بسم الله الرحمن الرحیم

یادگیری عمیق

مدرس: محمدرضا محمدی

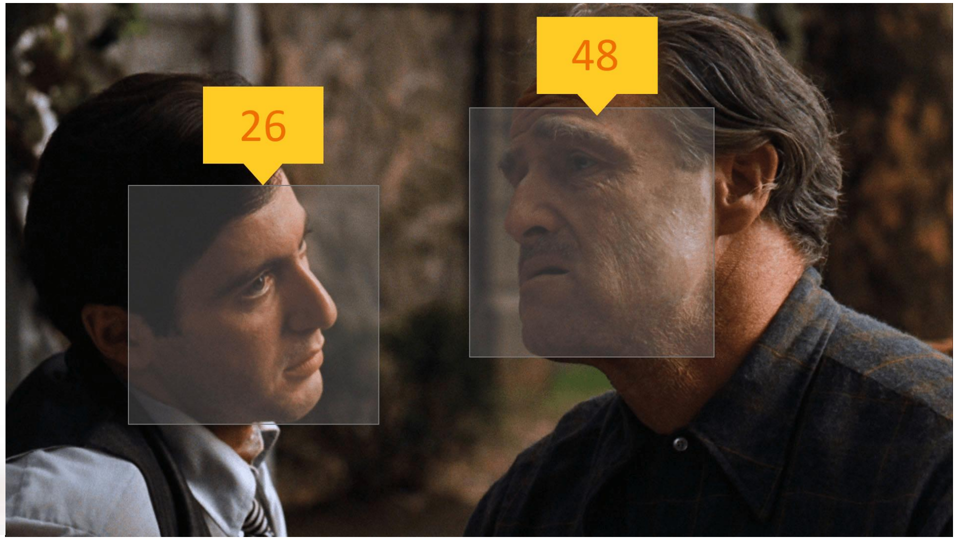
زمستان ۱۴۰۱

شبکه‌های عصبی خطی

Linear Neural Networks

رگرسیون

- رگرسیون به مجموعه‌ای از روش‌ها برای مدل‌سازی رابطه بین یک یا چند متغیر مستقل و یک متغیر وابسته اشاره دارد
- تخمین یک مقدار عددی مورد نظر است



رگرسیون خطی

- فرض می‌شود رابطه میان متغیرهای مستقل $\mathbf{x}$ و متغیر وابسته y خطی است
  - جمع وزن‌دار
- نویز موجود رفتار مناسبی دارد (مانند توزیع نرمال)
- فرض کنید می‌خواهیم قیمت خانه را برحسب مساحت و سال ساخت تخمین بزنیم

features		target
area	age	price
63	13	1.4
78	2	1.95
95	9	2.12
⋮	⋮	⋮

$$\text{price} = w_{\text{area}} . \text{area} + w_{\text{age}} . \text{age} + b$$

sample

$$\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}]^{\mathrm{T}}, \quad y^{(i)}$$

رگرسیون خطی

- رگرسیون خطی برای یک بردار ویژگی d بعدی

$$\hat{y} = w_1 x_1 + \cdots + w_d x_d + b$$

$$\hat{y} = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$$

$$\mathbf{x} \in \mathbb{R}^d \ , \mathbf{w} \in \mathbb{R}^d$$

- برای یک مجموعه داده شامل n نمونه

$$\mathbf{X} \in \mathbb{R}^{n \times d}$$

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b$$

- هدف این است که بتوانیم پارامترهای $\mathbf{w}$ و b را به گونه‌ای تخمین بزنیم که برای یک نمونه داده جدید که از همان توزیع $\mathbf{X}$ نمونه‌برداری شده باشد، پیش‌بینی با حداقل خطا را داشته باشد

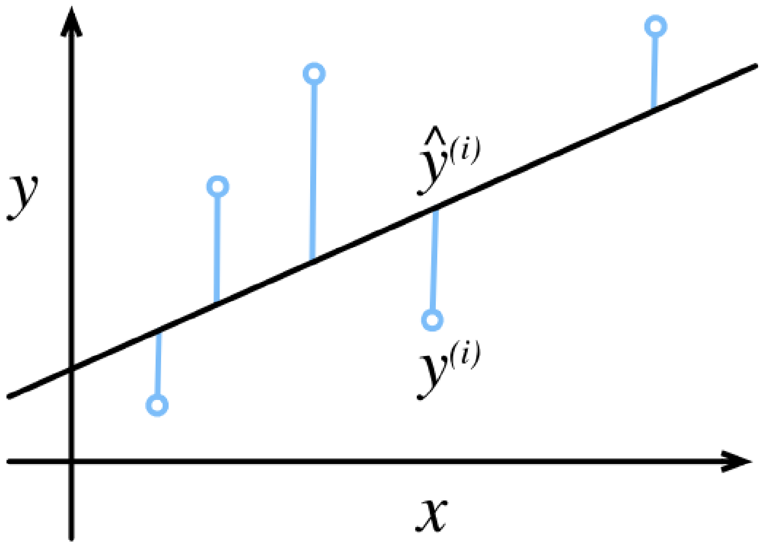
رگرسیون خطی

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b$$

- حتی اگر بهترین مدل برای پیش‌بینی y بر اساس $\mathbf{x}$ خطی باشد، نمی‌توان توقع داشت در مجموعه داده واقعی تمام مقادیر $\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + b$ دقیقا برابر با $y^{(i)}$ باشند
  - ممکن است مقداری خطای اندازه‌گیری داشته باشیم (نویز)
- برای یافتن مقادیر مناسب پارامترهای مدل نیاز است تا یک معیار مناسب برای ارزیابی عملکرد مدل تعریف کنیم

تابع ضرر

- تابع ضرر فاصله میان مقادیر واقعی و مقادیر پیش‌بینی شده را اندازه‌گیری می‌کند
- به طور معمول مقدار تابع ضرر یک عدد نامنفی است که مقادیر کوچکتر بهتر هستند و در حالت کاملا دقیق به ۰ می‌رسد
- مربع خطا یکی از توابع پرکاربرد در مسئله رگرسیون است



$$l^{(i)}(\mathbf{w}, b) = \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + b - y^{(i)}\right)^2$$

$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\arg\min}\, L(\mathbf{w}, b)$$

بهینه‌سازی

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + b - y^{(i)}\right)^2$$

- برای سادگی می‌توانیم b را به $\mathbf{w}$ و یک مقدار ۱ را به هر $\mathbf{x}^{(i)}$ الحاق کنیم

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$

- این مسئله یک مینیمم محلّی دارد که مشتق در آن صفر است و به سادگی قابل محاسبه است

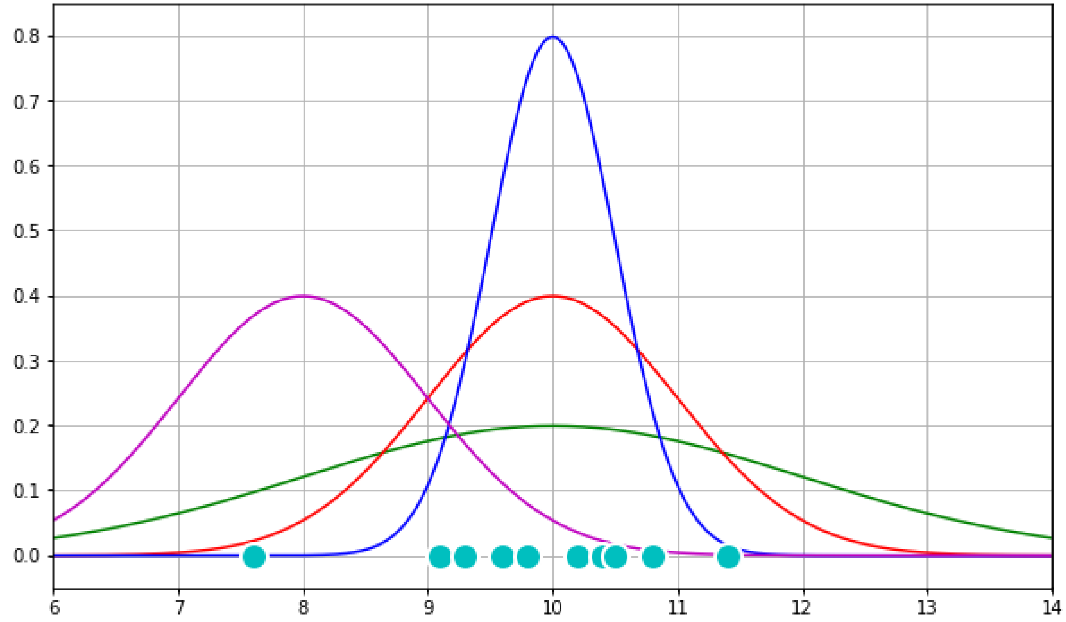
$$\nabla_{\mathbf{w}}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 = 2\mathbf{X}^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w}^* = \mathbf{X}^{\mathrm{T}}\mathbf{y} \Rightarrow \mathbf{w}^* = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} = \mathbf{X}^{\dagger}\mathbf{y}$$

برآوردگر Maximum Likelihood

- فرض کنید ۱۰ داده از یک فرآیند مشاهده شده است
  - به عنوان نمونه، زمان پاسخ دادن به یک سوال بر حسب ثانیه
- چگونه این فرآیند را مدل کنیم؟
  - یک راه این است که یک مدل برای توزیع آماری فرآیند فرض کنیم
  - سپس، پارامترهای مدل را به گونه‌ای تعیین کنیم که تابع درست‌نمایی مربوط به مشاهده این نمونه‌ها از آن توزیع را بیشینه کنند

$$\max \prod_{i=1}^{n} \mathcal{N}\left(x^{(i)} \mid \mu, \sigma^2\right)$$



رگرسیون خطی

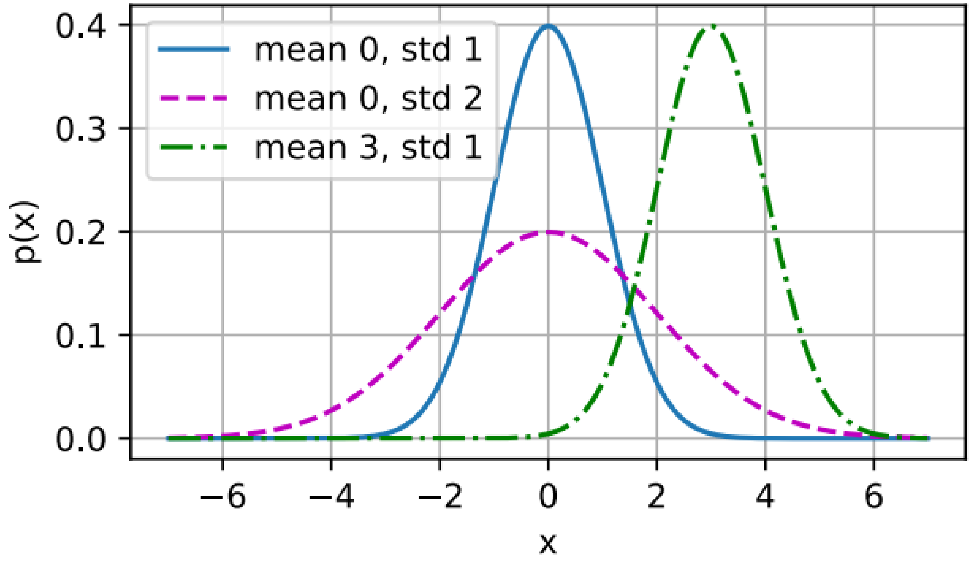
- فرض کنید مشاهدات از یک مدل خطی همراه با نویز نرمال (گاوسی) جمع‌آوری شده‌اند

$$\hat{y} = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b + \epsilon$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

- توزیع نرمال

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$



رگرسیون خطی

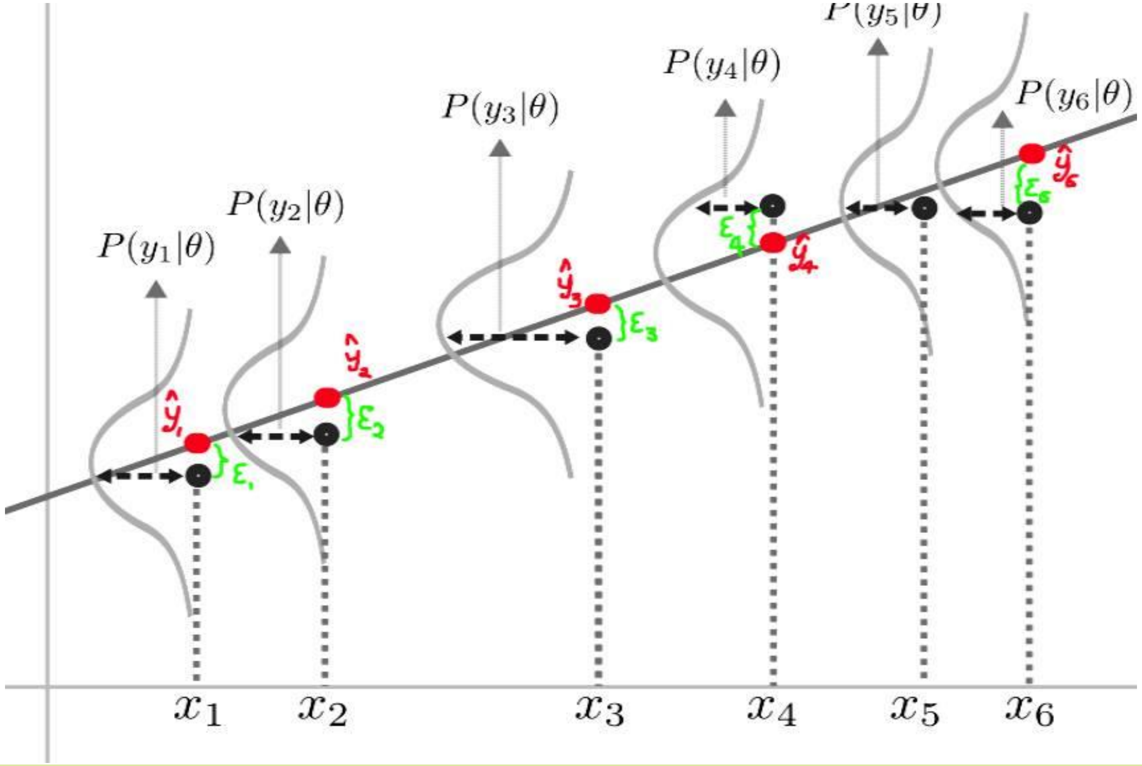
- فرض کنید مشاهدات از یک مدل خطی همراه با نویز نرمال (گاوسی) جمع‌آوری شده‌اند

$$\hat{y} = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b + \epsilon$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

$$P(y \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(y - \mathbf{w}^{\mathrm{T}}\mathbf{x} - b\right)^2\right)$$

$$P(\mathbf{y} \mid \mathbf{X}) = \prod_{i=1}^{n} P\left(y^{(i)} \mid \mathbf{x}^{(i)}\right)$$

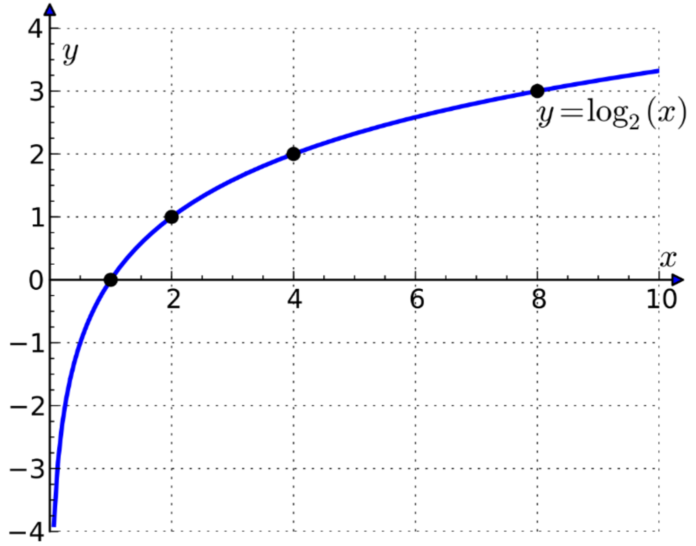


Maximum Likelihood

$$P(y \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(y - \mathbf{w}^{\mathrm{T}}\mathbf{x} - b\right)^2\right)$$

$$P(\mathbf{y} \mid \mathbf{X}) = \prod_{i=1}^{n} P\left(y^{(i)} \mid \mathbf{x}^{(i)}\right)$$

- بجای بیشینه‌سازی $P(\mathbf{y} \mid \mathbf{X})$ می‌توان لگاریتم آن را بیشینه کرد
  - ضرب به جمع تبدیل می‌شود



$$\log P(\mathbf{y} \mid \mathbf{X}) = \sum_{i=1}^{n} \log P\left(y^{(i)} \mid \mathbf{x}^{(i)}\right)$$

$$= \sum_{i=1}^{n} -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(y - \mathbf{w}^{\mathrm{T}}\mathbf{x} - b\right)^2$$

Maximum Likelihood

- به طور معمول منفی لگاریتم تابع درست‌نمایی را کمینه می‌کنیم

$$-\log P(\mathbf{y} \mid \mathbf{X}) = \sum_{i=1}^{n} \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\left(y - \mathbf{w}^{\mathrm{T}}\mathbf{x} - b\right)^2$$

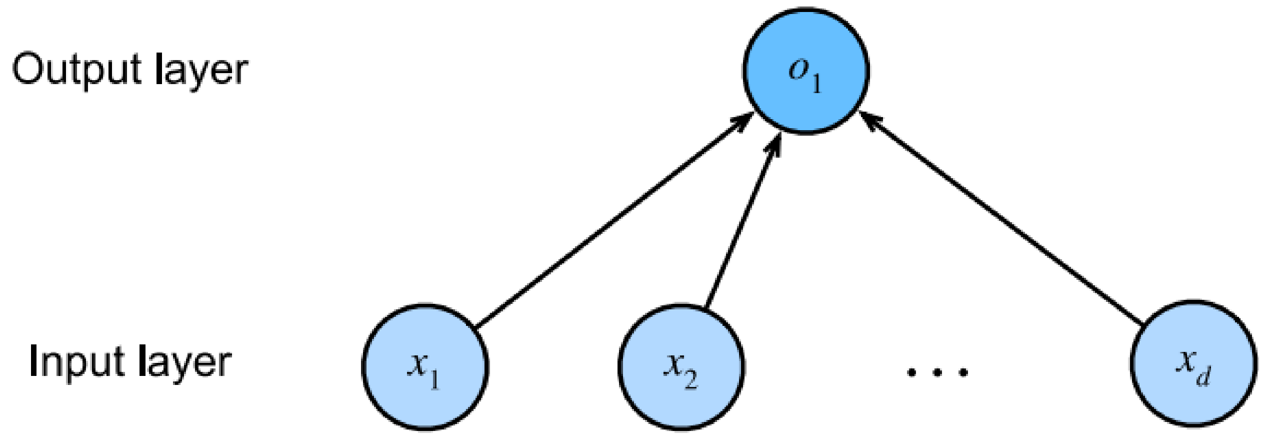
- اگر σ^2 را ثابت در نظر بگیریم

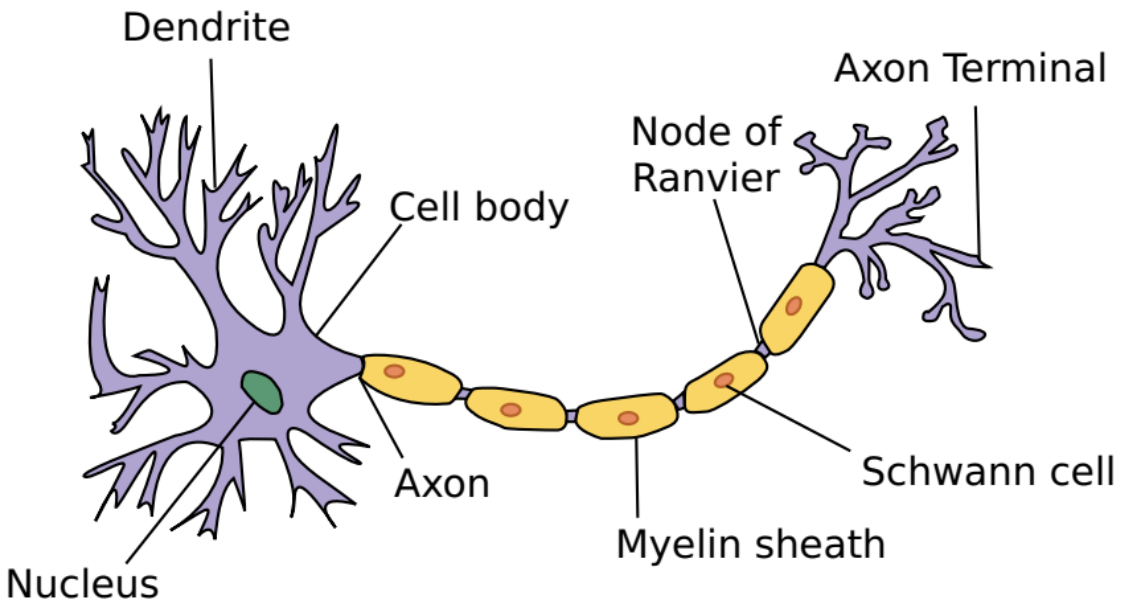
$$\min \sum_{i=1}^{n}\left(y - \mathbf{w}^{\mathrm{T}}\mathbf{x} - b\right)^2$$

- همان کمینه‌سازی مجموع مربعات خطا است!
- با استفاده از رویکرد ML می‌توان توابع ضرر متناسب با مسئله را بدست آورد

شبکه‌های عصبی

- رگرسیون خطی یک شبکه عصبی یک لایه است
- در این مثال واحد خروجی به تمام واحدهای ورودی متصل است و به آن لایه کاملا متصل می‌گوئیم
- مدل نورون‌های زیستی از لحاظ ساختاری مشابه با این ساختار است

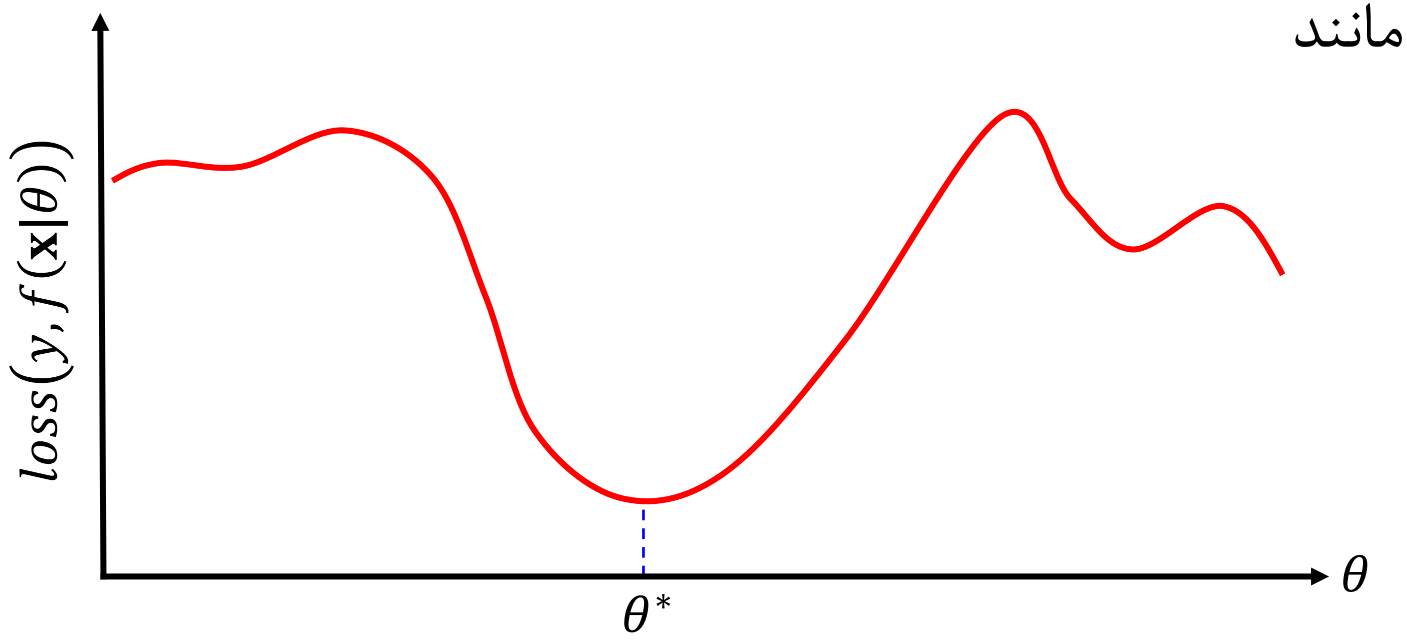




بهینه‌سازی

$$\theta^* = \min_{\theta} \; loss\big(y, f(\mathbf{x}|\theta)\big)$$

- در حالت‌هایی که فضای جستجو کوچک باشد می‌توان تمام فضا را جستجو کرد
- اگر فضای جستجو پیوسته اما ساده باشد، می‌توانیم مشتق بگیریم و مساوی با صفر قرار دهیم
- در غیر این صورت، باید از روش‌های تقریبی مانند الگوریتم گرادیان کاهشی استفاده کنیم



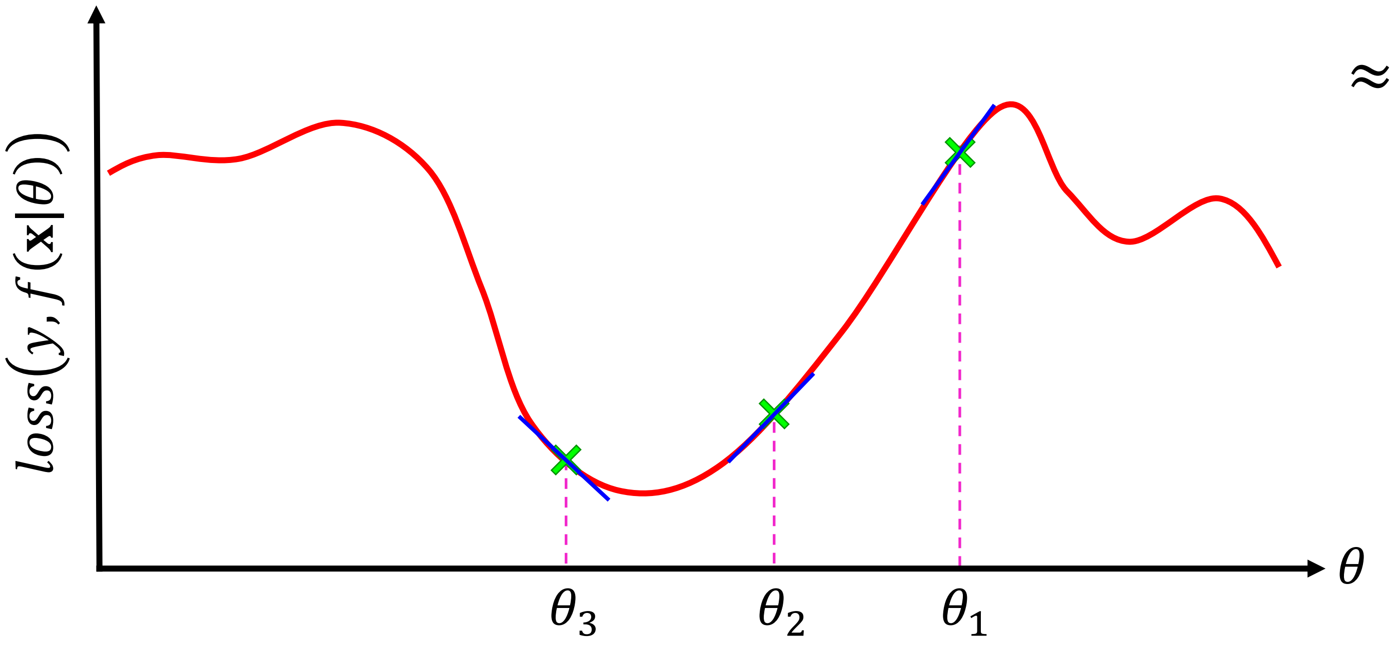
بهینه‌ساز گرادیان کاهشی

- با یک نقطه اولیه شروع می‌کنیم و در هر گام در جهتی حرکت می‌کنیم که منجر به کاهش تابع شود

$$L(\theta+\Delta\theta)=L(\theta)+\Delta\theta\frac{\partial L(\theta)}{\partial\theta}+\frac{(\Delta\theta)^2}{2!}\frac{\partial^2 L(\theta)}{\partial\theta^2}+\cdots$$

$$\approx L(\theta)+\Delta\theta\frac{\partial L(\theta)}{\partial\theta}$$

- در خلاف جهت گرادیان حرکت می‌کنیم



دسته‌بندی باینری

- در بسیاری از مسائل یادگیری ماشین نیاز به پیش‌بینی مقدار یک متغیر باینری است
- شبکه‌های عصبی باید تنها یک مقدار را پیش‌بینی کنند $P(y = 1 \mid \mathbf{x}) \in [0,1]$
- بهتر است خروجی محدود به بازه [0,1] باشد

	Two Class Classification	
$y \in \{0, 1\}$	**1 or Positive Class**	**0 or Negative Class**
Email	Spam	Not Spam
Tumor	Malignant	Benign
Transaction	Fraudulent	Not Fraudulent

$$\text{e.g. } P(y = 1|\mathbf{x}) = \max\{0, \min\{1, \mathbf{w}^{\mathrm{T}}\mathbf{x} + b\}\}$$

- بهینه‌سازهای مبتنی بر گرادیان نمی‌توانند پارامترهای مطلوب چنین شبکه‌ای را بیابند
  - مشتق تابع ضرر برای داده‌هایی که کاملا اشتباه پیش‌بینی شوند ۰ است

دسته‌بندی باینری

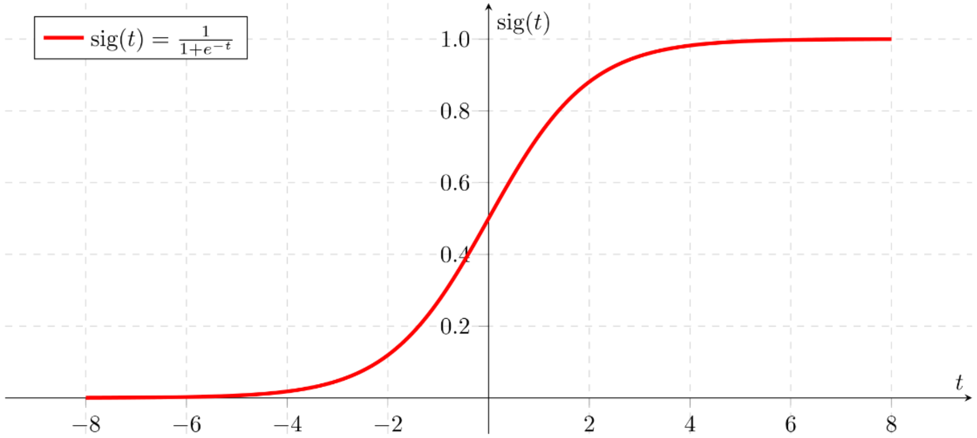
- بهتر است از تابعی برای محدود کردن خروجی استفاده کنیم که مشتق آن صفر نشود
- تابع سیگموئید برای این منظور پرکاربرد است $P(y=1\mid\mathbf{x})=\sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b\right)$
- آیا تابع ضرر MSE در این حالت مناسب است؟

$$l=\left(y-\sigma(o)\right)^2,\qquad o=\mathbf{w}^T\mathbf{x}+b$$

- مثال عددی: اگر $o=5.6$ و $y=0$

$$\frac{dl}{do}=-2\left(y-\sigma(o)\right)\sigma(o)\left(1-\sigma(o)\right)=0.0073$$

$$\approx -2(0-1)1(1-1)$$



Maximum Likelihood

$$-\log P(\mathbf{y} \mid \mathbf{X}) = -\sum_{i=1}^{n} \log P\left(y^{(i)} \mid \mathbf{x}^{(i)}\right)$$

- y یک متغیر باینری است و احتمال $P(y \mid \mathbf{x})$ به صورت زیر قابل بیان است

$$P(y \mid \mathbf{x}) = \begin{cases} \hat{y} & \text{if } y = 1 \\ 1 - \hat{y} & \text{if } y = 0 \end{cases} = \hat{y}^{y}(1 - \hat{y})^{1-y}$$

Bernoulli distribution

$$-\log P(\mathbf{y} \mid \mathbf{X}) = -\sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

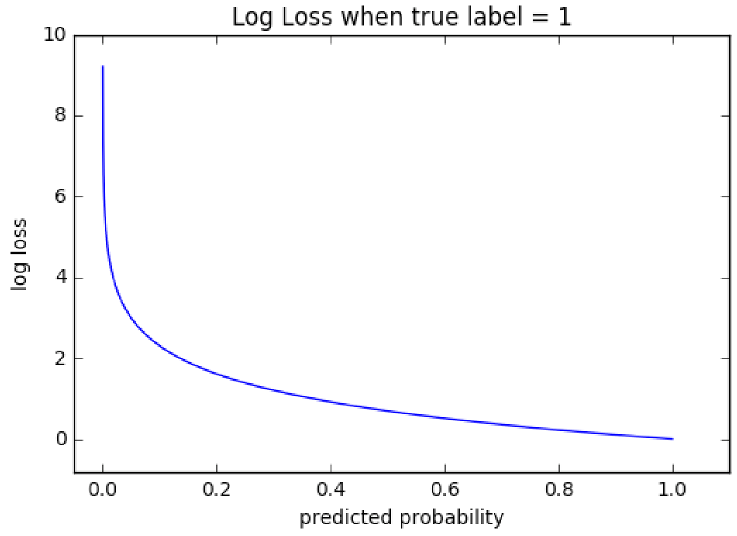
- تابع ضرر binary cross-entropy نامیده می‌شود که برای مسائل دسته‌بندی باینری پرکاربرد است

دسته‌بندی باینری

- بهتر است از تابعی برای محدود کردن خروجی استفاده کنیم که مشتق آن صفر نشود
- تابع سیگموئید برای این منظور پرکاربرد است $P(y=1\,|\,\mathbf{x}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b\right)$
- با استفاده از ML تابع ضرر binary cross-entropy بدست می‌آید

$$l = -y\log\sigma(o) - (1-y)\log\left(1-\sigma(o)\right)$$

- مثال عددی: اگر $o = 5.6$ و $y = 0$



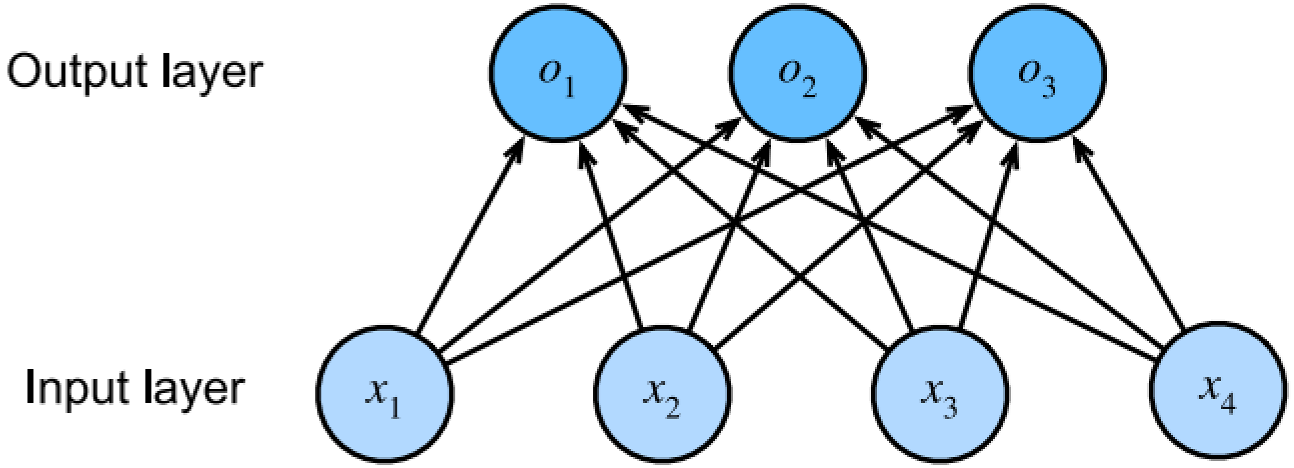
$$\frac{dl}{do} = -\frac{-\sigma(o)\left(1-\sigma(o)\right)}{1-\sigma(o)} = \sigma(o) = 0.9963$$

دسته‌بندی چندکلاسه

$$o_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1$$

$$o_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2$$

$$o_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3$$



- تعیین دسته یک نمونه جدید از میان چندین دسته دارای کاربردهای بسیار فراوانی است
- می‌توان برای هر کلاس یک نورون قرار داد تا احتمال تعلق نمونه به آن کلاس را تخمین بزند
- با استفاده از یک لایه خطی احتمال غیرنرمالیزه **o** را پیش‌بینی می‌کنیم که logit نامیده می‌شوند
- چطور نرمالیزه کنیم؟

دسته‌بندی چندکلاسه

- می‌خواهیم احتمال پسین مربوط به هر کلاس را تخمین می‌زنیم $P(y = i \mid \mathbf{x}) \in [0,1]$
- برای آنکه مقادیر خروجی از جنس احتمال باشند (هر کدام نامنفی و مجموع برابر با ۱) می‌توانیم از تابع فعال‌سازی Softmax استفاده کنیم که تعمیم تابع Sigmoid است

$$\mathbf{o} = \mathbf{W}^{\mathrm{T}}\mathbf{x} + \mathbf{b}$$

$$\hat{y}_i = \mathrm{softmax}(\mathbf{o})_i = \frac{\exp(o_i)}{\sum_{k=1}^{q} \exp(o_k)}$$

- تابع ضرر متناسب با این تابع فعال‌سازی، categorical cross-entropy است

$$l(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{q} y_i \log \hat{y}_i$$

https://drive.google.com/file/d/11FmT5RE3ujFax4ggIMtq62dYElOOx8B4/view?usp=sharing

مشتق softmax

$$l(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{q} y_i \log \frac{\exp(o_i)}{\sum_{k=1}^{q} \exp(o_k)} = \sum_{i=1}^{q} y_i \log \sum_{k=1}^{q} \exp(o_k) - \sum_{i=1}^{q} y_i o_i$$

$$= \log \sum_{k=1}^{q} \exp(o_k) \sum_{i=1}^{q} y_i - \sum_{i=1}^{q} y_i o_i = \log \sum_{k=1}^{q} \exp(o_k) - \sum_{i=1}^{q} y_i o_i$$

$$\partial_{o_i} l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\exp(o_i)}{\sum_{k=1}^{q} \exp(o_k)} - y_i = \mathrm{softmax}(\mathbf{o})_i - y_i$$

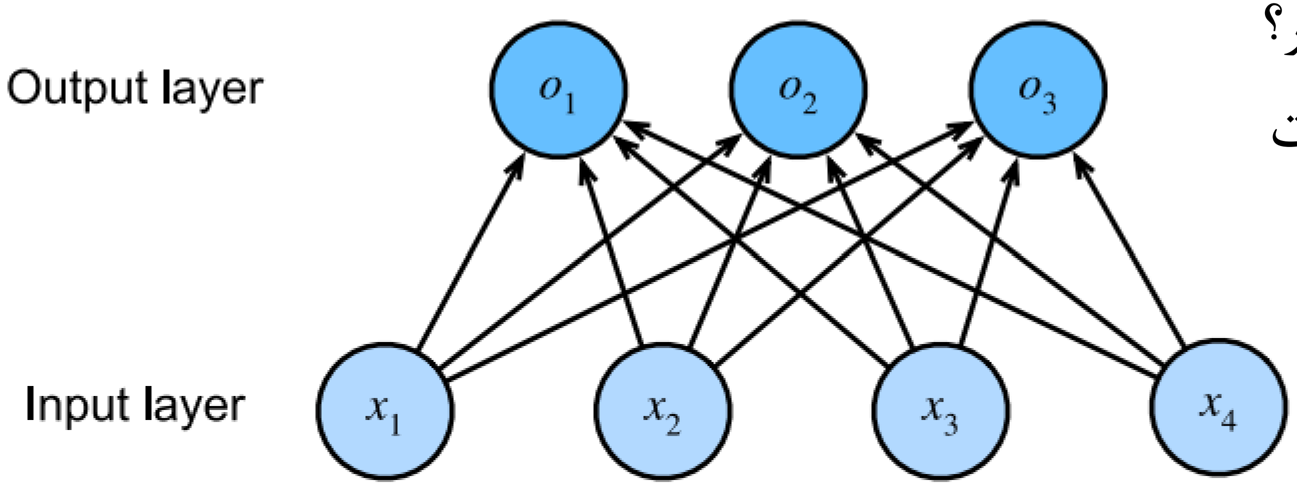
- مشابه با رگرسیون خطی، گرادیان تابع ضرر نسبت به خروجی بخش خطی برابر با میزان اختلاف پیش‌بینی با مقدار واقعی است

پرسپترون چندلایه

Multilayer Perceptron

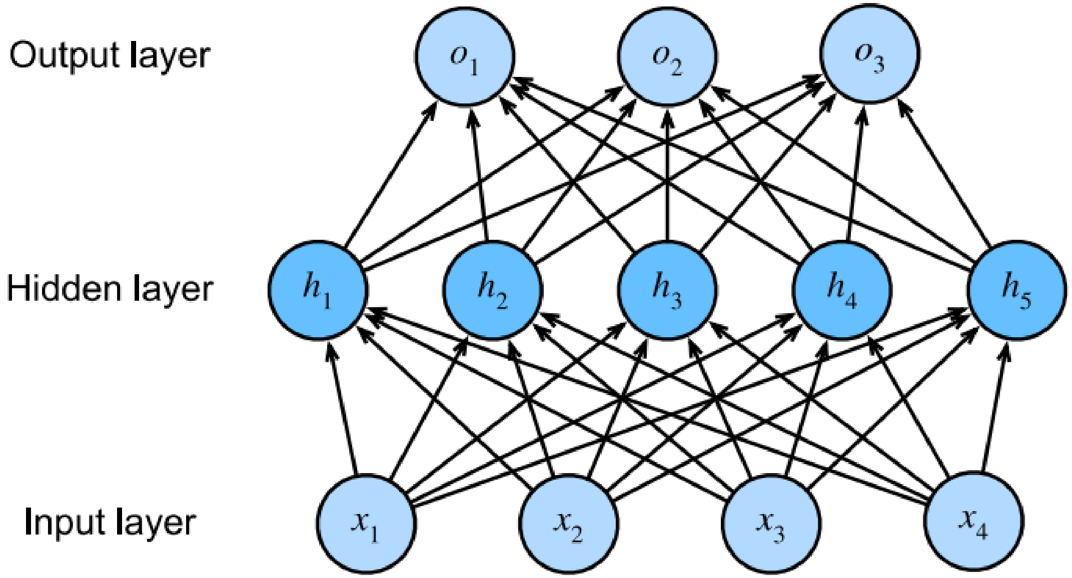
مدل‌های خطی

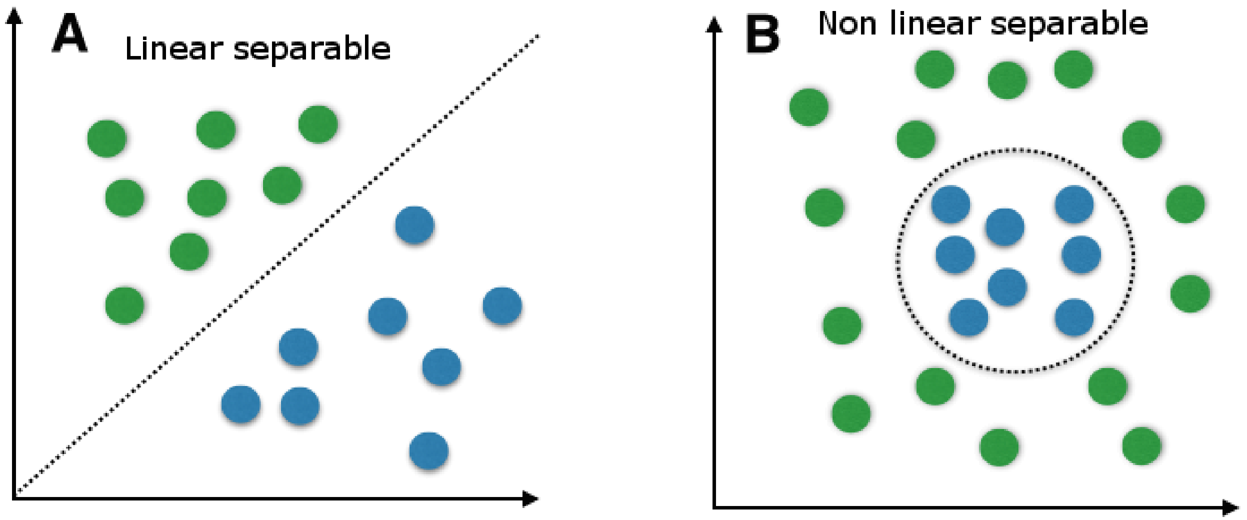
- بر فرض یکنواختی دلالت دارد
  - هر افزایشی در ویژگی همواره منجر به افزایش (کاهش) خروجی می‌شود اگر وزن مربوطه + (-) باشد
  - مثال: احتمال بازپرداخت اقساط بر حسب ویژگی‌هایی شامل درآمد
    - اثر افزایش درآمد از ۰ به ۵ میلیون متناظر با افزایش از ۱۰۰ به ۱۰۵ میلیون است
  - مثال: احتمال مرگ بر حسب ویژگی‌هایی شامل دمای بدن
    - هر چه دما بالاتر باشد ریسک بیشتری دارد یا کمتر؟
- استخراج ویژگی قبل از لایه خطی بسیار مهم است



پرسپترون چندلایه

- ساده‌ترین شبکه عمیق است که از چندین لایه کاملا متصل تشکیل می‌شود
- وظیفه لایه‌های میانی استخراج بازنمایی مناسبی است که مسئله مورد نظر در فضای جدید به صورت خطی قابل حل باشد



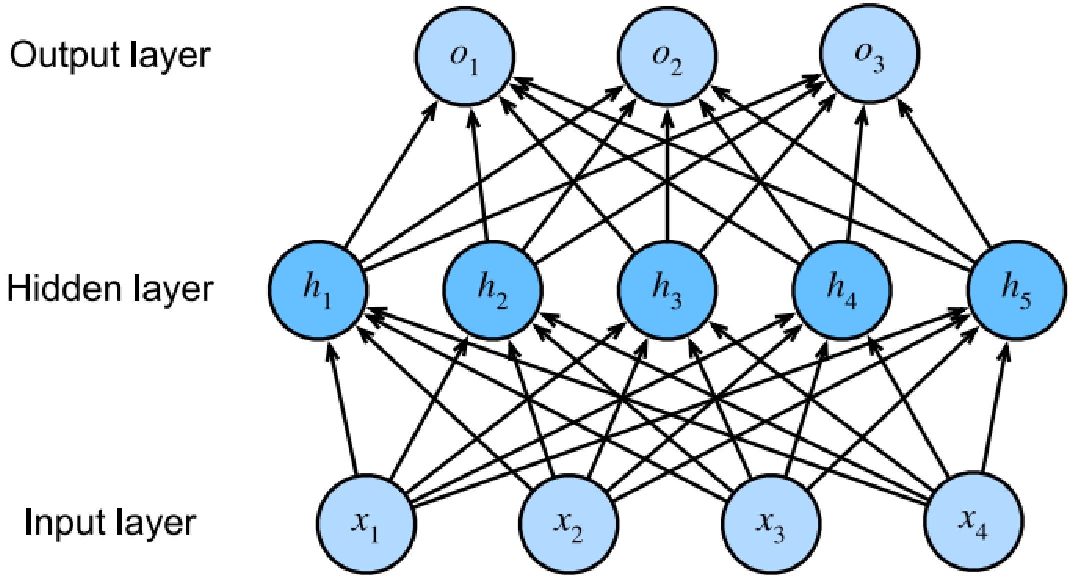


پرسپترون چندلایه

$$\mathbf{X} \in \mathbb{R}^{n\times d}$$

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)} \qquad \mathbf{H} \in \mathbb{R}^{n\times h} \qquad \mathbf{W}^{(1)} \in \mathbb{R}^{d\times h} \qquad \mathbf{b}^{(1)} \in \mathbb{R}^{1\times h}$$

$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \qquad \mathbf{O} \in \mathbb{R}^{n\times q} \qquad \mathbf{W}^{(2)} \in \mathbb{R}^{h\times q} \qquad \mathbf{b}^{(2)} \in \mathbb{R}^{1\times q}$$



$$\begin{aligned}\mathbf{O} &= \left(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right)\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W} + \mathbf{b}\end{aligned}$$

- دو لایه خطی متوالی هیچ تفاوتی با یک لایه ندارد

مثال

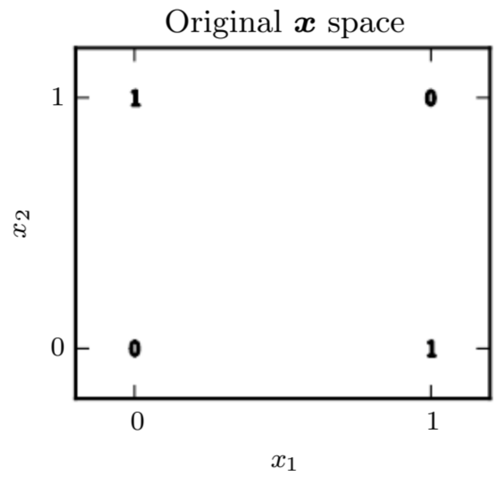
- تابع ضرر MSE:

$$L = \frac{1}{4}\sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

- مدل خطی:

$$\hat{y}^{(i)} = \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + b$$

- با استفاده از بهینه‌سازی تحلیلی به پاسخ زیر می‌رسیم
  - $\mathbf{w} = \mathbf{0}, b = 0.5$
  - $L = 0.25$ خواهد شد



مثال

- تابع ضرر MSE:

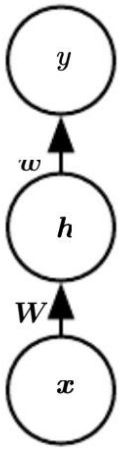
$$L = \frac{1}{4}\sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

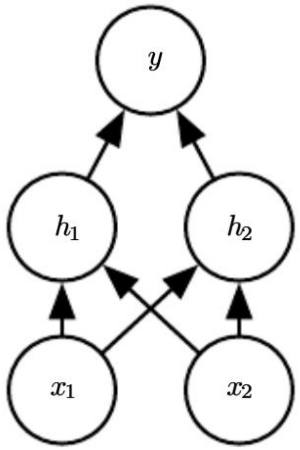
- مدل ۲ لایه:

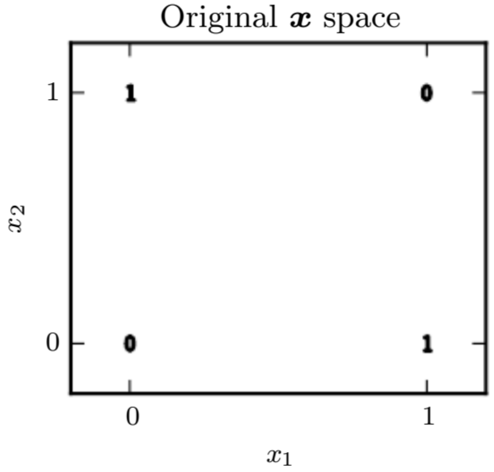
$$\hat{y}^{(i)} = f^{(2)}\left(f^{(1)}(\mathbf{x})\right)$$

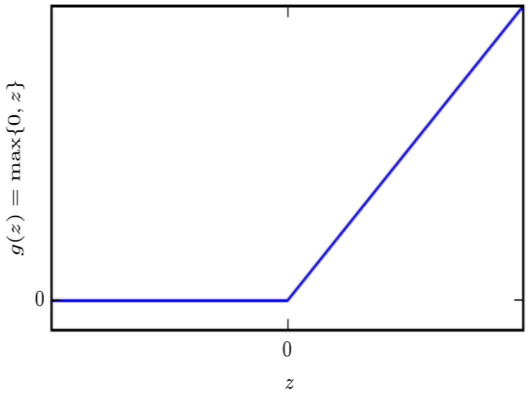
$$\hat{y}^{(i)} = \mathbf{w}^{\mathrm{T}} g\left(\mathbf{W}^{\mathrm{T}}\mathbf{x}^{(i)} + \mathbf{c}\right) + b$$

$$\hat{y}^{(i)} = \mathbf{w}^{\mathrm{T}} \max\left(0, \mathbf{W}^{\mathrm{T}}\mathbf{x}^{(i)} + \mathbf{c}\right) + b$$









مثال

- تابع ضرر MSE:

$$L = \frac{1}{4}\sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

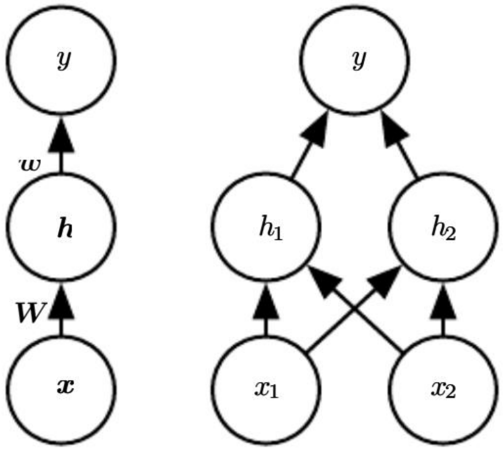
- مدل ۲ لایه:

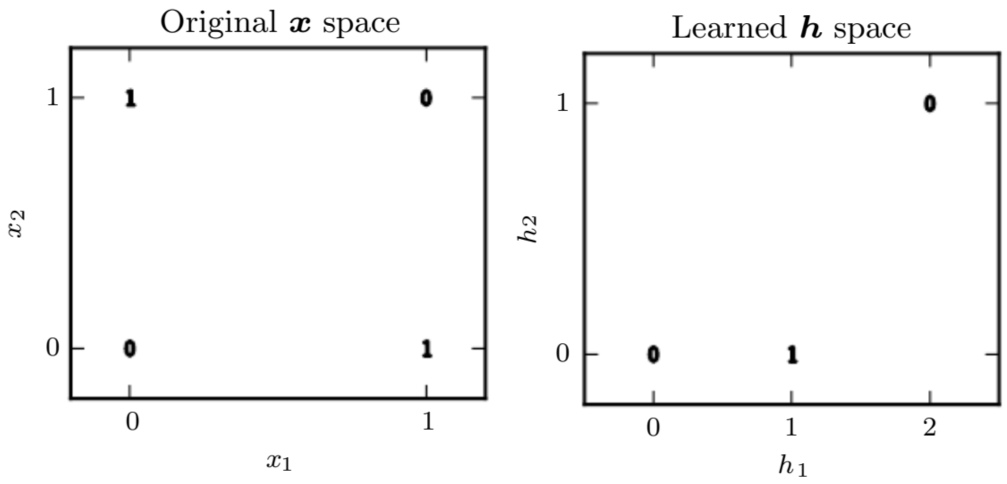
$$\hat{y}^{(i)} = \mathbf{w}^{\mathrm{T}} \max\left(0, \mathbf{W}^{\mathrm{T}}\mathbf{x}^{(i)} + \mathbf{c}\right) + b$$

- پاسخ زیر با $L = 0$ قابل محاسبه است

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

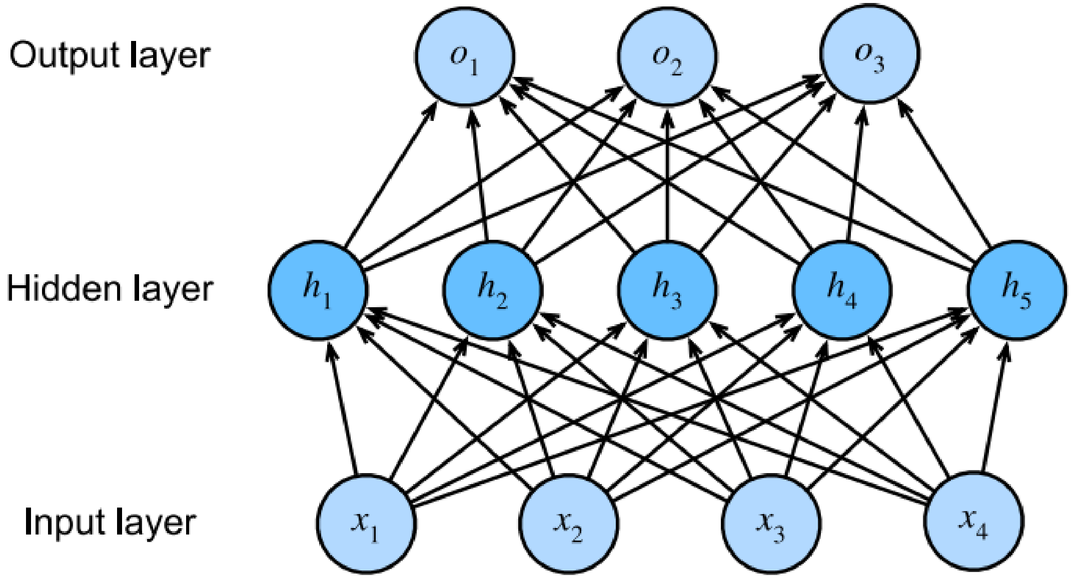
$$\boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \qquad b = 0$$





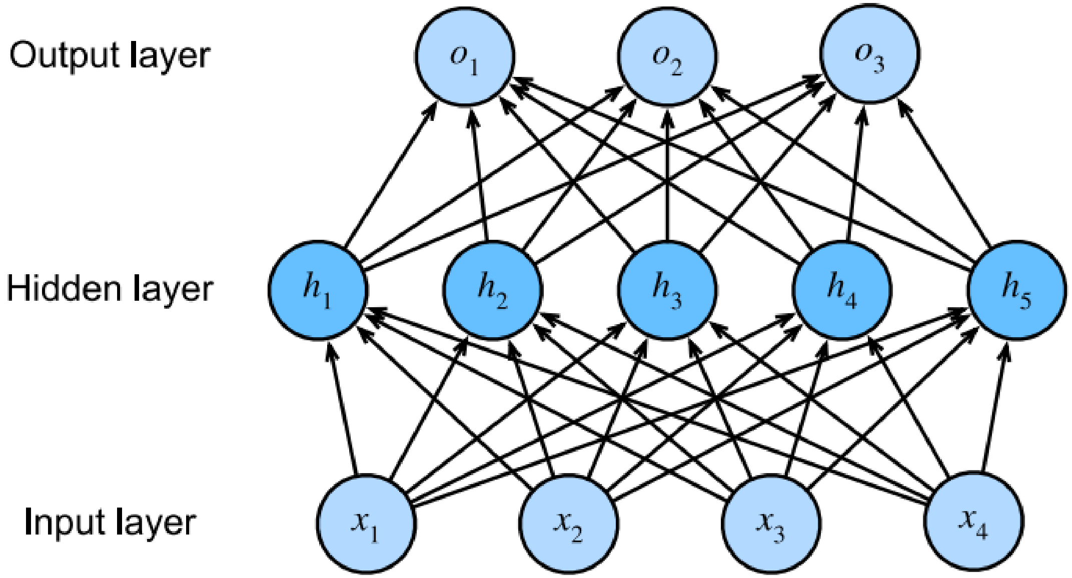
پرسپترون چندلایه

- یک شبکه MLP با تابع فعال‌سازی مناسب در لایه‌های میانی و تعداد نورون کافی می‌تواند هر تابعی را پیاده‌سازی کند
  - اصطلاحا Universal Approximator است
  - حتی فقط با یک لایه میانی با تعداد نورون به اندازه کافی زیاد
  - قسمت سخت کار آموزش وزن‌های مدل است
  - برای تقریب بسیاری از توابع می‌توانیم از شبکه‌های عمیق‌تر (بجای عریض‌تر) استفاده کنیم که بسیار فشرده‌تر و کاراتر باشد



توابع فعال‌سازی

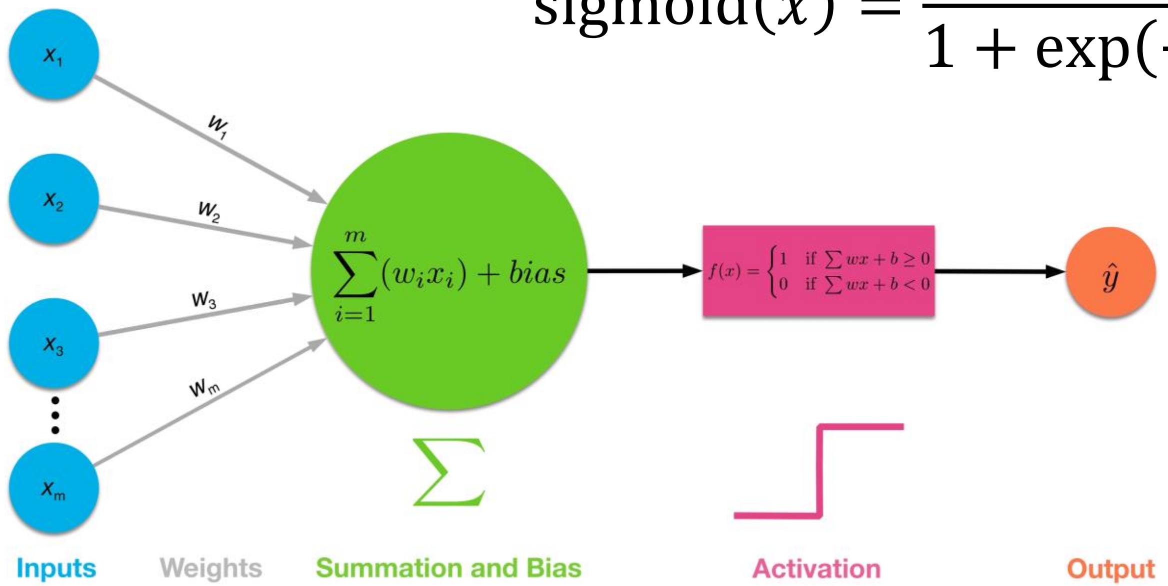
- توابع فعال‌سازی برای هر نورون مشخص می‌کنند که خروجی بخش خطی در چه شرایطی منجر به فعال شدن نورون شود
- به خصوص در شبکه‌های عمیق بسیار بااهمیت هستند

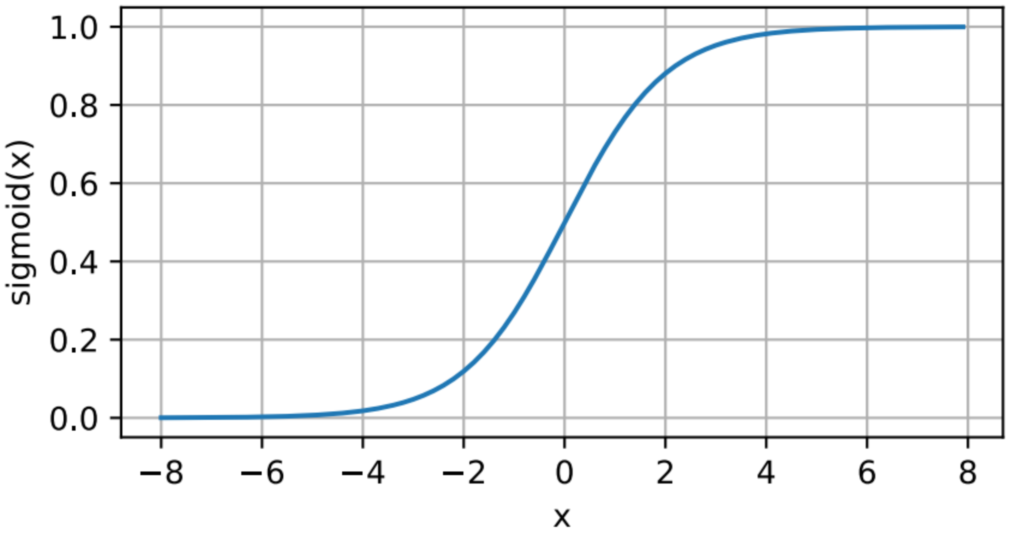


Sigmoid

- در اولین شبکه‌های عصبی، دانشمندان علاقه‌مند به مدل‌سازی نورون‌های بیولوژیکی بودند که یا فعال می‌شوند یا نمی‌شوند
- با توسعه روش‌های یادگیری مبتنی بر گرادیان، نیاز بود تا تقریب مشتق‌پذیر استفاده شود

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



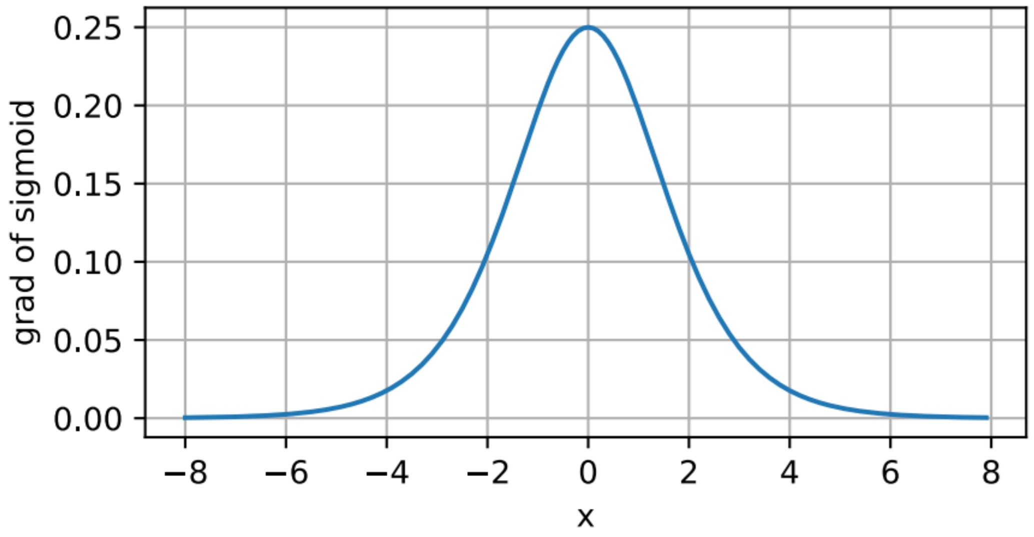


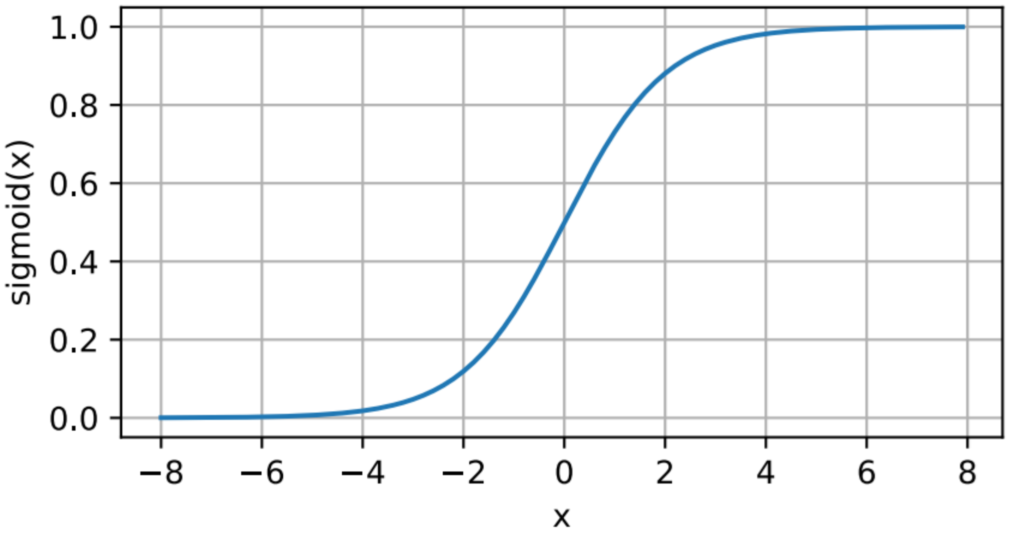
Sigmoid

- در بسیاری از مقادیر، گرادیان نزدیک به صفر است که بهینه‌سازی شبکه‌های عمیق را پیچیده می‌کند
- یکی از ایرادات sigmoid این است که خروجی آن همواره مثبت است

$$\frac{d}{dx}\text{sigmoid}(x) = \text{sigmoid}(x)(1 - \text{sigmoid}(x))$$

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





Sigmoid

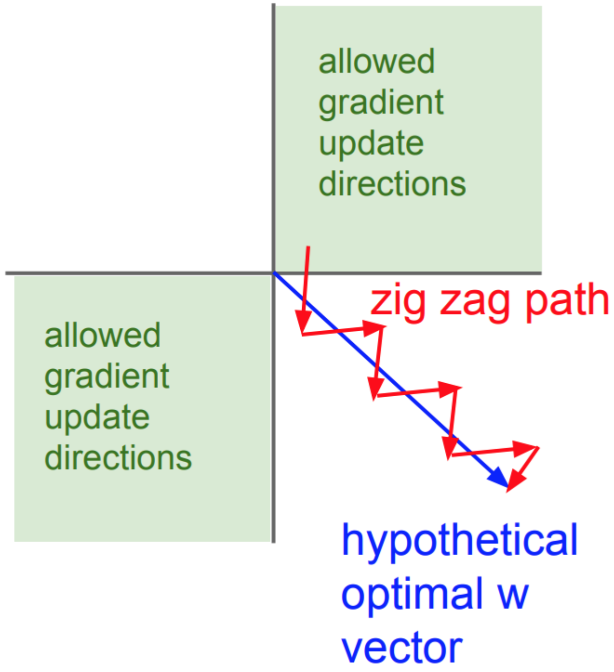
- در بسیاری از مقادیر، گرادیان نزدیک به صفر است که بهینه‌سازی شبکه‌های عمیق را پیچیده می‌کند
- یکی از ایرادات sigmoid این است که خروجی آن همواره مثبت است
  - فرض کنید تمام ورودی‌های یک لایه مثبت باشند

$$z = \sum_k w_k h_k + b$$

  - راجع به گرادیان نسبت به **w** چه می‌توانیم بگوئیم؟
    - تمام مقادیر مثبت یا تمام مقادیر منفی خواهند بود!

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_i} = \frac{\partial L}{\partial z} h_i$$

  - بهینه‌سازی وزن‌ها در برخی راستاها زیگ‌زاگی خواهد بود

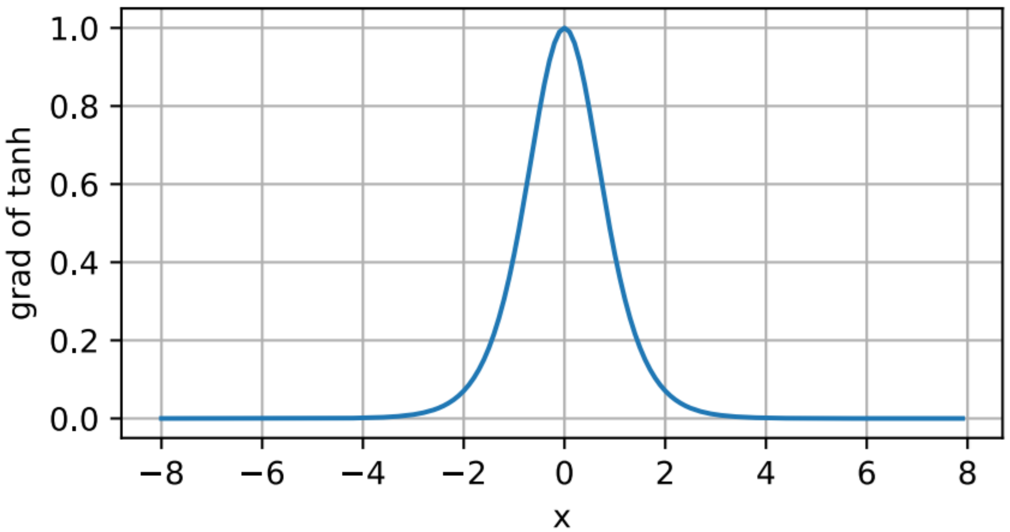


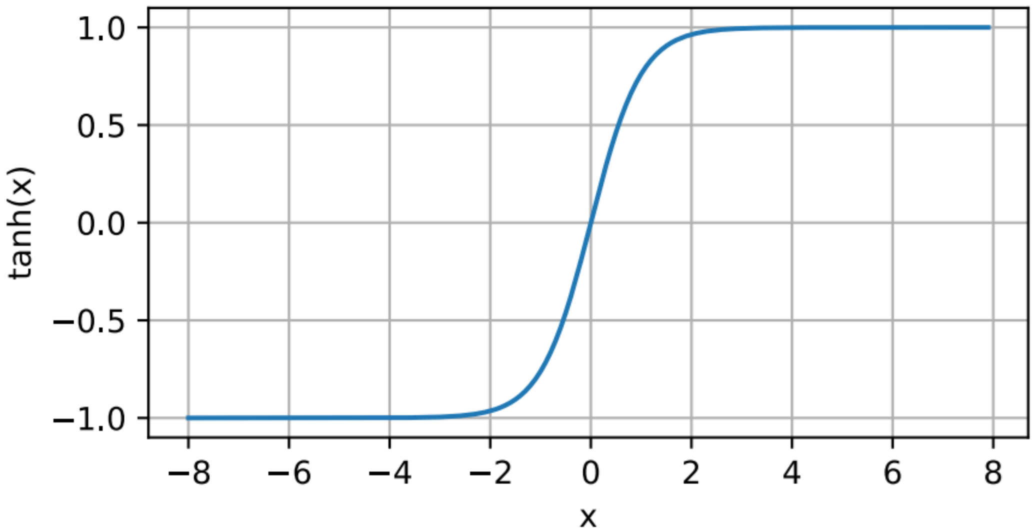
Tanh

- مشابه با تابع Sigmoid خروجی آن محدود است اما به بازه ۱- تا ۱+
  - اشباع شدن تابع tanh، بهینه‌سازی را دشوار می‌کند و از لحاظ محاسباتی هم به دلیل exp پرهزینه است

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

$$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x)$$

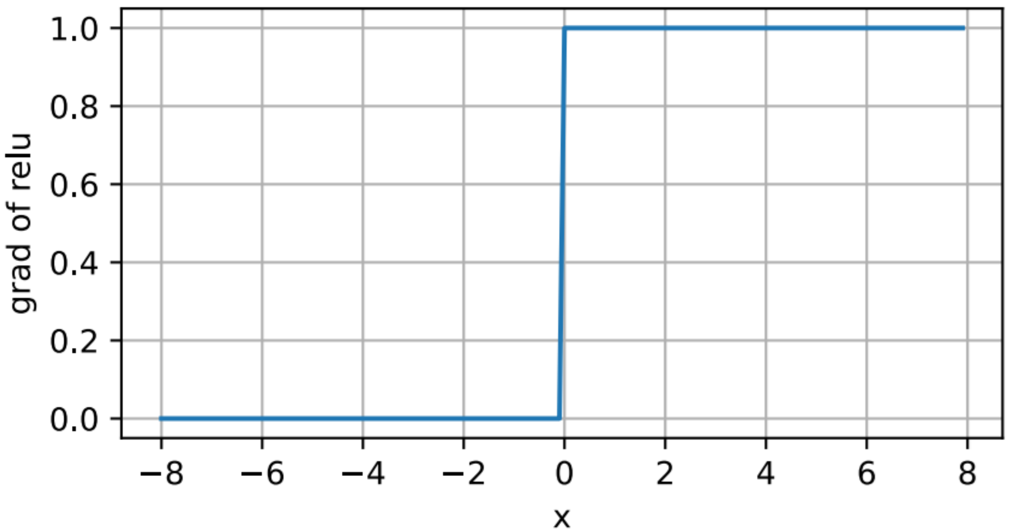




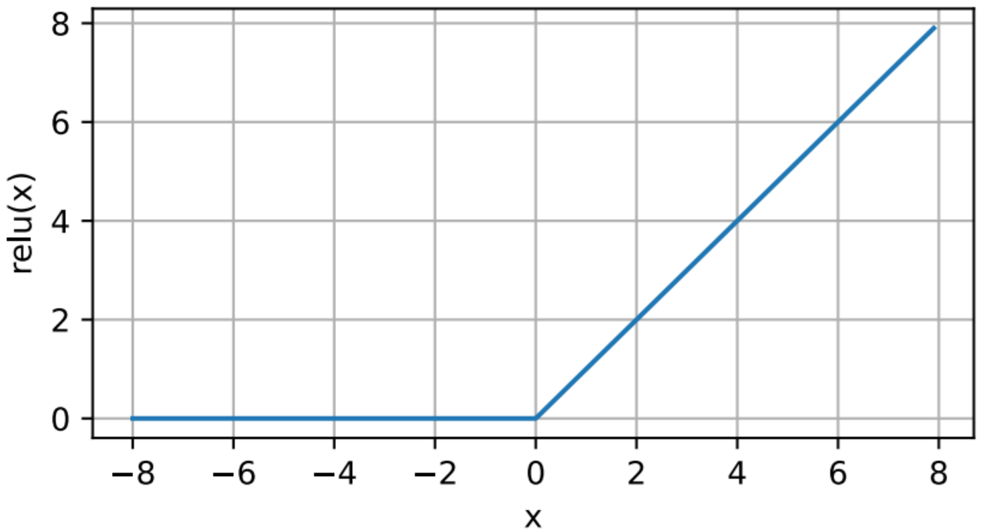
Rectifed Linear Unit (ReLU)

- یک تابع غیرخطی بسیار ساده و پرکاربرد است
- بهینه‌سازی ReLU بسیار آسان است زیرا بسیار شبیه به واحدهای خطی است
  - مشتق ReLU برای مقادیری که فعال است همواره بزرگ است

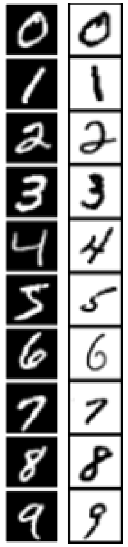
$$\frac{d}{dx}\mathrm{ReLU}(x) = x > 0$$



$$\mathrm{ReLU}(x) = \max(x, 0)$$



ReLU با شیب مخالف صفر

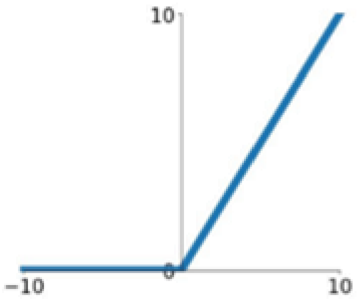


- تابع قدر مطلق ($g(z) = |z|$) با استفاده از $\alpha_i = -1$
- تابع Leaky ReLU از یک مقدار ثابت کوچک مانند $\alpha_i = 0.01$ استفاده می‌کند
- نسخه Parametric ReLU (PReLU) α_i را یک متغیر قابل آموزش در نظر می‌گیرد

$$h_i = g(\mathbf{z}, \boldsymbol{\alpha})_i = \max(0, z_i) + \alpha_i \min(0, z_i)$$

ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$