مدرسه پردازش و تحلیل داده دقیقه

شبکه‌های عصبی مصنوعی

Artificial Neural Networks

سعید مجیدی

شبکه‌های عصبی عمیق

پاییز - زمستان ۱۴۰۴

راهنمای نمادهای ریاضی استفاده شده

- **حرف کوچک و ساده:** یک عدد حقیقی

$$x = 2$$

- **حرف کوچک و بولدشده:** یک بردار عمودی

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$$

بردار $\boldsymbol{w}$

$$\boldsymbol{w}^T = [w_0,\ w_1,\ \ldots,\ w_m]$$

ترانهاده بردار $\boldsymbol{w}$

- **حرف بزرگ و بولدشده:** یک ماتریس

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$$

ماتریس $\boldsymbol{A}$

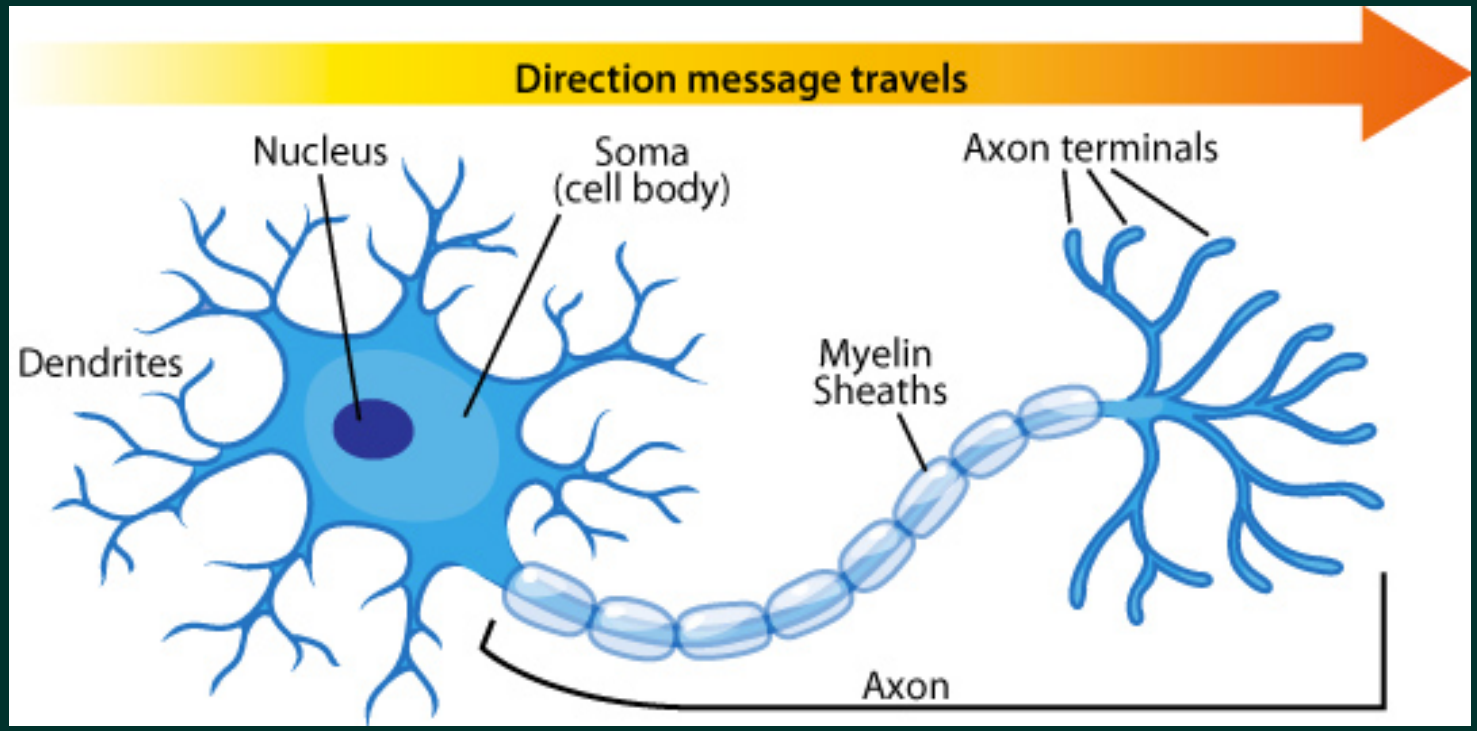
منشا شبکه‌های عصبی

سابقه

- الگوریتم‌هایی برای تقلید از عملکرد مغز
- استفاده بسیار در دهه‌های ۸۰ و ۹۰ میلادی
- کم‌شدن محبوبیت در اواخر دهه ۹۰
- تجدید حیات: افزایش محبوبیت دوباره از اواخر دهه اول قرن جاری با افزایش قدرت محاسباتی و افزایش تولید داده، علی‌الخصوص داده‌های غیرساختاریافته همچون متن، تصویر، ویدیو ...
- هم‌اکنون: پرچمدار روش‌های یادگیری ماشین و هوش‌مصنوعی با کاربردهای فراوان

نورون در مغز انسان

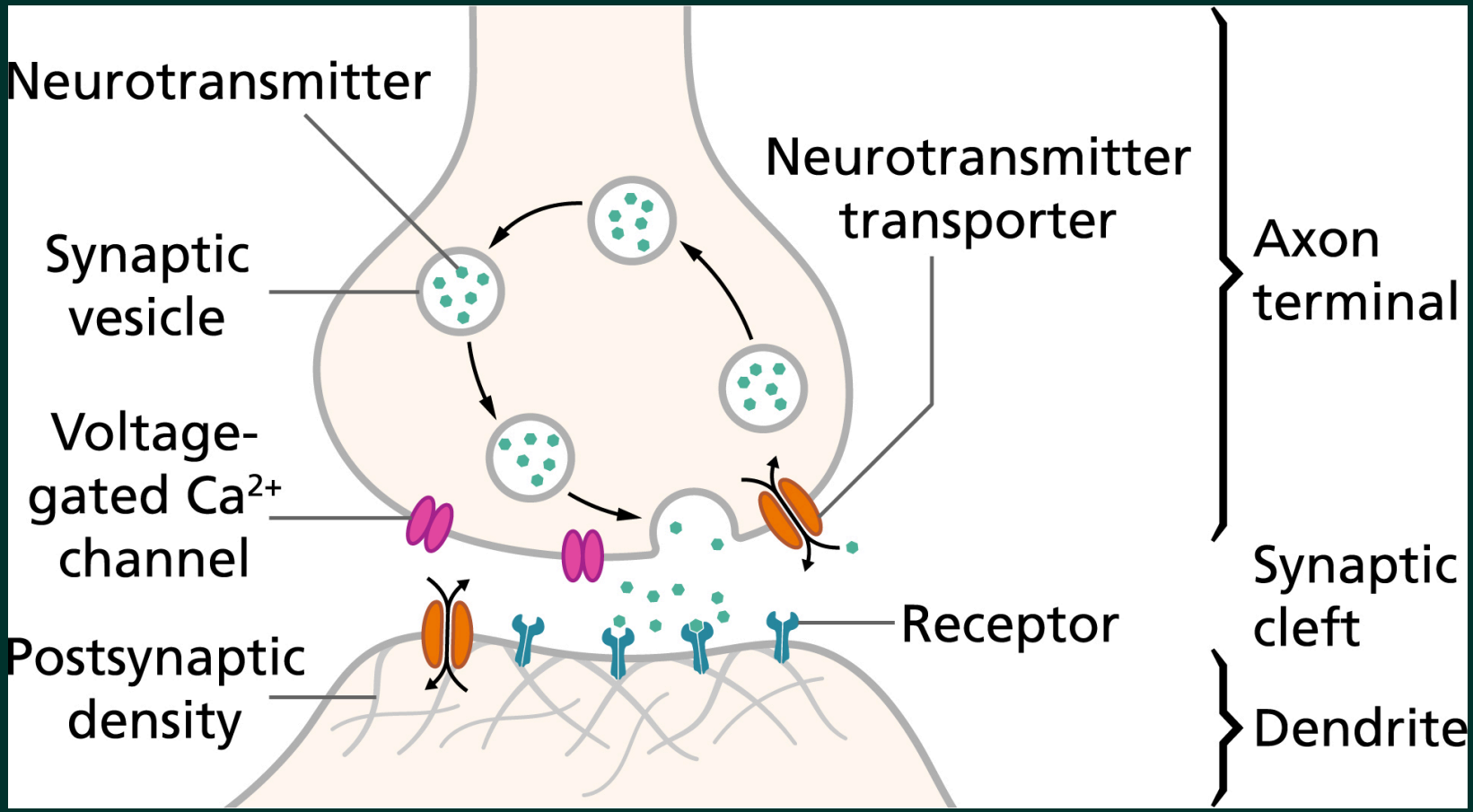
- نورون‌های (Neuron) واحدهای پردازشی در مغز انسان هستند
- مغز انسان از بیش از ۱۰۰میلیارد نورون تشکیل شده است.



- نورون‌ها سیگنال‌های الکتریکی را از دندریت‌ها گرفته و به آکسون ارسال می‌کنند.

ارتباط عصبی

- آکسون یک نورون به دندریت‌های تعداد زیادی از نورون‌های دیگر متصل است.

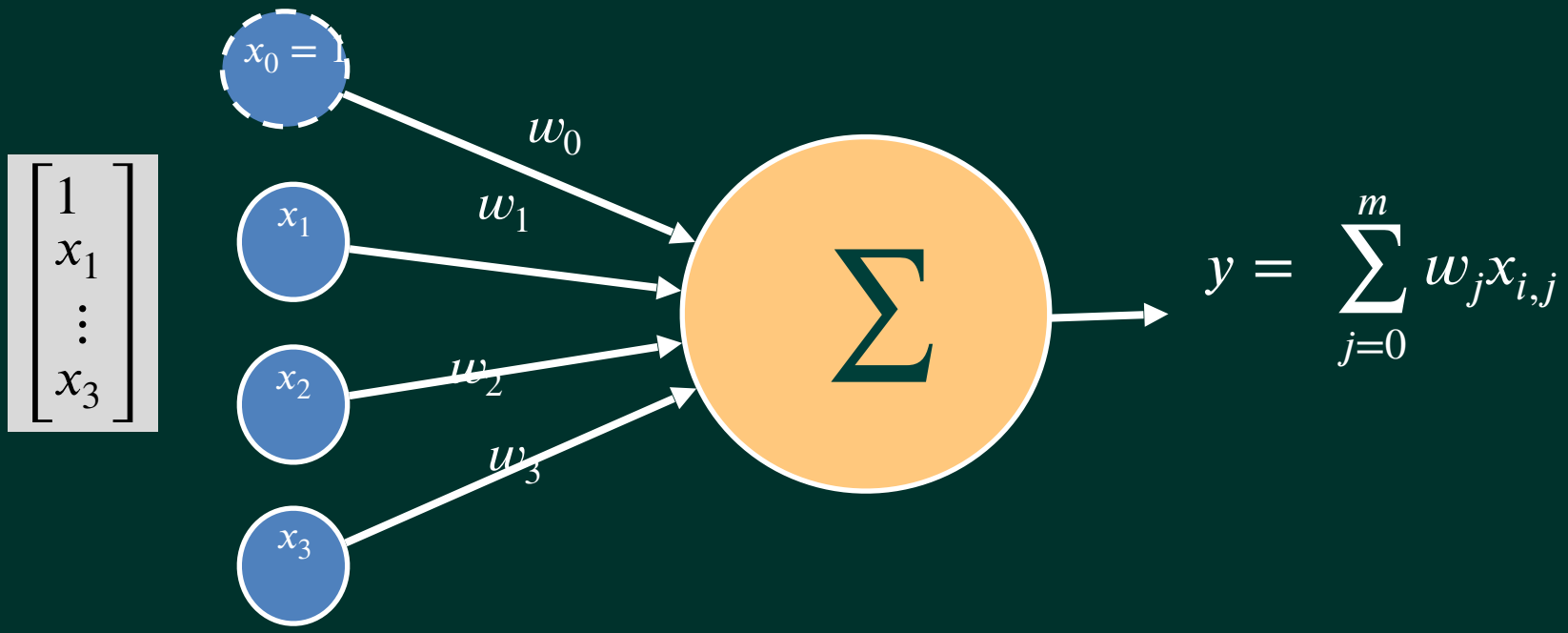


مدل ریاضی

دید کلی

- یک شبکه عصبی در حقیقت یک تابع است!
- هر شبکه عصبی، به‌طور کلی، از اجزای زیر تشکیل شده است:
- نورون‌ها (neurons): مقدار ورودی را از یک تابع عبور داده و نتیجه را در خروجی اعلان می‌کند.
- وزن‌ها: مقادیر بین نورون‌ها را حمل می‌کنند.

مدل ریاضی یک نورون



مدل خطی

- برای ورودی داده شده به شکل: $(x_1, y_1),\ (x_2, y_2),\ \cdots,\ (x_n,\ y_n)$
- هر کدام از x_i ها یک ورودی و هرکدام از y_i ها یک خروجی نامیده می‌شود.
- خروجی، یک ترکیب خطی وزن‌دار شده از ویژگی $x_{i,j}$ و وزن w_j است:

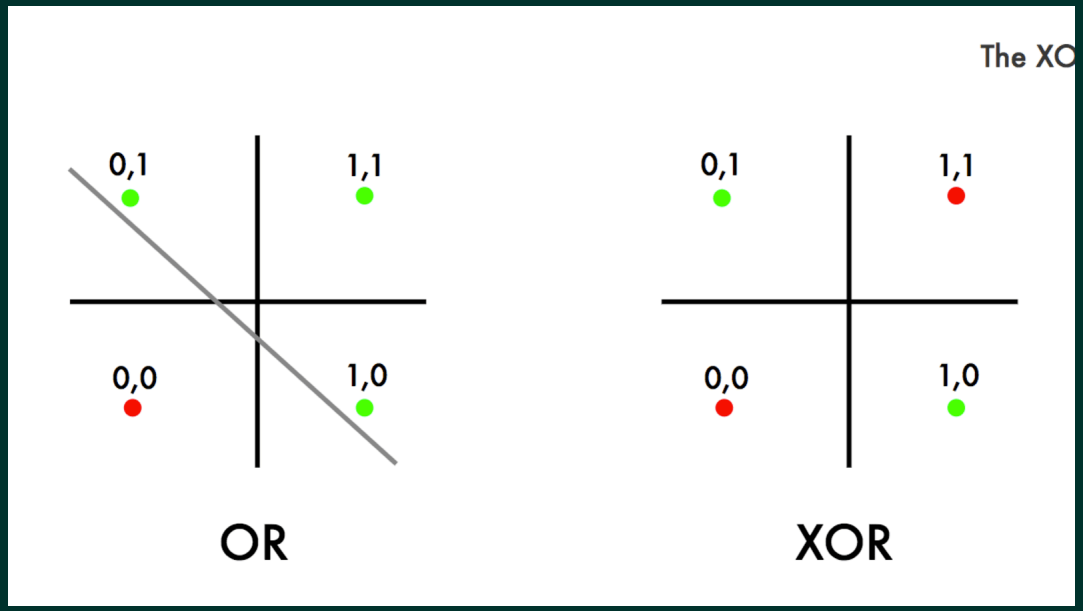
$$y(\boldsymbol{x_i}) = w_0 x_{i,0} + w_1 x_{i,1} + w_2 x_{i,2} + \cdots + w_m x_{i,m}, \qquad x_{i,0} = 1$$

$$y(\boldsymbol{x_i}) = \sum_{j=0}^{m} w_j x_{i,j} = \boldsymbol{w}^T . \boldsymbol{x_i}$$

محدودیت‌های مدل خطی

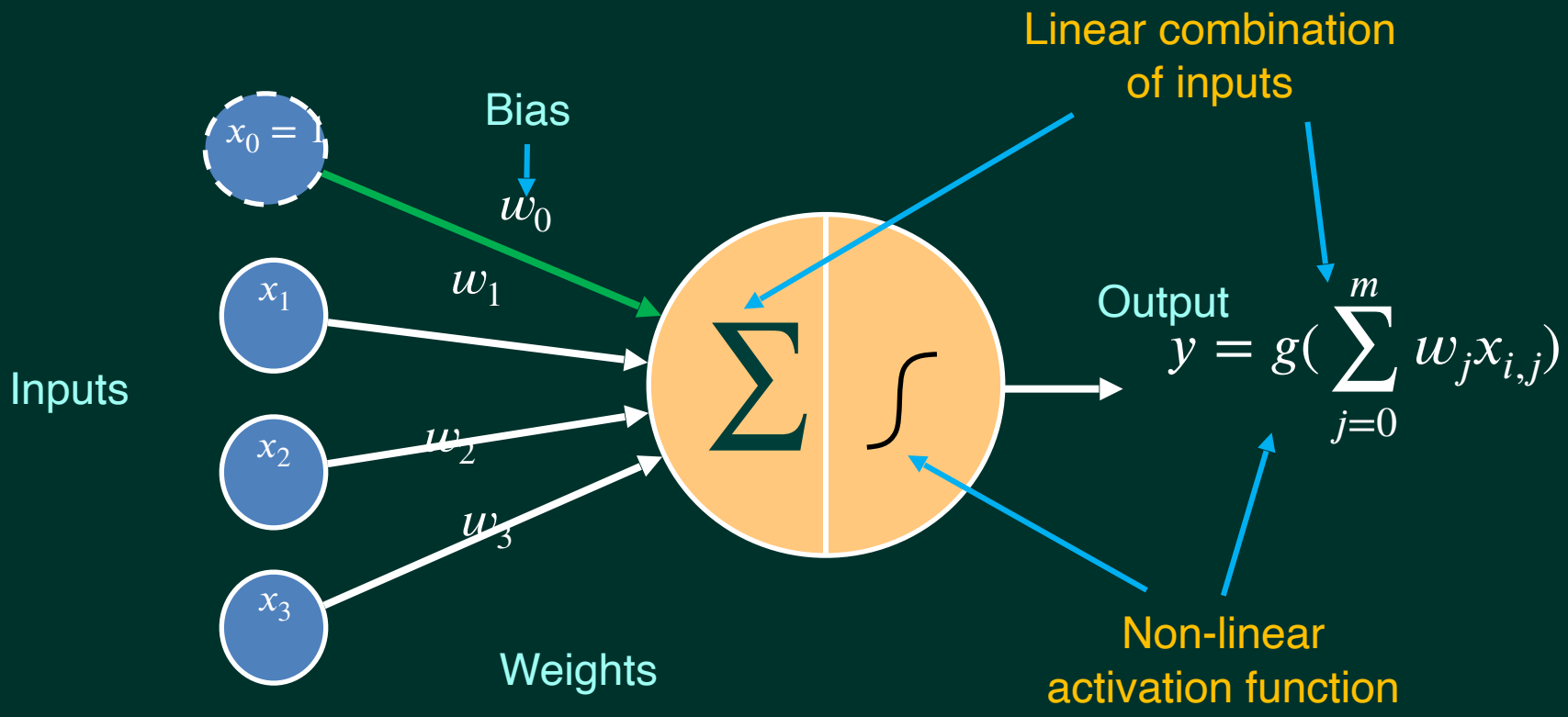
- مدل خطی تمامی توابع را نمی‌تواند مدل کند.
- مثال: تابع "یای انحصاری(XOR) "

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



- غیرخطی‌سازی مدل: استفاده از **توابع فعال‌سازی (Activation Functions)**

اجزای یک نورون غیرخطی

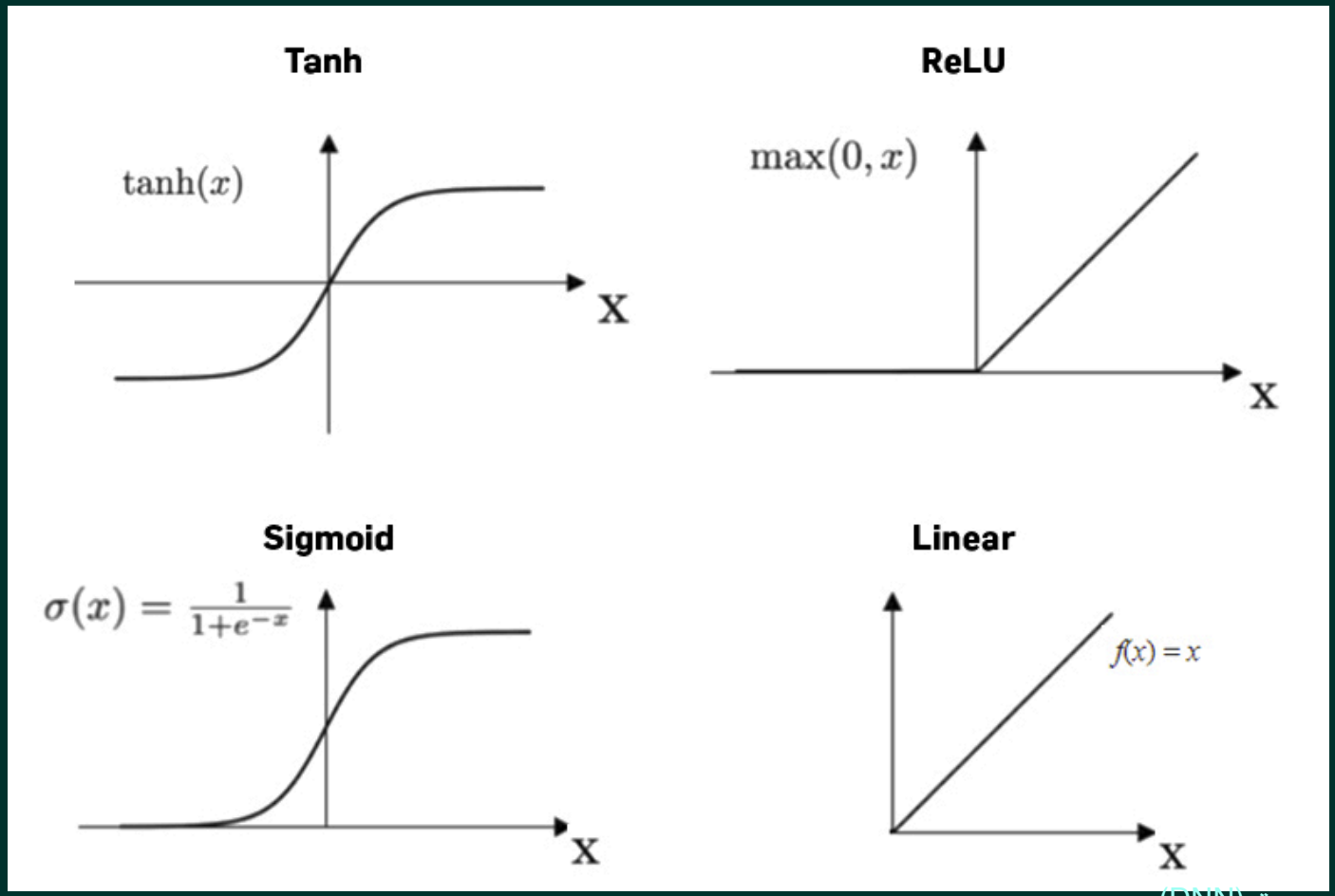


The Perceptron

توابع فعال‌سازی
ACTIVATION FUNCTIONS

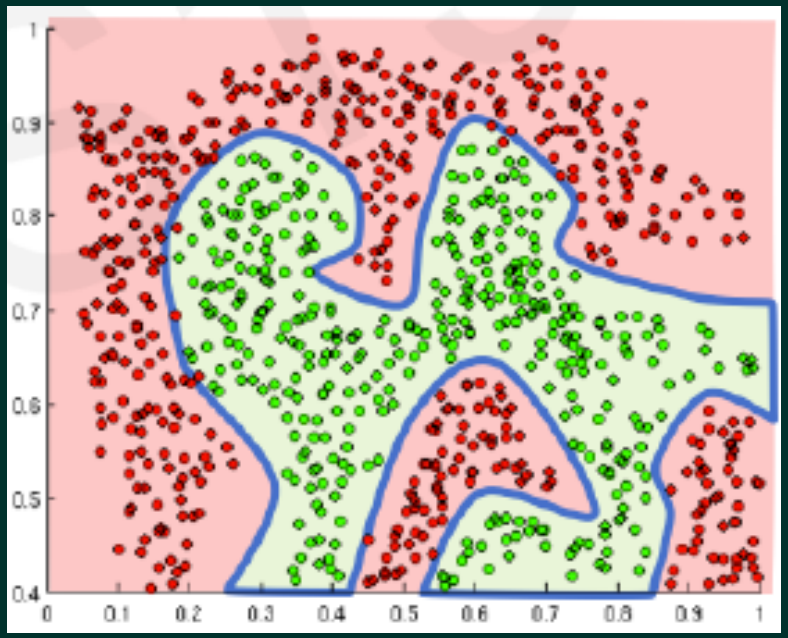
$$y(\boldsymbol{x_i}) = f(\sum_{j=0}^{m} w_j x_{i,j})$$

- ایده: تبدیل مدل خطی به غیرخطی

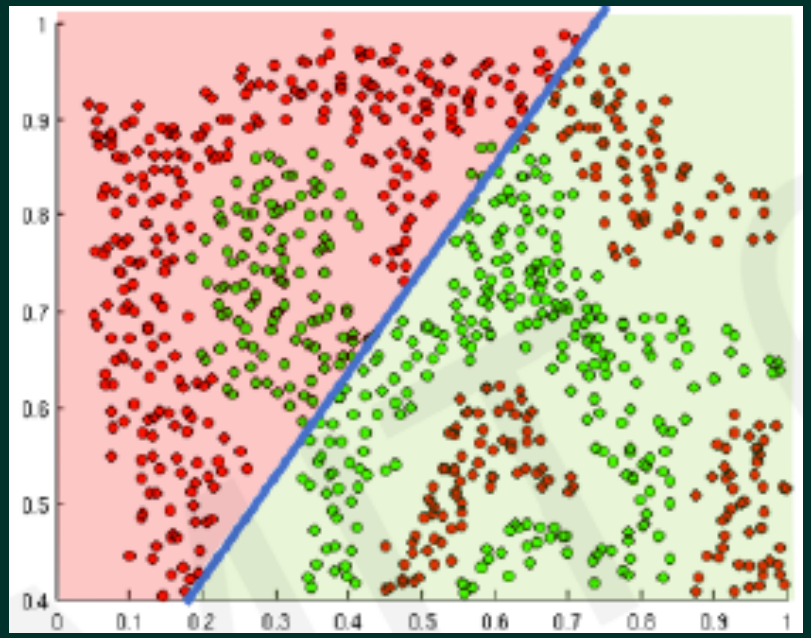


اهمیت توابع فعال‌سازی

- هدف از استفاده از توابع فعال‌سازی، اضافه‌نمودن خاصیت غیرخطی بودن به شبکه است.

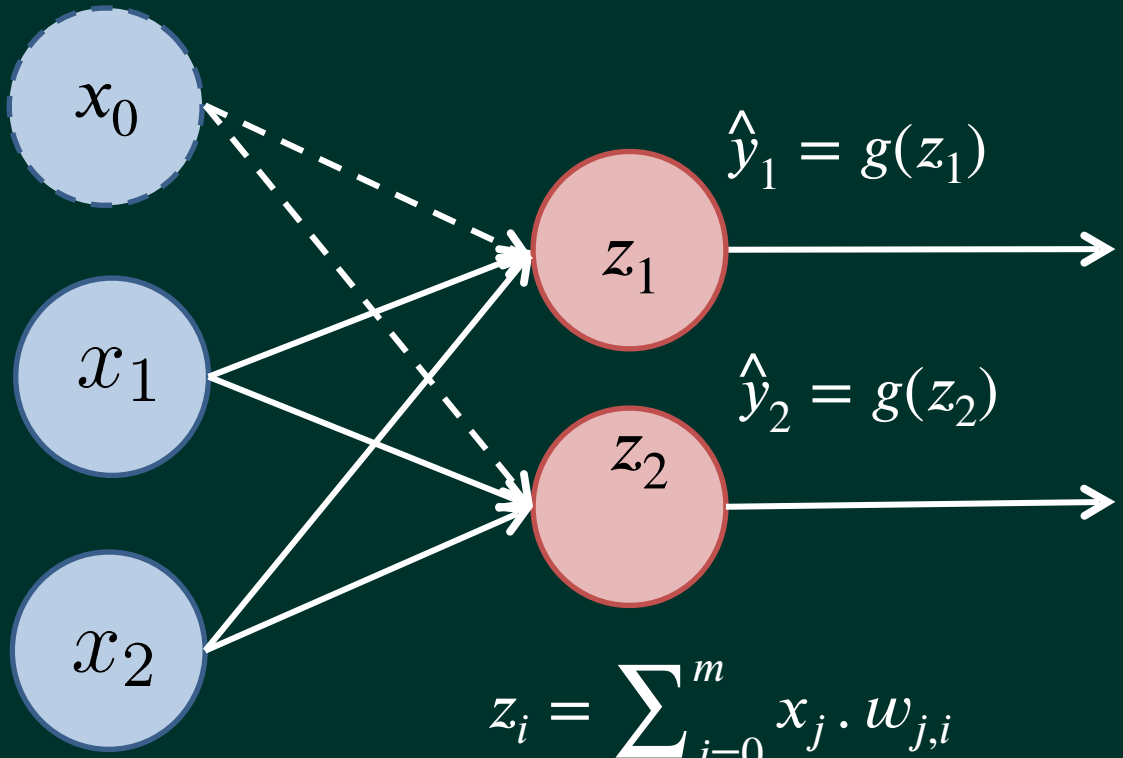


غیرخطی بودن اجازه می‌دهد که توابع پیچیده را به هر میزان تخمین بزنیم.



فارغ از اندازه شبکه، توابع خطی تنها قادر به تولید خط مرزهای خطی خواهند بود.

پرسپترون با چند خروجی

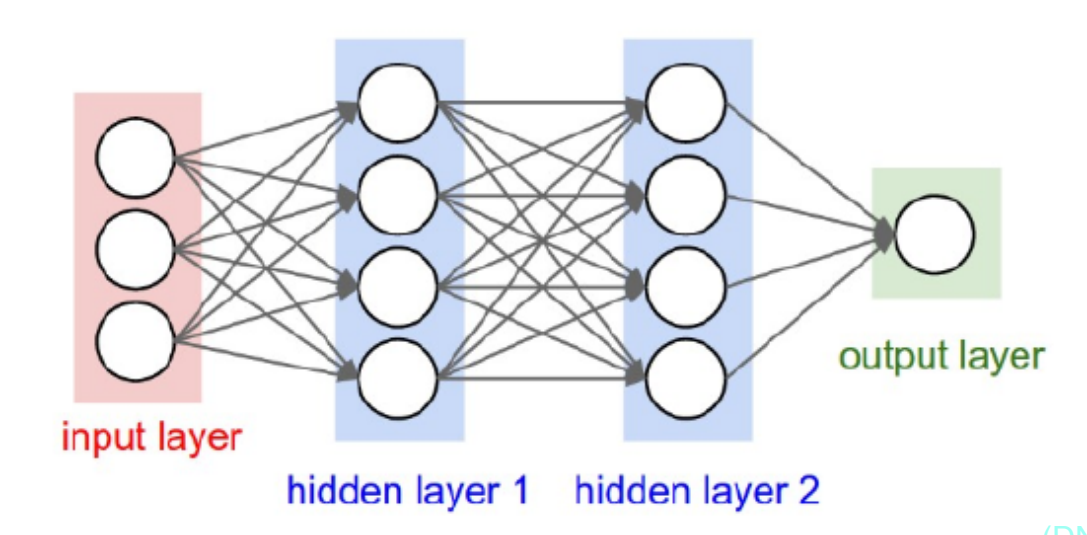


$$z_i = \sum_{j=0}^{m} x_j \cdot w_{j,i}$$

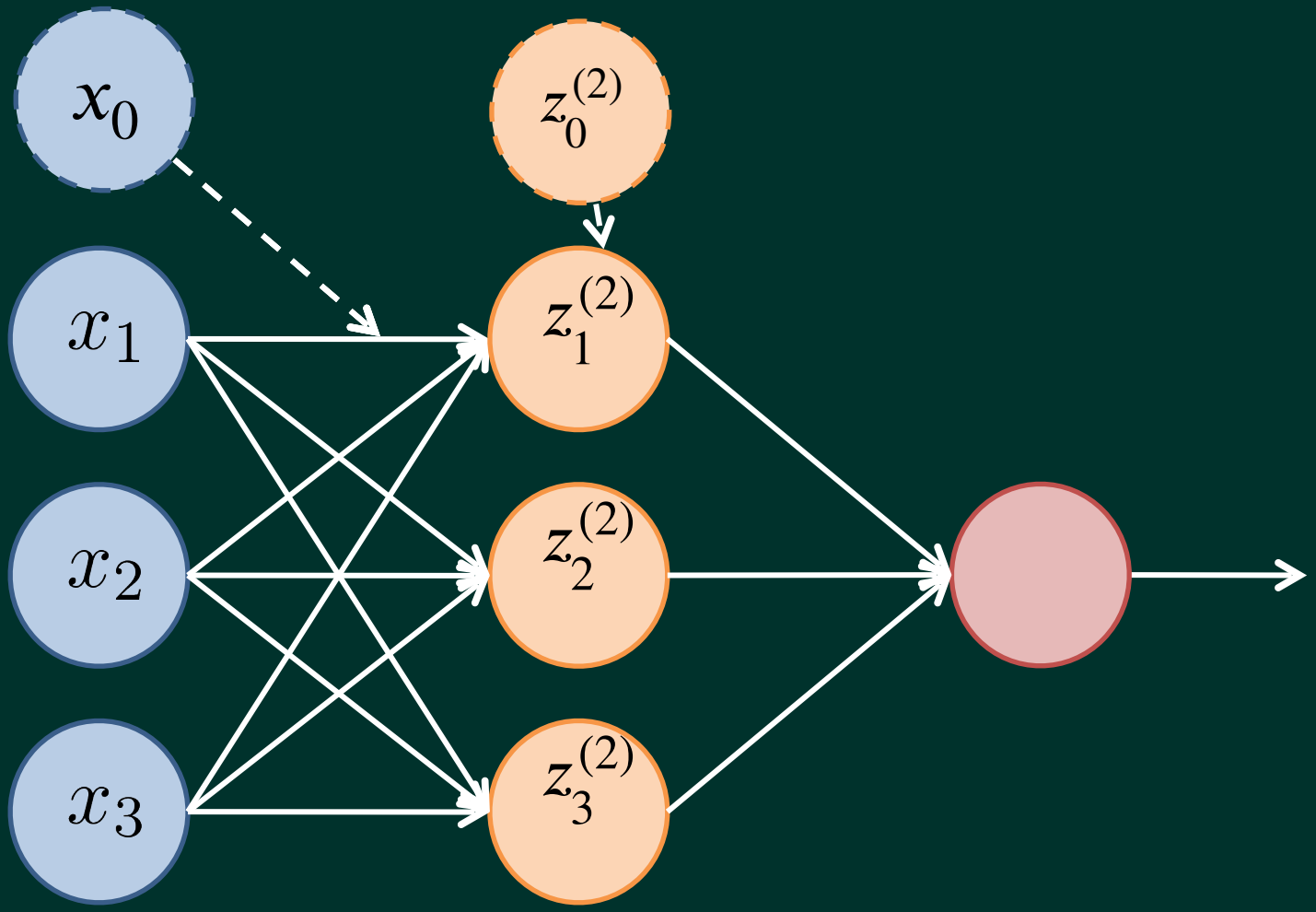
$$\hat{y}_i = g(z_i)$$

لایه‌ها

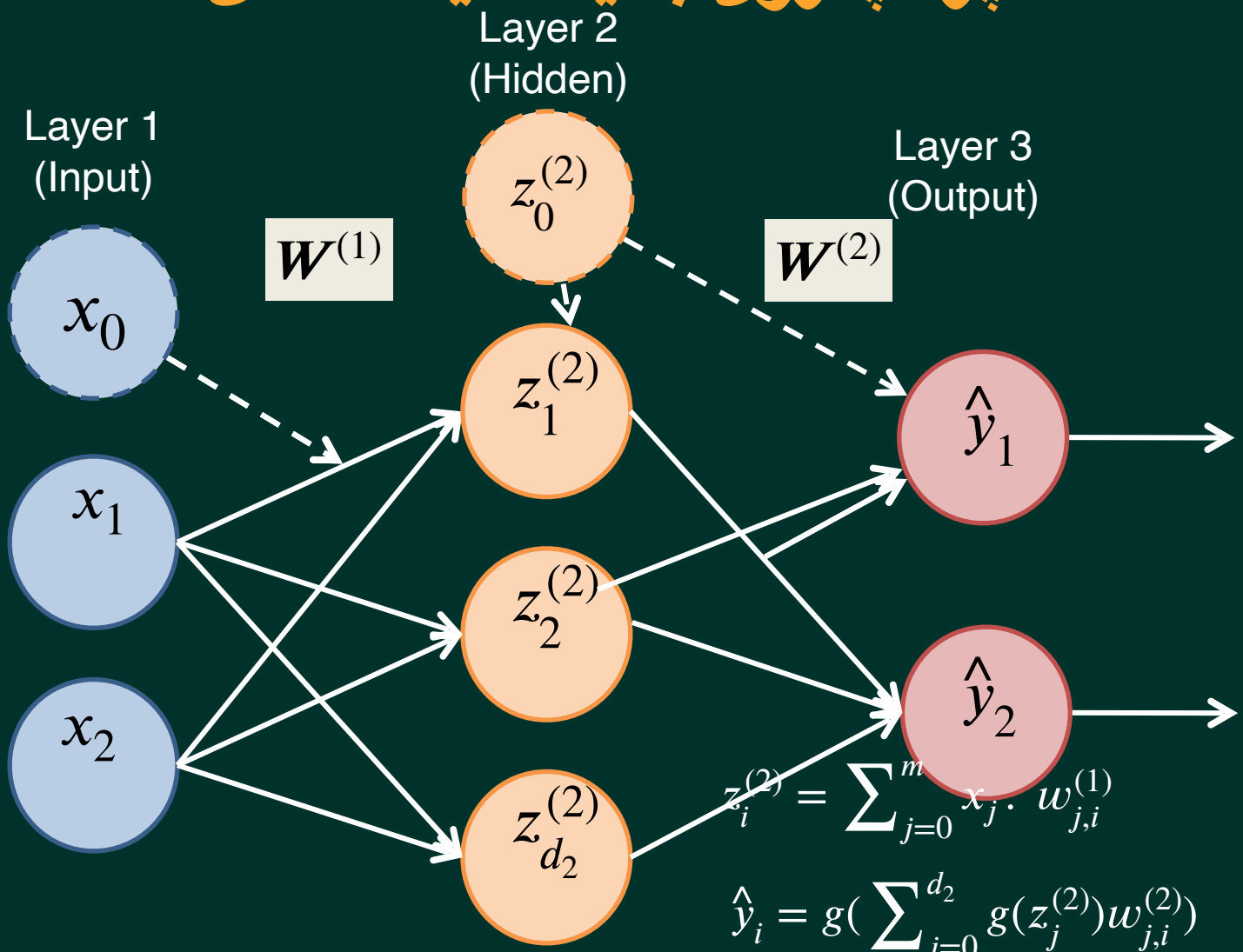
- نورون‌های یک شبکه عصبی در چندین لایه گروه‌بندی می‌شوند.
- در هر شبکه ۳ نوع لایه می‌تواند وجود داشته باشد:
  - لایه ورودی (Input layer)
  - لایه (های) مخفی (میانی) (Hidden Layer(s))
  - لایه خروج



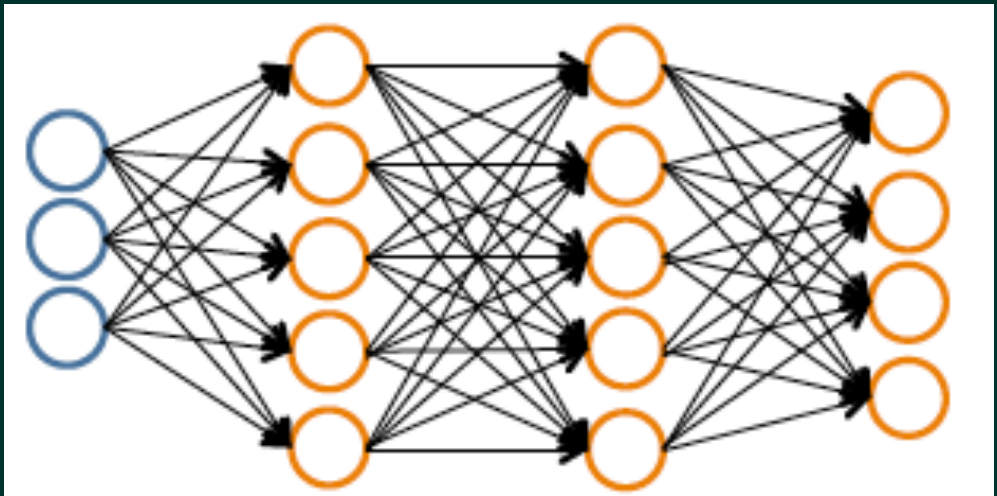
شبکه عصبی


x_0
x_1
x_2
x_3
$z_0^{(2)}$
$z_1^{(2)}$
$z_2^{(2)}$
$z_3^{(2)}$
Layer 1
(Input)
Layer 2
(Hidden)
Layer 3
(Output)

پرسپترون با یک لایه مخفی



پرسپترون با یک لایه مخفی



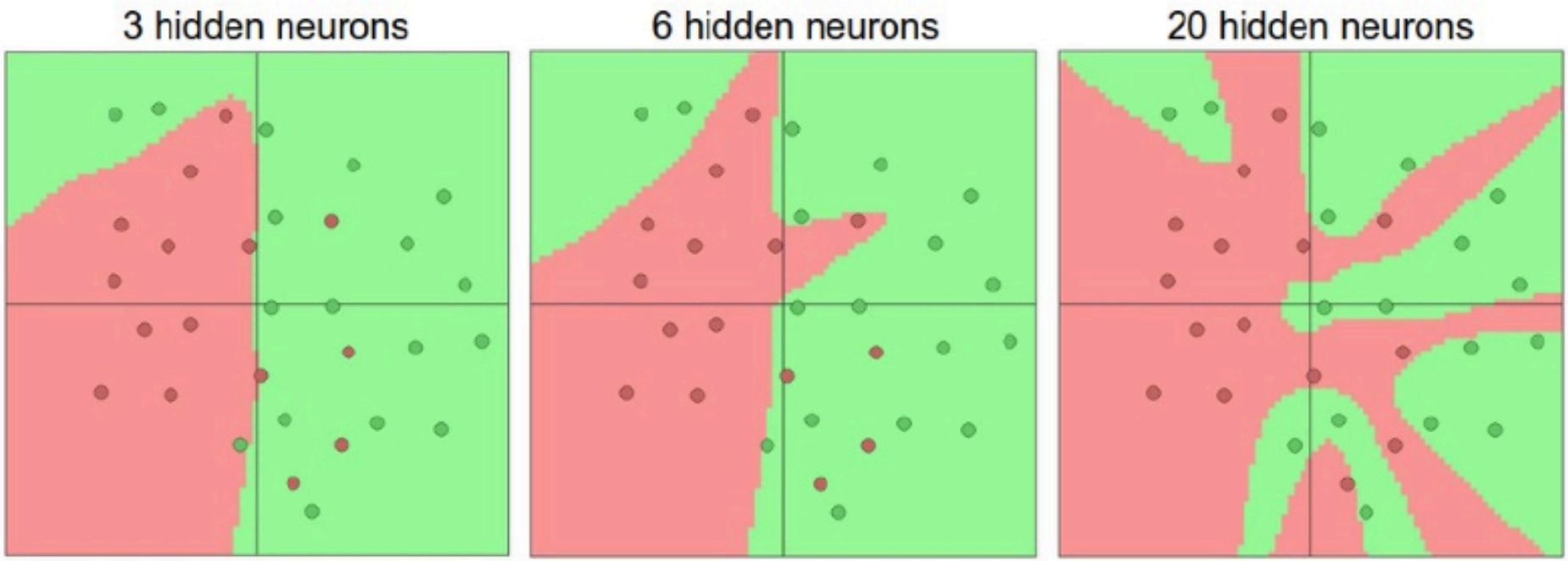
- d_i ام i تعداد نورون‌های لایه :
- d_1 :تعداد ویژگی‌ها : m
- d_L :تعداد واحدهای لایه آخر = تعداد دسته‌ها : C
- $W^{(i)}$ (بدون درنظر گرفتن بایاس) ام $i+1$ ام و لایه i ماتریس وزن‌ها بین لایه :
- اندازه $W^{(i)}$ برابر با $d_i \times d_{i+1}$

$$L = 4, d_1 = 3, d_2 = 5, d_3 = 5, d_4 = 4$$

$$W^{(1)} = 3 \times 5$$
$$W^{(2)} = 5 \times 5$$
$$W^{(3)} = 5 \times 4$$

قدرت نمایش

- یک شبکه عصبی با حداقل یک لایه‌ی مخفی، قادر به نمایش هر تابعی است.



- توانایی شبکه با افزایش تعداد لایه‌های مخفی و تعداد نورون‌های هر لایه افزایش می‌یابد

معنی؟

- شبکه‌های عصبی بسیار **قدرتمند**اند.
- به‌همین دلیل با افزایش توان محاسباتی در سال‌های اخیر تمایل استفاده از آنها به‌شدت افزایش یافته است.

- اما:
- "افزایش قدرت همزان است با افزایش بیش‌برازش *(Overfitting)*"

  – *Boris Ivanovic, 2016*

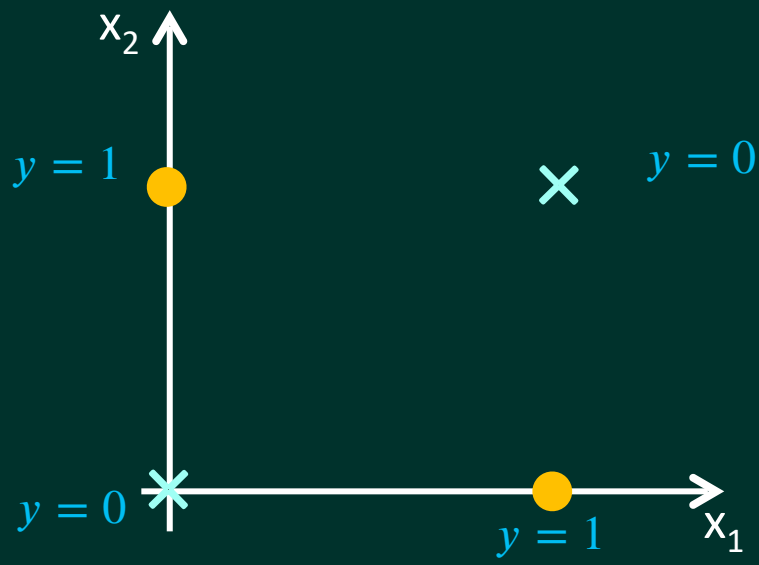
- مثال: ۲۰ "لایه مخفی" در اسلاید قبل

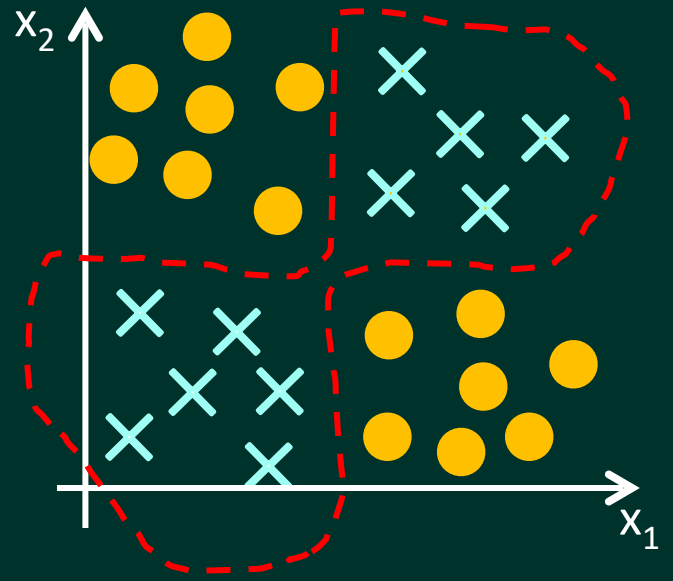
مثال

مثال: تابع یای انحصاری XOR

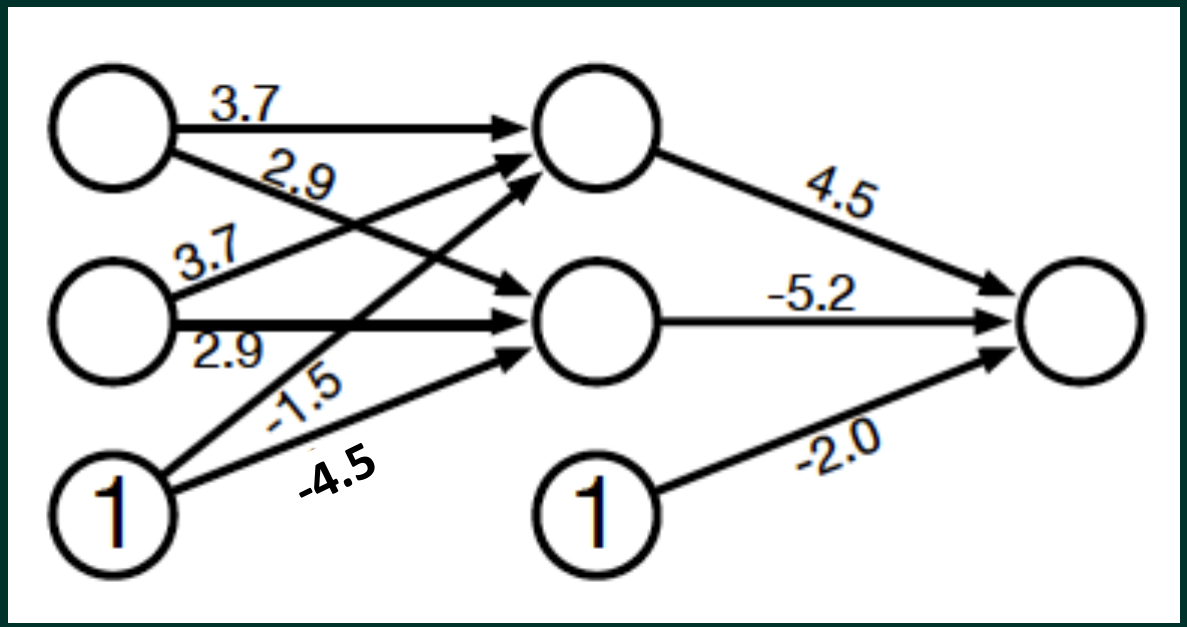
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

$$y = x_1 \ \mathbf{XOR} \ x_2$$

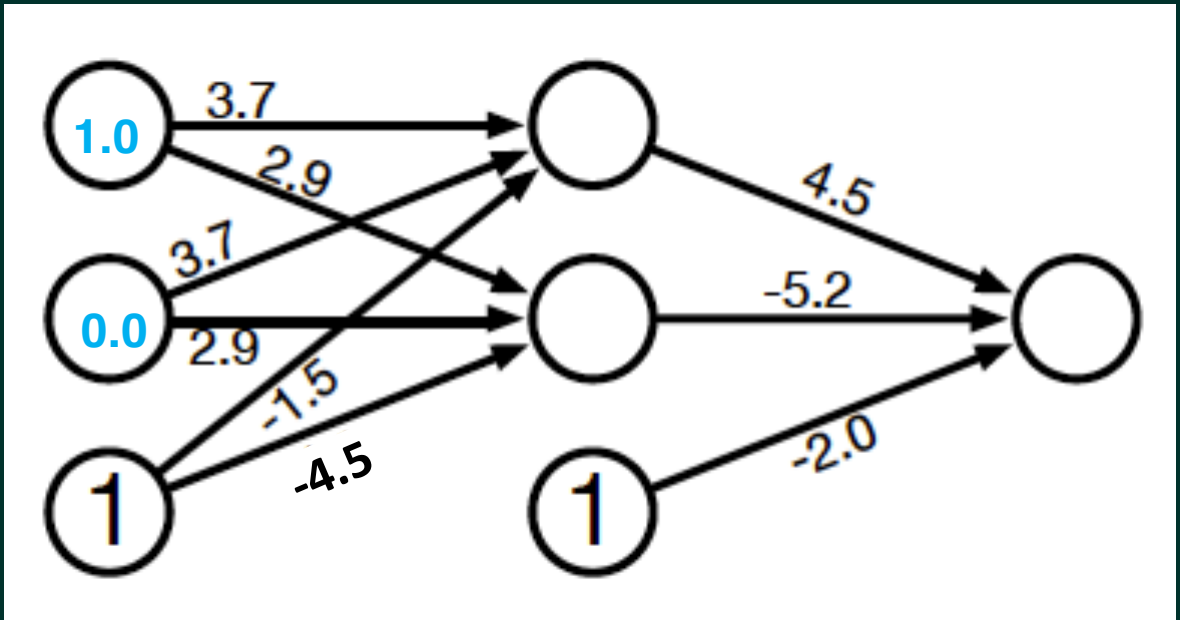




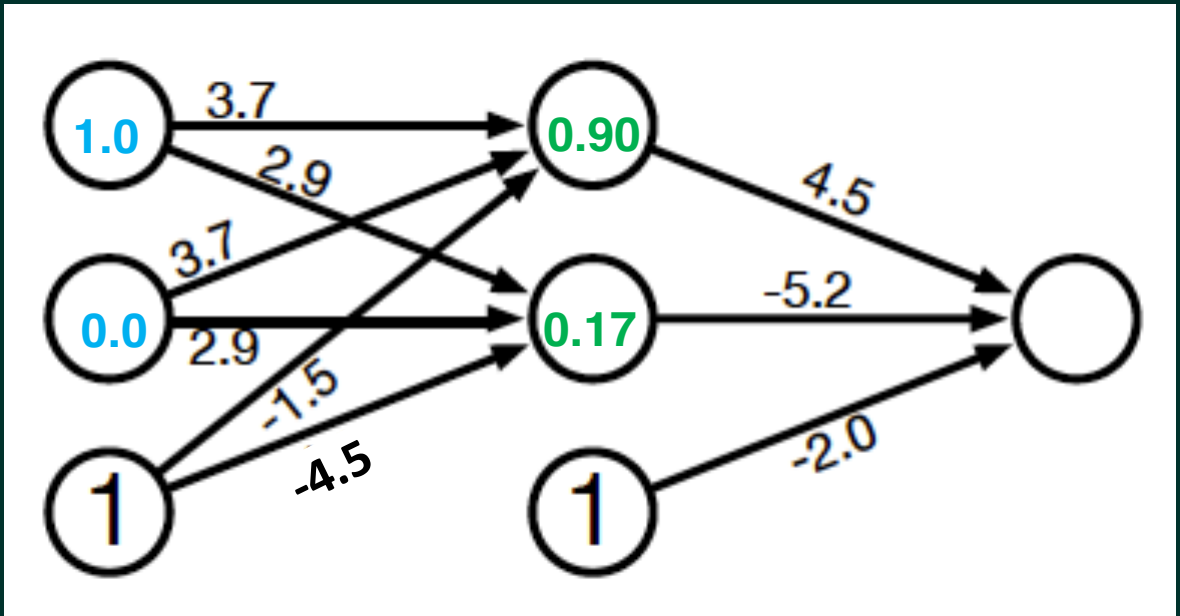
مثال: شبکه عصبیXOR



مثال: لایه ورودی



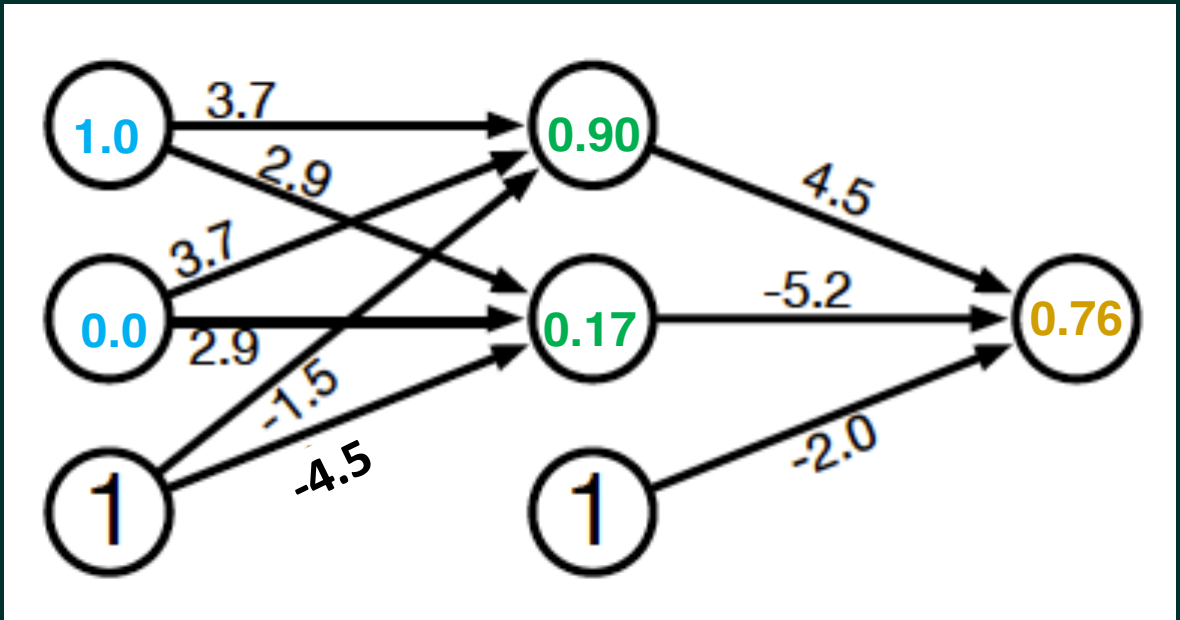
مثال: لایه میانی



$$\text{sigmoid}(1.0 \times 3.7 + 0.0 \times 3.7 + 1 \times -1.5) = \text{sigmoid}(2.2) = \frac{1}{1+e^{-2.2}} = 0.90$$

$$\text{sigmoid}(1.0 \times 2.9 + 0.0 \times 2.9 + 1 \times -4.5) = \text{sigmoid}(-1.6) = \frac{1}{1+e^{1.6}} = 0.17$$

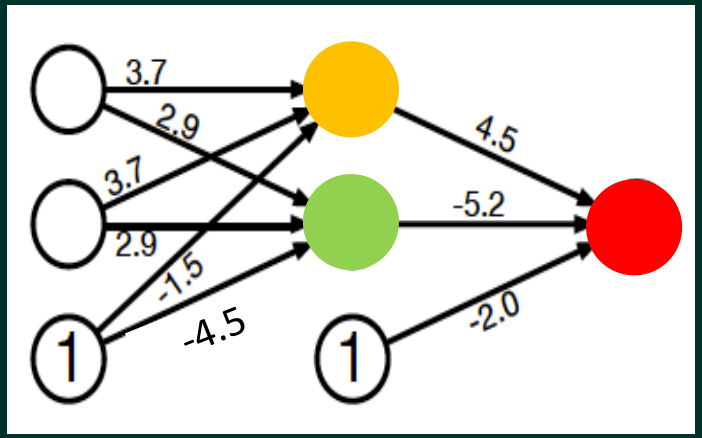
مثال: لایه خروجی



$$\text{sigmoid}(.90 \times 4.5 + .17 \times -5.2 + 1 \times -2.0) = \text{sigmoid}(1.17) = \frac{1}{1 + e^{-1.17}} = 0.76$$

خروجی به ازای تمام ترکیبات ممکن ورودی

Input	Input	H	H	O	
0	0	0.18	0.01	0.22	0
0	1	0.90	0.17	0.76	1
1	0	0.90	0.17	0.76	1
1	1	0.99	0.78	0.17	0



$$h_1 = x_1 \ \mathbf{OR}\ x_2$$

$$h_2 = x_1 \ \mathbf{AND}\ x_2$$

$$y = x_1 \ \mathbf{XOR}\ x_2$$

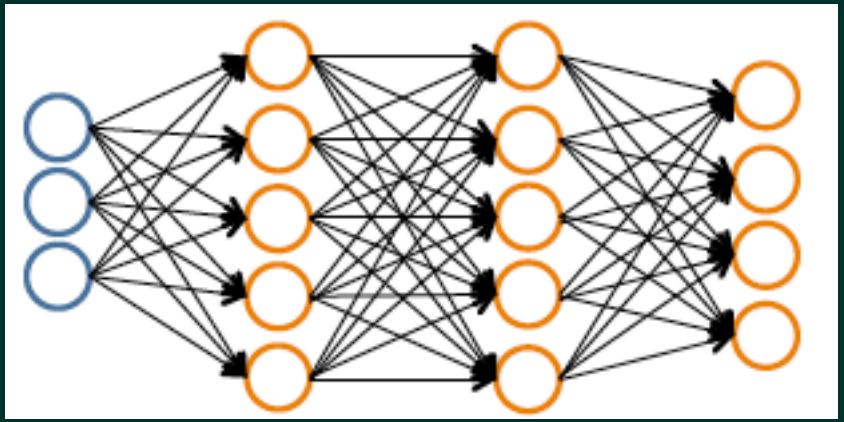
دسته‌بندی با شبکه عصبی

Pedestrian

Car

Motorcycle

Truck



$$\hat{y}_i \in \mathbb{R}^k$$

$$Pedestrain = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}; \quad Car = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$Motorcycle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}; \quad Truck = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Slide by Andrew NG

آموزش یک شبکه عصبی

چگونگی یادگیری بردار وزن‌ها

یادگیری بانظارت

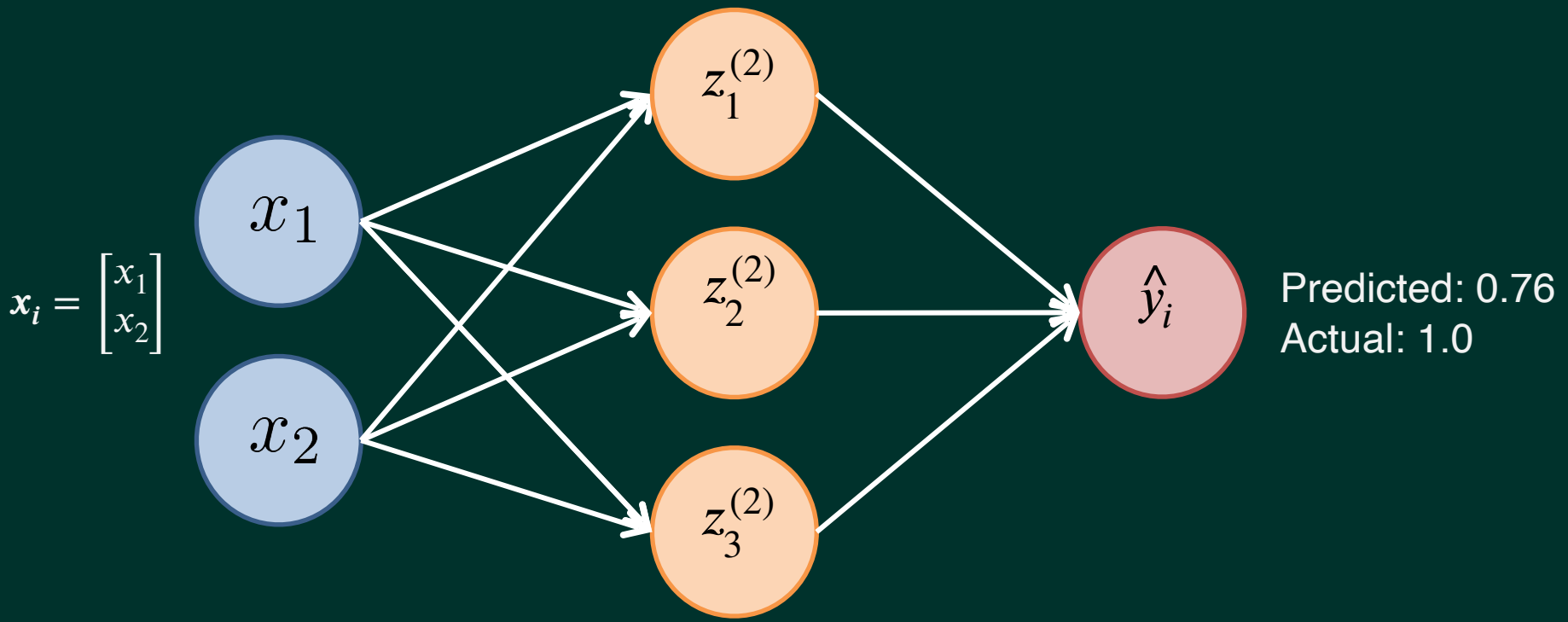
- برای هر نمونه آموزشی x، مقدار حقیقی برچسب y را می‌دانیم.
- مدل فراگرفته شده، تخمینی از برچسب ارائه می‌دهد: $\hat{y}$

هدف فراگرفتن مقادیر ماتریس‌های $W^{(i)}$ است به‌گونه‌ای که فاصله بین y و $\hat{y}$ تا حد امکان کم باشد.

- نیاز به تابعی برای تخمین این نزدیکی داریم: تابع زیان، تابع هزینه
- نیاز به الگوریتم بهینه‌سازی تا با به‌روزرسانی مقادیر ماتریس‌های $W^{(i)}$ مقدار تابع هزینه را کمینه کند.

تابع زیان

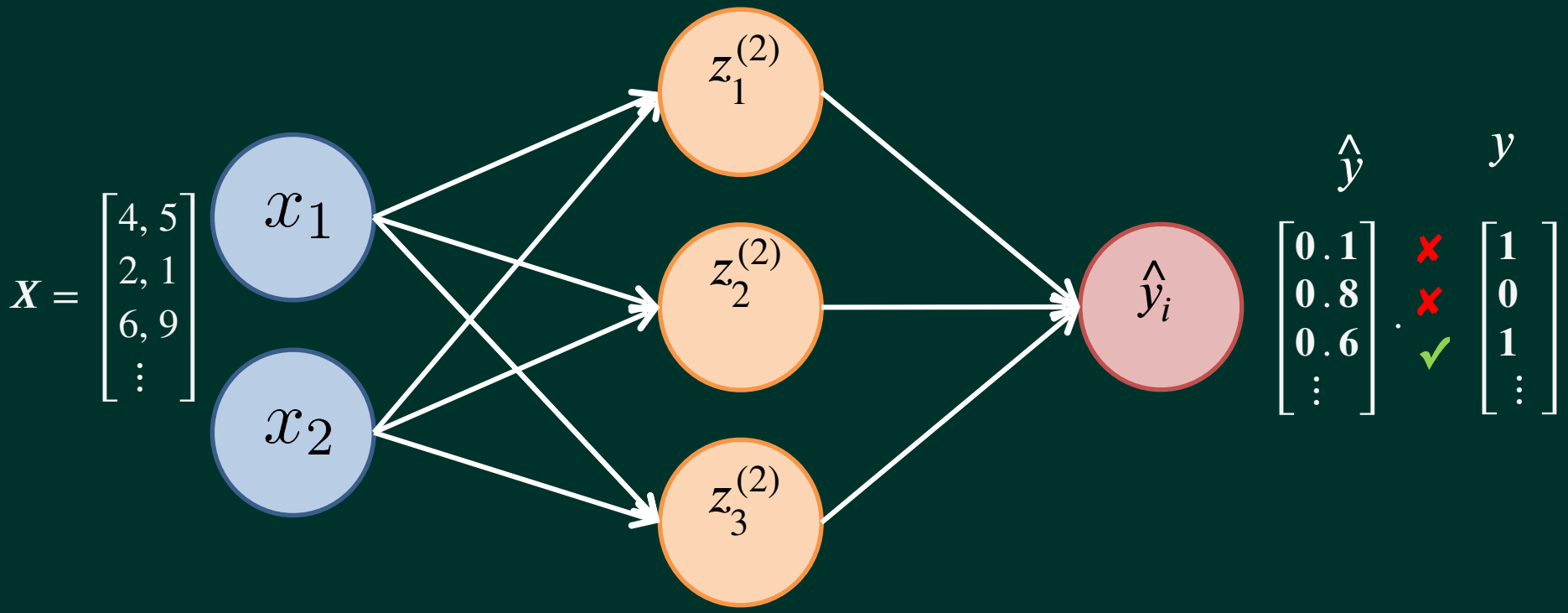
- **تابع زیان:** هزینه‌ی پیش‌بینی اشتباه توسط شبکه را اندازه‌گیری می‌کند.



$$\mathcal{L}(\hat{y}_i,\ y_i)$$

تابع هزینه

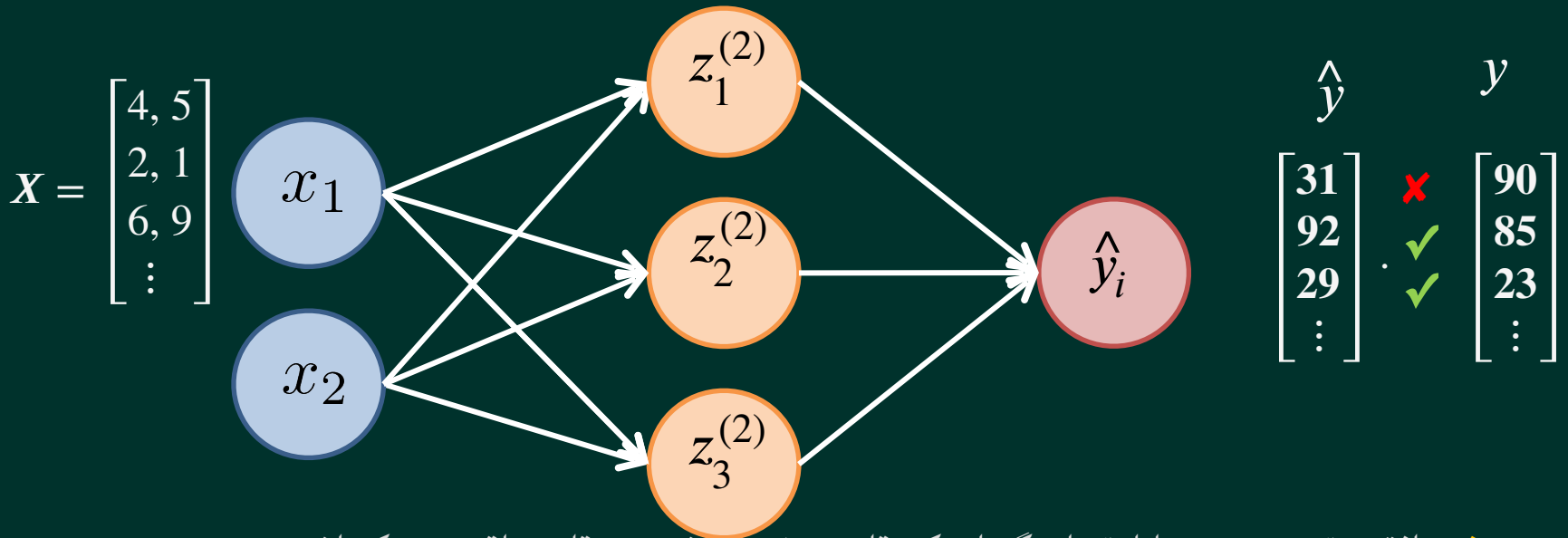
- **تابع هزینه:** زیان کلی حاصل از تمامی داده‌های آموزشی را اندازه‌گیری می‌کند.



$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}(\hat{y}_i,\ y_i)$$

تابع هزینه میانگین مربع خطا
MEAN SQUARED ERROR COST

- برای مدل‌هایی که خروجی یک عدد حقیقی است (رگرسیون)



- هدف: یافتن بهترین مجموعه پارامترها به‌گونه‌ای که مقادیر پیش‌بینی شده به مقادیر واقعی نزدیک باشند
- **تابع زیان (Loss Function)**: مربع مانده یا مربع خطا (بیان می‌کند که برازش چقدر خوب بوده است)

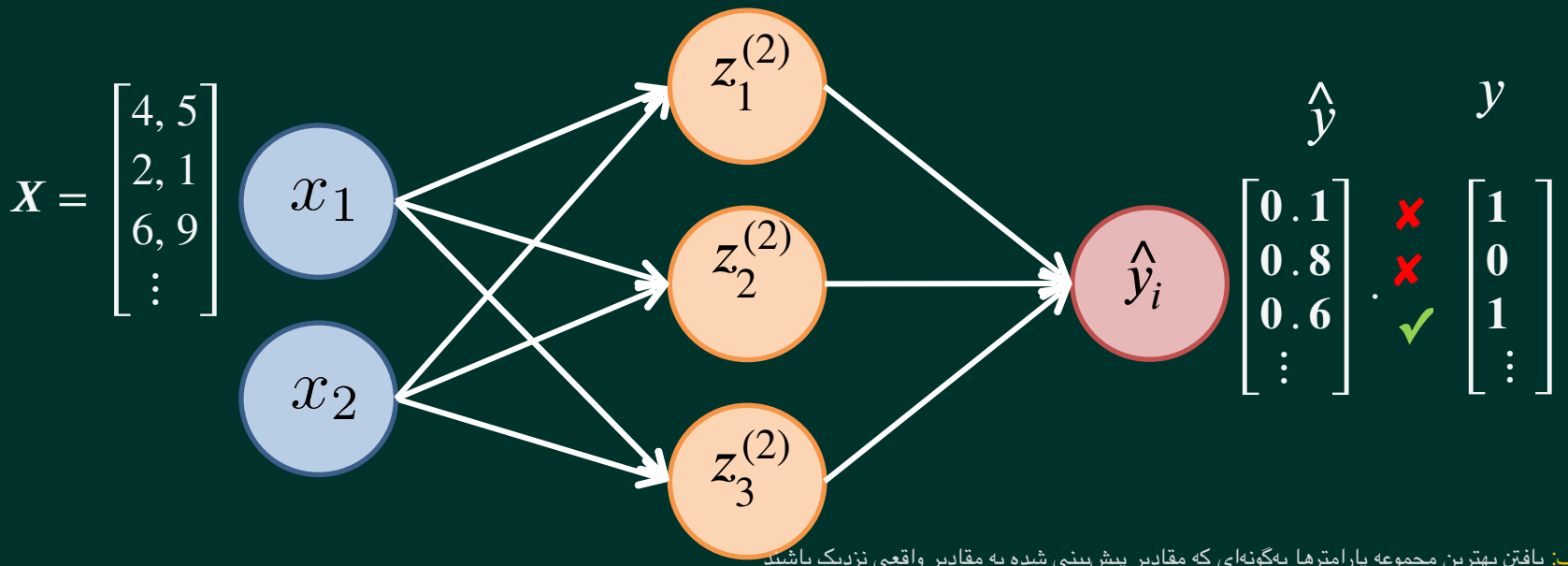
$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}\left(\hat{y} - y\right)^2$$

تابع هزینه (Cost Function): میانگین تابع زیان بر روی همه نمونه‌های آموزشی

$$J(w, b) = \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2$$

تابع هزینه آنتروپی متقاطع دوتایی
BINARY CROSS ENTROPY COST

- برای مدل‌هایی که خروجی به‌صورت احتمالی بین ۰ و ۱ است به‌کار می‌رود. (دسته‌بندی)



- هدف: یافتن بهترین مجموعه پارامترها به‌گونه‌ای که مقادیر پیش‌بینی شده به مقادیر واقعی نزدیک باشند
- تابع زیان (Loss Function)
- تابع هزینه (Cost Function): میانگین تابع زیان بر روی همه نمونه‌های آموزشی

$$\mathscr{L}\left(\hat{y},\ y\right) = \begin{cases} -\log\left(\hat{y}\right) . & if\ y = 1 \\ -\log\left(1 - \hat{y}\right) & if\ y = 0 \end{cases}$$

$$J(\boldsymbol{W}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\,.\,log\left(\hat{y}_i\right) + \left(1 - y_i\right)log(1\ - \hat{y}_i\right]$$

جمع‌بندی صورت مسئله

- **ورودی:** n داده به شکل

$$(x_1, y_1),\ (x_2, y_2),\ \cdots,\ (x_n,\ y_n);$$

- **پارامترها:**

$$W = \{W^{(0)}, W^{(1)},\ \ldots\}$$

- **تابع هزینه (Cost Function)**

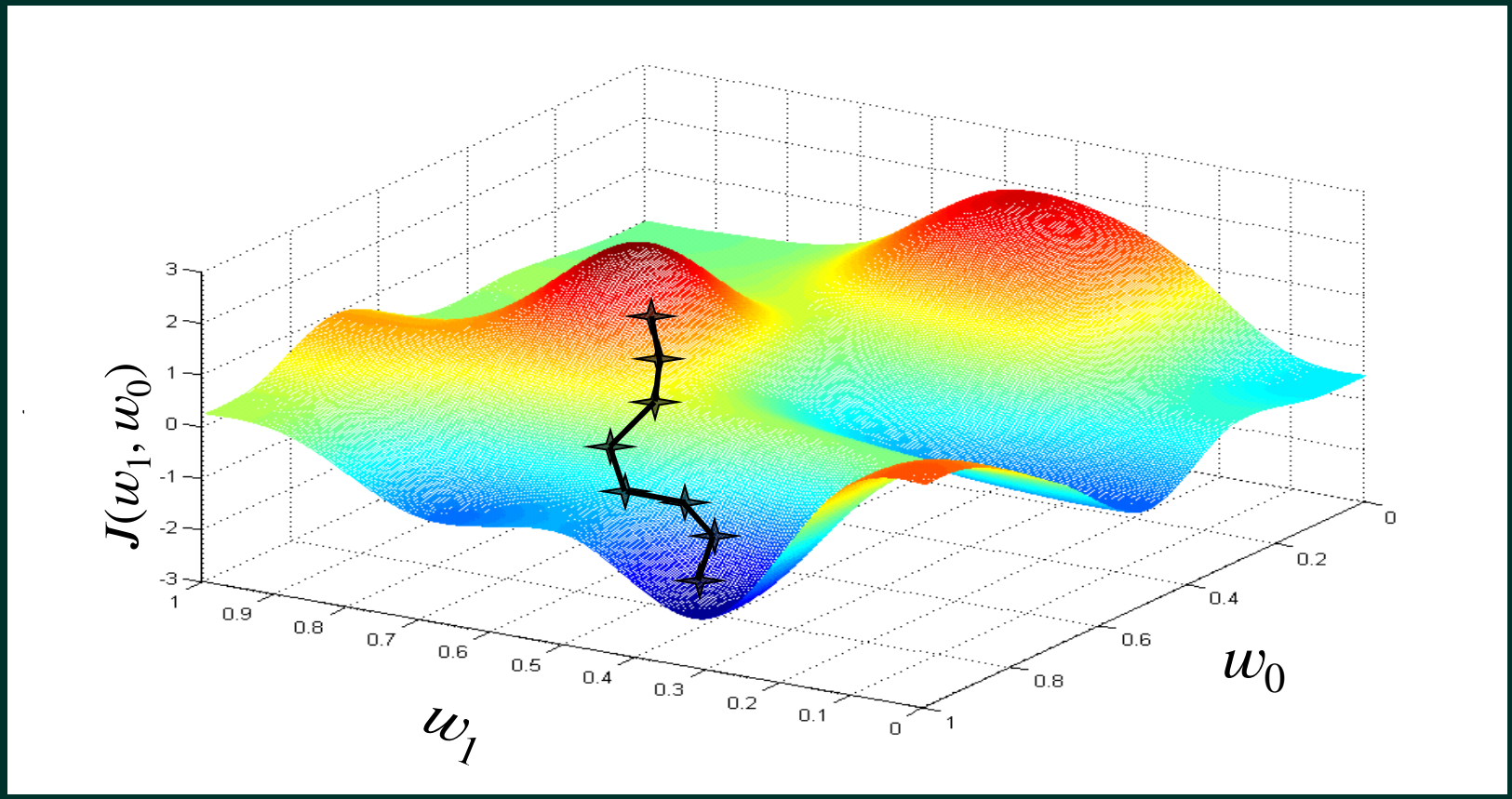
$$J(W) = -\frac{1}{n}\sum_{i=1}^{n}[y_i\,.\,log\left(\hat{y}_i\right) + \left(1 - y_i\right)log(1\ - \hat{y}_i]$$

- **هدف:** کمینه‌کردن تابع هزینه

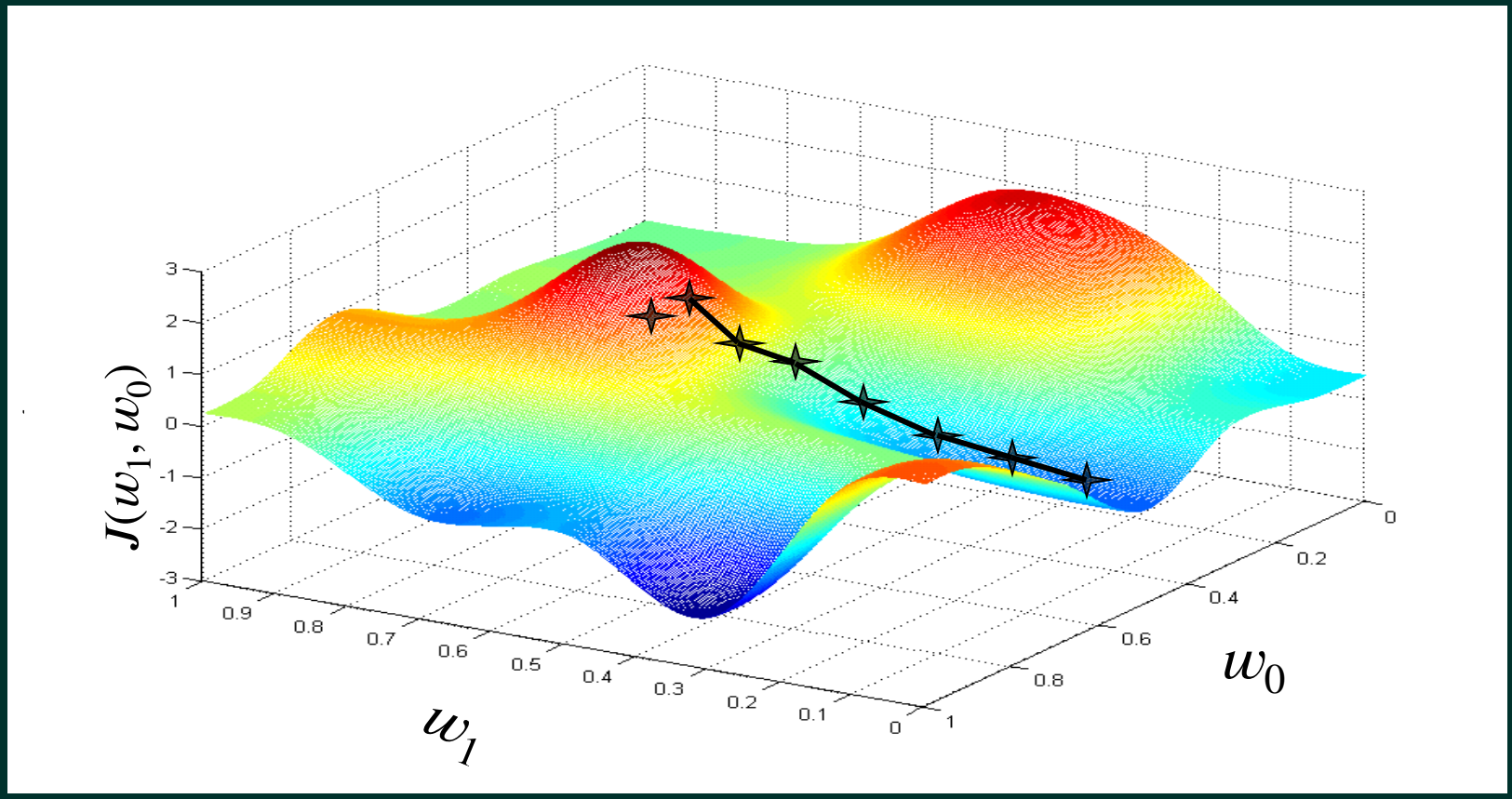
$$\min_{w} J(W)$$

گرادیان کاهشی

- **ورودی:** تابع $J(\boldsymbol{W})$
- **هدف:** $\min_{\boldsymbol{W}} J(\boldsymbol{W})$
- **گرادیان کاهشی:**
  - یک الگوریتم تکرارشونده (iterative)
  - از یک مقدار اولیه برای پارامترها (ماتریس‌های $\boldsymbol{W}^{(i)}$)شروع کرده و به صورت پیاپی مقادیر پارامترها را برای کاهش دادن مقدار $J(\boldsymbol{W})$ تا رسیدن به نقطه بهینه به‌روزرسانی می‌کند.

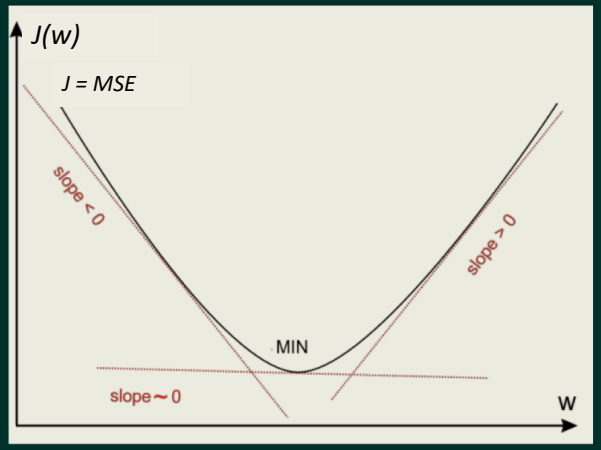


اگر تابع چندین نقطه بهینه محلی داشته باشد، نقطه‌ی شروع در رسیدن به هرکدام از آنها تاثیرگذار خواهد بود.


$J(w_1, w_0)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
w_1
0
0.2
0.4
0.6
0.8
1
w_0

فلسفه گرادیان کاهشی

- **مشاهده:**
  - اگر $\frac{\partial J}{\partial w} > 0$ ، مشتق مثبت، تابع صعودی است. درنتیجه افزایش w موجب افزایش J خواهد شد.
  - اگر $\frac{\partial J}{\partial w} < 0$ ، مشتق منفی، تابع نزولی است. درنتیجه افزایش w موجب کاهش J خواهد شد.



- به‌روزرسانی زیر موجب کاهش تابع هزینه خواهد شد:

$$w = w - \eta \frac{\partial J}{\partial w}$$

- الگوریتم نام خود را از بردار گرادیان می‌گیرد:

بردار گرادیان : $\frac{\partial J}{\partial \boldsymbol{w}} = \begin{pmatrix} \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_m} \end{pmatrix}$

الگوریتم گرادیان کاهشی

1. Initialize weights randomly
2. repeat until convergence {
3. *Compute gradient*, $\frac{\partial}{\partial \boldsymbol{w}} J(\boldsymbol{W})$
4. Update weights, $\boldsymbol{w} = \boldsymbol{w} - \eta \frac{\partial}{\partial \boldsymbol{w}} J(\boldsymbol{W})$

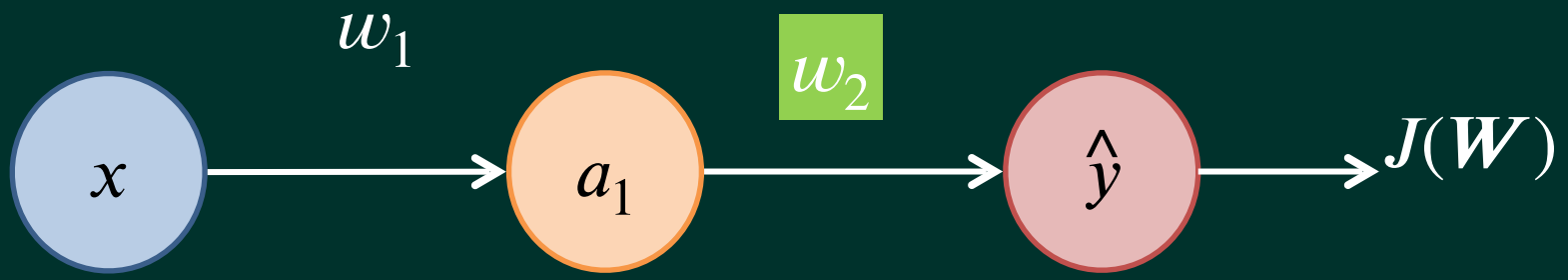
}

5. Return weights

η نرخ یادگیری (Learning rate)

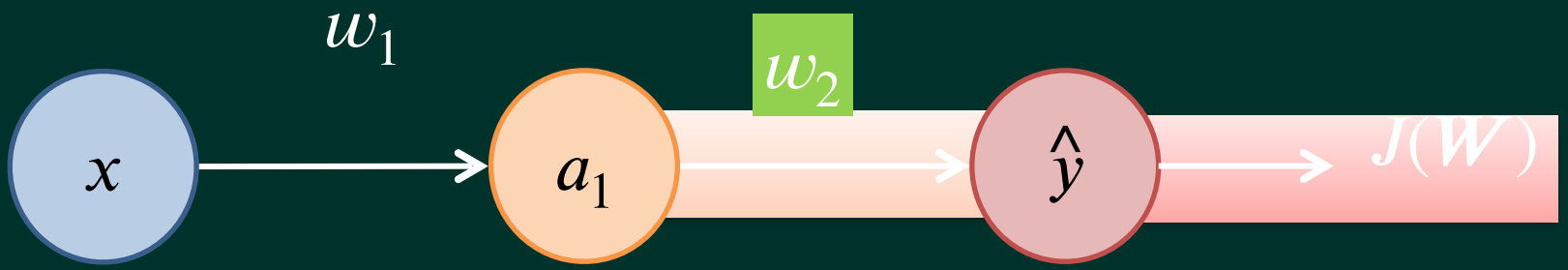
انتشار به عقب
BACKPROPAGATION

محاسبه گرادیان: انتشار به عقب



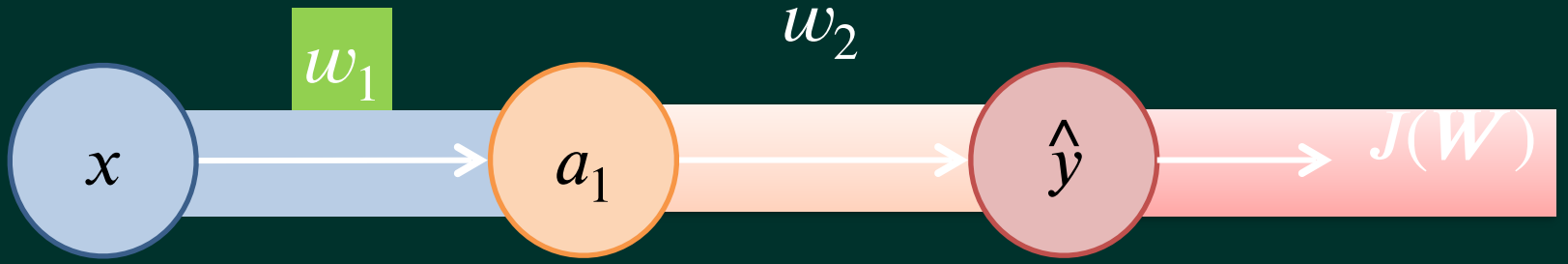
چگونه یک تغییر کوچک در یکی از وزن‌ها (مثلا w_2)، بر روی مقدار $J(\boldsymbol{W})$ اثرمی‌گذارد؟

محاسبه گرادیان: انتشار به عقب



$$\frac{\partial J(\boldsymbol{W})}{\partial w_2} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_2}$$

محاسبه گرادیان: انتشار به عقب

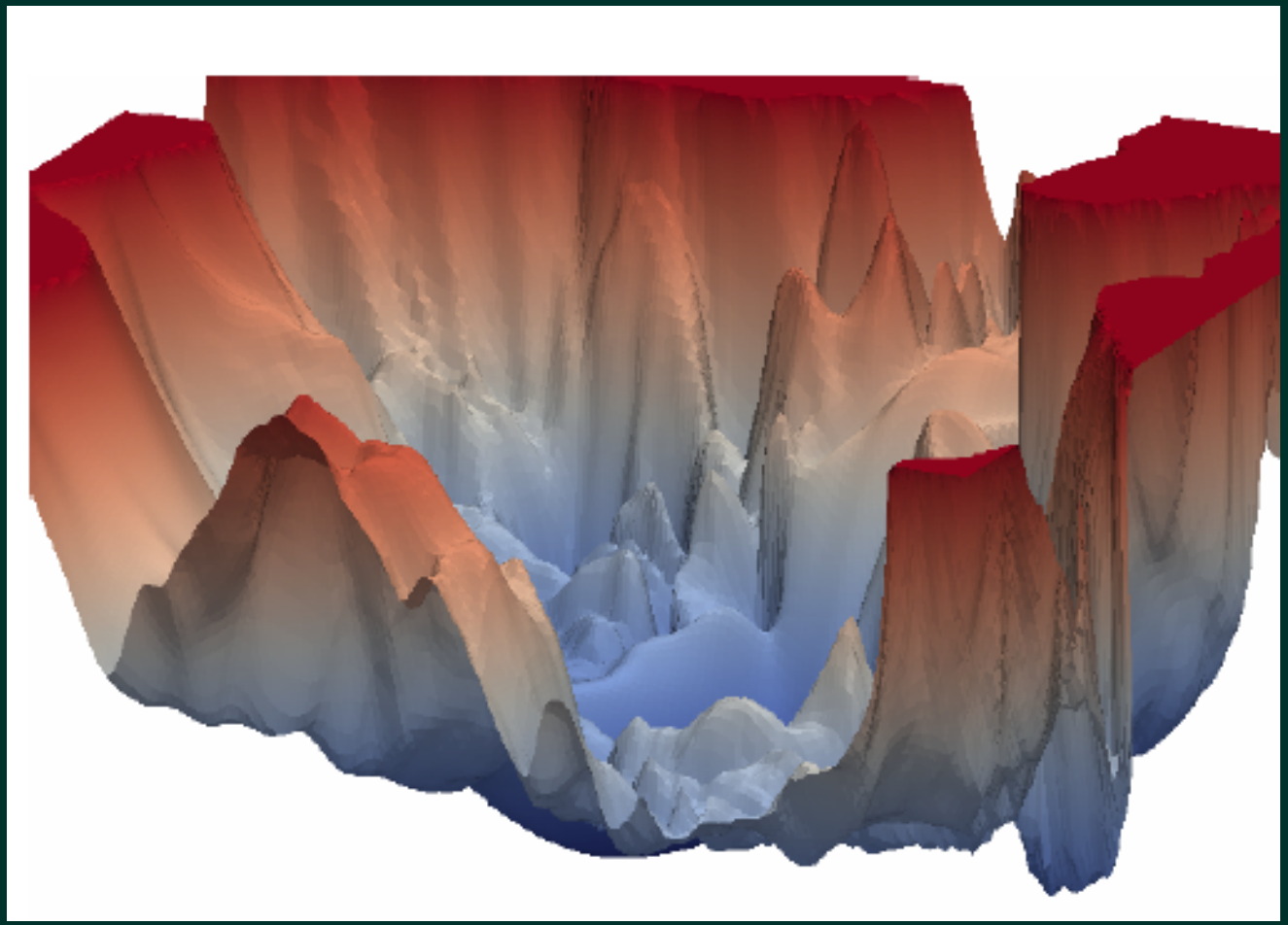


$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial a_1} \times \frac{\partial a_1}{\partial w_1}$$

این عملیات برای تمامی وزن‌ها با استفاده از گرادیان لایه‌های بعدی تکرار می‌شود.

آموزش شبکه‌های عصبی مشکل است

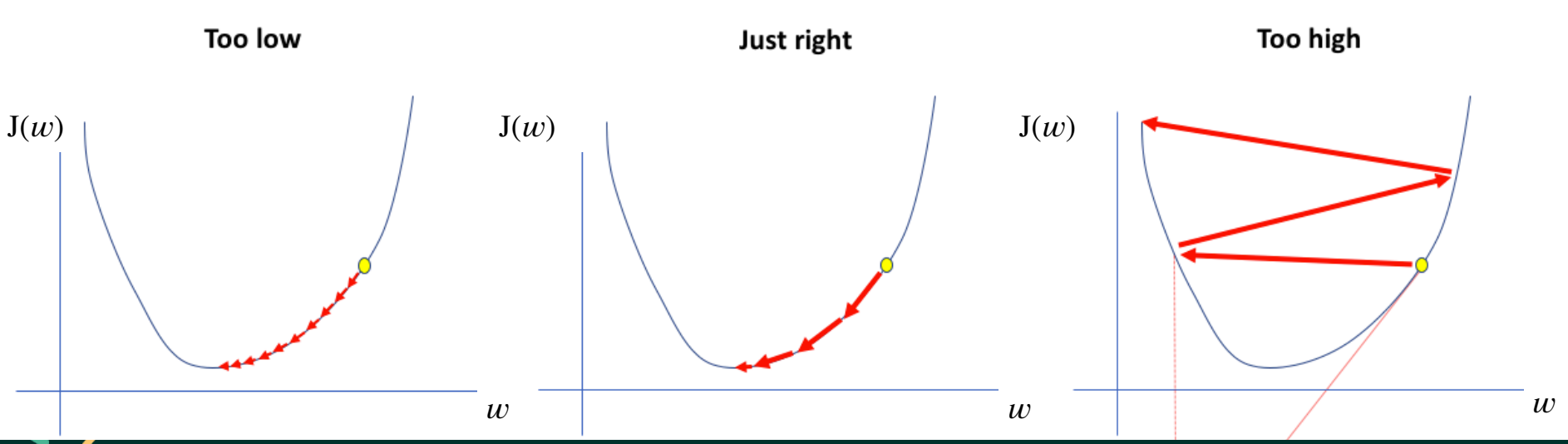
- تابع هزینه یک تابع محدب نیست و تعداد زیادی کمینه محلی دارد.



https://www.ijcai.org/proceedings/2020/0387.pdf

نرخ یادگیری

- هرچه مقدار η بیشتر باشد، w با سرعت بیشتری تغییر می‌کند.
- اگر خیلی کوچک باشد، گرادیان کاهشی ممکن است خیلی طول بکشد.
  - تعداد زیادی به‌روزرسانی
- اگر خیلی بزرگ باشد، گرادیان کاهشی ممکن است اطراف نقطه کمینه پرسه بزند ولی همگرا نشود.



نرخ یادگیری تطبیقی
ADAPTIVE LEARNING RATE

- می‌توان مقادیر مختلفی از نرخ‌های یادگیری را برای پیداکردن نرخ یادگیری بهینه امتحان نمود.

روش بهتر:

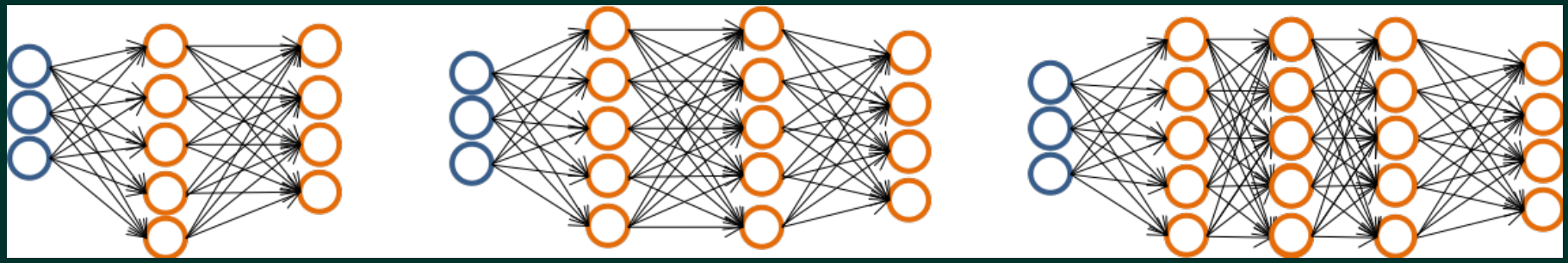
- طراحی یک نرخ یادگیری تطبیقی که با توجه به موقعیت مسئله تغییر کند.
- نرخ یادگیری با توجه به موارد مختلف می‌توان در طول فرآیند یادگیری کم و زیاد شود:
  - میزان بزرگی گرادیان
  - میزان سرعت فرآیند یادگیری
  - اندازه برخی وزن‌ها
  - ...

الگوریتم‌های مختلف گرادیان کاهشی

- Stochastic Gradient Descent
- Adam
- Adadelta
- Adagrad
- RMSProp

جمع‌بندی آموزش شبکه‌های عصبی

۱. انتخاب معماری مناسب



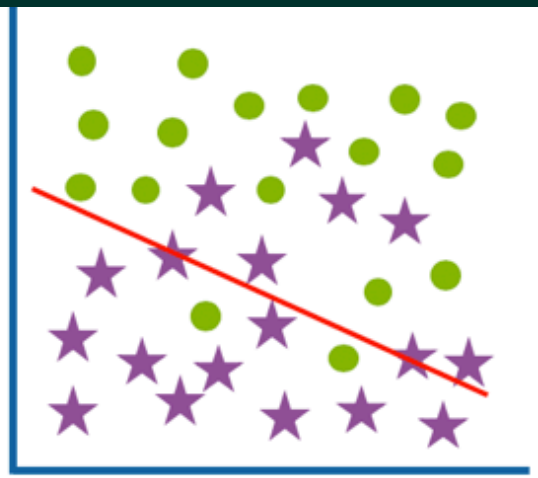
- تعداد نورون لایه اول: ابعاد ورودی (تعداد ویژگی‌ها)
- تعداد نورون لایه آخر: تعداد رده‌ها
- به‌طور پیش فرض، یک یا بیشتر لایه میانی.

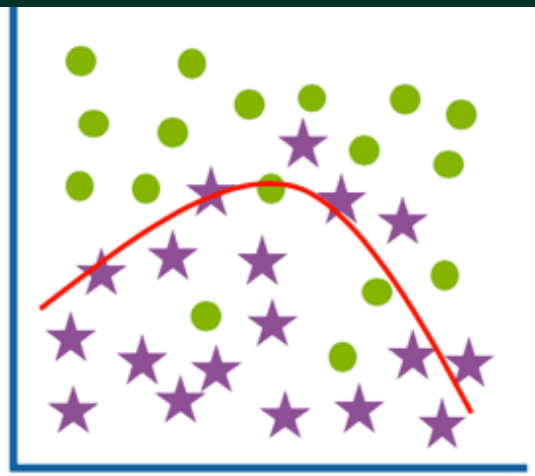
۲. آموزش

1. مقداردهی اولیه وزن‌های شبکه به‌صورت تصادفی
2. پیاده‌سازی انتشار به جلو (Forward Propagation) جهت محاسبه $\hat{\boldsymbol{y}}(\boldsymbol{x}_i)$ به ازای هر داده ورودی $\boldsymbol{x}_i$
3. پیاده‌سازی نحوه محاسبه تابع هزینه $\boldsymbol{J}(\boldsymbol{W})$
4. پیاده‌سازی انتشار به عقب (Backprop) برای محاسبه مشتقات $\frac{\partial J(\boldsymbol{W})}{\partial w_{i,j}^{(k)}}$
5. استفاده از الگوریتم گرادیان کاهشی یا هر الگوریتم بهینه‌سازی دیگر برای کمینه کردن مقدار تابع هزینه $\boldsymbol{J}(\boldsymbol{W})$

بیش‌برازش در شبکه‌های عصبی

بیش‌برازش





Underfit (High Bias)

- تعداد کم ویژگی در مجموعه داده
- عملکرد ضعیف بر روی هر دو مجموعه آموزش و آزمون

Just Right

Overfit (High Variance)

- تعداد زیاد ویژگی در مجموعه داده
- عملکرد خوب بر روی مجموعه آموزشی و عملکرد ضعیف بر روی مجموعه آزمون

شبکه‌های عصبی عمیق (DNN)

منظم‌سازی

- تکنیک‌هایی برای محدودسازی مسئله بهینه‌سازی از فراگرفتن مدل‌های پیچیده

- چرا در یک مسئله یادگیری به منظم‌سازی نیاز داریم؟
  - جهت بهبود عملکرد مدل در تعمیم برای داده‌های دیده نشده

- روش‌های استاندارد:
  - محدودسازی تعداد نورون‌ها در لایه‌های مخفی
  - محدودسازی مقدار وزن‌ها: تخریب وزن (Weight decay)
  - توقف زودهنگام فرآیند یادگیری قبل از آنکه شبکه دچار بیش‌برازش شود (Early stopping)

منظم‌سازی L^2
(RIDGE REGULARIZATION)

- با اضافه‌کردن یک عبارت مجازات (Penalty) به تابع هزینه، از افزایش مقادیر ضرایب (وزن‌ها) جلوگیری کرد:

$$\mathscr{R}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 = \frac{1}{2}\sum_{j=1}^{m} w_j^2$$

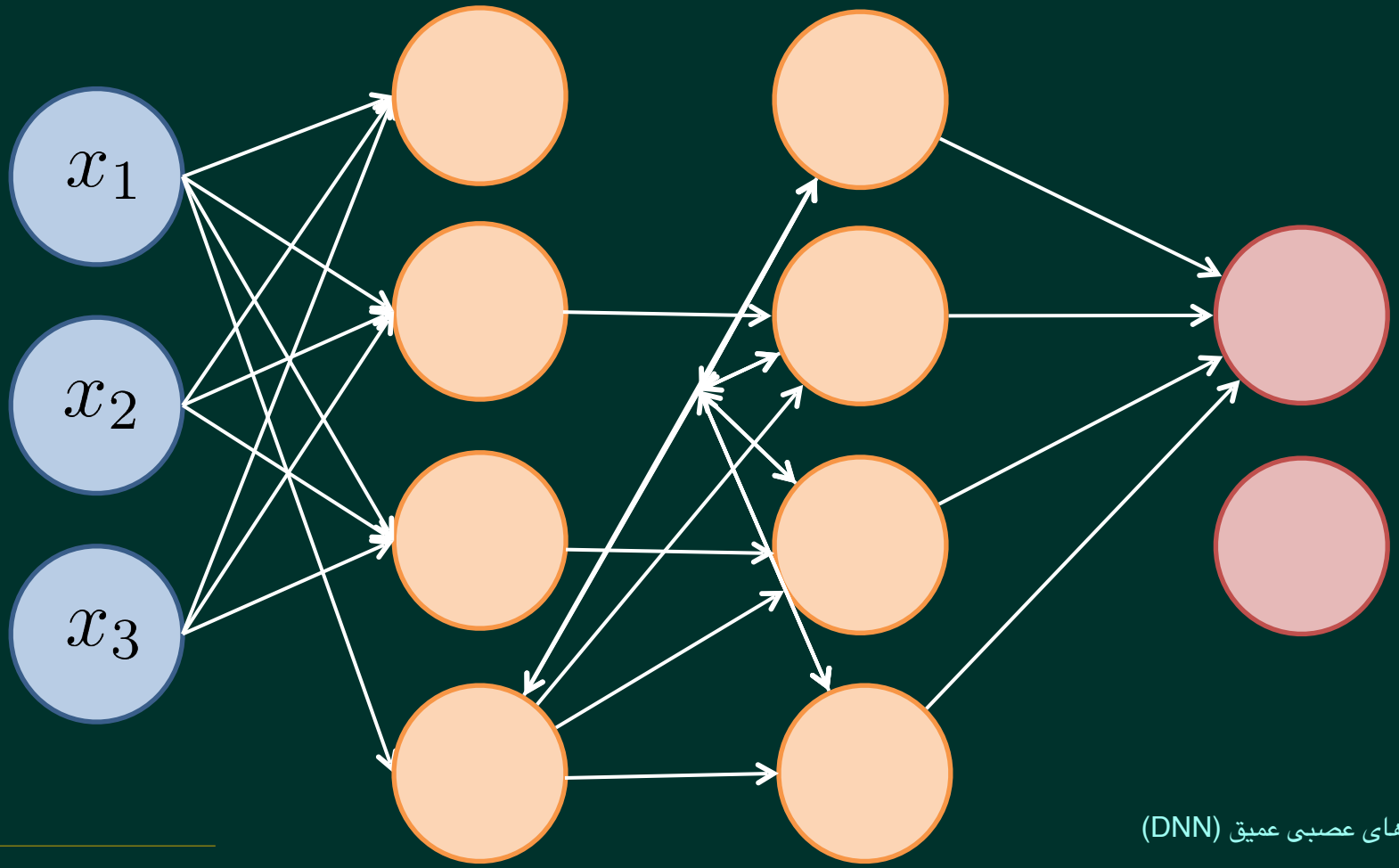
- تابع هزینه منظم‌شده عبارت است از:

$$J_{Logreg} = J + \lambda\mathscr{R} = J + \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2$$

- یک پارامتر اصل که با استفاده از مجموعه اعتبارسنجی می‌توان آن را تنظیم نمود λ.

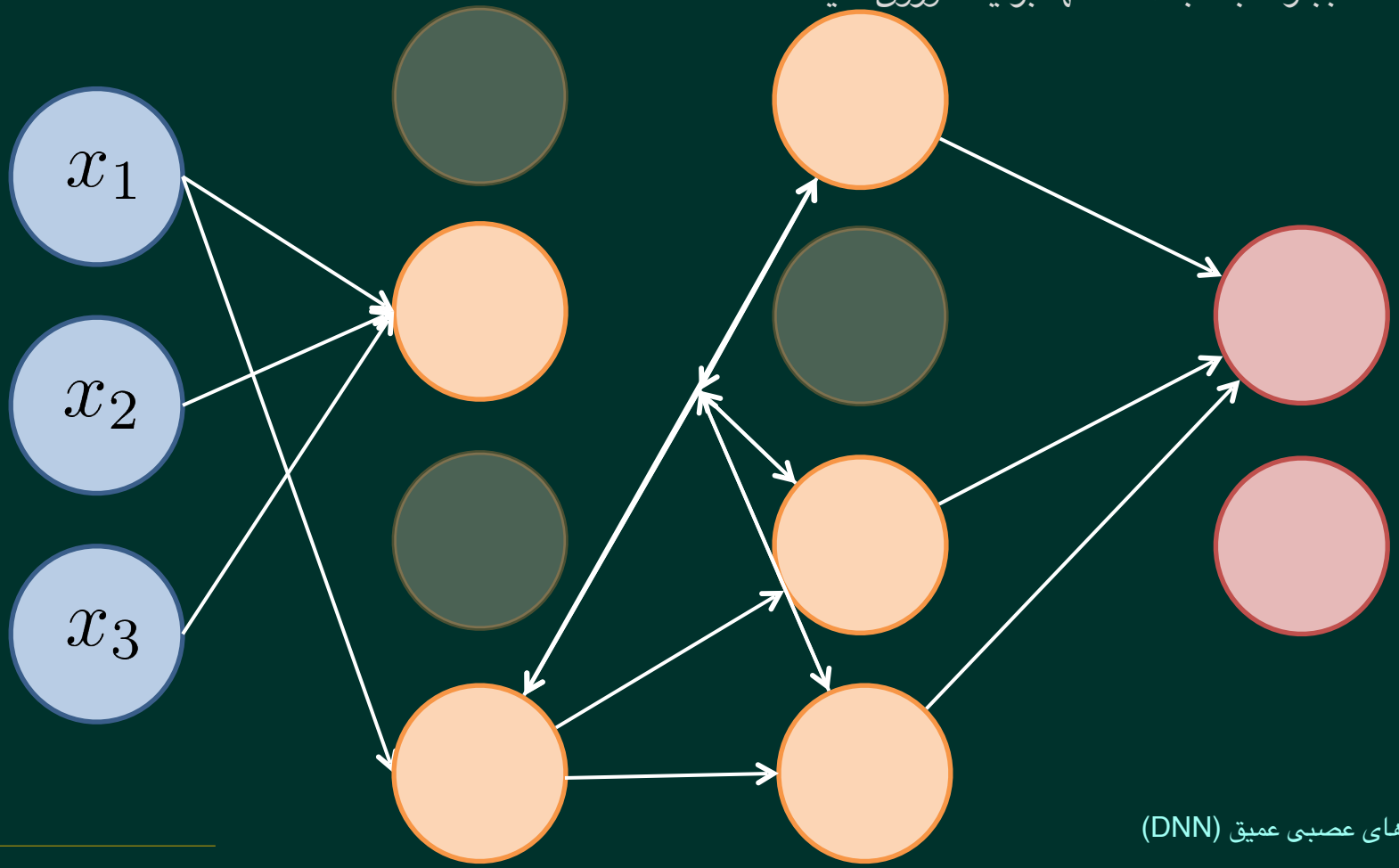
منظم‌سازی ۱: حذف تصادفی (DROPOUT)

- هنگام آموزش، به‌صورت تصادفی برخی از نورون‌ها غیرفعال می‌شوند.



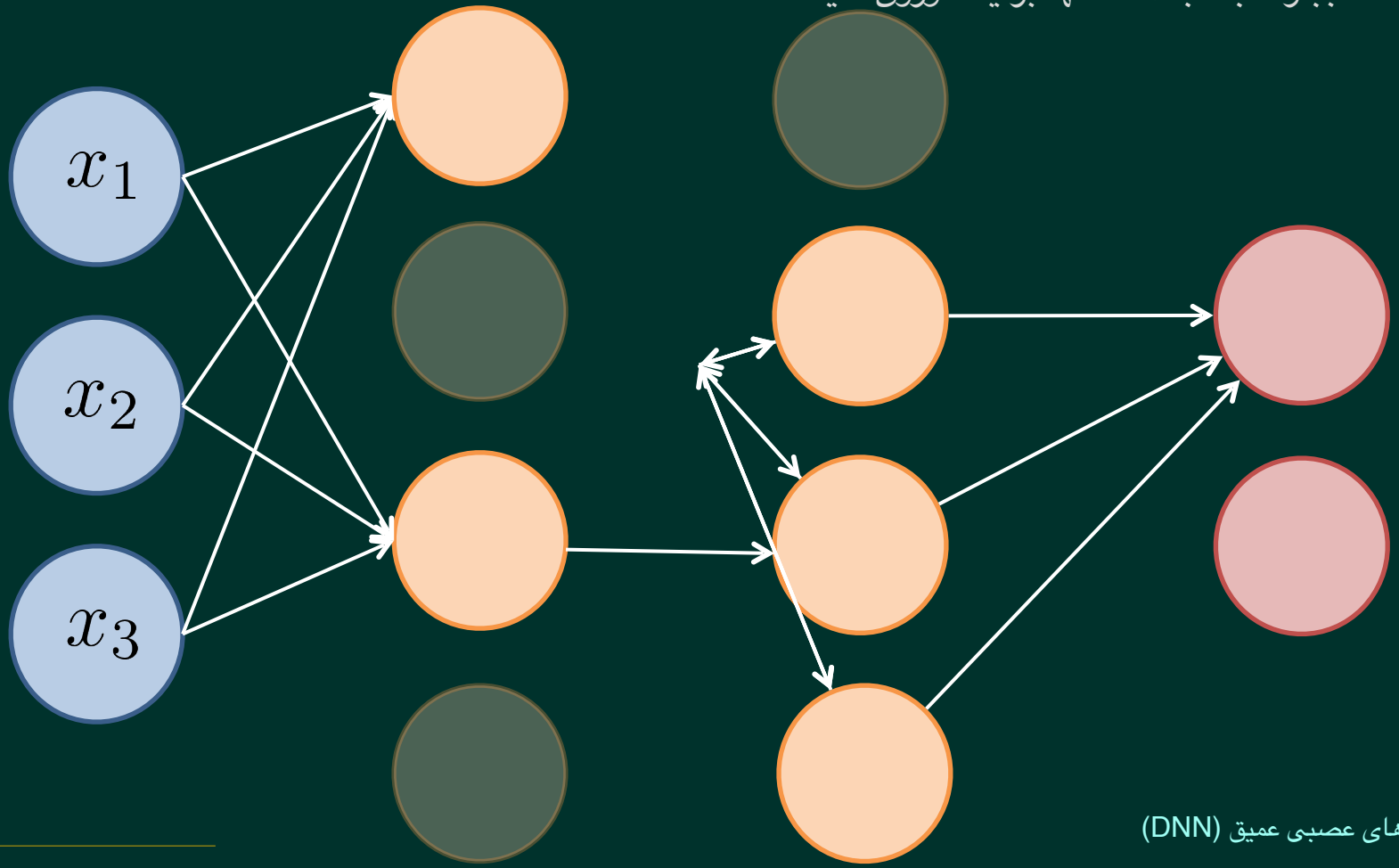
منظم‌سازی ۱: حذف تصادفی (DROPOUT)

- هنگام آموزش، به‌صورت تصادفی برخی از نورون‌ها غیرفعال می‌شوند.
  - به‌صورت معمول، نیمی (۵۰درصد) از نورون‌های هر لایه غیرفعال می‌شوند.
  - اجبار شبکه به آنکه تنها بر یک نورون تکیه نکند.



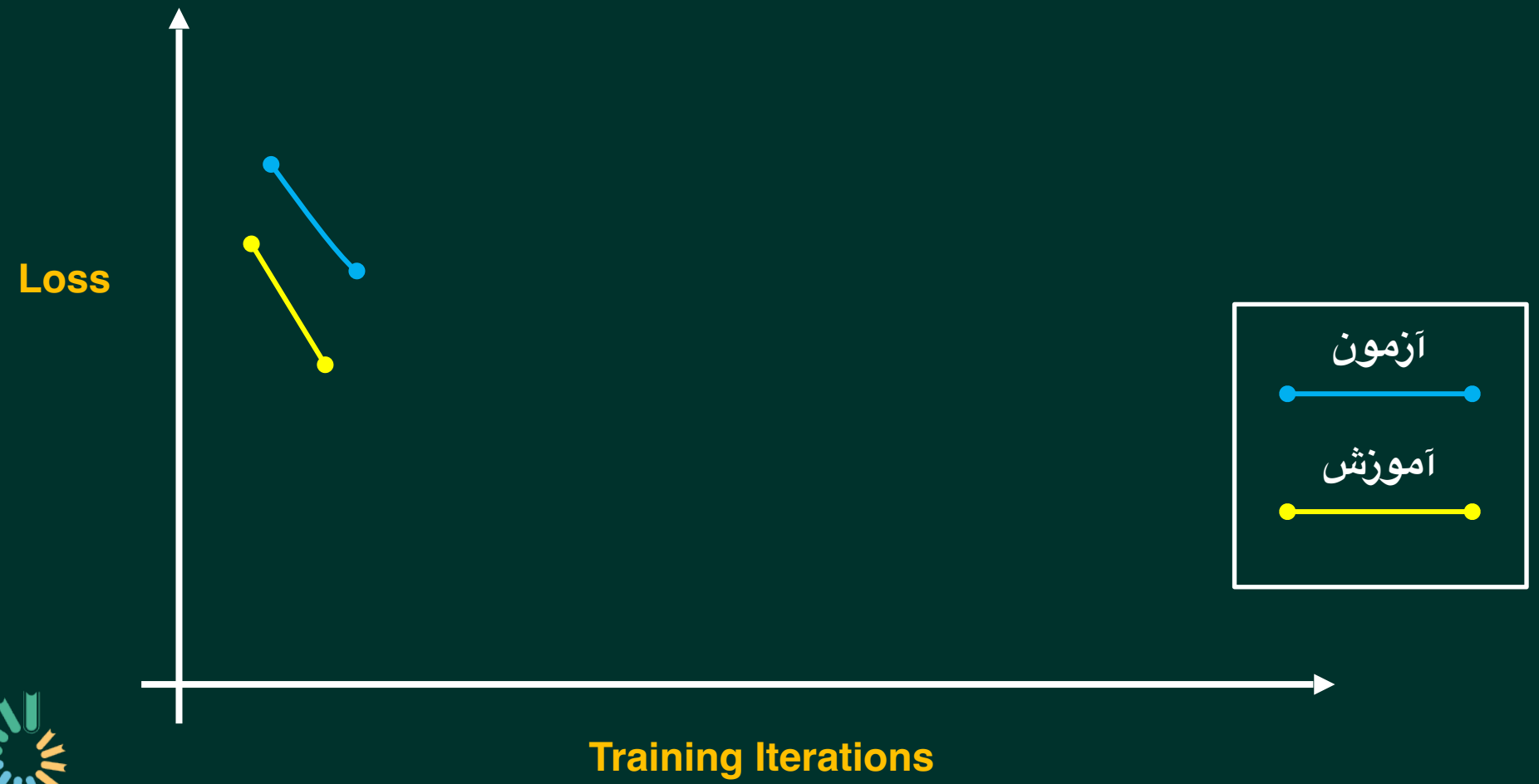
منظم‌سازی ۱: حذف تصادفی (DROPOUT)

- هنگام آموزش، به‌صورت تصادفی برخی از نورون‌ها غیرفعال می‌شوند.
  - به‌صورت معمول، نیمی (۵۰درصد) از نورون‌های هر لایه غیرفعال می‌شوند.
  - اجبار شبکه به آنکه تنها بر یک نورون تکیه نکند.



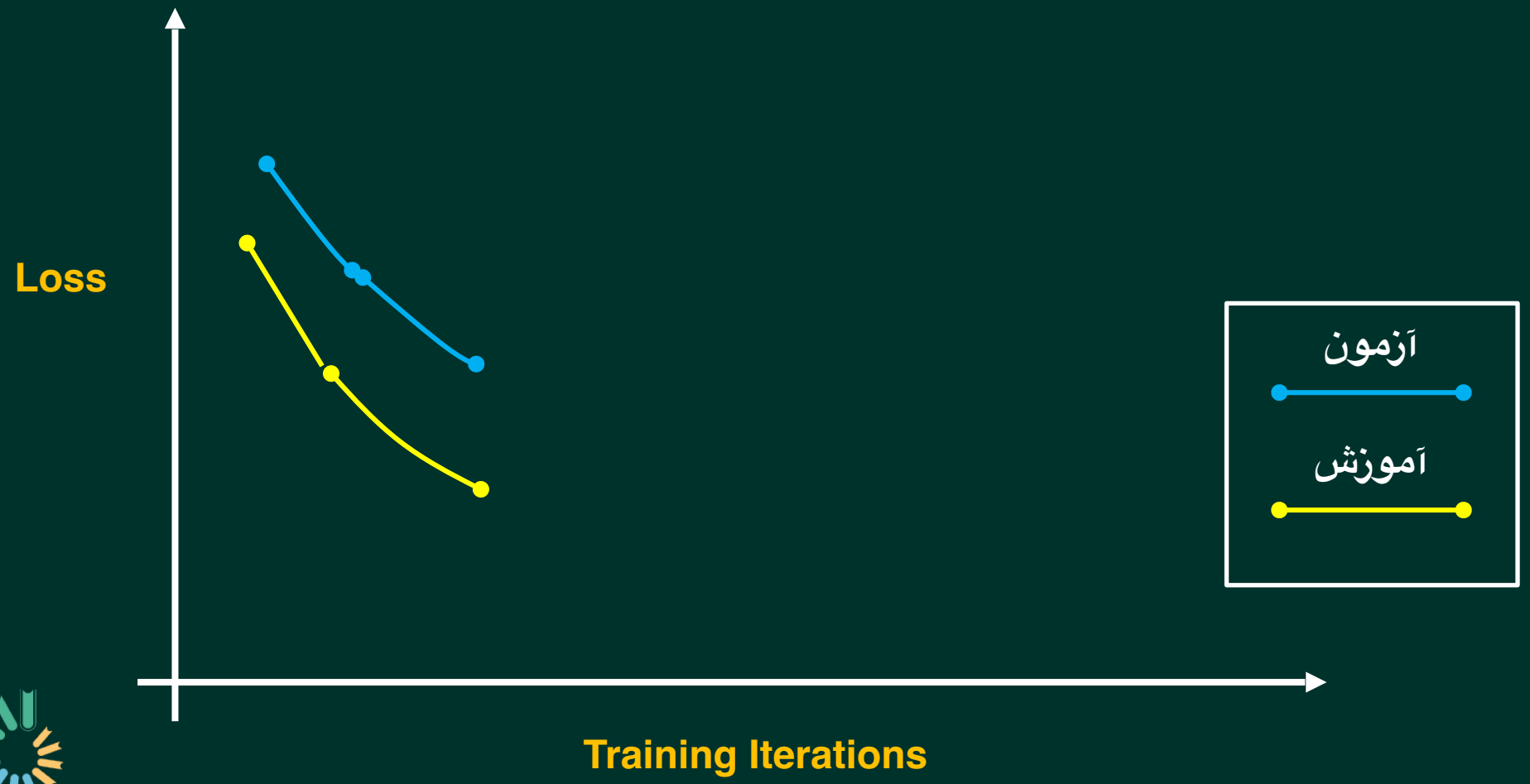
منظم‌سازی ۲: توقف زودرس

- توقف آموزش قبل از آنکه احتمال بیش‌برازش باشد.



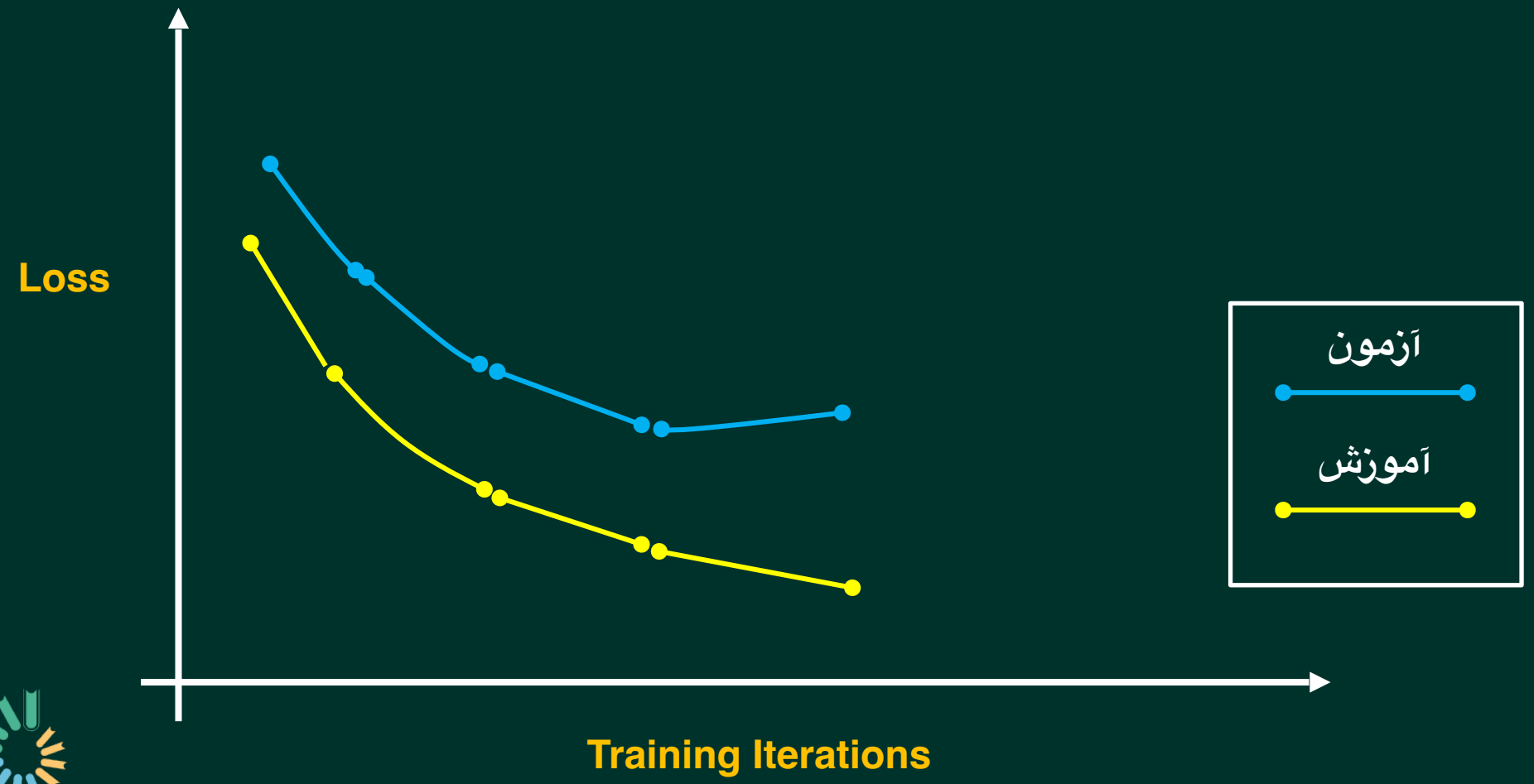
منظم‌سازی ۲: توقف زودرس

- توقف آموزش قبل از آنکه احتمال بیش‌برازش باشد.



منظم‌سازی ۲: توقف زودرس

- توقف آموزش قبل از آنکه احتمال بیش‌برازش باشد.



منظم‌سازی ۲: توقف زودرس

- توقف آموزش قبل از آنکه احتمال بیش‌برازش باشد.

منظم‌سازی ۲: توقف زودرس

- توقف آموزش قبل از آنکه احتمال بیش‌برازش باشد.

نکات اضافی

مقداردهی اولیه وزن‌ها

- مقادیر وزن‌ها به صورت تصادفی مقداردهی اولیه می‌شوند.
- وزن‌ها را نباید با مقدار ۰ مقدار دهی اولیه کرد.
- بهتر است به‌صورت یکنواخت با مقادیر تصادفی کم مقداردهی شوند.
  - مثلا در بازه $[-0.01, \quad 0.01]$

دسته‌بندی داده‌ها BATCHING

در فرآیند آموزش، ارائه هرداده آموزشی منجر به ایجاد تغییراتی در مقدار وزن‌ها می‌شود.

برای افزایش سرعت یادگیری، می‌توان داده‌ها را در بسته‌هایی کوچک (mini batches) دسته‌بندی کرد:

- در هربار به‌روزرسانی مقادیر وزن‌ها، جمع تغییرات حاصل از داده‌های این دسته به مدل اعمال می‌شود

- اندازه‌ی دسته‌ها: معمولا توانی از ۲
- الگوریتم گرادیان کاهشی با بسته‌بندی: گرادیان کاهشی تصادفی

(Stochastic Gradient Descent)

تفاوت EPOCH و ITERATION در آموزش شبکه‌های عصبی

- **دوره (Epoch)**
  - یک بار عبور کامل کل دادهٔ آموزشی
  - مدل تمام نمونه‌ها را حداقل یک بار می‌بیند
- **تکرار (Iteration)**
  - یک گام به‌روزرسانی وزن‌ها
  - هر iteration مربوط به یک batch است
- تعداد iterations در هر epoch:

$$Iteration\ per\ Epoch = \frac{Number\ of\ Samples}{Batch\ Size}$$

تفاوت EPOCH و ITERATION در آموزش شبکه‌های عصبی

- اگر:
  - تعداد نمونه‌ها = 10,000
  - Batch size = 100
- آنگاه:
  - دیدن کل 10,000 نمونه = ۱ epoch
  - پردازش 100 نمونه = ۱ iteration
  - تعداد iterations در هر epoch = 100

الگوریتم گرادیان کاهشی تصادفی
Stochastic Gradient Descent

1. Initialize weights randomly
2. repeat until convergence {
3. Pick Batch of B data points
4. *Compute gradient*, $\frac{\partial}{\partial \boldsymbol{w}} J(\boldsymbol{W}) = \frac{1}{B} \sum_{k=1}^{B} \frac{\partial}{\partial \boldsymbol{w}} J_k(\boldsymbol{W})$
5. Update weights, $\boldsymbol{w} = \boldsymbol{w} - \eta \frac{\partial}{\partial \boldsymbol{w}} J(\boldsymbol{W})$

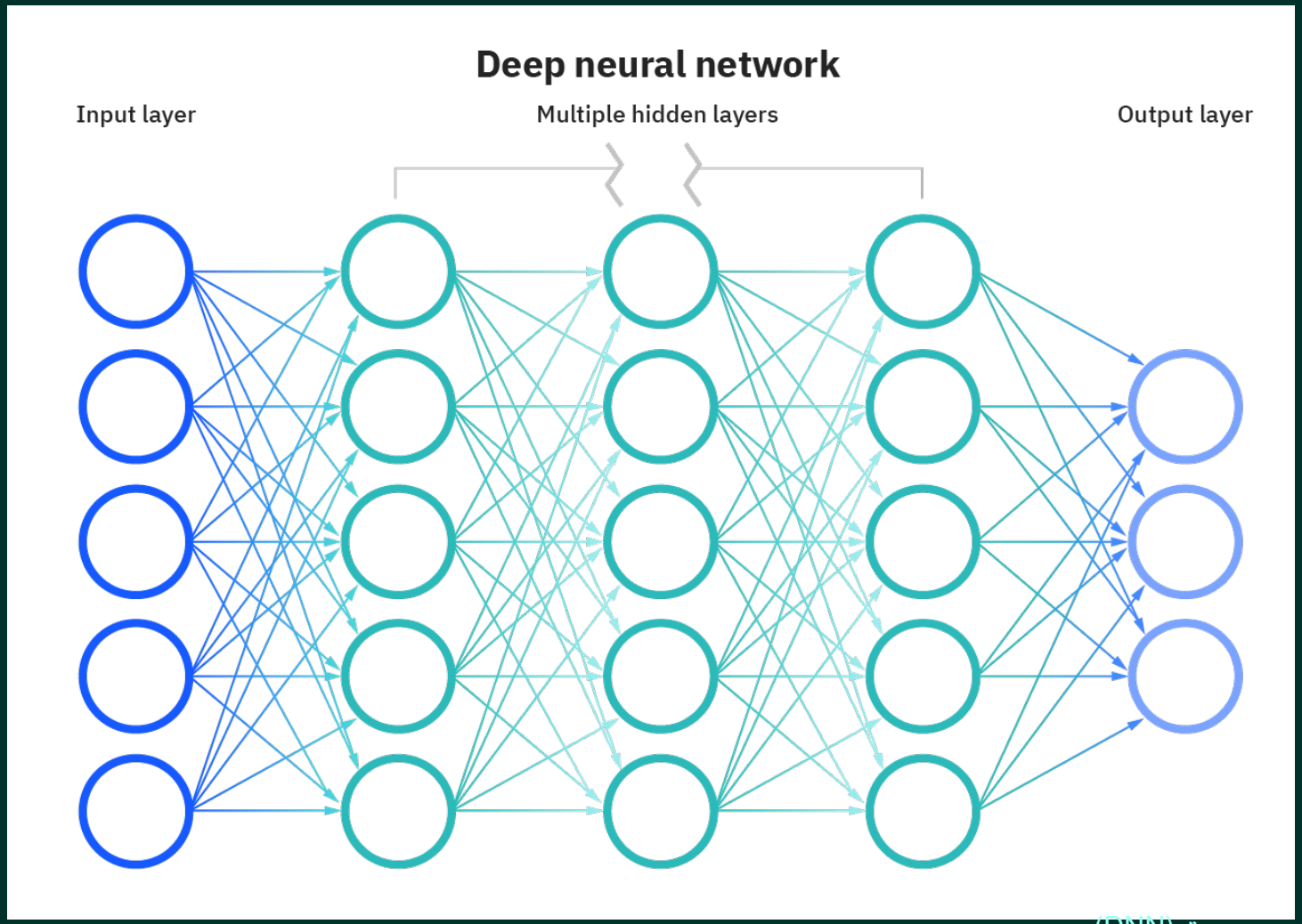
}

6. Return weights

شبکه‌های عصبی عمیق
DEEP NEURAL NETWORKS

یادگیری عمیق

- لایه‌های بیشتر = یادگیری عمیق



شبکه‌های عصبی عمیق (DNN)

شبکه عمیق در مقابل شبکه عصبی معمولی

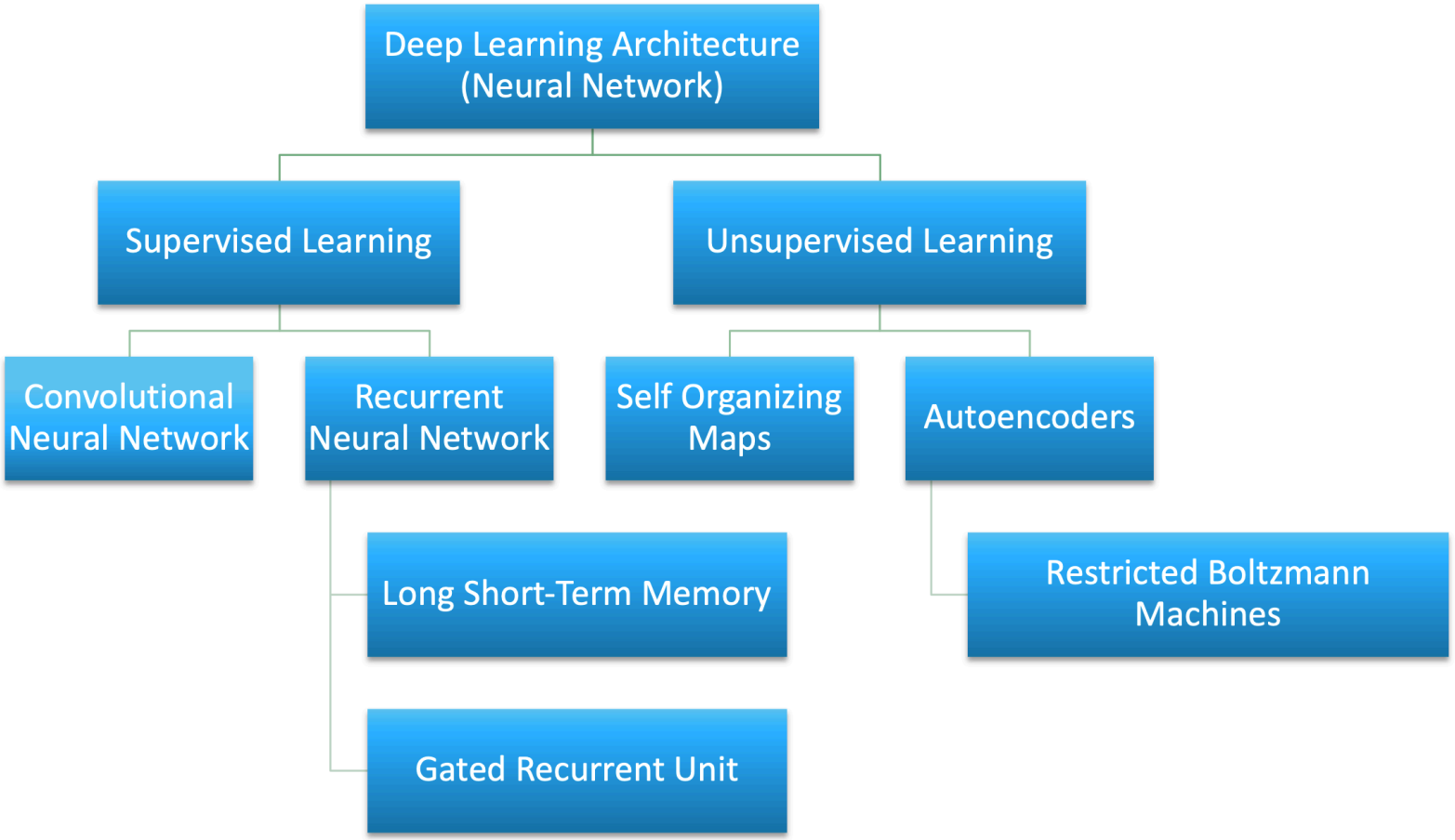
شبکه‌های عصبی عمیق، شبکه‌های عصبی با ابعاد بسیار زیادند.

- عمق: معمولا ۵ لایه (ولی ابعاد برخی تا ۲۰ لایه هم می‌رسد)
  - معمولا تمامی نورون‌ها به یکدیگر متصل نیستند.
- عرض: نورون‌های نهان در لایه‌های میانی معمولا به چندهزار می‌رسند.
- تعداد پارامترها: از میلیون تا میلیارد!

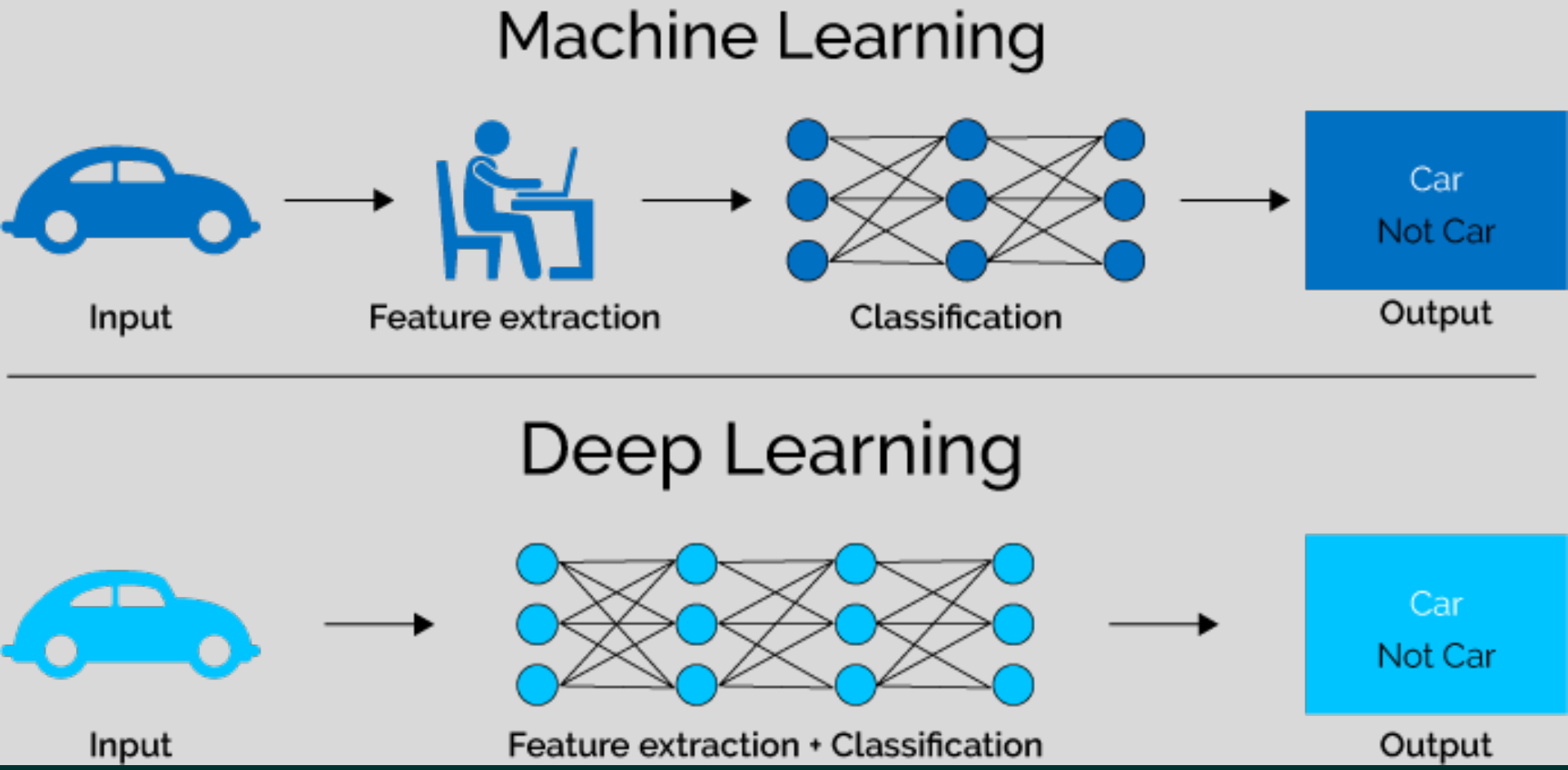
الگوریتم‌ها و محاسبات

- الگوریتم‌های جدید (پیش‌آموزش، آموزش لایه‌محور، ..)
- نیازمند سخت‌افزارهای محاسباتی ویژه (GPU)

انواع شبکه‌های عصبی عمیق



تفاوت یادگیری عمیق با یادگیری ماشین سنتی



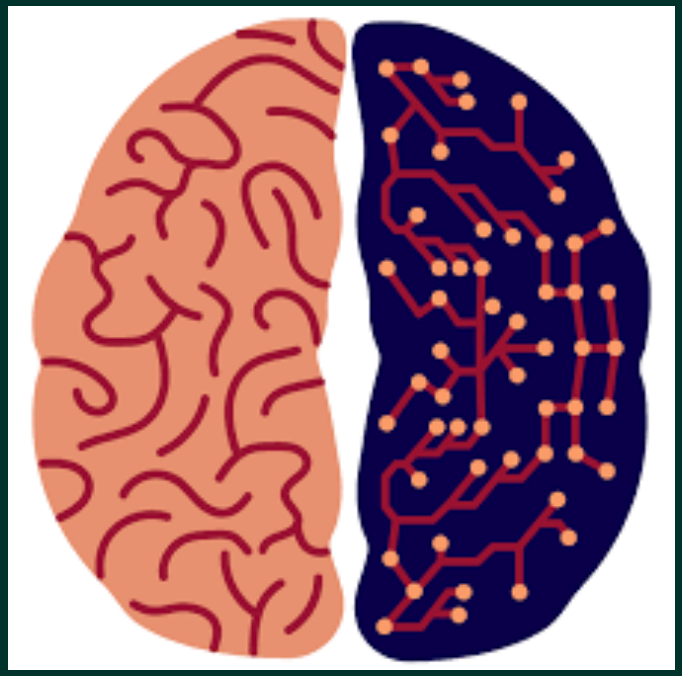
کتابخانه‌های شبکه‌های عصبی و یادگیری عمیق

- Scikit-Learn
- Tensorflow
- PyTorch
- Keras
- Thean
- Theano

مغز و شبکه‌های عصبی مصنوعی

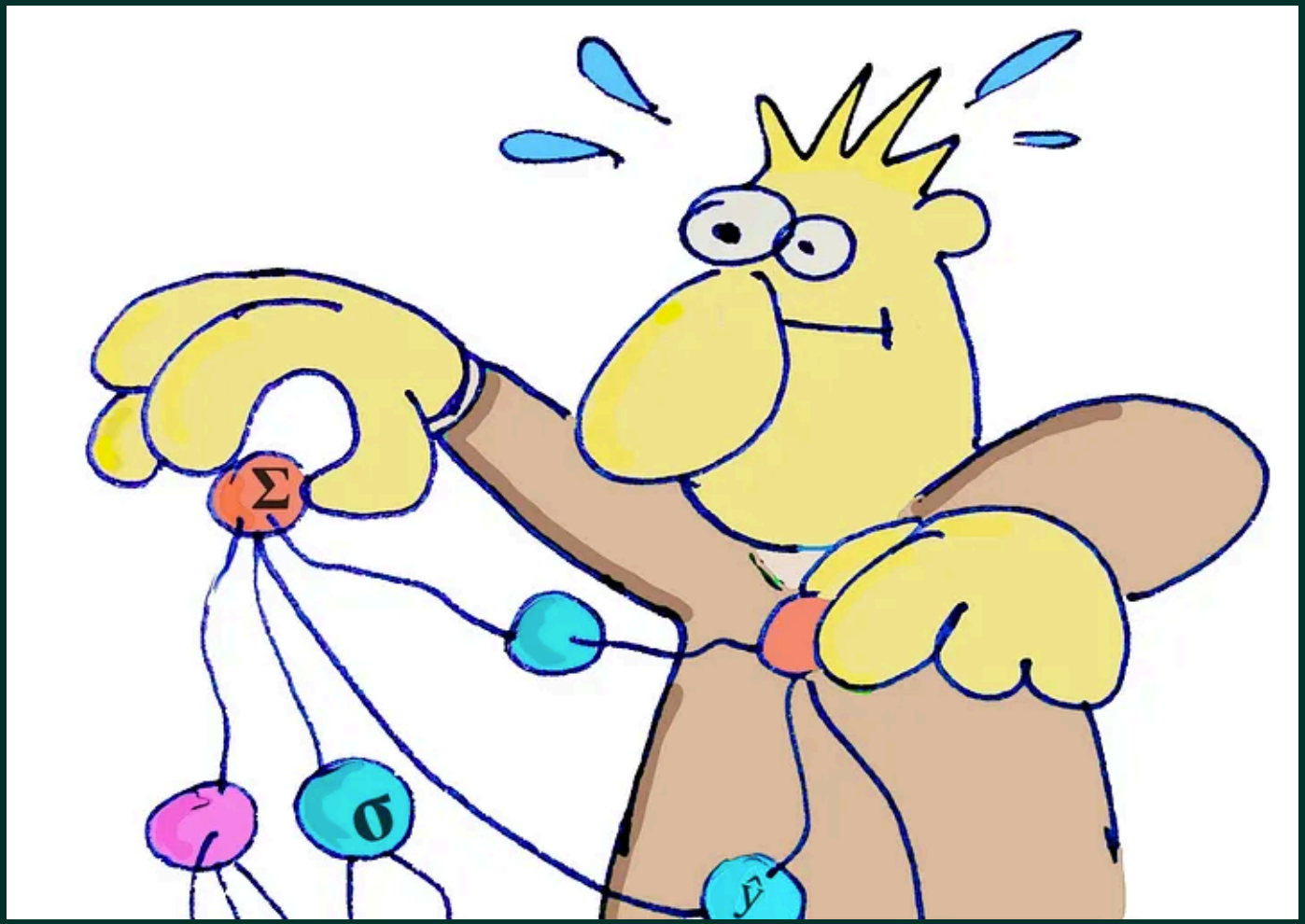
شباهت‌ها

- وجود نورون و ارتباطات میان نورون‌ها
- یادگیری: تغییر در ارتباطات، و نه تغییر در نورون‌ها
- حجم گسترده‌ای از پردازش موازی



تفاوت‌ها

- شبکه‌های عصبی مصنوعی بسیار ساده‌تر از مغزند
  - محاسبات در نورون بسیار ساده‌سازی شده است
  - قدم‌های زمانی گسسته



سوال ؟