بسم الله الرحمن الرحیم

یادگیری ماشین

مدرس: محمدرضا محمدی

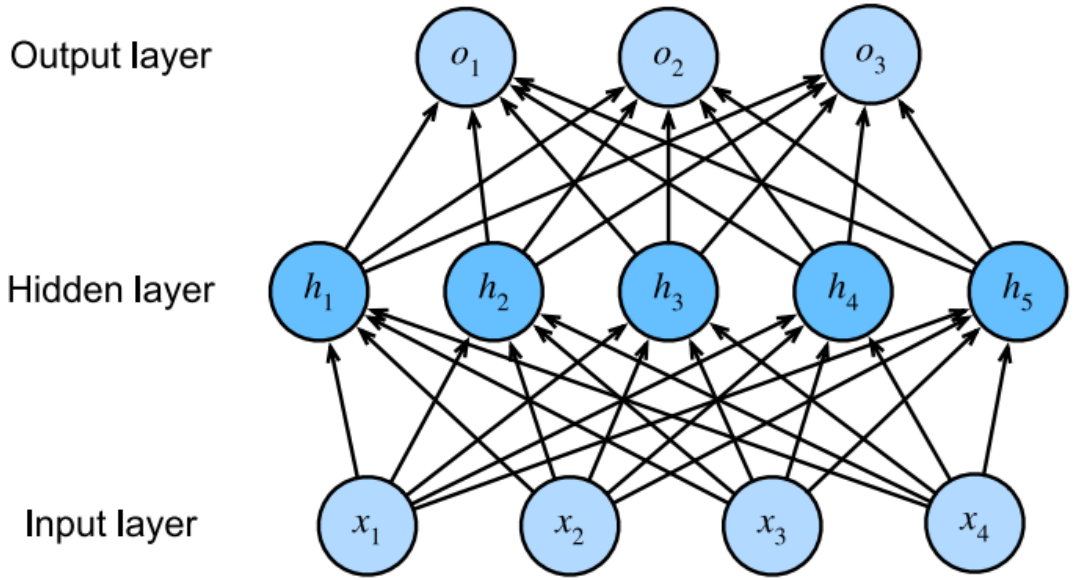
بهار ۱۴۰۲

شبکه‌های عصبی مصنوعی

Artificial Neural Networks

توابع فعال‌سازی

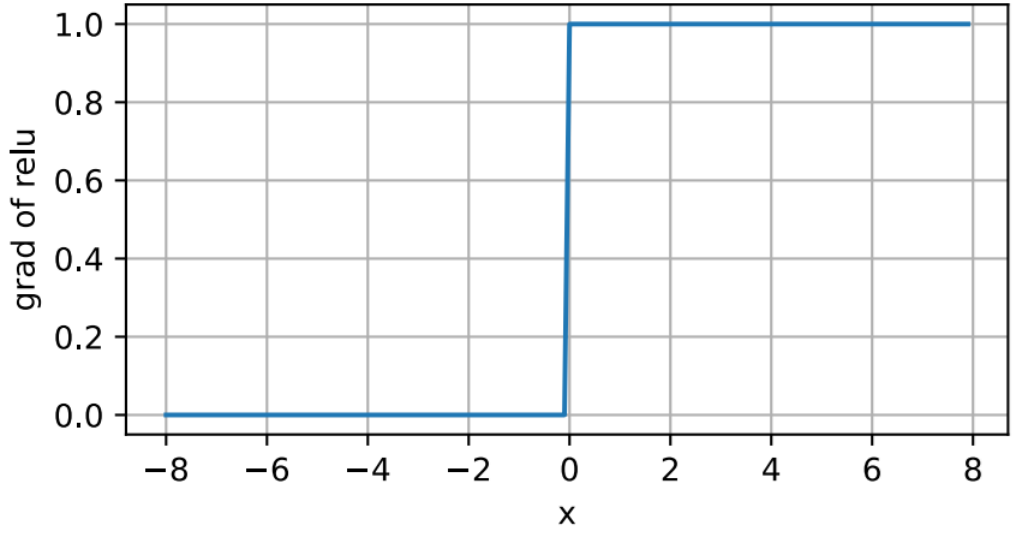
- توابع فعال‌سازی برای هر نورون مشخص می‌کنند که خروجی بخش خطی در چه شرایطی منجر به فعال شدن نورون شود
- به خصوص در شبکه‌های عمیق بسیار بااهمیت هستند



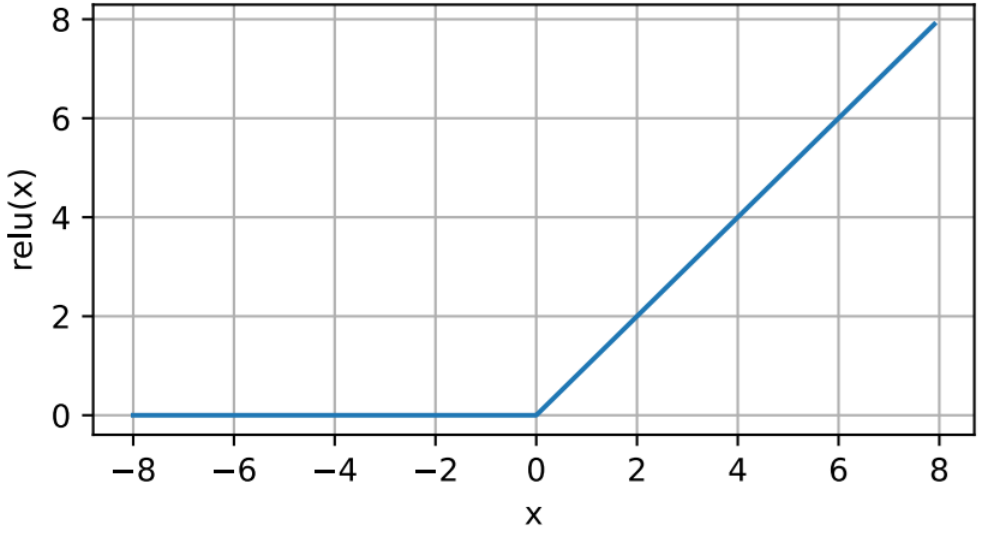
Rectifed Linear Unit (ReLU)

- 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

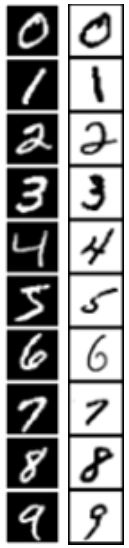
$$\frac{d}{dx}\mathrm{ReLU}(x) = x > 0$$



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



ReLU با شیب مخالف صفر

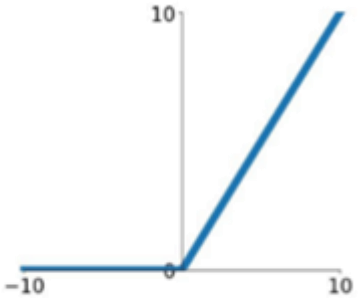


- تابع قدر مطلق ($g(z) = |z|$) با استفاده از $\alpha_i = -1$
- تابع Leaky ReLU از یک مقدار ثابت کوچک مانند $\alpha_i = 0.01$ استفاده می‌کند
- نسخه Parametric ReLU (PReLU) α_i را یک متغیر قابل آموزش در نظر می‌گیرد

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

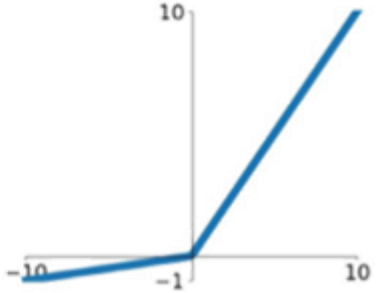
ReLU

$\max(0, x)$

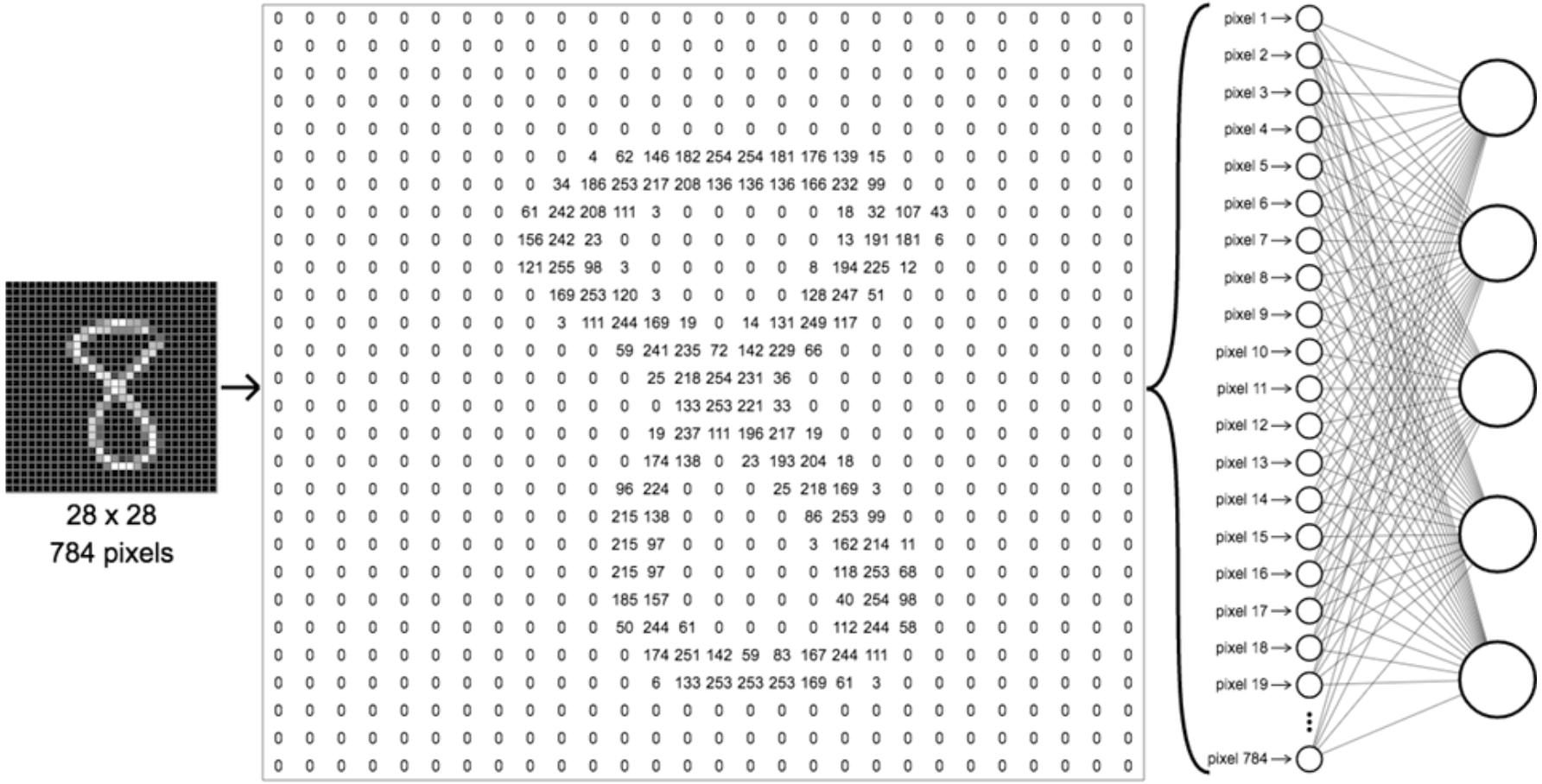


Leaky ReLU

$\max(0.1x, x)$

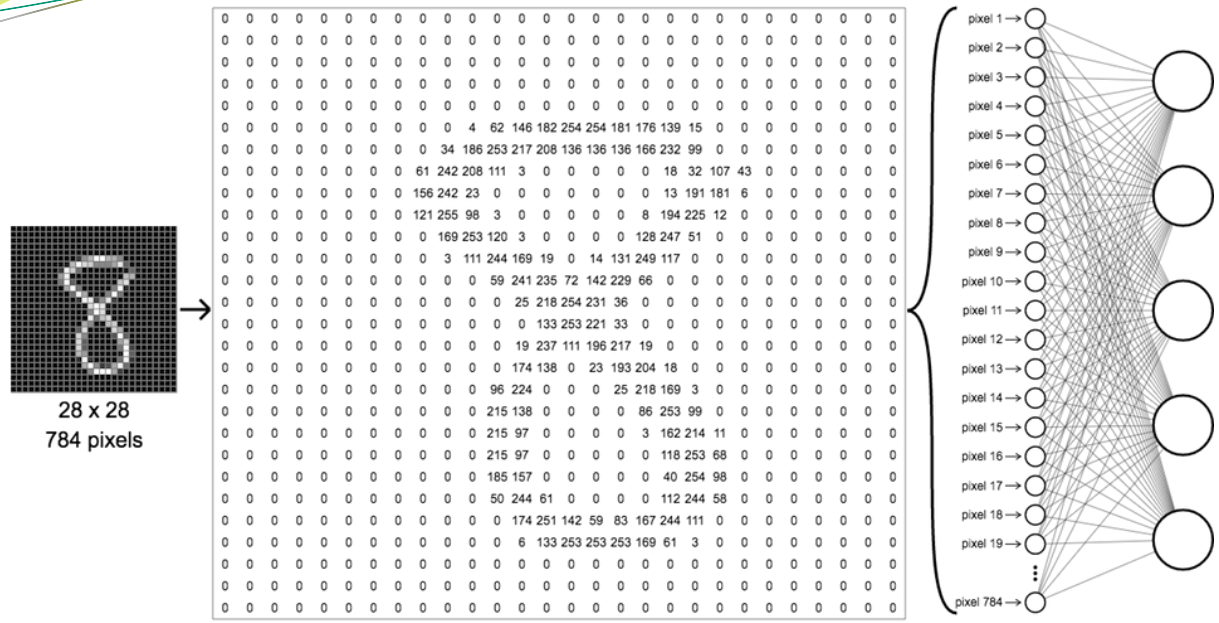


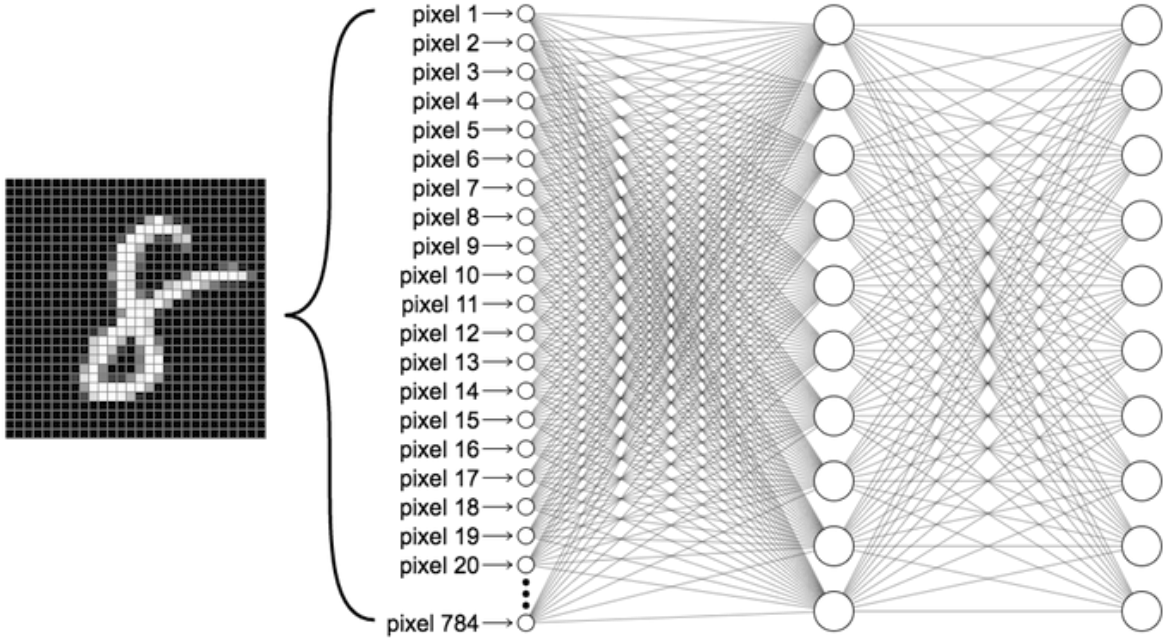
لایه کاملا متصل



https://becominghuman.ai/not-just-introduction-to-convolutional-neural-networks-part-1-56a36b938592

شبکه‌های عصبی





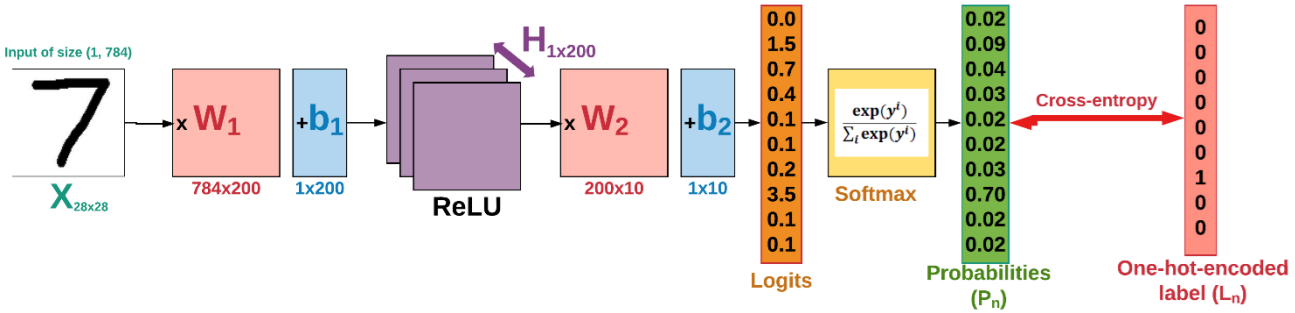
https://ml4a.github.io/ml4a/looking_inside_neural_nets/

آموزش شبکه‌های عصبی عمیق

Training Deep Neural Networks

چرخه آموزش

- این تنظیم تدریجی، که آموزش نیز نامیده می شود، پایه یادگیری در بسیاری از الگوریتم‌های یادگیری ماشین است
  - یک batch از نمونه‌های آموزشی x و خروجی‌های مربوطه y انتخاب می‌شود
  - شبکه بر روی x اعمال می‌شود (forward pass) تا $\hat{y}$ بدست بیاید
  - تابع ضرر شبکه برای این batch از مقایسه y و $\hat{y}$ محاسبه می‌شود
  - تمام وزن‌های شبکه به گونه‌ای به روز می‌شوند که مقدار ضرر برای این batch کمی کاهش بیابد



شبیه‌سازی

- کتابخانه TensorFlow یکی از ابزارهای قدرتمند در حوزه یادگیری عمیق است که امکانات زیادی را در اختیار کاربران قرار می‌دهد و مورد استقبال بسیاری از پژوهشگران است
- در این دوره ما از بخش Keras در TensorFlow برای شبیه‌سازی استفاده می‌کنیم
- برای اجرای کدها از سرویس رایگان Google Colab استفاده می‌کنیم

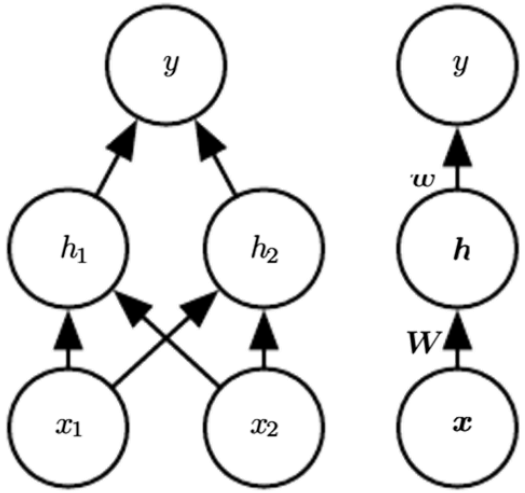
Example: https://colab.research.google.com/drive/1owxMjSGbcTtKydWKXCRCk_CPxZbMmlxG?usp=sharing

محاسبه گرادیان

$$h_i = g\left(\boldsymbol{x}^T \boldsymbol{W}_{:,i} + c_i\right) = f^{(1)}(\boldsymbol{x})$$

$$y = f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right) = f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^T \max\{0, \boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{c}\} + b$$

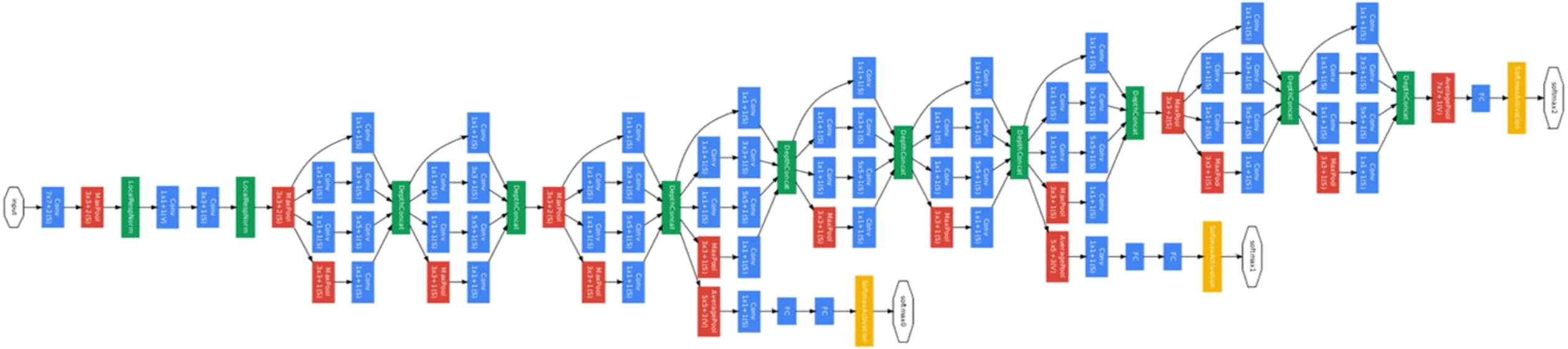
$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum \left(f^*(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b)\right)^2$$



- اگر بتوانیم $\frac{\partial J}{\partial \boldsymbol{W}}$ ، $\frac{\partial J}{\partial \boldsymbol{c}}$ ، $\frac{\partial J}{\partial \boldsymbol{w}}$ و $\frac{\partial J}{\partial b}$ را محاسبه کنیم، می‌توانیم پارامترهای مدل را آموزش بدهیم
- اگر بخواهیم تابع ضرر را تغییر بدهیم؟
  - نیاز است تا معادلات دوباره از ابتدا استخراج شوند

محاسبه گرادیان

- بسیار خسته‌کننده است و نیاز به محاسبات ماتریسی فوق‌العاده زیادی دارد



پس‌انتشار (Backpropagation)

$$f(x, y, z) = (x + y)z$$

e.g., $x = -2,\ y = 5,\ z = -4$

$$q = x + y \qquad \Rightarrow f = qz$$

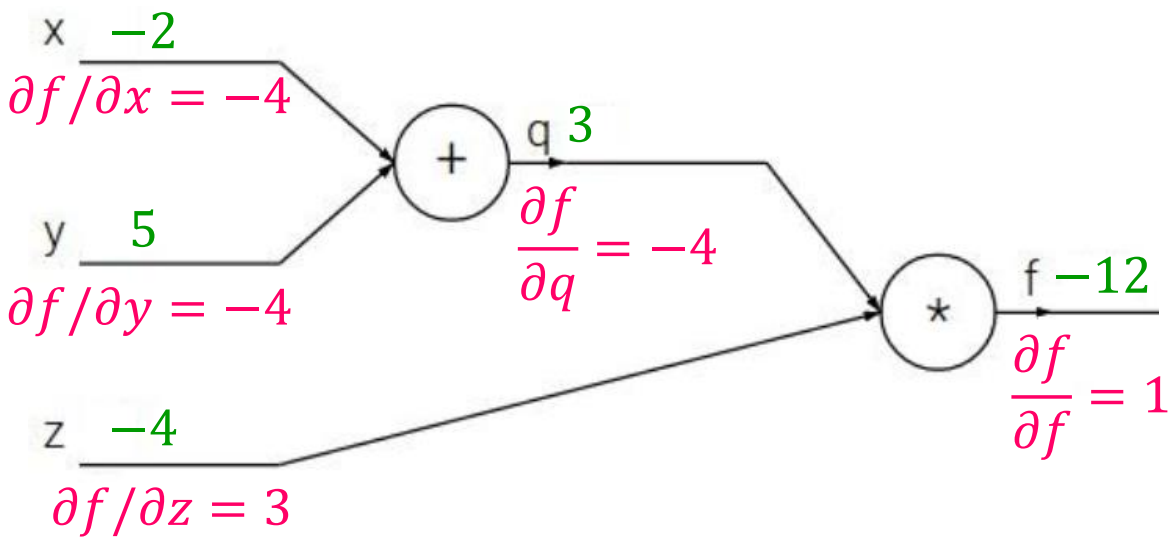
we want $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ and $\frac{\partial f}{\partial z}$

$$\frac{\partial f}{\partial q} = z \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

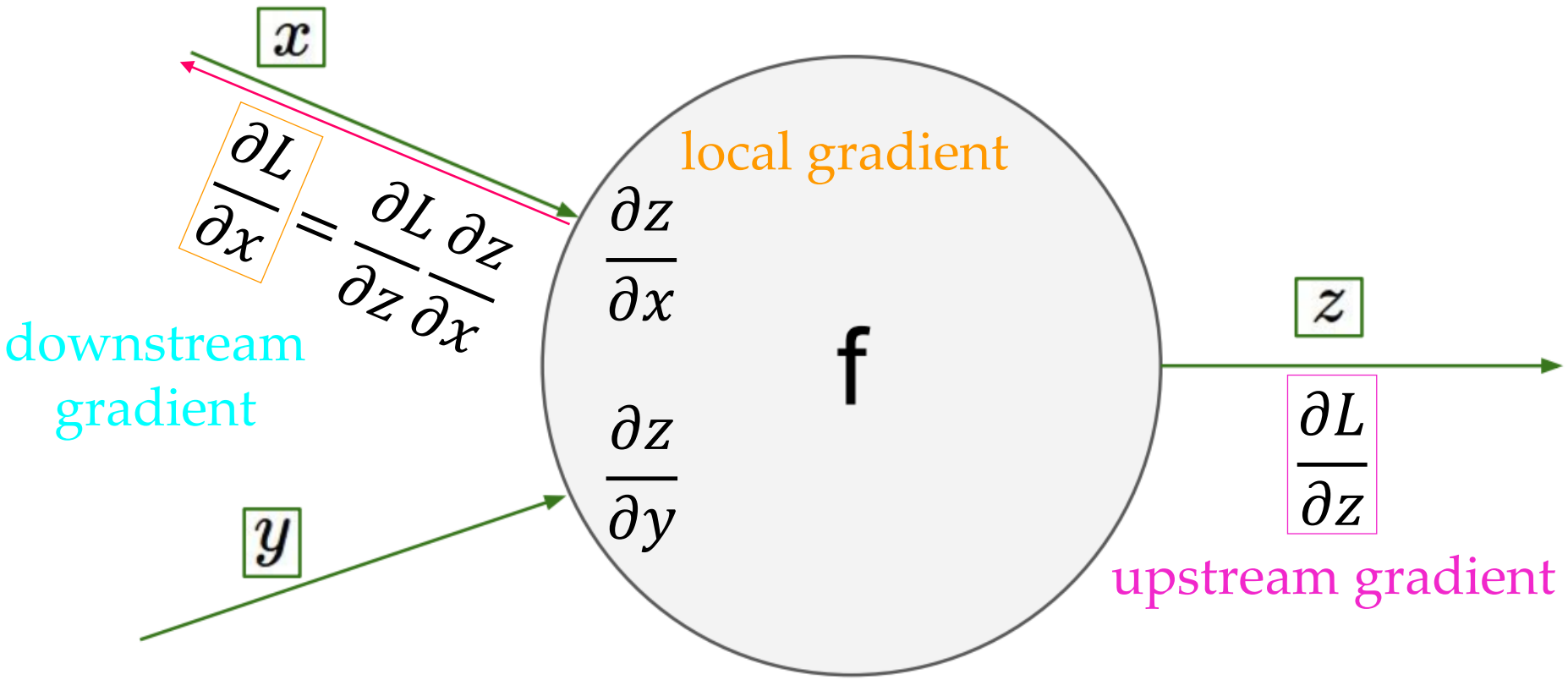
Chain rule

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} \qquad \frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$



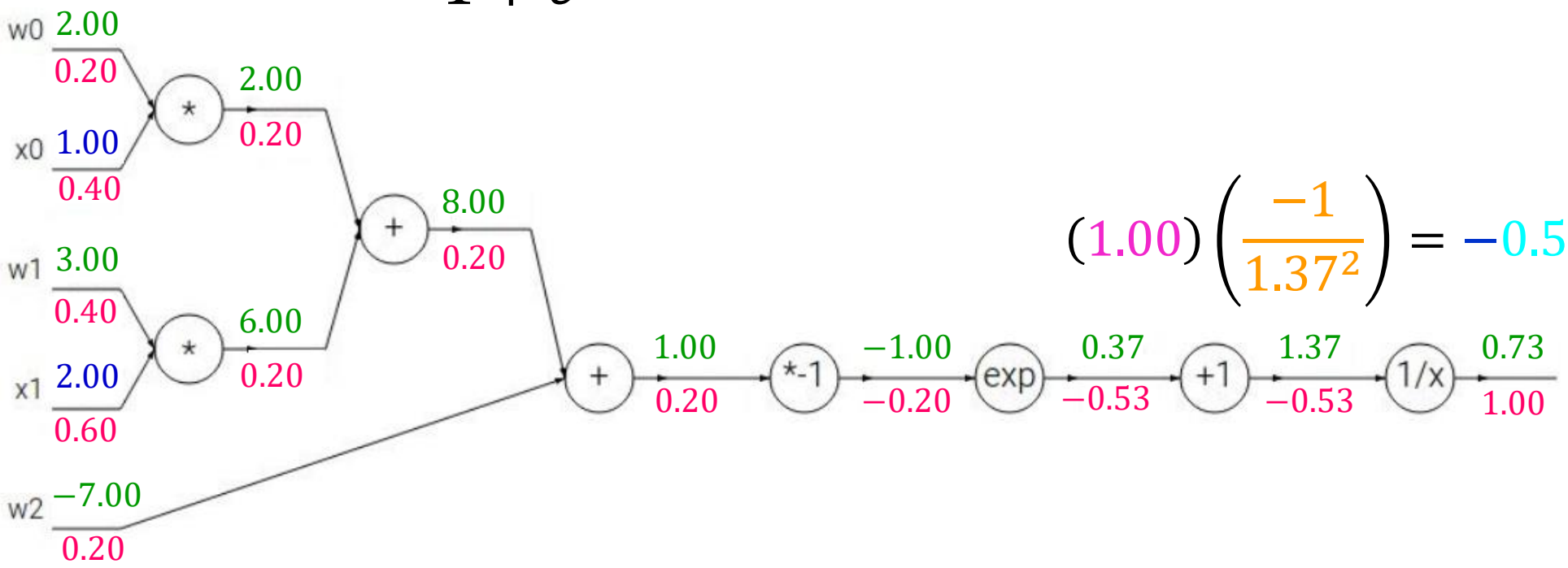
Based on cs231n by Fei-Fei Li & Justin Johnson & Serena Yeung https://cs231n.github.io/optimization-2/

پس‌انتشار (Backpropagation)



مثال: Linear + Sigmoid

$$f(\boldsymbol{w},\boldsymbol{x}) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



$$(1.00)\left(\frac{-1}{1.37^2}\right) = -0.53$$

$$\frac{\partial(ax)}{\partial x} = a \qquad \frac{\partial(c+x)}{\partial x} = 1 \qquad \frac{\partial(e^x)}{\partial x} = e^x \qquad \frac{\partial(1/x)}{\partial x} = \frac{-1}{x^2}$$