神经网络和反向传播
NNs and Back Propagation

任课教师：孔雨秋

yqkong@dlut.edu.cn

目 录 (CONTENT)

大连理工大学 Dalian University of Technology

01 多层感知机 (Multi-layer Perceptron)

02 前馈神经网络 (Feedforward Neural Network)

03 梯度反向传播 (Gradient Back Propagation)

04 闭形式解 (Close-form Solution)

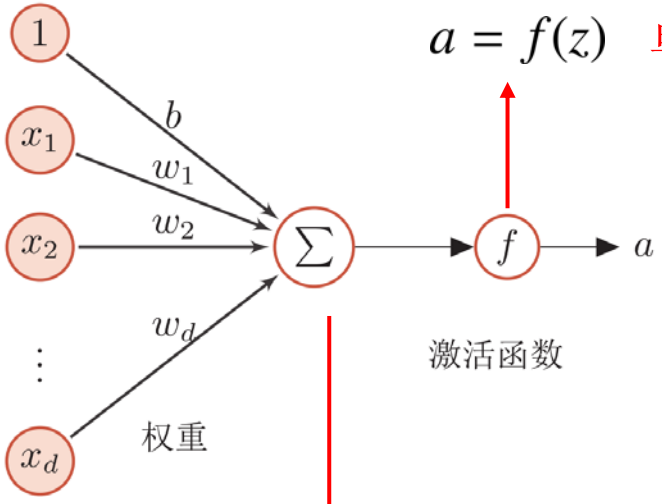
大连理工大学
Dalian University Of Technology

PART ONE 01

多层感知机
Multi-layer Perceptron

单层感知机

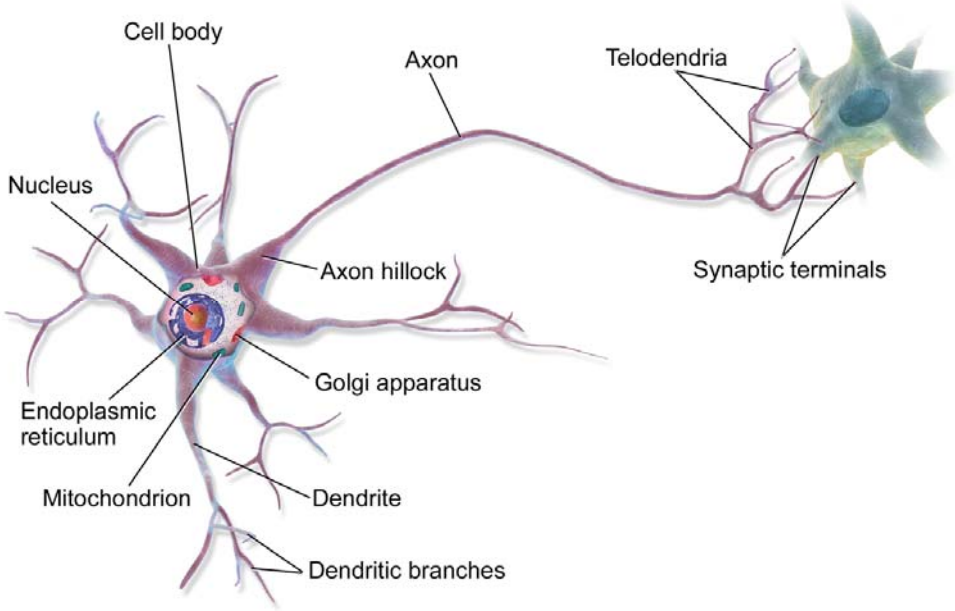
■ 单层感知机结构



$a = f(z)$ 单个神经细胞只有两种状态：兴奋和抑制

$$z = \sum_{i=1}^{d} w_i x_i + b = \mathbf{w}^T \mathbf{x} + b$$

一个简单的线性模型！



单层感知机

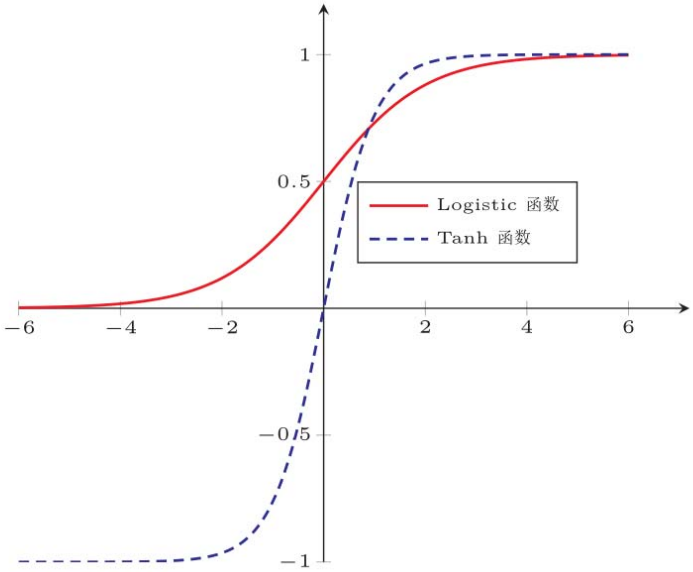
■ 常见激活函数

□ Sigmoid激活函数

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

□ Tanh激活函数

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



□ 性质

- ✓ 饱和函数：自变量绝对值较大的时候，值接近；
- ✓ tanh函数是零中心化的，而logistic函数的输出恒大于0；
- ✓ 非零中心化的输出会使得其后一层的神经元的输入发生偏置偏移（bias shift），并进一步使得梯度下降的收敛速度变慢。

单层感知机

大连理工大学 Dalian University of Technology

■ 常见激活函数

$$\mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

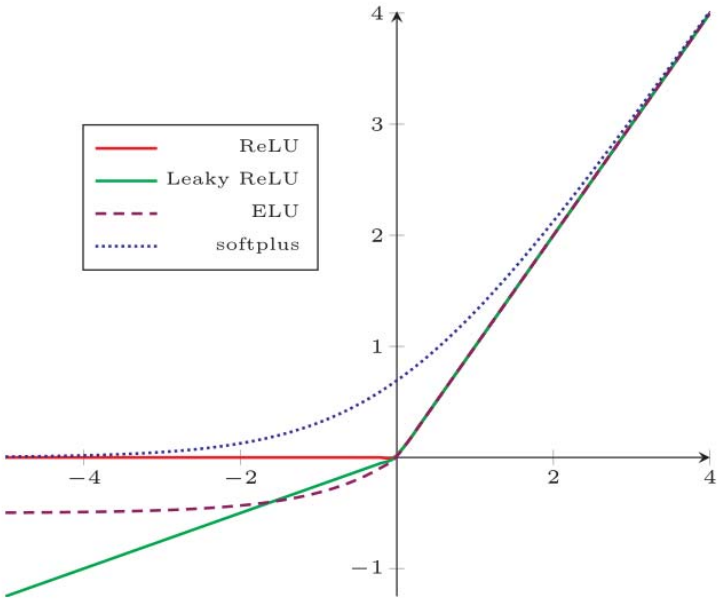
$$\mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma x & \text{if } x \leq 0 \end{cases}$$

$$\mathrm{PReLU}_i(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma_i x & \text{if } x \leq 0 \end{cases}$$

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

$$= \max(0, x) + \min(0, \gamma(\exp(x) - 1))$$

$$\mathrm{softplus}(x) = \log(1 + \exp(x))$$



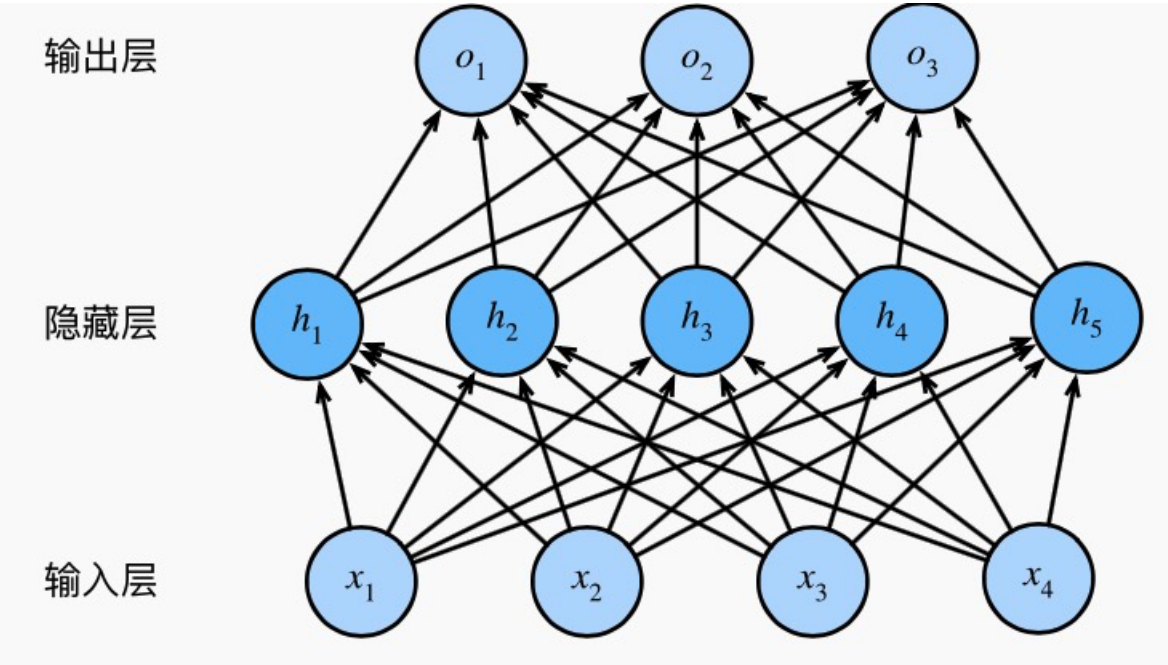
死亡ReLU问题（Dying ReLU Problem）

- 计算上更加高效
- 生物学合理性
- 单侧抑制、宽兴奋边界
- 在一定程度上缓解梯度消失问题

多层感知机

- **多层感知机**在单层感知机的基础上引入了一到多个隐藏层(hidden layer)。隐藏层位于输入层和输出层之间。不难发现，即便再添加更多的隐藏层, 以上设计依然只能与仅含输出层的单层感知机等价。

- 线性变换下，等价于一个单层感知机



$$\boldsymbol{O} = (\boldsymbol{X}\boldsymbol{W}_h + \boldsymbol{b}_h)\boldsymbol{W}_o + \boldsymbol{b}_o = \boldsymbol{X}\boldsymbol{W}_h\boldsymbol{W}_o + \boldsymbol{b}_h\boldsymbol{W}_o + \boldsymbol{b}_o.$$

大连理工大学
Dalian University Of Technology

PART TWO 02

前馈神经网络
Feedforward NNs

前馈神经网络

大连理工大学 Dalian University of Technology

■ 人工神经网络 (Artificial Neural Network)

人工神经网络主要由大量的神经元以及它们之间的有向连接构成。
因此考虑三方面：

- **神经元的激活规则**

  主要是指神经元输入到输出之间的映射关系，一般为非线性函数。

- **网络的拓扑结构**

  不同神经元之间的连接关系

- **学习算法**

  通过训练数据来学习神经网络的参数

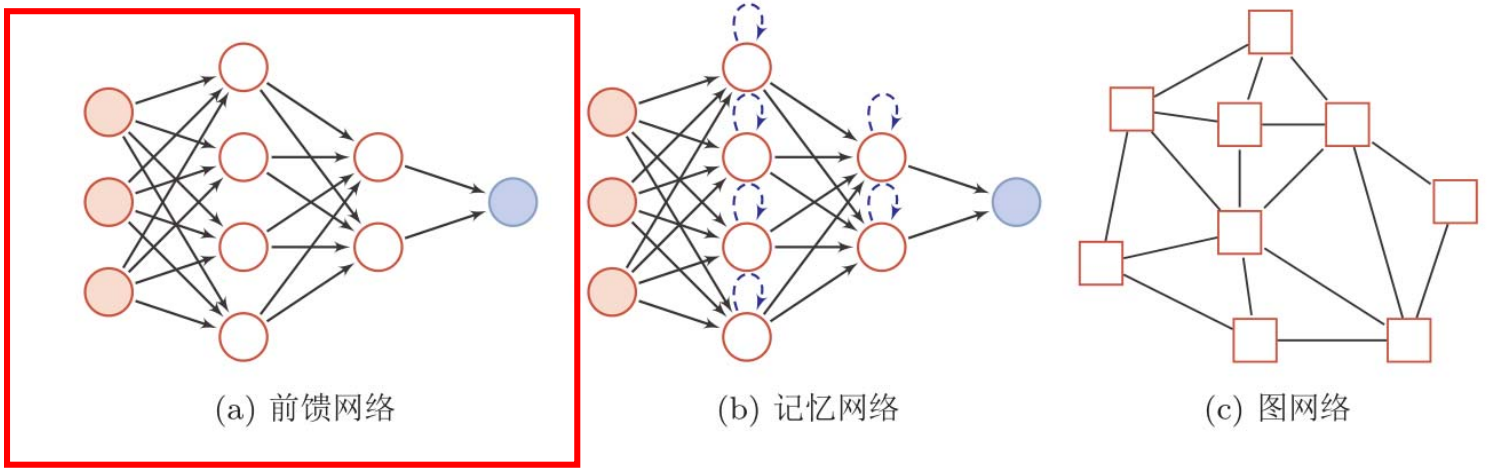
前馈神经网络

大连理工大学 Dalian University of Technology

■ 前馈神经网络 (Feedforward Neural Networks)

- **人工神经网络**由神经元模型构成，这种由许多神经元组成的信息处理网络具有并行分布结构。

- **前馈神经网络**是人工神经网络的一种形式，信息前向传递，实现逐层编码处理的网络结构。



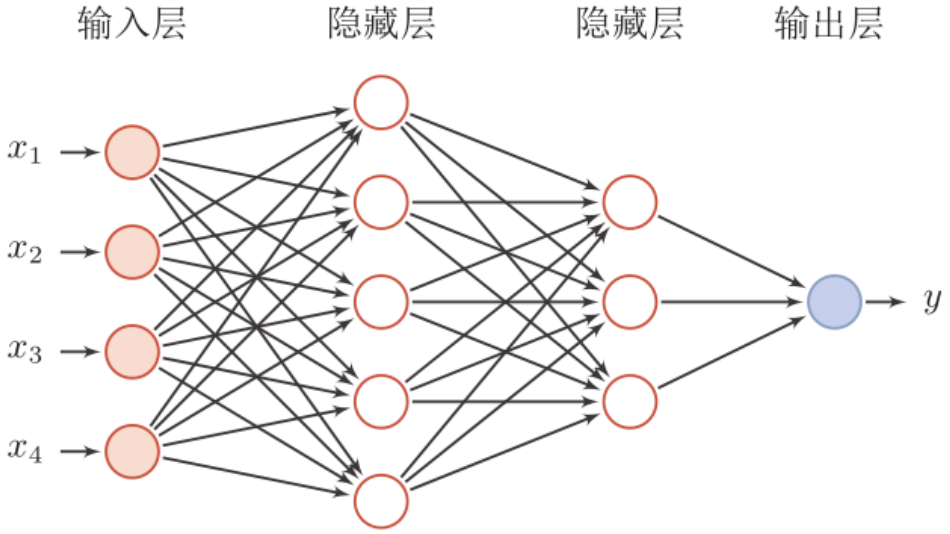
圆形节点表示一个神经元，方形节点表示一组神经元。

前馈神经网络

大连理工大学 Dalian University of Technology

■ 前馈神经网络（全连接神经网络、多层感知器）

- 各神经元分别属于不同的层，层内无连接。
- 相邻两层之间的神经元全部两两连接。
- 整个网络中无反馈，信号从输入层向输出层单向传播，可用一个有向无环图表示。



记号	含义
L	神经网络的层数
M_l	第 l 层神经元的个数
$f_l(\cdot)$	第 l 层神经元的激活函数
$\boldsymbol{W}^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$	第 $l-1$ 层到第 l 层的权重矩阵
$\boldsymbol{b}^{(l)} \in \mathbb{R}^{M_l}$	第 $l-1$ 层到第 l 层的偏置
$\boldsymbol{z}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的净输入（净活性值）
$\boldsymbol{a}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的输出（活性值）

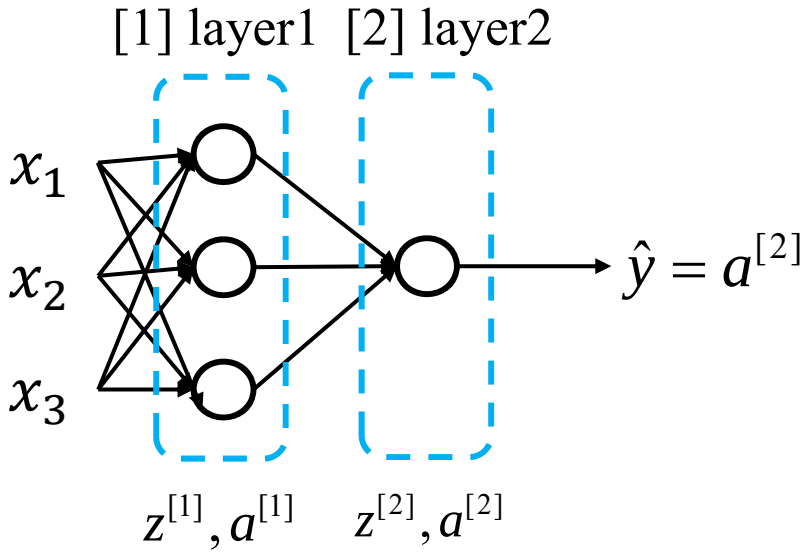
前馈神经网络

大连理工大学 Dalian University of Technology

- 逻辑回归模型

$$\left.\begin{matrix} x \\ w \\ b \end{matrix}\right\} \rightarrow z = w^T x + b \rightarrow \hat{y} = \sigma(z) \rightarrow L(\hat{y}, y)$$

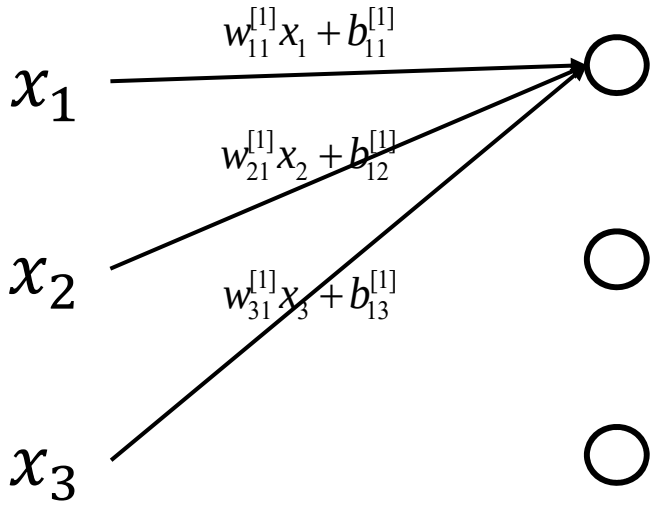
- 神经网络模型



前馈神经网络

大连理工大学 Dalian University of Technology

■ 第一层



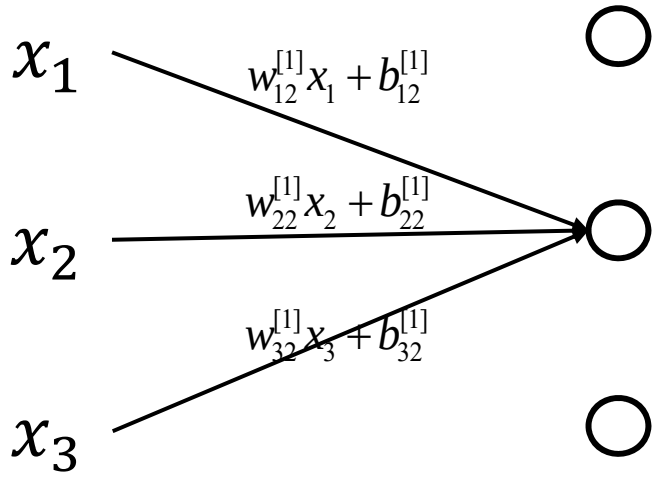
$$z_1^{[1]} = w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} = (w_{11}^{[1]}, w_{21}^{[1]}, w_{31}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_1^{[1]} = \mathbf{w}_1^{[1]T}\mathbf{x}^{[0]} + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

前馈神经网络

大连理工大学 Dalian University of Technology

■ 第一层



$$z_1^{[1]} = w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} = (w_{11}^{[1]}, w_{21}^{[1]}, w_{31}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_1^{[1]} = \mathbf{w}_1^{[1]T}\mathbf{x}^{[0]} + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

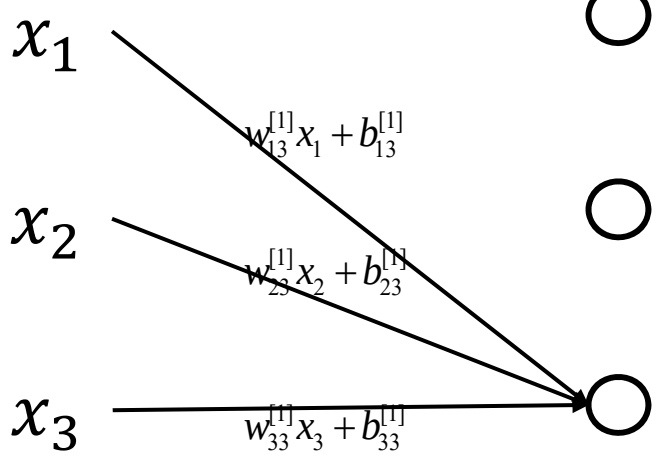
$$z_2^{[1]} = w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} = (w_{12}^{[1]}, w_{22}^{[1]}, w_{32}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_2^{[1]} = \mathbf{w}_2^{[1]T}\mathbf{x}^{[0]} + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

前馈神经网络

大连理工大学 Dalian University of Technology

■ 第一层



$$z_1^{[1]} = w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} = (w_{11}^{[1]}, w_{21}^{[1]}, w_{31}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_1^{[1]} = \mathbf{w}_1^{[1]T}\mathbf{x}^{[0]} + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} = (w_{12}^{[1]}, w_{22}^{[1]}, w_{32}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_2^{[1]} = \mathbf{w}_2^{[1]T}\mathbf{x}^{[0]} + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

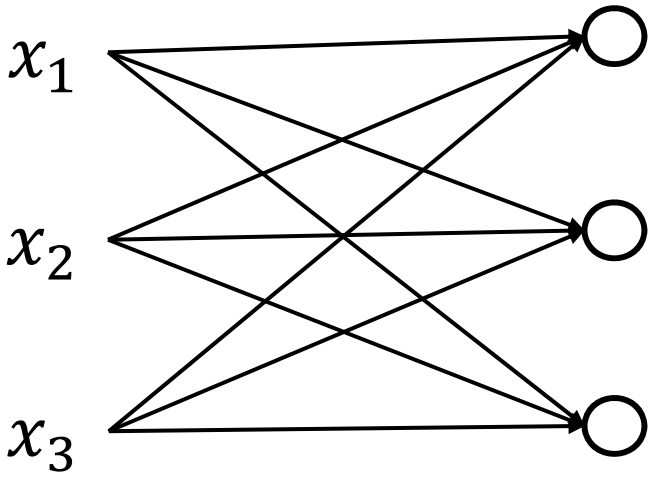
$$z_3^{[1]} = w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} = (w_{13}^{[1]}, w_{23}^{[1]}, w_{33}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_3^{[1]} = \mathbf{w}_3^{[1]T}\mathbf{x}^{[0]} + b_3^{[1]}$$

$$a_3^{[1]} = \sigma(z_3^{[1]})$$

前馈神经网络

大连理工大学 Dalian University of Technology

■ 第一层



$$z_1^{[1]} = w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} = (w_{11}^{[1]}, w_{21}^{[1]}, w_{31}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_1^{[1]} = \mathbf{w}_1^{[1]T}\mathbf{x}^{[0]} + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} = (w_{12}^{[1]}, w_{22}^{[1]}, w_{32}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_2^{[1]} = \mathbf{w}_2^{[1]T}\mathbf{x}^{[0]} + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} = (w_{13}^{[1]}, w_{23}^{[1]}, w_{33}^{[1]})\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + b_3^{[1]} = \mathbf{w}_3^{[1]T}\mathbf{x}^{[0]} + b_3^{[1]}$$

$$a_3^{[1]} = \sigma(z_3^{[1]})$$

$$\begin{pmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \end{pmatrix} = \begin{pmatrix} w_{11}^{[1]} & w_{21}^{[1]} & w_{31}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} & w_{32}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} & w_{33}^{[1]} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{pmatrix}$$

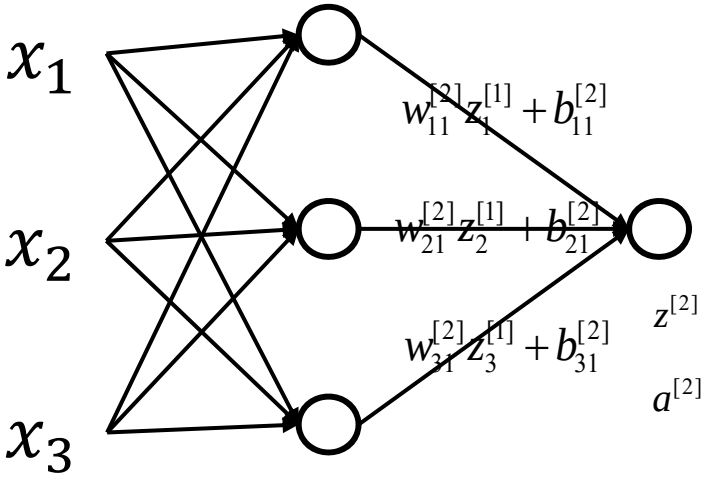
$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]T}\mathbf{z}^{[0]} + \mathbf{b}^{[1]}$$

$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

前馈神经网络

大连理工大学 Dalian University of Technology

■ 第二层



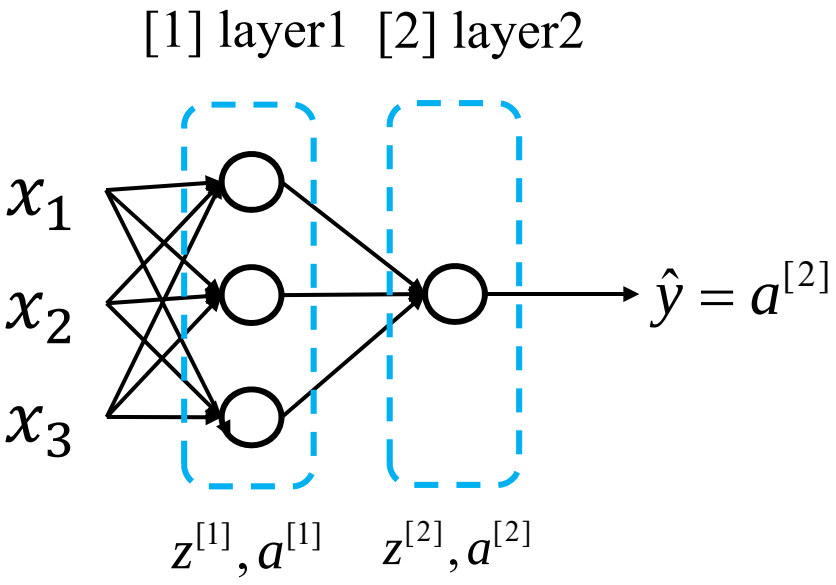
$$z^{[2]} = w_{11}^{[2]}z_1^{[1]} + w_{21}^{[2]}z_2^{[1]} + w_{31}^{[2]}z_3^{[1]} + b_1^{[2]} = (w_{11}^{[2]}, w_{21}^{[2]}, w_{31}^{[2]}) \begin{pmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \end{pmatrix} + b_1^{[2]} = \mathbf{w}_1^{[2]}\mathbf{z}^{[2]} + b_1^{[2]}$$

$$a^{[2]} = \sigma(z^{[2]})$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]T}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}$$

前馈神经网络

■ 第L层



$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]T}\mathbf{z}^{[0]} + \mathbf{b}^{[1]} \quad \text{第一层}$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]T}\mathbf{a}^{[1]} + \mathbf{b}^{[2]} \quad \text{第二层}$$

....

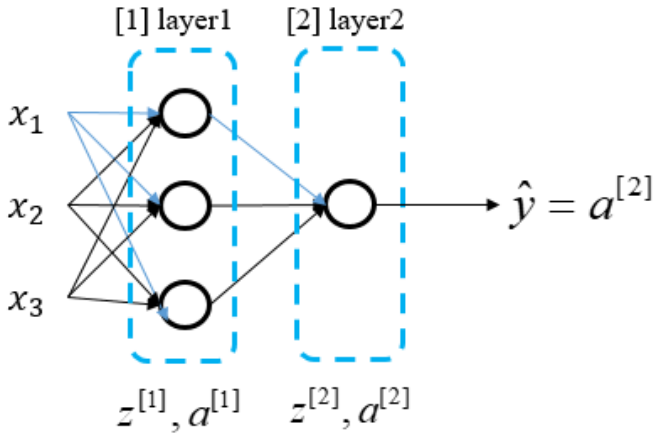
$$\mathbf{z}^{[L]} = \mathbf{W}^{[L]T}\mathbf{a}^{[L-1]} + \mathbf{b}^{[L]} \quad \text{第L层}$$

前馈神经网络

大连理工大学 Dalian University of Technology

■ 第一层

$$\left.\begin{matrix}\mathbf{x}\\ \mathbf{W}^{[1]}\\ \mathbf{b}^{[1]}\end{matrix}\right\}\rightarrow \mathbf{z}^{[1]}=\mathbf{W}^{[1]T}\mathbf{x}+\mathbf{b}^{[1]}\rightarrow \mathbf{a}^{[1]}=\sigma(\mathbf{z}^{[1]})$$



■ 第二层

$$\left.\begin{matrix}\mathbf{a}^{[1]}=\sigma(\mathbf{z}^{[1]})\\ \mathbf{W}^{[2]}\\ \mathbf{b}^{[2]}\end{matrix}\right\}\rightarrow \mathbf{z}^{[2]}=\mathbf{W}^{[2]T}\mathbf{a}^{[1]}+\mathbf{b}^{[2]}\rightarrow \mathbf{a}^{[2]}=\sigma(\mathbf{z}^{[2]})\rightarrow L(\mathbf{a}^{[2]},y)$$

前馈神经网络

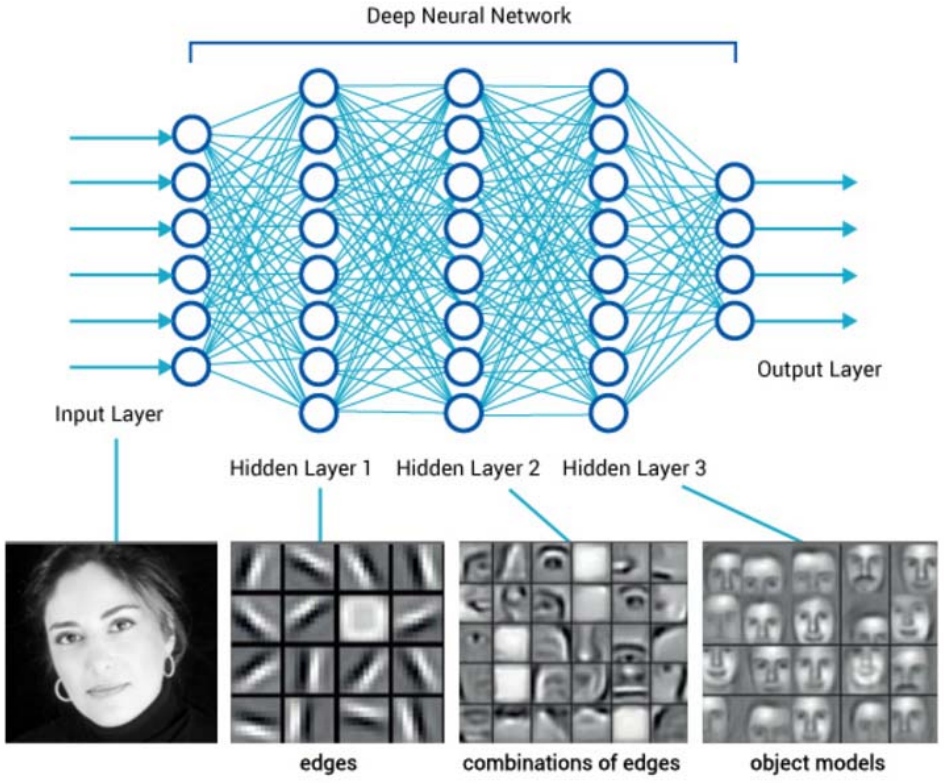
大连理工大学 Dalian University of Technology

■ 前馈计算—信息向前传递

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)},$$

$$\boldsymbol{a}^{(l)} = f_l(\boldsymbol{z}^{(l)}).$$

$$\boldsymbol{x} = \boldsymbol{a}^{(0)} \to \boldsymbol{z}^{(1)} \to \boldsymbol{a}^{(1)} \to \boldsymbol{z}^{(2)} \to \cdots \to \boldsymbol{a}^{(L-1)} \to \boldsymbol{z}^{(L)} \to \boldsymbol{a}^{(L)} = \phi(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}))$$



前馈神经网络

大连理工大学 Dalian University of Technology

■ 通用近似定理

定理 4.1 – 通用近似定理（Universal Approximation Theorem）[Cybenko, 1989, Hornik et al., 1989]： 令 $\varphi(\cdot)$ 是一个非常数、有界、单调递增的连续函数，$\mathcal{I}_d$ 是一个 d 维的单位超立方体 $[0,1]^d$，$C(\mathcal{I}_d)$ 是定义在 $\mathcal{I}_d$ 上的连续函数集合。对于任何一个函数 $f \in C(\mathcal{I}_d)$，存在一个整数 m，和一组实数 $v_i, b_i \in \mathbb{R}$ 以及实数向量 $\mathbf{w}_i \in \mathbb{R}^d$，$i = 1, \cdots, m$，以至于我们可以定义函数

$$F(\mathbf{x}) = \sum_{i=1}^{m} v_i \varphi(\mathbf{w}_i^{\mathrm{T}} \mathbf{x} + b_i), \tag{4.33}$$

作为函数 f 的近似实现，即

$$|F(\mathbf{x}) - f(\mathbf{x})| < \epsilon, \forall \mathbf{x} \in \mathcal{I}_d. \tag{4.34}$$

其中 $\epsilon > 0$ 是一个很小的正数。

根据通用近似定理，对于具有线性输出层和至少一个使用“挤压”性质的激活函数的隐藏层组成的前馈神经网络，只要其隐藏层神经元的数量足够，它可以以任意的精度来近似任何从一个定义在实数空间中的有界闭集函数。

前馈神经网络

大连理工大学 Dalian University of Technology

■ 通用近似定理

- 神经网络可以作为一个“万能”函数来使用，可以用来进行复杂的特征转换，或逼近一个复杂的条件分布

$$\hat{y} = g(\varphi(\mathbf{x}), \theta)$$

神经网络

前馈神经网络

大连理工大学 Dalian University of Technology

■ 通用近似定理

- 神经网络可以作为一个“万能”函数来使用，可以用来进行复杂的特征转换，或逼近一个复杂的条件分布

$$\hat{y} = g(\varphi(\mathbf{x}), \theta)$$

分类器

- 如果 $g(\cdot)$ 为Logistic回归，那么Logistic回归分类器可以看成神经网络的最后一层

前馈神经网络

大连理工大学 Dalian University of Technology

■ 分类问题

- 如果使用Softmax回归分类器，相当于网络最后一层设置C个神经元，其输出经过Softmax函数进行归一化后可以作为每个类的条件概率。

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(L)})$$

- 采用交叉熵损失函数，对于样本(x,y)，其损失函数为

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbf{y}^{\mathrm{T}} \log \hat{\mathbf{y}}$$

大连理工大学 Dalian University of Technology

■ 交叉熵损失函数

- **信息量**：信息是用来**消除**随机**不确定性**的东西。信息量的大小与信息发生的概率成反比。概率越大，信息量越小。概率越小，信息量越大。设某一事件发生的概率为P(x)，其信息量表示为

$$I(x) = -\log(P(x))$$

- **信息熵**：所有信息量的期望 $\mathbf{X} = (x_1, x_2, \cdots, x_n)$

$$H(\mathbf{X}) = \sum_{i=1}^{n} P(x_i)\log(P(x_i))$$

- **相对熵 （KL散度）**：

$$D_{KL}(p \| q) = \sum_{i=1}^{n} p(x_i)\log\left(\frac{p(x_i)}{q(x_i)}\right)$$

真实分布　　预测分布

前馈神经网络

大连理工大学 Dalian University of Technology

■ 交叉熵损失函数

- **交叉熵：**交叉熵等于KL散度加上一个常量（信息熵），且公式相比KL散度更加容易计算，所以在机器学习中常常使用交叉熵损失函数来计算loss

$$D_{KL}(p\|q)=\sum_{i=1}^{n}p(x_i)\log(\frac{p(x_i)}{q(x_i)})$$

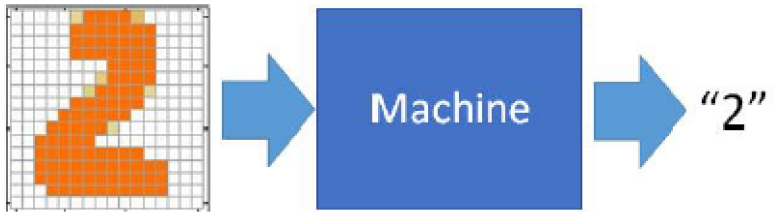
$$=\sum_{i=1}^{n}p(x_i)\log(p(x_i))-\sum_{i=1}^{n}p(x_i)\log(q(x_i))$$

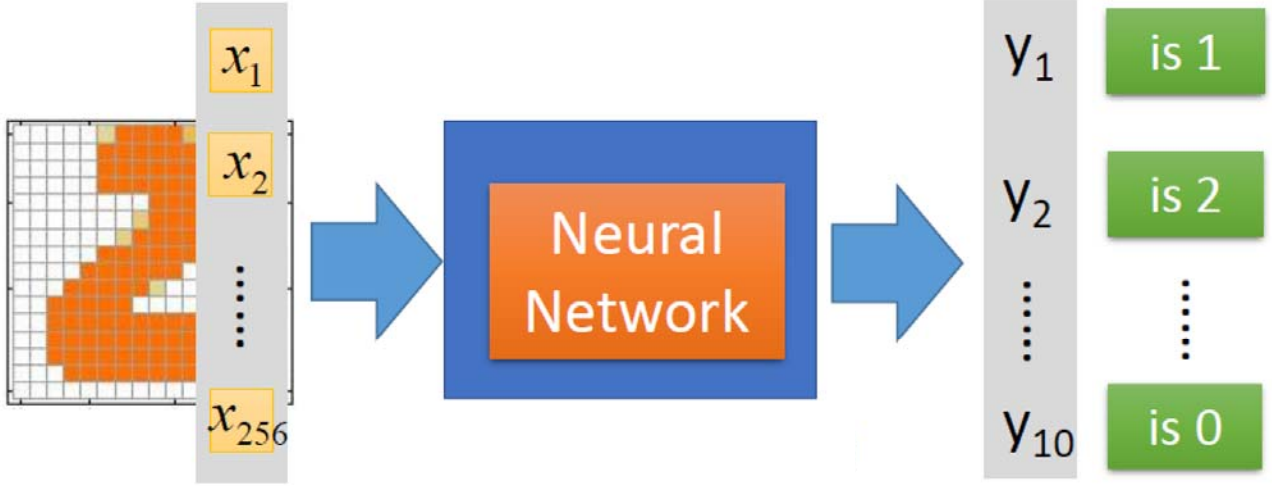
$$=\boxed{-H(p(x))}+\boxed{[-\sum_{i=1}^{n}p(x_i)\log(q(x_i))]}$$

信息熵　　交叉熵

前馈神经网络

■ 例：手写数字识别





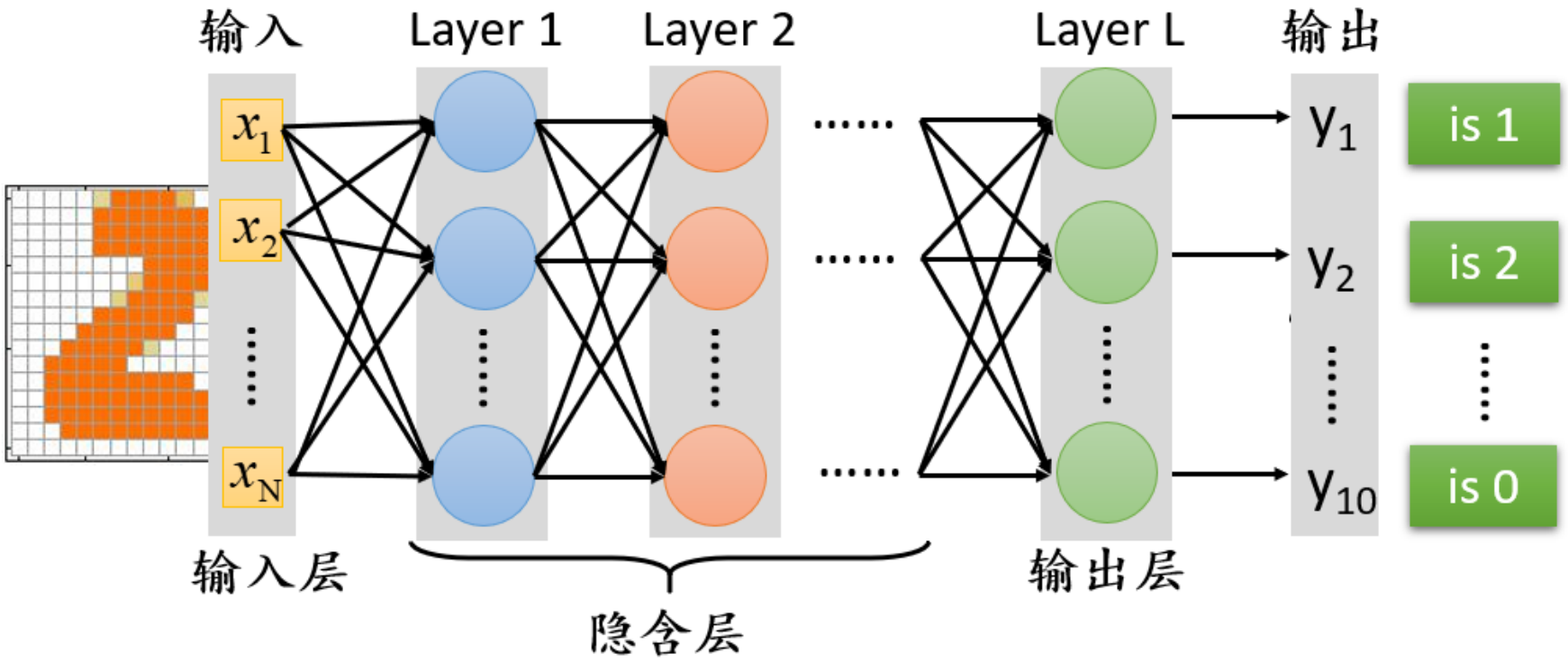
输入：256维向量

输出：10维向量

前馈神经网络

大连理工大学 Dalian University of Technology

■ 例：手写数字识别

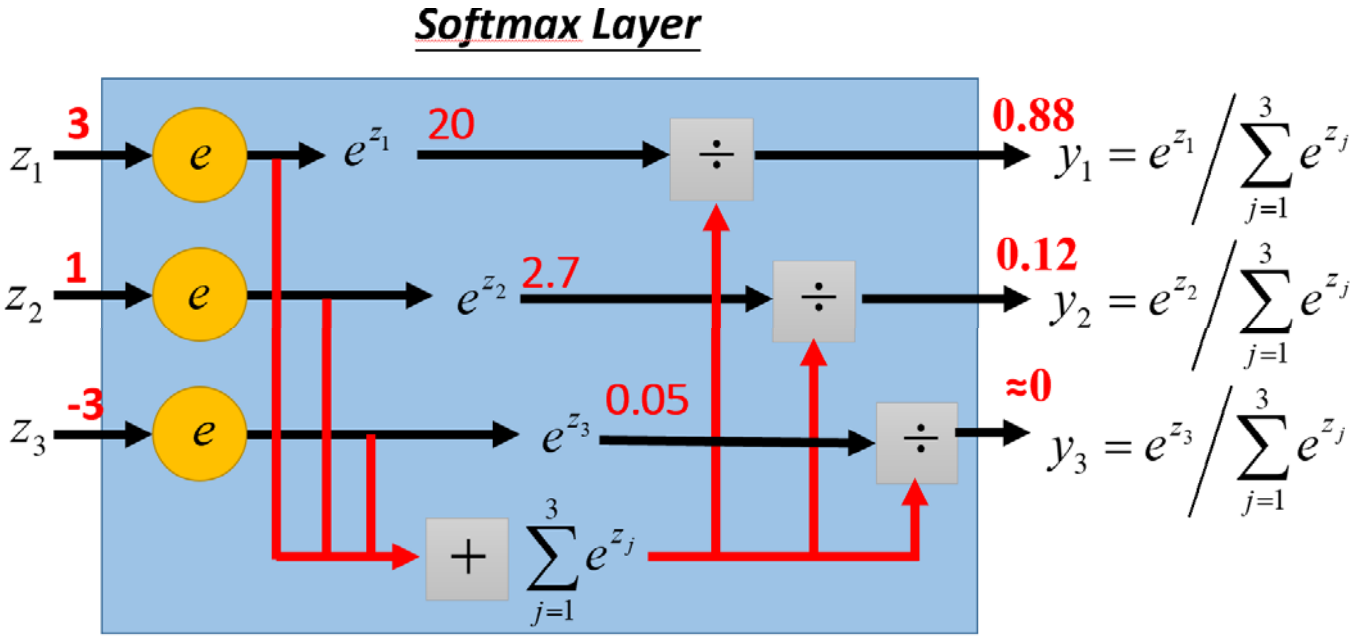


对网络结构进行合理设计，使其函数集合中能够有一个较好的函数

大连理工大学 Dalian University of Technology

■ 最后的输出层—Softmax Layer

- 图像分类：将输入图像的信息转换成其属于各个类别的概率



Probability:

- $1 > y_i > 0$
- $\sum_i y_i = 1$

大连理工大学
Dalian University Of Technology

PART THREE

梯度反向传播

Gradient Back Propagation

梯度反向传播

大连理工大学 Dalian University of Technology

■ 参数学习 (Parameter Learning)

- 给定训练集为 $D = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ ，将每个样本 $\mathbf{x}^{(n)}$ 输入给前馈神经网络，得到网络输出为 $\hat{y}^{(n)}$ ，骑在数据集D上的结构化风险函数为：

$$\mathcal{R}(W, \mathbf{b}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)}) + \frac{1}{2} \lambda \|W\|_F^2$$

- 梯度下降：

$$W^{(l)} \leftarrow W^{(l)} - \alpha \frac{\partial \mathcal{R}(W, \mathbf{b})}{\partial W^{(l)}}$$

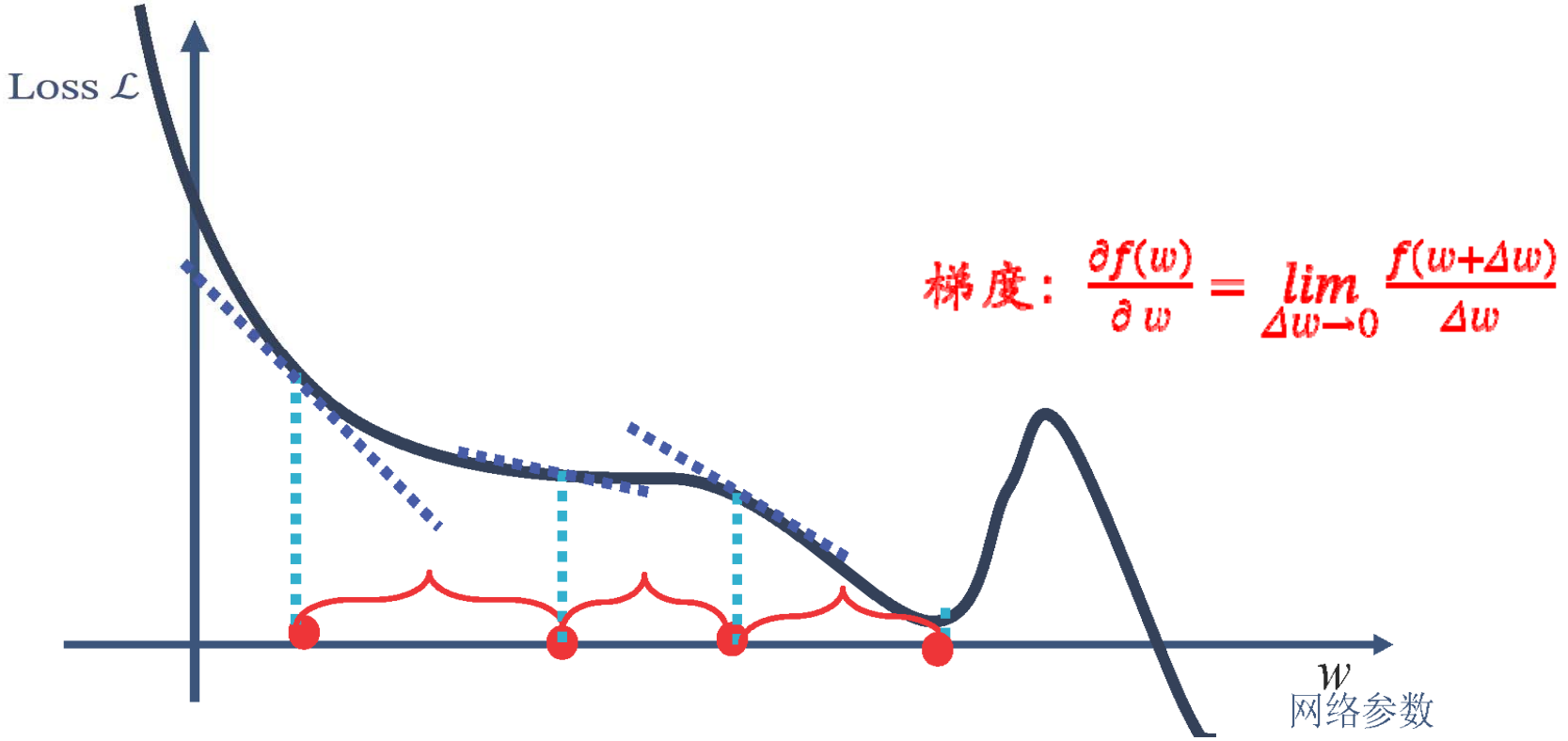
$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \alpha \frac{\partial \mathcal{R}(W, \mathbf{b})}{\partial \mathbf{b}^{(l)}}$$

梯度反向传播

大连理工大学 Dalian University of Technology

■ 梯度下降

□ 梯度下降类似盲人下山



梯度反向传播

大连理工大学 Dalian University of Technology

■ 梯度下降

- 神经网络是一个复杂的复合函数(链式法则)

$$y = f^5(f^4(f^3(f^2(f^1(x))))) \rightarrow \frac{\partial y}{\partial x} = \frac{\partial f^5}{\partial f^4}\frac{\partial f^4}{\partial f^3}\frac{\partial f^3}{\partial f^2}\frac{\partial f^2}{\partial f^1}\frac{\partial f^1}{\partial x}$$

- 梯度反向传播算法(Back Propagation)：根据前馈网络的特点而设计的高效方法

- 更加通用的计算方法：自动微分(Automatic Differentiation)

梯度反向传播

大连理工大学 Dalian University of Technology

■ 链式法则

□ 链式法则(Chain Rule)是在微积分中求复合函数导数的一种常用方法。

（1）若 $x \in \mathbb{R}, \boldsymbol{u} = u(x) \in \mathbb{R}^s, \boldsymbol{g} = g(\boldsymbol{u}) \in \mathbb{R}^t$，则

$$\frac{\partial \boldsymbol{g}}{\partial x} = \frac{\partial \boldsymbol{u}}{\partial x}\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} \quad \in \mathbb{R}^{1\times t}.$$

（2）若 $\boldsymbol{x} \in \mathbb{R}^p, \boldsymbol{y} = g(\boldsymbol{x}) \in \mathbb{R}^s, \boldsymbol{z} = f(\boldsymbol{y}) \in \mathbb{R}^t$，则

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} \quad \in \mathbb{R}^{p\times t}.$$

（3）若 $X \in \mathbb{R}^{p\times q}$ 为矩阵，$\boldsymbol{y} = g(X) \in \mathbb{R}^s, z = f(\boldsymbol{y}) \in \mathbb{R}$，则

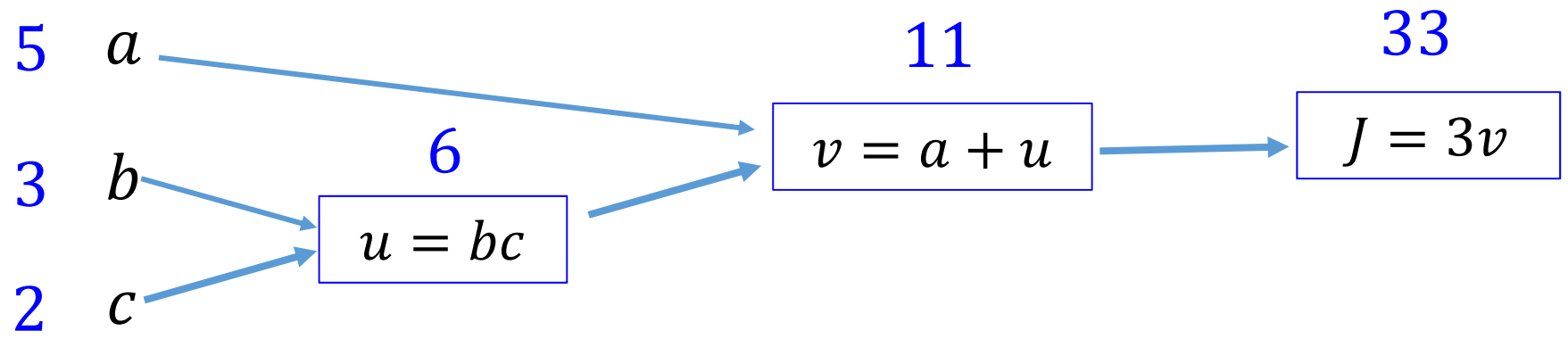
$$\frac{\partial z}{\partial X_{ij}} = \frac{\partial \boldsymbol{y}}{\partial X_{ij}}\frac{\partial z}{\partial \boldsymbol{y}} \quad \in \mathbb{R}.$$

梯度反向传播

■ 梯度下降—计算图

$$J(a,b,c) = 3\,(a + \underbrace{bc}_{u}) \quad = 3(5 + 3 \times 2) = 33$$

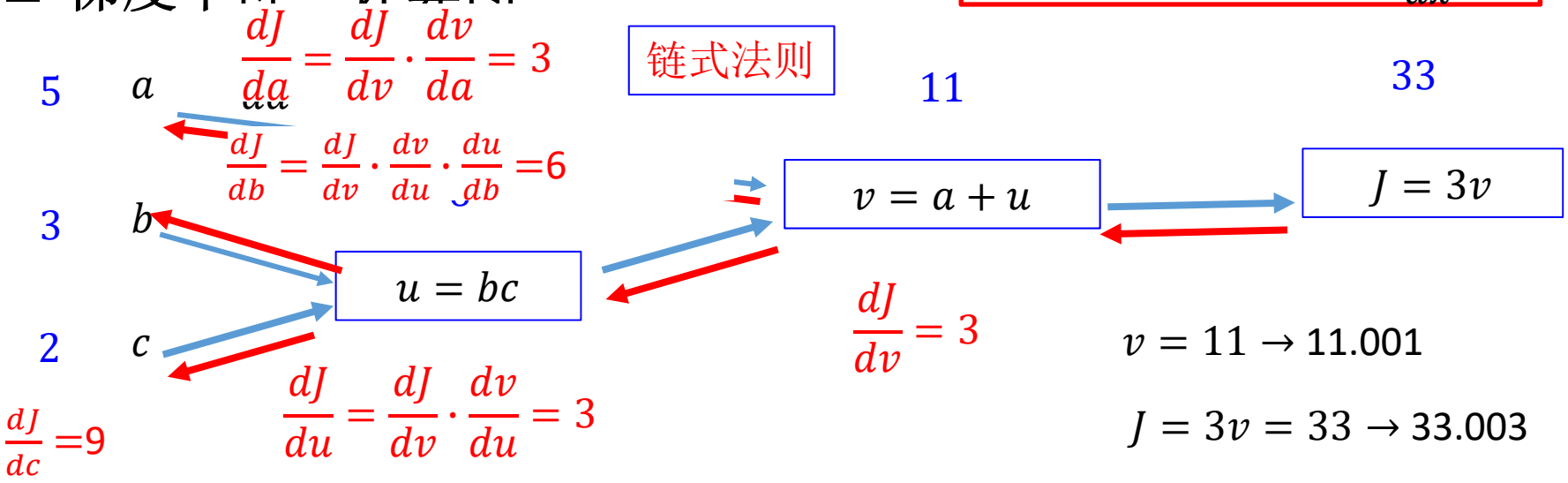
$$u = bc$$
$$v = a + u$$
$$J = 3v$$



梯度反向传播

$$y = f(x) = 3x, f'(x) = \frac{dy}{dx} = 3$$

■ 梯度下降—计算图



$v = 11 \rightarrow 11.001$

$J = 3v = 33 \rightarrow 33.003$

$\frac{dJ}{dv} = 3$

$b = 3 \rightarrow 3.001$

$u = 6 \rightarrow bc = 6.002$

$v = 11 \rightarrow 11.002$

$J = 3v = 33 \rightarrow 33.006$

$\frac{du}{db} = 2, \frac{dv}{du} = 1, \frac{dJ}{db} = 6$

$u = 6 \rightarrow 6.001$

$v = 11 \rightarrow 11.001$

$J = 3v = 33 \rightarrow 33.003$

$\frac{dv}{du} = 1, \quad \frac{dJ}{dv} = 3$

$a = 5 \rightarrow 5.001$

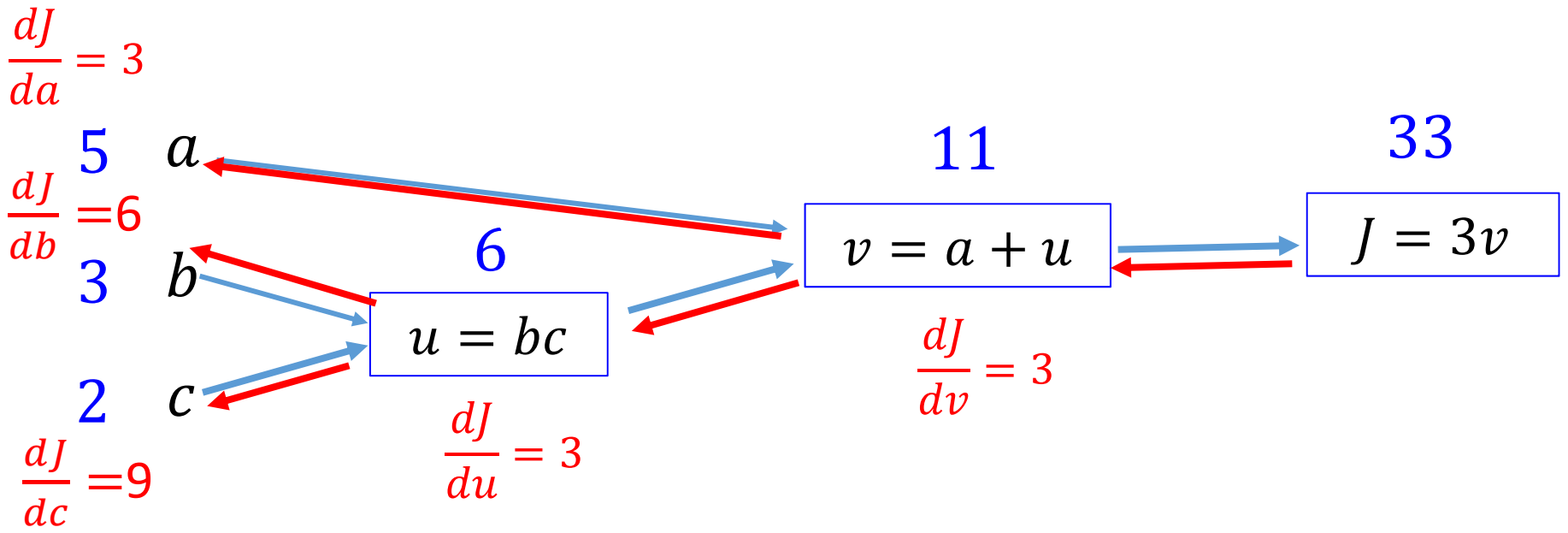
$v = 11 \rightarrow 11.001$

$J = 3v = 33 \rightarrow 33.003$

$\frac{dv}{da} = 1, \quad \frac{dJ}{da} = 3$

梯度反向传播

■ 梯度下降—计算图



梯度反向传播

大连理工大学 Dalian University of Technology

■ 梯度下降—计算图

□逻辑回归算法：$\hat{y} = f(w^T x + b)$，　其中$z = w^T x + b, f(z) = \frac{1}{1+e^{-z}}$

□损失函数：

$$L(\hat{y}^{(i)}, y^{(i)}) = -y^{(i)} log(\hat{y}^{(i)}) - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

□代价函数:

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

□梯度下降法参数更新：

$$w := w - \eta \frac{\partial L(w,b)}{\partial w}, b := b - \eta \frac{\partial L(w,b)}{\partial b}$$

梯度反向传播

大连理工大学 Dalian University of Technology

■ 梯度下降—计算图

□ 假设单个样本, 样本只有两个特征x_1和x_2；参数w_1、w_2和b：

$$z = w_1 x_1 + w_2 x_2 + b$$

$$\hat{y} = f(z)$$

□ 损失函数：

$$L(\hat{y}, y) = -y log(\hat{y}) - (1 - y)\log(1 - \hat{y})$$

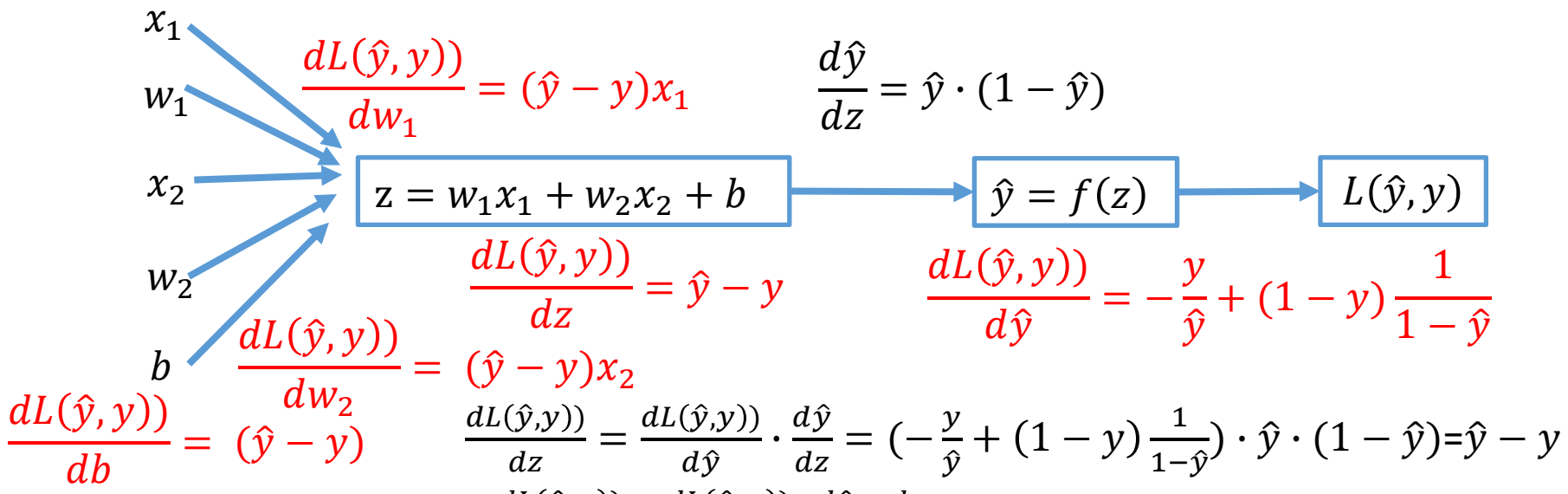
梯度反向传播

大连理工大学 Dalian University of Technology

■ 梯度下降—计算图

$$z = w_1x_1 + w_2x_2 + b$$
$$\hat{y} = f(z) = \frac{1}{1+e^{-z}}$$
$$L(\hat{y}, y) = -ylog(\hat{y}) - (1-y)\log(1-\hat{y})$$

$$f'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = f(z)(1-fz)$$



$$\frac{dL(\hat{y},y))}{dz} = \frac{dL(\hat{y},y))}{d\hat{y}} \cdot \frac{d\hat{y}}{dz} = (-\frac{y}{\hat{y}} + (1-y)\frac{1}{1-\hat{y}}) \cdot \hat{y}\cdot(1-\hat{y}) = \hat{y}-y$$

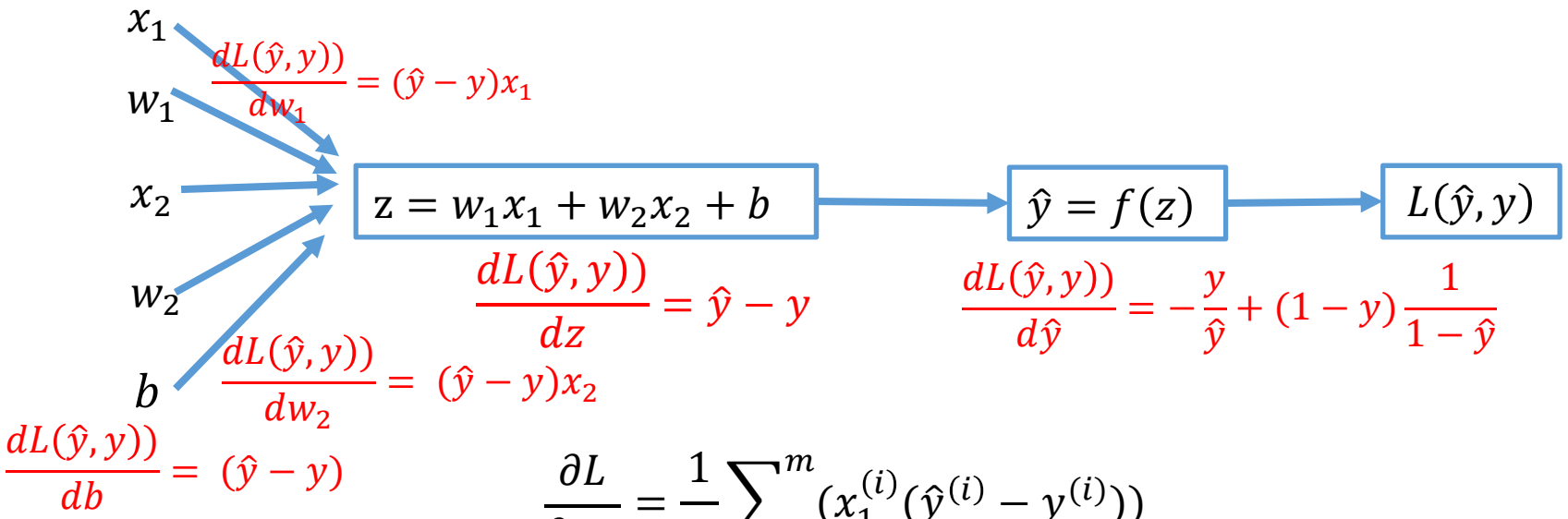
$$\frac{dL(\hat{y},y))}{dw_1} = \frac{dL(\hat{y},y))}{d\hat{y}} \cdot \frac{d\hat{y}}{dz} \cdot \frac{dz}{dw_1} = (\hat{y}-y)x_1$$

$$\frac{dL(\hat{y},y))}{dw_2} = \frac{dL(\hat{y},y))}{d\hat{y}} \cdot \frac{d\hat{y}}{dz} \cdot \frac{dz}{dw_2} = (\hat{y}-y)x_2$$

梯度反向传播

大连理工大学 Dalian University of Technology

■ 梯度下降—计算图



对于m个样本：

$$\frac{\partial L}{\partial w_1} = \frac{1}{m}\sum_i^m (x_1^{(i)}(\hat{y}^{(i)} - y^{(i)}))$$

$$\frac{\partial L}{\partial w_2} = \frac{1}{m}\sum_i^m (x_2^{(i)}(\hat{y}^{(i)} - y^{(i)}))$$

$$\frac{\partial L}{\partial b} = \frac{1}{m}\sum_i^m (\hat{y}^{(i)} - y^{(i)})$$

更新：$w_1 = w_1 - \eta\frac{\partial L}{\partial w_1}$, $w_2 = w_2 - \eta\frac{\partial L}{\partial w_2}$, $b = b - \eta\frac{\partial L}{\partial b}$

梯度反向传播

- **反向传播 (Back Propagation)**

$$\mathbf{z}^{(l)} = W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

梯度反向传播

大连理工大学 Dalian University of Technology

■ 反向传播 (Back Propagation)

$$\mathbf{z}^{(l)} = W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

误差项

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

$$\delta^{(l)} = \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \in \mathbb{R}^{m^{(l)}}$$

梯度反向传播

大连理工大学 Dalian University of Technology

■ 反向传播 (Back Propagation)

$$\mathbf{z}^{(l)} = W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

$$\frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

误差项

$$\delta^{(l)} = \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \in \mathbb{R}^{m^{(l)}}$$

权重梯度

$$\begin{aligned}
\frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} &= \left[\frac{\partial z_1^{(l)}}{\partial w_{ij}^{(l)}}, \cdots, \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}}, \cdots, \frac{\partial z_{m^{(l)}}^{(l)}}{\partial w_{ij}^{(l)}}\right] \\
&= \left[0, \cdots, \frac{\partial(\mathbf{w}_{i:}^{(l)}\mathbf{a}^{(l-1)} + b_i^{(l)})}{\partial w_{ij}^{(l)}}, \cdots, 0\right] \\
&= \left[0, \cdots, a_j^{(l-1)}, \cdots, 0\right] \\
&\triangleq \mathbb{I}_i(a_j^{(l-1)}) \quad \in \mathbb{R}^{m^{(l)}},
\end{aligned}$$

梯度反向传播

大连理工大学 Dalian University of Technology

■ 反向传播 (Back Propagation)

$$\mathbf{z}^{(l)} = W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

$$\frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

误差项

$$\delta^{(l)} = \frac{\partial \mathcal{L}(\mathbf{y},\hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \in \mathbb{R}^{m^{(l)}}$$

权重梯度

$$\begin{aligned}\frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} &= \left[\frac{\partial z_1^{(l)}}{\partial w_{ij}^{(l)}}, \cdots, \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}}, \cdots, \frac{\partial z_{m^{(l)}}^{(l)}}{\partial w_{ij}^{(l)}}\right] \\ &= \left[0, \cdots, \frac{\partial(\mathbf{w}_{i:}^{(l)}\mathbf{a}^{(l-1)} + b_i^{(l)})}{\partial w_{ij}^{(l)}}, \cdots, 0\right] \\ &= \left[0, \cdots, a_j^{(l-1)}, \cdots, 0\right] \\ &\triangleq \mathbb{I}_i(a_j^{(l-1)}) \quad \in \mathbb{R}^{m^{(l)}},\end{aligned}$$

偏置梯度

$$\frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} = \mathbf{I}_{m^{(l)}} \quad \in \mathbb{R}^{m^{(l)} \times m^{(l)}}$$

梯度反向传播

大连理工大学 Dalian University of Technology

■ 反向传播 (Back Propagation)

□ 神经网络结构

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)})$$

$$\mathbf{z}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{a}^{(l)} + \mathbf{b}^{(l+1)}$$

□ 逐层反向传播

$$\frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} = \frac{\partial f_l(\mathbf{z}^{(l)})}{\partial \mathbf{z}^{(l)}} = \mathrm{diag}(f_l'(\mathbf{z}^{(l)}))$$

$$\frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} = (W^{(l+1)})^{\mathrm{T}}$$

$$\begin{aligned}
\delta^{(l)} &\triangleq \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \\
&= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l+1)}} \\
&= \mathrm{diag}(f_l'(\mathbf{z}^{(l)})) \cdot (W^{(l+1)})^{\mathrm{T}} \cdot \delta^{(l+1)} \\
&= f_l'(\mathbf{z}^{(l)}) \odot \left((W^{(l+1)})^{\mathrm{T}} \delta^{(l+1)}\right),
\end{aligned}$$

梯度反向传播

大连理工大学 Dalian University of Technology

■ 反向传播 (Back Propagation)

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} = \mathbb{I}_i(a_j^{(l-1)})\delta^{(l)} = \delta_i^{(l)} a_j^{(l-1)}$$

- 进一步，$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ 关于第 l 层权重$\mathbf{W}^{(l)}$的梯度为

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^{\mathrm{T}}$$

- 同理，$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ 关于第 l 层偏置$\boldsymbol{b}^{(l)}$的梯度为

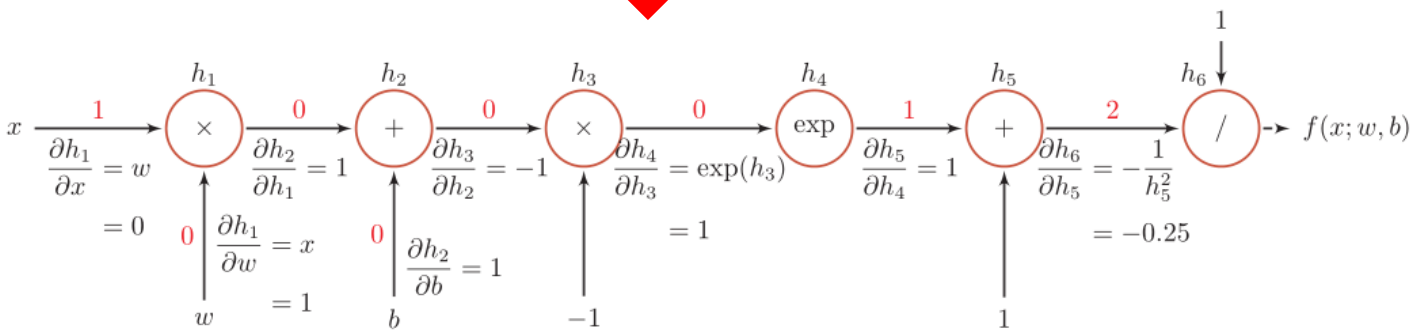
$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$

梯度反向传播

■ 自动微分 (Automatic Differentiation)

自动微分是利用链式法则来自动计算一个复合函数的梯度。

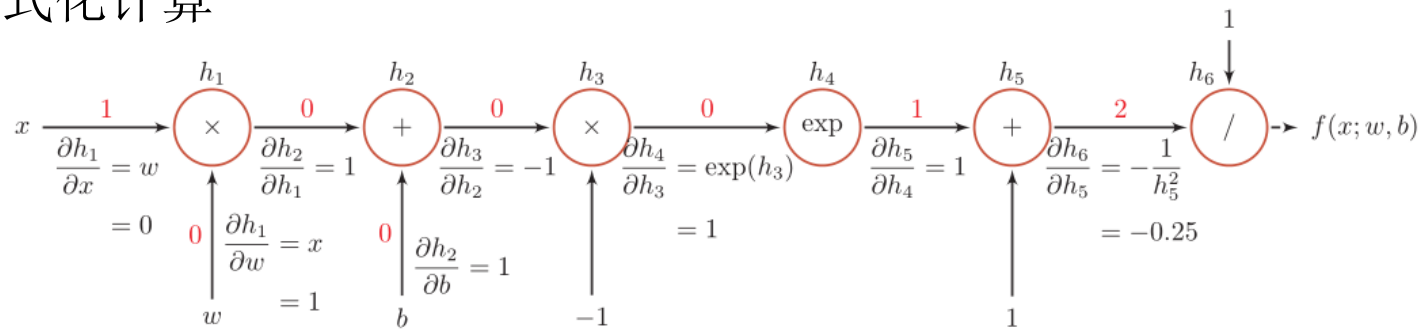
$$f(x;w,b)=\frac{1}{\exp(-(wx+b))+1}.$$



梯度反向传播

■ 自动微分 (Automatic Differentiation)

□ 形式化计算



函数	导数	
$h_1 = x \times w$	$\frac{\partial h_1}{\partial w} = x$	$\frac{\partial h_1}{\partial x} = w$
$h_2 = h_1 + b$	$\frac{\partial h_2}{\partial h_1} = 1$	$\frac{\partial h_2}{\partial b} = 1$
$h_3 = h_2 \times -1$	$\frac{\partial h_3}{\partial h_2} = -1$	
$h_4 = \exp(h_3)$	$\frac{\partial h_4}{\partial h_3} = \exp(h_3)$	
$h_5 = h_4 + 1$	$\frac{\partial h_5}{\partial h_4} = 1$	
$h_6 = 1/h_5$	$\frac{\partial h_6}{\partial h_5} = -\frac{1}{h_5^2}$	

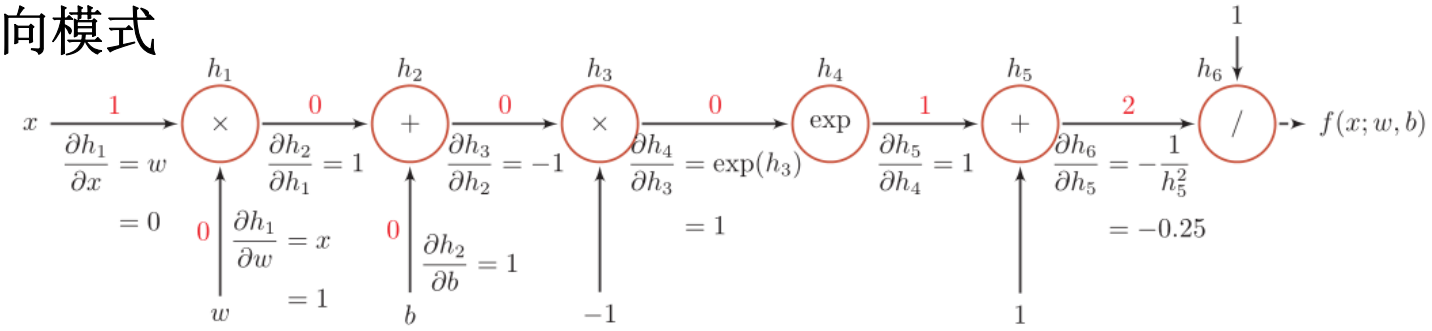
当x = 1,w = 0,b = 0时，可以得到

$$\frac{\partial f(x;w,b)}{\partial w}|_{x=1,w=0,b=0} = \frac{\partial f(x;w,b)}{\partial h_6}\frac{\partial h_6}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial w}$$
$$= 1 \times -0.25 \times 1 \times 1 \times -1 \times 1 \times 1$$
$$= 0.25.$$

梯度反向传播

■ 自动微分 (Automatic Differentiation)

□ 前向模式



□ 反向模式(与反向传播的计算梯度的方式相同)

前向计算每一层的状态和激活值，直到最后一层

反向计算每一层的参数的偏导数

更新参数

□ 如果函数和参数之间有多条路径，可以将这多条路径上的导数再进行相加，得到最终的梯度。

梯度反向传播

大连理工大学 Dalian University of Technology

■ 自动微分 (Automatic Differentiation)

□ 静态计算图

在编译时构建计算图，计算图构建好之后在程序运行时不能改变。代表性工具箱Theano和Tensorflow

□ 动态计算图

在程序运行时动态构建，可自动根据函数的结构调整计算图。代表性工具箱DyNet，Chainer和PyTorch

□ 优缺点

静态计算图在构建时可以进行优化，并行能力强，但灵活性比较差低。动态计算图则不容易优化，当不同输入的网络结构不一致时，难以并行计算，但是灵活性比较高。

大连理工大学
Dalian University Of Technology

PART FOUR 04

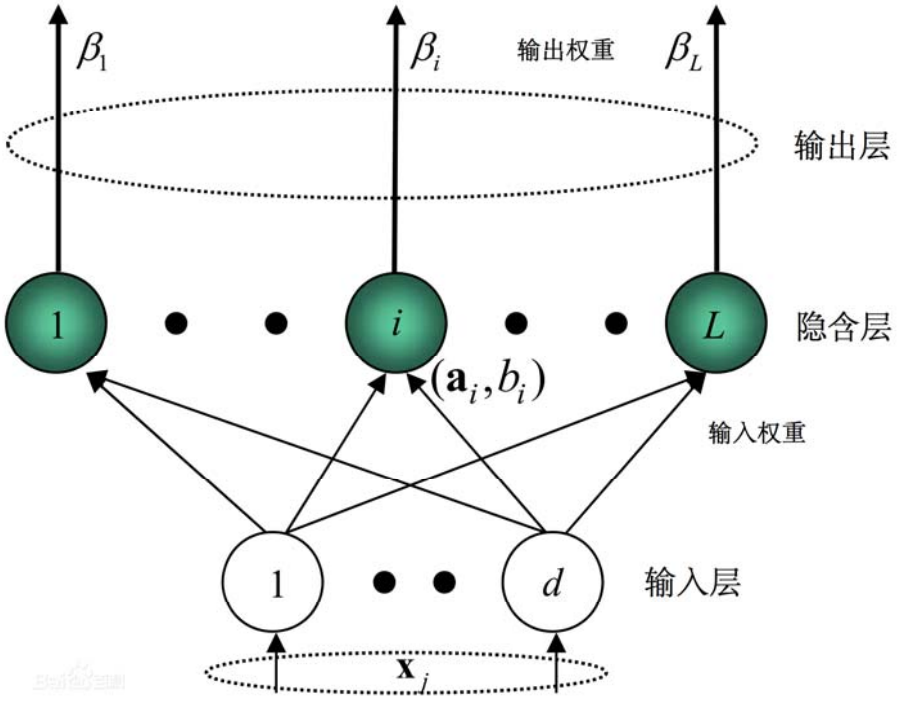
闭形式解

Close-form Solution

闭形式解

■ 极端学习机 (Extreme Learning Machine)

快速学习算法，对于单隐层神经网络(三层)，ELM可以随机初始化输入权重和偏置，并利用最小二乘得到相应的输出权重。



闭形式解

随机初始化

闭形式解

■ 极端学习机 (Extreme Learning Machine)

- 在一个 ELM 的单隐层结构中，第 i 个隐藏节点的输出为：

$$h_i(\mathbf{x}) = G(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x})$$

- 整个隐藏层的输出映射为：

$$\mathbf{h}(\mathbf{x}) = [G(h_1(\mathbf{x})), G(h_2(\mathbf{x})), \cdots, G(h_L(\mathbf{x}))]$$

- 给定 N 个训练样本，ELM 的隐藏层输出矩阵 H 给出为：

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_N) \end{bmatrix}$$

- 具有 L 个隐藏节点的 ELM 结构的输出为：

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i \mathbf{h}_i(\mathbf{x})$$

闭形式解

大连理工大学 Dalian University of Technology

■ 极端学习机 (Extreme Learning Machine)

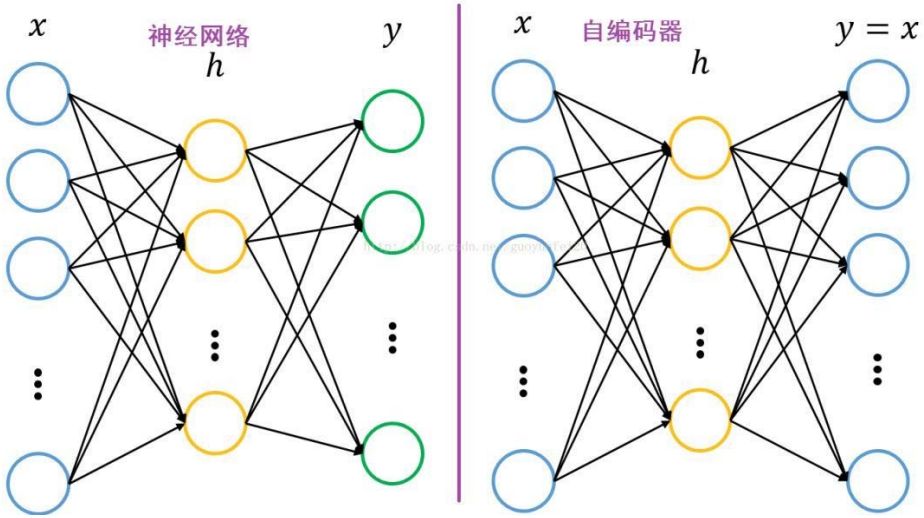
- 整个隐藏层的输出为： $\mathbf{H}\beta$
- 目标矩阵 T (标签矩阵)为： $\mathbf{T}=[\mathbf{t}_1,\mathbf{t}_2,\cdots,\mathbf{t}_N]^T$
- 目标函数为： $Minimize:\|\beta\|_p+C\|\mathbf{H}\beta-\mathbf{T}\|_q$
- 求解矩阵H的Moore-Penrose广义逆： $\hat{\beta}=\mathbf{H}^{+}\mathbf{T}$

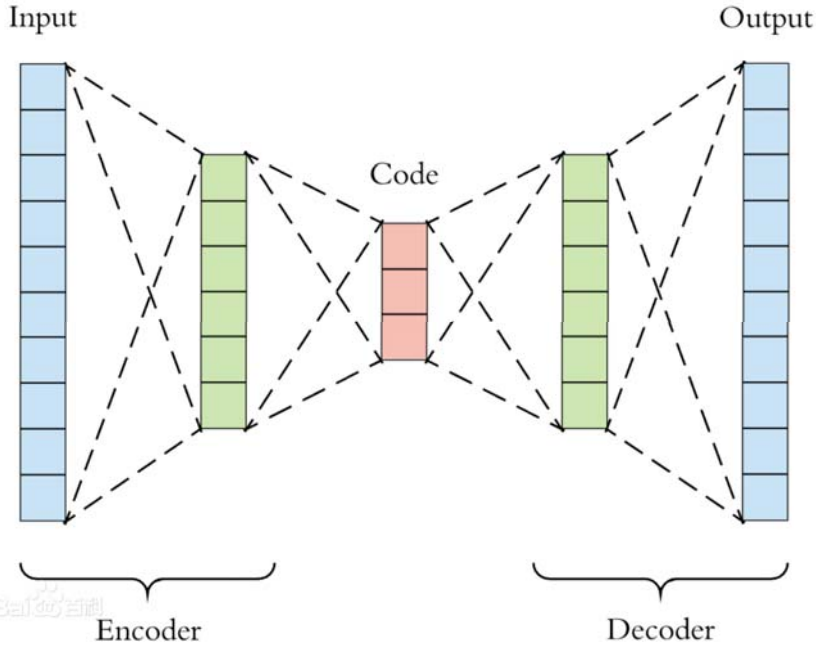
闭形式解！

闭形式解

■ 自编码器 (Auto-Encoder)

- 自编码器是一类在半监督学习和无监督学习中使用的人工神经网络，其功能是通过将输入信息作为学习目标，对输入信息进行表征学习
- 自编码器包含编码器(Encoder)和解码器(Decoder)两部分





闭形式解

■ 自编码器 (Auto-Encoder)

- 按学习范式，自编码器可以被分为收缩自编码器、正则自编码器和变分自编码器，其中前两者是判别模型、后者是生成模型
- 按构建类型，自编码器可以是前馈结构或递归结构的神经网络
- 自编码器具有一般意义上表征学习算法的功能，被应用于降维和异常值检测
- 包含卷积层结构的自编码器可被应用于计算机视觉问题，包括图像降噪、图像风格迁移等

闭形式解

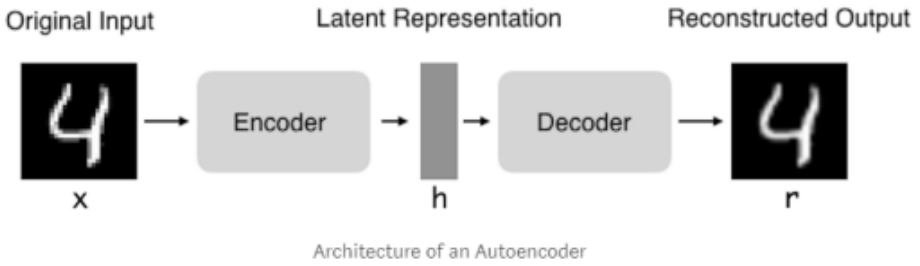
■ 自编码器 (Auto-Encoder)

- 自编码器是一个输入和输出相同的神经网络,给定输入空间和特征空间，自编码器求解两者的映射使输入特征的重建误差达成最小:

$$y = h(x)$$
$$r = f(y) = f(h(x))$$

重建误差最小

- 求解完成后，由编码器输出的隐含层特征，即“编码特征”可视为输入数据的表征



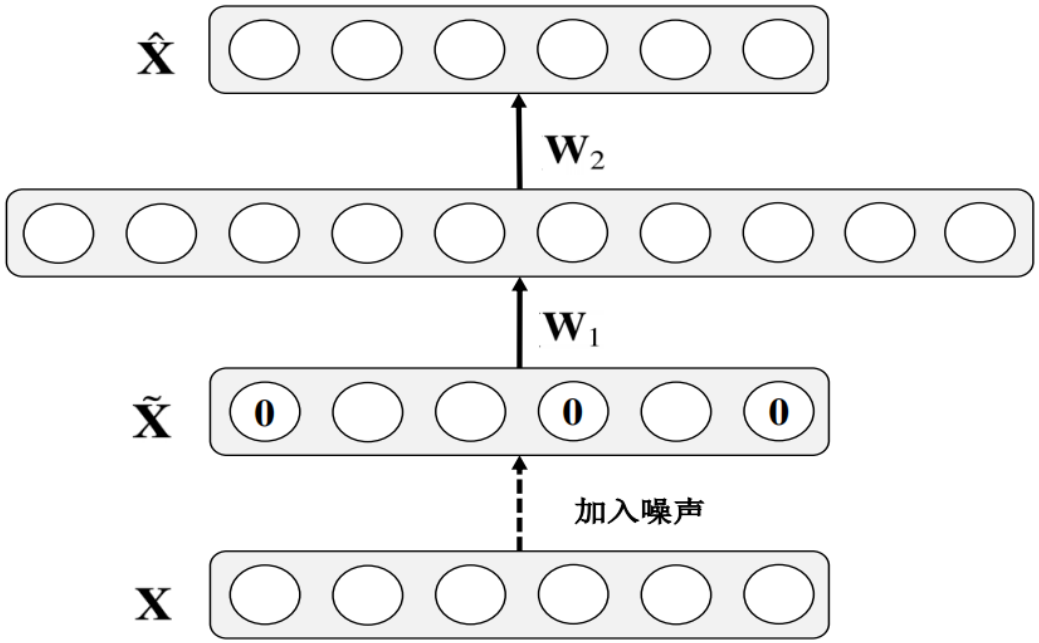
Architecture of an Autoencoder

- 当自编码器的编码、解码能力很强时，可以确保对训练样本的完美重建，但却难以发现数据的内在分布和结构规律

闭形式解

■ 自编码器 (Auto-Encoder)

□ 去噪自编码器(Denoising Auto-Encoder, DAE)

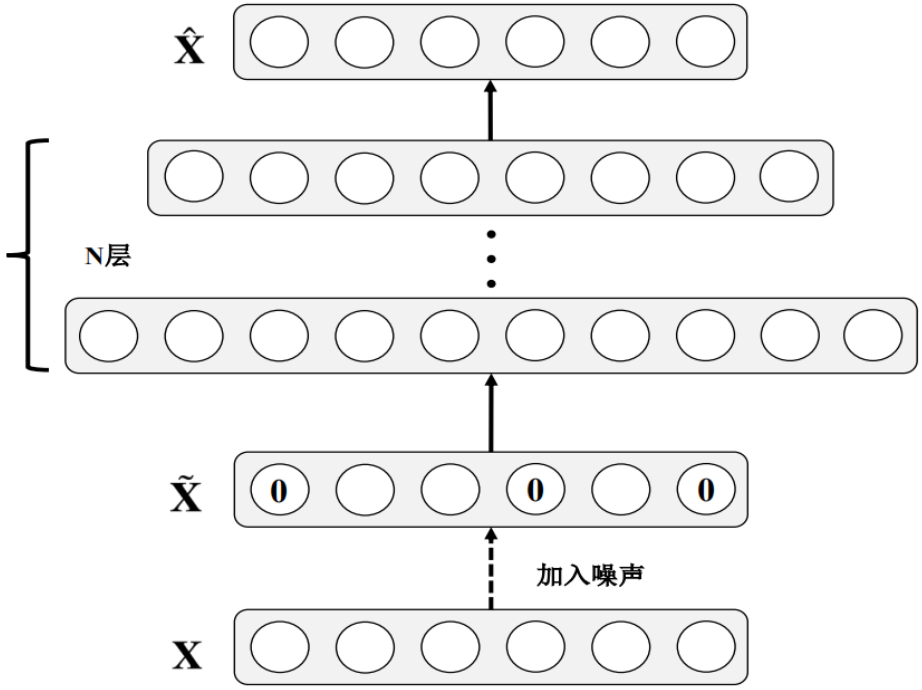


通常还可约束编码部分和解码部分的权重保持某种关系，比如捆绑约束(Tied Constraint)、稀疏约束(Sparse Constraint)等。

闭形式解

■ 自编码器 (Auto-Encoder)

□ 堆叠去噪自编码器(Stacked Denoising Auto-Encoder, DAE)



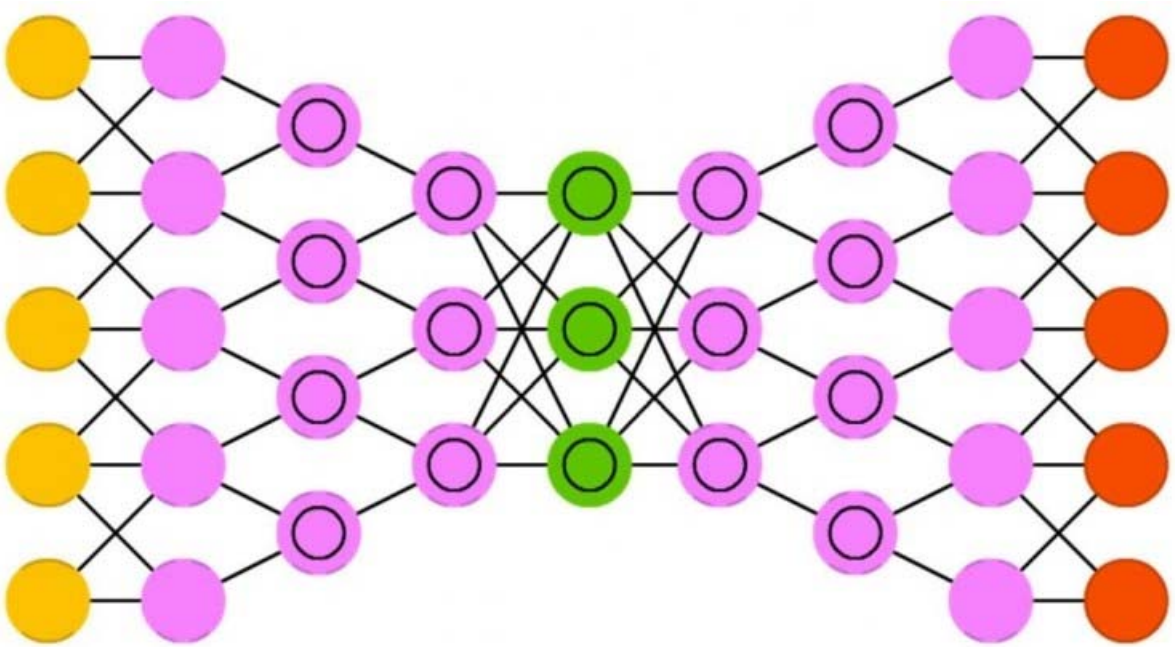
引入了更多的隐藏层，反复堆叠，从而获得了更多输入数据的深层次和抽象化特征。

闭形式解

大连理工大学 Dalian University of Technology

■ 自编码器 (Auto-Encoder)

□ 欠完备自编码器(Uncomplete Auto-Encoders, UAE)



编码维度小于输入维度的自编码器。学习欠完备的表示将强制自编码器捕捉训练数据中最显著的特征。

小结

大连理工大学 Dalian University of Technology

- 多层感知机
  - 单层感知机（连接方式，激活函数）
  - 多层感知机（等价形态，多层扩展）
- 前馈神经网络
  - 人工神经网络
  - 前馈神经网络(连接模式, 数据传递)
  - 通用近似定理
  - 分类问题
- 梯度反向传播
  - 参数学习
  - 梯度下降(链式法则, 反向传播, 自动微分)
- 闭形式解
  - 极端学习机
  - 自编码器(降噪自编码器，堆叠降噪自编码器，欠完备降噪自编码器)

基于全连接网络的手写数字识别

大连理工大学 Dalian University of Technology

```python
class FCNet(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(FCNet, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out
```

```python
class student(object):
    def __init__(self, name, age, score):
        self.name = name
        self.age = age
        self.score = score

    def print_age(self):
        print("%s's name: %s" % (self.name, self.age))

    def print_score(self):
        print("%s's age: %s" % (self.name, self.score))


stu1 = student('Amy', 13, 86)
stu2  =student('Tom', 15, 79)
```

基于全连接网络的手写数字识别

大连理工大学 Dalian University of Technology

```python
class FCNet(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(FCNet, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out
```

基于全连接网络的手写数字识别

大连理工大学 Dalian University of Technology

```python
class MyTestData(Dataset):
    def __init__(self, params):
        super(MyTestData, self).__init__()
        self.test_im_path = params['test_im_path']
        self.test_lb_path = params['test_lb_path']
        self.test_im_num = 10000
        self.test_labels = open(self.test_lb_path, 'r').read().splitlines()

    def __len__(self):
        return self.test_im_num

    def __getitem__(self, index):
        # load image
        img_file = os.path.join(self.test_im_path, str(index) + '.png')
        img = Image.open(img_file)
        im = self.transform(img)

        # load label
        lb = int(self.test_labels[index])

        return im, lb

    def transform(self, img):
        transform_img = transforms.Compose([transforms.Resize((28, 28)),
                                            transforms.ToTensor()])
        im = transform_img(img)
        return im
```

初始化
参数设置

读入图像和标签

对图像进行变换
(转化为tensor)

基于全连接网络的手写数字识别

大连理工大学

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
# 参数设置
params = {}
params['train_im_path'] = 'D:\\myproject\\mnist_pytorch\\mnist\\train-images\\'
params['test_im_path'] = 'D:\\myproject\\mnist_pytorch\\mnist\\t10k-images\\'
params['train_lb_path'] = 'D:\\myproject\\mnist_pytorch\\mnist\\train-labels.txt'
params['test_lb_path'] = 'D:\\myproject\\mnist_pytorch\\mnist\\t10k-labels.txt'
params['INPUT_SIZE'] = 784
params['HIDDEN_SIZE'] = 500
params['OUTPUT_SIZE'] = 10
params['NUM_EPOCHES'] = 10
params['BATCH_SIZE'] = 100
params['LEARNING_RATE'] = 0.001
```

参数设置

```
# load dataset
train_loader = DataLoader(MyData(params),
                          shuffle=True,
                          batch_size=params['BATCH_SIZE'])
test_loader = DataLoader(MyTestData(params),
                         shuffle=False,
                         batch_size=1)
```

加载训练数据和测试数据

```
model = FCNet(params['INPUT_SIZE'], params['HIDDEN_SIZE'], params['OUTPUT_SIZE']).to(device)
```

定义网络

```
criterion = nn.CrossEntropyLoss()
```

定义损失函数(交叉熵损失函数)

```
optimizer = torch.optim.Adam(model.parameters(), lr=params['LEARNING_RATE'])
```

定义优化方法(ADAM)

基于全连接网络的手写数字识别

大连理工大学 Dalian University of Technology

网络训练

```
for epoch in range(params['NUM_EPOCHES']):
    for i, (images, labels) in enumerate(train_loader):
        # Move tensors to the configured device
        images = images.reshape(-1, 28*28).cuda(device=0)  # # -1 是指模糊控制的意思，即固定784列，不知道多少行
        labels = labels.cuda(device=0)
        # images = images.reshape(-1, 28 * 28).to(device)  # # -1 是指模糊控制的意思，即固定784列，不知道多少行
        # labels = labels.to(device)

        # 前向传播
        outputs = model(images)
        loss = criterion(outputs, labels)

        # 反向传播和优化
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i+1) % 100 == 0:
            print('Epoch [{}/{}], Step [{}/{}], Loss: {:.4f}'
                  .format(epoch+1, params['NUM_EPOCHES'], i+1, total_step, loss.item()))
            losses.append(loss.item())
```

基于全连接网络的手写数字识别

大连理工大学 Dalian University of Technology

```
# 测试模型
# 在测试阶段，不用计算梯度
with torch.no_grad():
    correct = 0
    total = 0
    model = model.eval()
    for images, labels in test_loader:
        images = images.reshape(-1, 28 * 28).to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, predict = torch.max(outputs.data, 1)
        ##这里返回两组数据，最大image_data和最大值索引，可以用torch.argmax（）更为直观；这里去理解其
        ## 这个 _ , predicted是python的一种常用的写法，表示后面的函数其实会返回两个值
        # 但是我们对第一个值不感兴趣，就写个_在那里，把它赋值给_就好，我们只关心第二个值predicted
        # 比如 _ ,a = 1,2 这中赋值语句在python中是可以通过的，你只关心后面的等式中的第二个位置的值是
        total += labels.size(0)  ##更新测试图片的数量    size(0),返回行数
        correct += (predict == labels).sum().item()

    accuracy = correct / total
    accs.append(accuracy)
    print('Accuracy of the network on the 10000 test images: {} %'.format(100 * accuracy))
```

基于全连接网络的手写数字识别

大连理工大学 Dalian University of Technology

- 保存网络模型

```
# 保存模型
torch.save(model.state_dict(), 'model.pth')
```

基于全连接


```
Epoch [1/10], Step [100/600], Loss: 0.6180
Epoch [1/10], Step [200/600], Loss: 0.2788
Epoch [1/10], Step [300/600], Loss: 0.3504
Epoch [1/10], Step [400/600], Loss: 0.2611
Epoch [1/10], Step [500/600], Loss: 0.3334
Epoch [1/10], Step [600/600], Loss: 0.1110
Accuracy of the network on the 10000 test images: 93.2 %
Epoch [2/10], Step [100/600], Loss: 0.2007
Epoch [2/10], Step [200/600], Loss: 0.1345
Epoch [2/10], Step [300/600], Loss: 0.1862
Epoch [2/10], Step [400/600], Loss: 0.2383
Epoch [2/10], Step [500/600], Loss: 0.1317
Epoch [2/10], Step [600/600], Loss: 0.1385
Accuracy of the network on the 10000 test images: 95.5 %
Epoch [3/10], Step [100/600], Loss: 0.0772
Epoch [3/10], Step [200/600], Loss: 0.1712
Epoch [3/10], Step [300/600], Loss: 0.1935
Epoch [3/10], Step [400/600], Loss: 0.2046
Epoch [3/10], Step [500/600], Loss: 0.0839
Epoch [3/10], Step [600/600], Loss: 0.2274
Accuracy of the network on the 10000 test images: 95.67999999999999 %
Epoch [4/10], Step [100/600], Loss: 0.1628
Epoch [4/10], Step [200/600], Loss: 0.0509
Epoch [4/10], Step [300/600], Loss: 0.1581
Epoch [4/10], Step [400/600], Loss: 0.0813
Epoch [4/10], Step [500/600], Loss: 0.1619
Epoch [4/10], Step [600/600], Loss: 0.0967
Accuracy of the network on the 10000 test images: 96.12 %
Epoch [5/10], Step [100/600], Loss: 0.0991
Epoch [5/10], Step [200/600], Loss: 0.0900
Epoch [5/10], Step [300/600], Loss: 0.1194
Epoch [5/10], Step [400/600], Loss: 0.1203
Epoch [5/10], Step [500/600], Loss: 0.0709
Epoch [5/10], Step [600/600], Loss: 0.1261
Accuracy of the network on the 10000 test images: 96.72 %
Epoch [6/10], Step [100/600], Loss: 0.0240
Epoch [6/10], Step [200/600], Loss: 0.0739
Epoch [6/10], Step [300/600], Loss: 0.0865
Epoch [6/10], Step [400/600], Loss: 0.0295
Epoch [6/10], Step [500/600], Loss: 0.0645
Epoch [6/10], Step [600/600], Loss: 0.1076
```

基于全连接网络的手写数字识别

大连理工大学 Dalian University of Technology

```
# plot loss figure and accuracy figure
plt.figure()
plt.title('loss')
plt.plot(losses)

plt.figure()
plt.title('accuracy')
plt.plot(accs)

plt.show()

# 保存模型
torch.save(model.state_dict(), 'model.pth')
```

基于全连接网络的手写数字识别