ○ مثال: دسته‌بندی تصویر بیماری گیاهان

- وزن اولیه: از شبکه عصبی پیچشی AlexNet آموزش داده شده با داده ImageNet
  - ۱.۲ میلیون تصویر با ۱۰۰۰ دسته مختلف از موضوعات مختلف
- حوزه جدید: ۱۰ دسته (بیماری)، حدود ۳۵۰۰ تصویر برای هر دسته



# یادگیری عمیق: بسترها . . .

1994  
MLC++

2002  
Torch

2014-2016,  
Pattern, Caffe, cuDNN, DL4J, TensorFlow, MLlib



2000  
OpenCV

2006-2008  
Scikit, Theano, Accord

2010  
MOA

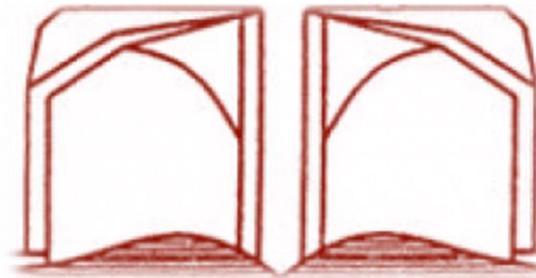
2012  
Mahout

Goldsborough, P., 2016. A tour of tensorflow. *arXiv preprint arXiv:1610.01178*.

Available at: <https://arxiv.org/pdf/1610.01178.pdf>

# پادگیری عمیق: بسترها

	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	



یادگیری ماشین

شبکه عصبی بازگشتی



# شبکه عصبی بازگشتی ...

## ● عقیده کاوی (دسته‌بندی نظرات کاربران)

### ● موافق/مخالف

+ 1

امید دلبر (1393/1/24) :

فوق العادس



+ 0

محمود ابراهیمی جاویدی (1393/1/22) :

من این لب تا 5 ماه دارم و از همه نظر عالی مثل: کیفیت صفحه نمایش با کیفیت لواش برای لمسی بودنش و ...



+ 1

مرتضی اکبری (1393/1/17) :

یکی از مشکلان ازینظر من کارت گرافیکشه.  
منتظوم باس کارت گرافیکها!

+ 0

هادی کریم (1392/12/22) :

سلام..اصلا تو خریدن شک نکنید..من خریدم خیلی عالیه..خیلی بیشتر از عکسیش خوشگله..وافعا زیباست..اصلا لولایی دیده نمیشه واستحکامش خیلی بیشتره..به قیمتیش هم ارزه..صدای فنیش هم اصلا اذیت نمیکنه..جون  
مده برای بازی

+ 5

ابودر هوفون (1392/12/17) :

با سلام

بعد از 4ماه کار مداوم

از هر نظر عالی

قررت خوبی داره..بسیار سریع  
تاج خیلی خوبی داره..کیفیت و افعا عالی..صفحه عجیبی داره که اصلا حسته نمیشنین موقع کار باهاس  
وزنیش هم که عالیه..یه همراه همیشگی میشه برآتون  
مرسی

+ 0

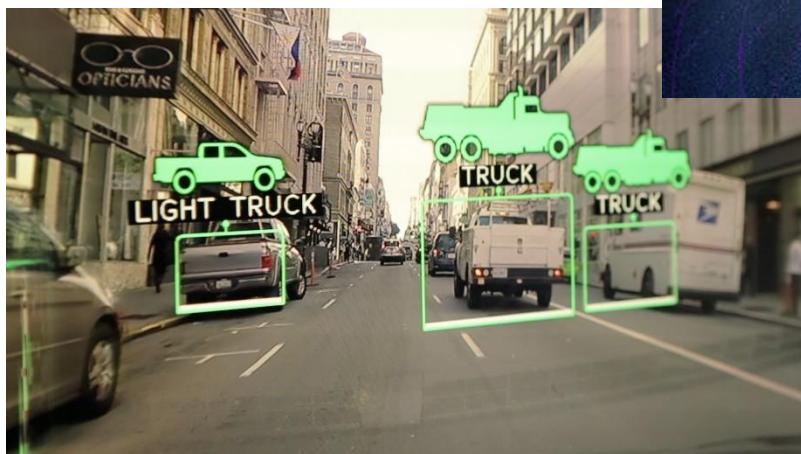
هادی کریم (1392/12/1) :

سلام..اصلا تو خریدن شک نکنید..من خریدم خیلی عالیه..خیلی بیشتر از عکسیش خوشگله..وافعا زیباست..اصلا لولایی دیده نمیشه استحکامش خیلی بیشتره..به قیمتیش هم ارزه..صدای فنیش هم اصلا اذیت نمیکنه..جون  
مده برای بازی

نموده دسته بندی با  
شبکه عصبی؟

# شبکه عصبی بازگشتی ...

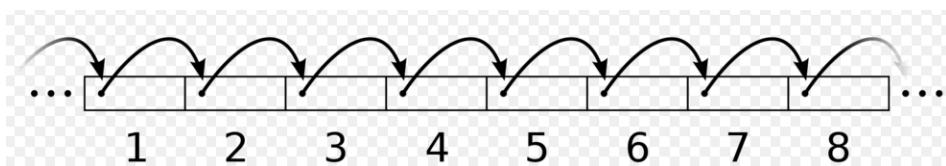
○ خودروهای خودران!



# شبکه عصبی بازگشتی ...

○ داده متوالی (Sequential Data): داده‌هایی که مقدار فعلی آنها به مقادیر قبلی وابسته است

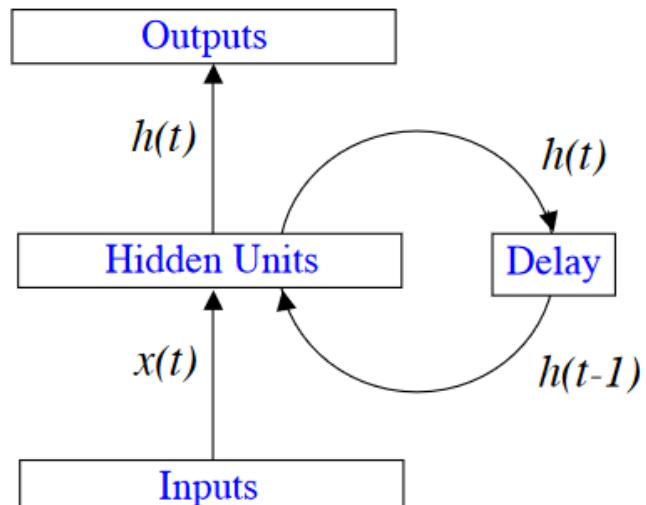
- فریم‌های (نمونه‌های) سیگنال گفتار
- فریم‌های (تصاویر) متوالی ویدئو
- وضعیت آب و هوا
- قیمت سهام یک شرکت/صنعت
- دنباله‌های تولید شده توسط گرامرها
- کلمات داخل یک متن
- ... ●



# شبکه عصبی بازگشتی . . .

## ◦ شبکه‌های عصبی بازگشتی (RNN: Recurrent Neural Networks)

- شبکه‌هایی که در ساختار آنها یال‌های بازگشت کننده وجود دارد
- بر خلاف شبکه‌های عصبی رو به جلو، یال‌ها می‌توانند تشکیل دور بدهند.
- به دلیل داشتن یال بازگشتی در ساختار خود، قدرت حافظه‌ای دارند.
- مناسب برای پردازش داده‌های متوالی (Sequential Data)



- ساختار: شبیه MLP با بازگشت از نرون‌های مخفی

# شبکه عصبی بازگشتی . . .

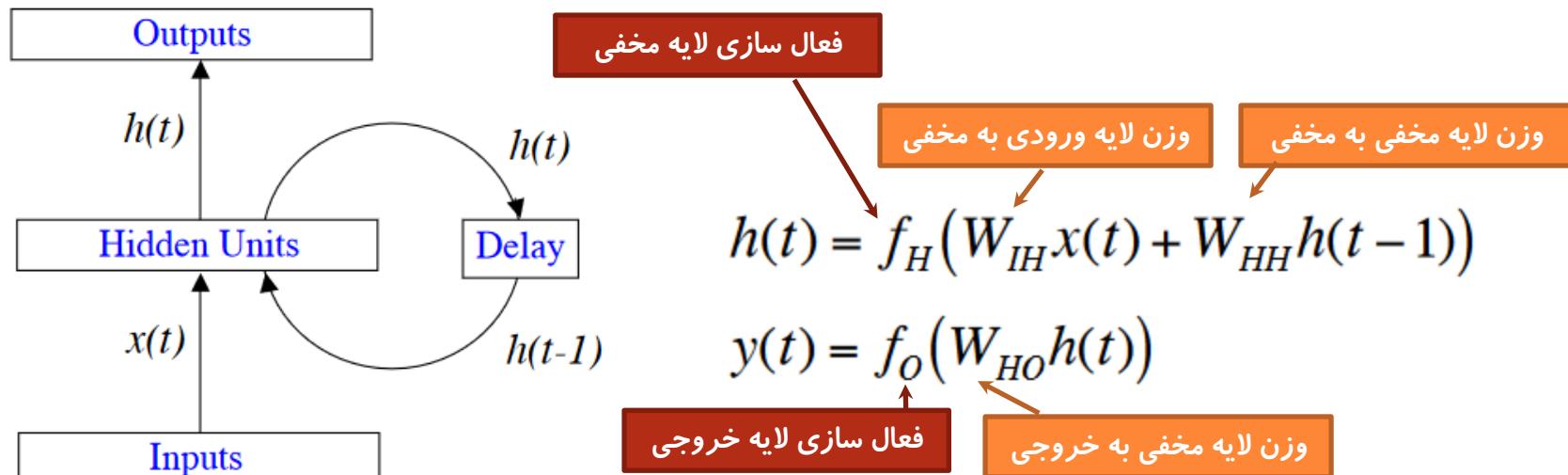
## ◦ شبکه عصبی بازگشتی به عنوان یک سیستم پویا (Dynamic System)

- تعریف کامل سیستم با حالت (State) سیستم: مجموعه‌ای مقادیر حاوی همه اطلاعات

◦ مقادیر فعال‌سازی‌های لایه مخفی:  $h(t)$

◦ مرتبه سیستم پویا = ابعاد فضای حالت

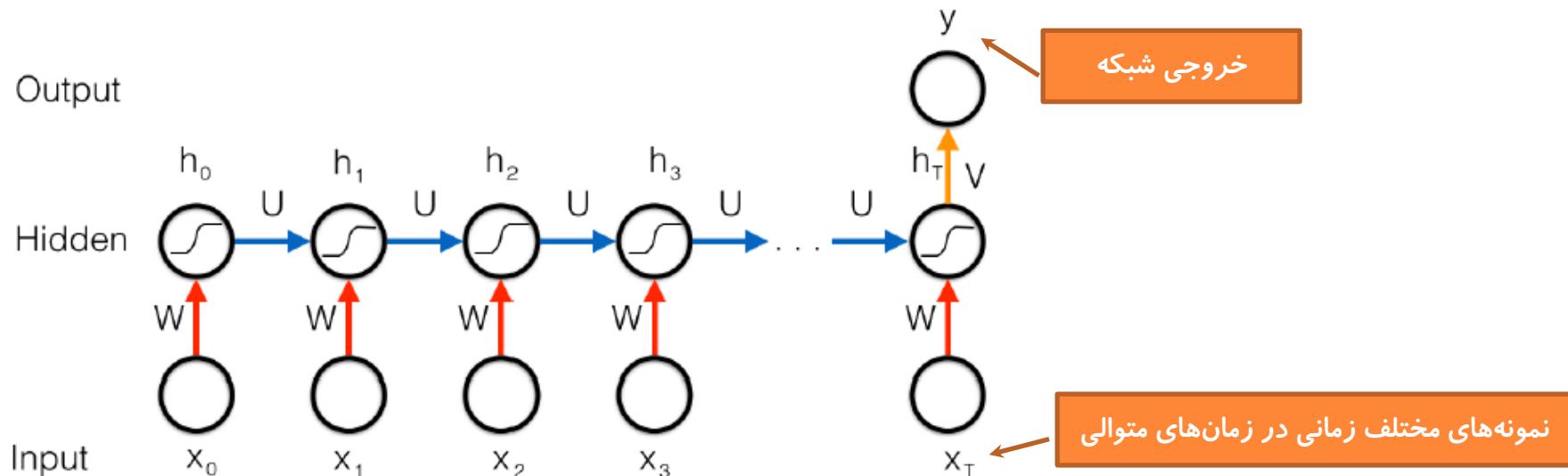
◦ در اینجا = تعداد نرون‌های لایه مخفی



# شبکه عصبی بازگشتی: آموزش ...

## ○ عملکرد شبکه

- وابستگی خروجی شبکه به خروجی‌های لایه مخفی به ازای همه ورودی‌ها



$$f(x) = Vh_T$$

$$h_t = \sigma(Uh_{t-1} + Wx_t), \text{ for } t = T, \dots, 1$$

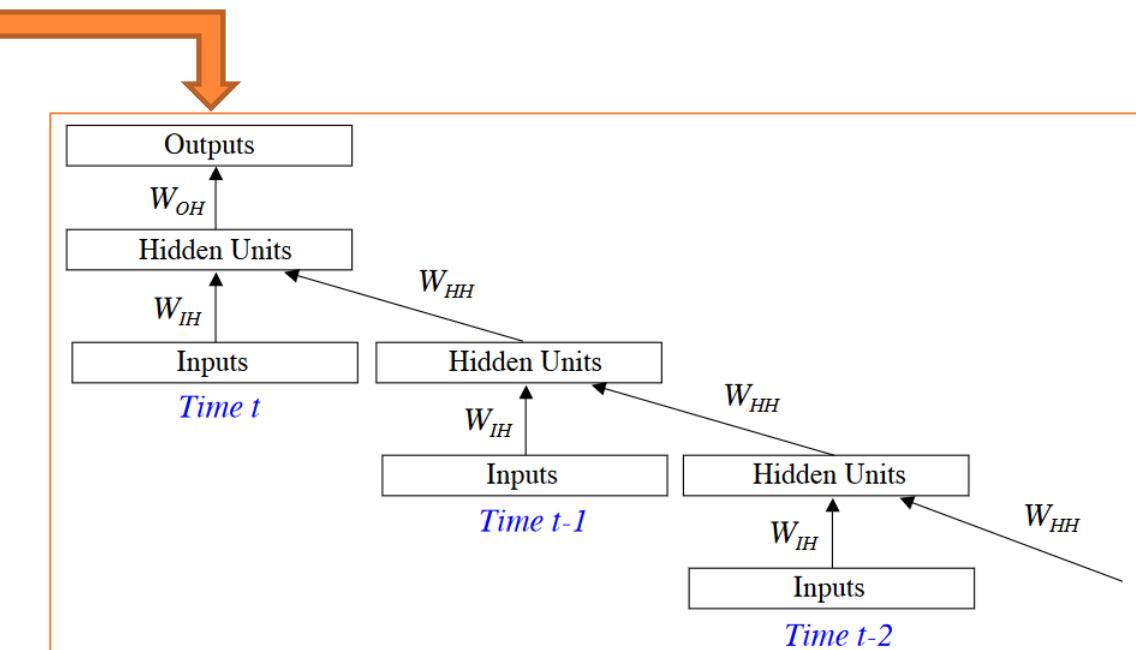
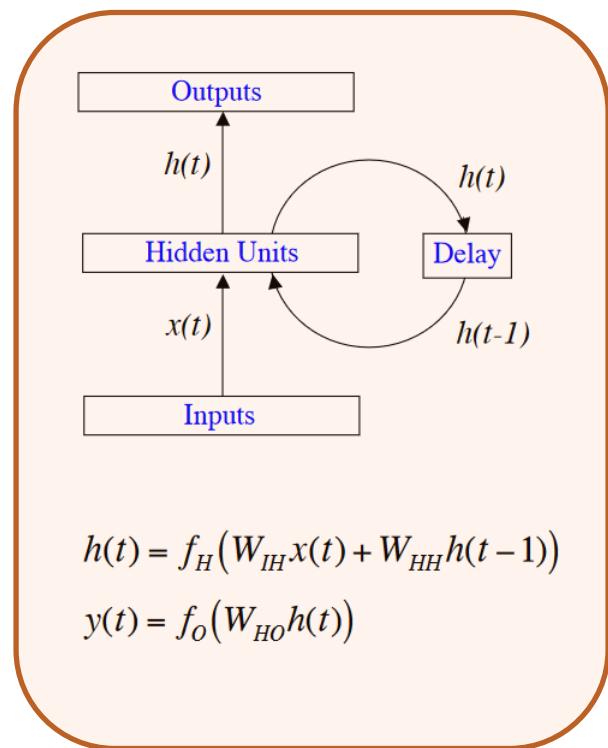
...

$$h_0 = \sigma(Wx_0)$$

# شبکه عصبی بازگشتی: آموزش ...

## ○ استفاده از الگوریتم پس‌انتشار . . .

- محاسبه خطای خروجی و استفاده از گرادیان برای تخمین وزن‌ها
- تبدیل شبکه بازگشتی به شبکه جلوسو
- باز کردن در زمان (Unfolding over Time)



# شبکه عصبی بازگشتی: آموزش ...

## ○ استفاده از الگوریتم پس انتشار ...

Backpropagation Through Time (BPTT) •

• محاسبه خطای شبکه برای همه نمونه‌های بین دو زمان شروع  $t_0$  و پایان  $t_1$

$$E_{total}(t_0, t_1) = \sum_{t=t_0}^{t_1} E_{sse/ce}(t)$$

• گرادیان وزن‌های شبکه برای بدست آوردن مقدار تغییرات وزن

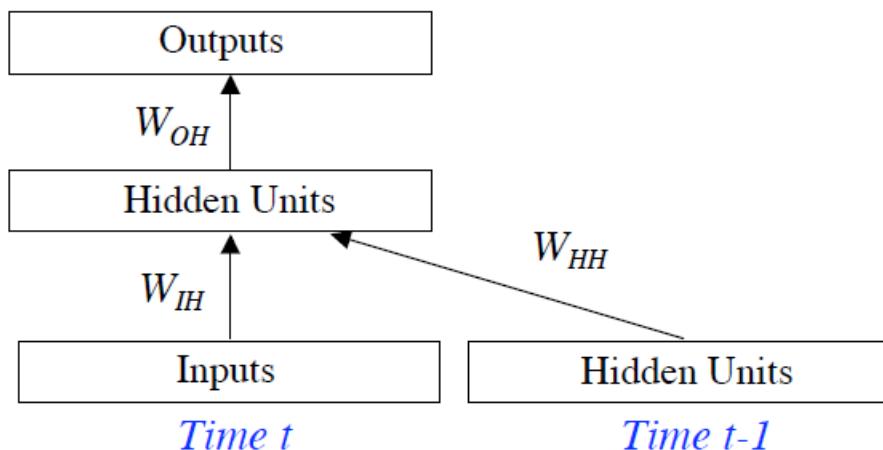
$$\Delta w_{ij} = -\eta \frac{\partial E_{total}(t_0, t_1)}{\partial w_{ij}} = -\eta \sum_{t=t_0}^{t_1} \frac{\partial E_{sse/ce}(t)}{\partial w_{ij}}$$

$$w_{ij} \in \{W_{IH}, W_{HH}\}$$

# شبکه عصبی بازگشتی: آموزش ...

## ○ استفاده از الگوریتم پس انتشار

- استفاده از این روش نیازمند نگهداری حالت‌های قبلی شبکه (خروجی لایه مخفی) و کلیه ورودی‌های قبلی
- در عمل نگهداری همه اطلاعات قبلی مشکل است و تنها از تعداد محدودی از آنها (مثلاً ۳۰ مقدار قبلی) استفاده می‌شود = truncation
- حالت ساده = نگهداری فقط یک مرحله قبل = شبکه المان (Elman Network)

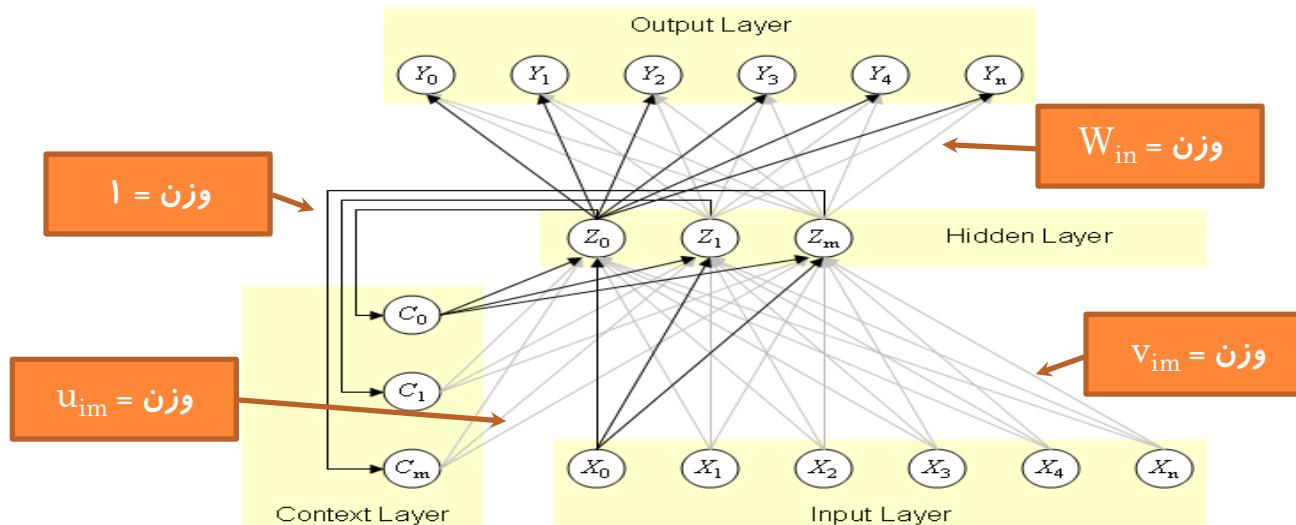


# شبکه عصبی الگان

## ○ ساختار

- دارای چهار لایه ورودی، مخفی، بافت و خروجی
- لایه بافت

- نرون‌های لایه بافت یک کپی از فعال‌سازهای نرون‌های لایه پنهان را دریافت می‌کنند
- اتصالات بازگشتی لایه بافت به لایه پنهان، یک حافظه کوتاه‌مدت را برای شبکه ایجاد می‌کند
- تعداد نرون‌های لایه بافت با تعداد نرون‌های لایه پنهان برابر است
- وزن یال‌هایی که لایه پنهان را به لایه بافت متصل می‌کنند برابر مقدار ثابت یک می‌باشد





# شبکه عصبی الگان: الگوریتم آموزش و کاربرد

## ○ مراجعه کنید به

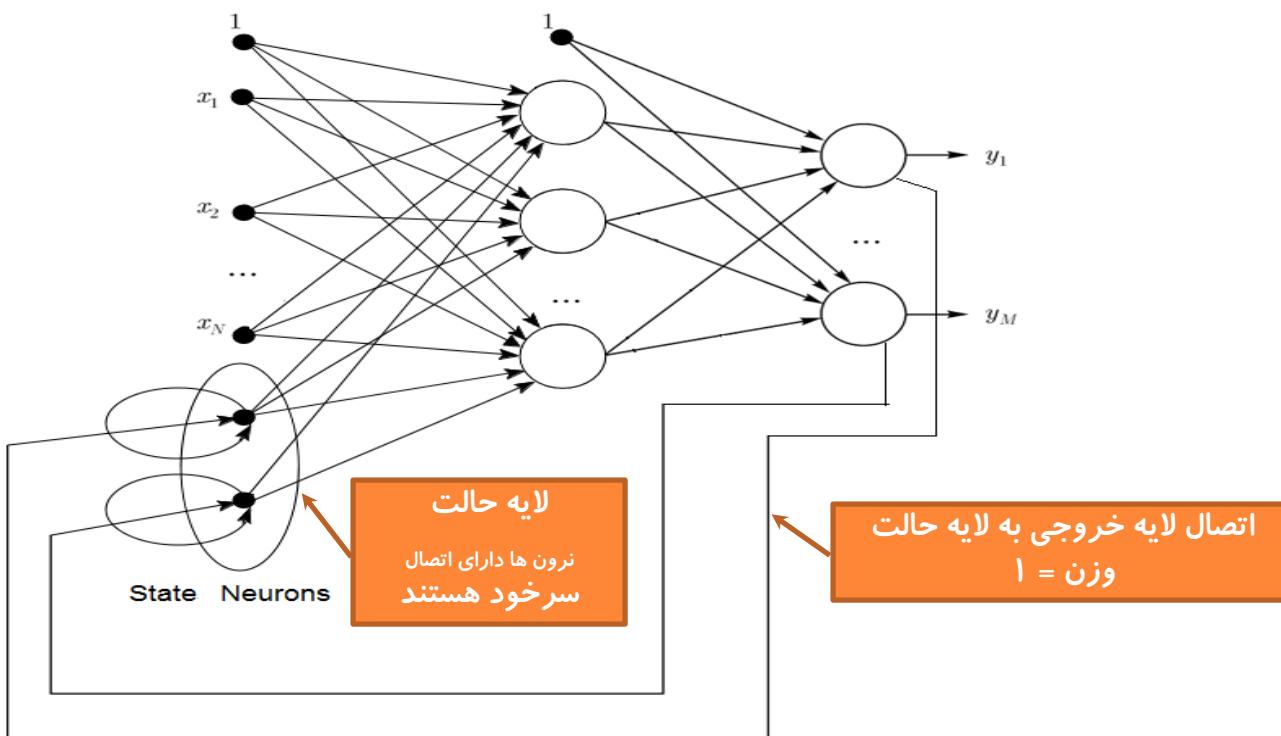
- <http://dsp.ut.ac.ir/en/wp-content/uploads/2018/05/ANN-Lecture5-RNN.pdf>



## شبکه عصبی جردن ...

### ● معرفی و ساختار

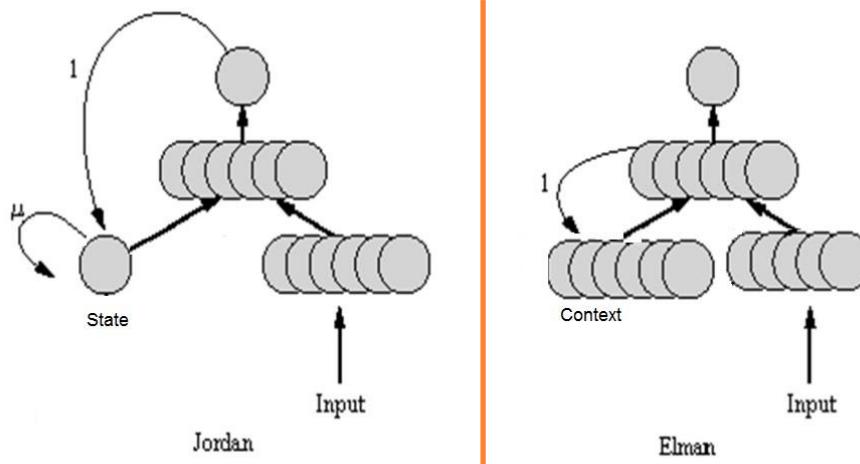
- ارائه شده در سال ۱۹۹۶ توسط مایکل جردن
- دارای شباهت بسیار زیاد به شبکه عصبی المان
- شبکه دارای اتصالات بازگشتی از لایه خروجی به لایه حالت و همچنین از لایه حالت به خودش می‌باشد



# شبکه عصبی جردن: تفاوت با شبکه المان

## ○ در شبکه جردن

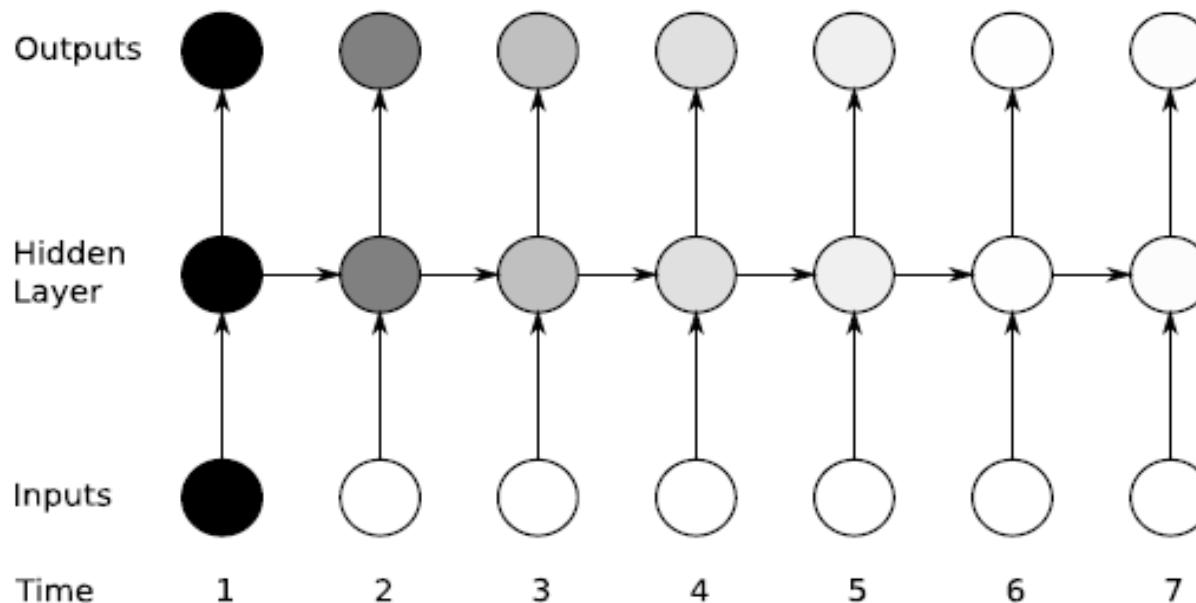
- اتصالات بازگشتی به جای لایه پنهان از لایه خروجی شروع می‌شود (با وزن ثابت یک)
- در این شبکه لایه بافت، لایه حالت (State Layer) نامیده می‌شود
- لایه حالت شامل اتصالات بازگشتی از خودش به خودش با وزن ثابت می‌باشد
- تعداد نرون‌های لایه حالت با تعداد نرون‌های لایه خروجی برابر است
- نرون‌های لایه حالت یک کپی از فعال‌سازهای نرون‌های لایه خروجی را دریافت می‌کنند.



# مشکل فراموش در شبکه‌های عصبی بازگشتی . . .

## ○ مشکل

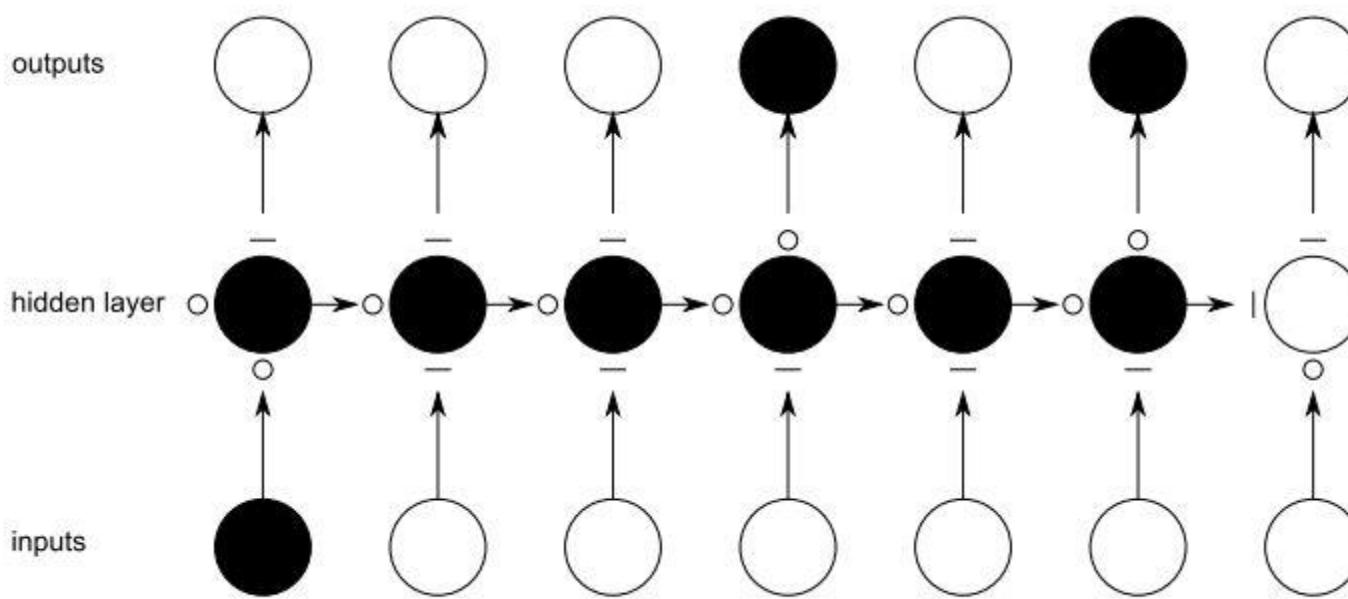
- با طولانی شدن دنباله ورودی، شبکه عصبی بازگشتی به مرور داده‌های اولیه را فراموش می‌کند که به آن مشکل فراموشی گفته می‌شود
- سایه‌های پررنگ‌تر به معنای تاثیر بیشتر بر لایه پنهان و خروجی می‌باشد



# مشکل فراموش در شبکه‌های عصبی بازگشتی

## ○ حل مشکل فراموش

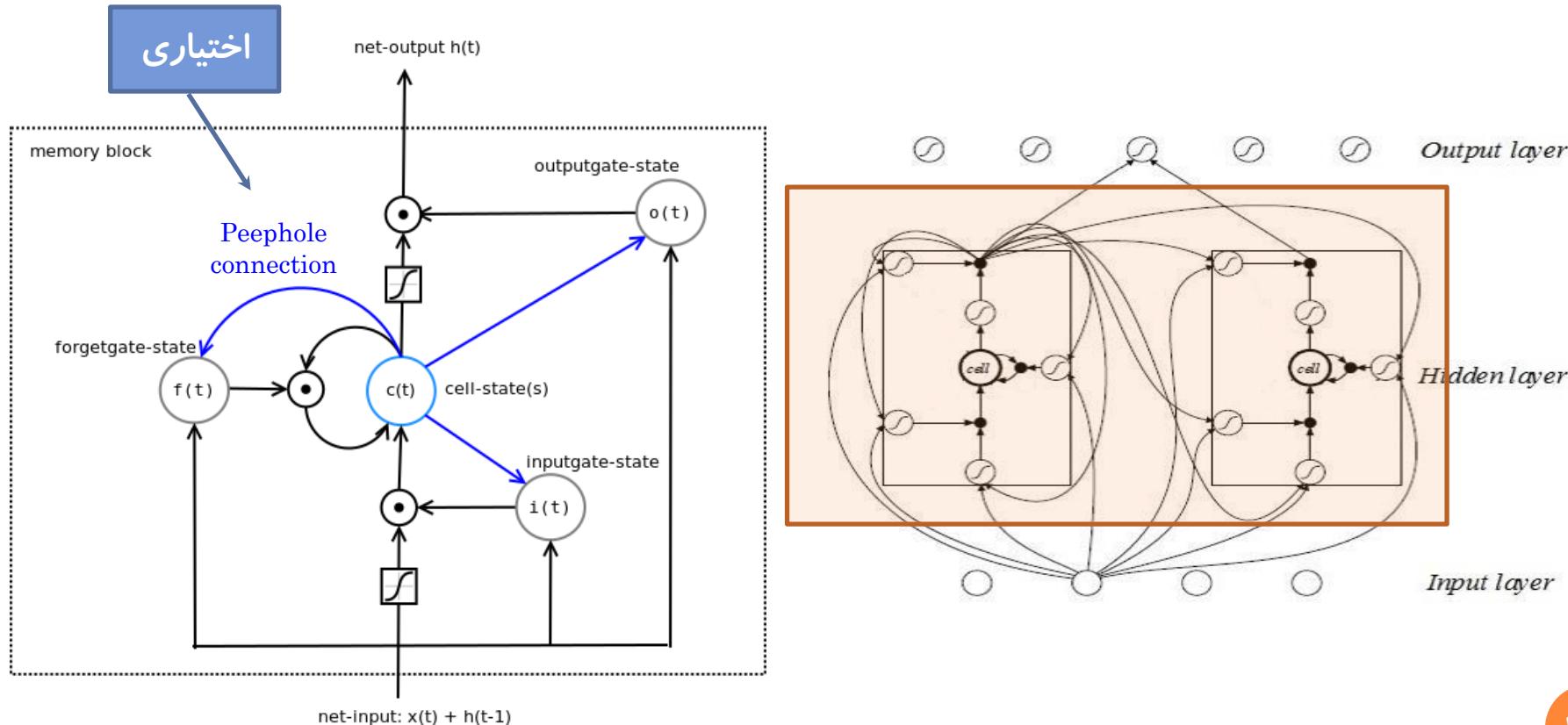
- کنترل ورود و خروج داده در واحدهای لایه میانی شبکه



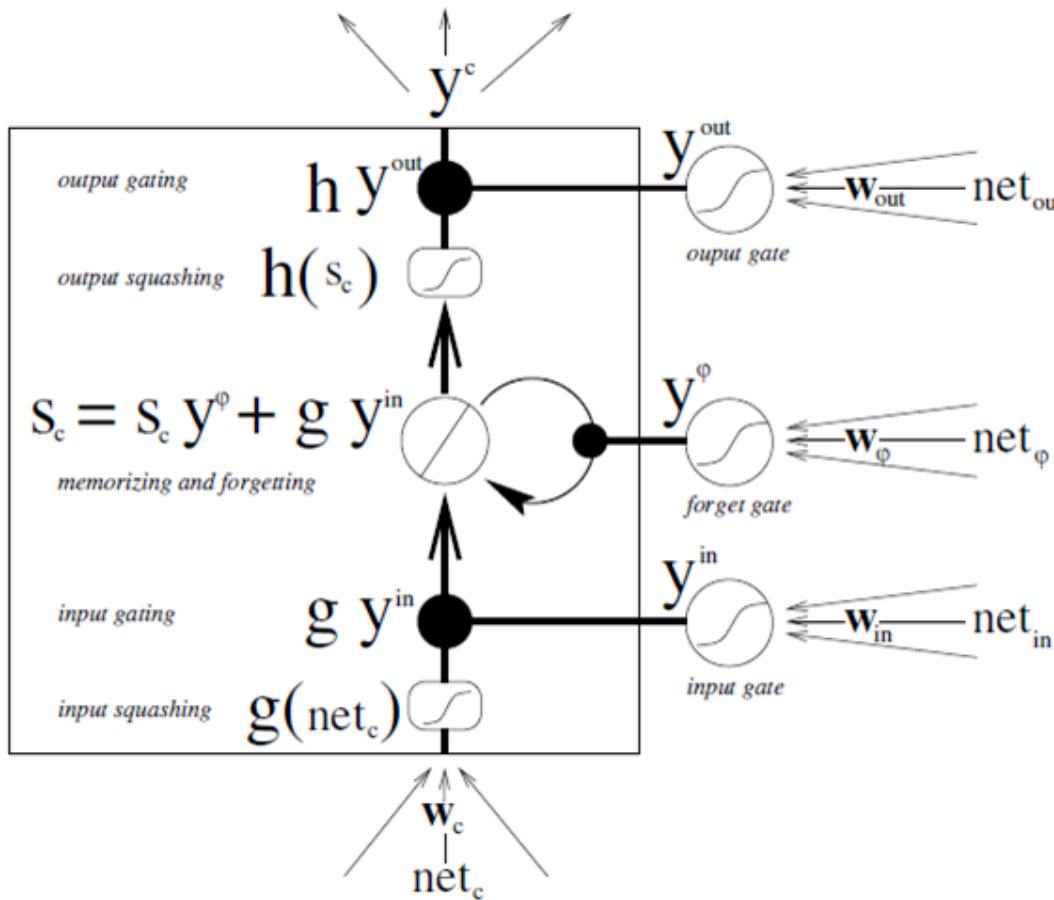
# شبکه عصبی حافظه کوتاه مدت ماندگار (LSTM) ...

## Long Short Term Memory (LSTM)

- نرون‌های لایه پنهان با بلوک‌های حافظه جایگزین شدند
- حل شدن مشکل فراموشی دنباله‌های طولانی



# شبکه عصبی LSTM: ساختار بلوک حافظه ...



○ هر بلوک حافظه، شامل

- سلول

- برای ذخیره اطلاعات در بلوک

- دروازه ورودی

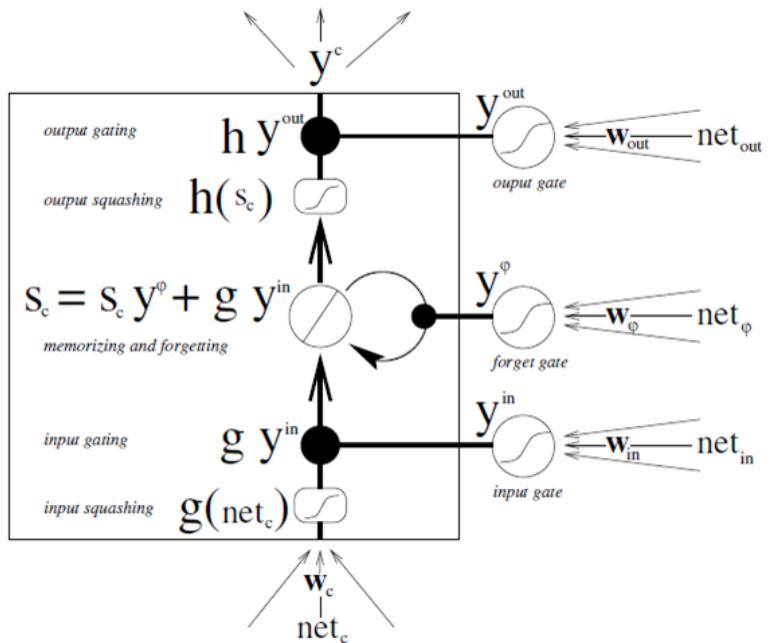
- دروازه فراموشی

- دروازه خروجی

◦ دروازه‌ها وظیفه کنترل ورود و خروج داده‌ها و پاک کردن حافظه بلوک را برعهده دارند

# شبکه عصبی LSTM: ساختار بلوک حافظه ...

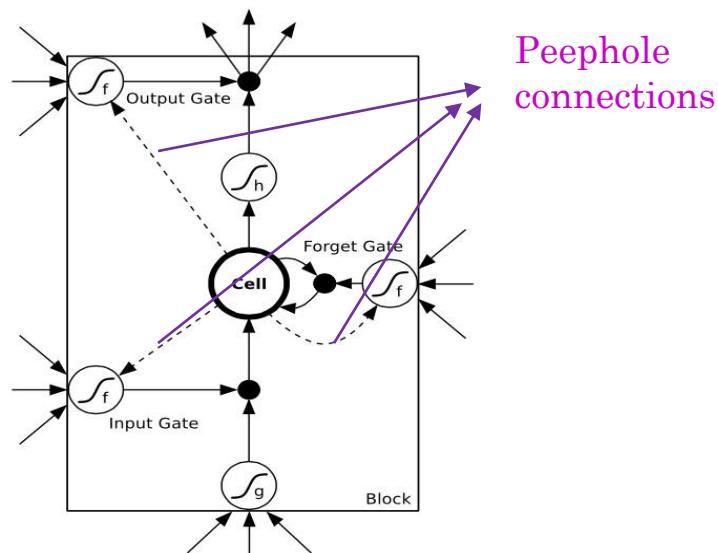
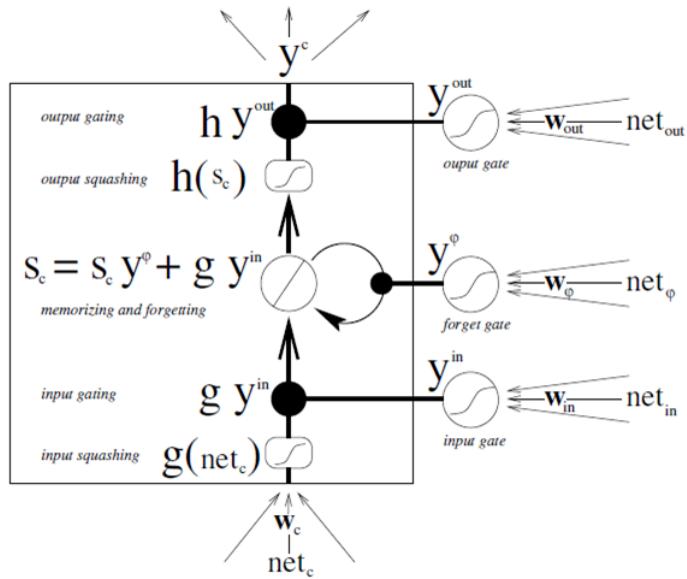
- وظیفه دروازه‌ها: کنترل ورود و خروج داده‌ها و پاک کردن حافظه‌ی بلوک
  - فعال‌ساز دروازه‌ها مقداری بین صفر و یک می‌گیرند.
  - در صورتی که دروازه کاملا باز باشد فعال‌ساز آن برابر یک و در صورتی که کاملا بسته باشد فعال‌ساز آن برابر صفر است
- هر سلول حافظه در مرکز خود یک واحد به نام CEC دارد که به فعال‌سازی آن **حالت سلول**,  $S_c$  می‌گویند



# شبکه عصبی LSTM: ساختار بلوک حافظه

## اتصالات روزنه (peephole)

- این اتصالات دلخواه هستند (optional) و حالت سلول،  $s_c$ ، را به دروازه‌ها متصل می‌سازند



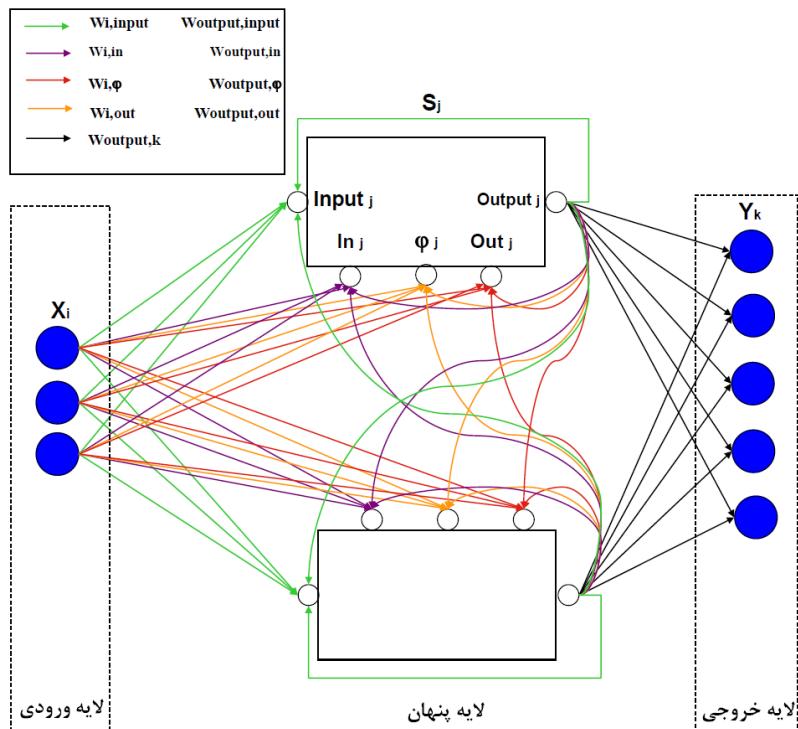
# شبکه عصبی LSTM: اتصالات (وزن ها)

## ○ از ورودی

- وزن بین لایه ورودی و بلوک حافظه
- وزن بین لایه ورودی و دروازه ورودی
- وزن بین لایه ورودی و دروازه فراموشی
- وزن بین لایه ورودی و دروازه خروجی

## ○ از خروجی بلوک (سلول حافظه)

- وزن بین خروجی بلوک‌ها و ورودی بلوک‌ها
- وزن بین خروجی بلوک‌ها و دروازه ورودی
- وزن بین خروجی بلوک‌ها و دروازه فراموشی
- وزن بین خروجی بلوک‌ها و دروازه خروجی
- وزن بین خروجی‌های بلوک‌ها و لایه خروجی





# شبکه عصبی LSTM: الگوریتم آموزش و کاربرد

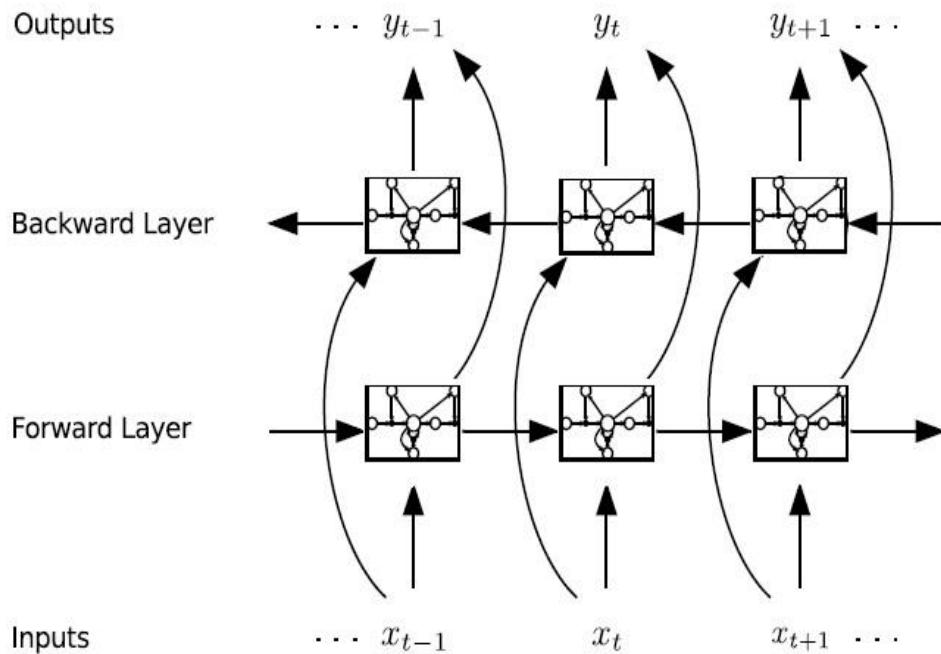
## ○ مراجعه کنید به

- <http://dsp.ut.ac.ir/en/wp-content/uploads/2018/05/ANN-Lecture5-RNN.pdf>

# شبکه‌های عصبی (Bidirectional LSTM: دوطرفه)

## ○ ساختار

- شامل دو لایه پنهان بازگشتی مجزا (هر لایه پنهان شامل بلوک‌های LSTM می‌باشد).
- بین این دو لایه پنهان هیچ اتصالی وجود ندارد.
- هر دو لایه پنهان به یک لایه خروجی متصل شده‌اند.



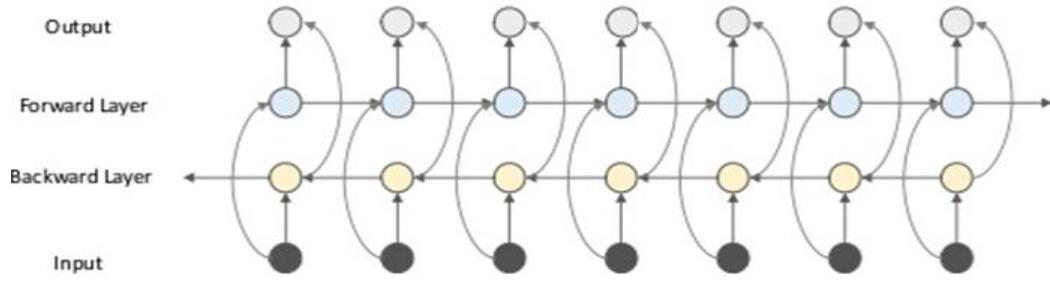
# شبکه‌های عصبی (Bidirectional LSTM) دوطرفه

## ○ ایده اصلی

هر دنباله ورودی در دو جهت زمانی رو به جلو و از انتهای پنهان بازگشته مجزا داده شود

- فرض کنید دنباله آموزشی به صورت  $X^T = (x_1, x_2, \dots, x_{T-1}, x_T)$  و دنباله هدف متناظر برابر  $t^T = (t_1, t_2, \dots, t_{T-1}, t_T)$  باشد
- در هر مرحله بردار  $x_i$  را به لایه Backward و  $x_{T-(i-1)}$  به لایه Forward ارسال کرده و مقدار هدف را بردار  $t_i$  قرار می‌دهیم.
- آموزش شبکه را با استفاده از الگوریتم مربوط به شبکه LSTM دنبال می‌کنیم

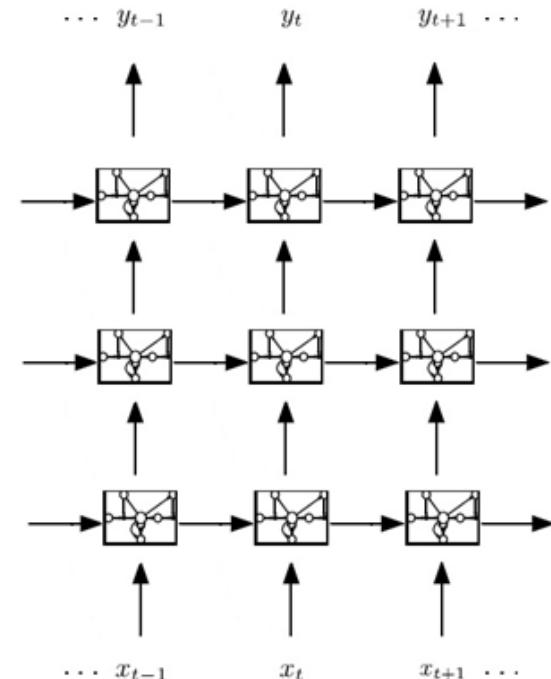
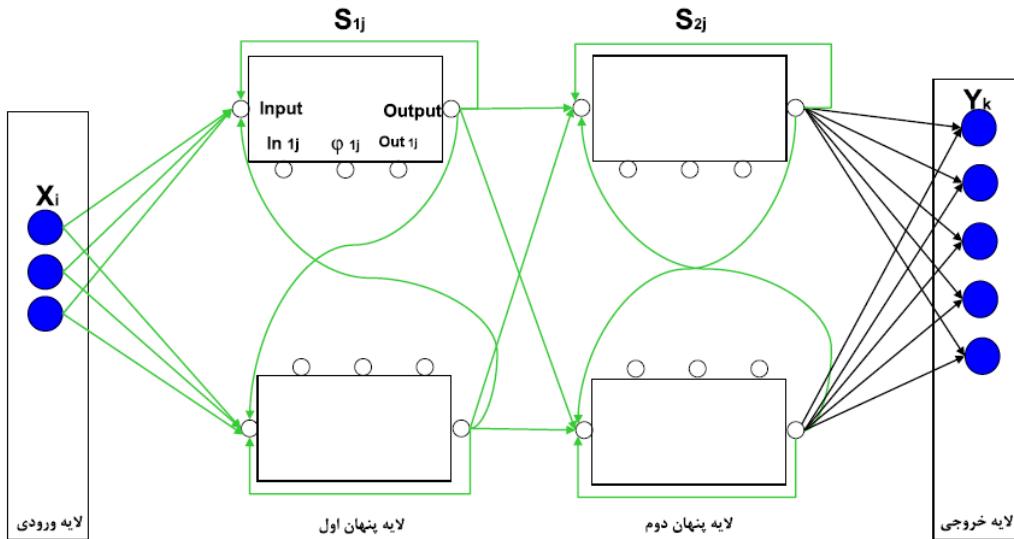
مقدار خالص رسیده به لایه خروجی جمع وزن دار مقدار خالص دو لایه Backward و Forward است



# شبکه‌های عصبی LSTM: عمیق

◦ شبکه عصبی با بیش از یک لایه مخفی

- خروجی هر لایه پنهان ورودی لایه پنهان بالاتر
- تقریب زننده جهانی
- قابلیت یادگیری بیشتر

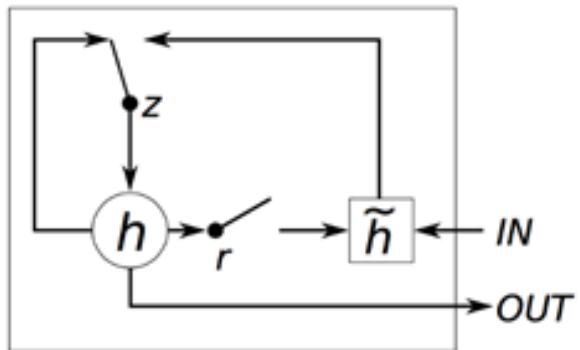


# شبکه عصبی واحد بازگشتی در واژه‌ای (GRU)

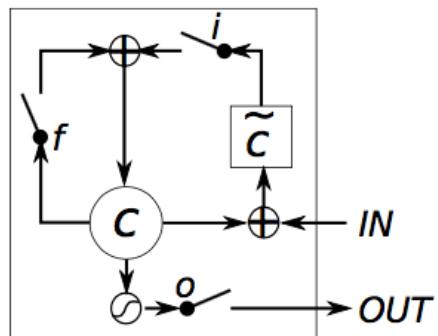
## ○ ساده شده LSTM استاندارد

- عدم وجود دروازه خروجی

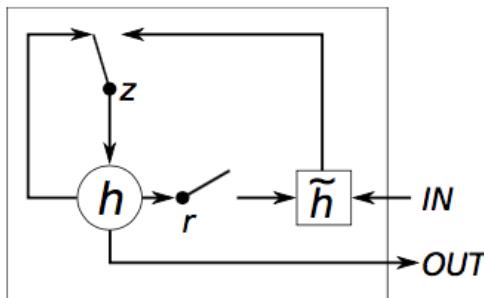
- دارای دو دروازه (LSTM دارای ۳ دروازه)
  - راه اندازی مجدد (reset) و بروزرسانی (update)
- عبور تمام مقدار سلول به خروجی یا ورودی سایر بلوك‌ها



• عبور تمام مقدار سلول به خروجی یا ورودی سایر بلوك‌ها



(a) Long Short-Term Memory



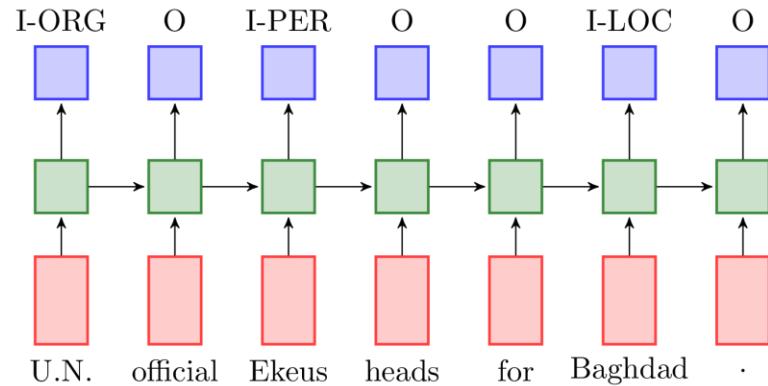
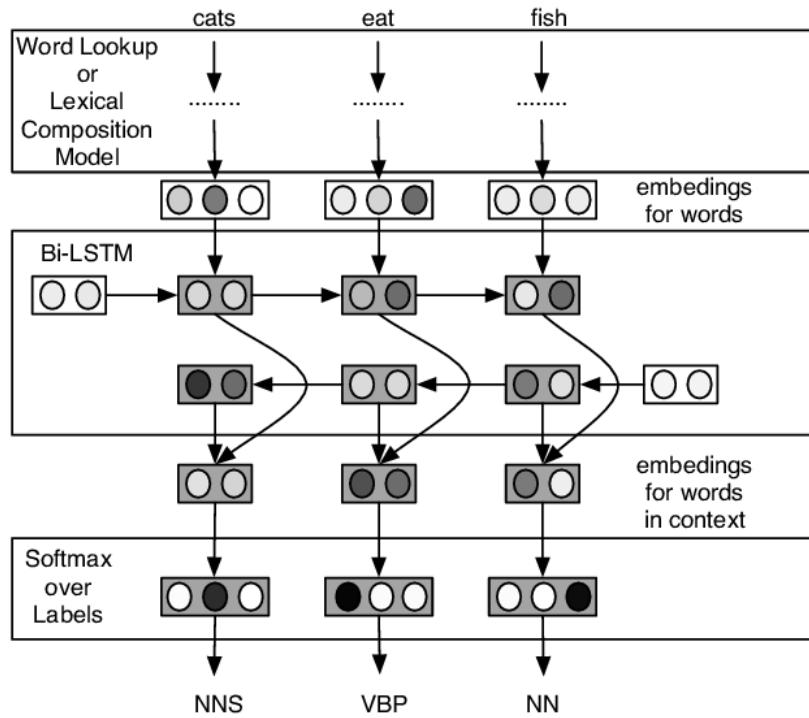
(b) Gated Recurrent Unit

Figure 1: Illustration of (a) LSTM and (b) gated recurrent units. (a)  $i$ ,  $f$  and  $o$  are the input, forget and output gates, respectively.  $c$  and  $\tilde{c}$  denote the memory cell and the new memory cell content. (b)  $r$  and  $z$  are the reset and update gates, and  $h$  and  $\tilde{h}$  are the activation and the candidate activation.

# شبکه عصبی LSTM: کاپردها...

برچسب زنی اجزای کلام (POS)/بازشناسی پدیده‌های اسمی (NER)

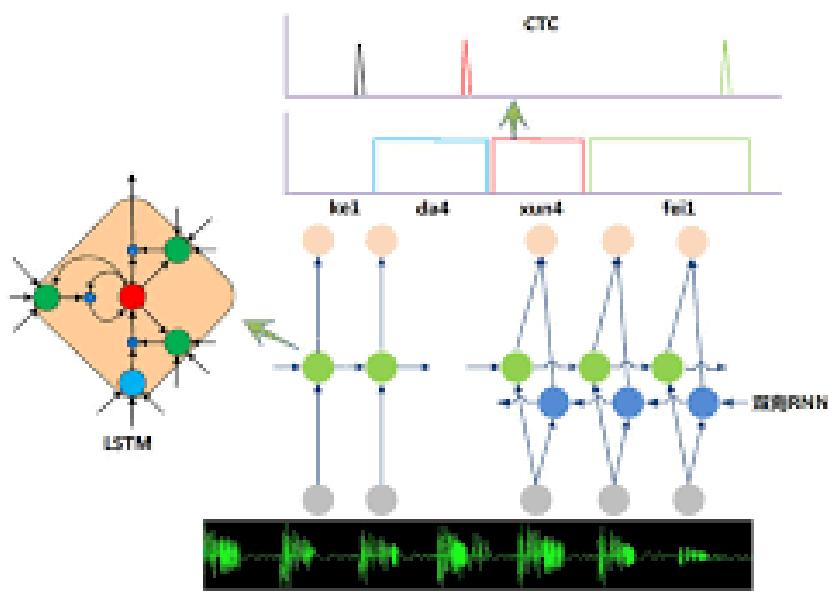
- ورودی: بردار یک کلمه (مثل Word Vector)
- خروجی: به تعداد برچسب‌ها (هر نرون یک برچسب)



# شبکه عصبی LSTM: کاپردها...

## ○ بازشناسی گفتار

- ورودی: بردار مربوط به یک فریم گفتار
- خروجی: به تعداد واحدهای بازشناسی شونده (هر نرون یک واژ)
- نیاز به روشی برای تبدیل دنباله برچسب فریم‌ها به دنباله واژ = CTC



# شبکه عصبی LSTM: کاپردها...

## ● بازشناسی دست خط / نویسه‌های نوری (OCR)

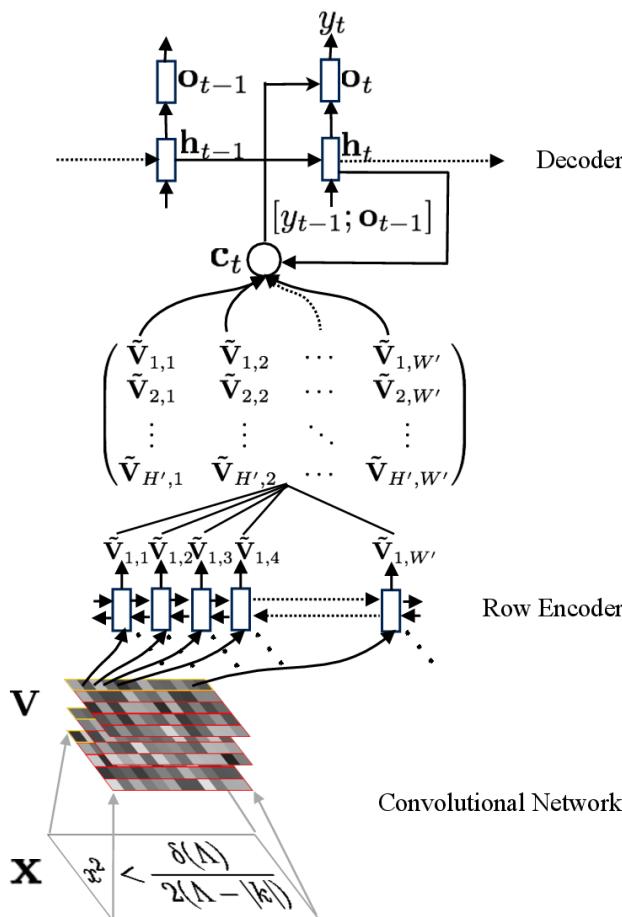
- ورودی: بردار مربوط به ویژگی یک فریم از تصویر

○ ویژگی های CNN

- خروجی: به تعداد واحدهای بازشناسی شونده

○ هر نرون یک کاراکتر

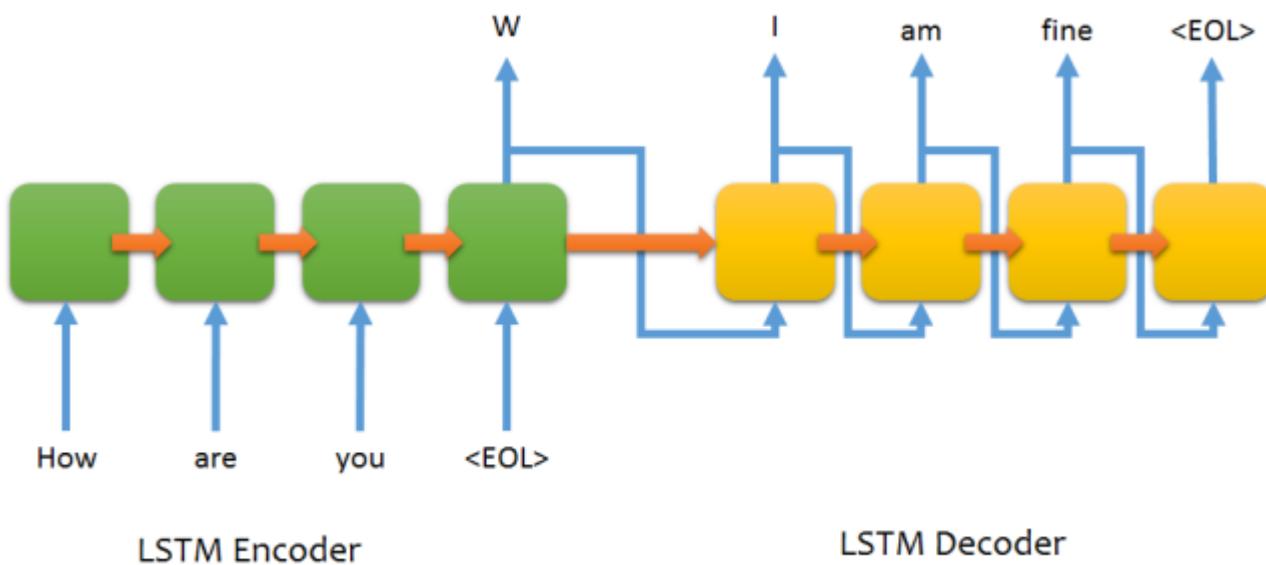
○ نیاز به تبدیل دنباله برچسب فریم ها به دنباله واژ = CTC



# شبکه عصبی LSTM: کاپردها...

## ○ ترجمه ماشینی

- ورودی: بردار یک کلمه (مثل Word Vector) در زبان مبدا
- خروجی: احتمال کلمات در زبان مقصد





# شبکه عصبی LSTM: کاربردها

روشی غالب در همه (!) کاربردهای مدل‌سازی دنباله

# شبکه عصبی LSTM: تشخیص واژه‌های فارسی ...

## داده‌ها: فارس‌داد

- شامل ۶۰۸۰ سیگنال صوتی
- داده آموزش: ۹۵٪ کل داده (معادل ۵۶۹۸ سیگنال)
- داده آزمون: ۵٪ کل داده (معادل ۳۸۲ سیگنال)

## استخراج ویژگی

- روش مورد استفاده: MFCC
- تعداد ضرایب هر فریم: ۳۹
- طول فریم: ۱۶ میلی ثانیه

## شبکه مورد استفاده: LSTM

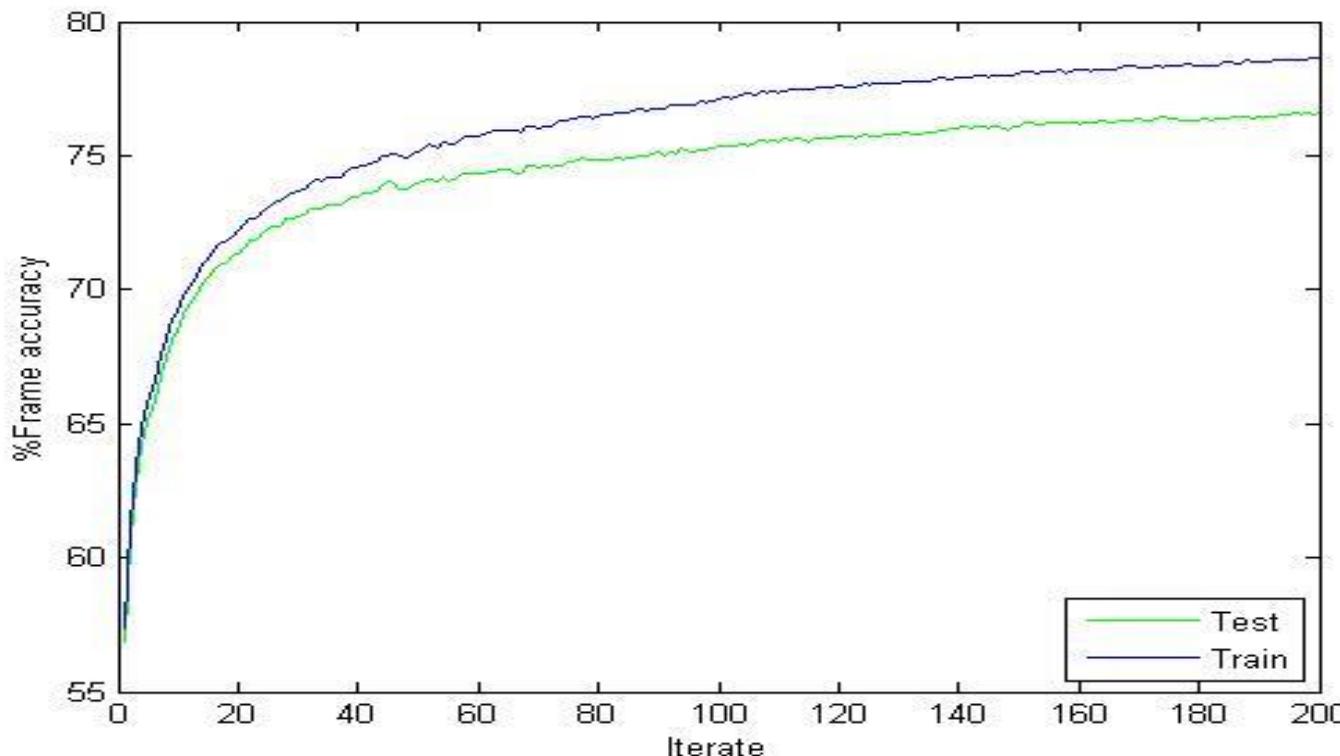
- ساختار شبکه
- نرون‌های لایه ورودی: ۳۹
- نرون‌های لایه خروجی: ۳۰ (تعداد واژه‌های فارسی + سکوت)

- وزن‌های اولیه: تصادفی در بازه  $[-1, 1]$
- نرخ یادگیری: ۰.۰۰۰۳
- تعداد بلوك حافظه: ۱۵۰

# شبکه عصبی LSTM: تشخیص واژه‌های فارسی ...

○ دقت به دست آمده روی فریم‌ها بعد از ۲۰۰ تکرار

- داده‌های آموزش:٪.۷۸/۶۲
- داده‌های آزمون:٪.۷۶/۵۹



# شبکه عصبی LSTM دو طرفه: تشخیص و اعماق فارسی . . .

## ○ داده‌ها: فارس‌دادات

- شامل ۶۰۸۰ سیگنال صوتی
- داده آموزش: ۸۰٪ کل داده (معادل ۴۸۶۴ سیگنال)
- داده آزمون: ۲۰٪ کل داده (معادل ۱۲۱۶ سیگنال)

## ○ استخراج ویژگی

- روش مورد استفاده: MFCC
- تعداد ضرایب هر فریم: ۳۹
- طول فریم: ۱۶ میلی ثانیه

## ○ شبکه مورد استفاده: Bidirectional LSTM

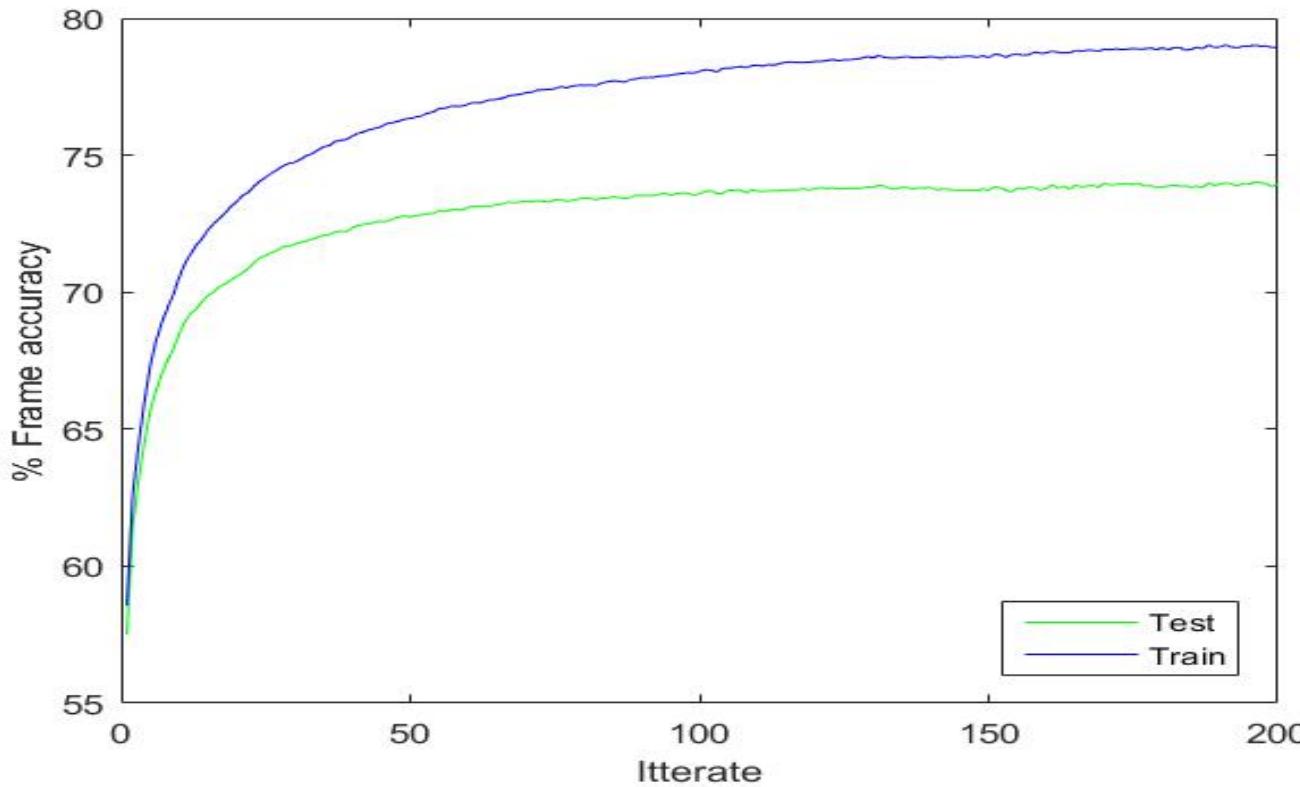
### ○ ساختار شبکه

- نرون‌های لایه ورودی: ۳۹
- نرون‌های لایه خروجی: ۳۰
- وزن‌های اولیه: تصادفی در بازه  $\left[\frac{-1}{10}, \frac{1}{10}\right]$
- نرخ یادگیری: ۰.۰۰۰۳
- تعداد بلوک حافظه: ۱۲۰

# شبکه عصبی LSTM دو طرفه: تشخیص و اعماق فارسی . . .

○ دقیقت به دست آمده روی فریم‌ها بعد از ۲۰۰ تکرار

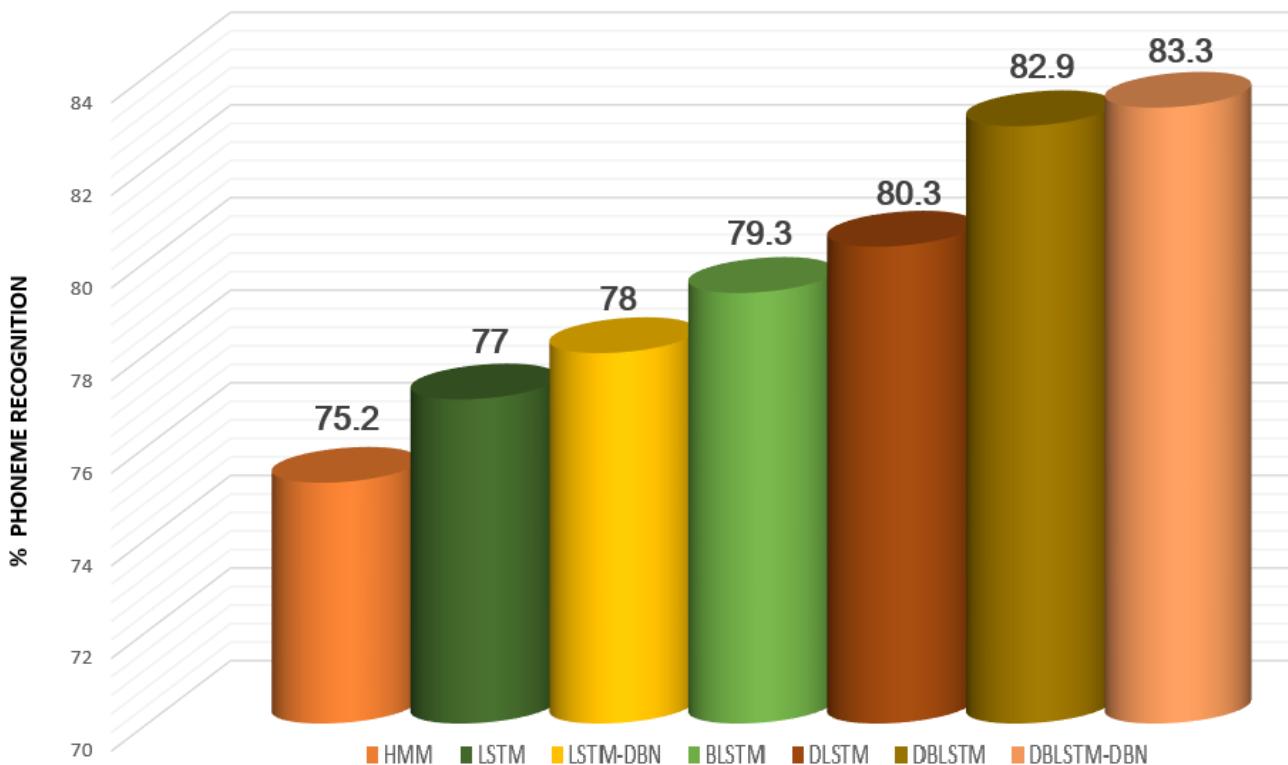
- داده‌های آموزش: ۷۹.۰٪
- داده‌های آزمون: ۷۳.۸٪



# شبکه عصبی LSTM دو طرفه: تشخیص واژه‌های فارسی

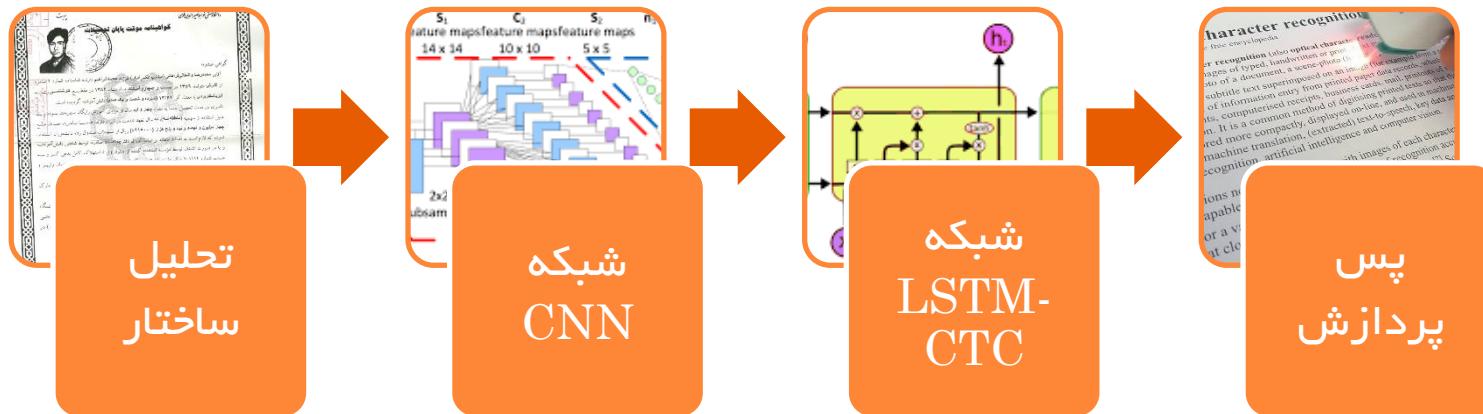
## ○ دقت روی واژ

- کارایی بالاتر شبکه‌های دو طرفه نسبت به شبکه‌های یک طرفه
- کارایی بالاتر شبکه‌های عمیق نسبت به شبکه‌های غیر عمیق



# یادگیری عمیق؛ نویسه خوان نوری فارسی ...

◦ تبدیل عکس (حاوی نوشته) به متن الکترونیکی



- استخراج ویژگی با CNN

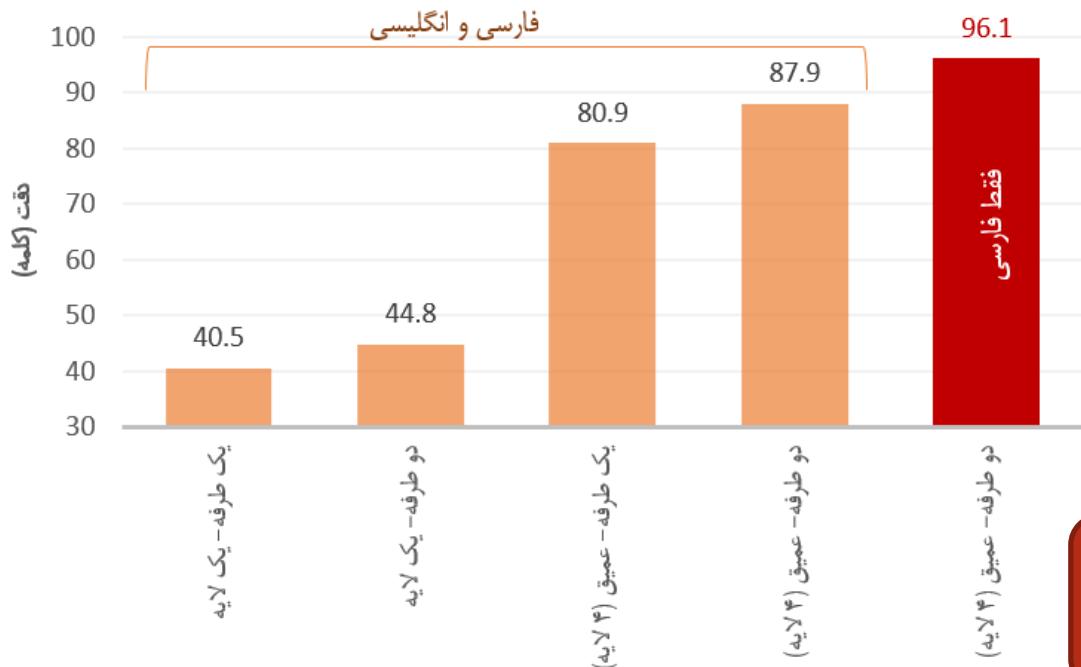
- یک لایه

- بازناسی با LSTM

# یادگیری عمیق؛ نویسه خوان نوری فارسی ...

## ○ دقت تشخیص کلمه

- ۱۱ فونت رایج فارسی و سه اندازه ۱۲، ۱۴ و ۱۸
- دادگان آموزش: ۲,۰۰۰,۰۰۰ تصویر
- دادگان آزمون
- فقط فارسی: ۶۰۰۰۰۰ تصویر
- فارسی-انگلیسی: ۶۰۰۰۰۰ تصویر



Try it !  
KhanaSoft.com