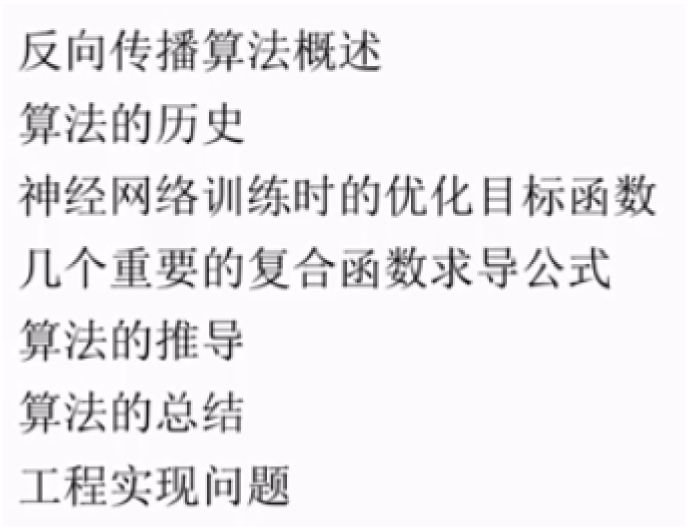
12 人工神经网络2

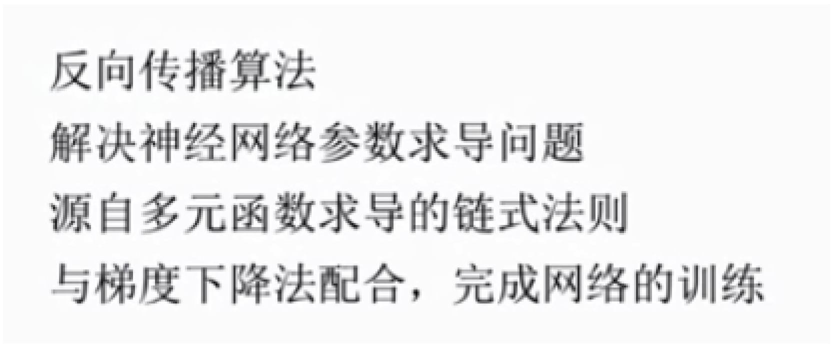
12.1 本集内容简介



反向传播算法概述
算法的历史
神经网络训练时的优化目标函数
几个重要的复合函数求导公式
算法的推导
算法的总结
工程实现问题

简称BP，反向传播算法。

12.2 反向传播算法概述

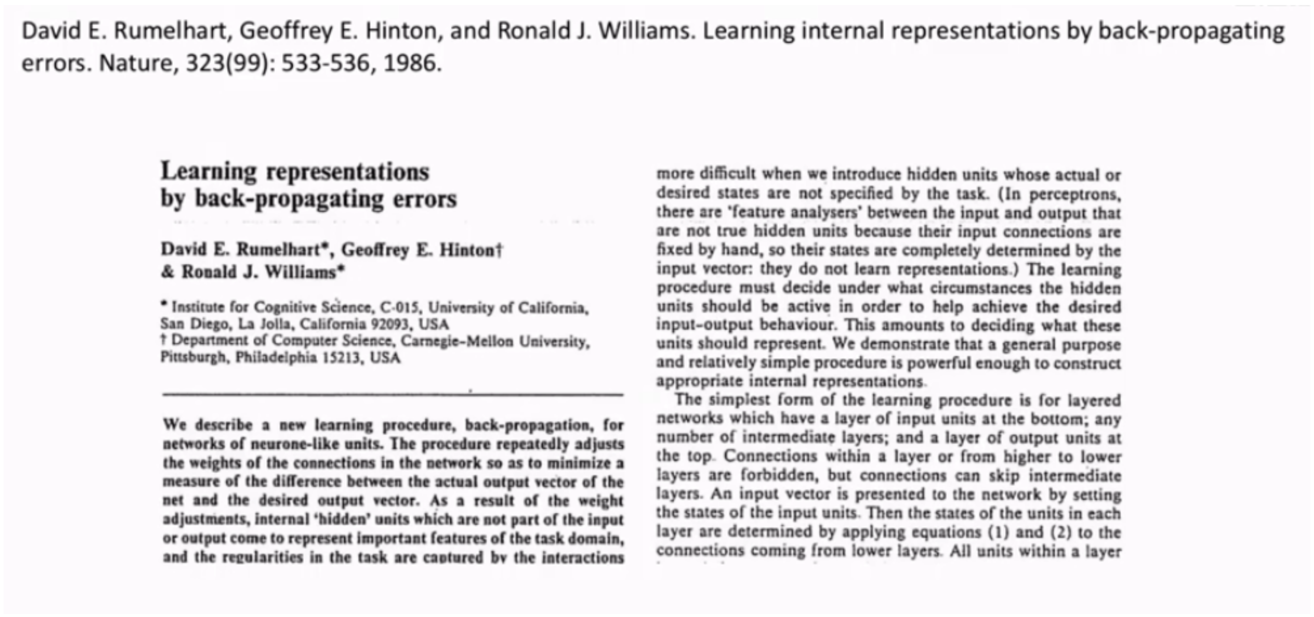

反向传播算法
解决神经网络参数求导问题
源自多元函数求导的链式法则
与梯度下降法配合，完成网络的训练

反向传播算法

难点就在于反向求导

12.3 反向传播算法的历史

我们在机器学习发展历史时已经说过，这里回忆一下。


David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by back-propagating errors. Nature, 323(99): 533-536, 1986.

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

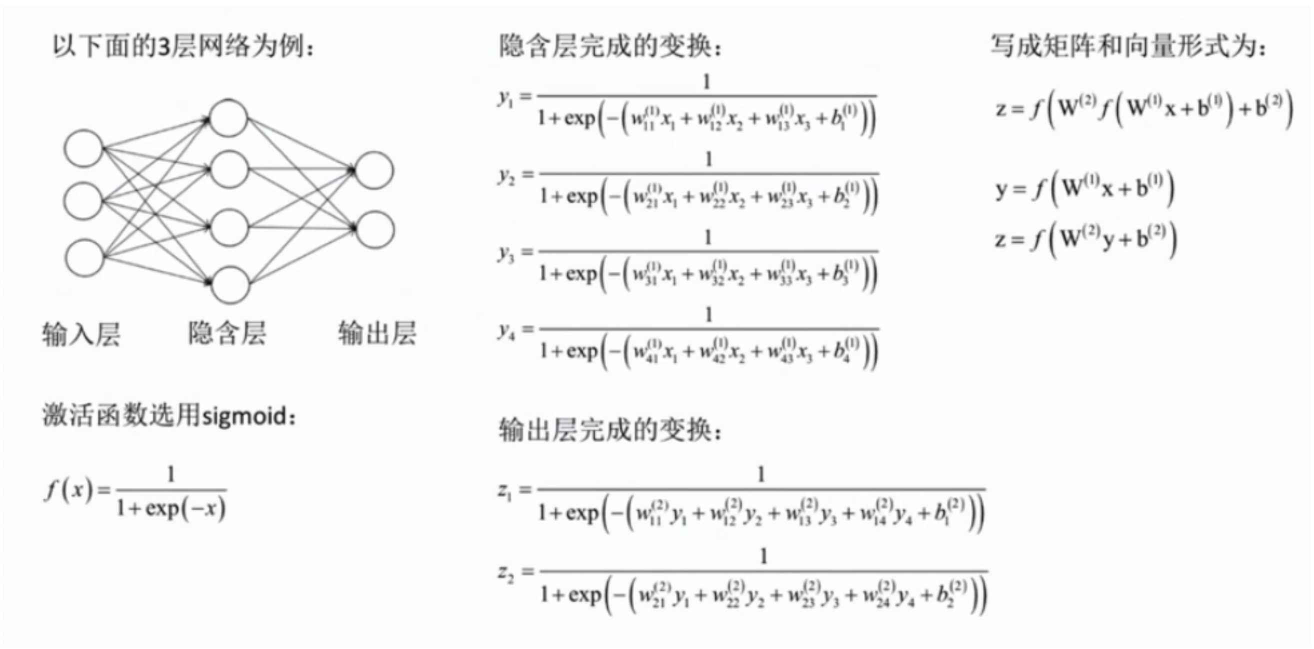
We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions

more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer

深度学习之父

12.4 正向传播算法回顾



以下面的3层网络为例：

激活函数选用sigmoid：

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

隐含层完成的变换：

$$y_1=\frac{1}{1+\exp\left(-\left(w_{11}^{(1)}x_1+w_{12}^{(1)}x_2+w_{13}^{(1)}x_3+b_1^{(1)}\right)\right)}$$

$$y_2=\frac{1}{1+\exp\left(-\left(w_{21}^{(1)}x_1+w_{22}^{(1)}x_2+w_{23}^{(1)}x_3+b_2^{(1)}\right)\right)}$$

$$y_3=\frac{1}{1+\exp\left(-\left(w_{31}^{(1)}x_1+w_{32}^{(1)}x_2+w_{33}^{(1)}x_3+b_3^{(1)}\right)\right)}$$

$$y_4=\frac{1}{1+\exp\left(-\left(w_{41}^{(1)}x_1+w_{42}^{(1)}x_2+w_{43}^{(1)}x_3+b_4^{(1)}\right)\right)}$$

输出层完成的变换：

$$z_1=\frac{1}{1+\exp\left(-\left(w_{11}^{(2)}y_1+w_{12}^{(2)}y_2+w_{13}^{(2)}y_3+w_{14}^{(2)}y_4+b_1^{(2)}\right)\right)}$$

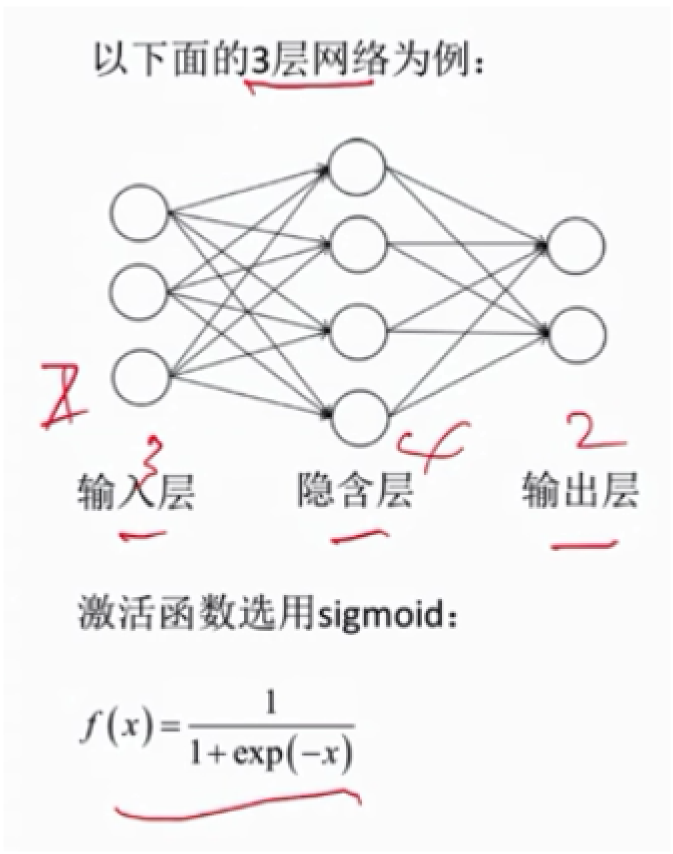
$$z_2=\frac{1}{1+\exp\left(-\left(w_{21}^{(2)}y_1+w_{22}^{(2)}y_2+w_{23}^{(2)}y_3+w_{24}^{(2)}y_4+b_2^{(2)}\right)\right)}$$

写成矩阵和向量形式为：

$$z=f\left(W^{(2)}f\left(W^{(1)}x+b^{(1)}\right)+b^{(2)}\right)$$

$$y=f\left(W^{(1)}x+b^{(1)}\right)$$

$$z=f\left(W^{(2)}y+b^{(2)}\right)$$

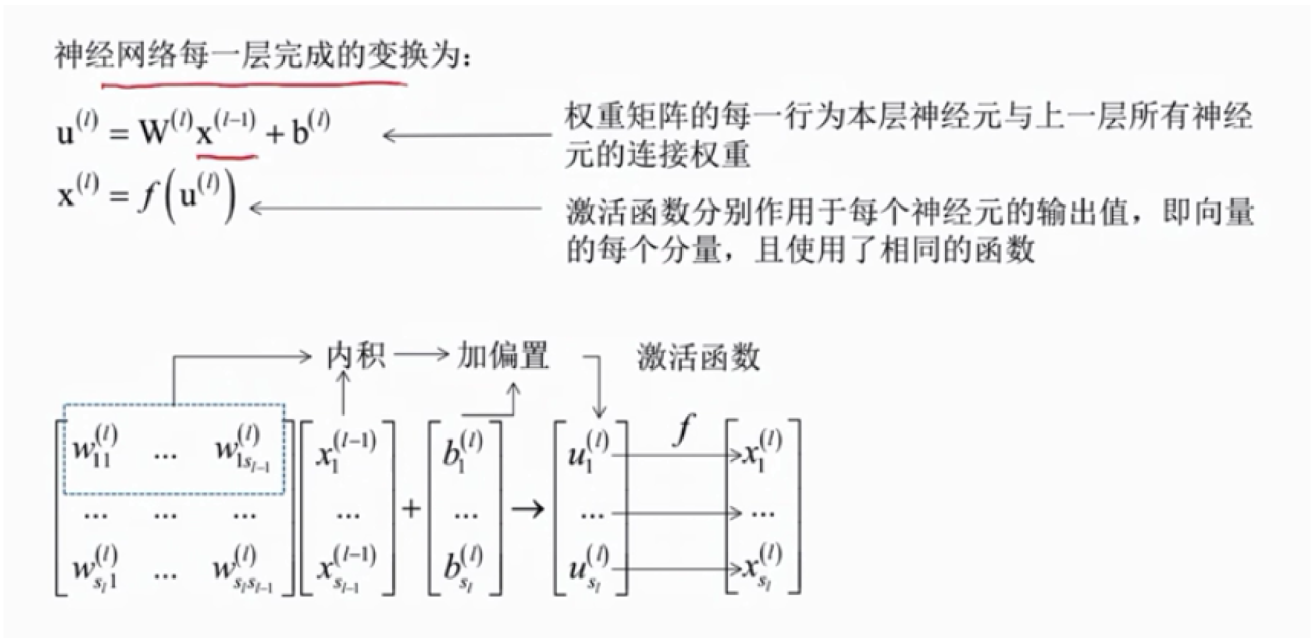


以下面的3层网络为例：

3
4
2
输入层
隐含层
输出层

激活函数选用sigmoid：

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



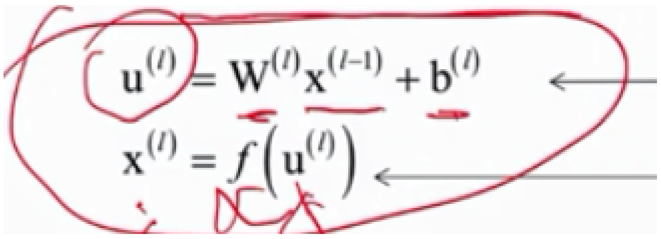
神经网络每一层完成的变换为：

$$\mathbf{u}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}$$ ← 权重矩阵的每一行为本层神经元与上一层所有神经元的连接权重

$$\mathbf{x}^{(l)} = f\left(\mathbf{u}^{(l)}\right)$$ ← 激活函数分别作用于每个神经元的输出值，即向量的每个分量，且使用了相同的函数

$$\begin{bmatrix} w_{11}^{(l)} & \cdots & w_{1s_{l-1}}^{(l)} \\ \cdots & \cdots & \cdots \\ w_{s_l1}^{(l)} & \cdots & w_{s_ls_{l-1}}^{(l)} \end{bmatrix} \begin{bmatrix} x_1^{(l-1)} \\ \cdots \\ x_{s_{l-1}}^{(l-1)} \end{bmatrix} + \begin{bmatrix} b_1^{(l)} \\ \cdots \\ b_{s_l}^{(l)} \end{bmatrix} \rightarrow \begin{bmatrix} u_1^{(l)} \\ \cdots \\ u_{s_l}^{(l)} \end{bmatrix} \xrightarrow{f} \begin{bmatrix} x_1^{(l)} \\ \cdots \\ x_{s_l}^{(l)} \end{bmatrix}$$

重要公式

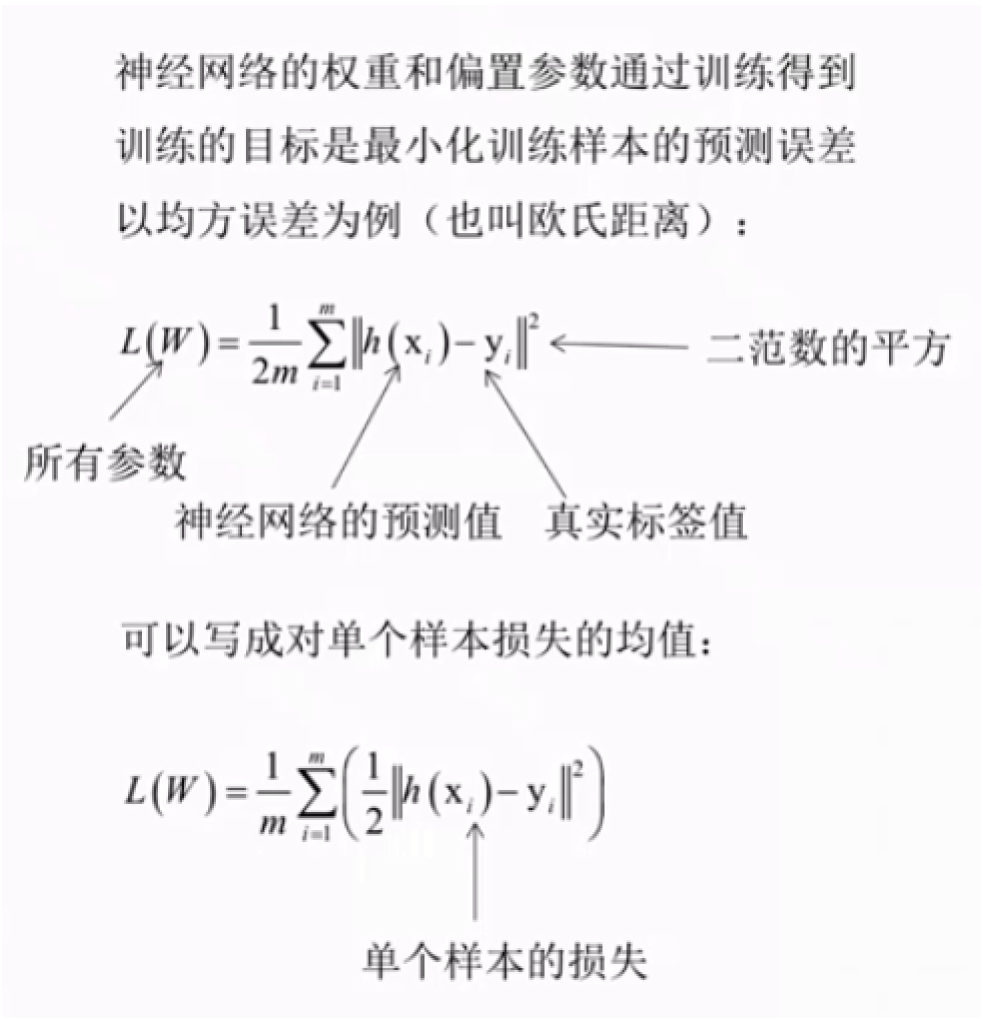


$$\mathbf{u}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{x}^{(l)} = f\left(\mathbf{u}^{(l)}\right)$$

12.5 优化目标函数

多层的符合函数，求解非常困难。



神经网络的权重和偏置参数通过训练得到

训练的目标是最小化训练样本的预测误差

以均方误差为例（也叫欧氏距离）：

$$L(W)=\frac{1}{2m}\sum_{i=1}^{m}\left\|h(\mathrm{x}_i)-\mathrm{y}_i\right\|^2 \longleftarrow \text{二范数的平方}$$

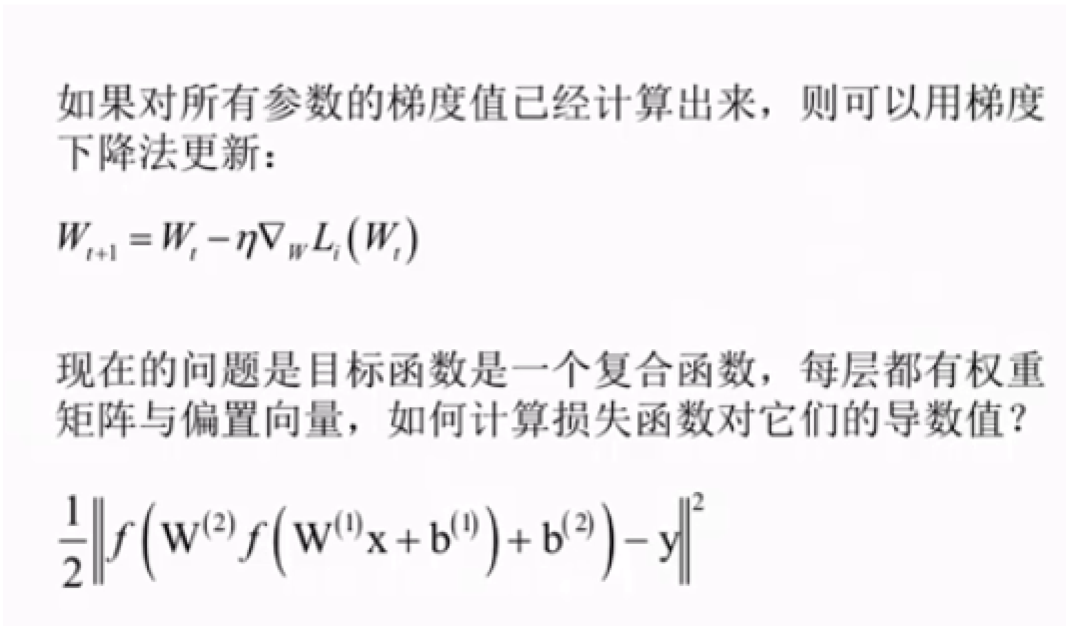
所有参数

神经网络的预测值　真实标签值

可以写成对单个样本损失的均值：

$$L(W)=\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h(\mathrm{x}_i)-\mathrm{y}_i\right\|^2\right)$$

单个样本的损失



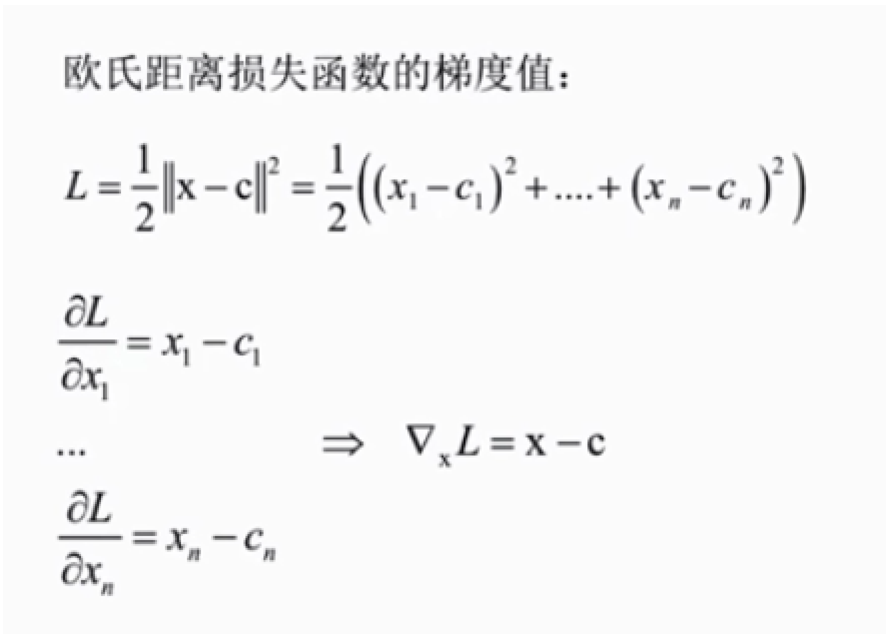
如果对所有参数的梯度值已经计算出来，则可以用梯度下降法更新：

$$W_{t+1}=W_t-\eta\nabla_W L_i(W_t)$$

现在的问题是目标函数是一个复合函数，每层都有权重矩阵与偏置向量，如何计算损失函数对它们的导数值？

$$\frac{1}{2}\left\|f\left(\mathrm{W}^{(2)}f\left(\mathrm{W}^{(1)}\mathrm{x}+\mathrm{b}^{(1)}\right)+\mathrm{b}^{(2)}\right)-\mathrm{y}\right\|^2$$

12.6 欧氏距离的求导公式



欧氏距离损失函数的梯度值：

$$L = \frac{1}{2}\|\mathrm{x}-\mathrm{c}\|^2 = \frac{1}{2}\left((x_1-c_1)^2+....+(x_n-c_n)^2\right)$$

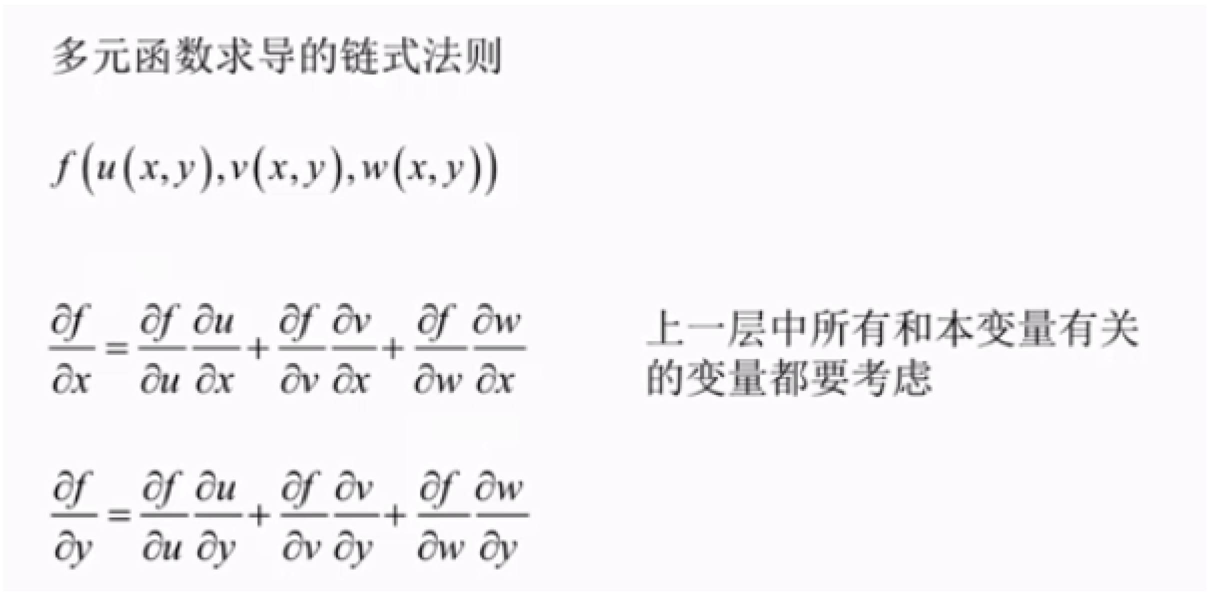
$$\frac{\partial L}{\partial x_1} = x_1 - c_1$$

$$\cdots \qquad \Rightarrow \quad \nabla_{\mathrm{x}} L = \mathrm{x}-\mathrm{c}$$

$$\frac{\partial L}{\partial x_n} = x_n - c_n$$

前面乘以一个二分之一

12.7 链式法则



多元函数求导的链式法则

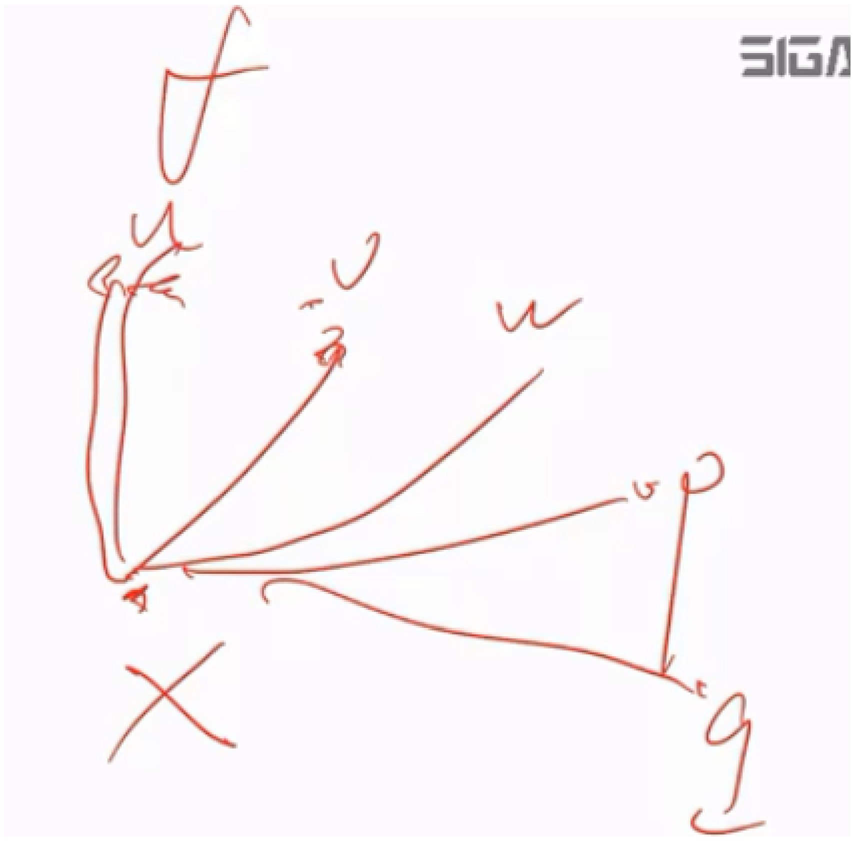
$$f\left(u(x,y),v(x,y),w(x,y)\right)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial x} + \frac{\partial f}{\partial w}\frac{\partial w}{\partial x}$$

上一层中所有和本变量有关的变量都要考虑

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial y} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial y} + \frac{\partial f}{\partial w}\frac{\partial w}{\partial y}$$

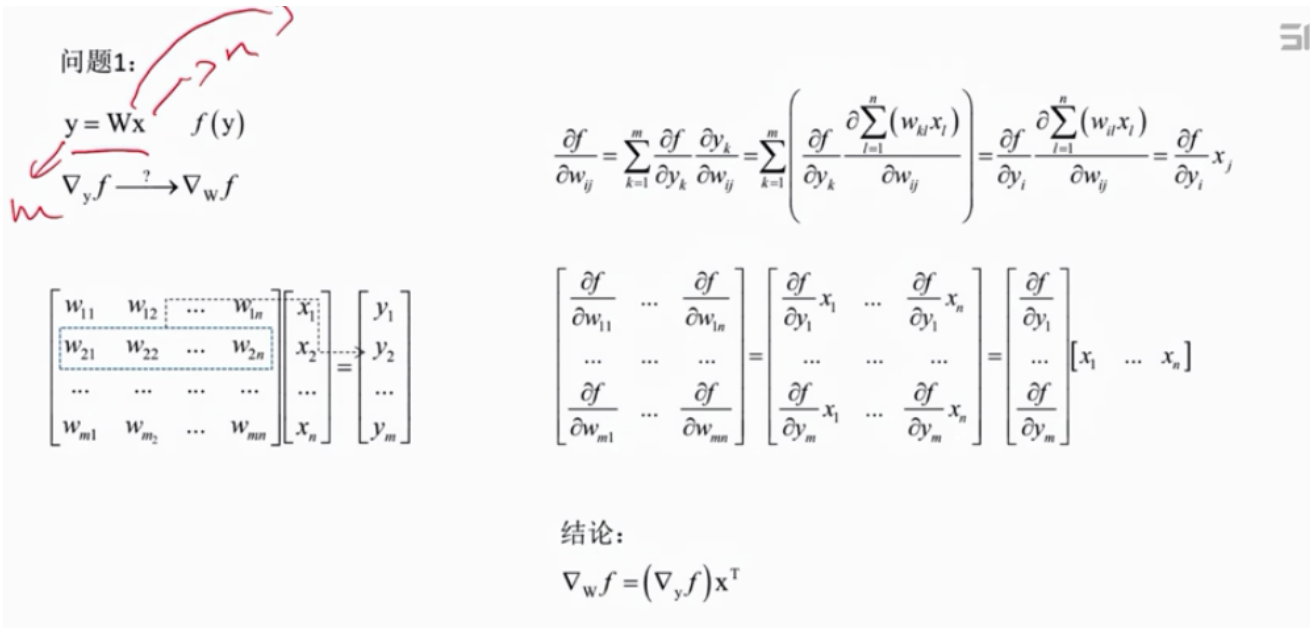
对较复杂的函数可以作图来求解，避免错漏



12.8 基础求导公式

问题1

多元函数的链式法则



问题1:

$$y = Wx \qquad f(y)$$

$$\nabla_y f \xrightarrow{?} \nabla_W f$$

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m1} & w_{m_2} & \cdots & w_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix}$$

$$\frac{\partial f}{\partial w_{ij}} = \sum_{k=1}^{m} \frac{\partial f}{\partial y_k} \frac{\partial y_k}{\partial w_{ij}} = \sum_{k=1}^{m} \left(\frac{\partial f}{\partial y_k} \frac{\partial \sum_{l=1}^{n} (w_{kl} x_l)}{\partial w_{ij}} \right) = \frac{\partial f}{\partial y_i} \frac{\partial \sum_{l=1}^{n} (w_{il} x_l)}{\partial w_{ij}} = \frac{\partial f}{\partial y_i} x_j$$

$$\begin{bmatrix} \frac{\partial f}{\partial w_{11}} & \cdots & \frac{\partial f}{\partial w_{1n}} \\ \cdots & \cdots & \cdots \\ \frac{\partial f}{\partial w_{m1}} & \cdots & \frac{\partial f}{\partial w_{mn}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial y_1} x_1 & \cdots & \frac{\partial f}{\partial y_1} x_n \\ \cdots & \cdots & \cdots \\ \frac{\partial f}{\partial y_m} x_1 & \cdots & \frac{\partial f}{\partial y_m} x_n \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial y_1} \\ \cdots \\ \frac{\partial f}{\partial y_m} \end{bmatrix} \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$$

结论:

$$\nabla_W f = \left(\nabla_y f \right) x^T$$

列向量就是f对y的梯度值。

结论：

$$\nabla_{\mathbf{W}} f = \left(\nabla_{\mathbf{y}} f\right) \mathbf{x}^{\mathrm{T}}$$

问题2

和前面的刚好相反

- y 因变量
- W 常数
- x 自变量

$$\underline{\mathbf{y} = \mathbf{W}\mathbf{x}} \qquad f(\mathbf{y})$$

$$\nabla_{\mathbf{y}} f \xrightarrow{?} \nabla_{\mathbf{x}} f$$

这里的情况比前面复杂一些

问题2：

$$\underline{\mathbf{y} = \mathbf{W}\mathbf{x}} \qquad f(\mathbf{y})$$

$$\nabla_{\mathbf{y}} f \xrightarrow{?} \nabla_{\mathbf{x}} f$$

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m1} & w_{m_2} & \cdots & w_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix}$$

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial f}{\partial y_j} \frac{\partial y_j}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial f}{\partial y_j} \frac{\partial \left(\sum_{k=1}^{n} w_{jk} x_k \right)}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial f}{\partial y_j} w_{ji} = \left[w_{1i} \quad \cdots \quad w_{mi} \right] \nabla_{\mathbf{y}} f$$

结论：

$$\nabla_{\mathbf{x}} f = \mathbf{W}^{\mathrm{T}} \nabla_{\mathbf{y}} f$$

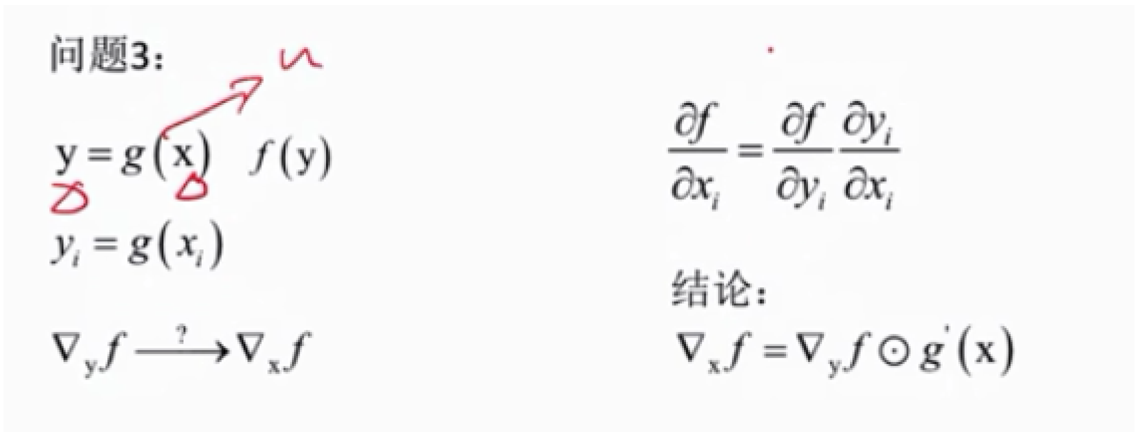
红色框中的分子只有一项是和xj 有关的，所以求导之后，只剩下w

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial f}{\partial y_j} \frac{\partial y_j}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial f}{\partial y_j} \boxed{\frac{\partial \left(\sum_{k=1}^{n} w_{jk} x_k \right)}{\partial x_i}} = \sum_{j=1}^{m} \frac{\partial f}{\partial y_j} w_{ji} = \left[w_{1i} \quad \cdots \quad w_{mi} \right] \nabla_{\mathbf{y}} f$$

结论：

$$\nabla_{\mathbf{x}} f = \mathbf{W}^{\mathrm{T}} \nabla_{\mathbf{y}} f$$

问题3



问题3:

$$\overset{\partial}{y} = g(\overset{\partial}{x}) \quad f(y) \qquad n$$

$$y_i = g(x_i)$$

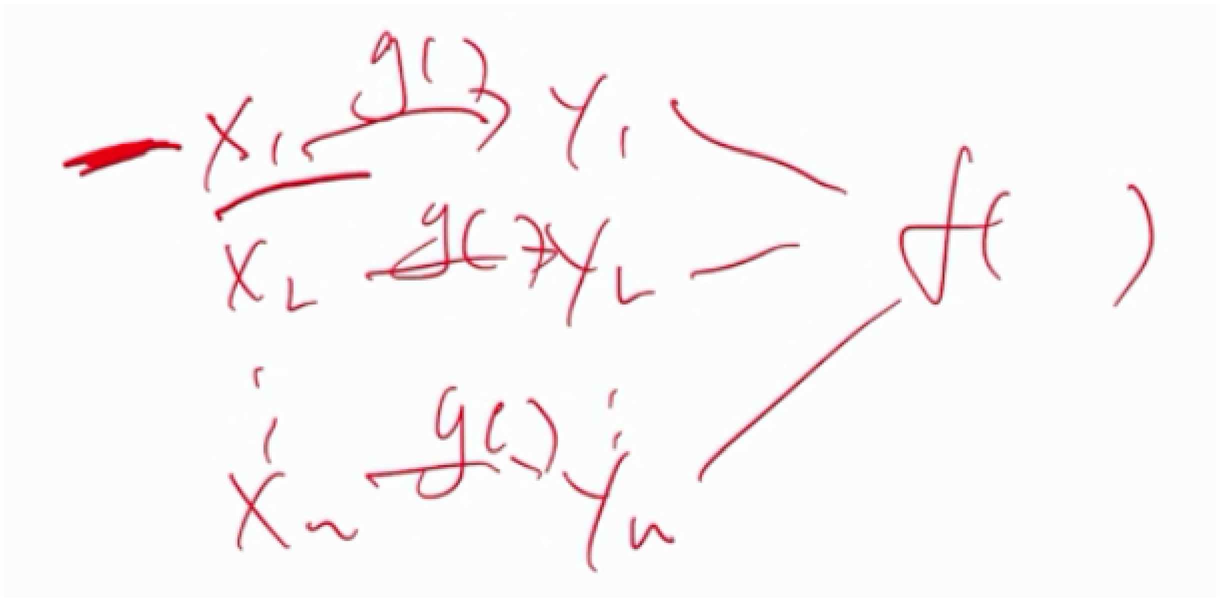
$$\nabla_y f \xrightarrow{?} \nabla_x f$$

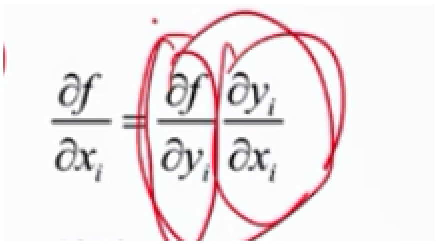
$$\frac{\partial f}{\partial x_i} = \frac{\partial f}{\partial y_i}\frac{\partial y_i}{\partial x_i}$$

结论:

$$\nabla_x f = \nabla_y f \odot g'(x)$$

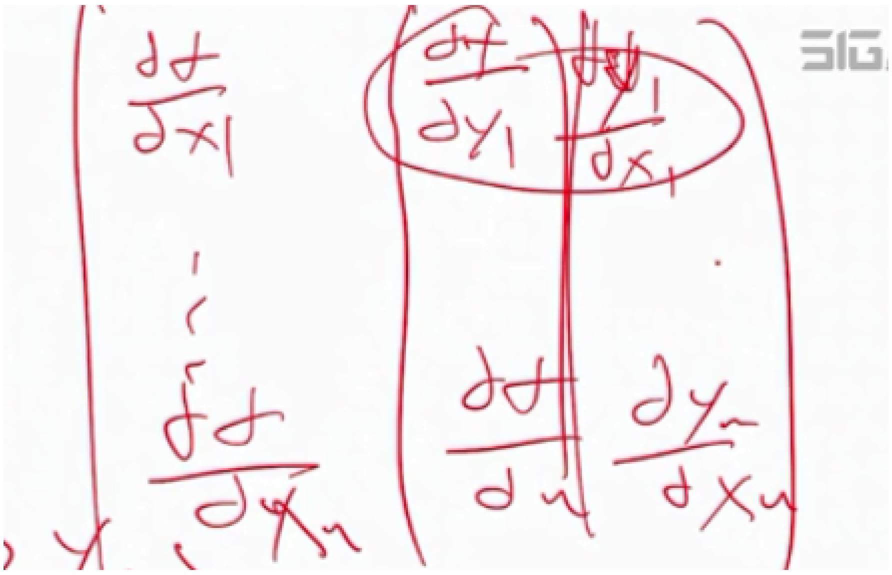
都要通过激活函数来映射



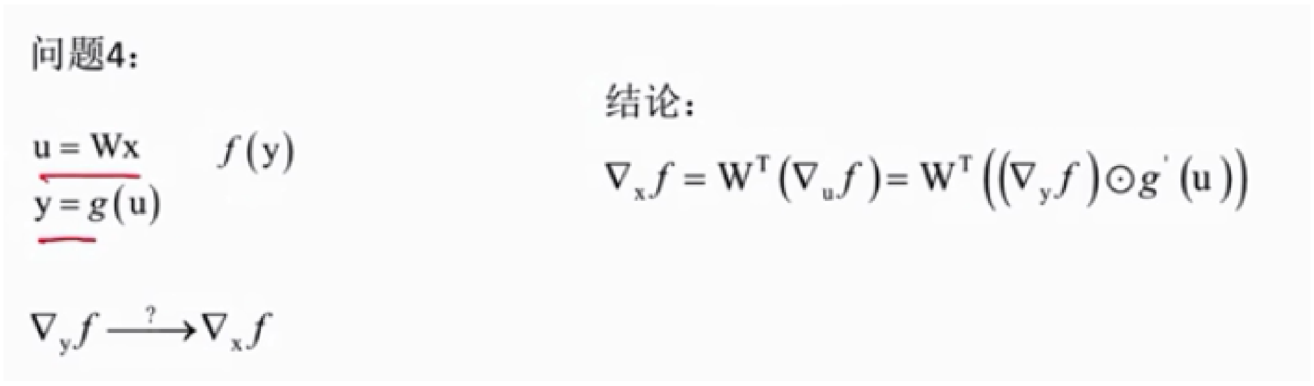


$$\frac{\partial f}{\partial x_i} = \boxed{\frac{\partial f}{\partial y_i}}\boxed{\frac{\partial y_i}{\partial x_i}}$$

看成两个向量对应相乘



问题4



问题4:

$\underline{u = Wx}$ $\quad f(y)$

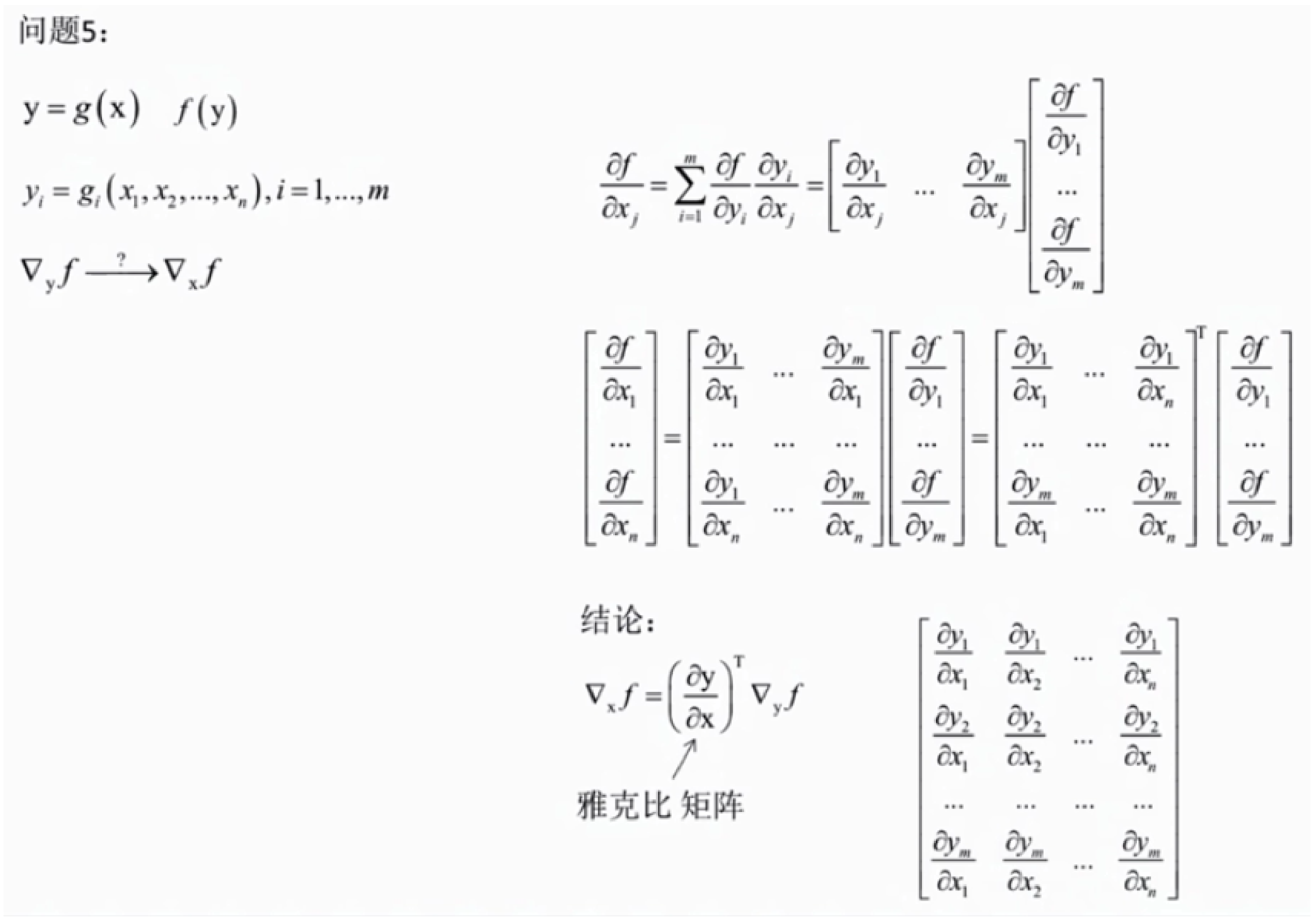
$\underline{y = g(u)}$

$\nabla_y f \xrightarrow{?} \nabla_x f$

结论:

$$\nabla_x f = W^T(\nabla_u f) = W^T\left((\nabla_y f)\odot g'(u)\right)$$

问题5


问题5:

$$y = g(x) \quad f(y)$$

$$y_i = g_i(x_1, x_2, \dots, x_n), i = 1, \dots, m$$

$$\nabla_y f \xrightarrow{?} \nabla_x f$$

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{m} \frac{\partial f}{\partial y_i}\frac{\partial y_i}{\partial x_j} = \begin{bmatrix} \frac{\partial y_1}{\partial x_j} & \cdots & \frac{\partial y_m}{\partial x_j} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial y_1} \\ \cdots \\ \frac{\partial f}{\partial y_m} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \cdots & \cdots & \cdots \\ \frac{\partial y_1}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial y_1} \\ \cdots \\ \frac{\partial f}{\partial y_m} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}^T \begin{bmatrix} \frac{\partial f}{\partial y_1} \\ \cdots \\ \frac{\partial f}{\partial y_m} \end{bmatrix}$$

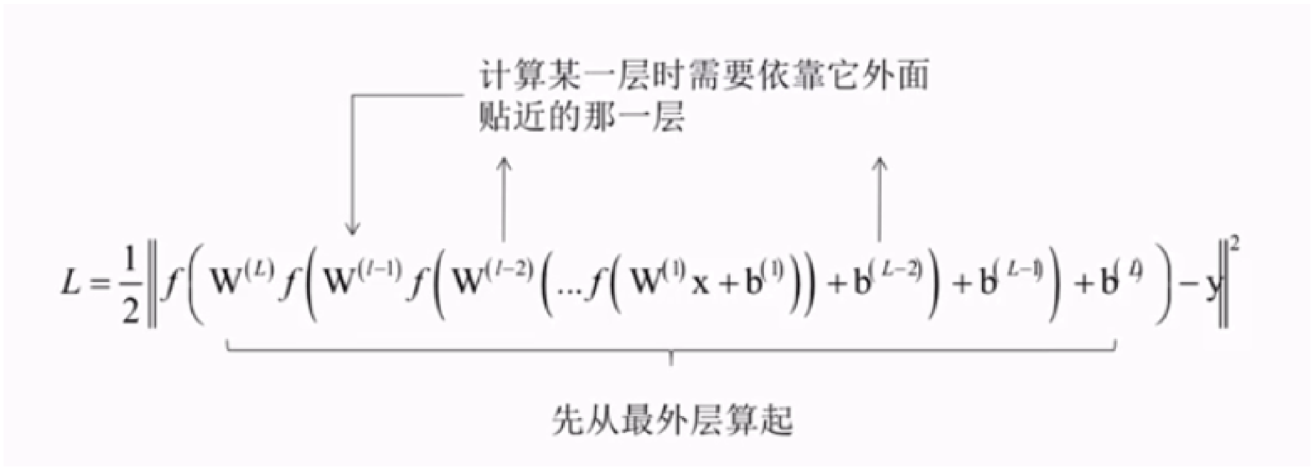
结论:

$$\nabla_x f = \underbrace{\left(\frac{\partial y}{\partial x}\right)}_{\text{雅克比 矩阵}}^T \nabla_y f$$

$$\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

这其实就是雅可比矩阵的转置

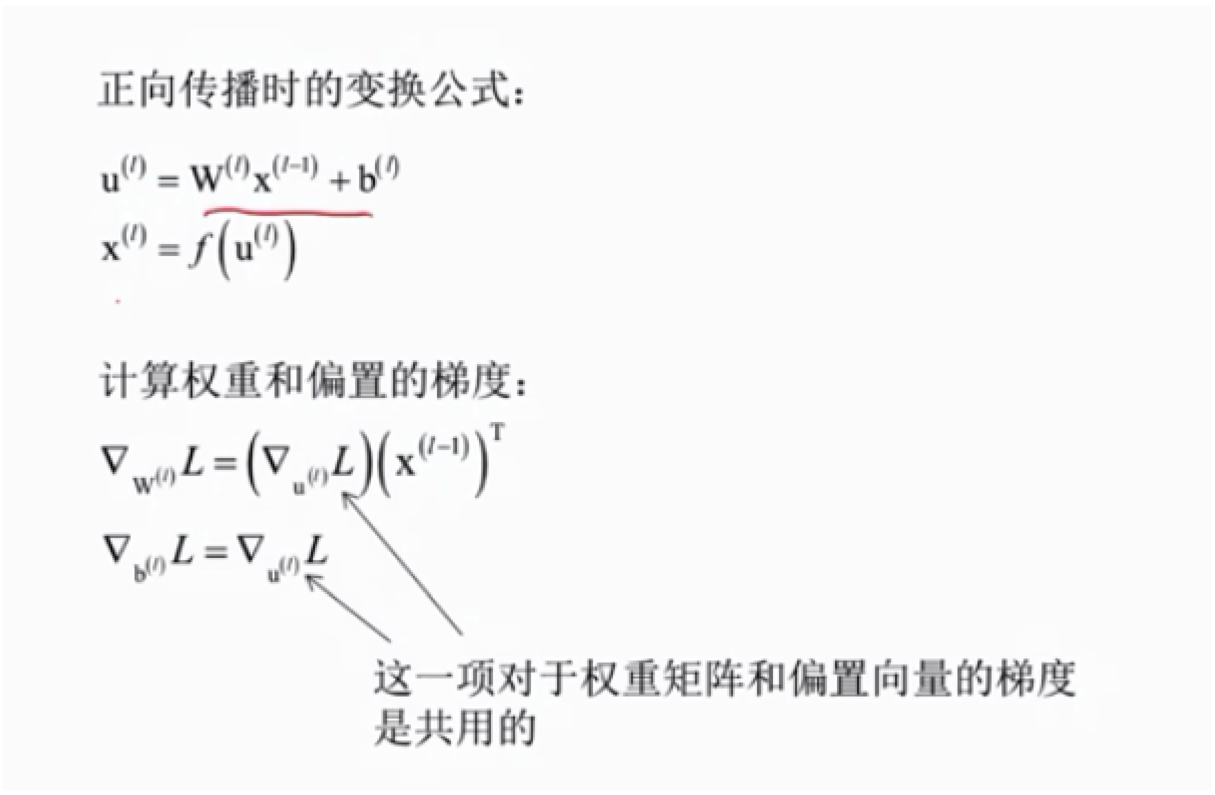
12.9 求导的整体思路


计算某一层时需要依靠它外面贴近的那一层

$$L = \frac{1}{2}\left\| \underbrace{f\left(W^{(L)} f\left(W^{(l-1)} f\left(W^{(l-2)}\left(\dots f\left(W^{(1)}x + b^{(1)}\right)\right) + b^{(L-2)}\right) + b^{(L-1)}\right) + b^{(L)}\right)}_{\text{先从最外层算起}} - y \right\|^2$$

求导就是要从外向内，逐层求导。这就是反向传播算法的基本原理。

12.10 权重和偏置的求导公式



正向传播时的变换公式：

$$u^{(l)} = \underline{W^{(l)}x^{(l-1)}} + b^{(l)}$$
$$x^{(l)} = f\left(u^{(l)}\right)$$

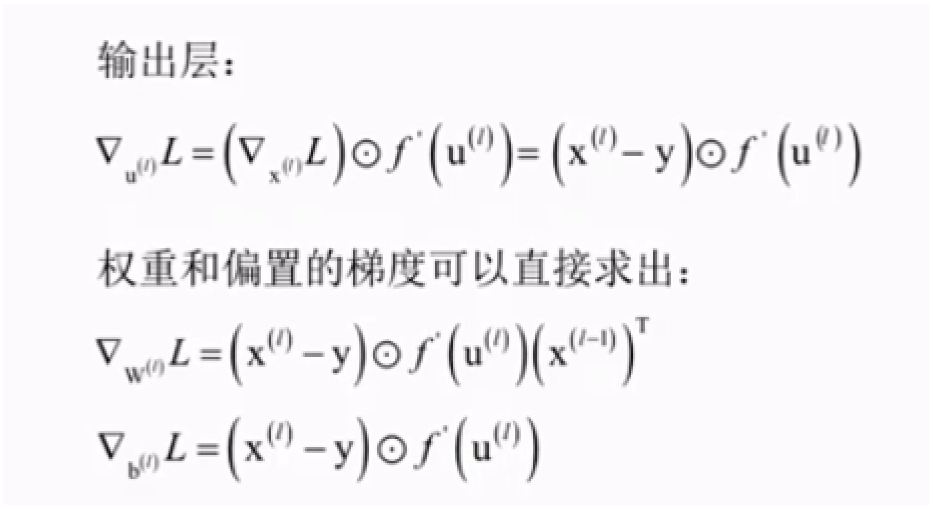
计算权重和偏置的梯度：

$$\nabla_{W^{(l)}} L = \left(\nabla_{u^{(l)}} L\right)\left(x^{(l-1)}\right)^{T}$$
$$\nabla_{b^{(l)}} L = \nabla_{u^{(l)}} L$$

这一项对于权重矩阵和偏置向量的梯度是共用的

12.11 输出层的求导

输出层



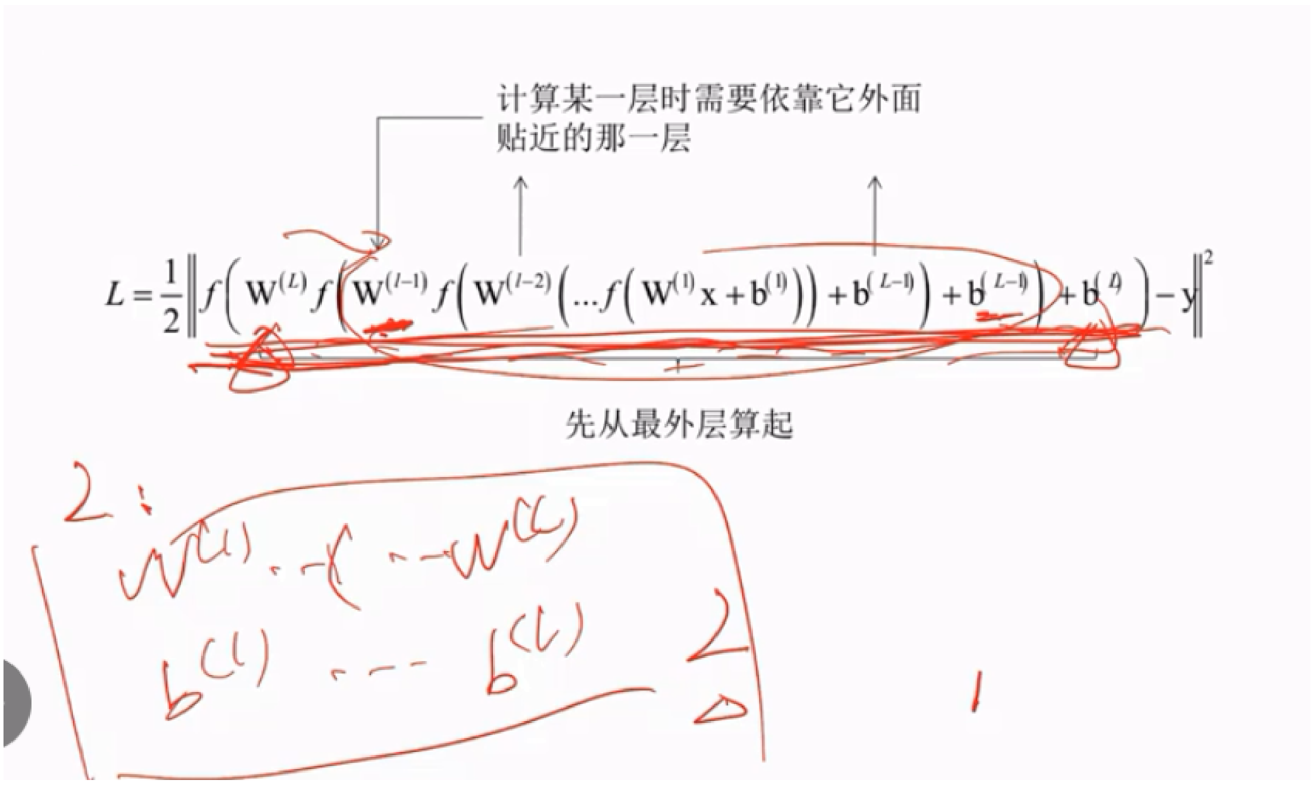
输出层：

$$\nabla_{u^{(l)}} L = \left(\nabla_{x^{(l)}} L\right) \odot f'\left(u^{(l)}\right) = \left(x^{(l)} - y\right) \odot f'\left(u^{(l)}\right)$$

权重和偏置的梯度可以直接求出：

$$\nabla_{W^{(l)}} L = \left(x^{(l)} - y\right) \odot f'\left(u^{(l)}\right)\left(x^{(l-1)}\right)^{T}$$
$$\nabla_{b^{(l)}} L = \left(x^{(l)} - y\right) \odot f'\left(u^{(l)}\right)$$

我们首先考虑最外层



计算某一层时需要依靠它外面贴近的那一层

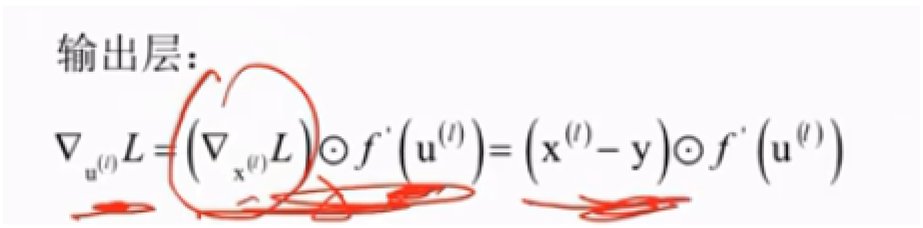
$$L=\frac{1}{2}\left\| f\left(W^{(L)} f\left(W^{(l-1)} f\left(W^{(l-2)}\left(\dots f\left(W^{(1)}x+b^{(1)}\right)\right)+b^{(L-1)}\right)+b^{(L-1)}\right)+b^{(L)}\right)-y\right\|^{2}$$

先从最外层算起

L:

$W^{(1)} \dots W^{(L)}$

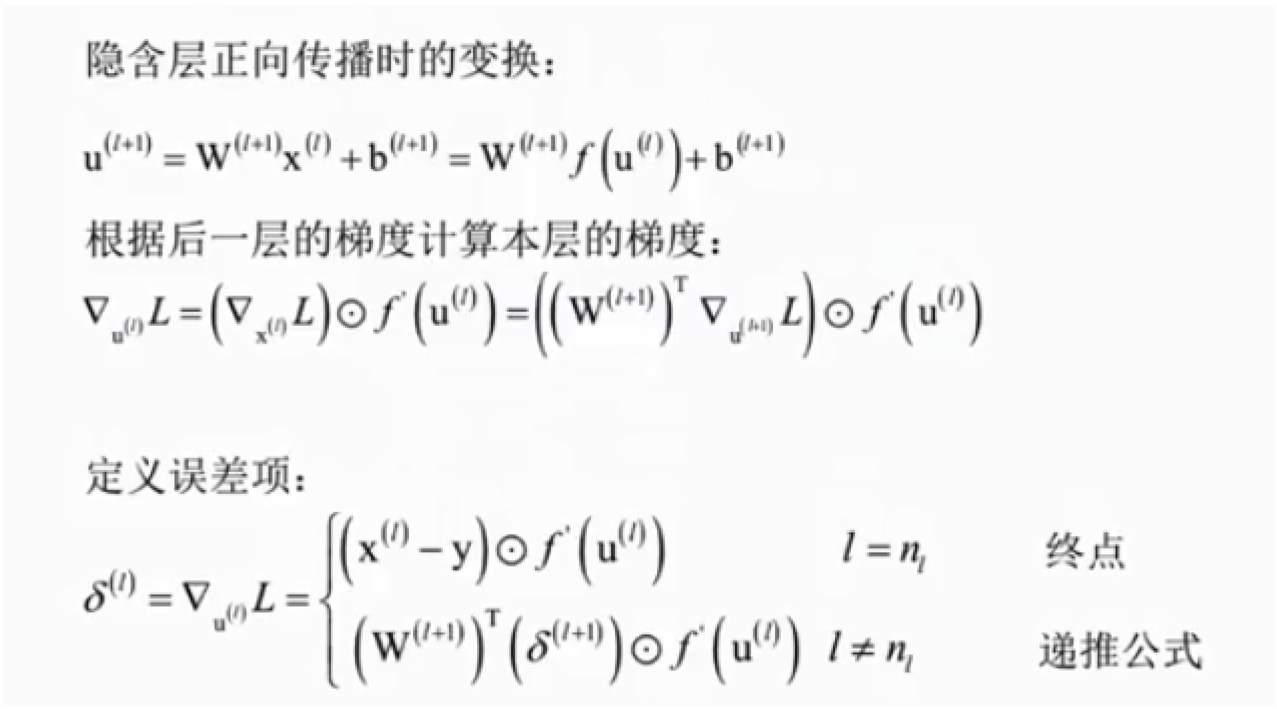
$b^{(1)} \dots b^{(L)}$



输出层:

$$\nabla_{u^{(l)}}L=\left(\nabla_{x^{(l)}}L\right)\odot f'\left(u^{(l)}\right)=\left(x^{(l)}-y\right)\odot f'\left(u^{(l)}\right)$$

隐含层



隐含层正向传播时的变换：

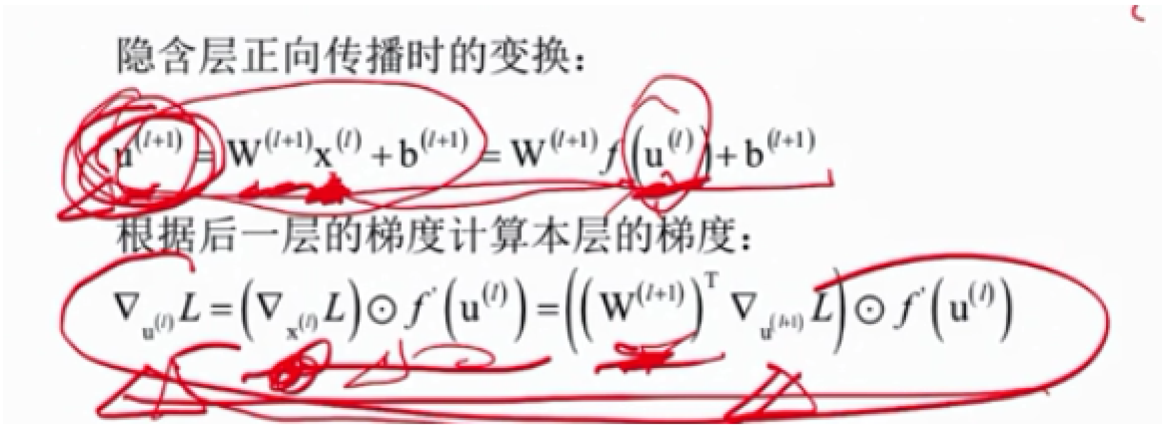
$$u^{(l+1)} = W^{(l+1)}x^{(l)} + b^{(l+1)} = W^{(l+1)} f\left(u^{(l)}\right) + b^{(l+1)}$$

根据后一层的梯度计算本层的梯度：

$$\nabla_{u^{(l)}} L = \left(\nabla_{x^{(l)}} L\right) \odot f'\left(u^{(l)}\right) = \left(\left(W^{(l+1)}\right)^T \nabla_{u^{(l+1)}} L\right) \odot f'\left(u^{(l)}\right)$$

定义误差项：

$$\delta^{(l)} = \nabla_{u^{(l)}} L = \begin{cases} \left(x^{(l)} - y\right) \odot f'\left(u^{(l)}\right) & l = n_l \quad \text{终点} \\ \left(W^{(l+1)}\right)^T \left(\delta^{(l+1)}\right) \odot f'\left(u^{(l)}\right) & l \neq n_l \quad \text{递推公式} \end{cases}$$



隐含层正向传播时的变换：

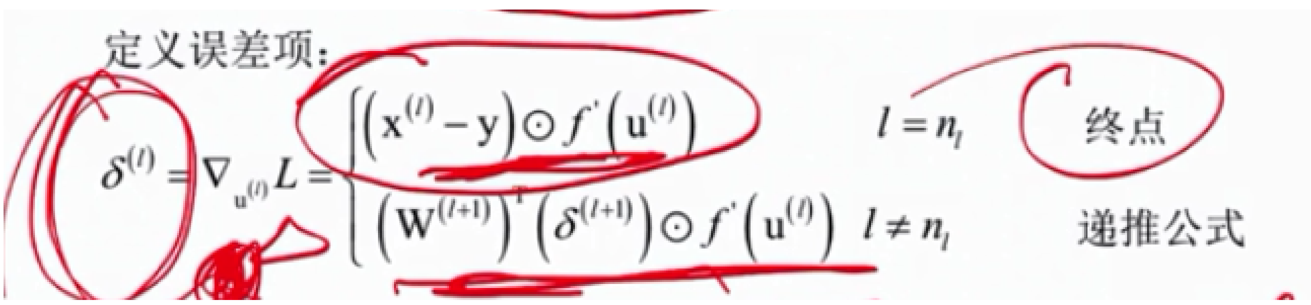
$$u^{(l+1)} = W^{(l+1)}x^{(l)} + b^{(l+1)} = W^{(l+1)} f\left(u^{(l)}\right) + b^{(l+1)}$$

根据后一层的梯度计算本层的梯度：

$$\nabla_{u^{(l)}} L = \left(\nabla_{x^{(l)}} L\right) \odot f'\left(u^{(l)}\right) = \left(\left(W^{(l+1)}\right)^T \nabla_{u^{(l+1)}} L\right) \odot f'\left(u^{(l)}\right)$$

从后往前算

从终点开始算起



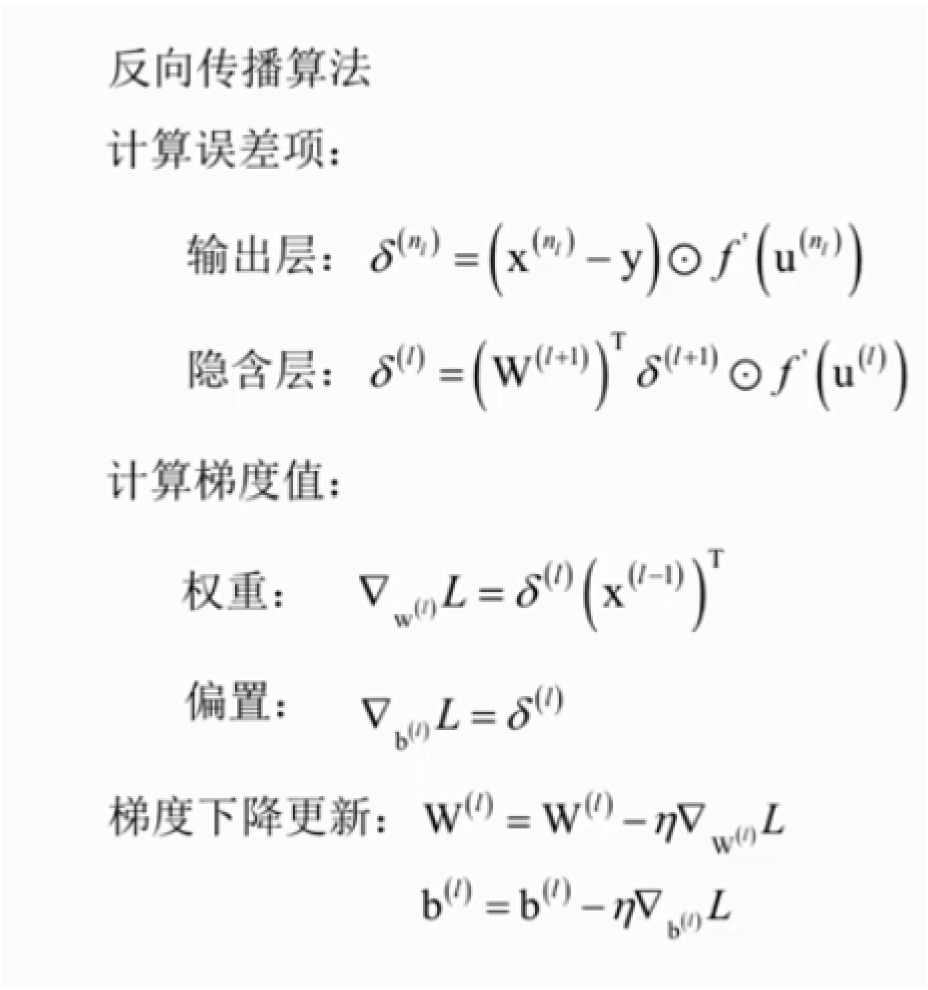
定义误差项：

$$\delta^{(l)} = \nabla_{u^{(l)}} L = \begin{cases} \left(x^{(l)} - y\right) \odot f'\left(u^{(l)}\right) & l = n_l \quad \text{终点} \\ \left(W^{(l+1)}\right)^T \left(\delta^{(l+1)}\right) \odot f'\left(u^{(l)}\right) & l \neq n_l \quad \text{递推公式} \end{cases}$$

梯度值算出来之后，我们就可以用梯度下降法来更新。

12.13 完整的算法

根据前面的推导，我们就可以得到完整的反向穿反驳



反向传播算法

计算误差项：

输出层：$\delta^{(n_l)} = \left(\mathrm{x}^{(n_l)} - \mathrm{y}\right) \odot f'\left(\mathrm{u}^{(n_l)}\right)$

隐含层：$\delta^{(l)} = \left(\mathrm{W}^{(l+1)}\right)^{\mathrm{T}} \delta^{(l+1)} \odot f'\left(\mathrm{u}^{(l)}\right)$

计算梯度值：

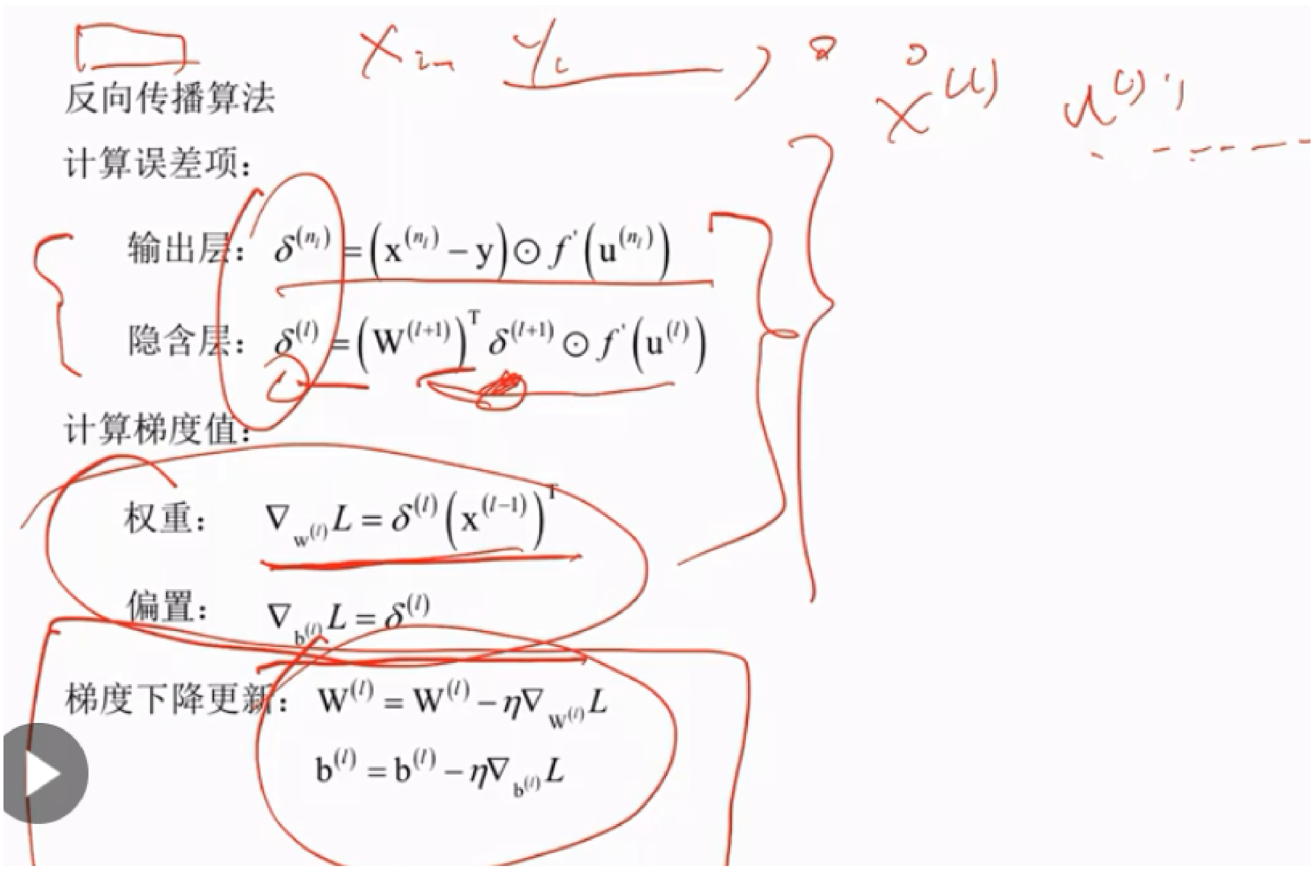
权重：$\nabla_{\mathrm{W}^{(l)}} L = \delta^{(l)} \left(\mathrm{x}^{(l-1)}\right)^{\mathrm{T}}$

偏置：$\nabla_{\mathrm{b}^{(l)}} L = \delta^{(l)}$

梯度下降更新：$\mathrm{W}^{(l)} = \mathrm{W}^{(l)} - \eta \nabla_{\mathrm{W}^{(l)}} L$

$\mathrm{b}^{(l)} = \mathrm{b}^{(l)} - \eta \nabla_{\mathrm{b}^{(l)}} L$

核心在前面圈出


反向传播算法

计算误差项：

输出层：$\delta^{(n_l)} = \left(\mathrm{x}^{(n_l)} - \mathrm{y}\right) \odot f'\left(\mathrm{u}^{(n_l)}\right)$

隐含层：$\delta^{(l)} = \left(\mathrm{W}^{(l+1)}\right)^{\mathrm{T}} \delta^{(l+1)} \odot f'\left(\mathrm{u}^{(l)}\right)$

计算梯度值：

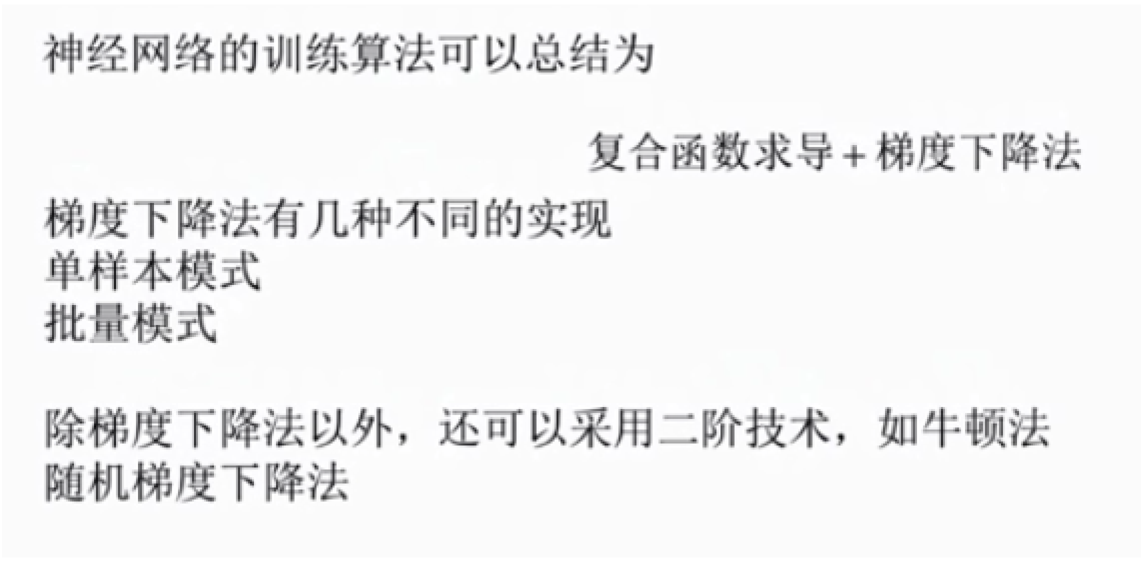
权重：$\nabla_{\mathrm{w}^{(l)}} L = \delta^{(l)} \left(\mathrm{x}^{(l-1)}\right)^{\mathrm{T}}$

偏置：$\nabla_{\mathrm{b}^{(l)}} L = \delta^{(l)}$

梯度下降更新：$\mathrm{W}^{(l)} = \mathrm{W}^{(l)} - \eta \nabla_{\mathrm{W}^{(l)}} L$

$\mathrm{b}^{(l)} = \mathrm{b}^{(l)} - \eta \nabla_{\mathrm{b}^{(l)}} L$

12.14 算法总结


神经网络的训练算法可以总结为

复合函数求导＋梯度下降法

梯度下降法有几种不同的实现
单样本模式
批量模式

除梯度下降法以外，还可以采用二阶技术，如牛顿法
随机梯度下降法

总结为符合函数求导和梯度下降法

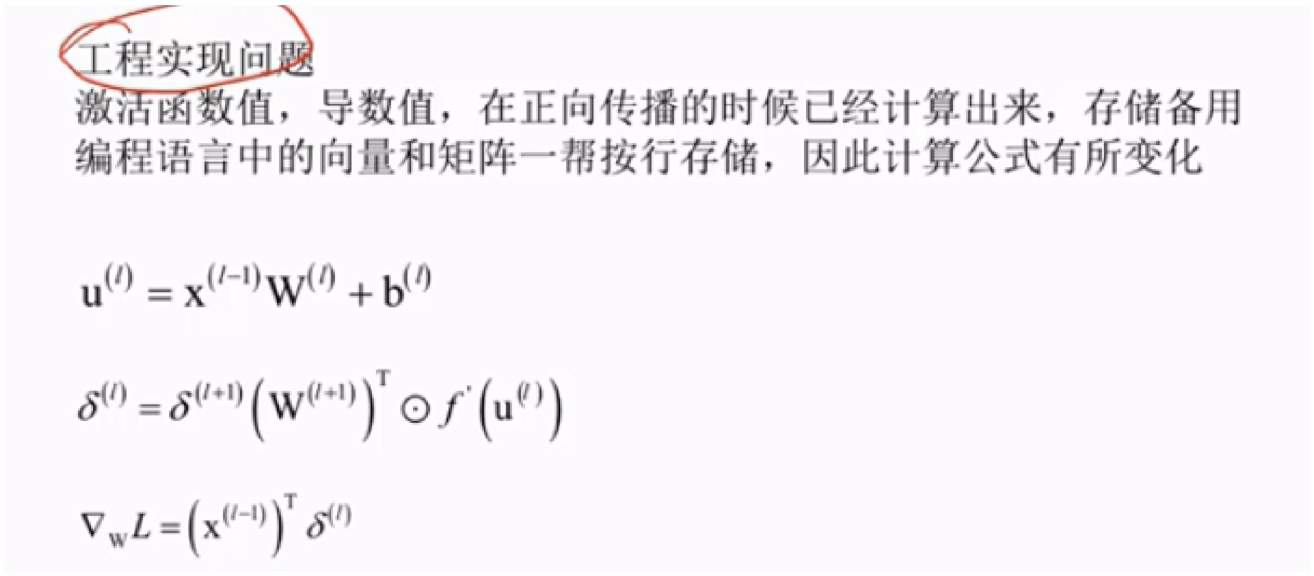
梯度下降法有几种不同的实现

- 单样本模式
- 批量模式 mini-batch

除了梯度下降法以外，还可以用其他方法。

随机梯度下降法 SCD

12.15 工程实现细节



工程实现问题

激活函数值，导数值，在正向传播的时候已经计算出来，存储备用

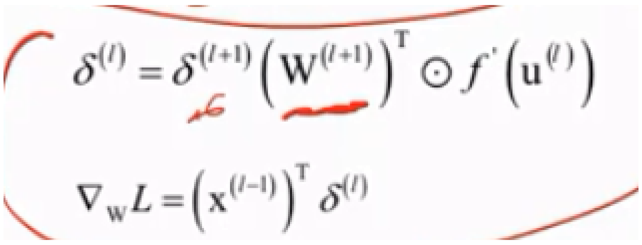
编程语言中的向量和矩阵一帮按行存储，因此计算公式有所变化

$$\mathbf{u}^{(l)} = \mathbf{x}^{(l-1)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}$$

$$\delta^{(l)} = \delta^{(l+1)}\left(\mathbf{W}^{(l+1)}\right)^{\mathrm{T}} \odot f'\left(\mathbf{u}^{(l)}\right)$$

$$\nabla_{\mathbf{W}} L = \left(\mathbf{x}^{(l-1)}\right)^{\mathrm{T}} \delta^{(l)}$$

反向传播



$$\delta^{(l)} = \delta^{(l+1)}\left(\mathbf{W}^{(l+1)}\right)^{\mathrm{T}} \odot f'\left(\mathbf{u}^{(l)}\right)$$

$$\nabla_{\mathbf{W}} L = \left(\mathbf{x}^{(l-1)}\right)^{\mathrm{T}} \delta^{(l)}$$

12.16 本集总结