2021-2022学年秋季学期

自然语言处理
Natural Language Processing

授课教师： 胡玥

助　教： 李运鹏

自然语言处理
Natural Language Processing

第 3 章　深度学习基础

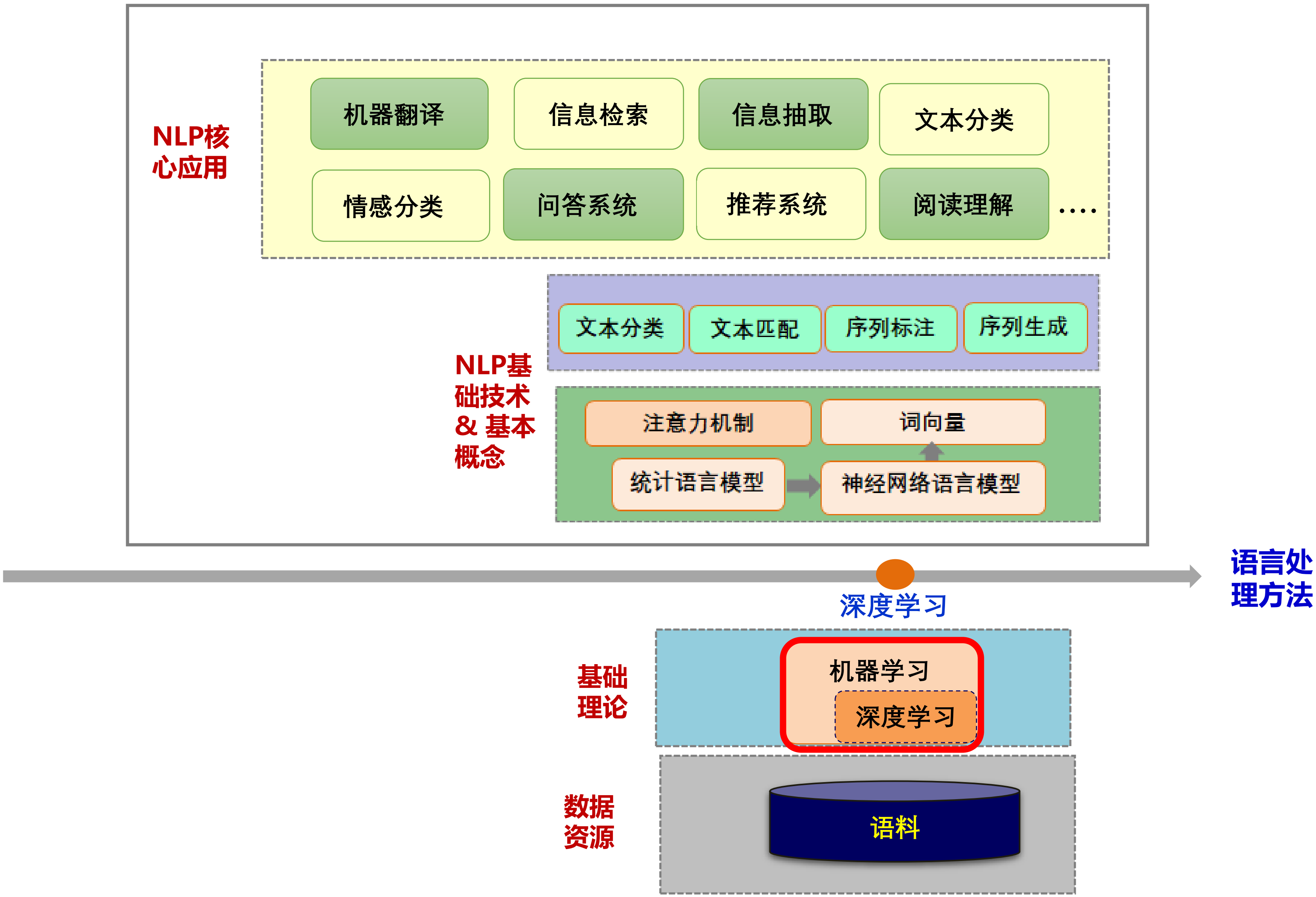
前馈神经网络

1/2

授课教师： 胡玥

授课时间：2021.9

基于深度学习的自然语言处理课程内容



概 要

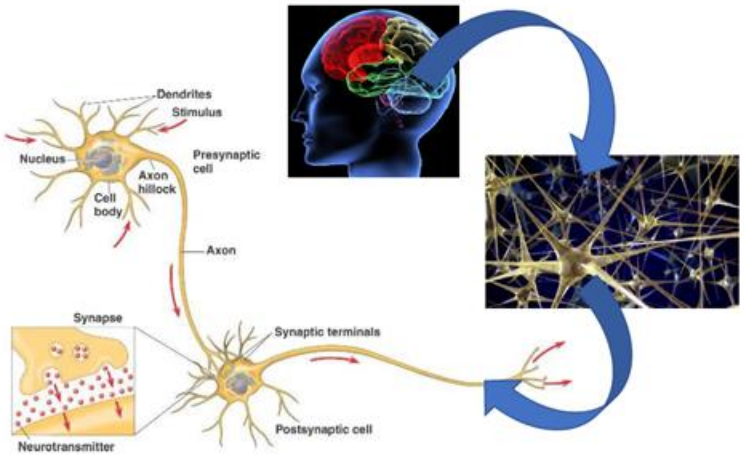
本章主要内容：本章主要介绍人工神经网络的基本概念，模型结构，参数学习算法及参数学习过程中存在问题。其中，模型结构包括：全连结前馈神经网络DNN；卷积神经网络CNN；图卷积神经网络GNN和循环神经网络 RNN

本章教学目的：通过本章教学使学生了解和掌握自然语言处理中常用的深度学习模型，为后继内容的讲解奠定基础 。

引言

■ 人工神经网络

人类大脑结构：可视作为1000多亿神经元组成的神经网络



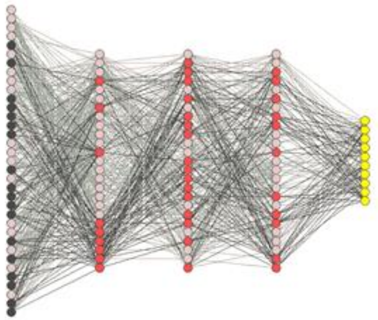
特点：巨量并行性；信息处理和存储单元结合在一起；自组织自学习功能

人工神经网络：模仿人脑网络构建不同的深度学习模型

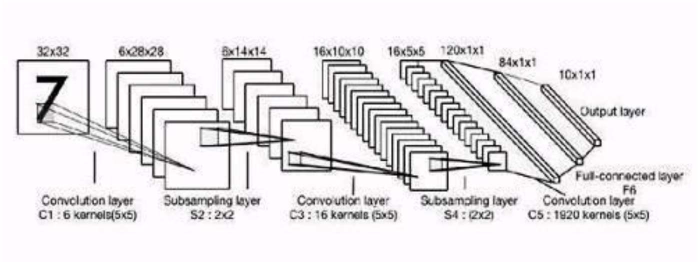
引言

■ **自然语言处理中常用的神经网络模型**

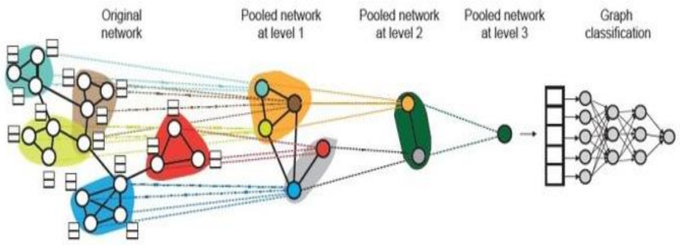
◆ 全连接前馈神经网络DNN



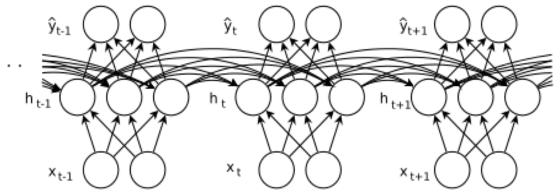
◆ 卷积神经网络CNN



◆ 图卷积神经网络GNN



◆ 循环神经网络RNN



内 容 提 要

3.1　全连接前馈神经网络 DNN

3.2　卷积神经网络 CNN

3.3　图卷积神经网络 GNN

3.4　循环神经网络 RNN

3.1 全连接前馈神经网络DNN

■ 前馈神经网络DNN

本节内容：

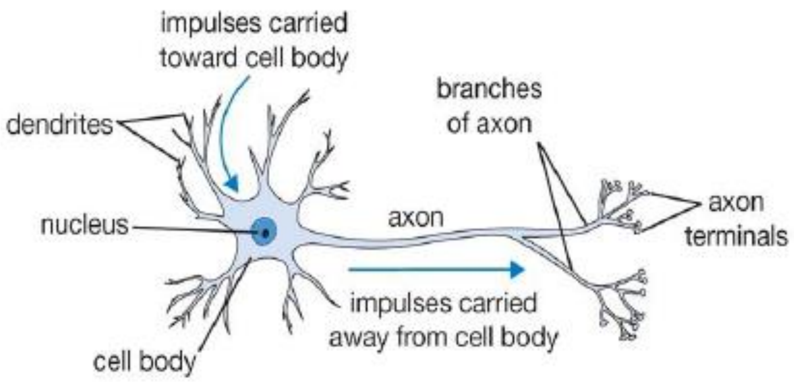
1. 人工神经元模型
2. 前馈神经网络DNN
3. 梯度下降法
4. 反向传播算法BP
5. 梯度消失问题
6. 示例

内容间逻辑关系：

模型结构	模型训练/学习	说明
前馈神经网络DNN	反向传播算法BP 梯度消失/溢出问题	DNN模型结构及训练算法
人工神经元模型	梯度下降法	神经网络基础知识

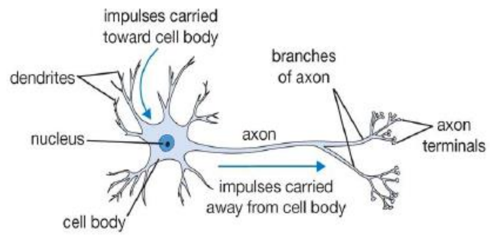
1. 人工神经元模型

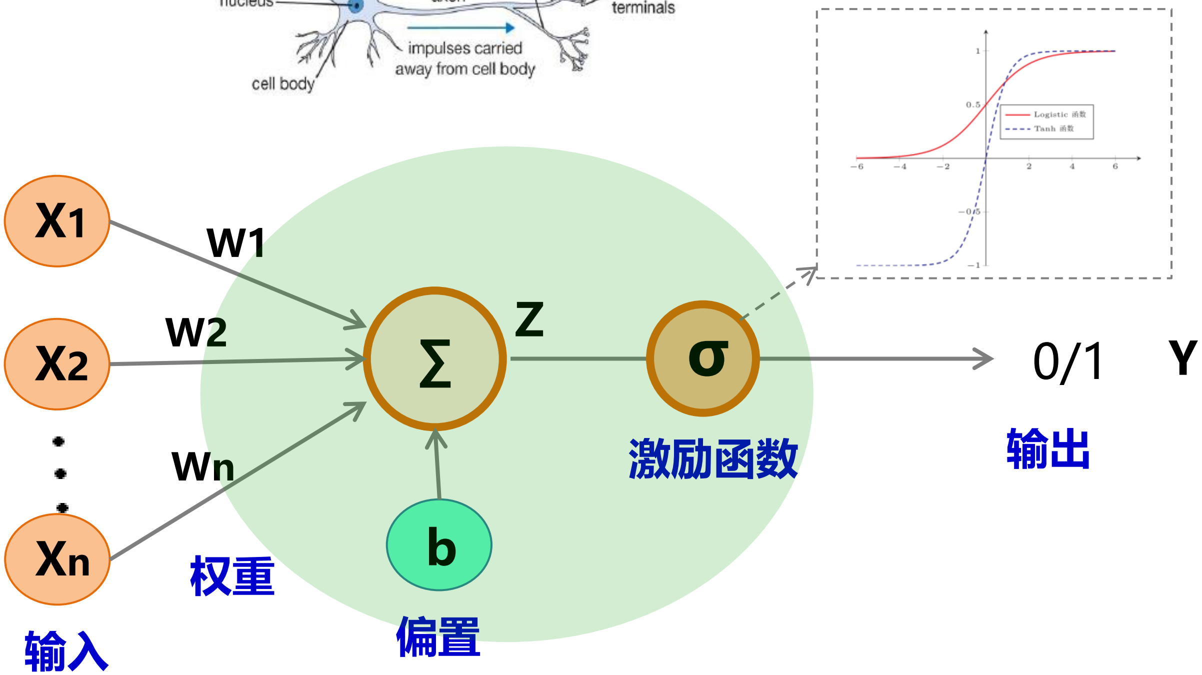
■ 生物神经元



单个神经细胞只有两种状态：兴奋和抑制

■ 人工神经元模型





输入: X

输出: Y

参数: w，b

函数关系：

$$Z = X1W1 + X2W2 + \cdots + XnWn + b$$

$$Y = \sigma(Z) = \sigma(W^T X + b)$$

1. 人工神经元模型

■ 激活函数

为了增强网络的表达能力，需要引入连续的非线性激活函数

激活函数的性质

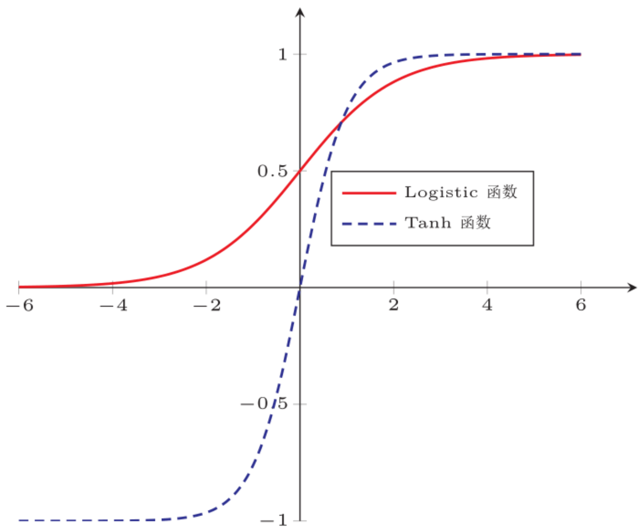
- 连续并可导（允许少数点上不可导）的非线性函数。
  - 可导的激活函数可以直接利用数值优化的方法来学习网络参数。
- 激活函数及其导函数要尽可能的简单
  - 有利于提高网络计算效率。
- 激活函数的导函数的值域要在一个合适的区间内
  - 不能太大也不能太小，否则会影响训练的效率和稳定性。

1. 人工神经元模型

常用激活函数

■ $$\sigma(x) = \frac{1}{1+\exp(-x)}$$ (Sigmoid / logistic)

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



- 饱和函数
- 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。

1. 人工神经元模型

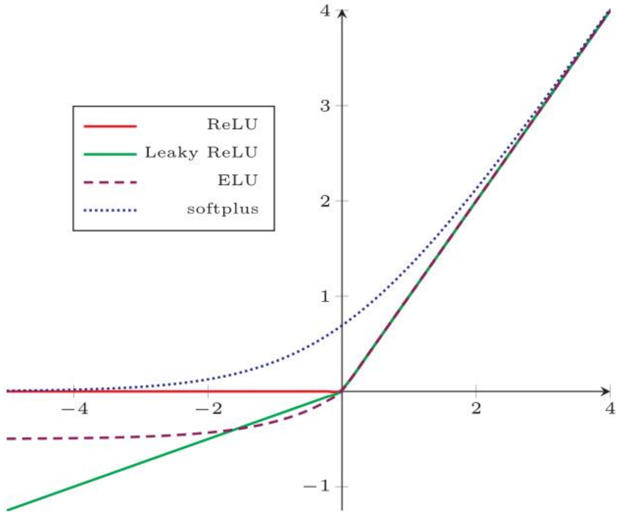
常用激活函数

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

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

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

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



- 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

1. 人工神经元模型

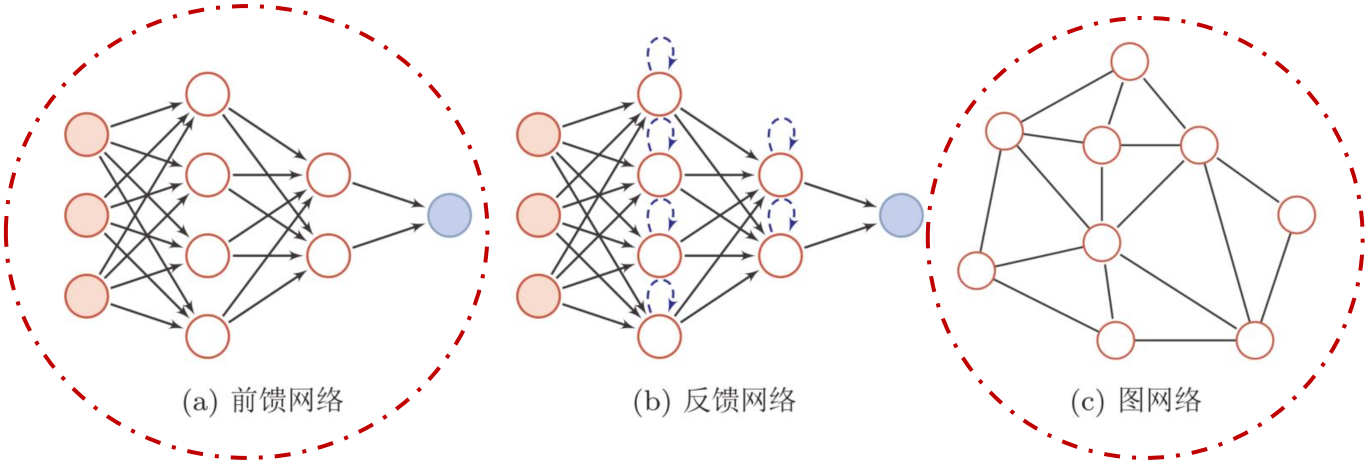
常用激活函数及导数

激活函数	函数	导数
Logistic 函数	$f(x)=\frac{1}{1+\exp(-x)}$	$f'(x)=f(x)(1-f(x))$
Tanh 函数	$f(x)=\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$f'(x)=1-f(x)^2$
ReLU 函数	$f(x)=\max(0,x)$	$f'(x)=I(x>0)$
ELU 函数	$f(x)=\max(0,x)+\min\left(0,\gamma(\exp(x)-1)\right)$	$f'(x)=I(x>0)+I(x\leq 0)\cdot\gamma\exp(x)$
SoftPlus 函数	$f(x)=\log\left(1+\exp(x)\right)$	$f'(x)=\frac{1}{1+\exp(-x)}$

1. 人工神经元模型

■ 人工神经网络

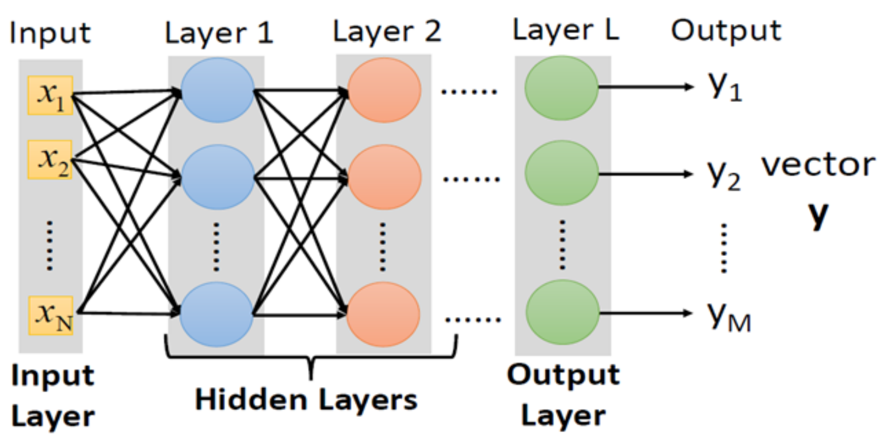
由多个神经元组成的具有并行分布结构的神经网络模型



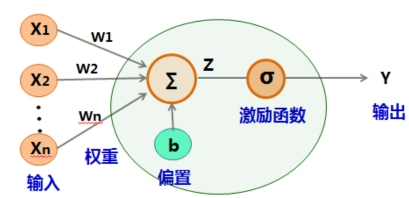
2. 前馈神经网络DNN

■ 前馈神经网络DNN

前馈神经网络中，各神经元分别属于不同的层。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 。

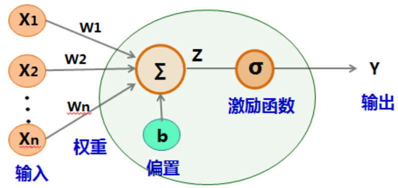


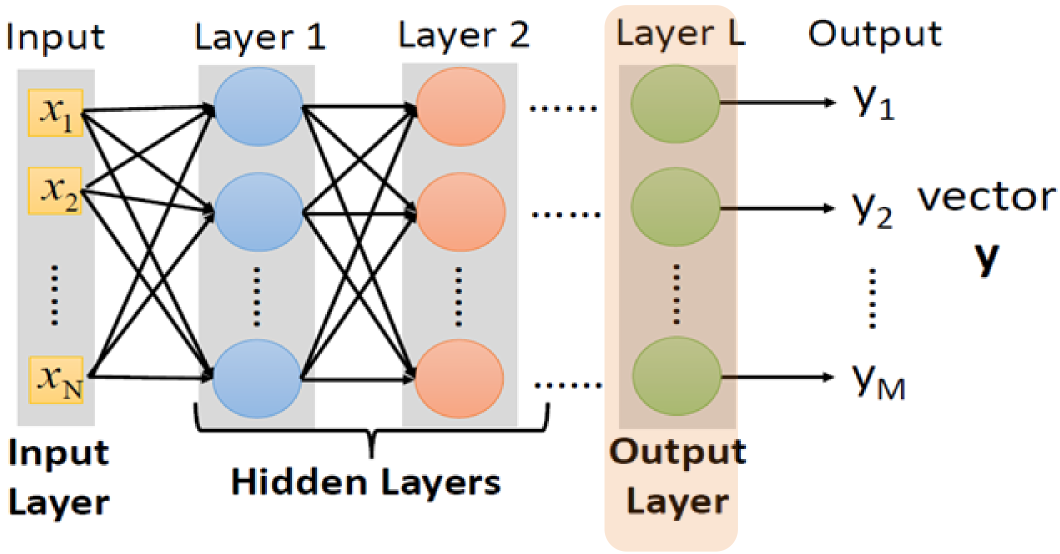
$$f: R^N \rightarrow R^M$$



2. 前馈神经网络DNN

■ DNN模型结构（输入/输出）





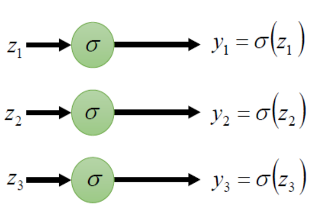
模型输入： X

模型输出： Y

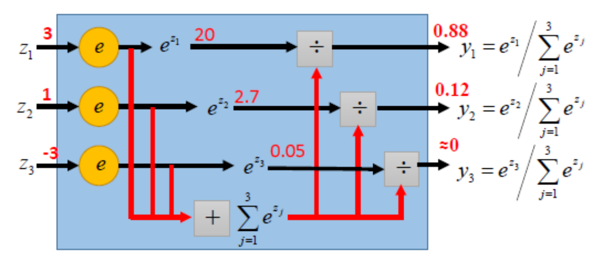
模型参数：

输出层：

一般情况：

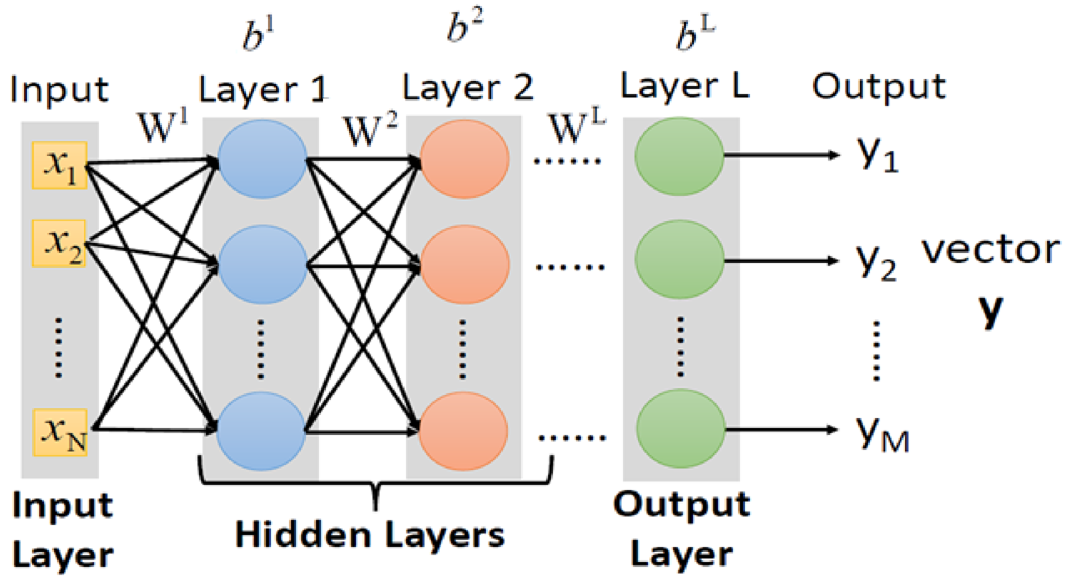


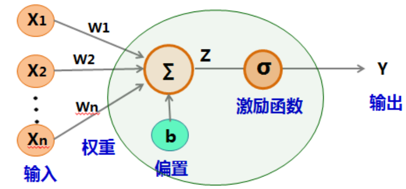
用Softmax 做输出层：



2. 前馈神经网络DNN

■ DNN模型结构 (参数)





模型输入： X

模型输出： Y

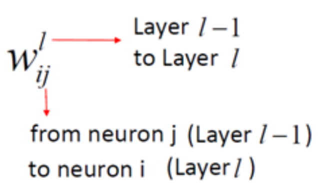
模型参数： 层间连线权重 $W^1, W^2, \cdots W^L$

各层偏置 b^1， $b^2 \cdots b^L$

参数表示说明

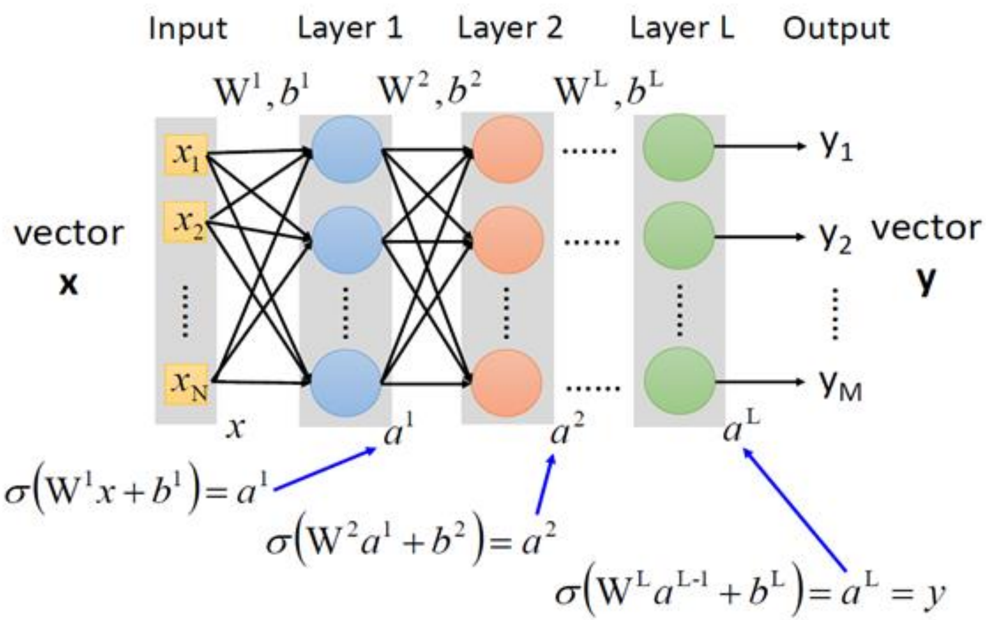
W^l : a weight matrix L-1层 到L层 权重

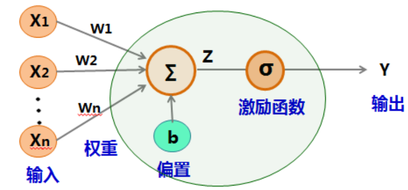
w_{ij}^l : a weight



2. 前馈神经网络DNN

■ DNN模型结构（函数关系）





输入、输出参数之间函数关系（信息传播方式）

$Y = f(x, \theta) \qquad \theta = \{W^1, b^1, W^2, b^2 \ldots W^L, b^L\}$

$$y = f(x) = \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^2\right) \ldots + b^L\right)$$

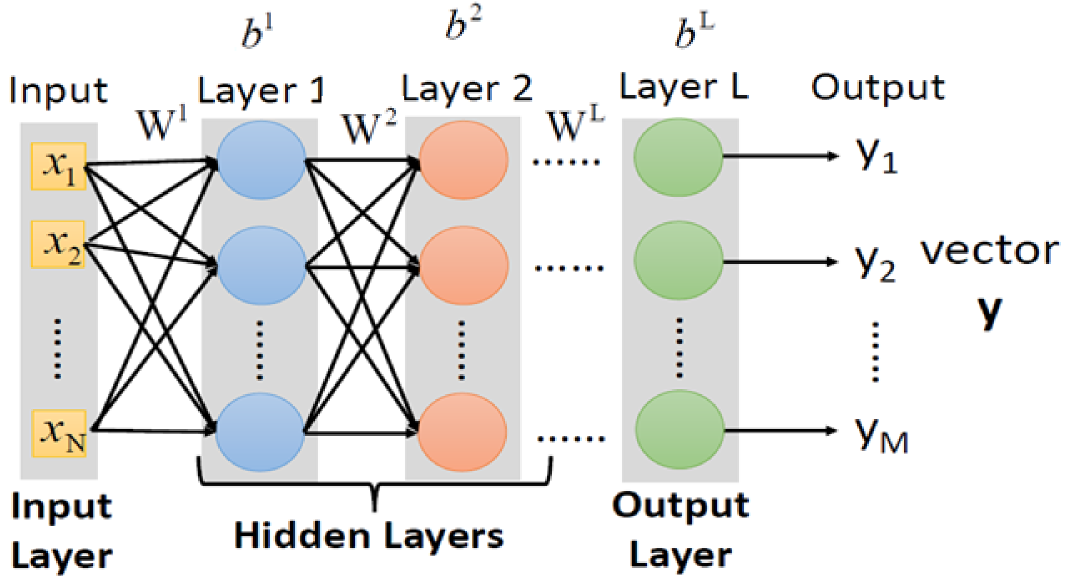
$$Z^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(Z^{(L)})$$

$$X = a^{(0)} \rightarrow Z^{(1)} \rightarrow a^{(1)} \rightarrow Z^{(2)} \rightarrow \ldots\ldots \rightarrow a^{(L-1)} \rightarrow Z^{(L)} \rightarrow a^{(L)} = Y$$

2. 前馈神经网络DNN

■ DNN模型结构



模型输入： X

模型输出： Y

模型参数： $\theta = \{W^1, b^1, W^2, b^2 \dots W^L, b^L\}$

函数关系： $y = f(x) = \sigma\left(W^L \dots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^2\right) \dots + b^L\right)$

3. 梯度下降法

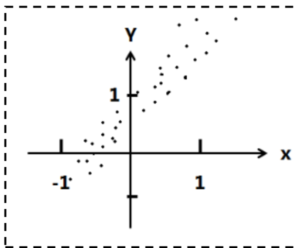
问题： 如何学习模型参数？

有监督训练：通过训练集的实例数据 （x^i; $\hat{y}^i$） 学习参数

例： y=ax+b 如何确定 a , b ？

设给定实例数据(x^i; $\hat{y}^i$)

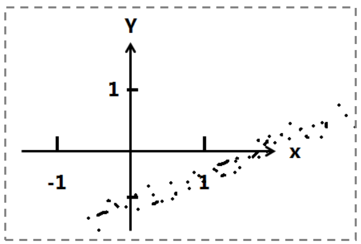
① (x^i; $\hat{y}^i$) : (1,3) , (2,5),(3,7) ⋯.



有： a =2 ; b=1

模型 : y=2x+1

② (x^i; $\hat{y}^i$) : (2,0) , (4,1),(6,2) ⋯.



有： a =1/2 ; b=-1

模型： y=(1/2)x-1

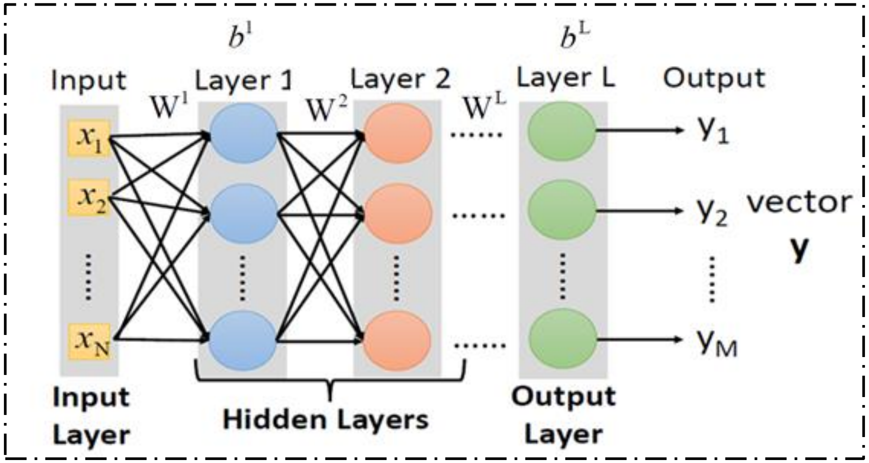
方法1： 通过列方程解决

3. 梯度下降法

神经网络参数学习问题：

有监督训练： 训练集数据 （x^i; $\hat{y}^i$）学习 θ={ W^1, b^1 ,W^2, b^2... W^L ,b^L}

特点： 参数量巨大（达到数百万） 用方法1 解方程方法不可行



神经网络一般用迭代调参方式进行参数学习

3. 梯度下降法

方法2： 迭代调参方法

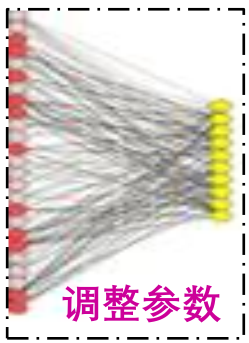
迭代调参方法：通过调整参数，让模型输出递归性地逼近标准输出。

例：设模型函数关系为 $y = f(x, \theta)$　　θ 为参数

训练集数据 $(x^i; \hat{y}^i)$

$y = f(x, \theta)$

输入： x^i



标准输出 $\hat{y}^i$

逼近

模型输出： $y^i = f(x^i, \theta)$

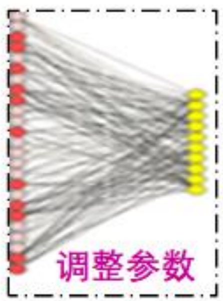
问题： 怎么调？调到什么逼近程度？

方法2： 迭代调参方法

具体方法：

$y = f(x, \theta)$

输入： x^i



标准输出 $\hat{y}^i$

逼近

模型输出： $y^i = f(x^i, \theta)$

① 用 y^i 与 $\hat{y}^i$ 的误差定义损失函数L(θ) 或 C(θ)

② 求 minC(θ)

迭代调参步骤：

① 定义目标函数（损失函数）：一般将问题转化为求极值问题

② 优化目标函数：通过求目标函数的极值 来确定参数

3. 梯度下降法

常用的损失函数有：

- 0-1损失
- 平方损失函数
- 绝对值损失函数
- 对数损失函数
- 交叉熵（负对数似然函数）
- Hinge损失
- 指数损失

……

3. 梯度下降法

◆ 绝对值损失函数：

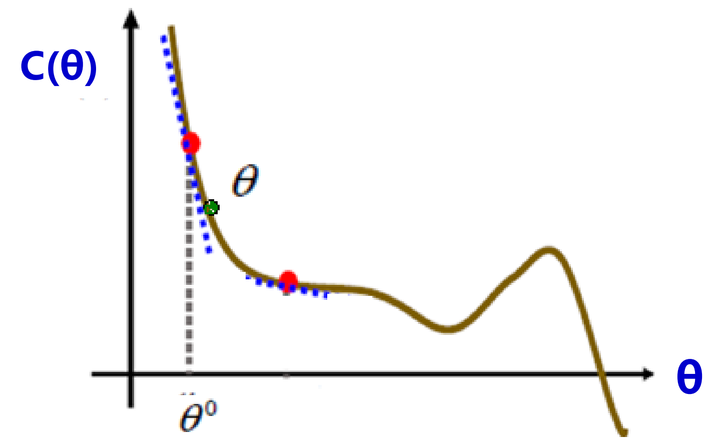
$$L(Y, f(X, \theta)) = |Y - f(X, \theta)|$$

◆ 平方损失函数：

$$L(Y, f(X, \theta)) = (Y - f(X, \theta))^2$$

◆ 交叉熵损失函数：

$$L(Y, f(X, \theta)) = -\sum_{i=1}^{C} y_i \log f_i(X, \theta)$$



如对于三类分类问题，一个样本的标签向量为 $\mathbf{y} = [0, 0, 1]^{\mathrm{T}}$，模型预测的标签分布为 $f(\mathbf{x}; \theta) = [0.3, 0.3, 0.4]^{\mathrm{T}}$，则它们的交叉熵为

$$\mathcal{L}(\theta) = -\big(0 \times \log(0.3) + 0 \times \log(0.3) + 1 \times \log(0.4)\big) = -\log(0.4).$$

用one-hot向量y来表示目标类别c其中只有y_c= 1，其余的向量元素都为 0

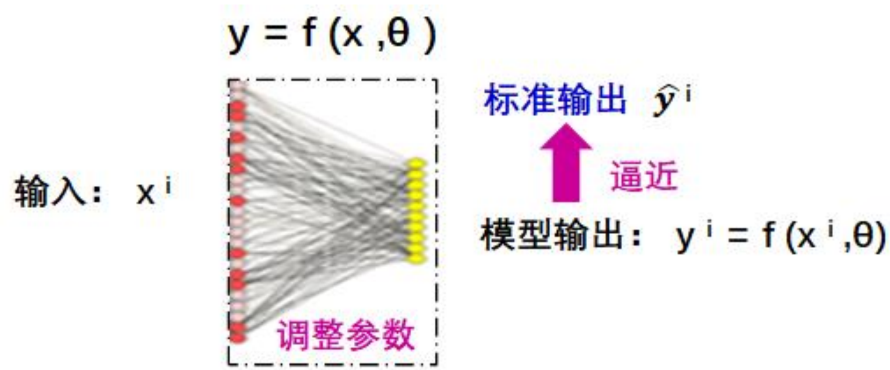
$$L(Y, f(X, \theta)) = -\log f_y(X, \theta)$$

交叉熵损失函数也是负对数似然损失函数

其中:$f_y(x,\theta)$ 为真实类别y 的似然函数。

3. 梯度下降法

① 定义目标函数（损失函数）：



① 用 y^i 与 $\hat{y}^i$ 的误差定义损失函数L(θ) 或 C(θ)

② 求 minC(θ)

绝对值损失函数：

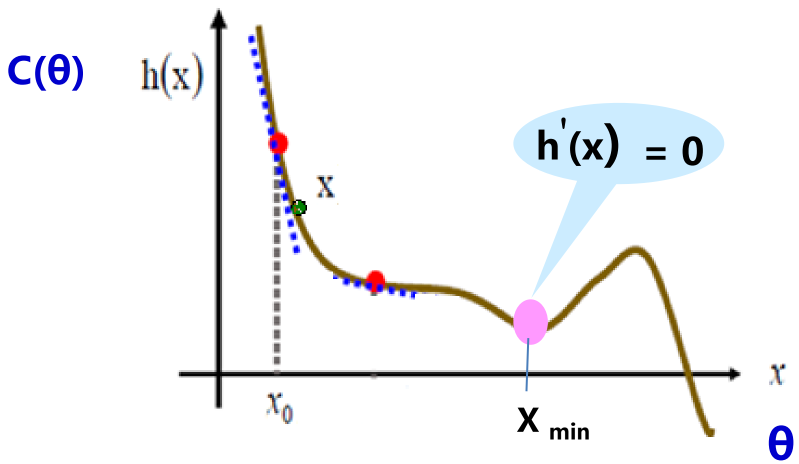
$C(\theta) = L(Y, f(X, \theta)) = |Y - f(X, \theta)|$

优化目标： 求 minC(θ)

3．梯度下降法

② 优化目标函数　　通过求目标函数的极值 来确定参数

求极值问题：有函数　y=h(x)，求 min h(x)



C(θ) =L(Y, f(X , θ)) = | Y – f(X , θ) |

原理：

泰勒展开：如 h(x) 在$x = x_0$ 附近无限可微

$$h(x) = \sum_{k=0}^{\infty} \frac{h^{(k)}(x_0)}{k!}(x - x_0)^k$$

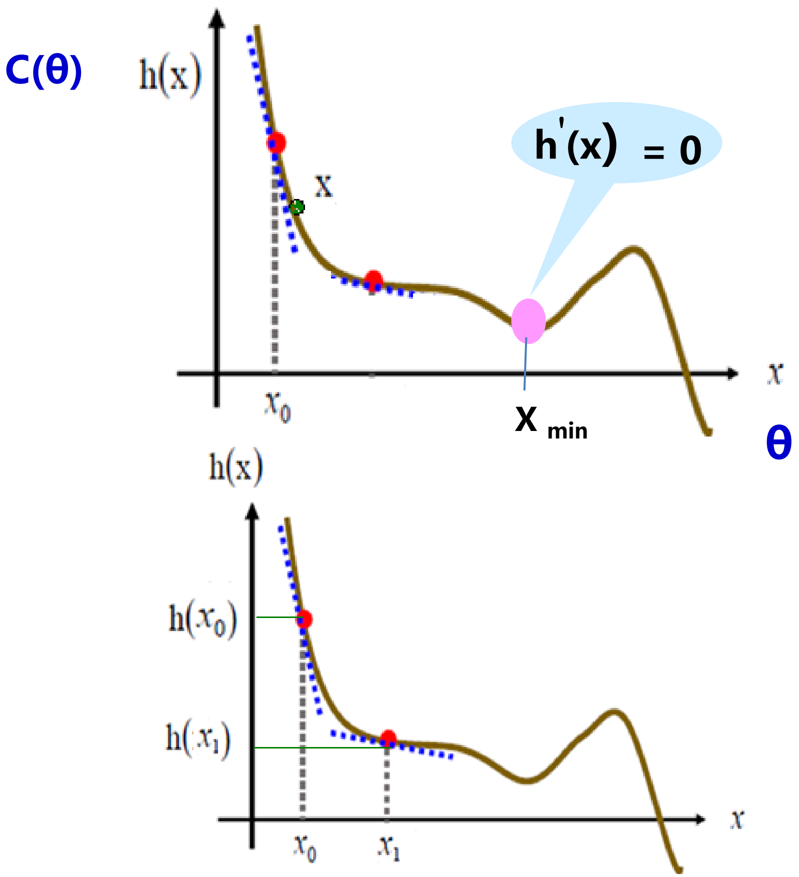
$$= h(x_0) + h'(x_0)(x - x_0) + \frac{h''(x_0)}{2!}(x - x_0)^2 + \dots$$

当 x 与 x_0 足够接近时

$$h(x) \approx h(x_0) + h'(x_0)(x - x_0)$$

$$h(x_{i+1}) = h(x_i) + h'(x_i)(x_{I+1} - x_i)$$

3. 梯度下降法



$h(x_{i+1}) = h(x_i) + h'(x_i)(x_{I+1} - x_i)$

目标：求 h(X) 极小值

每次取X_{i+1}应满足 $h(X_{i+1}) < h(X_i)$

$h(x_{i+1}) - h(x_i) = h'(x_i)(x_{I+1} - x_i) < 0$

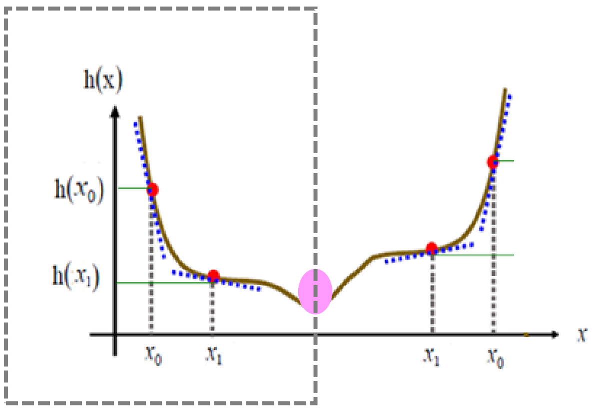
即满足 $h'(x_i)(x_{i+1} - x_i) < 0$ 条件

h(X) 将趋于变小

每步参数调整

$X_{i+1} = X_i - \eta\, h'(x_i)$

3. 梯度下降法

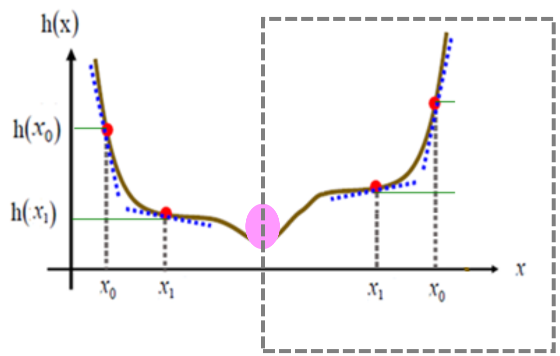


验证

从左向右调整：

$X_1 = X_0 - \eta\, h'(x_0)$

满足 $h'(x_i)(x_{i+1} - x_i) < 0$ 条件



从右向左调整：

$X_1 = X_0 - \eta\, h'(x_0)$

满足 $h'(x_i)(x_{i+1} - x_i) < 0$ 条件

参数调整方法 – 梯度下降：

$\boxed{X_{i+1} \leftarrow X_i - \eta\, h'(x_i)}$ 直到 $h'(x_i)=0$

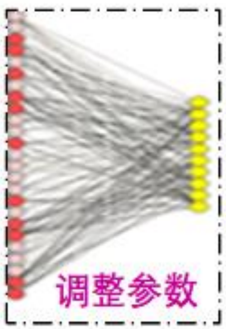
学习率 梯度

3. 梯度下降法

梯度下降方法步骤

$y = f(x, \theta)$

输入： x^i



标准输出 $\hat{y}^i$

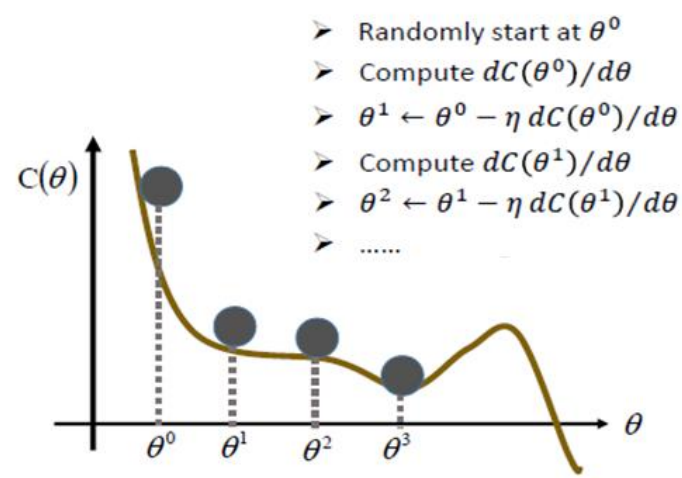
逼近

模型输出： $y^i = f(x^i, \theta)$

① 绝对值损失函数：

$C(\theta) = L(Y, f(X, \theta)) = |Y - f(X, \theta)|$

② 梯度下降过程：

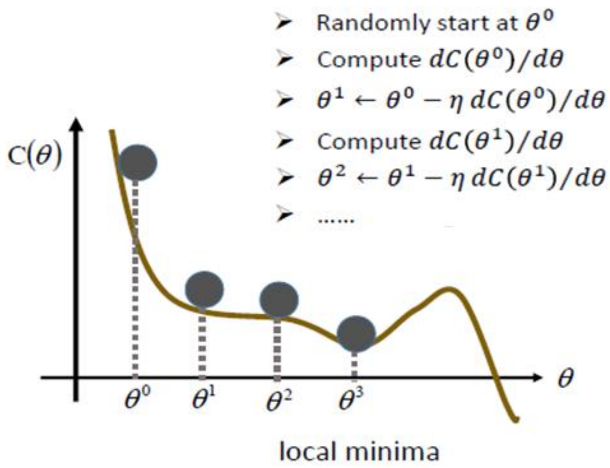


local minima

$$X_{i+1} \leftarrow X_i - \eta\, h'(x_i)$$

3. 梯度下降法

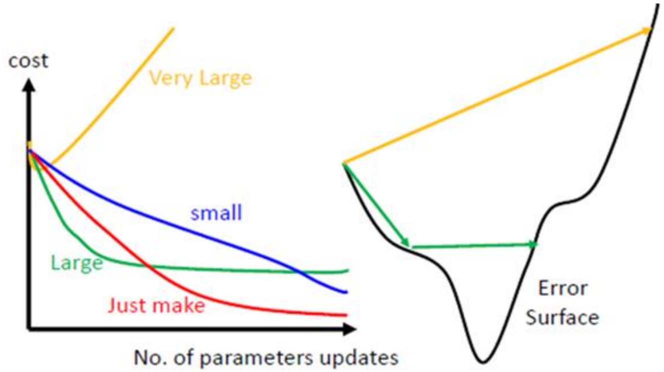
梯度下降中问题：



（1）参数初值

参数初值设置将影响参数学习效果

避免各参数初值设为相同值，参数初值设置尽量随机。



（2）学习率 η

学习率 η 设置时要注意不能过大或过小

3. 梯度下降法

■ 梯度下降法

给定训练集数据集 {(X^1 , $\hat{y}^1$)⋯ {(X^r, $\hat{y}^r$)⋯ {(X^R , $\hat{y}^R$)}

调参方法 $\theta^i = \theta^{i-1} - \eta \nabla C\left(\theta^{i-1}\right)$

◆ 梯度下降法（ Gradient Descent ）

$$C(\theta) = \frac{1}{R}\sum_r \left\| f\left(x^r;\theta\right) - \hat{y}^r \right\| \qquad \theta^i = \theta^{i-1} - \eta \nabla C\left(\theta^{i-1}\right) \qquad \nabla C\left(\theta^{i-1}\right) = \frac{1}{R}\sum_r \nabla C^r\left(\theta^{i-1}\right)$$

◆ 随机梯度下降法（ Stochastic Gradient Descent ）

$$C(\theta) = \left\| f\left(x^r;\theta\right) - \hat{y}^r \right\| \qquad \theta^i = \theta^{i-1} - \eta \nabla C^r\left(\theta^{i-1}\right) \qquad \nabla C\left(\theta^{i-1}\right) = \nabla C\left(\theta^{i-1}\right)$$

◆ mini-batch 梯度下降法（ mini batch Stochastic Gradient Descent ）

$$C(\theta) = \frac{1}{B}\sum_{x_r \in b} \left\| f\left(x^r;\theta\right) - \hat{y}^r \right\| \qquad \theta^i = \theta^{i-1} - \eta \nabla C^r\left(\theta^{i-1}\right) \qquad \nabla C\left(\theta^{i-1}\right) = \frac{1}{B}\sum_{x_r \in b} \nabla C^r\left(\theta^{i-1}\right)$$

3. 梯度下降法

随机梯度下降法

输入: 训练集 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, 验证集 $\mathcal{V}$, 学习率 α

随机初始化 θ;

repeat

  对训练集 $\mathcal{D}$ 中的样本随机重排序;

  for $n = 1 \cdots N$ **do**

    从训练集 $\mathcal{D}$ 中选取样本 $(\mathbf{x}^{(n)}, y^{(n)})$;

    // 更新参数

    $\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}\left(\theta; x^{(n)}, y^{(n)}\right)}{\partial \theta}$;

  end

until 模型 $f(\mathbf{x};\theta)$ 在验证集 $\mathcal{V}$ 上的错误率不再下降;

输出: θ

3. 梯度下降法

神经网络为一个复杂的复合函数

对于复杂的复合函数如何计算梯度？

采用链式法则

3. 梯度下降法

◆ 复合函数梯度

函数：y=h(g(x)) ，求 min h(g(x))

$$X_{i+1} \leftarrow X_i - \eta\, h'(g(x_i))$$

y=h(g(x))

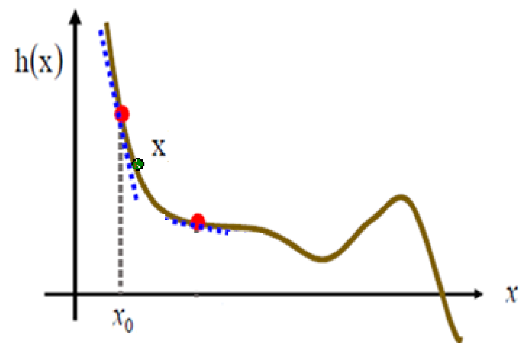
链式法则：

$$y = h(z),\ z = g(x)$$

$$\Delta x \to \Delta z \to \Delta y$$

$$\frac{dy}{dx} = \frac{dy}{dz} \cdot \frac{dz}{dx}$$

函数：y=h(x)，求 min h(x)



$$h(x_1) = h(x_0) + h'(x_0)(x_1 - x_0)$$

$$X_{i+1} \leftarrow X_i - \eta\, h'(x_i)$$

3. 梯度下降法

◆ 高维参数梯度

函数：y=h(g(x,w)) ，求 min h(g(x,w))

$$X_{i+1} \leftarrow X_i - \eta \frac{\partial y}{\partial x}$$

$$w_{i+1} \leftarrow w_i - \eta \frac{\partial y}{\partial w}$$

y=h(g(x，w))

链式法则：

$$y = h(z) \quad z = g(x, \mathrm{w})$$

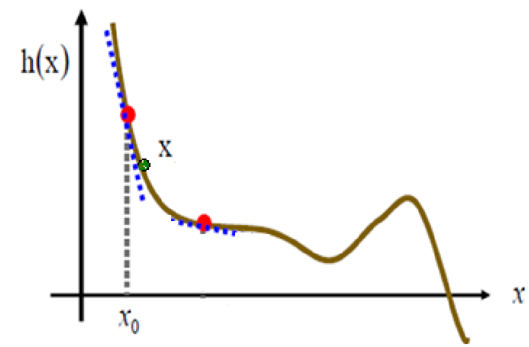
$$\Delta x \rightarrow \Delta z \rightarrow \Delta y$$

$$\Delta \mathrm{w} \rightarrow \Delta z \rightarrow \Delta y$$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z} \frac{\partial z}{\partial x}$$

$$\frac{\partial y}{\partial \mathrm{w}} = \frac{\partial y}{\partial z} \frac{\partial z}{\partial \mathrm{w}}$$

函数：y=h(x)，求 min h(x)



$$\mathrm{h}(\mathrm{x}_1) = h(x_0) + h'(x_0)(x_1 - x_0)$$

$$X_{i+1} \leftarrow X_i - \eta\, h'(x_i)$$

3. 梯度下降法

◆ 复合函数梯度

函数：z=k(g(s),h(s)) ，求 min k(g(s),h(s))

$$s_{i+1} \leftarrow s_i - \eta \frac{\partial z}{\partial s}$$

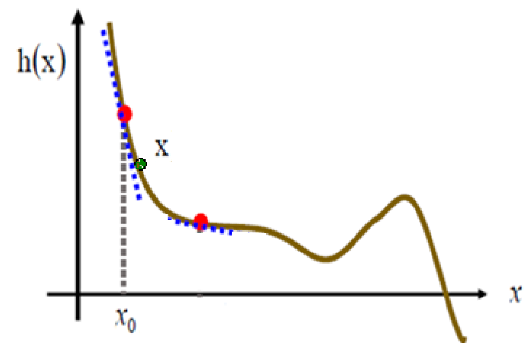
z=k(g(s),h(s))

链式法则：

$$x = g(s) \qquad y = h(s) \qquad z = k(x, y)$$

$$\begin{tikzcd} & \Delta x \arrow[dr] & \\ \Delta s \arrow[ur] \arrow[dr] & & \Delta z \\ & \Delta y \arrow[ur] & \end{tikzcd} \qquad \frac{\partial z}{\partial s} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial s}$$

函数：y=h(x)，求 min h(x)



$$h(x_1) = h(x_0) + h'(x_0)(x_1 - x_0)$$

$$\boxed{x_{i+1} \leftarrow x_i - \eta\, h'(x_i)}$$

4. 反向传播算法

■ 反向传播算法 （Back Propagation ）

1974年Webos在博士论文中首次提出BP算法，但未引发关注 .目前广泛使用的BP算法诞生于1986年 以全连接层为例：链式求 导，梯度反向传导 .

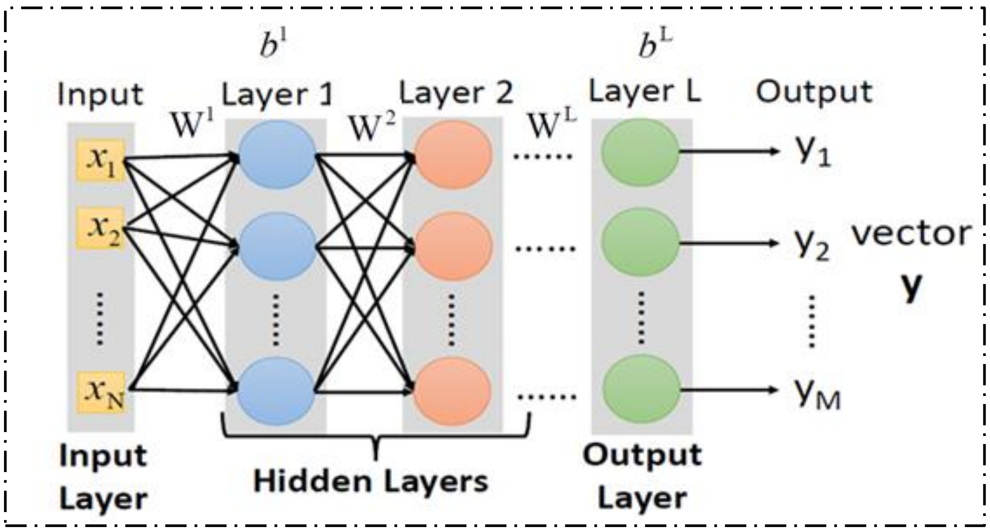
核心思想：

将输出误差以某种形式反传给各层所有的单元，各层按本层误差修正各单元连接权值。

有监督学习，采用梯度下降法调参

4. 反向传播算法

■ 反向传播算法 （Back Propagation ）



全连接前馈神经网

任务： 给定训练集数据集 $\{(x^1, \hat{y}^1) \cdots \{(x^r, \hat{y}^r) \cdots \{(x^R, \hat{y}^R)\}$

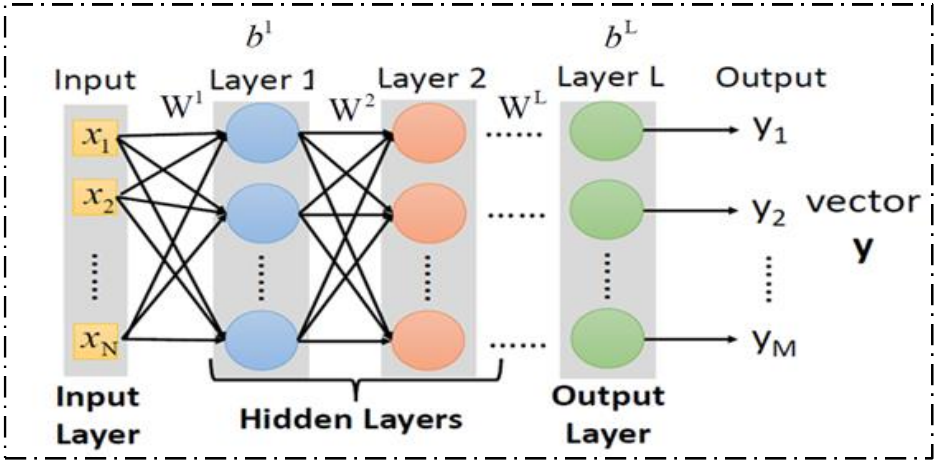
学习 网络参数 $\theta = \{W^1, b^1, W^2, b^2 \cdots W^L, b^L\}$

4. 反向传播算法

① 定义损失函数

训练集数据 $\{(x^1, \hat{y}^1) \cdots (x^r, \hat{y}^r) \cdots (x^R, \hat{y}^R)\}$

输入 x^i



标准输出 $\hat{y}^i$

模型输出 $y^i = f(x^i, \theta)$

函数： $y = f(x) = \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^2\right) \ldots + b^L\right)$

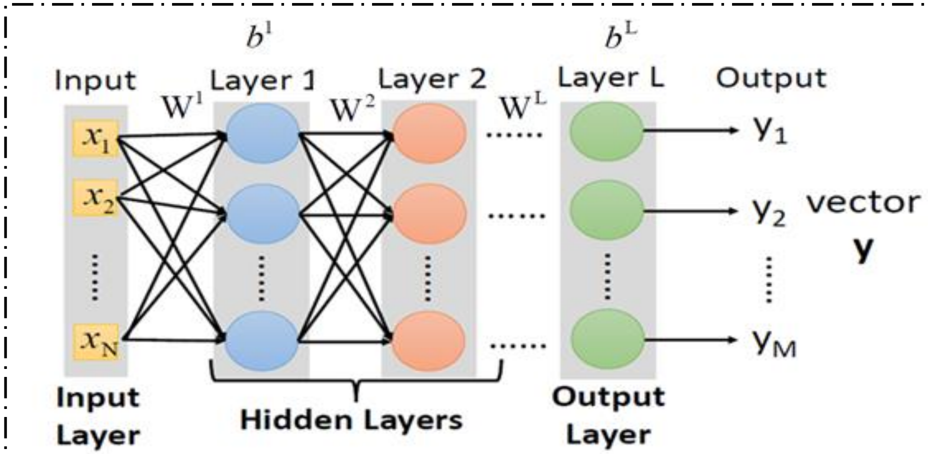
参数： $\theta = \{W^1, b^1, W^2, b^2 \cdots W^L, b^L\}$

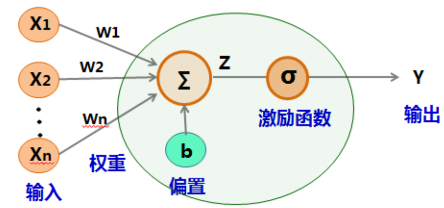
损失函数 $C(\theta) = \frac{1}{R}\sum_r \left\| f(x^r; \theta) - \hat{y}^r \right\| = \frac{1}{R}\sum_r \left\| \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^2\right) \ldots + b^L\right) - \hat{y}^r \right\|$

各层参数是损失函数的多层复合函数

4. 反向传播算法

各层参数与损失函数的关系：





损失函数：

$$C(\theta)=\frac{1}{R}\sum_r\left\|f(x^r;\theta)-\hat{y}^r\right\|$$

$$=\frac{1}{R}\sum_r\left\|\sigma\left(W^L\ldots\sigma\left(W^2\sigma\left(W^1x+b^1\right)+b^2\right)\ldots+b^L\right)-\hat{y}^r\right\|$$

$$y=f(x)=\sigma\left(W^L\ldots\sigma\left(W^2\sigma\left(W^1x+b^1\right)+b^2\right)\ldots+b^L\right)$$

各层参数是损失函数的多层复合函数

$Z^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)}$　　$a^{(L)} = \sigma(Z^{(L)})$

$W^1 \rightarrow Z^{(1)} \rightarrow a^{(1)} \rightarrow Z^{(2)} \rightarrow \ldots\ldots \rightarrow a^{(L-1)} \rightarrow Z^{(L)} \rightarrow a^{(L)} = Y$

$\Delta W^1 \rightarrow \Delta Z^{(1)} \rightarrow \Delta a^{(1)} \rightarrow \Delta Z^{(2)} \rightarrow \ldots\ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C\ (\theta)$

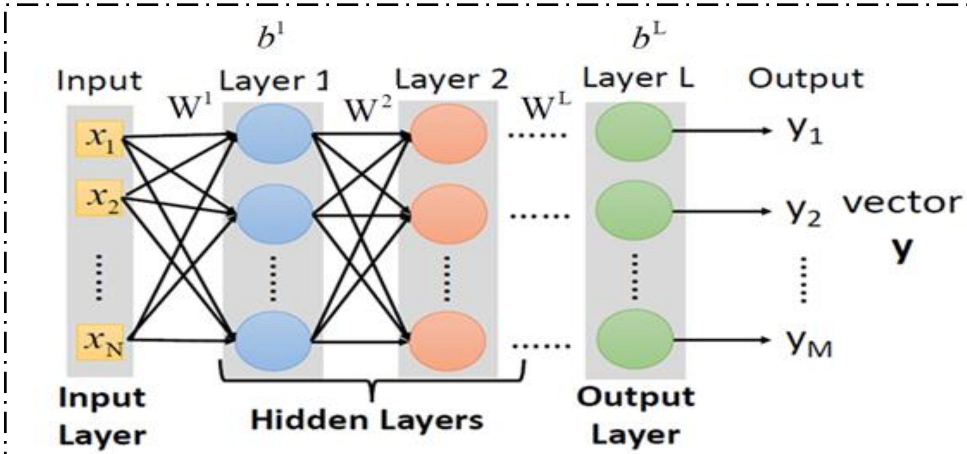
$\Delta W^2 \rightarrow \Delta Z^{(2)} \rightarrow \Delta a^{(2)} \rightarrow \Delta Z^{(3)} \rightarrow \ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C\ (\theta)$

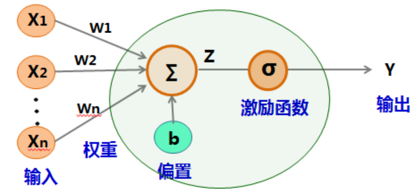
………

$\Delta W^L \rightarrow \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C\ (\theta)$

4. 反向传播算法

② 调参数优化目标函数





$$Z^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)}$$

各层均采用梯度下降法调整参数

$$\theta^* = arg \min_{\theta} C(\theta)$$

$$[W^l]^1 \longleftarrow [W^l]^0 - \eta \frac{\partial C(\theta)}{\partial wl}$$

$$[b^l]^1 \longleftarrow [b^l]^0 - \eta \frac{\partial C(\theta)}{\partial bl}$$

$y=h(g(x))$ $\min h(g(x))$

$X_{i+1} \leftarrow X_i - \eta\, h'(g(x_i))$

链式法则：

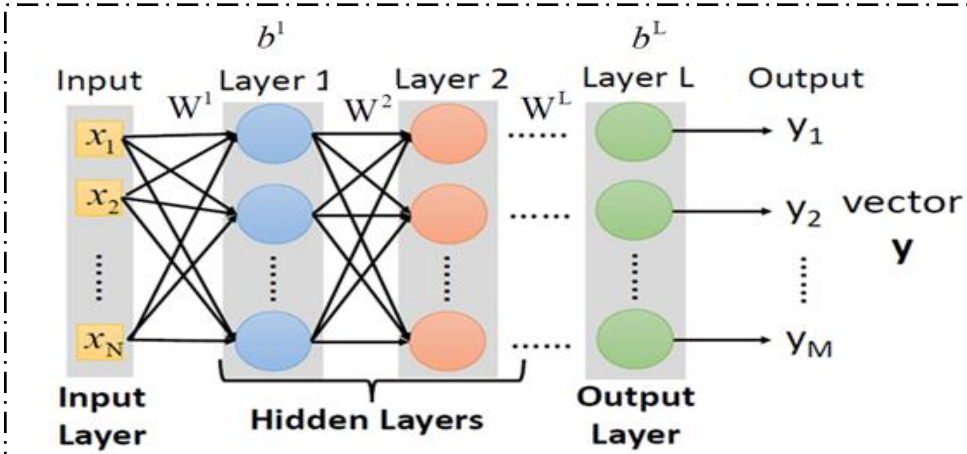
$y = h(z) \quad z = g(x)$

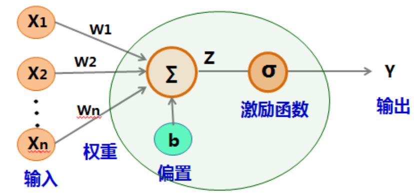
$\Delta x \rightarrow \Delta z \rightarrow \Delta y$

$$\frac{dy}{dx} = \frac{dy}{dz} \cdot \frac{dz}{dx}$$

4. 反向传播算法

② 调参数优化目标函数





$$Z^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)}$$

$$\Delta W^1 \rightarrow \Delta Z^{(1)} \rightarrow \Delta a^{(1)} \rightarrow \Delta Z^{(2)} \rightarrow \ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$$

$$\Delta W^2 \rightarrow \Delta Z^{(2)} \rightarrow \Delta a^{(2)} \rightarrow \Delta Z^{(3)} \rightarrow \ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$$

$$\ldots\ldots$$

$$\Delta W^L \rightarrow \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$$

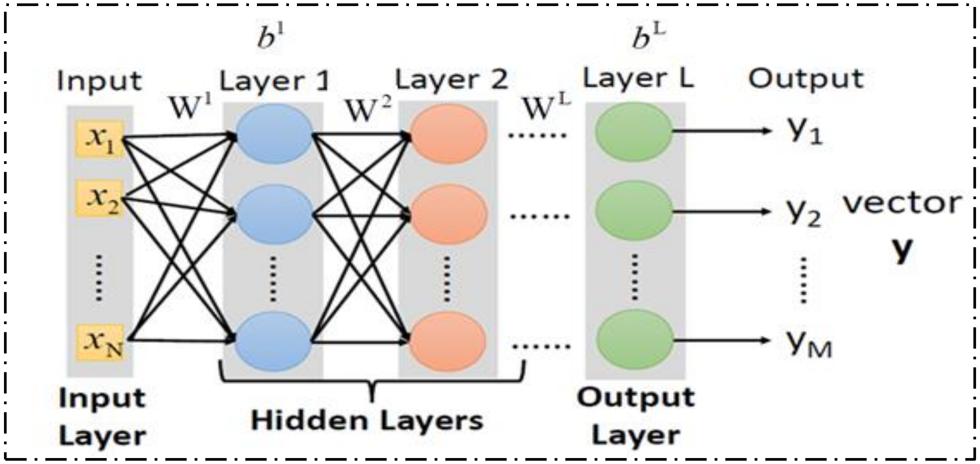
各层均采用梯度下降法调整参数

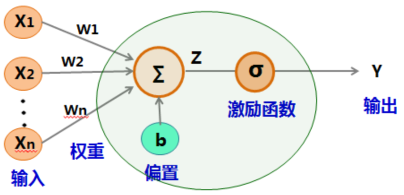
$$[W^l]^1 \leftarrow [W^l]^0 - \eta \frac{\partial C(\theta)}{\partial W^l}$$

$$\frac{\partial C(\theta)}{\partial W^l} = \underbrace{\boxed{\frac{\partial z^l}{\partial W^l}}}_{a^{l-1}} \frac{\partial C(\theta)}{\partial z^l} = a^{l-1} \boxed{\frac{\partial C(\theta)}{\partial z^l}}^{\,?}$$

4. 反向传播算法

求： $\frac{\partial C(\theta)}{\partial Z^l}$





$Z^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)}$

$Y^L = \sigma(Z^L)$

$\Delta W^1 \rightarrow \Delta Z^{(1)} \rightarrow \Delta a^{(1)} \rightarrow \Delta Z^{(2)} \rightarrow \ldots\ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$

$\Delta W^2 \rightarrow \Delta Z^{(2)} \rightarrow \Delta a^{(2)} \rightarrow \Delta Z^{(3)} \rightarrow \ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$

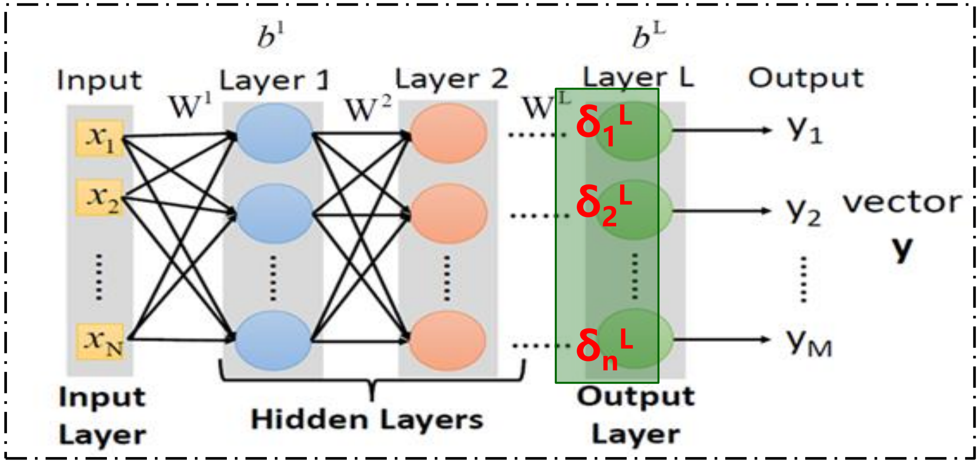
....

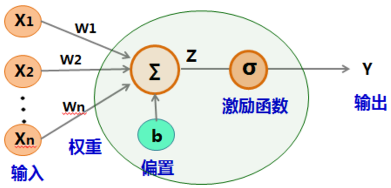
$\Delta W^L \rightarrow \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$

定义一个误差项 δ^l $\frac{\partial C(\theta)}{\partial Z^l} = \delta^l$ 先求最后一层误差 δ^L

4. 反向传播算法

最后层一个误差 δ^L





损失函数：

$$C(\theta)=\frac{1}{R}\sum_r \left\| y^r - \hat{y}^r \right\|$$

$$Z^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)} \qquad Y^L = \sigma(Z^L)$$

$$\Delta W^1 \rightarrow \Delta Z^{(1)} \rightarrow \Delta a^{(1)} \rightarrow \Delta Z^{(2)} \rightarrow \ldots\ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$$

$$\Delta W^2 \rightarrow \Delta Z^{(2)} \rightarrow \Delta a^{(2)} \rightarrow \Delta Z^{(3)} \rightarrow \ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$$

....

$$\Delta W^L \rightarrow \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$$

$$\delta^L = \frac{\partial C(\theta)}{\partial Z^L} = \frac{\partial C(\theta)}{\partial Y^L}\,\frac{\partial Y^L}{\partial Z^L}$$

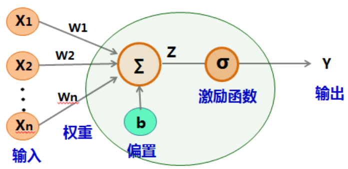
$$C(\theta)=\frac{1}{R}\sum_r \left\| y^r - \hat{y}^r \right\|$$

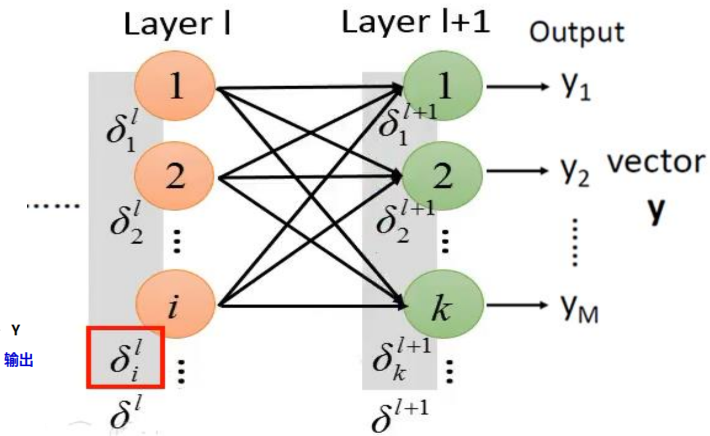
$$\sigma'(Z^L)$$

$$\boxed{\delta^L = \sigma'(z^l)\bullet\nabla C^r(y^r)}$$

4. 反向传播算法

l 层误差 δ^l 与 l +1 层误差 δ^{l+1} 的关系 (关键步骤)

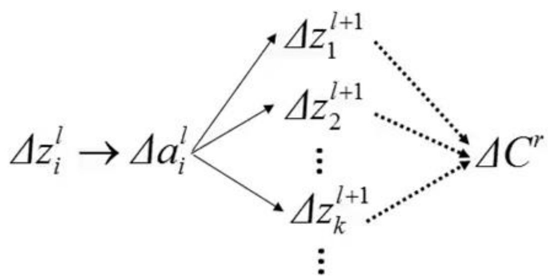




损失函数：

$$C(\theta)=\frac{1}{R}\sum_r\left\|f\left(x^r;\theta\right)-\hat{y}^r\right\|$$

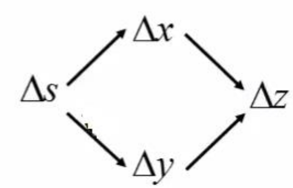
$$\delta_i^l=\frac{\partial C^r}{\partial z_i^l}$$



$$\delta_i^l=\frac{\partial C^r}{\partial z_i^l}=\frac{\partial a_i^l}{\partial z_i^l}\sum_k\frac{\partial z_k^{l+1}}{\partial a_i^l}\frac{\partial C^r}{\partial z_k^{l+1}}$$

链式法则：

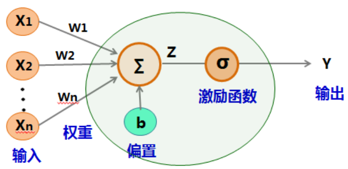
$x=g(s)$ $\quad y=h(s)$ $\quad z=k(x,y)$

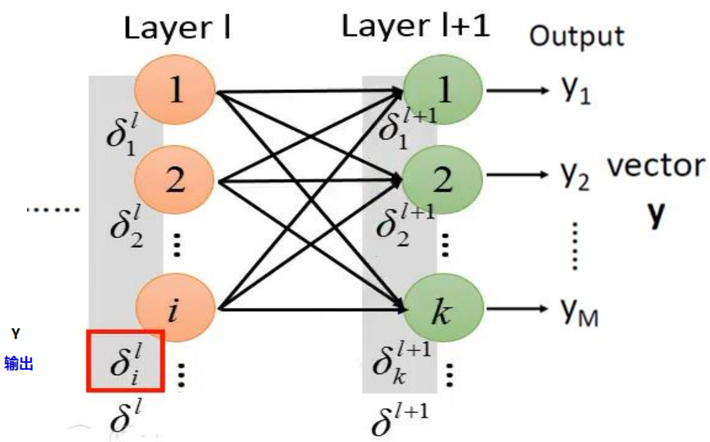


$$\frac{\partial z}{\partial s}=\frac{\partial z}{\partial x}\frac{\partial x}{\partial s}+\frac{\partial z}{\partial y}\frac{\partial y}{\partial s}$$

4. 反向传播算法

l 层误差 δ^l 与 l +1 层误差 δ^{l+1} 的关系 (关键步骤)





损失函数:

$$\boxed{C(\theta)=\frac{1}{R}\sum_r\left\|f\left(x^r;\theta\right)-\hat{y}^r\right\|}$$

$Z^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)}$ $\quad\quad$ $a^L = \sigma(Z^L)$

$$\boxed{z_k^{l+1}=\sum_i w_{ki}^{l+1}a_i^l+b_k^{l+1}}$$

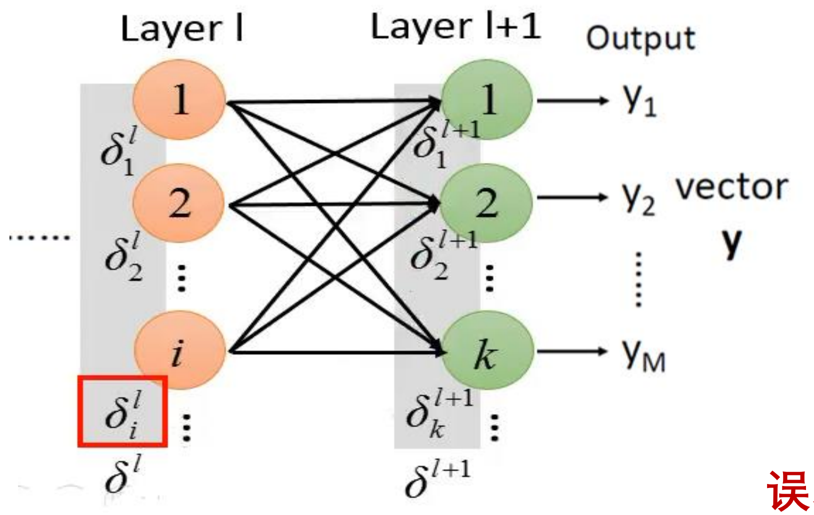
$$\delta_i^l=\frac{\partial C^r}{\partial z_i^l}=\underbrace{\boxed{\frac{\partial a_i^l}{\partial z_i^l}}}_{\sigma'\left(z_i^l\right)}\sum_k\underbrace{\boxed{\frac{\partial z_k^{l+1}}{\partial a_i^l}}}_{w_{ki}^{l+1}}\underbrace{\boxed{\frac{\partial C^r}{\partial z_k^{l+1}}}}_{\delta_k^{l+1}}=\sigma'\left(z_i^l\right)\sum_k w_{ki}^{l+1}\delta_k^{l+1}$$

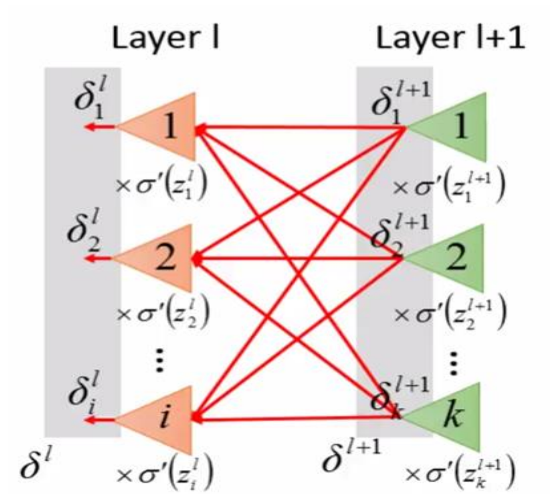
$$\boxed{\delta_i^l=\sigma'\left(z_i^l\right)\sum_k w_{ki}^{l+1}\delta_k^{l+1}}$$

4. 反向传播算法

根据 δ^{l+1} 求 δ^{l} （误差反传）

$$\delta_i^l = \sigma'\left(z_i^l\right)\sum_k w_{ki}^{l+1}\delta_k^{l+1}$$





误差反传公式

$$\boxed{\delta^l = \sigma'\left(z^l\right)\bullet\left(W^{l+1}\right)^T\delta^{l+1}}$$

$$\sigma'\left(z^l\right)=\begin{bmatrix}\sigma'\left(z_1^l\right)\\ \sigma'\left(z_2^l\right)\\ \vdots\\ \sigma'\left(z_i^l\right)\\ \vdots\end{bmatrix}$$

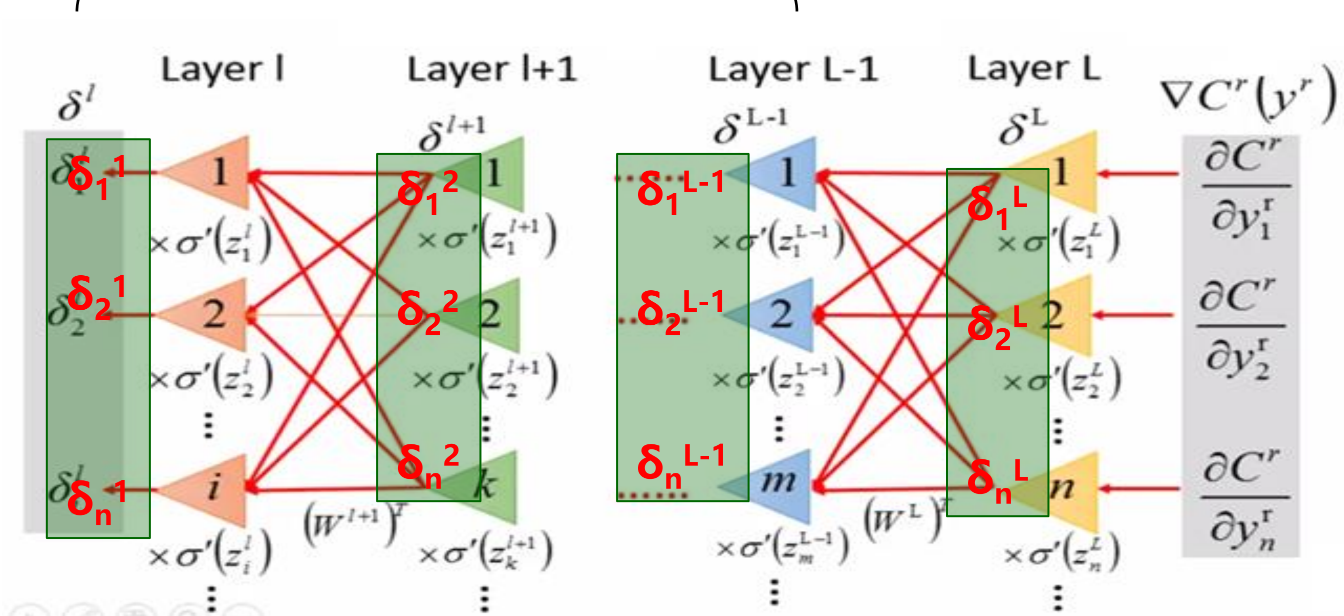
4. 反向传播算法

误差反传过程：

首先计算最后层误差δ^L，然后根据误差反传公式求得 倒数第二层误差δ^{L-1}直至第一层。

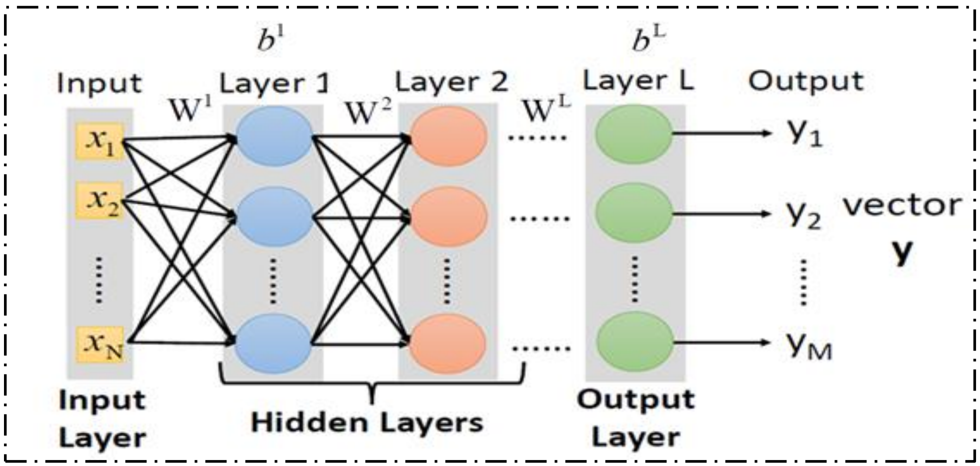
$$\boxed{\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}}$$

$$\boxed{\delta^L = \sigma'(z^l) \bullet \nabla C^r(y^r)}$$



4. 反向传播算法

② 调参数优化目标函数



损失函数:

$$C(\theta)=\frac{1}{R}\sum_r \left\| y^r-\hat{y}^r \right\|$$

$$\mathbf{Z^{(L)}=W^{(L)}\,a^{(L-1)}+b^{(L)}}$$

$\Delta W^1 \rightarrow \Delta Z^{(1)} \rightarrow \Delta a^{(1)} \rightarrow \Delta Z^{(2)} \rightarrow \ldots\ldots\ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$

$\Delta W^2 \rightarrow \Delta Z^{(2)} \rightarrow \Delta a^{(2)} \rightarrow \Delta Z^{(3)} \rightarrow \ldots\ldots \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$

……..

$\Delta W^L \rightarrow \Delta Z^{(L)} \rightarrow \Delta a^{(L)} \rightarrow \Delta Y \rightarrow \Delta C(\theta)$

各层均采用梯度下降法调整参数

$$[W^l]^1 \longleftarrow [W^l]^0 - \eta \frac{\partial C(\theta)}{\partial W^l} \qquad \frac{\partial C(\theta)}{\partial W^l} = \frac{\partial Z^l}{\partial W^l}\frac{\partial C(\theta)}{\partial Z^l} = a^{l-1}\,\delta^l$$

4. 反向传播算法

前馈神经网络的训练过程可以分为以下三步：

（1）先前馈计算每一层的状态和激活值，直到最后一层；

（2）反向传播计算每一层的误差；

（3）计算每一层参数的偏导数，并更新参数

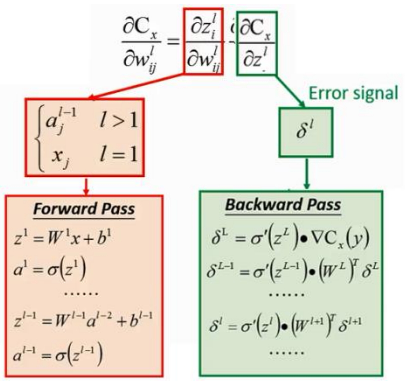
4. 反向传播算法

反向传播算法

输入：训练集：$(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \cdots, N$，最大迭代次数：$T$

输出：$W, \mathbf{b}$

初始化 $W, \mathbf{b}$;
for $t = 1 \cdots T$ **do**
    for $i = 1 \cdots N$ **do**
        (1) 前馈计算每一层的状态和激活值，直到最后一层;
        (2) 用公式(3)反向传播计算每一层的误差 $\delta^{(l)}$;
        (3) 用公式(1)和(2)每一层参数的导数;
$$\frac{\partial C(W, \mathbf{b}; \mathbf{x}, y)}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^T$$
$$\frac{\partial C(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$
        (4) 更新参数;
$$W^{(l)} = W^{(l)} - \alpha \sum_{i=1}^{N} \left(\frac{\partial C(W,\mathbf{b};\mathbf{x}^{(i)},y^{(i)})}{\partial W^{(l)}} \right)$$
$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \sum_{i=1}^{N} \left(\frac{\partial C(W,\mathbf{b};\mathbf{x}^{(i)},y^{(i)})}{\partial \mathbf{b}^{(l)}} \right);$$
    end
end



(1)
$$\frac{\partial C(W, \mathbf{b}; \mathbf{x}, y)}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^T$$

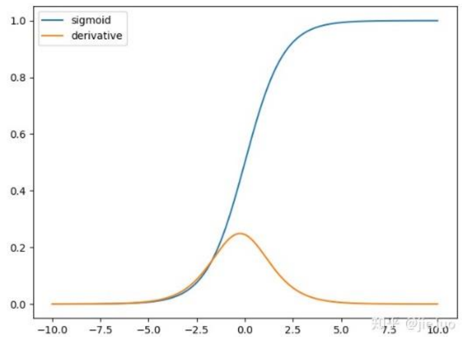
(2)
$$\frac{\partial C(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$

(3)
$$\delta^{l} = \sigma'(z^{l}) \bullet (W^{l+1})^T \delta^{l+1}$$

5. 梯度消失问题

激活函数分析

◆ Sigmoid函数

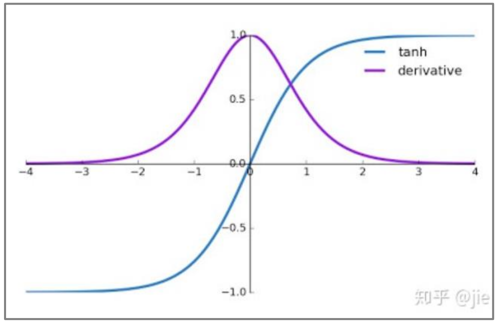


表达式：$\sigma(x) = \frac{1}{1+e^{-x}}$

值域：（0，1）

导数值域：（0，0.25）

◆ Tanh函数



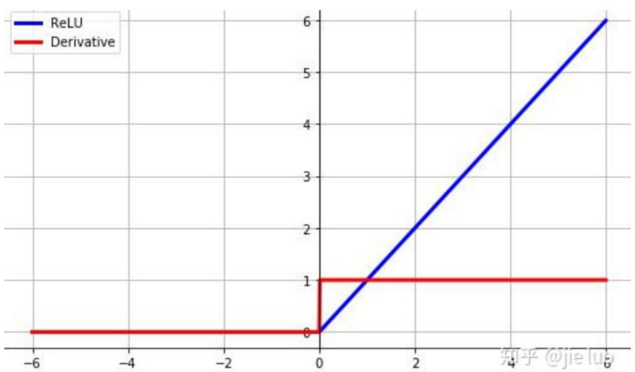
表达式：$\sigma(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$

值域：(-1,1), 当|x|>3时，函数容易饱和。

导数值域：（0，1），当|x|>3时，梯度几乎为0。

5．梯度消失问题

◆ ReLU函数



表达式： $\sigma(x) = max(0, x)$

- 值域：当x<0时，函数值为0，当x>0时，函数值跟x线性增长。
- 导数值域：当x<0时，导函数值为0，当x>0时，导函数值为0。

在神经网络中误差反向传播的迭代公式为 $\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$

其中需要用到激活函数σ(Z^L)的导数误差从输出层反向传播时每层都要乘激活函数导数。这样当激活函数导数值小于 1 时，误差经过每一层传递都会不断衰减，当网络很深时甚至消失

5. 梯度消失问题

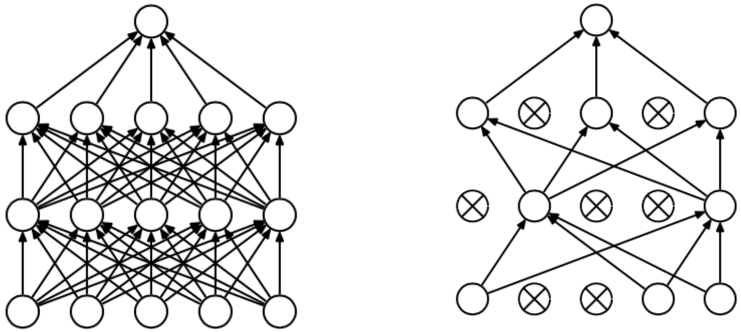
解决梯度梯度消失问题方法

- 选择合适的激活函数
- 用复杂的门结构代替激活函数
- 残差结构

解决过拟合问题方法

选择合适的正则方法

- Dropout



- 损失函数加入适当的正则项

6. 前馈神经网分类问题示例

任务：用前馈神经网实现花的分类

输入：花的 萼片长度、萼片宽度、花瓣长度、花瓣宽度

输出：花的种类

已知：数据集共包含150个实例，3个品种的花各有50个格式如下：

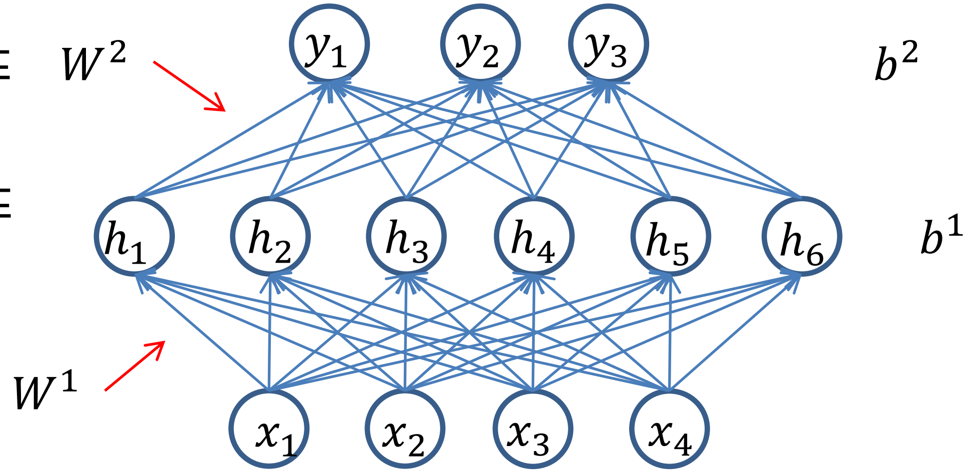
序号	萼片长度	萼片宽度	花瓣长度	花瓣宽度	类别
1	5.1	3.5	1.4	0.2	1
50	5	3.3	1.4	0.2	1
51	7	3.2	4.7	1.4	2
100	5.7	2.8	4.1	1.3	2
101	6.3	3.3	6	2.5	3
150	5.9	3	5.1	1.8	3

将每个类别的前40个，共120个实例组成训练集，其余30个实例组成测试集。

6. 前馈神经网分类问题示例

■ 模型结构

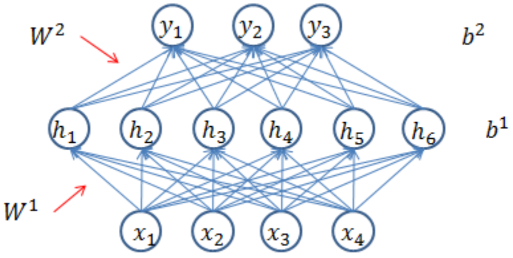
- **构建包含一个隐含层的神经网络DNN模型**
  - 输入层神经元数量：4，对应特征向量维度。
  - 隐含层神经元数量：6，根据经验公式($\sqrt{\boldsymbol{n}+\boldsymbol{m}}+\boldsymbol{a}$)取值。 $\boldsymbol{a} \epsilon [\mathbf{1}, \mathbf{10}]$
  - 输出层神经元数量：3，对应目标类别的数量。
- **输入、输出、参数**
  - x表示模型输入
  - H表示隐含状态
  - y表示模型输出
  - W^1表示输入-隐含层权值矩阵
  - b^1表示隐含层偏置
  - W^2表示隐含-输出层权值矩阵
  - b^2表示输出层偏置



6. 前馈神经网分类问题示例

- **参数包括 W^1，b^1，W^2，b^2**

$$W^1 = \begin{bmatrix} W^1_{(1,1)}, W^1_{(1,2)}, W^1_{(1,3)}, W^1_{(1,4)}, W^1_{(1,5)}, W^1_{(1,6)} \\ W^1_{(2,1)}, W^1_{(2,2)}, W^1_{(2,3)}, W^1_{(2,4)}, W^1_{(2,5)}, W^1_{(2,6)} \\ W^1_{(3,1)}, W^1_{(3,2)}, W^1_{(3,3)}, W^1_{(3,4)}, W^1_{(3,5)}, W^1_{(3,6)} \\ W^1_{(4,1)}, W^1_{(4,2)}, W^1_{(4,3)}, W^1_{(4,4)}, W^1_{(4,5)}, W^1_{(4,6)} \end{bmatrix}$$



$$b^1 = [b^1_1, b^1_2, b^1_3, b^1_4, b^1_5, b^1_6]^T$$

$$W^2 = \begin{bmatrix} W^2_{(1,1)}, W^2_{(1,2)}, W^2_{(1,3)} \\ W^2_{(2,1)}, W^2_{(2,2)}, W^2_{(2,3)} \\ W^2_{(3,1)}, W^2_{(3,2)}, W^2_{(3,3)} \\ W^2_{(4,1)}, W^2_{(4,2)}, W^2_{(4,3)} \\ W^2_{(5,1)}, W^2_{(5,2)}, W^2_{(5,3)} \\ W^2_{(6,1)}, W^2_{(6,2)}, W^2_{(6,3)} \end{bmatrix} \qquad b^2 = \begin{bmatrix} b^2_1 \\ b^2_2 \\ b^2_3 \end{bmatrix}$$

6. 前馈神经网分类问题示例

运算关系:

-- 隐含层

$$h_1 = sigmoid((\sum_{i=1}^{4} x_i W_{(i,1)}^1) + b_1^1)$$

隐含层神经元向量化表示为:

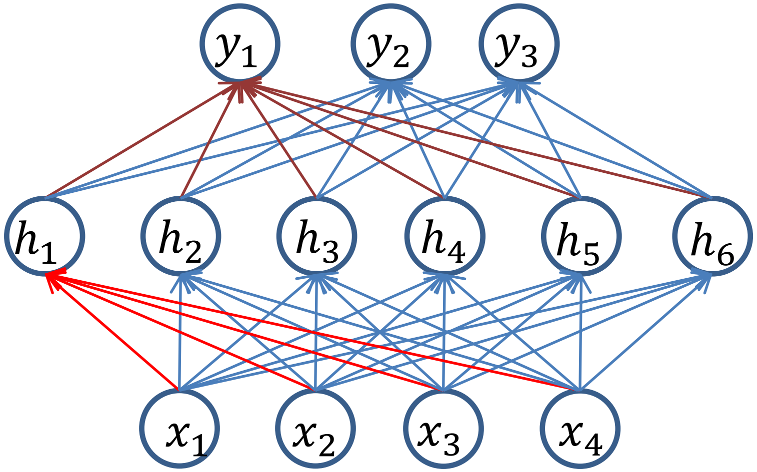
$$H = sigmoid(xW^1 + b^1)$$

--输出层

$$(y_{pred} \sim Z)_1 = \sum_{i=1}^{6} h_i W_{(i,1)}^2 + b_1^2$$

输出层神经元向量化表示为:

$$y_{pred} = softmax(HW^2 + b^2)$$



■ 模型学习

梯度下降法训练模型参数

训练集数据 (x^i , $\hat{y}^i$) 的格式?

定义损失函数

交叉熵损失: $J(\theta; x, y) = -\sum_{j=1}^{3} y_j \log((y_{pred})_j)$ $\boxed{\theta = [W^1, b^1, W^2, b^2]}$

整体损失: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} J(\theta; x^{(i)}, y^{(i)})$

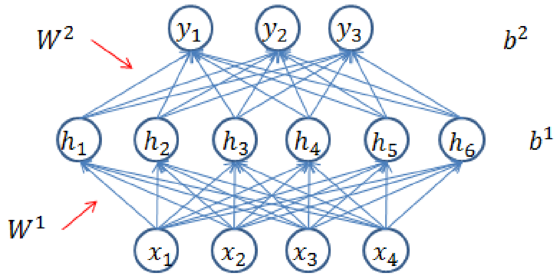
初始化参数: W^1, b^1, W^2, b^2

用BP算法训练参数 W^1, b^1, W^2, b^2。

6. 前馈神经网分类问题示例

训练结果（神经网络权值和阈值）：

```
W1:  [[-0.92051142 -0.26299602 -1.31182587 -0.01975909 -0.59453297 -1.11046791]
 [-0.28832591 -1.20107734 -0.9193843   2.20059323 -1.76573455 -0.85953856]
 [-2.33936191  2.61448216 -0.70324481 -2.55568218  2.42951894 -1.61432779]
 [-2.24077463  2.89018154  0.49342883 -2.72368193  4.8323493   1.31086338]]
b1:  [ -2.00212002  -2.24461389  -0.26081288   0.61751789 -11.42080021
  -0.88119394]
W2:  [[ -0.47563726  -0.80865479  -1.70860052]
 [ -4.02558851   4.80955172   1.79265082]
 [  0.18003793   0.4045147    0.40952846]
 [  8.4482317   -6.87298441  -4.0685277 ]
 [ -2.96541858  -5.34733152  11.1489048 ]
 [  0.08154635   0.21151544  -0.45402744]]
b2:  [ 0.90756476  2.25174689 -2.58430076]
```



6. 前馈神经网分类问题示例

预测

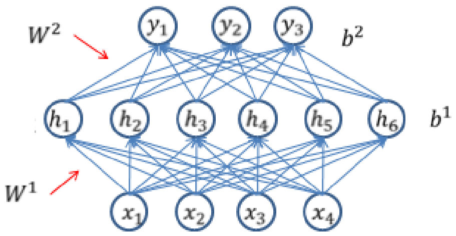
预测数据

序号	萼片长度	萼片宽度	花瓣长度	花瓣宽度	类别
1	5	3.5	1.3	0.3	1,0,0
2	4.5	2.3	1.3	0.3	1,0,0
3	5.5	2.6	4.4	1.2	0,1,0
4	6.1	3	4.6	1.4	0,1,0
5	6.7	3.1	5.6	2.4	0,0,1
6	6.9	3.1	5.1	2.3	0,0,1

x

预测结果

```
[[  9.99998450e-01   1.42261445e-06   1.62947060e-07]
 [  9.99949336e-01   4.81077950e-05   2.52183918e-06]
 [  4.37718809e-05   9.98948872e-01   1.00730266e-03]
 [  4.63805227e-05   9.98428643e-01   1.52486726e-03]
 [  8.70701982e-08   2.18645262e-04   9.99781311e-01]
 [  2.09509579e-07   6.10101502e-04   9.99389648e-01]]
```



附：深度学习框架(开源)

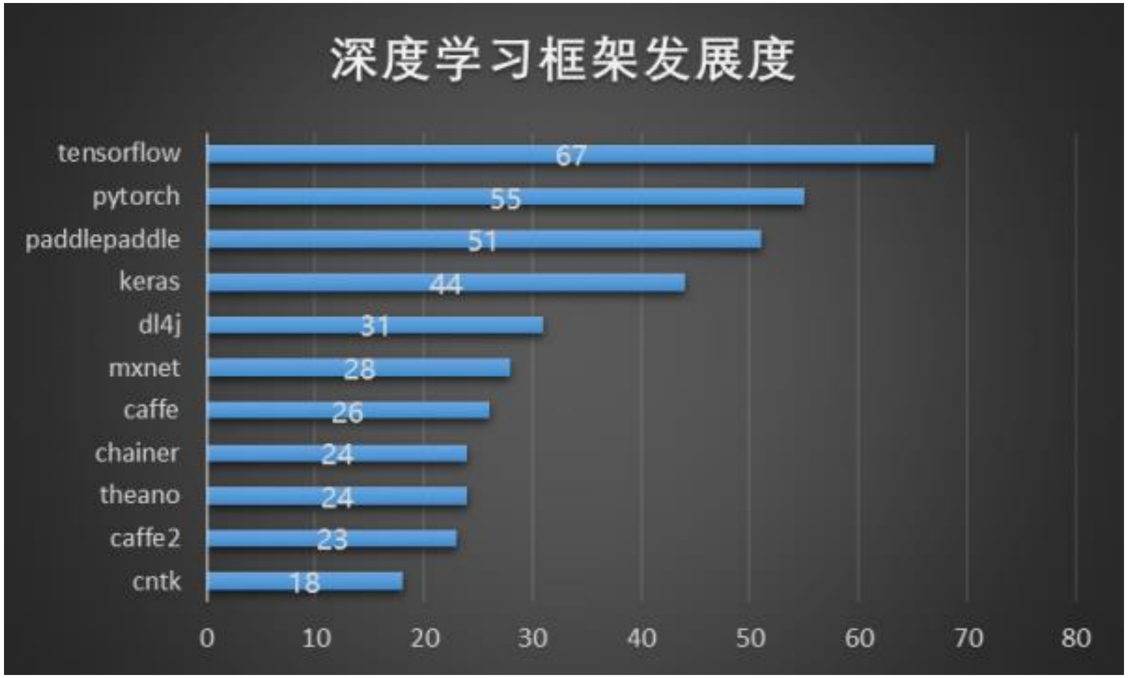
深度学习框架(开源)

十大深度学习框架

框架	发布时间	公司	开发语言
pytorch	2016	facebook	c++、lua
tensorflow	2015	google	c++
paddlepaddle	2016	baidu	c++
mxnet	2017	apache	c++
dl4j	2014	eclipse	java
caffe2	2017	facebook	c++
caffe	2014	berkeley vision	c++
cntk	2016	microsoft	c++
theano	2007	MILA	python
keras	2015	google	python
chainer	2015	chainer	python

深度学习框架(开源)

十大深度学习框架



了解各个深度学习框架请查阅相关资料

参考文献：

李宏毅课程

http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html

邱锡鹏，《神经网络与深度学习》讲义

车万翔，Deep Learning　Lecture 02: Neural Network

在此表示感谢!

谢谢各位！

Q&A