
بسم الله الرحمن الرحیم

یادگیری عمیق

مدرس: محمدرضا محمدی

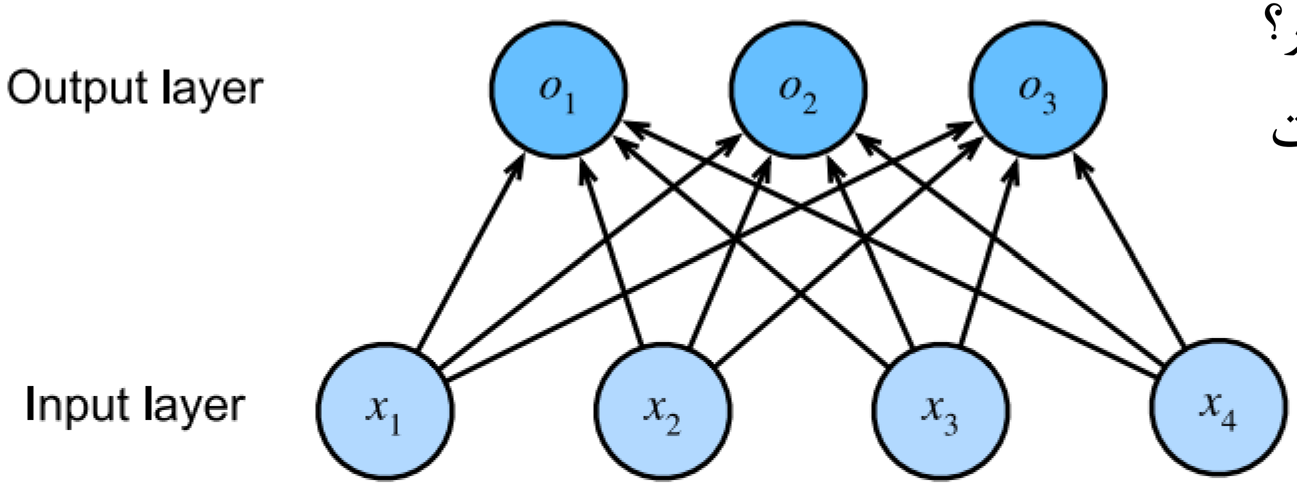
زمستان ۱۴۰۱

پرسپترون چندلایه

Multilayer Perceptron

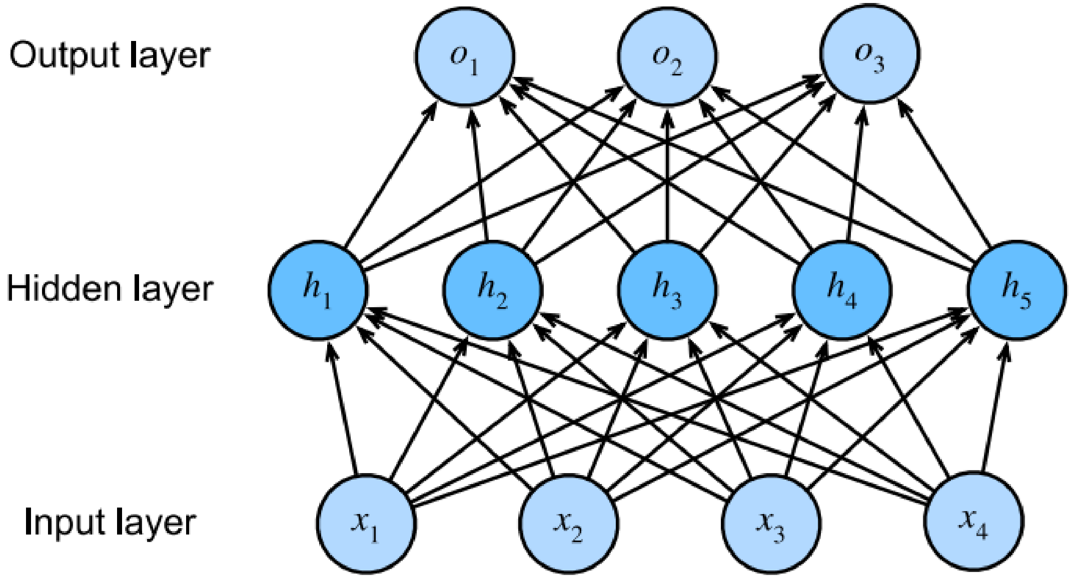
مدل‌های خطی

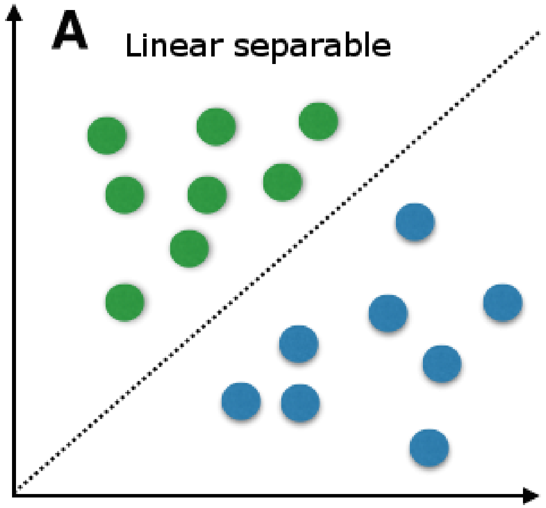
- بر فرض یکنواختی دلالت دارد
  - هر افزایشی در ویژگی همواره منجر به افزایش (کاهش) خروجی می‌شود اگر وزن مربوطه + (−) باشد
  - مثال: احتمال بازپرداخت اقساط بر حسب ویژگی‌هایی شامل درآمد
    - اثر افزایش درآمد از ۰ به ۵ میلیون متناظر با افزایش از ۱۰۰ به ۱۰۵ میلیون است
  - مثال: احتمال مرگ بر حسب ویژگی‌هایی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

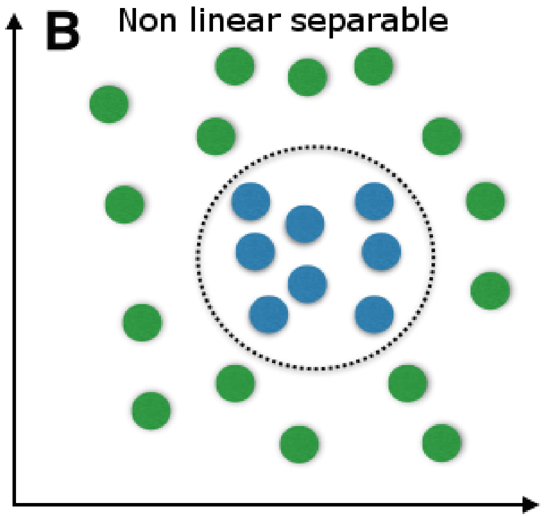


پرسپترون چندلایه

- 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





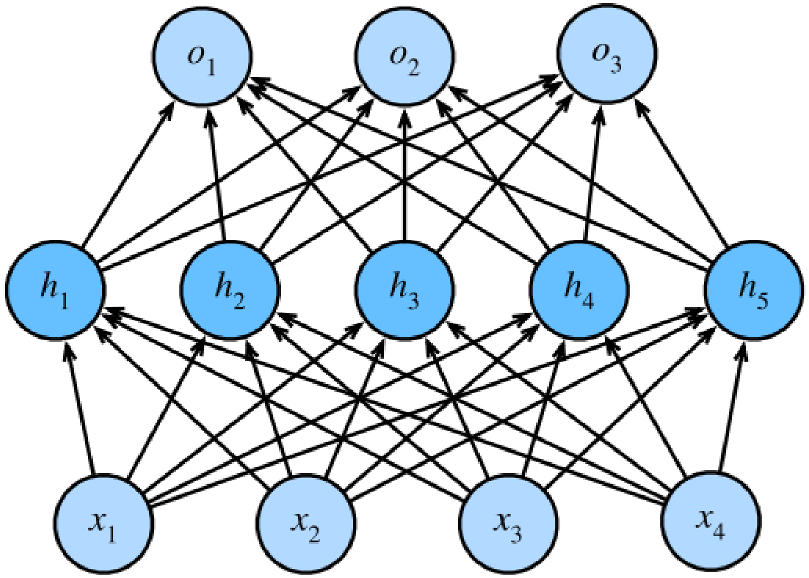


پرسپترون چندلایه

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

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

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



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

- دو لایه خطی متوالی هیچ تفاوتی با یک لایه ندارد

مثال

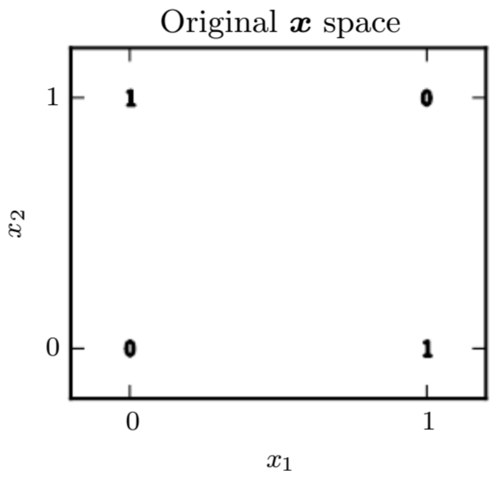
- تابع ضرر MSE:

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

- 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:

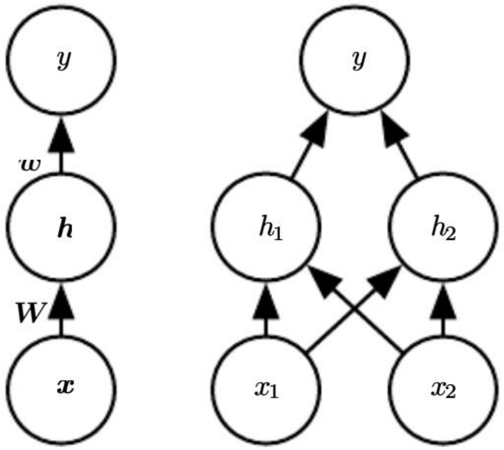
$$L = \frac{1}{4}\sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

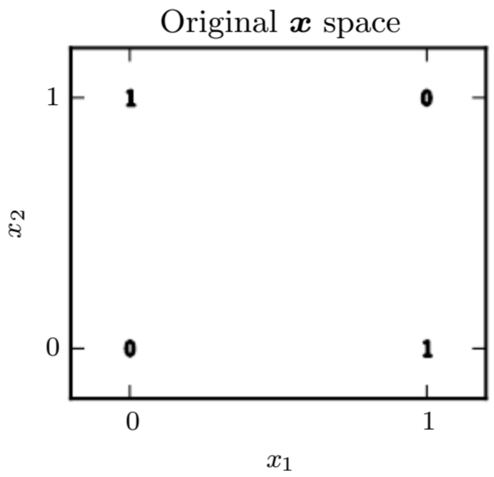
- مدل ۲ لایه:

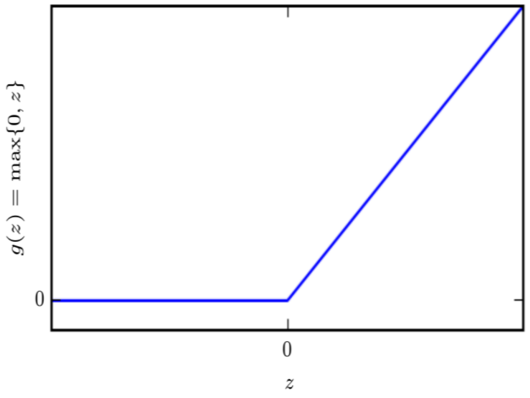
$$\hat{y}^{(i)} = f^{(2)}\left(f^{(1)}(\mathbf{x})\right)$$

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

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







مثال

- تابع ضرر MSE:

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

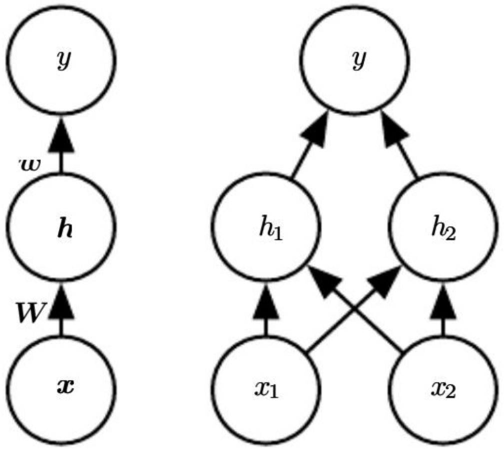
- مدل ۲ لایه:

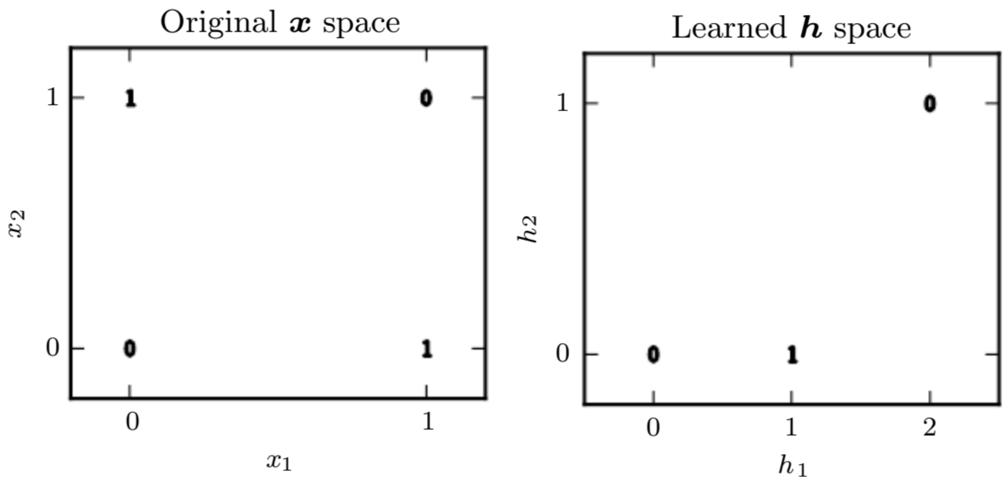
$$\hat{y}^{(i)} = \mathbf{w}^{\mathrm{T}} \max\left(0, \mathbf{W}^{\mathrm{T}}\mathbf{x}^{(i)} + \mathbf{c}\right) + b$$

- پاسخ زیر با $L = 0$ قابل محاسبه است

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

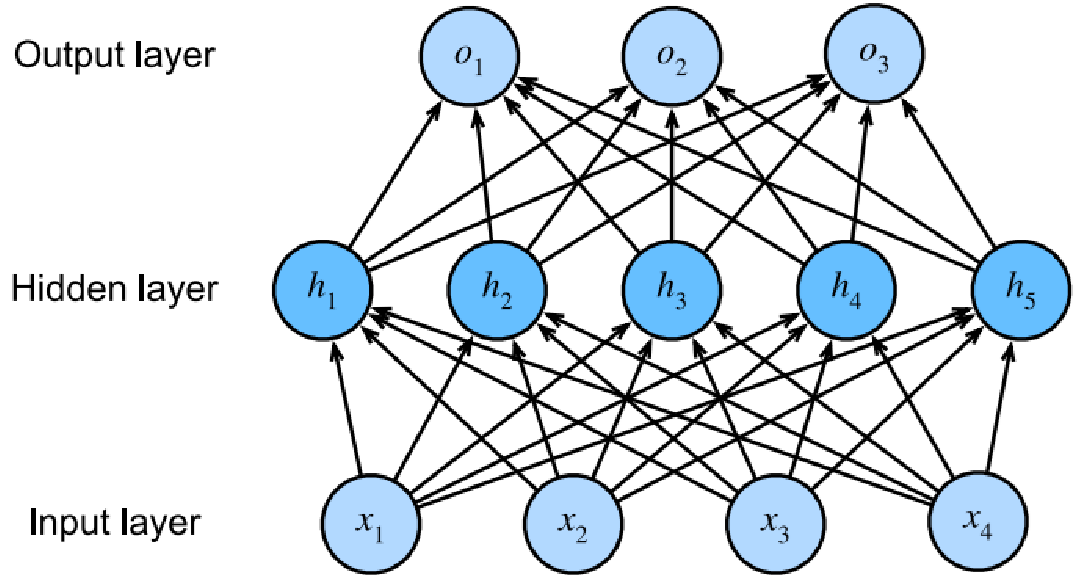
$$\boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \qquad b = 0$$





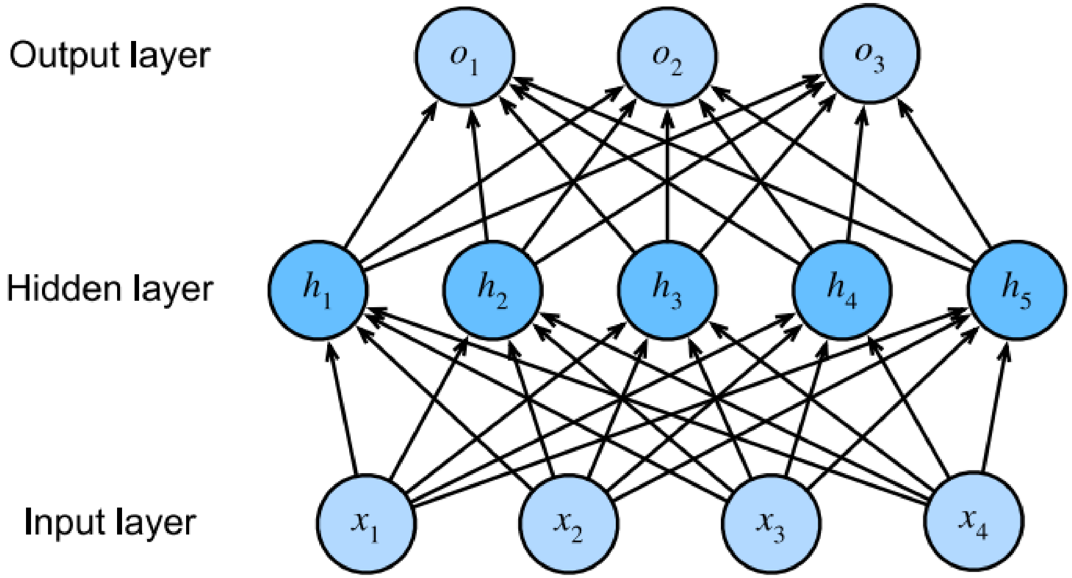
پرسپترون چندلایه

- 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

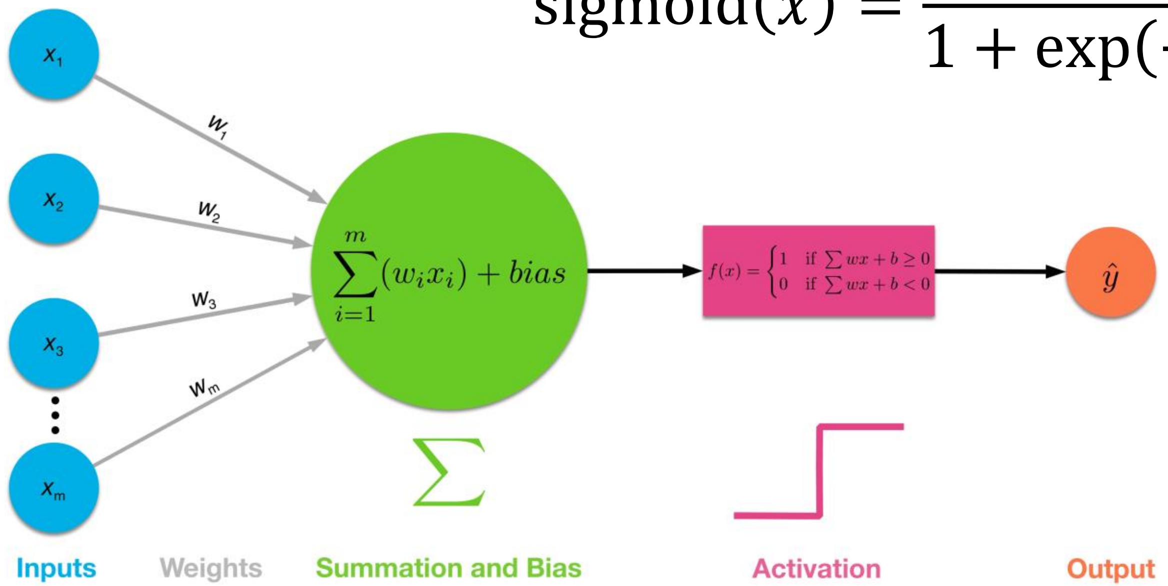
- توابع فعال‌سازی برای هر نورون مشخص می‌کنند که خروجی بخش خطی در چه شرایطی منجر به فعال شدن نورون شود
- به خصوص در شبکه‌های عمیق بسیار بااهمیت هستند

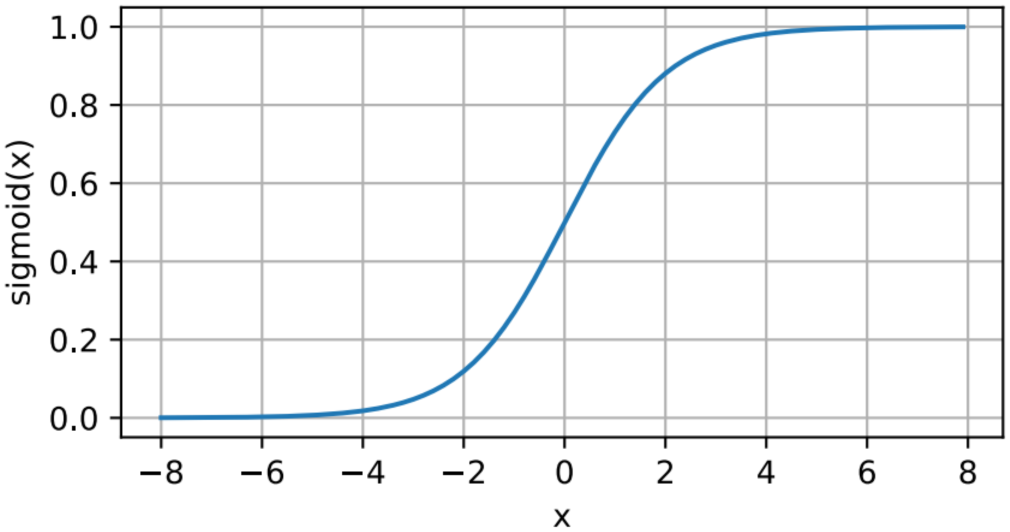


Sigmoid

- 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

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



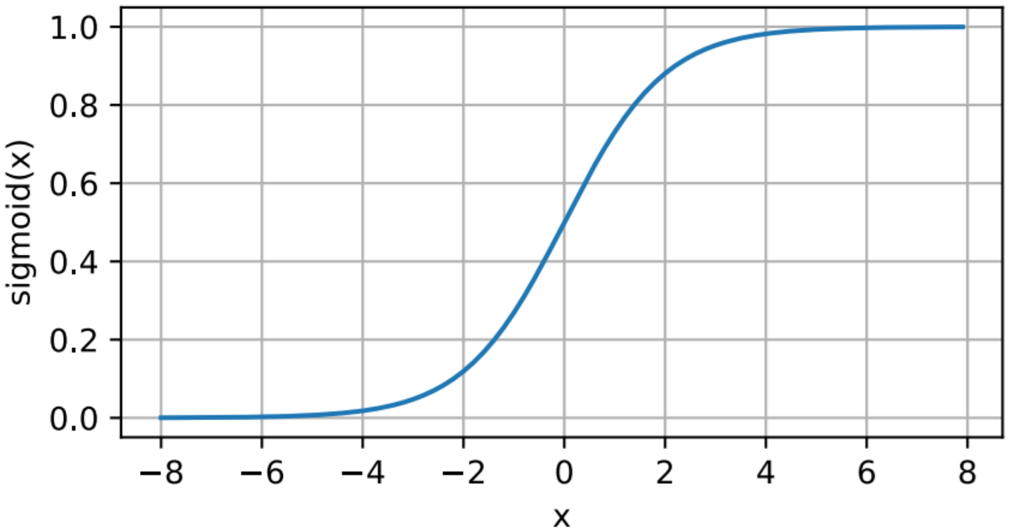


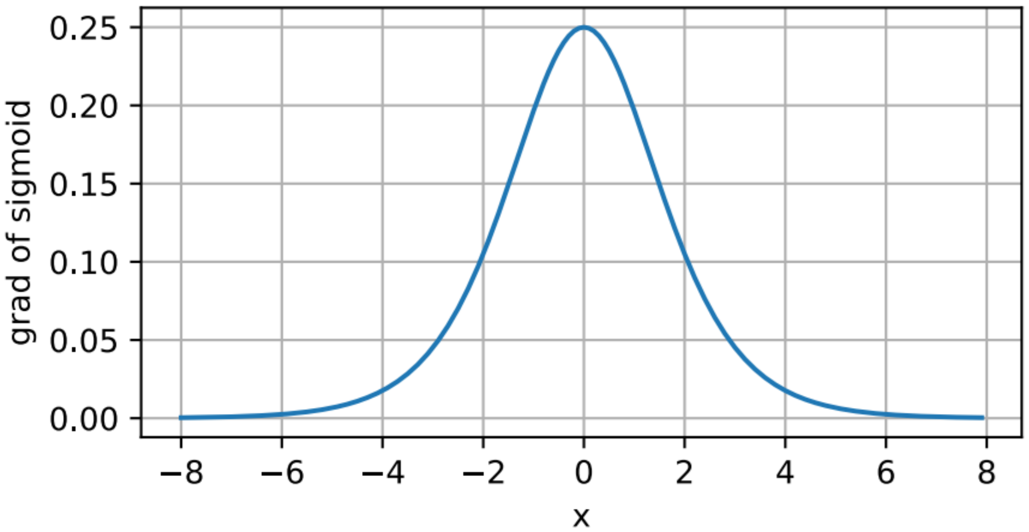
Sigmoid

- 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

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

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





Sigmoid

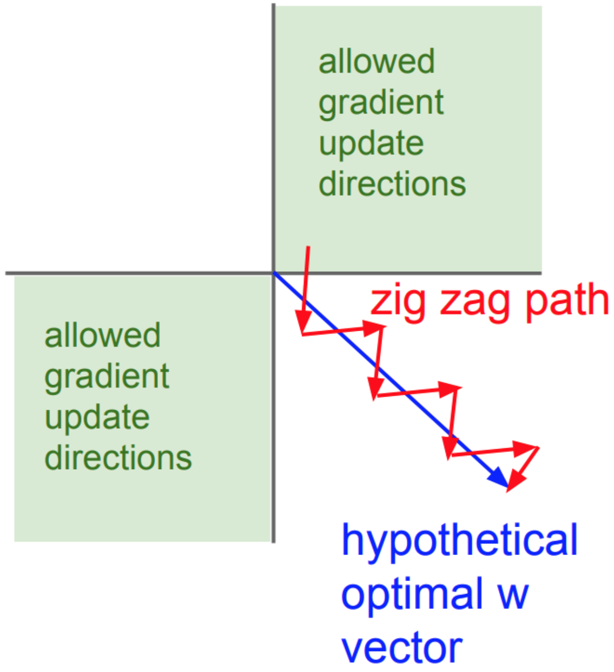
- در بسیاری از مقادیر، گرادیان نزدیک به صفر است که بهینه‌سازی شبکه‌های عمیق را پیچیده می‌کند
- یکی از ایرادات sigmoid این است که خروجی آن همواره مثبت است
  - فرض کنید تمام ورودی‌های یک لایه مثبت باشند

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

  - 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

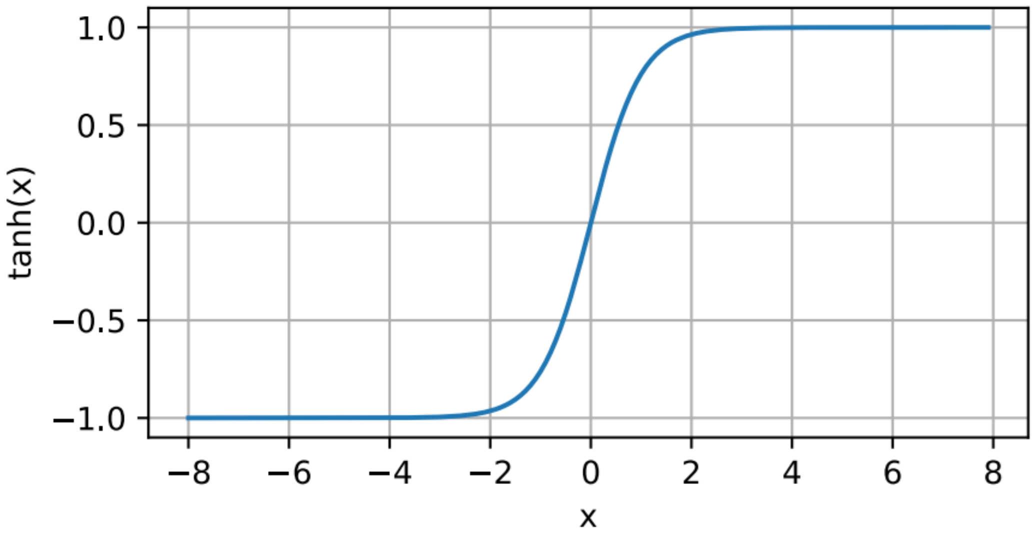


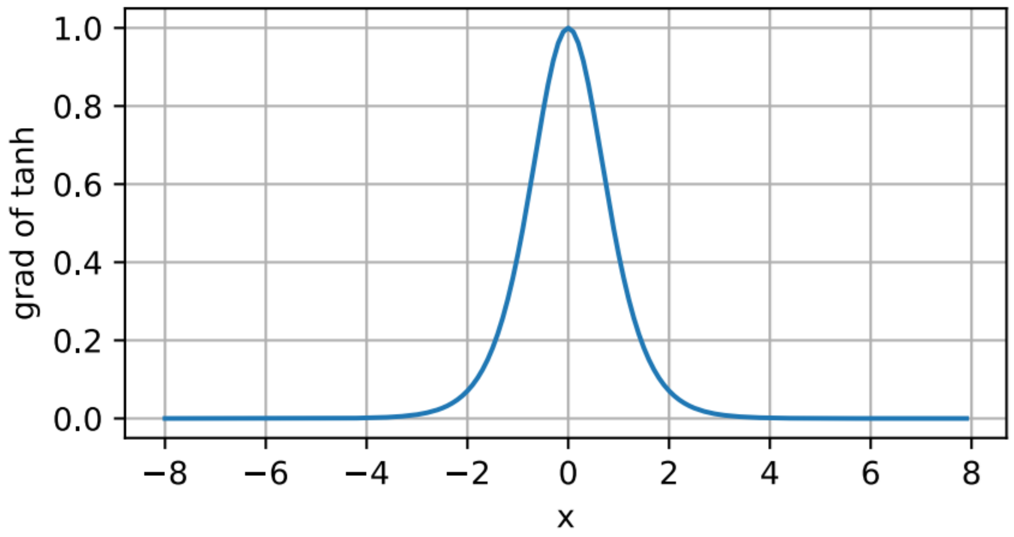
Tanh

- 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

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

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

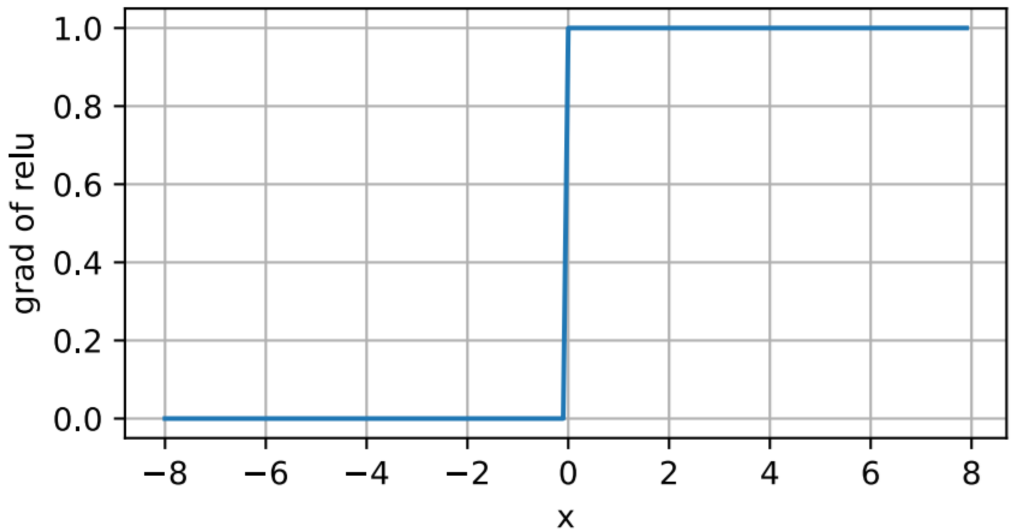




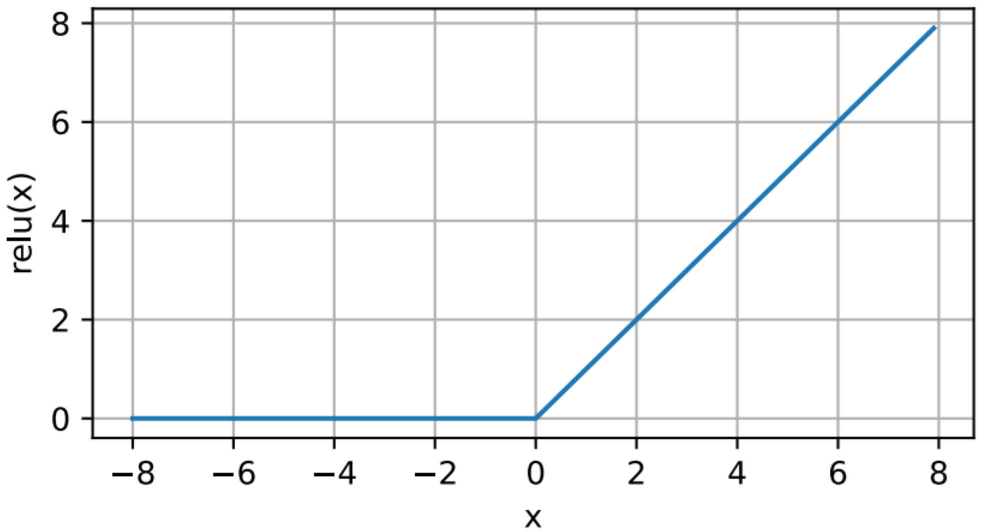
Rectifed Linear Unit (ReLU)

- 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

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

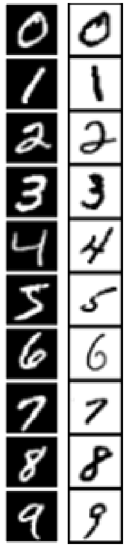


$$\mathrm{ReLU}(x) = \max(x, 0)$$

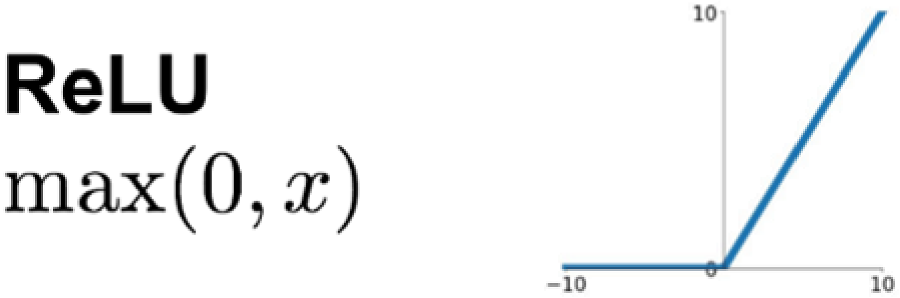


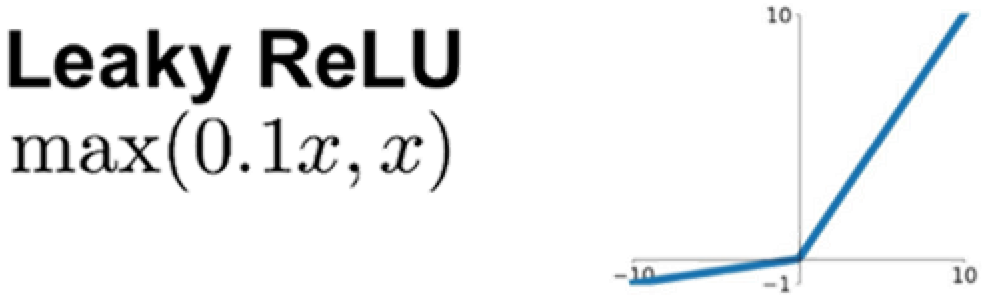
ReLU با شیب مخالف صفر

- 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



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





انتخاب مدل

- در یادگیری ماشین، هدف ما کشف الگوها است
- اما چگونه می‌توانیم مطمئن باشیم که واقعاً یک الگوی عمومی را کشف کرده‌ایم و صرفاً داده‌های خود را حفظ نکرده‌ایم؟

انتخاب مدل

- چگونه الگوهایی را کشف کنیم که تعمیم‌پذیر باشند؟
  - چالش بزرگ این است که در زمان آموزش مدل فقط به مجموعه کوچکی از داده‌ها دسترسی داریم