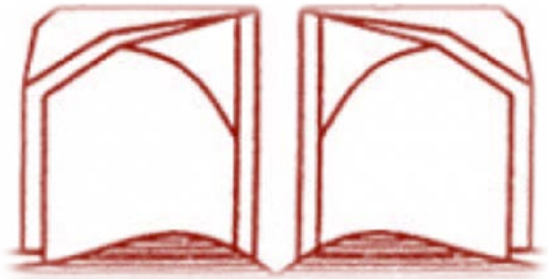


روش‌های یادگیری ماشین در پردازش زبان طبیعی

شبکه عصبی مصنوعی

هادی ویسی

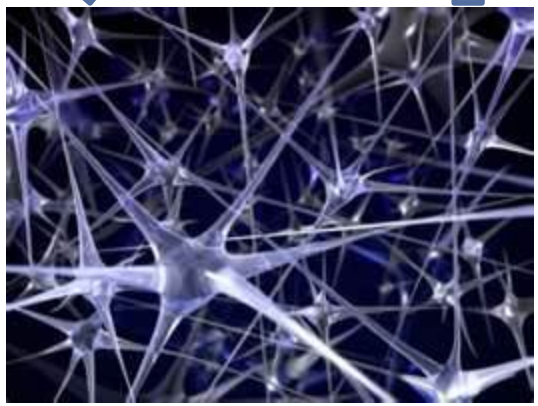
h.veisi@ut.ac.ir

دانشگاه تهران – دانشکده علوم و فنون نوین

فهرست

شبکه عصبی؟

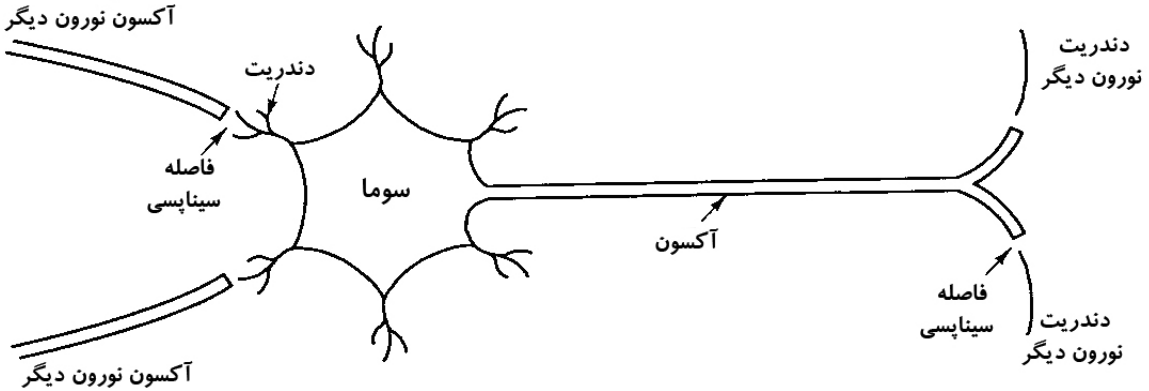
- مغز = شبکه‌ای بسیار بزرگ از عصب‌ها (نرون‌ها)
  - ۱۰۰.۰۰۰.۰۰۰.۰۰۰ نرون
  - ۱۰.۰۰۰ اتصال برای هر نرون
- شبکه عصبی مصنوعی = شبیه‌سازی شبکه عصبی طبیعی



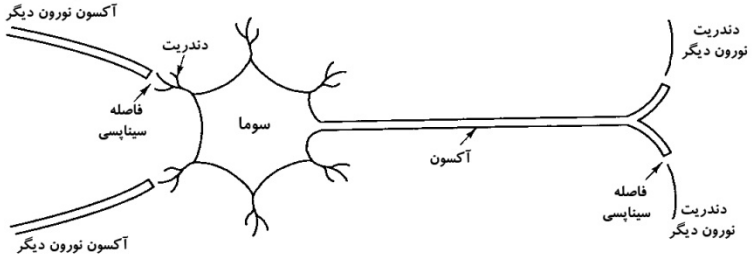
شبکه عصبی طبیعی . . .

- عنصر پردازشگر تشکیل‌دهنده یک شبکه عصبی مصنوعی
  - نرون (Neuron) = عصب طبیعی (سلول مغزی)

- سه جزء تشکیل دهنده یک نرون طبیعی
  - دندریت‌ها (Dendrite): دریافت سیگنال از سایر نرون‌ها
  - سوما (Soma) = بدنهٔ 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 فاصلهٔ سیناپسی (Synaptic Gap)
- عمل شیمیایی انتقال دهنده، سیگنال ورودی را تغییر می‌دهند (تضعیف/تقویت سیگنال)
- سوما سیگنال‌های ورودی به سلول را جمع می‌بندد
- زمانی‌که 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 نتیجهٔ غلظت‌های مختلف یون‌ها در اطراف پوشش آکسون نرون («مادهٔ سفید» مغز) می‌باشد.
  - یون‌ها = پتاسیم، سدیم و کلرید
- سیگنال‌ها به صورت ضربه‌های الکتریکی هستند

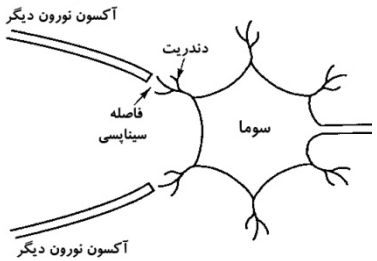
شبکه عصبی مصنوعی . . .

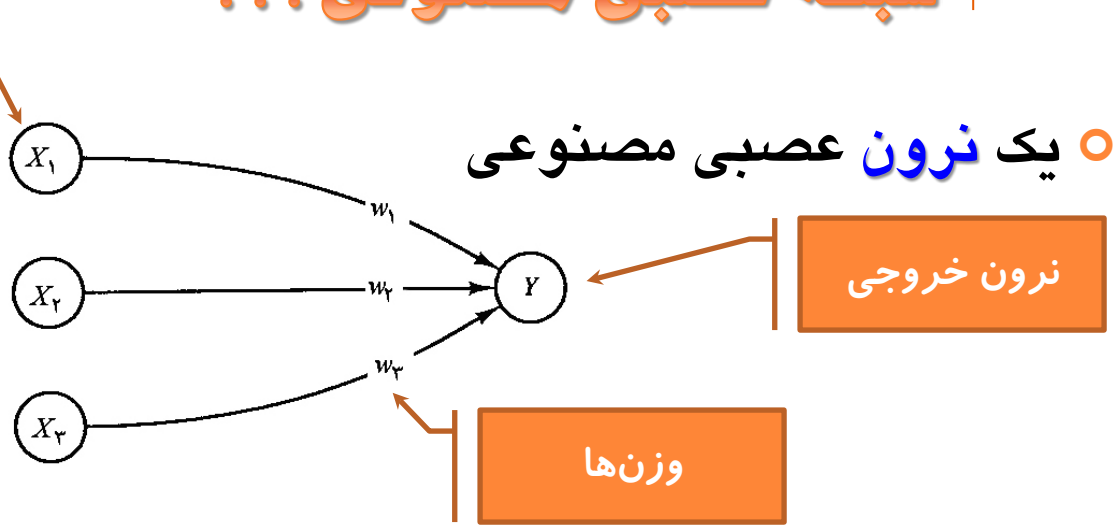
- شبکه عصبی مصنوعی [Artificial Neural Network]
  - یک سیستم پردازش اطلاعات با ویژگی‌های مشترکی با شبکه‌های عصبی طبیعی
  - تعمیم یافتهٔ مدل‌های ریاضی تشخیص انسان بر اساس زیست‌شناسی عصبی

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

شبکه عصبی مصنوعی . . .

- یک نرون عصبی مصنوعی





- فعال‌سازی‌ها یا سیگنال‌های خروجی نرون‌های ورودی به ترتیب x1، x2 و x3 هستند
- ورودی شبکه به نرون Y، حاصل‌جمع وزن‌دار سیگنال‌های ورودی و وزن‌هاست:

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

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

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

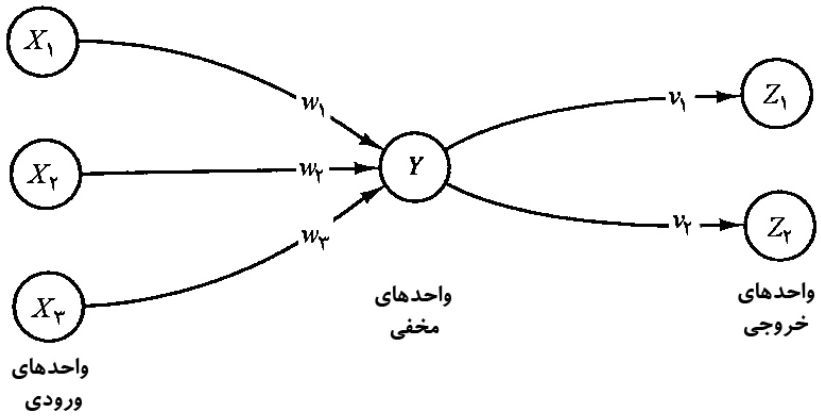
$$f(x) = \begin{cases} 1 & if \quad x > \theta \\ 0 & if \quad x < \theta \end{cases} \qquad f(x) = \frac{1}{1+\exp(-x)}$$

- تابع پله
- تابع سیگموید (Sigmoid)

شبکه عصبی مصنوعی . . .

- یک **شبکه** عصبی مصنوعی

- سه لایه: ورودی، مخفی و خروجی
- دو دسته وزن: Wها و Vها

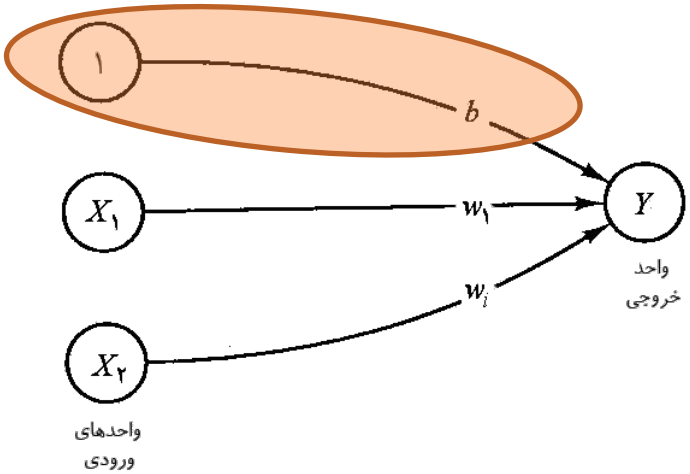


- در یک شبکه یک نرون می‌تواند ورودی‌های مختلفی را از چند نرون دریافت کند

شبکه عصبی مصنوعی . . .

بایاس

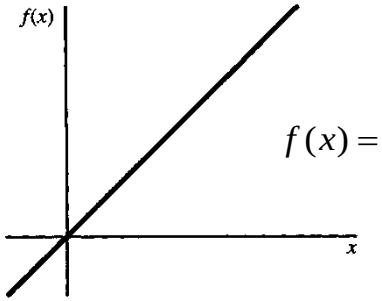
- در ورودی شبکه عصبی، علاوه بر ورودی‌های موردنظر، یک ورودی ثابت با مقدار ۱ نیز داشته باشیم.



$$y\_in = 1 \times b + w_1 x_1 + w_2 x_2 = b + \sum_i w_i x_i$$

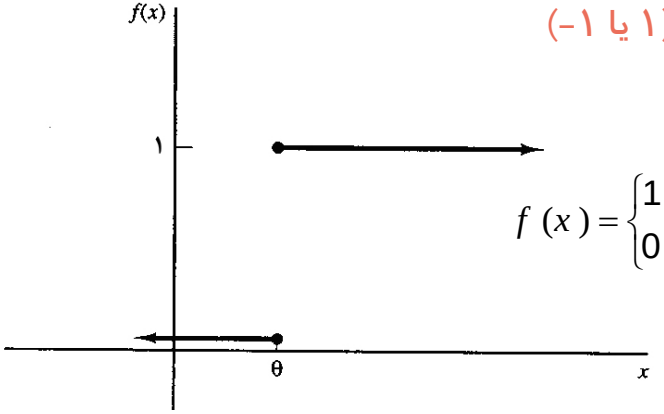
شبکه عصبی مصنوعی . . .

- توابع فعال‌سازی متداول . . .
  - تابع همانی (Identity Function)
    - برای واحدهای ورودی



$$f(x) = x \qquad for\ all\ x$$

  - 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-)



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

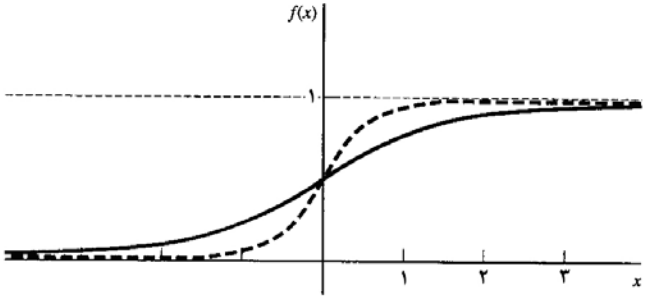
شبکه عصبی مصنوعی . . .

توابع فعال‌سازی متداول

- **توابع سیگموید** (Sigmoid Functions)
- **منحنی‌هایی به شکل S**
  - استفاده در شبکه‌های عصبی پس‌انتشار (نیاز به مشتق‌گیری)

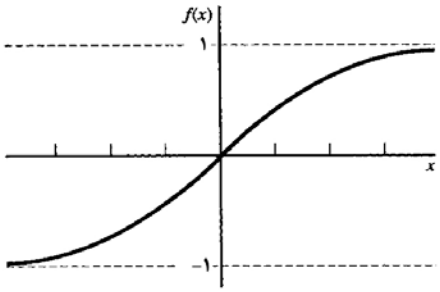
$$f(x) = \frac{1}{1+\exp(-\sigma x)}$$

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



- سیگموید دودویی- تابع لجستیک (Logistic Function)
  - دامنهٔ ۰ تا ۱، مقادیر مطلوب خروجی یا دودویی است و یا بین ۰ و ۱ است

- سیگموید دوقطبی- شبیه به تابع تانژانت هیپربولیک (Hyperbolic Tangent Function)
  - دامنهٔ ۱- تا ۱



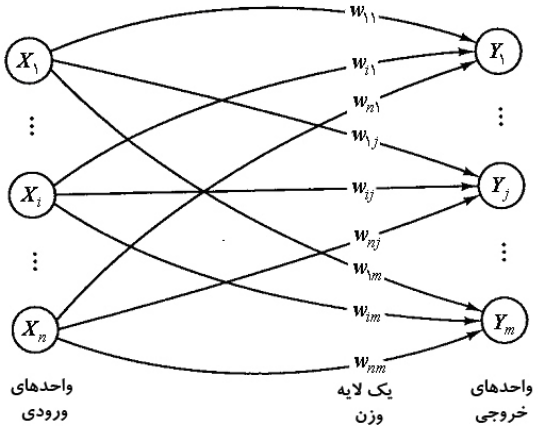
$$g(x) = 2f(x) - 1 = \frac{2}{1+\exp(-\sigma x)} - 1 = \frac{1-\exp(-\sigma x)}{1+\exp(-\sigma x)}$$

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

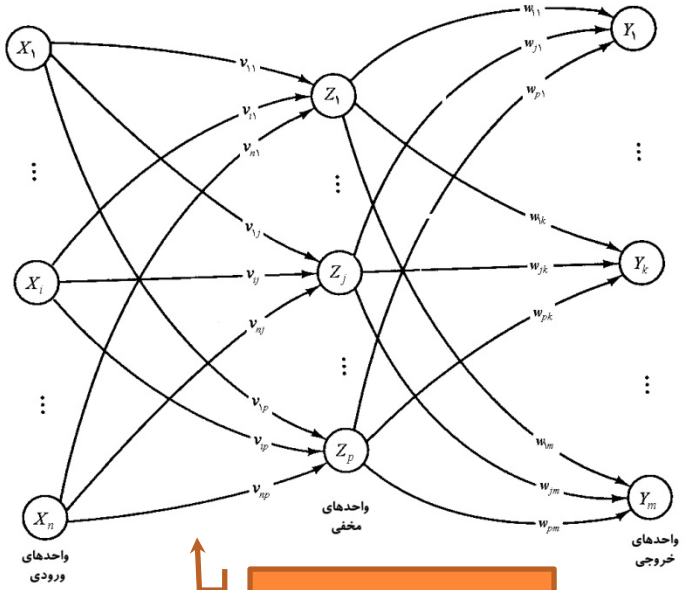
شبکه عصبی مصنوعی . . .

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



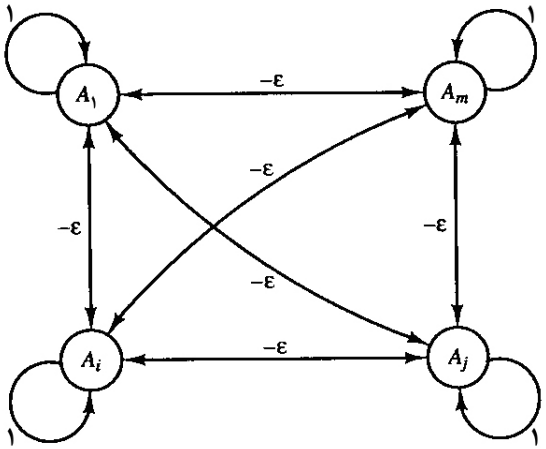
شبکه دولایه

شبکه عصبی مصنوعی . . .

ساختارهای رایج

- شبکهٔ بازگشتی (Recurrent)، مسیرهای بستهٔ سیگنال از یک واحد به خودش وجود دارد

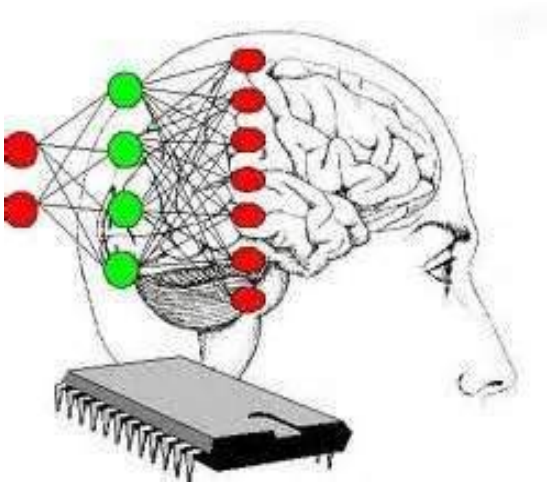
- شبکهٔ رقابتی: واحدهای آن کاملاً به هم مرتبط‌اند



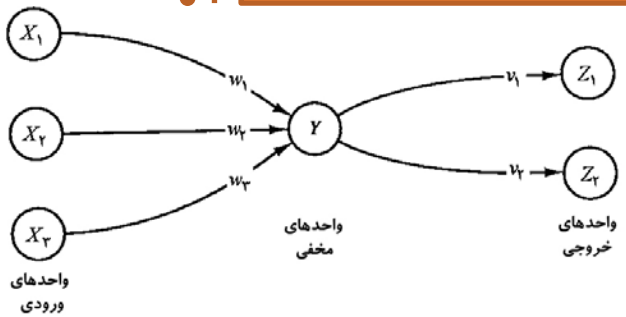
شبکه عصبی مصنوعی . . .

ویژگی‌های مشخص کننده یک شبکهٔ 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



- معماری ۳ لایه
- وزن‌های W و V
- تابع فعال‌سازی برای Y و Z1 و Z2



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)

- دههٔ ۷۰- سال‌های خاموش
  - ۱۹۷۲- اولین کار کوهونن (از هلسینکی)، روی شبکه‌های عصبی حافظهٔ پیوندی

شبکه عصبی مصنوعی: تاریخچه . . .

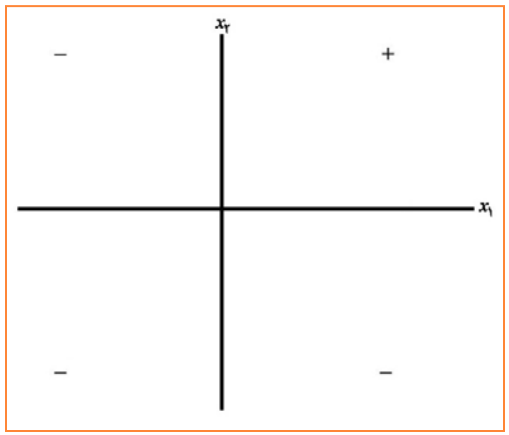
- دهه ۸۰- شکوفایی شبکه‌های عصبی
  - ۱۹۸۲- شبکه‌های هاپفیلد (جزو شبکه‌های حافظهٔ انجمنی)
  - ۱۹۸۲- نگاشت‌های خودسازمانده کوهونن (SOM)
  - ۱۹۸۵- الگوریتم پس‌انتشار خطا برای آموزش شبکه‌های چندلایه (MLP)
  - ۱۹۸۵- ماشین بولتزمن: تغییر وزن‌ها براساس تابع چگالی احتمال
  - ۱۹۸۷- شبکه‌های نظریهٔ نوسان وفقی (ART) توسط کارپنز و گراس‌برگ
  - ۱۹۸۷- شبکه Neocognitron توسط فوکوشیما برای بازشناسی نویسه‌ها
  - مطالعات ریاضیاتی و زیست‌شناختی
  - پیاده‌سازی سخت‌افزاری شبکه عصبی

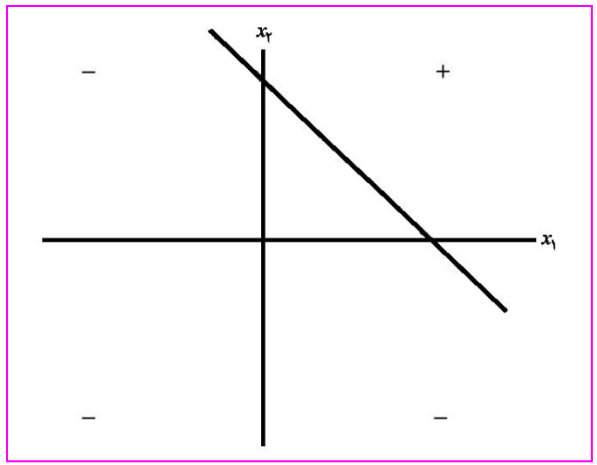
شبکه عصبی مصنوعی: تاریخچه

- دهه ۹۰ – دهه کاربرد
  - به کار گیری 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 . . .

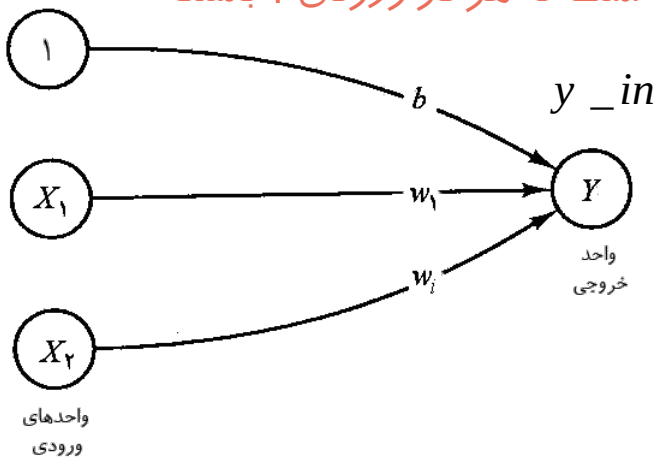
INPUT(x_1, x_2)	OUTPUT
(1, 1)	+1
(1 , −1)	−1
(−1 , 1)	−1
(−1 , −1)	−1





مثال: تابع AND

- دو ورودی (دوقطبی) و یک خروجی (دوقطبی)
  - وقتی خروجی ۱ است که هر دو ورودی ۱ باشند



$$y\_in = 1 \times b + w_1 x_1 + w_2 x_2$$

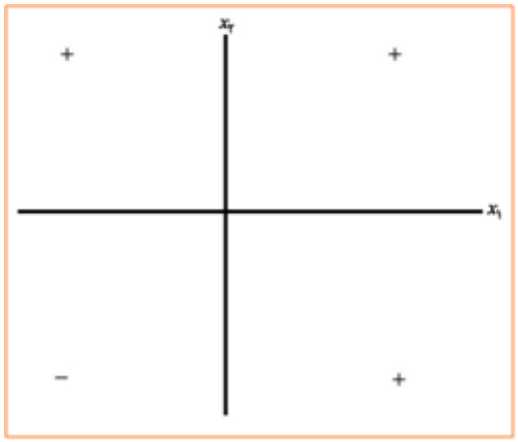
$$f(x) = \begin{cases} 1 & if \quad x > \theta \\ 0 & if \quad x < \theta \end{cases}$$

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

- پاسخ $b = -1,\ w_1 = 1, w_2 = 1$

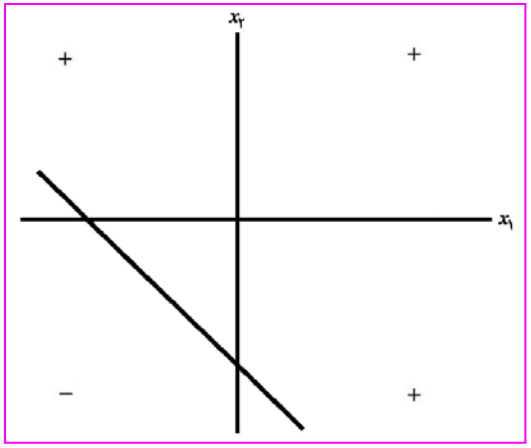
$$x_2 = -x_1 + 1$$

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



- مرز تصمیم‌گیری

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

$$x_2 = -x_1 - 1$$

- 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، آدالاین 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

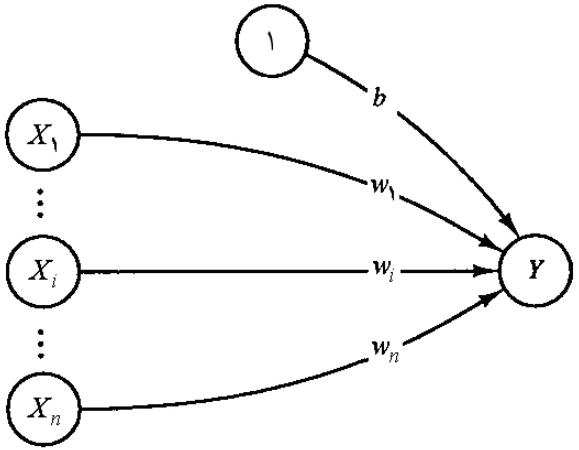
شبکه عصبی پرسپترون: ساختار . . .

ساختاراولیه

- **مدل تقریبی شبکیه چشم**
  - سه لایه نرون (واحدهای حسی، واحدهای پیونددهنده و واحد پاسخ)
    - فقط وزن‌های بین لایه‌های دوم و سوم آموزش داده می‌شود
    - خروجی واحدهای پیونددهنده به واحدهای پاسخ یک بردار دودویی است
    - عملاً 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 ۱+
  - متعلق نبودن با پاسخ ۱-

شبکه عصبی پرسپترون: الگوریتم . . .

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

  تعیین نرخ یادگیری $0 < \alpha \leq 1$ (مقدار ۱)
- مرحلهٔ ۱ – تا زمانی‌که شرایط توقف برقرار نیست، مراحل ۲ تا ۶ را انجام دهید
- مرحلهٔ ۲ – انجام مراحل ۳ تا ۵ برای هر جفت داده آموزش $s:t$
- مرحلهٔ ۳ – فعال‌سازی‌های واحدهای ورودی را مشخص کنید: $x_i = s_i$
- مرحلهٔ ۴ – پاسخ واحد خروجی را محاسبه کنید:

الگوریتم تکراری

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

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

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

Hadi Veisi (h.veisi@ut.ac.ir)

شبکه عصبی پرسپترون: الگوریتم . . .

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

به‌روز کردن مشروط وزن‌ها

اگر $y \neq t$ است، آنگاه:

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

در غیراین‌صورت:

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

- مرحلهٔ ۶- شرایط توقف را آزمایش کنید:
  - اگر در مرحلهٔ ۲ هیچ وزنی تغییر نکرد، الگوریتم را متوقف کنید، در غیراین‌صورت ادامه دهید.

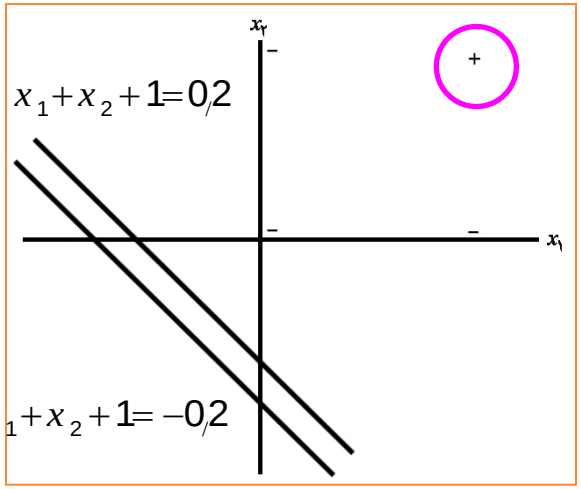
خطا = برابر نبودن پاسخ شبکه و مقدار هدف

نرخ یادگیری

شبکه عصبی پرسپترون: مثال . . .

- تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .
  - دودویی: مقادیر صفر و یک — دوقطبی: مقادیر ۱+ و ۱−
  - وزن‌های اولیه و بایاس را صفر؛ نرخ اولیه یادگیری = ۱؛ آستانه =۰.۲
  - ارائه ورودی اول

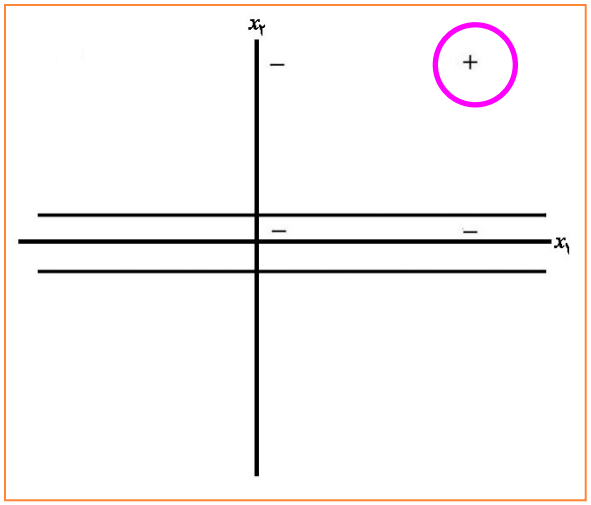
INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1



شبکه عصبی پرسپترون: مثال . . .

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

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	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

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	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

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	−1
0	0	1	−1	−1	−1	0	0	0	0	0	−1

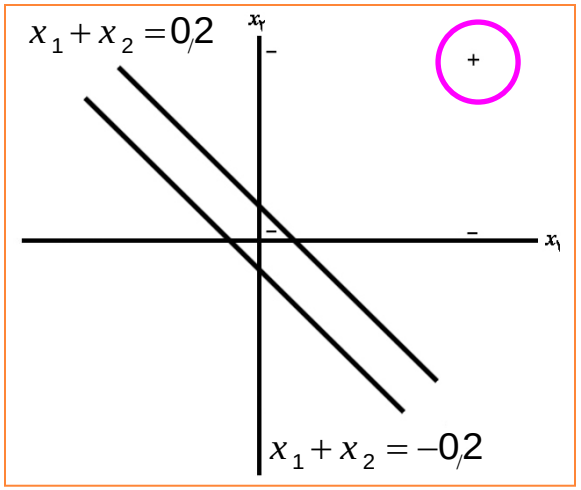
کامل شدن اولین دور آموزش (Epoch)

شبکه عصبی پرسپترون: مثال . . .

- تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .
  - نیاز به تکرار؟؟ صحیح نبودن پاسخ برای اولین الگوی ورودی
  - تکراری بودن فرآیند آموزش (Iterative)

- دومین دور آموزش –ارائه اولین ورودی

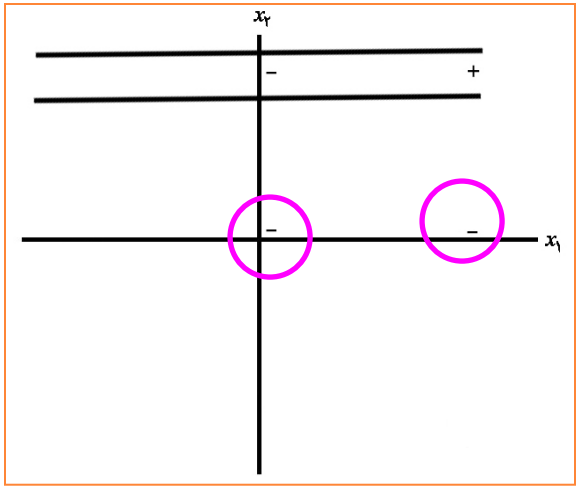
INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	–1
1	1	1	–1	–1	1	1	1	1	1	1	0



شبکه عصبی پرسپترون: مثال . . .

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

INPUT			NET	OUT	TARGET	WEIGHT 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

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	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

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	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)

Hadi Veisi (h.veisi@ut.ac.ir)

شبکه عصبی پرسپترون: مثال . . .

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

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	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 . . .

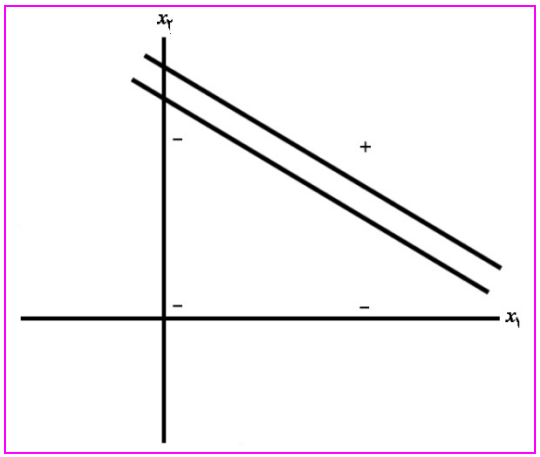
- تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .
  - پنجمین، ششمین، .... دور آموزش

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

  - نهمین دور آموزش

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

  - دهمین دور آموزش
    - عدم تغییر وزن‌ها = توقف الگوریتم
    - همگرایی وزن‌ها



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

شبکه عصبی پرسپترون: مثال . . .

- تابع AND با ورودی‌ها و هدف‌های دوقطبی
  - آستانه، بایاس و وزن‌های اولیهٔ برابر با صفر؛ نرخ یادگیری برابر با ۱

- دور اول آموزش

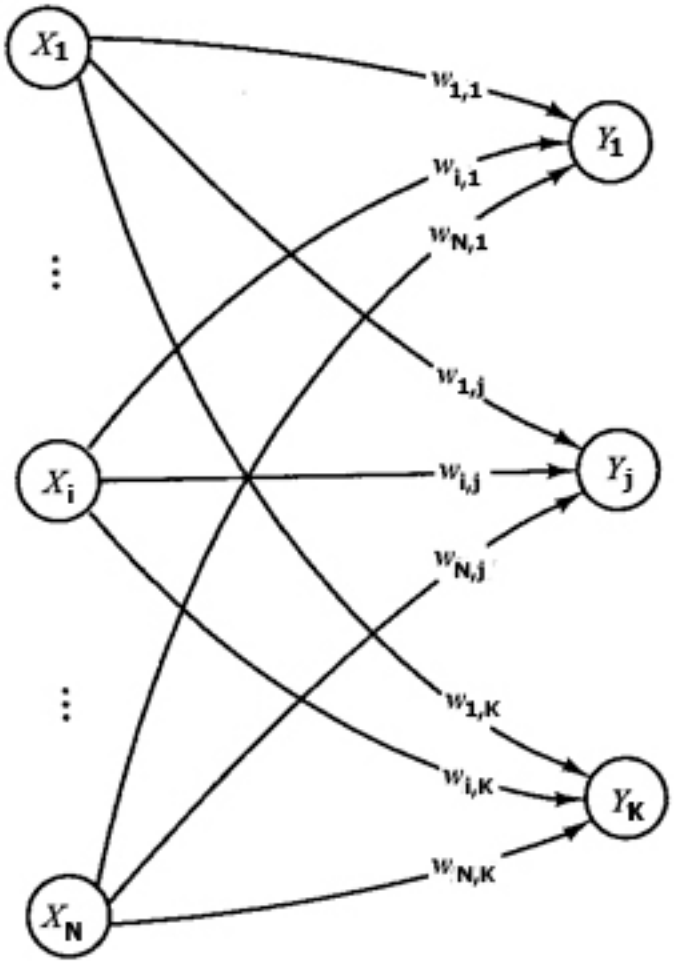
INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
1	−1	1	1	1	−1	−1	1	−1	0	2	0
−1	1	1	2	1	−1	1	−1	−1	1	1	−1
−1	−1	1	−3	−1	−1	0	0	0	1	1	−1

- دور دوم 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

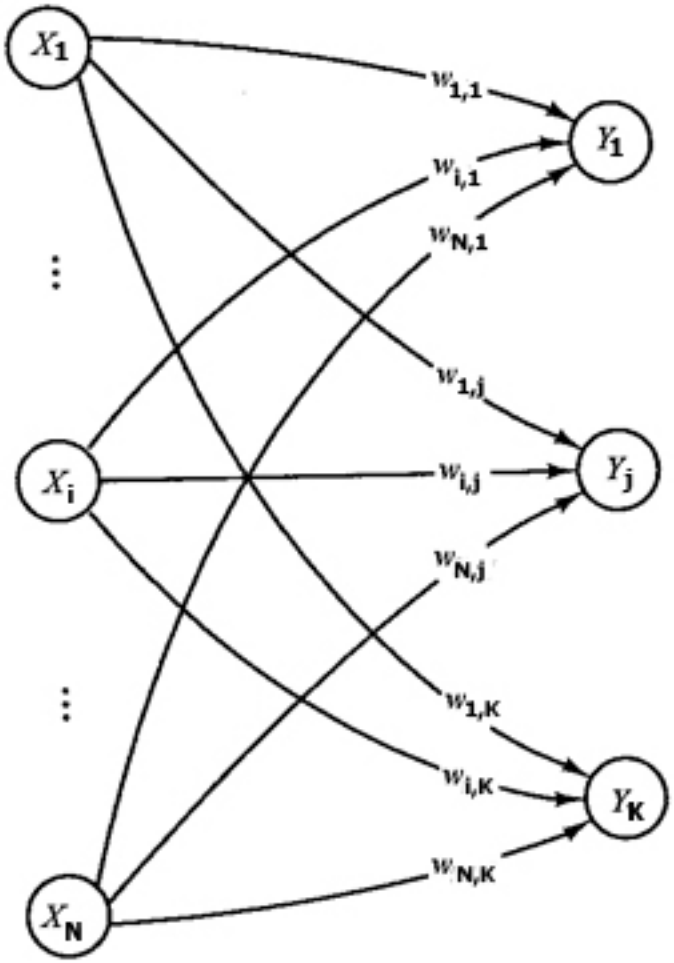
- بهبود نتایج با تغییر نمایش دودویی به دوقطبی

شبکه عصبی پرسپترون: مثال . . .



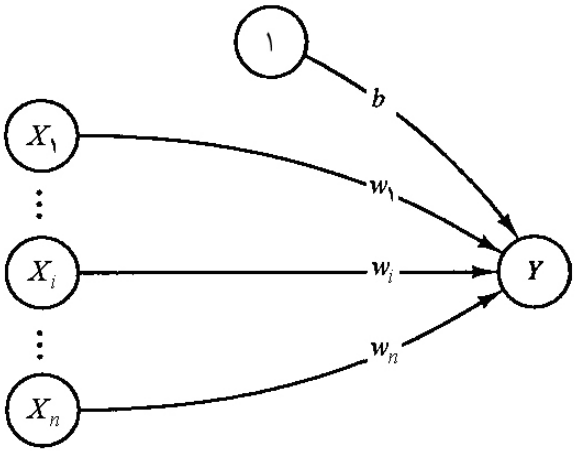
- تشخیص عنوان (موضوع) متن
  - تعداد K دسته (موضوع) $C = \{c_1, c_2, \ldots, c_K\}$
    - تعداد K نرون خروجی
  - داده = سند متنی
    - تبدیل هر سند به یک بردار ویژگی N بعدی
      - تعداد N نرون ورودی
  - آموزش
    - داده = تعداد M سند دارای برچسپ
    - خروجی: وزن‌های شبکه = یک ماتریس $N*K$ (مدل)
  - آزمون
    - ورودی: یک سند با عنوان نامشخص
      - تبدیل سند به یک بردار N بعدی و دادن آن به شبکه
    - خروجی: نرونی (دسته ای) که مقدار بزرگ‌تر دارد

شبکه عصبی پرسپترون: مثال . . .



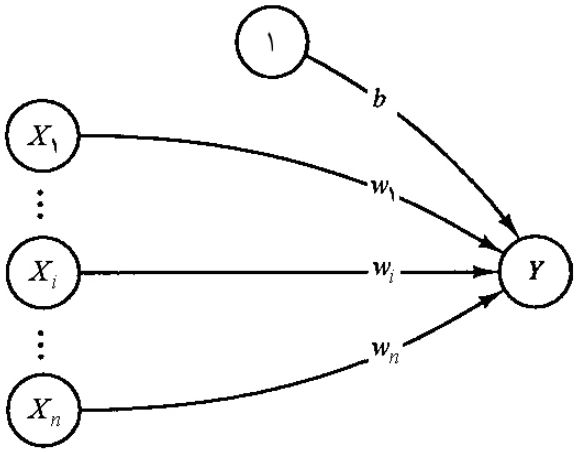
- تشخیص زبان
  - تعداد K دسته (زبان) $C = \{c_1, c_2, \ldots, c_K\}$
    - تعداد K نرون خروجی
  - داده = سند متنی یا سند صوتی
    - تبدیل هر سند به یک بردار ویژگی N بعدی
      - تعداد N نرون ورودی

شبکه عصبی پرسپترون: مثال . . .



- تشخیص جنسیت
  - تعداد ۲ دسته (زن و مرد) $C = \{c_1, c_2\}$
    - تعداد یک نرون خروجی (صفر و یک)
      - می توان 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)
      - می توان 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: همگرایی قانون یادگیری

قضیه

- اگر بردار وزن w^* وجود داشته باشد به‌طوری‌که برای تمام pها داشته باشیم:

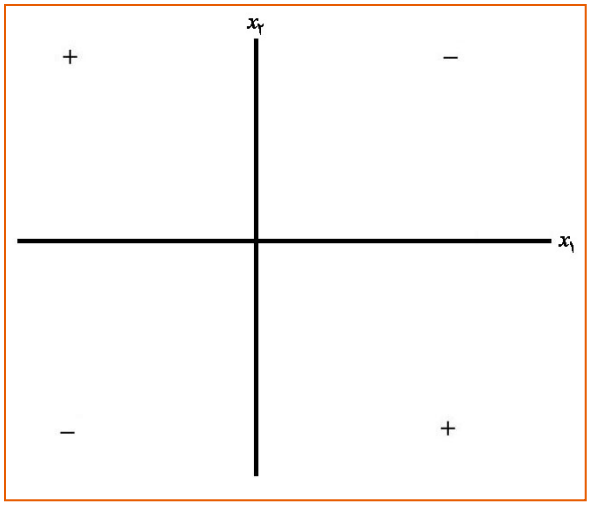
$$f(x(p).\ w^*) = t(p)$$

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

- p= تعداد بردارهای ورودی آموزش
- x(p) = بردارهای ورودی آموزش
- t(p)= مقدار هدف معادل بردارهای ورودی آموزش (دوقطبی)
- f = تابع فعال‌سازی خروجی

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

شبکه عصبی پرسپترون

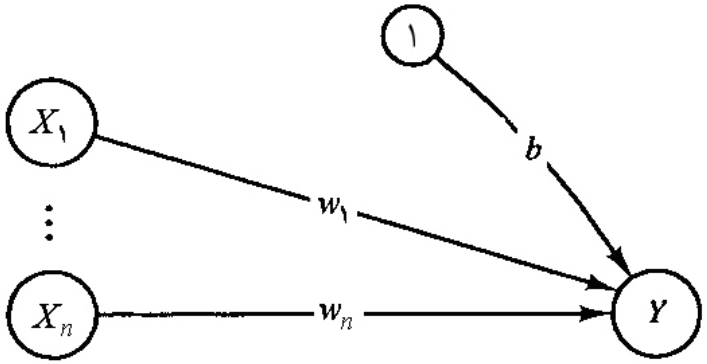


INPUT(x_1, x_2)	OUTPUT
(1, 1)	−1
(1 , −1)	+1
(−1 , 1)	+1
(−1 , −1)	−1

- **مثال: تابع XOR**
  - **دو ورودی (دوقطبی) و یک خروجی (دوقطبی)**
    - وقتی خروجی ۱ است که فقط یکی از ورودی‌ها ۱ باشد
  - **حل؟**
    - الگوریتم همگرا نمی شود (جواب درست نمی دهد)
  - **فضای داده‌های ورودی به صورت خطی جدایی‌پذیر (Linearly Spreadable) نیست.**
    - هیچ خط مستقیم نمی‌تواند نقاط مثبت و منفی را جدا کند: پرسپترون قادر به یافتن پاسخ نیست

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 وروی و خروجی
  - تابع فعال‌سازی خروجی = تابع همانی
  - ساختار مشابه با سایر شبکه‌های قبلی
    - چند ورودی
    - بایاس = ورودی برابر با ۱



شبکه آدالاین: الگوریتم . . .

- مرحلهٔ ۰- مقداردهی اولیه به وزن‌ها (مقادیر تصادفی کوچک)
  مقداردهی به نرخ یادگیری
- مرحلهٔ ۱- تا زمانی‌که شرایط توقف برقرار نیست، مراحل ۲ تا ۶ را انجام دهید.
- مرحلهٔ ۲- برای هر جفت آموزش دوقطبی $s:t$ مراحل ۳ تا ۵ را انجام دهید.
- مرحلهٔ ۳- فعال‌سازی‌های واحدهای ورودی را مشخص کنید: $x_i = s_i \qquad i = 1,\ldots,n$
- مرحلهٔ ۴- مقدار ورودی شبکه را به واحد خروجی محاسبه کنید: $y\_in = b + \sum_i x_i w_i$
- مرحلهٔ ۵- مقادیر وزن‌ها و بایاس را به‌روز کنید:
$$\begin{cases} b(new) = b(old) + \alpha.(t - y\_in) \\ w_i(new) = w_i(old) + \alpha.(t - y\_in).x_i \end{cases}$$
- مرحلهٔ ۶- شرایط توقف را آزمایش کنید:
  اگر بزرگ‌ترین تغییر وزنی که در مرحلهٔ 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

x_1	x_2	t
1	1	1
1	0	–1
0	1	–1
0	0	–1

- شبکه بعد از آموزش

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

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

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

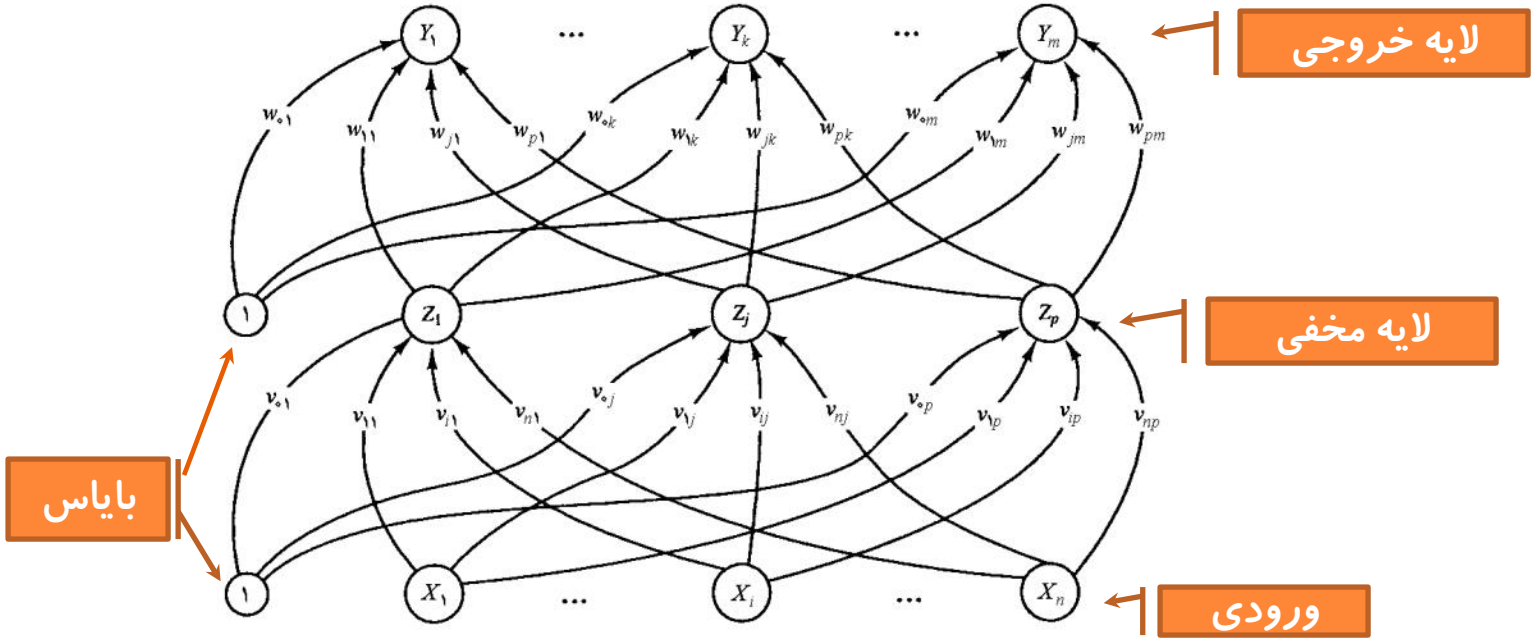
$$e = E\{(\hat{t} - t)^2\} = \sum_{p=1}^{4} [\{x_1(p).w_1 + x_2(p).w_2 + w_0\} - t(p)]^2$$

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 آدالاین
    - روش کاهش گرادیان برای به حداقل رساندن کل مربعات خطای خروجی
  - **(از) مهم‌ترین و پرکاربردترین شبکه(های) عصبی**

شبکه عصبی پرسپترون چندلایه (ساختار) . . .

- 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

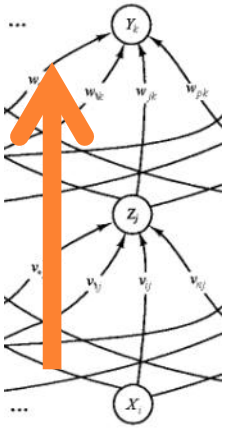
- مبنای ریاضی الگوریتم پس‌انتشار = بهینه‌سازی کاهش گرادیان (Gradient Descent)
  - گرادیان (شیب) یک تابع = نمایانگر جهتی که تابع در آن سریع‌تر افزایش می‌یابد
  - شیب با علامت منفی = جهتی نشان دهنده کاهش سریع‌تر آن تابع
  - در اینجا تابع مورد نظر = تابع خطای شبکه
  - متغیرهای مورد نظر = وزن‌های شبکه

شبکه عصبی پرسپترون چندلایه (الگوریتم آموزش) . . .

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

پیش‌خور

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

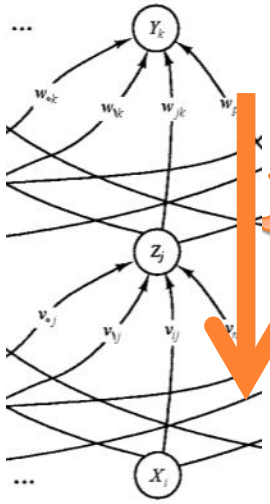


$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \qquad z_j = f(z\_in_j)$$

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

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

Hadi Veisi (h.veisi@ut.ac.ir)



شبکه عصبی پرسپترون چندلایه (الگوریتم آموز

پس‌انتشار خطا

- مرحلهٔ 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 لایهٔ قبلی (لایه مخفی)

- مرحلهٔ 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

- مرحلهٔ ۸- به‌روز کردن وزن‌ها و بایاس‌های واحدهای خروجی

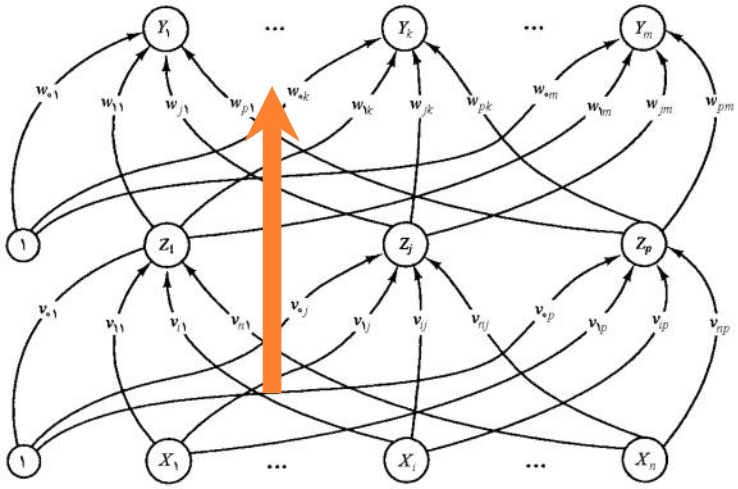
$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

به‌روز کردن وزن‌ها و بایاس‌های واحدهای مخفی

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

- مرحلهٔ ۹- شرایط توقف را بررسی کنید.

شبکه عصبی پرسپترون چندلایه (کاربرد) . . .



بعد از آموزش

- فقط مرحلهٔ پیش‌خور مورد نیاز است

- مرحلهٔ ۰: مقادیر وزن‌های شبکه را با استفاده از الگوریتم آموزش تعیین کنید.
- مرحلهٔ ۱: برای هر بردار ورودی، مراحل ۲ تا ۴ را انجام دهید.
- مرحلهٔ ۲: برای تمام نرون‌های ورودی، فعال‌سازی واحد ورودی را تعیین کنید،
- مرحلهٔ ۳: برای واحدهای مخفی:

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

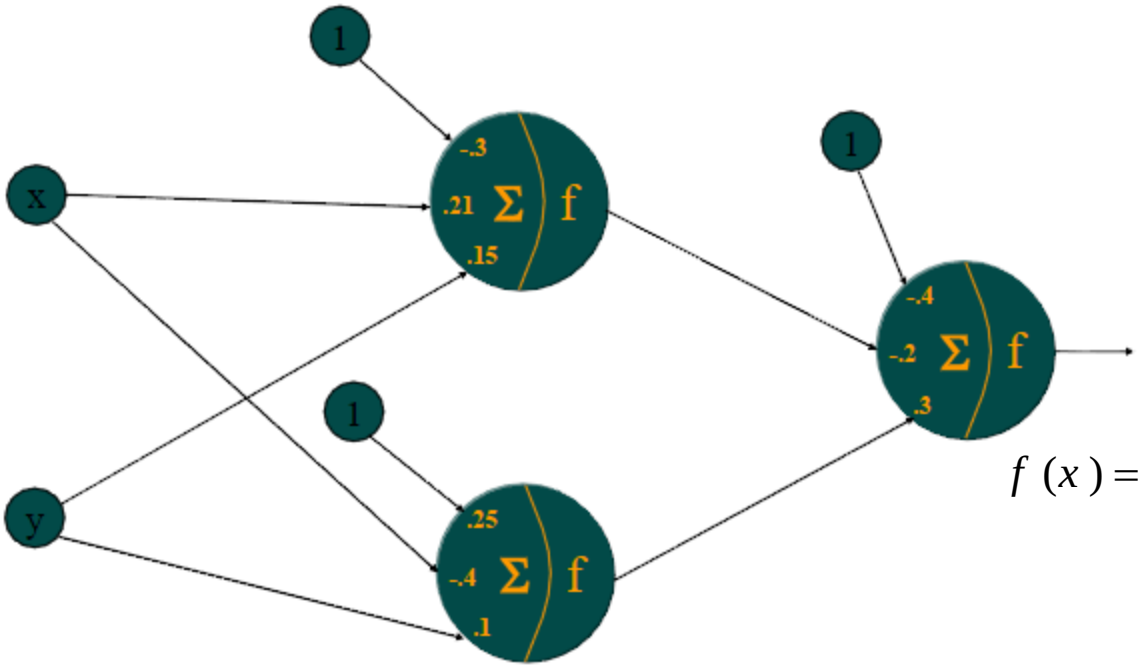
- مرحلهٔ ۴: برای واحدهای خروجی:

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

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)

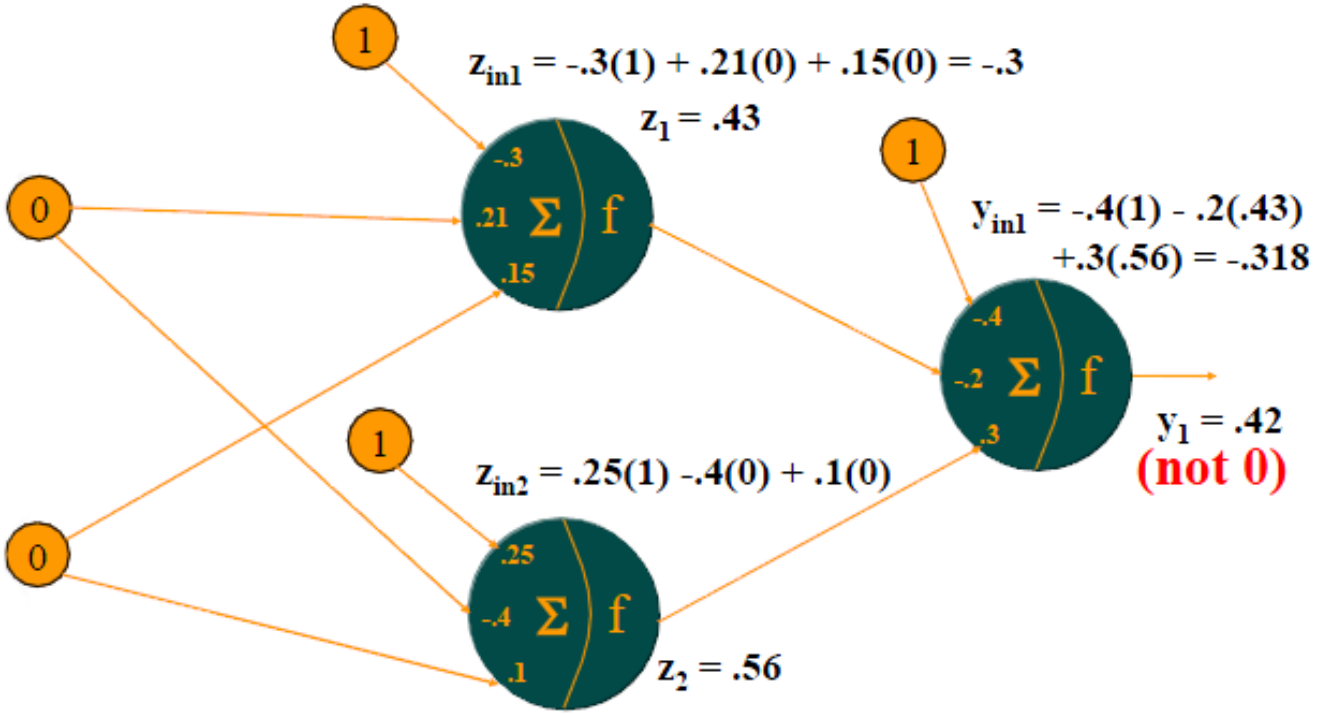


$$f(x) = \frac{1}{1+\exp(-x)}$$

شبکه عصبی پرسپترون چندلایه (مثال) . . .

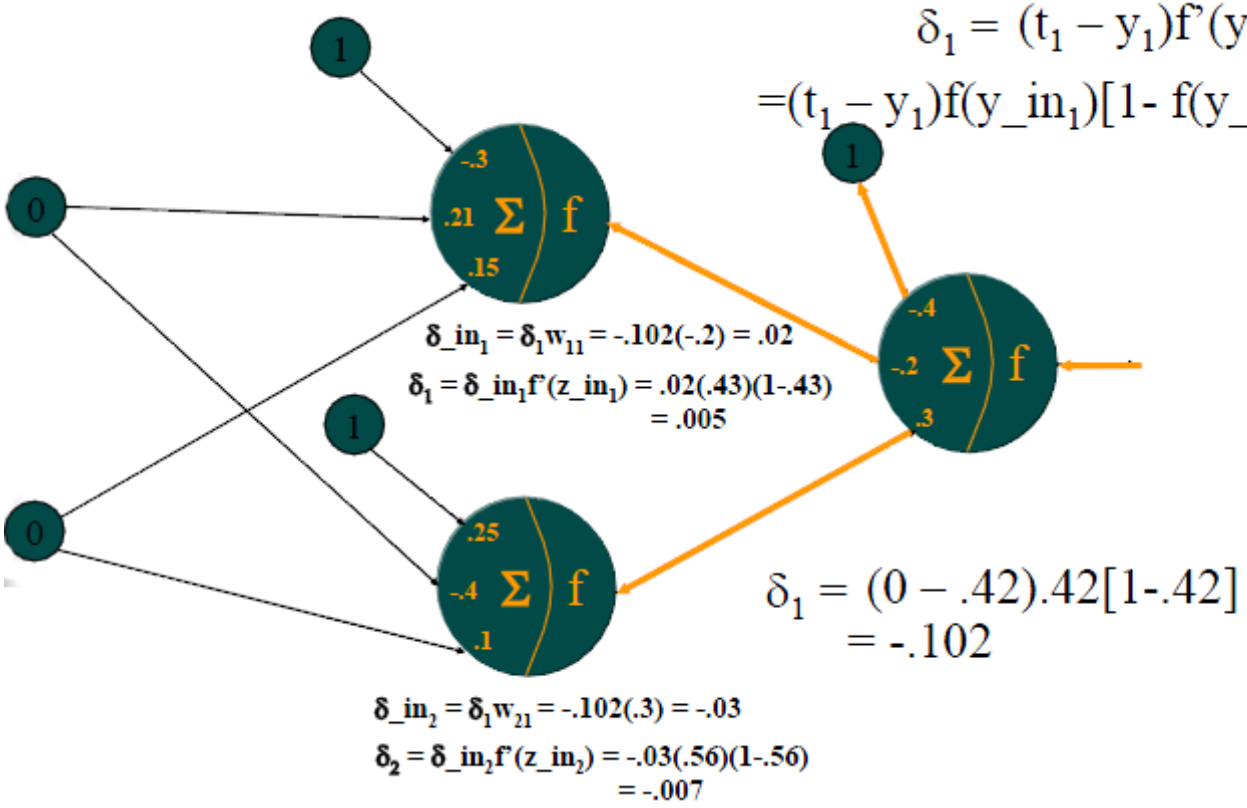
- تابع XOR: نمایش دودویی (۲ از ۶) . . .
  - پیشخور کردن ورودی

x_1	x_2	y
0	0	0



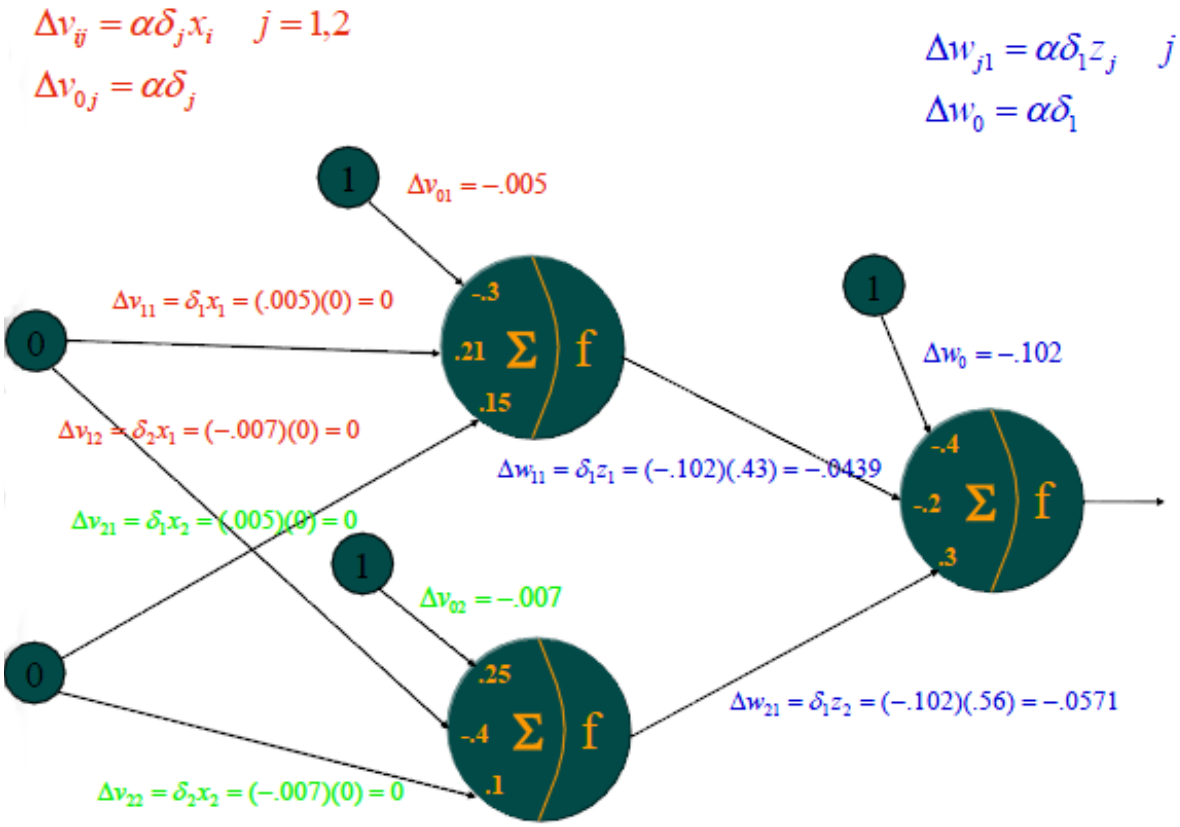
شبکه عصبی پرسپترون چندلایه (مثال) . . .

- تابع XOR: نمایش دودویی (۳ از ۶) . . .
  - پس‌انتشار خطا



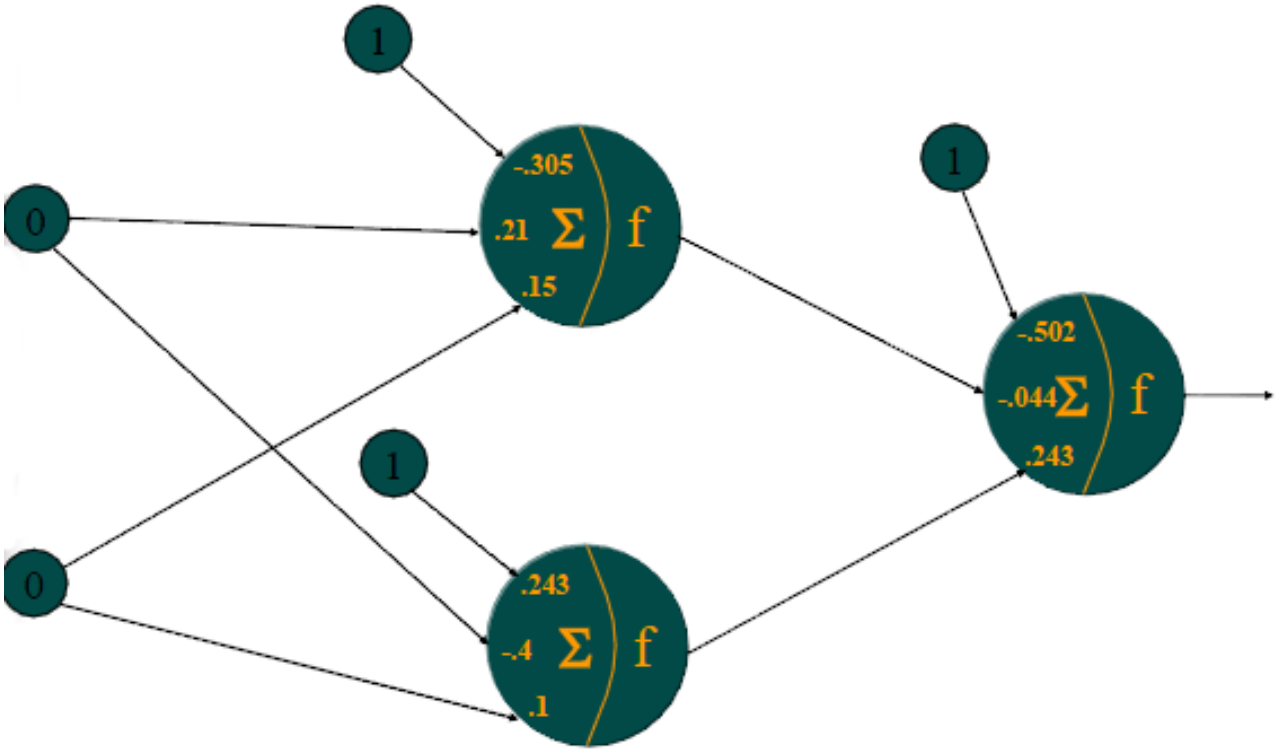
شبکه عصبی پرسپترون چندلایه (مثال) . . .

- تابع XOR: نمایش دودویی (۴ از ۶) . . .
  - محاسبه وزن‌ها



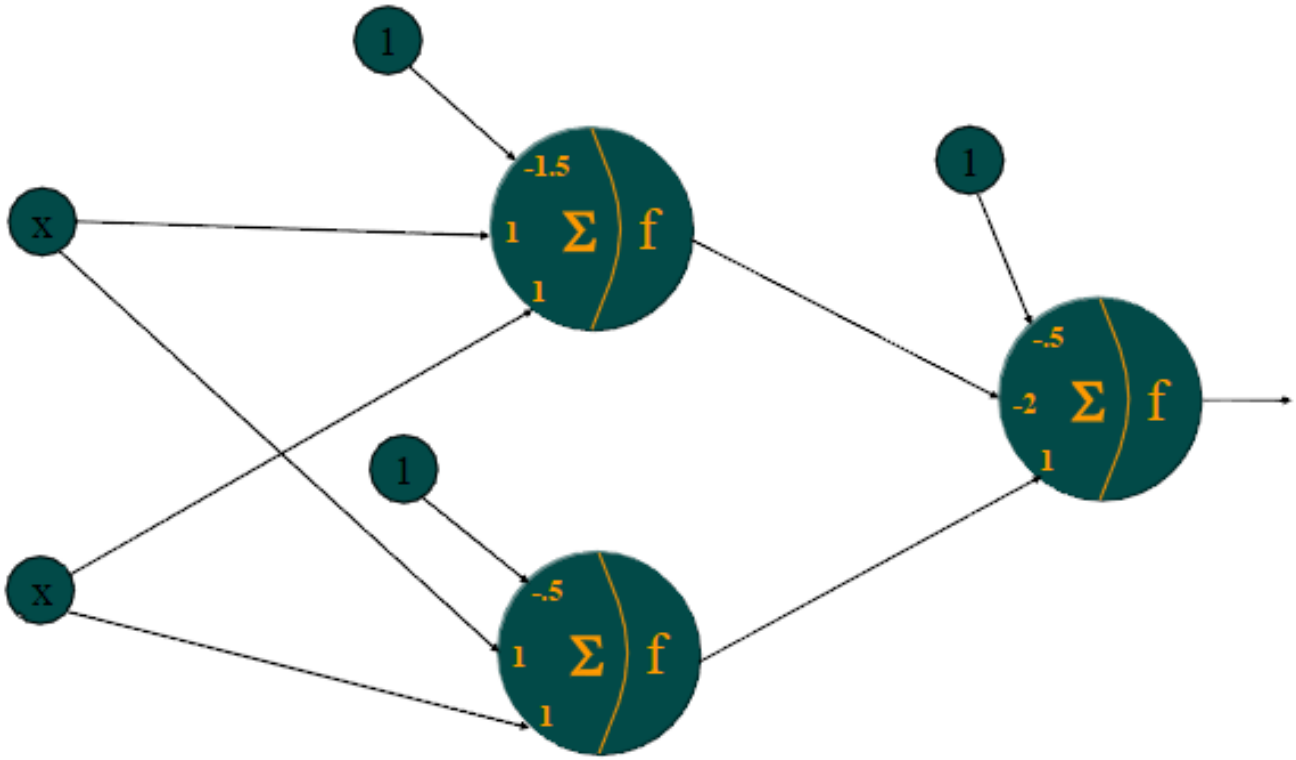
شبکه عصبی پرسپترون چندلایه (مثال) . . .

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



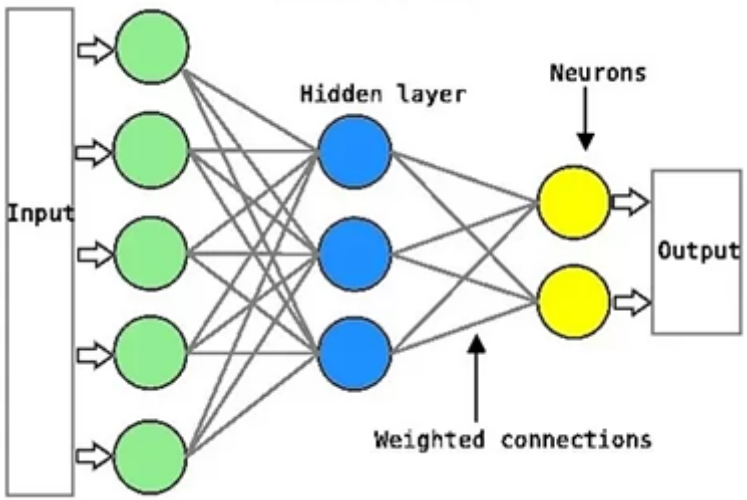
شبکه عصبی پرسپترون چندلایه (مثال) . . .

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



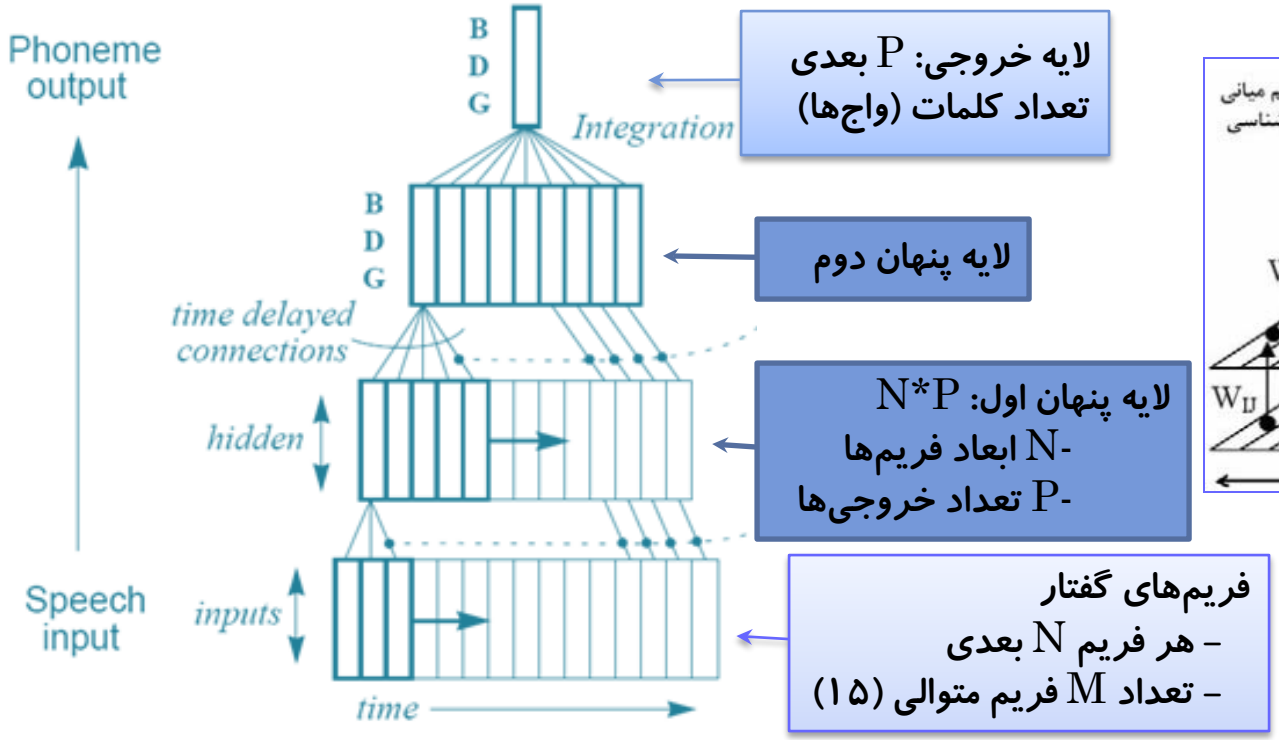
شبکه عصبی پرسپترون چندلایه (مثال) . . .

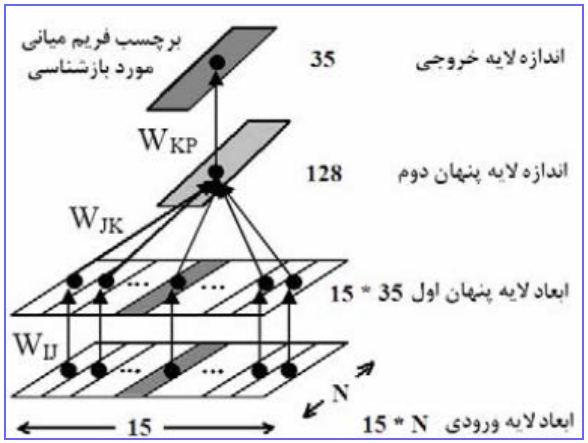
- کاربرد در پردازش گفتار و پردازش زبان
  - تشخیص گوینده
  - تشخیص جنسیت
  - جداسازی گفتار از غیر گفتار
  - تعیین نقش دستوری (POS Tagging)
  - تعیین شباهت دو متن
  - تعیین عنوان متن
  - ...



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)





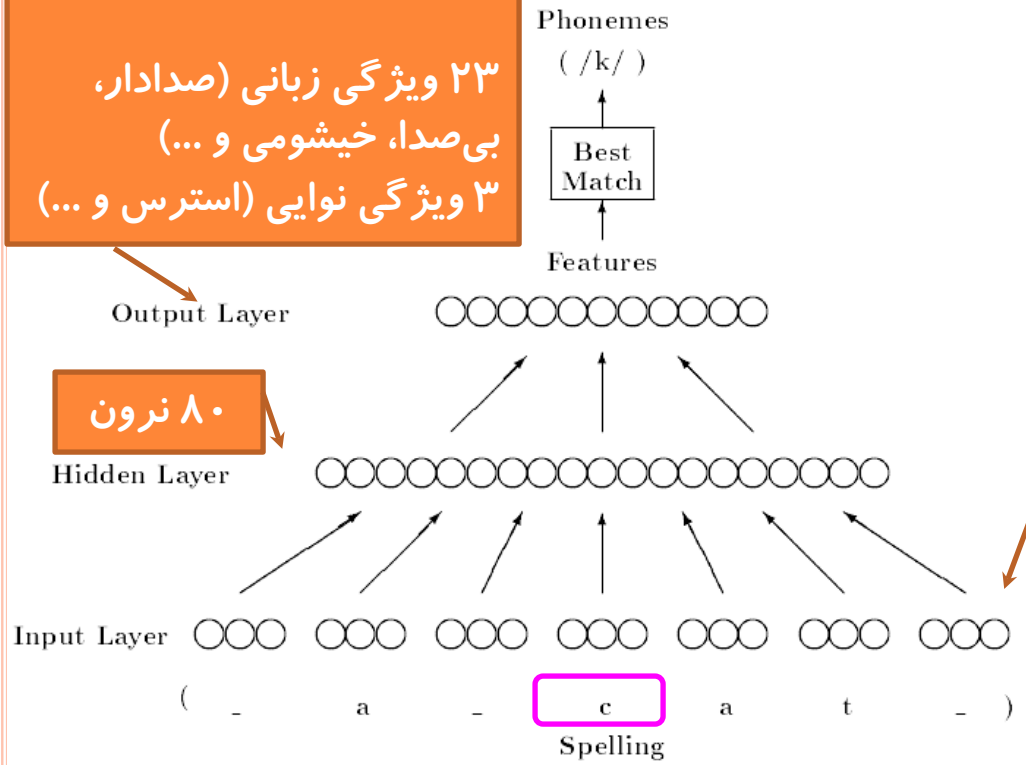
تشخیص واج‌های فارسی

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 "gave“, "brave" (کشیده) و ”have" (کوتاه)
  - تبدیل متن به صورت واجی
    - Phone → f o n (f-on-)

۲۶ نرون = ۲۶ ویژگی

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

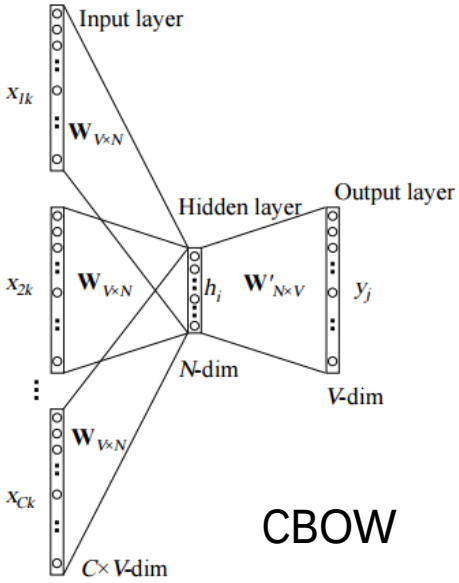


۷*۲۹=۲۰۳ نرون ورودی

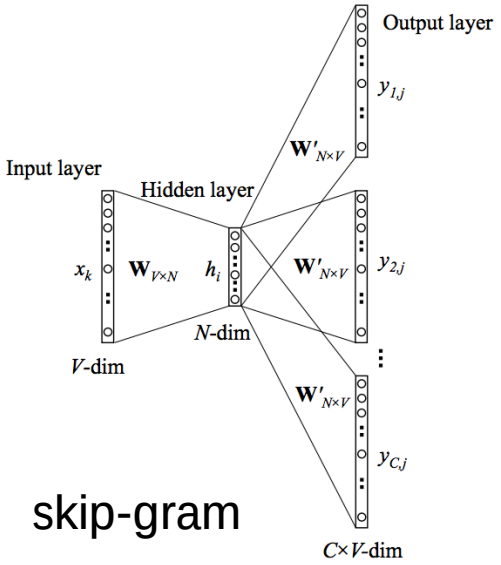
۷= واج موردنظر و ۳واج در دوطرف آن
۲۹ = بردار دودویی، هر بعد معادل یک واج

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



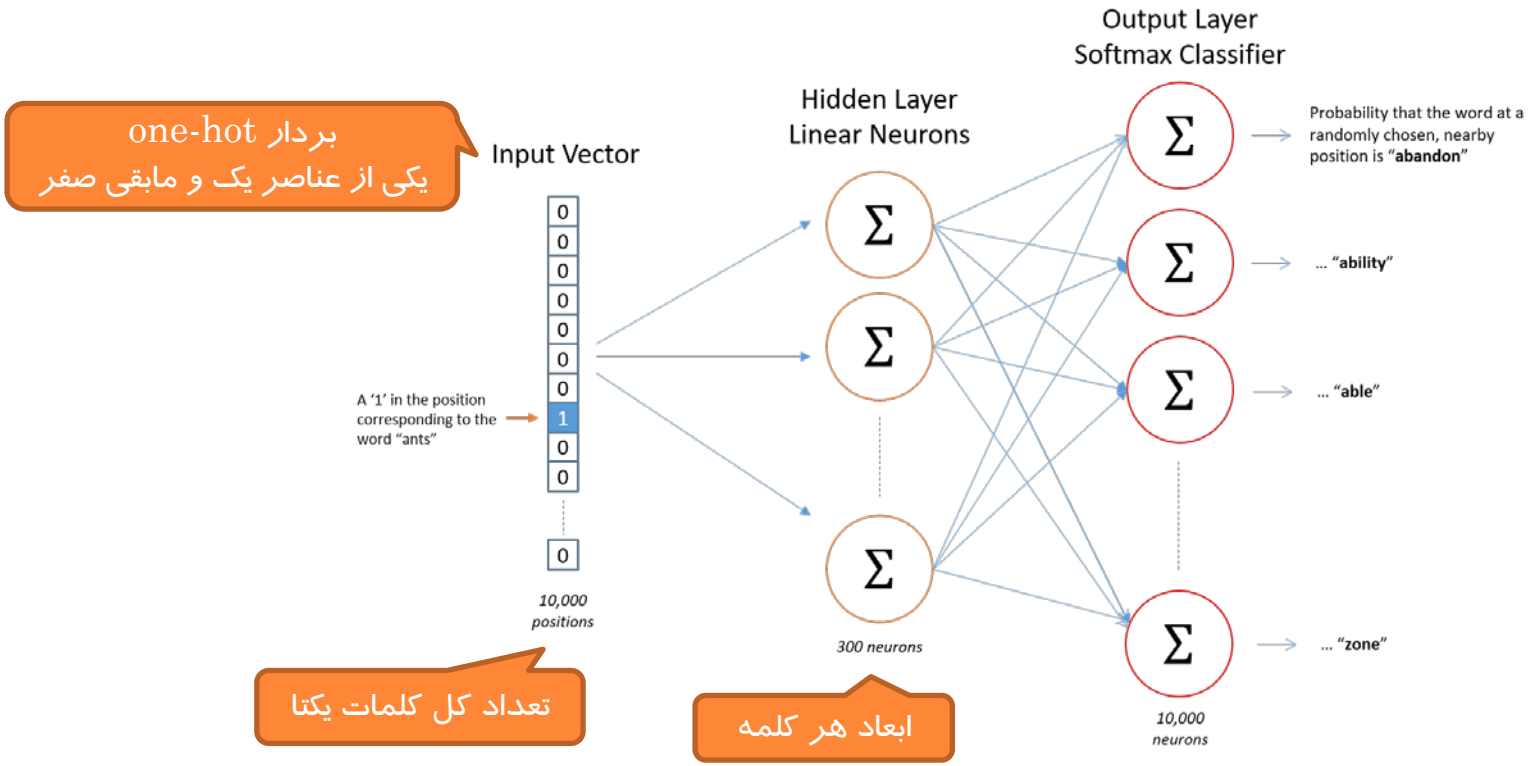
CBOW



skip-gram

شبکه عصبی پرسپترون چندلایه (مثال) . . .

- بردار کلمات: پرش چندتایی



شبکه عصبی پرسپترون چندلایه (مثال) . . .

- بردار کلمات: پرش چندتایی
  - داده ورودی

Source Text

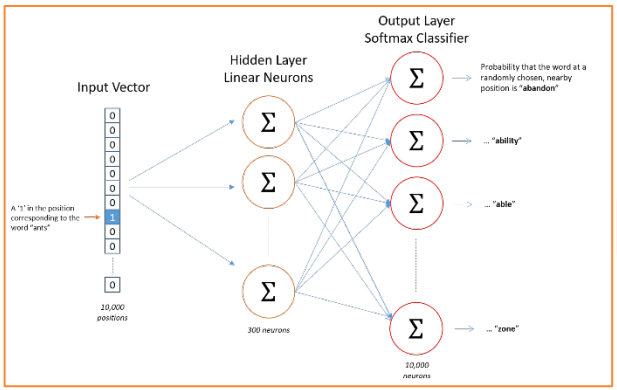
The quick brown fox jumps over the lazy dog.

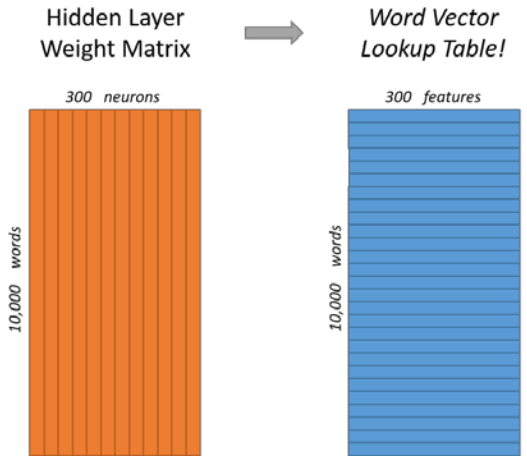
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

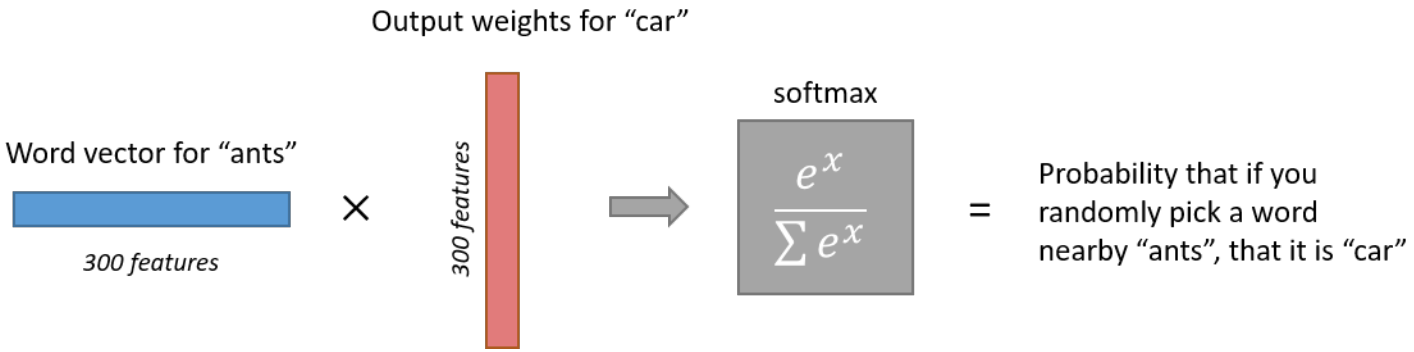
- بردار کلمات نهایی





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



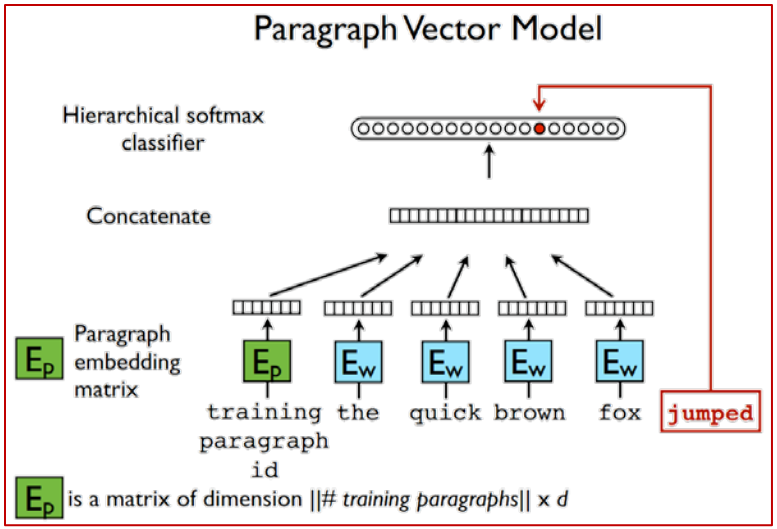
شبکه عصبی پرسپترون چندلایه (مثال) . . .

- بردار جمله/پاراگراف/سند
  - ترکیب بردار کلمات ساده
    - اصفهان+ رودخانه ➔زاینده رود

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

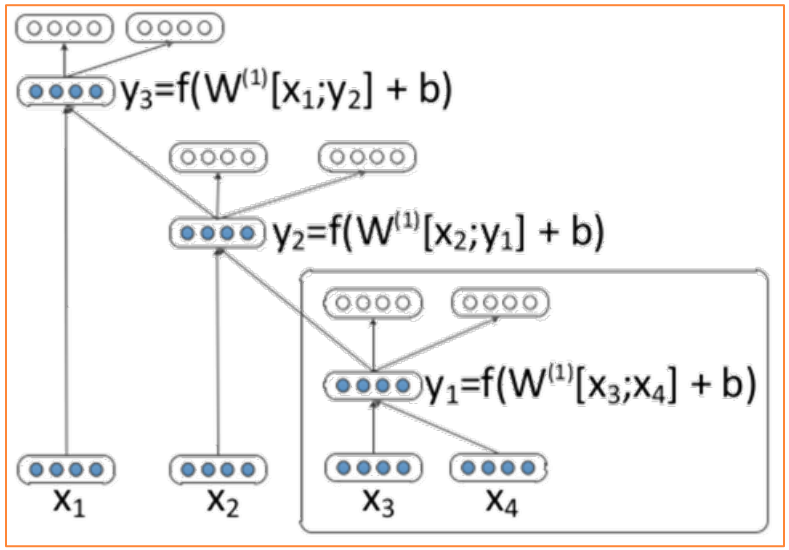
J. Mitchell and M. Lapata, "Composition in distributional models of semantics," Cognitive science, vol. 34, pp. 1388-1429, 2010.

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 خودرمزگذار 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 اولیهٔ تصادفی (مثبت یا منفی)
  - بازه متداول برای مقادیر تصادفی وزن‌ها و بایاس‌ها بین ۰.۵ و ۰.۵- (یا بین ۱- و ۱)

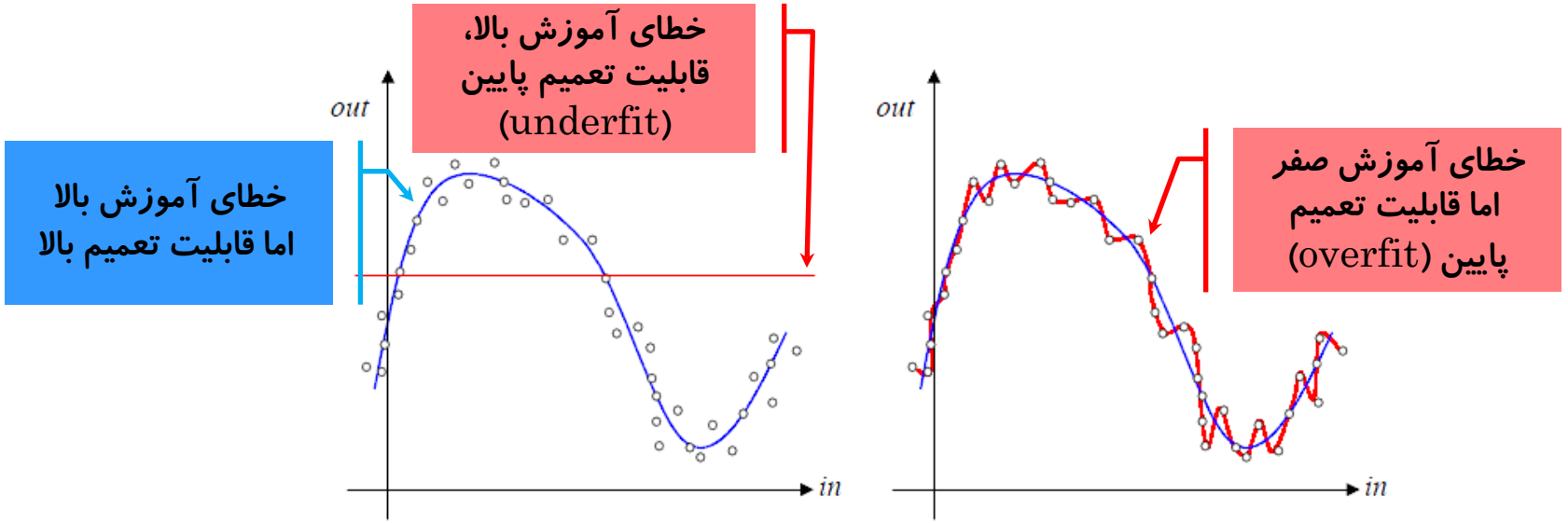
آموزش شبکهٔ عصبی با بیش از یک لایه مخفی

- مشابه الگوریتم آموزش با یک لایه مخفی
- محاسبهٔ ها برای هر لایهٔ مخفی اضافی مشابه لایه مخفی بیان شده در الگوریتم است
- برای هر لایهٔ مخفی، گام ۴ در مرحلهٔ پیش‌خور و گام ۷ در مرحلهٔ پس‌انتشار تکرار می‌شود.
- یک لایهٔ مخفی در شبکهٔ 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 لایهٔ مخفی، آموزش شبکه را آسان‌تر می‌کند.

شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

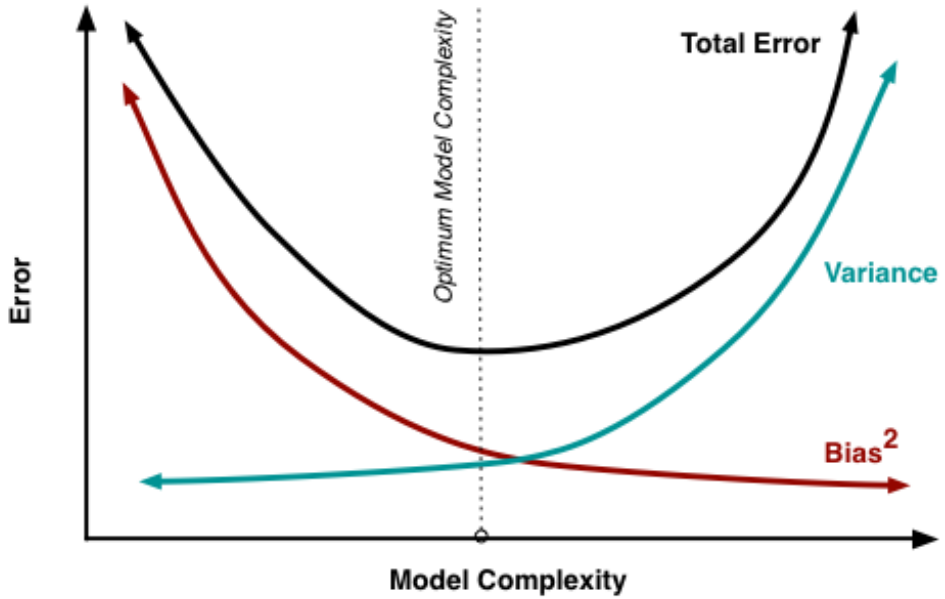
تعادل بین یادگیری الگوها و تعمیم

- 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 زمانی‌که مقدار مربعات خطا واقعاً حداقل شده، الزاماً مفید نمی‌باشد



شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

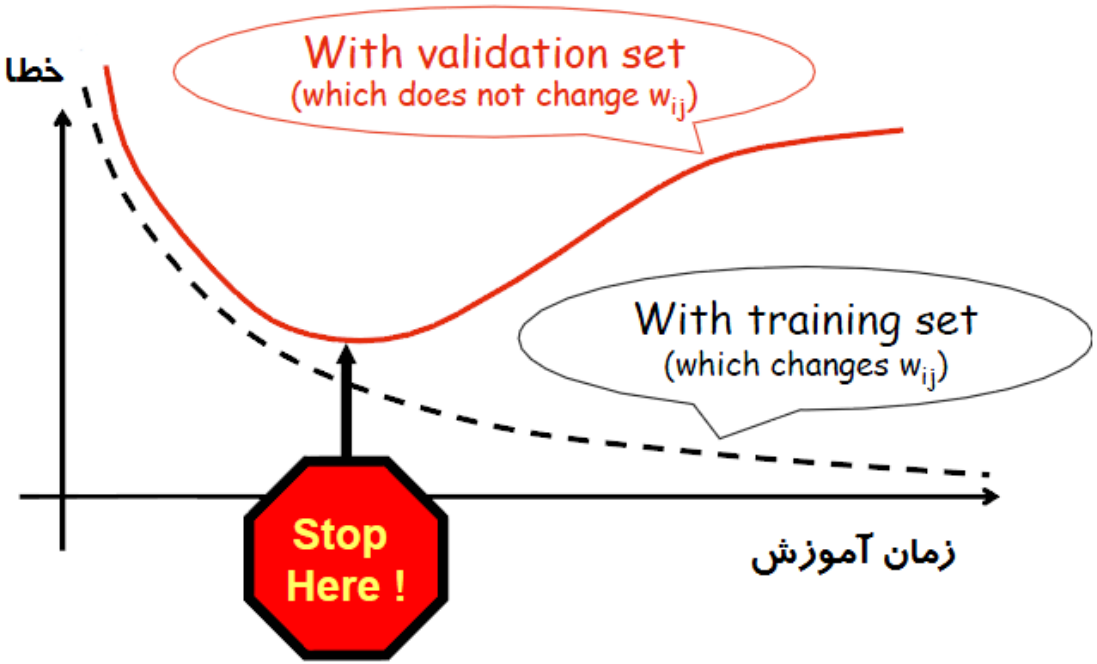
- تعادل بین یادگیری الگوها و تعمیم= جلوگیری از بیش برازش: راه حل ۱
  - پیچیده کردن مدل = بیش برازش = بایاس کمتر = واریانس بیشتر



- تعداد نرون‌های کمتر در لایه مخفی

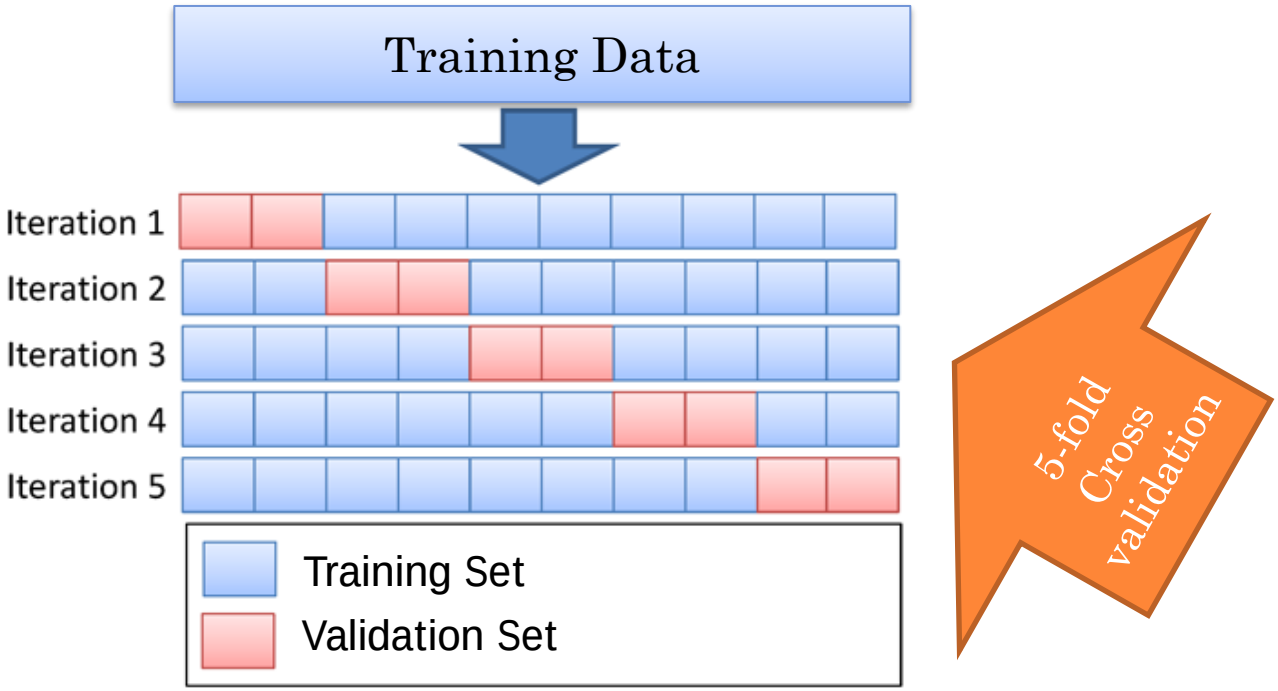
شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

- تعادل بین یادگیری الگوها و تعمیم= جلوگیری از بیش برازش: راه حل ۲
  - توقف زودهنگام شبکه با افزایش خطای مجموعه ارزیابی (تست)
    - استفاده از مجموعه ارزیابی



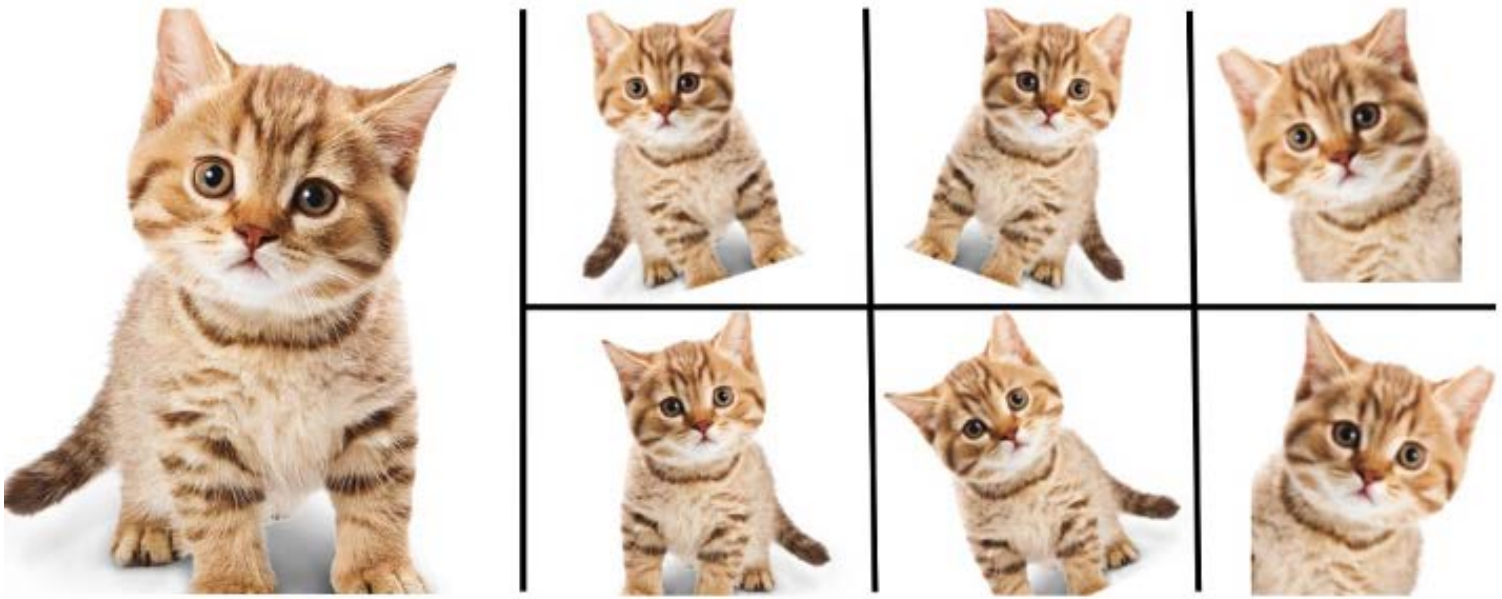
شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

- تعادل بین یادگیری الگوها و تعمیم= جلوگیری از بیش برازش: راه حل ۲
  - 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 زیر مجموعه‌ها برای تایید اعتبار استفاده می‌شود



شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

- تعادل بین یادگیری الگوها و تعمیم= جلوگیری از بیش برازش: راه حل ۳
  - افزایش حجم و تنوع داده‌های آموزش



شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

- **به‌روز کردن وزن با پس‌انتشار با گشتاور(Momentum)**
  - تغییر روش کاهش گرادیان: مقدار تغییر وزن ترکیبی از گرادیان (شیب) فعلی و گرادیان قبلی
  - به‌روز شدن وزن‌های زمان t+1 وابسته به وزن‌های زمان‌های قبل‌تر (مانند t و t-1)

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

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

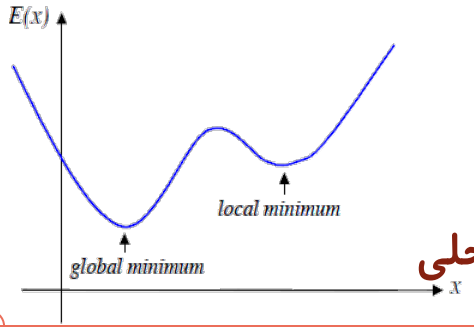
$$\Delta w_{jk}(t+1) = \alpha\,\delta_k\,z_j + \mu\,\Delta w_{jk}(t)$$

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

گرادیان فعلی

گرادیان قبلی

پارامتر ممان (بین ۰ تا ۱)



- همگرایی سریع‌تر + کاهش احتمال گیر کردن در نقطهٔ کمینهٔ محلی

شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

- **تعداد داده های آموزش: قاعدهٔ تجربی**
  - P= تعداد الگوهای آموزش موجود،
  - W= تعداد وزن‌های مورد آموزش در شبکه
  - e= صحت دسته‌بندی مورد نظر

  - آموزش شبکه برای دسته‌بندی صحیح کسری معادل (e/2)-1 از الگوهای آموزشی،
  - می‌توان مطمئن بود که شبکه 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.

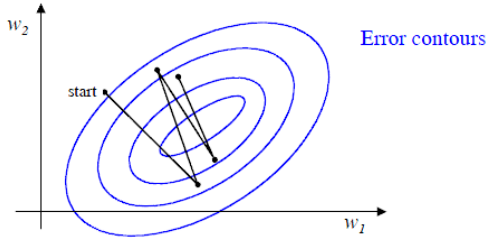
شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

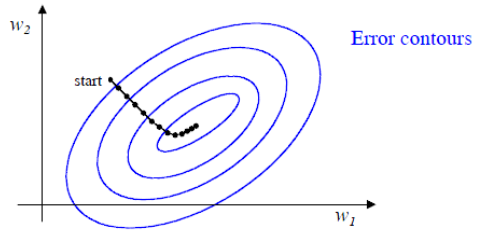
- به‌روز کردن دسته‌ای (Batch Updating)
  - به جای به‌روز کردن وزن‌های شبکه بعد از ارائهٔ هر الگوی آموزشی
  - ادغام مقدار تصحیح (تغییر) وزن را برای چند الگو یا برای تمام الگوها در یک دور کامل
  - تشکیل یک مقدار تنظیم وزن برای هر وزن، برابر با میانگین عبارات تصحیح وزن‌ها

  - آسان‌تر کردن تصحیح وزن‌ها
  - مقاوم بودن در برابر داده‌های نویزی
  - موازی‌سازی محاسبات
  - افزایش احتمال نزدیک شدن به کمینهٔ محلی

شبکه عصبی پرسپترون چندلایه: نکات تکمیلی . . .

- نرخ یادگیری ثابت
  - مقدار بزرگ = سرعت همگرایی بالا + همگرایی ناهموار (احتمال بالای واگرایی)





  - مقدار کوچک = سرعت همگرایی پایین + همگرایی هموار
    - مقدار مناسب حدود ۰.۰۰۱ و ۰.۰۰۰۱
- مقدار دینامیک
  - اوایل آموزش مقدار بزرگ و به مرور کوچک شود
  - هر وزن دارای نرخ اختصاصی خودش
    - روش دلتا-بار–دلتا (Delta-Bar-Delta)

شبکه عصبی پرسپترون چندلایه: نکات تکمیلی

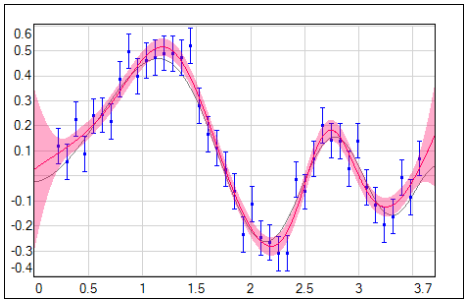
- **شبکه MLP تقریب‌زننده‌های جهانی است: قضیهٔ هچ-نیلسون**
  - هر تابع پیوستهٔ $f : I^n \to R^m$ را که در آن I بازهٔ بستهٔ $[0,1]$ است، می‌توان دقیقاً با یک شبکهٔ عصبی پیش‌خور با n واحد ورودی، $2n+1$ واحد مخفی و m واحد خروجی نمایش داد.
  - تابع فعال‌سازی برای واحد مخفی jام: $z_j = \sum_{i=1}^{n} \lambda^i \psi(x_i + \varepsilon j) + j$

عددی ثابت و حقیقی

تابع پیوسته، حقیقی و یکنوای صعودی، مستقل از f و وابسته به n

  - مقدار ثابت ε برای فراهم بودن شرایط قضیهٔ 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

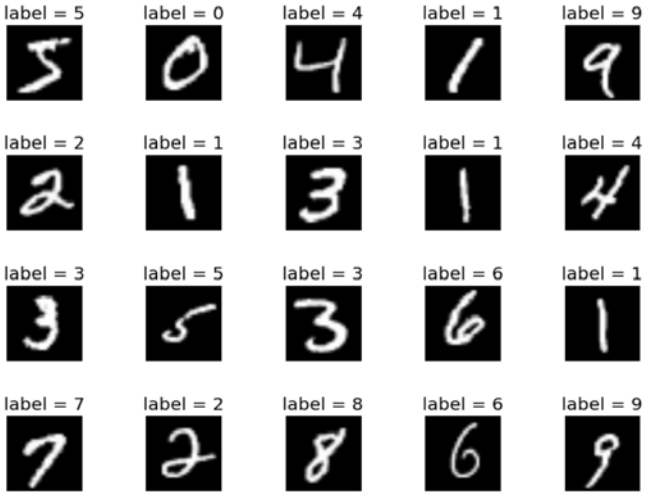


شبکه پرسپترون چندلایه: کد پایتون . . .

- استفاده از scikit-learn

```
import numpy as np
from scipy.ndimage import convolve
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import fetch_mldata
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.externals import joblib
from sklearn.utils import shuffle
import os.path

PATH = 'mlp_model'

if __name__ == '__main__':
    print('Fetching and loading MNIST data')
    mnist = fetch_mldata('MNIST original')
    X, y = mnist.data, mnist.target
    X, y = shuffle(X, y, random_state=0)
    X2=X[0:4000,:] # select a part of data
    y2=y[0:4000] # select a part of data labels
    X_train, X_test, y_train, y_test = train_test_split(X2 / 255., y2, test_size=0.25)

    print('Got MNIST with %d training- and %d test samples' % (len(y_train), len(y_test)))
    print('Digit distribution in whole dataset:', np.bincount(y2.astype('int32')))

    clf = None
    if os.path.exists(PATH):
        print('Loading model from file.')
        clf = joblib.load(PATH).best_estimator_
    else:
        print('Training model.')
        # params = {'hidden_layer_sizes': [(256,), (512,), (128, 256, 128,)]}
        params = {'hidden_layer_sizes': [(512,)]}
        mlp = MLPClassifier(verbose=10, momentum=0.9, learning_rate='adaptive', activation='relu')
        clf = GridSearchCV(mlp, params, verbose=10, n_jobs=-1, cv=5)
        clf.fit(X_train, y_train)
        print('Finished with grid search with best mean cross-validated score:', clf.best_score_)
        # print('Best params appeared to be', clf.best_params_)
        joblib.dump(clf, PATH)
        clf = clf.best_estimator_

    print('Test accuracy:', clf.score(X_test, y_test))
```

شبکه پرسپترون چندلایه: کد پایتون

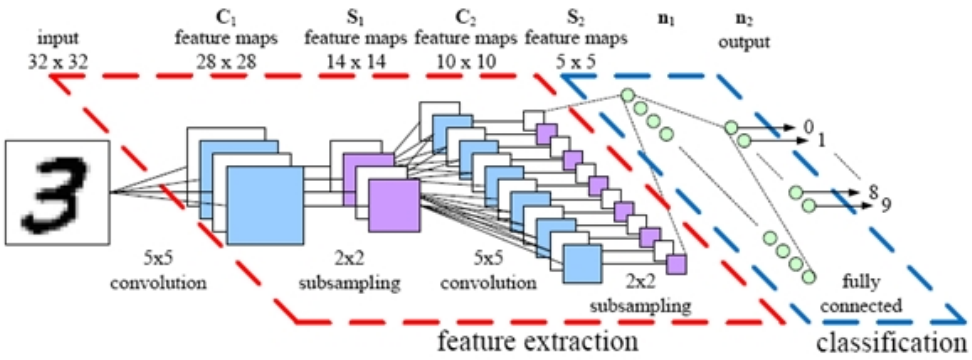


- دادگان MNIST
  - تعداد ۷۰ هزار تصویر از اعداد دست نویس ۰ تا ۱۰
    - ۱۰ دسته
    - ۶۰ هزار آموزش و ۱۰ هزار آزمون
  - هر عدد یک تصویر ۲۸*۲۸ = ۷۸۴ پیکسلی
- دانلود از http://yann.lecun.com/exdb/mnist
- کارایی روش‌های مختلف

روش	درصد خطا روی داده آزمون
linear classifier (1-layer NN)	۱۲.۰
K-nearest-neighbors, Euclidean (L2)	۵.۰
SVM, Gaussian Kernel	۱.۴
2-layer NN, 1000 hidden units	۴.۵
Convolutional net LeNet-4	۱.۱
committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	۰.۲۳

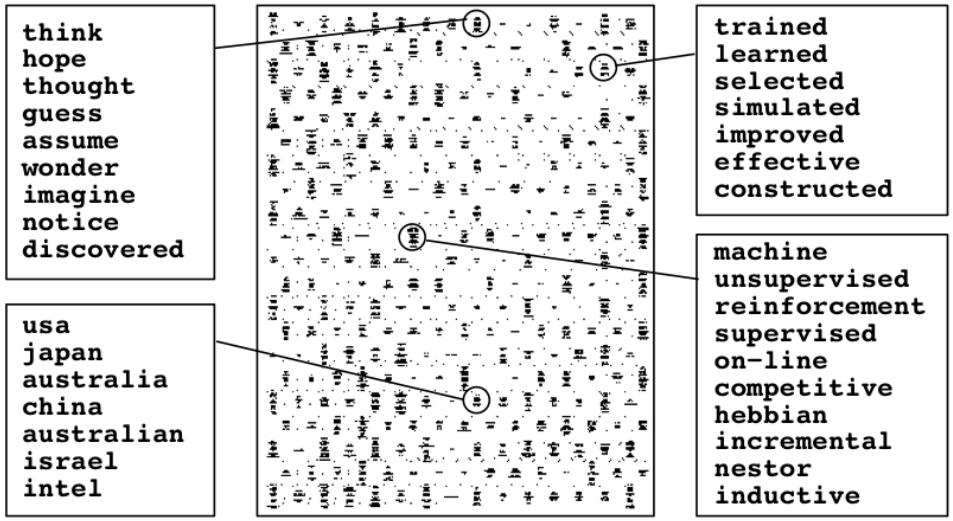
سایر شبکه های عصبی در پردازش زبان و گفتار . . .

- 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 شبکه های عصبی در پردازش زبان و گفتار

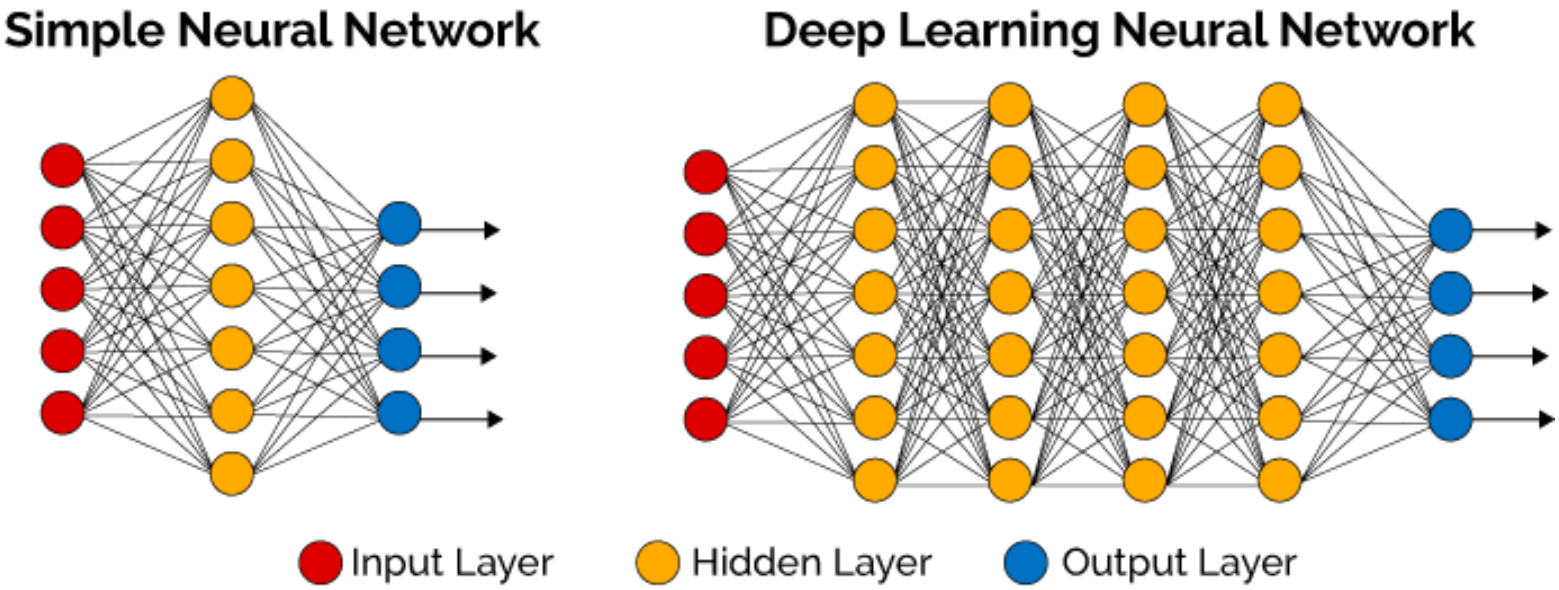
- شبکه‌های عصبی نگاشت‌های خودسازمانده کوهونن (SOM)
  - کاربرد در دسته‌بندی خودکار متون
  - یادگیری بدون نظارت (خوشه بندی)



  - حالت با نظارت
    - یادگیری چندی‌سازی برداری (LVQ)

یادگیری عمیق . . .

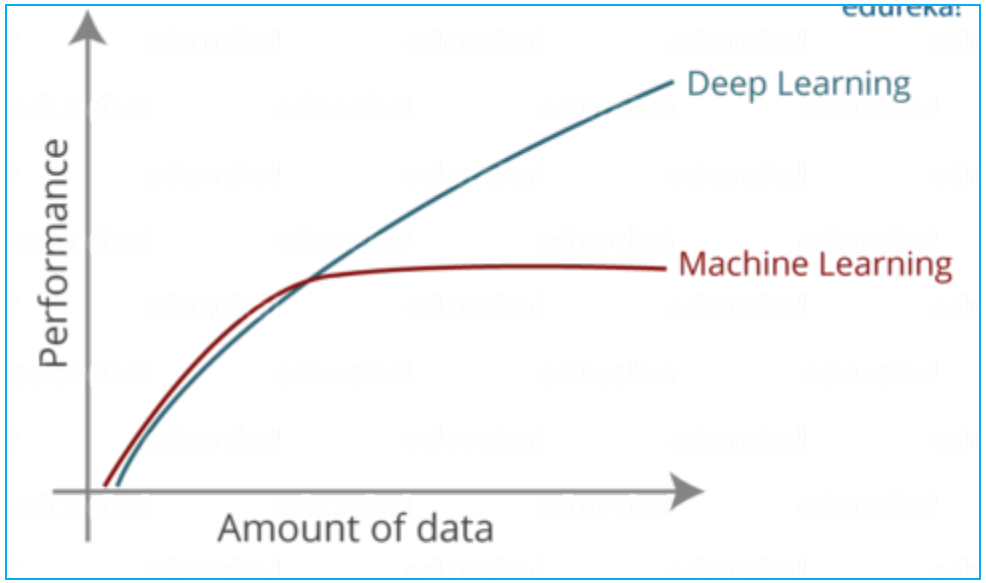
- افزایش تعداد لایه های مخفی

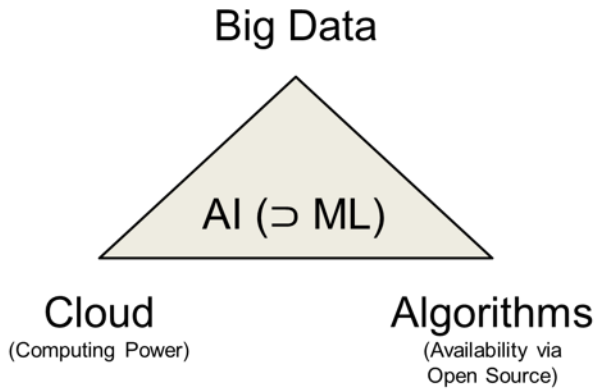


- مدل پیچیده = تعداد پارامترهای زیاد = توان محاسباتی بالا

یادگیری عمیق . . .

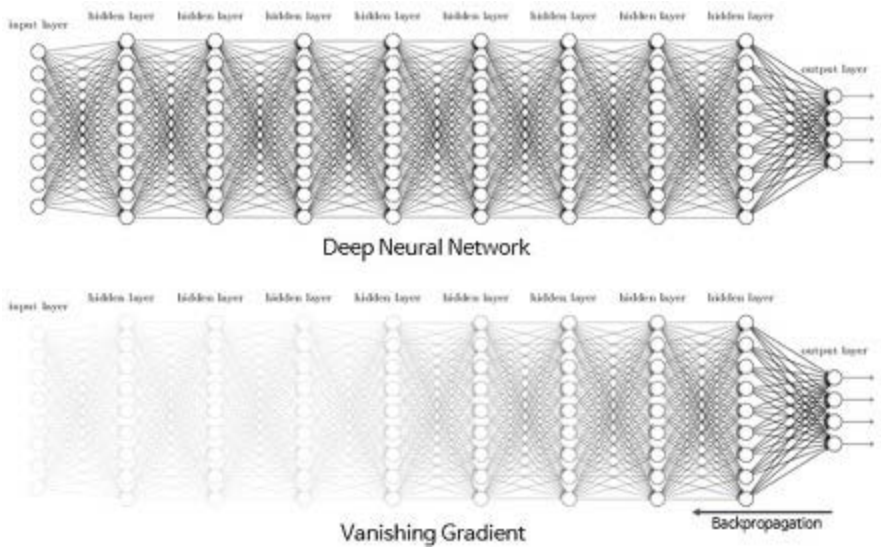
- سه رکن اصلی
  - داده حجیم (Big Data)
  - توان پردازشی (GPU)
  - الگوریتم یادگیری





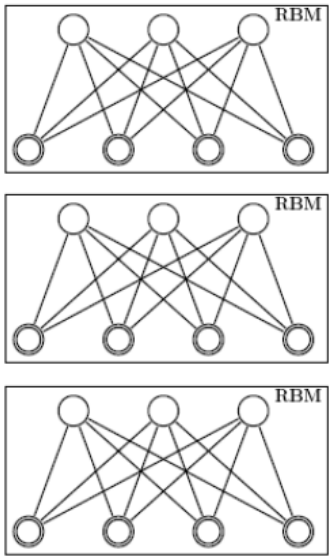
یادگیری عمیق . . .

- 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 ویژگیِ ویژگی ها استخراج می شود
- ادامه این روند تا رسیدن به لایه آخر

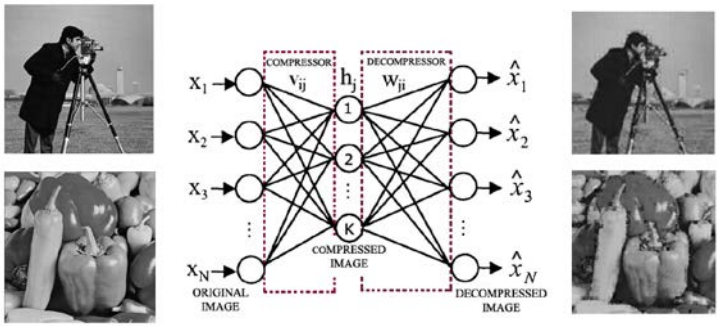
یادگیری عمیق: شبکه باور عمیق (DBN)

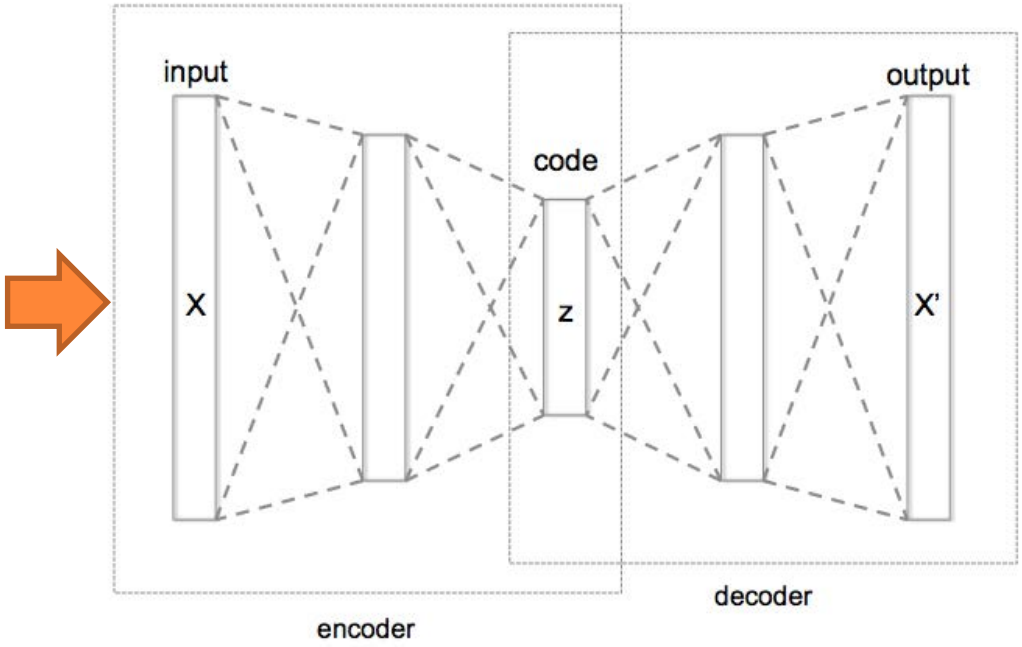
- برای الگوریتم آموزش و مثال، مراجعه کنید به

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

شبکه های خودرمزگذار (AE) . . .

- شبکه های خودرمزگذار (AE: AutoEncoders)
  - استفاده از لایه وسط به عنوان ویژگی

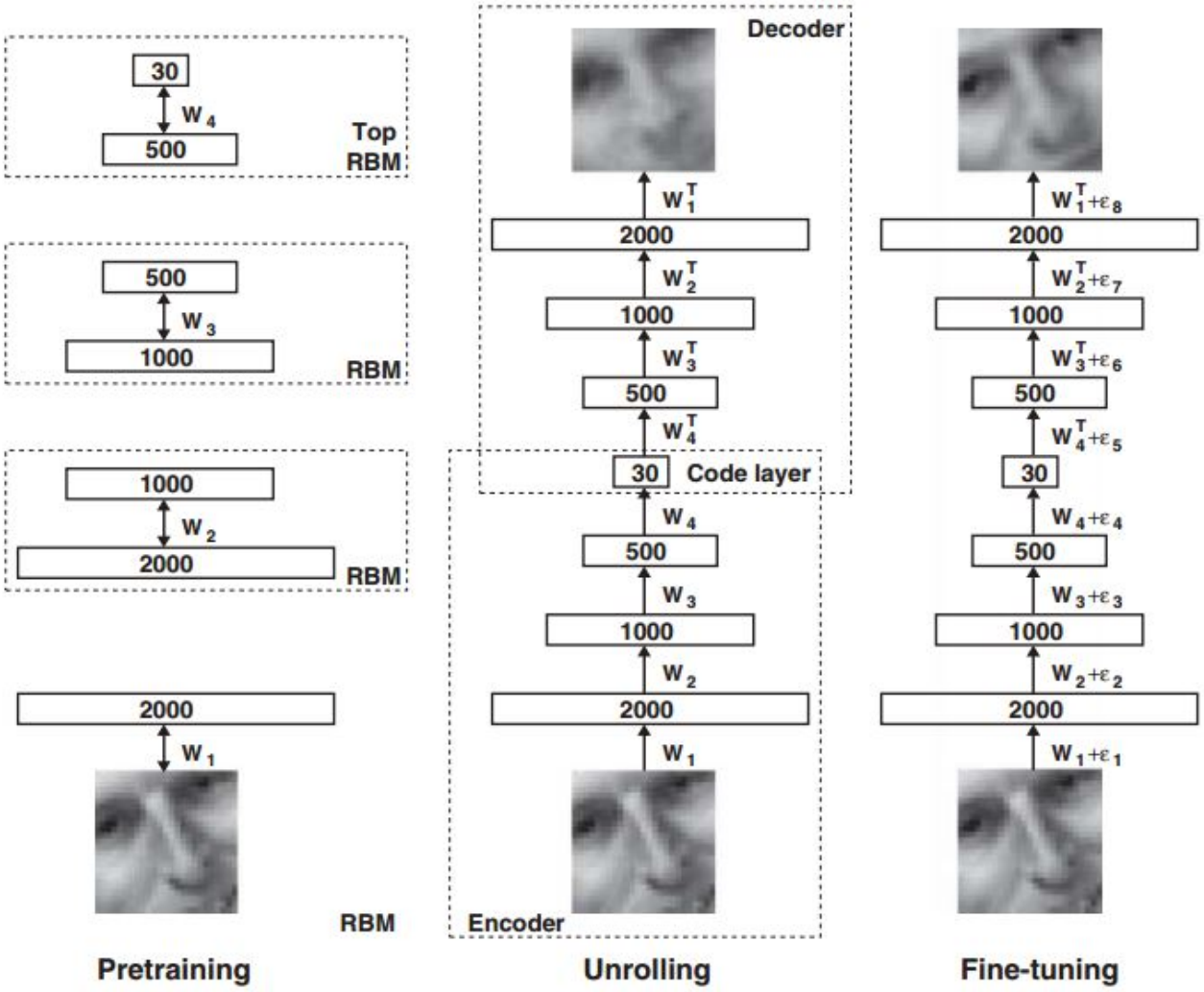




Veisi, Hadi, and Mansour Jamzad. "A complexity-based approach in image compression using neural networks." *Int. J. Sign. Process* 5 (2009): 82-92.

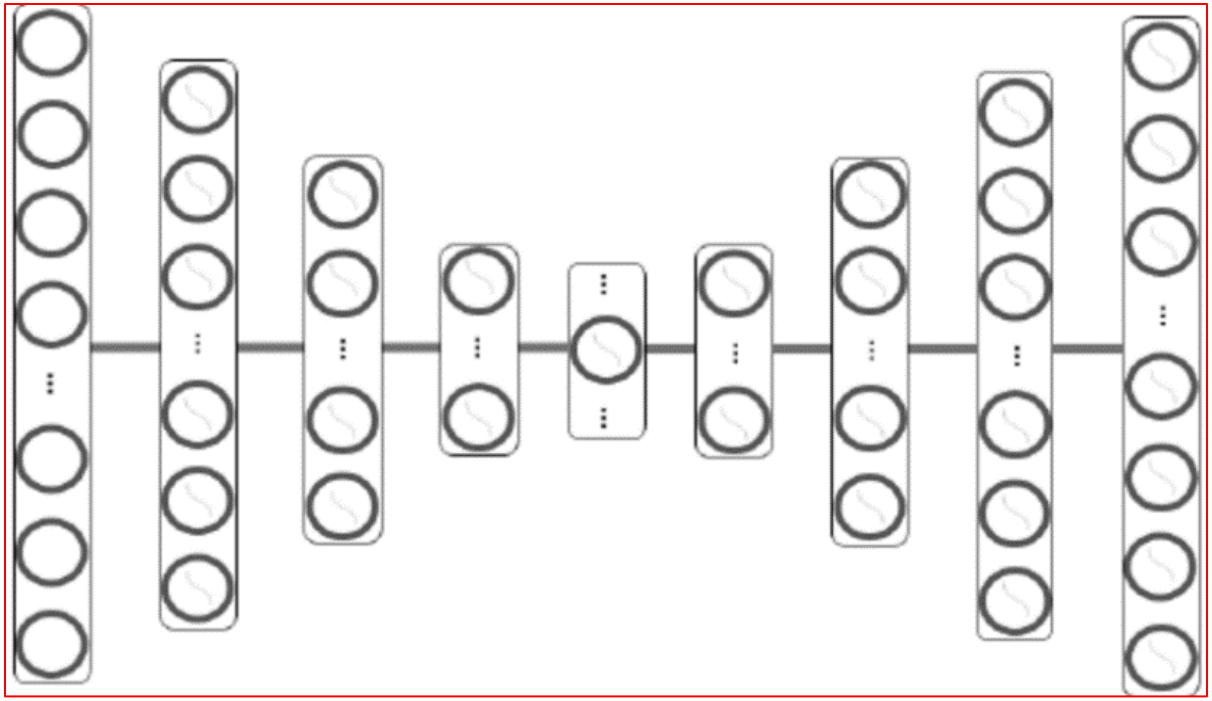
شبکه های خودرمزگذار (AE) . . .

- خودرمزگذار



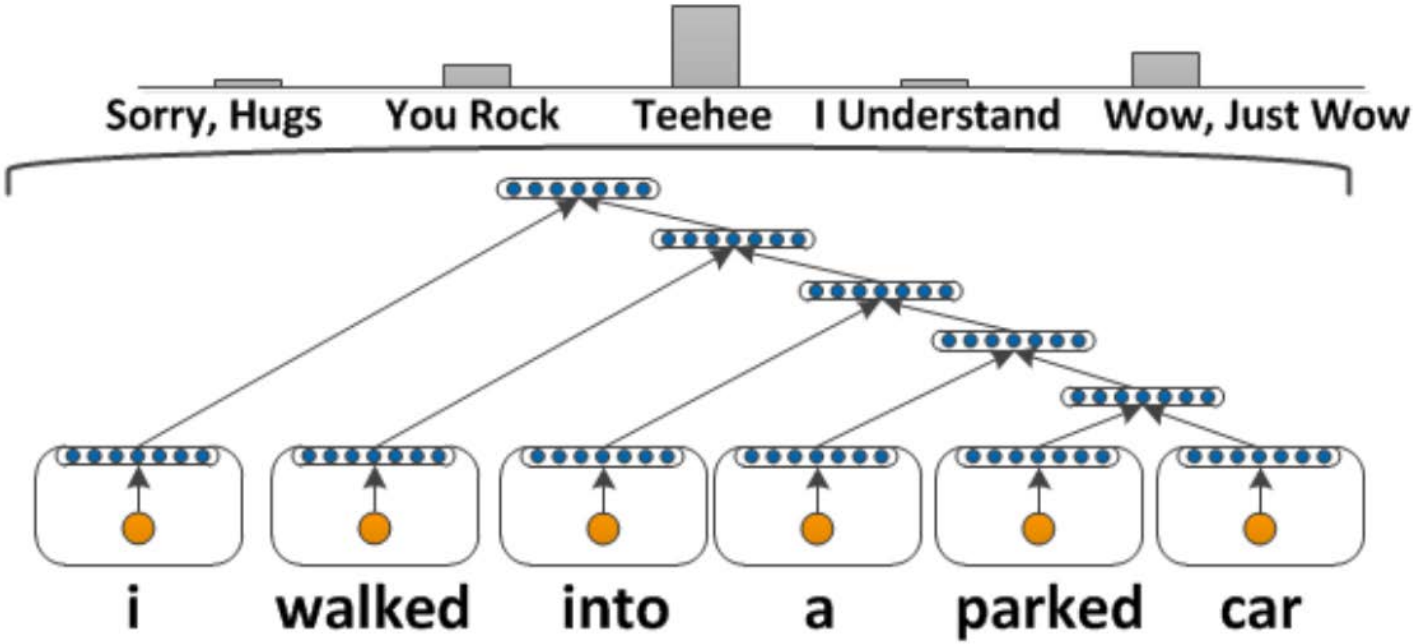
شبکه های خودرمزگذار (AE) . . .

- شبکه های خودرمزگذار (AE: AutoEncoders)
  - استفاده از لایه وسط به عنوان ویژگی



شبکه های خودرمزگذار (AE) . . .

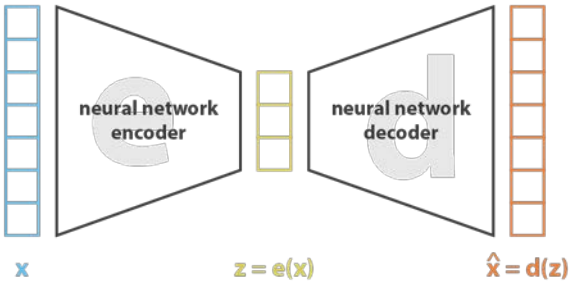
- خودرمزگذار بازگشتی (RAE: Recursive AutoEncoders)
  - ترکیب دو به دوی بردار کلمات برای تولید بردا جمله/سند



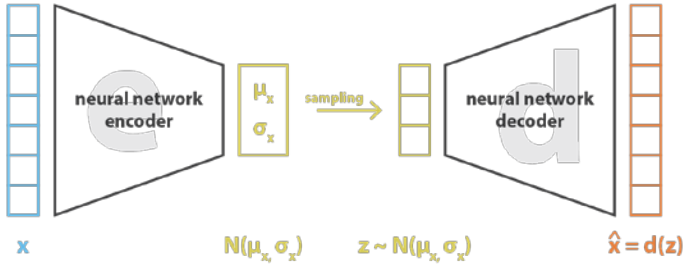
شبکه های خودرمزگذار متغیر (VAE) . . .

شبکه Variational Autoencoders

- استفاده از عبارت تنظیم (regularization) برای جلوگیری از بیش برازش و اطمینان از مناسب بودن متغیر پنهان برای تولید داده
- یادگیری تابع توزیع به جای خود داده



$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$



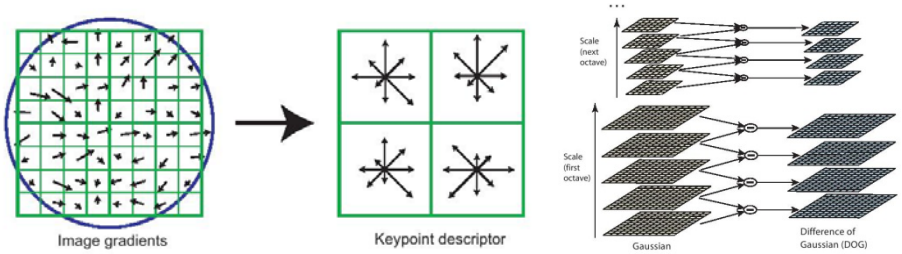
$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

شبکه های خودرمزگذار متغیر (VAE)

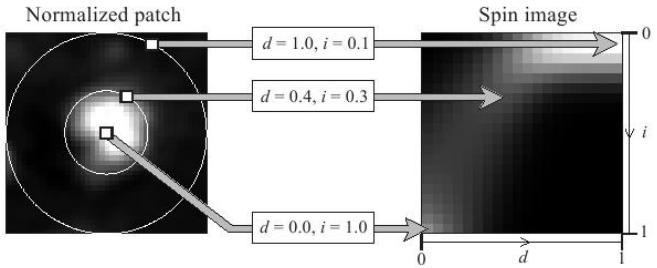
- تولید تصویر

https://github.com/WojciechMormul/vae

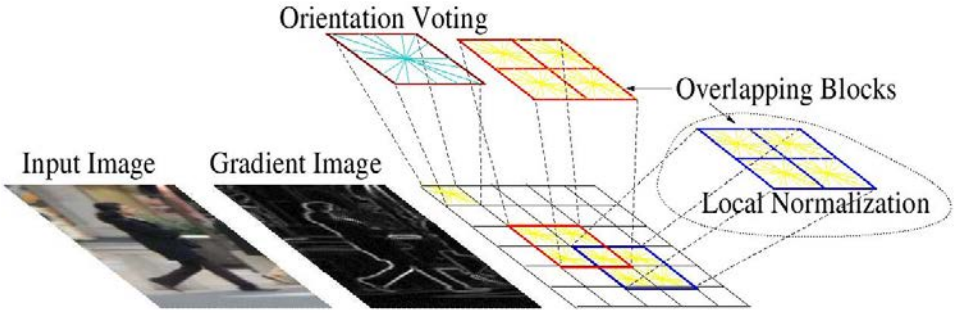
یادگیری عمیق: استخراج ویژگی (تصویر)



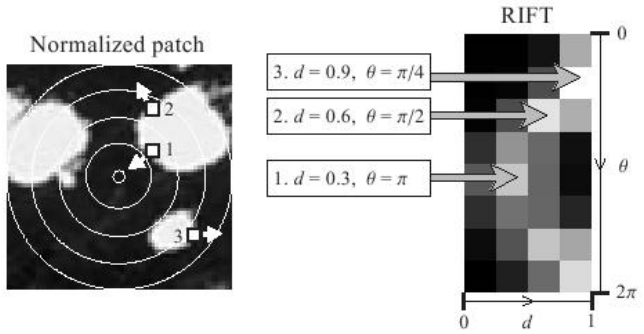
SIFT



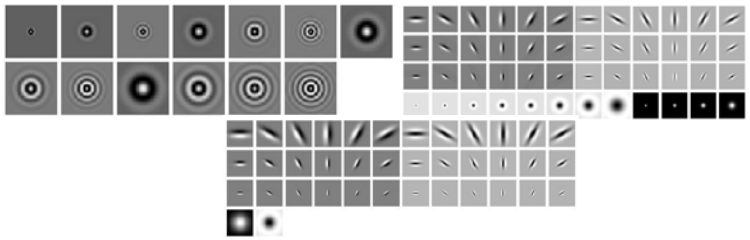
Spin image



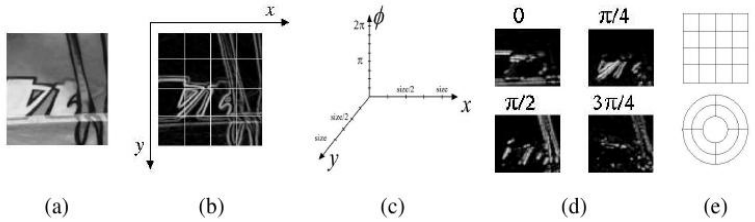
HoG



RIFT

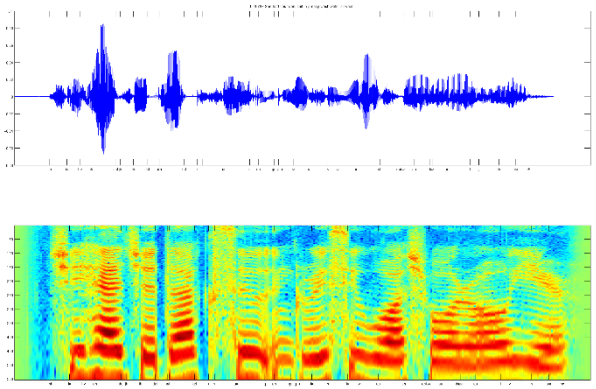


Textons

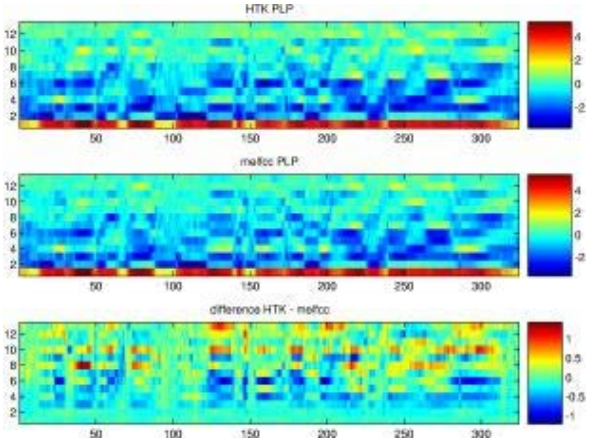


GLOH

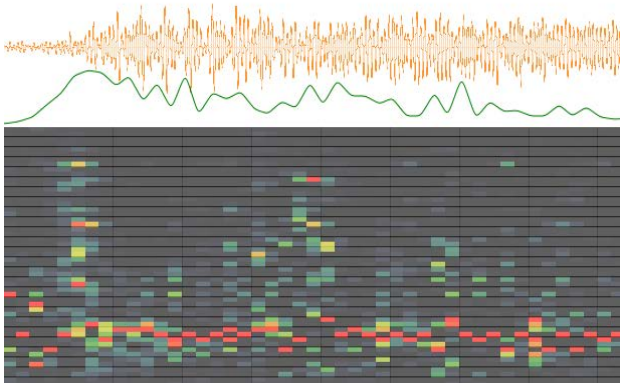
یادگیری عمیق: استخراج ویژگی (صدا)



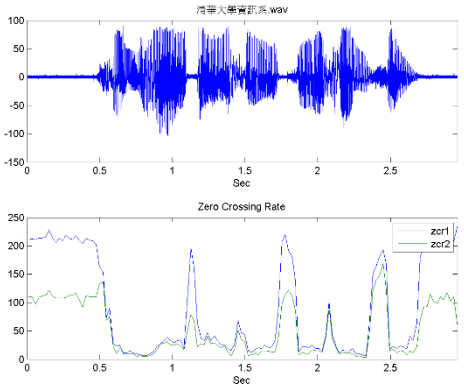
Spectrogram



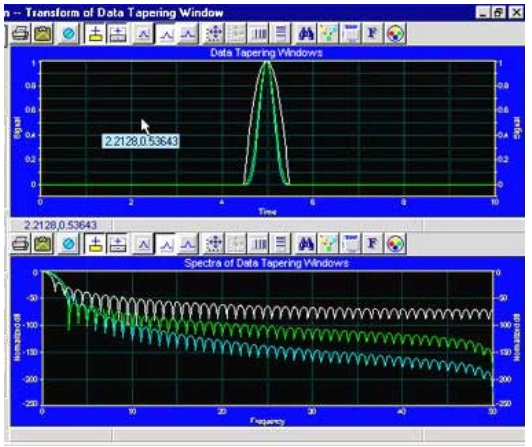
MFCC



Flux



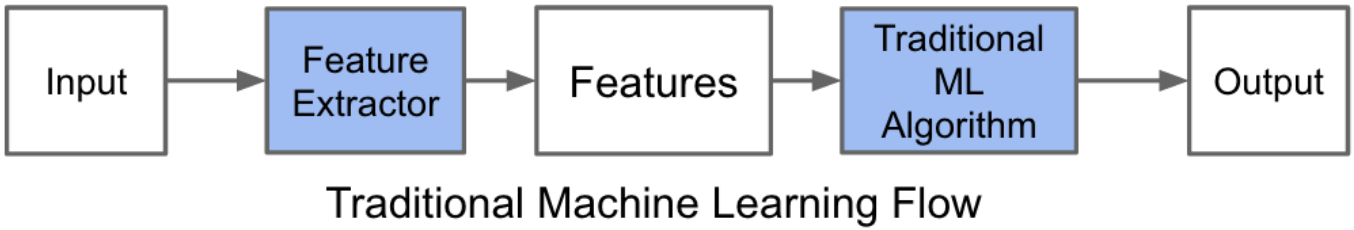
ZCR



Rolloff

یادگیری عمیق . . .

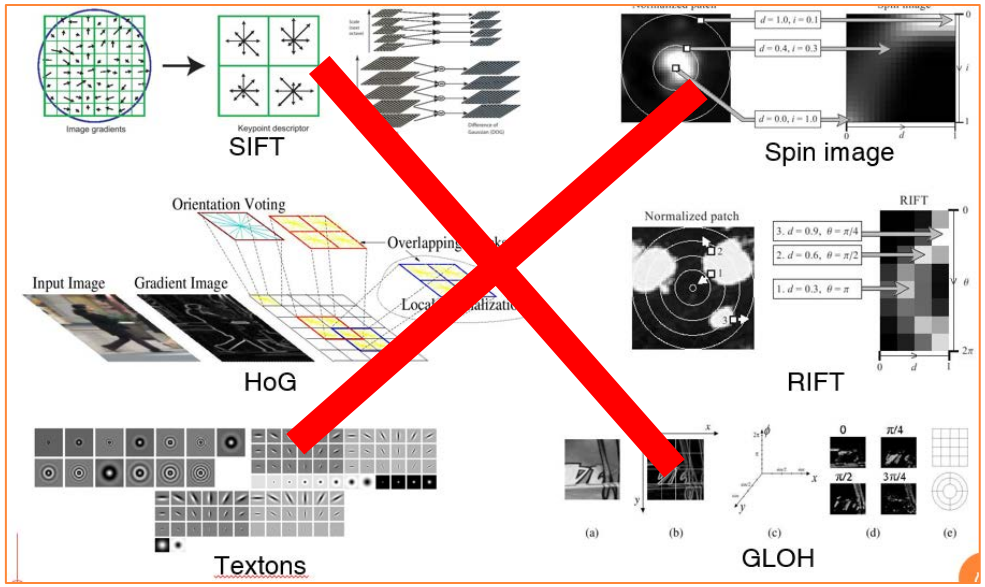
- ترکیب استخراج ویژگی و دسته‌بندی



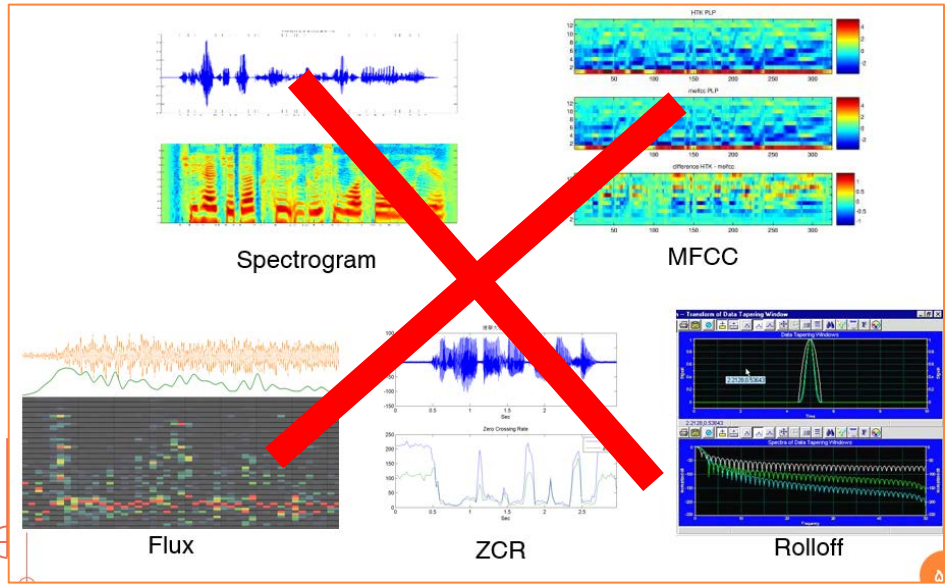
Traditional Machine Learning Flow

Deep Learning Flow

یادگیری عمیق: استخراج ویژگی و دسته‌بندی



Unlabeled images

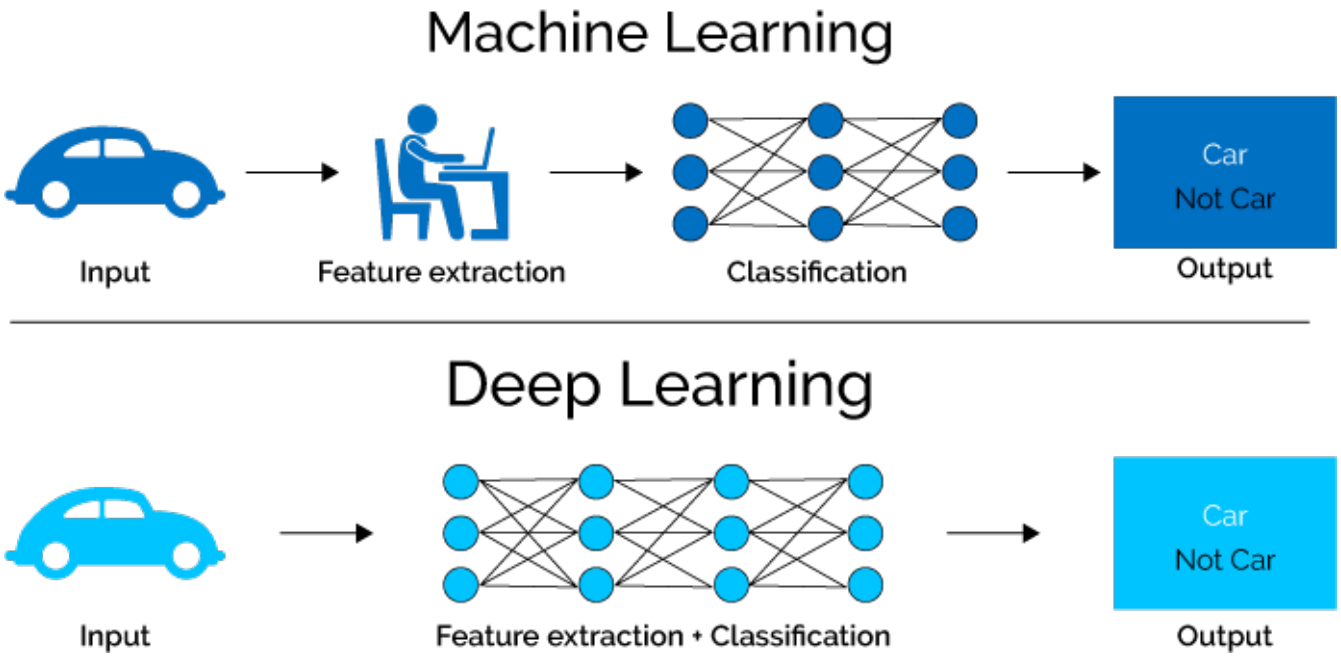


Learning algorithm

Feature representation

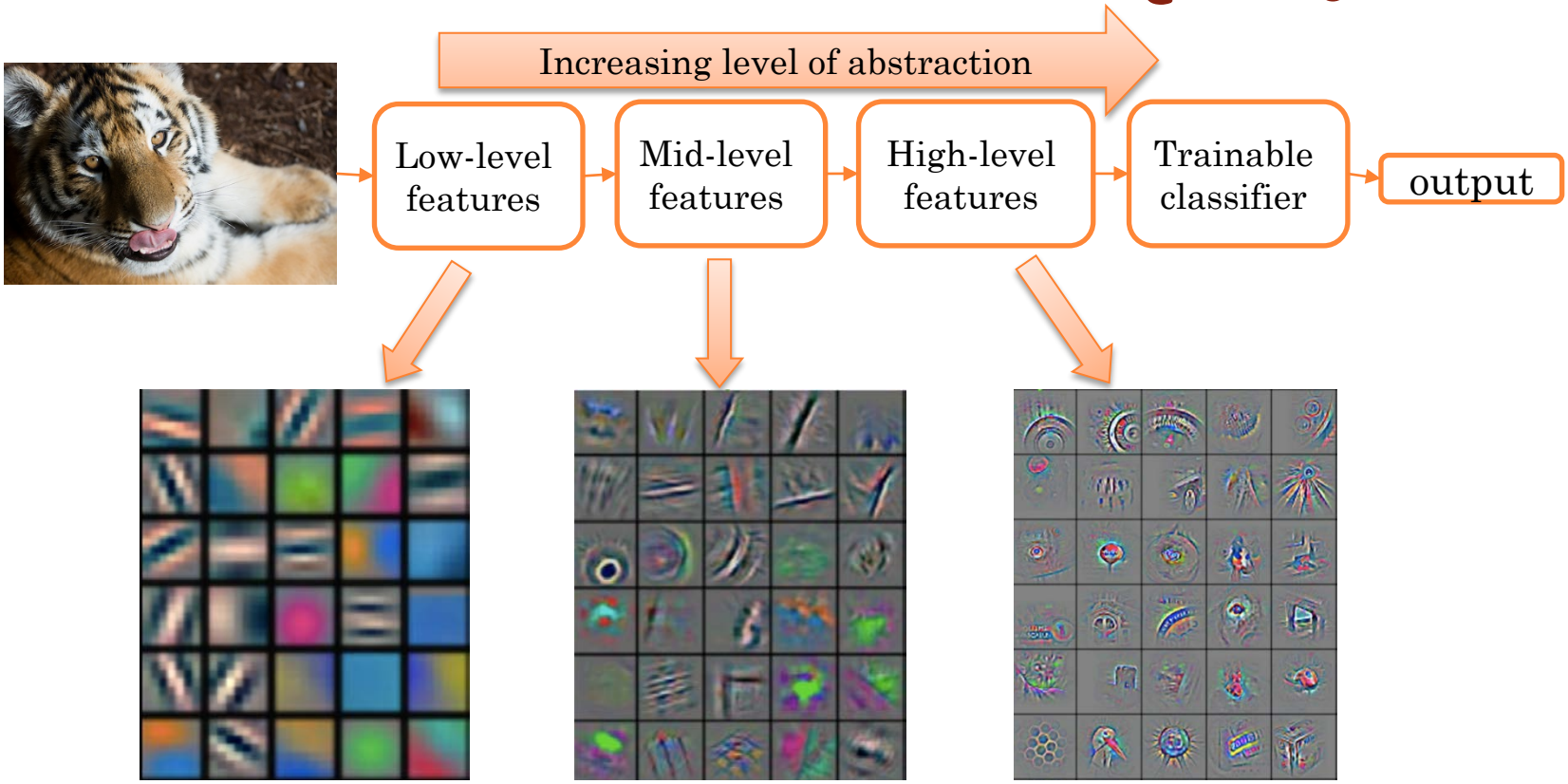
یادگیری عمیق . . .

- ترکیب استخراج ویژگی و دسته‌بندی



یادگیری عمیق: استخراج ویژگی . . .

- استخراج ویژگی سلسله مراتبی . . .
  - از ورودی خام (جزئیات)
  - تا ویژگی های سطح بالا (کلیات)



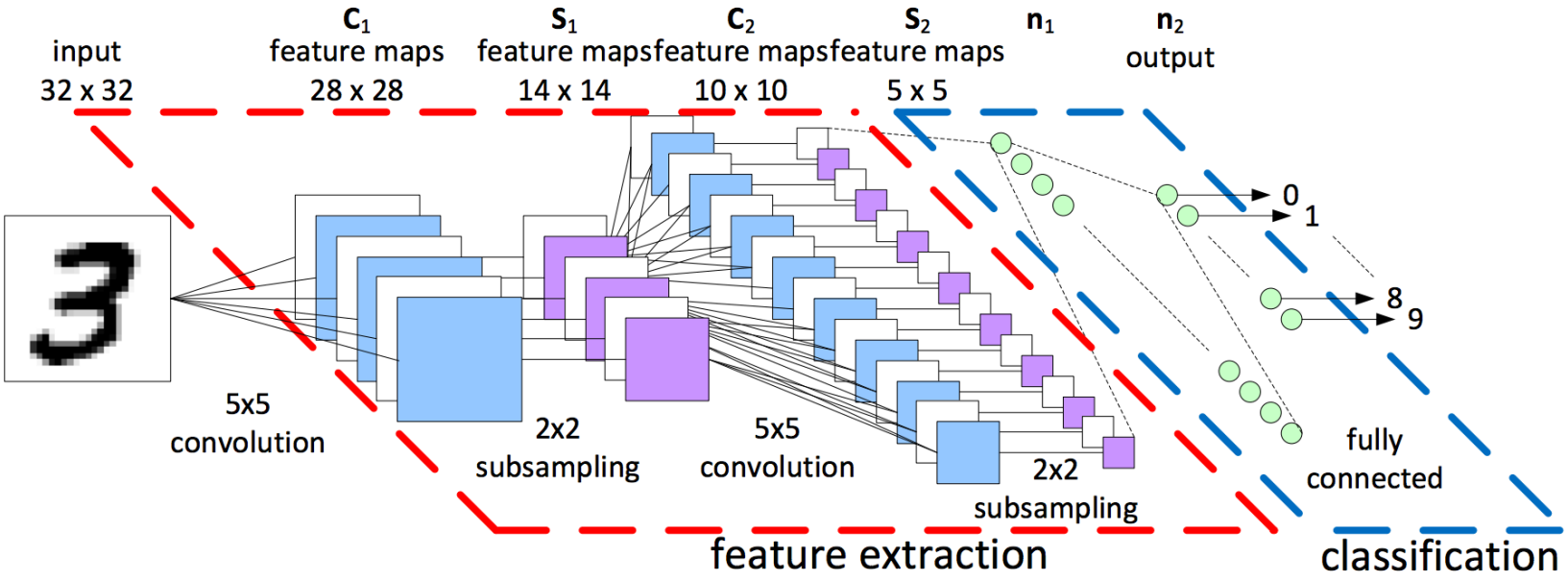
یادگیری عمیق: استخراج ویژگی

- استخراج ویژگی سلسله مراتبی

- تصویر
  - Pixel → edge → texton → motif → part → object
- متن
  - Character → word → word group → clause → sentence → story

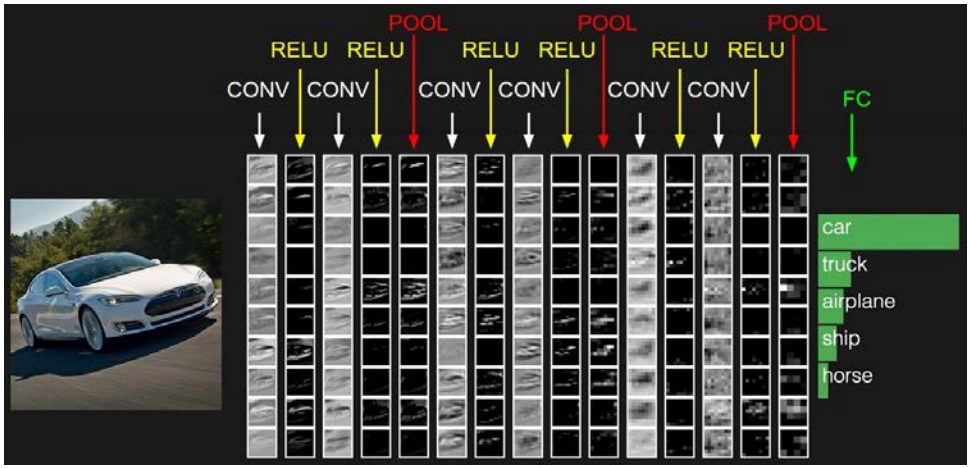
یادگیری عمیق: شبکه عصبی پیچشی . . .

- شبکه عصبی پیچشی (CNN: Convolutional Neural Network)
  - استخراج ویژگی از تصاویر (و دسته‌بندی)



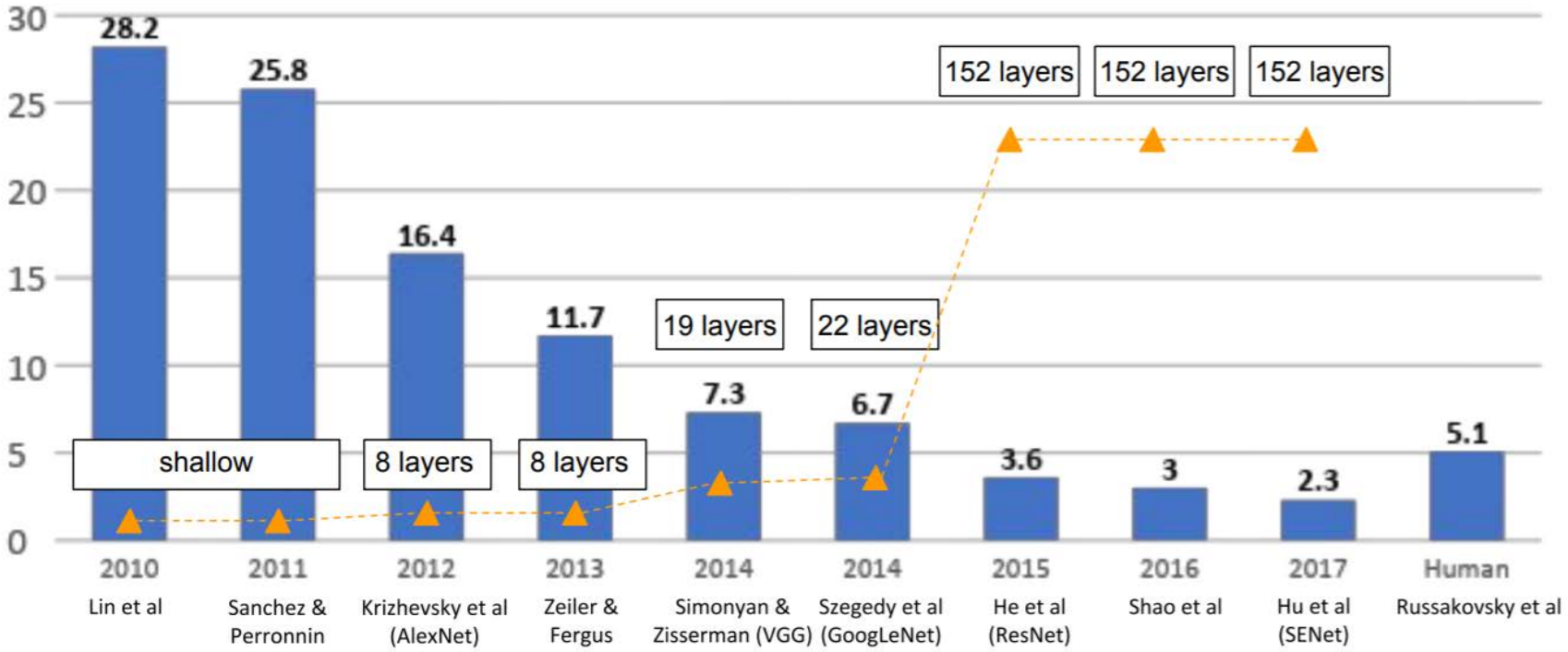
یادگیری عمیق: شبکه عصبی پیچشی ...

- لایه‌های رایج در این شبکه (به ترتیب)
  - ورودی (Input)
    - یک تصویر ۳ بعدی: طول، عرض و عمق (مثلا معادل ۳ برابر با سه مولفه RGB)
  - پیچش (Convolution)
    - هر نرون بلوکی از پیکسل‌های نزدیک به هم را می بیند
    - تابع فعال‌سازی ReLU
      - $f(x)=max(0,x)$
      - غیرخطی کردن رفتار شبکه
  - نمونه‌برداری (Pooling)
    - کاهش اندازه فضای ویژگی‌ها
  - تمام متصل (Fully Connected)
    - برای دسته‌بندی، مانند شبکه‌های عصبی MLP



یادگیری عمیق: شبکه عصبی پیچشی . . .

- نرخ خطای دسته‌بندی روی ImageNet
  - ۱.۲ میلیون تصویر در ۱۰۰۰ دسته
  - کارایی بالاتر از عملکرد انسان !



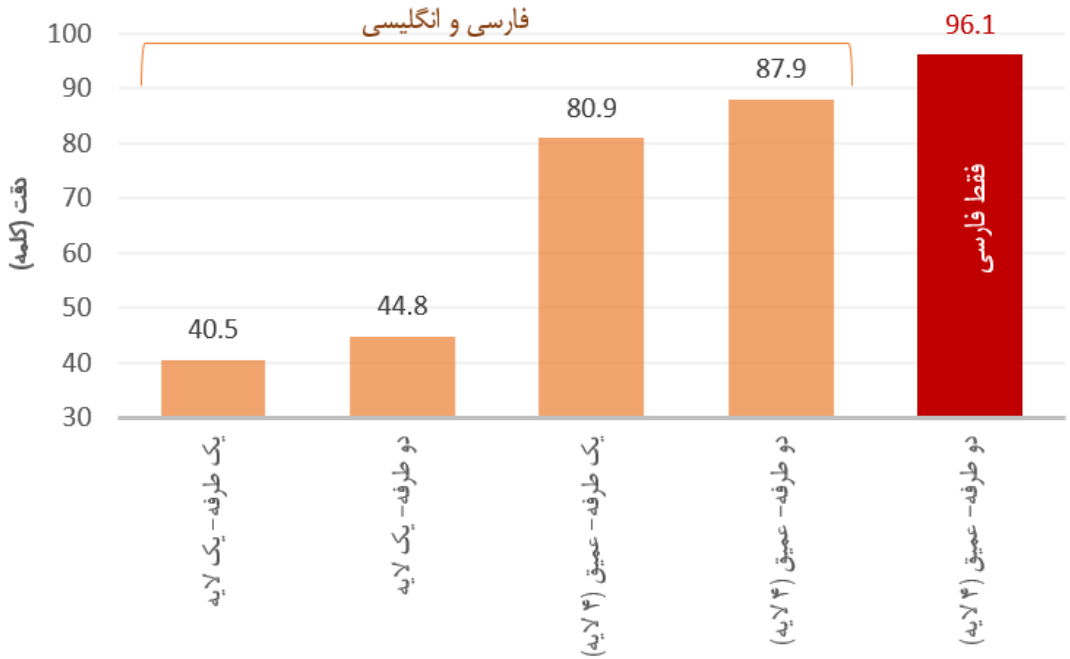
یادگیری عمیق: شبکه عصبی پیچشی . . .

- برای جزئیات بیشتر (آموزش و مثال) مراجعه کنید به

- http://dsp.ut.ac.ir/en/wp-content/uploads/2019/ANN-Lecture7-CNN.pdf

یادگیری عمیق: شبکه عصبی پیچشی (مثال) . . .

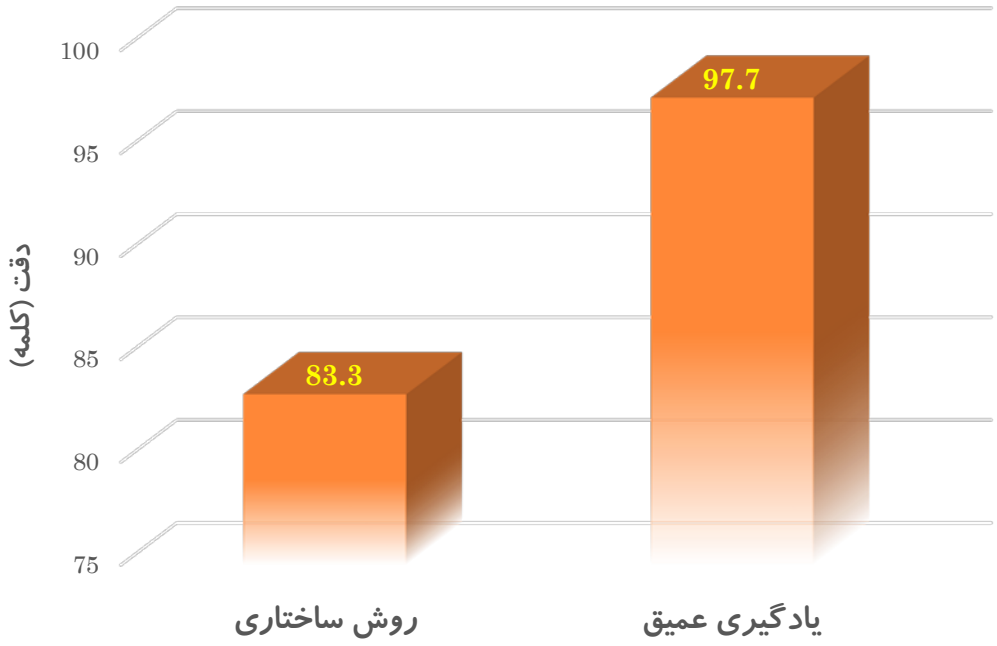
- نویسه خوان نوری فارسی
  - استخراج ویژگی با شبکه CNN و دسته بندی با شبکه LSTM
  - ۱۱ فونت رایج فارسی و سه اندازه ۱۲، ۱۴ و ۱۸
  - دادگان آموزش: ۲،۰۰۰،۰۰۰ تصویر
  - دادگان آزمون
    - فقط فارسی: ٤،٦٠٠ تصویر
    - فارسی-انگلیسی: ٤،٦٠٠ تصویر



زهره خسروبیگی، نویسه‌خوان نوری فارسی با استفاده از شبکه عصبی حافظه کوتاه‌مدت ماندگار، پایان‌نامه کارشناسی ارشد، دانشگاه تهران، ۱۳۹۷

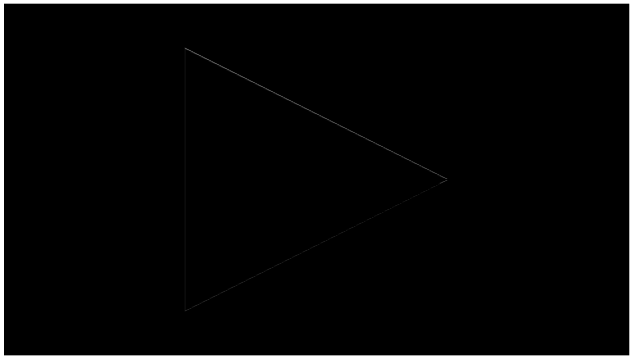
یادگیری عمیق: شبکه عصبی پیچشی (مثال) . . .

- نویسه خوان نوری فارسی: مقایسه یادگیری عمیق و روش سنتی
  - برای فونت B Mitra

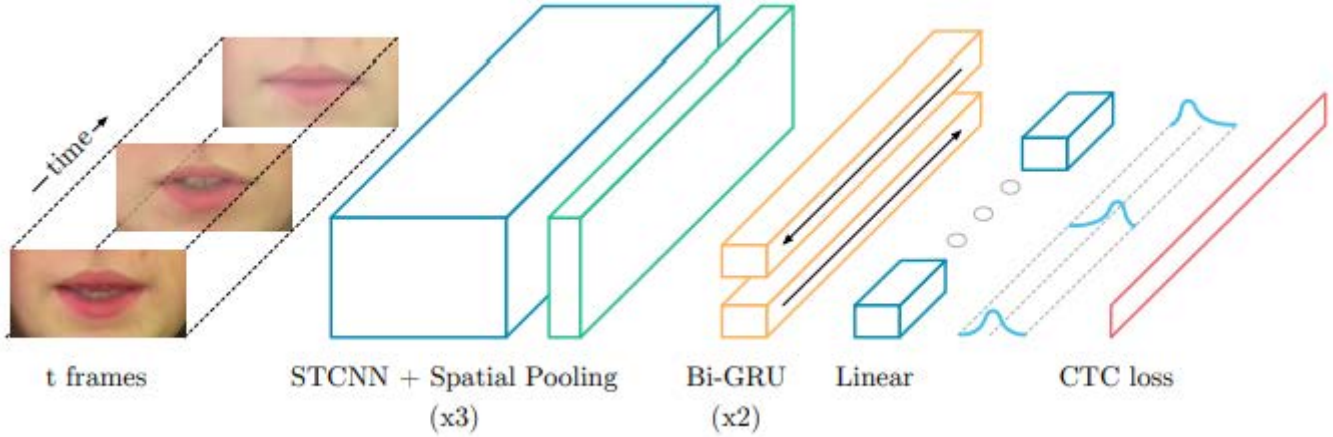


زهره خسرویگی، نویسه‌خوان نوری فارسی با استفاده از شبکه عصبی حافظه کوتاه‌مدت ماندگار، پایان‌نامه کارشناسی ارشد، دانشگاه تهران، ۱۳۹۷

یادگیری عمیق-شبکه عصبی پیچشی: لب‌خوانی



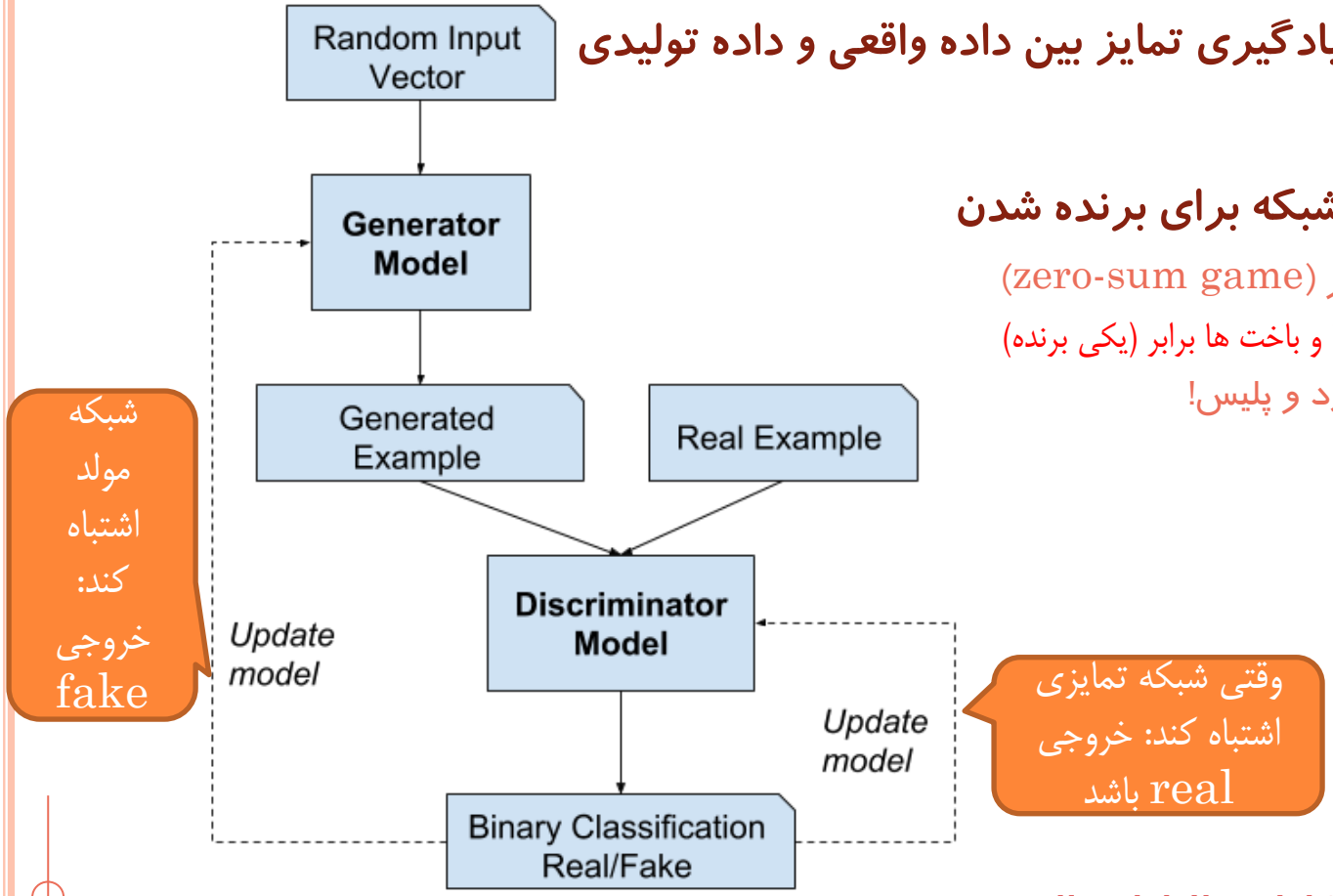
- مدل LipNet
  - CNN
  - (GRU) Gated Recurrent Unit
  - CTC loss



[Assael, Yannis M., et al. "Lipnet: End-to-end sentence-level lipreading.“, 2016]

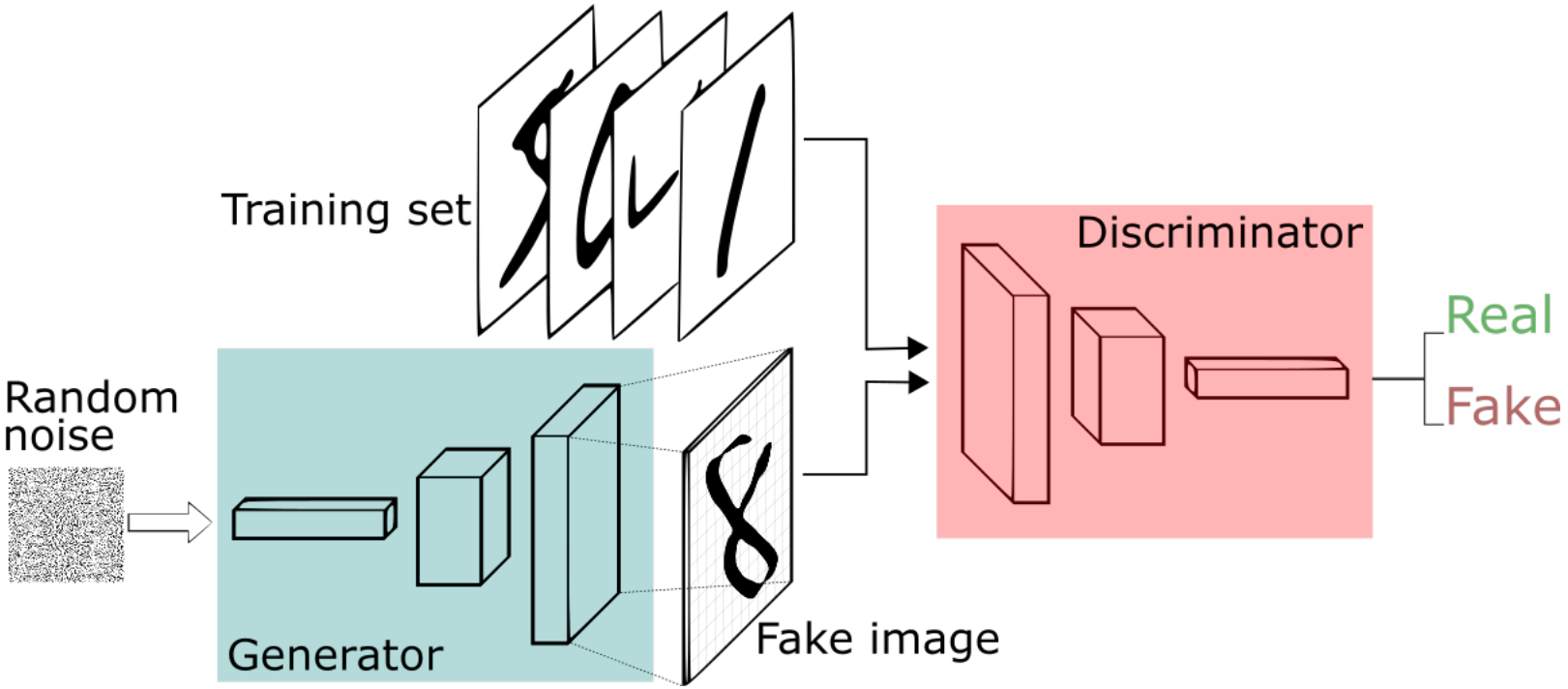
یادگیری عمیق: شبکه GAN ...

- شبکه مولد مقابله‌ای (GAN: Generative Adversarial Networks)
  - شبکه مولد: تولید داده مصنوعی
  - شبکه تمایزی: یادگیری تمایز بین داده واقعی و داده تولیدی
  - مسابقه بین دو شبکه برای برنده شدن
    - بازی جمع صفر (zero-sum game)
      - مجموع بردها و باخت ها برابر (یکی برنده)
    - مسابقه بین دزد و پلیس!



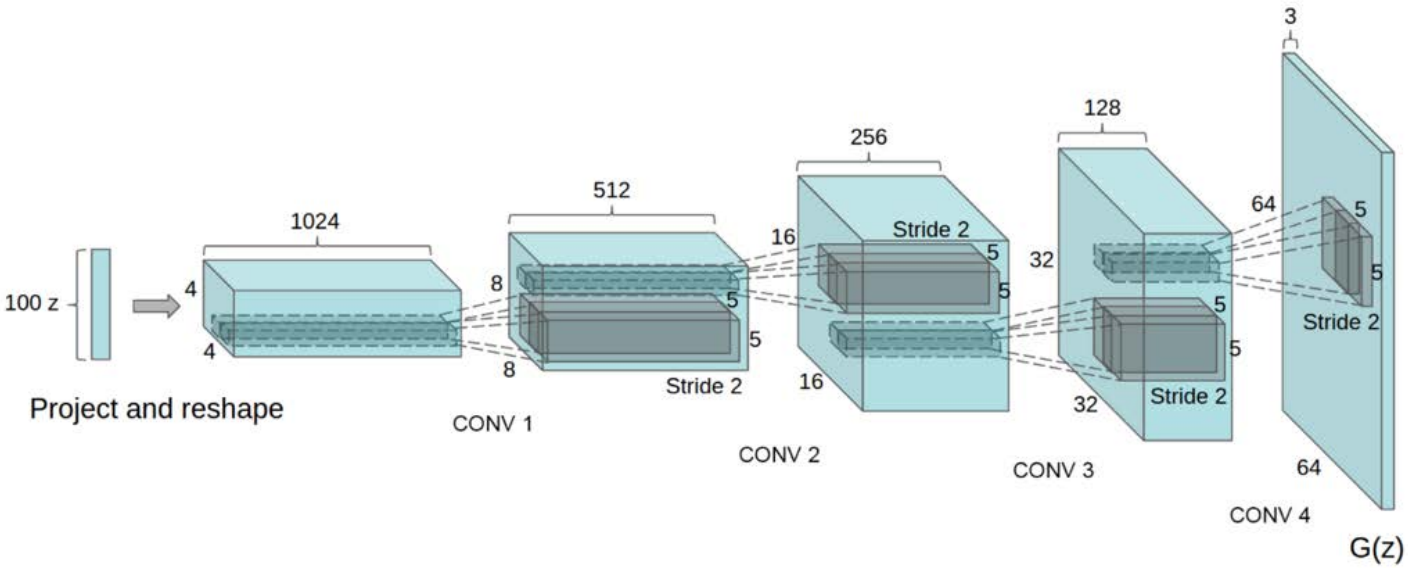
یادگیری عمیق: شبکه GAN ...

- شبکه مولد مقابله‌ای
  - حالت ایده آل: خروجی برای دو حالت خروجی اطمینان ۵۰٪ داشته باشد
  - در پایان آموزش، شبکه تمایزی دور ریخته می شود



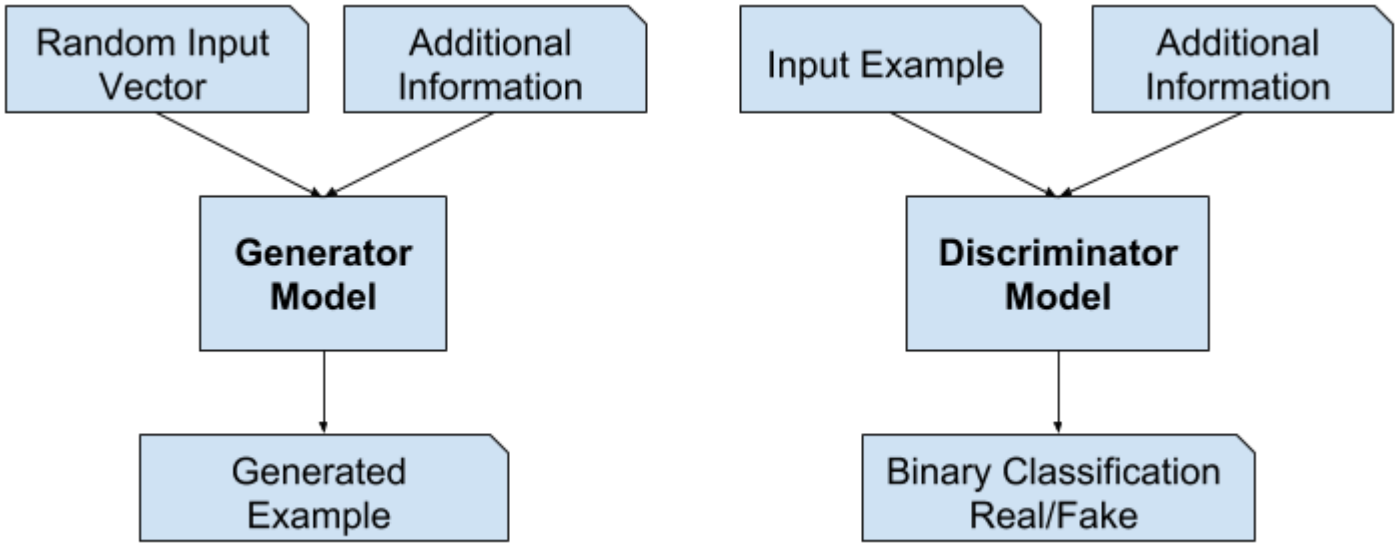
یادگیری عمیق: شبکه GAN . . .

- شبکه مولد در DCGAN
  - Deep Convolutional Generative Adversarial Networks



یادگیری عمیق: شبکه GAN . . .

- شبکه GAN شرطی (Conditional GAN)
  - **تولید یا تمایز مشروط است (مثلا به دادن داده اضافی مانند اسم دسته)**
    - اگر دسته زن و مرد باشد، این شرط باعث تولید چهره زن/مرد می شود
    - شرط می تواند خودش یک نمونه داده باشد، مثل تصویر زمستان یا بهار برای تولید تصویر مرتبط با آن تصویر (فصل)
      - قابل استفاده در ترجمه های تصویر به تصویر مانند یادگیری رنگ کردن تصویر، یادگیری سبک نویسنده/گوینده/نقاش



یادگیری عمیق: شبکه GAN . . .

- تولید چهره . . .

Examples of Photorealistic GAN-Generated Faces.Taken from Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017.

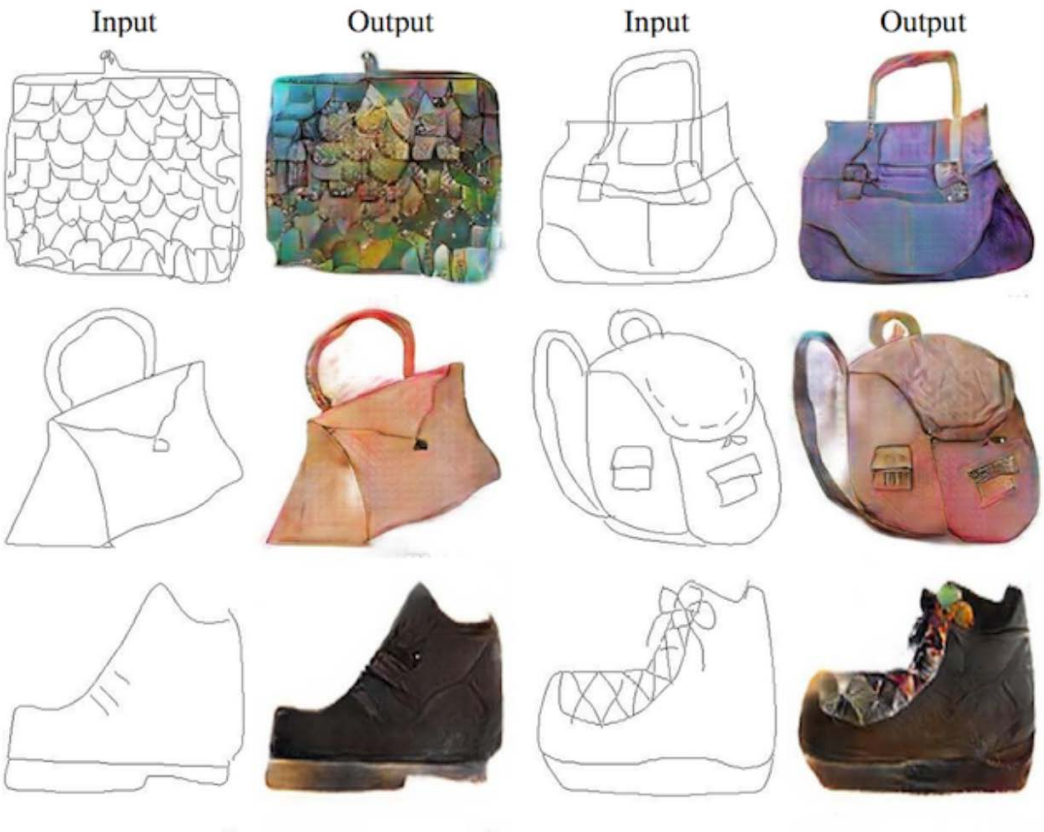
یادگیری عمیق: شبکه GAN . . .

- تولید چهره

Example of the Progression in the Capabilities of GANs from 2014 to 2017.Taken from The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation, 2018.

یادگیری عمیق: شبکه GAN . . .

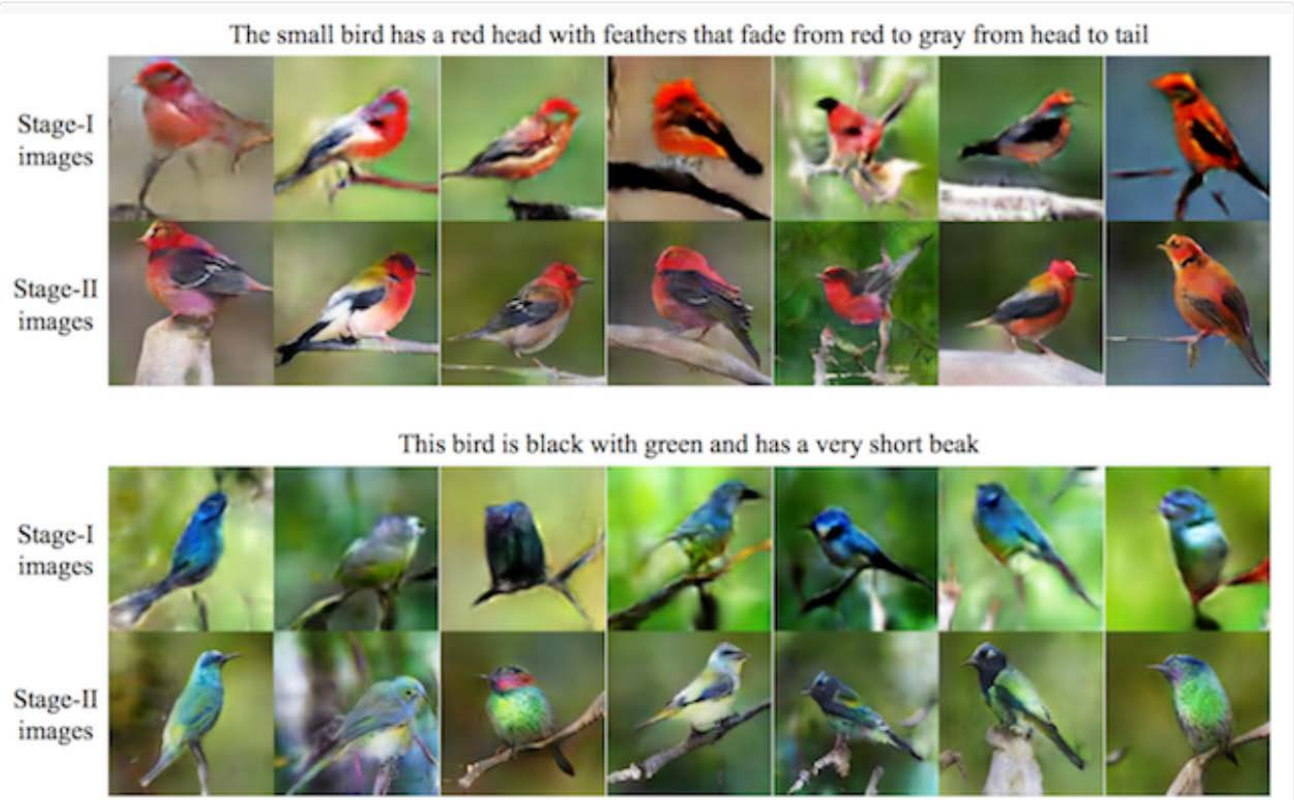
- ترجمه تصویر به تصویر . . .



Example of Sketches to Color Photographs With pix2pix.Taken from Image-to-Image Translation with Conditional Adversarial Networks, 2016.

یادگیری عمیق: شبکه GAN . . .

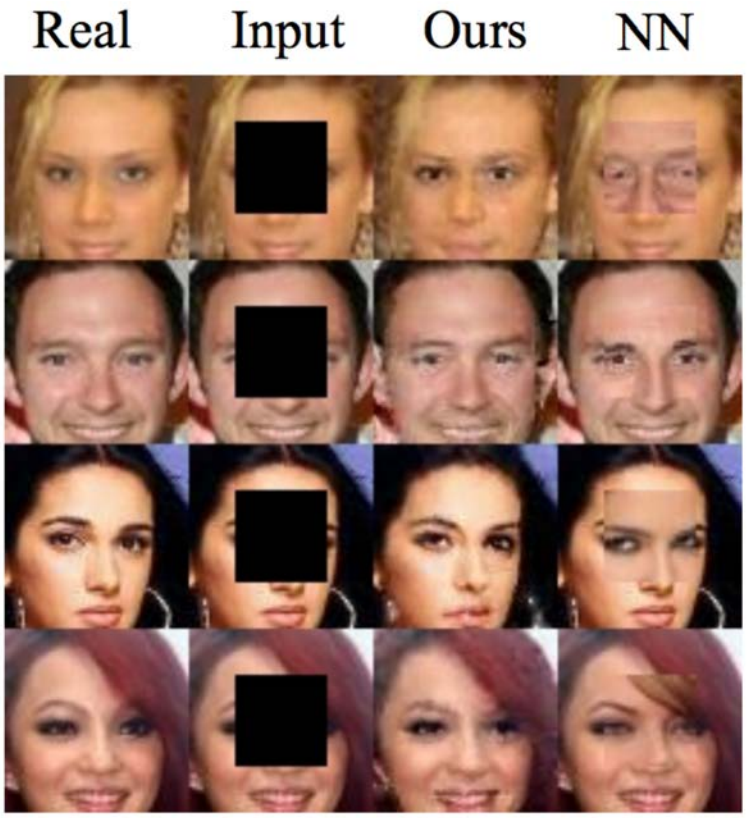
- تبدیل متن به تصویر (text2image)



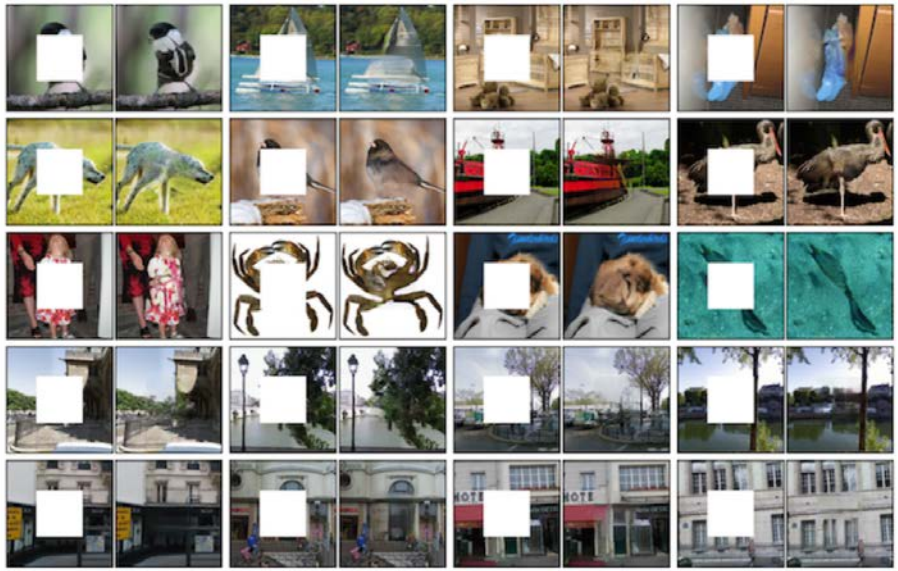
Example of Textual Descriptions and GAN-Generated Photographs of BirdsTaken from StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, 2016.

یادگیری عمیق: شبکه GAN . . .

- بازسازی عکس (Photo Inpainting)
  - بازسازی ناحیه از دست رفته در تصویر



Example of GAN-based Inpainting of Photographs of Human FacesTaken from Semantic Image Inpainting with Deep Generative Models, 2016.



Example of GAN-Generated Photograph Inpainting Using Context Encoders.Taken from Context Encoders: Feature Learning by Inpainting describe the use of GANs, specifically Context Encoders, 2016.

یادگیری عمیق: شبکه GAN

تولید متن فارسی

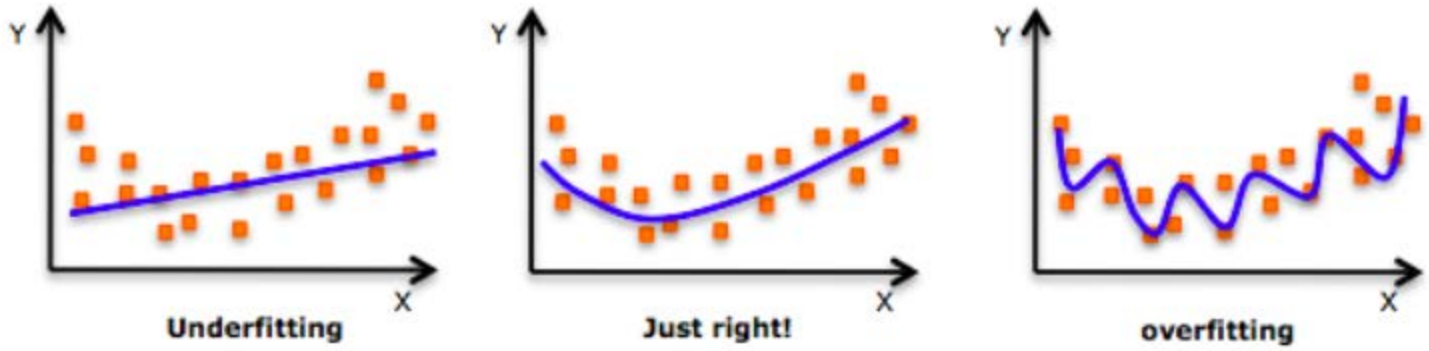
- شعر فردوسی

- به پیش سپاه اندر امد ز جای	که ای نامور شاه را دل ز جای
- به پیران چنین گفت کای پرخرد	که از من بدین داستان بگذرد
- بدو گفت بهرام کای نامجوی	به دل گفت کای در جهان کینه جوی
- برامد ز ایوان شاه امدند	به نزدیک او با سپاه امدند
- به کار اندر امد ز دریای خون	بگویم که باشد به دل پر ز خون
- بدو گفت بر من بدی در جهان	زمین را همی داشت اندر نهان
- بدین گونه با او سپه را بخواند	سوی شاه گردن فراوان براند
- به روی اندر امد ز من در جهان	ز توران به درگاه شاه جهان
- چو بشنید ازو بازگشتن به راه	بیاراست لشکر برامد ز راه
- بدو گفت بازار ایران زمین	که ای پهلوان سر به سر بر زمین
- به داد و به دیدار او شد سپاه	به پیش سپاهش به تخت و کلاه
- بران کوه بر شد ز هر سو سپاه	که بیژن نیاید به پیش سپاه
- به پیش اندر امد سوی کارزار	که از کار او در جهان کارزار
- به ایوان بیاراست گرد سپاه	چو خورشید تابان به درگاه شاه

احسان منتهایی، مهدیه سلیمانی باغشاه، تولید متن با رویکرد جمعی مبتنی بر شبکه‌های مولد مقابله‌ای محورهای موضوعی، مهندسی برق و کامپیوتر ایران، 1399

یادگیری عمیق: تنظیم ...

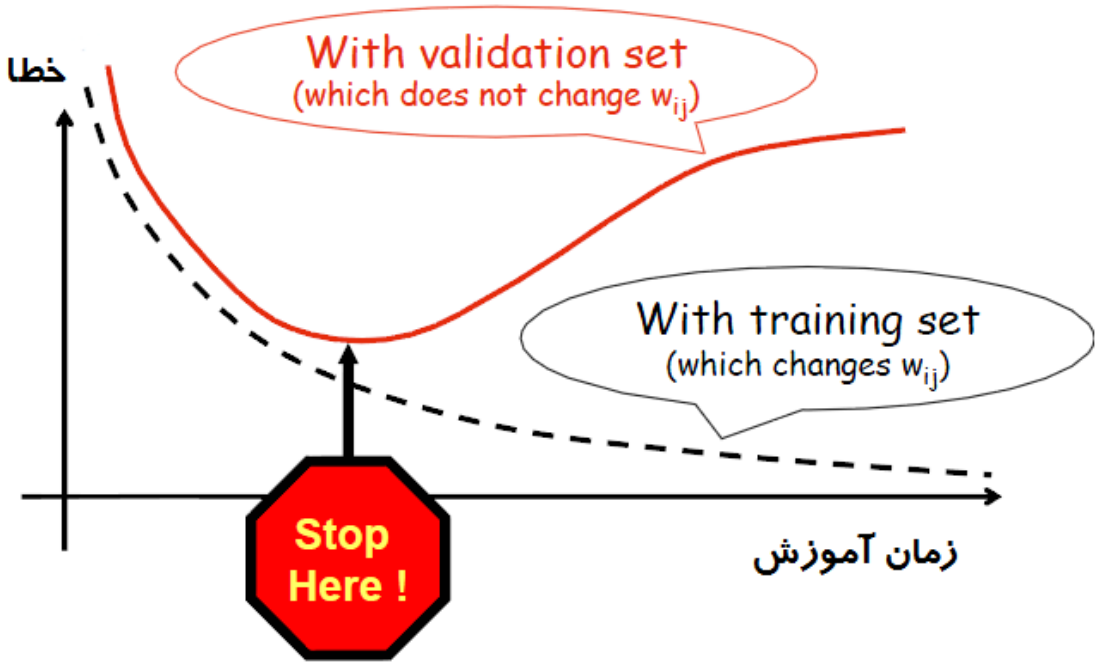
- تنظیم شبکه (Regularization)
  - روش‌هایی برای افزایش قدرت تعمیم پذیری شبکه و جلوگیری از بیش برازش



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

یادگیری عمیق: تنظیم ...

- توقف زودهنگام (Early Stopping)
  - جلوگیری از تکرار زیاد و وابسته شدن کامل شبکه به داده آموزشی
  - استفاده از مجموعه اعتبارسنجی یا اعتبارسنجی متقاطع
    - توقف در صورت افزایش خطا روی داده اعتبارسنجی



یادگیری عمیق: تنظیم . . .

- تغییر تابع هزینه
  - افزودن مقداری به مقدار خطای مورد استفاده در تابع هزینه

- $Cost\ function = Loss + Regularization\ term$

مقدار خطای MSE

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

ضریب تنظیم

- تنظیم L2

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

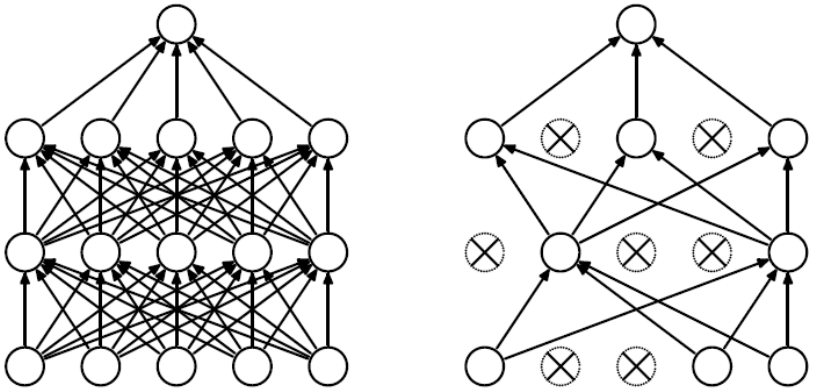
  - به آن weight decay هم می‌گویند
  - اثر کاهش وزن به صورت W += -λ* W به سمت صفر

- تنظیم L1

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

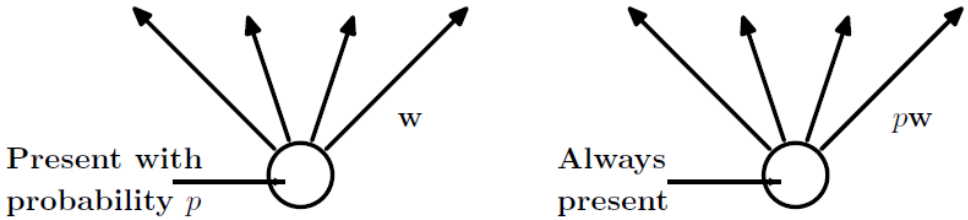
یادگیری عمیق: تنظیم ...

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



(a) Standard Neural Net (b) After applying dropout.

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



(a) At training time (b) At test time

- عملکرد مشابه آموزش گروهی (Ensemble Learning)

یادگیری عمیق: تنظیم

- تقویت داده (Data Augmentation)
  - افزایش تنوع و حجم داده‌ها به صورت مصنوعی

- برای تصویر
  - جابجایی، تغییر مقیاس، معکوس، چرخاندن، برش و ...

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

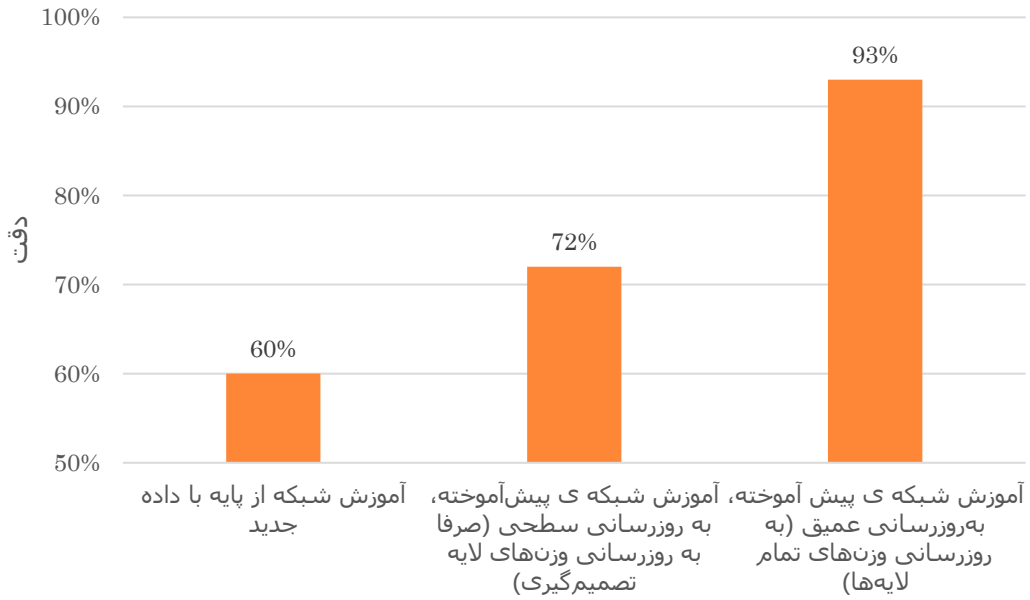
یادگیری عمیق: پیش آموزش . . .

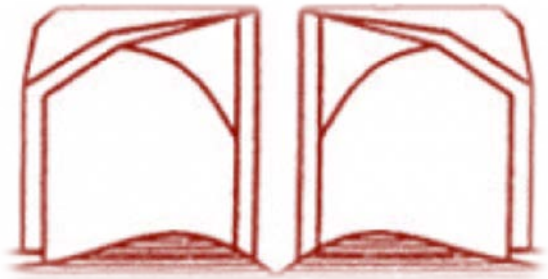
- ایده پیش آموزش (Pre-Train)
  - آموزش شبکه با داده (حجیم) از یک حوزه و انتقال وزن‌های آموزش دیده برای استفاده از حوزه دیگر (به عنوان وزن‌های اولیه)
    - انتقال وزن‌های مربوط به لایه‌های استخراج ویژگی
    - صرفنظر کردن از لایه دسته بندی
  - بهینه کردن وزن‌های انتقال یافته با داده حوزه جدید (Fine-tune)

- مرتبط با مفهوم یادگیری انتقالی (Transfer Learning)

یادگیری عمیق: پیش آموزش

- **مثال: دسته‌بندی تصویر بیماری گیاهان**
  - **وزن اولیه: از شبکه عصبی پیچشی AlexNet آموزش داده شده با داده ImageNet**
    - ۱.۲ میلیون تصویر با ۱۰۰۰ دسته مختلف از موضوعات مختلف
  - **حوزه جدید: ۱۰ دسته (بیماری)، حدود ۳۵۰ الی ۶۰۰ تصویر برای هر دسته**





یادگیری ماشین

شبکه عصبی بازگشتی

شبکه عصبی بازگشتی . . .

عقیده کاوی (دسته‌بندی نظرات کاربران)

موافق/مخالف

امید دلبر (1393/1/24) : 1

فوق العادس

محمود ابراهیمی جاویدی (1393/1/22) : 0

من این لپ تاپ 5 ماه دارم و از همه نظر عالی مثل: کیفیت صفحه نمایش یا کیفیت لولاش برای لمسی بودنش و...

مرتضی اکبری (1393/1/17) : 1

یکی از مشکلاش ازنظر من کارت گرافیکشه.
منظورم باس کارت گرافیکه!

هادی کریم (1392/12/22) : 0

سلام..اصلا تو خریدنش شک نکنید.من خریدم خیلی عالیه.خیلی بیشتر از عکسش خوشگله.واقعا زیباست.اصلا لولایی دیده نمیشه واستحکامش خیلی بیشتره.به قیمتش می ارزه.صدای فنش هم اصلا اذیت نمیکنه.جون میده برای بازی

ابوذر هومن (1392/12/17) : 5

با سلام
بعد از 4ماه کار مداوم
از هر نظر عالی
قدرت خوبی داره.بسیار سریع
تاچ خیلی خوبی داره.کیفیت واقعا عالی.صفحه عجیبی داره که اصلا خسته نمیشین موقع کار باهاش
وزنش هم که عالیه.یه همراه همیشگی میشه براتون
مرسی

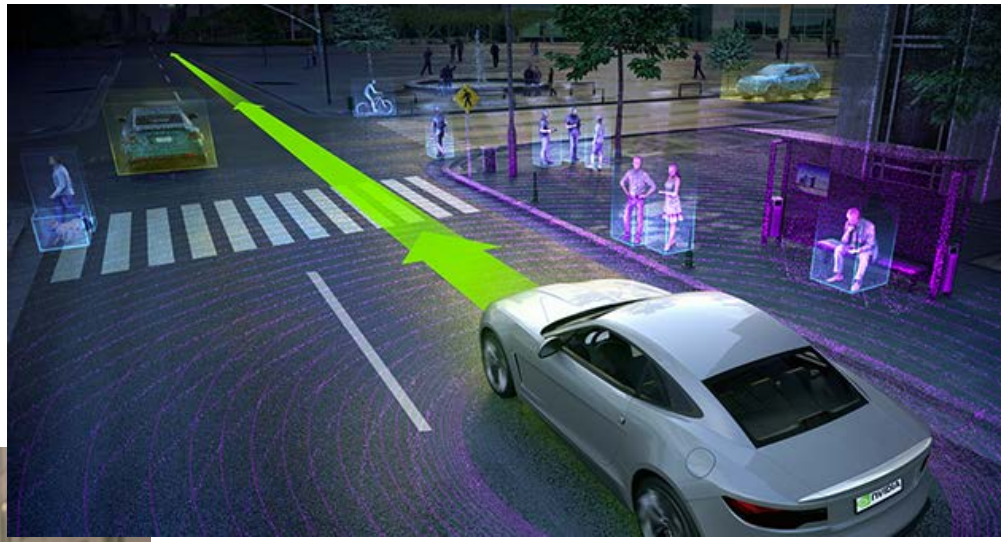
هادی کریم (1392/12/1) : 0

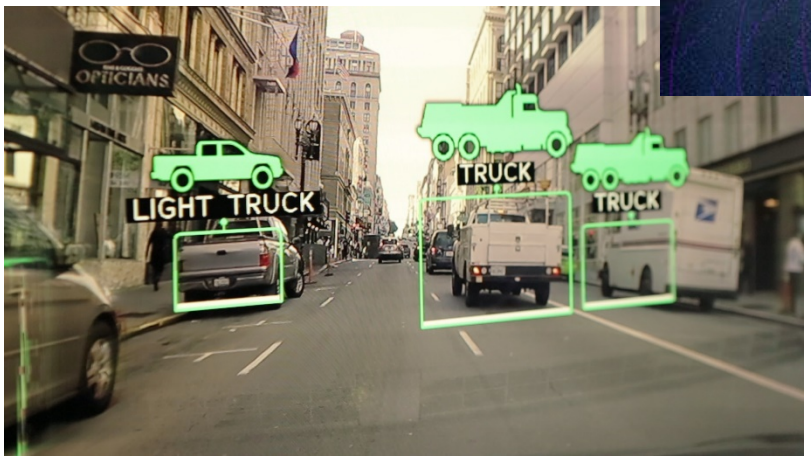
سلام..اصلا تو خریدنش شک نکنید.من خریدم خیلی عالیه.خیلی بیشتر از عکسش خوشگله.واقعا زیباست.اصلا لولایی دیده نمیشه واستحکامش خیلی بیشتره.به قیمتش می ارزه.صدای فنش هم اصلا اذیت نمیکنه.جون میده برای بازی

نحوه دسته بندی با شبکه عصبی؟

شبکه عصبی بازگشتی . . .

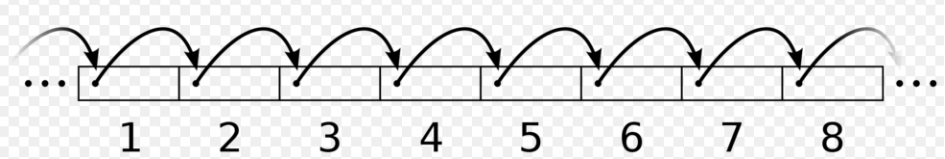
- خودروهای خودران!





شبکه عصبی بازگشتی . . .

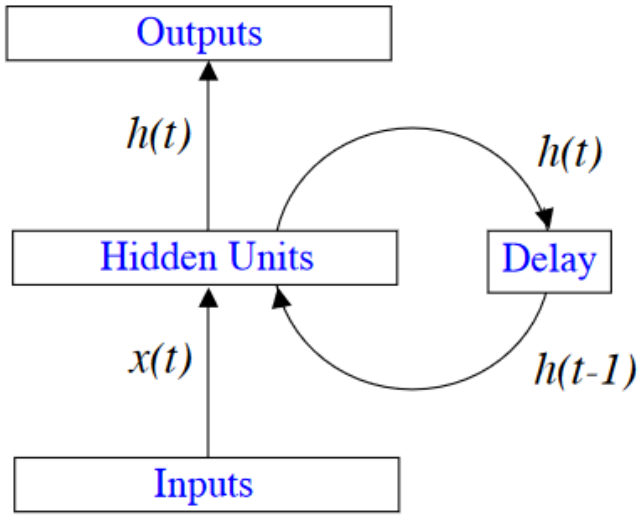
- داده متوالی (Sequential Data): داده‌هایی که مقدار فعلی آنها به مقادیر قبلی وابسته است
  - فریم‌های (نمونه‌های) سیگنال گفتار
  - فریم‌های (تصاویر) متوالی ویدئو
  - وضعیت آب و هوا
  - قیمت سهام یک شرکت/صنعت
  - دنباله‌های تولید شده توسط گرامرها
    - کلمات داخل یک متن
  - ...



شبکه عصبی بازگشتی . . .

- **شبکه‌های عصبی بازگشتی (RNN: Recurrent Neural Networks)**
  - شبکه‌هایی که در ساختار آنها یال‌های بازگشت کننده وجود دارد
  - بر خلاف شبکه‌های عصبی رو به جلو، یال‌ها می‌توانند تشکیل دور بدهند.

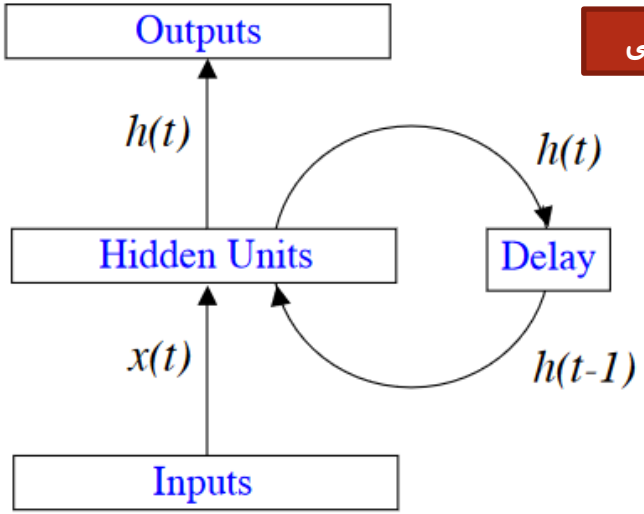
- به دلیل داشتن یال بازگشتی در ساختار خود، قدرت **حافظه**‌ای دارند.
  - مناسب برای پردازش داده‌های متوالی (Sequential Data)



- ساختار: شبیه MLP با بازگشت از نرون های 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



فعال سازی لایه مخفی

وزن لایه ورودی به مخفی

وزن لایه مخفی به مخفی

$$h(t) = f_H\left(W_{IH}x(t) + W_{HH}h(t-1)\right)$$

$$y(t) = f_O\left(W_{HO}h(t)\right)$$

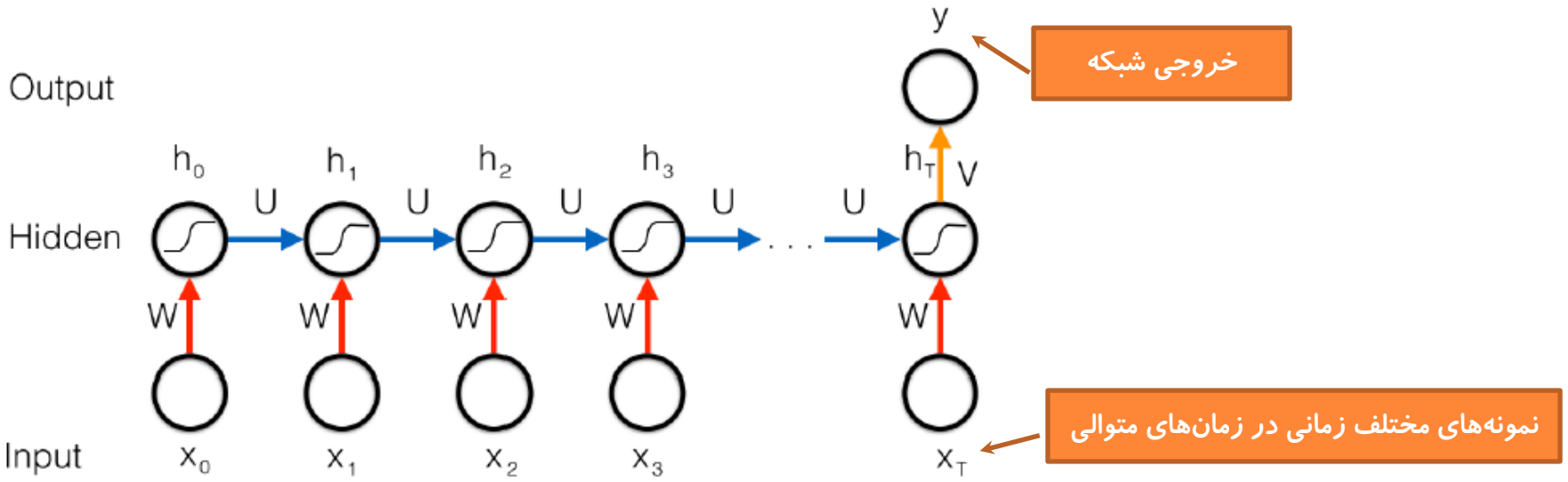
فعال سازی لایه خروجی

وزن لایه مخفی به خروجی

شبکه عصبی بازگشتی: آموزش ...

عملکرد شبکه

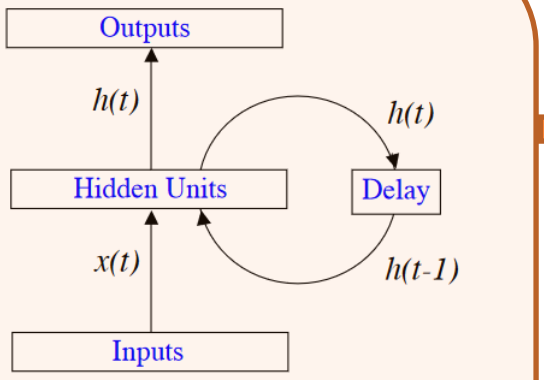
- وابستگی خروجی شبکه به خروجی‌های لایه مخفی به ازای همه ورودی‌ها



$$f(x) = Vh_T$$
$$h_t = \sigma(Uh_{t-1} + Wx_t), \text{ for } t = T, .., 1$$
$$\dots$$
$$h_0 = \sigma(Wx_0)$$

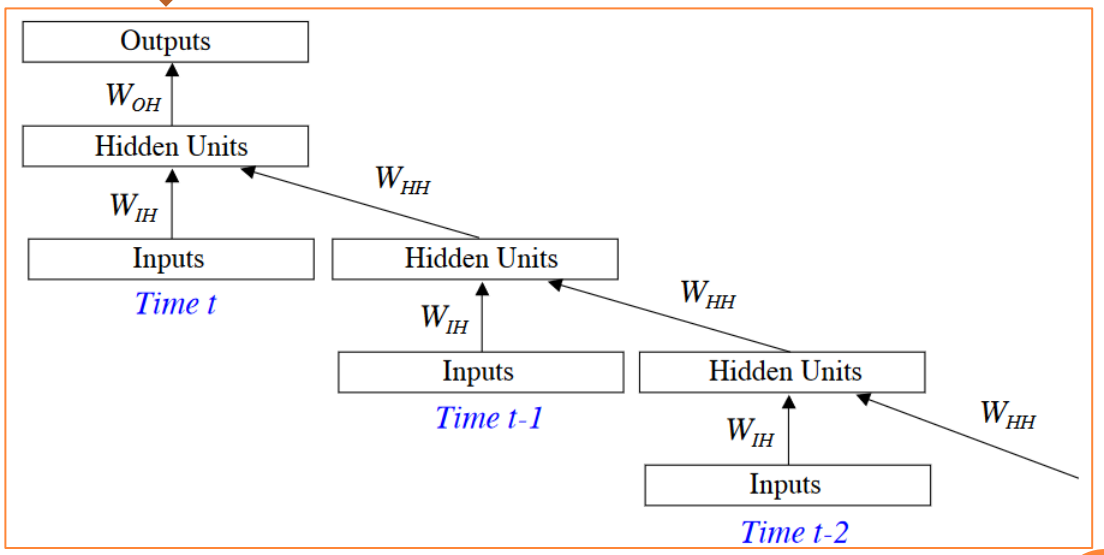
شبکه عصبی بازگشتی: آموزش ...

- 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)



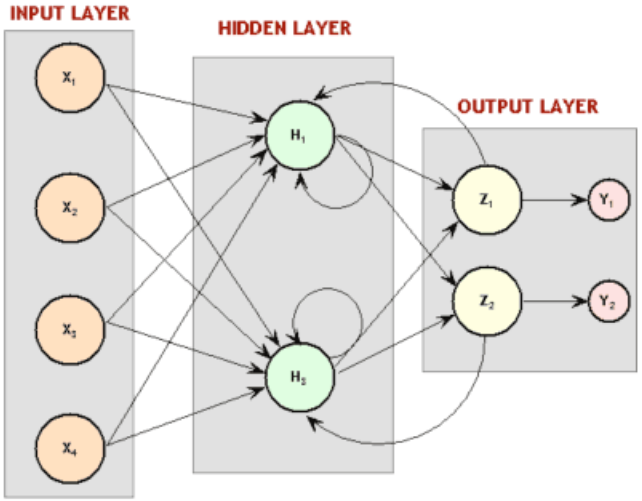
$$h(t) = f_H\left(W_{IH}x(t) + W_{HH}h(t-1)\right)$$

$$y(t) = f_O\left(W_{HO}h(t)\right)$$

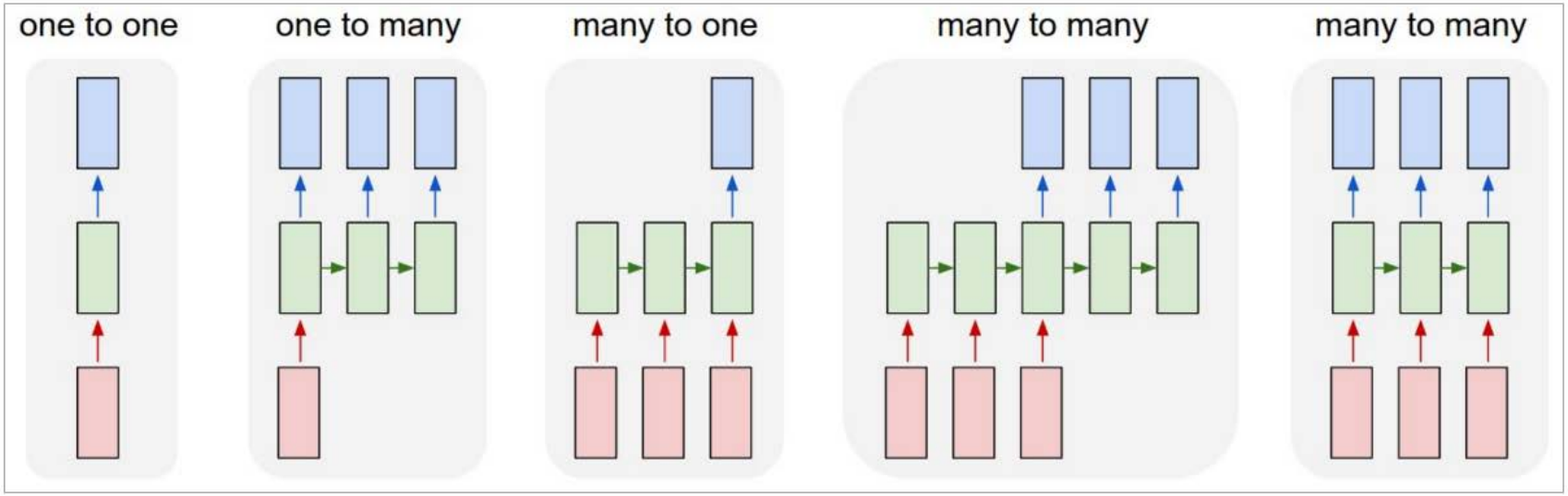


شبکه عصبی بازگشتی: کاربردها . . .

- مدل‌سازی و پیش‌بینی داده‌های متوالی (Sequential Data) و سری زمانی (Time Series)
  - سیگنال گفتار: تشخیص گفتار
  - دست‌نوشته: تبدیل دست نوشته به متن الکترونیکی
  - امضا: تایید/تشخیص هویت
  - دنباله نویسه‌ها یا واژه‌ها در یک متن: تشخیص موضوع متن
  - تصاویر پشت سر هم (ویدئو): دنبال کردن اشیا
  - پیش‌بینی نرخ ارز/شاخص سهام
  - پیش‌بینی بارش باران در روزهای متوالی سال
  - پیش‌بینی مقدار مصرف آب در روزهای متوالی
  - ...



شبکه عصبی بازگشتی: انواع استفاده . . .



ورودی ثابت و خروجی ثابت
مثال: دسته‌بندی تصویر

ورودی ثابت و خروجی متغیر
مثال: عنوان گذاری تصویر

ورودی متغیر و خروجی ثابت
مثال: تحلیل احساس در متن

ورودی متغیر و خروجی متغیر
مثال: ترجمه ماشینی متن

ورودی متغیر و خروجی متغیر (همزمان)
مثال: تعیین نقش دستوری کلمات در متن

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

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

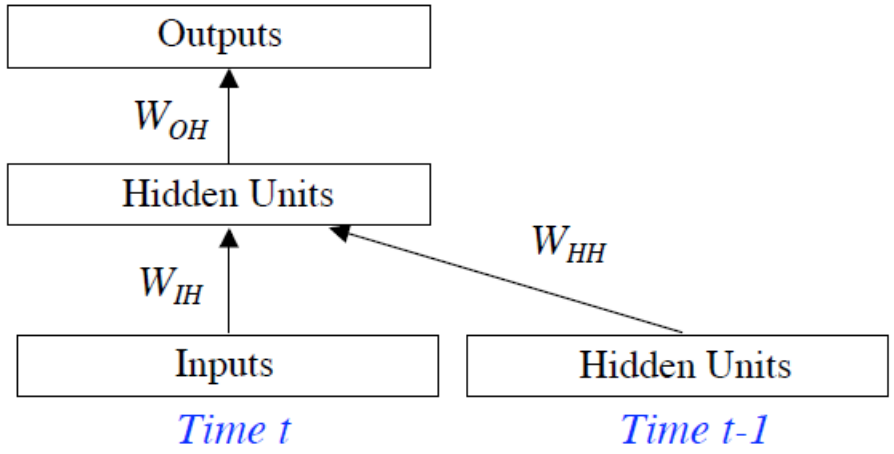
  - گرادیان وزن‌های شبکه برای بدست آوردن مقدار تغییرات وزن

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

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

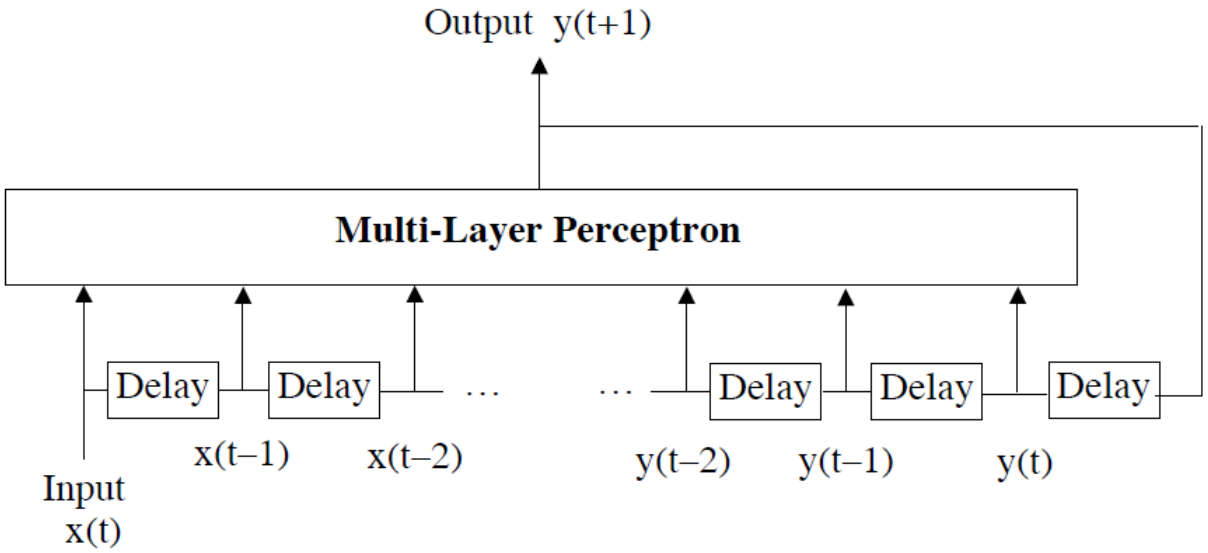
شبکه عصبی بازگشتی: آموزش ...

- **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 (مثلا 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

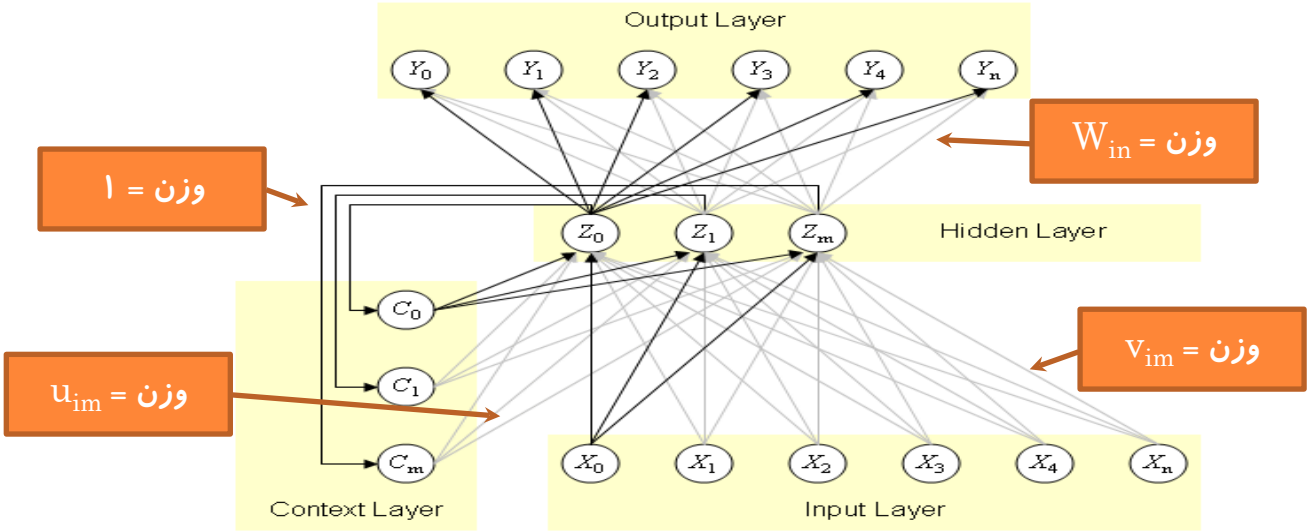


- این ساختار برای پیش‌بینی سری‌های زمانی مناسب است

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
  - وزن یال‌هایی که لایه پنهان را به لایه بافت متصل می‌کنند برابر مقدار ثابت یک می‌باشد



شبکه عصبی المان: الگوریتم آموزش و کاربرد

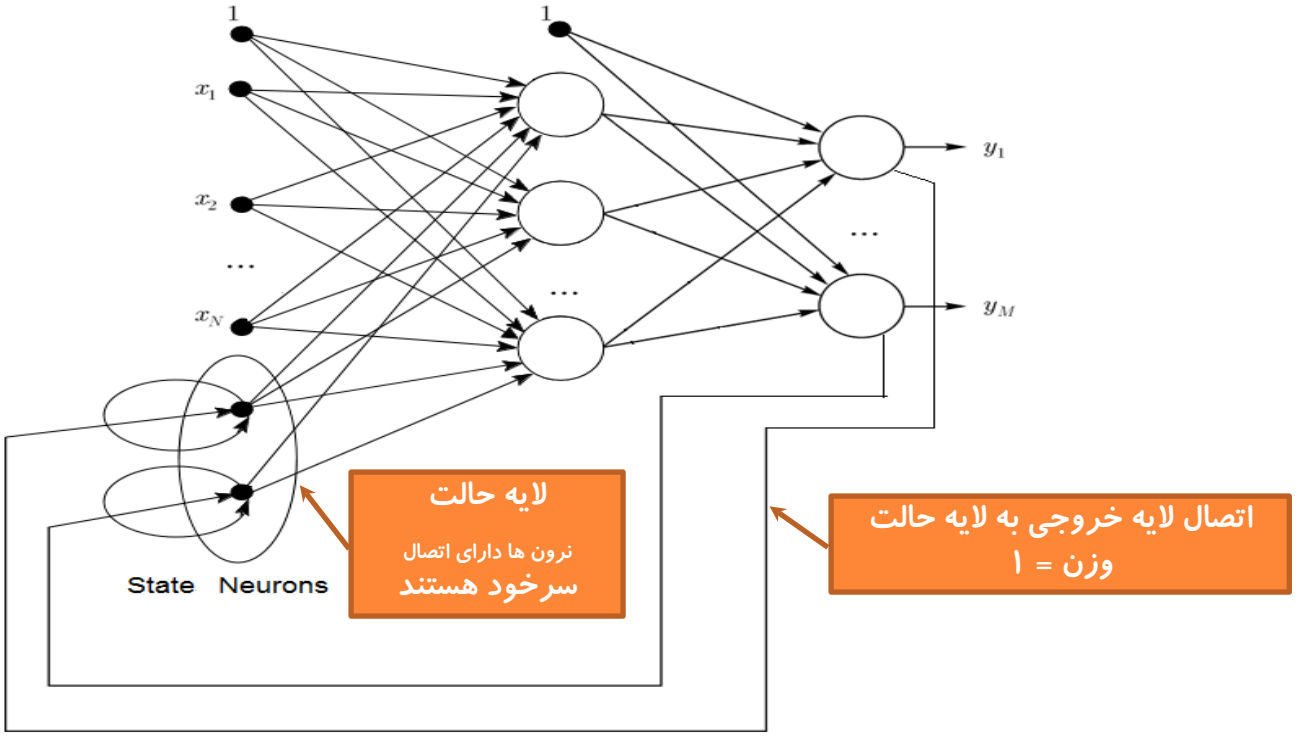
- مراجعه کنید به

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

شبکه عصبی جردن ...

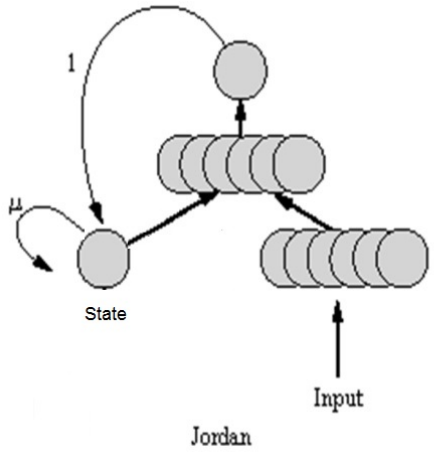
معرفی و ساختار

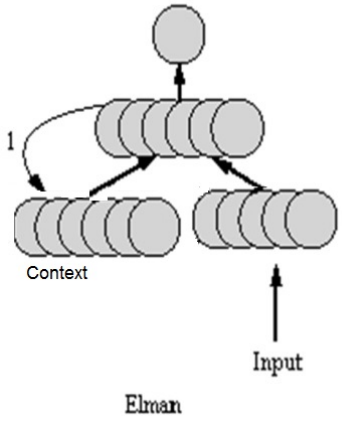
- ارائه شده در سال ۱۹۹۶ توسط مایکل جردن
- دارای شباهت بسیار زیاد به شبکه عصبی المان
- شبکه دارای اتصالات بازگشتی از لایه خروجی به لایه حالت و همچنین از لایه حالت به خودش می‌باشد



شبکه عصبی جردن: تفاوت با شبکه المان

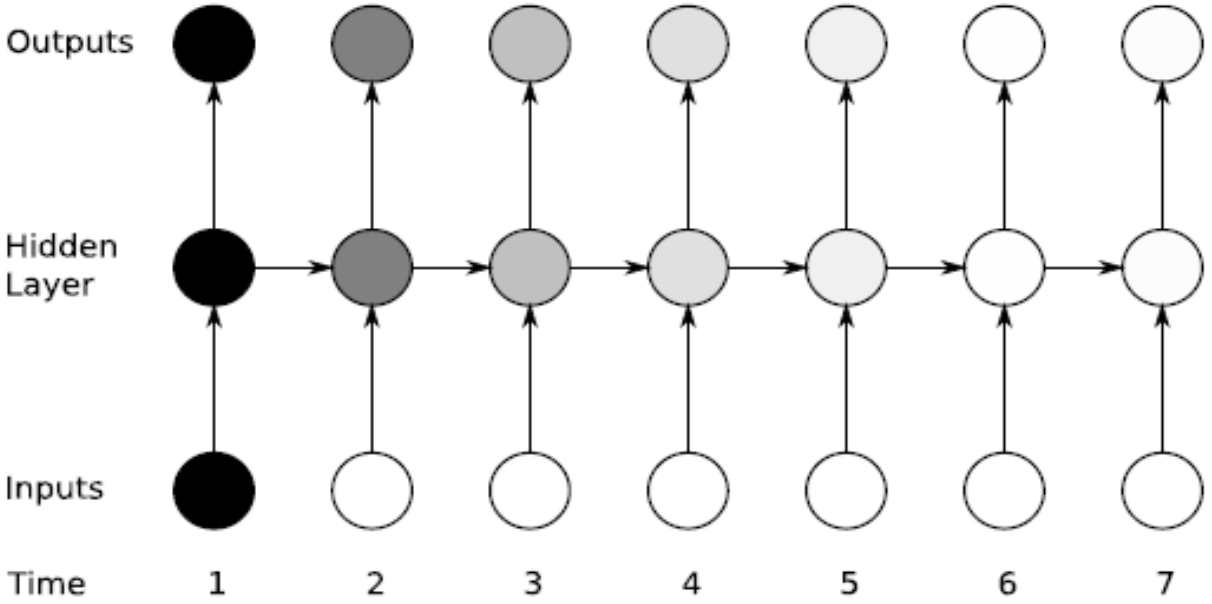
- در شبکه جردن
  - اتصالات بازگشتی به جای لایه پنهان از لایه خروجی شروع می‌شود (با وزن ثابت یک)
  - در این شبکه لایه بافت، لایه حالت (State Layer) نامیده می‌شود
  - لایه حالت شامل اتصالات بازگشتی از خودش به خودش با وزن ثابت می‌باشد
  - تعداد نرون‌های لایه حالت با تعداد نرون‌های لایه خروجی برابر است
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





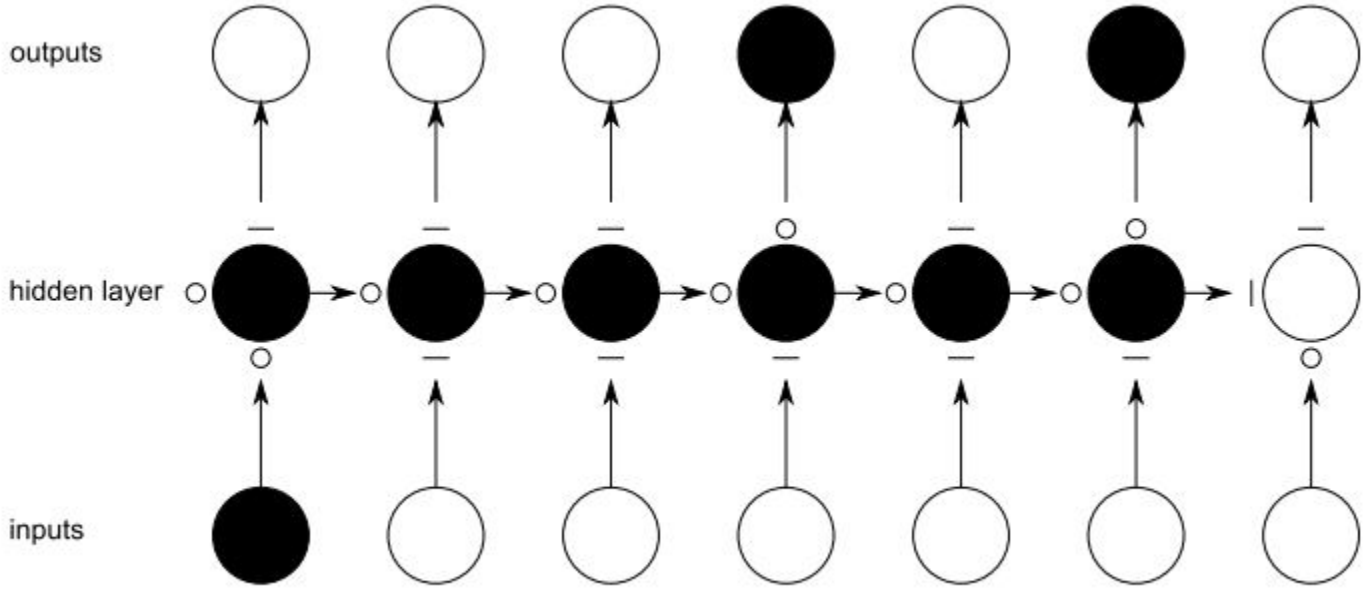
مشکل فراموشی در شبکه‌های عصبی بازگشتی . . .

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



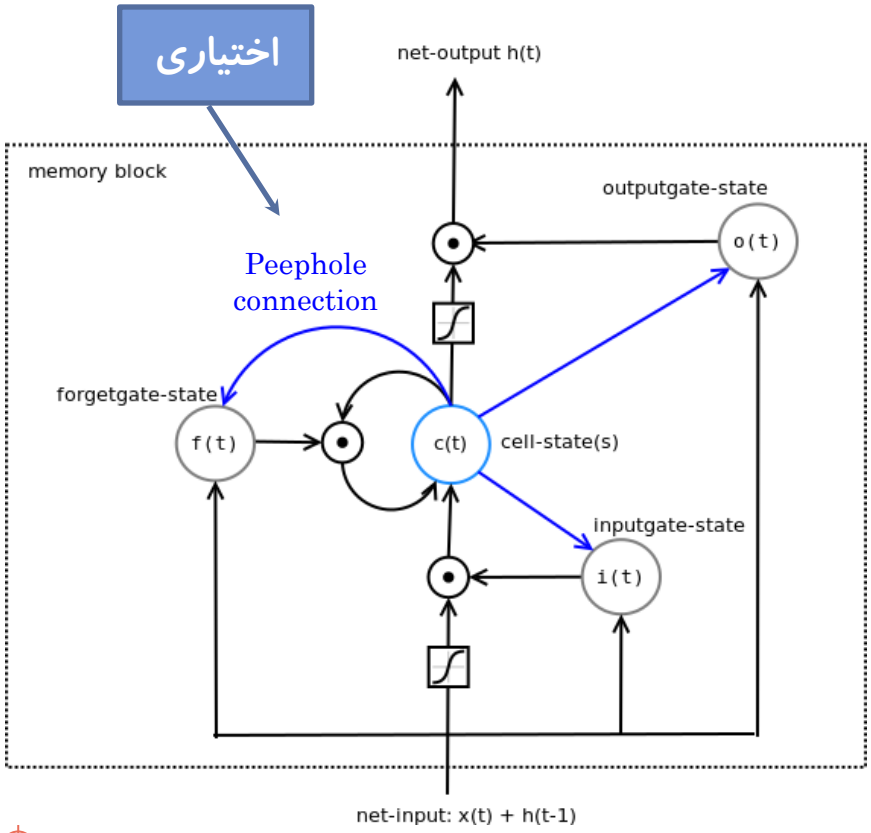
مشکل فراموشی در شبکه‌های عصبی بازگشتی

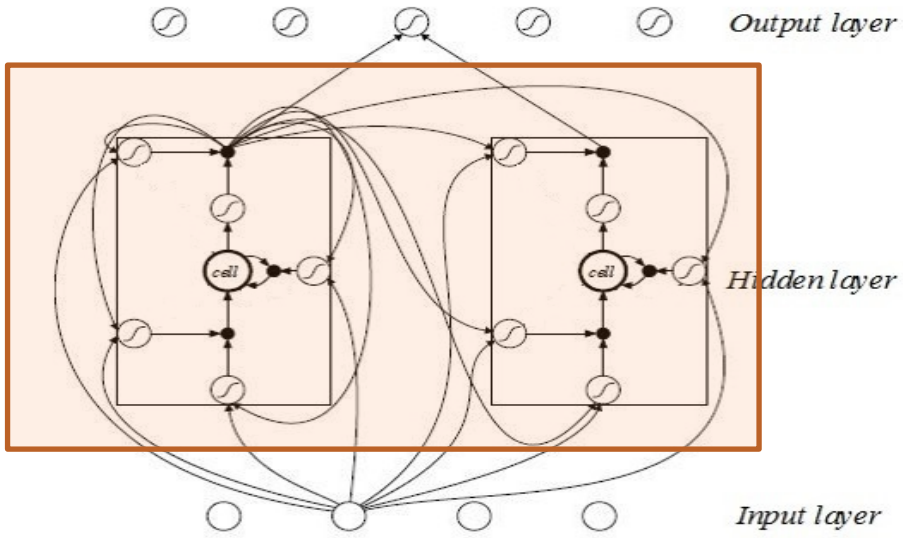
- حل مشکل فراموش
  - کنترل ورود و خروج داده در واحدهای لایه میانی شبکه



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

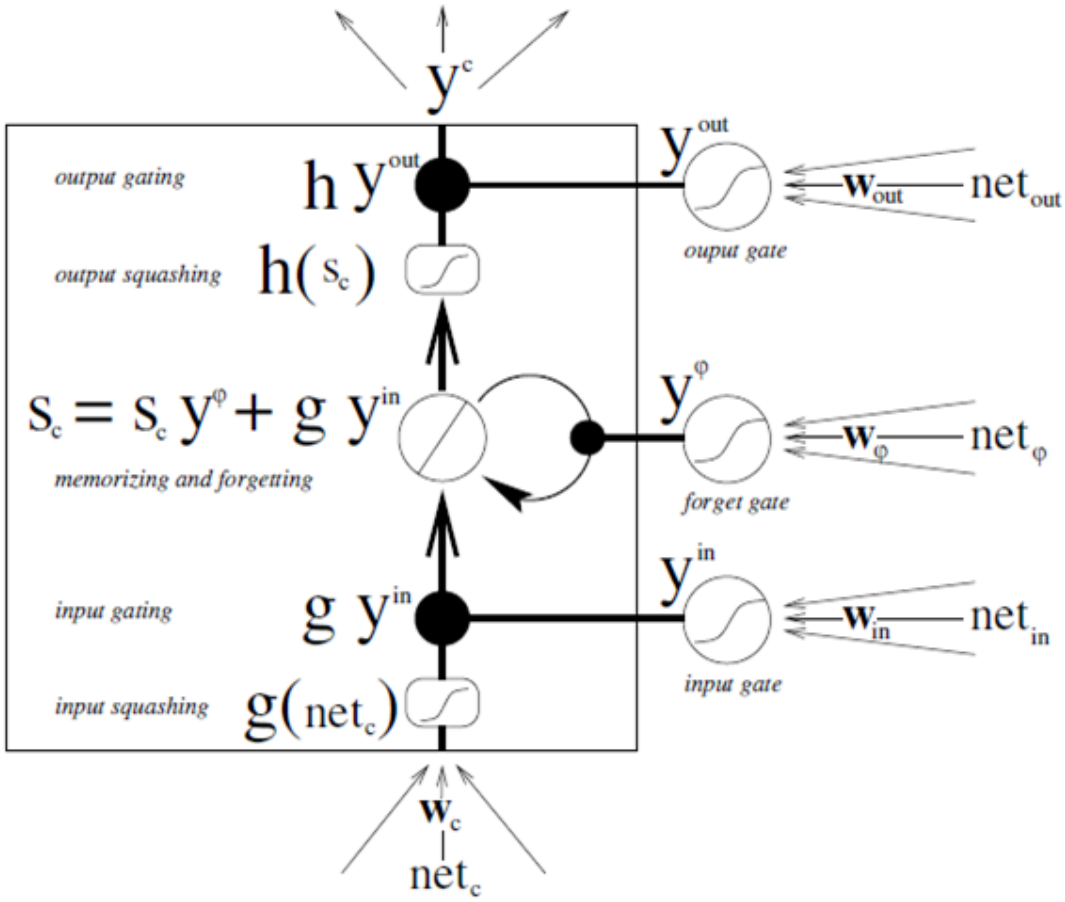
- شبکه Long Short Term Memory (LSTM)
  - نرون‌های لایه پنهان با بلوک‌های حافظه جایگزین شدند
    - حل شدن مشکل فراموشی دنباله‌های طولانی





شبکه عصبی LSTM: ساختار بلوک حافظه ...

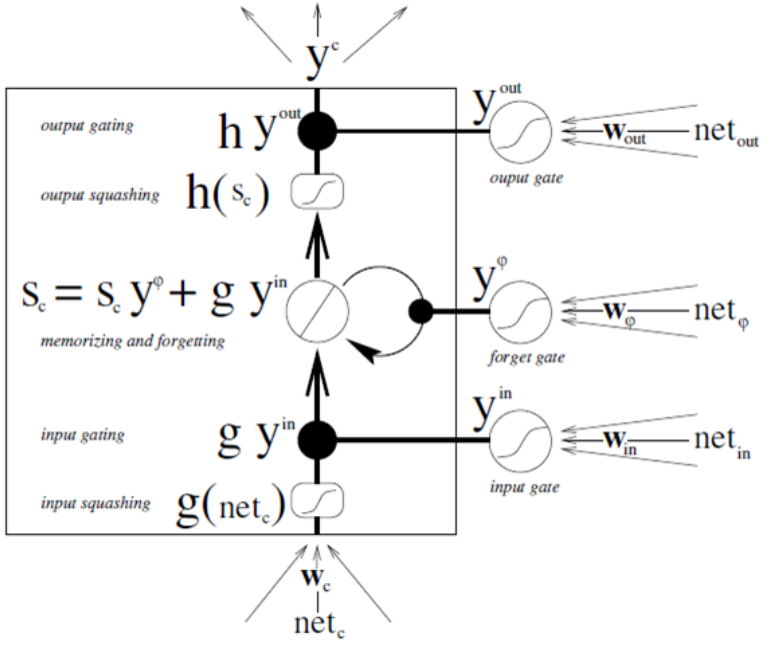
- هر بلوک حافظه، شامل
  - سلول
    - برای ذخیره اطلاعات در بلوک
  - دروازه ورودی
  - دروازه فراموشی
  - دروازه خروجی



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

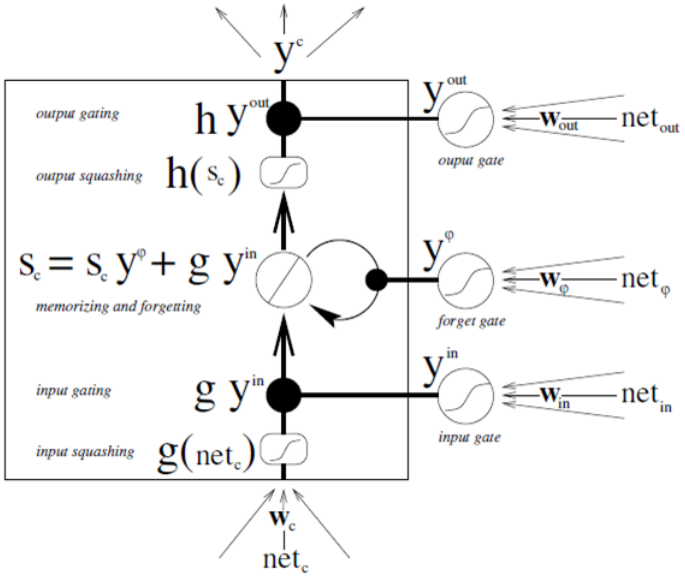
شبکه عصبی LSTM: ساختار بلوک حافظه . . .

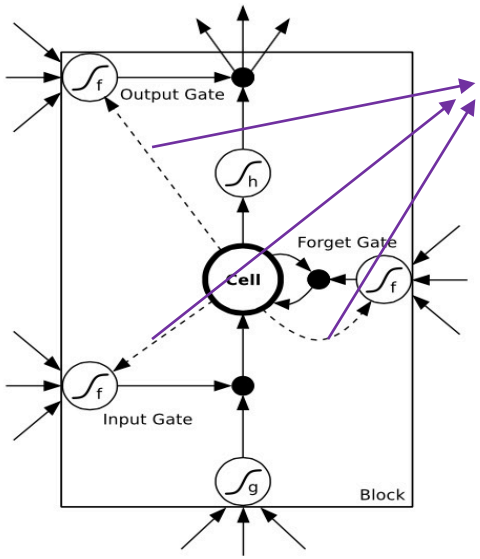
- وظیفه دروازه‌ها: کنترل ورود و خروج داده‌ها و پاک کردن حافظه‌ی بلوک
  - فعال‌ساز دروازه‌ها مقداری بین صفر و یک می‌گیرند.
  - در صورتی که دروازه کاملا باز باشد فعال‌ساز آن برابر یک و در صورتی‌که کاملا بسته باشد فعال‌ساز آن برابر صفر است
- هر سلول حافظه در مرکز خود یک واحد به نام CEC دارد که به فعال‌سازی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



شبکه عصبی LSTM: ساختار بلوک حافظه

- اتصالات روزنه (peephole)
  - این اتصالات دلخواه هستند (optional) و حالت سلول، S_c، را به دروازه‌ها متصل می‌سازند
  - در این اسلایدها از این اتصالات صرفنظر شده است

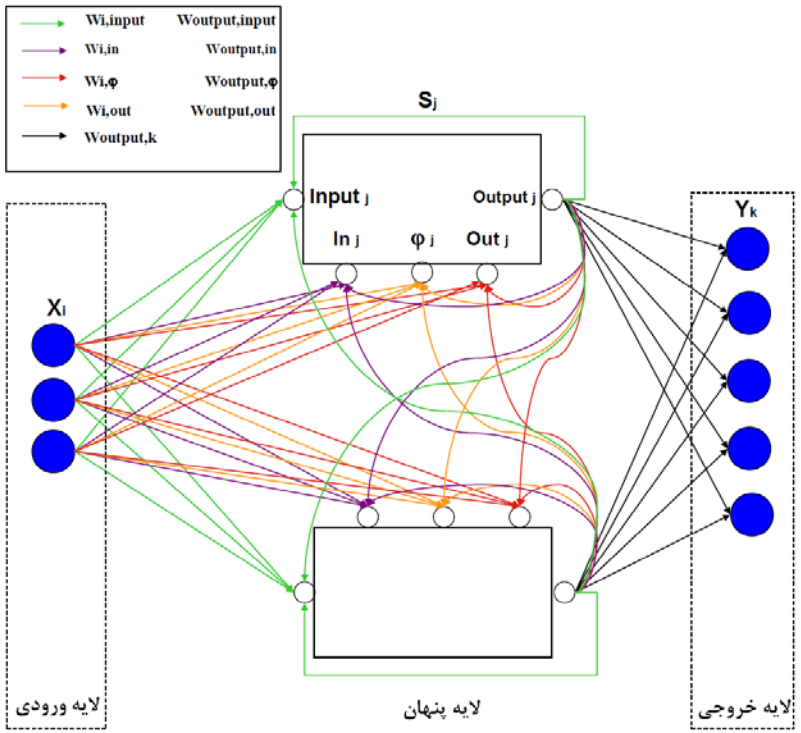




Peephole connections

شبکه عصبی LSTM: اتصالات (وزن ها)

- از ورودی
  - وزن بین لایه ورودی و بلوک حافظه
  - وزن بین لایه ورودی و دروازه ورودی
  - وزن بین لایه ورودی و دروازه فراموشی
  - وزن بین لایه ورودی و دروازه خروجی
- از خروجی بلوک (سلول حافظه)
  - وزن بین خروجی بلوک‌ها و ورودی بلوک‌ها
  - وزن بین خروجی بلوک‌ها و دروازه ورودی
  - وزن بین خروجی بلوک‌ها و دروازه فراموشی
  - وزن بین خروجی بلوک‌ها و دروازه خروجی
  - وزن بین خروجی‌های بلوک‌ها و لایه خروجی



شبکه عصبی LSTM: الگوریتم آموزش و کاربرد

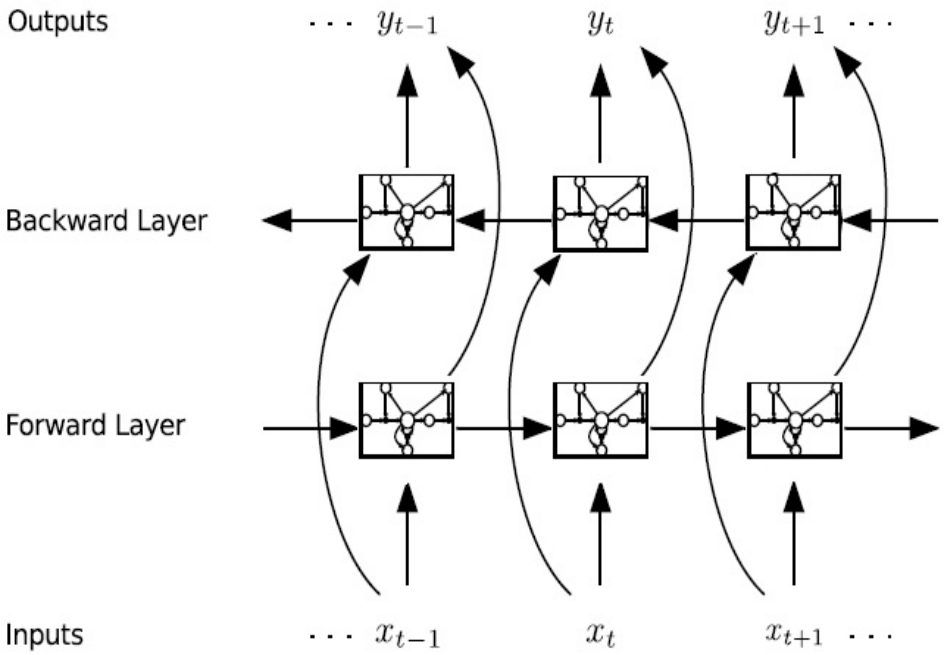
- مراجعه کنید به

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

شبکه‌های عصبی 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.



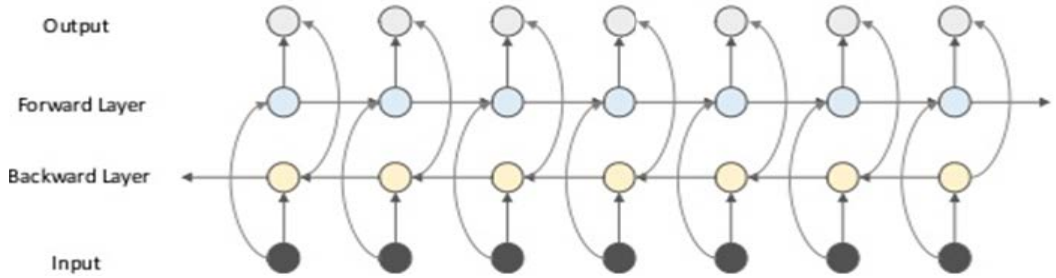
شبکه‌های عصبی LSTM دوطرفه (Bidirectional)

ایده اصلی

هر دنباله ورودی در دو جهت زمانی رو به جلو و از انتها به دو لایه پنهان بازگشتی مجزا داده شود

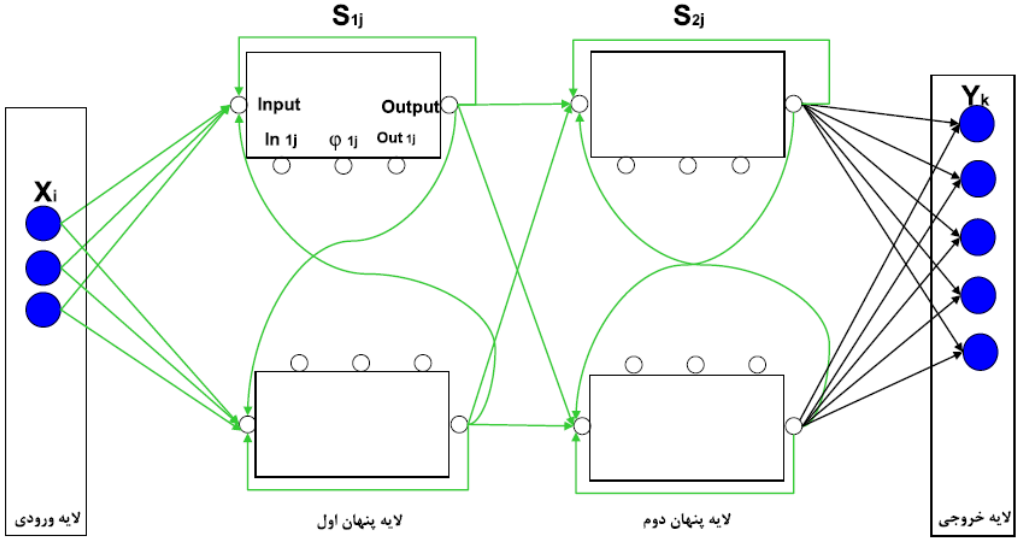
- فرض کنید دنباله آموزشی به صورت $X^T = (x_1, x_2, \dots, x_{T-1}, x_T)$ و دنباله هدف متناظر برابر $t^T = (t_1, t_2, \dots, t_{T-1}, t_T)$ باشد
- در هر مرحله بردار x_i را به لایه Forward و $x_{T-(i-1)}$ را به لایه Backward ارسال کرده و مقدار هد ف را بردار t_i قرار می‌دهیم.
- آموزش شبکه را با استفاده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

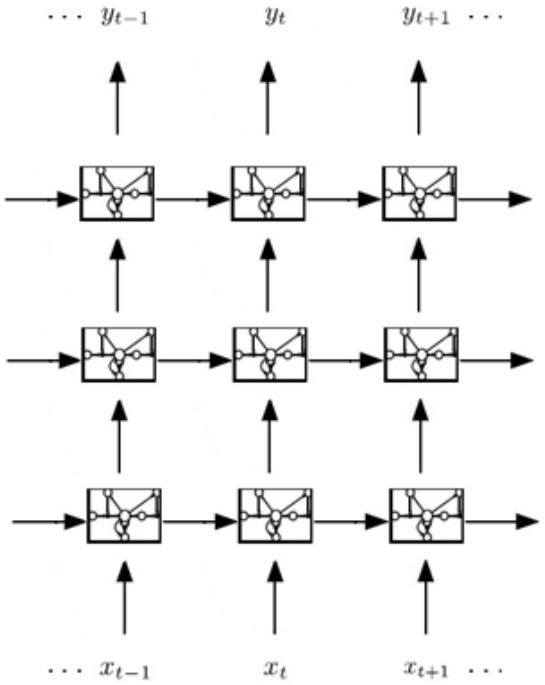
مقدار خالص رسیده به لایه خروجی جمع وزن‌دار مقدار خالص دو لایه Forward وBackward است



شبکه‌های عصبی LSTM: عمیق

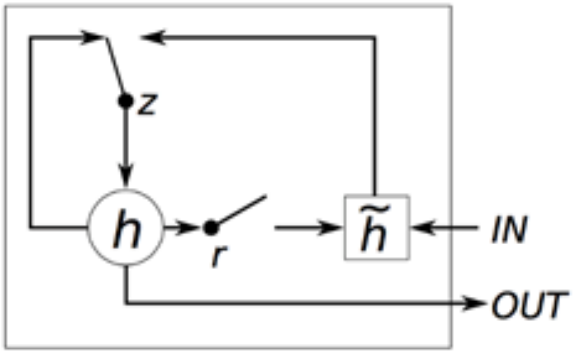
- شبکه عصبی با بیش از یک لایه مخفی
  - خروجی هر لایه پنهان ورودی لایه پنهان بالاتر
  - تقریب زننده جهانی
  - قابلیت یادگیری بیشتر

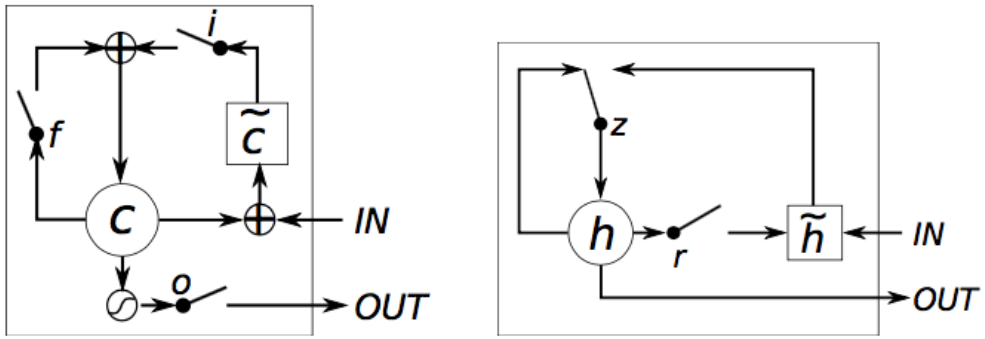




شبکه عصبی واحد بازگشتی دروازه‌ای(GRU)

- 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



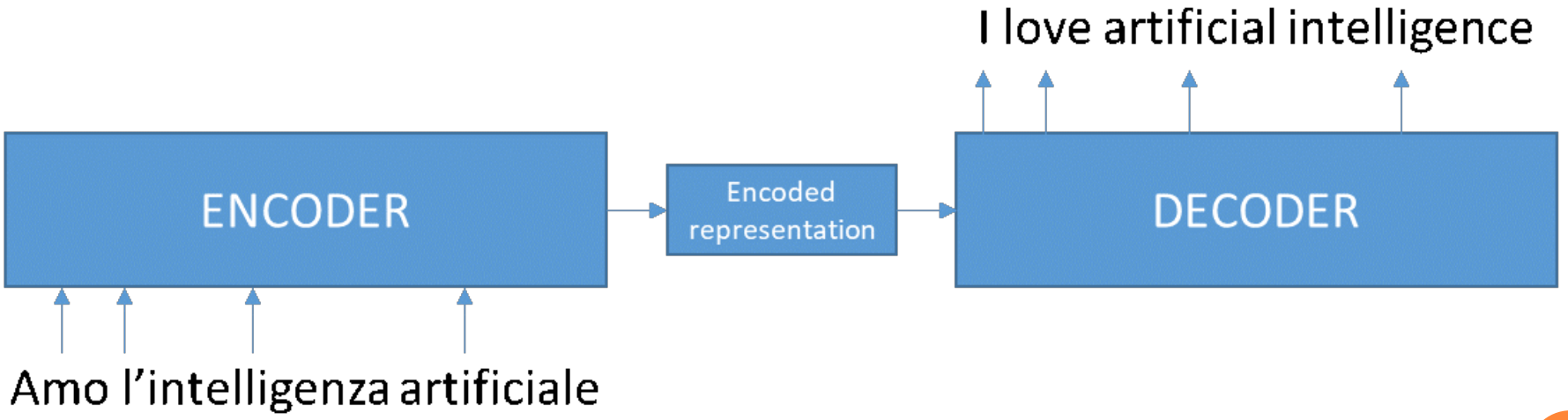


(a) Long Short-Term Memory (b) Gated Recurrent Unit

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.

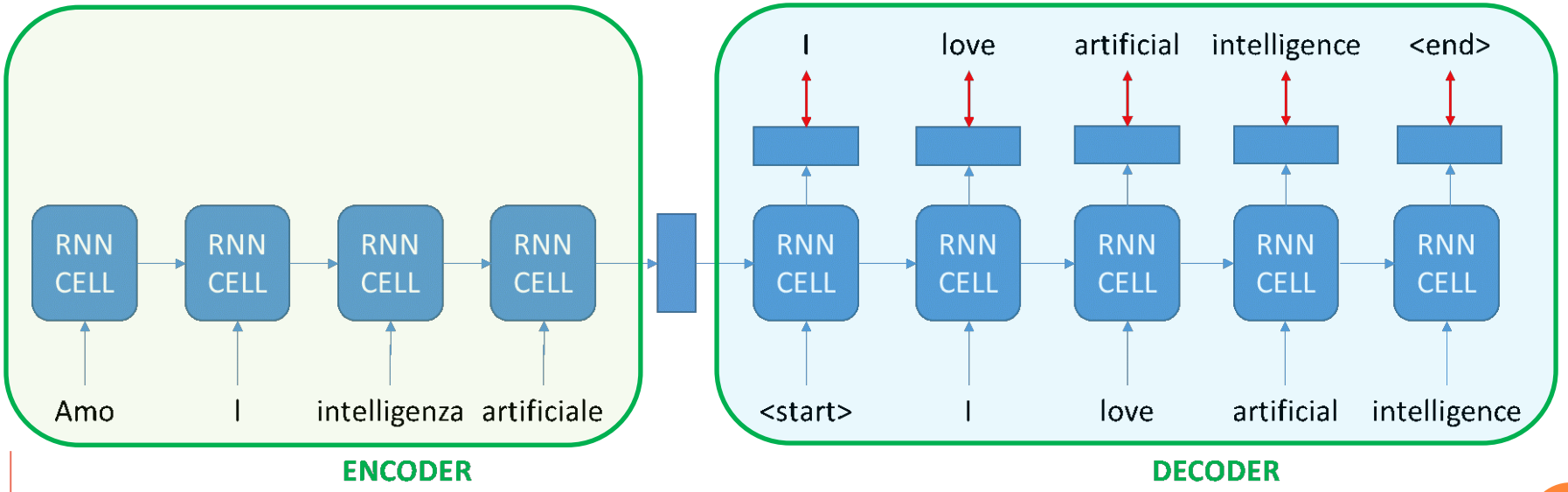
شبکه‌های بازگشتی: Seq2Seq . . .

- نگاشت کردن دو دنباله به همدیگر
  - ترجمه ماشینی
  - چت بات
  - خلاصه سازی متن
  - زیرنویس ویدئو
  - ...



شبکه‌های بازگشتی: Seq2Seq

- نحوه کار کردن
  - رمزگذار (Encoder): کد کردن دنباله ورودی در یک بردار بافت (Context)
  - رمزگشا (Decoder): پیش بینی کلمه بعدی با دریافت بردار بافت و کلمه قبلی



شبکه‌های بازگشتی: سازوکار توجه . . .

- توجه (Attention) . . .
  - توجه بیشتر انسان به اطلاعات مهم در پردازش داده (تصویر، متن و ...)

  - مثال: عنوان گذاری تصویر

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

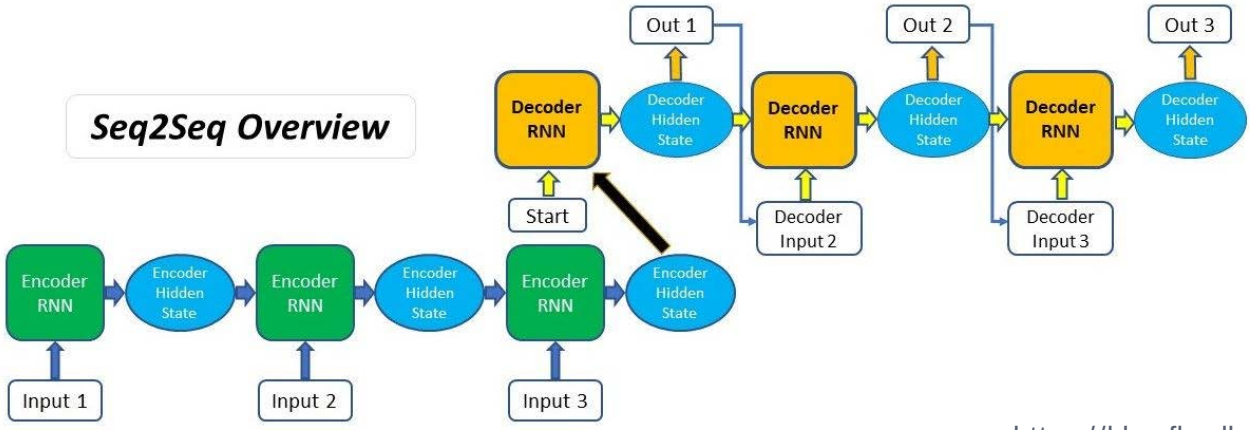
  - مثال: تحلیل احساس

شبکه‌های بازگشتی: سازوکار توجه . . .

- توجه (Attention)
  - نواحی توجه در مساله عنوان گذاری تصویر
    - جمله *A woman is throwing a frisbee in a park*

شبکه‌های بازگشتی: سازوکار توجه . . .

- مشکل
  - عدم امکان مدلسازی دقیق دنباله طولانی با یک بردار بافت به عنوان ورودی رمزگشا
- راه حل
  - در نظر گرفته خروجی همه حالت‌های میانی رمزگذار در فرایند رمزگشایی
    - برای تولید هر خروجی توسط رمزگشا، از همه حالت‌های میانی رمزگذار استفاده می شود
      - وزن دادن (توجه) بیشتر به حالت‌های میانی مرتبط با خروجی جاری



https://blog.floydhub.com/attention-mechanism

شبکه‌های بازگشتی: سازوکار توجه . . .

سازوکار توجه . . .

- فرض: ترجمه متن n کلمه‌ای x در زبان مبدا به متن m کلمه‌ای y در زبان مقصد

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$
$$\mathbf{y} = [y_1, y_2, \ldots, y_m]$$

- رمزگذار: یک BiLSTM با حالت‌های مخفی جلورو $\overrightarrow{h}_i$ و عقب‌رو $\overleftarrow{h}_i$
  - حالت کلی رمزگذار: اتصال دو دو حالت جلورو و عقب رو

$$h_i = [\overrightarrow{h}_i^\top ; \overleftarrow{h}_i^\top]^\top, i = 1, \ldots, n$$

- حالت مخفی رمزگشا برای کلمه خروجی t=1,…,m

$$s_t = f(s_{t-1}, y_{t-1}, \mathbf{c}_t)$$

  - تابعی از حالت قبلی رمزگشا، کلمه خروجی قبلی و بردار بافت $\mathbf{c}_t$
    - بردار بافت: جمع وزن‌دار حالت‌های رمزگذار برای همه کلمات ورودی

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$$

$$\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\exp(\text{score}(s_{t-1}, h_i))}{\sum_{i'=1}^{n} \exp(\text{score}(s_{t-1}, h_{i'}))}$$

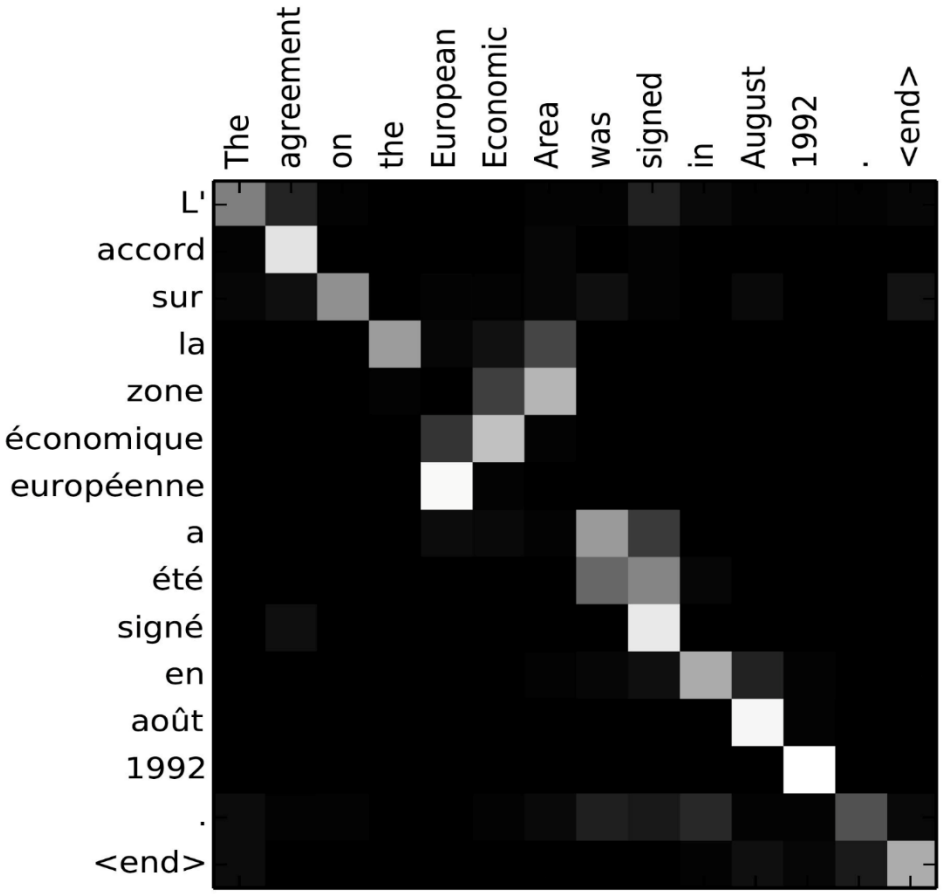
وزن‌های قابل آموزش با فرایند یادگیری

$$\text{score}(s_t, h_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a [s_t; h_i])$$

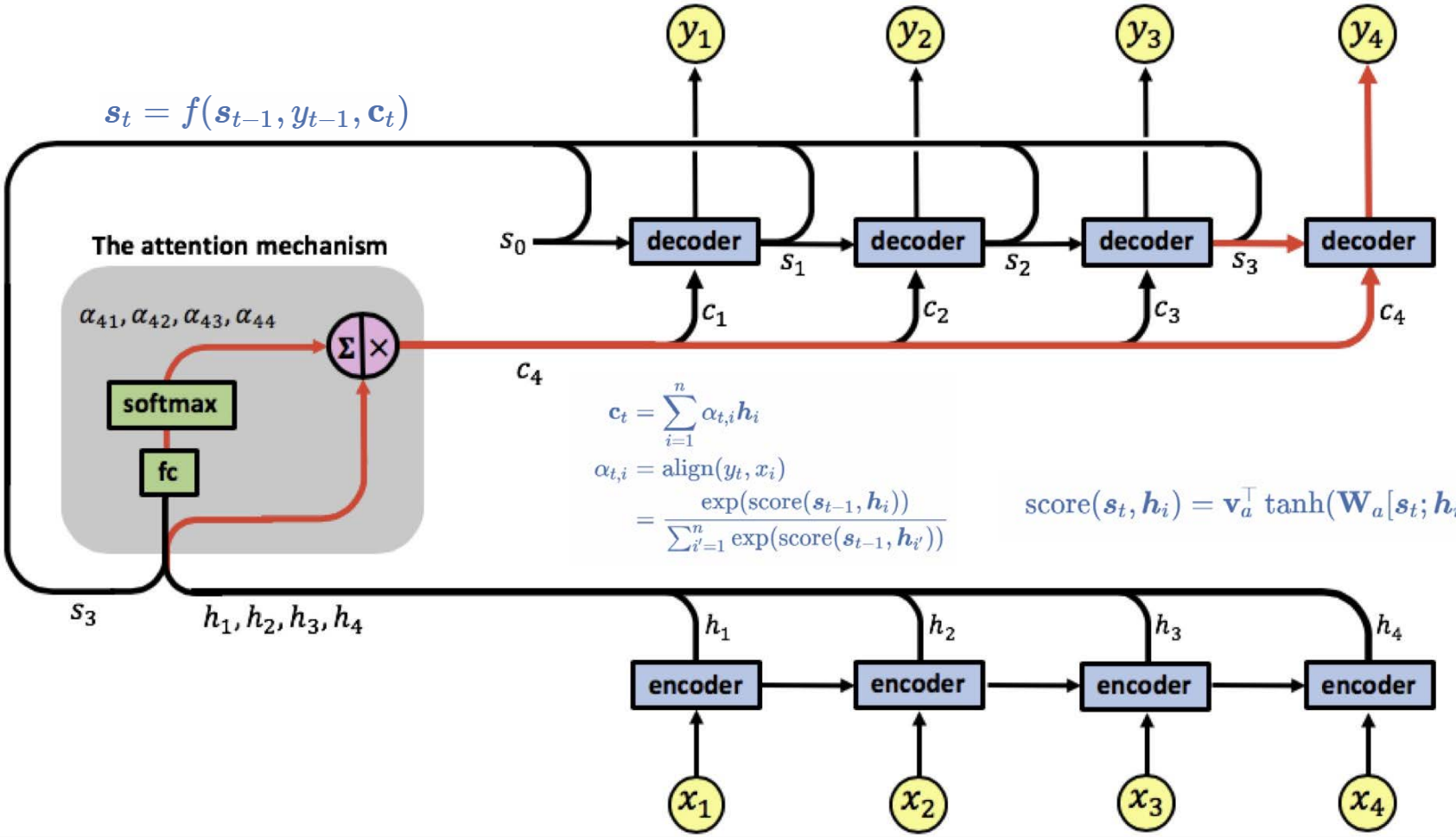
- وزن‌های $\alpha_{t,i}$= میزان توجه (تراز بودن) کلمه ورودی i با کلمه خروجی t

شبکه‌های بازگشتی: سازوکار توجه . . .

- سازوکار توجه . . .



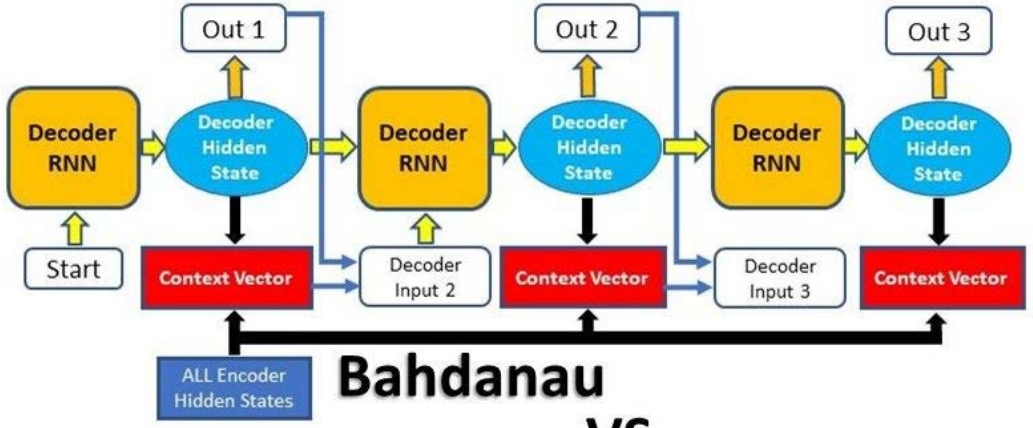
شبکه‌های بازگشتی: سازوکار توجه . . .



شبکه‌های بازگشتی: سازوکار توجه . . .

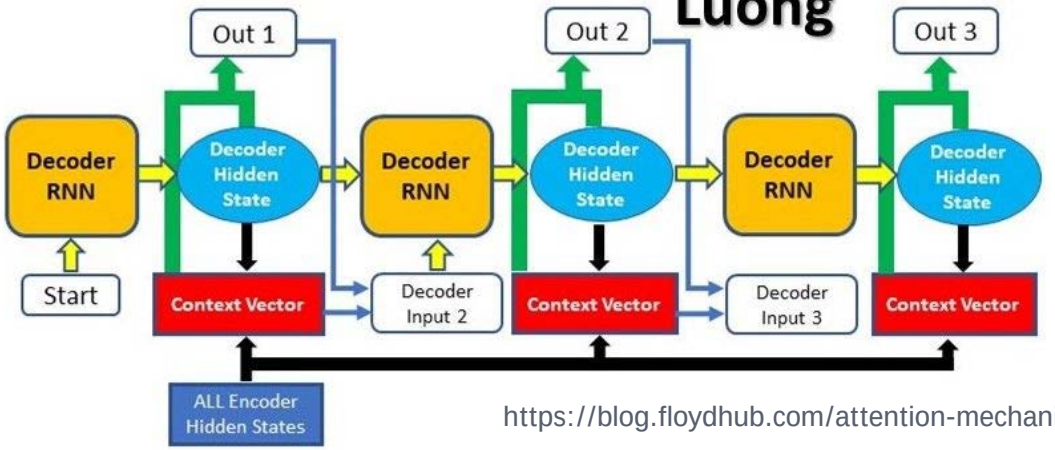
- انواع توجه

Additive Attention



Bahdanau VS Luong

Multiplicative Attention



https://blog.floydhub.com/attention-mechanism

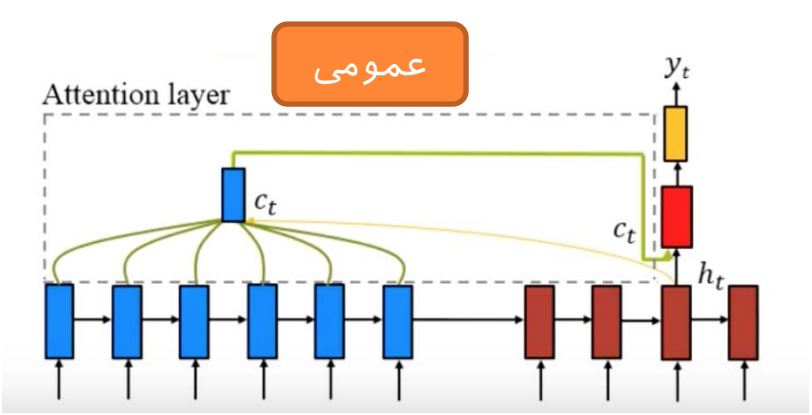
شبکه‌های بازگشتی: سازوکار توجه . . .

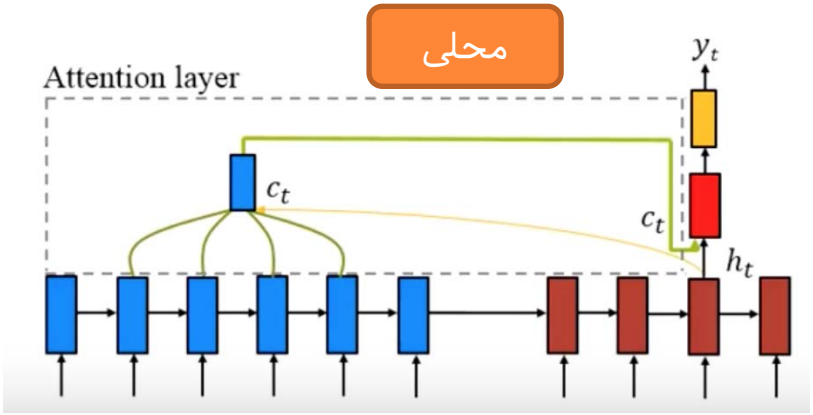
- روش‌های مختلف محاسبه امتیاز توجه

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

شبکه‌های بازگشتی: سازوکار توجه . . .

- **انواع توجه . . .**
  - **نرم (Soft) و سخت (Hard)**
    - نرم: یادگیری وزن‌ها و محاسبه آنها با در نظر گرفتن کل داده (کل تصویر یا کل جمله)
    - سخت: در نظر گرفتن تنها بخشی از داده در هر مرحله
  - **محلی (Local) و عمومی (Global)**
    - عمومی = توجه نرم = در نظر گرفتن کل داده در محاسبات وزن‌ها
    - محلی: پیش بینی فقط یک خروجی برای تراز کردن با ورودی جاری و سپس در نظر گرفتن یک پنجره اطراف این ورودی برای محاسبه بردار بافت





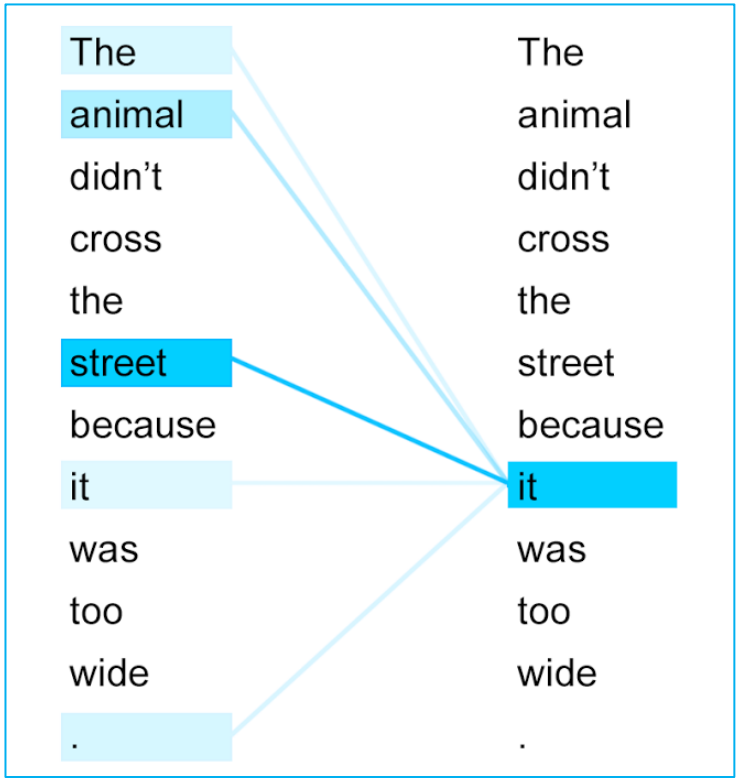
شبکه‌های بازگشتی: سازوکار توجه . . .

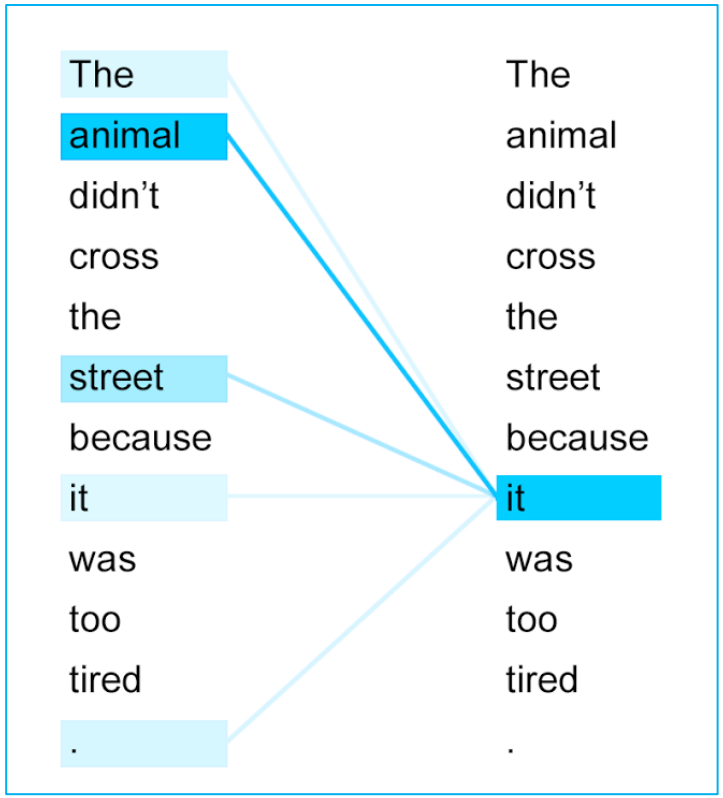
- انواع توجه . . .
  - توجه به خود (Self-Attention)
    - بیانگر میزان توجه مابین واحدهای مختلف یک دنباله (مانند کلمات یک جمله)
      - کاربرد در کاربردهای پردازش زبان مانند خلاصه سازی متن

The	FBI	is	chasing	a	criminal	on	the	run .
The	FBI	is	chasing	a	criminal	on	the	run .
The	FBI	is	chasing	a	criminal	on	the	run .
The	FBI	is	chasing	a	criminal	on	the	run .
The	FBI	is	chasing	a	criminal	on	the	run .
The	FBI	is	chasing	a	criminal	on	the	run .
The	FBI	is	chasing	a	criminal	on	the	run .
The	FBI	is	chasing	a	criminal	on	the	run .
The	FBI	is	chasing	a	criminal	on	the	run .

شبکه‌های بازگشتی: سازوکار توجه . . .

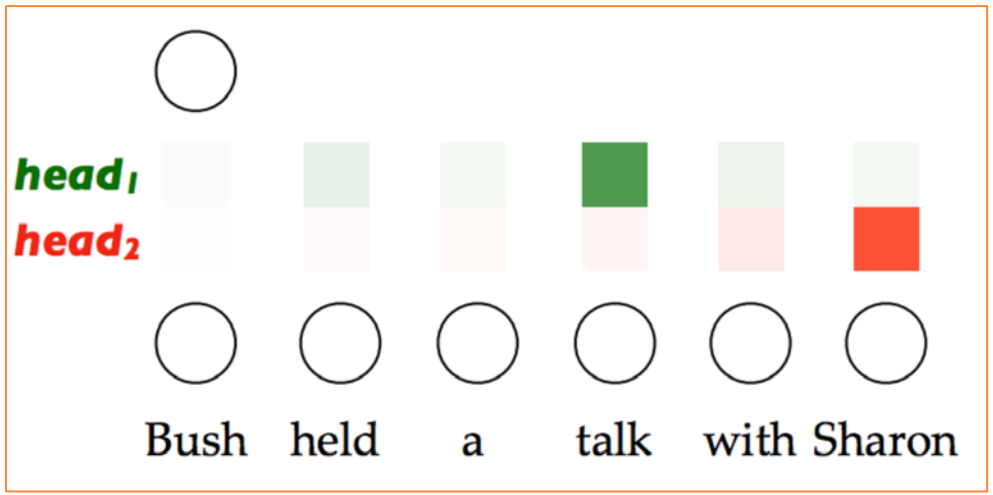
- انواع توجه . . .
  - توجه به خود (Self-Attention)
    - یاد گرفتن اینکه it معادل کدام کلمه است (مرجع ضمیر)

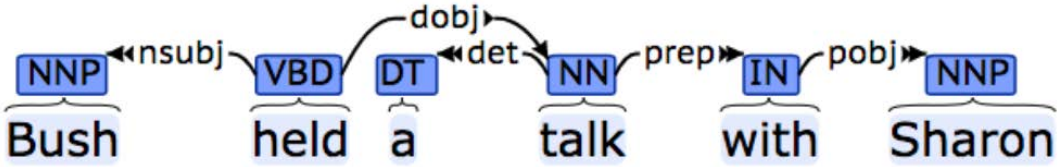




شبکه‌های بازگشتی: سازوکار توجه . . .

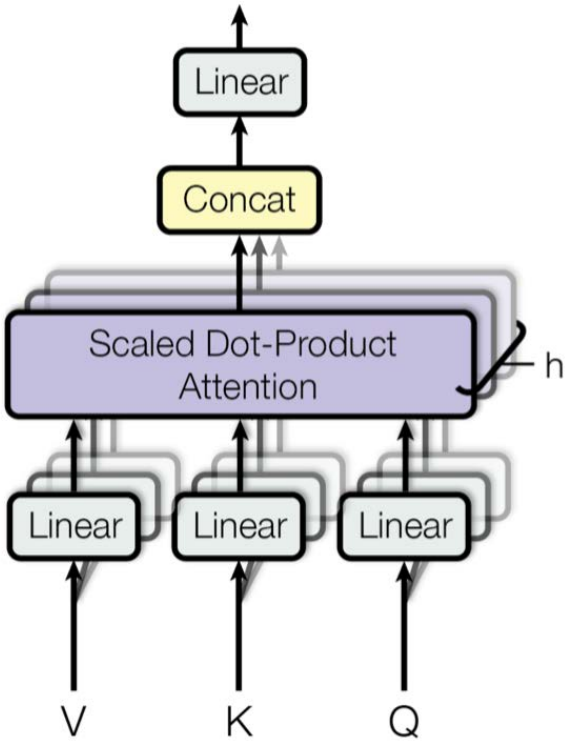
- انواع توجه . . .
  - توجه چند هسته‌ای (Multi-Head Attention)
    - حالت توسعه یافته Self-Attention که در آن چند لایه توجه نرم به صورت موازی اجرا شده و نتایج آنها با هم ادغام می‌شود





شبکه‌های بازگشتی: سازوکار توجه . . .

- انواع توجه . . .
  - توجه چند هسته‌ای (Multi-Head Attention)
    - اجرای چند Self-Attention به صورت موازی: وزن‌ها (Subspace)ی متفاوت
    - الحاق آنها به هم
    - ادغام با یک لایه



مثلا h=8

scaled dot-product attention

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

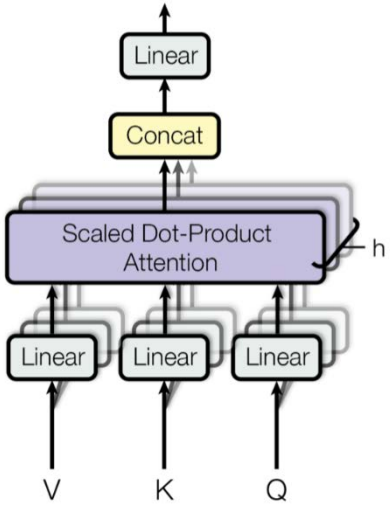
ابعاد Key

- ایده: ensembling

لایه وزنی خطی: تنظیم از طریق یادگیری

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\text{head}_1;\ldots;\text{head}_h]\mathbf{W}^O$$
$$\text{where head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

شبکه‌های بازگشتی: سازوکار توجه . . .



$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

انواع توجه . . .

- **توجه چند هسته‌ای(Multi-Head Attention): مثال**
  - برای عبارت Action gets results
  - محاسبه برای هر کلمه نسبت به سایر کلمات
  - برای کلمه اول (Action)
    - بردار Query = کلمه Action (در ترجمه خروجی قبلی رمزگشا)
    - بردارهای Keys: همه کلمات (در ترجمه بردارهای حالت مخفی رمزگذار)
    - بردارهای Value: همه کلمات (در ترجمه بردارهای حالت مخفی رمزگذار)

n= 64

میزان توجه

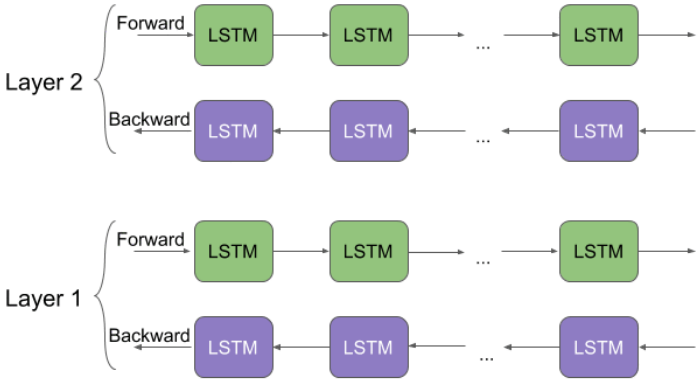
Word	q vector	k vector	v vector	score	score / 8	Softmax	Softmax * v	Sum
Action	q_1	k_1	v_1	$q_1 . k_1$	$q_1 . k_1 / 8$	x_{11}	$x_{11} * v_1$	z_1
gets		k_2	v_2	$q_1 . k_2$	$q_1 . k_2 / 8$	x_{12}	$x_{12} * v_2$	
results		k_3	v_3	$q_1 . k_3$	$q_1 . k_3 / 8$	x_{13}	$x_{13} * v_3$	

خروجی MHA

محاسبه به صورت مشابه برای بقیه کلمات

شبکه عصبی LSTM: کاربردها . . .

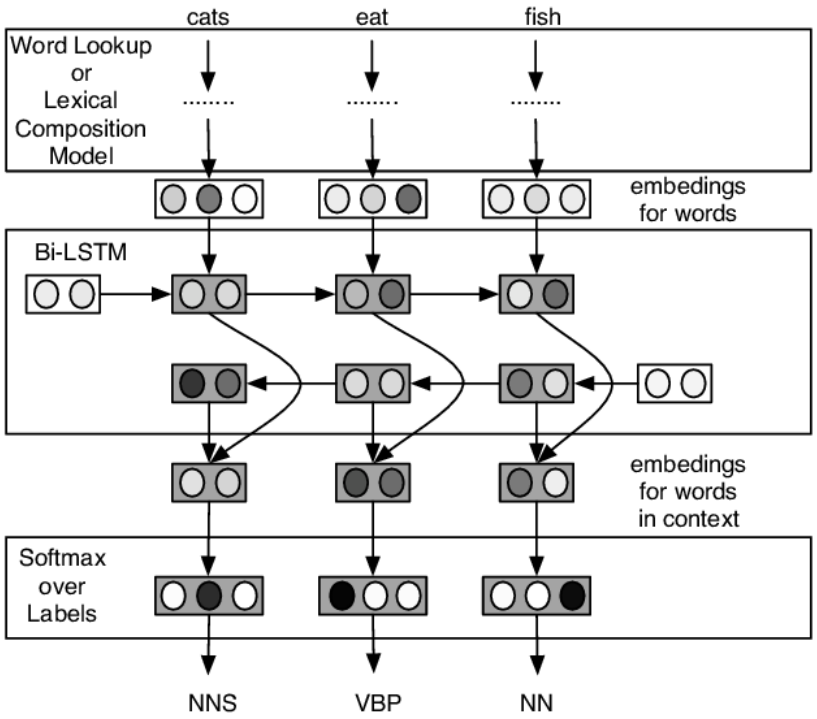
- مدل زبانی و بردار تعبیه کلمات . . .
  - استفاده از LSTM استاندارد
  - بردارهای تعبیه جدا برای کلمات Polysemy (ظاهر یکسان و معنی متفاوت)
    - شبکه Embeddings from Language Models (ELMo)

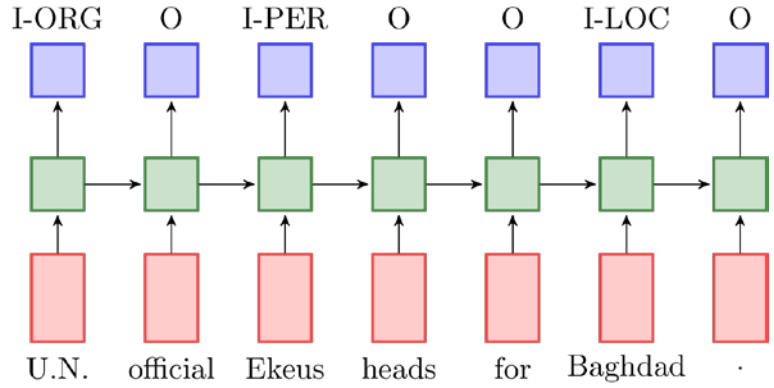


Peters, Matthew E., et al. "Deep contextualized word representations." *arXiv preprint arXiv:1802.05365* (2018).

شبکه عصبی LSTM: کاربردها . . .

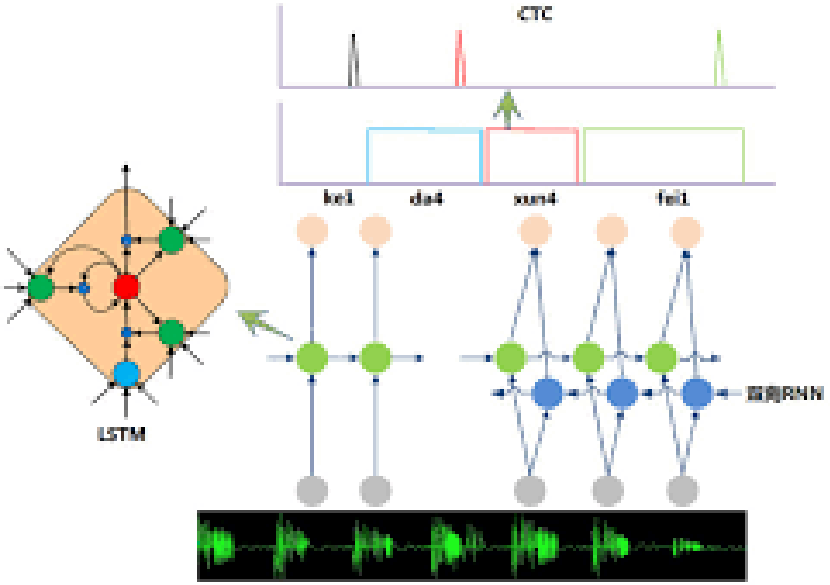
- برچسب زنی اجزای کلام (POS)/بازشناسی پدیده‌های اسمی (NER)
  - ورودی: بردار یک کلمه (مثل Word Vector)
  - خروجی: به تعداد برچسب‌ها (هر نرون یک برچسب)





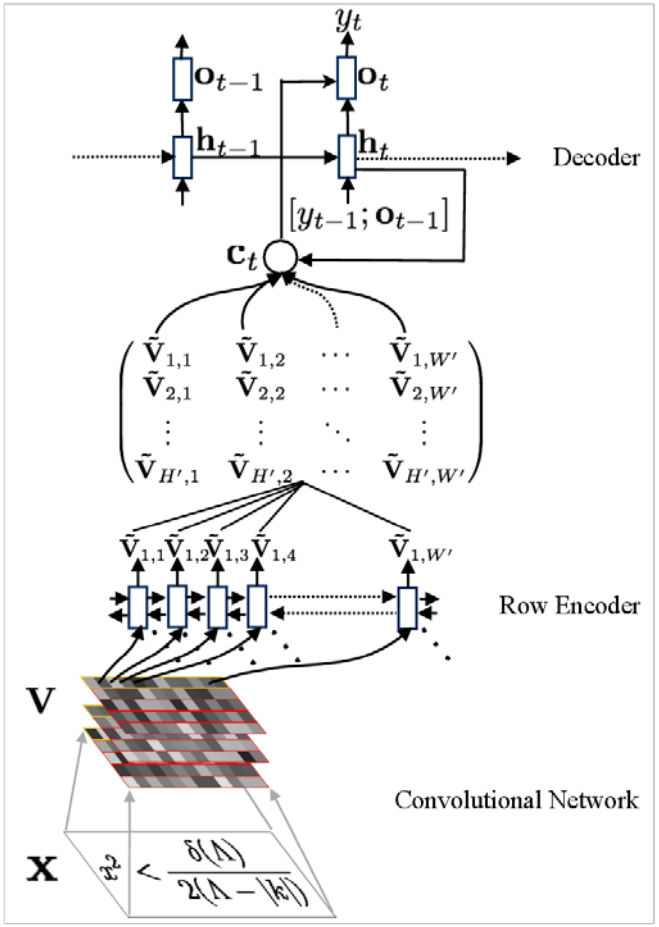
شبکه عصبی LSTM: کاربردها . . .

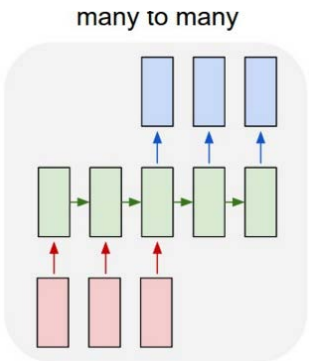
- بازشناسی گفتار
  - ورودی: بردار مربوط به یک فریم گفتار
  - خروجی: به تعداد واحدهای بازشناسی شونده (هر نرون یک واج)
    - نیاز به روشی برای تبدیل دنباله برچسب فریم ها به دنباله واج = CTC



شبکه عصبی LSTM: کاربردها . . .

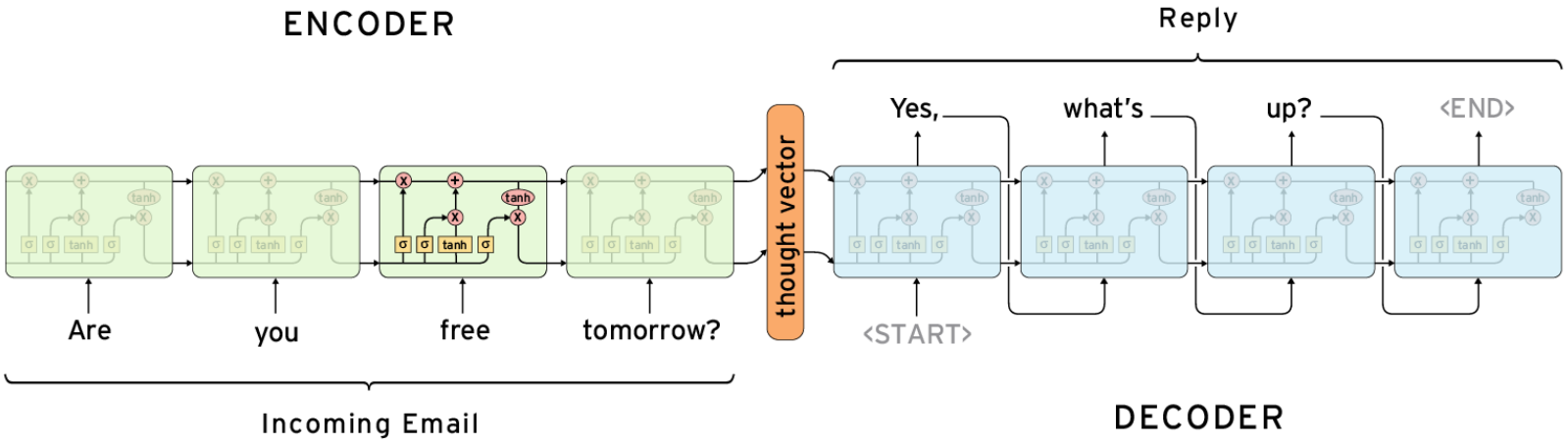
- بازشناسی دست خط/ نویسه‌های نوری (OCR)
  - ورودی: بردار مربوط به ویژگی یک فریم از تصویر
    - ویژگی های CNN
  - خروجی: به تعداد واحدهای بازشناسی شونده
    - هر نرون یک کاراکتر
    - نیاز به تبدیل دنباله برچسب فریم ها به دنباله واج = CTC

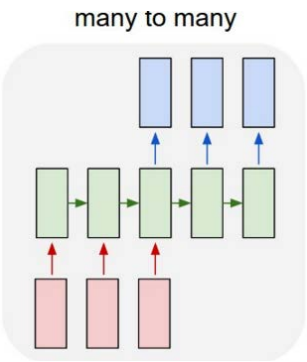




شبکه عصبی LSTM: کاربردها . . .

- چت بات
  - ورودی: دنباله کلمات ورودی (تبدیل هر کلمه به بردار با روشهای Word Vector)
  - خروجی: دنباله کلمات پاسخ

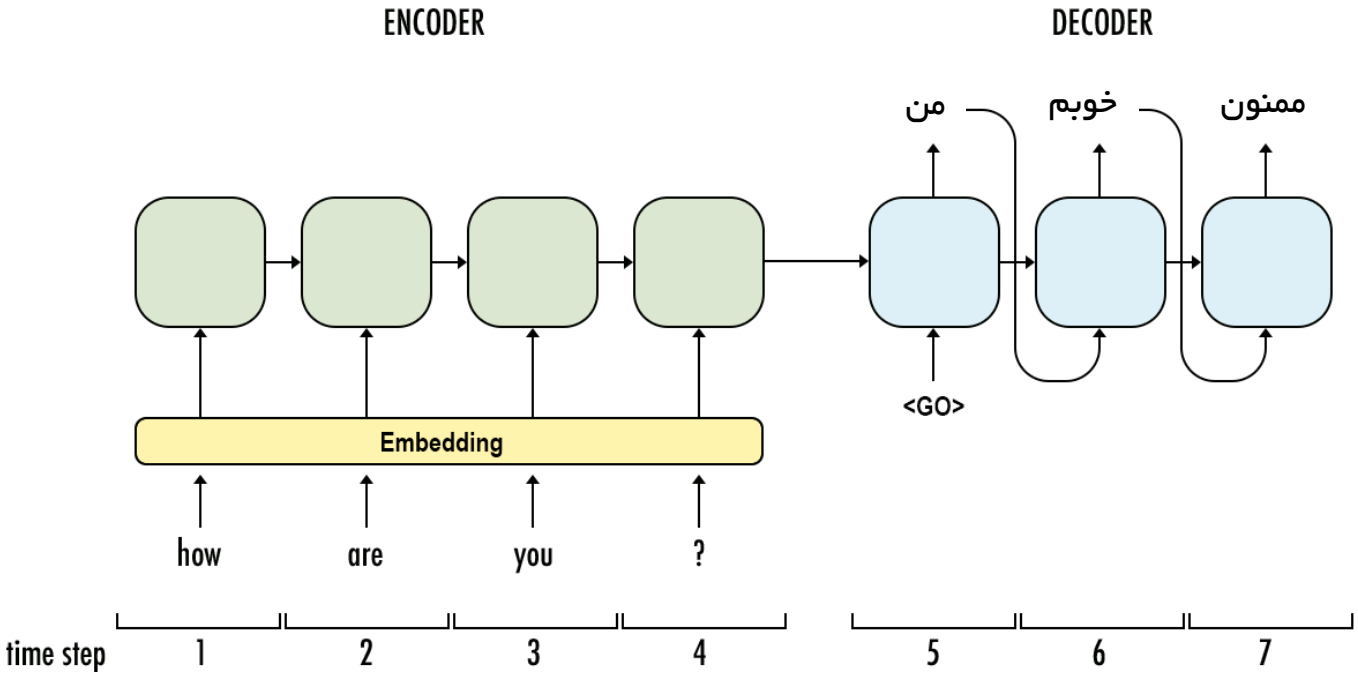




شبکه عصبی LSTM: کاربردها . . .

ترجمه ماشینی

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



شبکه عصبی LSTM: تشخیص واج‌های فارسی . . .

- **داده‌ها: فارس‌دات**
  - 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 $\left[\frac{-1}{10}, \frac{1}{10}\right]$
  - نرخ یادگیری: ۰/۰۰۰۳
  - تعداد بلوک حافظه: ۱۵۰

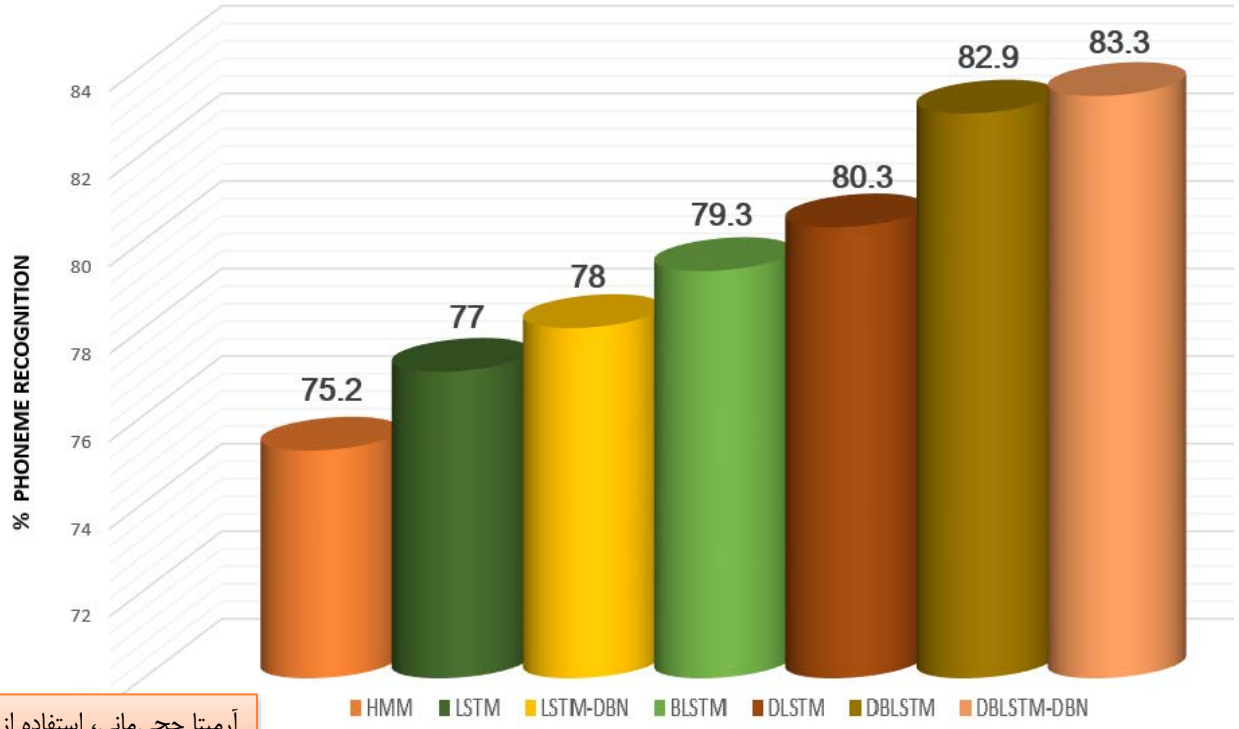
آرمیتا حجی‌مانی، استفاده از یادگیری عمیق برای بازشناسی گفتار فارسی، پایان‌نامه کارشناسی ارشد، دانشگاه تهران، ۱۳۹۵

شبکه عصبی LSTM دوطرفه: تشخیص واج‌های فارسی . . .

- **داده‌ها: فارس‌دات**
  - شامل ۶۰۸۰ سیگنال صوتی
  - داده آموزش: ۸۰٪ کل داده (معادل ۴۸۶۴ سیگنال)
  - داده آزمون: ۲۰٪ کل داده (معادل ۱۲۱۶ سیگنال)
- **استخراج ویژگی**
  - روش مورد استفاده: MFCC
  - تعداد ضرایب هر فریم: ۳۹
  - طول فریم: ۱۶ میلی ثانیه
- **شبکه مورد استفاده: Bidirectional LSTM**
- **ساختار شبکه**
  - نرون‌های لایه ورودی: ۳۹
  - نرون‌های لایه خروجی: ۳۰
  - وزن‌های اولیه: تصادفی در بازه $\left[\frac{-1}{10}, \frac{1}{10}\right]$
  - نرخ یادگیری: ۰/۰۰۰۳
  - تعداد بلوک حافظه: ۱۲۰

شبکه عصبی LSTM دوطرفه: تشخیص واج‌های فارسی

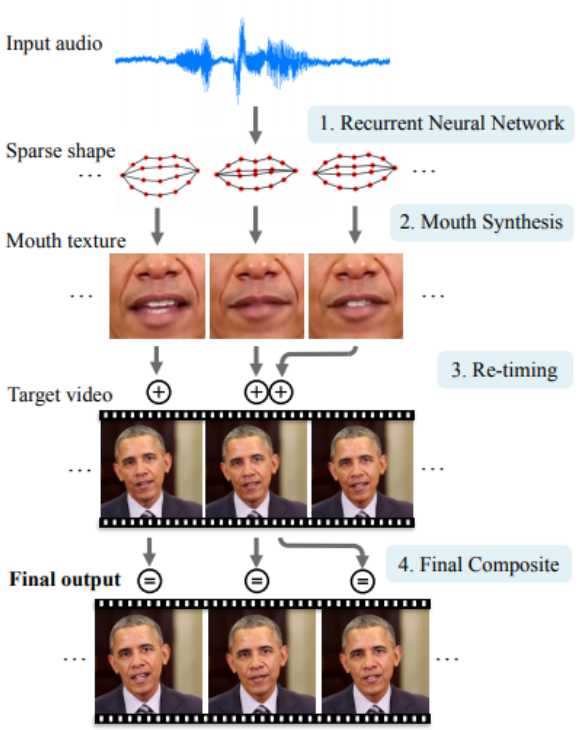
- دقت روی واج
  - کارایی بالاتر شبکه‌های دوطرفه نسبت به شبکه‌های یک‌طرفه
  - کارایی بالاتر شبکه‌های عمیق نسبت به شبکه‌های غیر عمیق



آرمیتا حجی‌مانی، استفاده از یادگیری عمیق برای بازشناسی گفتار فارسی، پایان‌نامه کارشناسی ارشد، دانشگاه تهران، ۱۳۹۵

شبکه عصبی LSTM: تولید سخنرانی ساختگی

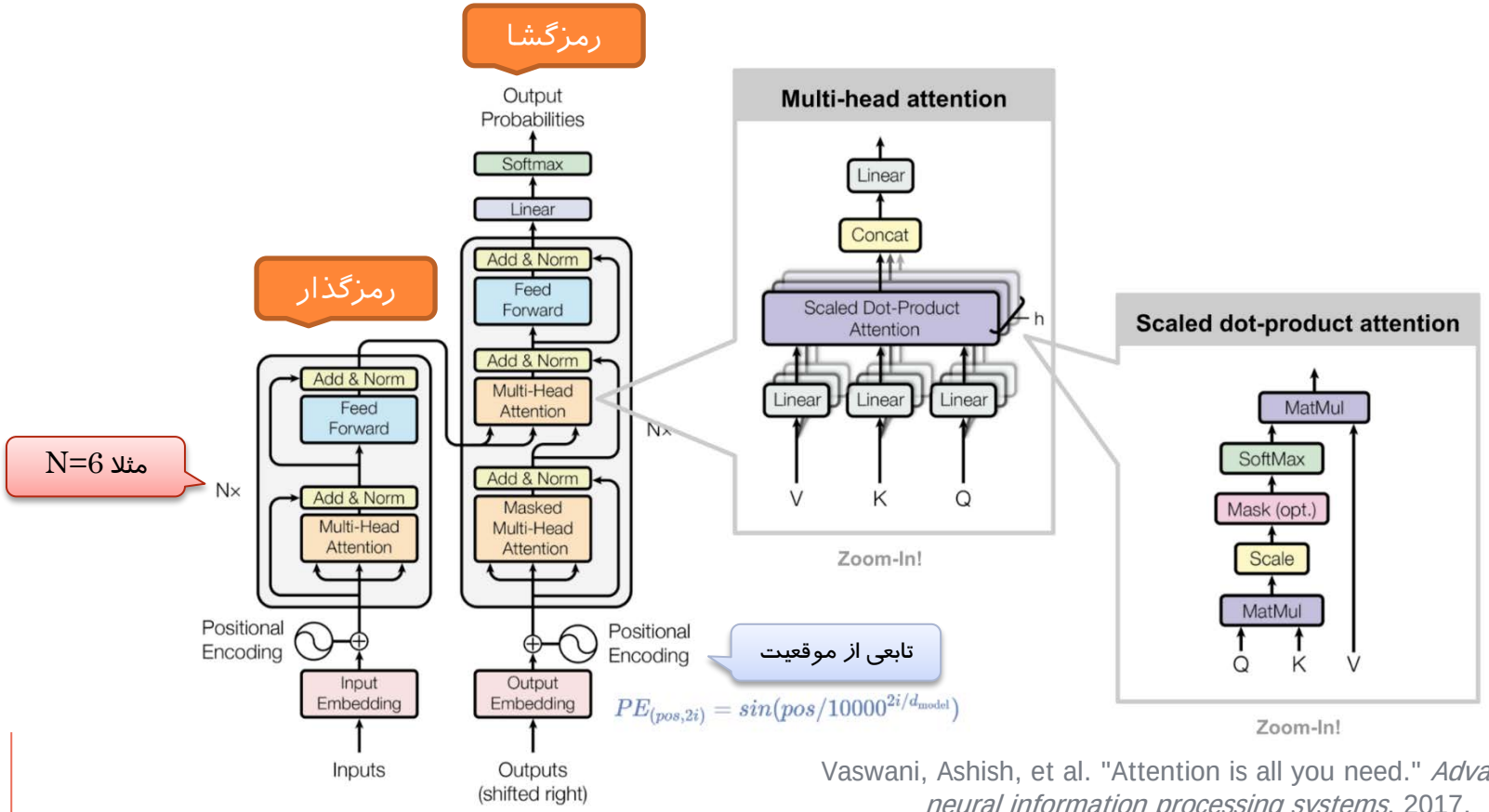
- تولید سخنرانی ساختگی برای اوباما: یادگیری حرکت لب با توجه به صدا



[Suwajanakorn, Supasorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. "Synthesizing obama: learning lip sync from audio.", 2017]

شبکه عصبی: مبدل‌ها ...

- مبدل (Transformer)
  - تبدیل ورودی به خروجی و مدل کردن وابستگی بین آنها با سازوکار توجه (بدون RNN)

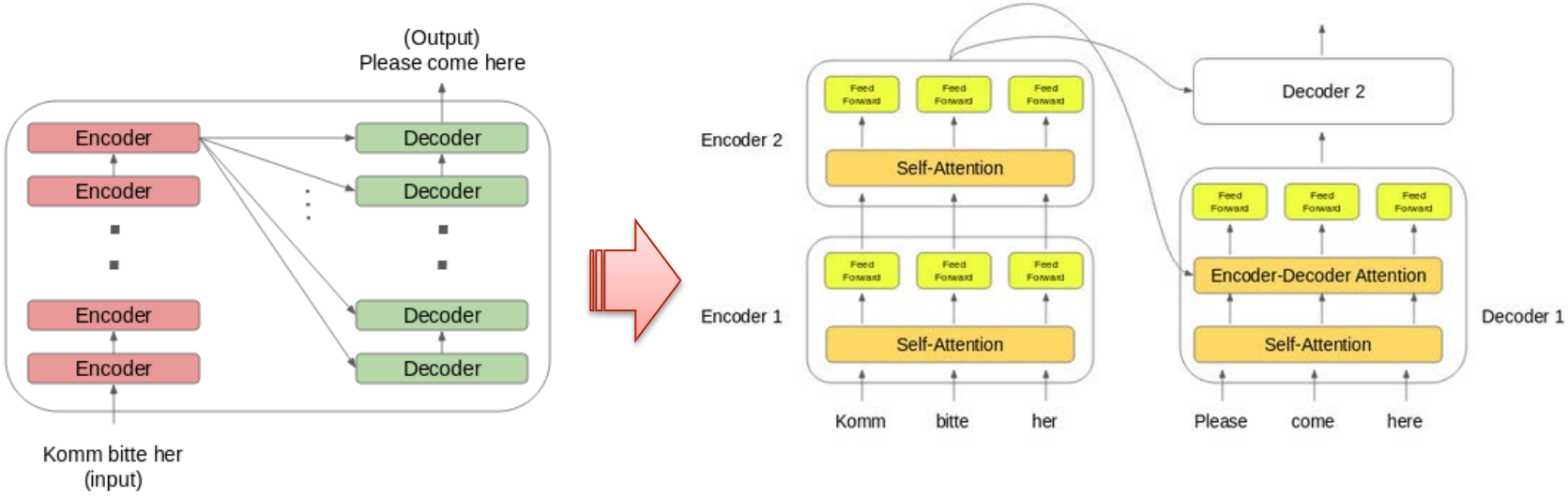


Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

شبکه عصبی: مبدل‌ها ...

ساختار

- تعداد N (مثلا ۶) رمزگذار و رمزگشا پشته شده
- دنباله (همه کلمات) ورودی به اولین رمزگذار داده می‌شود
- خروجی هر رمزگذار به رمزگذار بعدی داده می‌شود
- خروجی آخرین رمزگذار به همه رمزگشاها داده می‌شود



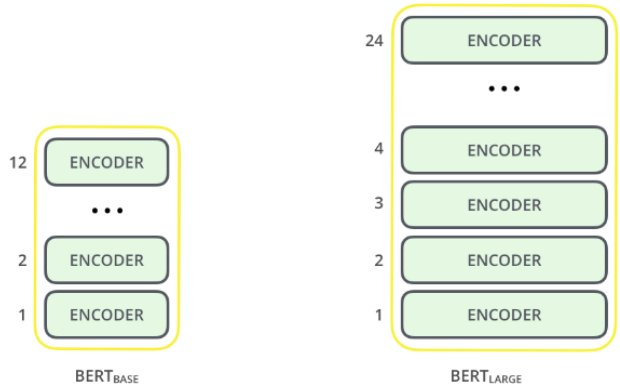
پیاده سازی در http://nlp.seas.harvard.edu/2018/04/03/attention.html

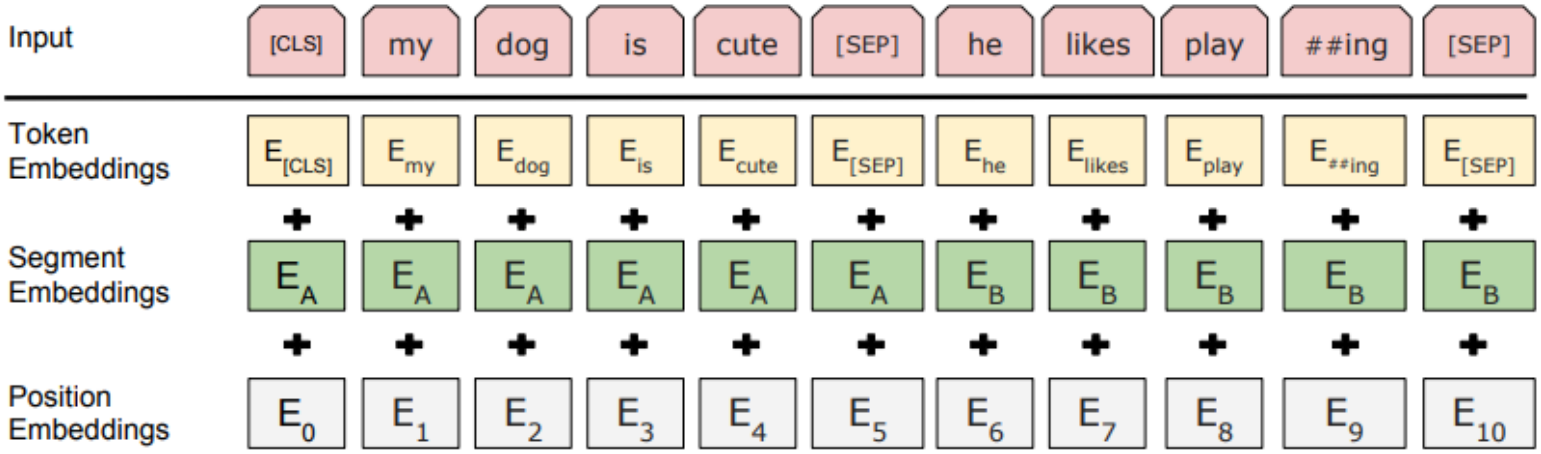
شبکه عصبی: مبدل‌ها ...

- بهبودها
  - رفع مشکل ثابت بودن طول دنباله ورودی (مشکل context fragmentation)
    - شبکه Transformer-XL: در نظر گرفتن بردار بافت قطعه قبلی به عنوان یکی دیگر از ورودی‌ها
  - مدل کردن ترتیب در دنباله به صورت مستقیم (مورد نیاز در یادگیری تقویتی)
    - Simple Neural Attention Meta-Learner (SNAIL)
  - شبکه Self-Attention GAN
    - مدل کردن وابستگی های خارج از محدوده فیلتر (وابستگی نواحی بزرگتر)

شبکه عصبی: کاربردهای مبدل‌ها . . .

- مدل زبانی و بردار تعبیه کلمات . . .
  - مدل Bidirectional Encoder Representations from Transformers (BERT)





شبکه عصبی: کاربردهای مدل‌ها . . .

- مزایای BERT در مقایسه با Word2Vec
  - در نظر گرفتن بافت
    - تولید بردارهای متفاوت برای کلمه «شیر» با معانی مختلف
  - در نظر گرفتن ترتیب
    - ترتیب آمدن کلمات در جمله به صورت مستقیم مدل می شود
  - نوع تعبیه مبتنی بر جمله است
    - برای در نظر گرفتن بافت لازم است کل جمله به مدل داده شود تا برای جمله و کلمات بردار تعبیه ایجاد شود (در نظر گرفتن بافت) اما Word2Vec برای هر کلمه یک بردار تولید می‌کند (روش BERT هم می‌تواند برای تولید بردار کلمات به تنهایی استفاده شود اما در این حالت بافت در نظر گرفته نمی‌شود)
  - بردار تعبیه برای کلمات خارج از واژگان (OOV)
    - امکان تولیدبردار برای هر کلمه با توجه به تولید بردار بر اساس زیرکلمات (نویسه‌ها)

شبکه عصبی: کاربردهای مبدل‌ها . . .

- پروژه Generative Pre-trained Transformer 3 (GPT-3)
  - مدل زبانی برای تولید متن طبیعی: تولید داستان، خلاصه سازی، شعر گفتن، کد نوشتن و ...
  - توسط OpenAI در ۲۰۲۰
  - دارای ۱۷۵ میلیارد پارامتر = بزرگترین شبکه عصبی ساخته شده تاکنون
    - هزینه آموزش شبکه = ۴.۶ میلیون دلار
    - آموزش روی حدود ۵۰۰ میلیارد واحد (کلمه) از متون مختلف (کتاب، اینترنت و ...)

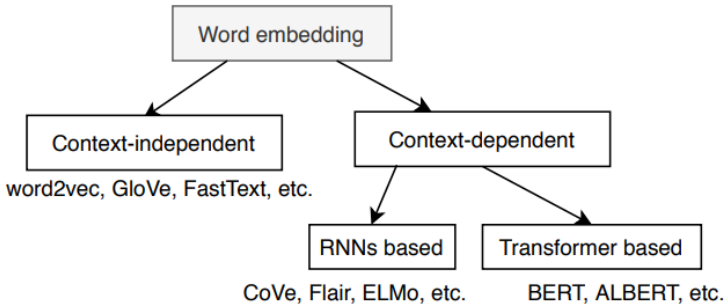
https://gpt3examples.com
https://gpt3demo.com

شبکه عصبی: کاربردهای مبدل‌ها . . .

- پروژه Generative Pre-trained Transformer 3 (GPT-3)

شبکه عصبی: کاربردهای مبدل‌ها . . .

مقایسه روش‌های مختلف تعبیه کلمات



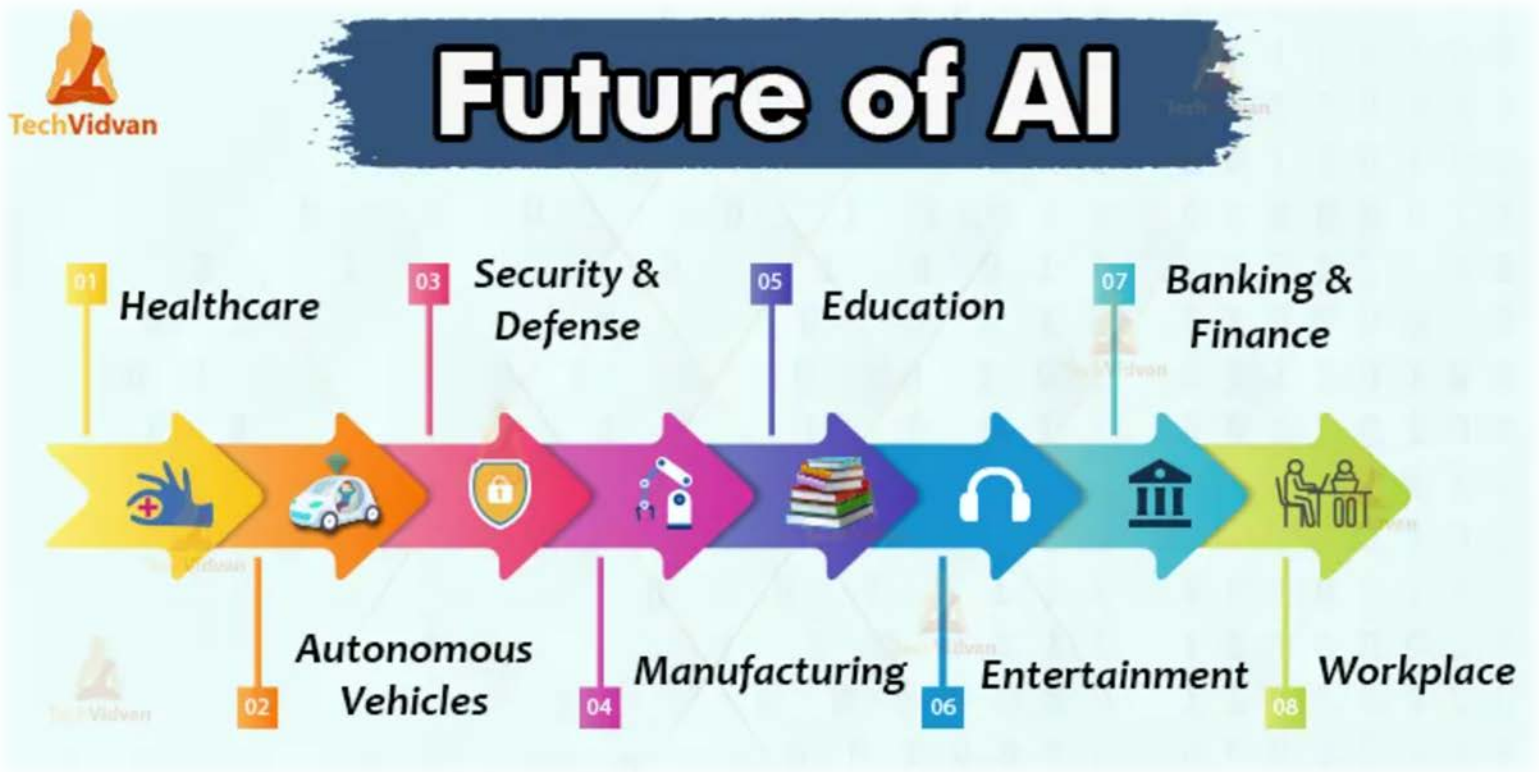
عنوان	سال	روش	تعبیه حساس به بافت؟	واحد یادگرفته شده
Word2Vec	۲۰۱۳	شبکه MLP	خیر	کلمه
Glove	۲۰۱۴	تجزیه ماتریسی	خیر	کلمه
FastText	۲۰۱۶	شبکه MLP	خیر	زیرکلمه (n-gram نویسه‌ها)
ELMO	۲۰۱۸	شبکه LSTM دوطرفه	بله	کلمه
BERT	۲۰۱۸	شبکه مبدل	بله	زیرکلمه

شبکه عصبی: کاربردهای مبدل‌ها

روش غالب در بیشتر (!) کاربردهای مدل‌سازی دنباله

بسترهای یادگیری عمیق

Name	Platform	Written In	Cuda	Parallel Execution	Trained Model	RNN	CNN
Tensorflow	Linux, Window,MacOs, Rasbian,Mobile, Webapp	Python, C++, Cuda	Yes	Yes	Yes	Yes	Yes
Pytorch	Linux, Window, MacOs	Python, C++, Cuda	Yes	Yes	Yes	Yes	Yes
Keras	Linux, MacOs, window	Python	Yes	Yes	Yes	Yes	Yes
Mxnet	Linux, Window, Mac,Mobile, Webapp	C++, Python, R, Julia, Scala, Go, Perl	Yes	Yes	Yes	Yes	Yes
Deeplearning4j	Window, Linux,Mac, Mobile	Java, Scala, Cuda, C++, Perl, Python, Closure	Yes	Yes	Yes	Yes	Yes
Microsoft CNTK	Window, Linux	C++	Yes	Yes	Yes	Yes	Yes


TechVidvan
Future of AI
01
Healthcare
02
Autonomous Vehicles
03
Security & Defense
04
Manufacturing
05
Education
06
Entertainment
07
Banking & Finance
08
Workplace