

神经网络基础

黄政宇

北京大学北京国际数学研究中心

北京大学国际机器学习研究中心



本堂课大纲

- 全连接神经网络
 - 激活函数
 - 万能逼近定理
- 神经网络中的模块
- 神经网络的训练
 - 初始化
 - 反向传播
 - 凸优化



利用非线性模型的监督学习

➤ 监督学习

训练数据： $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

线性模型： $f(x) = x^T w$
 $f(x) = \Psi(x)^T w$

非线性模型： $f(x; \theta)$

优化均方误差函数：

$$J(\theta) = \frac{1}{n} \sum_{j=1}^n J^{(i)}(\theta) \quad J^{(i)}(\theta) = \frac{1}{2} (f(x_i; \theta) - y_i)^2$$



神经网络

➤ 全连接神经网络

$$y = f(x; \theta): R^{n_0} \rightarrow R^{n_L}$$

$$f(x; \theta) = f_L \circ f_{L-1} \circ \dots \circ f_1(x)$$

$$f_i(z) = h^{(i)}(W^{(i)}z + b^{(i)}): R^{n_{i-1}} \rightarrow R^{n_i}$$

$$\text{矩阵: } W^{(i)} \in R^{n_i \times n_{i-1}}$$

$$\text{向量: } b^{(i)} \in R^{n_i}$$

对每个元素作用的(非线性)激活函数: $h^{(i)}$



激活函数

➤ 激活函数

对每个元素作用的(非线性) 激活函数： h

$$h(z) = [\sigma(z_1), \sigma(z_2), \dots \sigma(z_n)]^T$$

$$\text{RELU: } \sigma(z) = \max\{0, z\}$$

$$\text{Sigmoid: } \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\text{Tanh: } \sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\text{Leaky RELU: } \sigma(z) = \max\{z, \gamma z\} (\gamma \in (0, 1))$$

$$\text{GELU: } \sigma(z) = \frac{z}{2} \left[1 + \operatorname{erf}\left(\frac{z}{\sqrt{2}}\right) \right]$$

$$\text{SoftPlus: } \sigma(z) = \frac{1}{\beta} \log(1 + e^{\beta z}), \beta > 0$$

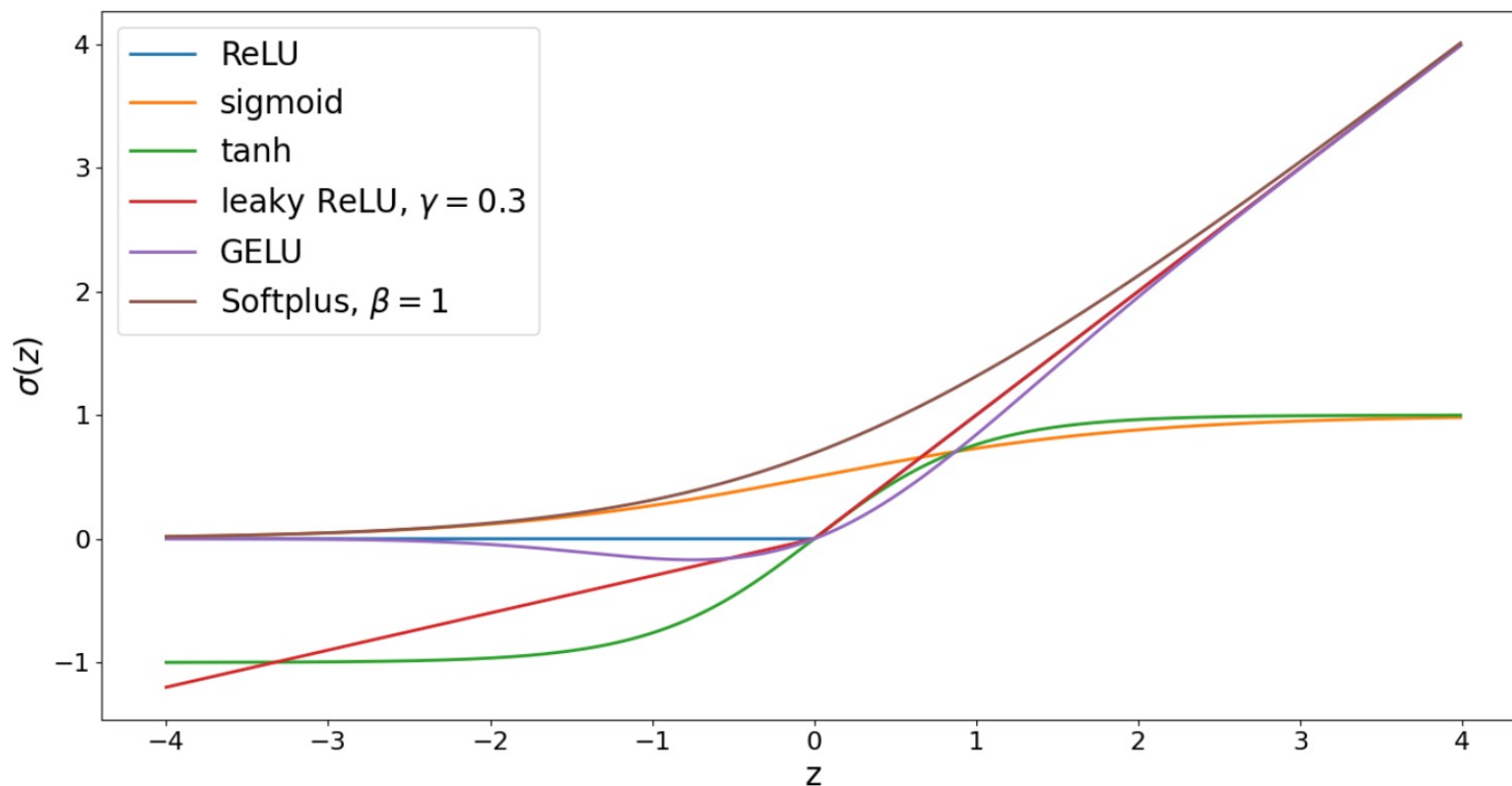


激活函数

➤ 激活函数

对每个元素作用的(非线性) 激活函数： h

$$h(z) = [\sigma(z_1), \sigma(z_2), \dots, \sigma(z_n)]^T$$

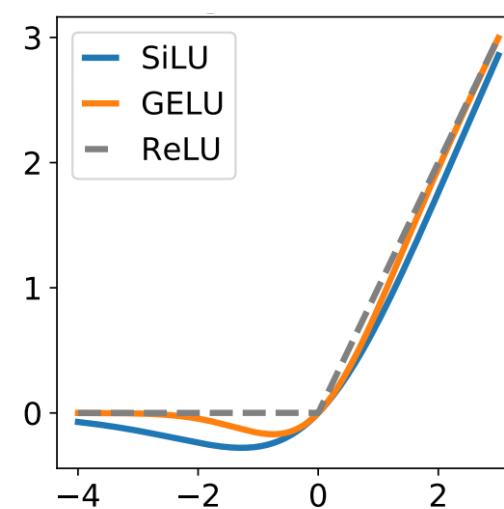
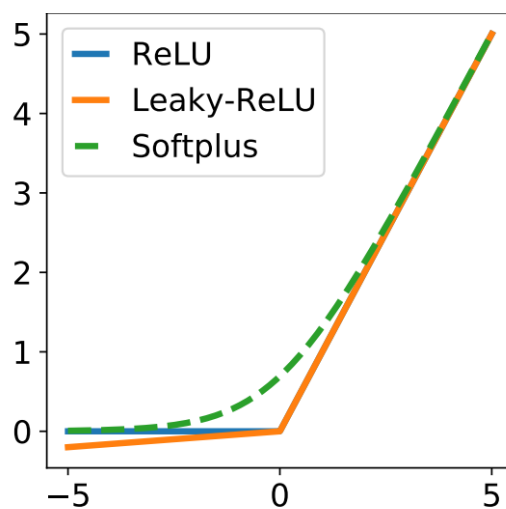
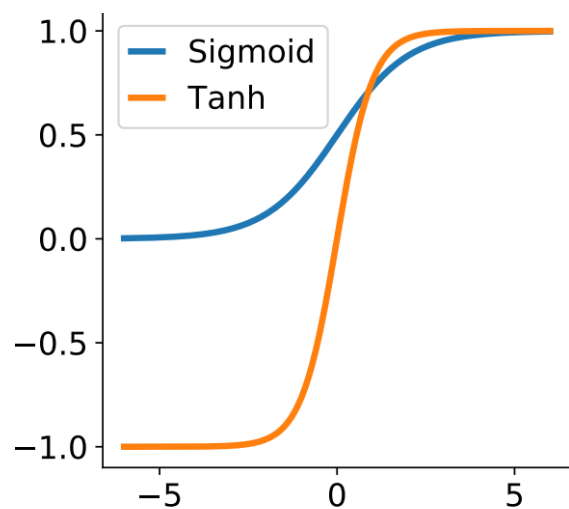




激活函数

➤ 激活函数饱和(saturating)

- 饱和(saturating)
- 导数消失





激活函数

➤ 激活函数

对每个元素作用的(非线性) 激活函数： h

$$h(z) = [\sigma(z_1), \sigma(z_2), \dots \dots \sigma(z_n)]^T$$

$$\text{RELU: } \sigma(z) = \max\{0, z\}$$

$$\text{Sigmoid: } \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\text{Tanh: } \sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\text{Leaky RELU: } \sigma(z) = \max\{z, \gamma z\} (\gamma \in (0, 1))$$

$$\text{GELU: } \sigma(z) = \frac{z}{2} \left[1 + \text{erf}\left(\frac{z}{\sqrt{2}}\right) \right]$$

$$\text{SoftPlus: } \sigma(z) = \frac{1}{\beta} \log(1 + e^{\beta z}), \beta > 0$$



神经网络

➤ 全连接神经网络(multi-layer perceptron)

在最后一层 f_L ，激活函数的选择由函数最终输出的约束条件所决定

-多类分类问题：
$$h^{(L)}(x) = \frac{e^{x_j}}{\sum_{j=1}^{R^{n_L}} e^{x_j}}$$

-正输出问题：
$$h^{(L)}(x) = e^x$$

-回归问题：没有激活函数
$$h^{(L)}(x) = x$$

➤ 回顾：回归问题

$$f(x; \theta) = W^{(L)} \phi_{\theta'}(x) + b^{(L)}$$

$\phi_{\theta'}(x)$: 特征

同时学习权重和特征



万能逼近定理

通用逼近器

\mathcal{X} 是一个紧致集。函数族 F 被称为是通用逼近器，如果 F 在函数空间 $C(\mathcal{X})$ 中稠密。这等价于说对于任意的 $f \in C(\mathcal{X})$ 和任意的 $\epsilon > 0$ ，存在 $\hat{f} \in F$ 使得

$$\sup_x |f(x) - \hat{f}(x)| < \epsilon$$

Weierstrass逼近定理

假设 $f(x)$ 是定义在实数区间 $\mathcal{X} = [a, b]$ 上的实值连续函数。对于任意的 $\epsilon > 0$ ，存在一个多项式 $p(x)$ ，使得对于所有 $[a, b]$ 区间内的 x ，都有 $|f(x) - p(x)| < \epsilon$ 。



万能逼近定理

万能逼近定理 (Cybenko 1989, Hoira 1990)

设 $\sigma: R \rightarrow R$ 是一个非常数、有界且连续的函数。令 I_d 表示 d -维单位超立方体 $[0,1]^d$ 。实数值连续函数在 I_d 上的空间记为 $C(I_d)$ 。那么，给定任何 $\epsilon > 0$ 和任何连续函数 $f \in C(I_d)$ 。存在宽度为 D 的两层神经网络，

$$F(x) = W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}$$

作为函数 f 的近似满足

$$\forall x \in I_d, |F(x) - f(x)| < \epsilon。$$



万能逼近定理

➤ 维数灾难 (Curse of dimensionality)

对于 L -Lipschitz连续函数，两层的神经网络

$$F(x) = W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}$$

对 $f \in C(I_d)$ 近似，达到 ϵ 的 L_∞ 误差估计，宽度需要为

$$D \sim \left(\frac{L}{\epsilon}\right)^d$$

对于线性函数，线性回归能很好近似

对于 C^∞ 光滑的函数，核岭回归能达到指数收敛

对于什么函数类，神经网络近似能克服维数灾难？



万能逼近定理

逼近定理 (Jones 1992, Barron 1993)

考虑cos激活函数。给定全空间的实值函数 f ，它的 Fourier 变换

$$\hat{f}(w) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} f(x) e^{-iw^T x} dx$$

满足

$$C_f = \int_{\mathbb{R}^d} |\hat{f}(w)| dw < \infty$$

给定任意概率密度函数 p ，那么存在宽度为 D 的两层神经网络 $F(x)$ ，作为函数 f 的近似满足

$$\|F(x) - f(x)\|_{L_2(p)}^2 \leq \frac{C_f^2}{D}$$



万能逼近定理

万能逼近定理 (Jones 1992, Barron 1993)

考虑sigmoid激活函数。给定紧集 Ω 上的函数 f ，考虑它的全空间 L_1 延拓 f_e ，以及相应的Fourier变换

$$\hat{f}_e(w) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} f_e(x) e^{-iw^T x} dx$$

满足

$$C_f = \inf_{f_e} \int_{\mathbb{R}^d} (1 + \|w\|) |\hat{f}_e(w)| dw < \infty$$

其中 $\|w\| = \sup_{x \in \Omega} |w^T x|$ 。给定任意概率密度函数 p ，那么存在宽度为 D 的两层神经网络 $F(x)$ ，作为函数 f 的近似满足

$$\|F(x) - f(x)\|_{L_2(p)}^2 \leq \frac{C_f^2}{D}$$



本堂课大纲

- 全连接神经网络
 - 激活函数
 - 万能逼近定理
- 神经网络中的模块
- 神经网络的训练
 - 初始化
 - 反向传播
 - 凸优化



神经网络中的模块

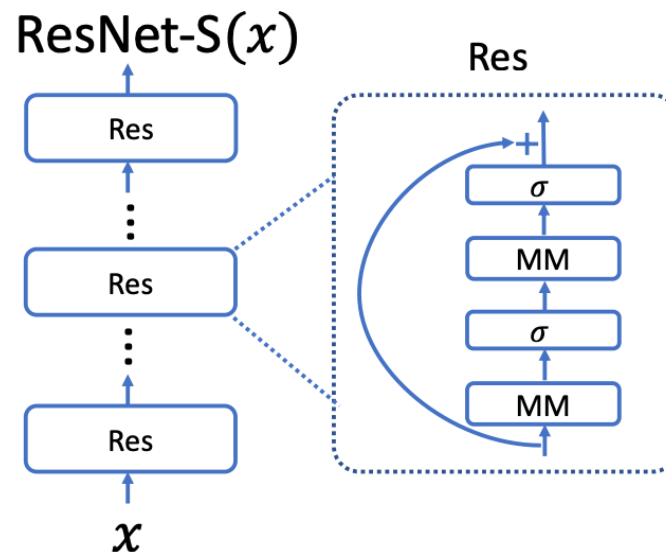
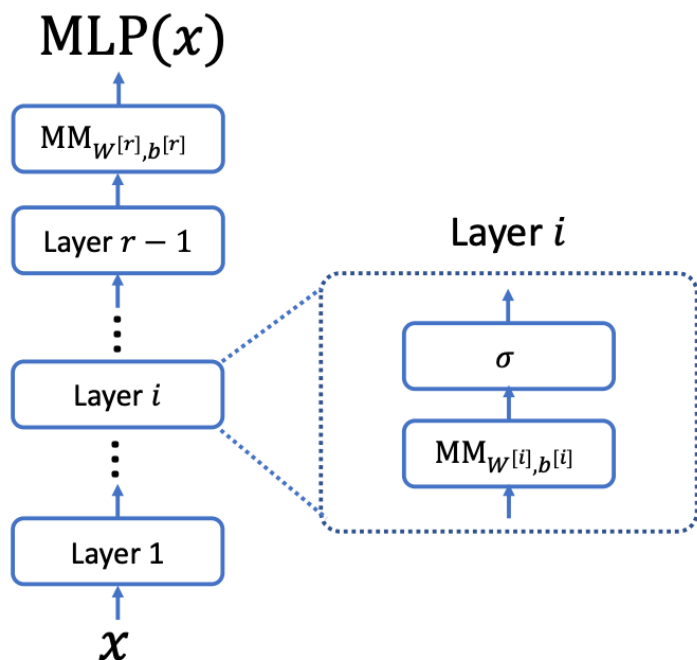
➤ 矩阵乘法模块：

$$MM(z) = Wz + b$$

➤ 残差连接模块：

$$Res(z) = z + \sigma(MM(\sigma(MM(z))))$$

ODE解释





神经网络中的模块

➤ 归一化(Layer normalization)层：

$$LN-S(z) = \begin{bmatrix} \frac{z_1 - \hat{\mu}}{\hat{\sigma}} \\ \frac{z_2 - \hat{\mu}}{\hat{\sigma}} \\ \vdots \\ \frac{z_m - \hat{\mu}}{\hat{\sigma}} \end{bmatrix} \quad \hat{\mu} = \frac{\sum_{i=1}^m z_i}{m} \quad \hat{\sigma} = \sqrt{\frac{\sum_{i=1}^m (z_i - \hat{\mu})^2}{m}}$$

$$LN(z) = \beta + \gamma LN-S(z)$$

- 尺度不变性(scaling-invariant):

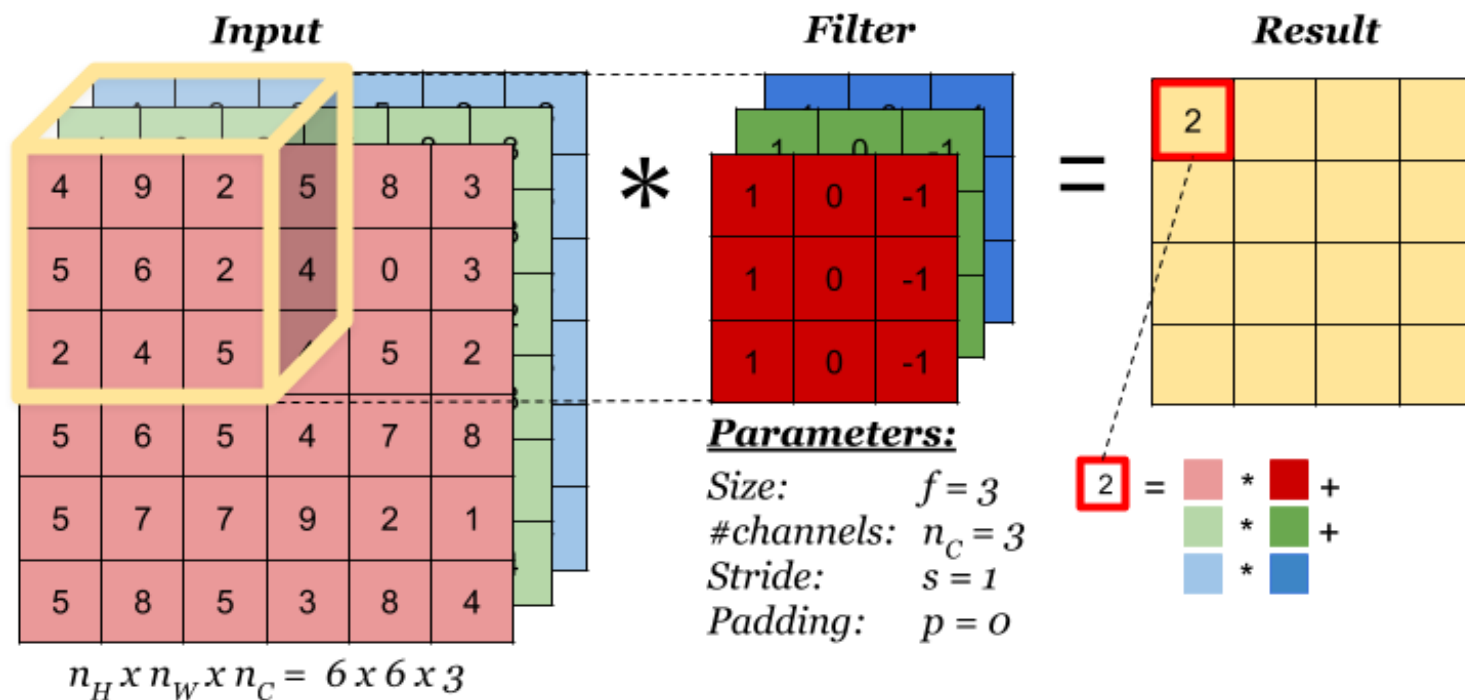
$$LN(MM_{\alpha W, \alpha b}(z)) = LN(MM_{W, b}(z))$$

- 前归一化层、后归一化层（在激活函数之后）



神经网络中的模块

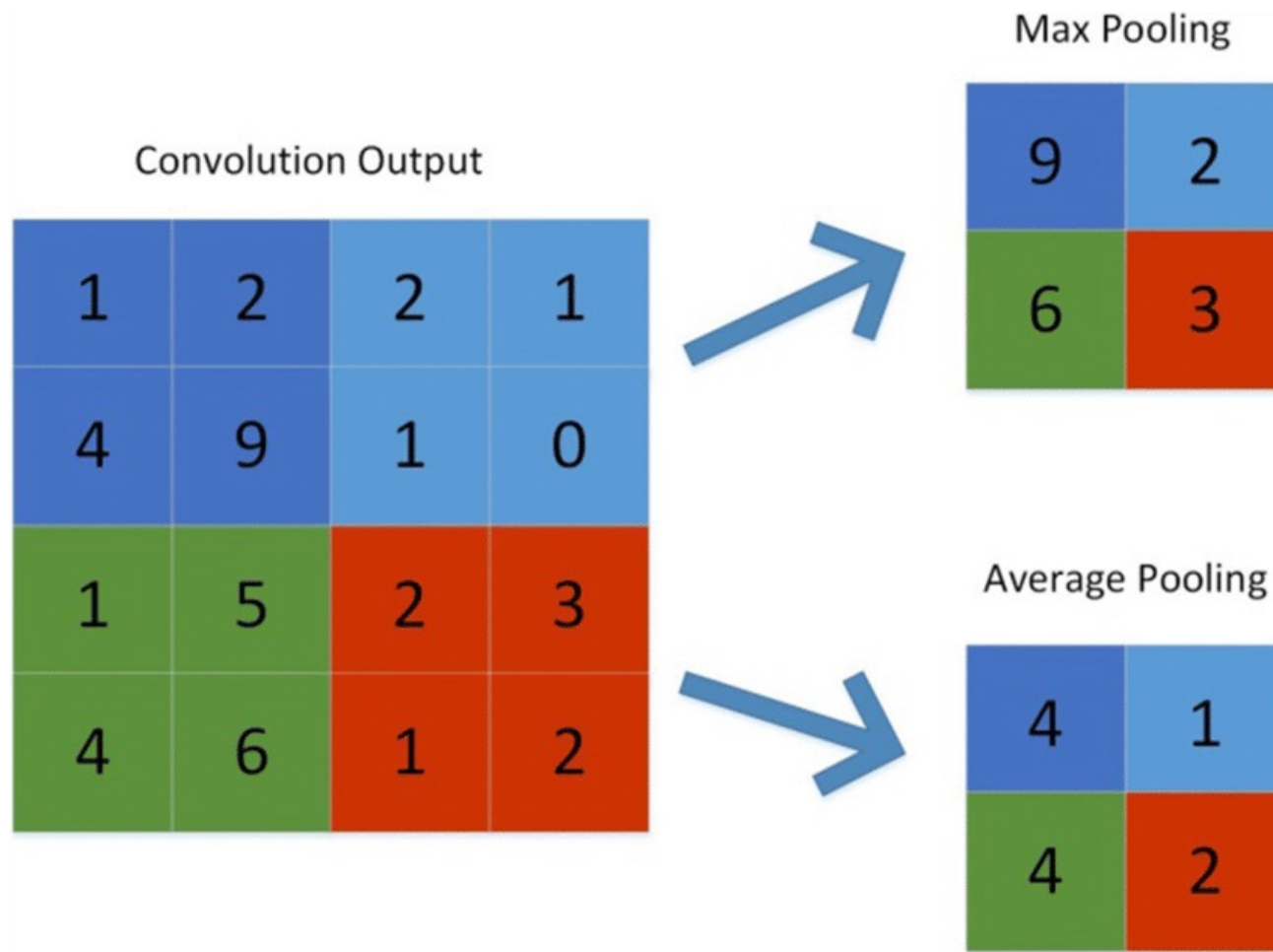
➤ 卷积模块





神经网络中的模块

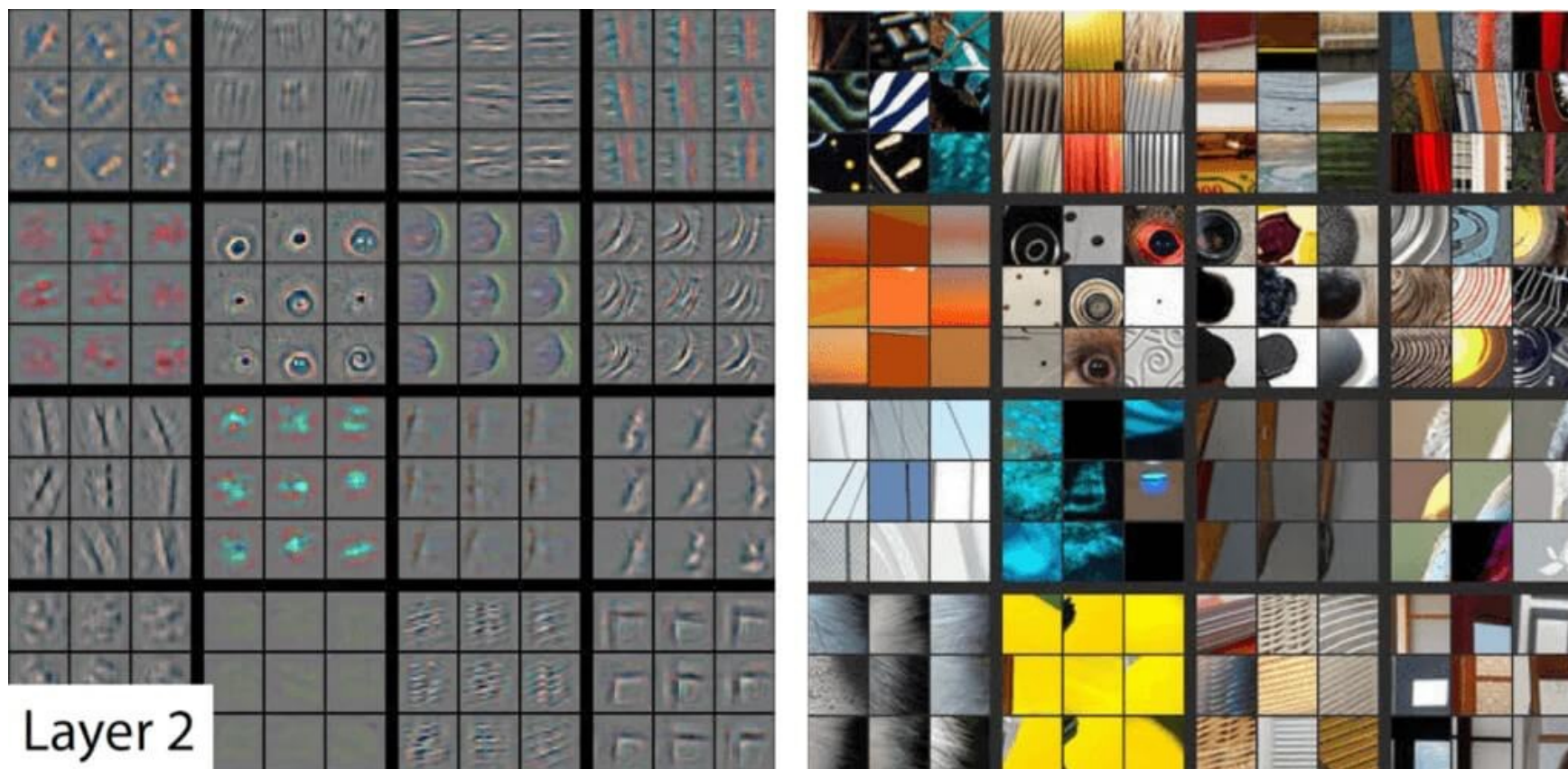
➤ 卷积模块 (池化层)





神经网络中的模块

➤ 卷积模块

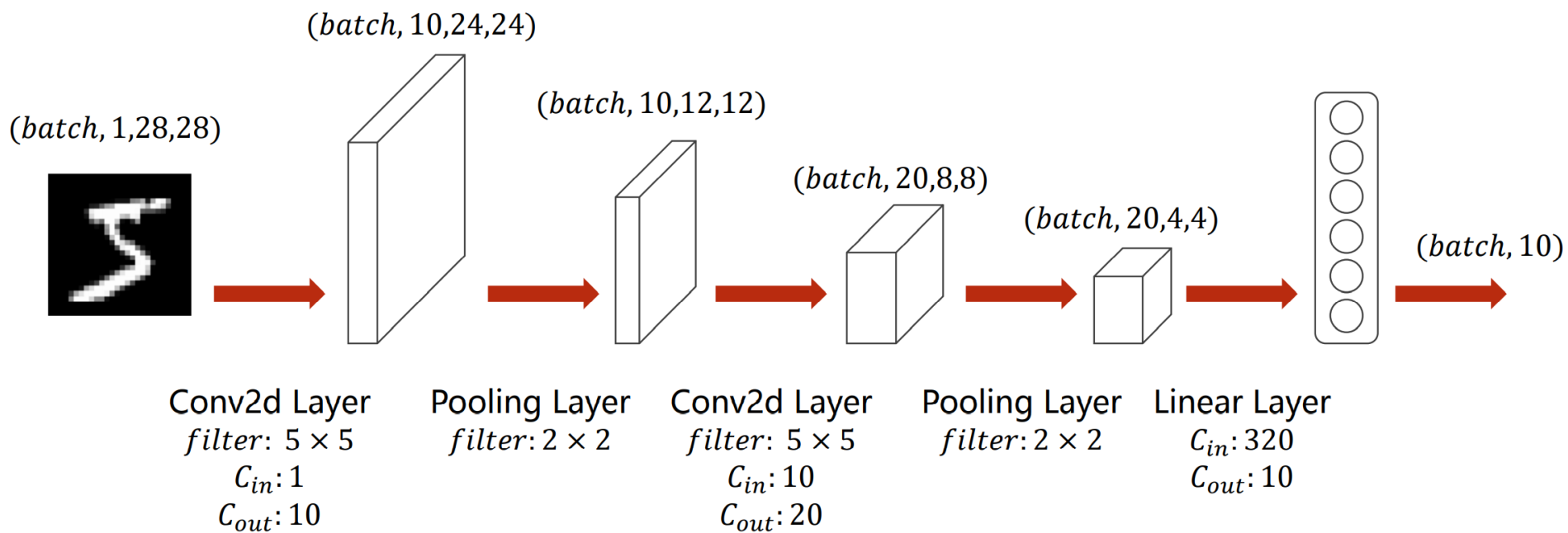


特征提取



神经网络中的模块

➤ 卷积模块

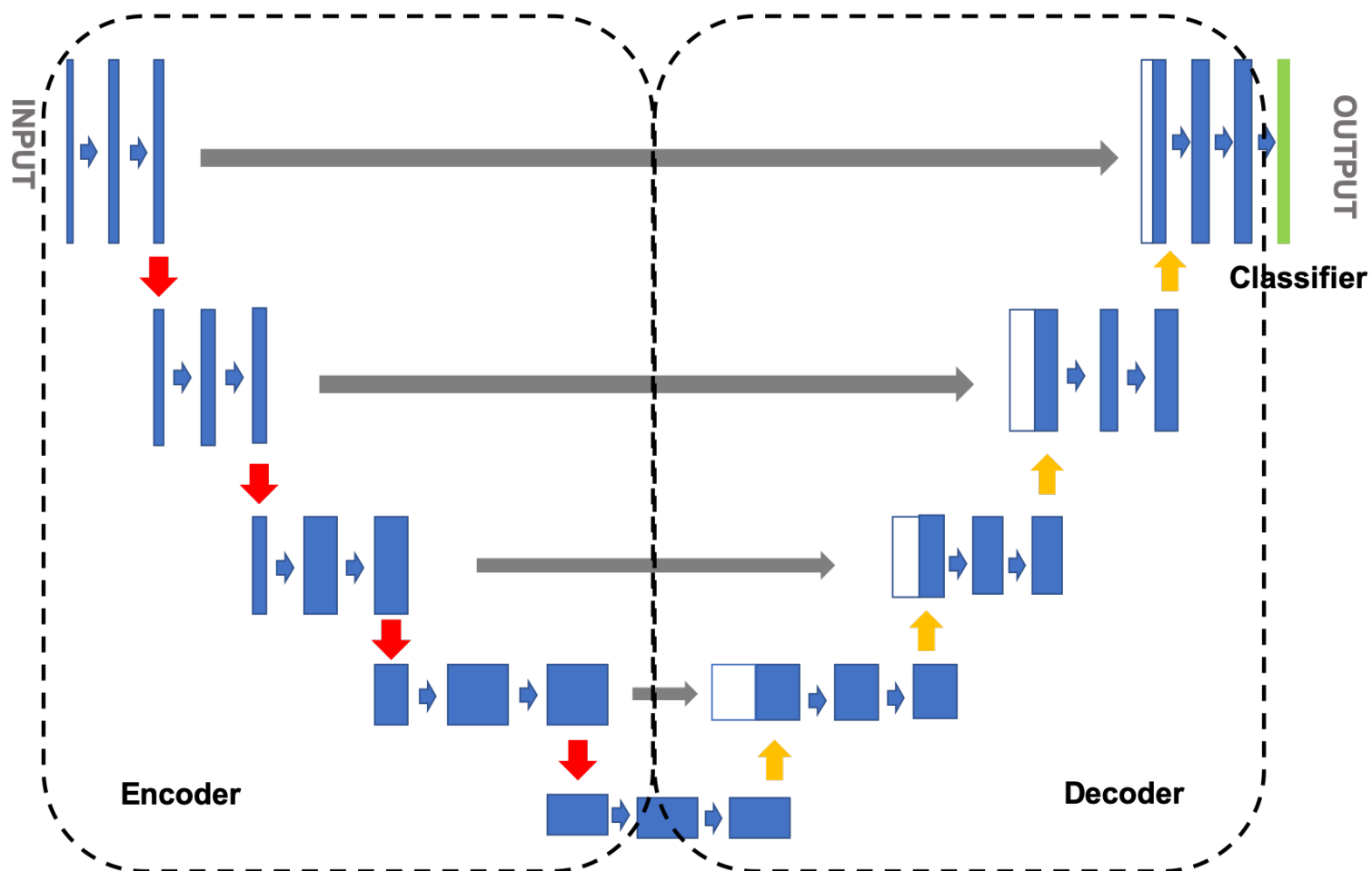


手写数字识别



神经网络中的模块

➤ U-Net





神经网络中的模块

➤ 注意力机制(Attention)

编码器状态 (encoder states) $x_1, x_2, \dots, x_n \in R^c$

解码器状态 (decoder states) $h \in R^o$

投影：

$$f = \langle h, x_1 \rangle x_1 + \langle h, x_2 \rangle x_2 + \dots + \langle h, x_n \rangle x_n$$

计算编码器状态和解码状态的相关性：

$$\text{score}(h, x_i)$$

注意力权重：

$$a_i = \frac{\exp(\text{score}(h, x_i))}{\sum_{i=1}^n \exp(\text{score}(h, x_i))}$$

输出： $f = a_1x_1 + a_2x_2 + \dots + a_nx_n$



神经网络中的模块

➤ 自注意力机制(Attention)

编码器状态 (encoder states) $x_1, x_2, \dots, x_n \in R^c$

查询 (query): $q_i = W^q x_i$ ($W^q \in R^{m \times c}$)

键 (key): $k_i = W^k x_i$ ($W^k \in R^{m \times c}$)

值 (value): $v_i = W^v x_i$ ($W^v \in R^{m \times c}$)

相关性 (alignment \ score):

$$e_{ij} = \frac{q_i \cdot k_j}{\sqrt{m}} \quad a_{ij} = \frac{\exp(e_{ij})}{\sum_{j'=1}^n \exp(e_{ij'})}$$

输出 :

$$\begin{aligned} \text{Attention}(q, k, v)_i &= \sum_{j=1}^n a_{ij} v_j \\ &= \langle q_i, k_1 \rangle v_1 + \langle q_i, k_2 \rangle v_2 + \dots + \langle q_i, k_n \rangle v_n \end{aligned}$$



神经网络中的模块

➤ 多头自注意力机制(Attention)

编码器状态 (encoder states) $x_1, x_2, \dots, x_n \in R^c$

对每一个头 $W_i^q, W_i^k, W_i^v \in R^{\frac{m}{h} \times c}$ ($i = 1, \dots, h$)

有输出：

$$\text{Attention}_i(x) \in R^{\frac{m}{h} \times n}$$

把所有头的结果合并，输乘以权重 $W_o \in R^{m \times m}$ 输出：

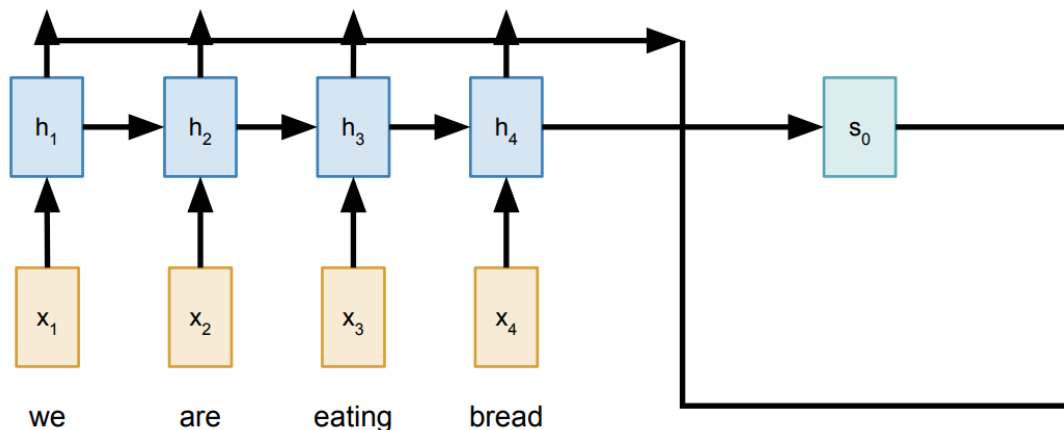
$$\text{MultiheadAttention}(x) = W_o \begin{bmatrix} \text{Attention}_1(x) \\ \text{Attention}_2(x) \\ \vdots \\ \text{Attention}_h(x) \end{bmatrix} \in R^{m \times n}$$



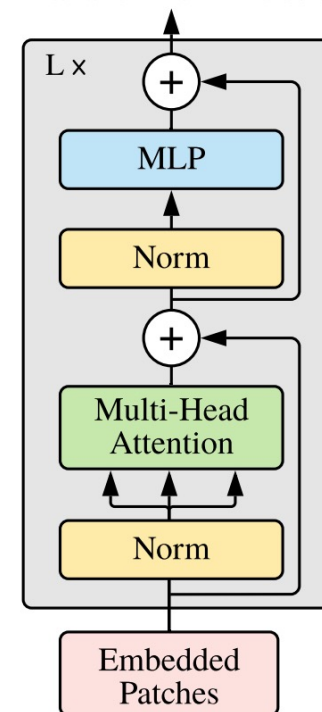
神经网络中的模块

➤ 自注意力机制(Attention)

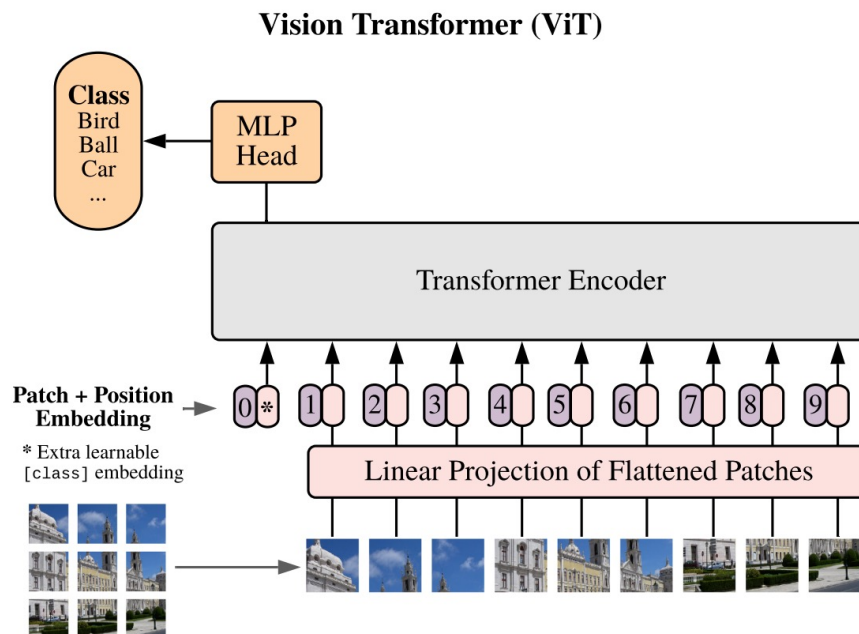
自然语言处理



Transformer Encoder



图像处理





本堂课大纲

- 全连接神经网络
 - 激活函数
 - 万能逼近定理
- 神经网络中的模块
- 神经网络的训练
 - 初始化
 - 反向传播
 - 凸优化

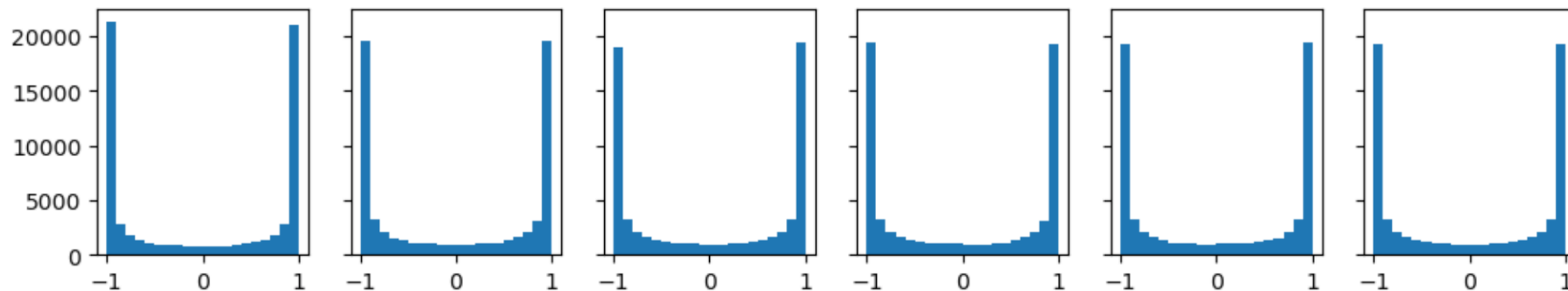


初始化

➤ 例子

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])

for n_in, n_out in zip(dims[:-1], dims[1:]):
    W = 0.05 * np.random.randn(n_in, n_out)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

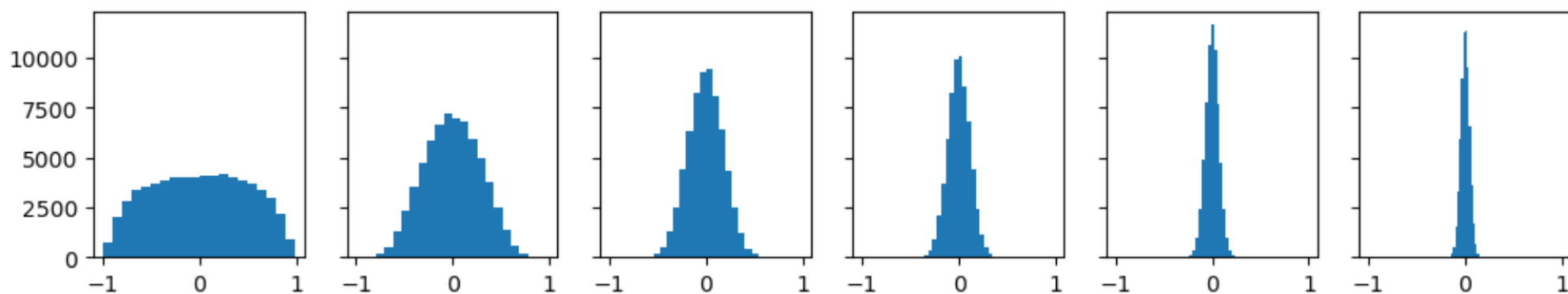




初始化

➤ 例子

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for n_in, n_out in zip(dims[:-1], dims[1:]):
    W = 0.01 * np.random.randn(n_in, n_out)
    x = np.tanh(x.dot(W))
    hs.append(x)
```





初始化

➤ Xavier初始化

考虑 $y = Wz$ ，假设 $W_{i,j} \sim \mathcal{N}(0, w^2)$

$$\text{Cov}[y_i] = n_{in} w^2 \times \text{Cov}[z_i] \Rightarrow w = \frac{1}{\sqrt{n_{in}}}$$

如果激活函数为ReLU，考虑 $y = \sigma(Wz)$ ，假设 $W_{i,j} \sim \mathcal{N}(0, w^2)$

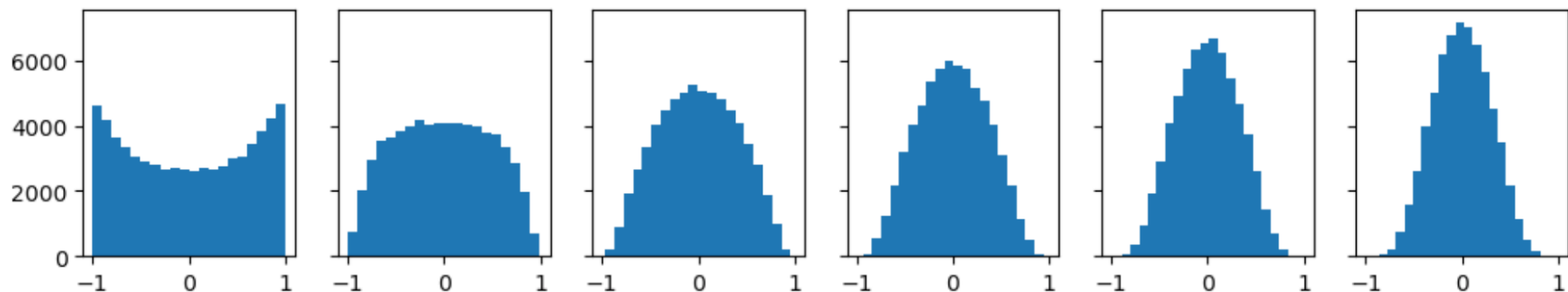
$$\text{Cov}[y_i] = \frac{1}{2} n_{in} w^2 \times \text{Cov}[z_i] \Rightarrow w = \frac{1}{\sqrt{\frac{n_{in}}{2}}}$$



初始化

➤ 例子

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for n_in, n_out in zip(dims[:-1], dims[1:]):
    W = np.random.randn(n_in, n_out) / np.sqrt(n_in)
    x = np.tanh(x.dot(W))
    hs.append(x)
```





反向传播

➤ 链式法则

$$r(x, y, z) = [z(x^2 + y)]^2$$

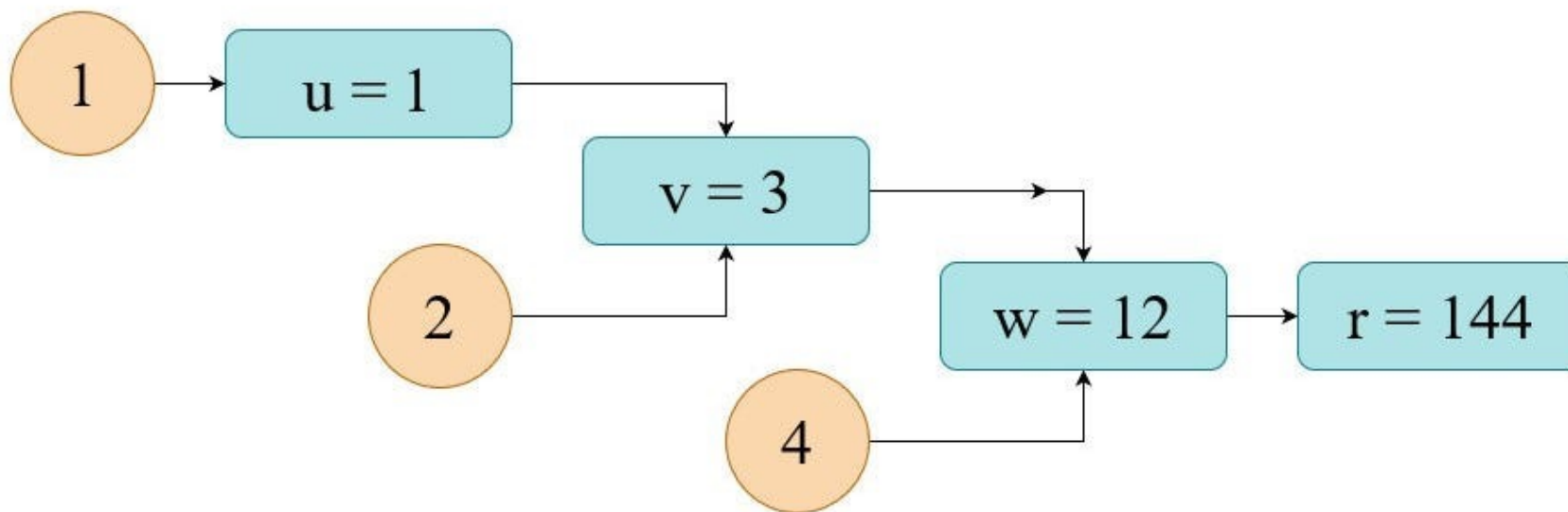
$$u = x^2$$

$$v = u + y$$

$$w = zv$$

$$r = w^2$$

$$(x, y, z) = (1, 2, 4)$$





反向传播

链式法则

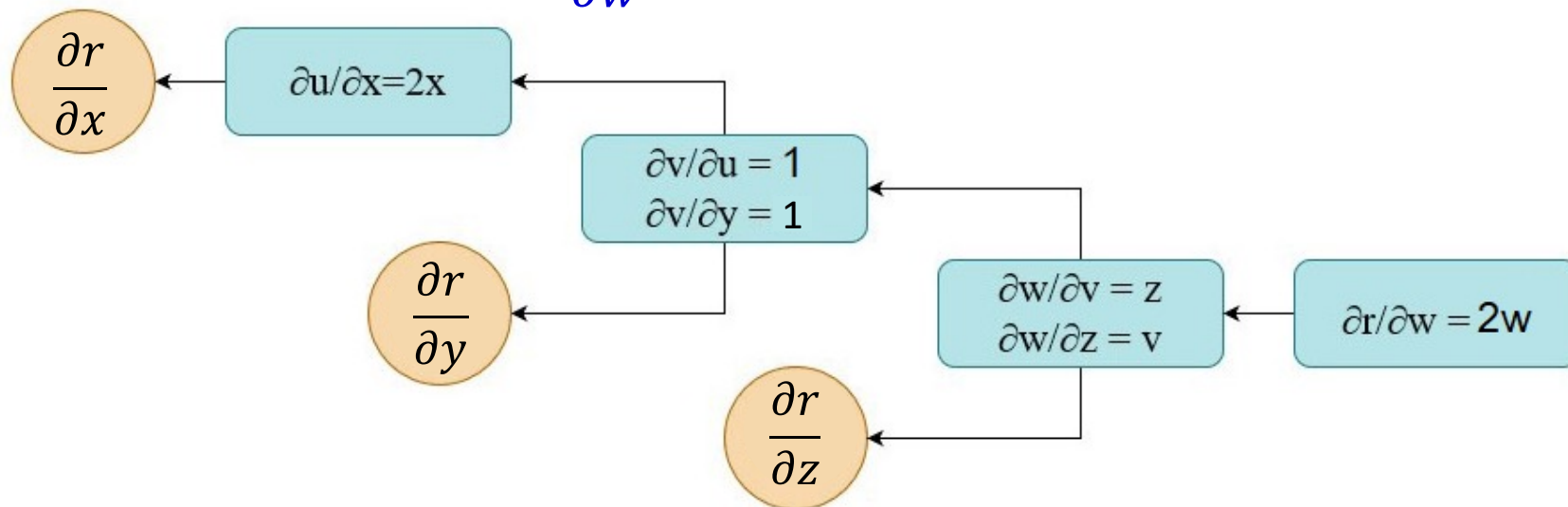
$$r(x, y, z) = z^2(x^2 + y)^2$$

$$u = x^2 \quad \frac{\partial u}{\partial x} = 2x$$

$$v = u + y \quad \frac{\partial v}{\partial u} = 1 \quad \frac{\partial v}{\partial y} = 1$$

$$w = zv \quad \frac{\partial w}{\partial v} = z \quad \frac{\partial w}{\partial z} = v$$

$$r = w^2 \quad \frac{\partial r}{\partial w} = 2w$$





反向传播

➤ 链式法则

$$f(x; \theta) = f_L \circ f_{L-1} \circ \dots \circ f_1(x)$$

我们神经网络中的操作(加减乘除激活函数等), 以及它们的导数或偏导数, 都可以在 $O(1)$ 时间内计算出来。如果 $f(x; \theta)$ 的复杂度为 $O(N)$, 那么计算 $\nabla_{\theta} f(x; \theta)$ 的复杂度也为 $O(N)$, 但是需要额外存储中间变量。



非凸优化

➤ 监督学习

极小化均方误差函数：

$$J(\theta) = \frac{1}{n} \sum_{j=1}^n J^{(i)}(\theta) \quad J^{(i)}(\theta) = \frac{1}{2} (f(x_i; \theta) - y_i)^2$$

➤ 最速度梯度下降法 (steepest gradient descent)

$$\begin{aligned} \theta &:= \theta - \alpha \nabla_{\theta} J(\theta) \\ &:= \theta - \alpha \frac{1}{n} \sum_{i=1}^n \nabla_{\theta} J^{(i)}(\theta) \end{aligned}$$



非凸优化

➤ 随机梯度下降法 (stochastic gradient descent)

均匀地 (无放回) 从 $\{1, 2, \dots, n\}$ 中选取 B 个实例 j_1, j_2, \dots, j_B

$$\theta := \theta - \alpha \frac{1}{|B|} \sum_{k=1}^B \nabla_{\theta} J^{(j_k)}(\theta)$$

每次把 $\{1, 2, \dots, n\}$ 分为 k 组 B_1, B_2, \dots, B_k

$$\theta := \theta - \alpha \frac{1}{|B_j|} \sum_{i \in B_j} \nabla_{\theta} J^{(i)}(\theta) \quad j = 1, 2, \dots, k$$

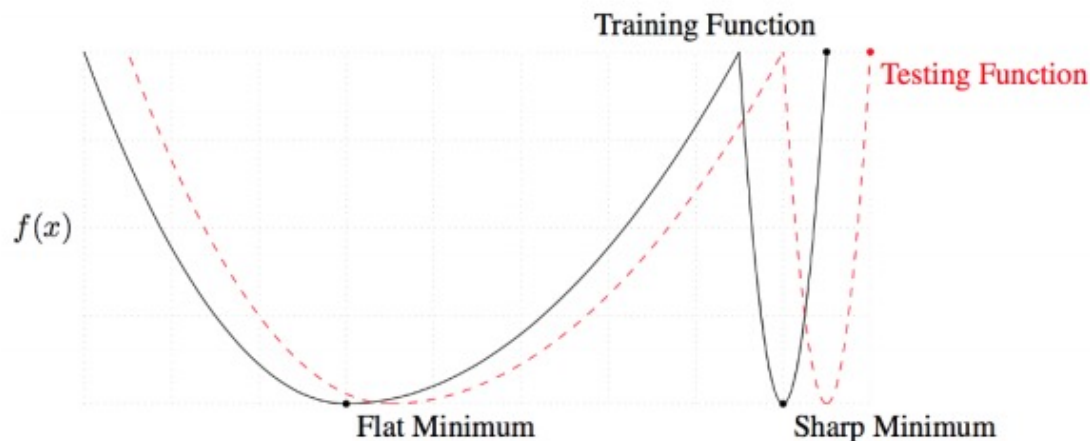


非凸优化

➤ 随机梯度下降法 (stochastic gradient descent)

$$\frac{1}{|B_j|} \sum_{i \in B_j} \nabla_{\theta} J^{(i)}(\theta) \approx \nabla_{\theta} E[J(\theta)]$$

$$\frac{1}{|B_j|} \sum_{i \in B_j} \nabla_{\theta} J^{(i)}(\theta) = \nabla_{\theta} E[J(\theta)] + \text{Random perturbation}$$

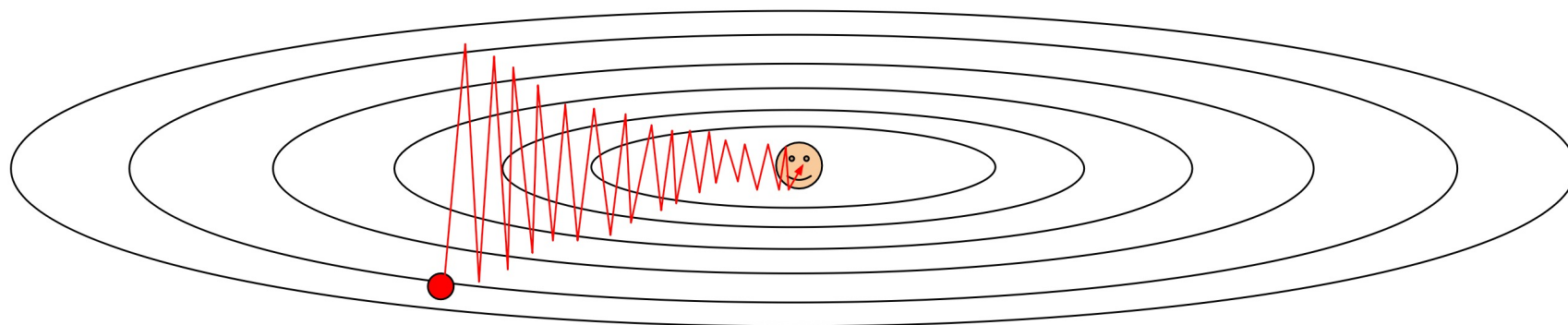


To fall into a sharp minimum, the gradient must be precise. The noise in SGD prevents that. It is hypothesized that flat minima generalize better.



非凸优化

➤ 最速度梯度下降法



- 损失函数具有高条件数，损失函数在一个方向上变化很快，而在另一个方向上变化很慢，会出现震荡。

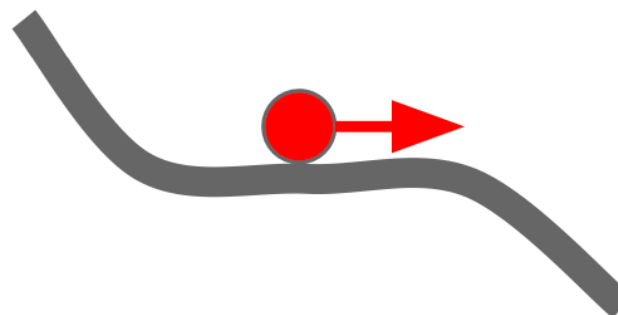
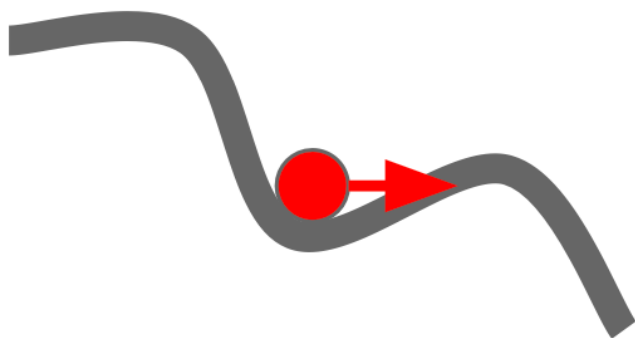
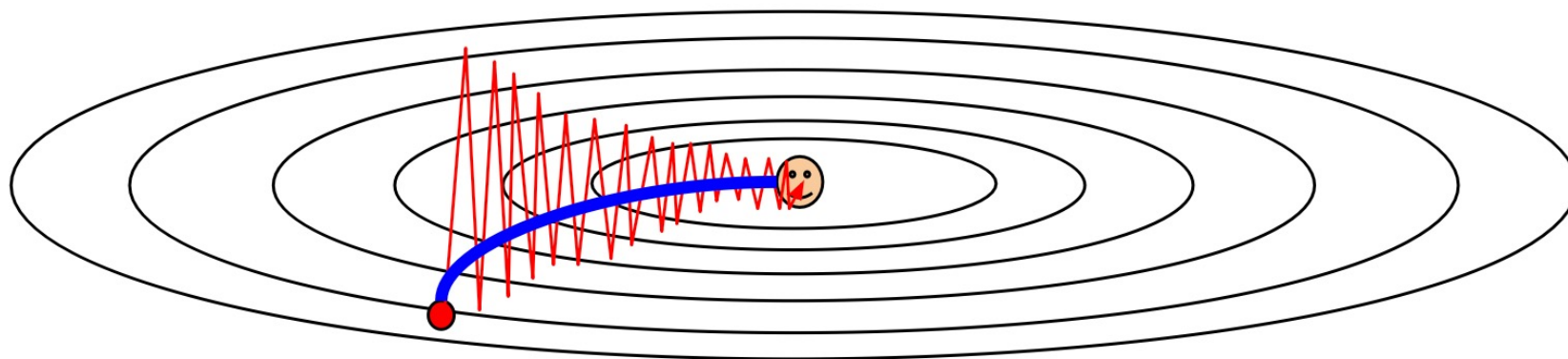


- 鞍点和局部极小点



非凸优化

➤ 动量更新





非凸优化

➤ 动量更新

Polyak重球法(heavy ball method)

$$\begin{aligned}y_k &= x_k + \beta(x_k - x_{k-1}) \\x_{k+1} &= y_k - \alpha \nabla f(x_k)\end{aligned}$$

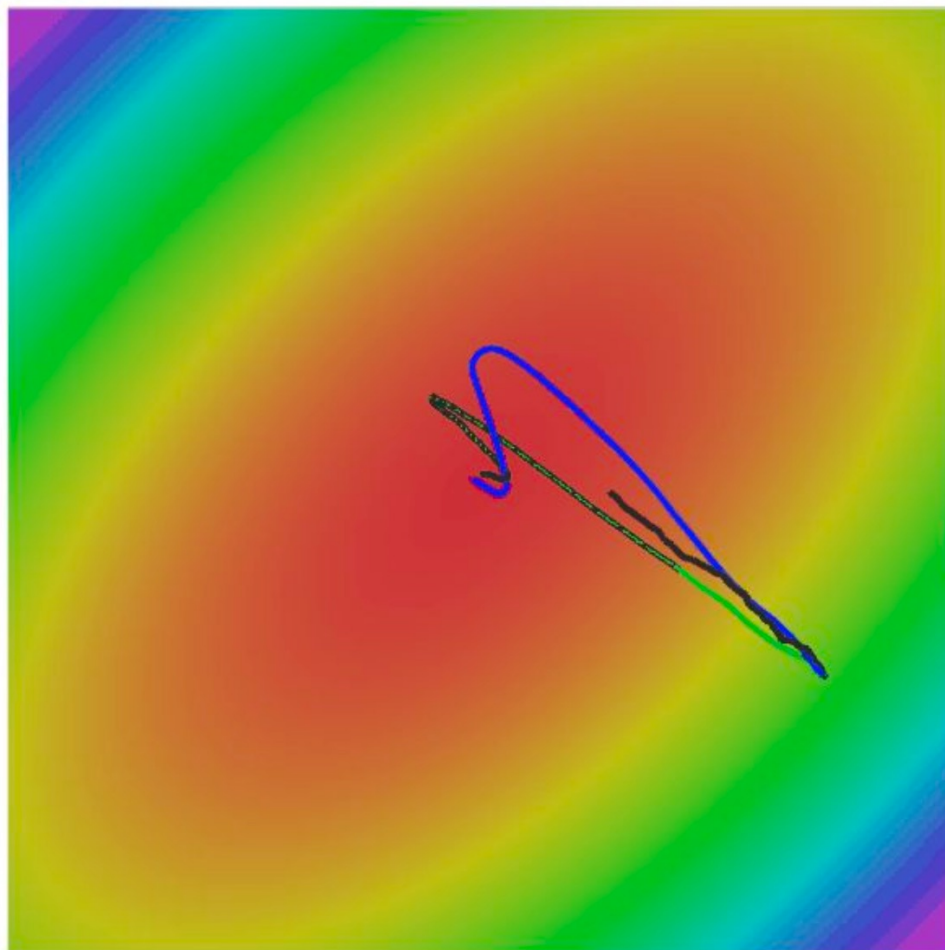
Nesterov 加速梯度下降(accelerated gradient descent)

$$\begin{aligned}y_k &= x_k + \beta(x_k - x_{k-1}) \\x_{k+1} &= y_k - \alpha \nabla f(y_k)\end{aligned}$$



非凸优化

➤ 动量更新



- SGD
- SGD+Momentum
- Nesterov



非凸优化

➤ 步长选取

AdaGrad

$$dx_k = \nabla f(x_k)$$

$$s_{k+1}^2 = s_k^2 + dx_k \odot dx_k$$

每个参数导数累积量

$$x_{k+1} = x_k - \frac{\alpha}{\sqrt{s_{k+1}^2 + 10^{-7}}} \odot dx_k$$

RMSProp

$$dx_k = \nabla f(x_k)$$

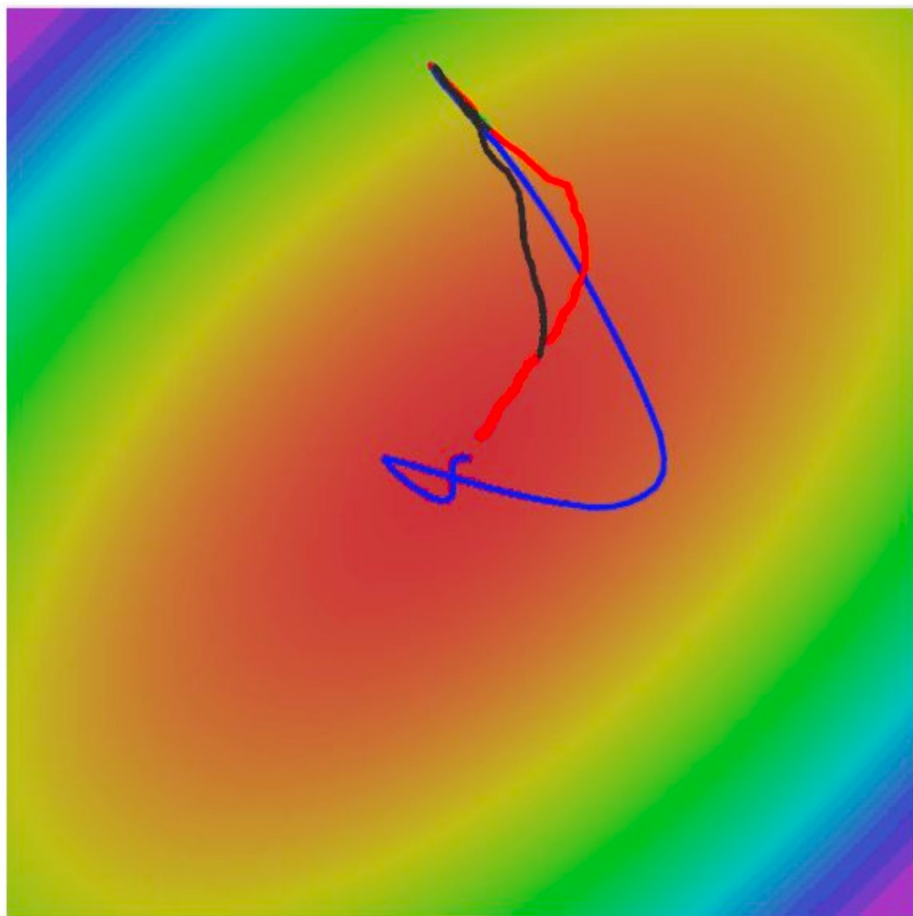
$$s_{k+1}^2 = \beta s_k^2 + (1 - \beta) dx_k \odot dx_k$$

$$x_{k+1} = x_k - \frac{\alpha}{\sqrt{s_{k+1}^2 + 10^{-7}}} \odot dx_k$$



非凸优化

➤ 步长选取



- SGD
- SGD+Momentum
- RMSProp



非凸优化

自适应矩估计(Adaptive Moment Estimation, Adam)

$$dx_k = \nabla f(x_k)$$

$$v_{k+1} = \beta_1 v_k + (1 - \beta_1) dx_k$$

$$s_{k+1} = \beta_2 s_k + (1 - \beta_2) dx_k \odot dx_k$$

偏差修正：

$$\hat{v}_{k+1} = \frac{v_{k+1}}{1 - \beta_1^{k+1}} \quad \hat{s}_{k+1} = \frac{s_{k+1}}{1 - \beta_2^{k+1}}$$

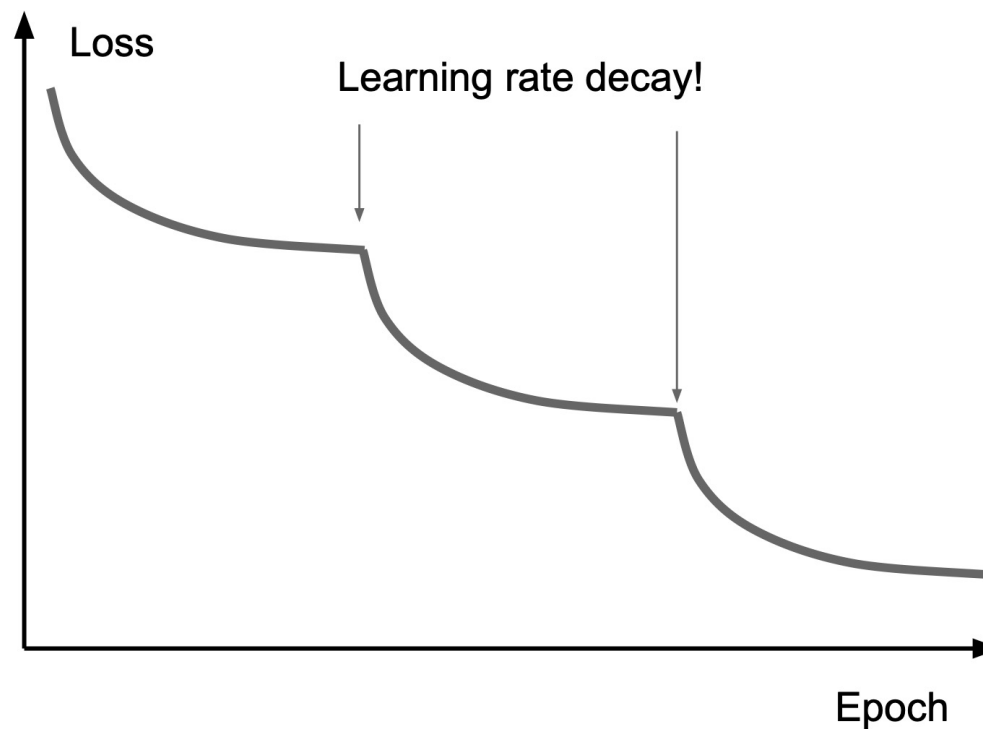
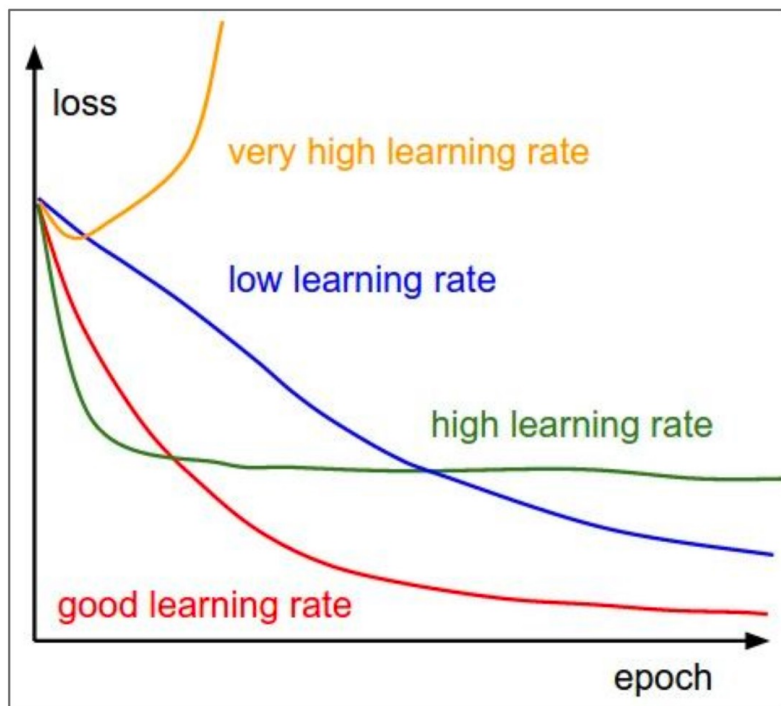
$$x_{k+1} = x_k - \frac{\alpha}{\sqrt{\hat{s}_{k+1} + 10^{-7}}} \odot \hat{v}_{k+1}$$

参数 $\beta_1 = 0.9$ ， $\beta_2 = 0.999$ 和 α



非凸优化

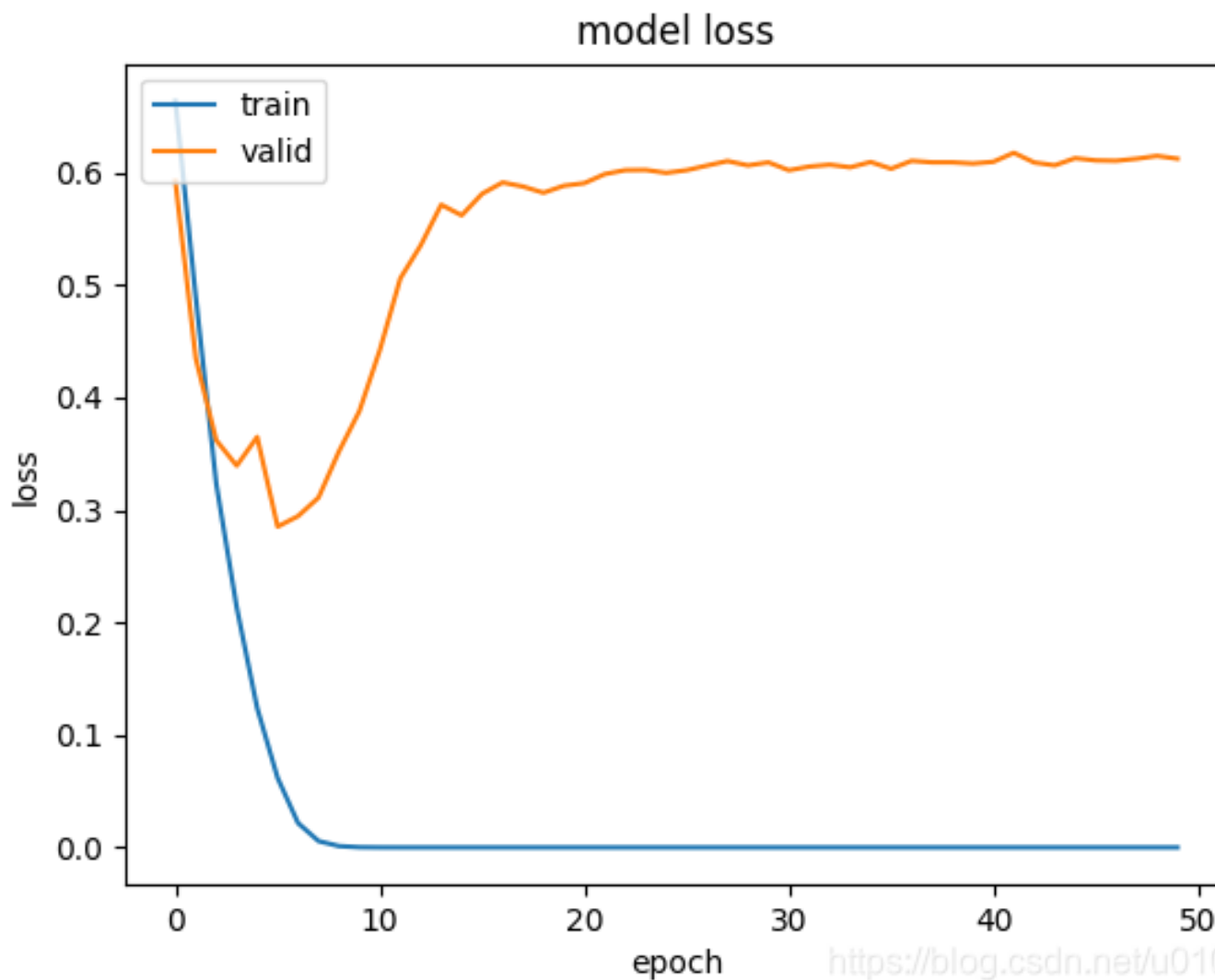
➤ 学习率 α 的取





非凸优化

过拟合





非凸优化

➤ 过拟合：正则化

$$J(\theta) = J(\theta) + \lambda R(\theta)$$

L_2 正则化 (weight decay) :

$$R(\theta) = \sum \theta_i^2$$

L_1 正则化 :

$$R(\theta) = \sum |\theta_i|$$

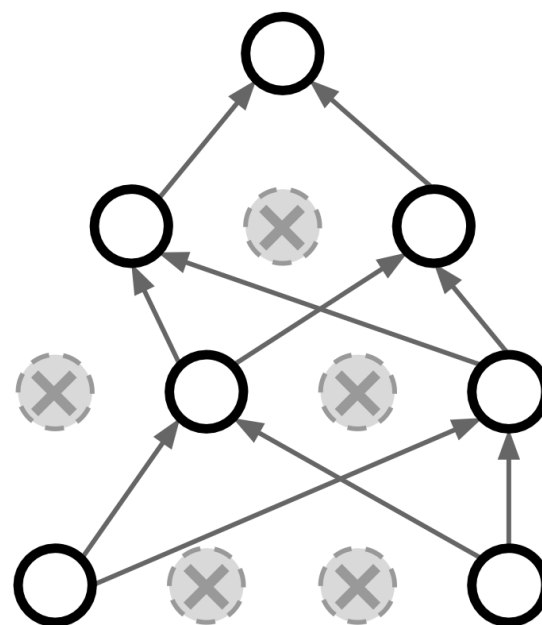
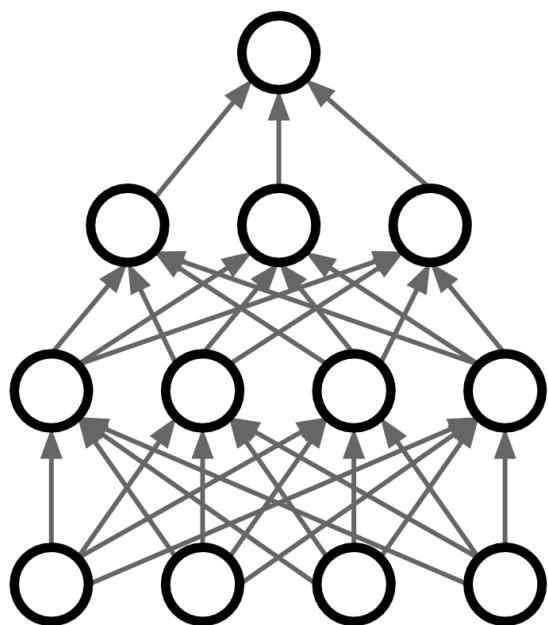
弹性网络 (Elastic net) :

$$R(\theta) = \beta \sum \theta_i^2 + \sum |\theta_i|$$



非凸优化

过拟合：丢弃法(Dropout)



- 在训练的时候(每一个批次(batch))，以概率 $1 - p$ 随机删去该层的一些输出，并把其余输出乘以 $1/p$
- 训练了多个模型



非凸优化

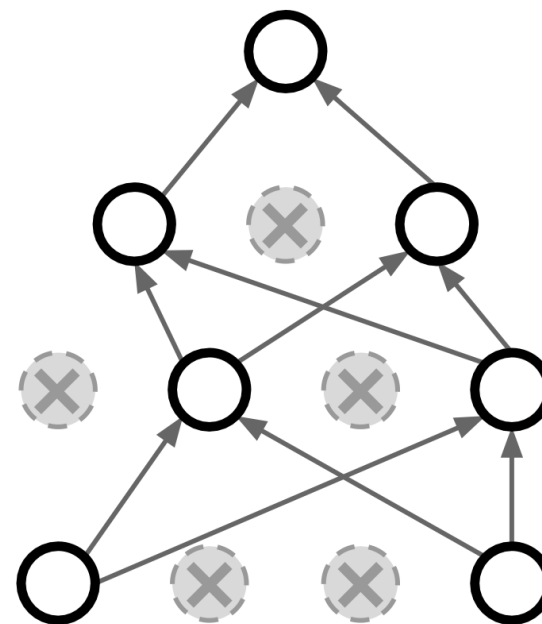
过拟合：丢弃法(Dropout)

```
p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
    # forward pass for example 3-layer neural network
    H1 = np.maximum(0, np.dot(W1, X) + b1)
    U1 = (np.random.rand(*H1.shape) < p) / p # first dropout mask. Notice /p!
    H1 *= U1 # drop!
    H2 = np.maximum(0, np.dot(W2, H1) + b2)
    U2 = (np.random.rand(*H2.shape) < p) / p # second dropout mask. Notice /p!
    H2 *= U2 # drop!
    out = np.dot(W3, H2) + b3

    # backward pass: compute gradients... (not shown)
    # perform parameter update... (not shown)

def predict(X):
    # ensembled forward pass
    H1 = np.maximum(0, np.dot(W1, X) + b1) # no scaling necessary
    H2 = np.maximum(0, np.dot(W2, H1) + b2)
    out = np.dot(W3, H2) + b3
```



测试时保持不变



参考文献

➤ 参考文献

斯坦福CS231课件：cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Cybenko, George. "Approximation by superpositions of a sigmoidal function." *Mathematics of control, signals and systems* 2, no. 4 (1989): 303-314.

Hornik, Kurt. "Approximation capabilities of multilayer feedforward networks." *Neural networks* 4, no. 2 (1991): 251-257.

Jones, Lee K. "A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training." *The annals of Statistics* (1992): 608-613.

Barron, Andrew R. "Universal approximation bounds for superpositions of a sigmoidal function." *IEEE Transactions on Information theory* 39, no. 3 (1993): 930-945.

威斯康星麦迪逊CS726课件：[Lecture 9 – 10: Accelerated Gradient Descent \(wisc.edu\)](https://wisc.edu)