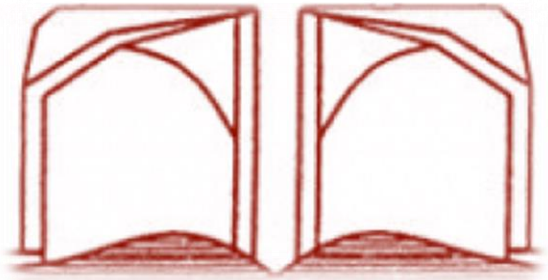


پردازش گفتار

شبکه عصبی مصنوعی

هادی ویسی

h.veisi@ut.ac.ir

دانشگاه تهران – دانشکده علوم و فنون نوین

فهرست

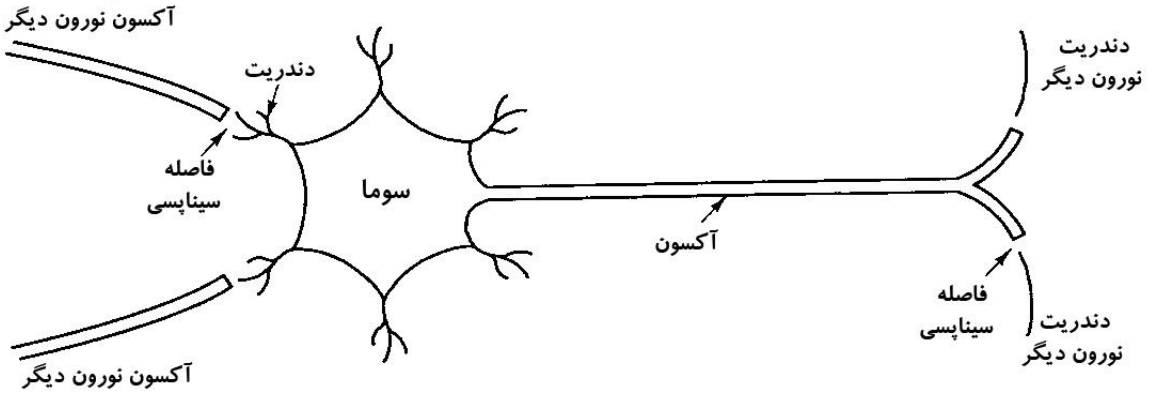
شبکه عصبی؟

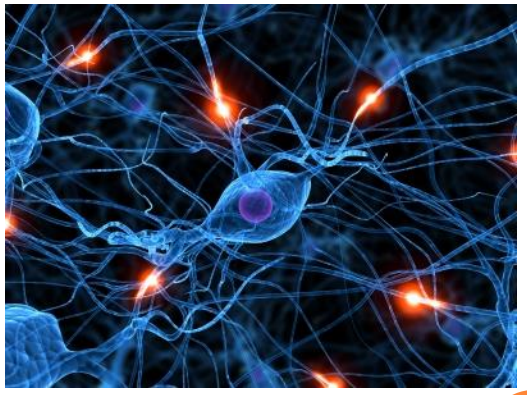
- مغز = شبکه‌ای بسیار بزرگ از عصب‌ها (نرون‌ها)
  - ۱۰۰.۰۰۰.۰۰۰.۰۰۰ نرون
  - ۱۰.۰۰۰ اتصال برای هر نرون
- شبکه عصبی مصنوعی = شبیه‌سازی شبکه عصبی طبیعی

شبکه عصبی طبیعی . . .

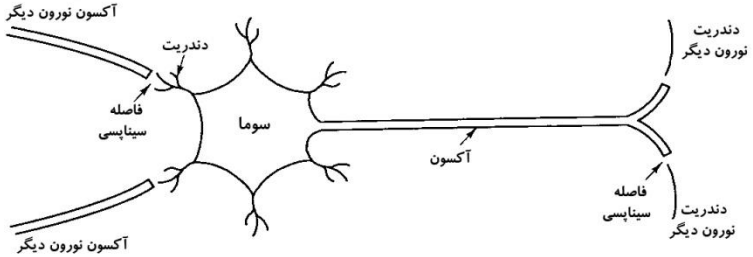
- عنصر پردازشگر تشکیل‌دهنده یک شبکه عصبی مصنوعی
  - نرون (Neuron) = عصب طبیعی (سلول مغزی)

- 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

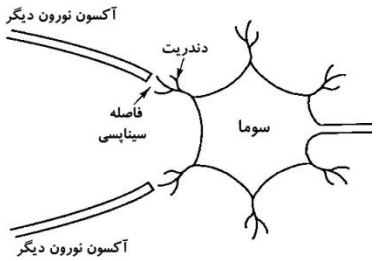
شبکه عصبی مصنوعی . . .

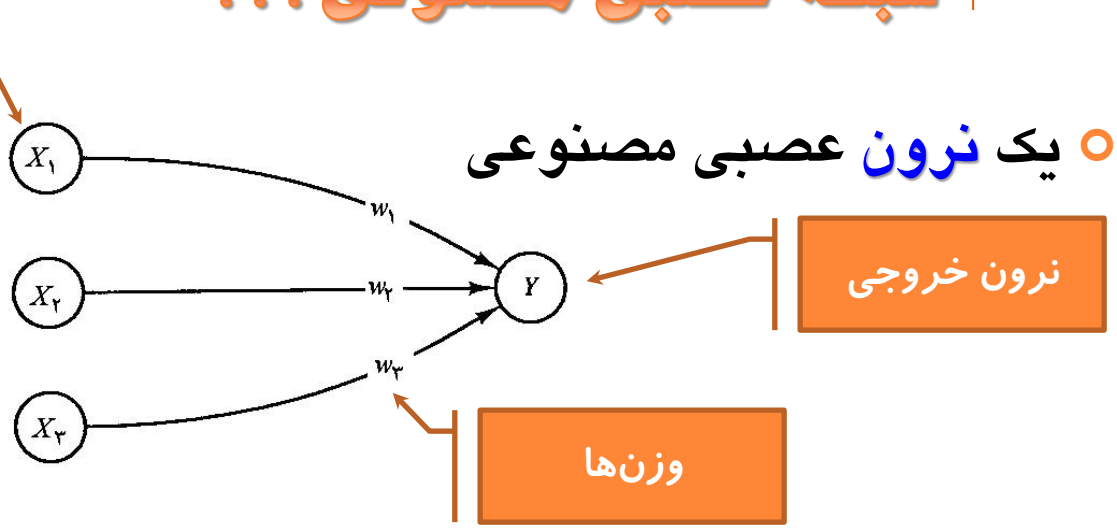
- **شبکه عصبی مصنوعی [Artificial Neural Network]**
  - یک سیستم پردازش اطلاعات با ویژگی‌های مشترکی با شبکه‌های عصبی طبیعی
  - تعمیم یافتهٔ مدل‌های ریاضی تشخیص انسان بر اساس زیست‌شناسی عصبی

- **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

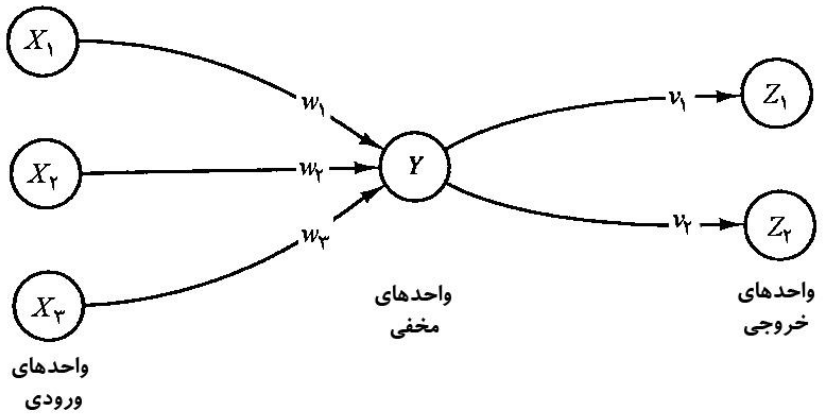
$$y = f(y\_in)$$

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

- تابع پله
- تابع سیگموید (Sigmoid)

شبکه عصبی مصنوعی . . .

- یک **شبکه** عصبی مصنوعی



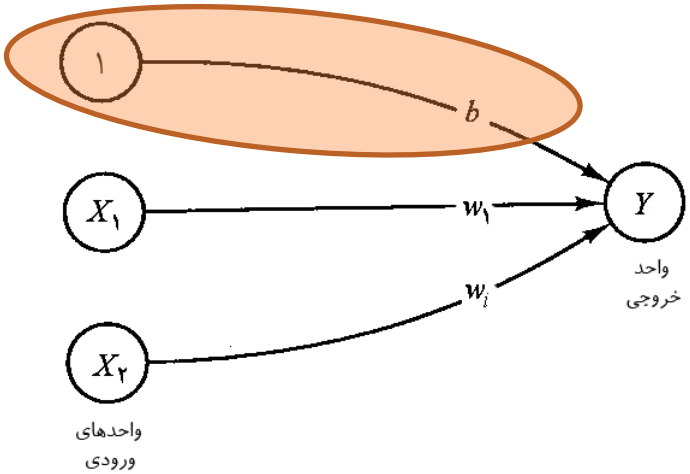
- سه لایه: ورودی، مخفی و خروجی
- دو دسته وزن: Wها و Vها

- 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.

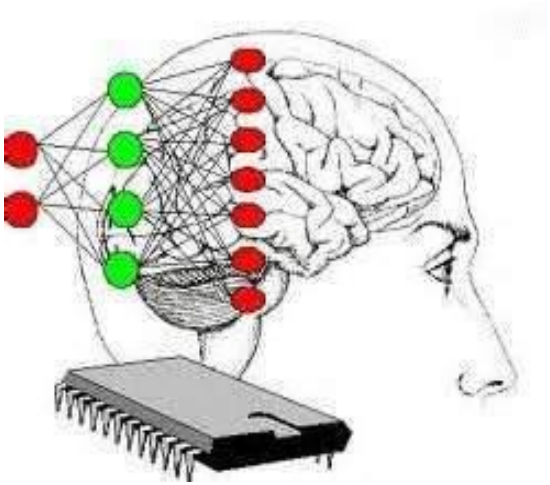


$$y\_in = 1 \times b + w_1 x_1 + w_2 x_2 = b + \sum_i w_i x_i$$

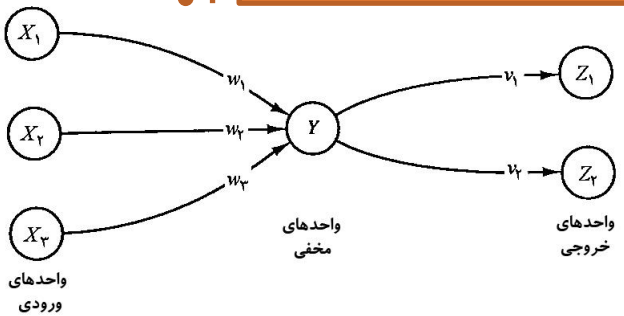
شبکه عصبی مصنوعی . . .

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



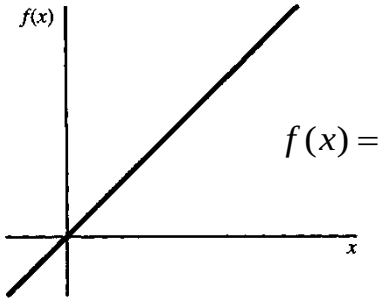
- معماری ۳ لایه
- وزن‌های W و V
- تابع فعال‌سازی برای Y و Z1 و Z2



شبکه عصبی مصنوعی . . .

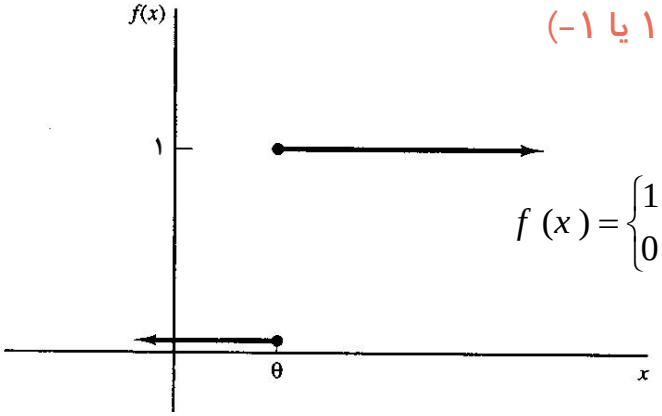
توابع فعال‌سازی متداول . . .

- **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-)



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

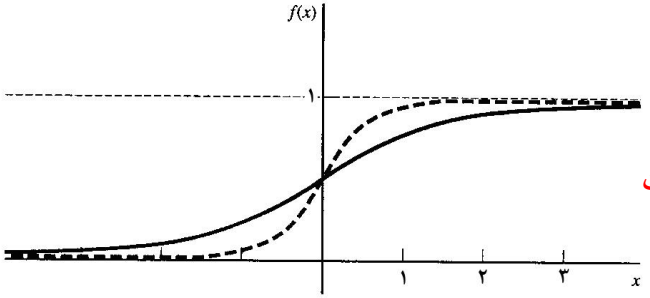
شبکه عصبی مصنوعی . . .

توابع فعال‌سازی متداول . . .

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

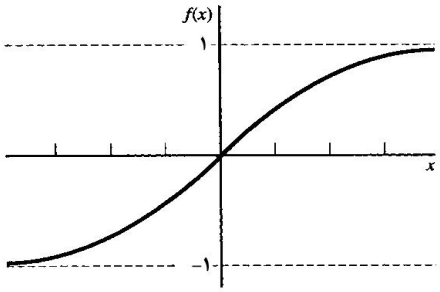
$$f(x) = \frac{1}{1+\exp(-\sigma x)}$$

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



- 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



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

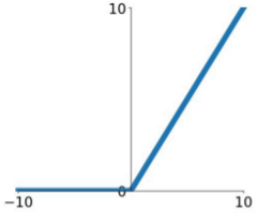
$$g'(x) = \frac{\sigma}{2}[1+g(x)][1-g(x)]$$

شبکه عصبی مصنوعی . . .

- توابع فعال‌سازی متداول . . .
  - ReLU: Rectified Linear Unit
    - سرعت محاسبه بالاتر نسبت به exp
    - رفع مشکل اشباع gradient (تا حدودی)
    - پرکاربرد در یادگیری عمیق
  - Softmax
    - تبدیل خروجی شبکه به احتمال
      - اعداد بین ۰ و ۱
      - جمع مقادیر برابر با ۱

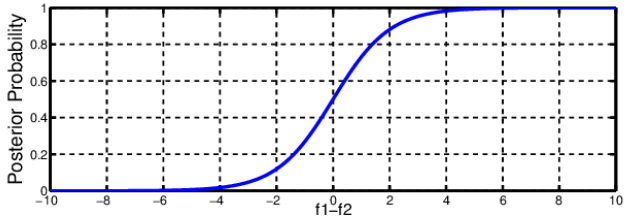
ReLU

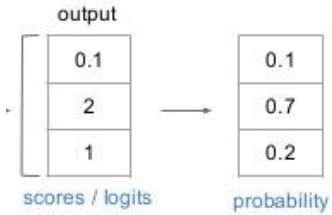
$$\max(0, x)$$



$$S(l_i) = \frac{e^{l_i}}{\sum_k e^{l_k}}$$

scores (logits)

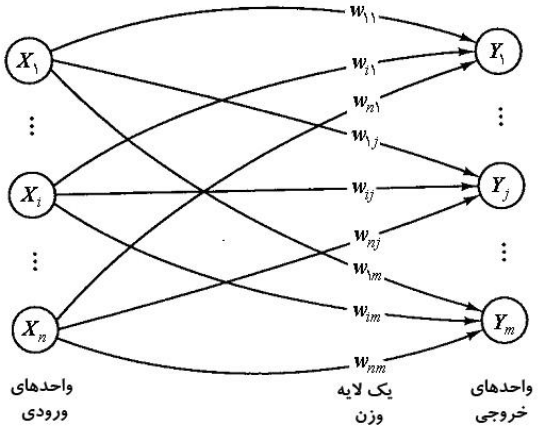




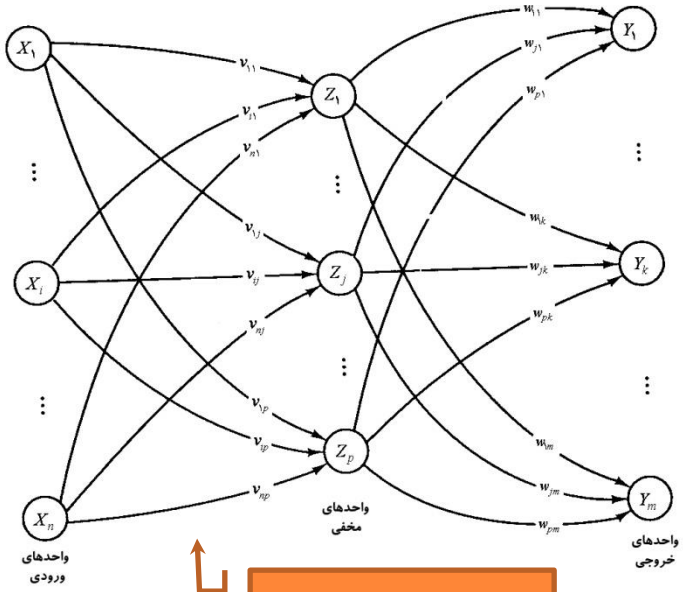
شبکه عصبی مصنوعی . . .

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

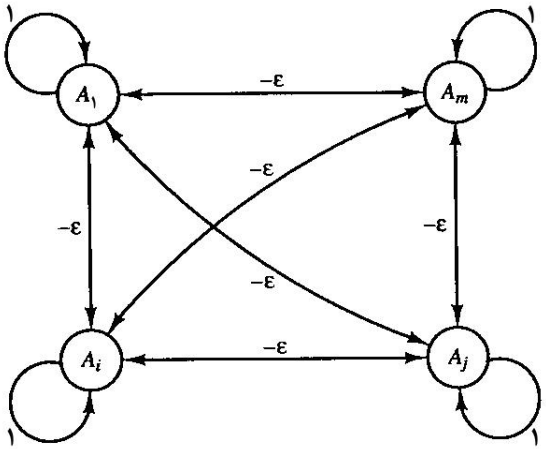


شبکه دولایه

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

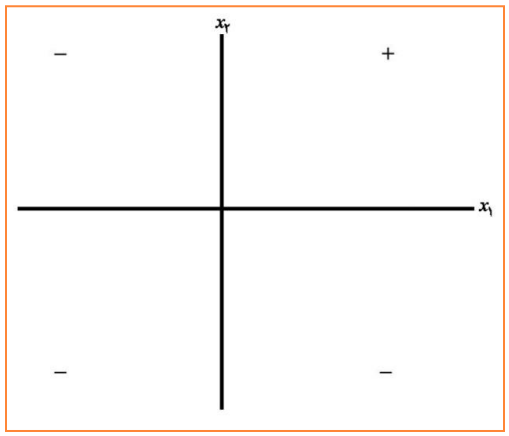


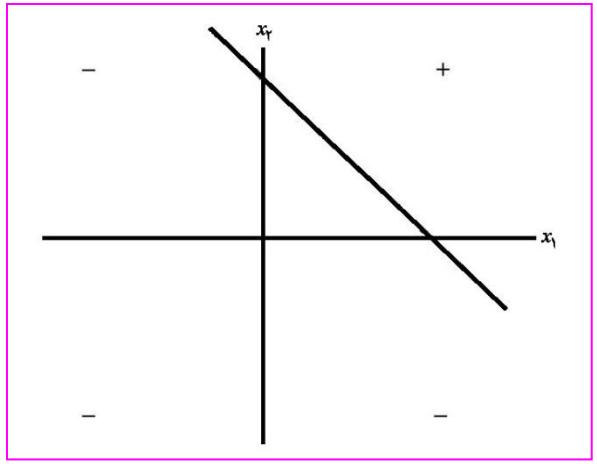
شبکه عصبی مصنوعی . . .

- الگوریتم آموزش/یادگیری (Training/Learning Algorithm)
  - یادگیری با ناظر
    - وزن های ثابت: تنظیم توسط طراح شبکه
      - استفاده در شبکه مک کلاچ–پیتز و هاپفیلد
    - هب: الهام از فرایند یادگیری انسان
      - استفاده در شبکه های هب و پرسپترون و شبکه های انجمنی (هاپفیلد)
    - دلتا: کمینه کردن خطای شبکه (کاهش گرادیان خطا)
      - استفاده در شبکه آدالاین
    - دلتای توسعه یافته: کمینه کردن خطای شبکه (کاهش گرادیان خطا)
      - استفاده در الگوریتم پس انتشار خطا و شبکه پرسپترون چند لایه
      - مبنای نظری یادگیری در بیشتر شبکه های عصبی مانند شبکه های بازگشتی
  - یادگیری بدون ناظر
    - کمینه کردن فاصله
      - استفاده در SOM و ART
    - بولتزمن: الهام از شبیه‌سازی سردشدن تدریجی (Simulated Annealing)

شبکه عصبی: مثال . . .

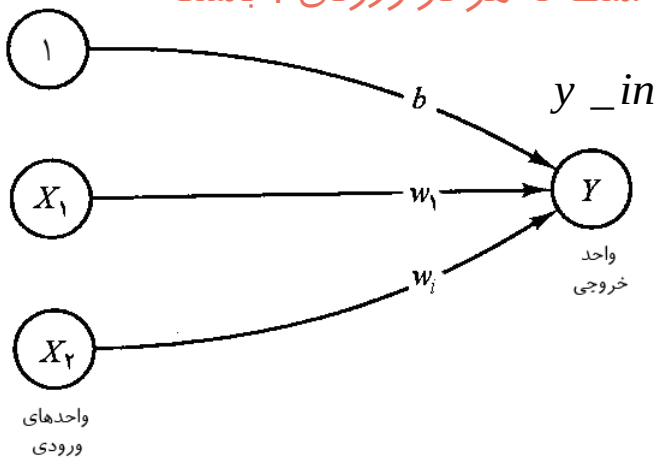
INPUT(x_1, x_2)	OUTPUT
(1 , 1)	+1
(1 , −1)	−1
(−1 , 1)	−1
(−1 , −1)	−1





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

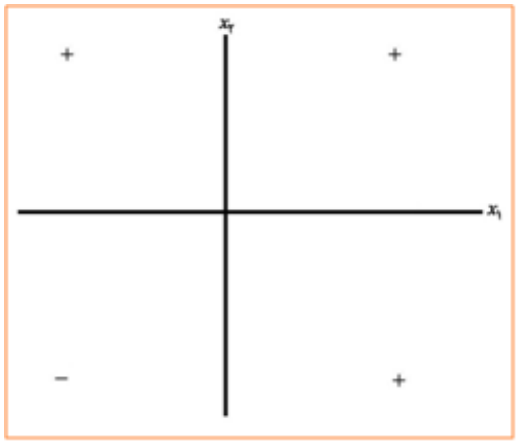
$$b + w_1 x_1 + w_2 x_2 = 0$$

- پاسخ

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

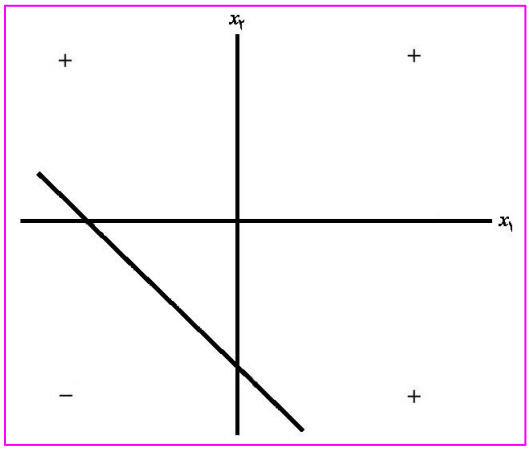
$$x_2 = -x_1 + 1$$

شبکه عصبی: مثال . . .



- 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

شبکهٔ هب ...

- اولین (و ساده‌ترین) قانون یادگیری برای شبکهٔ عصبی
- ایده اصلی یادگیری هب
  - یادگیری با تغییر استحکامات سیناپس‌های نرون‌ها (وزن‌های شبکه‌های عصبی) است
  - اگر دو نرون متصل به هم به طور هم‌زمان «فعال» باشند، وزن بین آنها باید افزایش یابد
  - هب دربارهٔ نرون‌هایی که به طور هم‌زمان برانگیخته نمی‌شوند، چیزی نمی‌گوید
    - یادگیری قوی‌تر = اگر دو نرون به طور هم‌زمان «غیرفعال» باشند، وزن‌ها افزایش یابد

- شبکهٔ هب یک لایه است
- به‌روز شدن (Update) وزن‌ها

$$w_i(new) = w_i(old) + x_i y$$

خروجی

وزن جدید

وزن قبلی

ورودی

- برای داده دودویی، اگر ورودی یا خروجی (یا هر دو) «غیرفعال» باشند، یادگیری صورت نمی‌گیرد

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

شبکهٔ هب: الگوریتم . . .

- مرحله ۰ – به تمام وزن‌ها مقدار اولیه صفر بدهید $w_i = 0 \quad (i = 1, \ldots, n)$
- مرحله ۱ – برای هر بردار آموزش ورودی و خروجی هدف، $s:t$، مراحل ۲ تا ۴ را انجام بده
- مرحله ۲ – فعال‌سازی‌های واحدهای ورودی را تعیین کن $x_i = s_i \quad (i = 1, \ldots, n)$
- مرحله ۳ – برای واحد خروجی فعال‌سازی را تعیین کن $y = t$
- مرحله ۴ – وزن‌ها و بایاس را به‌روز کن

$$w_i(new) = w_i(old) + x_i y \quad (i = 1, \ldots, n)$$

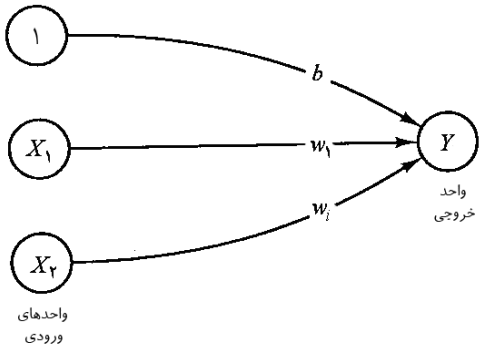
$$b(new) = b(old) + y$$

$$\mathbf{w}(new) = \mathbf{w}(old) + \mathbf{x}.y \quad \overset{\Delta w = x.y}{\Rightarrow} \quad \mathbf{w}(new) = \mathbf{w}(old) + \Delta \mathbf{w}$$

داده‌های آموزشی فقط یک بار به شبکه نشان داده شده و آموزش به اتمام می‌رسد

شبکهٔ هب: کاربرد . . .

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



- تابع AND با ورودی‌ها و هدف‌های دودویی . . .
  - تغییر وزن

$$\Delta w_1 = x_1 t, \quad \Delta w_2 = x_2 t, \quad \Delta b = 1.t = t$$

$$\mathbf{w}(new) = \mathbf{w}(old) + \Delta\mathbf{w}$$

  - برای ورودی اول

$$x_1 = 1, \quad x_2 = 1, \quad b = 1, \quad t = 1$$

INPUT			TARGET	WEIGHT CHANGES			WEIGHTS		
$(x_1$	x_2	1)	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
							(0	0	0)
(1	1	1)	1	(1	1	1)	(1	1	1)

مقدار اولیه

$$x_2 = -x_1 - 1$$

شبکهٔ هب: کاربرد . . .

- تابع AND با ورودی‌ها و هدف‌های دودویی
  - برای دومین، سومین و چهارمین ورودی

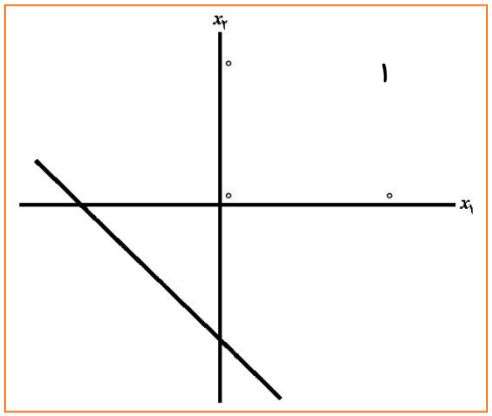
INPUT			TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	t	Δw_1	Δw_2	b	w_1	w_2	b
1	0	1	0	0	0	0	1	1	1
0	1	1	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1	1

یادگیری رخ نمی‌دهد
وزن‌ها تغییر نمی‌کند

الگوهایی با مقدار هدف صفر یا «غیرفعال»

استفاده از نمایش دودویی

پاسخ نادرست



$$x_2 = -x_1 - 1$$

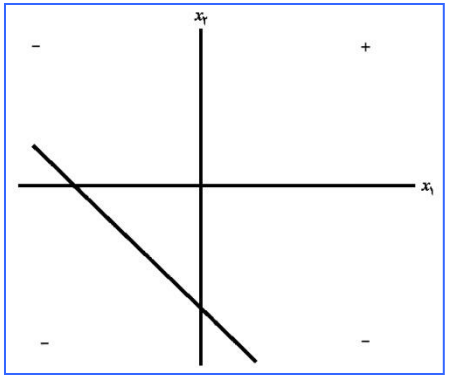
شبکهٔ هب: کاربرد . . .

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

- تابع AND برای ورودی‌ها و هدف‌های دوقطبی . . .
  - اولین ورودی

INPUT			TARGET	WEIGHT CHANGES			WEIGHTS		
$(x_1$	x_2	$1)$	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
							$(0$	0	$0)$
$(1$	1	$1)$	1	$(1$	1	$1)$	$(1$	1	$1)$

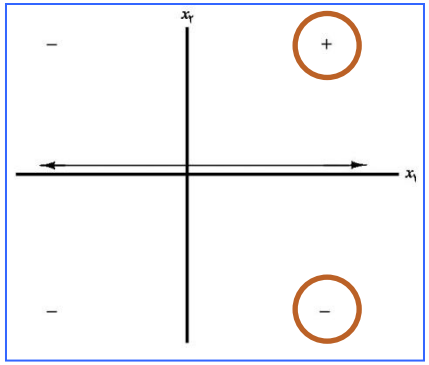
$$x_2 = -x_1 - 1$$



INPUT			TARGET	WEIGHT CHANGES			WEIGHTS		
$(x_1$	x_2	$1)$	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
							$(1$	1	$1)$
$(1$	-1	$1)$	-1	$(-1$	1	$-1)$	$(0$	2	$0)$

$$x_2 = 0$$

  - دومین ورودی



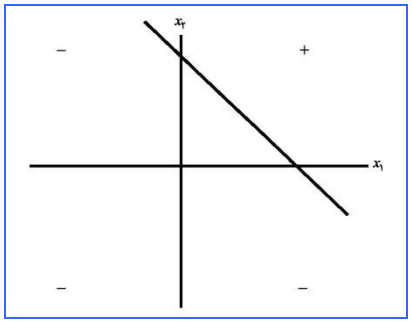
پاسخ درست برای
دو نمونه آموزش

شبکهٔ هب: کاربرد . . .

- تابع AND برای ورودی‌ها و هدف‌های دوقطبی
  - سومین ورودی

INPUT			TARGET	WEIGHT CHANGES			WEIGHTS		
$(x_1$	x_2	$1)$	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
							$(0$	2	$0)$
$(-1$	1	$1)$	-1	$(1$	-1	$-1)$	$(1$	1	$-1)$

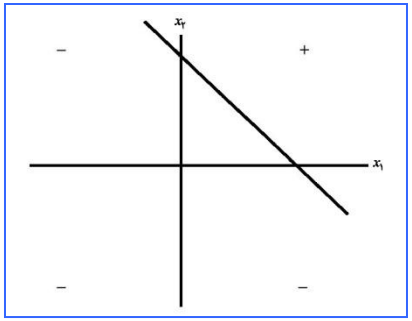
$$x_2 = -x_1 + 1$$



  - چهارمین ورودی

INPUT			TARGET	WEIGHT CHANGES			WEIGHTS		
$(x_1$	x_2	$1)$	t	$(\Delta w_1$	Δw_2	$\Delta b)$	$(w_1$	w_2	$b)$
							$(1$	1	$-1)$
$(-1$	-1	$1)$	-1	$(1$	1	$-1)$	$(2$	2	$-2)$

$$x_2 = -x_1 + 1$$



شبکه عصبی پرسپترون . . .

- 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

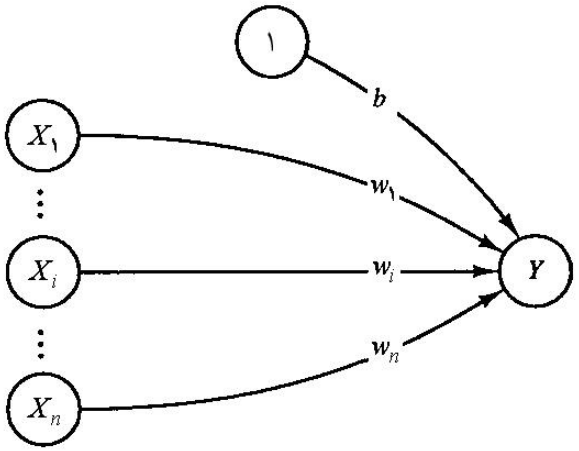
شبکه عصبی پرسپترون: ساختار . . .

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.

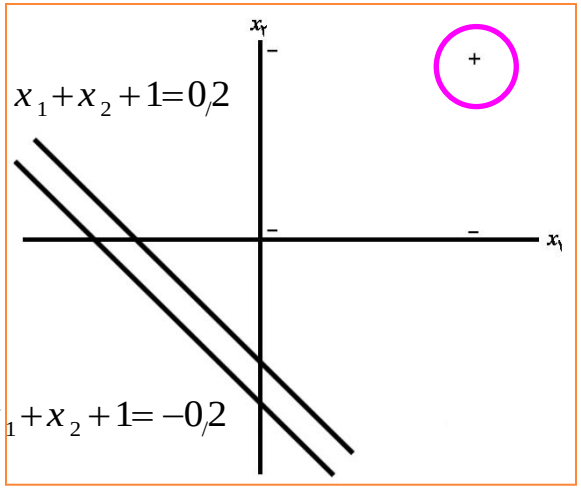
خطا = برابر نبودن پاسخ شبکه و مقدار هدف

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

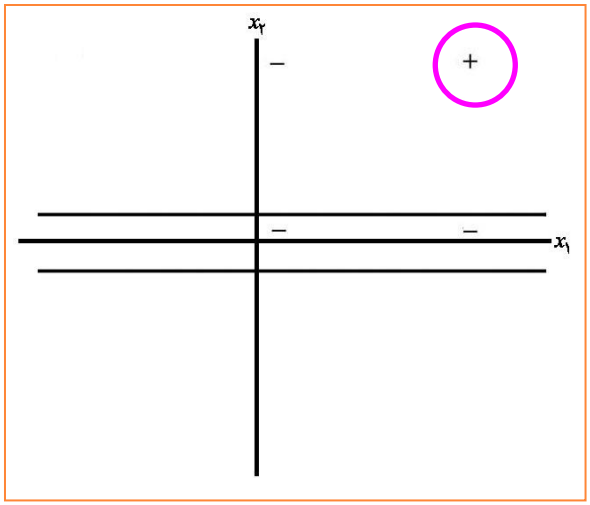
INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1



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

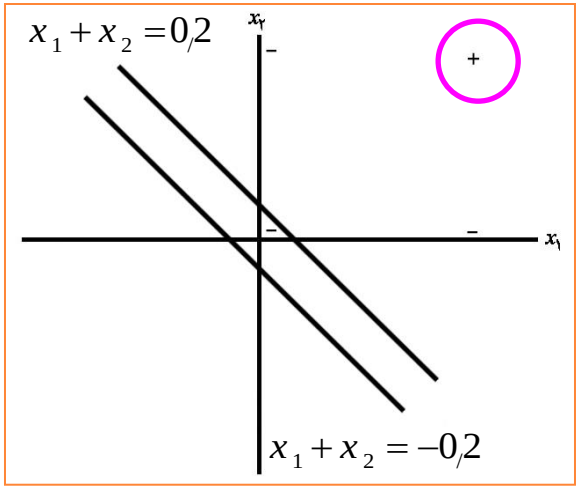
کامل شدن اولین دور آموزش (Epoch)

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

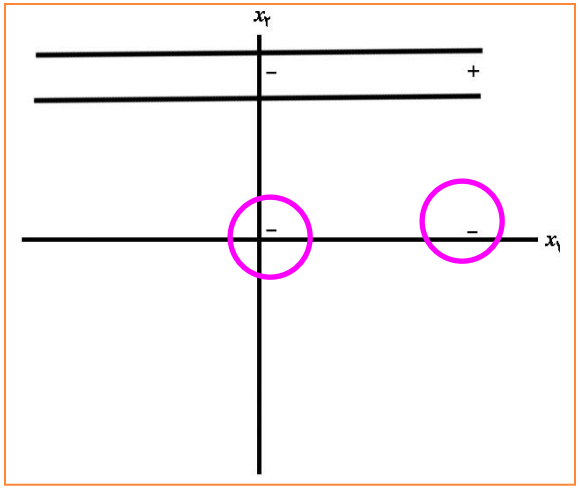
INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	–1
1	1	1	–1	–1	1	1	1	1	1	1	0



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	Δw_1	Δw_2	Δb	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)

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 . . .

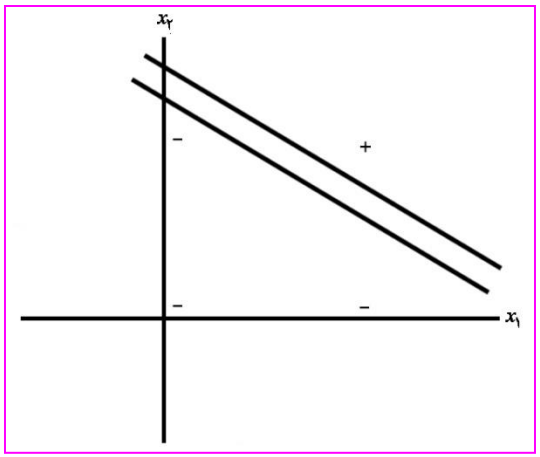
- تابع AND با ورودی‌های دودویی و هدف‌های دوقطبی . . .
  - پنجمین، ششمین، .... دور آموزش

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

  - نهمین دور آموزش

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

  - 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

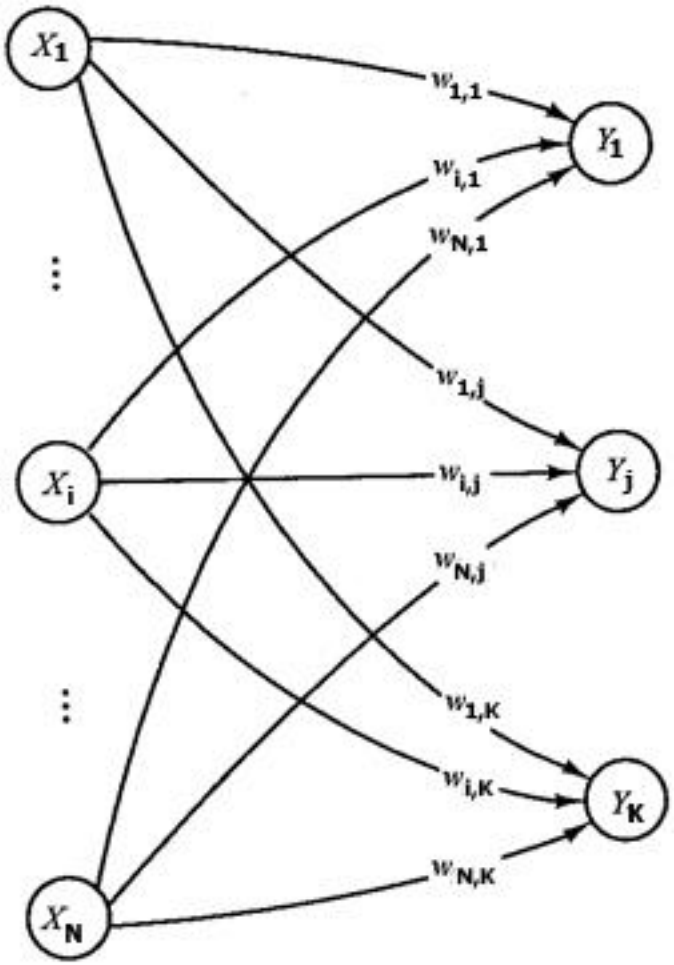
INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
x_1	x_2	1	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
1	−1	1	1	1	−1	−1	1	−1	0	2	0
−1	1	1	2	1	−1	1	−1	−1	1	1	−1
−1	−1	1	−3	−1	−1	0	0	0	1	1	−1

- 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

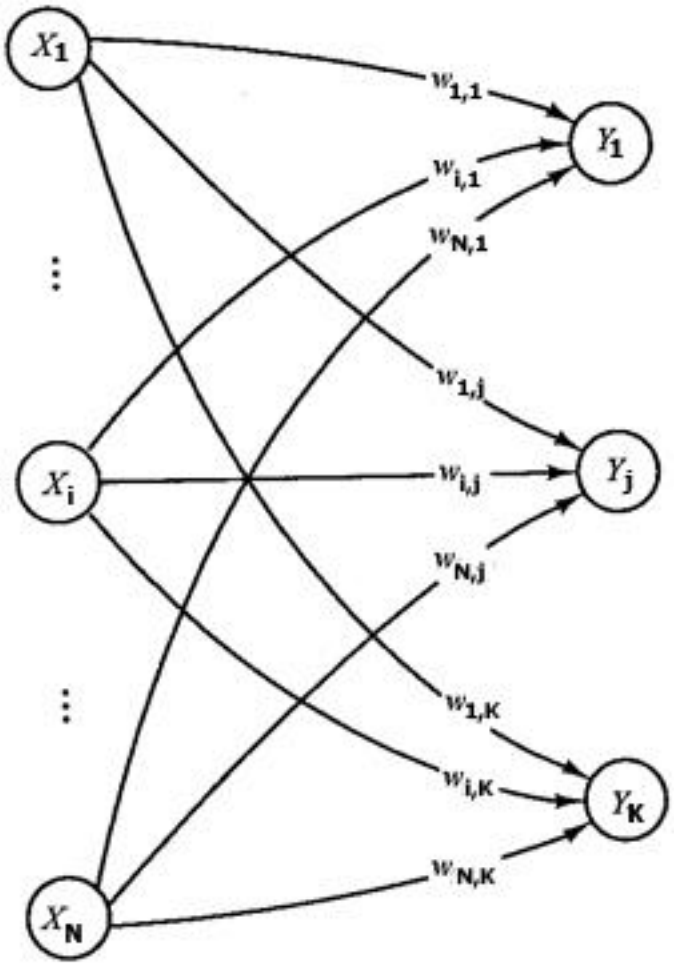
- بهبود نتایج با تغییر نمایش دودویی به دوقطبی

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 برچسب
    - خروجی: وزن‌های شبکه = یک ماتریس $N*K$ (مدل)
  - آزمون
    - ورودی: یک سند با عنوان نامشخص
      - تبدیل سند به یک بردار N بعدی و دادن آن به شبکه
    - خروجی: نرونی (دسته ای) که مقدار بزرگ‌تر دارد

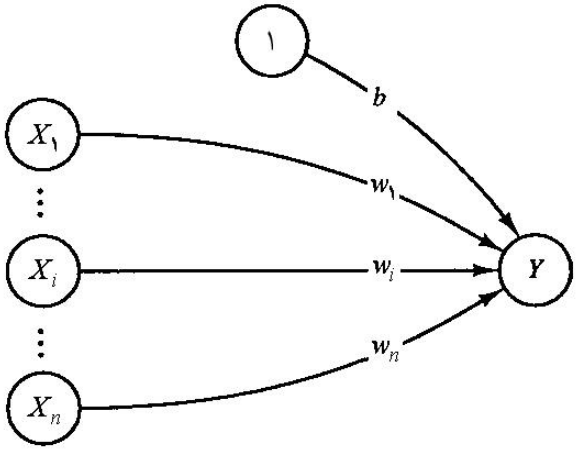
شبکه عصبی پرسپترون: مثال . . .



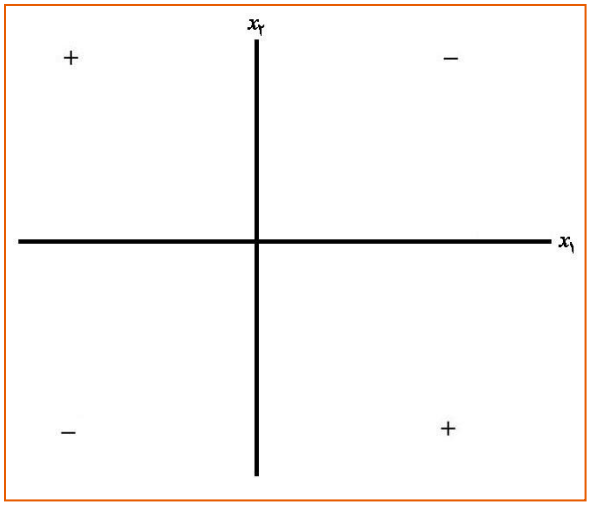
- تشخیص زبان
  - تعداد K دسته (زبان) $C = \{c_1, c_2, \ldots, c_K\}$
    - تعداد K نرون خروجی
  - داده = سند متنی یا سند صوتی
    - تبدیل هر سند به یک بردار ویژگی N بعدی
      - تعداد N نرون ورودی

شبکه عصبی پرسپترون: مثال . . .

- تشخیص 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: مثال



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 عصبی پرسپترون: همگرایی قانون یادگیری

قضیه

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

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

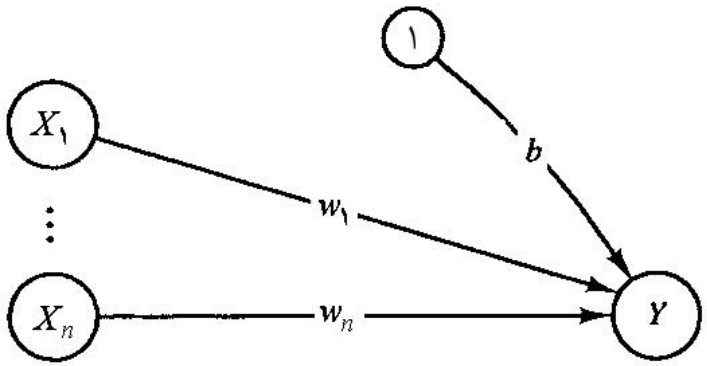
آنگاه برای هر بردار اولیه w، قانون یادگیری پرسپترون به بردار وزنی نزدیک می‌شود (نه الزاماً منحصر به فرد و نه الزاماً w^*) که برای تمام الگوهای آموزش پاسخ صحیحی می‌دهد و این کار در مراحلی با تعداد متناهی انجام می‌شود.

- 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 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
  - شبکه بعد از آموزش

x_1	x_2	t
1	1	1
1	0	−1
0	1	−1
0	0	−1

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

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

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

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

شبکه آدالاین: قانون یادگیری

قانون دلتا

- کمینه کردن خطای بین خروجی شبکه و مقدار هدف متناظر
- خطا = مربعات تفاضل

$$E = (t - y\_in)^2$$

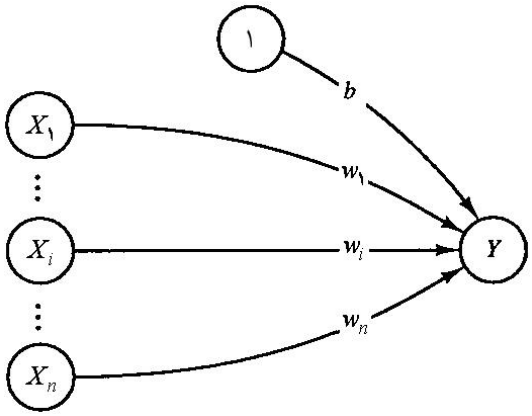
$$y\_in = \sum_{i=1}^{n} x_i w_i$$

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

$$-\frac{\partial E}{\partial w_I} = -2(t - y\_in)\frac{\partial y\_in}{\partial w_I} = -2(t - y\_in)x_I$$

$$\Delta w_I = \alpha(t - y\_in)x_I$$

الگوریتم‌های آموزش . . .

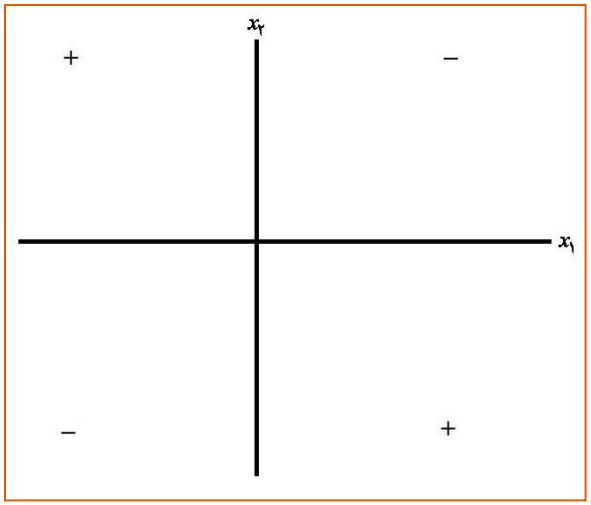


- الگوریتم آموزش = نحوه بدست آوردن وزن‌ها
  - صورت کلی $w\,(new\,) = w\,(old\,) + \Delta w$
  - روش‌ها

روش	ساختار	قانون یادگیری
هب	یک لایه	تغییر وزن در صورت فعال شدن همزمان ورودی (x) و خروجی (t) $w\,(new\,) = w\,(old\,) + x \cdot t$
پرسپترون	یک لایه	مشابه قانون یادگیری هب $w\,(new\,) = w\,(old\,) + \alpha\, x \cdot t$ + تکرار (Iterative): مجموعه آموزش چندین بار به شبکه داده می‌شود + در نظر گرفتن خطا: تغییر وزن‌ها فقط وقتی که خطا داریم + در نظر گرفتن نرخ یادگیری
آدالاین	یک لایه	قانون یادگیری دلتا: کمینه کردن میانگین مربعات خطا (LMS) +تکرار+مفهوم خطا $w\,(new\,) = w\,(old\,) + \alpha\,.\,(t - y\,).x$

مشکل: فقط آموزش شبکه یک لایه

الگوریتم‌های آموزش



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

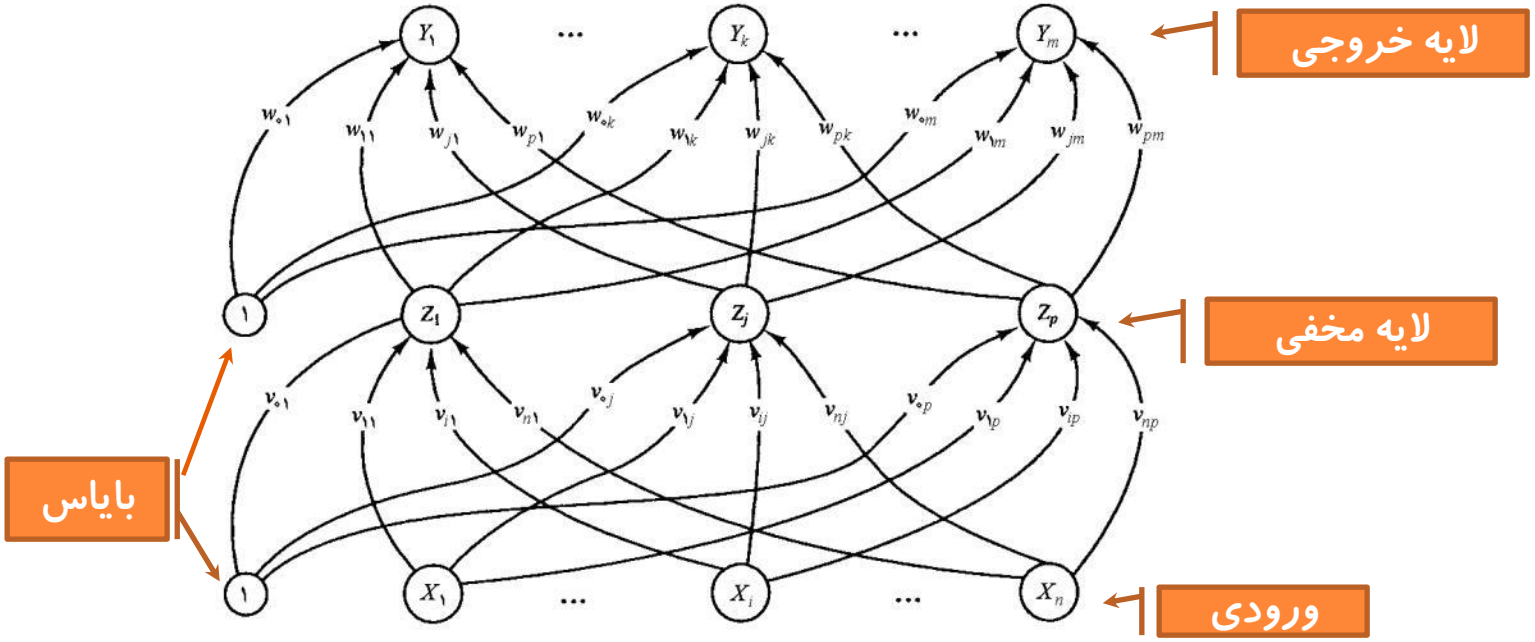
- **مثال: تابع XOR**
  - **دو ورودی (دوقطبی) و یک خروجی (دوقطبی)**
    - وقتی خروجی ۱ است که فقط یکی از ورودی‌ها ۱ باشد
  - **حل؟**
    - الگوریتم‌های هب، پرسپترون و آدالاین همگرا نمی شوند (جواب درست نمی دهند)
  - **فضای داده‌های ورودی به صورت خطی جدایی‌پذیر (Linearly Spreadable) نیست.**
    - هیچ خط مستقیم نمی‌تواند نقاط مثبت و منفی را جدا کند
    - نیاز به شبکه با بیش از یک لایه (بیش از یک خط جداکننده یا یک مرز غیرخطی)
      - تلاش برای حل با شبکه مادالاین = آدالاین چند لایه

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**

پرسپترون چندلایه (ساختار) . . .

- شبکه سه لایه
  - یک لایه ورودی (واحدهای X)،
  - یک لایه واحدهای مخفی (واحدهای Z)
  - یک لایه خروجی (واحدهای Y)



پرسپترون چندلایه (الگوریتم آموزش) . . .

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

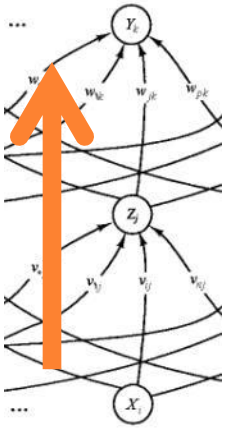
- مبنای ریاضی الگوریتم پس‌انتشار = بهینه‌سازی کاهش گرادیان (Gradient Descent)
  - گرادیان (شیب) یک تابع = نمایانگر جهتی که تابع در آن سریع‌تر افزایش می‌یابد
  - شیب با علامت منفی = جهتی نشان دهنده کاهش سریع‌تر آن تابع
  - در اینجا تابع مورد نظر = تابع خطای شبکه
  - متغیرهای مورد نظر = وزن‌های شبکه

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

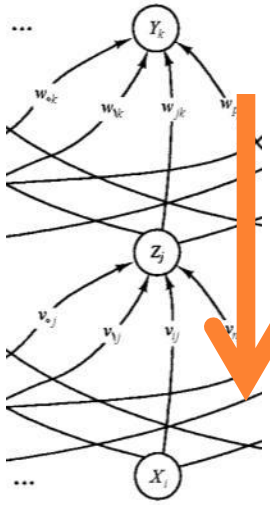


$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \qquad z_j = f(z\_in_j)$$

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

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

پرسپترون چندلایه (الگوریتم آموزش) . . .



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)

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

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.

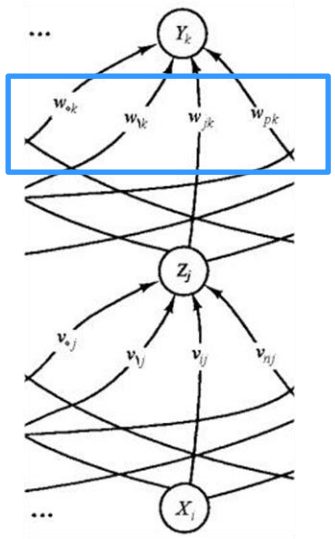
پرسپترون چندلایه: استخراج قوانین یادگیری . . .

- برای وزن‌های واحدهای خروجی

$$y\_in_k = \sum_j z_j w_{jk} \Rightarrow y_k = f(y\_in_k)$$

- خطا (تابعی از وزن‌ها): باید حداقل شود

$$E = 0{,}5 \sum_k [t_k - y_k]^2$$



$$\frac{\partial E}{\partial w_{JK}} = \frac{\partial}{\partial w_{JK}} 0{,}5 \sum_K [t_k - y_k]^2$$

$$= \frac{\partial}{\partial w_{JK}} 0{,}5 [t_K - f(y\_in_K)]^2$$

$$= -[t_K - y_K] \frac{\partial}{\partial w_{JK}} f(y\_in_K)$$

$$= -[t_K - y_K] f'(y\_in_K) \frac{\partial}{\partial w_{JK}} (y\_in_K)$$

$$= -[t_K - y_K] f'(y\_in_K) z_J = -\delta_K z_J$$

$$\Delta w_{jk} = -\alpha \frac{\partial E}{\partial w_{jk}}$$

$$= \alpha [t_k - y_k] f'(y\_in_k) z_j$$

$$= \alpha \delta_k z_j$$

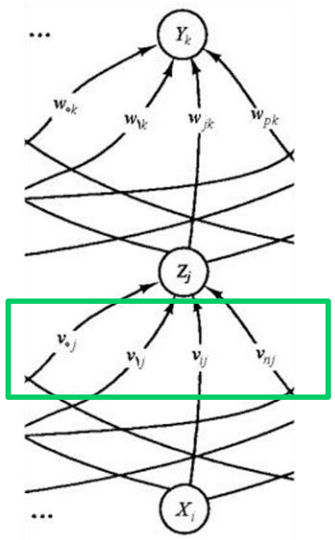
پرسپترون چندلایه: استخراج قوانین یادگیری

- برای وزن‌های واحدهای مخفی
  - داریم (از اسلاید قبلی)

$$y\_in_k = \sum_j z_j w_{jk} \Rightarrow y_k = f(y\_in_k)$$

  - خطا (تابعی از وزن‌ها): باید حداقل شود

$$E = 0{,}5 \sum_k [t_k - y_k]^2$$



$$\frac{\partial E}{\partial v_{IJ}} = -\sum_k [t_k - y_k] \frac{\partial}{\partial v_{IJ}} y_k$$

$$= -\sum_k [t_k - y_k] f'(y\_in_k) \frac{\partial}{\partial v_{IJ}} y\_in_k$$

$$= -\sum_k \delta_k \frac{\partial}{\partial v_{IJ}} y\_in_k$$

$$= -\sum_k \delta_k w_{Jk} \frac{\partial}{\partial v_{IJ}} z_J$$

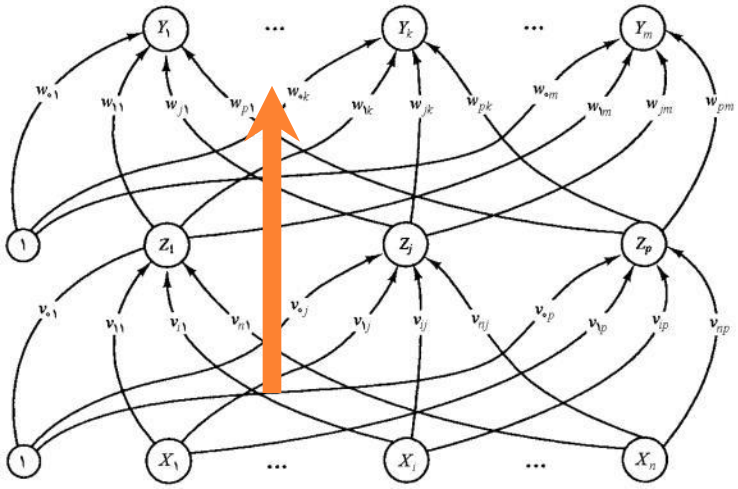
$$= -\sum_k \delta_k w_{Jk} f'(z\_in_J) x_I = -\delta_J x_I$$

$$\Delta v_{ij} = -\alpha \frac{\partial E}{\partial v_{ij}}$$

$$= \alpha f'(z\_in_j) x_i \sum_k \delta_k w_{jk}$$

$$= \alpha \delta_j x_i$$

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:

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

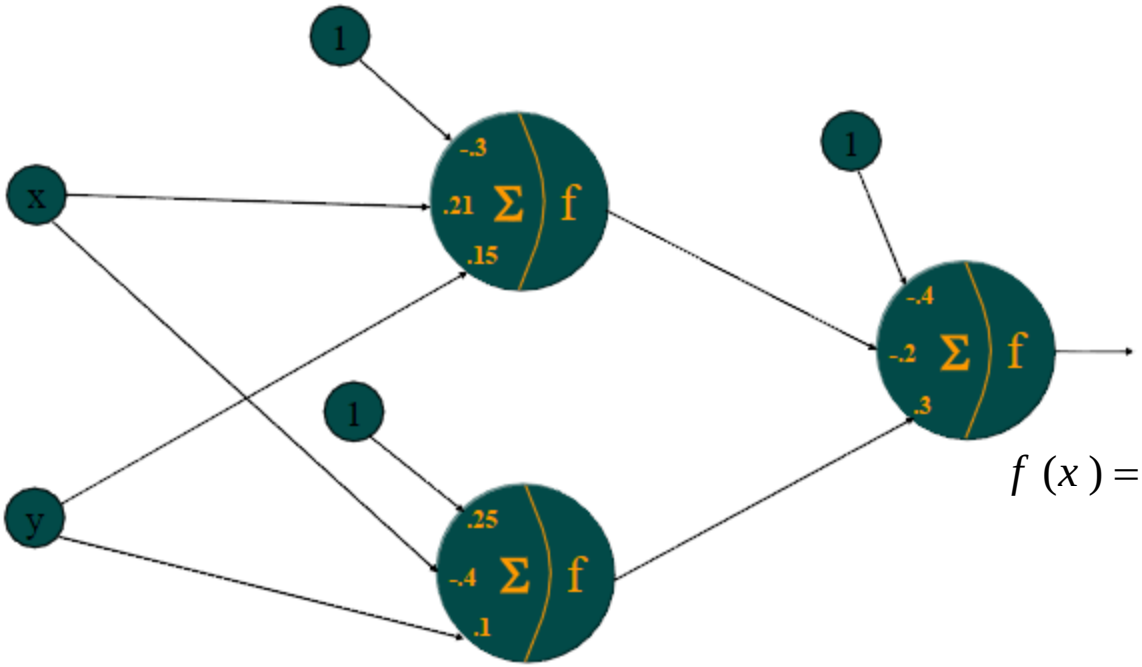
- مرحلهٔ ۴: برای واحدهای خروجی:

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

پرسپترون چندلایه (مثال) . . .

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

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

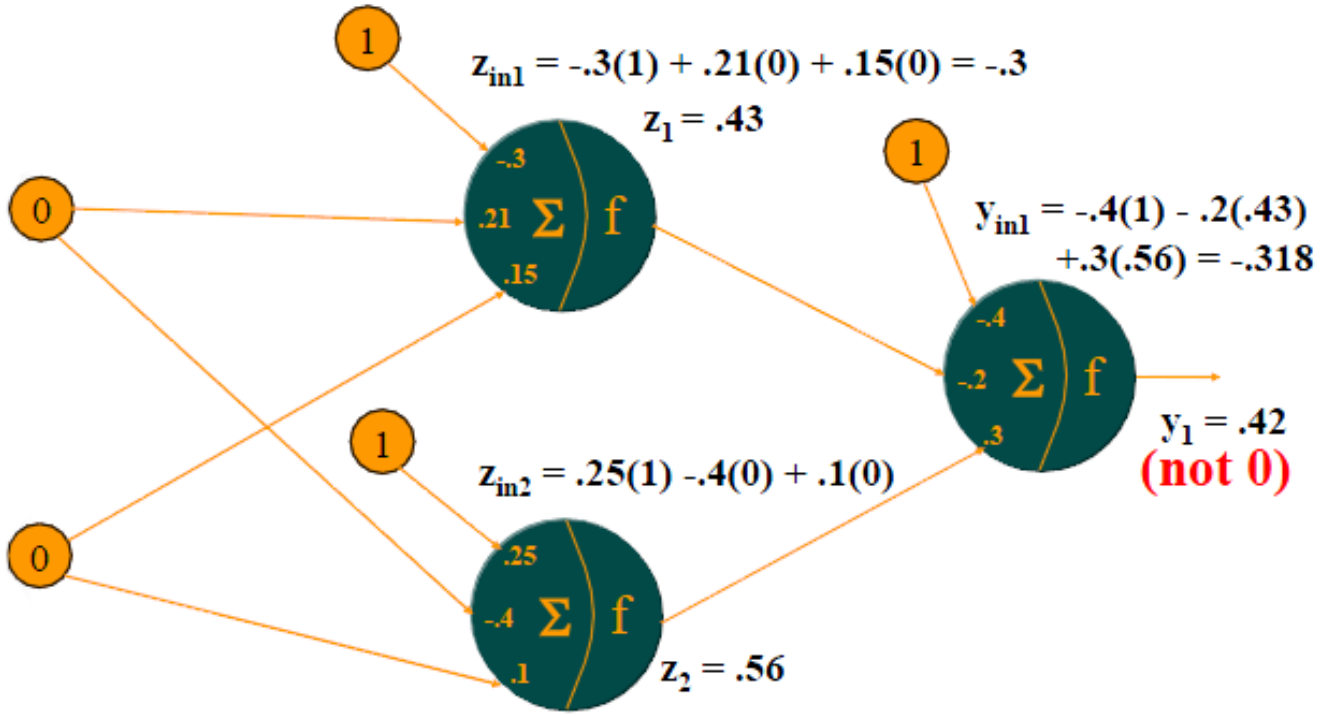


$$f(x) = \frac{1}{1+\exp(-x)}$$

پرسپترون چندلایه (مثال) . . .

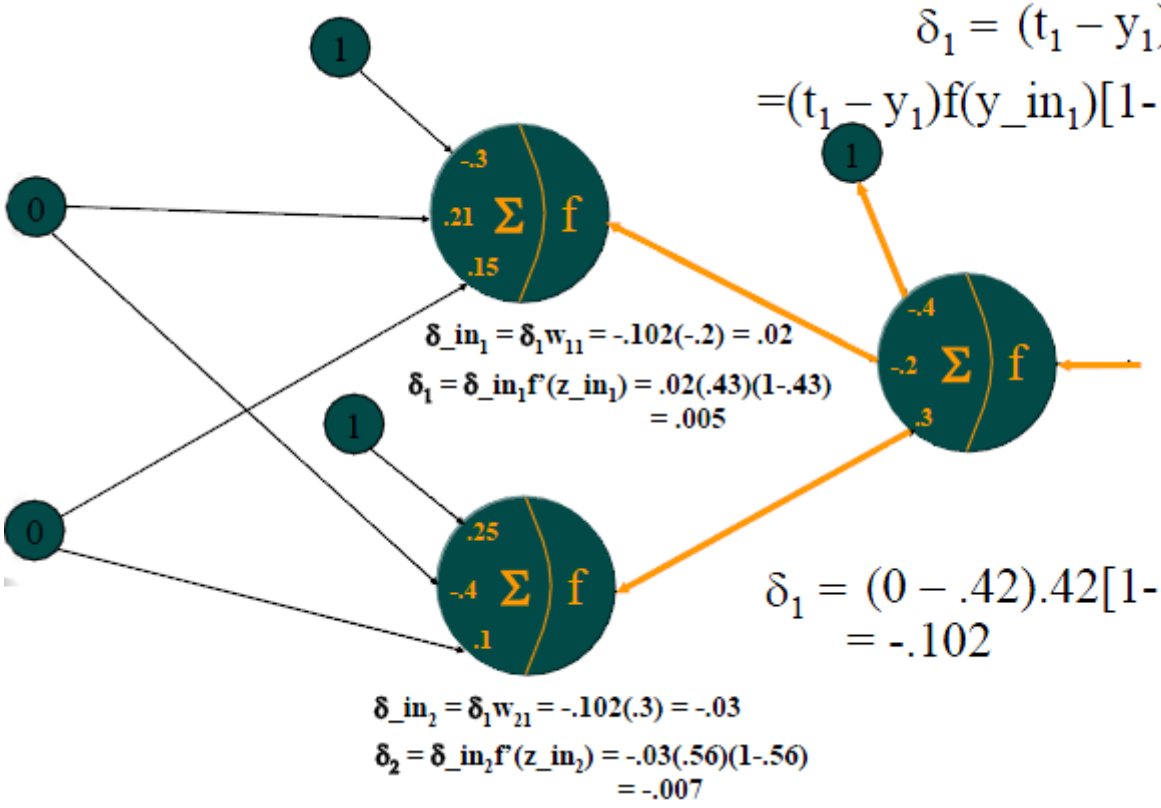
- تابع XOR: نمایش دودویی (۲ از ۶) . . .
  - پیشخور کردن ورودی

x_1	x_2	y
0	0	0



پرسپترون چندلایه (مثال) . . .

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



پرسپترون چندلایه (مثال) . . .

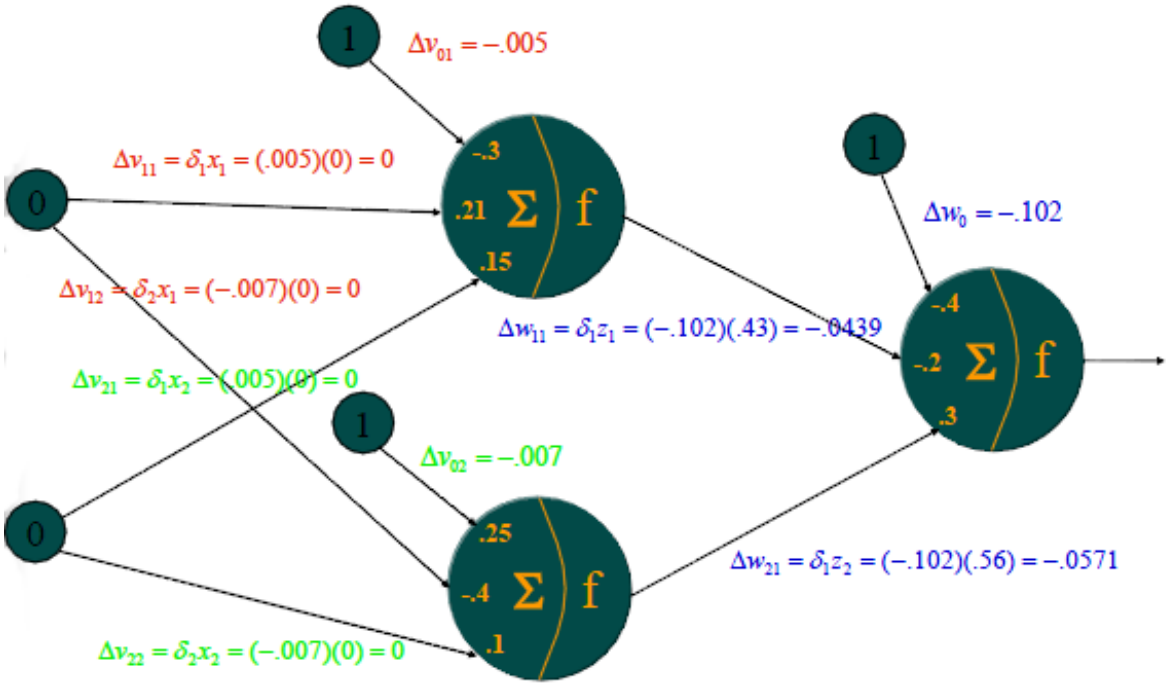
- تابع XOR: نمایش دودویی (۴ از ۶) . . .
  - محاسبه وزن‌ها

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

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

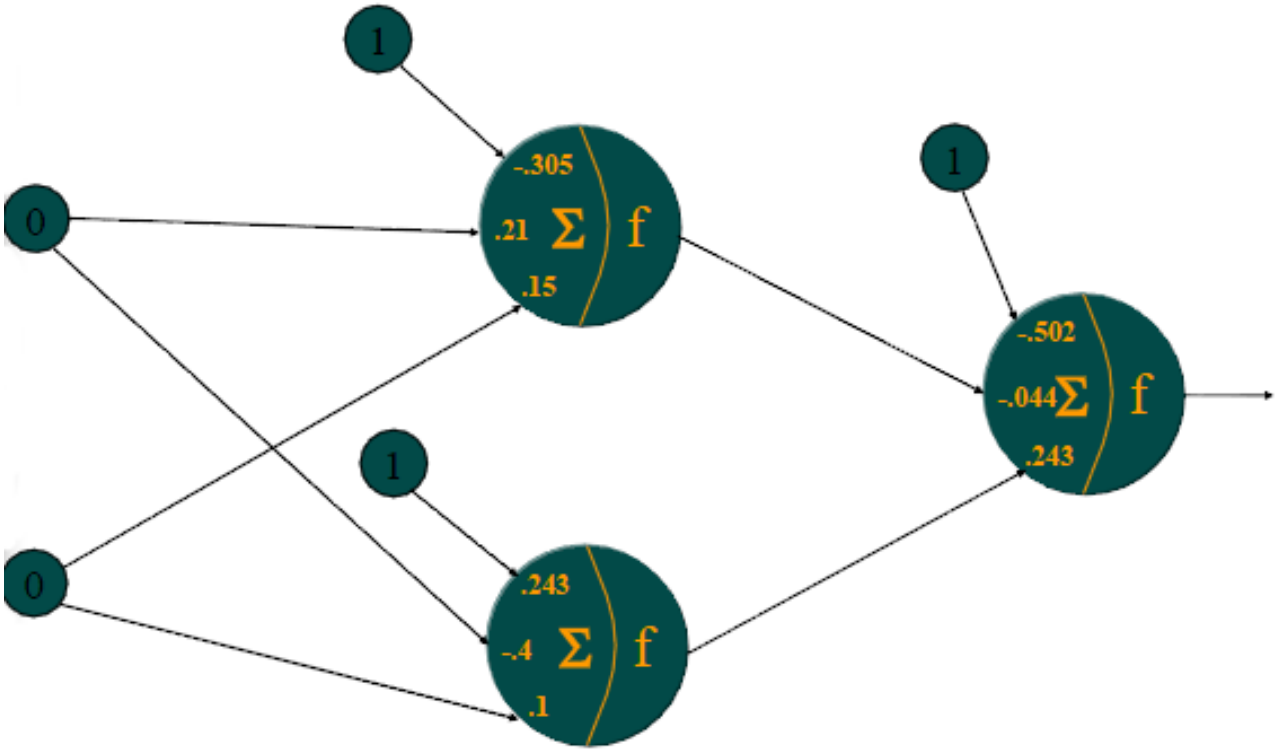
$\Delta w_{j1} = \alpha \delta_1 z_j \quad j = 1,2$

$\Delta w_0 = \alpha \delta_1$



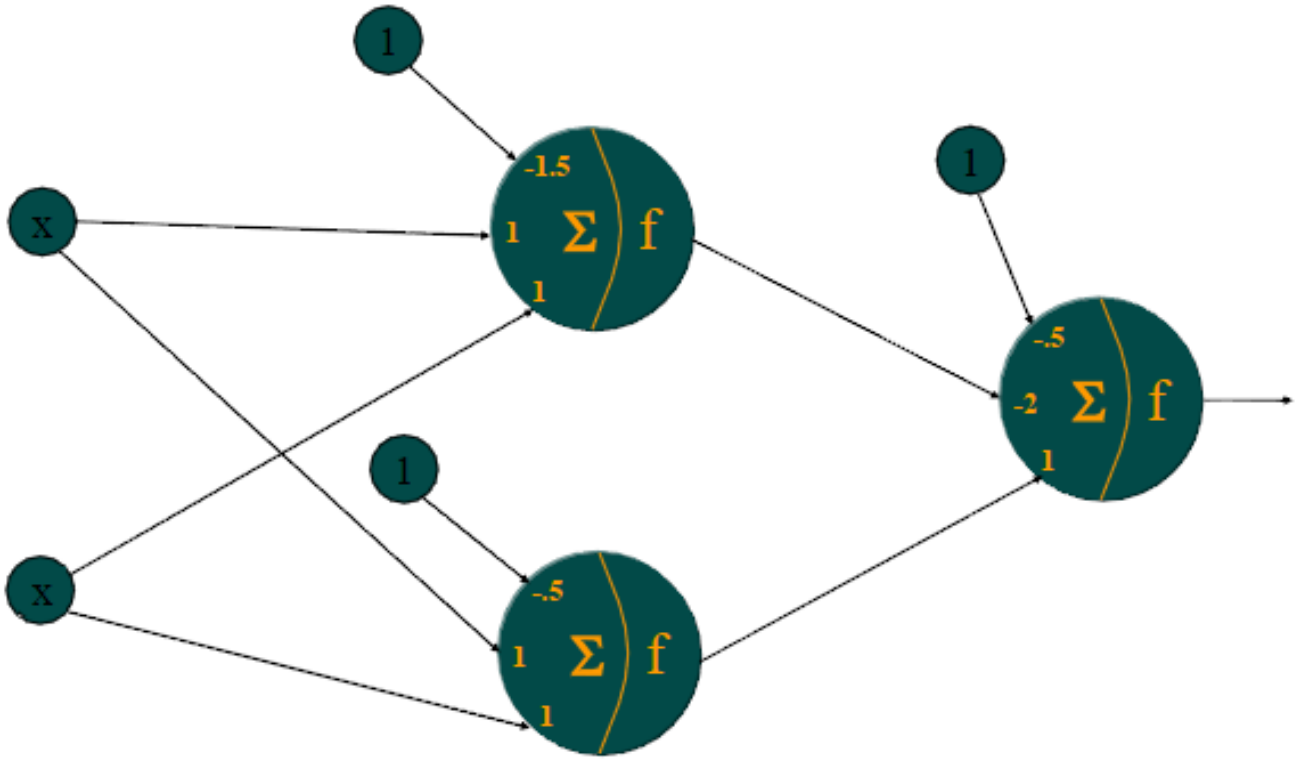
پرسپترون چندلایه (مثال) . . .

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

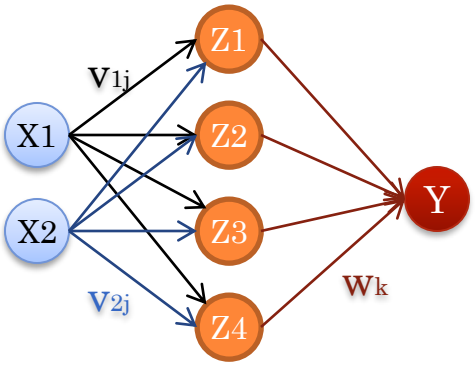


پرسپترون چندلایه (مثال) . . .

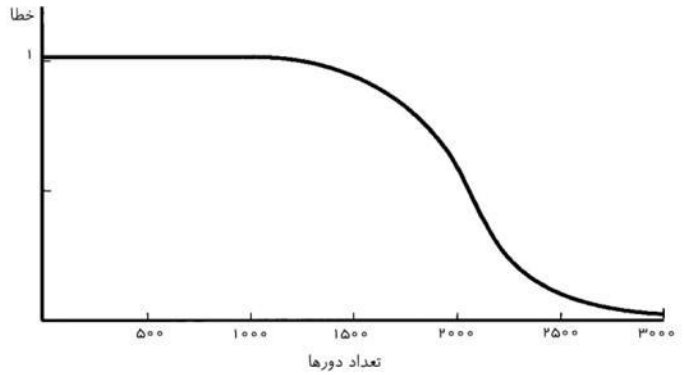
- تابع XOR: نمایش دودویی (۶ از ۶)
  - وزن‌های نهایی (بعد از ۵۰۰ تکرار)



پرسپترون چندلایه (مثال) . . .

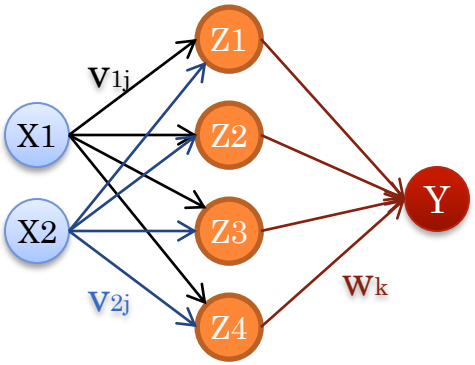


- تابع XOR: نمایش دودویی
  - ساختار ۱-۴-۲ (۲واحد ورودی، ۴واحد مخفی در یک لایهٔ مخفی، ۱واحد خروجی)
  - وزن‌های اولیه تصادفی
    - بایاس‌های چهار واحد مخفی: $-0{,}3329 \quad 0{,}2859 \quad 0{,}2771 \quad -0{,}3378$
    - وزن‌های اولین واحد ورودی به لایه مخفی: $0{,}3594 \quad -0{,}1448 \quad 0{,}3191 \quad 0{,}1970$
    - وزن‌های دومین واحد ورودی به لایه مخفی: $-0{,}4861 \quad -0{,}0347 \quad 0{,}1904 \quad 0{,}3099$
    - وزن‌های (و بایاس) واحدهای مخفی به واحد خروجی: $0{,}3581 \quad -0{,}3979 \quad -0{,}2913 \quad 0{,}4919 \quad -0{,}1401$
  - نرخ یادگیری = ۰.۰۲
  - ادامه آموزش تا زمانی که کل مربعات خطا برای چهار الگوی آموزش کمتر از ۰.۰۵ باشد

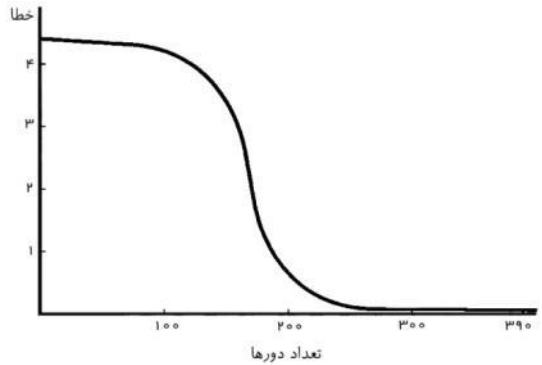


- آموزش نسبتاً آهسته
- تقریباً در ۳۰۰۰ دور

پرسپترون چندلایه (مثال) . . .

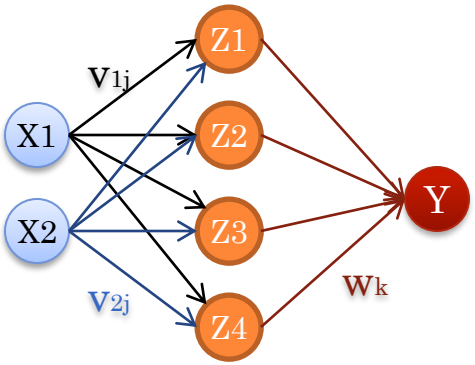


- تابع XOR: نمایش دوقطبی
  - ساختار ۱-۴-۲ (۲واحد ورودی، ۴واحد مخفی در یک لایهٔ مخفی، ۱واحد خروجی)
  - وزن‌های اولیه تصادفی
    - بایاس‌های چهار واحد مخفی: $-0{,}3329 \quad 0{,}2859 \quad 0{,}2771 \quad -0{,}3378$
    - وزن‌های اولین واحد ورودی به لایه مخفی: $0{,}3594 \quad -0{,}1448 \quad 0{,}3191 \quad 0{,}1970$
    - وزن‌های دومین واحد ورودی به لایه مخفی: $-0{,}4861 \quad -0{,}0347 \quad 0{,}1904 \quad 0{,}3099$
    - وزن‌های (و بایاس) واحدهای مخفی به واحد خروجی: $0{,}3581 \quad -0{,}3979 \quad -0{,}2913 \quad 0{,}4919 \quad -0{,}1401$
  - نرخ یادگیری = ۰.۰۲
  - ادامه آموزش تا زمانی که کل مربعات خطا برای چهار الگوی آموزش کمتر از ۰.۰۵ باشد



- آموزش سریع‌تر به نسبت حالت دودویی
- آموزش در ۳۸۷ دور

پرسپترون چندلایه (مثال) . . .

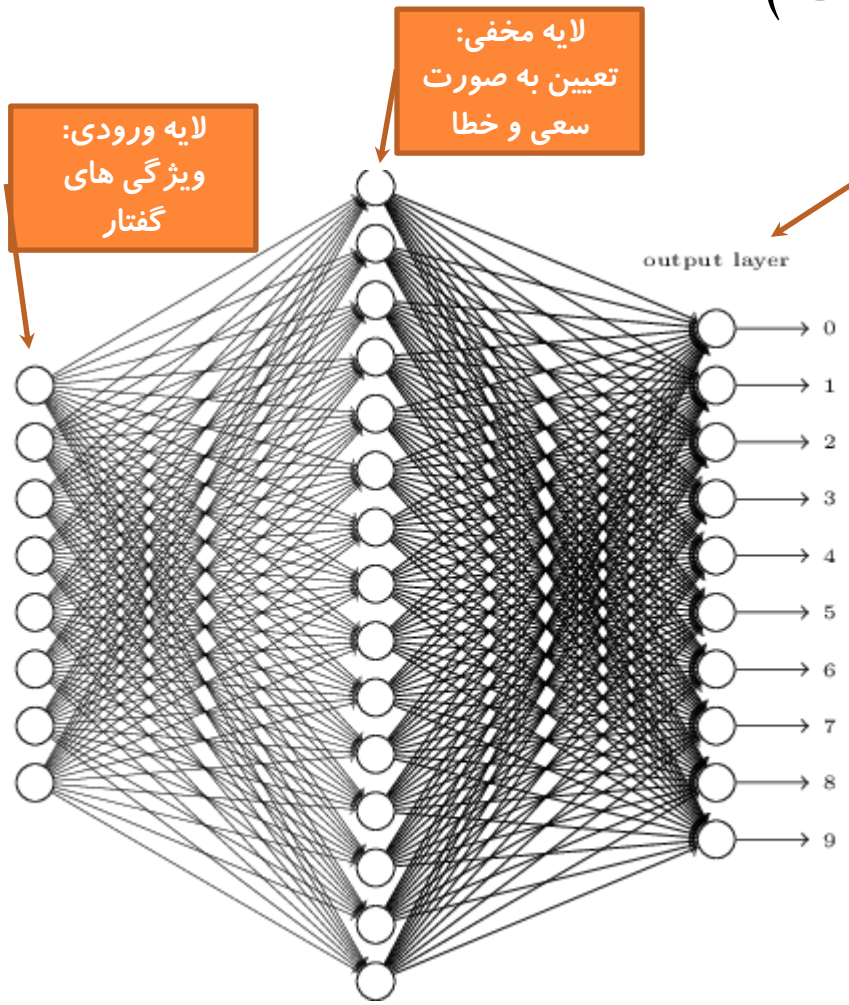


- تابع XOR: نمایش دوقطبی تغییر داده شده
  - ساختار ۱-۴-۲ (۲واحد ورودی، ۴واحد مخفی در یک لایهٔ مخفی، ۱واحد خروجی)
  - وزن‌های اولیه تصادفی
    - بایاس‌های چهار واحد مخفی: $-0{,}3329 \quad 0{,}2859 \quad 0{,}2771 \quad -0{,}3378$
    - وزن‌های اولین واحد ورودی به لایه مخفی: $0{,}3594 \quad -0{,}1448 \quad 0{,}3191 \quad 0{,}1970$
    - وزن‌های دومین واحد ورودی به لایه مخفی: $-0{,}4861 \quad -0{,}0347 \quad 0{,}1904 \quad 0{,}3099$
    - وزن‌های (و بایاس) واحدهای مخفی به واحد خروجی: $0{,}3581 \quad -0{,}3979 \quad -0{,}2913 \quad 0{,}4919 \quad -0{,}1401$
  - نرخ یادگیری = ۰.۰۲
  - ادامه آموزش تا زمانی که کل مربعات خطا برای چهار الگوی آموزش کمتر از ۰.۰۵ باشد

- ایده: اگر مقادیر هدف در مجانب قرار نداشته باشند، همگرایی بهبود می‌یابد
- مقادیر هدف‌های بین ۰.۸ و ۰.۸-
- آموزش در ۲۶۴ دور

پرسپترون چندلایه (مثال) . . .

- تشخیص گفتار (حالت کلمات گسسته)

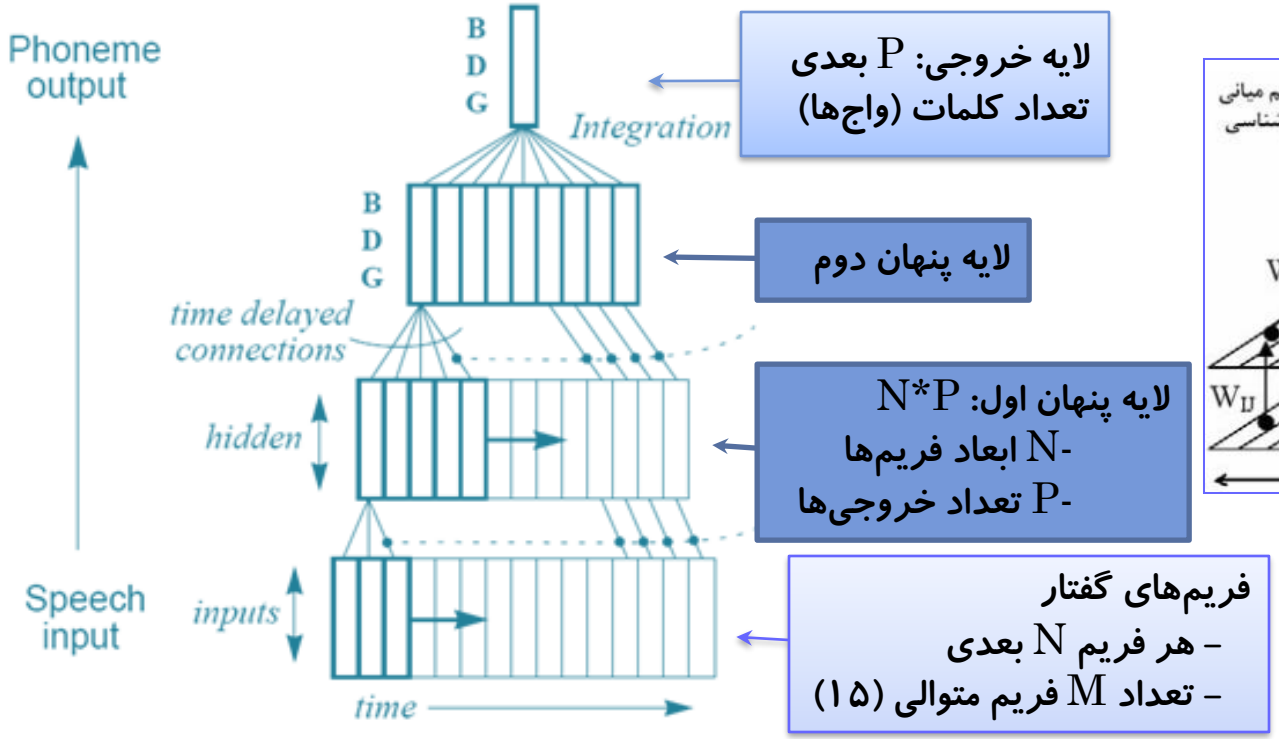


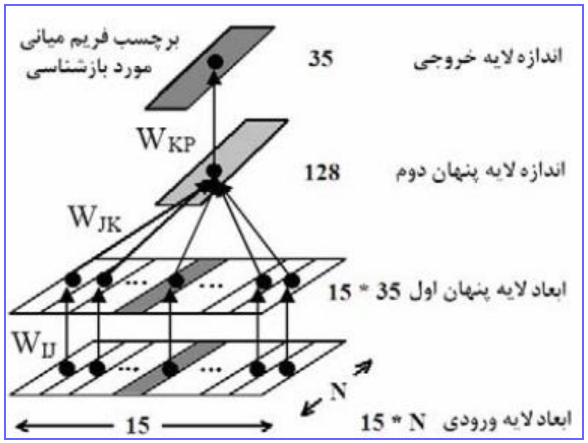
- فرض: طول ورودی ثابت است
  - فرض غیرواقعی برای گفتار!
  - راه حل
    - فریم بندی گفتار
    - دادن تعداد ثابتی فریم به ورودی (۵ فریم)

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)





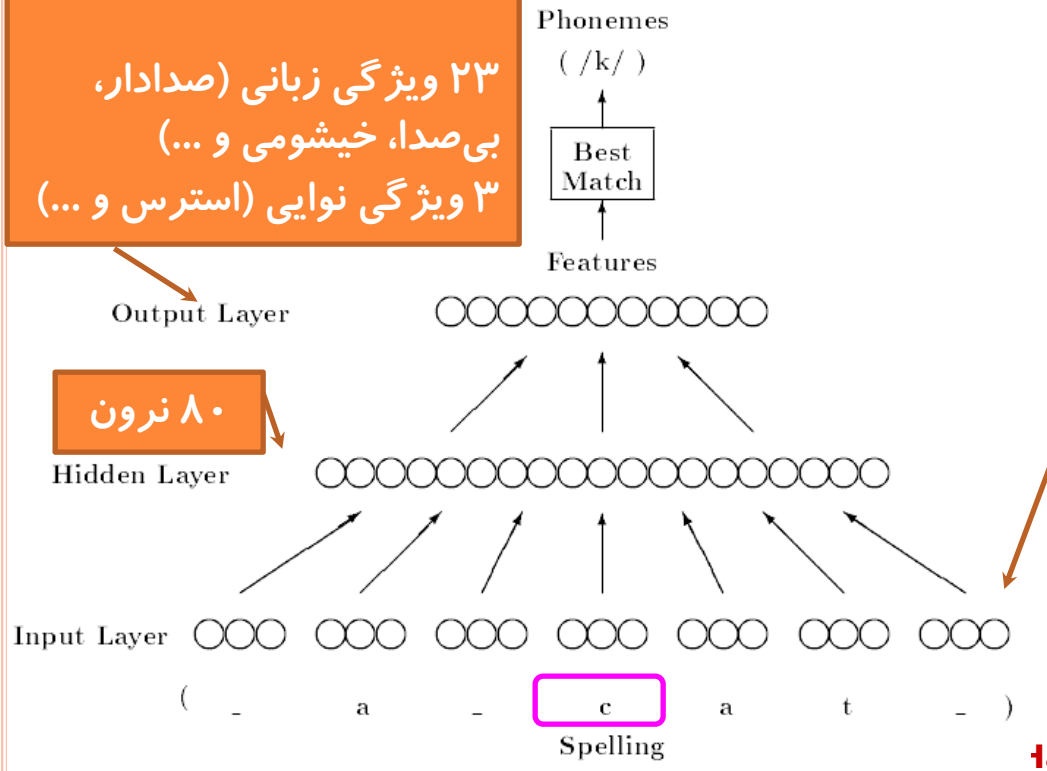
تشخیص واج‌های فارسی

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 ...)

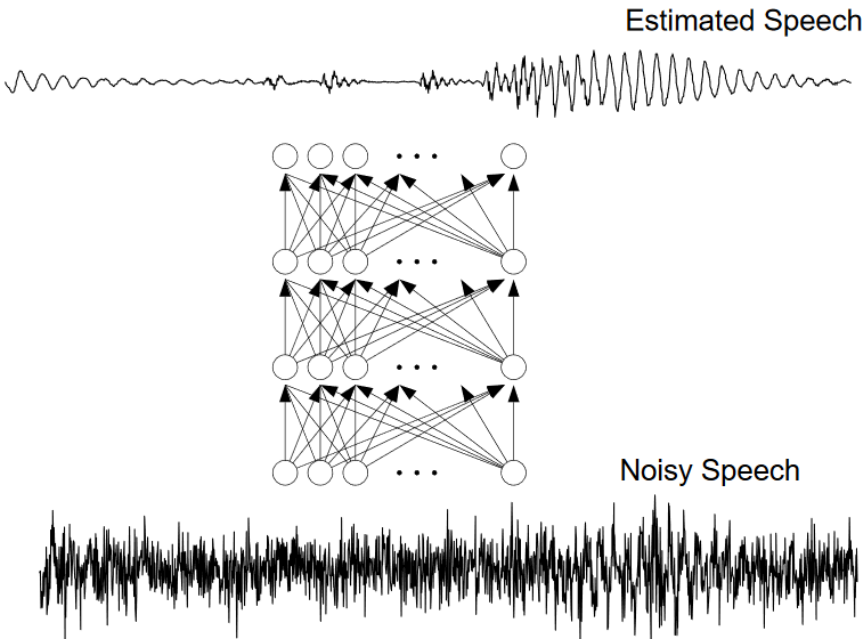


۷*۲۹=۲۰۳ نرون ورودی

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

پرسپترون چندلایه (مثال) . . .

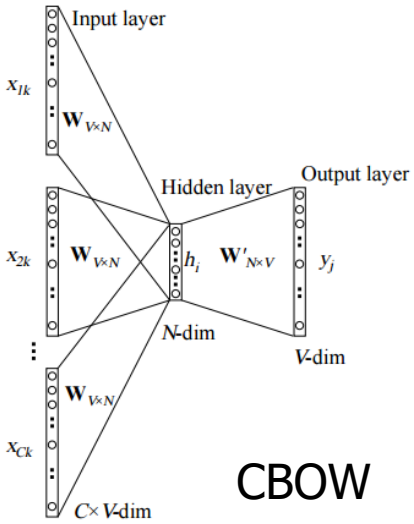
- بهسازی گفتار
  - ورودی: سیگنال نویزی (یا ویژگی های آن)
  - خروجی: سیگنال تمیز شده



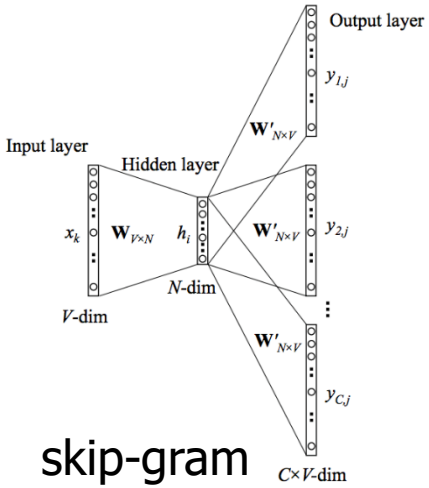
- بهسازی بیشتر با بیش از یک لایه مخفی

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

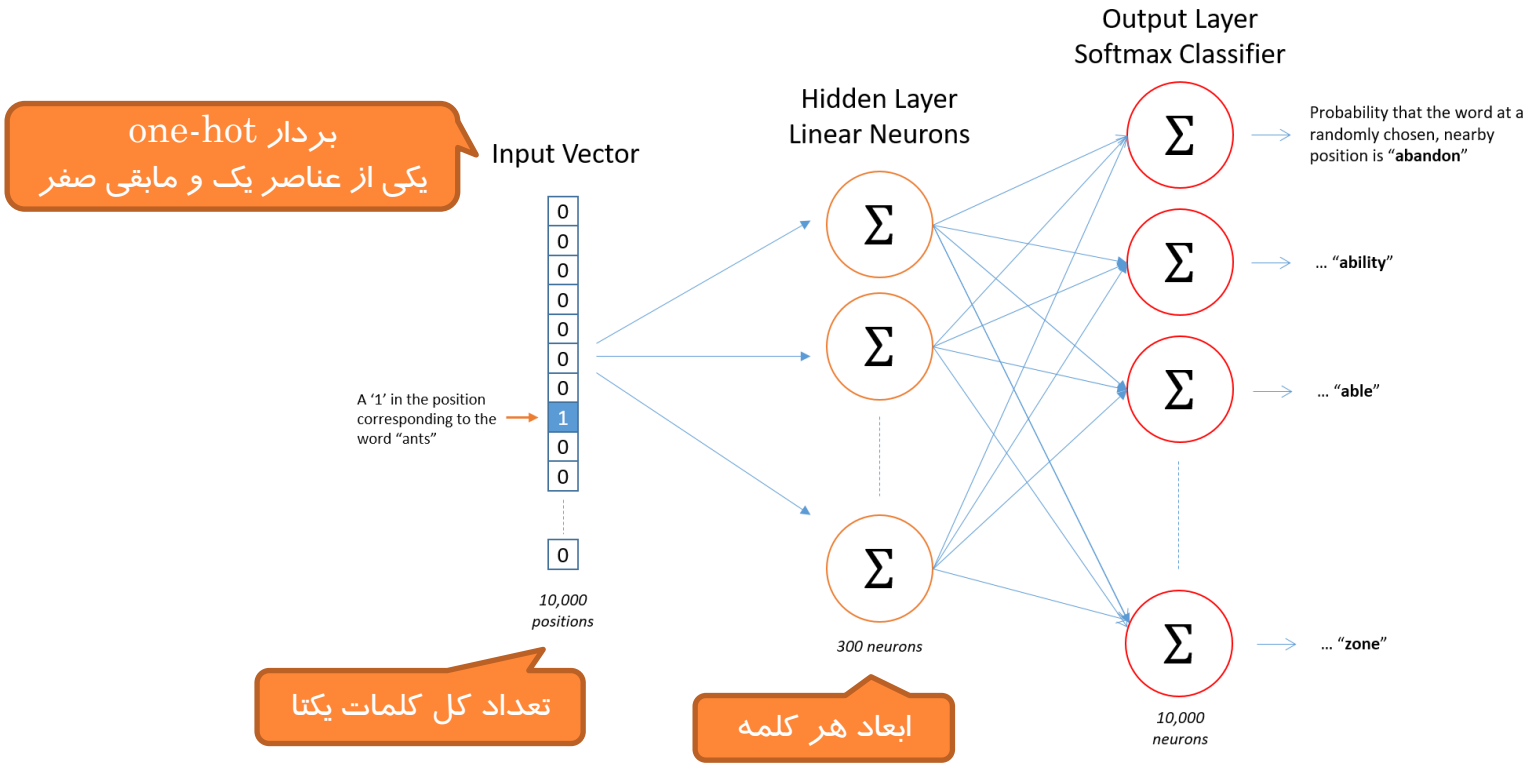


skip-gram

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In *Advances in neural information processing systems*, pp. 3111-3119. 2013.

پرسپترون چندلایه (مثال) . . .

- بردار کلمات: پرش چندتایی



پرسپترون چندلایه (مثال) . . .

بردار کلمات: پرش چندتایی

- داده ورودی

Source Text

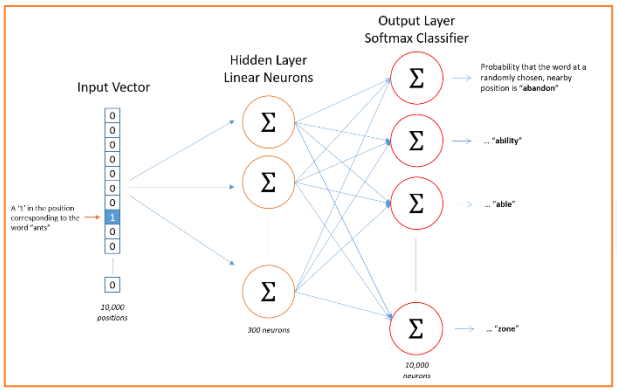
The quick brown fox jumps over the lazy dog.

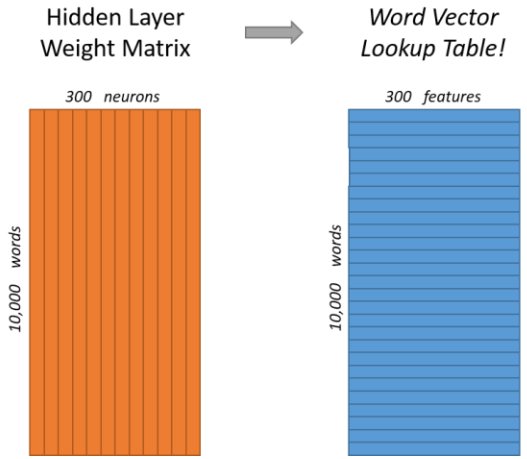
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

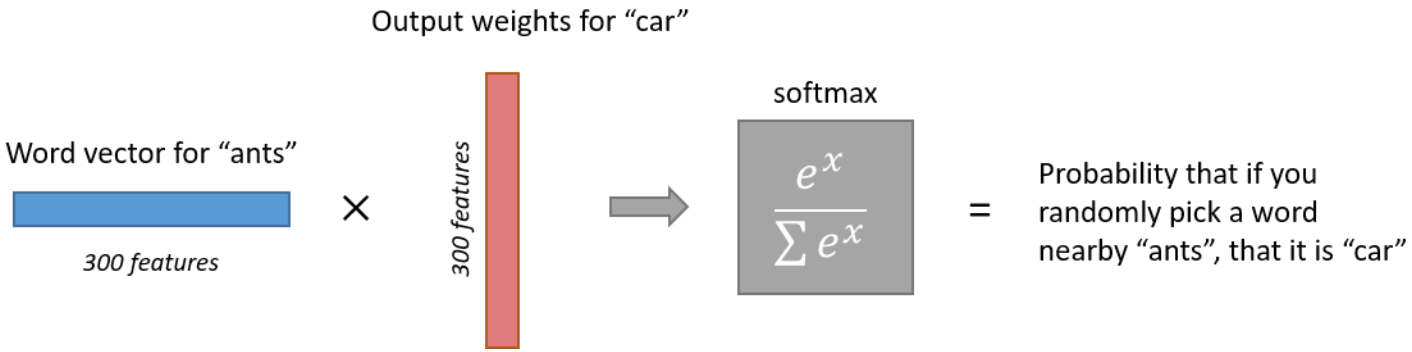
- بردار کلمات نهایی





پرسپترون چندلایه (مثال) . . .

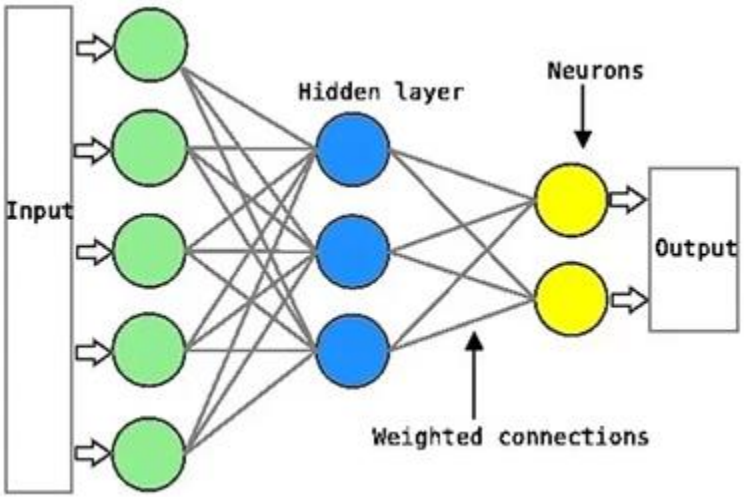
- بردار کلمات: پرش چندتایی
  - لایه خروجی
    - تابع فعال سازی SoftMax: تولید خروجی بین صفر و یک و جمع همه خروجی ها برابر با یک
    - به ازای هر کلمه یک نرون



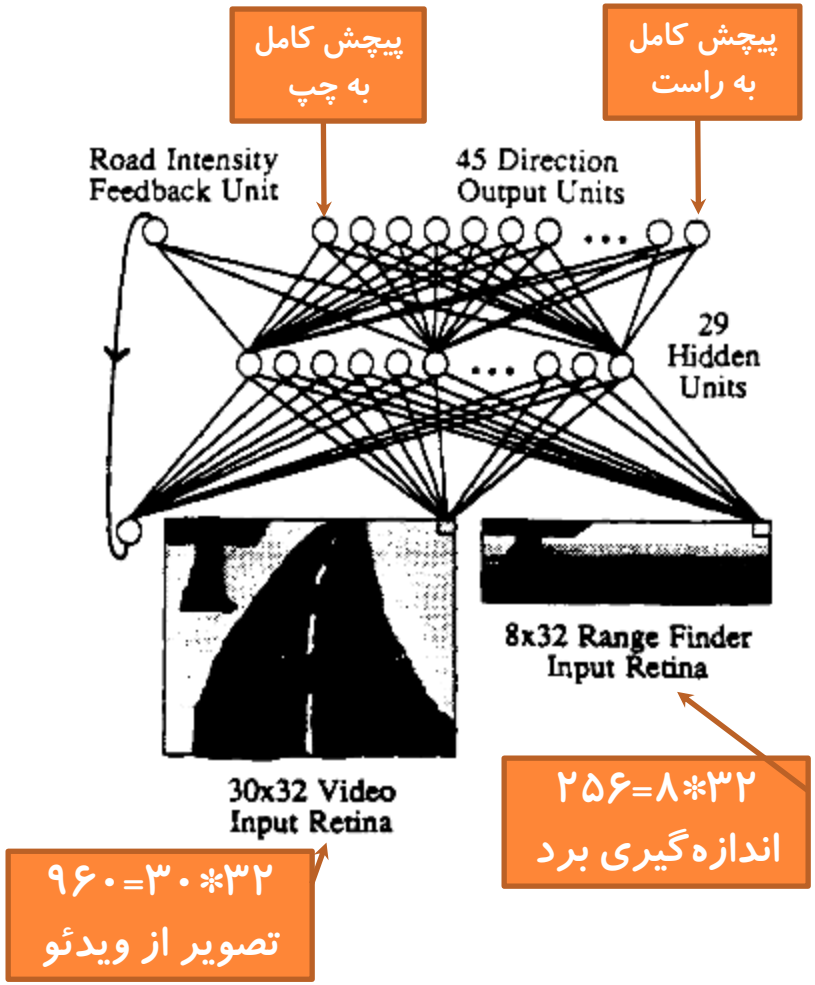
پرسپترون چندلایه (مثال) . . .

- 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
  - ...



شبکه‌های پس‌انتشار: کاربردها . . .

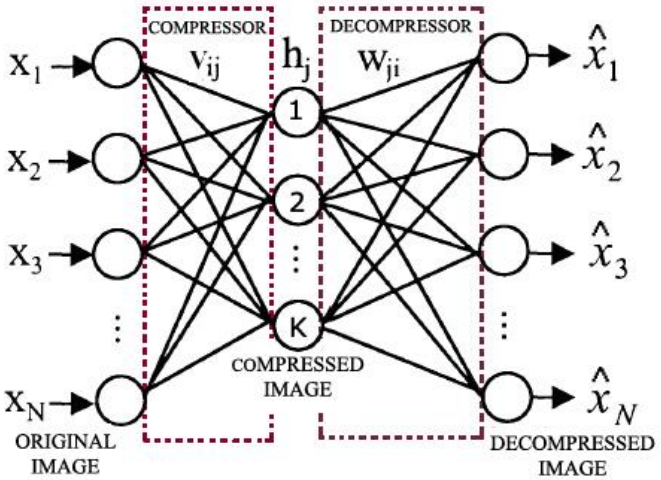


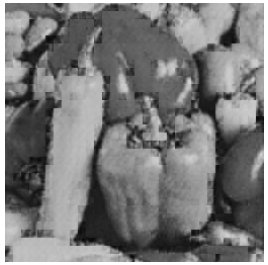
- هدایت خودکار خودرو (ALVINN) . . .
  - ALVINN(Autonomous Land Vehicle in a Neural Network)
- رانندگی خودکار خودرو در جاده مارپیچ
- ورودی: ۲۵۶+۹۶۰+۱ = ۱۲۱۷
- نرون‌های مخفی: ۲۹
- نرون‌های خروجی: ۴۵ (جهت فرمان)
- آموزش: ۱۲۰۰ تصویر از جاده‌های مختلف
- آموزش در ۴۰ تکرار
- دقت ۹۰٪
- رانندگی با سرعت ۵ کیلومتر بر ساعت!
  - دو برابر سرعت روش‌های دیگر!!

Dean A.Pomerleau, "ALVINN: AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK", Technical Report, AIP-77 (15213-3890), Department of Psychology Carnegie Mellon University, January 1989

پرسپترون چندلایه (مثال) . . .

- فشرده‌سازی داده‌ها
  - فشرده‌سازی با ضرر تصویر
  - شکستن تصویر به بلوک‌های کوچک (مثلا ۸*۸=۶۴ نرون ورودی و خروجی)
  - مقدار ورودی و خروجی (خروجی تابع فعال‌سازی) پیوسته هستند





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

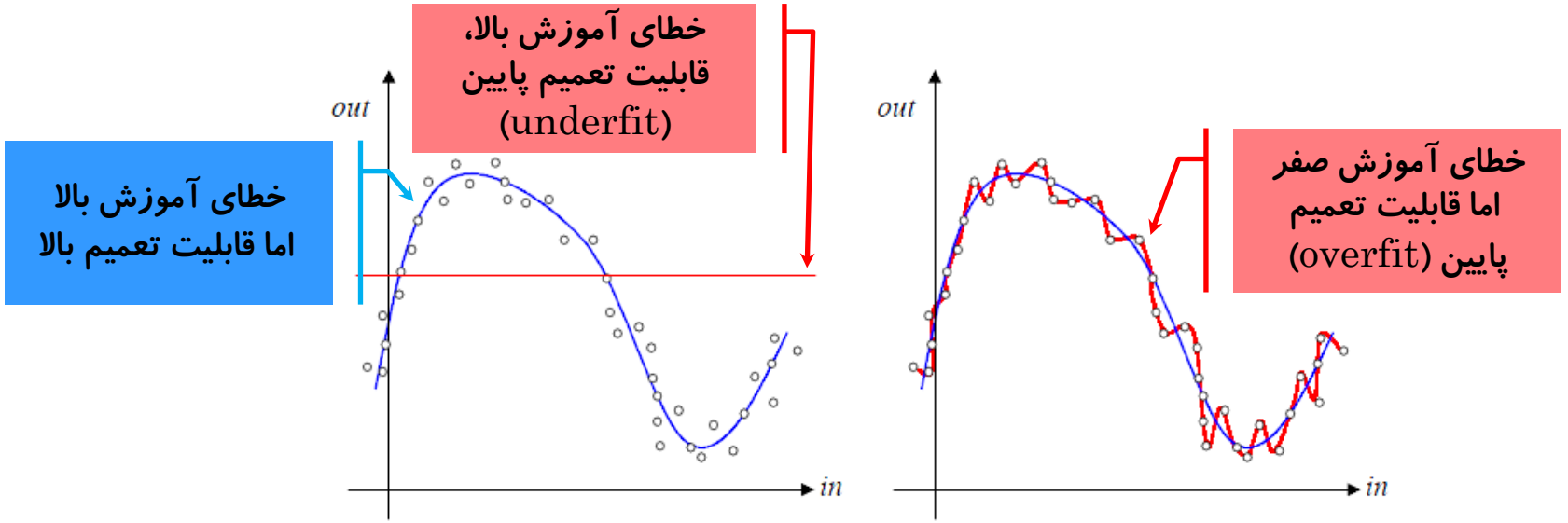
پرسپترون چندلایه: نکات تکمیلی . . .

- **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.

پرسپترون چندلایه: نکات تکمیلی . . .

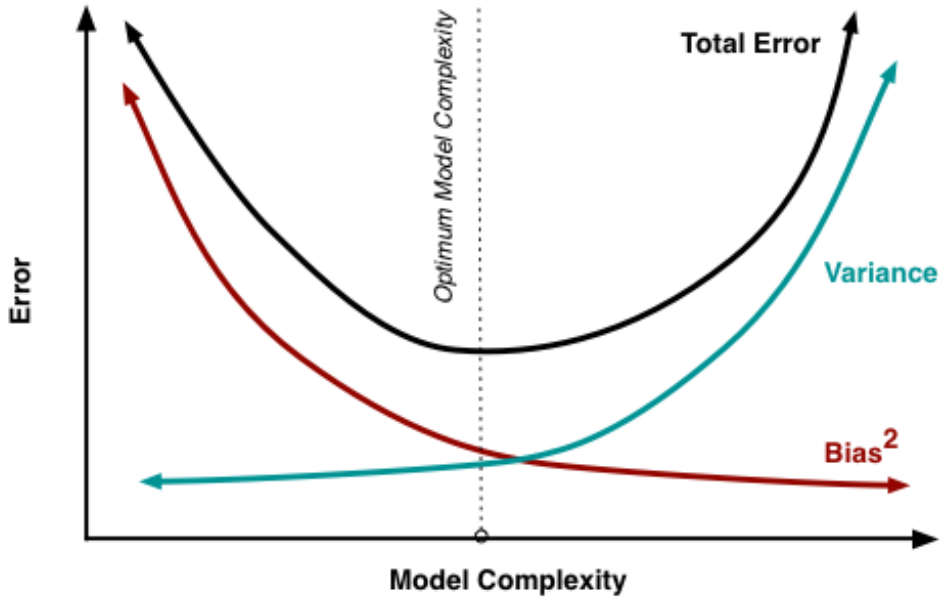
تعادل بین یادگیری الگوها و تعمیم

- 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



پرسپترون چندلایه: نکات تکمیلی . . .

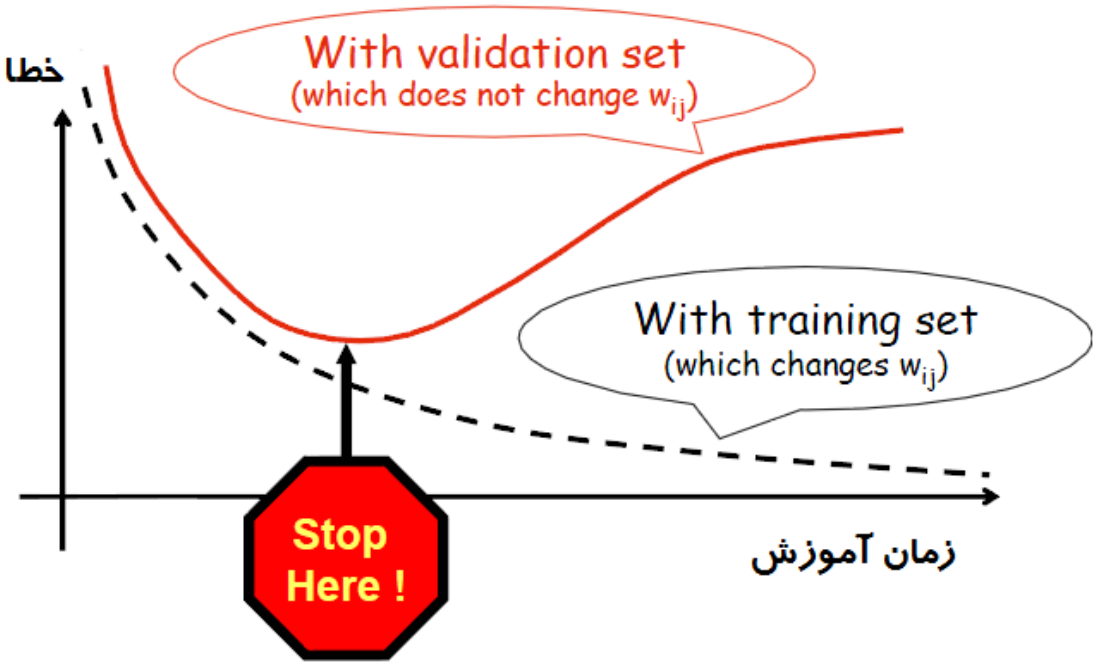
- تعادل بین یادگیری الگوها و تعمیم= جلوگیری از بیش برازش: راه حل ۱
  - پیچیده کردن مدل = بیش برازش = بایاس کمتر = واریانس بیشتر



- تعداد نرون‌های کمتر در لایه مخفی

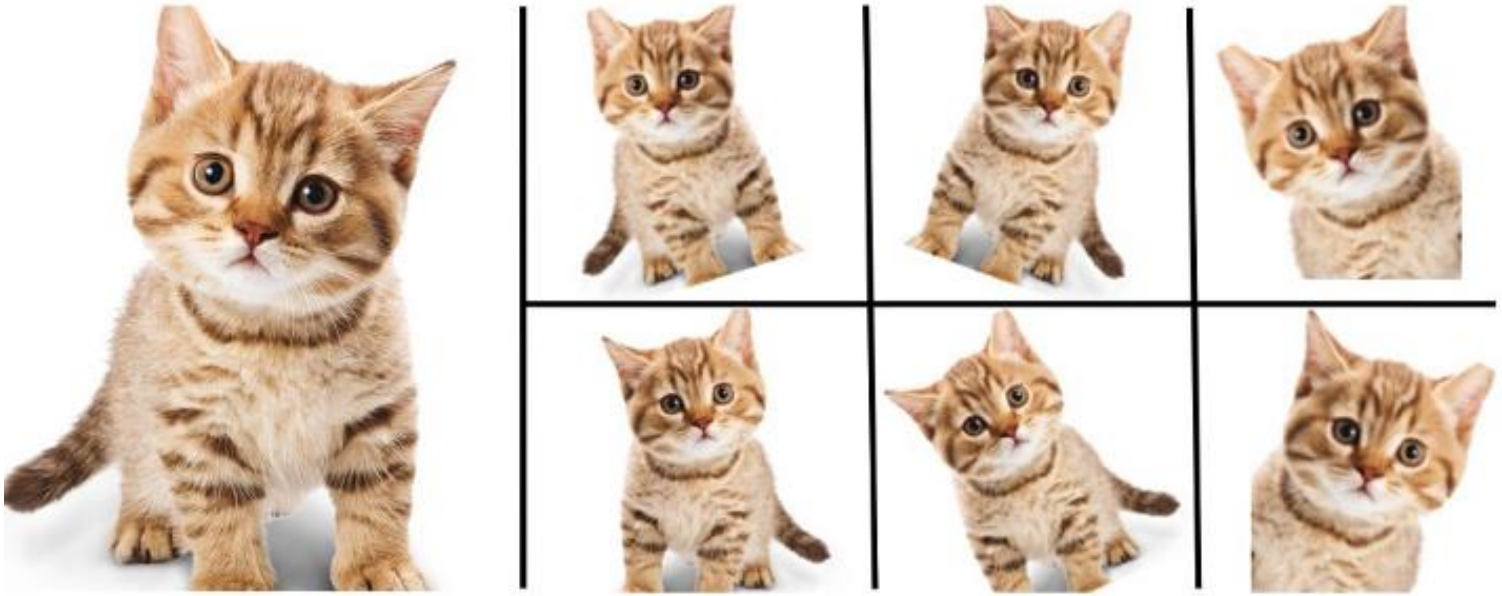
پرسپترون چندلایه: نکات تکمیلی . . .

- 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



پرسپترون چندلایه: نکات تکمیلی . . .

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



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.

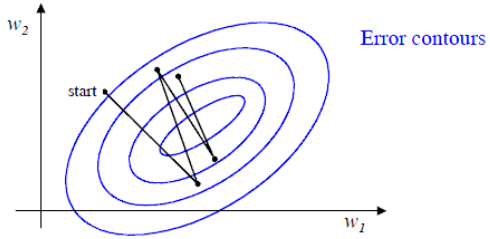
پرسپترون چندلایه: نکات تکمیلی . . .

- 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

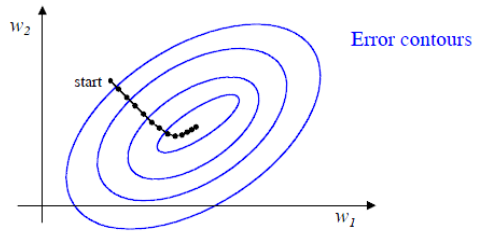
  - آسان‌تر کردن تصحیح وزن‌ها
  - مقاوم بودن در برابر داده‌های نویزی
  - موازی‌سازی محاسبات
  - افزایش احتمال نزدیک شدن به کمینهٔ محلی

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)

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)

گرادیان فعلی

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

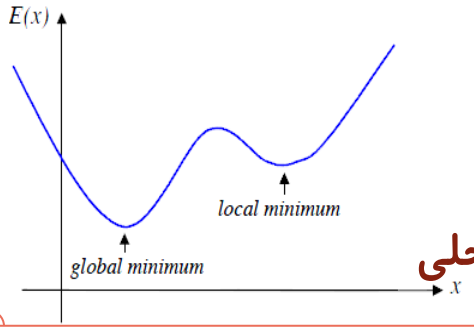
$$w_{jk}(t+1) = w_{jk}(t) + \alpha \delta_k z_j + \mu [w_{jk}(t) - w_{jk}(t-1)]$$

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

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

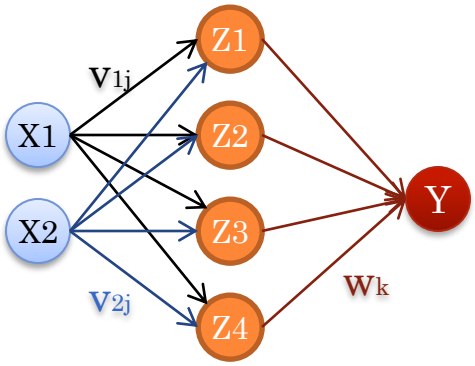
گرادیان قبلی

پارامتر ممان (بین ۰ تا ۱)



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

پرسپترون چندلایه: نکات تکمیلی . . .



- پس‌انتشار با گشتاور(Momentum): مثال
  - تابع XOR: نمایش دوقطبی
    - ساختار ۱-۴-۲
    - وزن‌های اولیه تصادفی
      - بایاس‌های چهار واحد مخفی: $-0{,}3378 \quad 0{,}2771 \quad 0{,}2859 \quad -0{,}3329$
      - وزن‌های اولین واحد ورودی به لایه مخفی: $0{,}1970 \quad 0{,}3191 \quad -0{,}1448 \quad 0{,}3594$
      - وزن‌های دومین واحد ورودی به لایه مخفی: $0{,}3099 \quad 0{,}1904 \quad -0{,}0347 \quad -0{,}4861$
      - وزن‌های (و بایاس) واحدهای مخفی به واحد خروجی: $-0{,}1401 \quad 0{,}4919 \quad -0{,}2913 \quad -0{,}3979 \quad 0{,}3581$
    - نرخ یادگیری = ۰.۰۲
    - ادامه آموزش تا زمانی که کل مربعات خطا برای چهار الگوی آموزش کمتر از ۰.۰۵ باشد
  - استفاده از ممان = افزایش سرعت یادگیری
    - پارامتر ممان = ۰.۹
    - آموزش از ۳۸۷ دور به ۳۸ دور کاهش می‌یابد

پرسپترون چندلایه: نکات تکمیلی

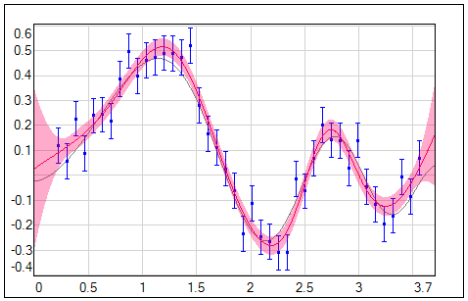
- شبکه MLP تقریب‌زننده‌های جهانی است: قضیهٔ هچ-نیلسون
  - هر تابع پیوستهٔ $f : I^n \to R^m$ را که در آن I بازهٔ بستهٔ $[0,1]$ است، می‌توان دقیقاً با یک شبکهٔ عصبی پیش‌خور با n واحد ورودی، $2n+1$ واحد مخفی و m واحد خروجی نمایش داد.
  - تابع فعال‌سازی برای واحد مخفی jام: $z_j = \sum_{i=1}^{n} \lambda^i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

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

عددی ثابت و حقیقی

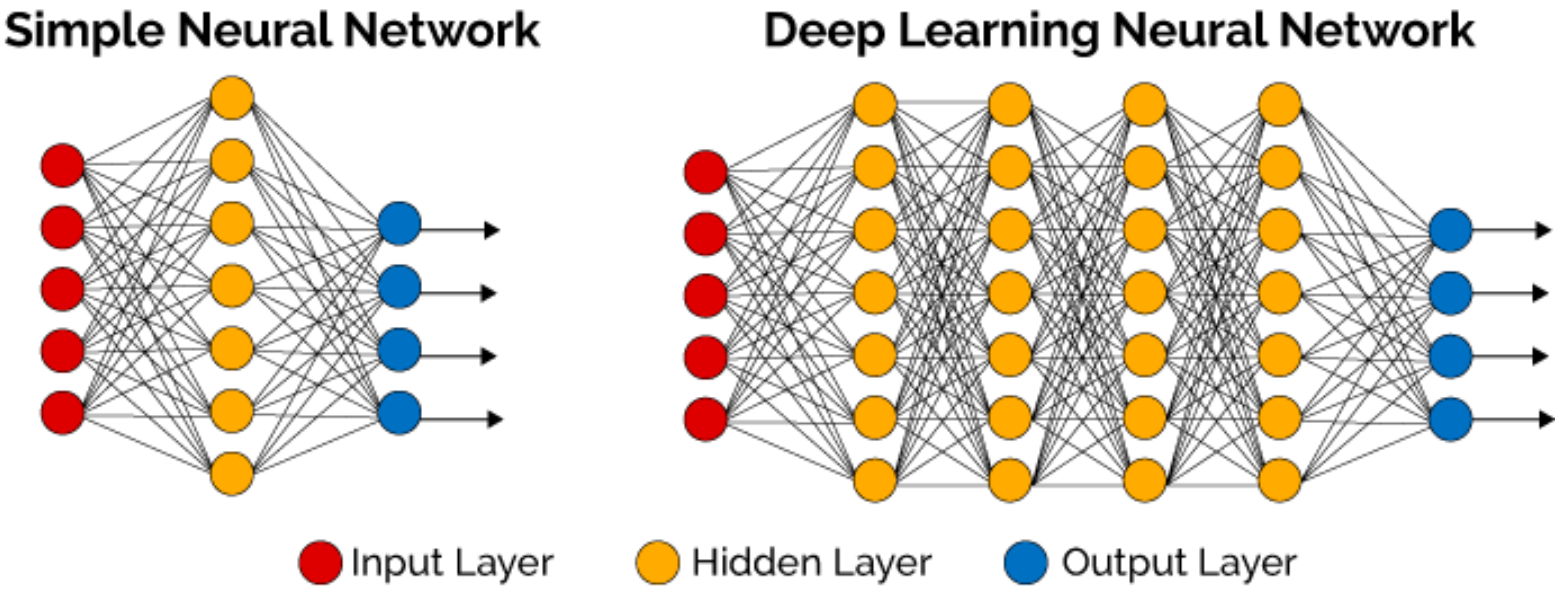
  - مقدار ثابت ε برای فراهم بودن شرایط قضیهٔ اسپرچر است.
  - تابع فعال‌سازی برای واحدهای خروجی: $y_k = \sum_{j=1}^{2n+1} g_k z_j$

تابع پیوسته، حقیقی و وابسته به f و ε



یادگیری عمیق . . .

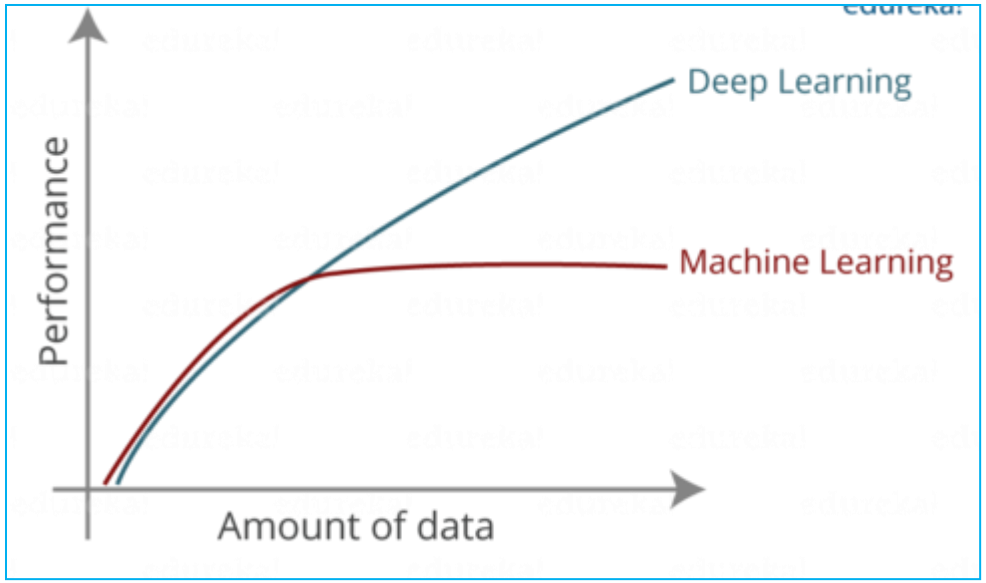
- افزایش تعداد لایه های مخفی

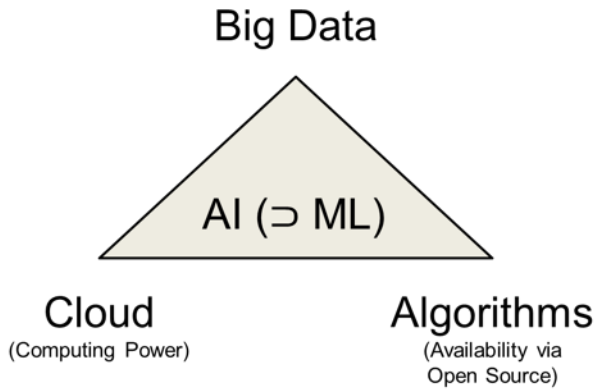


- مدل پیچیده = تعداد پارامترهای زیاد = توان محاسباتی بالا

یادگیری عمیق . . .

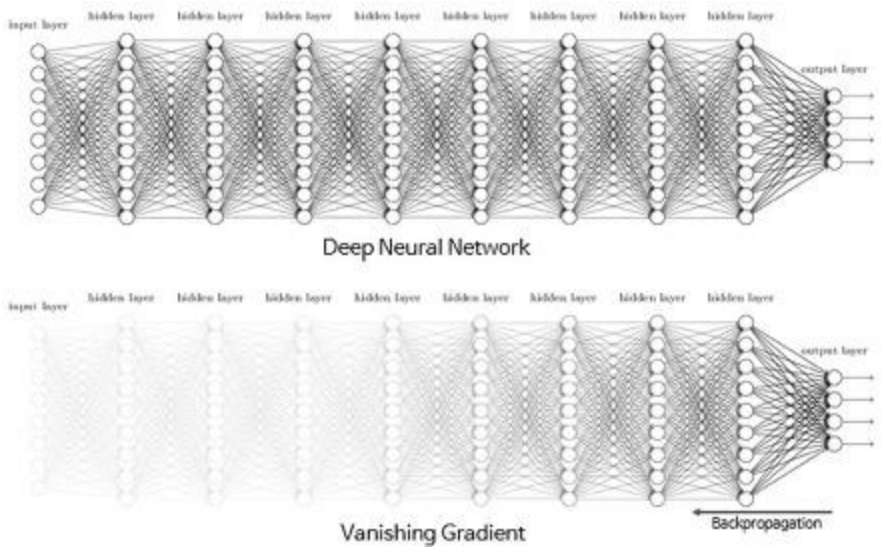
- سه رکن اصلی
  - داده حجیم (Big Data)
  - توان پردازشی (GPU)
  - الگوریتم یادگیری





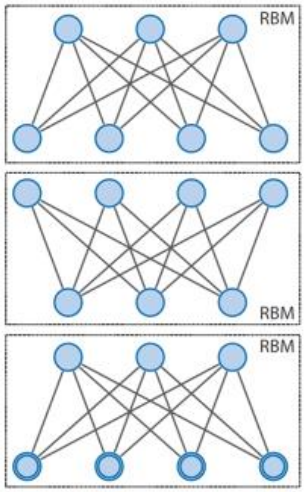
یادگیری عمیق . . .

- 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

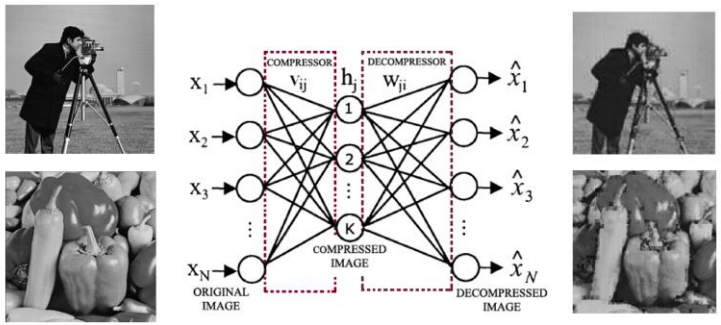
شبکه باور عمیق (DBN): الگوریتم آموزش و جزئیات

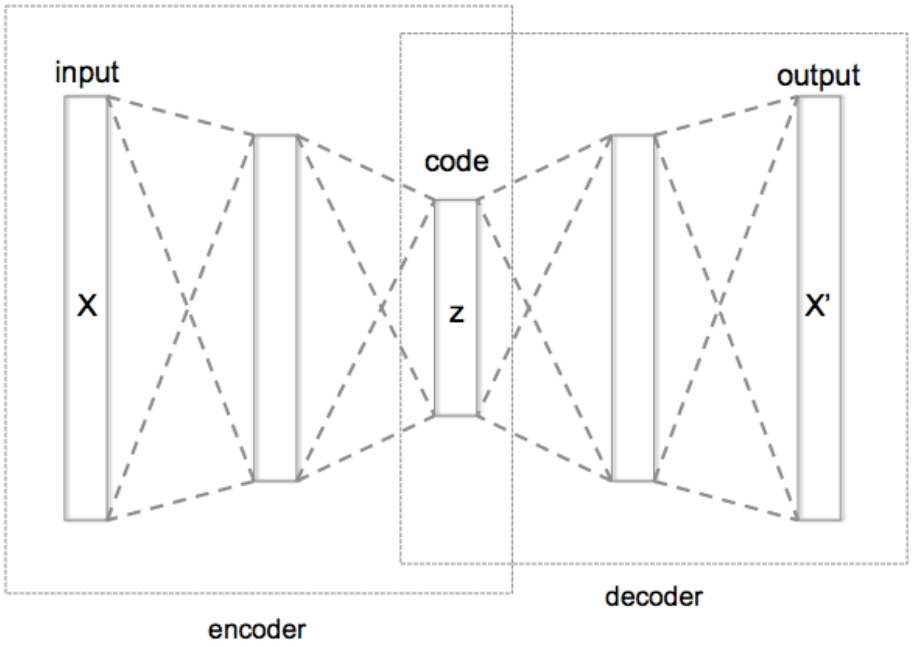
- مراجعه کنید به

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

شبکه های خودرمزگذار (AE)

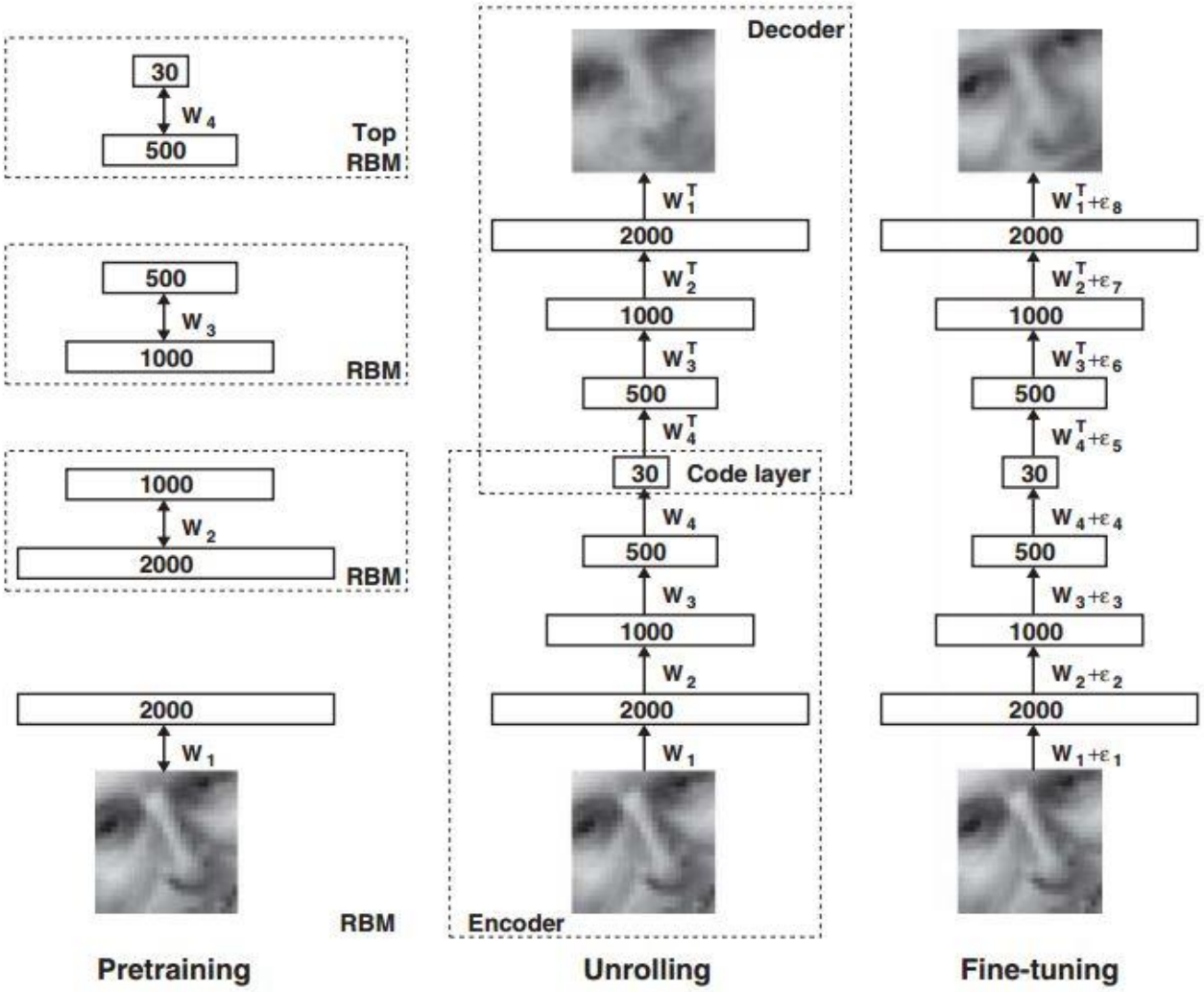
- شبکه های خودرمزگذار (AE: AutoEncoders)
  - استفاده از لایه وسط به عنوان ویژگی



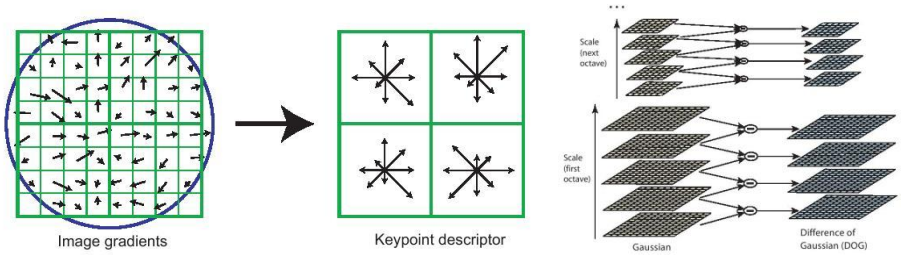


یادگیری عمیق . . .

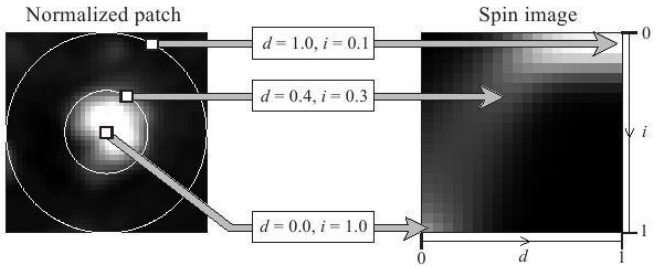
- خودرمزگذار



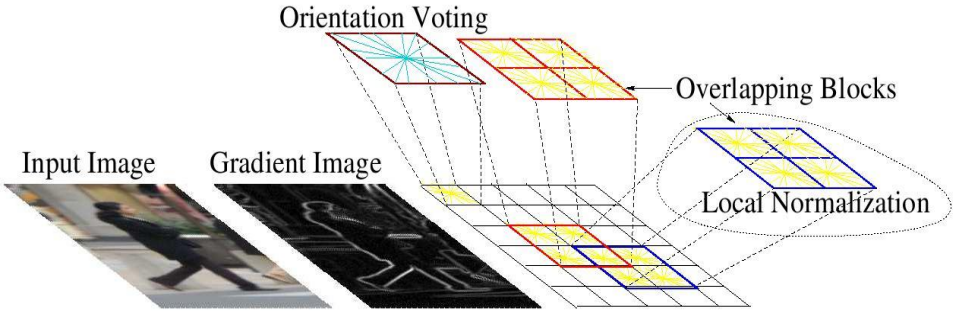
یادگیری عمیق: استخراج ویژگی (تصویر)



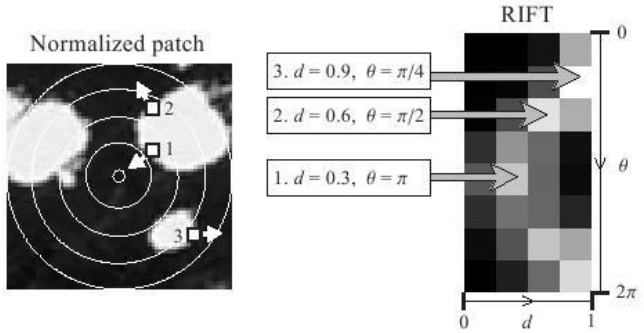
SIFT



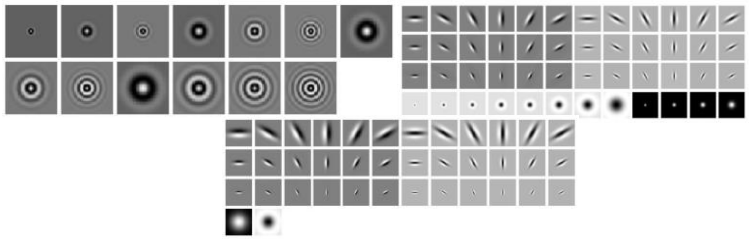
Spin image



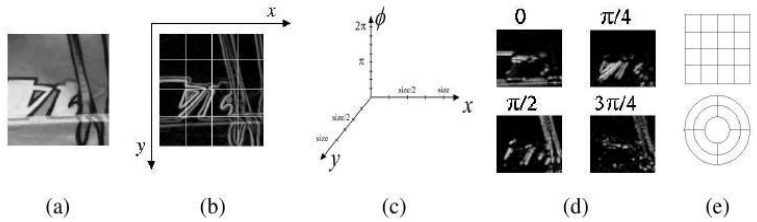
HoG



RIFT

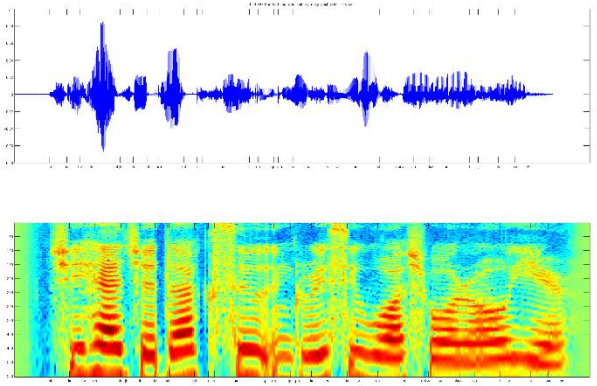


Textons

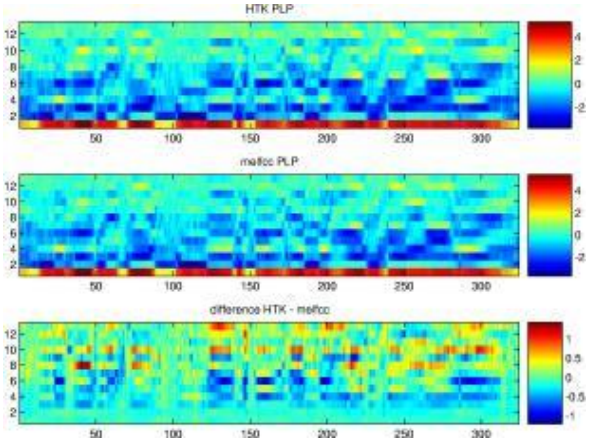


GLOH

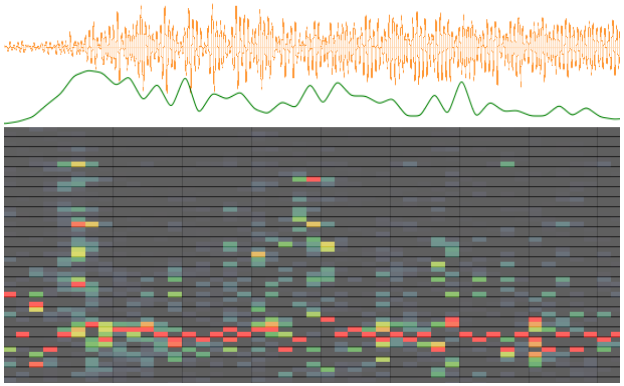
یادگیری عمیق: استخراج ویژگی (صدا)



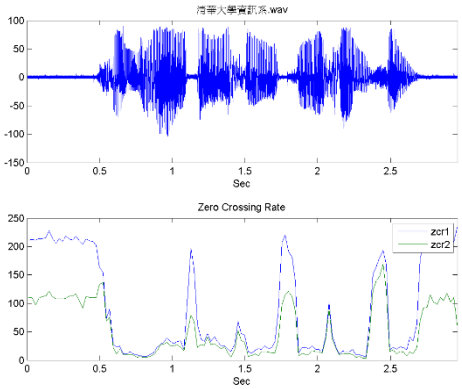
Spectrogram



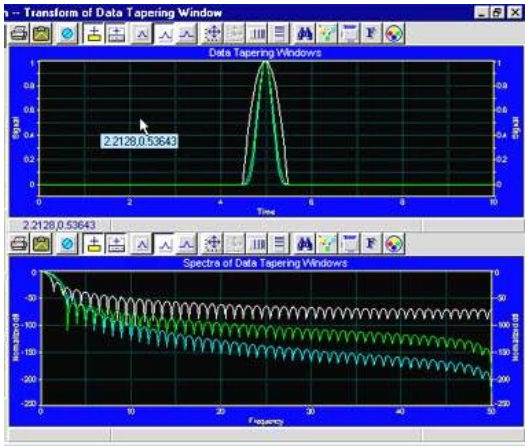
MFCC



Flux



ZCR



Rolloff

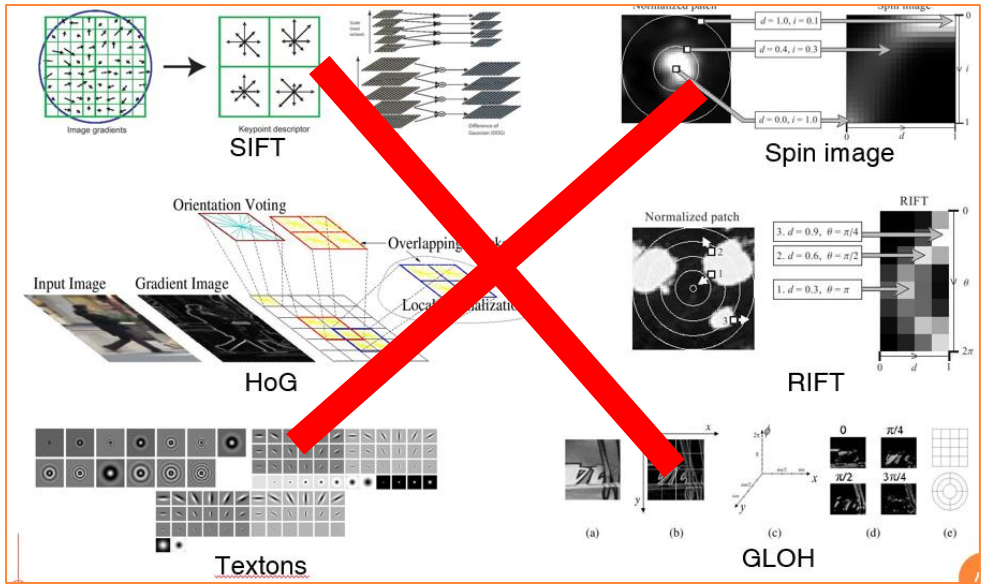
یادگیری عمیق . . .

- ترکیب استخراج ویژگی و دسته‌بندی

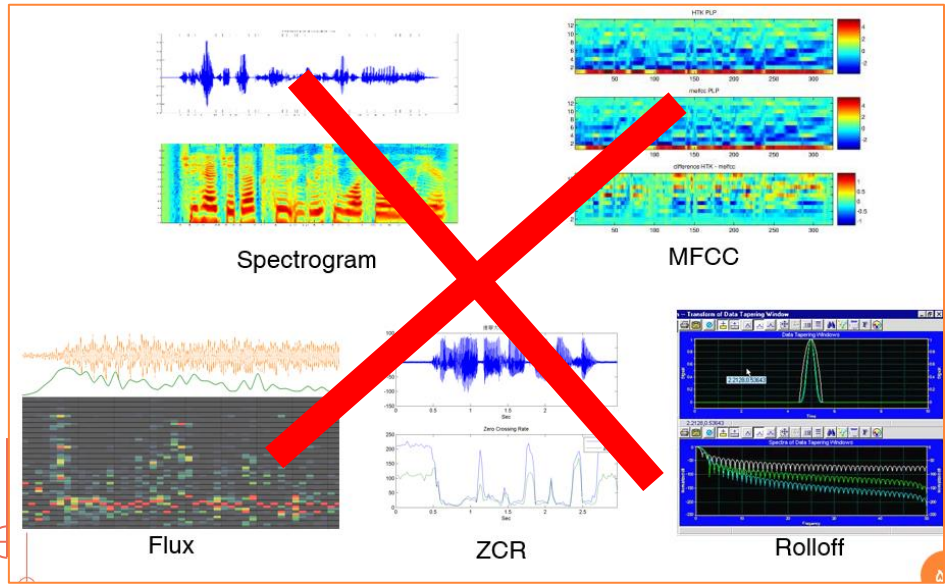
Traditional Machine Learning Flow

Deep Learning Flow

یادگیری عمیق: استخراج ویژگی و دسته‌بندی



Unlabeled images



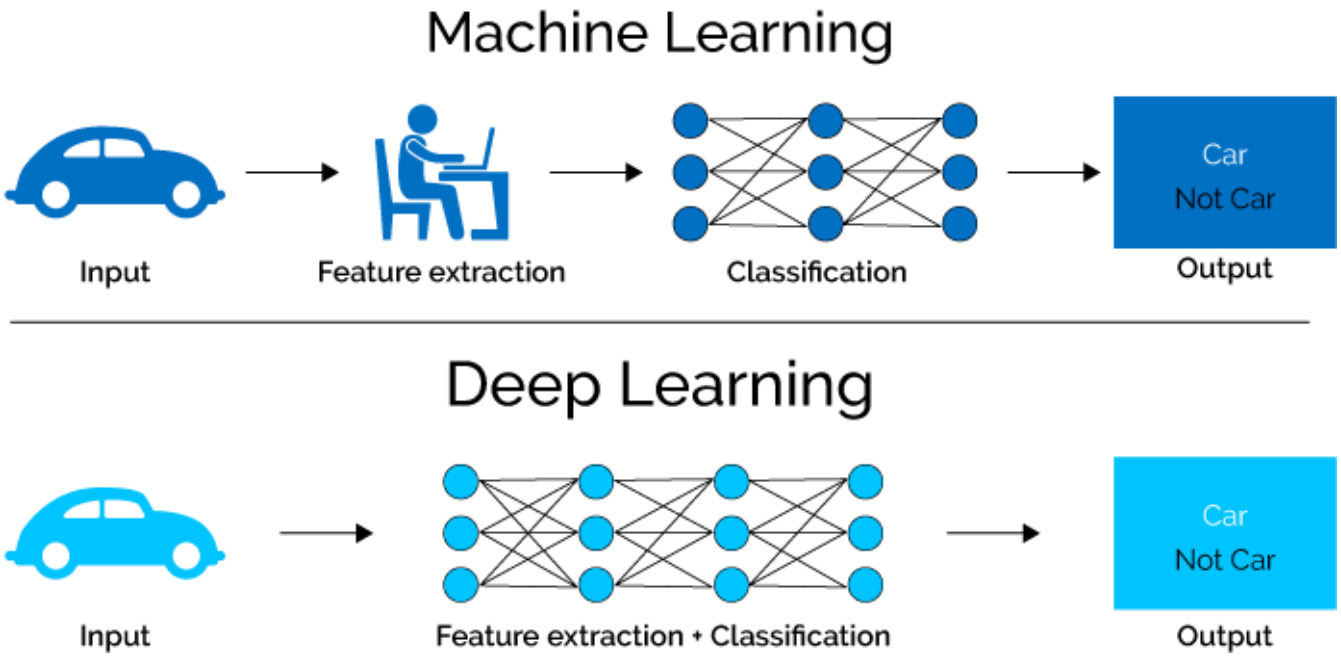
Learning algorithm

Feature representation

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

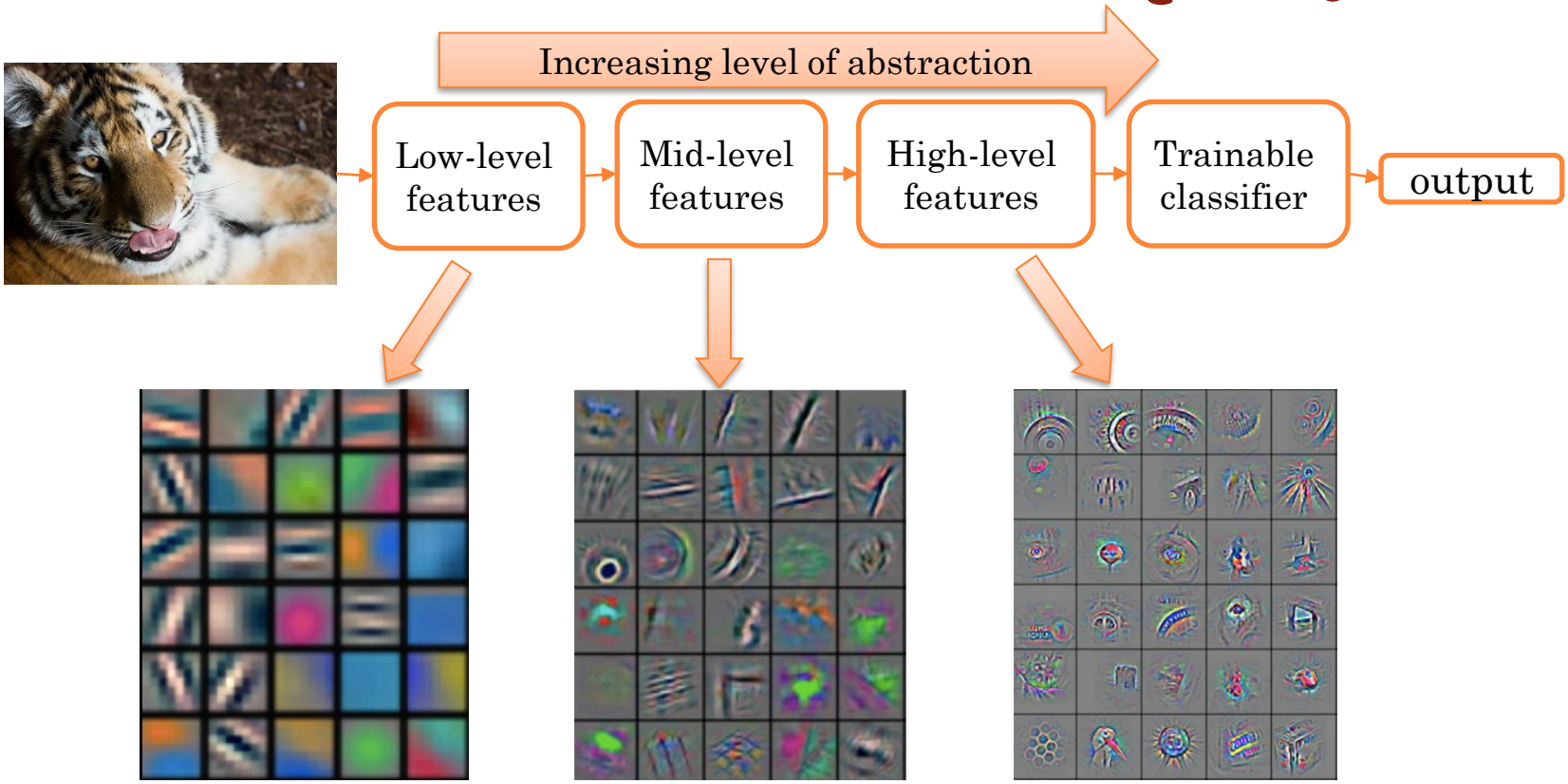
یادگیری عمیق . . .

- ترکیب استخراج ویژگی و دسته‌بندی



یادگیری عمیق: استخراج ویژگی . . .

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



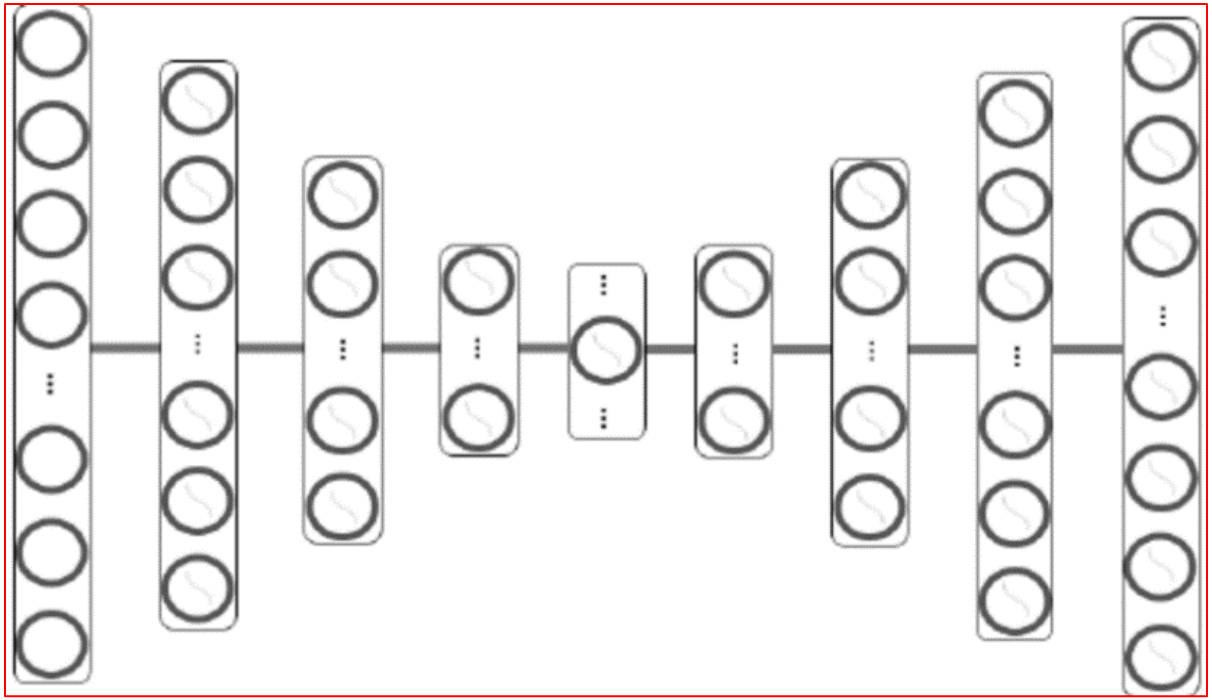
یادگیری عمیق: استخراج ویژگی

- استخراج ویژگی سلسله مراتبی

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

شبکه های خودرمزگذار (AE)

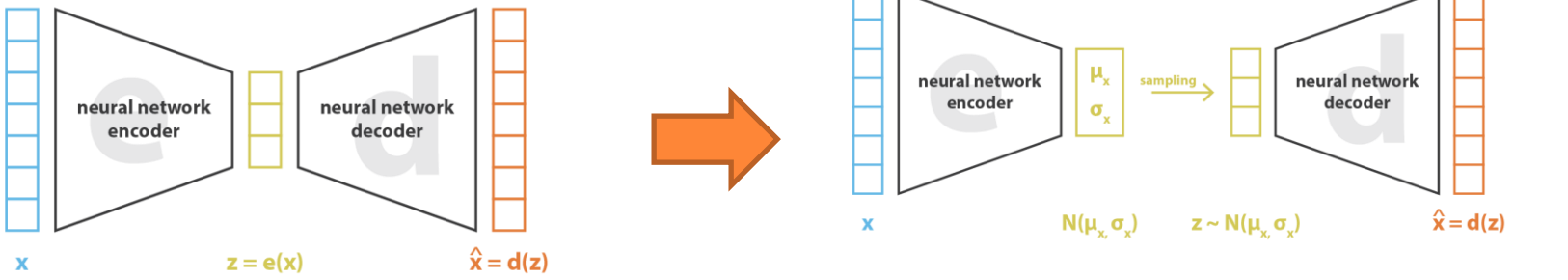
- شبکه های خودرمزگذار (AE: AutoEncoders)
  - استفاده از لایه وسط به عنوان ویژگی



شبکه های خودرمزگذار متغیر (VAE) . . .

شبکه Variational Autoencoders

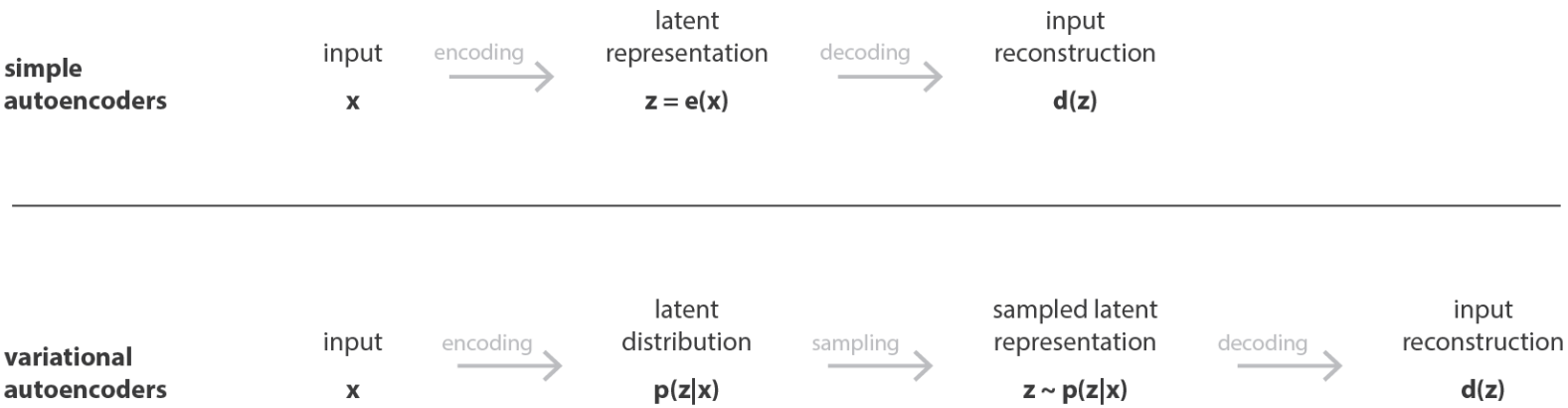
- استفاده از عبارت تنظیم (regularization) برای جلوگیری از بیش برازش و اطمینان از مناسب بودن متغیر پنهان برای تولید داده
- یادگیری تابع توزیع به جای خود داده



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

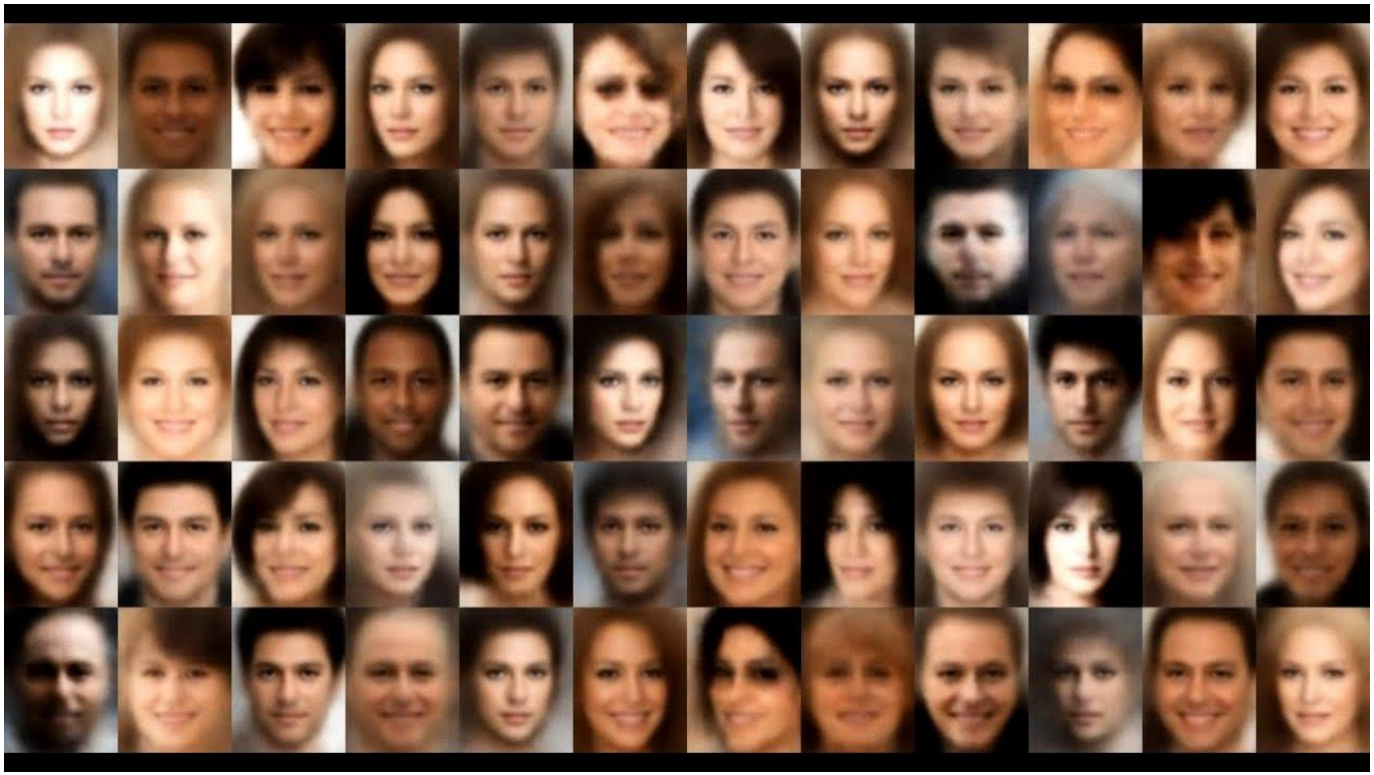
$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

شبکه های خودرمزگذار متغیر (VAE) . . .



شبکه های خودرمزگذار متغیر (VAE)

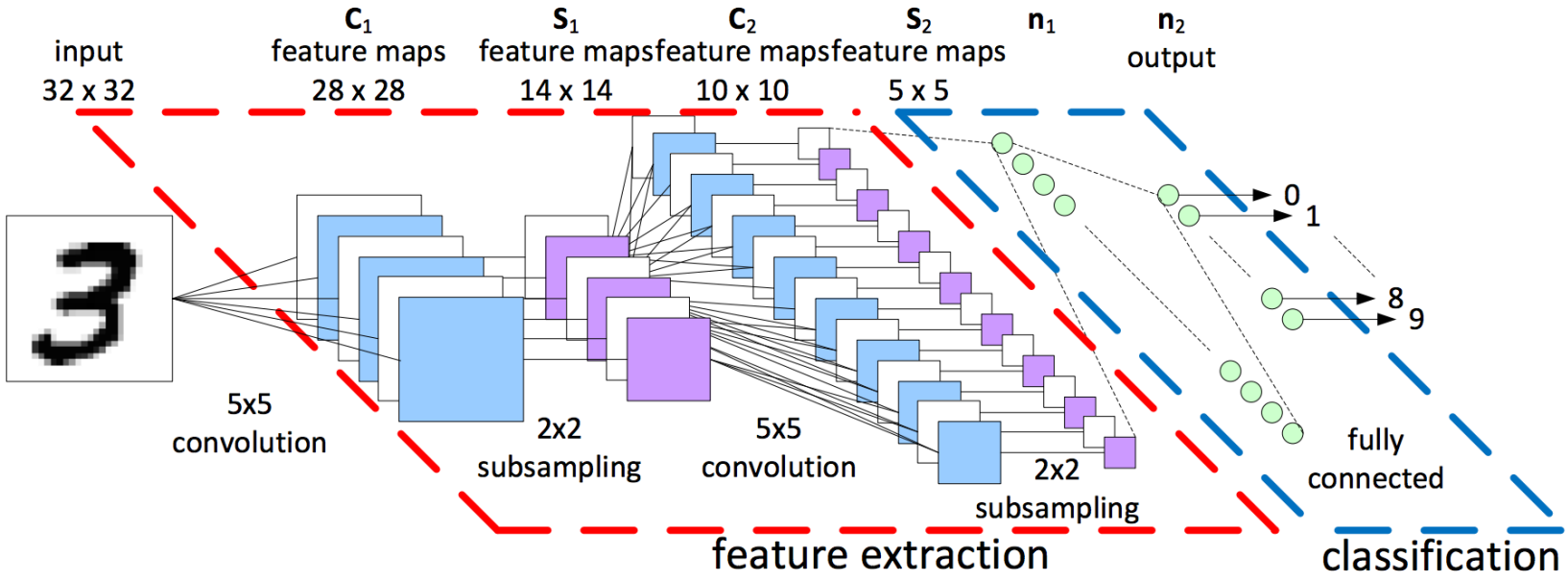
- تولید تصویر



https://github.com/WojciechMormul/vae

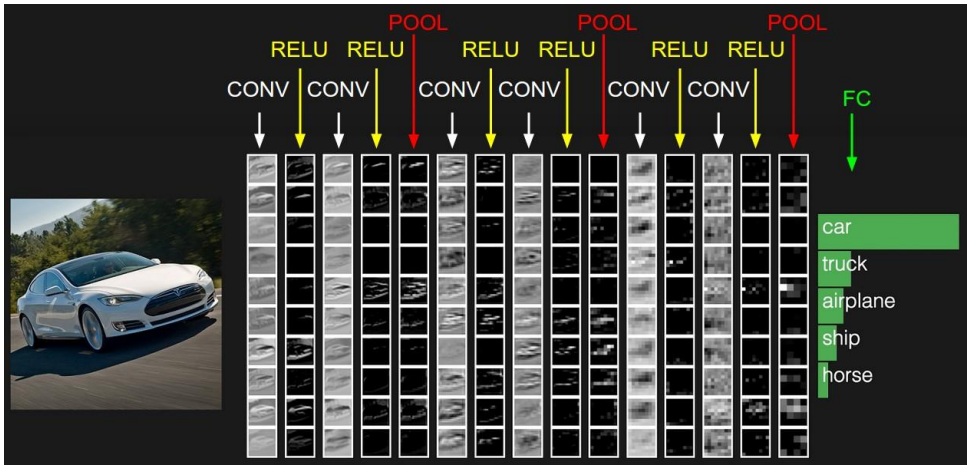
یادگیری عمیق: شبکه عصبی پیچشی . . .

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



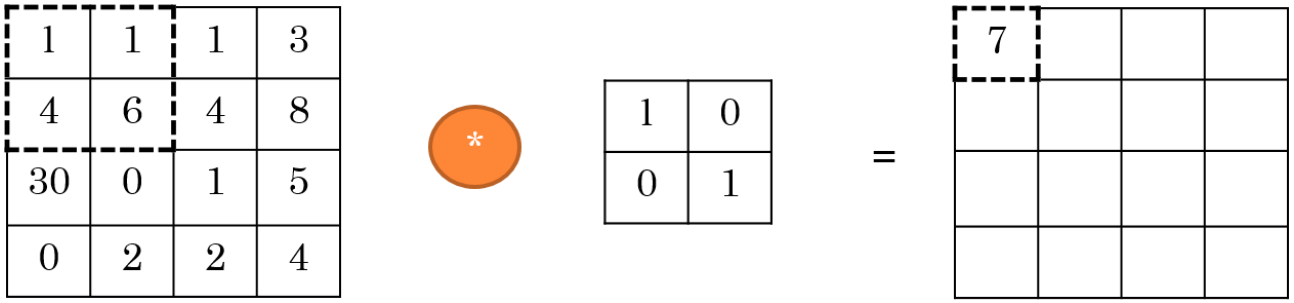
یادگیری عمیق: شبکه عصبی پیچشی ...

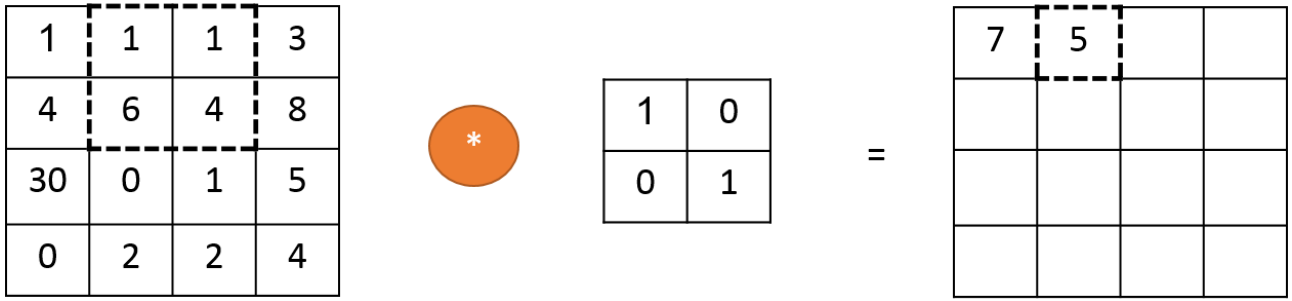
- لایه‌های رایج در این شبکه (به ترتیب)
  - ورودی (Input)
    - یک تصویر ۳ بعدی: طول، عرض و عمق (مثلا معادل ۳ برابر با سه مولفه RGB)
  - پیچش (Convolution)
    - هر نرون بلوکی از پیکسل‌های نزدیک به هم را می بیند
    - تابع فعال‌سازی ReLU
      - $f(x)=max(0,x)$
      - غیرخطی کردن رفتار شبکه
  - نمونه‌برداری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



یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

- به کمک فیلترهایی، وظیفه استخراج ویژگی‌ها را بر عهده دارد
- فیلترها = وزن‌ها
  - هر فیلتر برای یادگیری یک ویژگی است





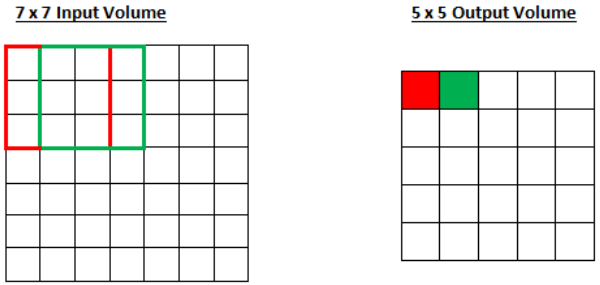
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

- **اندازه فیلتر = receptive field**
  - **اندازه ناحیه‌ای از ورودی که با هم بعد از ضرب در فیلتر به یک ویژگی تبدیل می‌شوند**
  - **مثال: تصویر ورودی [32x32x3] (۳۲*۳۲ پیکسل و RGB)**
    - با فیلترهای 5*5 آنگاه هر نرون در لایه پیچش دارای وزنهایی با ابعاد [5x5x3] در لایه ورودی است
      - تعداد وزن ها = 5*5*3 = 75 + یک بایاس

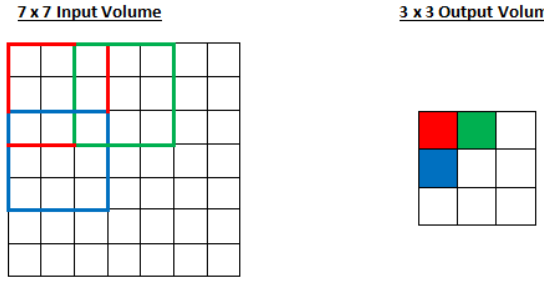
- **تعداد فیلتر = depth**
  - **تعداد فیلترهایی که می خواهیم استفاده کنیم**
  - **تعیین توسط طراح شبکه**
  - **مثال: تصویر ورودی [32x32x3] (۳۲*۳۲ پیکسل و RGB)**
    - استفاده از ۱۲ فیلتر، اندازه [32x32x12] را در لایه پیچش نتیجه می‌دهد
      - تبدیل تصویر سه بعدی [32x32x3] به تصویر سه بعدی [32x32x12]
      - اعمال تابع ReLU روی این تصویر جدید

یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

- گام = stride
  - **تعداد پیکسل‌های حرکت روی تصویر ورودی**
    - برای حالتی که اندازه تصویر از اندازه فیلتر بزرگ‌تر است (که در عمل معمولا اینگونه است)
  - **تاثیر روی اندازه تصویر خروجی**
    - هرچه بزرگتر باشد، تصویر خروجی کوچک‌تر می شود
  - **مقدار معمول ۱ و گاهی ۲**



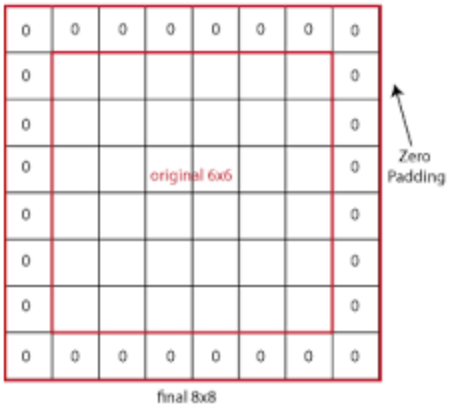
- مثال: مقدار ۱
  - خروجی ۵*۵
    - کوچکتر از ورودی

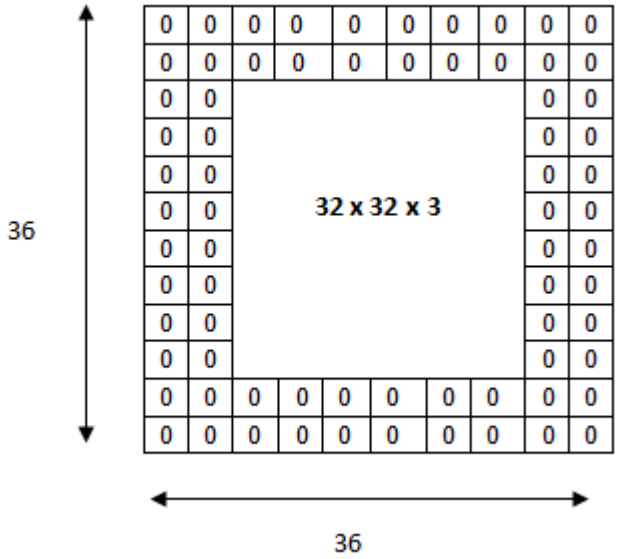


- مثال: مقدار ۲
  - افقی و عمودی
  - خروجی ۳*۳

یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

- افزودن صفر = zero-padding
  - افزودن سطرها و ستون‌های صفر در اطراف تصویر
  - کمک به کنترل اندازه تصویر خروجی
    - برای اینکه برابر با تصویر ورودی باشد
  - می‌تواند مقادیر مختلف (۱ یا ۲ یا بیشتر!) باشد





یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

- تنظیم مقادیر پارامترها . . .
  - اندازه تصویر ورودی = W
  - اندازه فیلتر = F
  - اندازه گام = S
  - اندازه افزودن صفر = P

  - اندازه تصویر خروجی: (W−F+2P)/S+1
    - مثال: اگر تصویر ورودی 7*7 باشد و فیلترها 3*3 باشد
      - با گام ۱ و صفر افزوده نشده باشد، اندازه خروجی 5*5 است: (7-3+2*0)/1+1
      - با گام ۲ و صفر افزوده نشده باشد، اندازه خروجی ۳*۳ است: (7-3+2*0)/2+1

  - برای گام ۱، تعداد صفرهای لازم جهت برابری اندازه تصویر ورودی و خروجی: (F-1)/2
    - مثال: اگر تصویر ورودی 7*7 باشد، فیلترها 3*3 باشد و گام ۱ باشد
      - تعداد صفر لازم = (3-2)/2=1

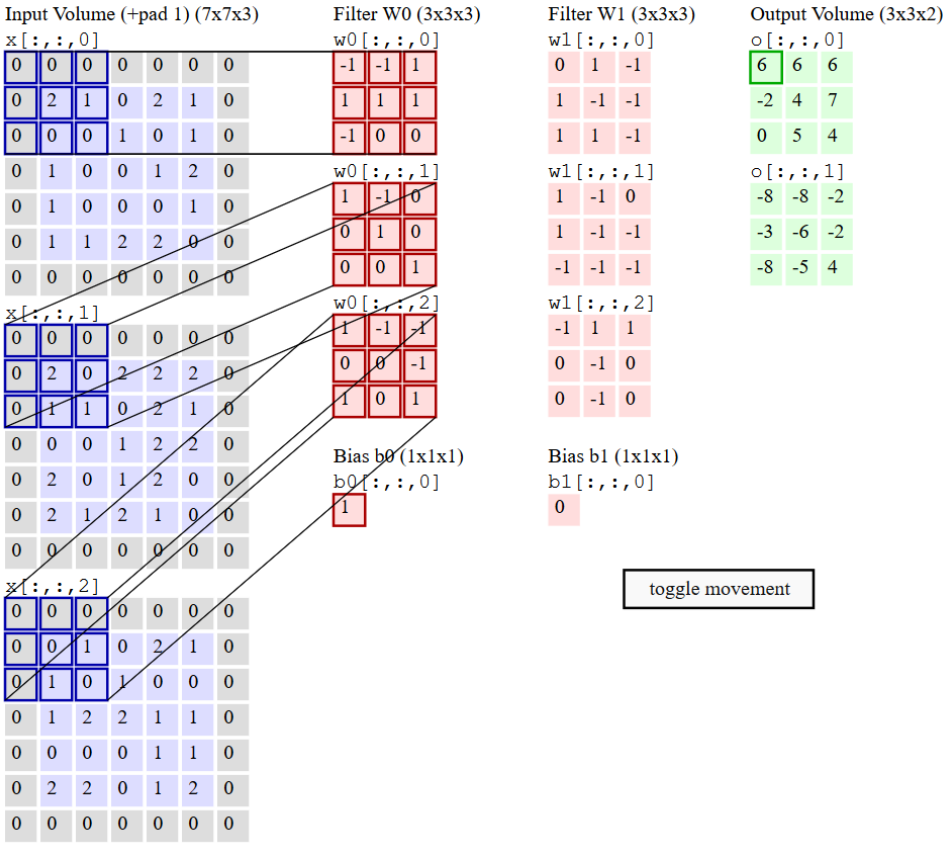
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

- **تنظیم مقادیر پارامترها: نمونه عملی**
  - **شبکه AlexNet در مسابقه ImageNet در ۲۰۱۲**
  - **تصاویر ورودی:** [227x227x3]
  - **در لایه پیچش اول**
    - اندازه فیلتر = ۱۱
    - اندازه گام = ٤
    - تعداد صفرهای افزوده شده = ۰
    - تعداد فیلتر = ۹۶
    - تصویر لایه پیچش اول = [55x55x96]
      - (227 - 11)/4 + 1 = 55
      - هر کدام معادل ناحیه [11x11x3] از تصویر ورودی

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

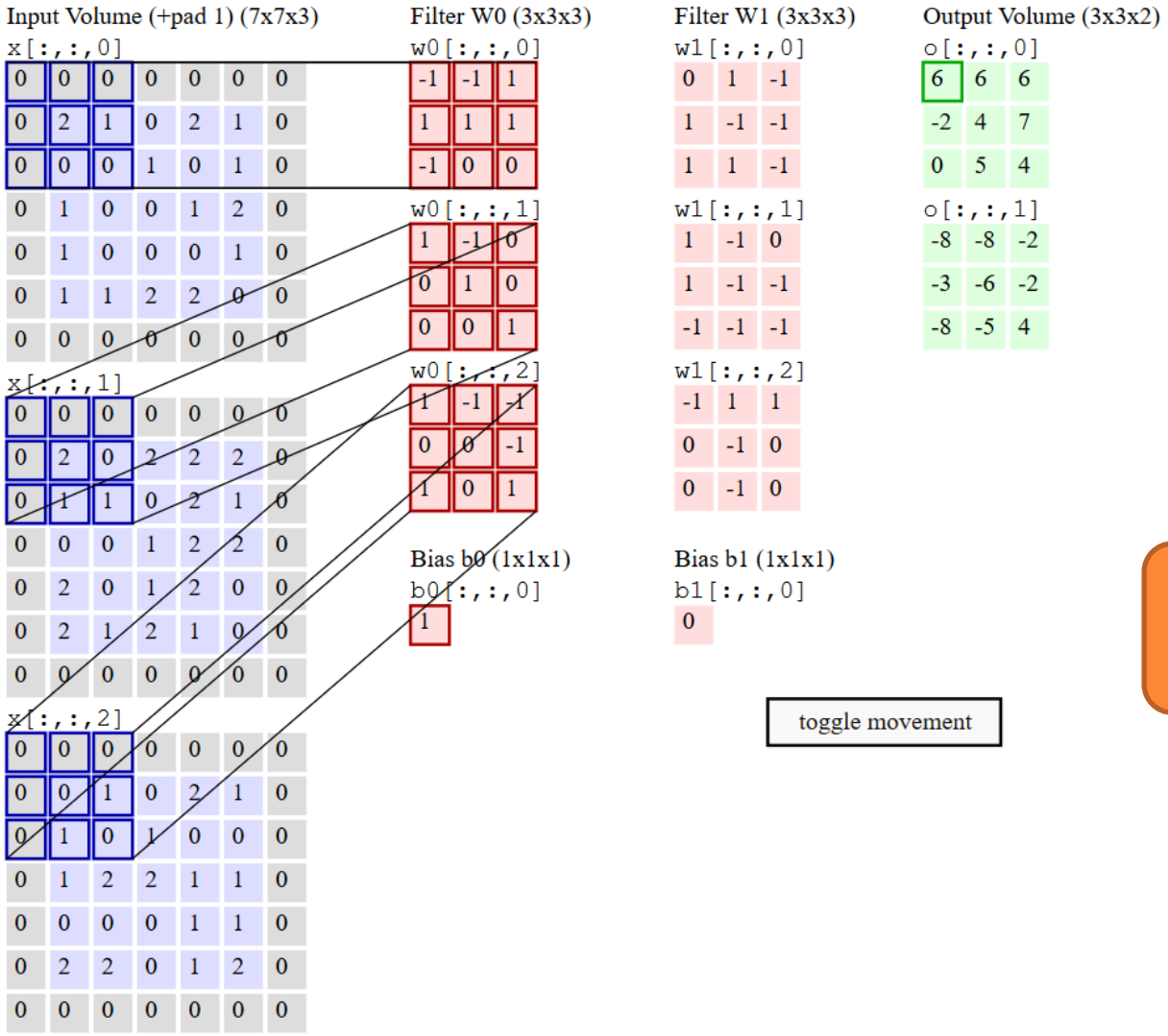
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

- دمو!
  - ورودی: تصویر سه بعدی [5x5x3]
  - اندازه فیلتر =F=3
  - تعداد فیلتر = K=2
  - اندازه گام = S=2
  - افزودن صفر = P=1



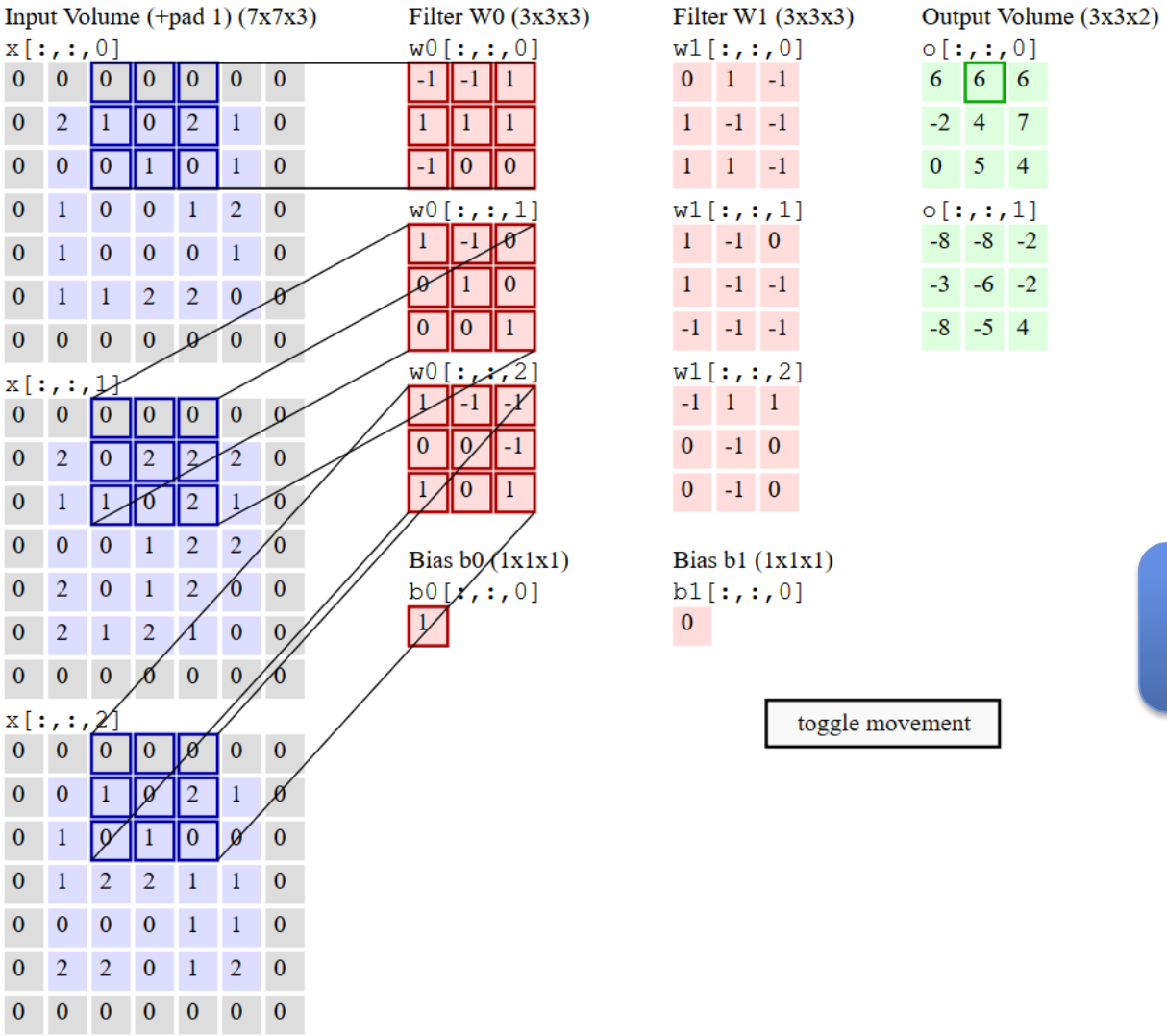
http://cs231n.github.io/convolutional-networks

یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



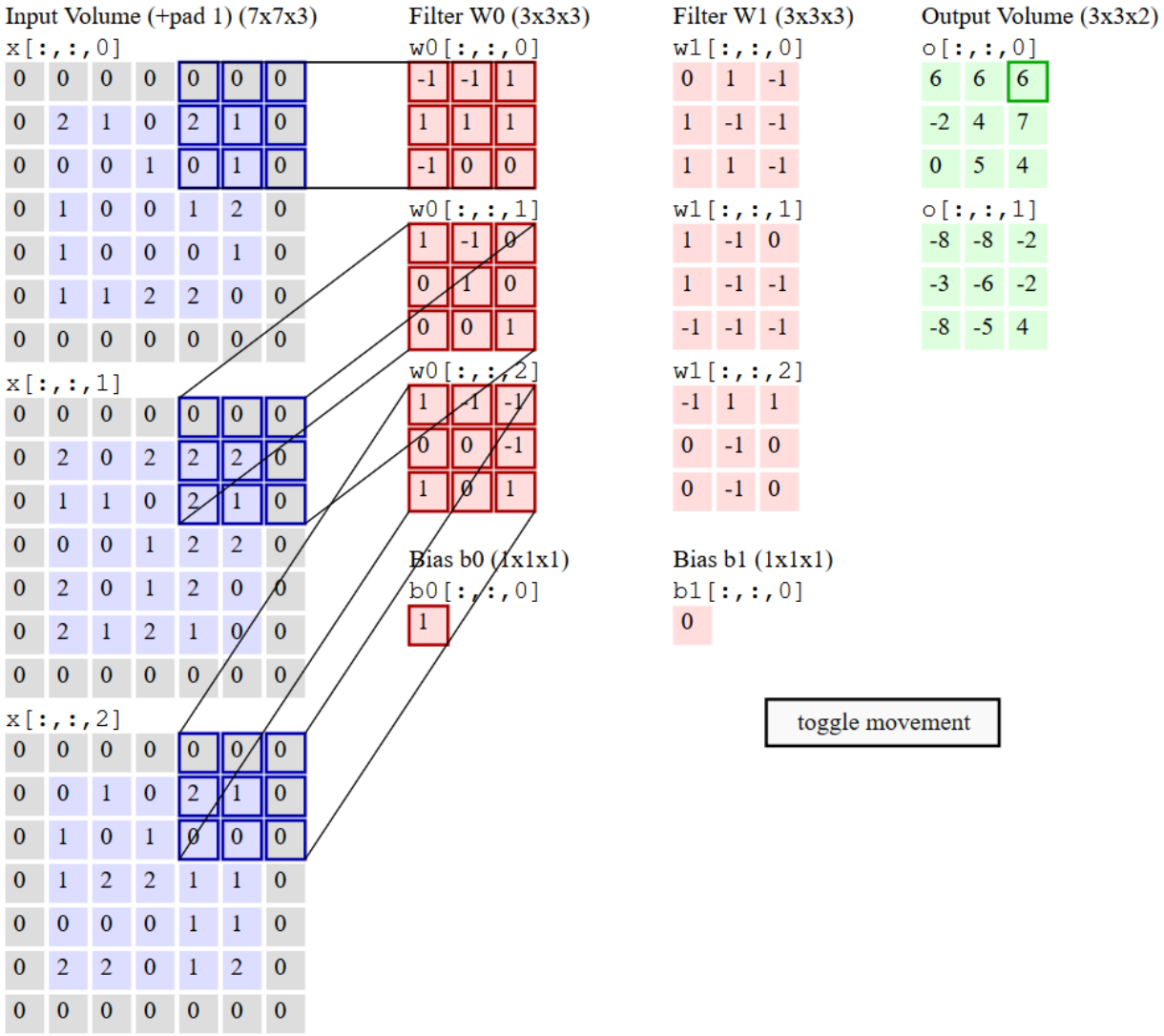
فیلتر اول
اولین نرون برش اول

یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

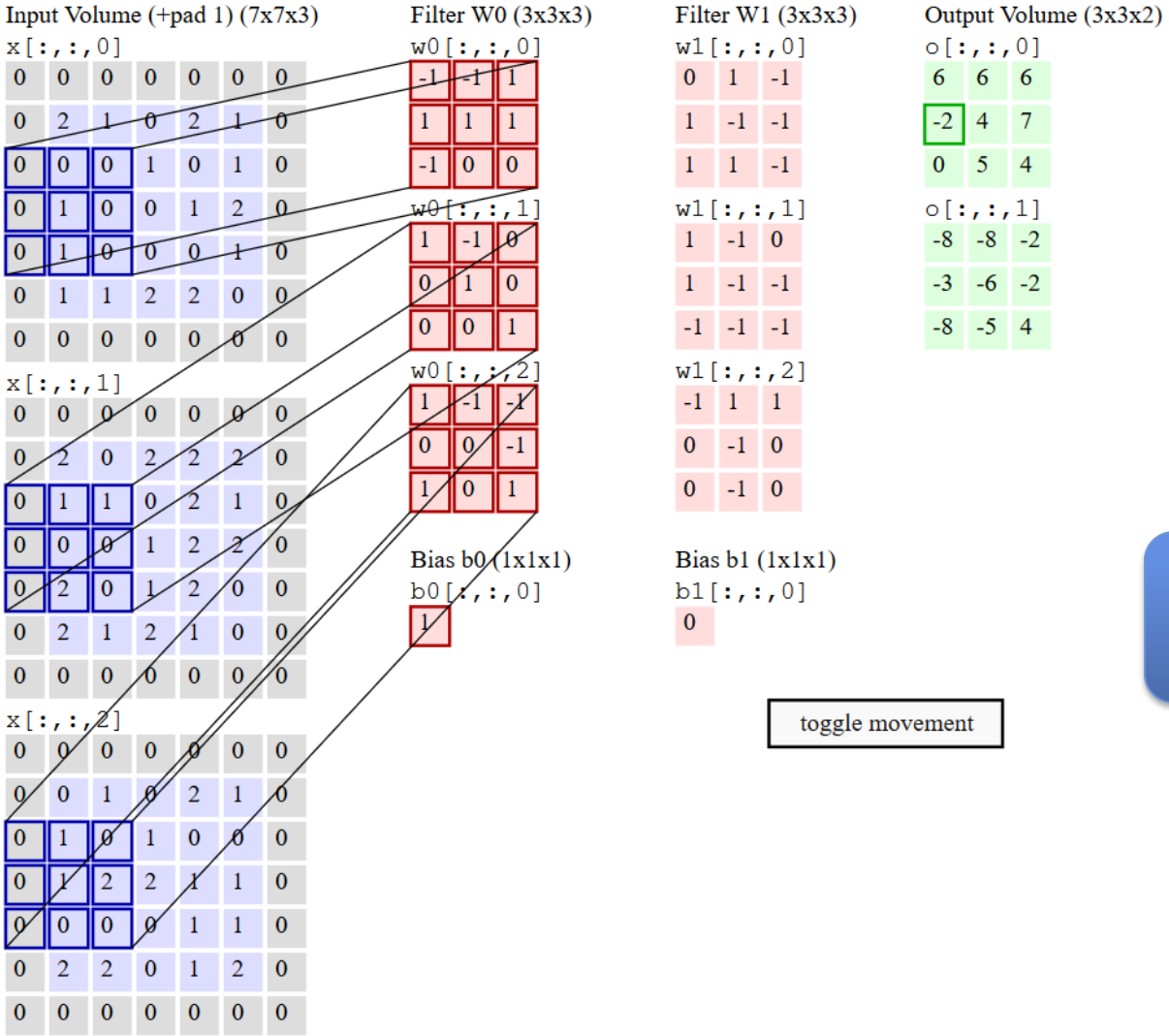


فیلتر اول
دومین نرون برش اول
گام = ۲ (افقی)

یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

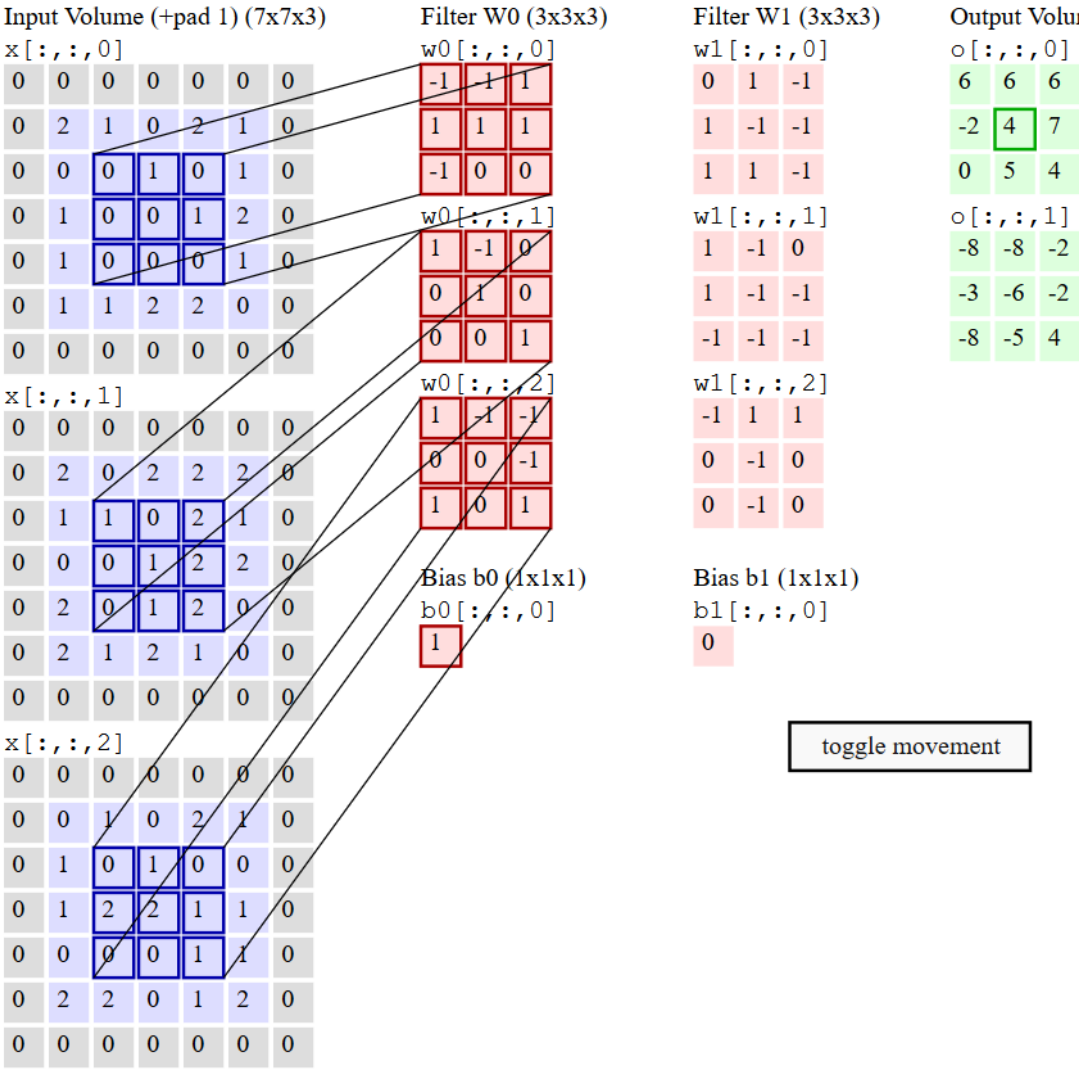


یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

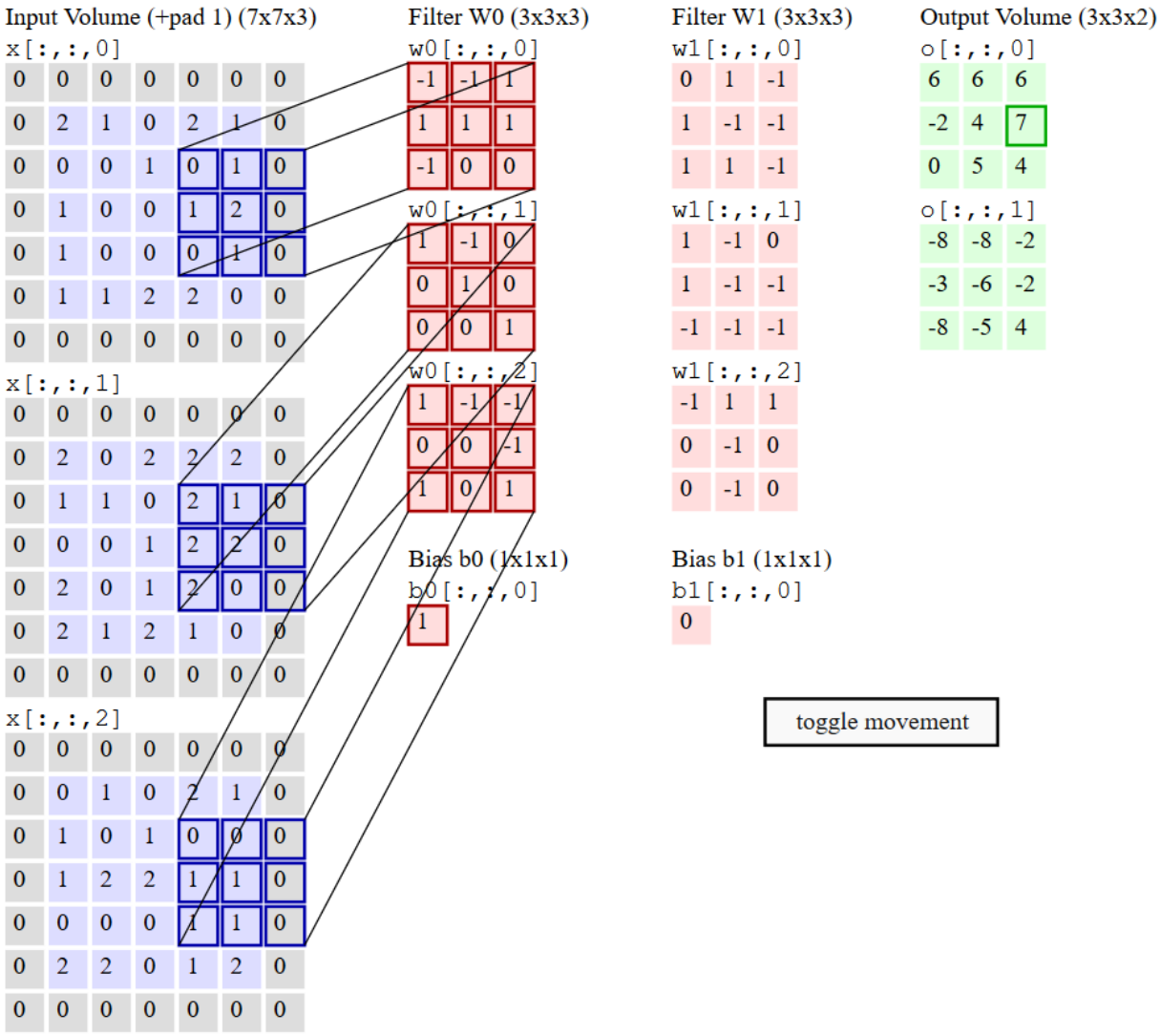


فیلتر اول
چهارمین نرون برش اول
گام = ۲ (عمودی)

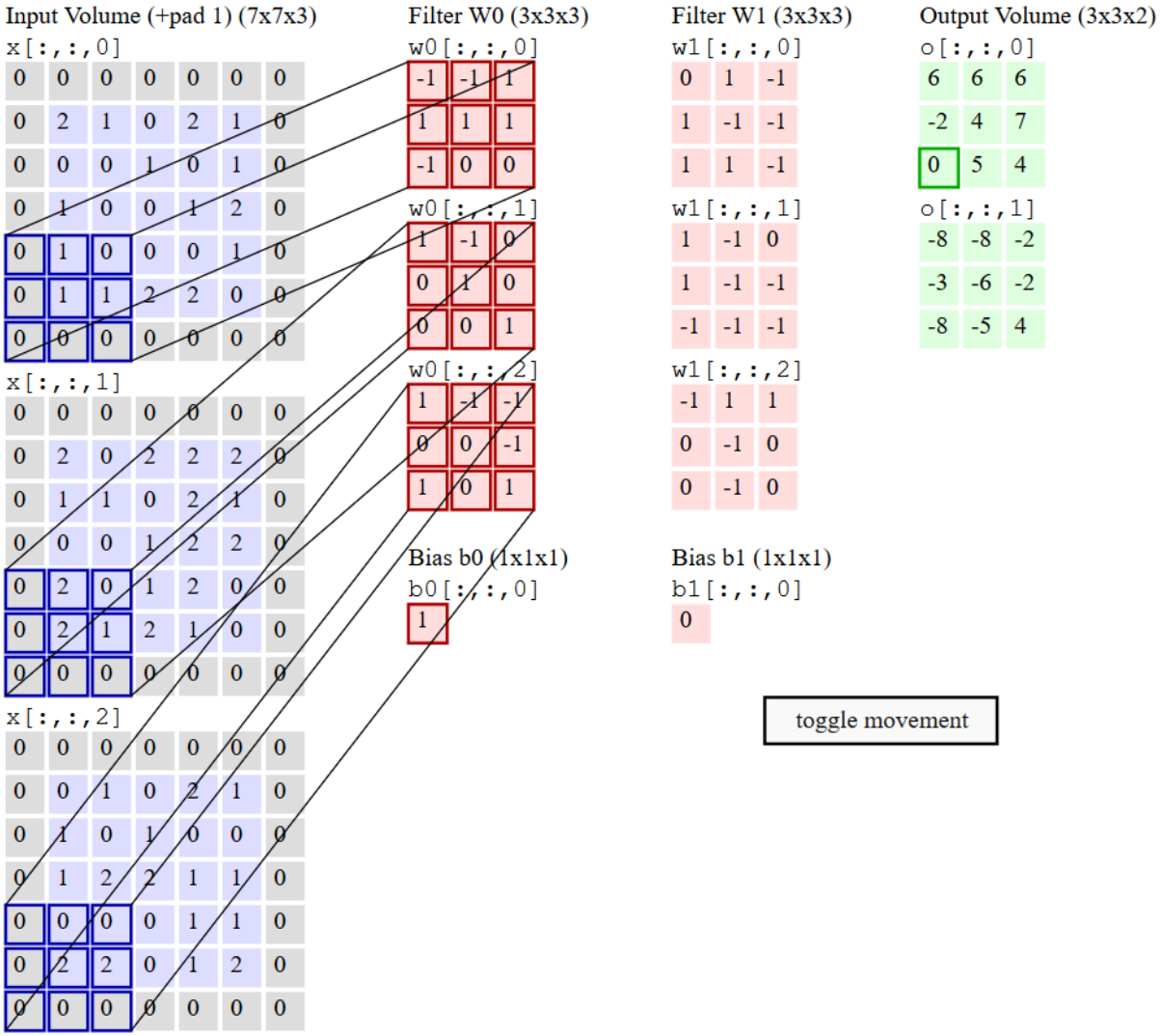
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



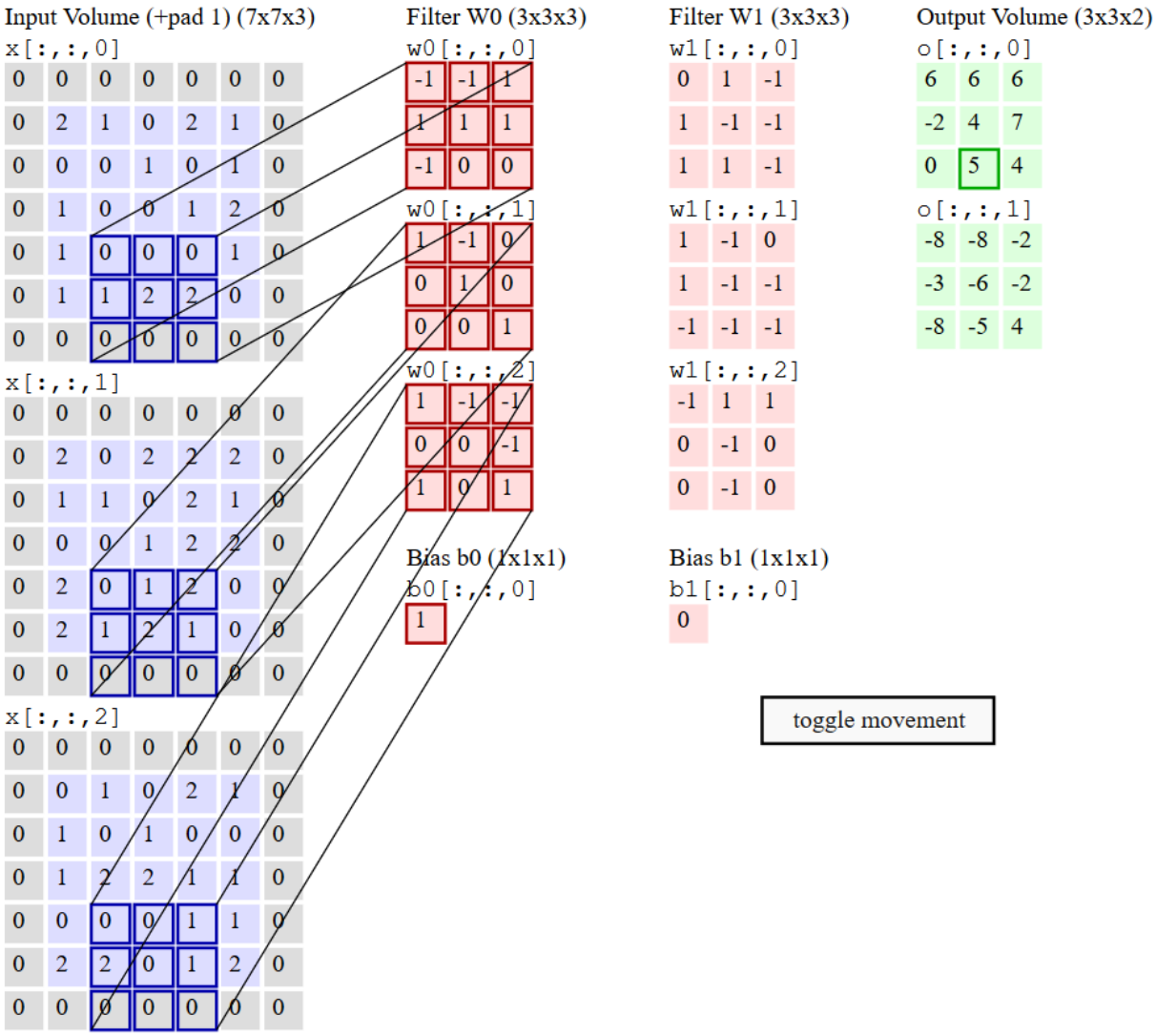
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



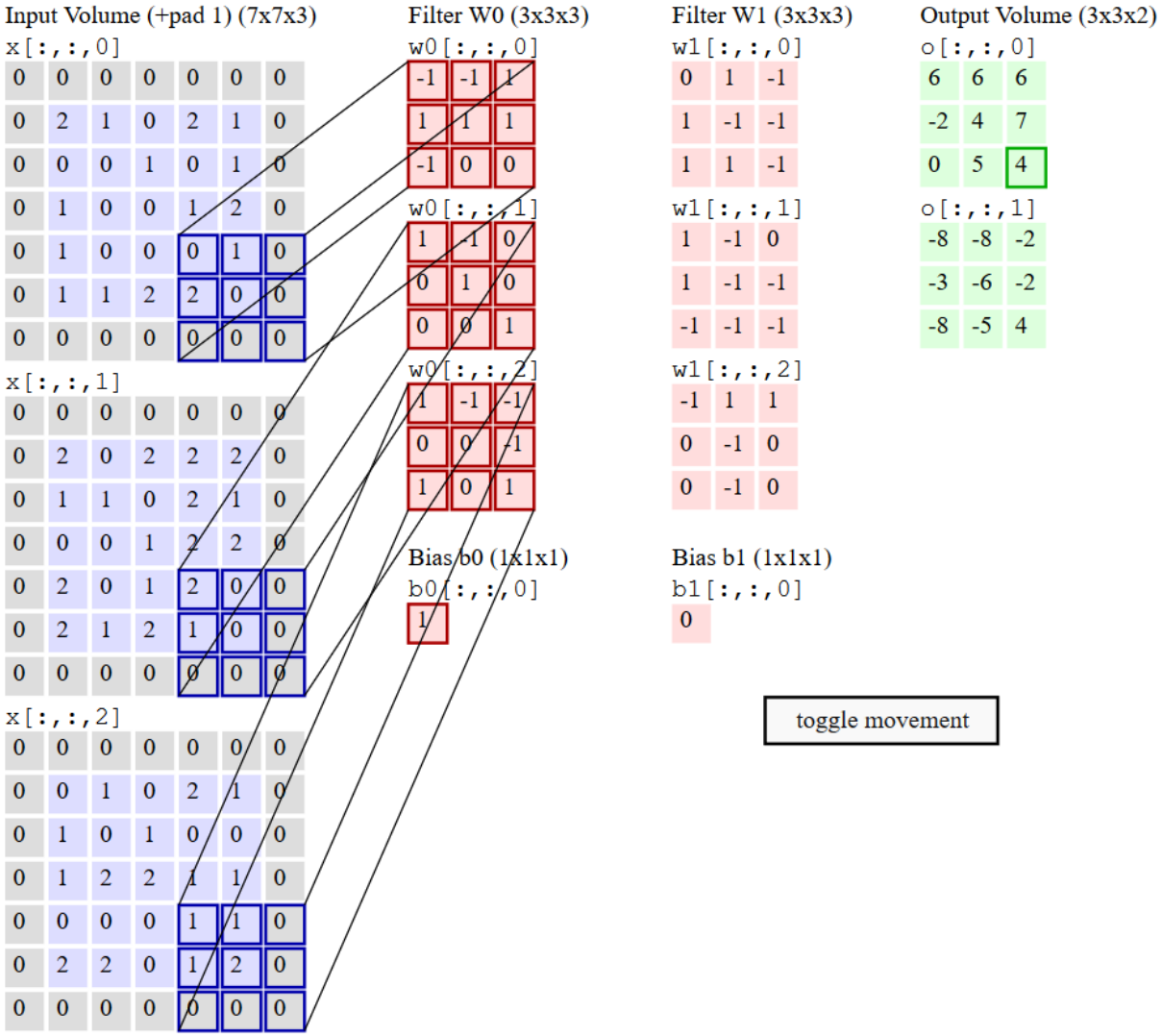
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



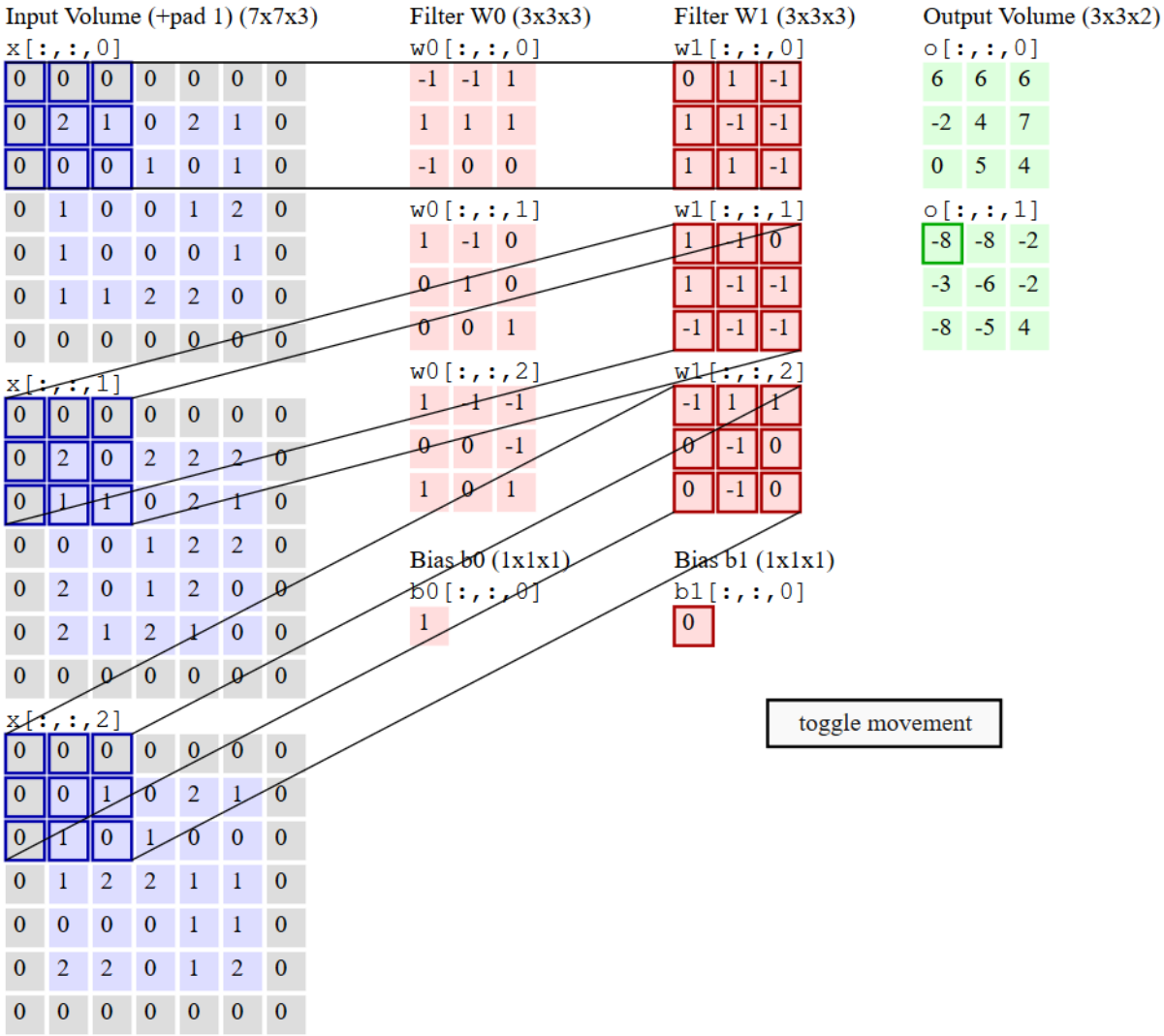
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

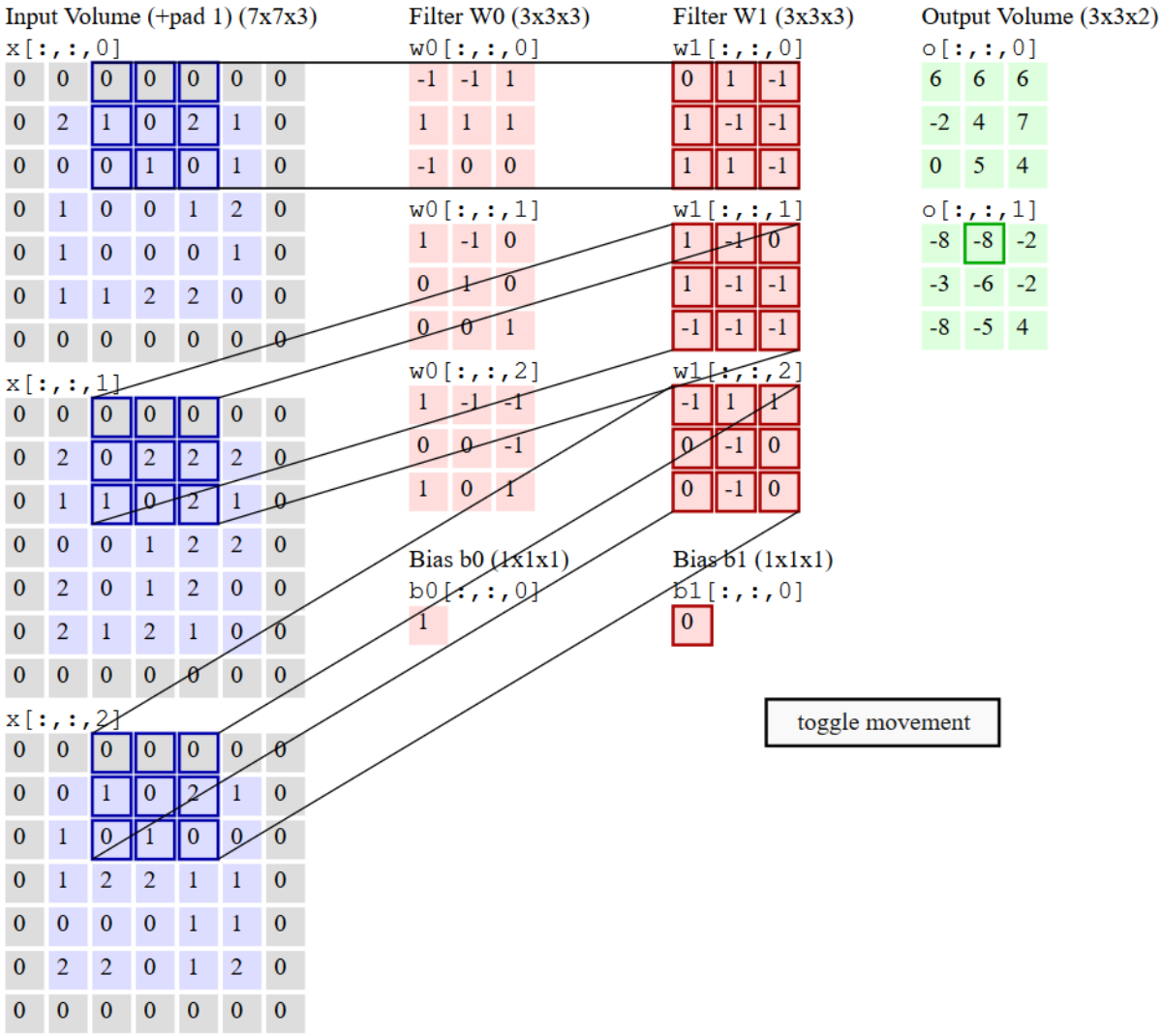


یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .

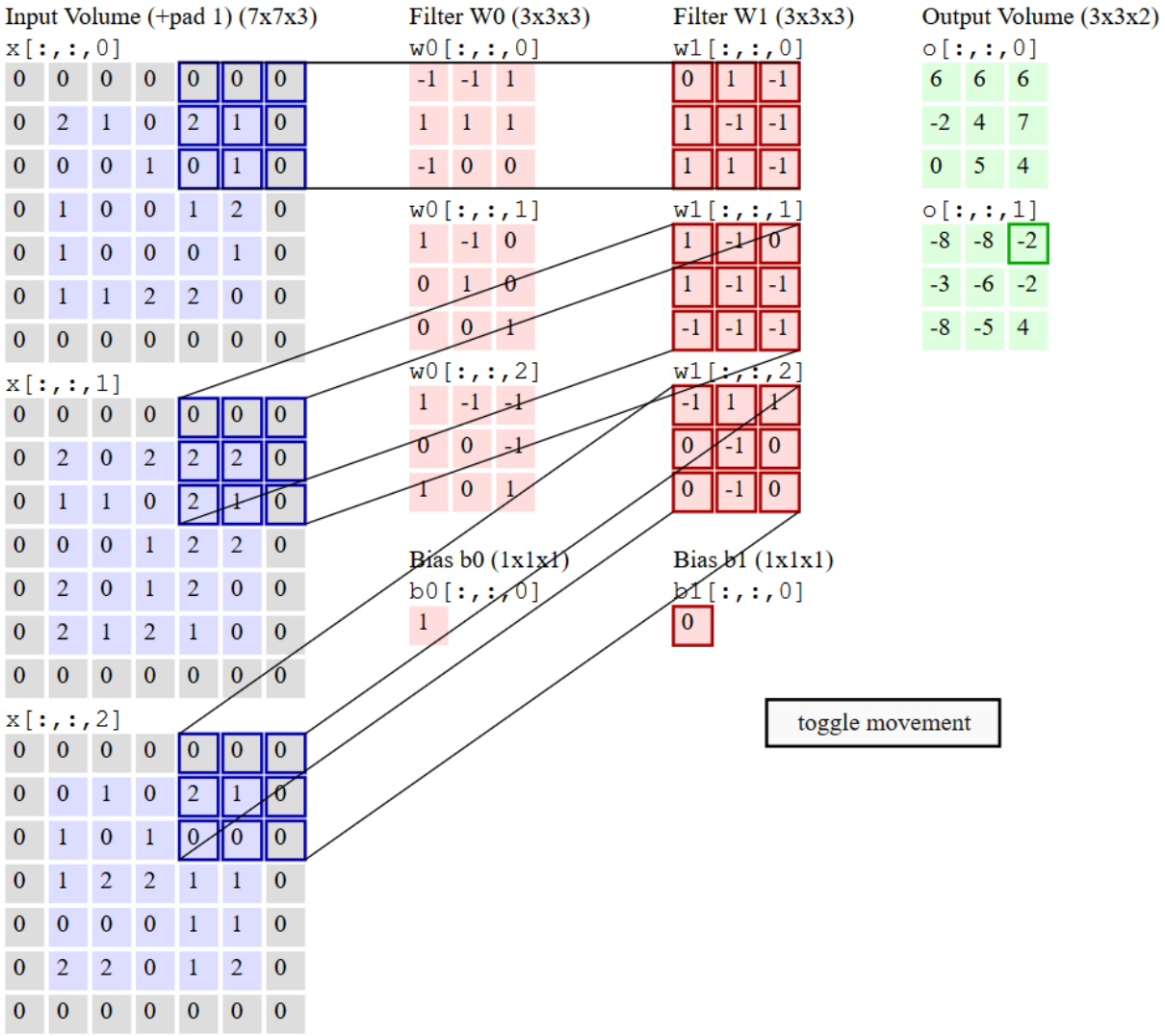


فیلتر دوم

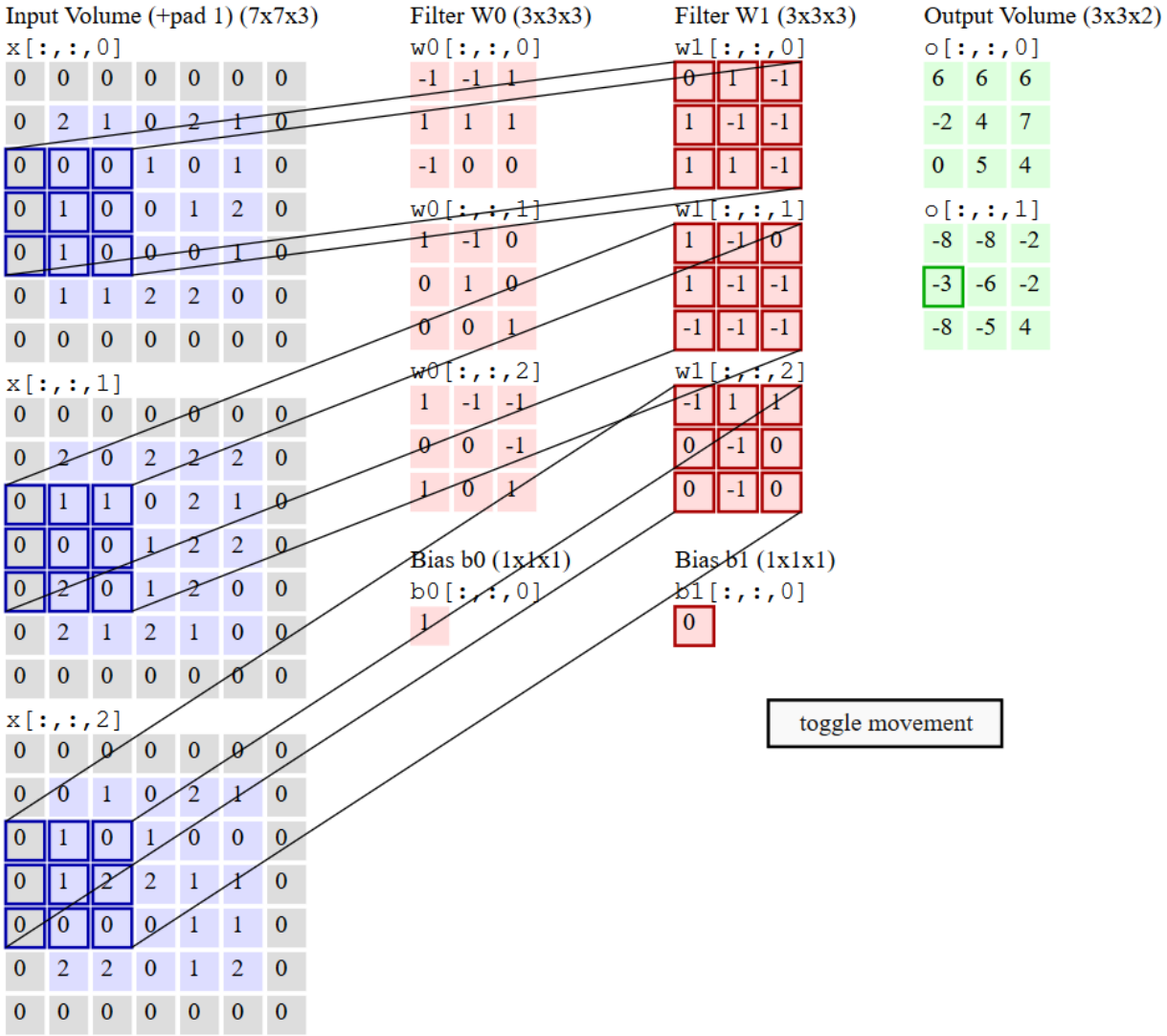
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



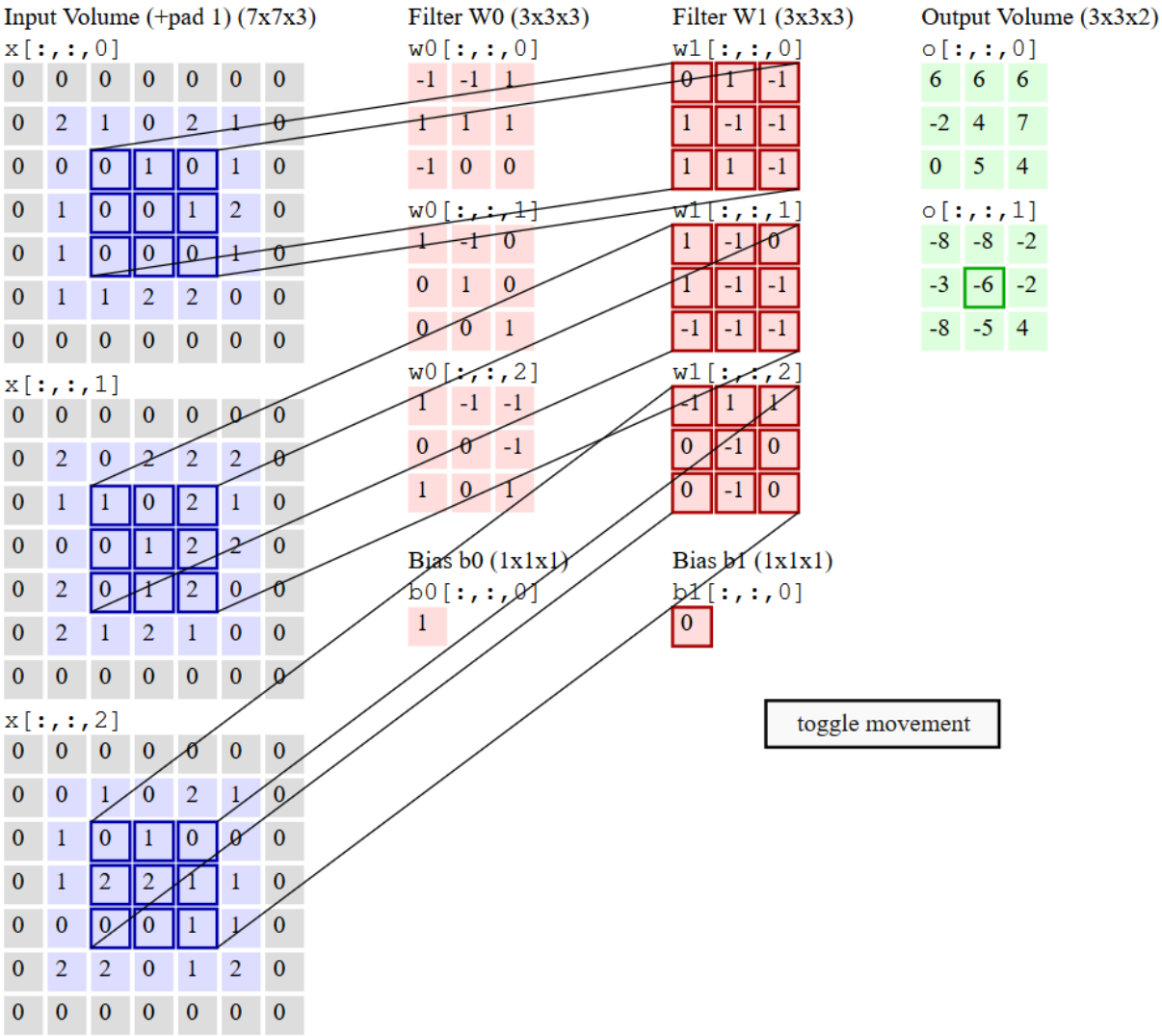
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



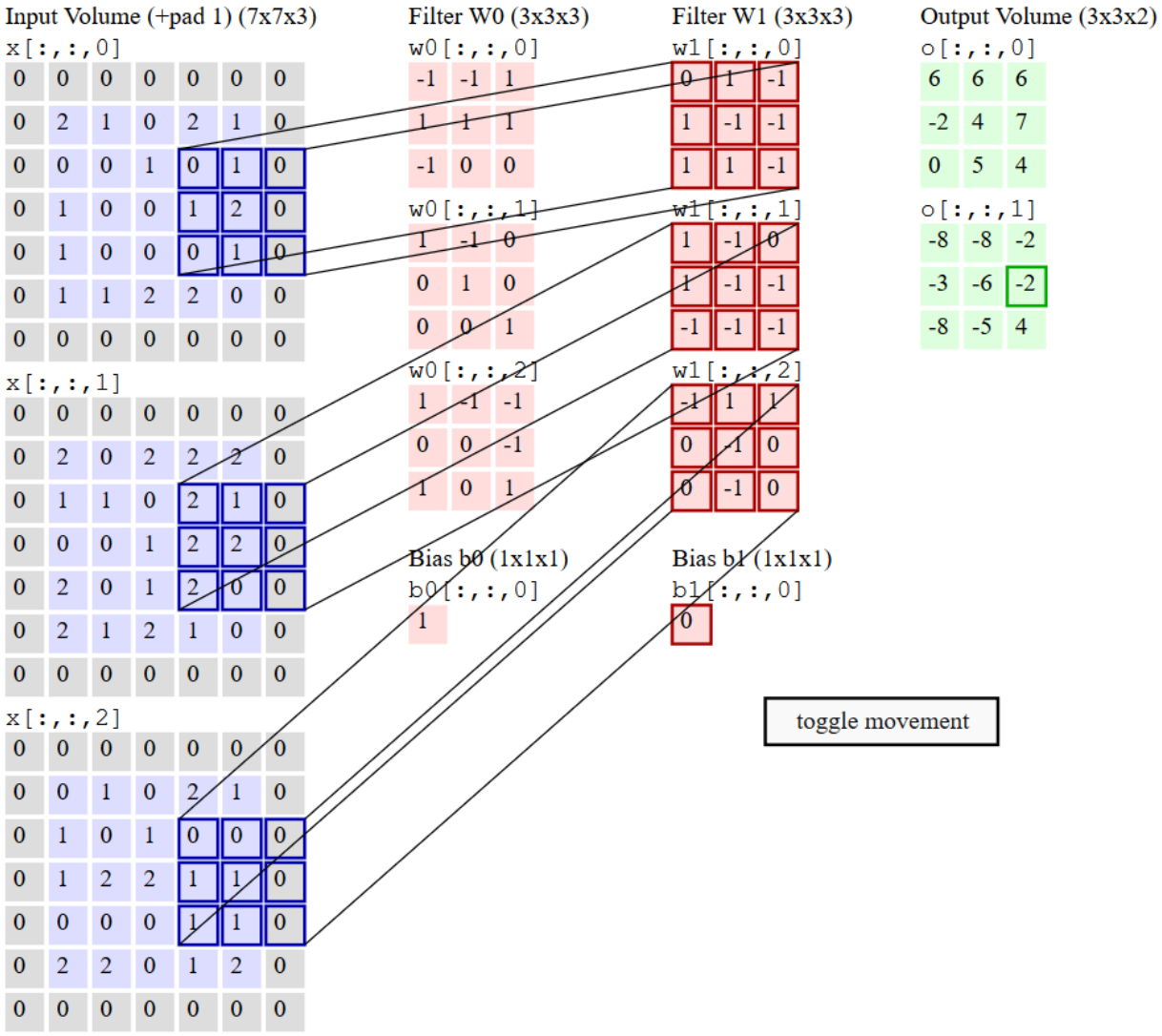
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



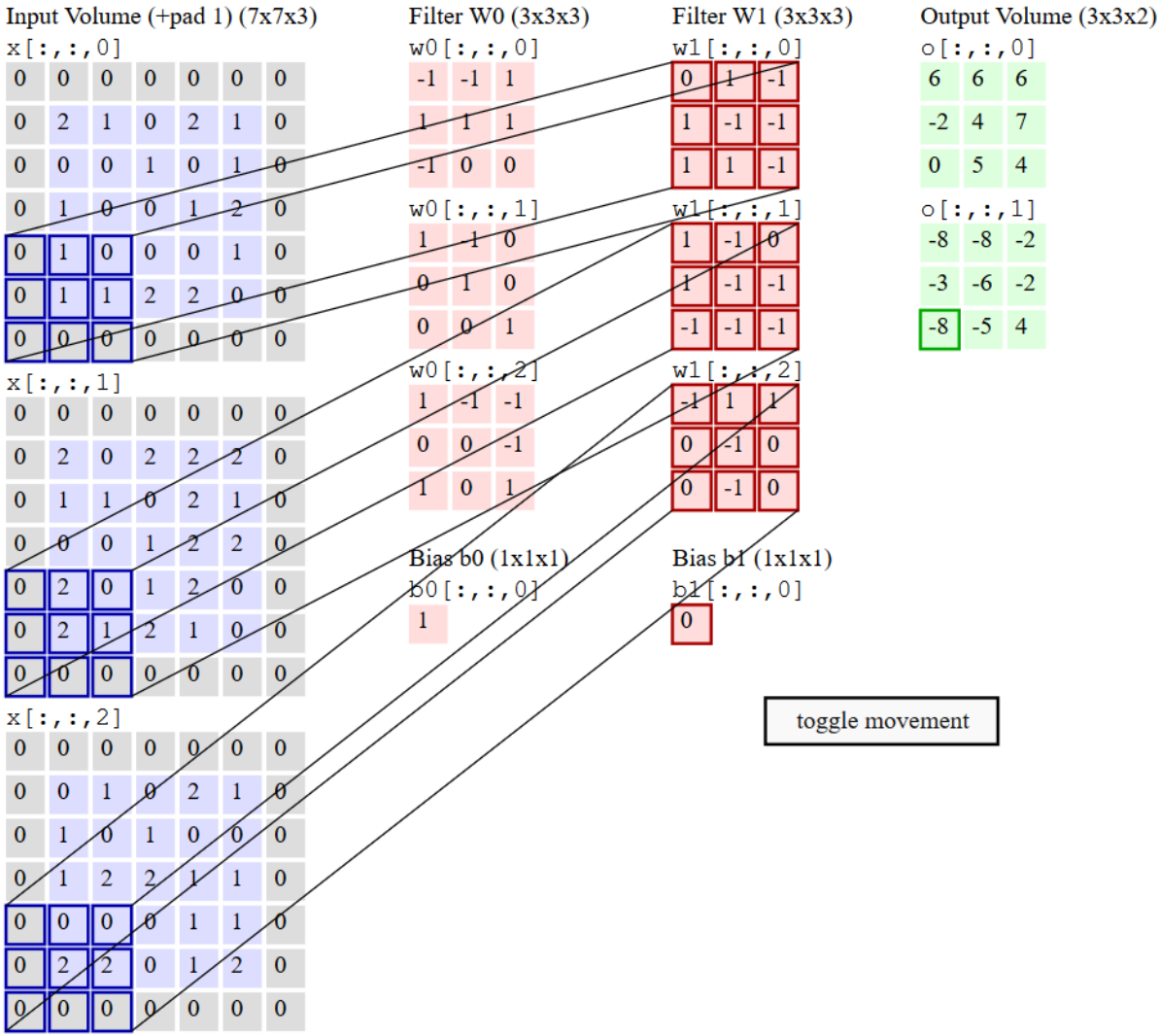
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



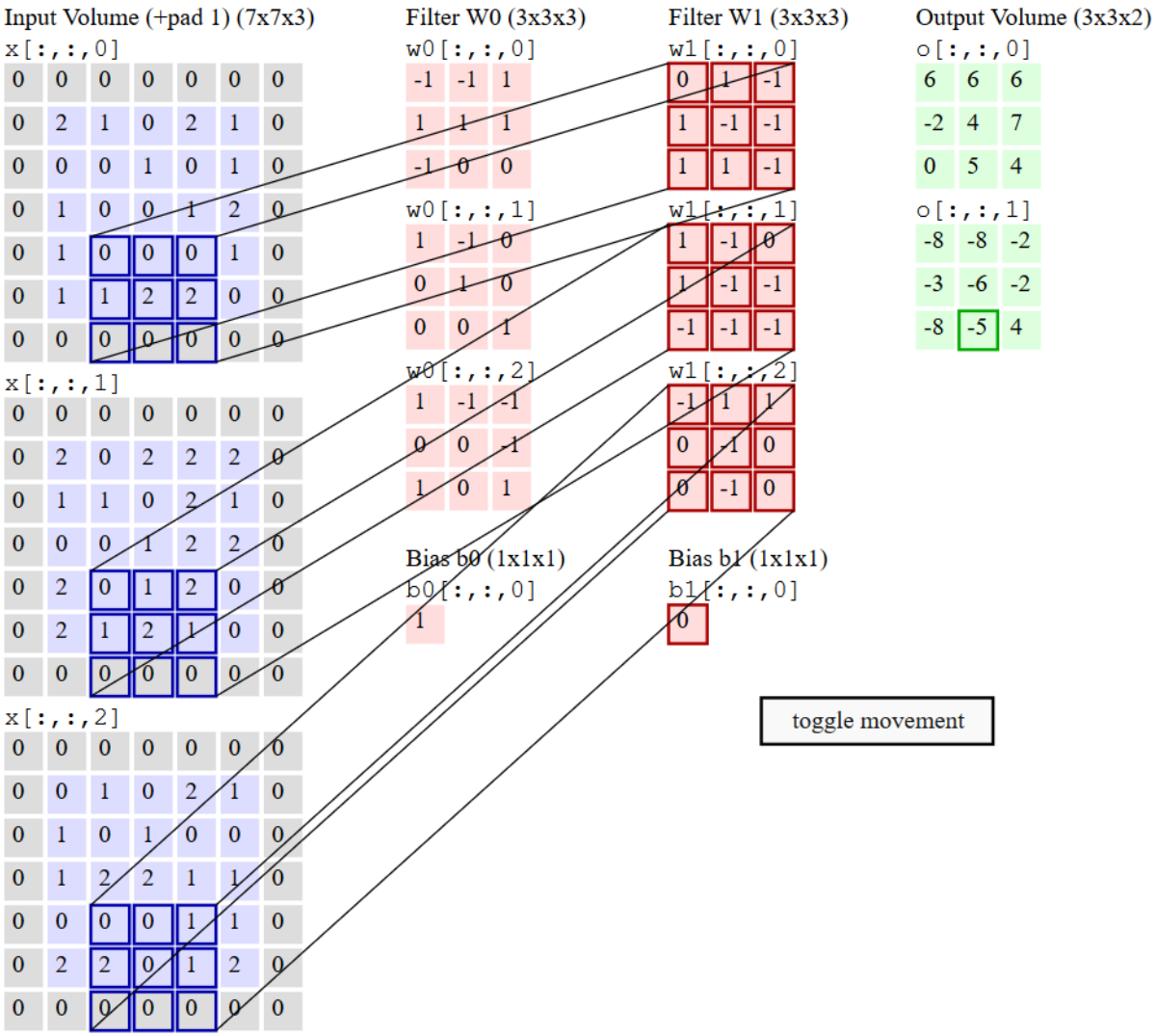
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



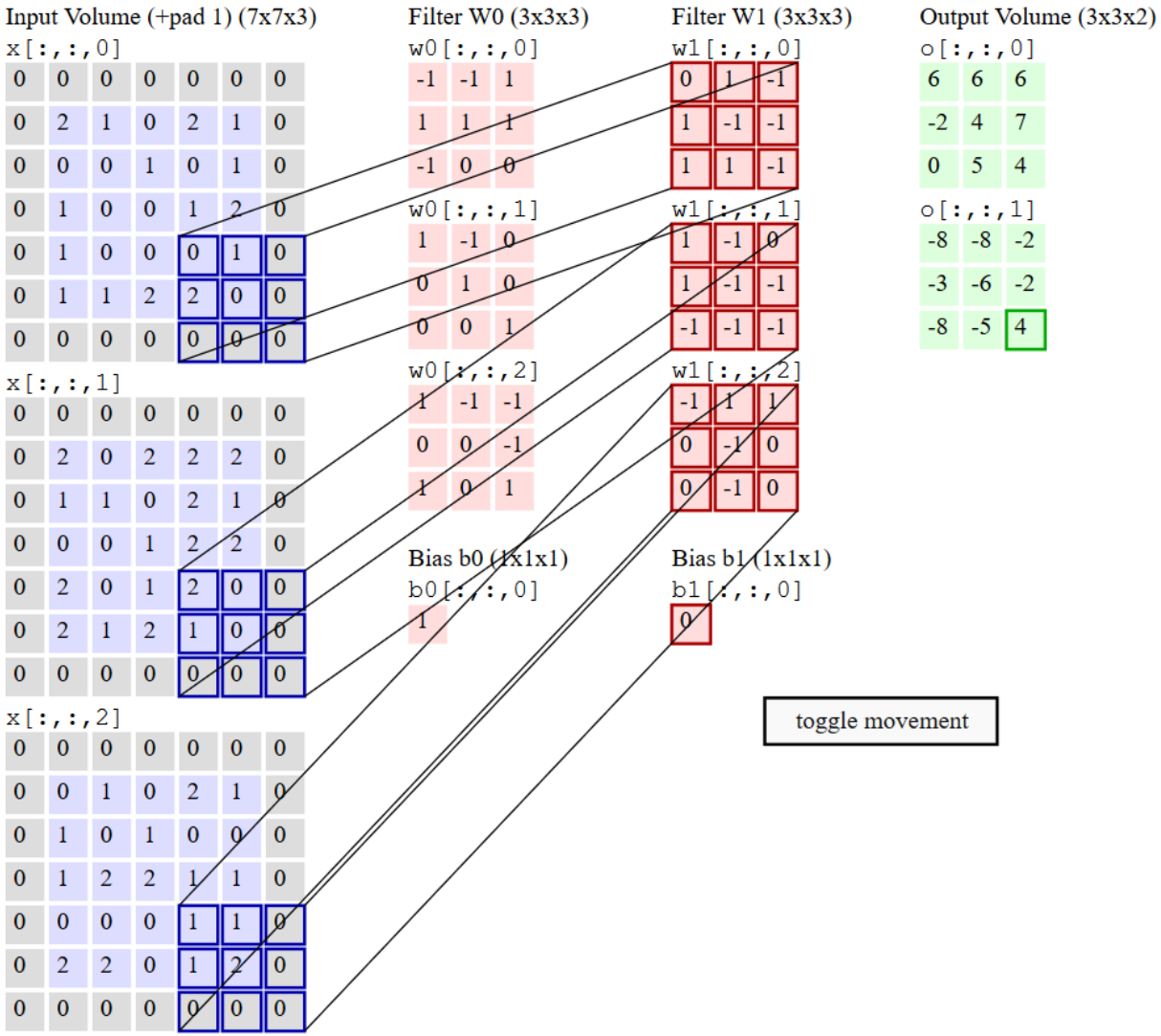
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



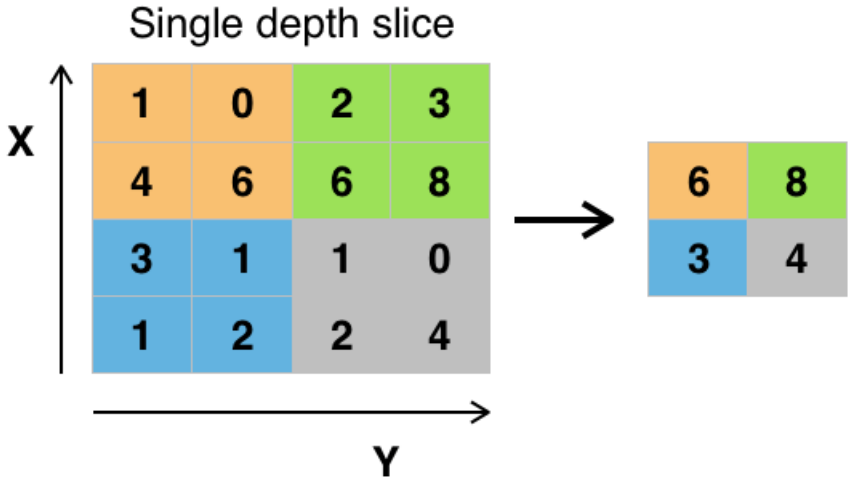
یادگیری عمیق- شبکه عصبی پیچشی: لایه پیچش . . .



یادگیری عمیق- شبکه عصبی پیچشی: لایه نمونه‌برداری . . .

- استفاده بعد از لایه پیچش
  - برای کاهش ابعاد تصویر
  - برای کنترل بیش برازش

- اعمال یک فیلتر 2*2 با گام ۲ برای بیشینه (Max) گرفتن
  - کاهش اندازه تصویر به نصف



یادگیری عمیق- شبکه عصبی پیچشی: لایه نمونه‌برداری . . .

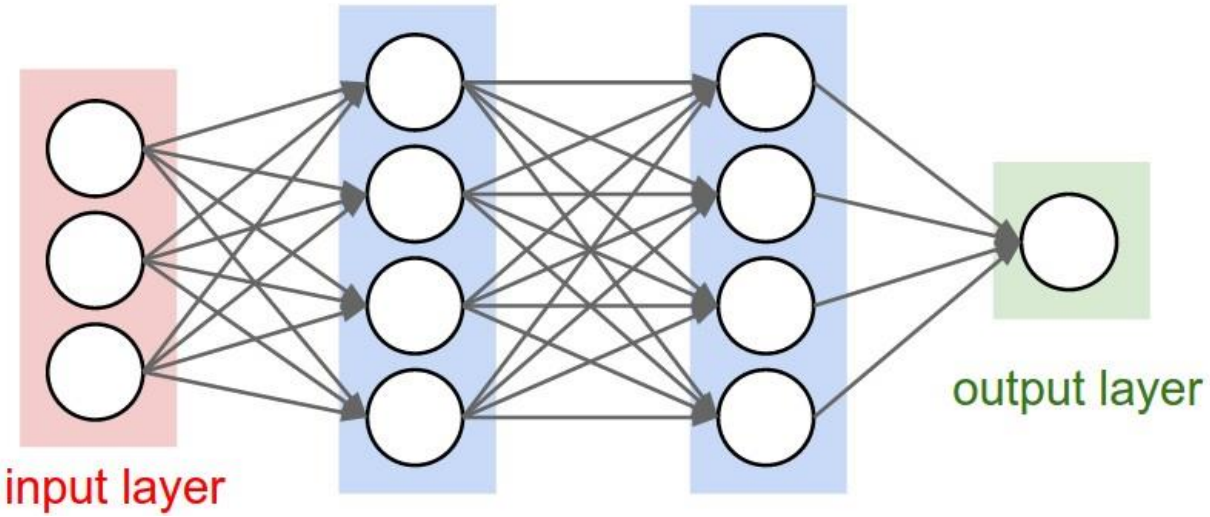
- روش‌های کاهش ابعاد
  - بیشینه (MAX): روش رایج
  - میانگین (Average)
  - نرم L2
- پارامترها
  - تصویر ورودی: W1×H1×D1
  - اندازه فیلتر F
  - اندازه گام S
  - تصویر خروجی: W2×H2×D2
    - W2=(W1−F)/S+1
    - H2=(H1−F)/S+1
    - D2=D1

یادگیری عمیق- شبکه عصبی پیچشی: لایه نمونه‌برداری . . .

- **مقادیر رایج پارامترها**
  - F=3, S=2 (**نمونه‌برداری همپوشان** overlapping pooling)
  - F=2, S=2 (**روش رایج**)
- **عدم استفاده از لایه نمونه‌برداری**
  - **در برخی موارد به جای استفاده از لایه نمونه‌برداری برای کاهش بعد، از همان لایه‌های پیچش با گام بزرگ استفاده می‌شود**
    - کارایی بهتر در آموزش مدل های تولیدی مانند variational autoencoders (VAEs) و generative adversarial networks (GANs)

یادگیری عمیق- شبکه عصبی پیچشی: لایه تمام متصل

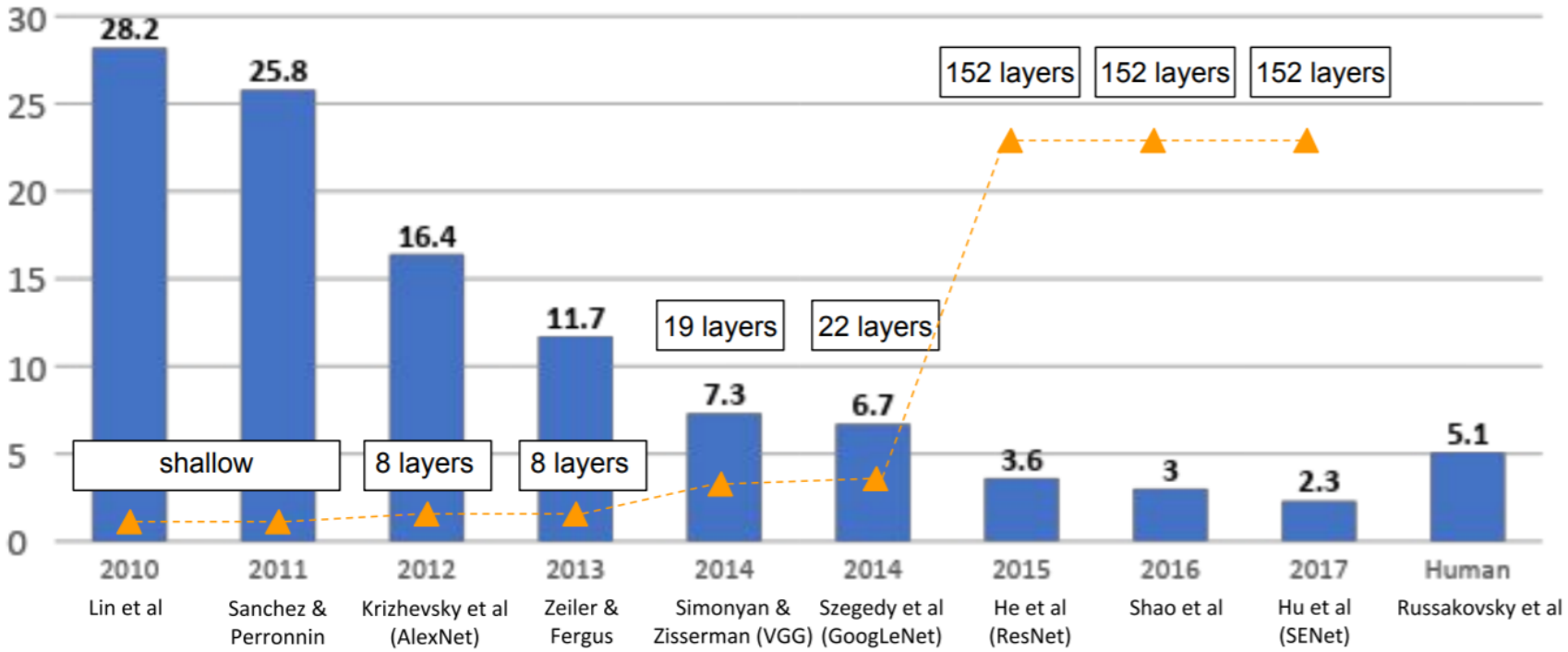
- استفاده به عنوان دسته بند
  - ورودی: تصویر آرایه شده آخرین لایه نمونه‌برداری/پیچش
  - خروجی: دسته های مختلف



- کارکرد با توابع فعال‌سازی مختلف مانند سیگموید و SoftMax

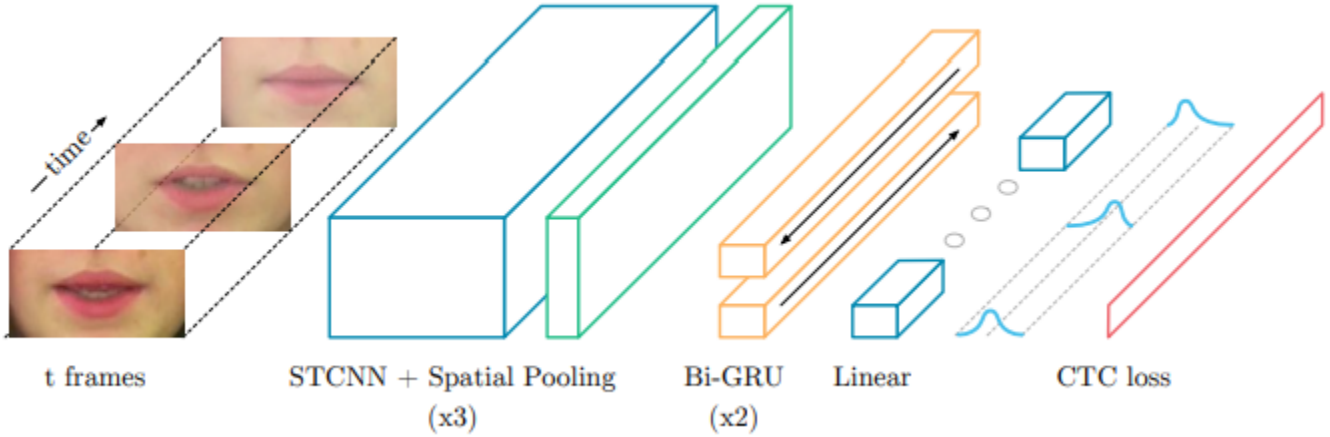
یادگیری عمیق: شبکه عصبی پیچشی . . .

- نرخ خطای دسته‌بندی روی ImageNet
  - ۱.۲ میلیون تصویر در ۱۰۰۰ دسته
  - نزدیک شدن کارایی شبکه پیچشی به عملکرد انسان



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]

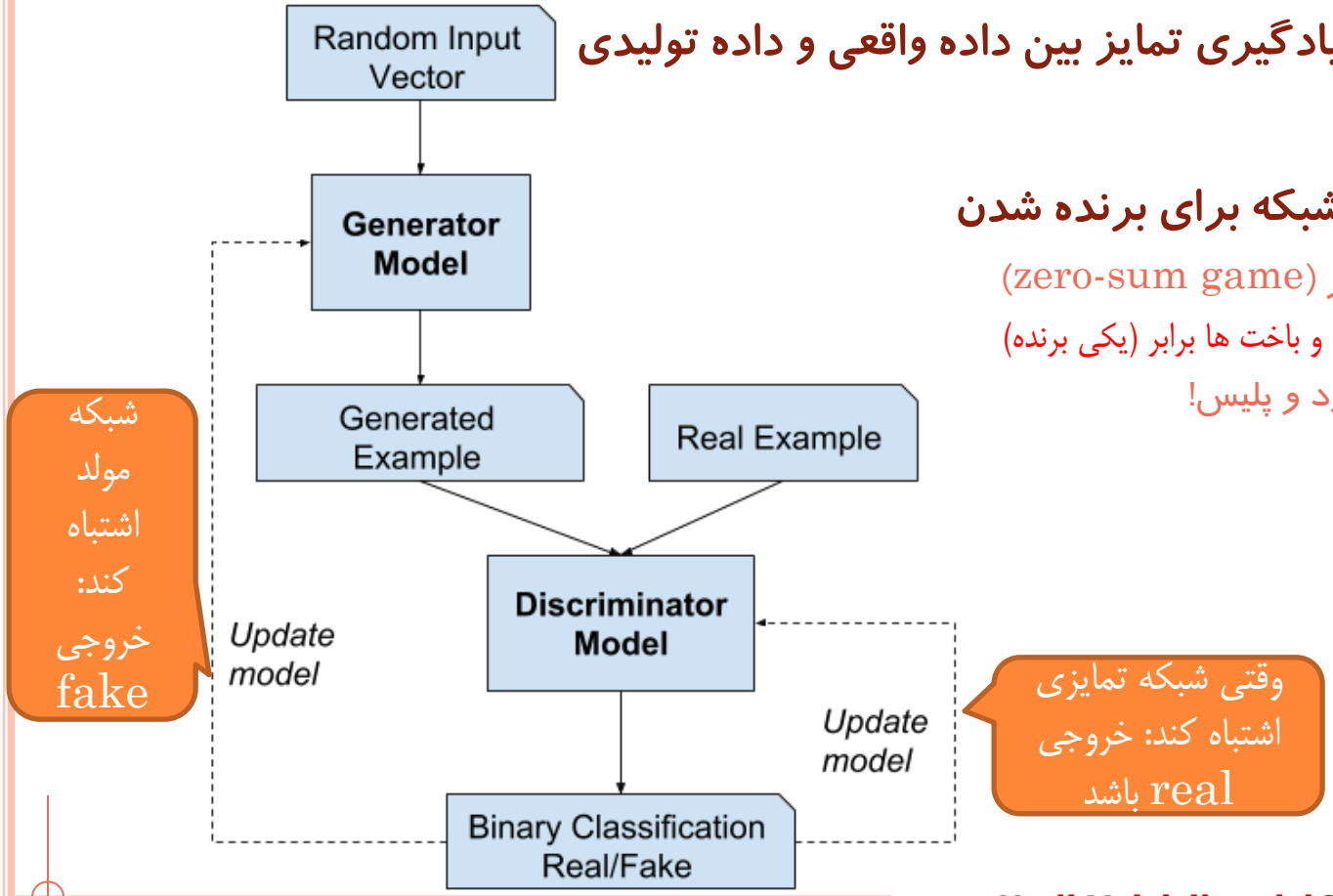
یادگیری عمیق-شبکه عصبی پیچشی: ماشین‌های خودران

- ماشین‌های خودران شرکت Tesla
  - تشخیص اشیا
  - پیش‌بینی
  - مسیریابی

[Csongor, Rob, "Tesla Raises the Bar for Self-Driving Carmakers", 2019]

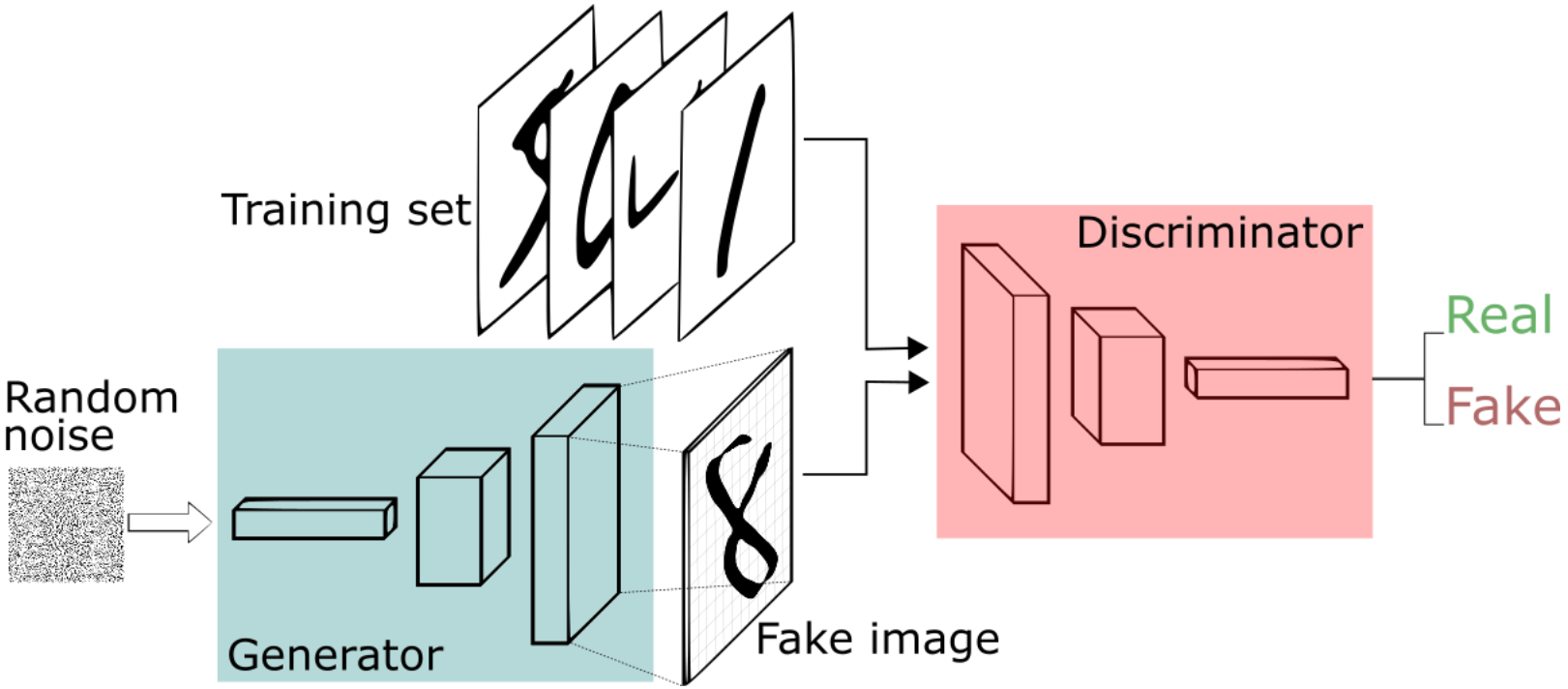
یادگیری عمیق: شبکه GAN ...

- 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!



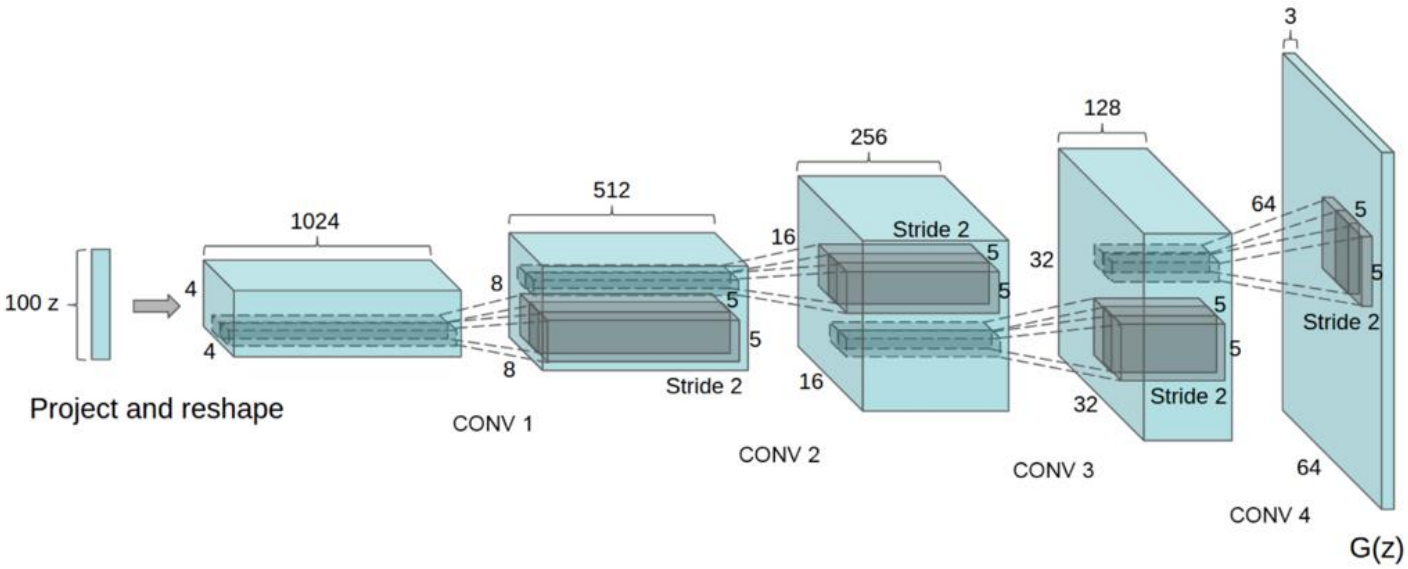
یادگیری عمیق: شبکه GAN ...

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



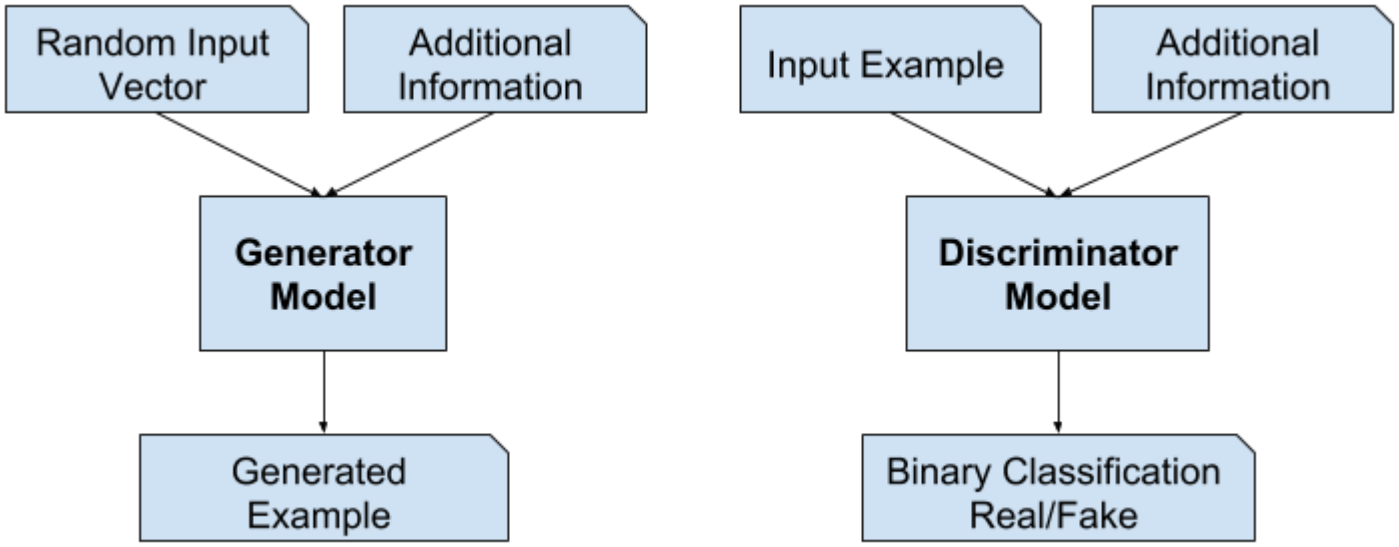
یادگیری عمیق: شبکه GAN ...

- 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.

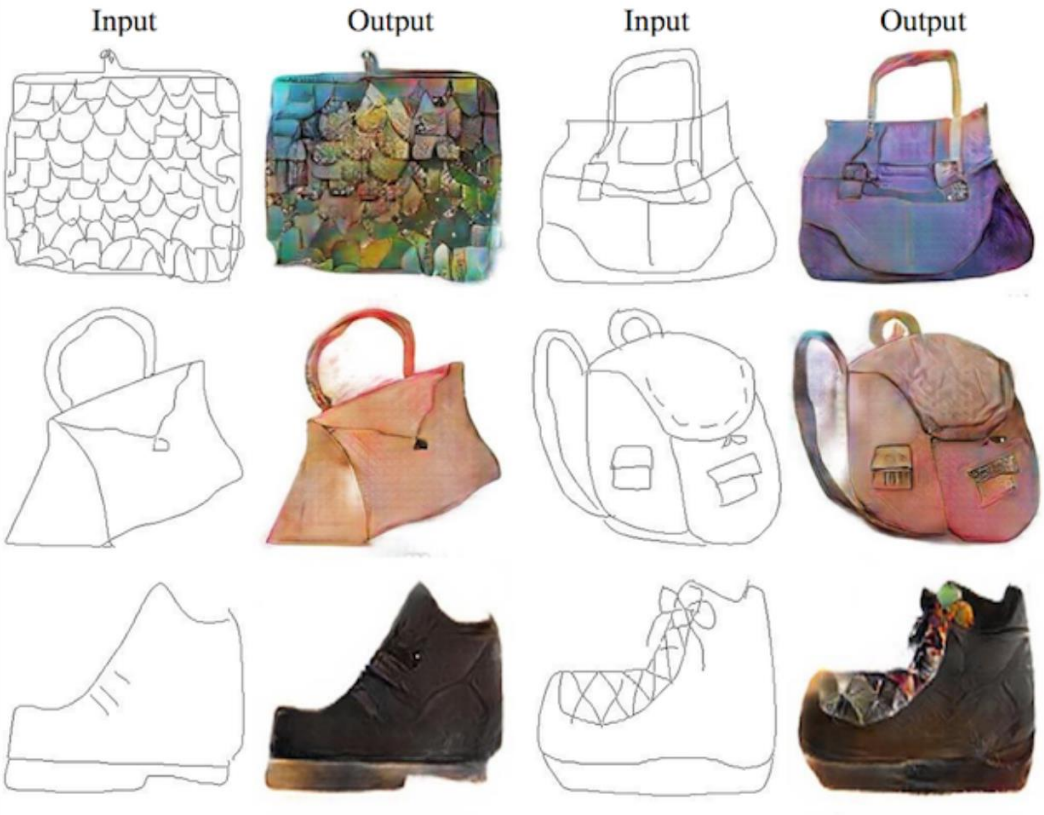
یادگیری عمیق: شبکه GAN . . .

- 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.

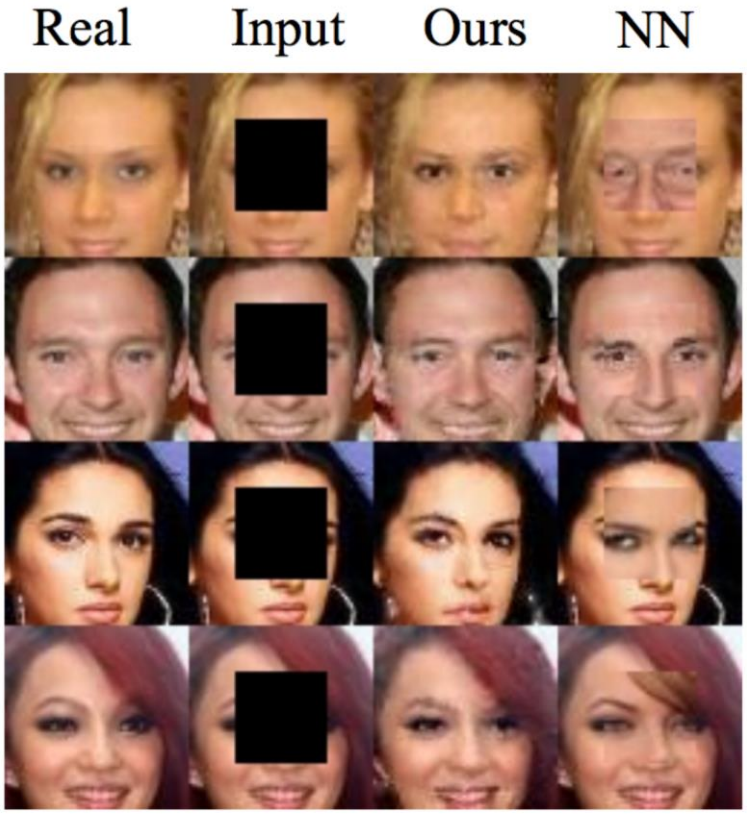
یادگیری عمیق: شبکه GAN . . .

- تبدیل متن به تصویر (text2image)

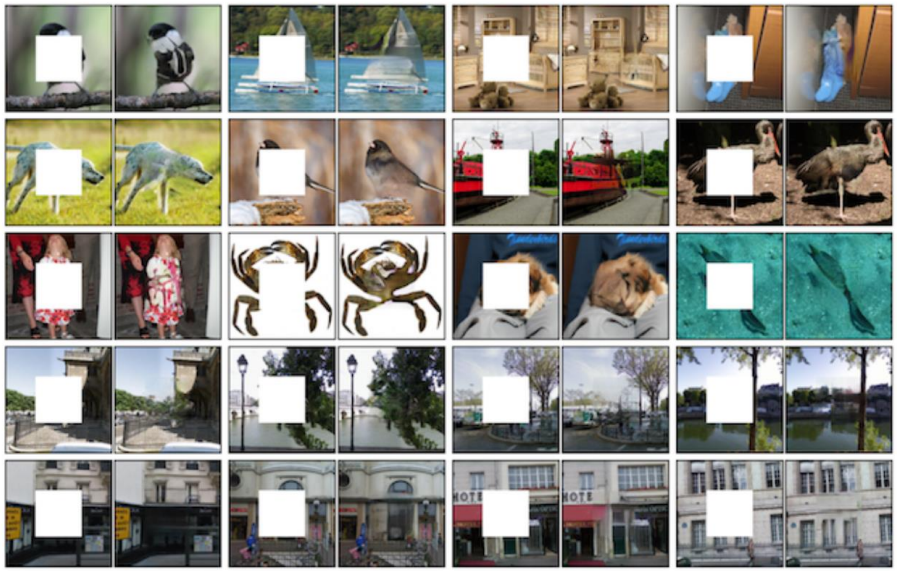
Example of Textual Descriptions and GAN-Generated Photographs of BirdsTaken from StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, 2016.

یادگیری عمیق: شبکه GAN

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



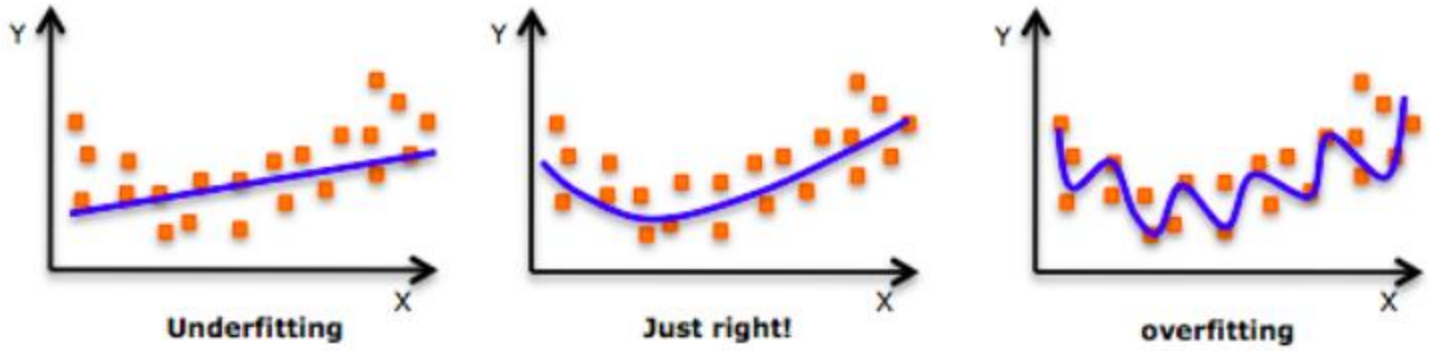
Example of GAN-based Inpainting of Photographs of Human FacesTaken from Semantic Image Inpainting with Deep Generative Models, 2016.



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: تنظیم ...

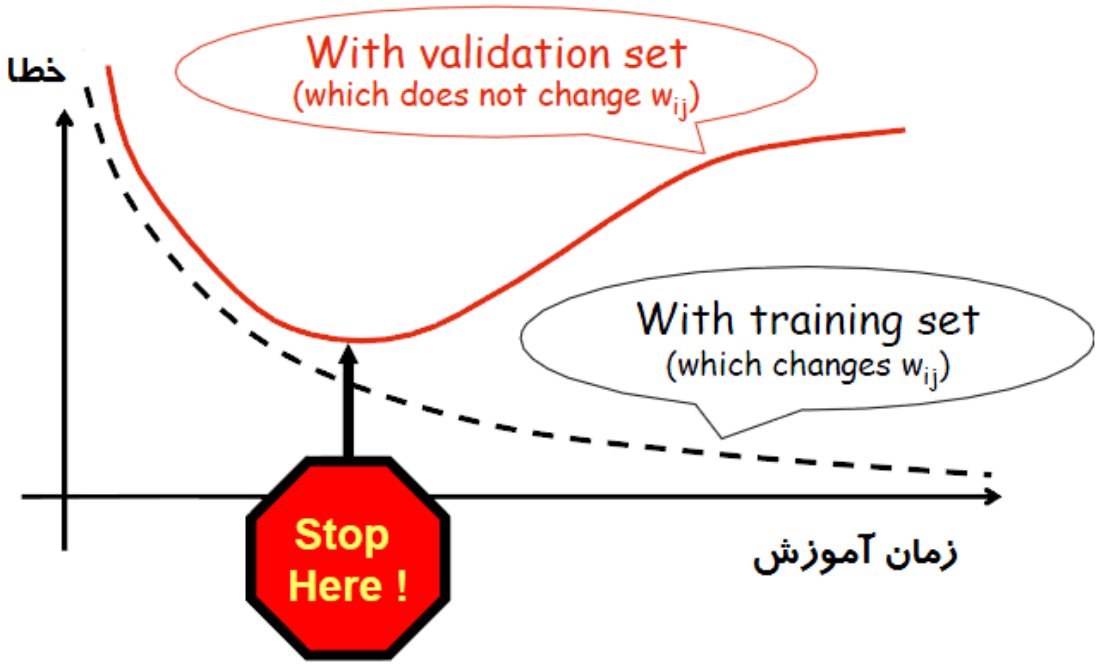
- تنظیم شبکه (Regularization)
  - روش‌هایی برای افزایش قدرت تعمیم پذیری شبکه و جلوگیری از بیش برازش



- 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

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

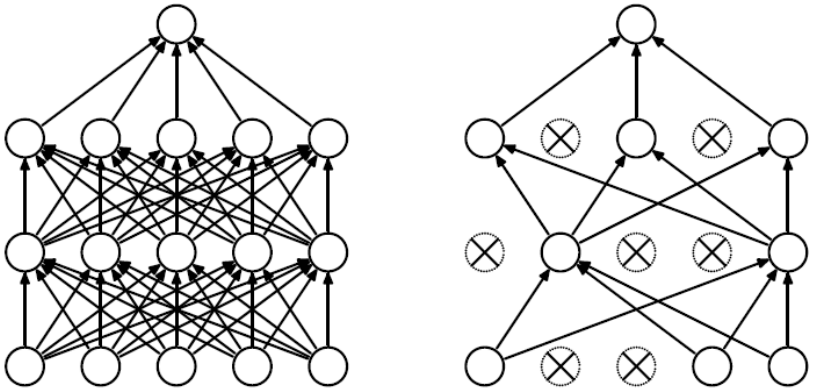
  - به آن weight decay هم می‌گویند
  - اثر کاهش وزن به صورت W += -λ* W به سمت صفر

- تنظیم L1

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

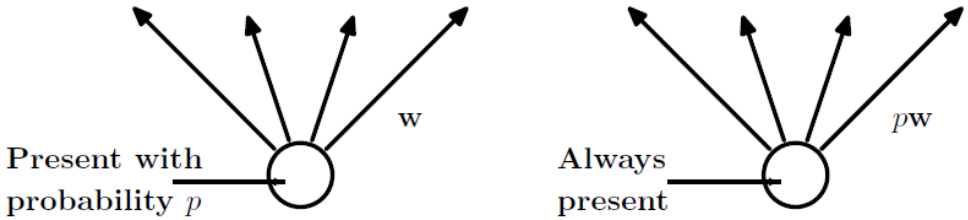
یادگیری عمیق: تنظیم ...

- 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 ...

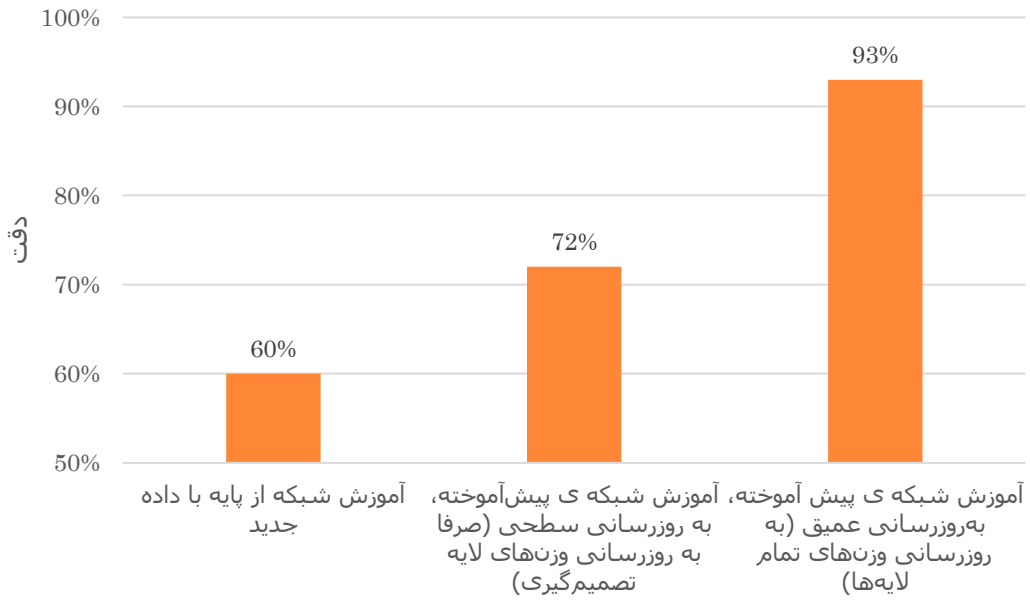
یادگیری عمیق: پیش آموزش . . .

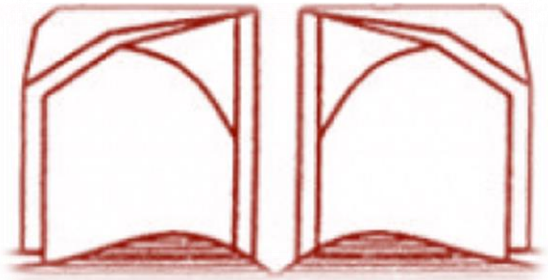
- **ایده پیش آموزش (Pre-Train)**
  - **آموزش شبکه با داده (حجیم) از یک حوزه و انتقال وزن‌های آموزش دیده برای استفاده از حوزه دیگر (به عنوان وزن‌های اولیه)**
    - انتقال وزن‌های مربوط به لایه‌های استخراج ویژگی
    - صرفنظر کردن از لایه دسته بندی
  - **بهینه کردن وزن‌های انتقال یافته با داده حوزه جدید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





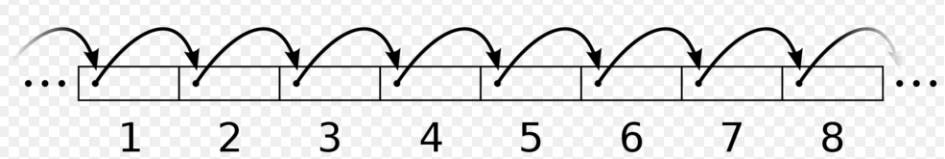
یادگیری ماشین

شبکه عصبی بازگشتی

شبکه عصبی بازگشتی . . .

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

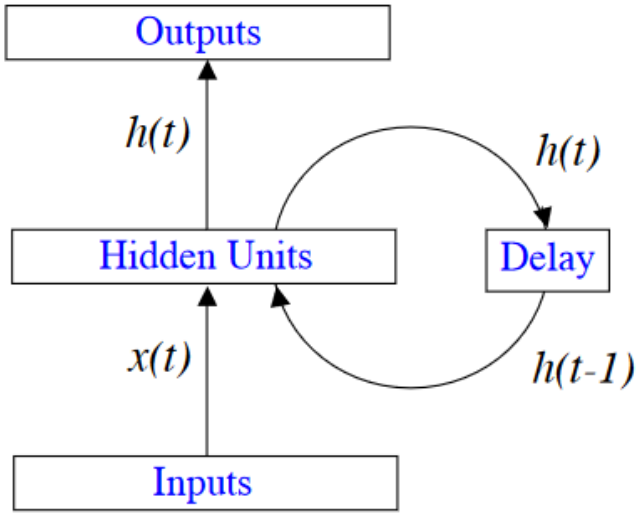
- فریم‌های (نمونه‌های) سیگنال گفتار
- فریم‌های (تصاویر) متوالی ویدئو
- وضعیت آب و هوا
- قیمت سهام یک شرکت/صنعت
- دنباله‌های تولید شده توسط گرامرها
  - کلمات داخل یک متن
- ...



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.

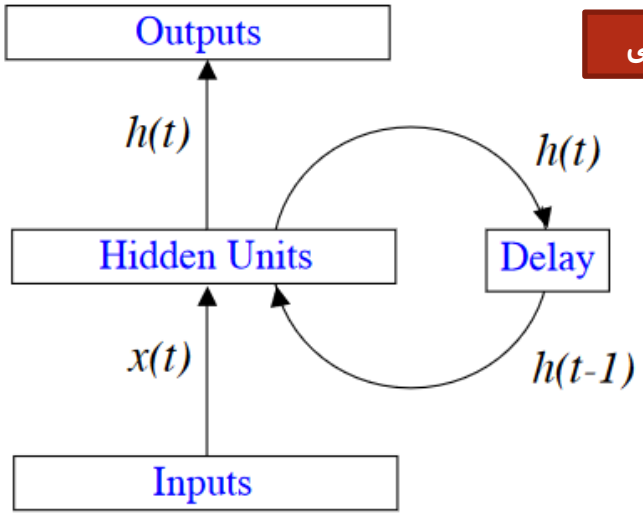
- به دلیل داشتن یال بازگشتی در ساختار خود، قدرت **حافظه**‌ای دارند.
  - مناسب برای پردازش داده‌های متوالی (Sequential Data)



- 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

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

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

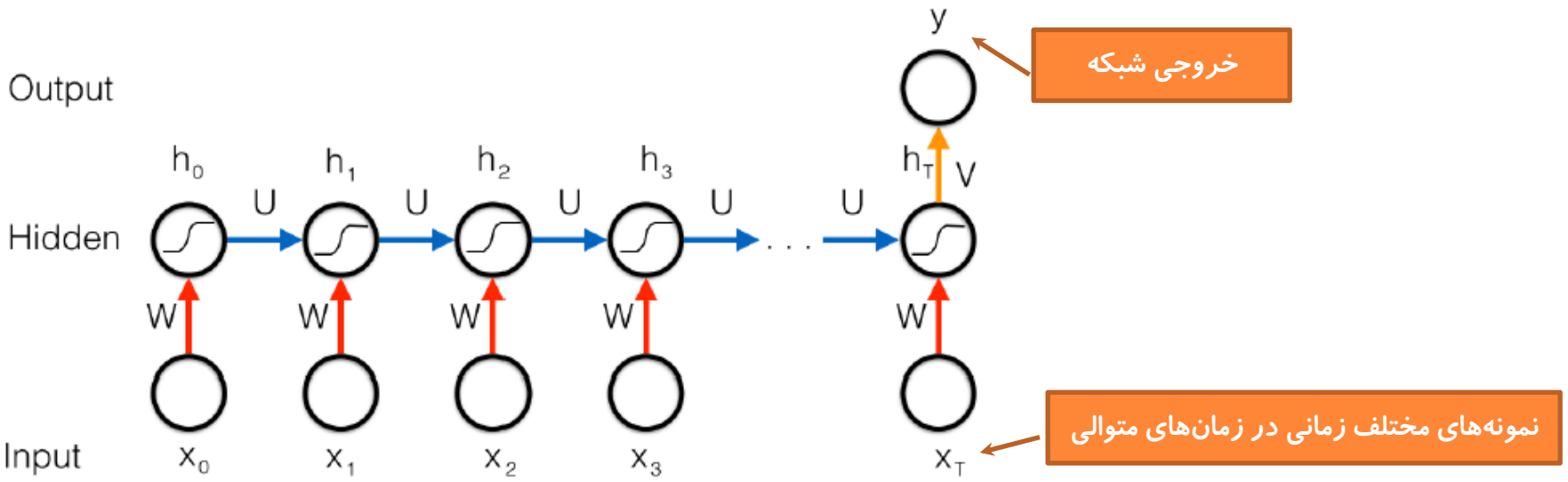
فعال سازی لایه خروجی

وزن لایه مخفی به خروجی

شبکه عصبی بازگشتی: آموزش ...

عملکرد شبکه

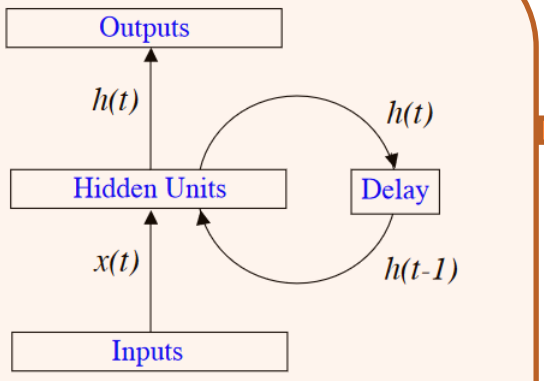
- وابستگی خروجی شبکه به خروجی‌های لایه مخفی به ازای همه ورودی‌ها



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

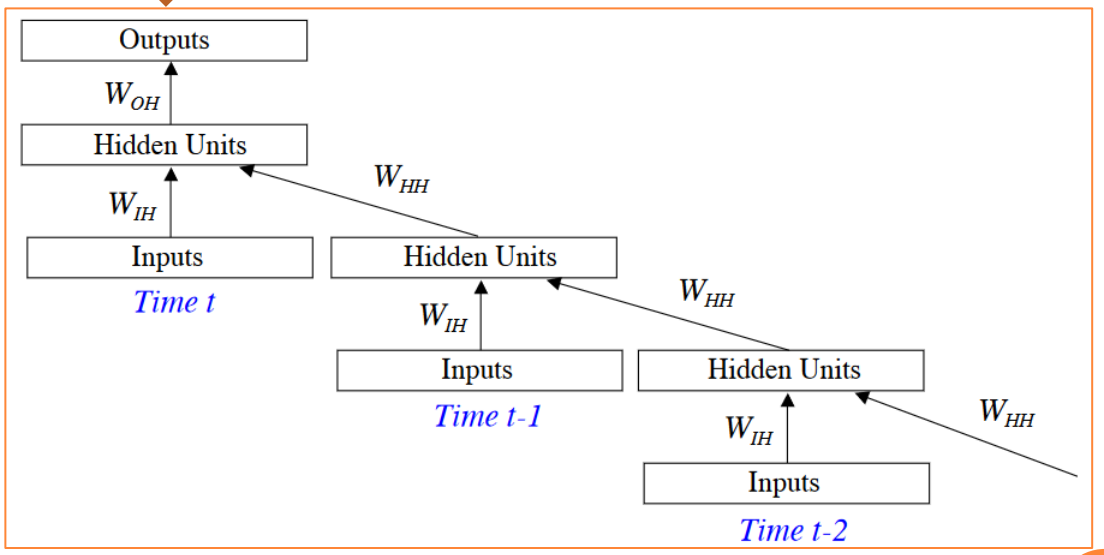
شبکه عصبی بازگشتی: آموزش ...

- 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: 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

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

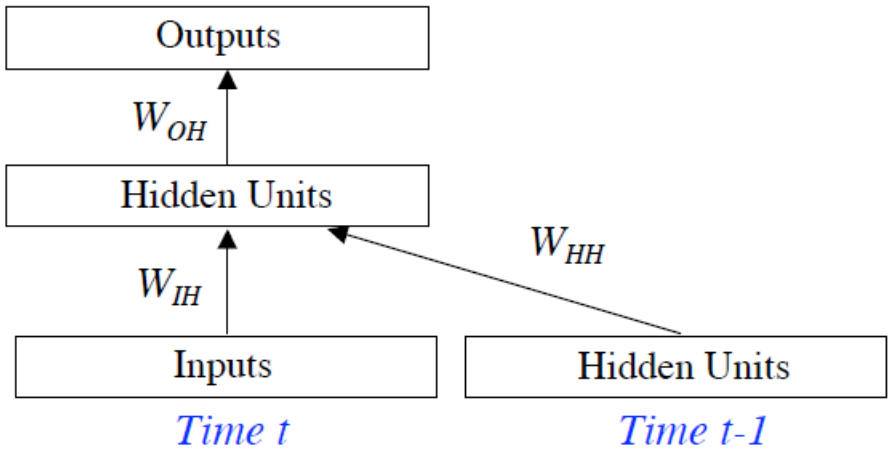
  - گرادیان وزن‌های شبکه برای بدست آوردن مقدار تغییرات وزن

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

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

شبکه عصبی بازگشتی: آموزش ...

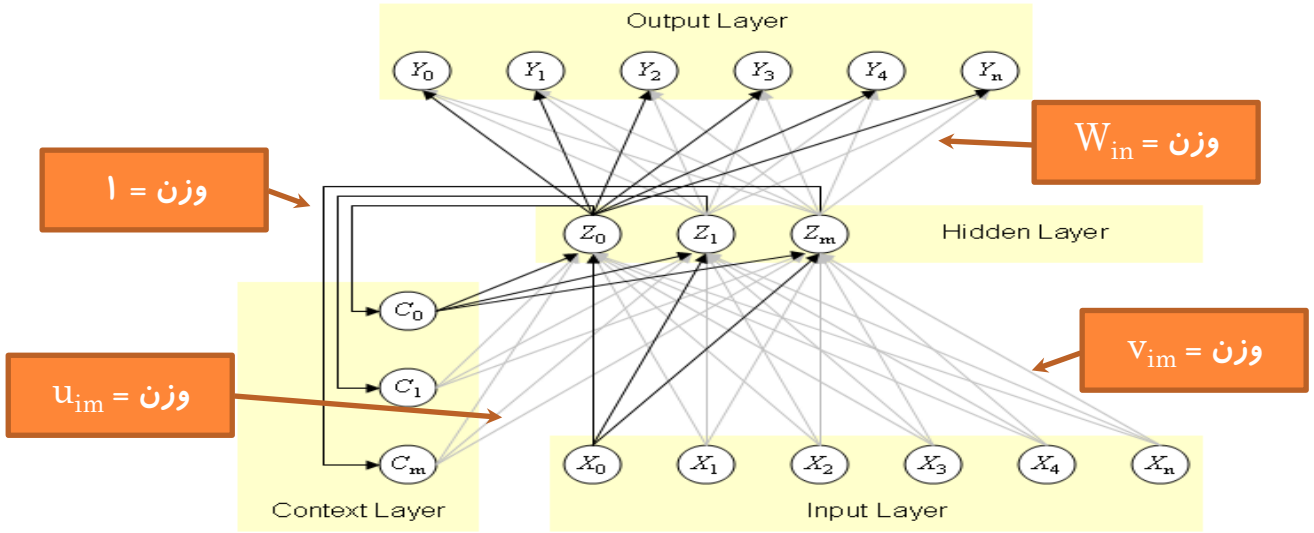
- **استفاده از الگوریتم پس‌انتشار**
  - استفاده از این روش نیازمند نگهداری حالت‌های قبلی شبکه (خروجی لایه مخفی) و کلیه ورودی‌های قبلی
  - در عمل نگهداری همه اطلاعات قبلی مشکل است و تنها از تعداد محدودی از آنها (مثلا ۳۰ مقدار قبلی) استفاده می‌شود = truncation
  - حالت ساده = نگهداری فقط یک مرحله قبل =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



شبکه عصبی 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

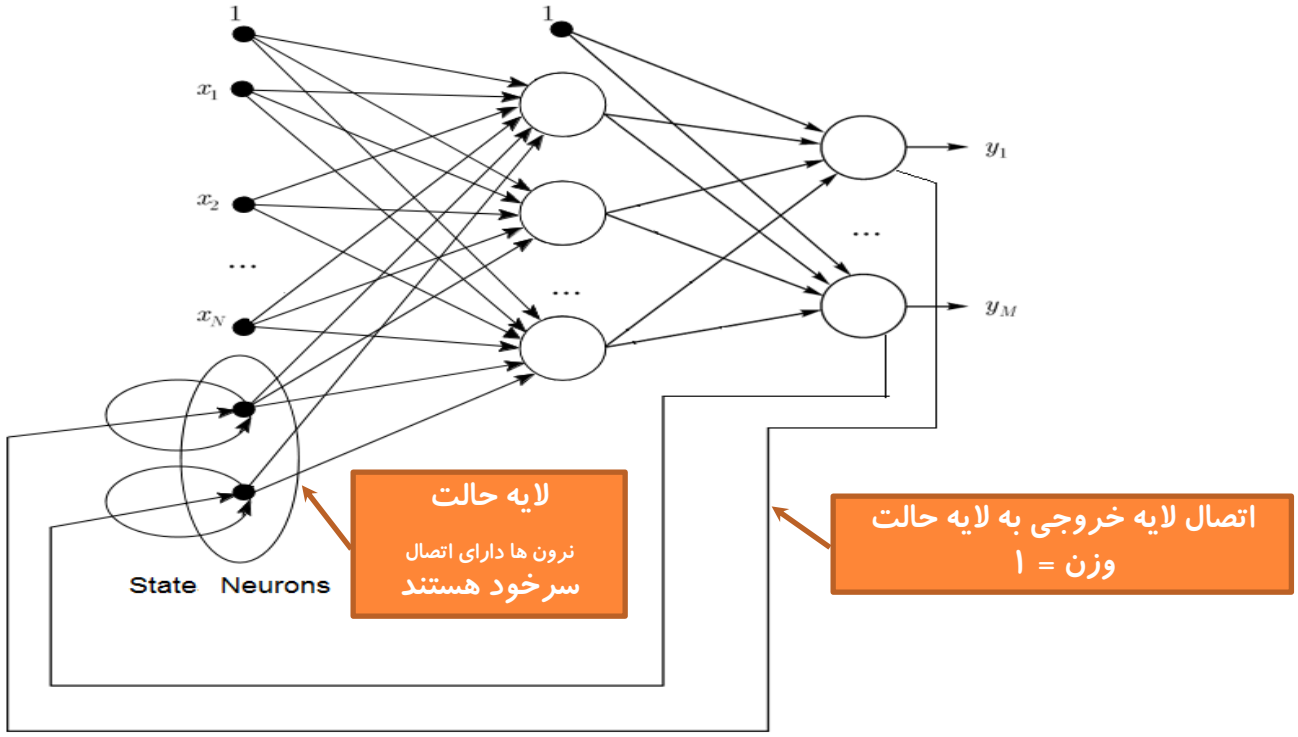
- مراجعه کنید به

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

شبکه عصبی جردن ...

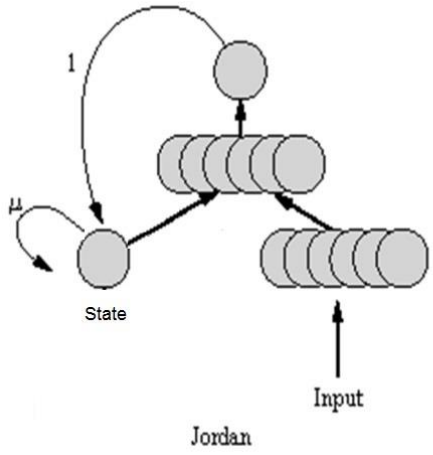
معرفی و ساختار

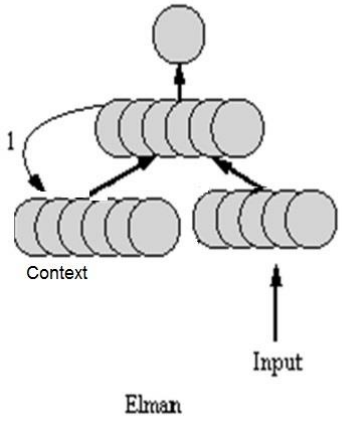
- ارائه شده در سال ۱۹۹۶ توسط مایکل جردن
- دارای شباهت بسیار زیاد به شبکه عصبی المان
- شبکه دارای اتصالات بازگشتی از لایه خروجی به لایه حالت و همچنین از لایه حالت به خودش می‌باشد



شبکه عصبی جردن: تفاوت با شبکه المان

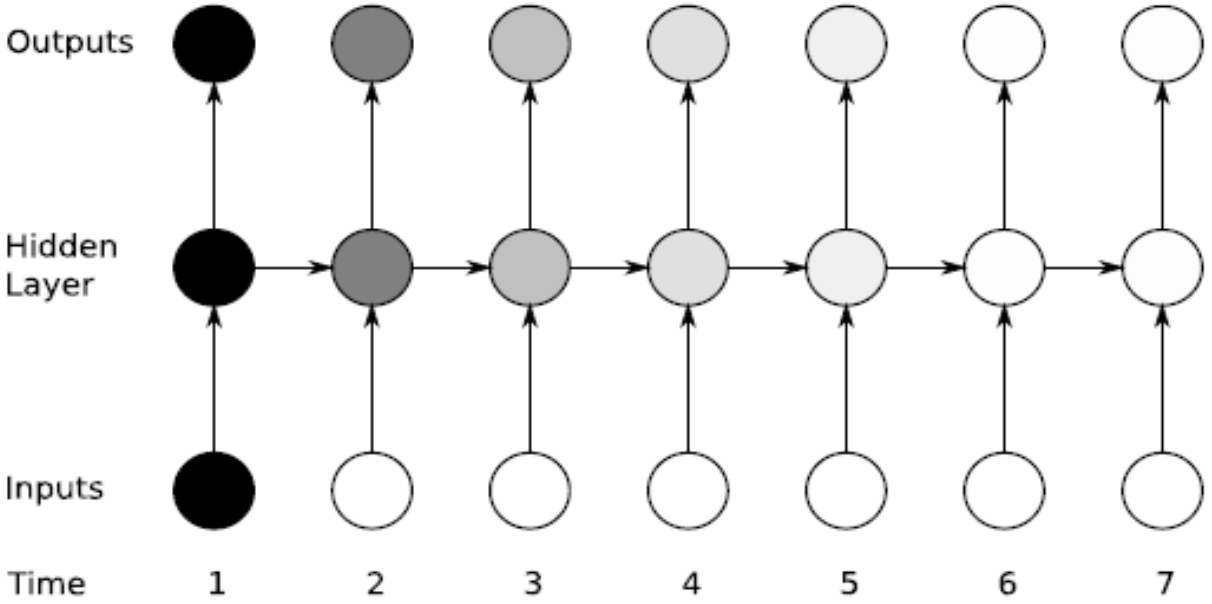
- در شبکه جردن
  - اتصالات بازگشتی به جای لایه پنهان از لایه خروجی شروع می‌شود (با وزن ثابت یک)
  - در این شبکه لایه بافت، لایه حالت (State Layer) نامیده می‌شود
  - لایه حالت شامل اتصالات بازگشتی از خودش به خودش با وزن ثابت می‌باشد
  - تعداد نرون‌های لایه حالت با تعداد نرون‌های لایه خروجی برابر است
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





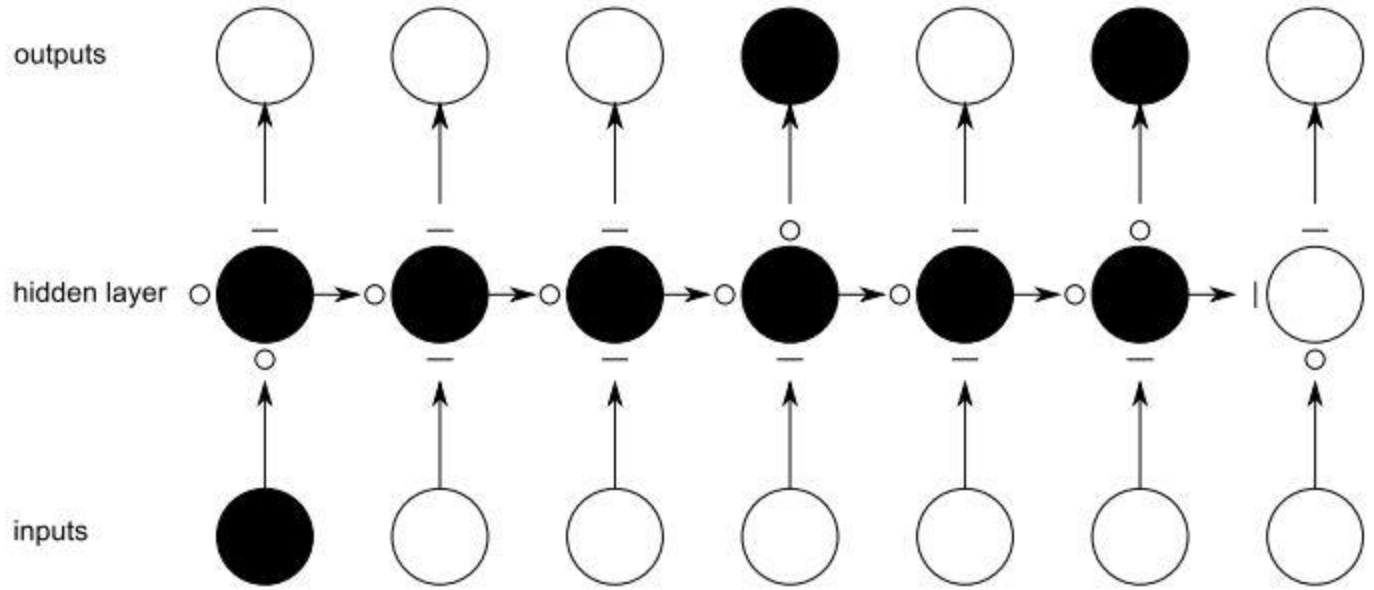
مشکل فراموشی در شبکه‌های عصبی بازگشتی . . .

- 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



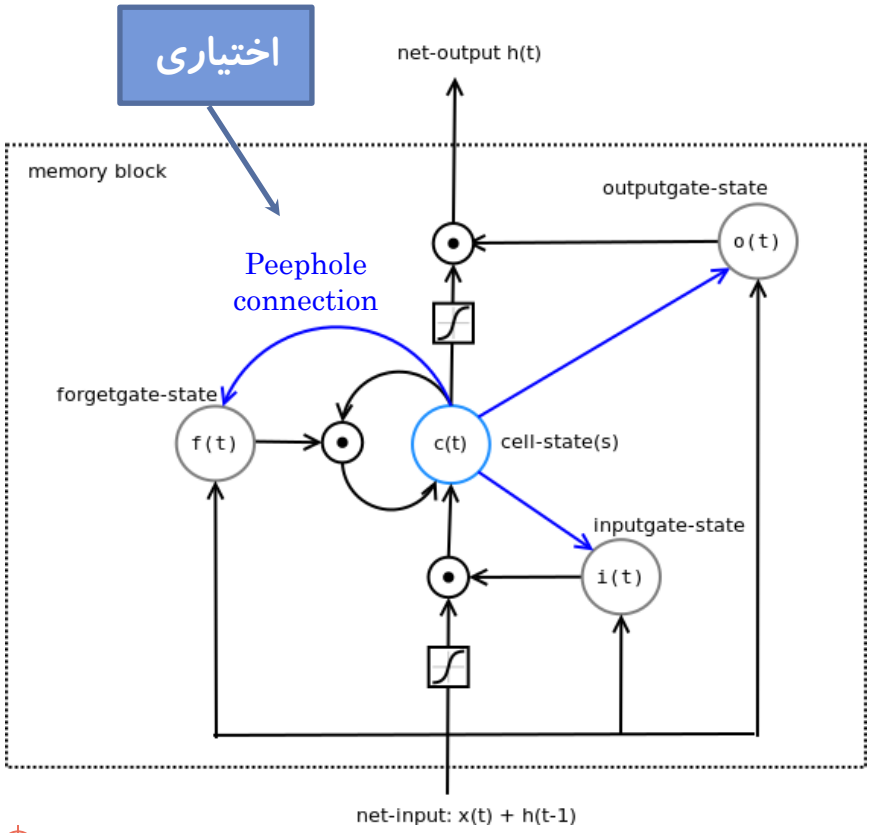
مشکل فراموشی در شبکه‌های عصبی بازگشتی

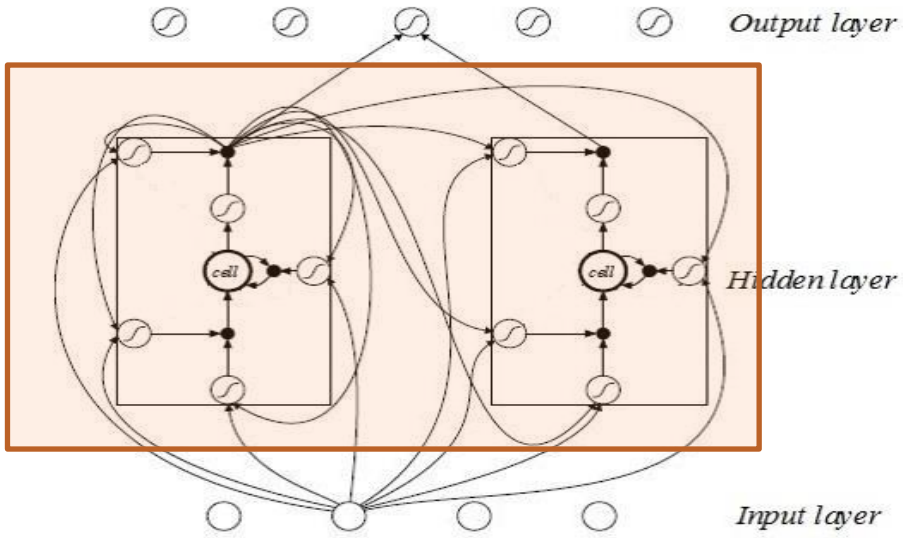
- حل مشکل فراموش
  - کنترل ورود و خروج داده در واحدهای لایه میانی شبکه



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

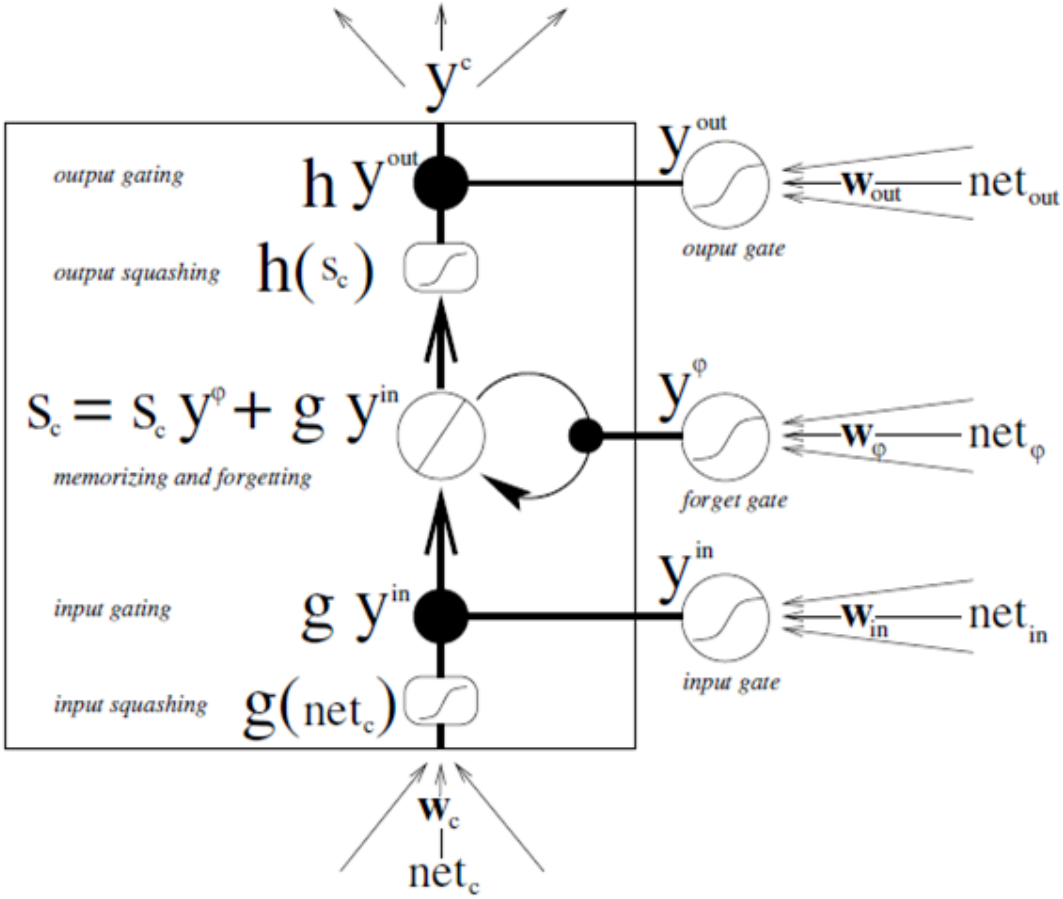
- شبکه Long Short Term Memory (LSTM)
  - نرون‌های لایه پنهان با بلوک‌های حافظه جایگزین شدند
    - حل شدن مشکل فراموشی دنباله‌های طولانی





شبکه عصبی LSTM: ساختار بلوک حافظه ...

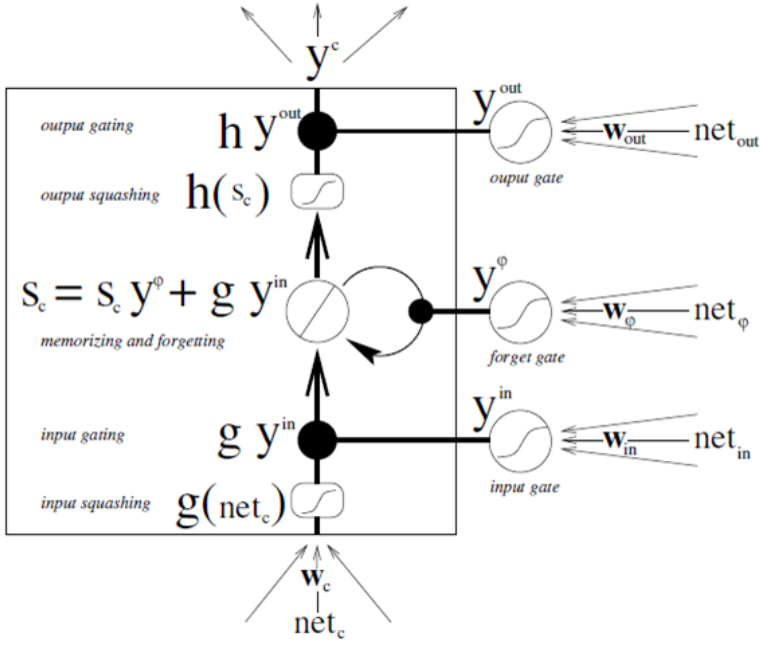
- هر بلوک حافظه، شامل
  - سلول
    - برای ذخیره اطلاعات در بلوک
  - دروازه ورودی
  - دروازه فراموشی
  - دروازه خروجی



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

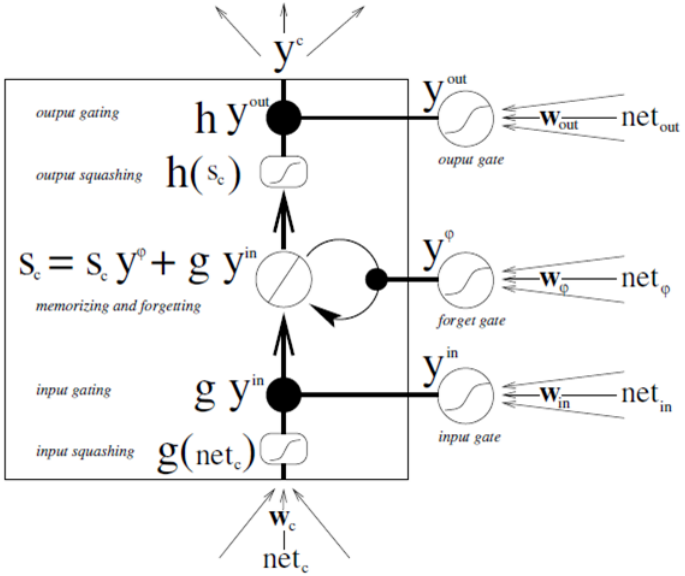
شبکه عصبی LSTM: ساختار بلوک حافظه . . .

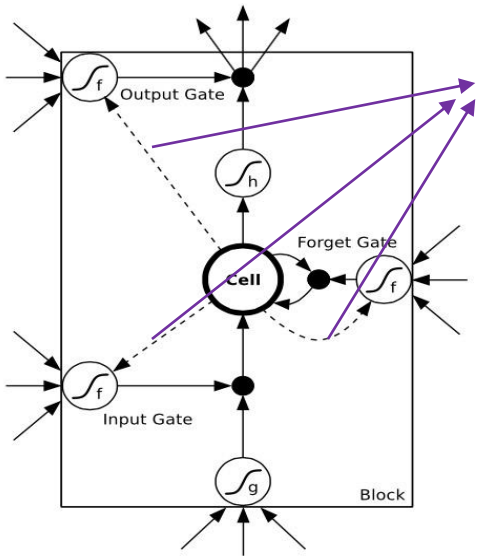
- وظیفه دروازه‌ها: کنترل ورود و خروج داده‌ها و پاک کردن حافظه‌ی بلوک
  - فعال‌ساز دروازه‌ها مقداری بین صفر و یک می‌گیرند.
  - در صورتی که دروازه کاملا باز باشد فعال‌ساز آن برابر یک و در صورتی‌که کاملا بسته باشد فعال‌ساز آن برابر صفر است
- هر سلول حافظه در مرکز خود یک واحد به نام CEC دارد که به فعال‌سازی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

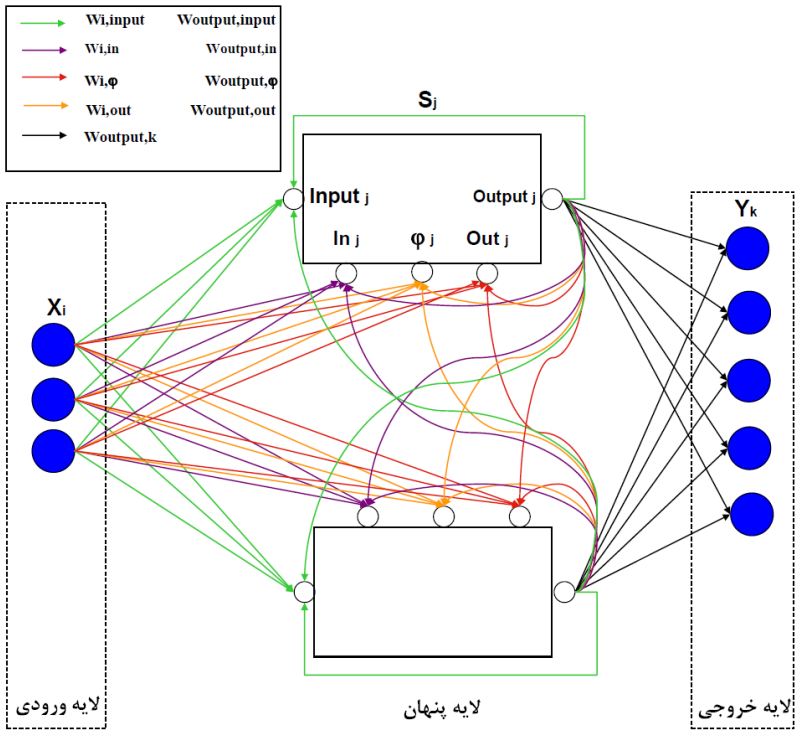




Peephole connections

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

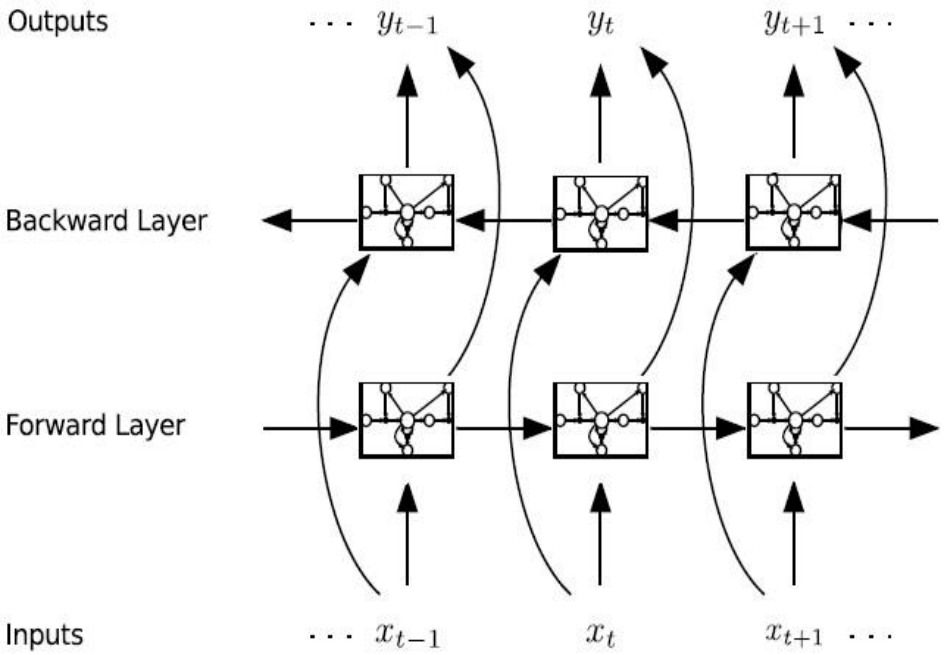
- مراجعه کنید به

- 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.



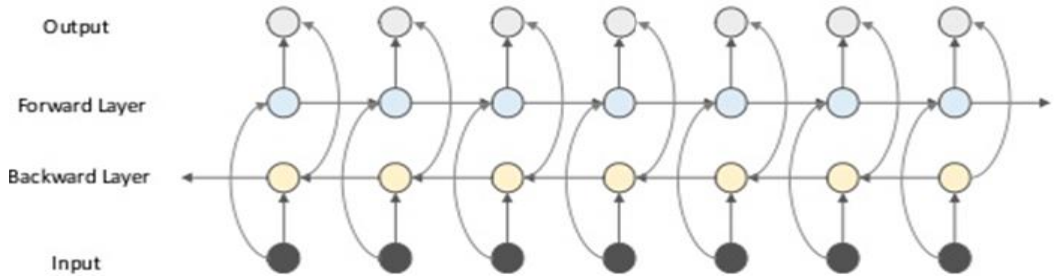
شبکه‌های عصبی LSTM دوطرفه (Bidirectional)

ایده اصلی

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

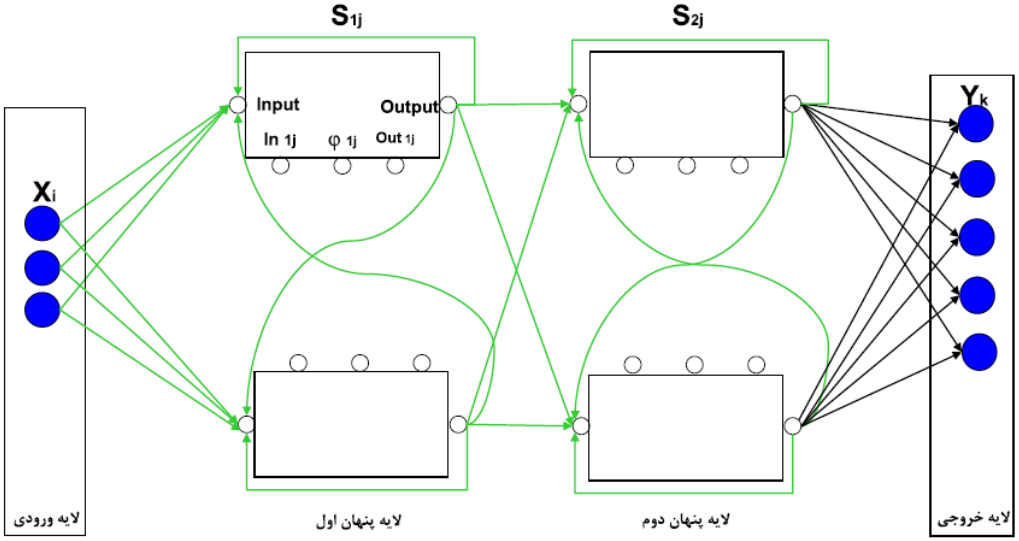
- فرض کنید دنباله آموزشی به صورت $\boldsymbol{X^T} = (x_1, x_2, \dots, x_{T-1}, x_T)$ و دنباله هدف متناظر برابر $\boldsymbol{t^T} = (t_1, t_2, \dots, t_{T-1}, t_T)$ باشد
- در هر مرحله بردار x_i را به لایه Forward و $x_{T-(i-1)}$ را به لایه Backward ارسال کرده و مقدار هد ف را بردار t_i قرار می‌دهیم.
-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

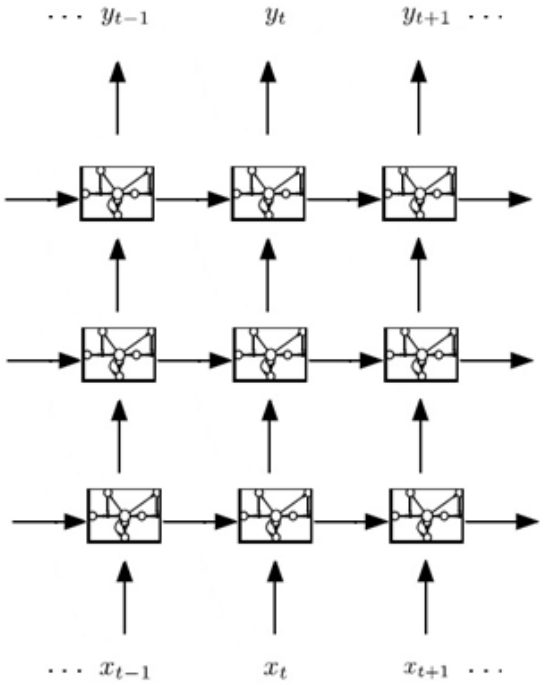
مقدار خالص رسیده به لایه خروجی جمع وزن‌دار مقدار خالص دو لایه Forward وBackward است



شبکه‌های عصبی LSTM: عمیق

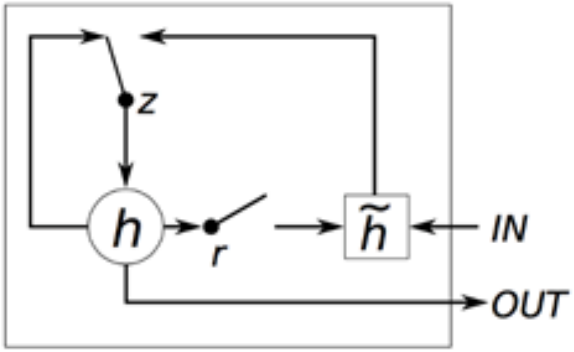
- شبکه عصبی با بیش از یک لایه مخفی
  - خروجی هر لایه پنهان ورودی لایه پنهان بالاتر
  - تقریب زننده جهانی
  - قابلیت یادگیری بیشتر



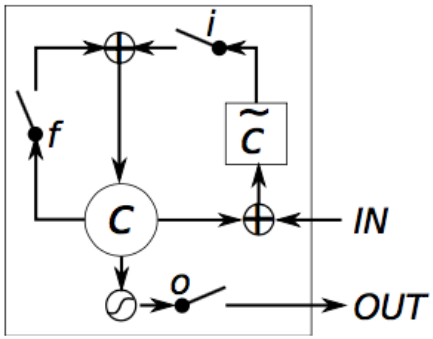


شبکه عصبی واحد بازگشتی دروازه‌ای(GRU)

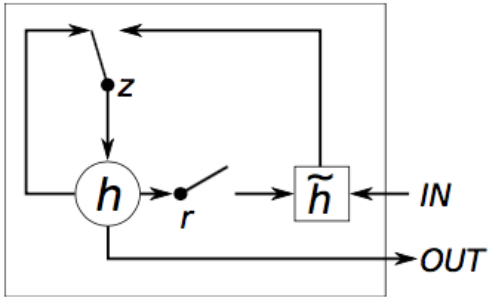
- ساده شده LSTM استاندارد
  - عدم وجود دروازه خروجی



- 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

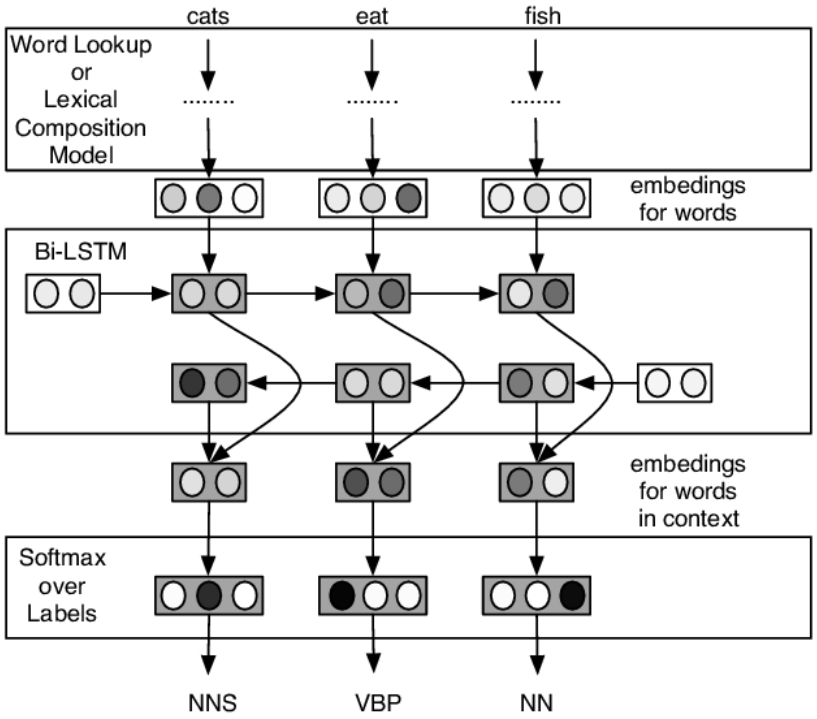


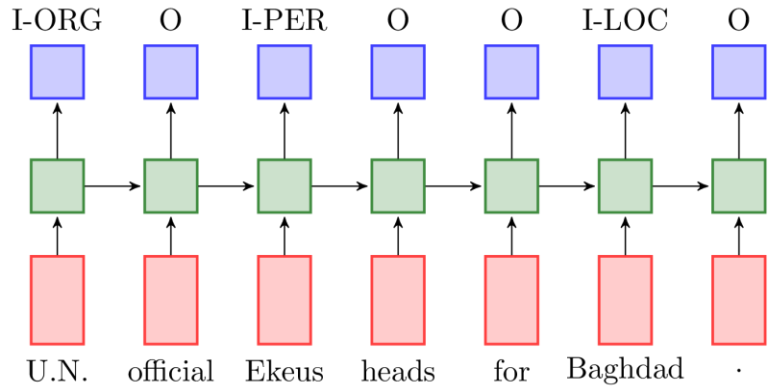
(b) Gated Recurrent Unit

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.

شبکه عصبی LSTM: کاربردها . . .

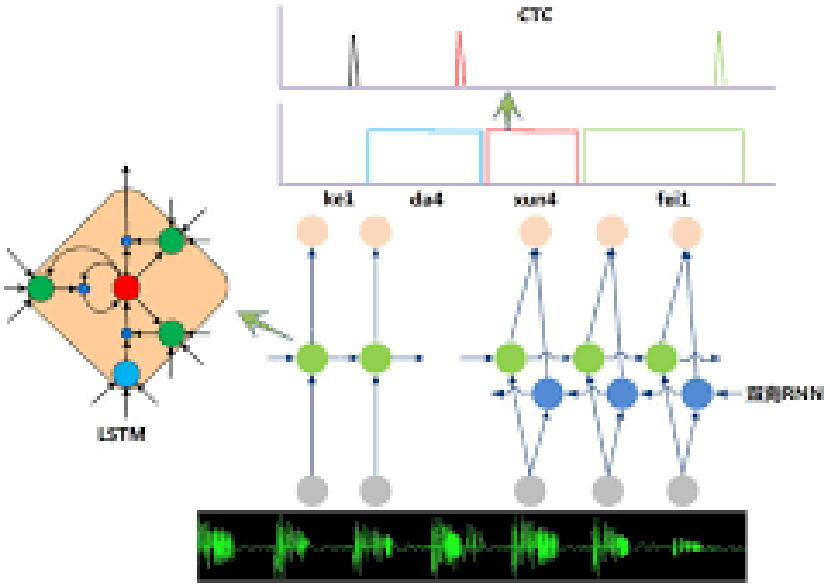
- برچسب زنی اجزای کلام (POS)/بازشناسی پدیده‌های اسمی (NER)
  - ورودی: بردار یک کلمه (مثل Word Vector)
  - خروجی: به تعداد برچسب‌ها (هر نرون یک برچسب)





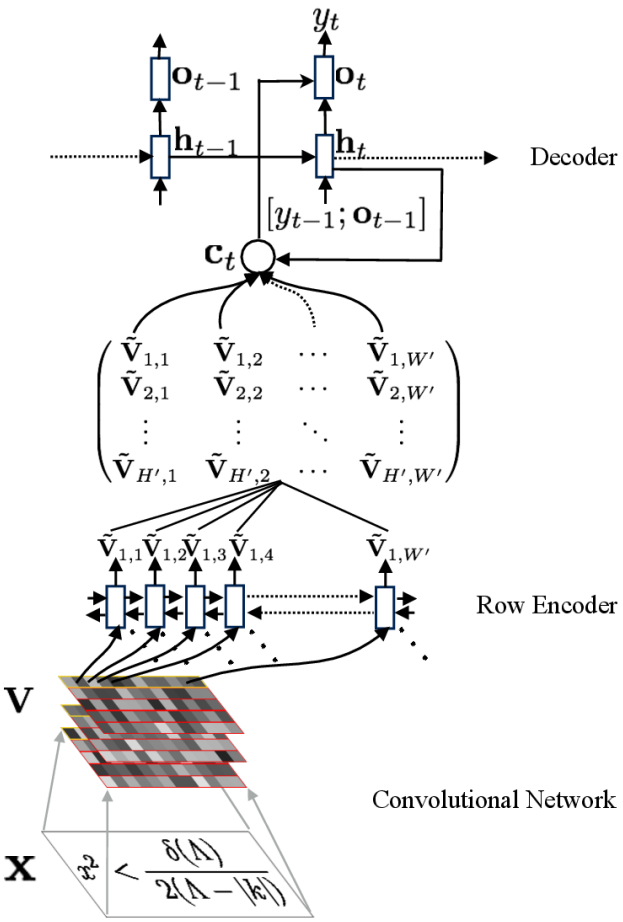
شبکه عصبی LSTM: کاربردها . . .

- 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



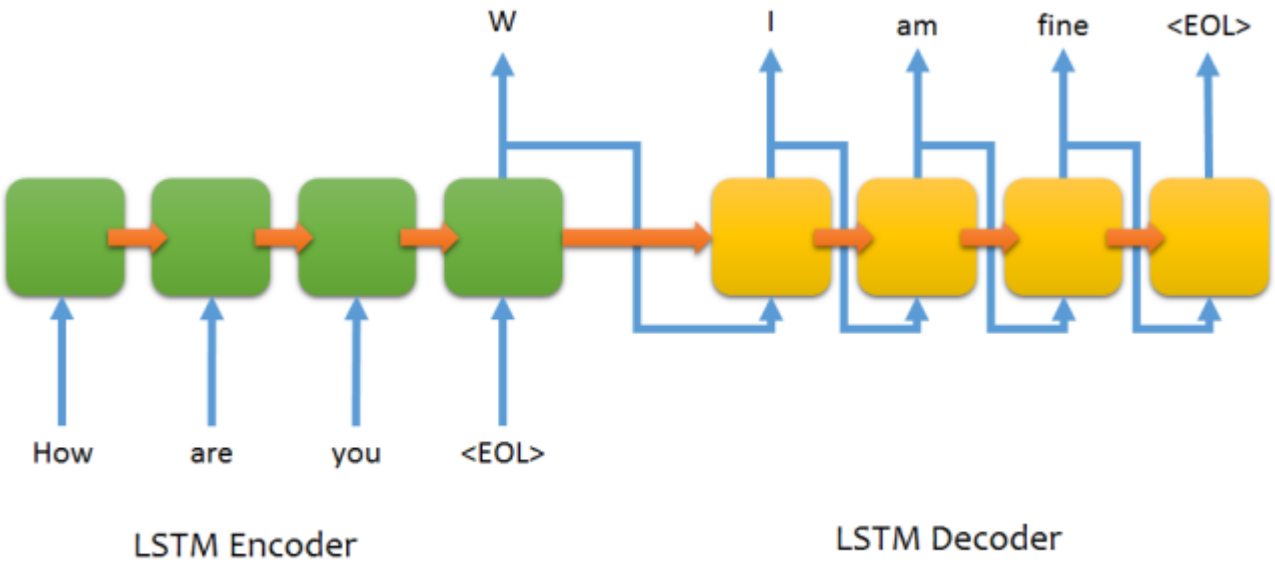
شبکه عصبی LSTM: کاربردها . . .

- 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 . . .

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



شبکه عصبی LSTM: کاربردها

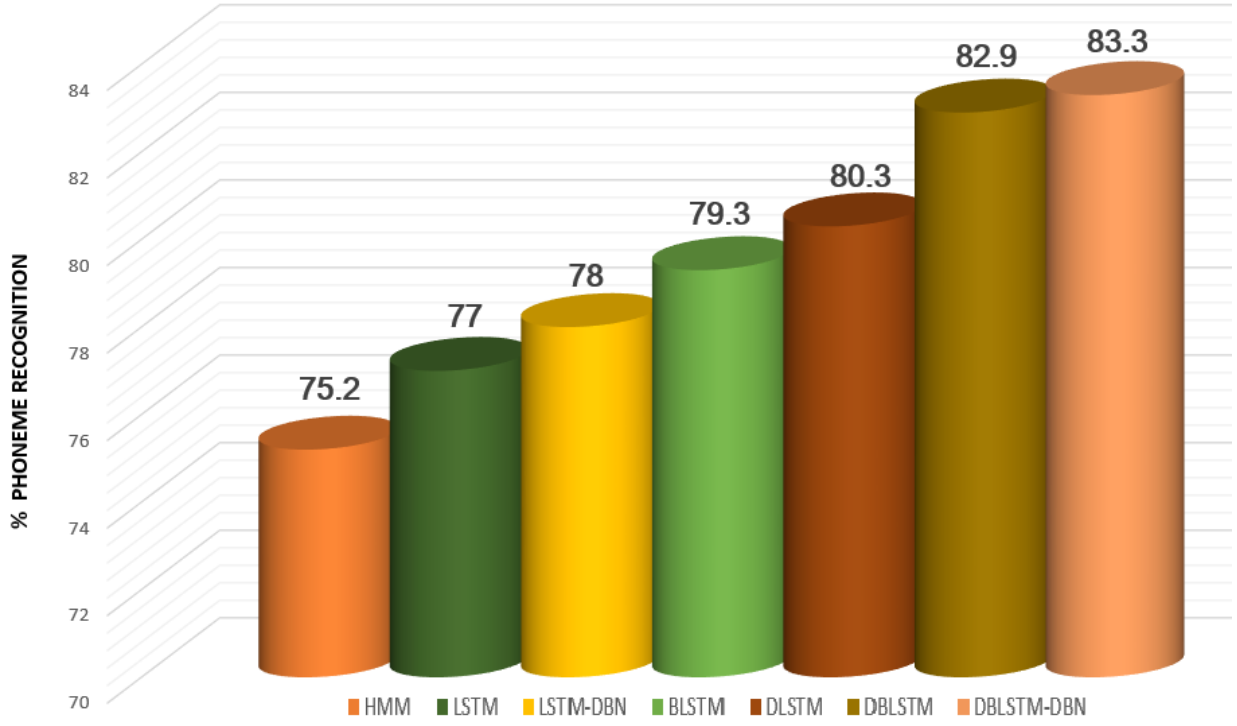
روشی غالب در همه (!) کاربردهای مدل‌سازی دنباله

شبکه عصبی LSTM: تشخیص واج‌های فارسی . . .

- **داده‌ها: 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

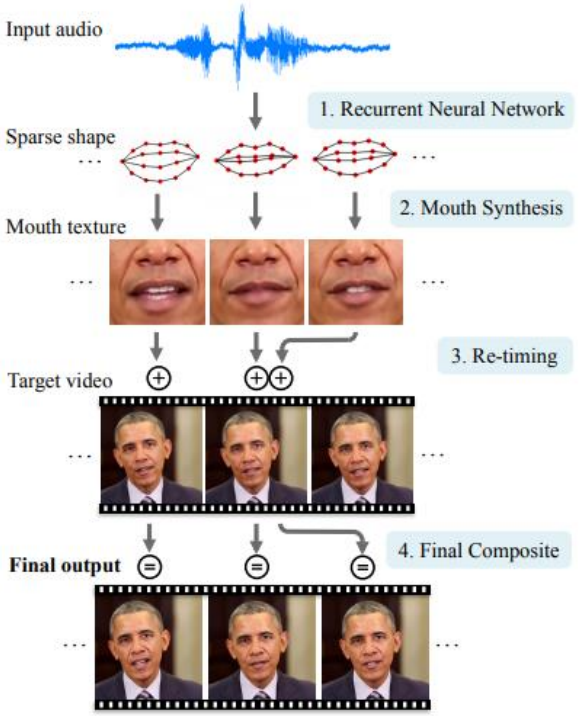
شبکه عصبی LSTM دوطرفه: تشخیص واج‌های فارسی

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



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]

بسترهای یادگیری عمیق . . .

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

کلاس استاد شفیعی کدکنی- دانشگاه تهران