

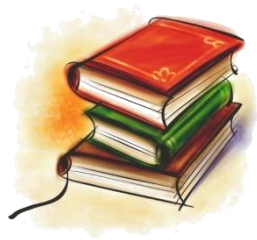
روش‌های آماری در پردازش زبان طبیعی

شبکه عصبی مصنوعی

هادی ویسی

h.veisi@ut.ac.ir

دانشگاه تهران - دانشکده علوم و فنون نوین



- شبکه عصبی مصنوعی
 - مقدمه و معرفی + تاریخچه
- شبکه عصبی پرسپترون
 - آموزش + مثال
- شبکه عصبی آدالین
 - 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)

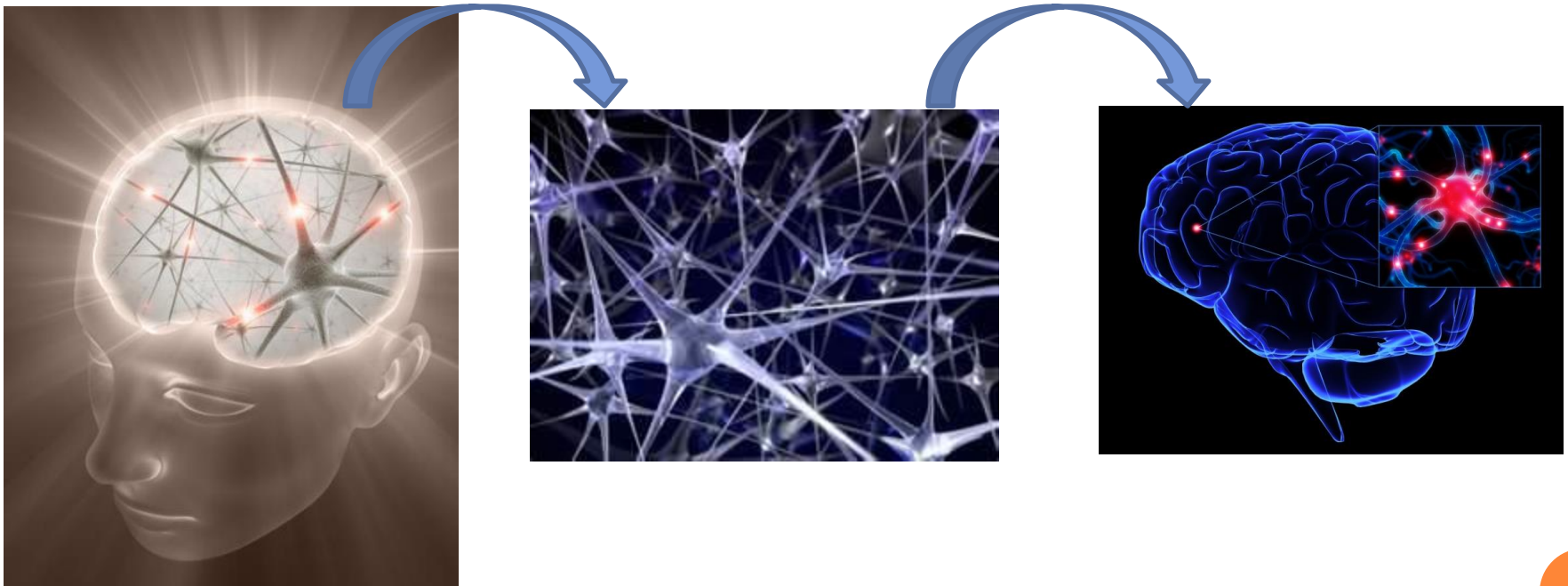
شبکه عصبی؟

○ 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



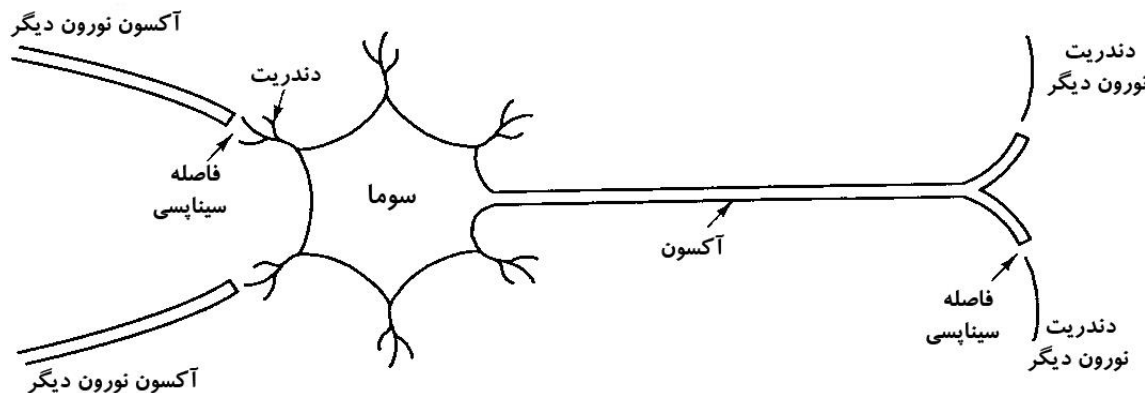
شبکه عصبی طبیعی ...

○ عنصر پردازشگر تشکیل‌دهنده یک شبکه عصبی مصنوعی

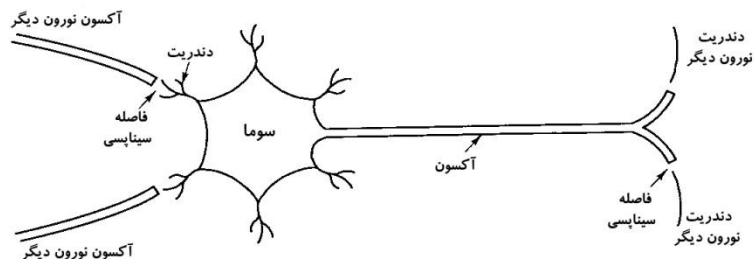
• نرون (Neuron) = عصب طبیعی (سلول مغزی)

○ سه جزء تشکیل‌دهنده یک نرون طبیعی

- دندریت‌ها (Dendrite): دریافت سیگنال از سایر نرون‌ها
- سوما (Soma) = بدنه سلول: سیگنال‌های 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 (تضعیف/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]

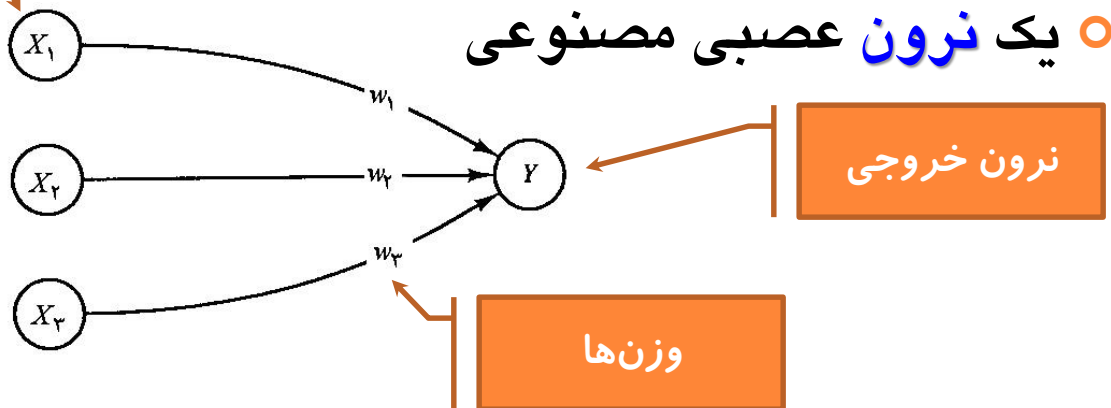
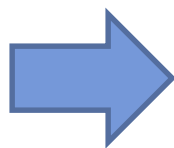
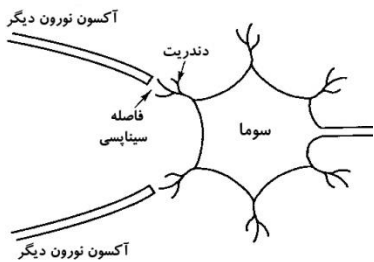
- یک سیستم پردازش اطلاعات با ویژگی‌های مشترکی با شبکه‌های عصبی طبیعی
- تعمیم یافته مدل‌های ریاضی تشخیص انسان بر اساس زیست‌شناسی عصبی

○ 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



- فعال‌سازی‌ها یا سیگنال‌های خروجی نرون‌های ورودی به ترتیب x_1 ، x_2 و x_3 هستند
- ورودی شبکه به نرون Y ، حاصل جمع وزن دار 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

○ تابع پله

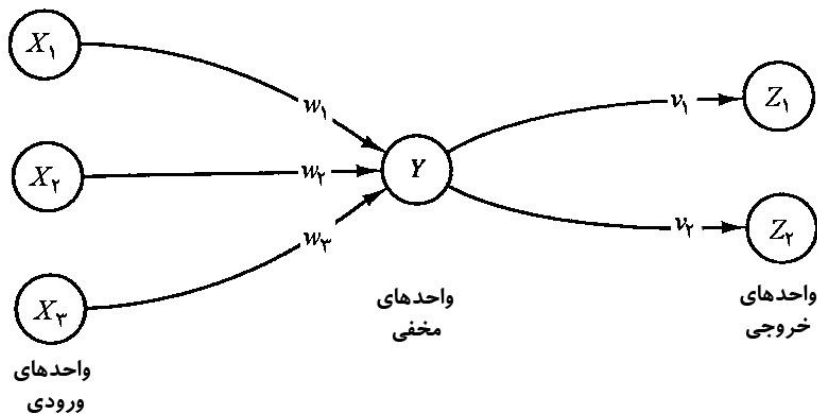
○ تابع سیگموئید (Sigmoid)

$$y = f(y_{in})$$

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

شبکه عصبی مصنوعی ...

○ یک شبکه عصبی مصنوعی



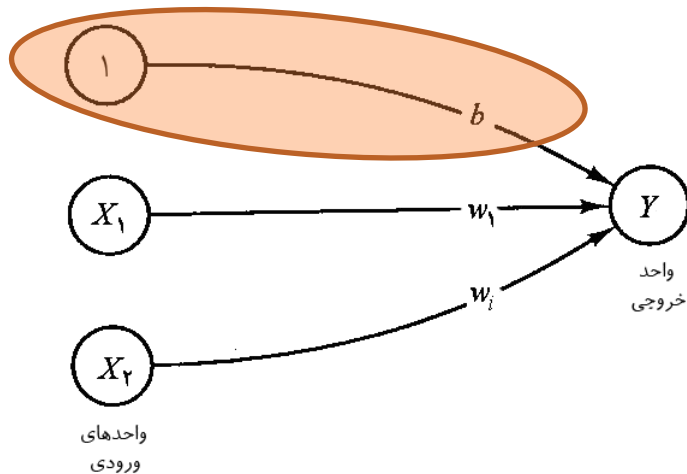
- سه لایه: ورودی، مخفی و خروجی
- دو دسته وزن: w ها و v ها

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

شبکه عصبی مصنوعی ...

○ بایاس

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



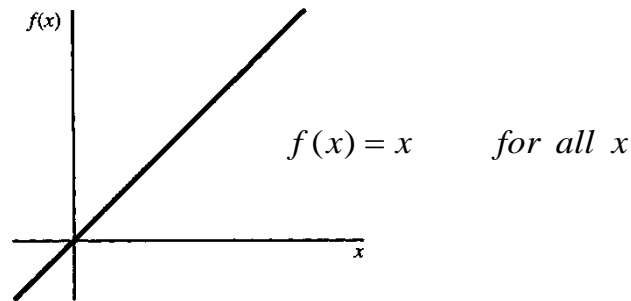
$$y_{in} = 1 \times b + w_1 x_1 + w_2 x_2 = b + \sum_i w_i x_i$$

شبکه عصبی مصنوعی ...

○ توابع فعال‌سازی متداول ...

- تابع همانی (Identity Function)

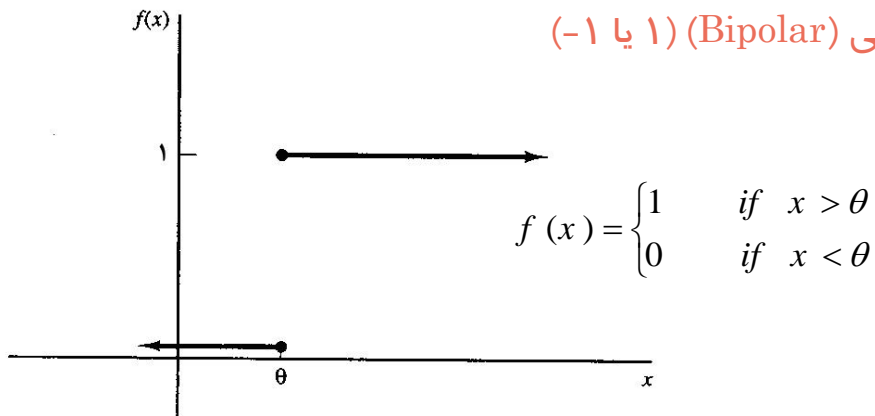
- برای واحدهای ورودی



- تابع پله‌ای دودویی (Step Function)

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

- خروجی = سیگنال دودویی (۱ یا ۰) یا دوقطبی (Bipolar) (۱ یا -۱)



شبکه عصبی مصنوعی ...

توابع فعال‌سازی متداول

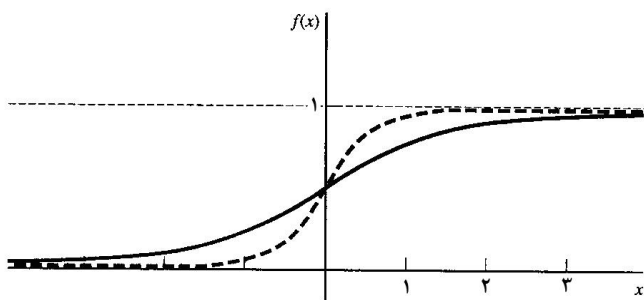
• توابع سیگموئید (Sigmoid Functions)

• منحنی‌هایی به شکل S

• استفاده در شبکه‌های عصبی پس‌انتشار (نیاز به مشتق‌گیری)

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

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

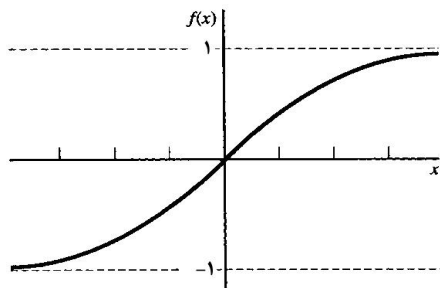


• سیگموئید دودویی - تابع لجستیک (Logistic Function)

• دامنه 0 تا 1، مقادیر مطلوب خروجی یا دودویی است و یا بین 0 و 1 است

• سیگموئید دوقطبی - شبیه به تابع تانژانت هیپربولیک (Hyperbolic Tangent Function)

• دامنه -1 تا 1



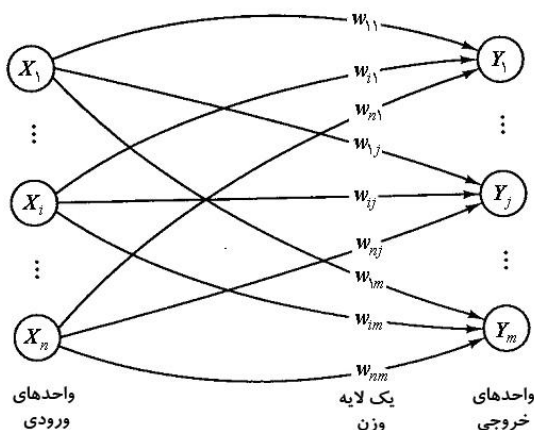
$$g(x) = 2f(x) - 1 = \frac{2}{1 + \exp(-\sigma x)} - 1 = \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)}$$

$$g'(x) = \frac{\sigma}{2} [1 + g(x)][1 - g(x)]$$

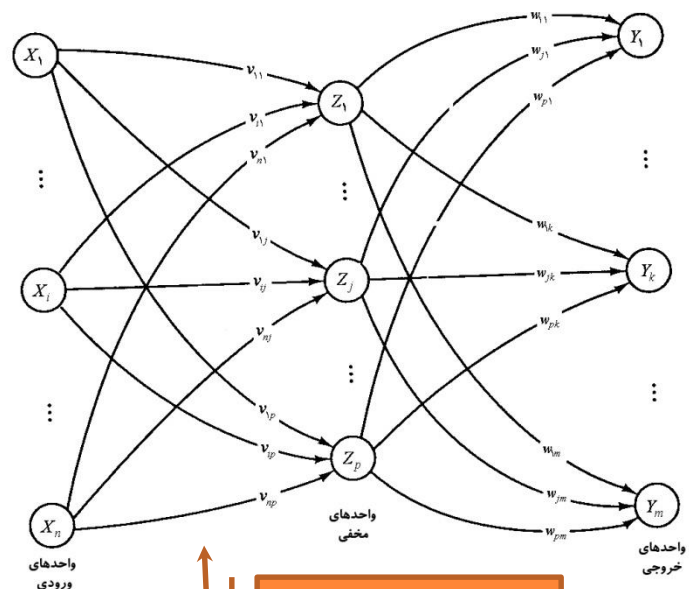
شبکه عصبی مصنوعی ...

○ 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

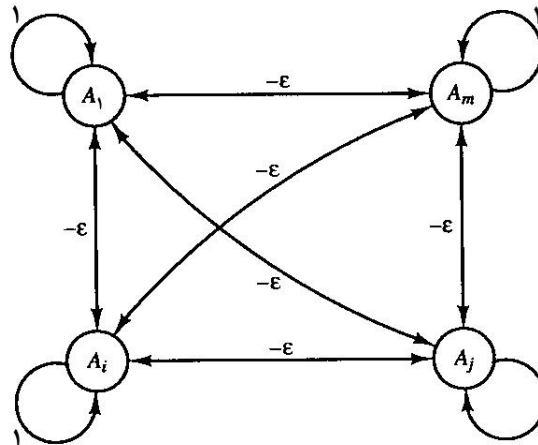


شبکه دولایه

شبکه عصبی مصنوعی ...

○ ساختارهای رایج

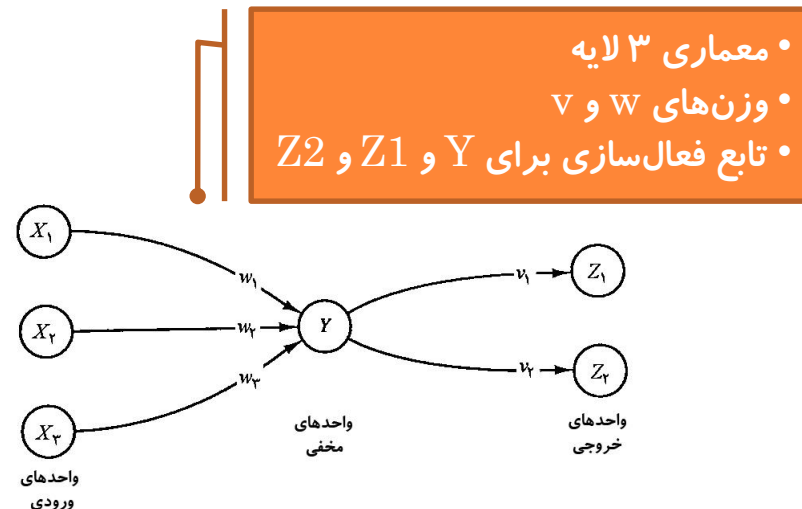
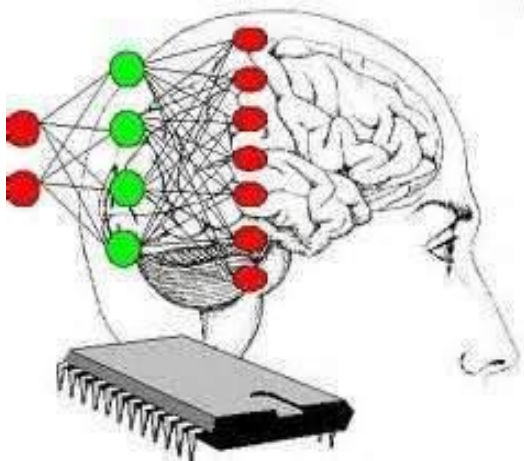
- شبکه بازگشتی (Recurrent)، مسیرهای بسته سیگنال از یک واحد به خودش وجود دارد
- شبکه رقابتی: واحدهای آن کاملاً به هم مرتبطند



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





شبکه عصبی مصنوعی: تاریخچه ...

○ دهه ۴۰ - اولین شبکه‌های عصبی مصنوعی

- ۱۹۴۳ - معرفی نرون مک‌کلاچ-پیتز (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 حافظه پیوندی



شبکه عصبی مصنوعی: تاریخچه ...

○ دهه ۸۰ - شکوفایی شبکه‌های عصبی

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



شبکه عصبی مصنوعی: تاریخچه

○ دهه ۹۰ - دهه کاربرد

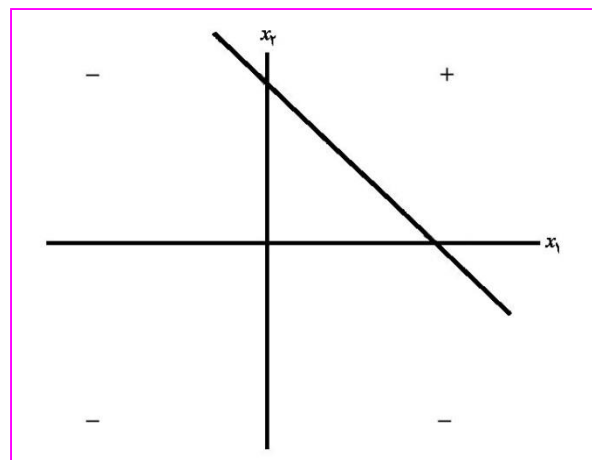
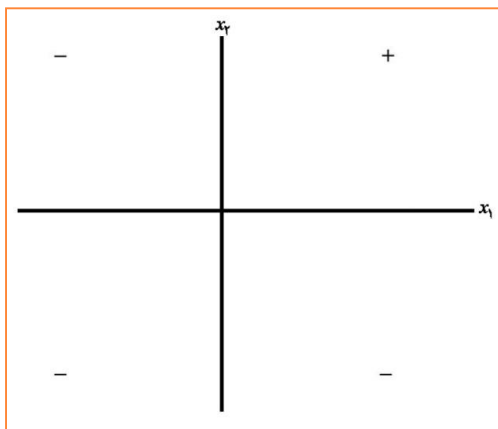
- به کارگیری شبکه‌های عصبی در کاربردهای مختلف
- توسعه شبکه توابع پایه شعاعی (RBF)
- ۱۹۹۷ - شبکه بازگشتی LSTM

○ ۲۰۰۰ به بعد

- یادگیری عمیق (Deep Learning)
- شبکه‌های بازگشتی

شبکه عصبی پرسپترون ...

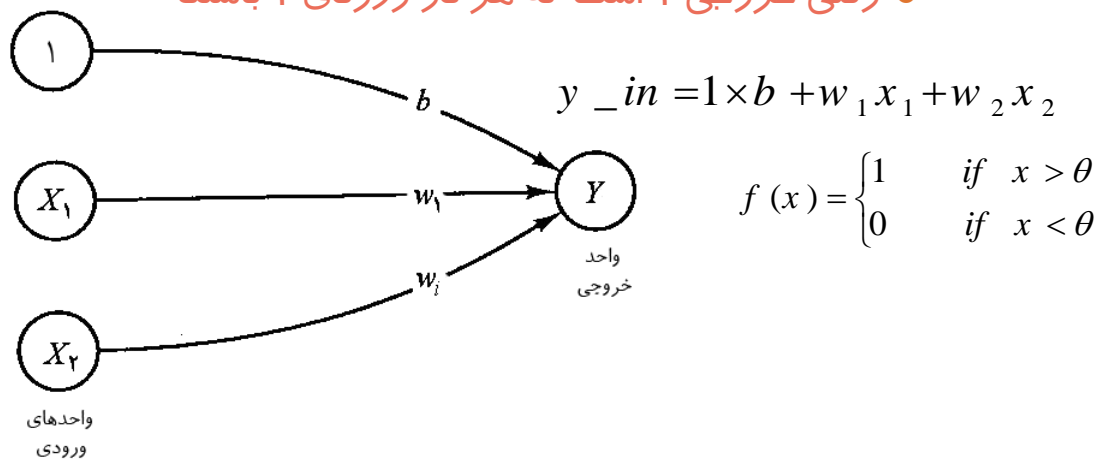
INPUT(x_1, x_2)	OUTPUT
(1, 1)	+1
(1, -1)	-1
(-1, 1)	-1
(-1, -1)	-1



مثال: تابع AND

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

• وقتی خروجی ۱ است که هر دو ورودی ۱ باشند

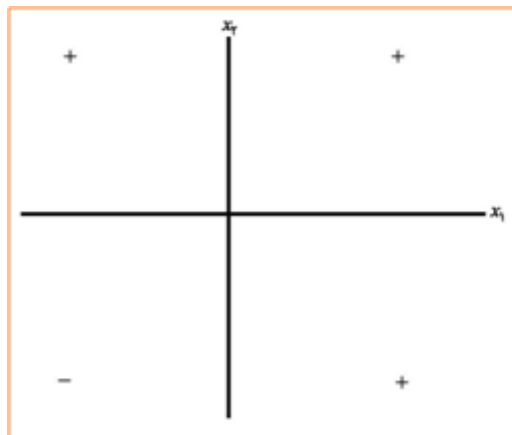


• مرز تصمیم‌گیری $b + w_1 x_1 + w_2 x_2 = 0$

• پاسخ $b = -1, w_1 = 1, w_2 = 1$

$$x_2 = -x_1 + 1$$

شبکه عصبی پرسپترون ...



مثال: تابع OR

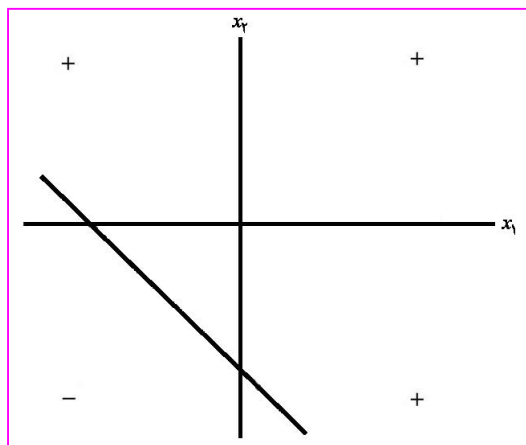
- دو ورودی (دوقطبی) و یک خروجی (دوقطبی)

• وقتی خروجی ۱ است که حداقل یکی از ورودی‌ها ۱ باشد

INPUT(x_1, x_2)

OUTPUT

(1, 1)	+1
(1, -1)	+1
(-1, 1)	+1
(-1, -1)	-1



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

$$x_2 = -x_1 - 1$$

- مرز تصمیم‌گیری

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



شبکه عصبی پرسپترون ...

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

- الگوریتم‌های یادگیری شبکه عصبی: هب، پرسپترون، آدالین 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 پرسپترون: ساختار...

○ ساختار اولیه

- مدل تقریبی شبکه چشم

- سه لایه نرون (واحدهای حسی، واحدهای پیونددهنده و واحد پاسخ)

- فقط وزن‌های بین لایه‌های دوم و سوم آموزش داده می‌شود

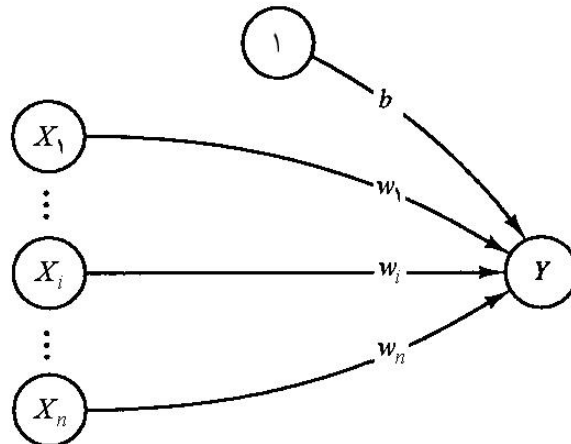
- خروجی واحدهای پیونددهنده به واحدهای پاسخ یک بردار دودویی است

- عملاً شبکه ای 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 +1

- متعلق نبودن با پاسخ -1



شبکه عصبی پرسپترون: الگوریتم ...

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

تعیین نرخ یادگیری $0 < \alpha \leq 1$ (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:

الگوریتم
تکراری

$$y_{in} = b + \sum_i x_i w_i$$

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

۱ = تعلق به دسته -۱ = عدم تعلق به دسته ۰ = ناحیه عدم تصمیم‌گیری (2θ)

شبکه عصبی پرسپترون: الگوریتم ...

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

اگر $y \neq t$ است، آنگاه: $w_i(new) = w_i(old) + \alpha x_i t$

$$b(new) = b(old) + \alpha t$$

به‌روز کردن
مشروط وزن‌ها

$$w_i(new) = w_i(old)$$

$$b(new) = b(old)$$

در غیراین صورت:

- 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

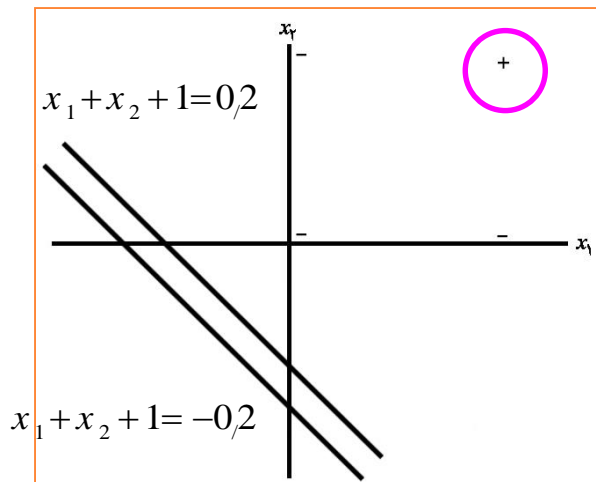
نرخ یادگیری

شبکه عصبی پرسپترون: مثال ...

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

- **دودویی**: مقادیر صفر و یک - **دوقطبی**: مقادیر +1 و -1
- وزن‌های اولیه و بایاس را صفر؛ نرخ اولیه یادگیری = 1؛ آستانه = 0.2.
- ارائه ورودی اول

			WEIGHT					
INPUT	NET	OUT	TARGET	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	$(1 \ 1 \ 1)$	$(1 \ 1 \ 1)$			



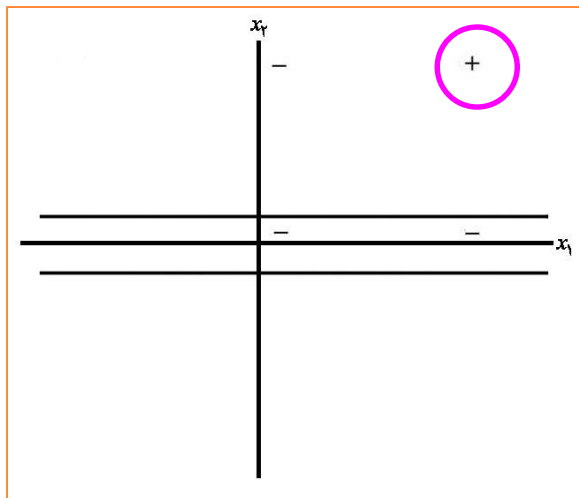


شبکه عصبی پرسپترون: مثال ...

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

• ارائه دومین ورودی

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



$$x_2 = 0.2$$

$$x_2 = -0.2$$



شبکه عصبی پرسپترون: مثال ...

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

• ارائه سومین ورودی

						WEIGHT					
INPUT			NET	OUT	TARGET	CHANGES			WEIGHTS		
$(x_1$	x_2	1)	y_{in}	y	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
									(0	1	0)
(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، وزن‌ها تغییری نمی‌کنند

						WEIGHT					
INPUT			NET	OUT	TARGET	CHANGES			WEIGHTS		
$(x_1$	x_2	$1)$	y_{in}	y	t	$(\Delta w_1$	Δw_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)



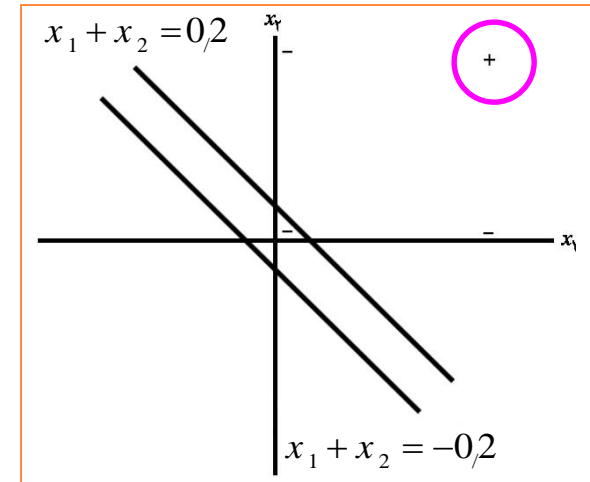
شبکه عصبی پرسپترون: مثال ...

○ 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

				WEIGHT							
INPUT			NET	OUT	TARGET	CHANGES			WEIGHTS		
$(x_1$	x_2	$1)$	y_{in}	y	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
									$(0$	0	$-1)$
$(1$	1	$1)$	-1	-1	1	$(1$	1	$1)$	$(1$	1	$0)$

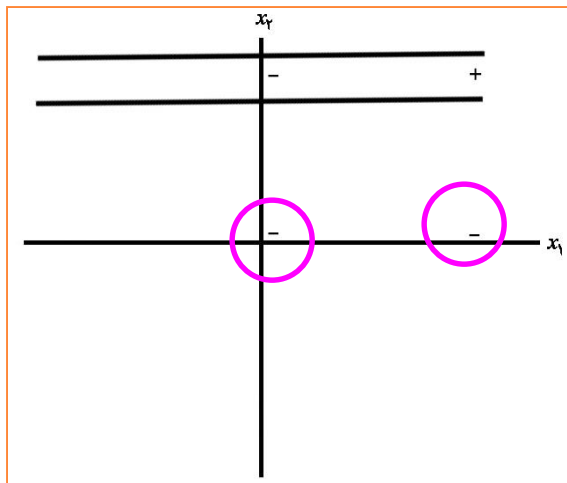


شبکه عصبی پرسپترون: مثال ...

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

• دومین دور آموزش - ارائه دومین ورودی

				WEIGHT							
INPUT			NET	OUT	TARGET	CHANGES			WEIGHTS		
$(x_1$	x_2	1)	y_{in}	y	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
									(1	1	0)
(1	0	1)	1	1	-1	(-1	0	-1)	(0	1	-1)



$$x_2 - 1 = 0/2$$

$$x_2 - 1 = -0/2$$



شبکه عصبی پرسپترون: مثال ...

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

• دومین دور آموزش - ارائه سومین ورودی

○ پاسخ برای تمام ورودی‌ها منفی

						WEIGHT					
INPUT			NET	OUT	TARGET	CHANGES			WEIGHTS		
$(x_1$	x_2	$1)$	y_{in}	y	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
									$(0$	1	$-1)$
$(0$	1	$1)$	0	0	-1	$(0$	-1	$-1)$	$(0$	0	$-2)$

• دومین دور آموزش - ارائه چهارمین ورودی

○ پاسخ برای تمام ورودی‌ها منفی

						WEIGHT					
INPUT			NET	OUT	TARGET	CHANGES			WEIGHTS		
$(x_1$	x_2	1)	y_{in}	y	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
									(0	0	-2)
(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

				WEIGHT							
INPUT			NET	OUT	TARGET	CHANGES			WEIGHTS		
$(x_1$	x_2	$1)$	y_{in}	y	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
									$(0$	0	$-2)$
$(1$	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 پرسپترون: مثال ...

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

• پنجمین، ششمین، ... دور آموزش

(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)

• نهمین دور آموزش

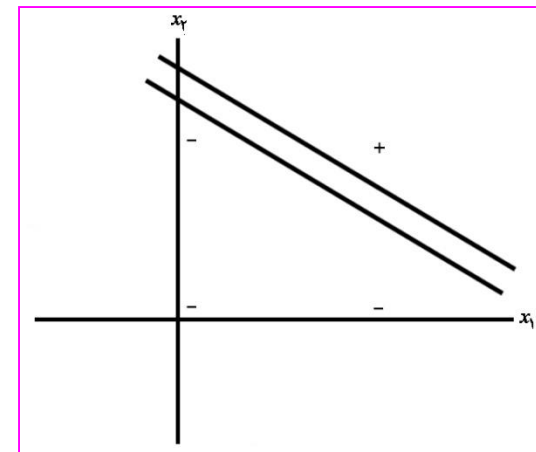
• دهمین دور آموزش

○ عدم تغییر وزن‌ها = توقف الگوریتم

○ همگرایی وزن‌ها

(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)

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





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			WEIGHTS		
				CHANGES					
$(x_1 \ x_2 \ 1)$	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	$(0 \ 0 \ 0)$			$(1 \ 1 \ -1)$		

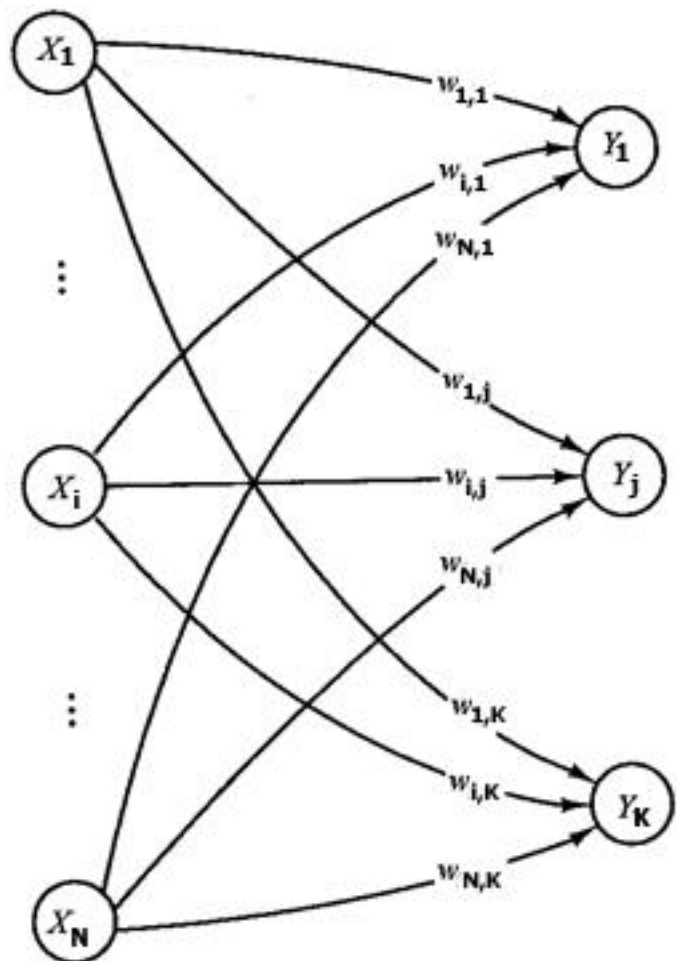
- دور اول آموزش

$(1 \ 1 \ 1)$	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 پرسپترون: مثال ...



تشخیص عنوان (موضوع) متن

- تعداد K دسته (موضوع) $C = \{c_1, c_2, \dots, c_K\}$

• تعداد K نرون خروجی

- داده = سند متنی

• 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 $N \times K$ (مدل)

آزمون

• ورودی: یک سند با عنوان نامشخص

• تبدیل سند به یک بردار N بعدی و دادن آن به شبکه

• خروجی: نرونی (دسته ای) که مقدار بزرگ‌تر دارد

شبکه عصبی پرسپترون: مثال ...

تشخیص زبان

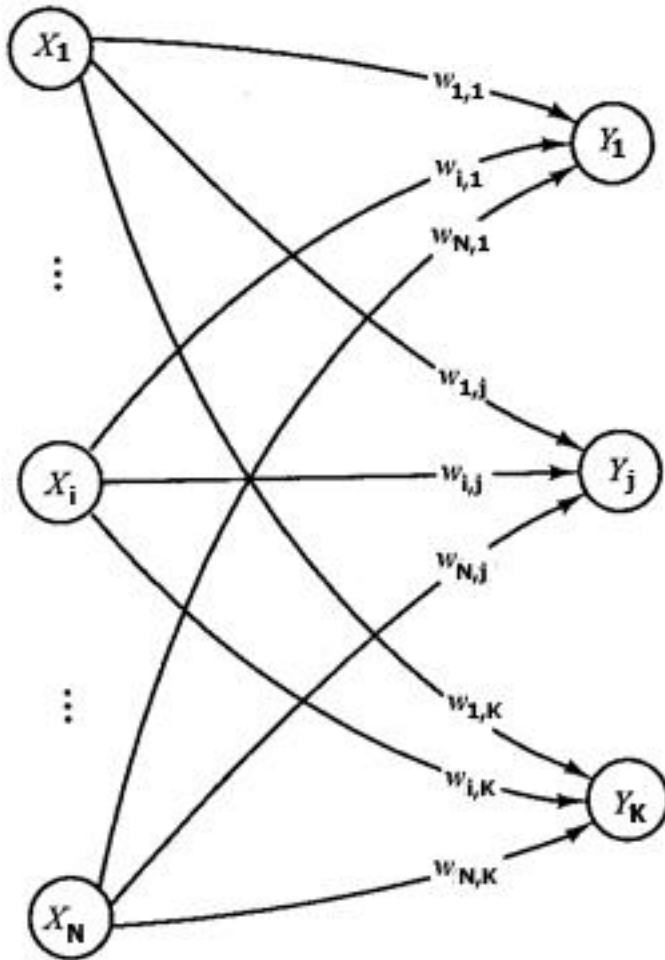
- تعداد K دسته (زبان) $C = \{c_1, c_2, \dots, c_K\}$

○ تعداد K نرون خروجی

- داده = سند متنی یا سند صوتی

○ تبدیل هر سند به یک بردار ویژگی N بعدی

○ تعداد N نرون ورودی



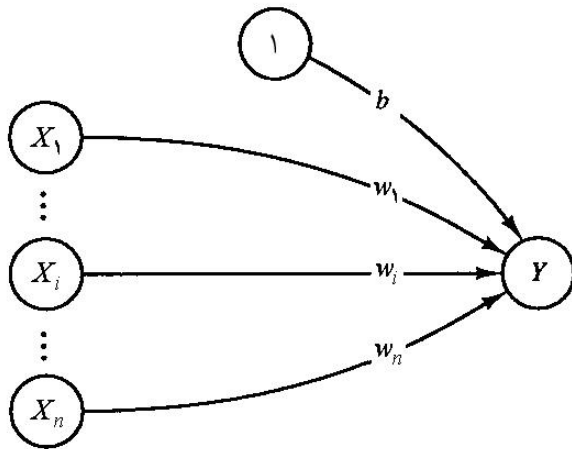
شبکه عصبی پرسپترون: مثال ...

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 ...

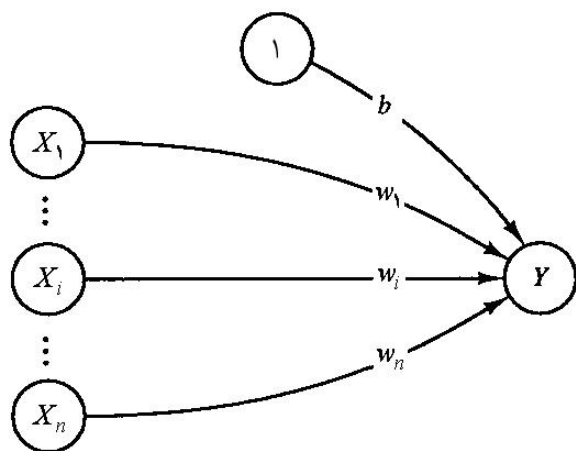
شبکه عصبی پرسپترون: مثال ...

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 ...

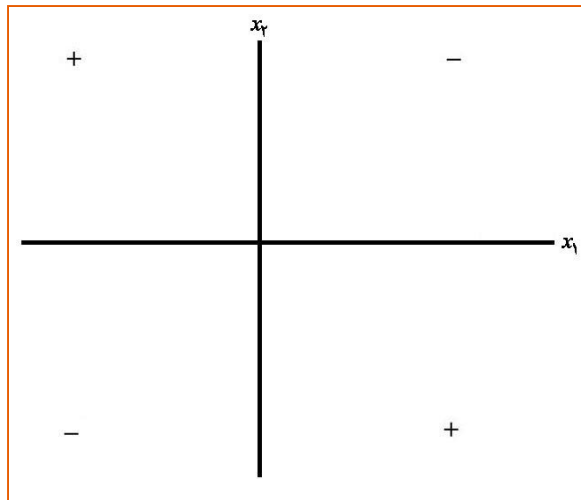


شبکه عصبی پرسپترون: مثال

● مثال: تابع XOR

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

○ وقتی خروجی ۱ است که فقط یکی از ورودی‌ها ۱ باشد



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

• حل؟

○ الگوریتم همگرا نمی‌شود (جواب درست نمی‌دهد)

• فضای داده‌های ورودی به صورت خطی جدایی‌پذیر (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 به طوری که 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 مراحل 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

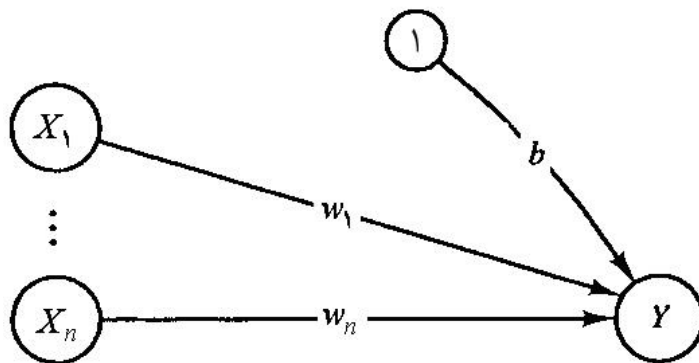
- برقراری این قضیه فقط برای مسائل خطی جدایی‌پذیر (Linearly Separable)

شبکه آدالاین ...

○ آدالاین = نرون خط وفقی (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 آدالاین: 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 $s:t$ 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 به‌روز 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، وگرنه ادامه دهید.



شبکه آدالاین: الگوریتم

○ تفاوت یادگیری آدالاین با یادگیری پرسپترون

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

○ نرخ یادگیری

- تاثیر بر سرعت و روند همگرایی الگوریتم
- روش: ابتدا مقدار را بزرگ فرض کرده (مثلاً 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 آدالاین: مثال ...

○ تابع AND: ورودی‌های **دودویی**، هدف‌های **دوقطبی**

• شبکه بعد از آموزش

x_1	x_2	t
1	1	1
1	0	-1
0	1	-1
0	0	-1

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

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

• مربعات خطا برای چهار الگوی آموزش با این وزن‌ها = ۱

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



شبکه عصبی پرسپترون چندلایه (MLP) ...

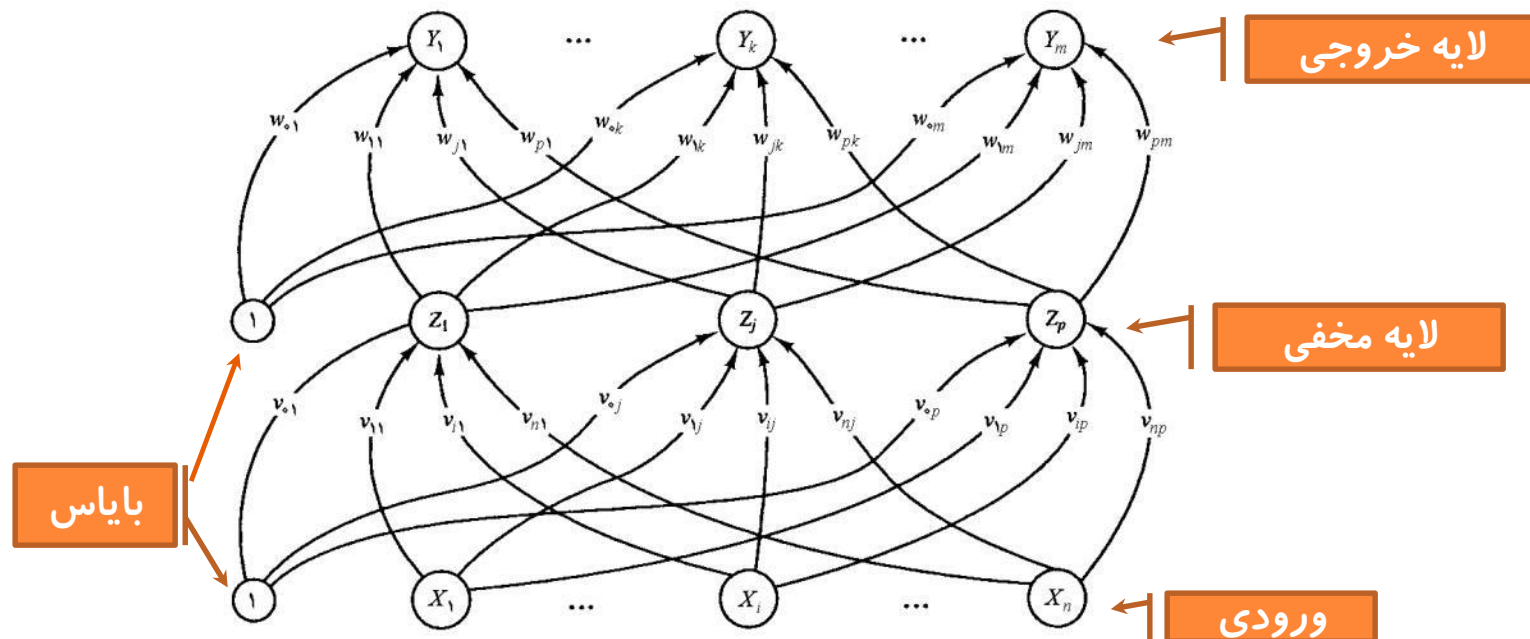
○ شبکه عصبی پرسپترون چندلایه (MLP: Multi-Layer Perceptron)

- توسعه شبکه‌های عصبی به حالت چند لایه
- آموزش با الگوریتم پس‌انتشار 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 شبکه‌(های) عصبی

شبکه عصبی پرسپترون چندلایه (ساختمار) ...

○ شبکه سه لایه

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





شبکه عصبی پرسپترون چندلایه (الگوریتم آموزش)...

○ مراحل

- پیش‌خور کردن الگوی آموزش ورودی
- پس‌انتشار خطای مربوط
- تنظیم وزن‌ها

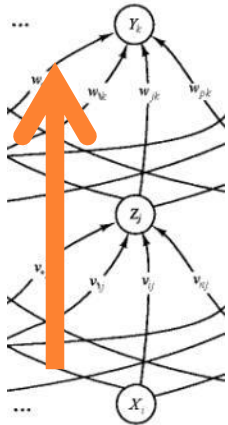
- مبنای ریاضی الگوریتم پس‌انتشار = بهینه‌سازی کاهش گرادیان (Gradient Descent)
 - گرادیان (شیب) یک تابع = نمایانگر جهتی که تابع در آن سریع‌تر افزایش می‌یابد
 - شیب با علامت منفی = جهتی نشان دهنده کاهش سریع‌تر آن تابع
 - در اینجا تابع مورد نظر = تابع خطای شبکه
 - متغیرهای مورد نظر = وزن‌های شبکه

شبکه عصبی 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

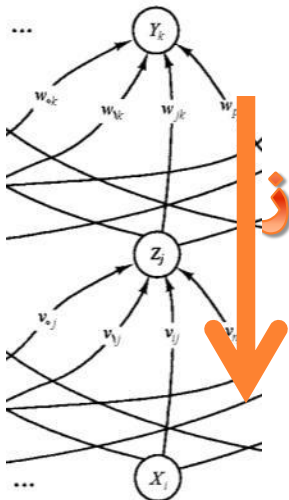


$$z_in_j = v_{0j} + \sum_{i=1}^n x_i v_{ij} \quad z_j = f(z_in_j)$$

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

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

شبکه عصبی پرسپترون چندلایه (الگوریتم آموزش)



○ پس‌انتشار 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 می‌رود) $\Delta w_{jk} = \alpha \delta_k z_j$

محاسبه پارامتر تصحیح بایاس (بعداً در به‌روز کردن به کار می‌رود) $\Delta w_{0k} = \alpha \delta_k$

ارسال δ_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 به‌روز کردن)

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

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



شبکه عصبی پرسپترون چندلایه (الگوریتم آموزش) ...

○ به‌روز کردن وزن‌ها و بایاس‌ها

• مرحله ۸- به‌روز کردن وزن‌ها و بایاس‌های واحدهای خروجی

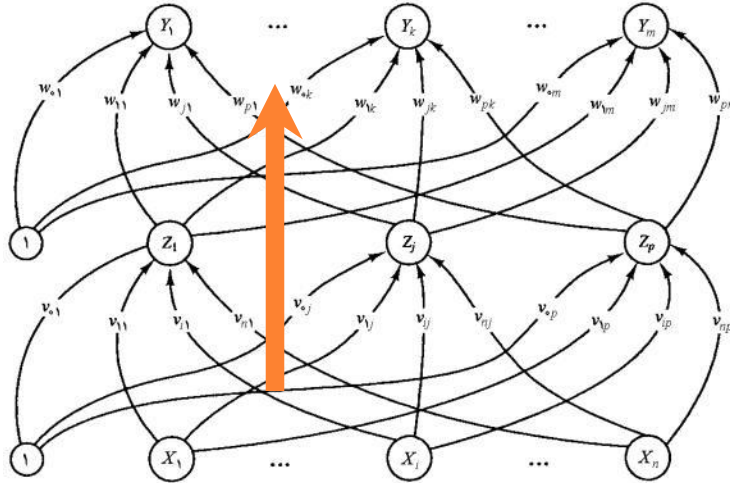
$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

به‌روز کردن وزن‌ها و بایاس‌های واحدهای مخفی

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

• مرحله ۹- شرایط توقف را بررسی کنید.

شبکه عصبی 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:

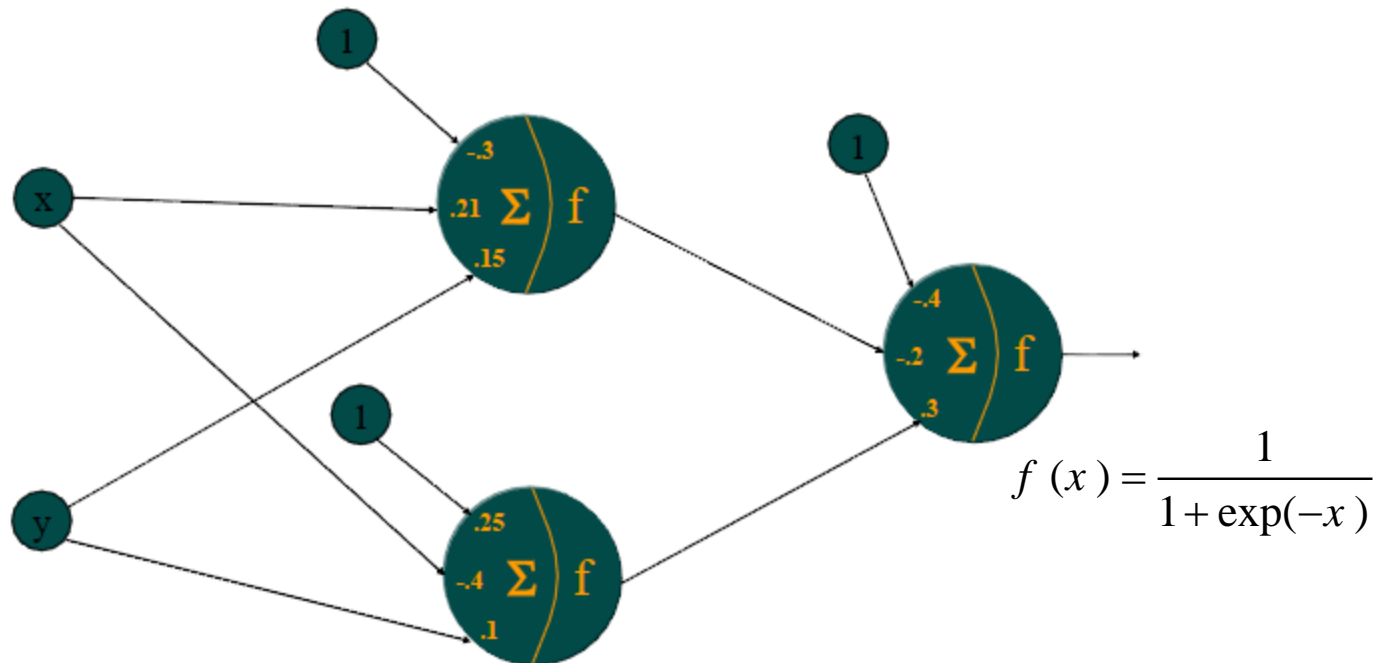
$$y_in_k = w_{0k} + \sum_{j=1}^p z_j w_{jk} \Rightarrow y_k = f(y_in_k)$$

شبکه عصبی 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)

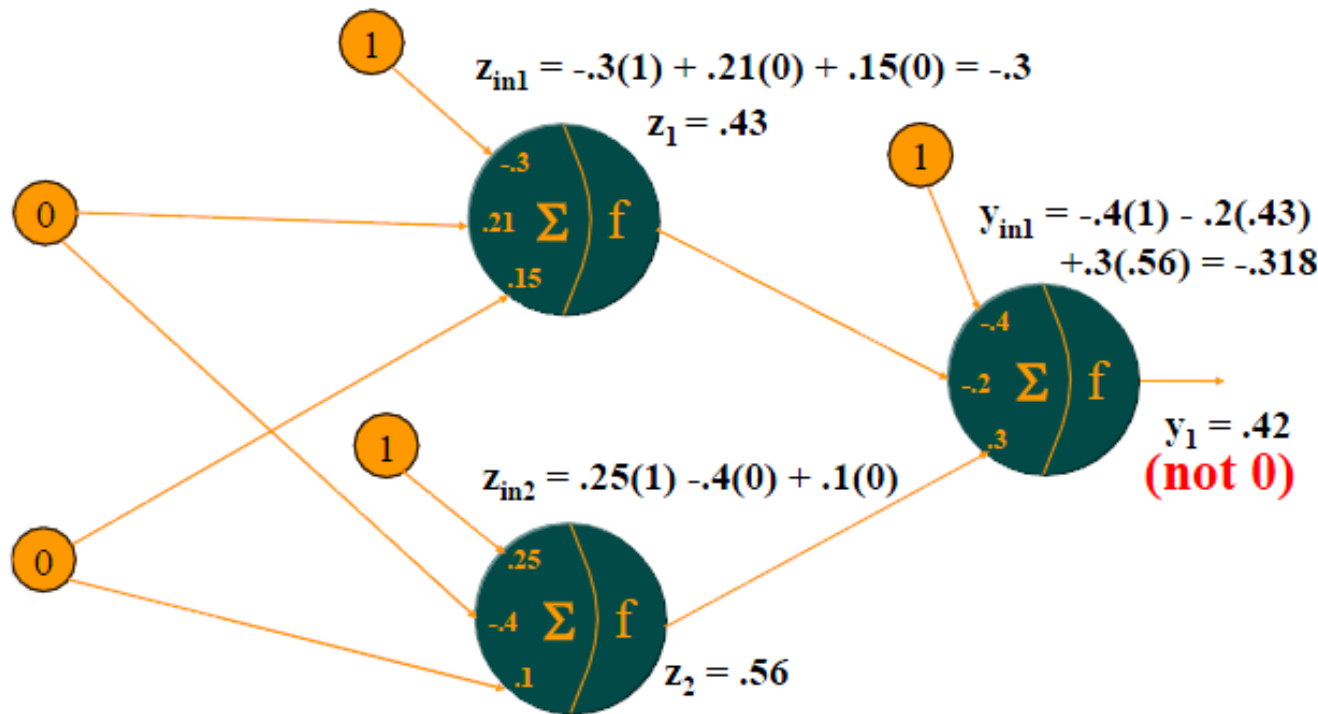


شبکه عصبی پرسپترون چندلایه (مثال) ...

○ تابع XOR: نمایش دودویی (۲ از ۶) ...

• پیش‌خور کردن ورودی

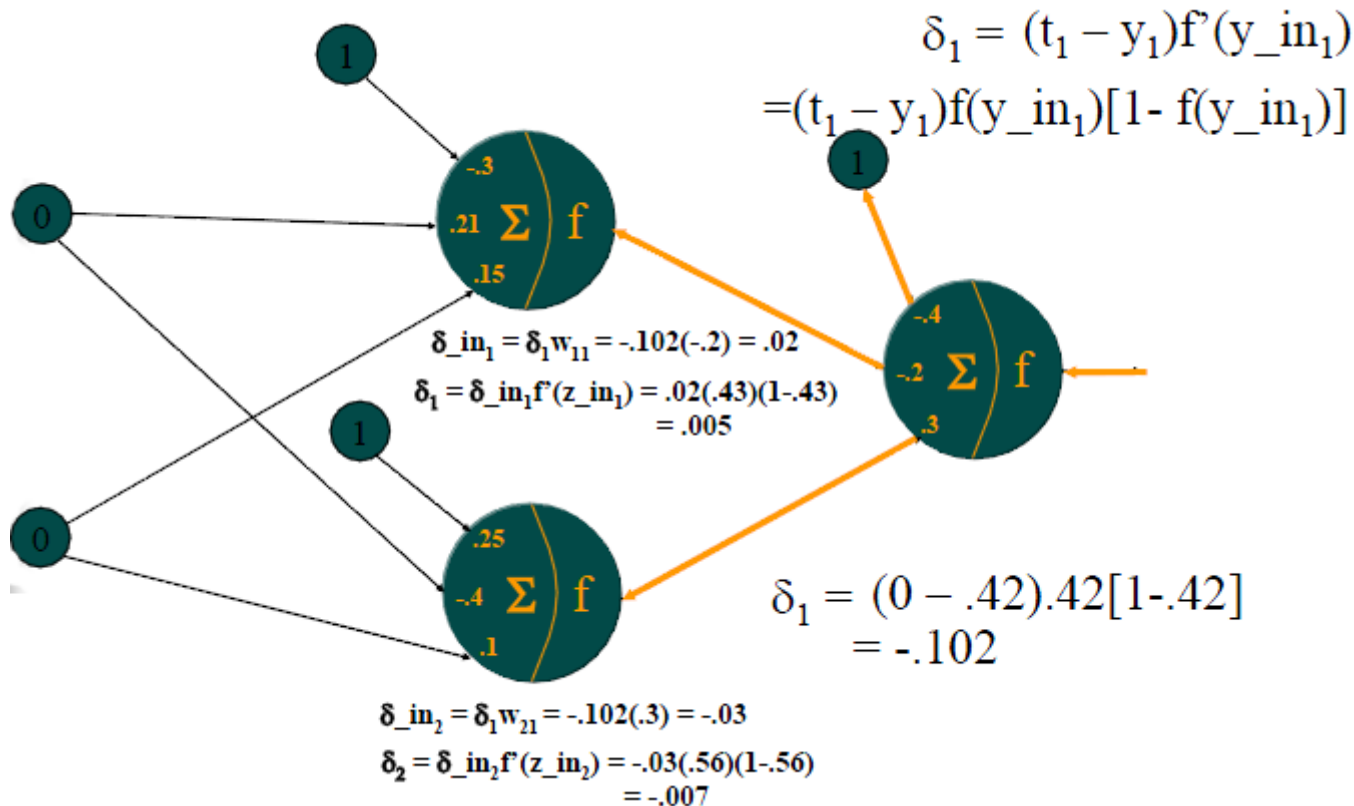
x_1	x_2	y
0	0	0



شبکه عصبی پرسپترون چندلایه (مثال) ...

○ تابع XOR: نمایش دودویی (۳ از ۶) ...

• پس انتشار خطا



شبکه عصبی پرسپترون چندلایه (مثال) ...

○ تابع XOR: نمایش دودویی (۴ از ۶) ...

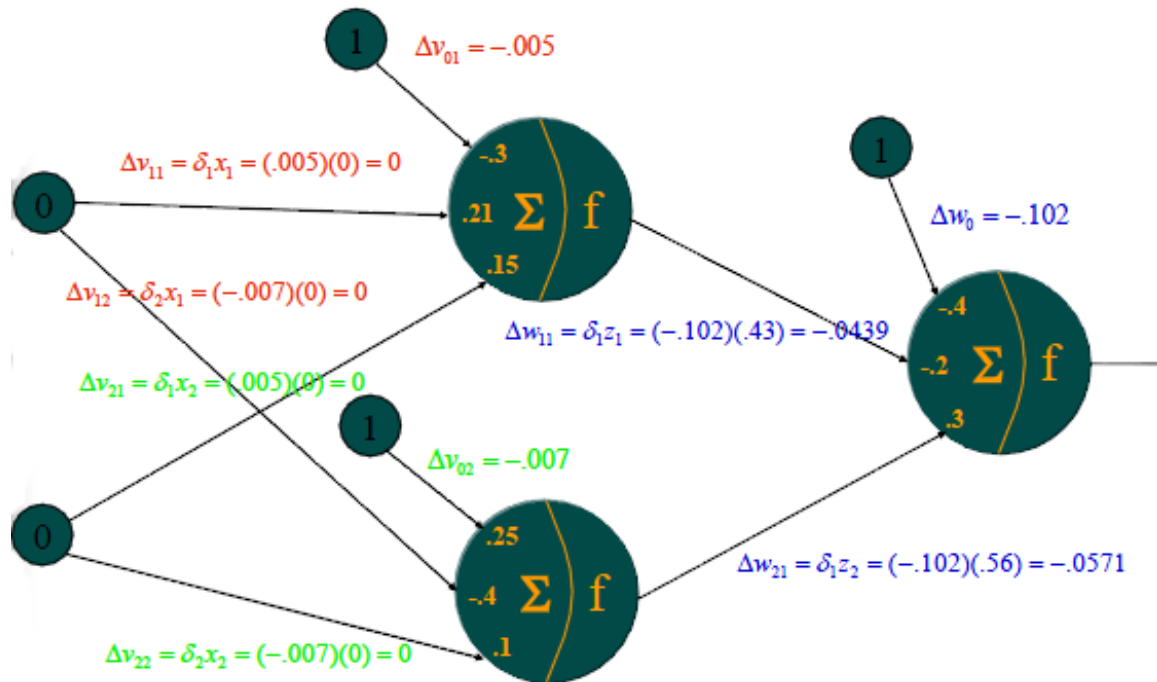
• محاسبه وزن‌ها

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

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

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

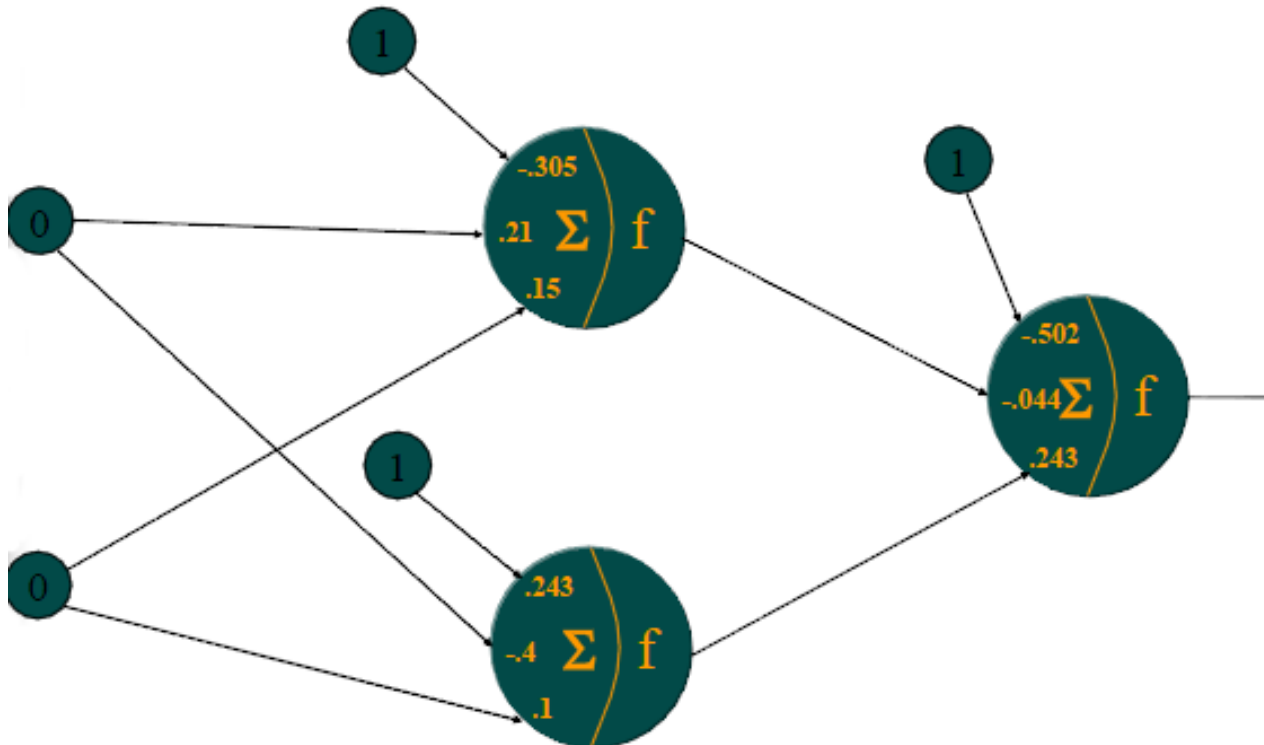
$$\Delta w_0 = \alpha \delta_1$$



شبکه عصبی پرسپترون چندلایه (مثال) ...

○ تابع XOR: نمایش دودویی (۵ از ۶) ...

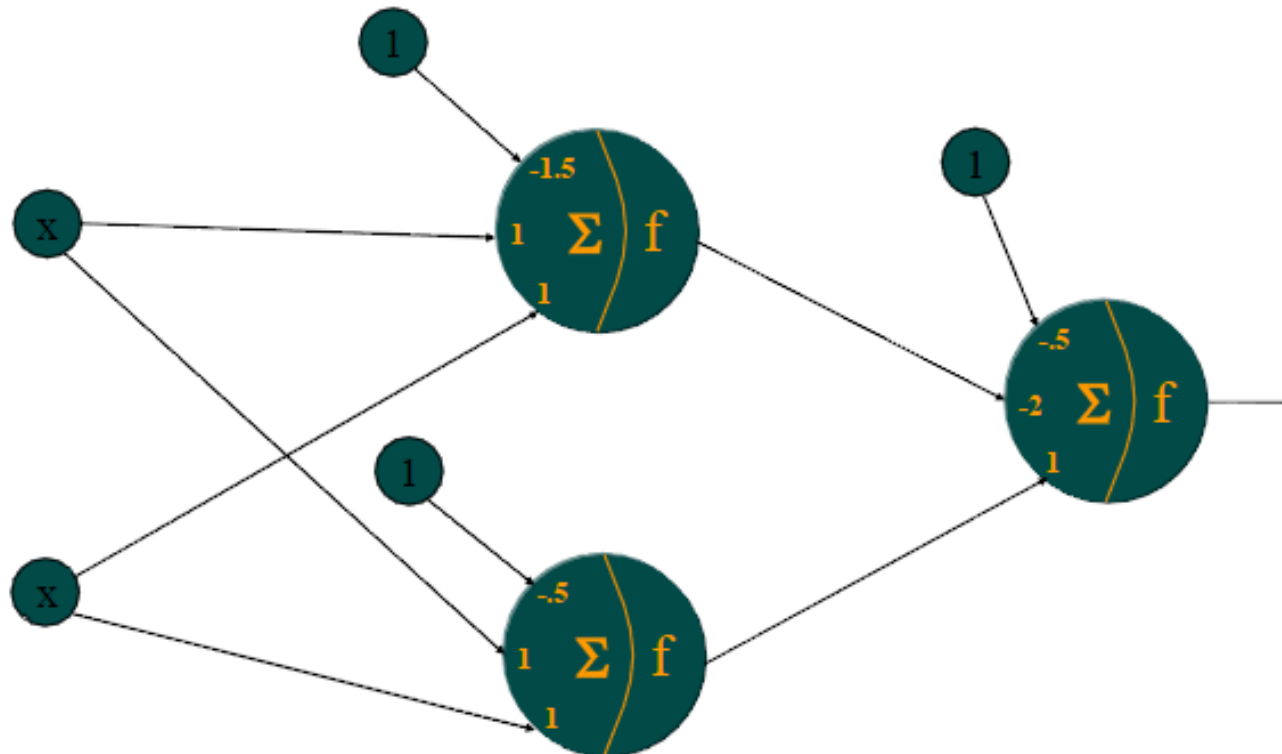
• به‌روز کردن وزن‌ها



شبکه عصبی پرسپترون چندلایه (مثال) ...

○ تابع XOR: نمایش دودویی (۶ از ۶)

• وزن‌های نهایی (بعد از ۵۰۰ تکرار)



شبکه عصبی پرسپترون چندلایه (مثال) ...

○ کاربرد در پردازش گفتار و پردازش زبان

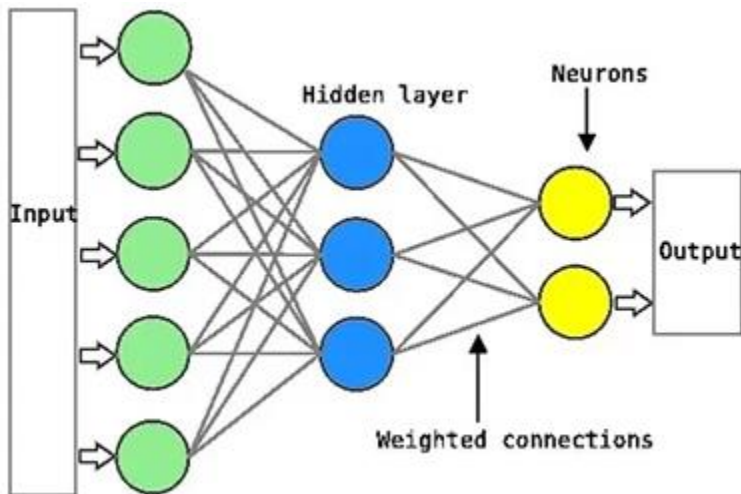
- تشخیص گوینده
- تشخیص جنسیت
- جداسازی گفتار از غیر گفتار

• تعیین نقش دستوری (POS Tagging)

• تعیین شباهت دو متن

• تعیین عنوان متن

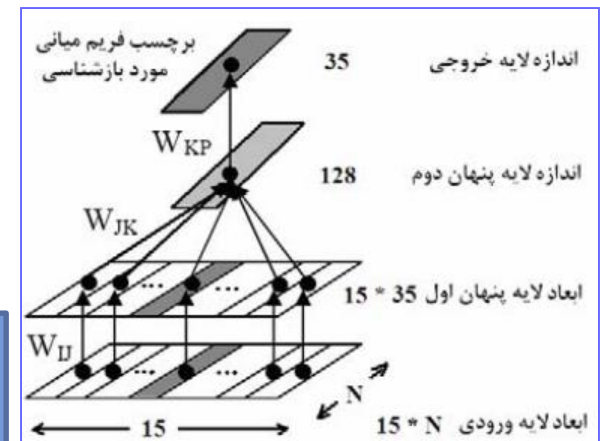
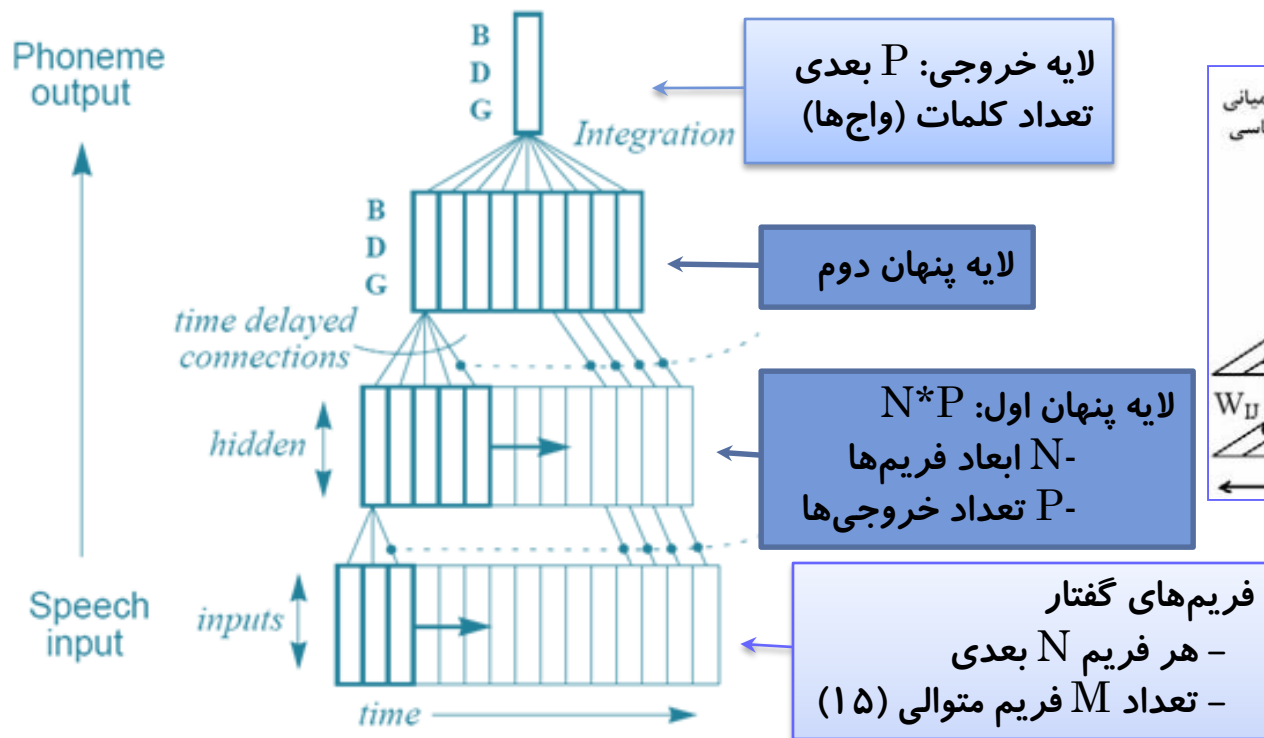
• ...



شبکه عصبی پرسپترون چندلایه (مثال) ...

○ تشخیص گفتار با TDNN: Time-Delay Neural Network

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



تشخیص واج‌های فارسی

شبکه عصبی پرسپترون چندلایه (مثال) ...

متن‌خوان انگلیسی NETtalk

- ورودی: متن نوشته شده و خروجی: واج‌هایی که صدا را می‌سازند
- وابستگی واج‌ها به محتوا

○ تفاوت تلفظ 'a' در "brave"، "gave" (کشیده) و "have" (کوتاه)

- تبدیل متن به صورت واجی

Phone → f o n (f-on-)

۲۶ نرون = ۲۶ ویژگی

۲۳ ویژگی زبانی (صدا دار،

بی صدا، خیشومی و ...)

۳ ویژگی نوایی (استرس و ...)

Phonemes

(/k/)

Best Match

Features

Output Layer

۸۰ نرون

Hidden Layer

Input Layer

(_ a _ c a t _)

Spelling

۲۰۳ = ۲۹ * ۷ نرون ورودی

۷ = واج موردنظر و ۳ واج در دوطرف آن

۲۹ = بردار دودویی، هر بعد معادل یک واج

شبکه عصبی پرسپترون چندلایه (مثال) ...

بردار کلمات ...

• Word Embedding, Word2Vec

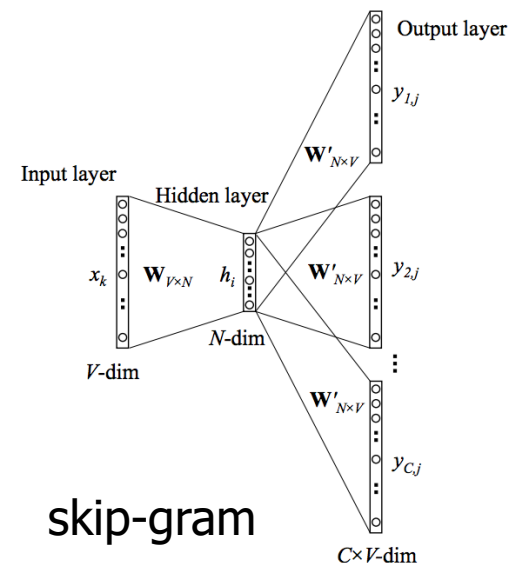
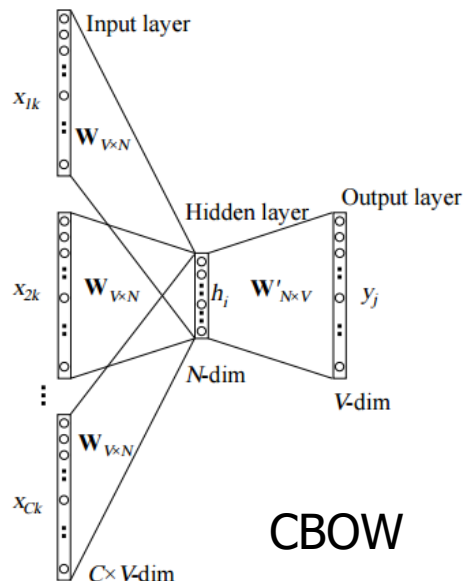
• روش‌ها

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

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

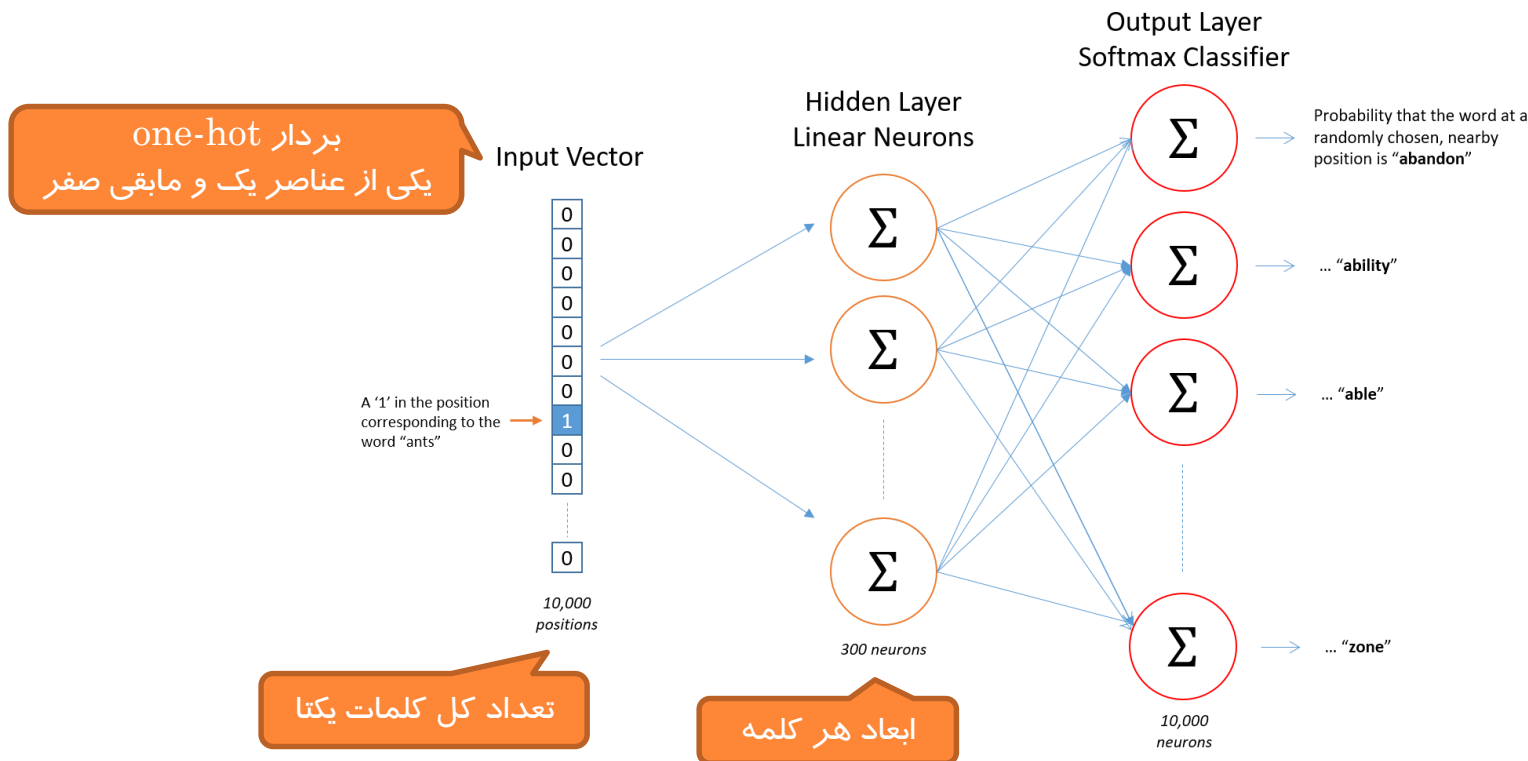
• پرش چندتایی (skip-gram)

• پیش بینی کلمات (احتمال همه کلمه های بعدی) از روی کلمه فعلی



شبکه عصبی پرسپترون چندلایه (مثال) ...

بردار کلمات: پرس چندتایی





شبکه عصبی پرسپترون چندلایه (مثال) ...

بردار کلمات: پرس چندتایی

• داده ورودی

Source Text

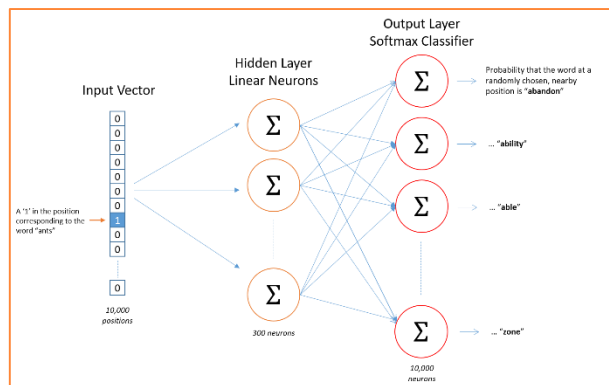
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

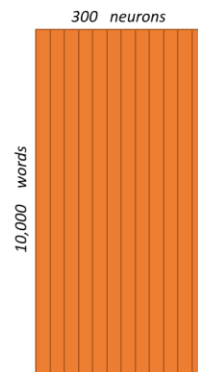
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

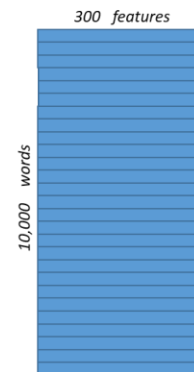
• بردار کلمات نهایی



Hidden Layer Weight Matrix



Word Vector Lookup Table!



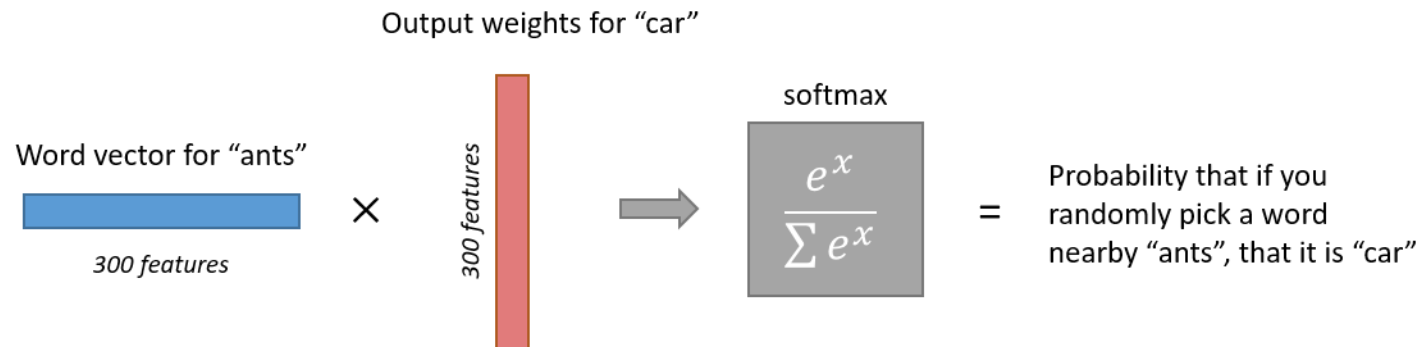


شبکه عصبی پرسپترون چندلایه (مثال) ...

○ 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





شبکه عصبی پرسپترون چندلایه (مثال) ...

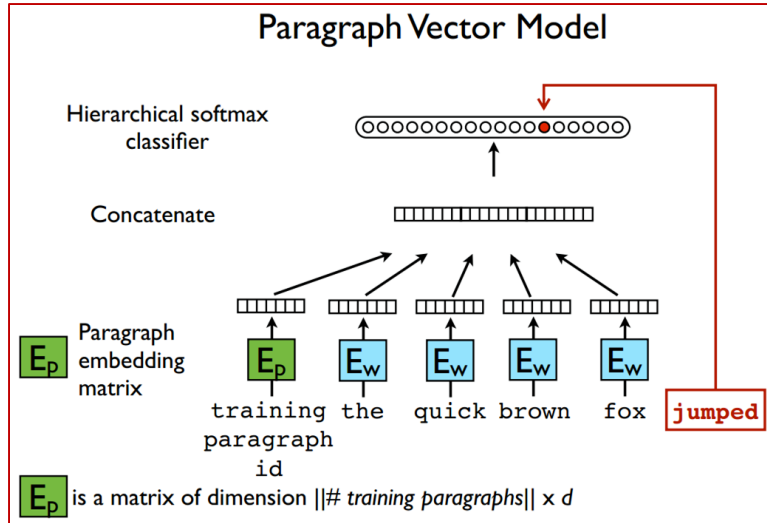
○ بردار جمله/پاراگراف/سند

• ترکیب بردار کلمات ساده

○ اصفهان + رودخانه → زاینده رود

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	$P_i = v_i(t_1 t_2)$

شبکه عصبی پرسپترون چندلایه (مثال)

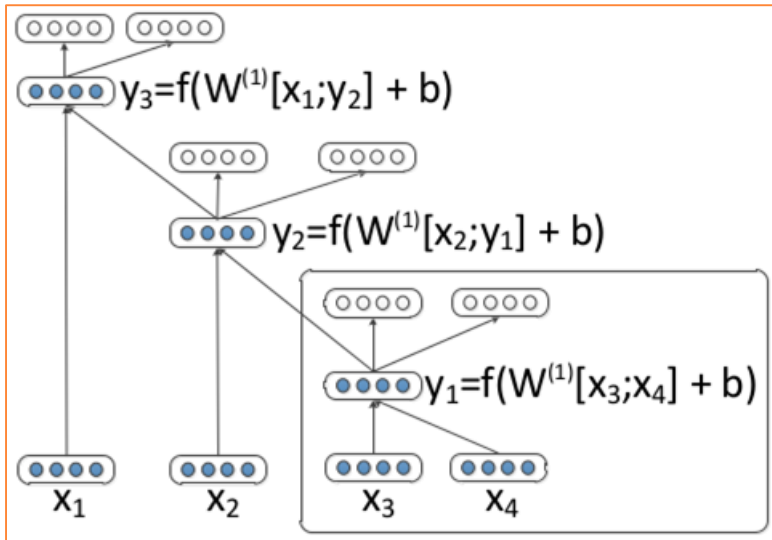


○ بردار جمله / پاراگراف / سند

• بردار پاراگراف

Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," In Proceedings of ICML, 2014.

• شبکه عصبی خودرمز گذار بازگشتی



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.



شبکه عصبی 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 اولیه تصادفی (مثبت یا منفی)
- بازه متداول برای مقادیر تصادفی وزن‌ها و بایاس‌ها بین ۰.۵ و -۰.۵ (یا بین ۱- و ۱)

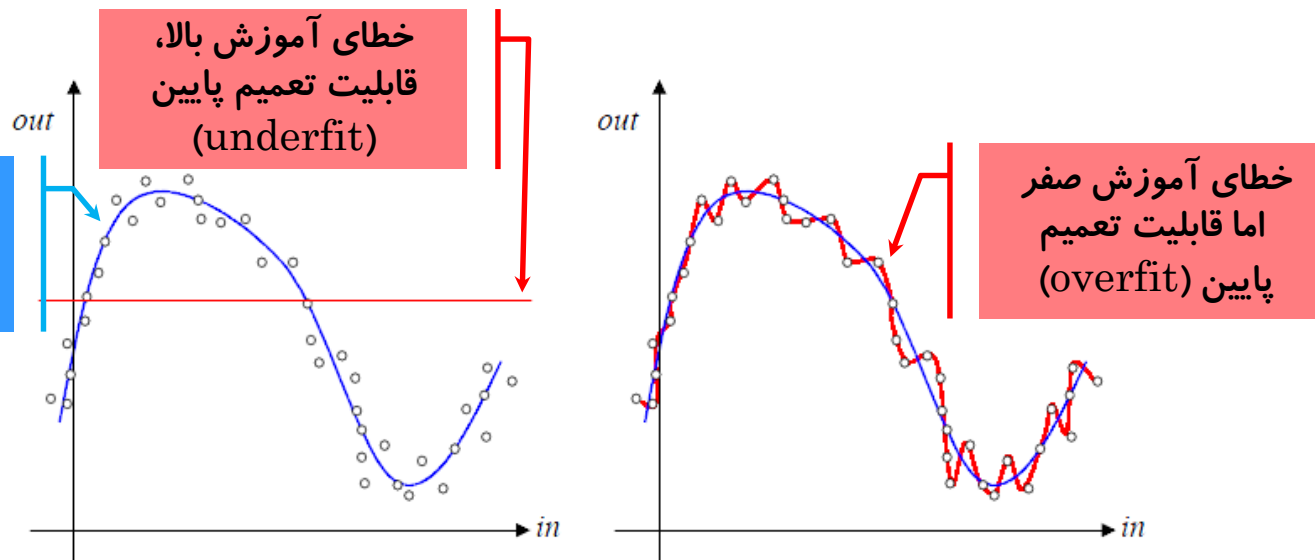
○ آموزش شبکه عصبی با بیش از یک لایه مخفی

- 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 پس‌انتشار تکرار می‌شود.
- یک لایه مخفی در شبکه پس‌انتشار 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.

شبکه عصبی پرسپترون چندلایه: 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



خطای آموزش بالا
اما قابلیت تعمیم بالا

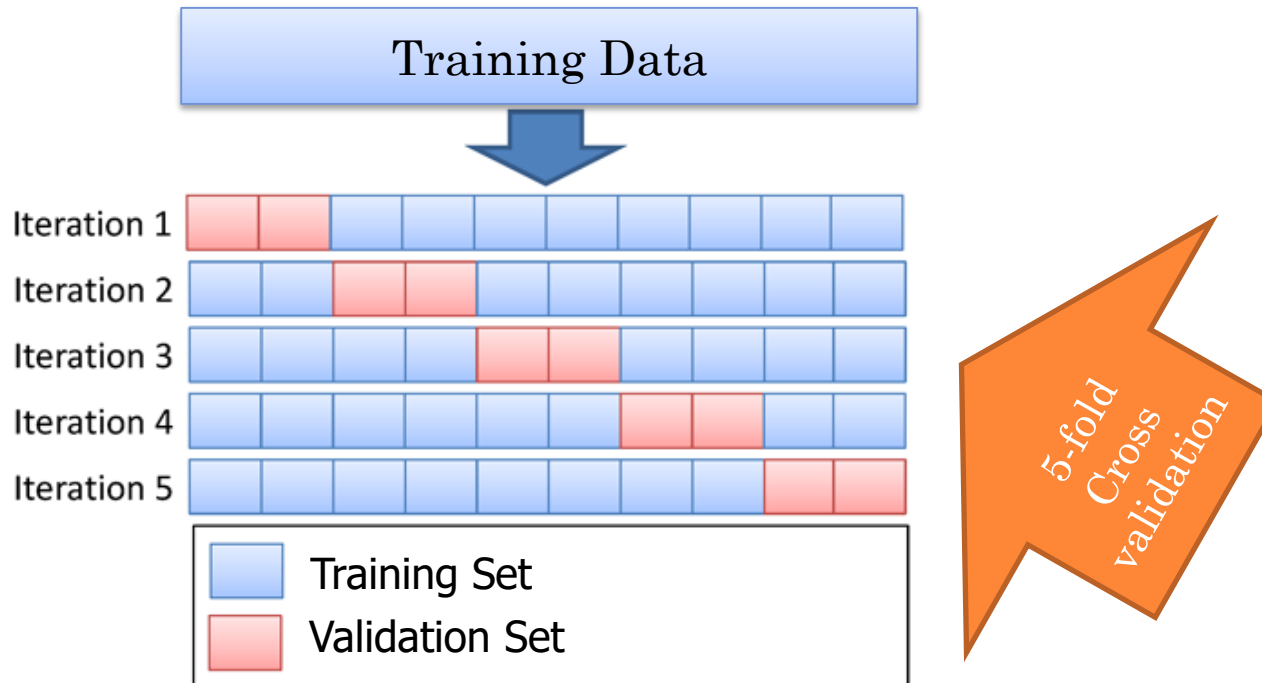


شبکه عصبی پرسپترون چندلایه: نکات تکمیلی...

○ تعادل بین یادگیری الگوها و تعمیم

• استفاده از دو مجموعه داده مجزا در زمان آموزش شبکه

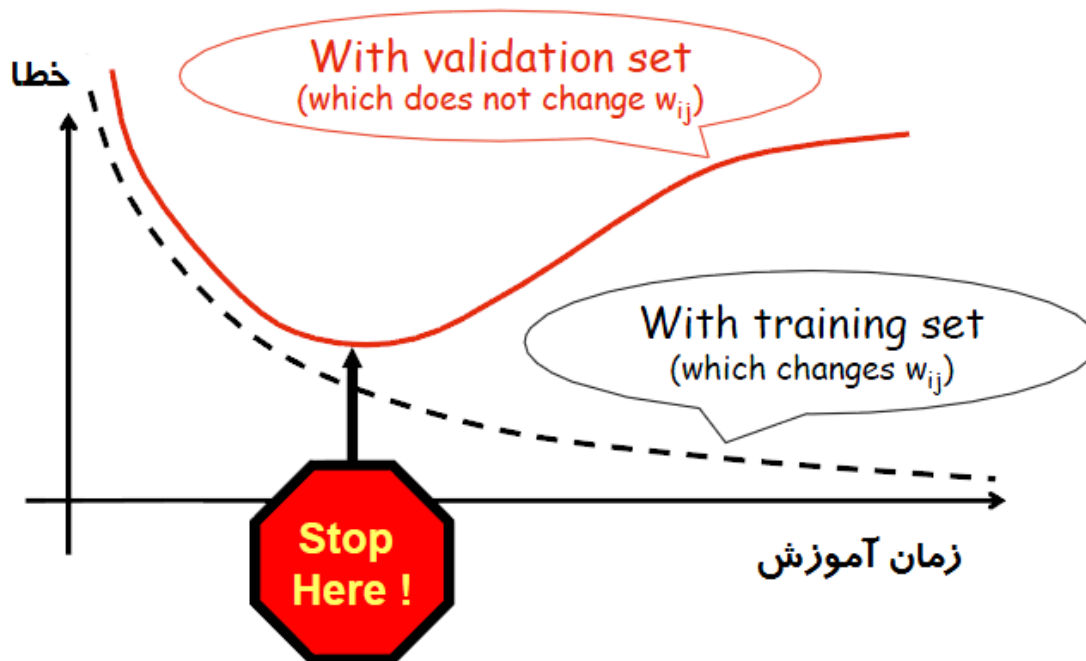
- یک مجموعه برای آموزش الگوها و یک مجموعه برای آموزش-آزمون الگوها (مجموعه validation)
- روش cross validation: تقسیم داده آموزش به K زیرمجموعه
- هر بار یکی از زیرمجموعه‌ها برای تایید اعتبار استفاده می‌شود



شبکه عصبی پرسپترون چندلایه: نکات تکمیلی...

○ نکاتی که قابلیت تعمیم را افزایش می‌دهد

- تعداد نرون‌های کمتر در لایه مخفی
- overfit نکردن: توقف شبکه با افزایش خطای مجموعه ارزیابی (تست)
- داده‌های آموزش پوششی از انواع و تنوع نمونه‌ها باشد





شبکه عصبی پرسپترون چندلایه: 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 قبل‌تر (مانند t و $t-1$)

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

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

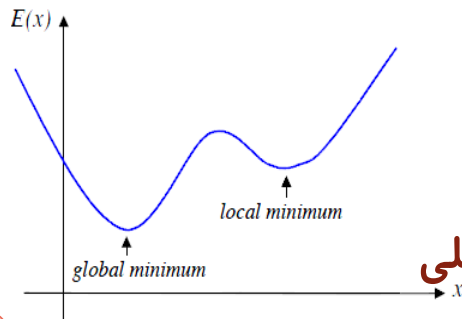
$$\Delta w_{jk}(t+1) = \alpha \delta_k z_j + \mu \Delta w_{jk}(t)$$

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

گرادیان فعلی

گرادیان قبلی

پارامتر ممان (بین ۰ تا ۱)



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



شبکه عصبی پرسپترون چندلایه: نکات تکمیلی...

○ تعداد داده‌های آموزش: قاعده 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 $1 - (e/2)$ از الگوهای آموزشی،
- می‌توان مطمئن بود که شبکه $1 - e$ الگوی آزمایش را نیز به درستی دسته‌بندی کند؟
- کافی بودن الگوهای آموزشی : $\frac{W}{P} = e$ یا $P = \frac{W}{e}$
- 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.



شبکه عصبی پرسپترون چندلایه: نکات تکمیلی...

○ به‌روز کردن دسته‌ای (Batch Updating)

- به جای به‌روز کردن وزن‌های شبکه بعد از ارائه هر الگوی آموزشی
- ادغام مقدار تصحیح (تغییر) وزن را برای چند الگو یا تمام الگوها 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



شبکه عصبی پرسپترون چندلایه: نکات تکمیلی

○ شبکه MLP تقریب‌زننده‌های جهانی است: قضیه هچ-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.

- تابع فعال‌سازی برای واحد مخفی j ام:
$$z_j = \sum_{i=1}^n \lambda^i \psi(x_i + \varepsilon j) + j$$

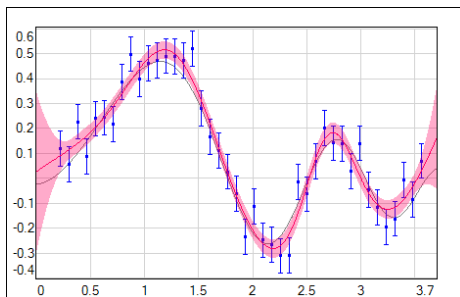
عددی ثابت و
حقیقی

تابع پیوسته، حقیقی و یکنواصعودی،
مستقل از f و وابسته به n

- مقدار ثابت ε برای فراهم بودن شرایط قضیه اسپرچر است.

- تابع فعال‌سازی برای واحدهای خروجی:
$$y_k = \sum_{j=1}^{2n+1} g_k z_j$$

تابع پیوسته، حقیقی و وابسته به f و ε



سایر شبکه‌های عصبی در پردازش زبان و گفتار...

○ شبکه‌های یادگیری عمیق (Deep NN) و شبکه‌های بازگشتی

- شبکه MLP با تعداد لایه‌های مخفی زیاد

- حافظه کوتاه-مدت ماندگار (LSTM: Long Short-Term Memory)

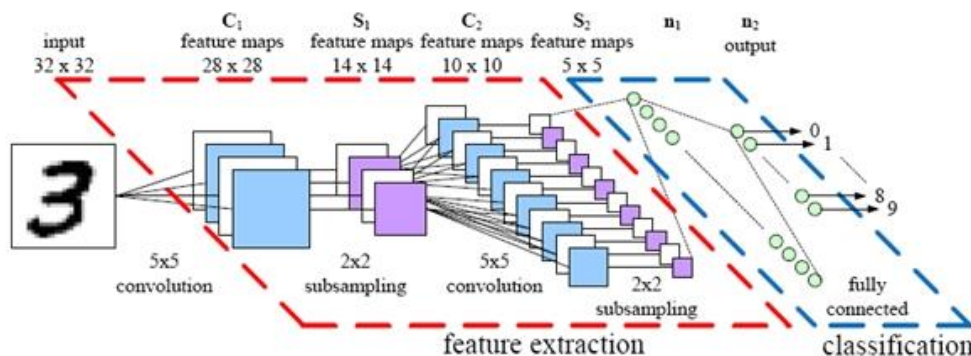
- بولتزمن محدود شده (RBN: Restricted Boltzmann Machine)

- باور عمیق (DBN: Deep Belief Network)

- رمزکننده خودکار (Auto-Encoder)

- ترکیب با روش‌های آماری (HMM)

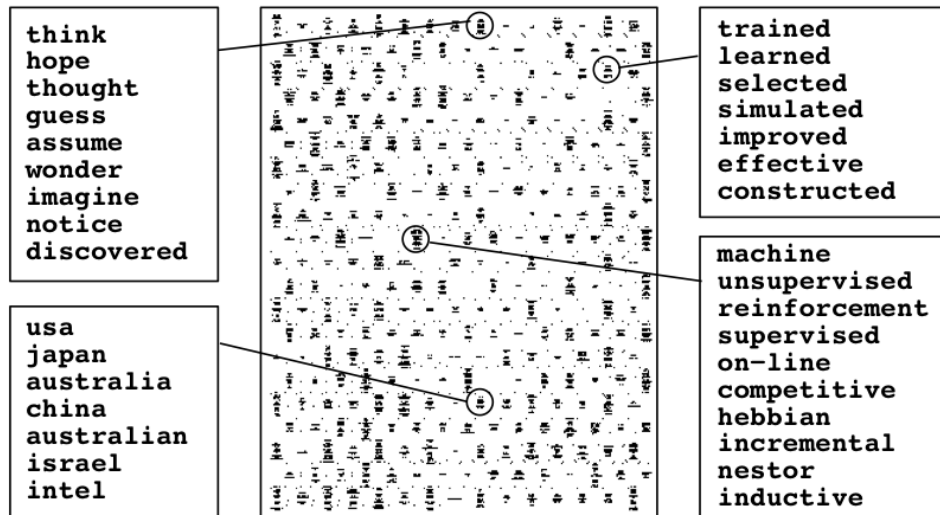
- استفاده در پردازش گفتار، تصویر و متن



سایر شبکه‌های عصبی در پردازش زبان و گفتار

○ شبکه‌های عصبی نگاشت‌های خودسازمانده کوهونن (SOM)

- کاربرد در دسته‌بندی خودکار متون
- یادگیری بدون نظارت (خوشه بندی)



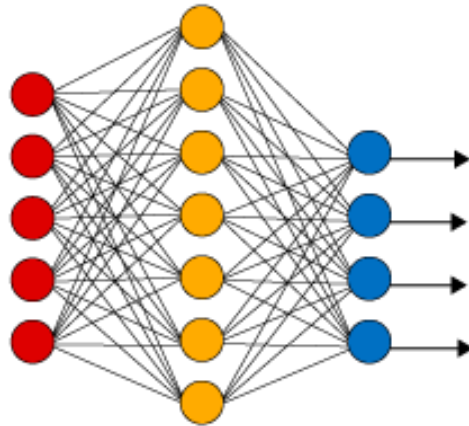
- حالت با نظارت

- یادگیری چندی‌سازی برداری (LVQ)

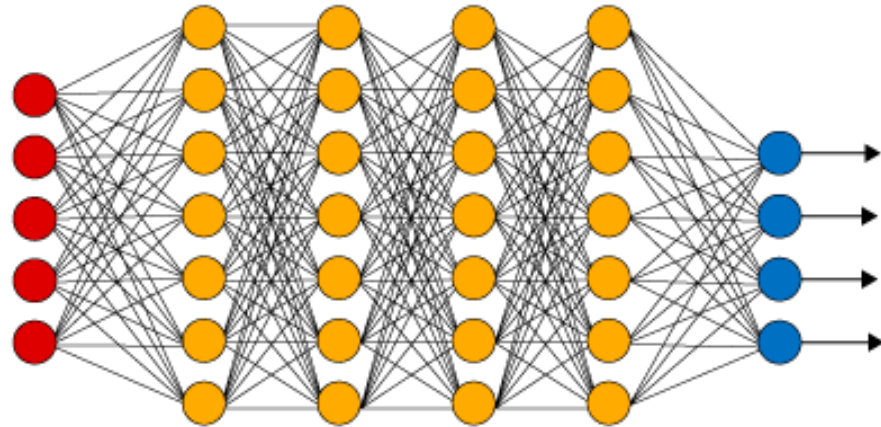
یادگیری عمیق ...

○ افزایش تعداد لایه های مخفی

Simple Neural Network



Deep Learning Neural Network



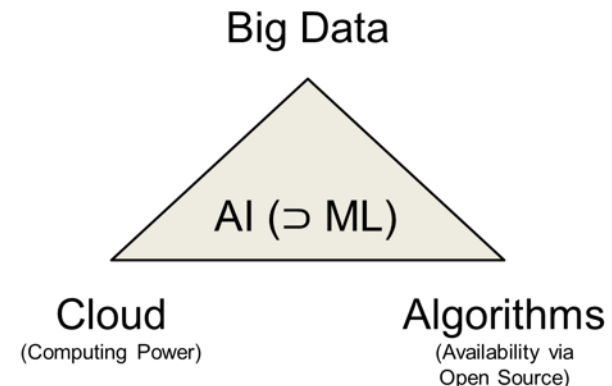
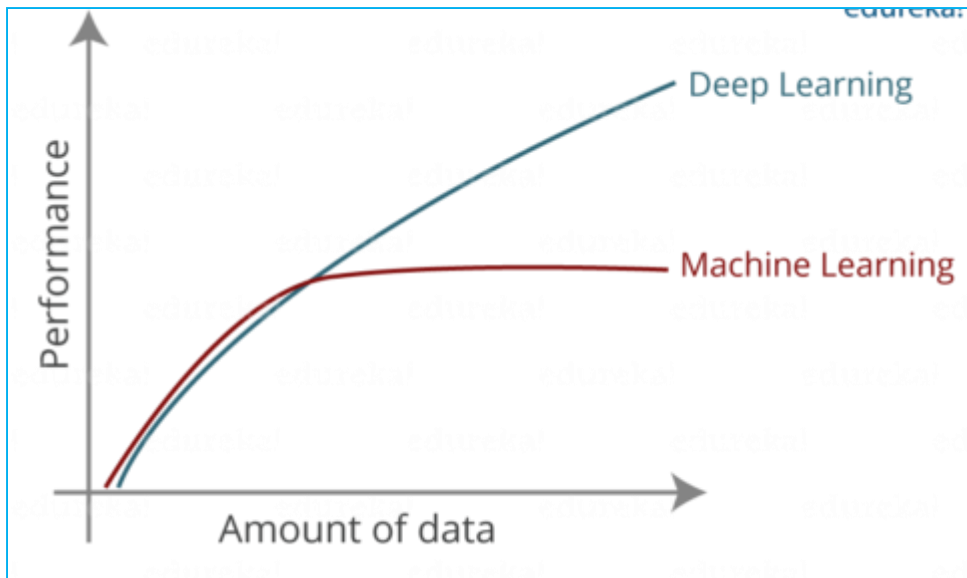
● Input Layer ● Hidden Layer ● Output Layer

• مدل پیچیده = تعداد پارامترهای زیاد = توان محاسباتی بالا

یادگیری عمیق ...

سه رکن اصلی

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

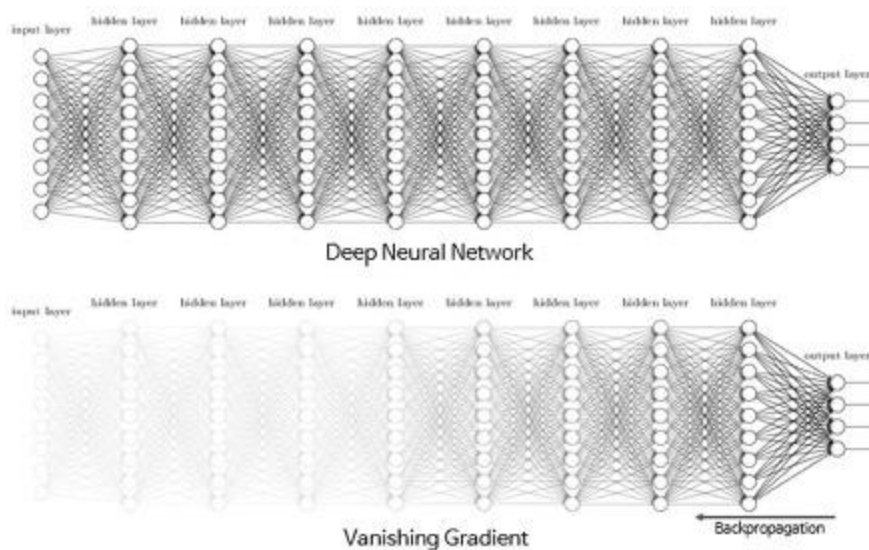


یادگیری عمیق ...

○ مشکل: یادگیری

• محو گرادیان (Vanishing Gradient)

○ کاهش گرادیان برگشتی از لایه آخر به لایه اول



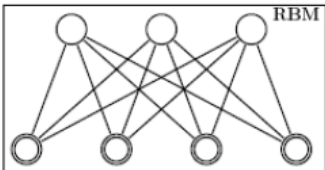
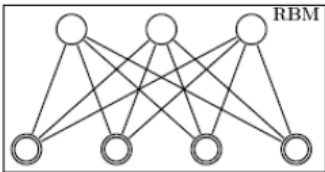
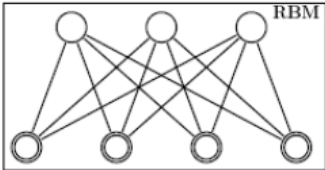
• راه حل

○ پیش آموزش شبکه برای تعیین وزن های اولیه مناسب

○ استفاده از ماشین بولتزمن محدود (RBM)

○ بهبود وزن ها با پس انتشار خطا

یادگیری عمیق: شبکه باور عمیق (DBN) ...



○ معماری

- از چندلایه ماشین بولتزمن محدود (RBM) ساخته می شود

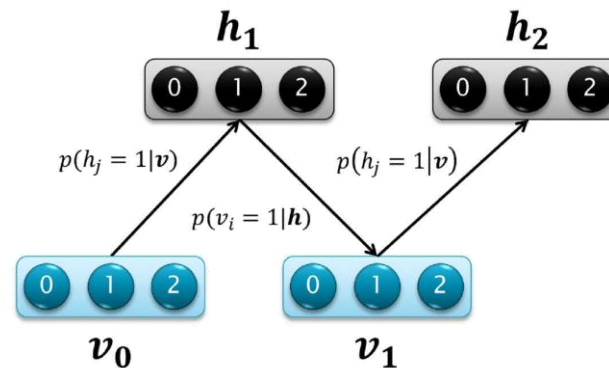
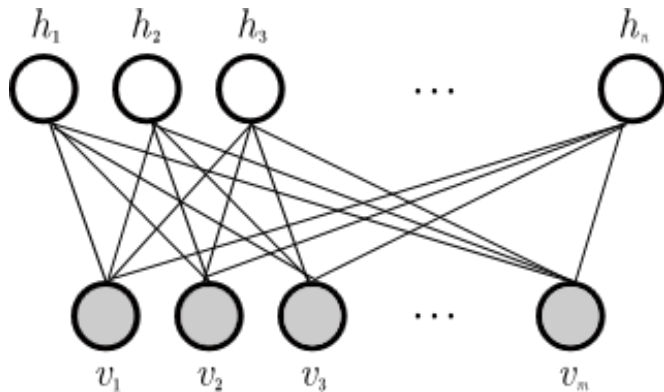
○ RBM: Restricted Boltzman Machine

○ آموزش

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

ماشین بولتزمن محدود (RBM)...

○ ساختار ماشین بولتزمن محدود (RBM)



تعداد واحدهای مشاهده پذیر

بایاس واحد مشاهده پذیر i

تعداد واحدهای پنهان

$$E(v, h) = - \sum_{i=1}^{g_v} \sum_{j=1}^{g_h} W_{ij} v_i h_j - \sum_{i=1}^{g_v} a_i v_i - \sum_{j=1}^{g_h} b_j h_j$$

بایاس واحد پنهان j



ماشین بولتزمن محدود (RBM)...

آموزش RBM

- انرژی حالت $\{v, h\}$ در ماشین بولتزمن محدود به صورت زیر است

$$E(v, h) = - \sum_{i=1}^{g_v} \sum_{j=1}^{g_h} W_{ij} v_i h_j - \sum_{i=1}^{g_v} a_i v_i - \sum_{j=1}^{g_h} b_j h_j$$

تعداد واحدهای مشاهده پذیر g_v

تعداد واحدهای پنهان g_h

بایاس واحد مشاهده پذیر i

بایاس واحد پنهان j

- شبکه به هر حالت ممکن مقادیر بردارهای مشاهده‌پذیر و مخفی با تابع انرژی، یک مقدار احتمال نسبت می‌دهد

$$P(v, h) = \frac{1}{Z} \exp(-E(v, h))$$

- انرژی کمتر = احتمال بیشتر

$$Z = \sum_v \sum_h \exp(-E(v, h))$$

ثابت نرمال‌سازی

ماشین بولتزمن محدود (RBM)...

- احتمالی که مدل به بردار قابل مشاهده (v) نسبت می‌دهد برابر است با

$$P(v) = \sum_h P(v, h) = \frac{1}{Z} \sum_h \exp(-E(v, h))$$

جمع تمام حالات ممکن بردارهای مخفی

- تابع هدف به صورت زیر تعریف می‌شود:

جهت رسیدن به انرژی کمتر برای داده‌های آموزشی و انرژی بیشتر برای سایر داده‌ها

$$\text{maximize}_{\{w_{ij}, a_i, b_j\}} \frac{1}{m} \sum_{l=1}^m \log \left(\sum_h P(\mathbf{v}^{(l)}, \mathbf{h}^{(l)}) \right)$$

تعداد نمونه های آموزشی

i امین بعد داده آموزشی l

$$\frac{\partial}{\partial w_{ij}} \left(\frac{1}{m} \sum_{l=1}^m \log \left(\sum_h P(\mathbf{v}^{(l)}, \mathbf{h}^{(l)}) \right) \right) = \frac{1}{m} \sum_{l=1}^m \sum_h x_{il} h_j P(h | v = x) - \sum_{v'} \sum_{h'} v'_i h'_j P(v', h')$$

غیر قابل محاسبه است

ماشین بولتزمن محدود (RBM)...

- با توجه به غیر قابل محاسبه بودن مشتق تابع هدف، این مقدار به صورت زیر تخمین زده می‌شود

$$\frac{\partial \log P(v)}{\partial w_{ij}} = \boxed{\langle v_i h_j \rangle_{data}} - \boxed{\langle v_i h_j \rangle_{model}}$$

فاز مثبت

فاز منفی

$\langle \rangle$: امید ریاضی بر روی ضرب مقادیر مشاهده پذیر و مخفی

- قانون اصلاح وزن‌ها به صورت زیر محاسبه می‌گردد

$$\Delta w_{ij} = \epsilon (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

$$\Delta a_i = \epsilon (\langle v_i \rangle_{data} - \langle v_i \rangle_{model})$$

$$\Delta b_j = \epsilon (\langle h_j \rangle_{data} - \langle h_j \rangle_{model})$$

ماشین بولتزمن محدود (RBM)...

محاسبه $\langle v_i h_j \rangle_{data}$

- حالت دودویی هر واحد مخفی با شرط داشتن واحدهای مشاهده پذیر به احتمال زیر یک می شود. در صورتیکه این مقدار از یک عدد تصادفی در بازه $[0-1]$ بزرگتر بود برابر یک و در غیر این صورت برابر صفر است.

تابع سیگموئید

$$P(h_j = 1 | \mathbf{v}) = \varphi \left(b_j + \sum_i v_i w_{ij} \right)$$

به دلیل نبود اتصال بین واحدهای مخفی این واحدها به شرط واحد مشاهده پذیر مستقل هستند

- حالت دودویی هر واحد مخفی با شرط داشتن واحدهای مشاهده پذیر به احتمال زیر یک می شود. در صورتیکه این مقدار از یک عدد تصادفی در بازه $[0-1]$ بزرگتر بود برابر یک و در غیر این صورت برابر صفر است.

$$P(v_i = 1 | \mathbf{h}) = \varphi \left(a_i + \sum_j h_j w_{ij} \right)$$

به دلیل نبود اتصال بین واحدهای مشاهده پذیر این واحدها به شرط واحد پنهان مستقل هستند

- با ضرب این دو مقدار $\langle v_i h_j \rangle_{data}$ محاسبه می گردد.

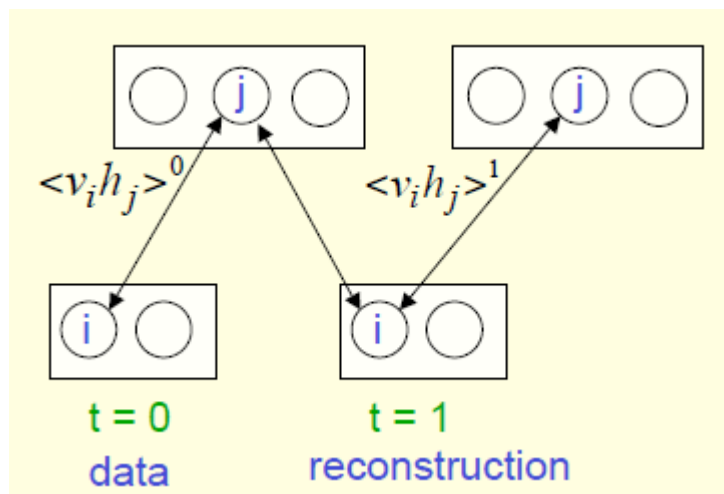
ماشین بولتزمن محدود (RBM)...

روش واگرایی متقابل (Contrastive Divergence)

• اجرای نمونه برداری گیبز تا گام زمانی $t=1$

- قرار دادن یک بردار از ورودی در لایه مشاهده پذیر
- بروز رسانی تمام واحدهای مخفی به صورت موازی
- بروز رسانی تمام واحدهای مشاهده پذیر به صورت موازی
- دوباره بروز رسانی واحدهای مخفی
- و در نهایت اصلاح وزن‌ها:

$$\frac{\partial \log p(v)}{\partial W_{ij}} = \langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^1$$



$$\Delta w_{ij} = \alpha (\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^1)$$

$$\Delta a_i = \alpha (\langle v_i \rangle^0 - \langle v_i \rangle^1)$$

$$\Delta b_j = \alpha (\langle h_j \rangle^0 - \langle h_j \rangle^1)$$



ماشین بولتزمن محدود (RBM)...

○ الگوریتم آموزش واگرایی متقابل

- گام ۰ - مقداردهی اولیه وزن‌ها به صورت تصادفی و صفر قرار دادن مقادیر بایاس
- گام ۱ - برای هر بردار ورودی گام‌های ۲ تا ۹ را تکرار کنید
- گام ۲ - یک بردار ورودی (v) را در لایه مشاهده پذیر قرار دهید.
- گام ۳ - احتمال فعال شدن واحدهای پنهان را به شرط بردار v محاسبه نمایید

تابع سیگموید

$$P(h_j = 1|v) = \sigma(b_j + \sum_{i=1}^m w_{ij} v_i)$$

$positive_{hidden}$

- گام ۴ - مقدار واحدهای پنهان را در صورتی که احتمال به دست آمده بزرگتر از یک مقدار تصادفی در بازه $[0,1]$ بود برابر یک و در غیر اینصورت برابر صفر قرار دهید (بردار h)
- گام ۵ - احتمال فعال شدن واحدهای مشاهده‌پذیر را به شرط بردار h محاسبه نمایید

$$P(v_i = 1|h) = \sigma(a_i + \sum_{j=1}^n w_{ij} h_j)$$



ماشین بولتزمن محدود (RBM)...

○ الگوریتم آموزش (ادامه...)

- گام ۶- در صورتی که احتمال به دست آمده بزرگتر از یک مقدار تصادفی در بازه $[0,1]$ بود، مقدار واحدهای مشاهده‌پذیر را برابر یک و در غیر اینصورت برابر صفر قرار دهید. (بردار v')

- گام ۷- احتمال فعال شدن واحدهای پنهان را به شرط بردار v' محاسبه نمایید.

$$P(h'_j = 1 | v') = \sigma(b_j + \sum_{i=1}^m w_{ij} v'_i)$$

negative_{hidden}

- گام ۸- اگر نرخ یادگیری برابر α باشد تغییرات وزن را به کمک رابطه‌های زیر محاسبه کنید.

$$\Delta W = \alpha (v^T * \text{Positive}_{\text{Hidden}} - v'^T * \text{Negative}_{\text{Hidden}})$$

$$\Delta a = \alpha (v - v')$$

$$\Delta b = \alpha (\text{Positive}_{\text{Hidden}} - \text{Negative}_{\text{Hidden}})$$



ماشین بولتزمن محدود (RBM)...

○ الگوریتم آموزش (ادامه...)

• گام ۹- وزن‌ها را بروز رسانی کنید.

$$W = W + \Delta W$$

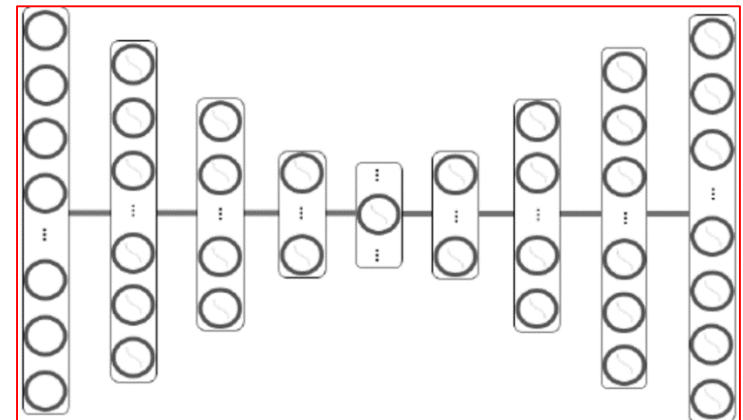
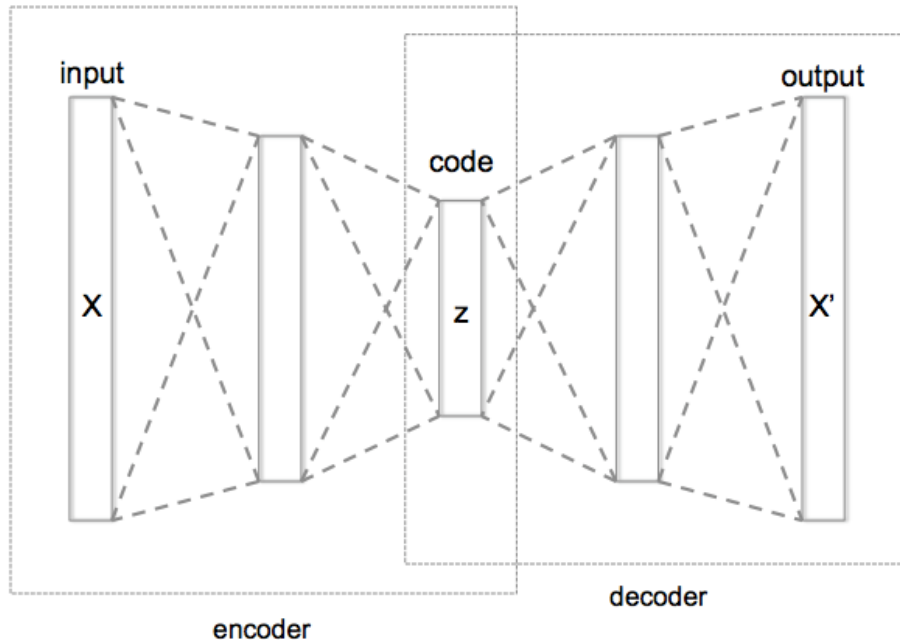
$$b = b + \Delta b$$

$$a = a + \Delta a$$

شبکه‌های خودرمزگذار بازگشتی (RAE)

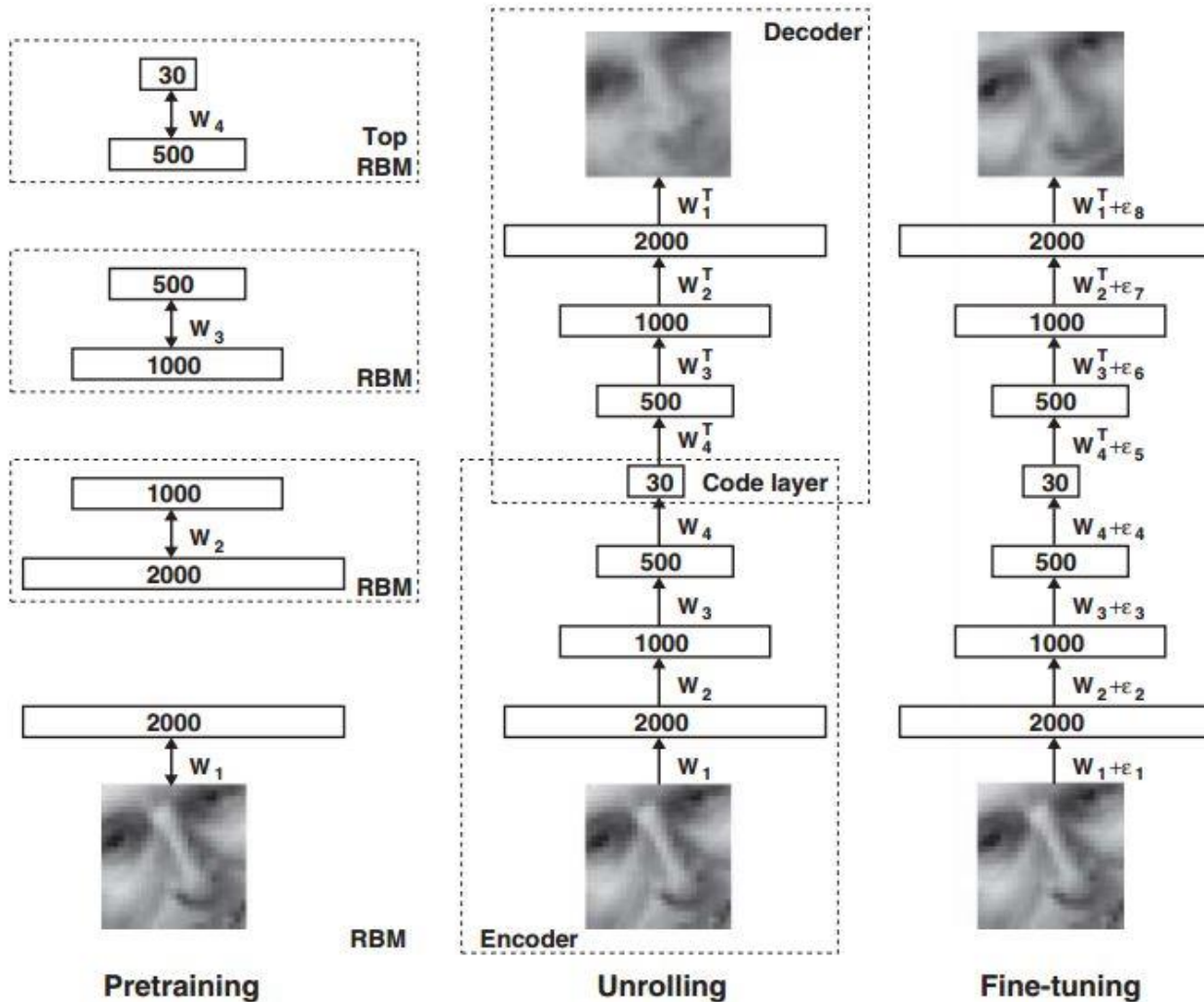
○ شبکه‌های خودرمزگذار بازگشتی (RAE: Recursive AutoEncoders)

- استفاده از لایه وسط به عنوان ویژگی

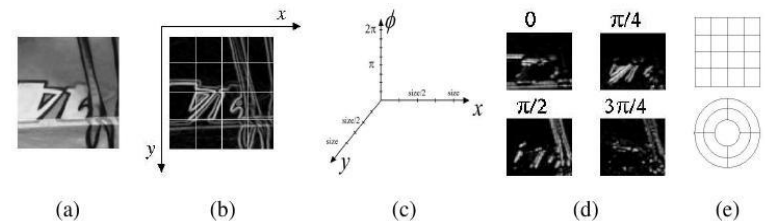
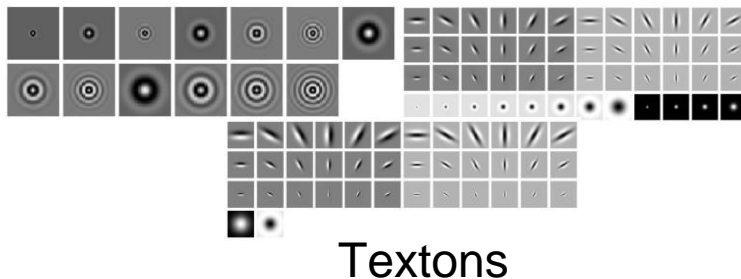
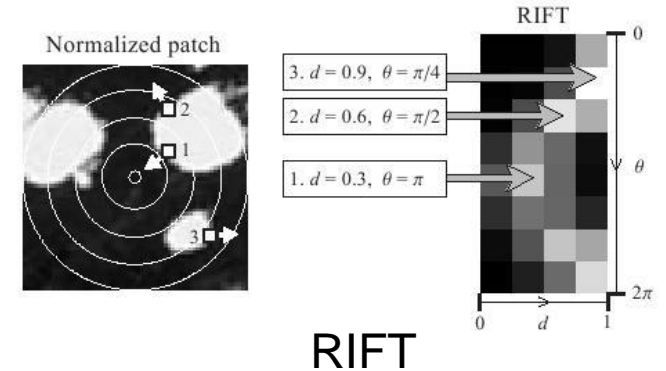
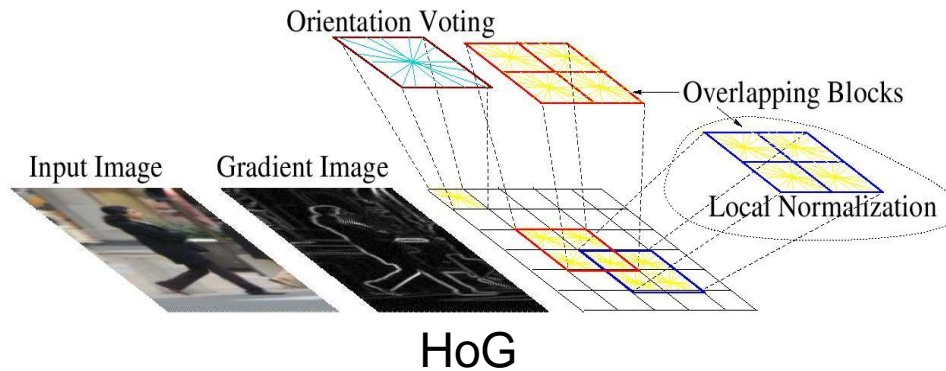
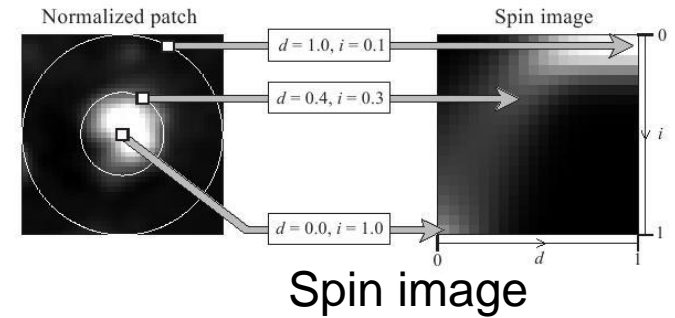
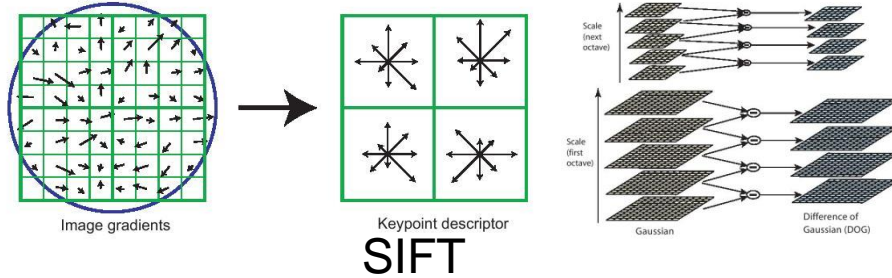


یادگیری عمیق ...

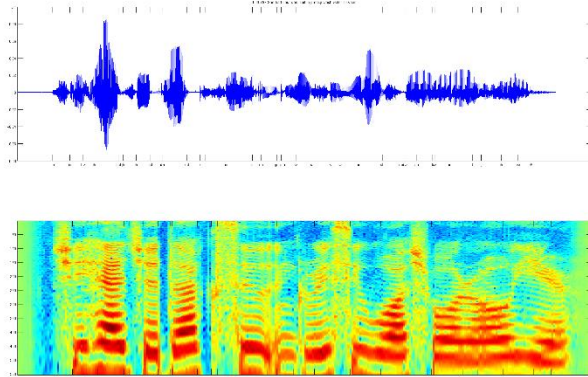
○ خودرمزگذار



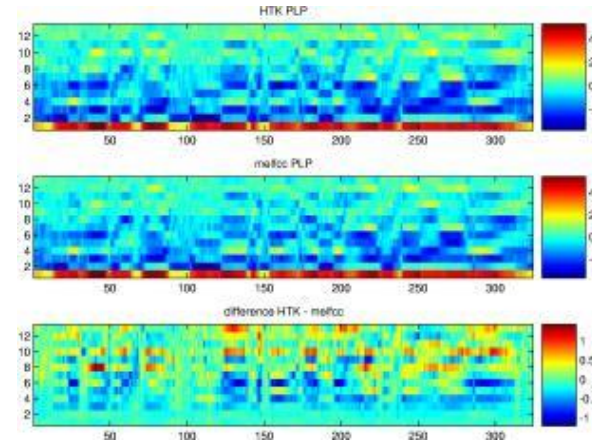
یادگیری عمیق: استخراج ویژگی (تصویر)



پادگیری عمیق: استخراج ویژگی (صدا)



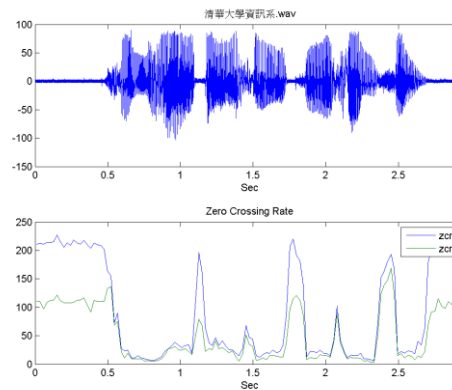
Spectrogram



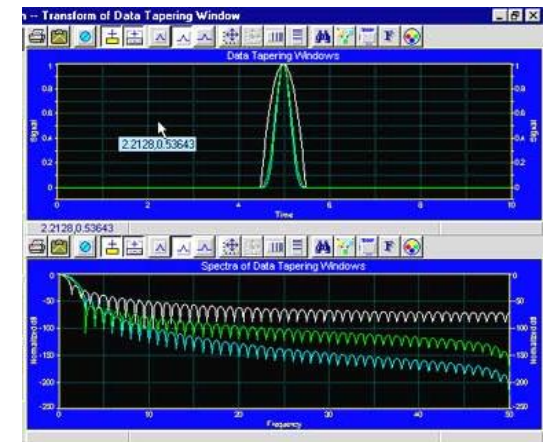
MFCC



Flux



ZCR



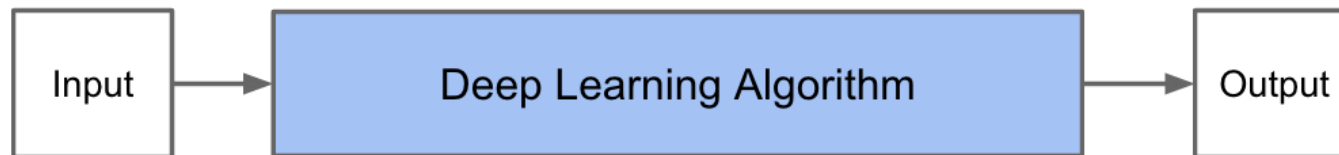
Rolloff

یادگیری عمیق ...

ترکیب استخراج ویژگی و دسته‌بندی

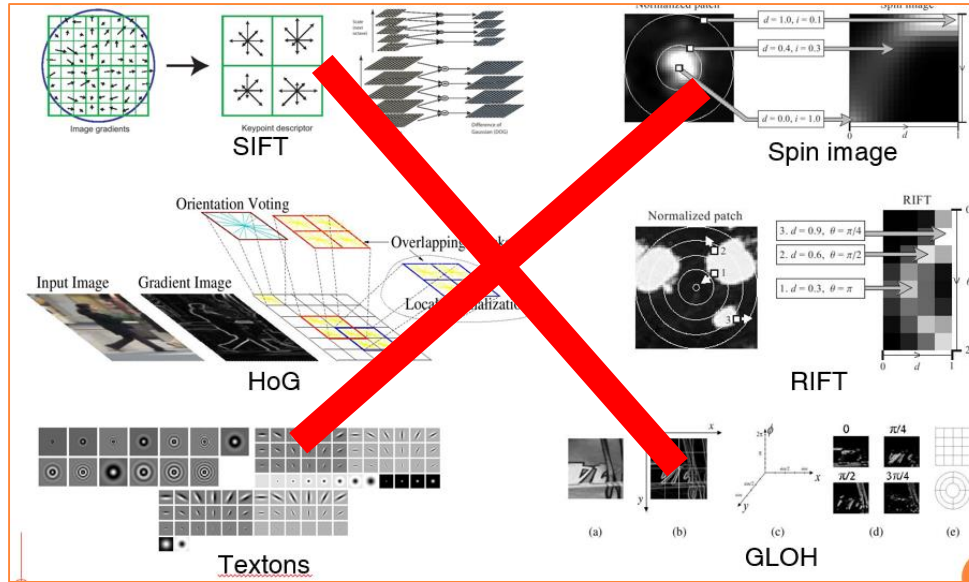


Traditional Machine Learning Flow



Deep Learning Flow

یادگیری عمیق: استخراج ویژگی و دسته‌بندی



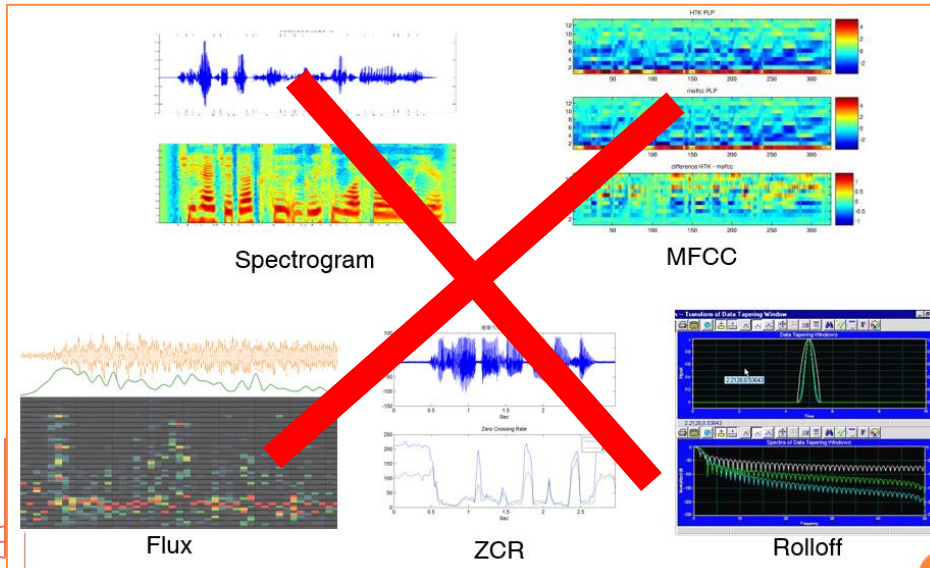
Unlabeled images



Learning algorithm

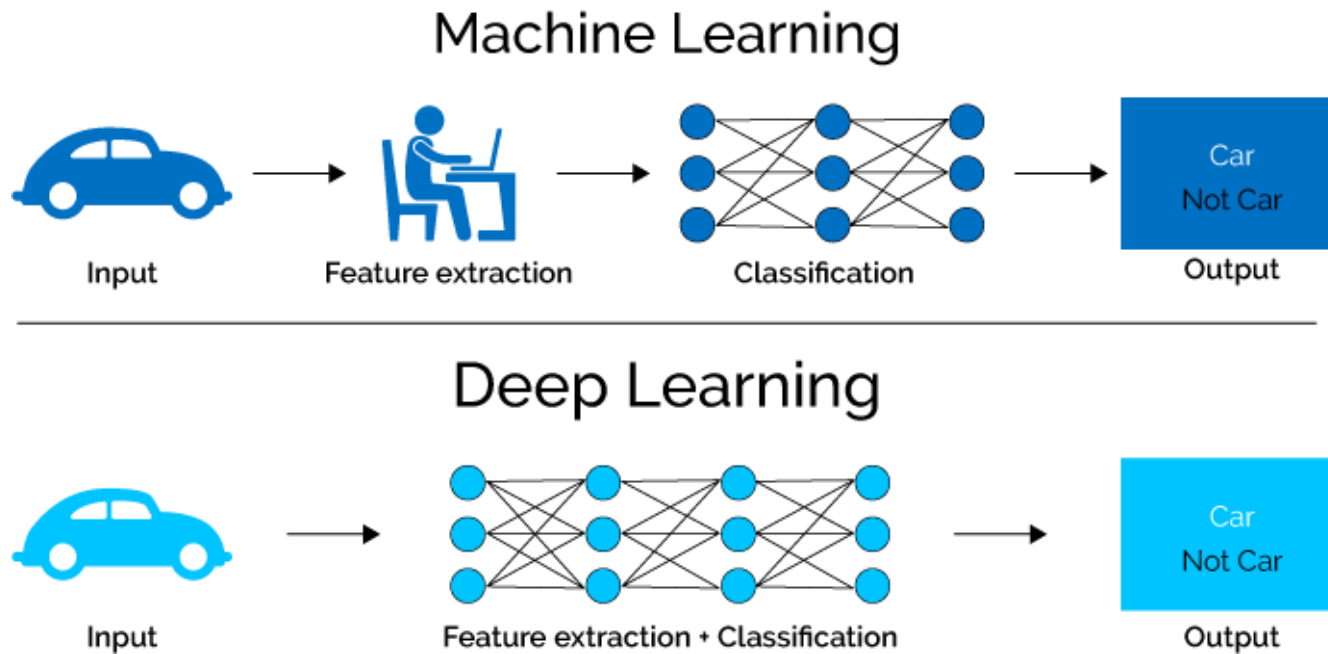


Feature representation



یادگیری عمیق ...

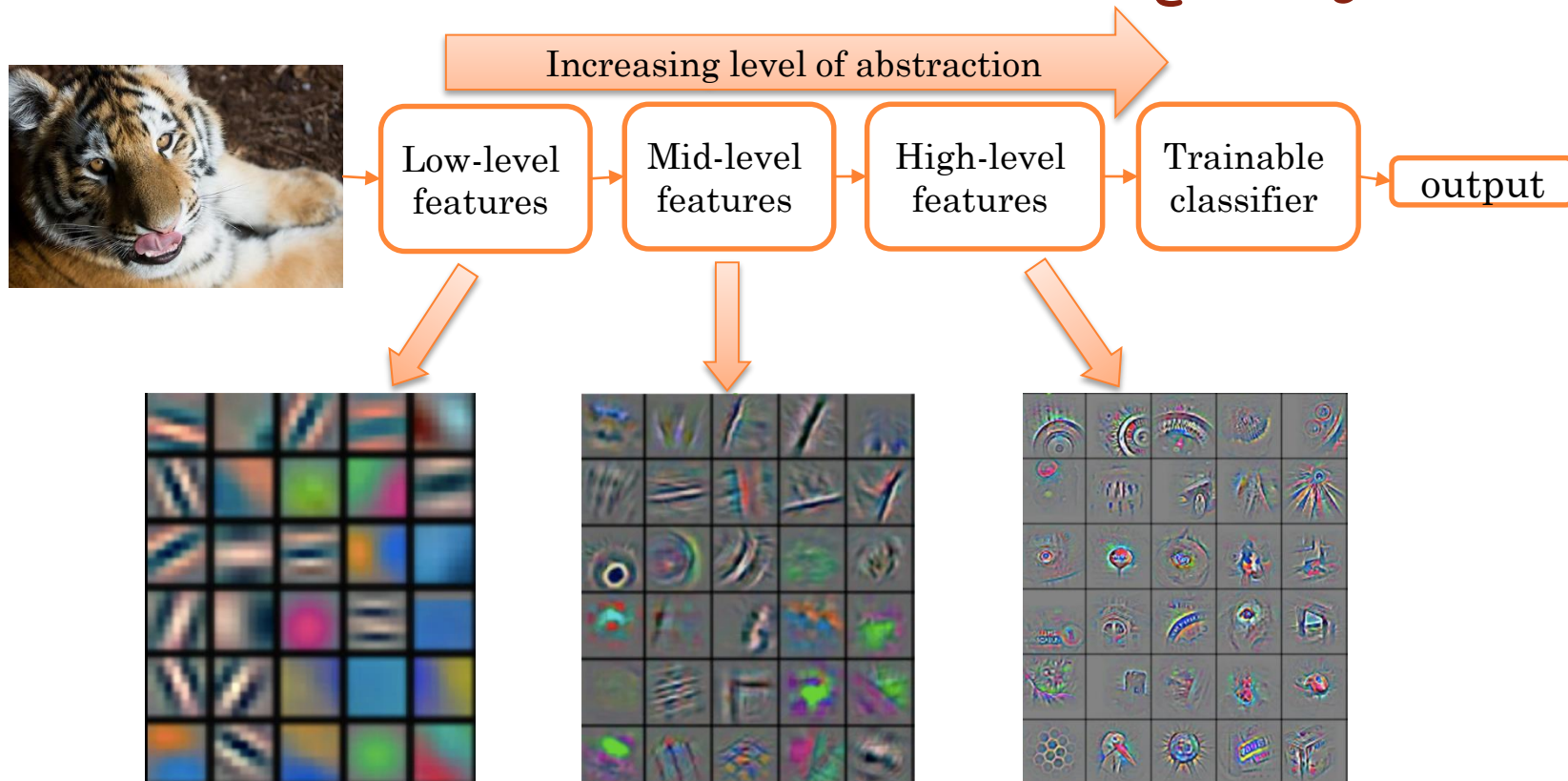
ترکیب استخراج ویژگی و دسته‌بندی



یادگیری عمیق: استخراج ویژگی ...

○ استخراج ویژگی سلسله مراتبی ...

- از ورودی خام (جزئیات)
- تا ویژگی‌های سطح بالا (کلیات)





یادگیری عمیق: استخراج ویژگی

○ استخراج ویژگی سلسله مراتبی

- تصویر

Pixel → edge → texon → motif → part → object ○

- متن

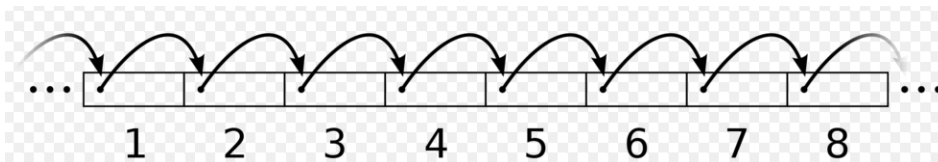
Character → word → word group → clause → sentence → story ○



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 ...

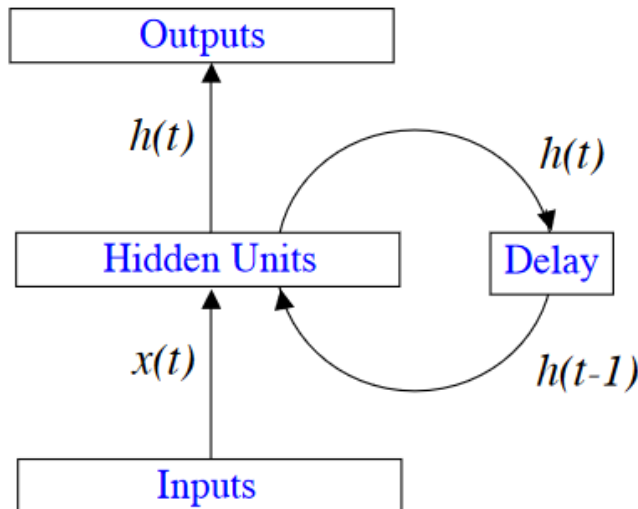
○ شبکه‌های عصبی بازگشتی (RNN: Recurrent Neural Networks)

- 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 ...

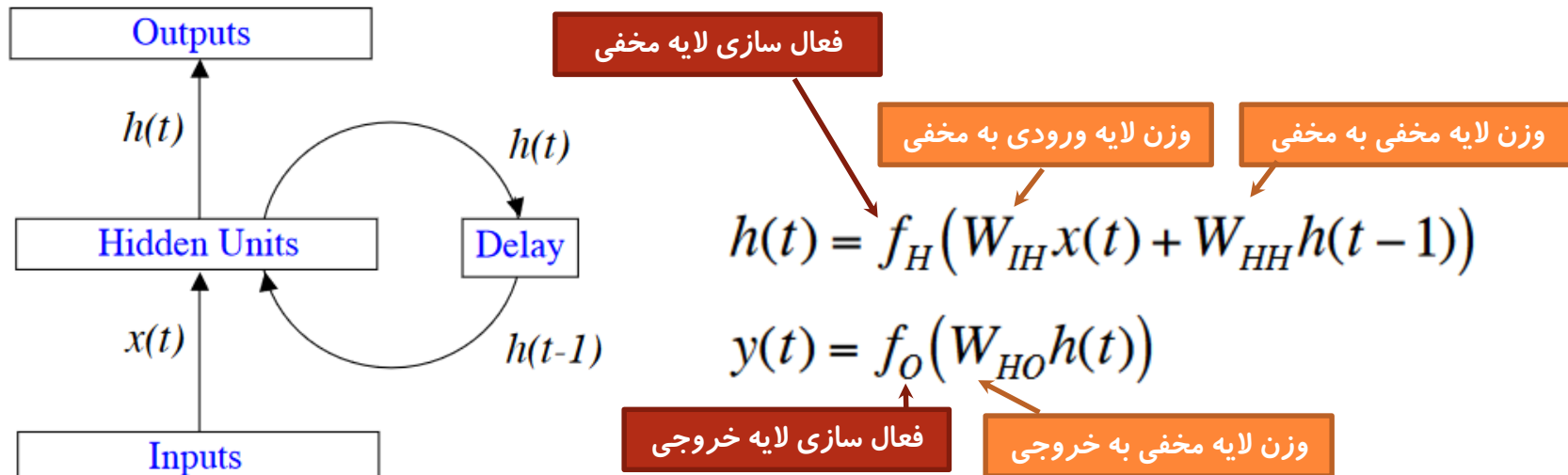
○ شبکه عصبی بازگشتی به عنوان یک سیستم پویا (Dynamic System)

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

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

○ مرتبه سیستم پویا = ابعاد فضای حالت

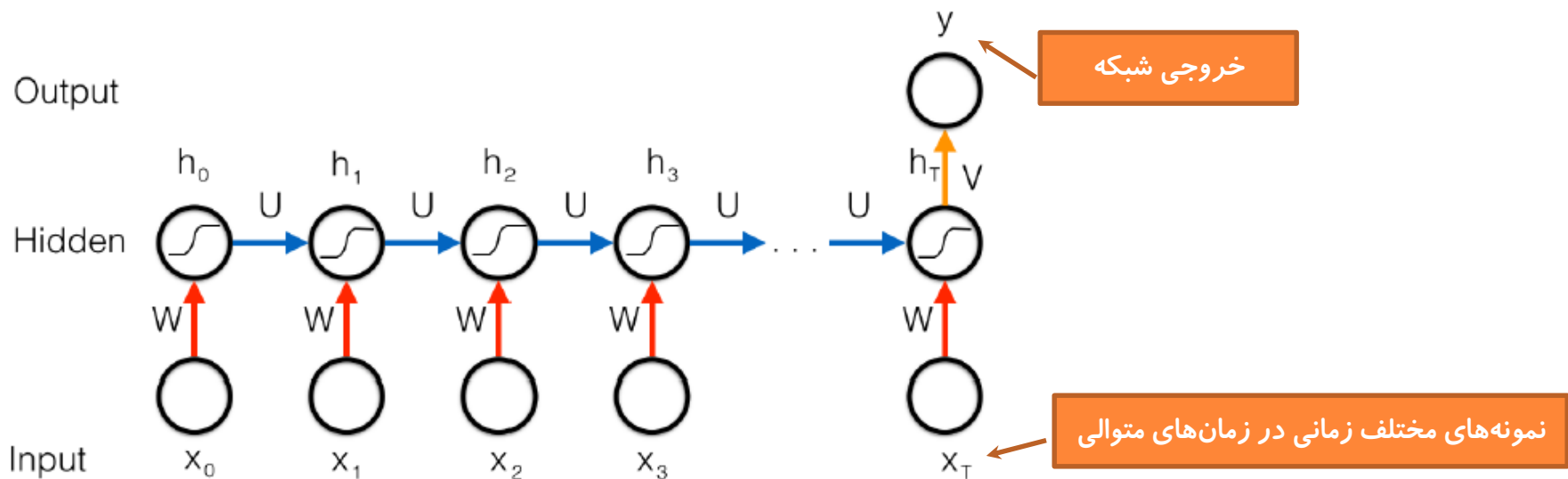
○ در اینجا = تعداد نرون‌های لایه مخفی



شبکه عصبی بازگشتی: آموزش ...

عملکرد شبکه

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



$$f(x) = Vh_T$$

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

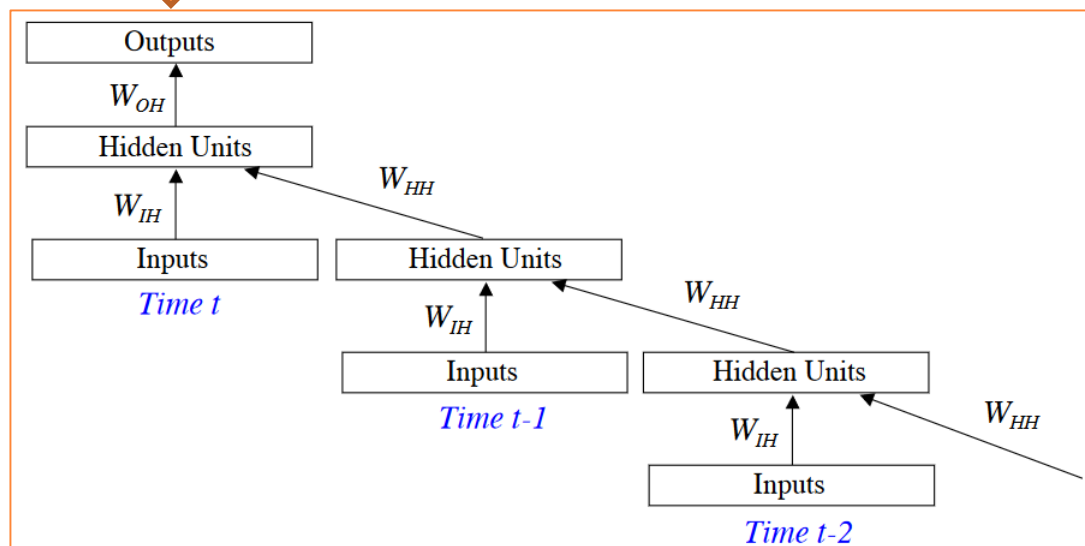
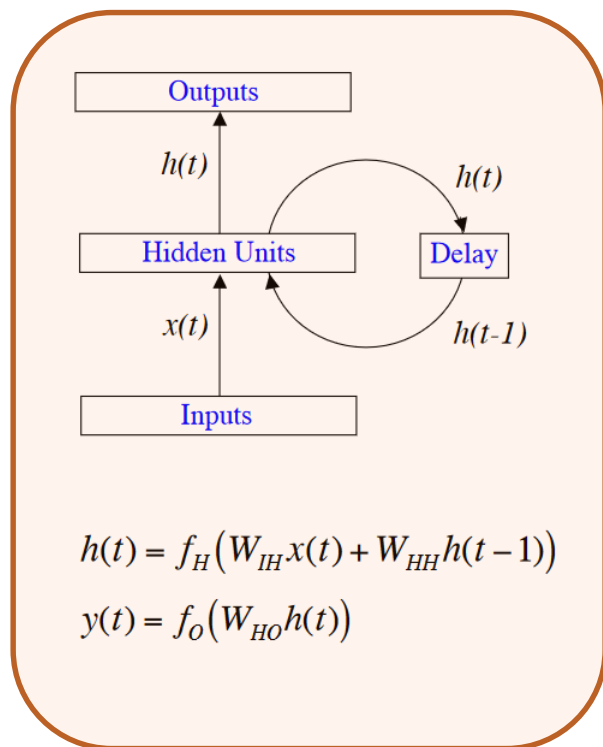
...

$$h_0 = \sigma(Wx_0)$$

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 پس‌انتشار ...

- 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

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

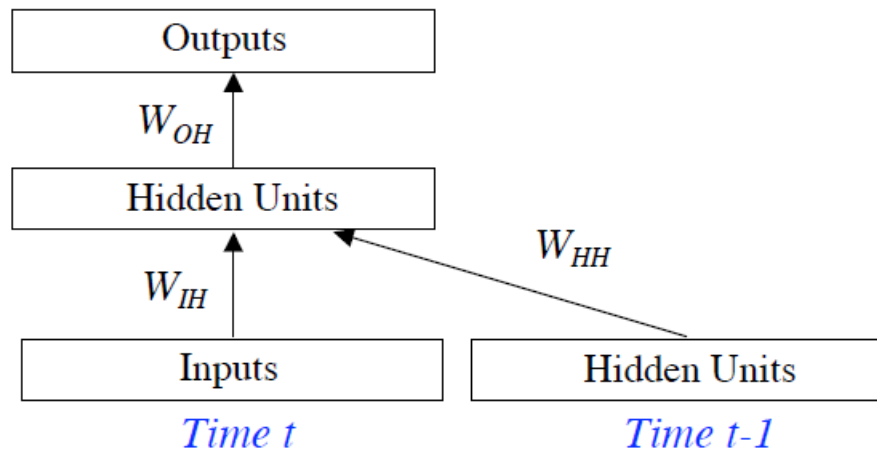
$$w_{ij} \in \{W_{IH}, W_{HH}\}$$



شبکه عصبی بازگشتی: آموزش ...

○ استفاده از الگوریتم پس‌انتشار

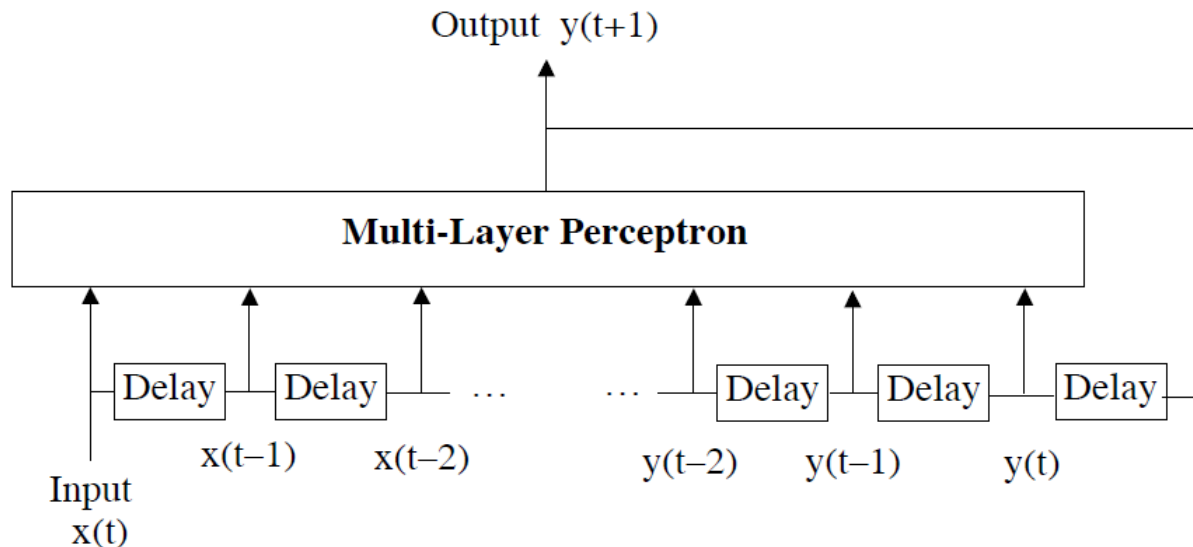
- استفاده از این روش نیازمند نگهداری حالت‌های قبلی شبکه (خروجی لایه مخفی) و کلیه ورودی‌های قبلی
- در عمل نگهداری همه اطلاعات قبلی مشکل است و تنها از تعداد محدودی از آنها (مثلاً ۳۰ مقدار قبلی) استفاده می‌شود = truncation
- حالت ساده = نگهداری فقط یک مرحله قبل = شبکه المان (Elman Network)



شبکه عصبی بازگشتی: آموزش ...

○ مدل NARX: Non-linear Auto-Regressive with eXogeneous inputs

- یک شبکه MLP با یک ورودی و یک خروجی
- ورودی‌ها و خروجی‌ها در زمان‌های مختلف تاخیر داده می‌شوند



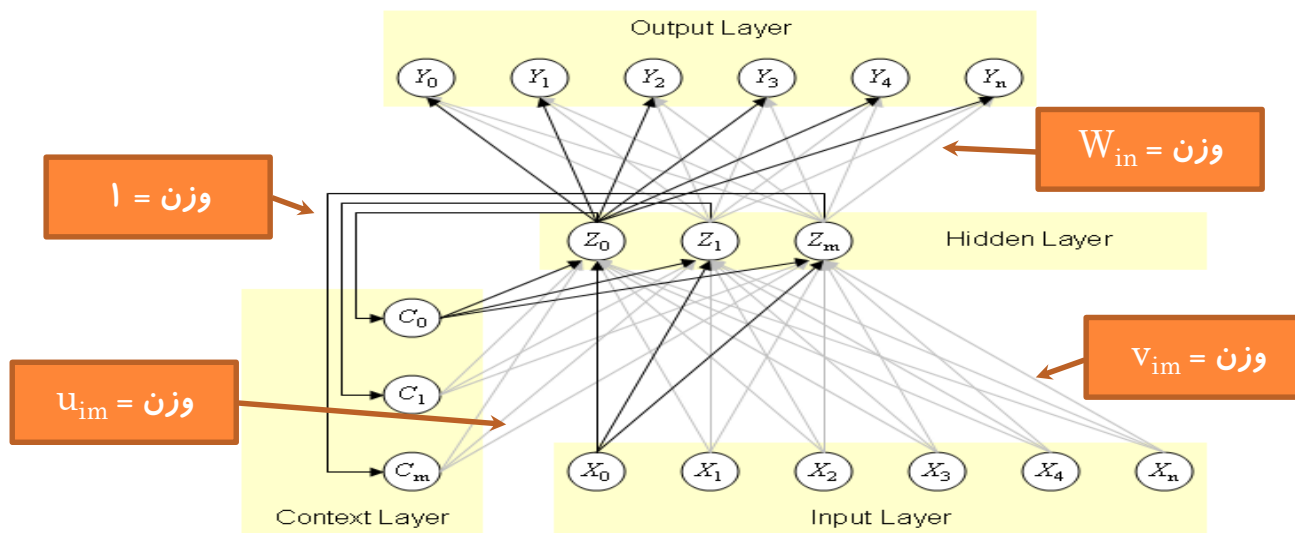
- این ساختار برای پیش‌بینی سری‌های زمانی مناسب است

شبکه عصبی المان ...

○ 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
- وزن‌یال‌هایی که لایه پنهان را به لایه بافت متصل می‌کنند برابر مقدار ثابت یک می‌باشد





شبکه عصبی المان: آموزش ...

○ به ابتدا و انتهای هر دنباله، نماد آغازین و نماد پایانی اضافه کنید

○ الگوریتم آموزش همان پس انتشار استاندارد است

- مرحله ۰: برای هر دنباله آموزش $X = (x_1, x_i, \dots, x_k)$ که x_1 و x_k به ترتیب نمادهای مربوط به شروع و خاتمه دنباله می‌باشد، مراحل ۱ تا ۱۴ را انجام دهید
- مرحله ۱: مقدار فعال‌ساز تمامی نرون‌های لایه بافت (C_i) را برابر ۰.۵ قرار دهید.
- مرحله ۲: مراحل ۳ تا ۱۴ را انجام دهید تا به انتهای دنباله برسید
- مرحله ۳: بردار x_i را به شبکه ارائه دهید.
- مرحله ۴: بردار x_{i+1} در دنباله را به عنوان پاسخ هدف به واحدهای خروجی ارائه دهید.
- مرحله ۵: مقدار فعال‌ساز نرون‌های لایه مخفی را محاسبه کنید

$$z_{in_q} = \sum_{i=1}^n x_i v_{iq} + \sum_{i=1}^m C_i u_{iq} \quad , \quad Z_q = f(z_{in_q})$$



شبکه عصبی المان: آموزش ...

- مرحله ۶: مقدار فعال‌ساز نرون‌های لایه خروجی را محاسبه کنید

$$y_{in_j} = \sum_{i=1}^m Z_i w_{ij} \quad , \quad y_j = f(y_{in_j})$$

- مرحله ۷: دلتای نرون‌های لایه خروجی را محاسبه کنید

$$\delta_j = (t_j - y_j) f'(y_{in_j})$$

- مرحله ۸: تغییرات وزن نرون‌های لایه پنهان به لایه خروجی را محاسبه کنید

$$\Delta W_{ij} = \alpha \delta_j Z_i$$

- مرحله ۹: خطای ورودی به نرون‌های لایه مخفی را محاسبه کنید (پس انتشار خطا)

$$\delta_{in_q} = \sum_{i=1}^n \delta_i w_{qi}$$

- مرحله ۱۰: دلتای نرون‌های لایه پنهان را محاسبه کنید

$$\delta_q = \delta_{in_q} f'(z_{in_q})$$

- مرحله ۱۱: تغییرات وزن نرون‌های لایه ورودی به لایه پنهان را محاسبه کنید

$$\Delta V_{iq} = \alpha x_i \delta_q$$



شبکه عصبی المان: آموزش

- مرحله ۱۲: تغییرات وزن نرون‌های لایه بافت به لایه پنهان را محاسبه کنید

$$\Delta U_{iq} = \alpha C_i \delta_q$$

- مرحله ۱۳: کلیه وزن‌ها را بروز رسانی کنید

- مرحله ۱۴: شرط توقف را بررسی کنید:

- اگر هدف نماد پایانی باشد، الگوریتم را خاتمه دهید، در غیر این صورت فعال‌ساز نرون‌های لایه مخفی را در نرون‌های لایه بافت کپی کنید.



شبکه عصبی المان: الگوریتم کاربرد

○ الگوریتم کاربرد (آزمون)

○ مرحله ۰: برای هر دنباله آزمون $X = (x_1, x_i, \dots, x_k)$ مراحل ۱ تا ۱۴ را انجام دهید

○ مرحله ۱: مقدار فعال‌ساز تمامی نرون‌های لایه بافت را برابر ۰.۵ قرار دهید.

○ مرحله ۲: برای تمامی بردارهای x_i مراحل ۳ تا ۶ را انجام دهید

○ مرحله ۳: بردار x_i را به شبکه ارائه دهید.

○ مرحله ۴: مقدار فعال‌ساز نرون‌های لایه مخفی را محاسبه کنید

$$z_in_m = \sum_{i=1}^n x_i v_{i m} + \sum_{i=1}^m C_i u_{i m} \quad , \quad Z_m = f(z_in_m)$$

○ مرحله ۵: مقدار فعال‌ساز نرون‌های لایه خروجی را محاسبه کنید

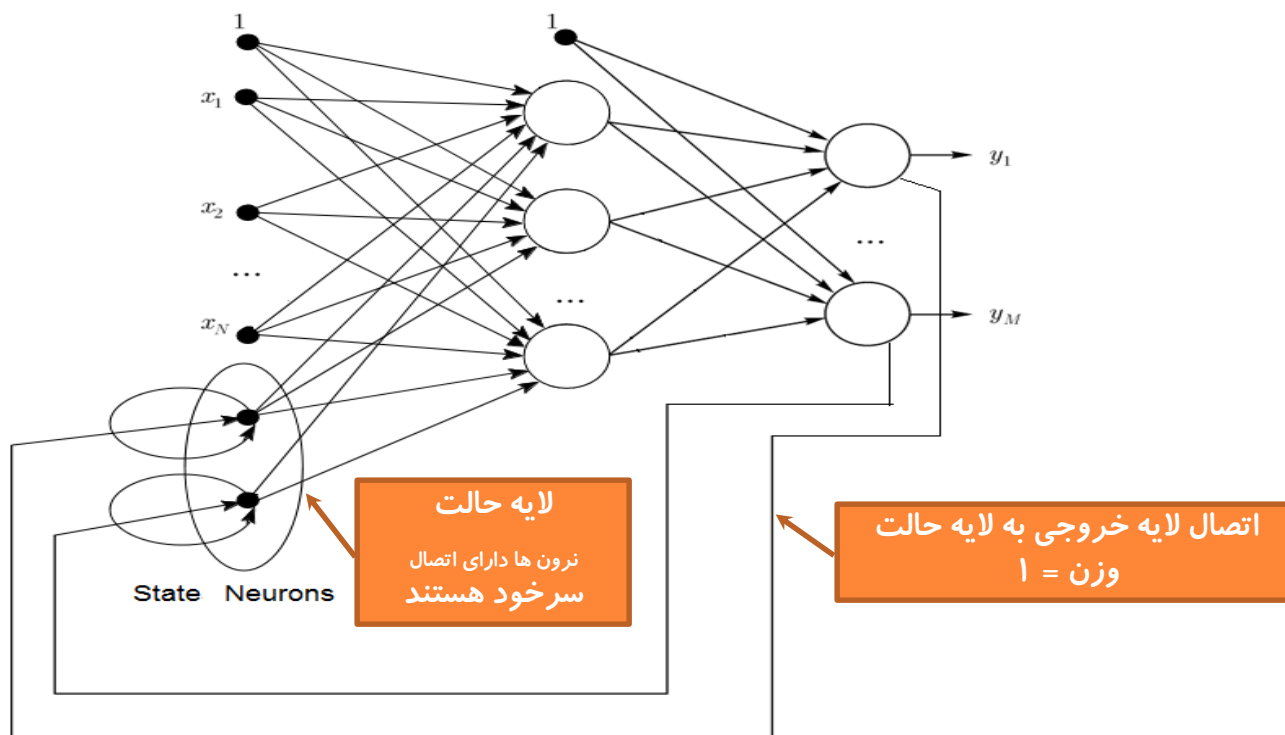
$$y_in_n = \sum_{i=1}^m Z_i w_{i n} \quad , \quad Y_n = f(y_in_n)$$

• مرحله ۶: نماد با بیشترین مقدار فعال‌ساز را به عنوان خروجی شبکه اعلام کنید

شبکه عصبی جردن ...

معرفی و ساختار

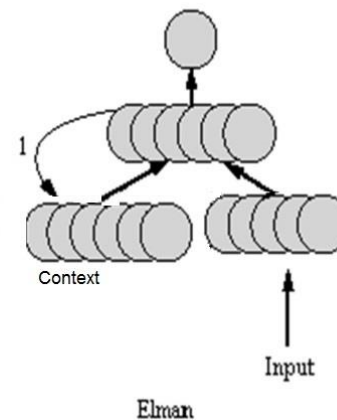
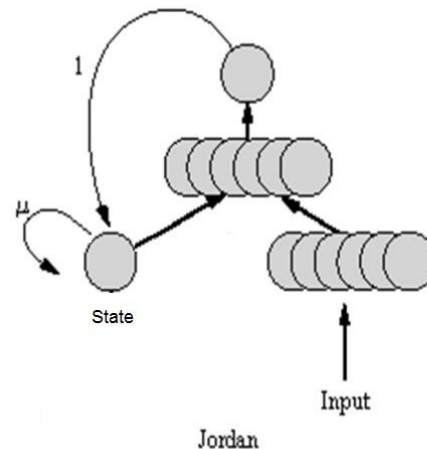
- ارائه شده در سال ۱۹۹۶ توسط مایکل جردن
- دارای شباهت بسیار زیاد به شبکه عصبی المان
- شبکه دارای اتصالات بازگشتی از لایه خروجی به لایه حالت و همچنین از لایه حالت به خودش می‌باشد



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.



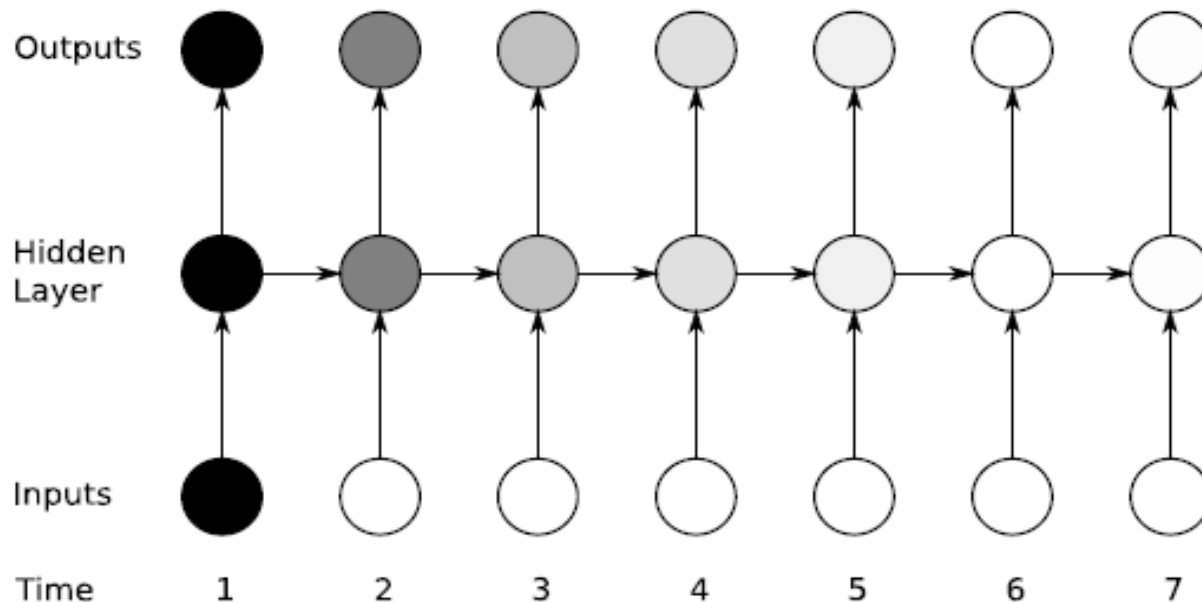


مشکل فراموشی در شبکه‌های عصبی بازگشتی ...

○ مشکل

- با طولانی شدن دنباله ورودی، شبکه عصبی بازگشتی به مرور داده‌های اولیه را فراموش می‌کند که به آن مشکل فراموشی گفته می‌شود

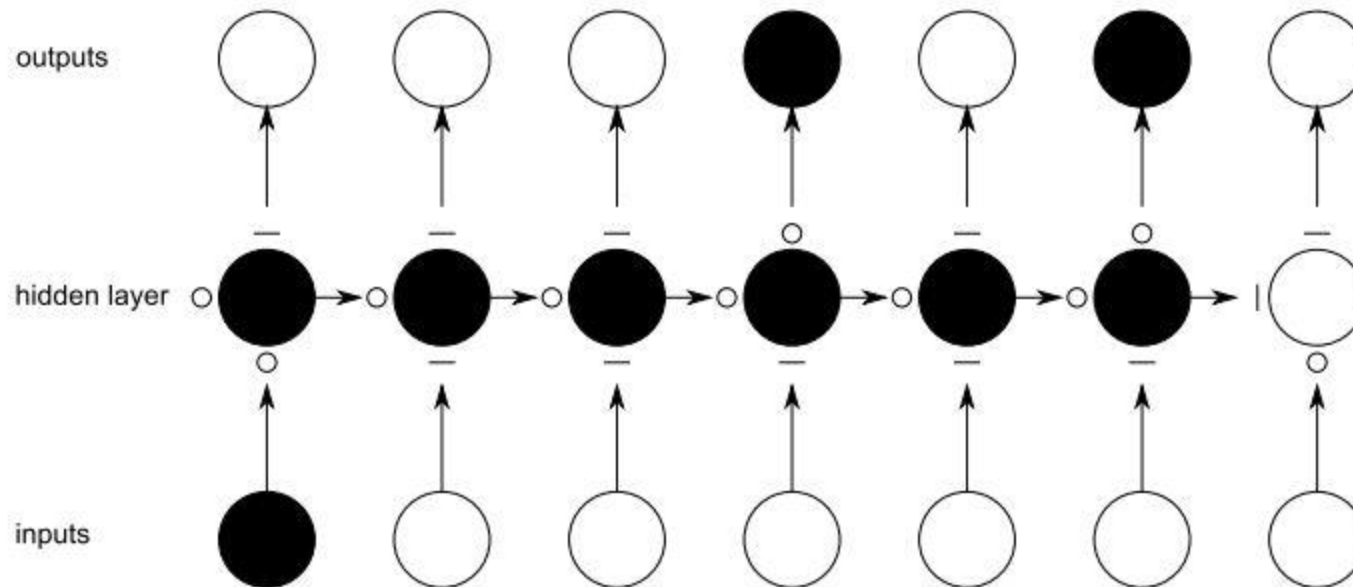
○ سایه‌های پررنگ‌تر به معنای تاثیر بیشتر بر لایه پنهان و خروجی می‌باشد



مشکل فراموشی در شبکه‌های عصبی بازگشتی

○ حل مشکل فراموشی

- کنترل ورود و خروج داده در واحدهای لایه میانی شبکه

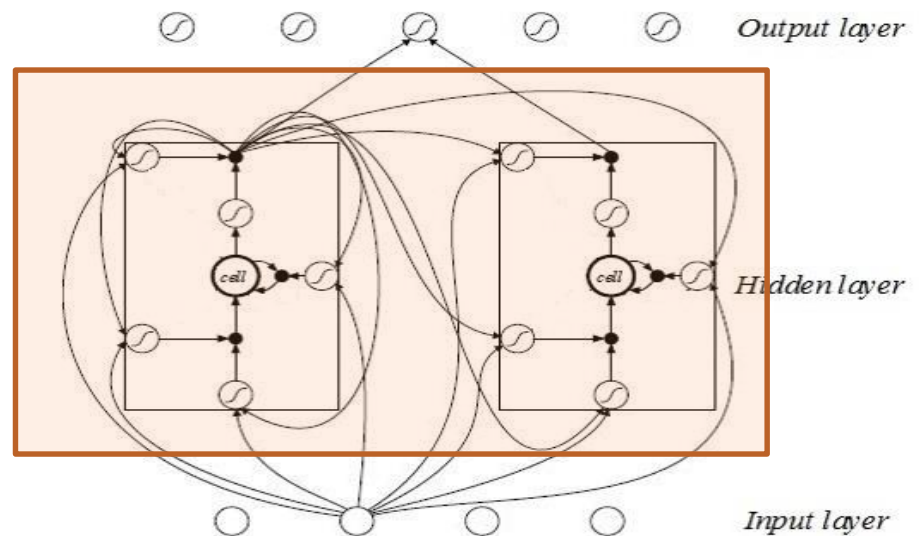
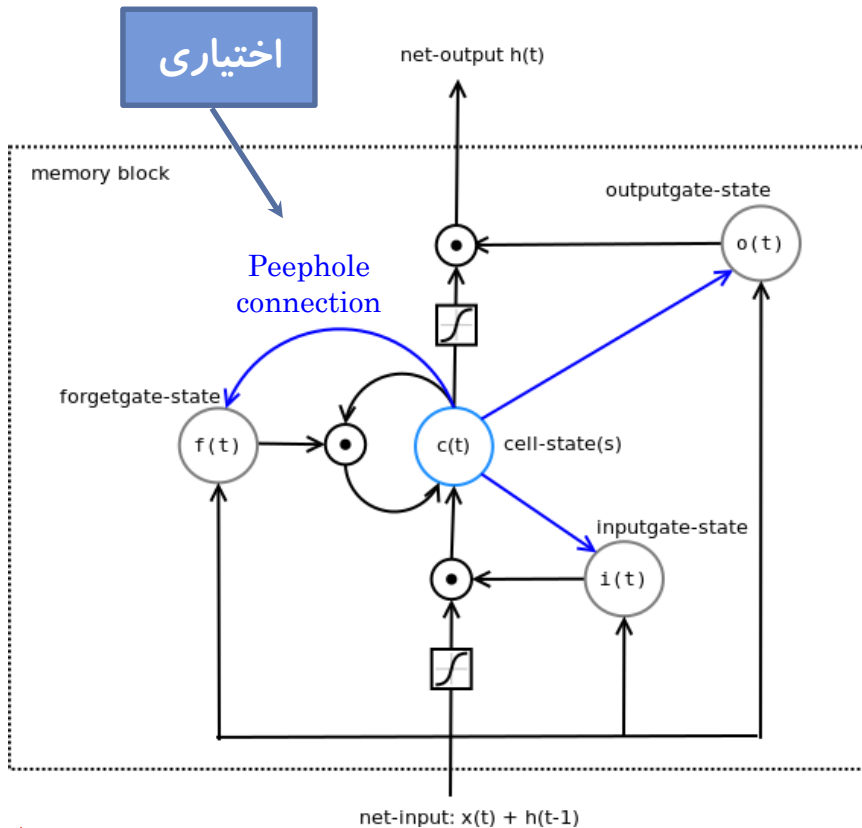


شبکه عصبی حافظه کوتاه مدت ماندگار (LSTM) ...

○ شبکه 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

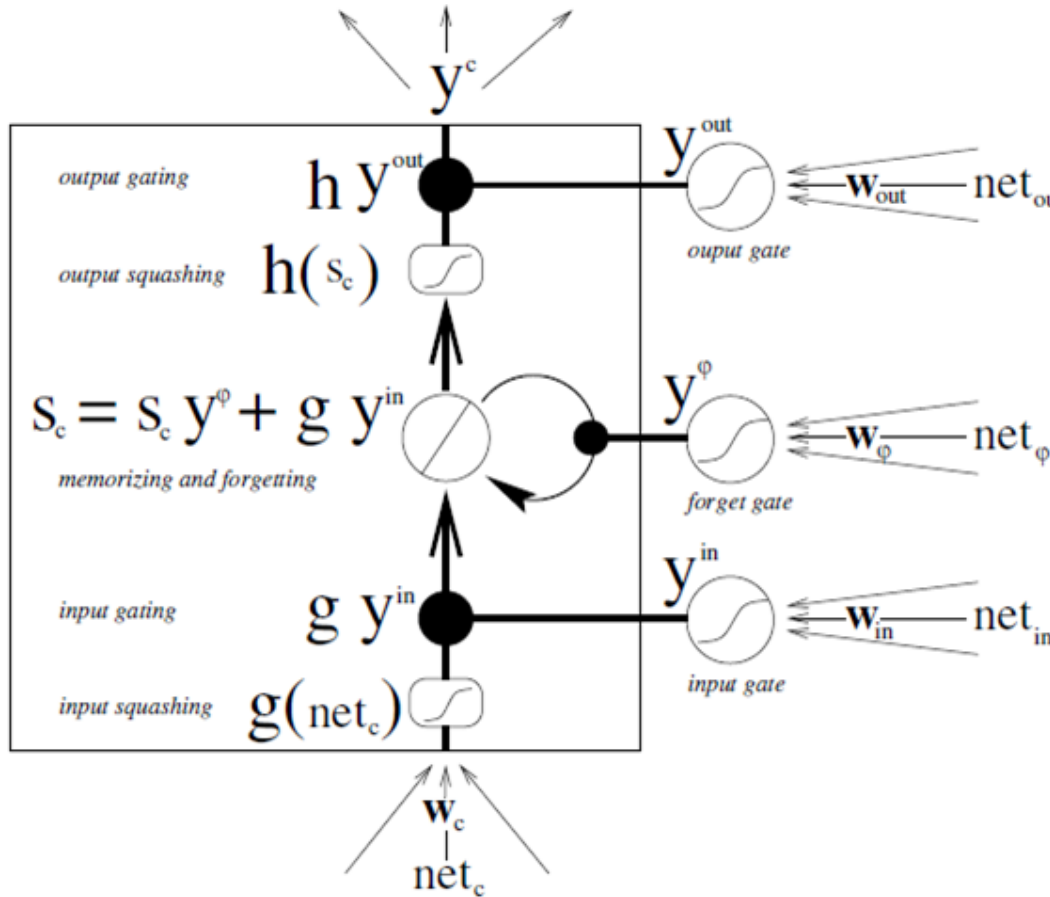
• سلول

○ برای ذخیره اطلاعات در بلوک

• دروازه ورودی

• دروازه فراموشی

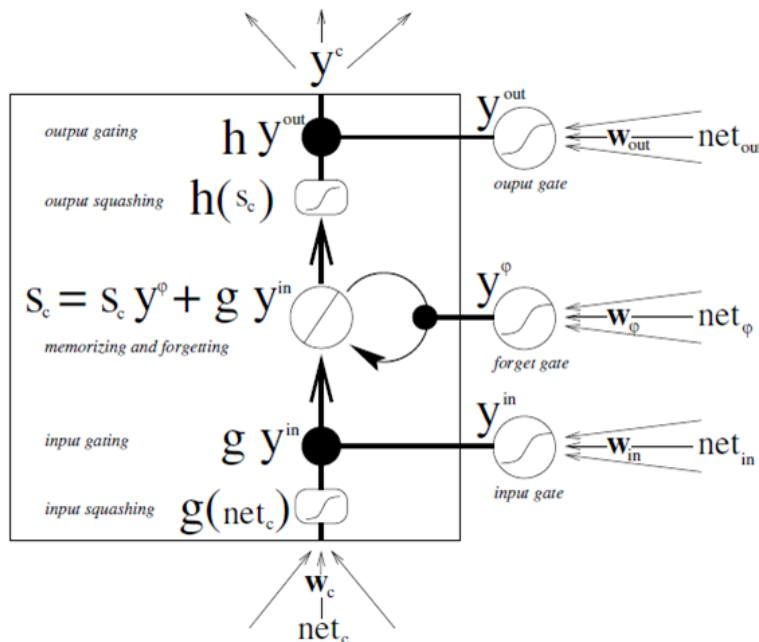
• دروازه خروجی



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

شبکه عصبی LSTM: ساختار بلوک حافظه ...

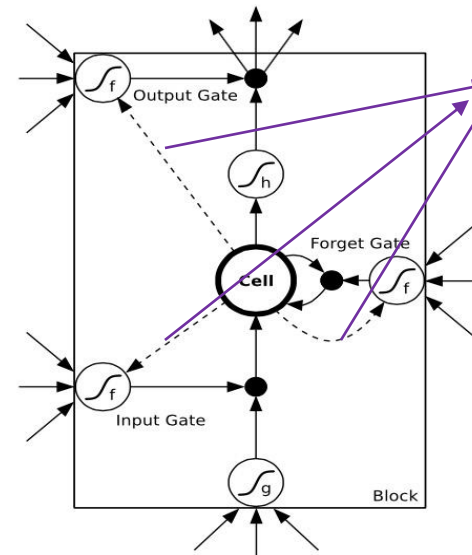
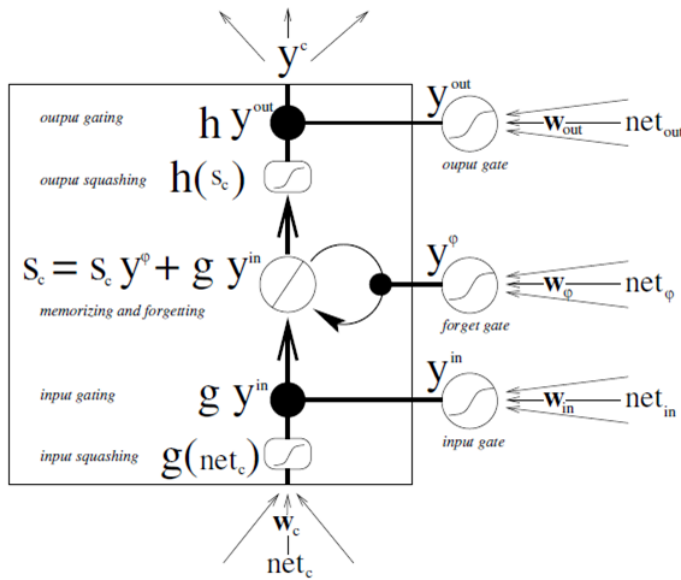
- وظیفه دروازه‌ها: کنترل ورود و خروج داده‌ها و پاک کردن حافظه‌ی بلوک
 - فعال‌ساز دروازه‌ها مقداری بین صفر و یک می‌گیرند.
 - در صورتی که دروازه کاملاً باز باشد فعال‌ساز آن برابر یک و در صورتی که کاملاً بسته باشد فعال‌ساز آن برابر صفر است
- هر سلول حافظه در مرکز خود یک واحد به نام CEC دارد که به فعال‌سازی 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
- در این اسلایدها از این اتصالات صرف‌نظر شده است



Peephole connections

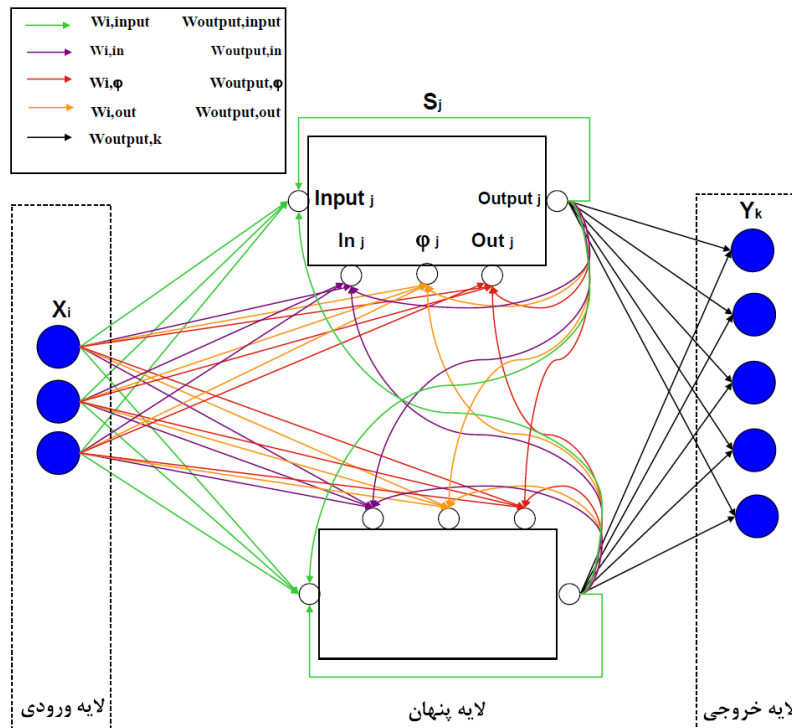
شبکه عصبی LSTM: اتصالات (وزن‌ها)

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

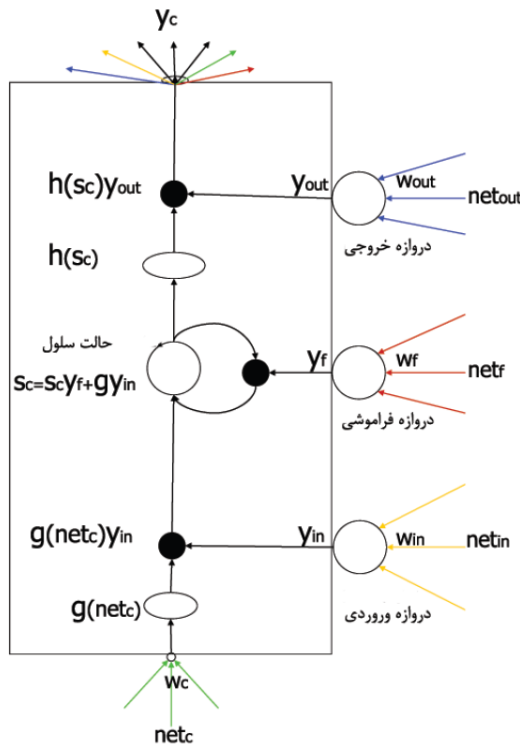
از خروجی بلوک (سلول حافظه)

- 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



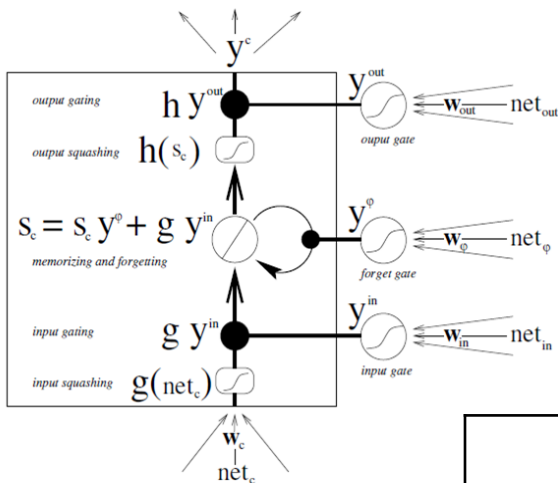
شبکه عصبی LSTM: آموزش ...

○ نمادها



$net_{c_j}(t)$	مقدار خالص ورودی به سلول j
$net_{in_j}(t)$	مقدار خالص ورودی به دروازه ورودی سلول j
$net_{\phi_j}(t)$	مقدار خالص ورودی به دروازه فراموشی سلول j
$net_{out_j}(t)$	مقدار خالص ورودی به دروازه خروجی سلول j
$s_{c_j}(t)$	حالت سلول j
$y^{in_j}(t)$	فعال ساز دروازه ورودی سلول j
$y^{out_j}(t)$	فعال ساز دروازه خروجی سلول j
$y^{\phi_j}(t)$	فعال ساز دروازه فراموشی سلول j
$y^{c_j}(t)$	فعال ساز خروجی سلول j
$net_k(t)$	مقدار خالص ورودی به نرون خروجی شماره k
$y^k(t)$	فعال ساز نرون خروجی شماره k
w_{lm}	یال اتصالی از واحد m به واحد l (اندیس کلی وزن‌ها)
$\delta_{out_j}(t)$	خطای دروازه خروجی سلول j
$\delta_k(t)$	خطای نرون خروجی شماره k
$e_{s_{c_j}}(t)$	خطای حالت سلول j
$0 \leq \alpha \leq 1$	نرخ یادگیری

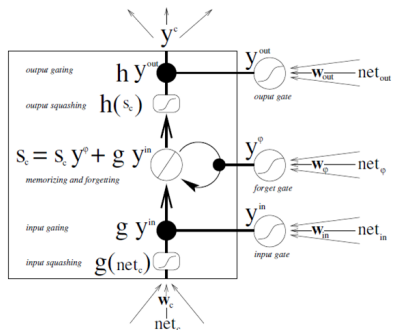
شبکه عصبی LSTM: آموزش ...



○ نمادها

$f(x) = \frac{1}{1 + e^{-x}}$	فعال ساز $f(x)$
$h(x) = \frac{2}{1 + e^{-x}} - 1$	فعال ساز $h(x)$
$g(x) = \frac{4}{1 + e^{-x}} - 2$	فعال ساز $g(x)$
به فعال ساز واحد m در گام زمانی قبل اشاره می‌کند. در صورتی که واحد m معادل نرون i از لایه ورودی باشد این مقدار معادل ورودی i^{am} شبکه (یعنی x_i) در مرحله زمانی فعلی خواهد بود.	
	$y^m(t - 1)$

شبکه عصبی LSTM: آموزش ...



• برای هر دنباله آموزش $X^T = (x_1, x_2, \dots, x_{T-1}, x_T)$ مراحل ۱ تا ۱۵ را انجام دهید.

• مرحله ۰: انتخاب وزن‌های اولیه به صورت تصادفی

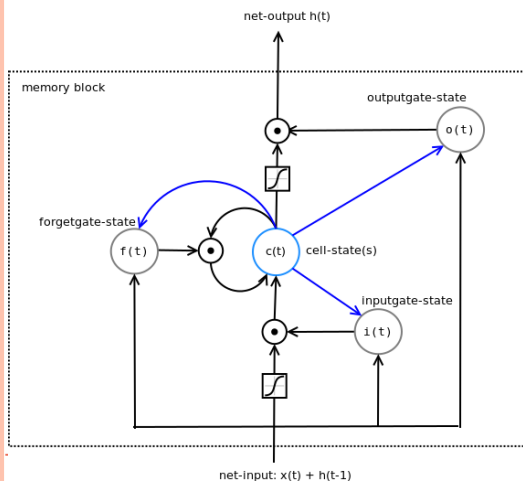
• مرحله ۱: مقدار اولیه حالت سلول‌ها، s_c ، را برابر صفر قرار دهید

• مرحله ۲: مقدار خالص همه ورودی‌های ممکن به دروازه‌های ورودی و فعال‌سازهای آن‌ها را محاسبه کنید

$$net_{in_j}(t) = \sum_m w_{in_j m} y^m(t-1), \quad y^{in_j}(t) = f(net_{in_j}(t))$$

فعال سازی بین صفر و یک

• net_{in_j} معادل جمع وزندار کلیه ورودی‌های ممکن به دروازه ورودی j آمین بلوک حافظه می‌باشد که این ورودی‌ها عبارتند از:

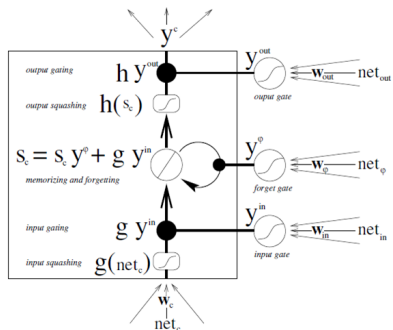


1. نرون‌های لایه ورودی

که در این حالت $y^m(t-1)$ معادل بردار X خواهد بود.

1. خروجی مرحله قبل کلیه بلوک‌های شبکه از جمله خود بلوک j

که در این حالت $y^m(t-1)$ معادل بردار $y^c(t-1)$ خواهد بود.



شبکه عصبی LSTM: آموزش ...

- مرحله ۳: مقدار خالص همه ورودی‌های ممکن به دروازه‌های فراموشی و فعال‌سازهای آن‌ها را محاسبه کنید

$$net_{\phi_j}(t) = \sum_m w_{\phi_j m} y^m(t-1) \quad , \quad y^{\phi_j}(t) = f(net_{\phi_j}(t))$$

فعال سازی بین صفر و یک

- net_{ϕ_j} معادل جمع وزندار کلیه ورودی‌های ممکن به دروازه فراموشی j آمین بلوک حافظه می‌باشد که این ورودی‌ها شامل:

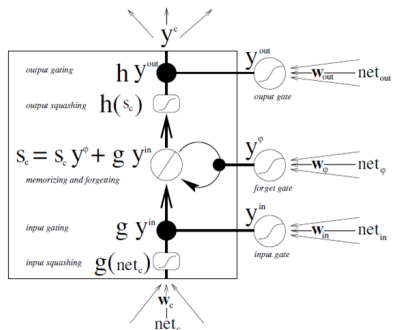
1. نرون‌های لایه ورودی: که در این حالت $y^m(t-1)$ معادل بردار X خواهد بود.
2. خروجی مرحله قبل کلیه بلوک‌های شبکه از جمله خود بلوک j : که در این حالت $y^m(t-1)$ معادل بردار $y^c(t-1)$ خواهد بود.

- مرحله ۴: مقدار خالص ورودی به سلول‌ها و همچنین حالت سلول‌ها را محاسبه کنید

$$net_{c_j}(t) = \sum_m w_{c_j m} y^m(t-1)$$

$$s_{c_j}(t) = s_{c_j}(t-1)y^{\phi_j}(t) + y^{in_j}(t) g\left(net_{c_j}(t)\right) ; \text{ for } t > 0$$

شبکه عصبی LSTM: آموزش ...



- مرحله ۴: مقدار خالص همه ورودی‌های ممکن به سلول‌ها و همچنین حالت سلول‌ها را محاسبه کنید

$$net_{c_j}(t) = \sum_m w_{c_j m} y^m(t-1)$$

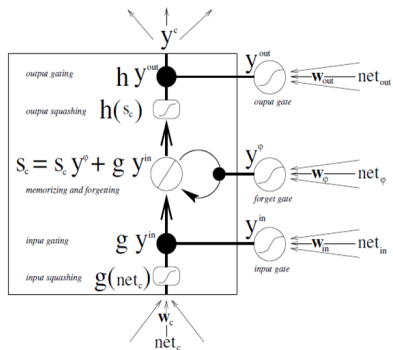
- net_{c_j} معادل جمع وزندار کلیه ورودی‌های ممکن به j آمین بلوک حافظه می‌باشد که این ورودی‌ها عبارتند از:

1. نرون‌های لایه ورودی: که در این حالت $y^m(t-1)$ معادل بردار X خواهد بود.
2. خروجی مرحله قبل کلیه بلوک‌های شبکه از جمله خود بلوک j : که در این حالت $y^m(t-1)$ معادل بردار $y^c(t-1)$ خواهد بود.

$$s_{c_j}(t) = s_{c_j}(t-1)y^{\phi_j}(t) + y^{in_j}(t)g\left(net_{c_j}(t)\right) ; for t > 0$$

فعال سازی بین ۲ و -۲

شبکه عصبی LSTM: آموزش ...



- مرحله ۵: مقدار خالص همه ورودی‌های ممکن به دروازه‌های خروجی و فعال‌سازهای آن‌ها را محاسبه کنید

$$net_{out_j}(t) = \sum_m w_{out_j m} y^m(t-1) \quad , \quad y^{out_j}(t) = f\left(net_{out_j}(t)\right)$$

فعال سازی بین صفر و یک

- net_{out_j} معادل جمع وزندار کلیه ورودی‌های ممکن به دروازه خروجی j آمین بلوک حافظه می‌باشد که این ورودی‌ها شامل:

- نرون‌های لایه ورودی: که در این حالت $y^m(t-1)$ معادل بردار X خواهد بود.
- خروجی مرحله قبل کلیه بلوک‌های شبکه از جمله خود بلوک j : که در این حالت $y^m(t-1)$ معادل بردار $y^c(t-1)$ خواهد بود.

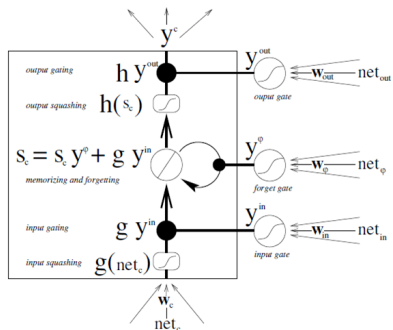
است

- مرحله ۶: خروجی سلول‌ها را محاسبه کنید

$$y^{c_j}(t) = y^{out_j}(t) h(s_{c_j}(t))$$

فعال سازی بین +۱ و -۱

شبکه عصبی LSTM: آموزش ...



○ مرحله ۷: مقدار خالص ورودی به نرون‌های لایه خروجی و فعال‌سازهای آن‌ها را محاسبه کنید

$$net_k(t) = \sum_m w_{km} y^m(t) \quad , \quad y^k(t) = f(net_k(t))$$

○ مرحله ۸: خطای نرون‌های لایه خروجی را محاسبه کنید

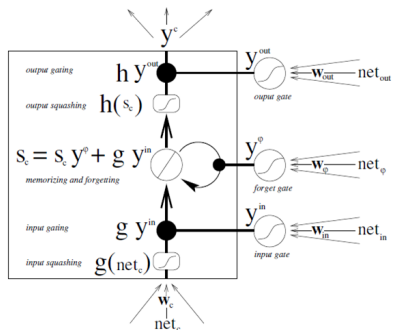
$$\delta_k(t) = f'_k(net_k(t)) e_k(t) \quad , \quad e_k(t) := t^k(t) - y^k(t)$$

○ مرحله ۹: خطای دروازه‌های خروجی را محاسبه کنید

$$\delta_{out_j}(t) = f'_{out_j}(net_{out_j}t)) h(s_{c_j}(t)) \left(\sum_K w_{k c_j} \delta_k(t) \right)$$

وزن یال اتصالی از خروجی بلوک j به نرون k از لایه خروجی

جمع وزن‌دار خطاهای رسیده از لایه خروجی به بلوک j ام حافظه



شبکه عصبی LSTM: آموزش ...

- مرحله ۱۰: خطای حالت سلول‌ها را به کمک رابطه زیر محاسبه کنید

$$e_{s_{c_j}}(t) = y^{out_j}(t) h'(s_{c_j}(t)) \left(\sum_k w_{k \ c_j} \delta_k(t) \right)$$

- مرحله ۱۱: برای محاسبه تغییرات وزنی یال‌های متصل به ورودی سلول‌ها، دروازه‌های ورودی و دروازه‌های فراموشی رندهای زیر را محاسبه کنید

$$\frac{\partial s_{c_j}(t=0)}{\partial w_{lm}} = 0 \quad ; \text{ for } l \in \{\varphi, in, c_j\}$$

مقدار اولیه کلیه رندها را برابر صفر قرار می‌دهیم

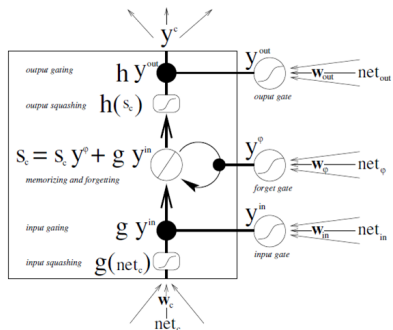
$$\frac{\partial s_{c_j}(t)}{\partial w_{c_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{c_j m}} y^{\varphi_j}(t) + g' \left(net_{c_j}(t) \right) y^{in_j}(t) y^m(t-1)$$

$$\frac{\partial s_{c_j}(t)}{\partial w_{in_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{in_j m}} y^{\varphi_j}(t) + g \left(net_{c_j}(t) \right) f'_{in_j} \left(net_{in_j}(t) \right) y^m(t-1)$$

$$\frac{\partial s_{c_j}(t)}{\partial w_{\varphi_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{\varphi_j m}} y^{\varphi_j}(t) + s_{c_j}(t-1) f'_{\varphi_j} \left(net_{\varphi_j}(t) \right) y^m(t-1)$$

- مرحله ۱۲: تغییرات وزنی یال‌های متصل به لایه خروجی (خروجی سلول به نرون‌های خروجی) را محاسبه کنید

$$\Delta w_{k \ c_j}(t) = \alpha \delta_k(t) y^{c_j}(t)$$



شبکه عصبی LSTM: آموزش ...

- مرحله ۱۳: تغییرات وزنی یال‌های متصل (وارده) به دروازه‌های ورودی و دروازه‌های فراموشی را محاسبه کنید

$$\Delta w_{lm}(t) = \alpha e_{s_{c_j}}(t) \frac{\partial s_{c_j}(t)}{\partial w_{lm}} \text{ for } l \in \{\varphi, in\}$$

- مرحله ۱۴: تغییرات وزنی یال‌های متصل به سلول‌ها (لایه ورودی به سلول) را محاسبه کنید

$$\Delta w_{c_j m}(t) = \alpha e_{s_{c_j}}(t) \frac{\partial s_{c_j}(t)}{\partial w_{c_j m}}$$

- مرحله ۱۵: تغییرات وزنی یال‌های متصل (وارده) به دروازه‌های خروجی را محاسبه کنید

$$\Delta w_{out_j m}(t) = \alpha \delta_{out_j}(t) y^m(t-1)$$

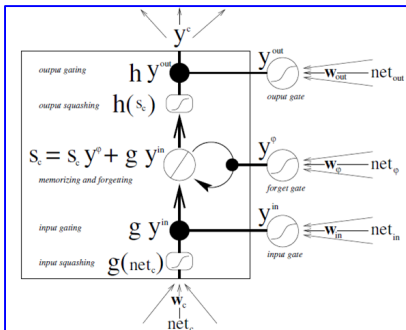


شبکه عصبی LSTM: آموزش

○ نکات کاربردی مهم

- در کلیه روابط، $y^m(t-1)$ به کلیه منابع ورودی به سلول در گام زمانی قبل اشاره می‌کند
 - اگر $y^m(t-1)$ به نرون‌های لایه ورودی اشاره کند آن را برابر مقدار نرون‌های لایه ورودی در مرحله زمانی فعلی قرار می‌دهیم.
- در انتهای هر دنباله آموزش، مقدار حالت و خروجی تمامی سلول‌ها و همچنین تمامی رُندها را برابر صفر قرار می‌دهیم
- به‌روز رسانی وزن‌ها در انتهای هر دنباله آموزشی صورت می‌گیرد.
 - مجموع تغییرات وزن‌ها به ازای همه دنباله‌ها به وزن‌های قبلی اضافه می‌شود
- بهتر است وزن‌های اولیه به‌صورت تصادفی و در بازه -0.1 تا 0.1 انتخاب شود.
- مقدار نرخ یادگیری را کوچک بگیرید، مثلاً 0.01 یا 0.001

شبکه عصبی LSTM: کاربرد ...



- مرحله ۰: برای هر دنباله آزمون $X^T = (x_1, x_2, \dots, x_{T-1}, x_T)$ مراحل ۱ تا ۷ را انجام دهید

- مرحله ۱: مقدار اولیه حالت تمامی سلول‌ها را برابر صفر قرار دهید

- مرحله ۲: برای تمامی بردارهای x_i مراحل ۳ تا ۸ را انجام دهید

- مرحله ۳: مقدار خالص ورودی به دروازه‌های ورودی و فعال‌سازهای آن‌ها را محاسبه کنید

$$net_{in_j}(t) = \sum_m w_{in_j m} y^m(t-1), \quad y^{in_j}(t) = f(net_{in_j}(t))$$

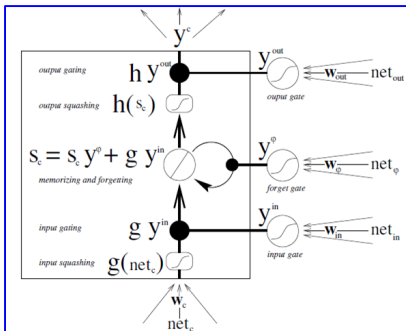
- net_{in_j} معادل جمع وزن‌دار کلیه ورودی‌های ممکن به دروازه ورودی j آمین بلوک حافظه می‌باشد که این ورودی‌ها شامل:

۱. نرون‌های لایه ورودی: که در این حالت $y^m(t-1)$ معادل بردار X خواهد بود.

۲. خروجی مرحله قبل کلیه بلوک‌های شبکه از جمله خود بلوک j : که در این حالت $y^m(t-1)$ معادل بردار $y^c(t-1)$ خواهد بود.

است

شبکه عصبی LSTM: کاربرد ...



- مرحله ۴: مقدار خالص ورودی به دروازه‌های فراموشی و فعال‌سازهای آن‌ها را محاسبه کنید

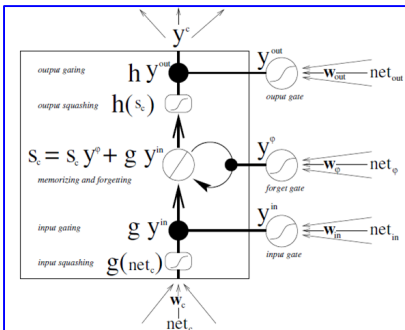
$$net_{\varphi_j}(t) = \sum_m w_{\varphi_j m} y^m(t-1) \quad , \quad y^{\varphi_j}(t) = f(net_{\varphi_j}(t))$$

- معادل جمع وزندار کلیه ورودی‌های ممکن به دروازه فراموشی j آمین بلوک حافظه می‌باشد که این ورودی‌ها شامل:

1. نرون‌های لایه ورودی: که در این حالت $y^m(t-1)$ معادل بردار X خواهد بود.
2. خروجی مرحله قبل کلیه بلوک‌های شبکه از جمله خود بلوک j : که در این حالت $y^m(t-1)$ معادل بردار $y^c(t-1)$ خواهد بود.

است

شبکه عصبی LSTM: کاربرد ...



- مرحله ۵: مقدار خالص ورودی به سلول‌ها و همچنین حالت سلول‌ها را محاسبه کنید

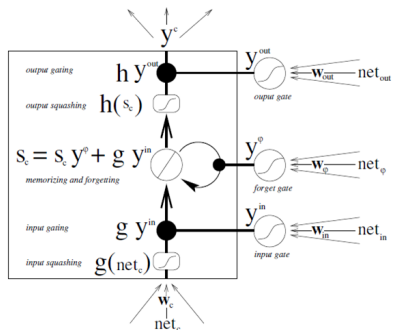
$$net_{c_j}(t) = \sum_m w_{c_j m} y^m(t-1)$$

- معادل جمع وزندار کلیه ورودی‌های ممکن به j آمین بلوک حافظه می‌باشد که این ورودی‌ها شامل:

1. نرون‌های لایه ورودی: که در این حالت $y^m(t-1)$ معادل بردار X خواهد بود.
2. خروجی مرحله قبل کلیه بلوک‌های شبکه از جمله خود بلوک j : که در این حالت $y^m(t-1)$ معادل بردار $y^c(t-1)$ خواهد بود.

است

$$s_{c_j}(t) = s_{c_j}(t-1)y^{\phi_j}(t) + y^{in_j}(t)g\left(net_{c_j}(t)\right) ; for t > 0$$



شبکه عصبی LSTM: کاربرد ...

○ مرحله ۶: مقدار خالص ورودی به دروازه‌های خروجی و فعال‌سازهای آن‌ها را محاسبه کنید

$$net_{out_j}(t) = \sum_m w_{out_j m} y^m(t-1) \quad , \quad y^{out_j}(t) = f(net_{out_j}(t))$$

○ net_{out_j} معادل جمع وزندار کلیه ورودی‌های ممکن به دروازه خروجی j اُمین بلوک حافظه می‌باشد که این ورودی‌ها شامل:

1. نرون‌های لایه ورودی: که در این حالت $y^m(t-1)$ معادل بردار X خواهد بود.

2. خروجی مرحله قبل کلیه بلوک‌های شبکه از جمله خود بلوک j : که در این حالت $y^m(t-1)$ معادل بردار $y^c(t-1)$ خواهد بود.

است

○ مرحله ۷: خروجی سلول‌ها را محاسبه کنید

$$y^{c_j}(t) = y^{out_j}(t) h(s_{c_j}(t))$$

○ مرحله ۸: مقدار خالص ورودی به نرون‌های لایه خروجی و فعال‌سازهای آن‌ها را محاسبه کنید

$$net_k(t) = \sum_m w_{km} y^m(t) \quad , \quad y^k(t) = f(net_k(t))$$

شبکه عصبی LSTM: مثال ...

گرامر ربر

• سال ۱۹۶۷

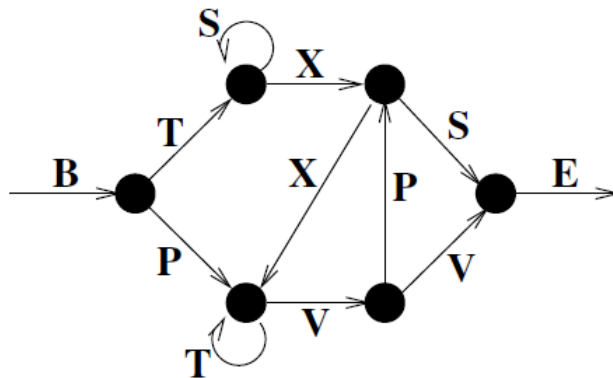
• تولید دنباله‌های با طول‌های مختلف (حتی نامحدود)

• شروع با B و خاتمه با E

• BTSSXXTVVE

• BPVVE

• BPVPXVPXVPXVVE



• هدف: ارائه هر نماد دنباله به شبکه و پیش‌بینی نماد بعدی توسط شبکه

• دنباله آموزش: (B, T, X, S, E)

• نماد آغازین: B

• نماد پایانی: E

• سایر نمادها: S, T, X, V, P

شبکه عصبی LSTM: مثال ...

هدف: تمامی عناصر ممکن با توجه به
عنصر ورودی فعلی. یعنی دو کاراکتر
P و T

ورودی: کاراکتر B

ورودی

هدف

x_1	B	T	P	S	X	V	E		B	T	P	S	X	V	E	t_1
	1	0	0	0	0	0	0		0	1	1	0	0	0	0	
	0	1	0	0	0	0	0		1	0	0	0	0	0	0	
	1	0	0	0	0	0	0		0	1	1	0	0	0	0	
	0	0	1	0	0	0	0		0	1	0	0	0	1	0	
	0	1	0	0	0	0	0		0	1	0	0	0	1	0	
	0	0	0	0	0	1	0		0	0	1	0	0	1	0	
	0	0	1	0	0	0	0		0	0	0	1	1	0	0	
	0	0	0	0	1	0	0		0	1	0	0	0	1	0	
	0	0	0	0	0	1	0		0	0	1	0	0	1	0	
	0	0	0	0	0	1	0		0	0	0	0	0	0	1	
	0	0	0	0	0	0	1		0	1	0	0	0	0	0	
x_{10}	0	1	0	0	0	0	0		0	0	0	0	0	0	1	t_{10}

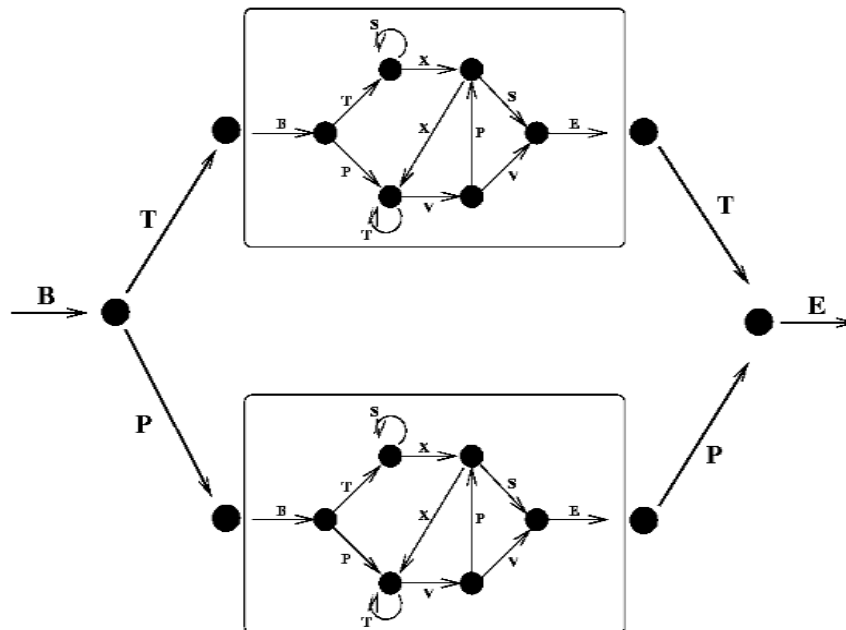
پیش بینی دنباله تولید شده توسط گرامر ربر

• دنباله آموزشی و دنباله هدف متناظر

$$X = (X_1, X_2, \dots, X_{10})$$

$$X_1 = (1, 0, 0, 0, 0, 0) = B$$

$$t_1 = (0, 1, 1, 0, 0, 0, 0) = T, P$$



شبکه عصبی LSTM: مثال ...

○ ساختار شبکه

- تعداد نرون‌های لایه ورودی: ۷

○ مربوط به ۷ نماد (B,P,S,T,V,X,E) که در دنباله ورودی می‌آید

- تعداد نرون‌های لایه خروجی: ۷

○ مربوط به ۷ نماد (B,P,S,T,V,X,E) که در دنباله هدف می‌آید

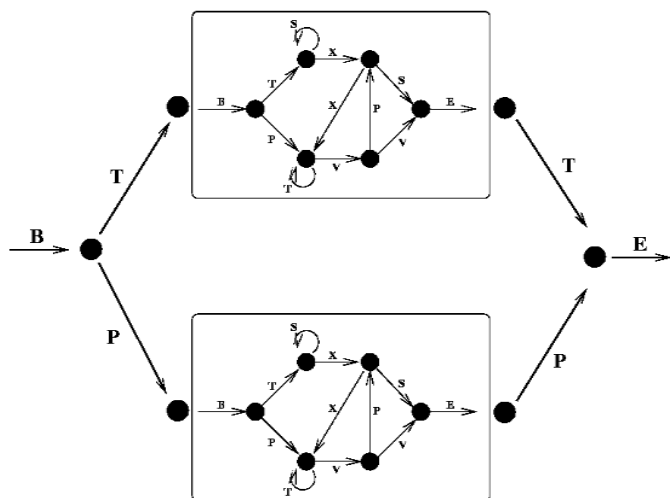
- تعداد بلوک‌های لایه پنهان: ۳

- نرخ یادگیری: ۰.۰۱

- وزن‌های اولیه: تصادفی در بازه $[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}]$

- تعداد داده‌های آموزشی

○ 60000 دنباله تصادفی



شبکه عصبی LSTM: مثال ...

آموزش ...

مرحله ۰: انتخاب وزن‌های اولیه به صورت تصادفی

وزن اولیه بین لایه ورودی و بلوک حافظه:

بصورت تصادفی بین -0.1 تا 0.1

بلوک حافظه

-0.1551	0.0359	0.0667
-0.1798	0.0956	-0.0473
-0.1779	-0.0722	0.0875
-0.1895	0.1393	-0.0325
0.1822	-0.0025	-0.1776
0.0101	-0.1541	0.0340
0.1025	0.1249	-0.0805

لایه ورودی

دروازه ورودی

-0.0602	-0.1342	-0.0259
-0.1914	0.0028	0.0784
-0.0636	0.0033	-0.1934
-0.0462	-0.1794	-0.1390
-0.1627	-0.0532	0.1598
0.1877	0.0339	-0.1814
-0.1240	0.1979	-0.1883

لایه ورودی

وزن اولیه بین لایه ورودی و دروازه ورودی:

بصورت تصادفی بین -0.1 تا 0.1

شبکه عصبی LSTM: مثال ...

دروازه فراموشی

-0.0362	-0.1848	-0.1138
0.0344	0.1690	-0.0059
-0.0738	-0.0507	-0.1122
-0.1411	0.1957	0.1747
0.1966	0.0174	-0.0890
-0.1570	0.0253	0.1812
-0.0663	-0.1248	0.1260

○ وزن اولیه بین لایه ورودی و دروازه فراموشی:

○ بصورت تصادفی بین -0.1 تا 0.1

دروازه خروجی

0.0298	-0.1180	-0.1193
-0.0638	-0.1049	-0.1492
-0.1123	0.1537	0.0751
0.1616	0.1685	0.1643
0.0732	0.1127	0.1004
-0.0513	-0.1032	0.1292
-0.0396	0.0334	-0.1100

○ وزن اولیه بین لایه ورودی و دروازه خروجی:

○ بصورت تصادفی بین -0.1 تا 0.1

ورودی بلوک‌ها

-0.0503	0.0632	-0.0233
-0.1866	0.0301	0.1111
-0.0375	0.1904	0.0278

○ وزن اولیه بین خروجی بلوک‌ها و ورودی بلوک‌ها:

○ بصورت تصادفی بین -0.1 تا 0.1

شبکه عصبی LSTM: مثال ...

دروازه ورودی

خروجی بلوک‌ها

-0.0453	-0.1578	-0.0879
-0.0325	0.0125	-0.1209
0.1743	0.0485	-0.0366

وزن اولیه بین خروجی بلوک‌ها و دروازه ورودی:

بصورت تصادفی بین -0.1 تا 0.1

دروازه فراموشی

خروجی بلوک‌ها

-0.0550	-0.0238	-0.0115
-0.1352	0.0681	-0.1800
0.1574	-0.1158	-0.0016

وزن اولیه بین خروجی بلوک‌ها و دروازه فراموشی:

بصورت تصادفی بین -0.1 تا 0.1

دروازه خروجی

خروجی بلوک‌ها

-0.0921	-0.0433	0.0446
0.0737	0.0235	0.0325
0.0992	0.0929	-0.1718

وزن اولیه بین خروجی بلوک‌ها و دروازه خروجی:

بصورت تصادفی بین -0.1 تا 0.1

لایه خروجی

خروجی بلوک‌ها

-0.1916	-0.1121	0.1680	-0.0545	-0.1061	-0.0808	0.1258
0.0049	-0.1078	-0.0724	-2.7298e-...	-0.1911	0.1304	0.1444
-0.0370	-0.0246	-0.0863	0.1254	-0.0796	-0.1454	0.0212

وزن اولیه بین خروجی‌های بلوک‌ها و لایه خروجی:

بصورت تصادفی بین -0.1 تا 0.1



شبکه عصبی LSTM: مثال ...

○ آموزش ...

- مرحله ۱: مقدار اولیه حالت و خروجی تمامی سلول‌ها را برابر صفر قرار دهید
- مرحله ۲: مقدار خالص ورودی به دروازه‌های ورودی و فعال‌سازهای آن‌ها را محاسبه کنید

$$((-0.0602*1)+(-0.1914*0)+(-0.0636*0)+(-0.0462*0)+(0.1627*0)+(0.1877*0)+(-0.1240*0))+((0*(-0.0453))*(0*(-0.0325))*(0*0.1743))=-0.0602$$

$net_{in} =$

-0.0602	-0.1342	-0.0259
---------	---------	---------

○ مقدار خالص ورودی به دروازه‌های ورودی

$$net_{in_j}(t) = \sum_m w_{in_j m} y^m(t-1)$$

○ فعال‌ساز دروازه‌های ورودی

$y^{in} =$

0.4850	0.4665	0.4935
--------	--------	--------

$$y^{in_j}(t) = f_{in_j}(net_{in_j}(t))$$

$Sigmoid(-0.0602)=0.4850$



شبکه عصبی LSTM: مثال ...

- مرحله ۳: مقدار خالص ورودی به دروازه‌های فراموشی و فعال‌سازهای آن‌ها را محاسبه کنید

$$((-0.0362*1)+(0.0344*0)+(-0.0738*0)+(-0.1411*0)+(0.1966*0)+(-0.1570*0)+(-0.0633*0))+((0*(-0.0550))* (0*(-0.1352)) * (0*0.1574)) = -0.0362$$

$net_{\varphi} =$

-0.0362	-0.1848	-0.1138
---------	---------	---------

○ مقدار خالص ورودی به دروازه‌های فراموشی

$$net_{\varphi_j}(t) = \sum_m w_{\varphi_j m} y^m(t-1)$$

○ فعال‌سازهای دروازه‌های فراموشی

$y^{\varphi} =$

0.4910	0.4539	0.4716
--------	--------	--------

$$\text{Sigmoid}(-0.0362) = 0.4910$$

$$y^{\varphi_j}(t) = f_{\varphi_j}(net_{\varphi_j}(t))$$



شبکه عصبی LSTM: مثال ...

- مرحله ۴: مقدار خالص ورودی به سلول‌ها و همچنین حالت سلول‌ها را محاسبه کنید

○ مقدار خالص ورودی به سلول‌ها

$$(0.0359 * 1) + (0.0956 * 0) + (-0.0722 * 0) + (0.1393 * 0) + (-0.0025 * 0) + (-0.1541 * 0) + (0.1249 * 0) = 0.0359$$

$$net_c = \begin{bmatrix} -0.1551 & 0.0359 & 0.0667 \end{bmatrix}$$

$$net_{c_j}(t) = \sum_m w_{c_j m} y^m(t-1)$$

○ حالت سلول‌ها

$$S_c = \begin{bmatrix} -0.0751 & 0.0168 & 0.0329 \end{bmatrix}$$

$$s_{c_j}(t) = s_{c_j}(t-1)y^{\phi_j}(t) + y^{in_j}(t) g\left(net_{c_j}(t)\right)$$

$$(0 * 0.4910) + (0.485 * (-0.1548)) = -0.0751$$



شبکه عصبی LSTM: مثال ...

- مرحله ۵: مقدار خالص ورودی به دروازه‌های خروجی و فعال‌سازهای آنها را محاسبه کنید
- مقدار خالص ورودی به دروازه‌های خروجی

$$(0.0298 * 1) + (-0.0638 * 0) + (-0.1123 * 0) + (0.1616 * 0) + (0.0732 * 0) + (-0.0513 * +(-0.0396 * 0) + ((0 * (-0.0921)) * (0 * 0.0737) * (0 * 0.0992)) = 0.0298$$

$$net_{out} = \begin{bmatrix} 0.0298 & -0.1180 & -0.1193 \end{bmatrix}$$

$$net_{out_j}(t) = \sum_m w_{out_j m} y^m(t-1)$$

- فعال‌سازهای دروازه‌های خروجی

$$y^{out} = \begin{bmatrix} 0.5074 & 0.4705 & 0.4702 \end{bmatrix}$$

$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t))$$

$$\text{Sigmoid}(0.0298) = 0.5074$$

شبکه عصبی LSTM: مثال ...

$$0.5074 * (-0.0375) = -0.190$$

$$y^c = \begin{bmatrix} -0.0190 & 0.0039 & 0.0077 \end{bmatrix}$$

- مرحله ۶: خروجی سلول‌ها را محاسبه کنید

$$y^{cj}(t) = y^{outj}(t) h(s_{cj}(t))$$

- مرحله ۷: مقدار خالص ورودی به نرون‌های لایه خروجی و فعال‌سازهای آن‌ها را محاسبه کنید

○ مقدار خالص ورودی به نرون‌های لایه خروجی

$$\begin{aligned} &(-0.0190 * (-0.1916)) + (0.0039 * 0.0049) \\ &+ (0.0077 * (-0.0370)) = 0.0034 \end{aligned}$$

$$net_k(t) = \sum_m w_{km} y^m(t)$$

$$net_k = \begin{bmatrix} 0.0034 & 0.0015 & -0.0042 & 0.0020 & 6.5059e-04 & 9.2639e-04 & -0.0017 \end{bmatrix}$$

○ فعال‌سازهای نرون‌های لایه خروجی

$$y^k =$$

$$\begin{bmatrix} 0.5008 & 0.5004 & 0.4990 & 0.5005 & 0.5002 & 0.5002 & 0.4996 \end{bmatrix}$$

$$\text{Sigmoid}(0.0034) = 0.5008$$

$$y^k(t) = f(net_k(t))$$



شبکه عصبی LSTM: مثال ...

مرحله ۸: خطای نرون‌های لایه خروجی را محاسبه کنید

$$\delta_k(t) = f'_k(\text{net}_k(t)) e_k(t), e_k(t) := t^k(t) - y^k(t)$$

$$\delta_k = \begin{bmatrix} -0.1252 & 0.1249 & 0.1253 & -0.1251 & -0.1250 & -0.1251 & -0.1249 \end{bmatrix}$$

$$(0 - 0.5008) * (0.5008) * (1 - 0.5008) = -0.1251$$

مرحله ۹: خطای دروازه‌های خروجی را محاسبه کنید

$$\delta_{out} = \begin{bmatrix} -4.2666\text{e-}04 & -7.0063\text{e-}05 & 2.2690\text{e-}06 \end{bmatrix}$$

$$\delta_{out_j}(t) = f'_{out_j}(\text{net}_{out_j}t)) h(s_{c_j}(t)) \left(\sum_K w_{k c_j} \delta_k(t) \right)$$

$$\begin{aligned} & 0.5074 * (1 - 0.5074) * (-0.0375) * \\ & ((-0.1252) * (-0.1916)) + (0.1249 * (-0.1121)) + (0.1253 * 0.1680) + ((-0.1251) * (-0.0545)) \\ & + ((-0.1250) * (-0.1061)) + ((-0.1251) * (-0.0808)) + (-0.1249 * 0.1258) = -0.00042 \end{aligned}$$



شبکه عصبی LSTM: مثال ...

- مرحله ۱۰: خطای حالت سلول‌ها را محاسبه کنید

$$e_{s_{c_j}}(t) = y^{out_j}(t) h'(s_{c_j}(t)) \left(\sum_k w_{k \ c_j} \delta_k(t) \right)$$

$e_{s_c} =$

0.0115	-0.0079	1.3007e-04
--------	---------	------------

$$0.5074 * 0.4993 * ((-0.1252) * (-0.1916)) + (0.1249 * (-0.1121)) + (0.1253 * 0.1680) + ((-0.1251)$$

شبکه عصبی LSTM: مثال ...

• مرحله ۱۱: رُندهای زیر را محاسبه کنید

○ محاسبه رُنْد یال‌های بین لایه ورودی و ورودی بلوک‌ها:

$$\frac{\partial s_{c_j}(t)}{\partial w_{c_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{c_j m}} y^{\varphi_j}(t) + g' \left(net_{c_j}(t) \right) y^{in_j}(t) y^m(t-1)$$

ورودی بلوک‌ها

$$(0 * 0.4910) + (0.9940 * 0.4850 * 1) = 0.4821$$

لایه ورودی

0.4821	0.4664	0.4930
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

شبکه عصبی LSTM: مثال ...

محاسبه رُند یال‌های بین خروجی بلوک‌ها و ورودی بلوک‌ها:

$$\frac{\partial s_{c_j}(t)}{\partial w_{c_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{c_j m}} y^{\varphi_j}(t) + g' \left(net_{c_j}(t) \right) y^{in_j}(t) y^m(t-1)$$

ورودی بلوک‌ها

0	0	0
0	0	0
0	0	0

خروجی بلوک‌ها

$$(0 * 0.4910) + (0.9940 * 0.4850 * 0) = 0$$

محاسبه رُند یال‌های بین لایه ورودی و دروازه‌های ورودی:

$$\frac{\partial s_{c_j}(t)}{\partial w_{in_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{in_j m}} y^{\varphi_j}(t) + g \left(net_{c_j}(t) \right) f'_{in_j} \left(net_{in_j}(t) \right) y^m(t-1)$$

دروازه ورودی

-0.0387	0.0089	0.0167
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

لایه ورودی

$$(0 * 0.4910) + (-0.1548 * 0.4850 * (1 - 0.4850) * 1) = -0.0387$$

شبکه عصبی LSTM: مثال ...

محاسبه رُند یال‌های بین خروجی بلوک‌ها و دروازه‌های ورودی:

$$\frac{\partial s_{c_j}(t)}{\partial w_{in_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{in_j m}} y^{\varphi_j}(t) + g\left(\text{net}_{c_j}(t)\right) f'_{in_j}\left(\text{net}_{in_j}(t)\right) y^m(t-1)$$

خروجی بلوک‌ها

دروازه ورودی

0	0	0
0	0	0
0	0	0

$$(0 * 0.4910) + (-0.1548 * 0.4850 * (1 - 0.4850) * 0) = 0$$

محاسبه رُند یال‌های بین لایه ورودی و دروازه‌های فراموشی:

$$\frac{\partial s_{c_j}(t)}{\partial w_{\varphi_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{\varphi_j m}} y^{\varphi_j}(t) + s_{c_j}(t-1) f'_{\varphi_j}\left(\text{net}_{\varphi_j}(t)\right) y^m(t-1)$$

دروازه فراموشی

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

$$(0 * 0.4910) + (0 * 0.4910 * (1 - 0.4910) * 1) = 0$$

لایه ورودی

شبکه عصبی LSTM: مثال ...

محاسبه رُند یال‌های بین خروجی بلوک‌ها و دروازه‌های فراموشی

$$\frac{\partial s_{c_j}(t)}{\partial w_{\varphi_j m}} = \frac{\partial s_{c_j}(t-1)}{\partial w_{\varphi_j m}} y^{\varphi_j}(t) + s_{c_j}(t-1) f'_{\varphi_j}(\text{net}_{\varphi_j}(t)) y^m(t-1)$$

دروازه فراموشی

0	0	0
0	0	0
0	0	0

$$(0 * 0.4910) + (0 * 0.4910 * (1 - 0.4910) * 0) = 0$$

مرحله ۱۲: تغییرات وزنی یال‌های ورودی به لایه خروجی را محاسبه کنید

$$\Delta w_{k c_j}(t) = \alpha \delta_k(t) y^{c_j}(t)$$

$$0.1 * (-0.1251) * (-0.190) = 0.00238$$

لایه خروجی

2.3841e-05	-2.3783e-05	-2.3850e-05	2.3825e-05	2.3808e-05	2.3812e-05	2.3781e-05
-4.9366e-06	4.9246e-06	4.9385e-06	-4.9333e-06	-4.9299e-06	-4.9306e-06	-4.9242e-06
-9.6905e-06	9.6668e-06	9.6942e-06	-9.6838e-06	-9.6773e-06	-9.6786e-06	-9.6661e-06

شبکه عصبی LSTM: مثال ...

- مرحله ۱۳: تغییرات وزنی یال‌های ورودی به دروازه‌های ورودی و دروازه‌های فراموشی را محاسبه کنید

محاسبه تغییر وزن یال‌های بین لایه ورودی و دروازه‌های ورودی

$$\Delta w_{in\ m}(t) = \alpha e_{s_{c_j}}(t) \frac{\partial s_{c_j}(t)}{\partial w_{l\ m}}$$

$$0.1 * 0.0115 * (-0.0387) = -0.000044$$

دروازه ورودی

-4.4571e-06	-7.0592e-07	2.1689e-08
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

لایه ورودی

محاسبه تغییر وزن یال‌های بین خروجی بلوک‌ها و دروازه‌های ورودی

$$\Delta w_{in\ m}(t) = \alpha e_{s_{c_j}}(t) \frac{\partial s_{c_j}(t)}{\partial w_{l\ m}}$$

$$0.1 * 0.0115 * 0 = 0$$

دروازه ورودی

0	0	0
0	0	0
0	0	0

خروجی بلوک‌ها

شبکه عصبی LSTM: مثال ...

محاسبه تغییر وزن یال‌های بین لایه ورودی و دروازه‌های فراموشی: $(\Delta w_{\phi m})$

$$\Delta w_{\phi m}(t) = \alpha e_{s_{c_j}}(t) \frac{\partial s_{c_j}(t)}{\partial w_{l m}}$$

دروازه فراموشی

$$0.1 * 0.0115 * 0 = 0$$

لایه ورودی

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

محاسبه تغییر وزن یال‌های بین خروجی بلوک‌ها و دروازه‌های فراموشی:

$$0.1 * 0.0115 * 0 = 0$$

دروازه فراموشی

خروجی بلوک‌ها

0	0	0
0	0	0
0	0	0

$$\Delta w_{\phi m}(t) = \alpha e_{s_{c_j}}(t) \frac{\partial s_{c_j}(t)}{\partial w_{l m}}$$



شبکه عصبی LSTM: مثال ...

- مرحله ۱۴: تغییرات وزنی یال‌های ورودی به بلوک‌ها را محاسبه کنید

محاسبه تغییر وزن یال‌های بین لایه ورودی و ورودی بلوک‌ها:

$$0.1 * 0.0115 * 0.4821 = 0.00055$$

ورودی بلوک‌ها

5.5564e-05	-3.6822e-05	6.4122e-07
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

لایه ورودی

$$\Delta w_{c_j m}(t) = \alpha e_{s_{c_j}}(t) \frac{\partial s_{c_j}(t)}{\partial w_{c_j m}}$$

محاسبه تغییر وزن یال‌های بین خروجی بلوک‌ها و ورودی بلوک‌ها:

$$0.1 * 0.0115 * 0 = 0$$

ورودی بلوک‌ها

0	0	0
0	0	0
0	0	0

خروجی بلوک‌ها

$$\Delta w_{c_j m}(t) = \alpha e_{s_{c_j}}(t) \frac{\partial s_{c_j}(t)}{\partial w_{c_j m}}$$

شبکه عصبی LSTM: مثال ...

- مرحله ۱۵: تغییرات وزنی یال‌های ورودی به دروازه‌های خروجی را محاسبه کنید

○ محاسبه تغییر وزن یال‌های بین لایه ورودی و دروازه‌های خروجی:

$$0.1 * (-0.00042) * 1 = -0.000042$$

دروازه خروجی

-4.2666e-06	-7.0063e-07	2.2690e-08
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

لایه ورودی

$$\Delta w_{out_j m}(t) = \alpha \delta_{out_j}(t) y^m(t-1)$$

○ محاسبه تغییر وزن یال‌های بین خروجی بلوک‌ها و دروازه‌های خروجی

دروازه خروجی

$$0.1 * (-0.00042) * 0 = 0$$

0	0	0
0	0	0
0	0	0

خروجی بلوک‌ها

$$\Delta w_{out_j m}(t) = \alpha \delta_{out_j}(t) y^m(t-1)$$



شبکه عصبی LSTM: مثال

- مرحله ۱۶: تغییرات وزن محاسبه شده در این گام زمانی تنها ذخیره کنید (با وزن‌های قبلی شبکه جمع نشود)
- مرحله ۱۷: برای تمامی گام‌های زمانی تمامی مراحل ۱ تا ۱۶ را تکرار کنید
- مرحله ۱۸: در انتهای آخرین گام زمانی مجموع تغییرات وزن‌های ذخیره شده در مرحله ۱۶ را به وزن‌های شبکه اضافه نمایید

برای دنباله آموزشی بعدی تمامی گام‌های ۱ تا ۱۸ را تکرار کنید.

شبکه عصبی LSTM: مثال از کاربرد ...

فرض کنید وزن‌های بدست آمده از مرحله آموزش گرامر بصورت زیر باشد:
بلوک حافظه

○ وزن بین لایه ورودی و بلوک حافظه:

لایه ورودی

3.9151	-3.2348	-3.0269
0.9615	1.2860	4.2637
4.7583	0.7312	3.1965
4.3264	-2.0329	-2.1017
2.6357	-1.6746	0.8470
2.6612	-1.5102	-3.3226
4.5521	-0.3698	-0.7786

○ وزن بین لایه ورودی و دروازه ورودی:

دروازه ورودی

لایه ورودی

4.8625	2.0639	0.9591
1.4073	0.4284	1.7322
0.8088	-0.9127	-0.5602
0.3574	1.1875	0.7015
1.7834	0.8217	2.1451
-0.3265	0.8664	0.3634
1.0398	-0.0357	0.2478

شبکه عصبی LSTM: مثال از کاربرد ...

دروازه فراموشی

○ وزن بین لایه ورودی و دروازه فراموشی:

لایه ورودی

-4.2569	0.7874	2.2658
0.3351	0.7287	3.9433
2.2501	0.7359	-4.0696
-0.4863	1.3139	-0.2710
-0.5954	1.0097	0.3341
3.0903	0.9282	-2.2383
2.1039	0.2374	-0.2583

دروازه خروجی

○ وزن بین لایه ورودی و دروازه خروجی:

لایه ورودی

4.7166	1.0076	-0.1894
3.2706	3.2495	2.7557
-0.9194	0.9137	3.1509
-2.2420	3.6590	2.5515
1.5076	3.6636	4.4850
2.1366	-3.3384	1.4218
2.0888	2.3657	1.5189

○ وزن بین خروجی بلوک‌ها و ورودی بلوک‌ها:

ورودی بلوک‌ها

خروجی بلوک‌ها

-1.7355	0.3540	-5.7765
-3.1873	1.9654	-2.9279
8.1901	-1.0391	4.1037

شبکه عصبی LSTM: مثال از کاربرد ...

دروازه ورودی

خروجی بلوکها

1.2279	0.2777	1.0664
-4.1167	-2.1342	-3.4557
0.2021	0.3564	-0.1380

○ وزن بین خروجی بلوکها و دروازه ورودی:

دروازه فراموشی

خروجی بلوکها

0.5370	2.6669	-0.0521
1.1137	-2.7448	2.6063
-1.2385	-0.7886	-1.8696

○ وزن بین خروجی بلوکها و دروازه فراموشی:

دروازه خروجی

خروجی بلوکها

-1.0283	2.0204	-0.7760
-3.2926	4.3884	-0.2293
3.8964	-1.5900	1.4185

○ وزن بین خروجی بلوکها و دروازه خروجی:

لایه خروجی

خروجی بلوکها

-2.6071	6.4393	10.3822	-16.7756	-16.7744	8.1711	-2.4454
2.4884	-3.3992	14.0191	-2.1289	-2.1267	11.1964	2.1982
0.1496	14.6971	-5.2526	-2.0799	-2.0826	8.8300	-0.8454

○ وزن بین خروجی‌های بلوکها و لایه خروجی:



شبکه عصبی LSTM: مثال از کاربرد ...

- مرحله ۰: دنباله آزمون را معادل دنباله زیر قرار دهید:

ورودی

هدف

	B	T	P	S	X	V	E		B	T	P	S	X	V	E
x_1	1	0	0	0	0	0	0		0	1	1	0	0	0	0
	0	1	0	0	0	0	0		1	0	0	0	0	0	0
	1	0	0	0	0	0	0		0	1	1	0	0	0	0
	0	1	0	0	0	0	0		0	0	0	1	1	0	0
	0	0	0	1	0	0	0		0	0	0	1	1	0	0
	0	0	0	1	0	0	0		0	0	0	1	1	0	0
	0	0	0	0	1	0	0		0	0	0	1	1	0	0
	0	0	0	1	0	0	0		0	0	0	0	0	0	1
	0	0	0	0	0	0	1		0	1	0	0	0	0	0
x_{10}	0	1	0	0	0	0	0		0	0	0	0	0	0	1

t_1

t_{10}

- مرحله ۱: مقدار اولیه حالت و خروجی تمامی سلول‌ها را برابر صفر قرار دهید
- مرحله ۲: برای تمامی بردارهای x_i مراحل ۳ تا ۶ را انجام دهید



شبکه عصبی LSTM: مثال ...

مرحله ۳: دنباله ورودی برابر x_i و هدف را برابر t_i قرار دهید

$$X_1 = (1, 0, 0, 0, 0, 0) = \mathbf{B}$$

$$t_1 = (0, 1, 1, 0, 0, 0, 0) = \mathbf{T}, \mathbf{P}$$

مرحله ۴: مقدار خالص ورودی به دروازه‌های ورودی و فعال‌سازهای آنها را محاسبه کنید

$$(4.8625 * 1) + (1.4073 * 0) + (0.8088 * 0) + (0.3574 * 0) + (1.7834 * 0) + (-0.3265 * 0) + (1.0398$$

$$net_{in} = \begin{bmatrix} 4.8625 & 2.0639 & 0.9591 \end{bmatrix}$$

$$y^{in} = \begin{bmatrix} 0.9923 & 0.8873 & 0.7229 \end{bmatrix}$$

$$\text{Sigmoid}(4.8625) = 0.9923$$

مقدار خالص ورودی به دروازه‌های ورودی

$$net_{in_j}(t) = \sum_m w_{in_j m} y^m(t-1)$$

فعال‌ساز دروازه‌های ورودی

$$y^{in_j}(t) = f_{in_j}(net_{in_j}(t))$$



شبکه عصبی LSTM: مثال ...

- مرحله ۳: مقدار خالص ورودی به دروازه‌های فراموشی و فعال‌سازهای آن‌ها را محاسبه کنید

$$((-4.2569 * 1) + (0.3351 * 0) + (2.2501 * 0) + (-0.4863 * 0) + (-0.5954 * 0) + (3.0903 * 0) + (2.1039 * 0)) + ((0 * 0.5370) + (0 * 1.1137) + (0 * (-1.2385))) = -4.2569$$

$$net_{\varphi} = \begin{bmatrix} -4.2569 & 0.7874 & 2.2658 \end{bmatrix}$$

$$y^{\varphi} = \begin{bmatrix} 0.0140 & 0.6873 & 0.9060 \end{bmatrix}$$

$$\text{Sigmoid}(-4.2569) = 0.014$$

○ مقدار خالص ورودی به دروازه‌های فراموشی

$$net_{\varphi_j}(t) = \sum_m w_{\varphi_j m} y^m(t-1)$$

○ فعال‌سازهای دروازه‌های فراموشی

$$y^{\varphi_j}(t) = f_{\varphi_j}(net_{\varphi_j}(t))$$



شبکه عصبی LSTM: مثال ...

- مرحله ۴: مقدار خالص ورودی به سلول‌ها و همچنین حالت سلول‌ها را محاسبه کنید

○ مقدار خالص ورودی به سلول‌ها

$$((3.9151 * 1) + (0.9615 * 0) + (4.7583 * 0) + (4.3264 * 0) + (2.6357 * 0) + (2.6612 * 0) + (4.5521 * 0)) + ((0 * (-1.7355)) + (0 * (-3.1873)) + (0 * 8.1901)) = 3.9151$$

$$net_c = \begin{bmatrix} 3.9151 & -3.2348 & -3.0269 \end{bmatrix}$$

$$net_{c_j}(t) = \sum_m w_{c_j m} y^m(t-1)$$

○ حالت سلول‌ها

$$S_c = \begin{bmatrix} 1.9071 & -1.6403 & -1.3122 \end{bmatrix}$$

$$(0 * 0.014) + (0.9923 * 1.9218) = 1.907$$

$$s_{c_j}(t) = s_{c_j}(t-1)y^{\phi_j}(t) + y^{in_j}(t) g\left(net_{c_j}(t)\right)$$



شبکه عصبی LSTM: مثال ...

- مرحله ۵: مقدار خالص ورودی به دروازه‌های خروجی و فعال‌سازهای آنها را محاسبه کنید
- مقدار خالص ورودی به دروازه‌های خروجی

$$((4.7166 * 1) + (3.2706 * 0) + (-0.9194 * 0) + (-2.2420 * 0) + (1.5076 * 0) + (2.1366 * 0) + (2.0888 * 0)) + ((0 * (-1.0283)) + (0 * (-3.2926)) + (0 * 3.8964)) = 4.7166$$

$$net_{out} = \begin{bmatrix} 4.7166 & 1.0076 & -0.1894 \end{bmatrix}$$

$$net_{out_j}(t) = \sum_m w_{out_j m} y^m(t-1)$$

- فعال‌سازهای دروازه‌های خروجی

$$y^{out} = \begin{bmatrix} 0.9911 & 0.7326 & 0.4528 \end{bmatrix}$$

$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t))$$

$$\text{Sigmoid}(4.7166) = 0.9911$$



شبکه عصبی LSTM: مثال ...

$$0.9911 * (0.7414) = 0.7348$$

$$y^c = \begin{bmatrix} 0.7348 & -0.4946 & -0.2607 \end{bmatrix}$$

- مرحله ۶: خروجی سلول‌ها را محاسبه کنید

$$y^{c_j}(t) = y^{out_j}(t) h(s_{c_j}(t))$$

- مرحله ۷: مقدار خالص ورودی به نرون‌های لایه خروجی و فعال‌سازهای آن‌ها را محاسبه کنید

مقدار خالص ورودی به نرون‌های لایه خروجی

$$(0.7348 * (-2.6071)) + (-0.4946 * 2.4884) + (-0.2670 * 0.1496) = -3.1854$$

$$net_k(t) = \sum_m w_{km} y^m(t)$$

$$net_k = \begin{bmatrix} -3.1854 & 2.5813 & 2.0647 & -10.7316 & -10.7312 & -1.8352 & -2.6637 \end{bmatrix}$$

فعال‌سازهای نرون‌های لایه خروجی

$$y^k = \begin{bmatrix} 0.0397 & 0.9296 & 0.8874 & 2.1843e-05 & 2.1853e-05 & 0.1376 & 0.0652 \end{bmatrix}$$

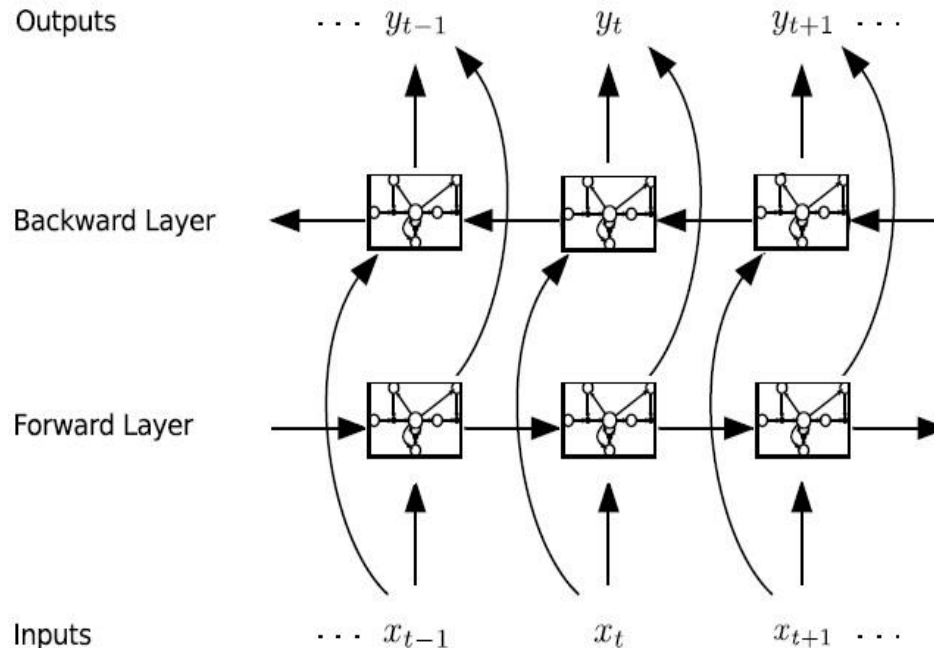
$$\text{Sigmoid}(-3.1854) = 0.0397$$

$$y^k(t) = f(net_k(t))$$

شبکه‌های عصبی LSTM: دوطرفه (Bidirectional) ...

○ 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.



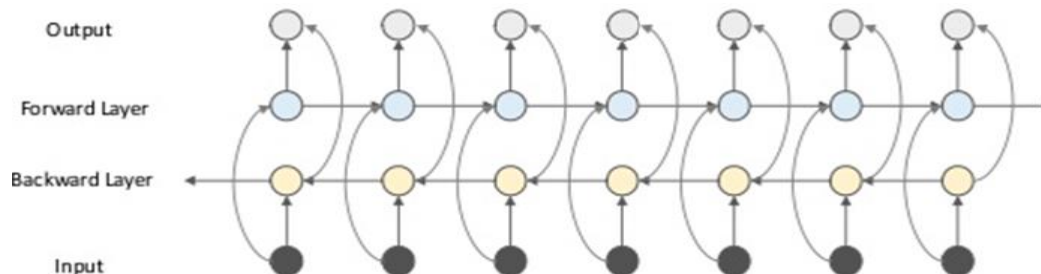
شبکه‌های عصبی LSTM: دوطرفه (Bidirectional)

○ ایده اصلی

هر دنباله ورودی در دو جهت زمانی رو به جلو و از انتها به دو لایه پنهان بازگشتی 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 Forward و $x_{T-(i-1)}$ را به لایه Backward ارسال کرده و مقدار هدف را بردار t_i قرار می‌دهیم.
- آموزش شبکه را با استفاده از الگوریتم مربوط به شبکه LSTM دنبال می‌کنیم

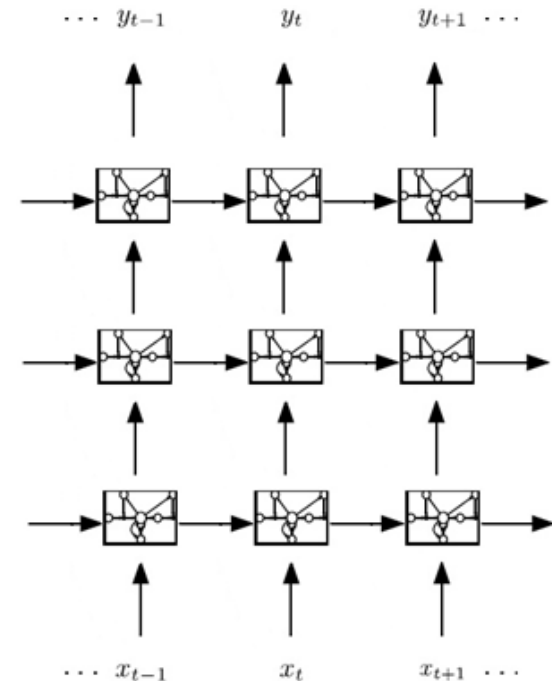
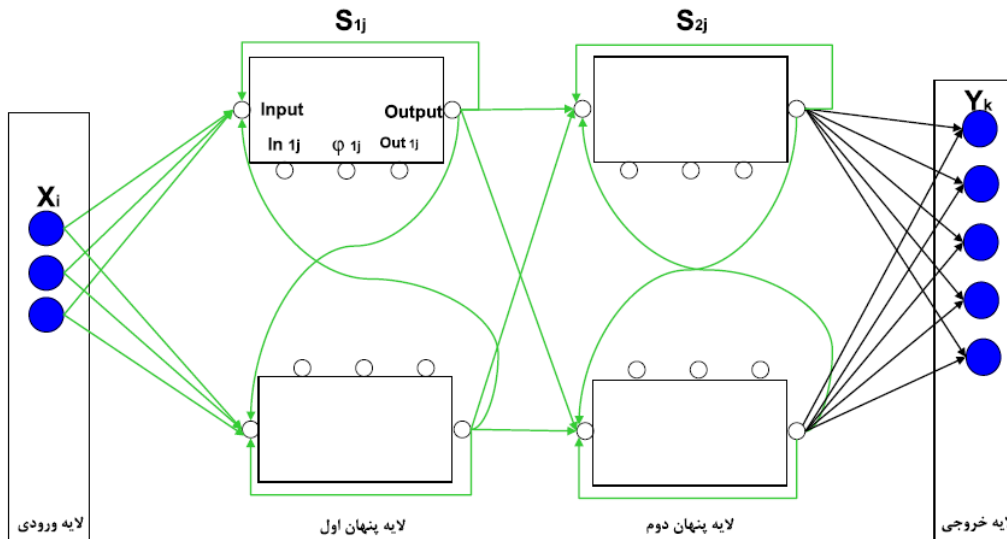
مقدار خالص رسیده به لایه خروجی جمع وزن‌دار مقدار خالص دو لایه Forward و Backward است



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 دروازه‌ای (GRU)

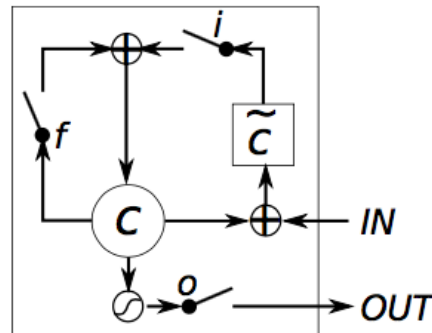
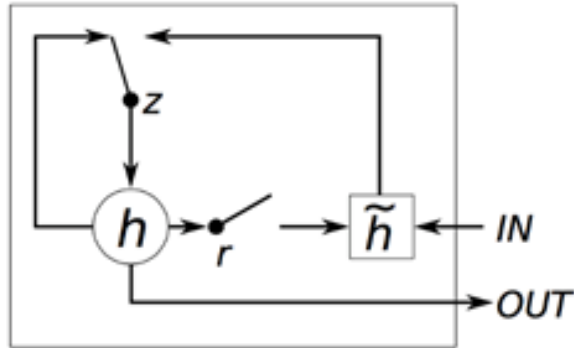
○ ساده شده LSTM استاندارد

• عدم وجود دروازه خروجی

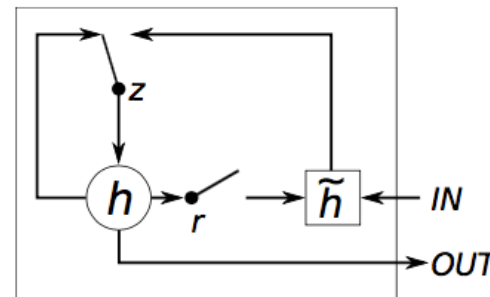
• دارای دو دروازه (LSTM دارای ۳ دروازه)

○ راه اندازی مجدد (reset) و بروزرسانی (update)

• عبور تمام مقدار سلول به خروجی یا ورودی سایر بلوک‌ها



(a) Long Short-Term Memory



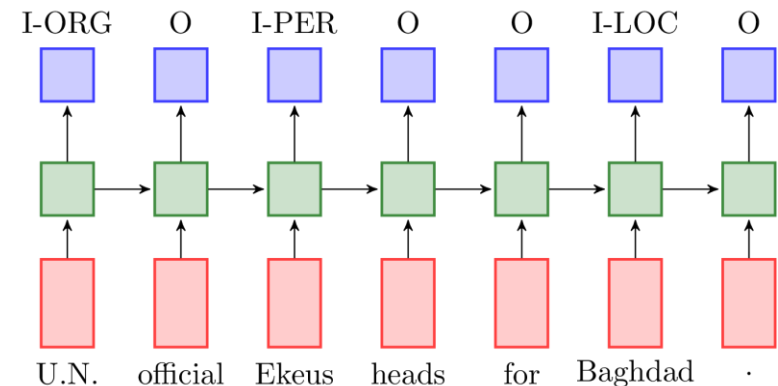
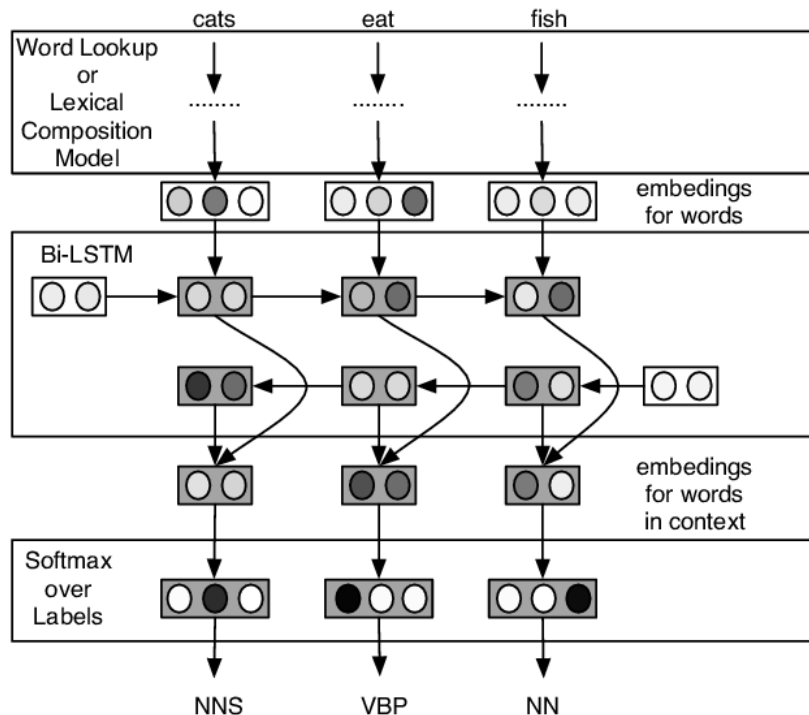
(b) Gated Recurrent Unit

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: کاربردها ...

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

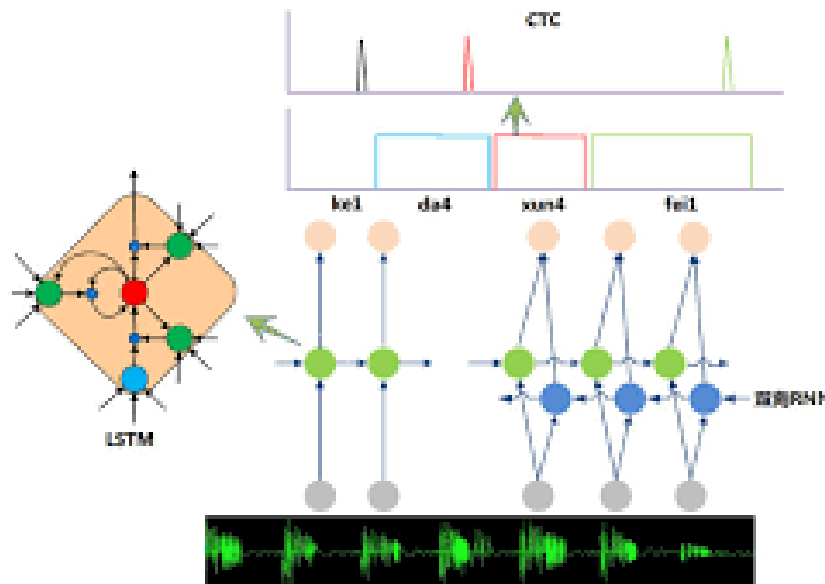
- ورودی: بردار یک کلمه (مثل Word Vector)
- خروجی: به تعداد برچسب‌ها (هر نرون یک برچسب)



شبکه عصبی LSTM: کاربردها ...

○ بازشناسی گفتار

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



شبکه عصبی LSTM: کاربردها ...

○ بازشناسی دست خط / نویسه‌های نوری (OCR)

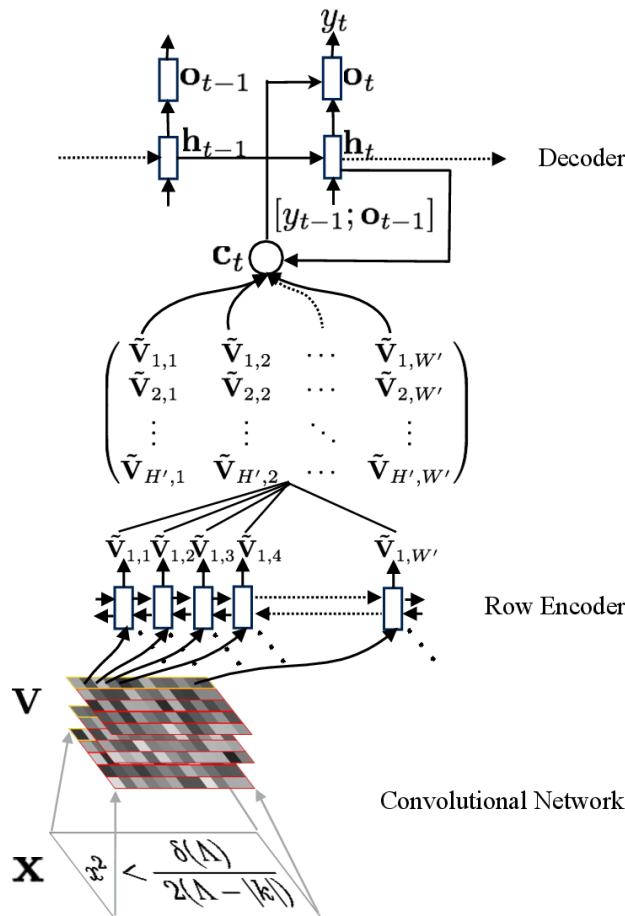
• ورودی: بردار مربوط به ویژگی یک فریم از تصویر

○ ویژگی‌های CNN

• خروجی: به تعداد واحدهای بازشناسی شونده

○ هر نرون یک کاراکتر

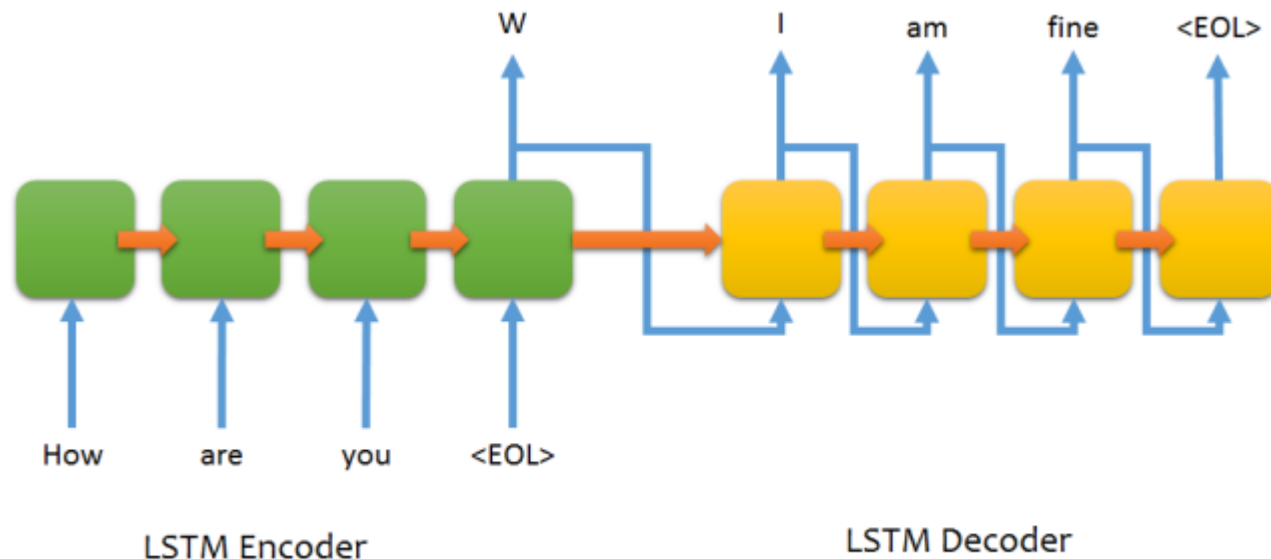
○ نیاز به تبدیل دنباله برچسب فریم‌ها به دنباله واج = CTC



شبکه عصبی LSTM: کاربردها ...

○ 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 واج‌های فارسی ...

○ داده‌ها: فارسی‌دات

- 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

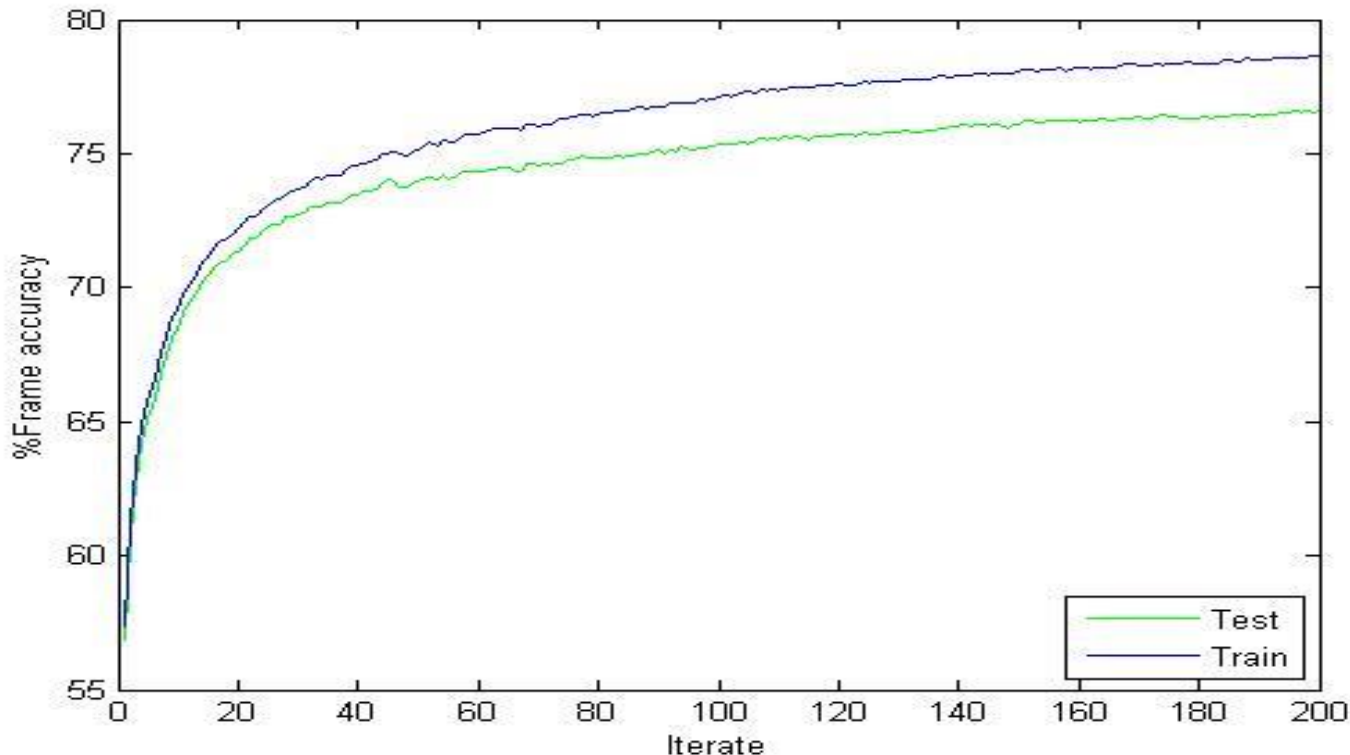
- نرون‌های لایه ورودی: ۳۹
- نرون‌های لایه خروجی: ۳۰ (تعداد واج‌های فارسی + سکوت)
- وزن‌های اولیه: تصادفی در بازه $[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}]$
- نرخ یادگیری: ۰/۰۰۰۳
- تعداد بلوک حافظه: ۱۵۰



شبکه عصبی LSTM: تشخیص واج‌های 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 واج‌های فارسی ...

○ داده‌ها: فارسی‌دات

- 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

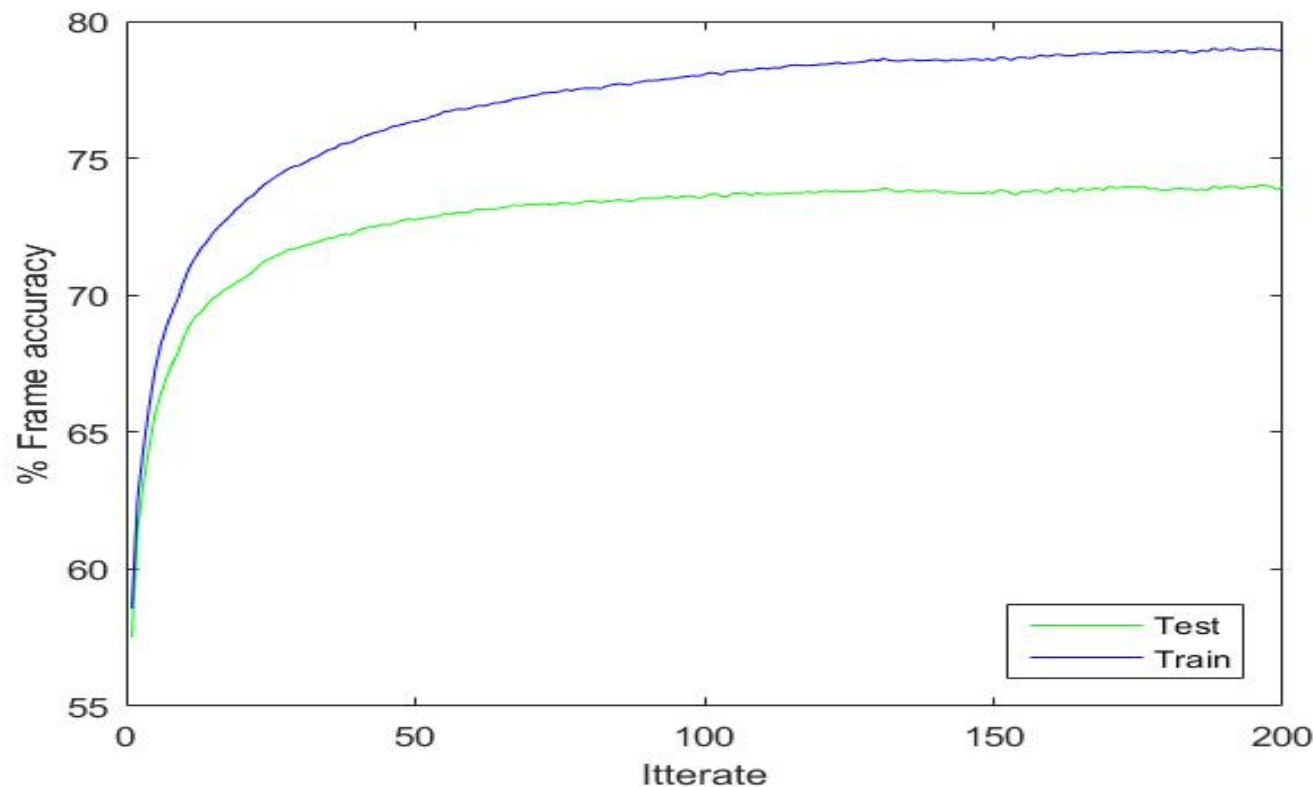
- نرون‌های لایه ورودی: ۳۹
- نرون‌های لایه خروجی: ۳۰
- وزن‌های اولیه: تصادفی در بازه $[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}]$
- نرخ یادگیری: ۰/۰۰۰۳
- تعداد بلوک حافظه: ۱۲۰



شبکه عصبی LSTM دوطرفه: تشخیص واج‌های فارسی ...

○ دقت به دست آمده روی فریم‌ها بعد از ۲۰۰ تکرار

- داده‌های آموزش: ۷۹.۰۱٪
- داده‌های آزمون: ۷۳.۸۴٪





شبکه عصبی LSTM دوطرفه: تشخیص واج‌های فارسی

○ دقت روی واج

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

