生命科学中的机器学习

深度学习入门 (2)

黄　强

复旦大学 生命科学学院

本次课程内容

1 深度学习的基本概况

2 感知机模型与单层人工神经网络

3 多层神经网络与误差反向传播算法：

核心训练算法—梯度下降法的原理

4 深度卷积神经网络与应用

3.1 核心训练算法：梯度下降（gradient descent）法

上次课程回顾：

给定n个样本数据，神经网络参数可视为损失函数*L*的自变量, “训练”即是求得使损失函数最小化的参数集。

求损失函数最小化的参数集 $$w_1^*, w_2^*, b^* = \underset{w_1, w_2, b}{\operatorname{argmin}}\ \ell(w_1, w_2, b).$$

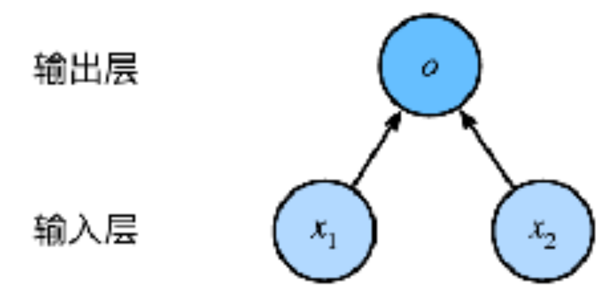


图 3.1: 线性回归是一个单层神经网络

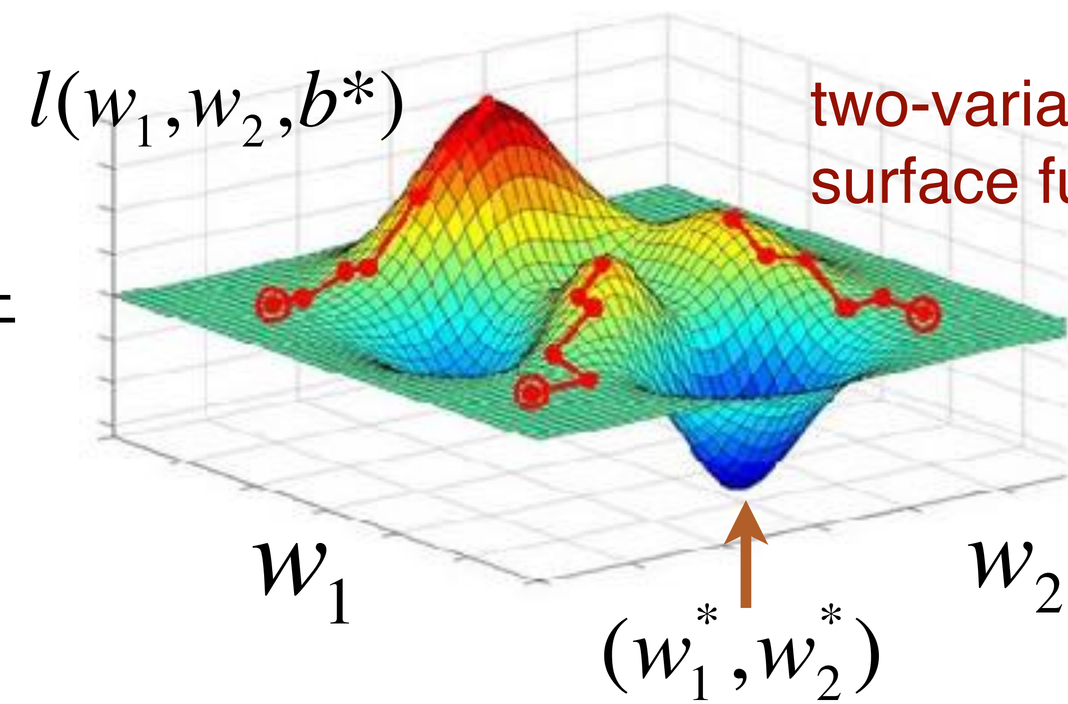
样本 *i* 的平方误差　　　　　　　　网络计算值　样本实验值

$$\ell^{(i)}(w_1, w_2, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2,$$

全部n个样本的平方误差

$$\ell(w_1, w_2, b) = \frac{1}{n}\sum_{i=1}^{n} \ell^{(i)}(w_1, w_2, b) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)}\right)^2.$$

为直观，假定给定了参数b*，只求参数w_1 和 w_2



two-variable surface function

则计算机对w1、w2的迭代求取过程类似于人在“山地”中搜索寻找到“谷底”的过程。

更多参数类似，只是在高维空间进行计算而已!

3.2 函数最优化算法的基本模式

神经网络训练过程实质是一个损失函数的最小化过程，是计算数学最优化方法的具体应用！

$$\min f(x)$$ 目标函数 （如损失函数）

subject to (s.t.) $x \in X$ $\quad X \in R^n$ n维空间

（神经网络模型参数空间巨大！）

（如权重和偏置参数）变量 约束集或可行域（如变量为角度，则只能取0-360度）

（1）无约束最优化（变量无任何约束条件）：

$$\min_{x \in R^n} f(x)$$

（迭代过程中，权重与偏置参数可以取任何值！）

（2）约束最优化（变量有约束条件）：

$$\min f(x)$$

s.t. $c_i(x) = 0, i \in E$ 等式约束指标集

$c_i(x) \geq 0, i \in I$ 不等式约束指标集

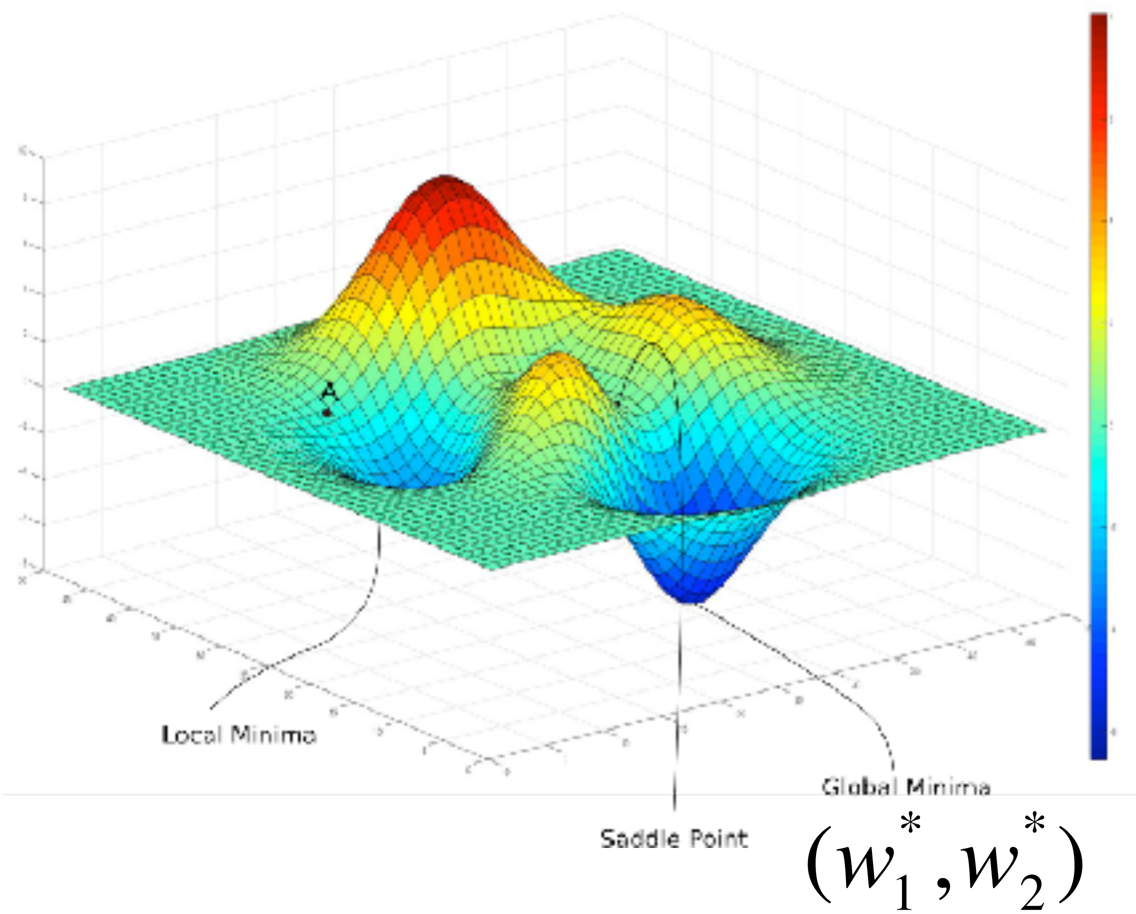
（如在迭代过程中，可以设定权重与偏置参数只能取某一区间的值！）

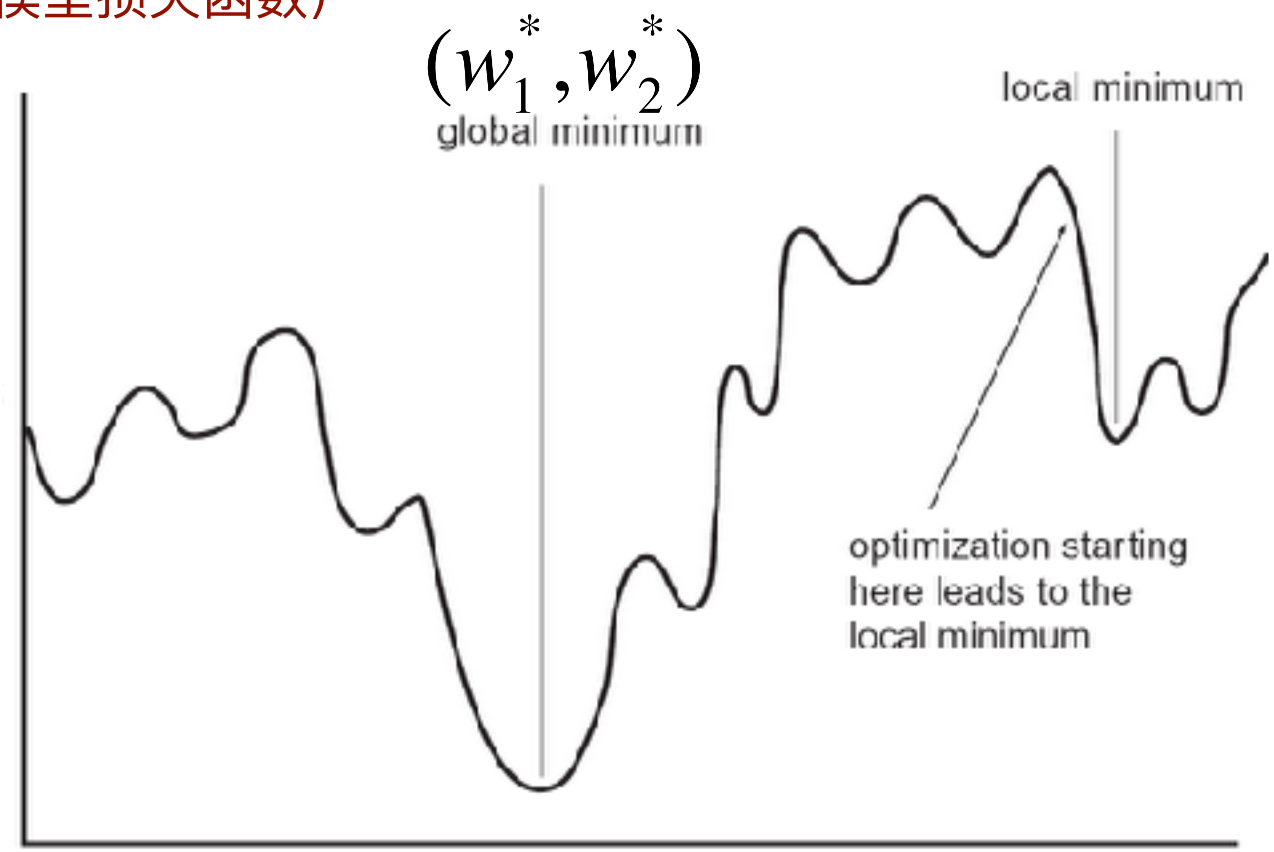
课后深度阅读计算数学参考书,了解各种最优化方法及其原理?

3.3 最优化算法的难点

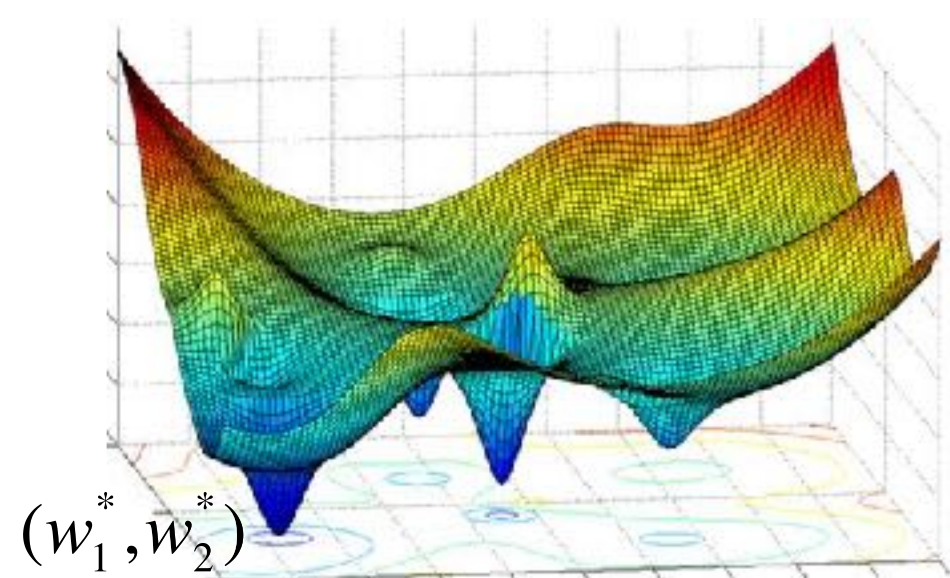
神经网络参数变量空间巨大，难于在有效的时间内遍历所有可能组合，找出全局最小点。

(神经网络模型损失函数)



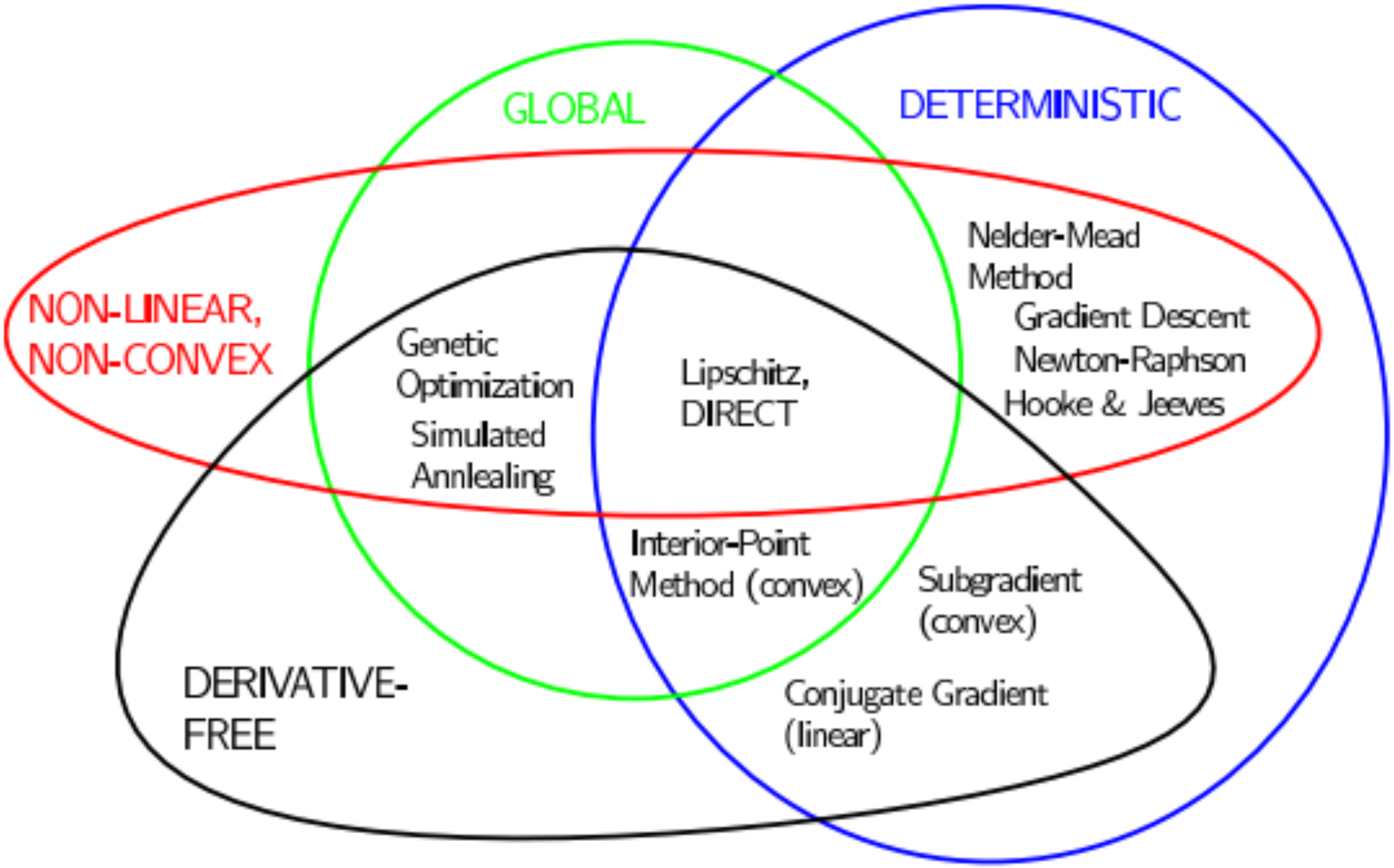


(神经网络模型参数空间)



损失函数面复杂，局部极小点多，初始值设定不好，一般的优化方法易于搜索到局部极小，无法越出，不能获得最优参数。

知识拓展：最优化方法的分类



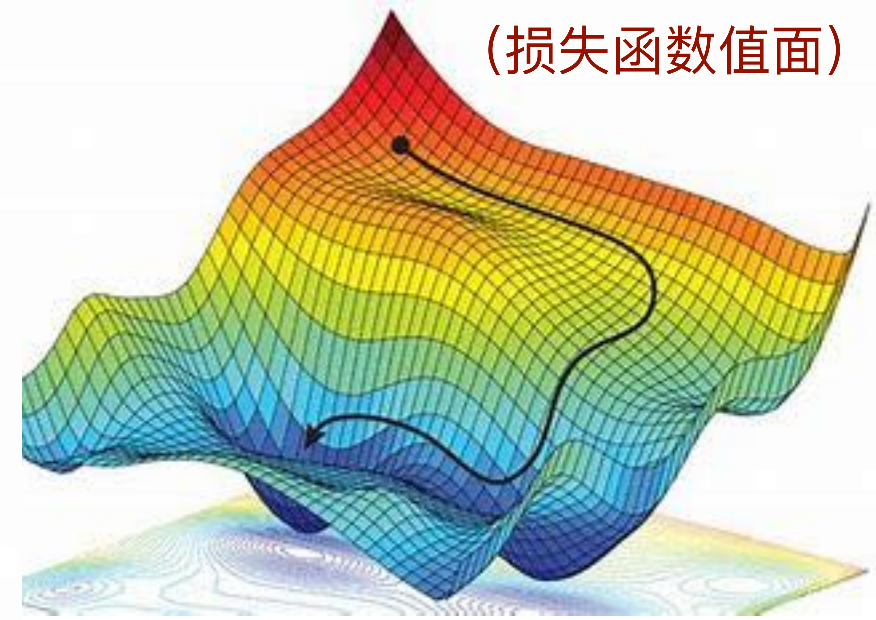
3.4 优化算法的迭代计算与下山过程十分类似

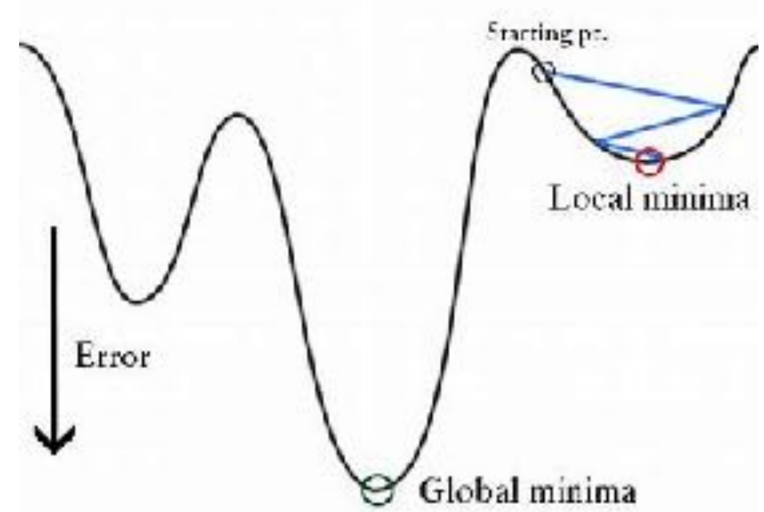
想象：在大雾天，只能看到很近的距离处，人从山顶出发，搜索下山，逐步走到谷底的过程。

下山：“高度函数”最小化

（“高度函数值面”）

（损失函数值面）





关键：根据每一步站立点周围的“局部信息”决定下一步的路径！

思考：如何把下山过程数学化和计算机化？

3.5 优化迭代计算过程的一般模式

给定n维的变量空间，最优化算法的迭代模式一般为（设迭代步骤 $k = 0, 1, 2, ...$）：

(1) 给定坐标空间的初始点 $\vec{x}_0$ (初始参数)

(2) 如果 $\vec{x}_k$ 满足停机条件，跳至 (6)

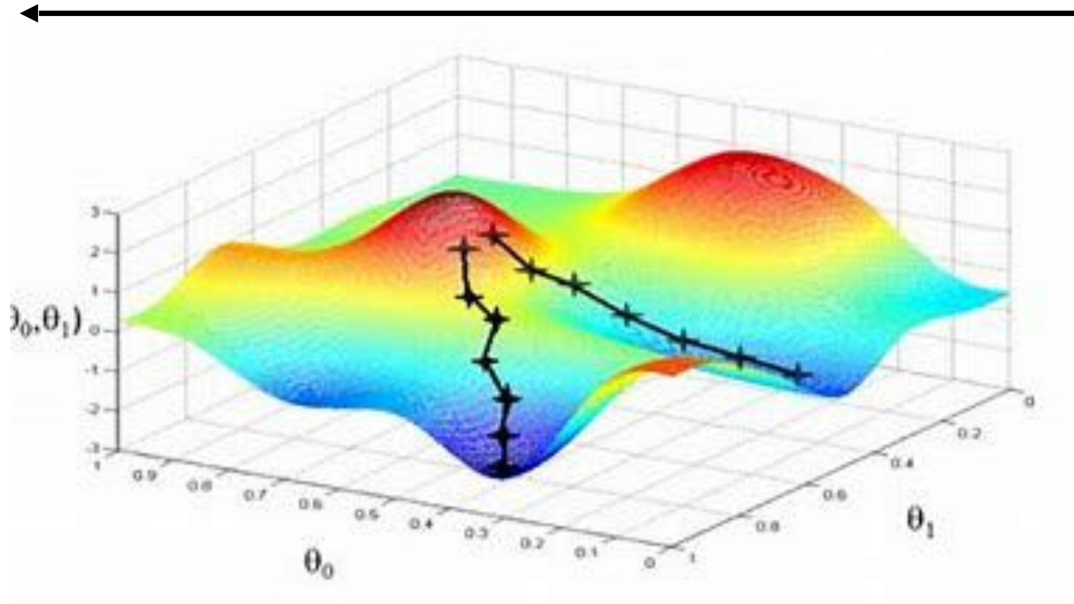
(3) 按照某种规则构造目标函数 $f(x)$ 的下降搜索方向 $\mathbf{p}_k$

(4) 用一维搜索算法或其它规则确定步长因子 α_k

(5) 算出下一迭代点 $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \mathbf{P}_k$ ，并跳至(2)

(6) 终止迭代并输出

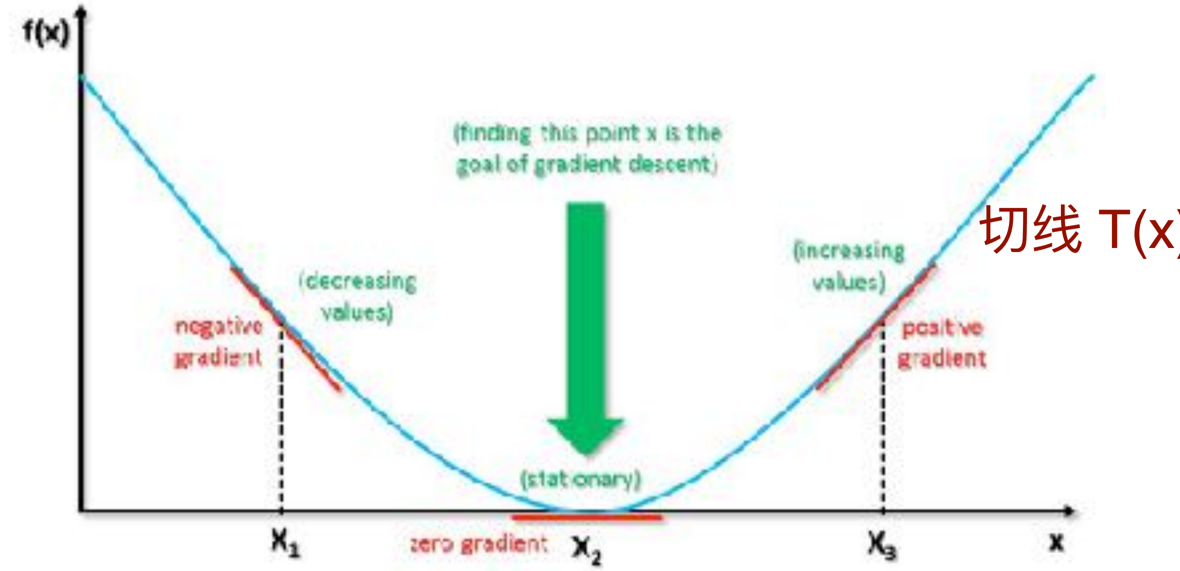
$\vec{x}^*$



关键：
如何决定
每一步的
下降方向？

思考：哪些量是n维向量，哪些是标量？

3.6 函数面上一点的方向导数、偏导数—关于下降方向局部信息

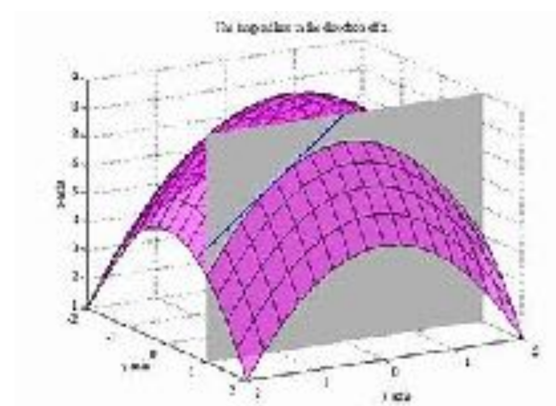


单变量连续曲线上一点x0的导数

$$f'(x_0)=\lim_{\delta x\to 0}\frac{f(x_0+\delta x)-f(x_0)}{\delta x}$$

在切线(tangent) *T*(x) 上: $\dfrac{\Delta T(x)}{\Delta x}=f'(x_0)$

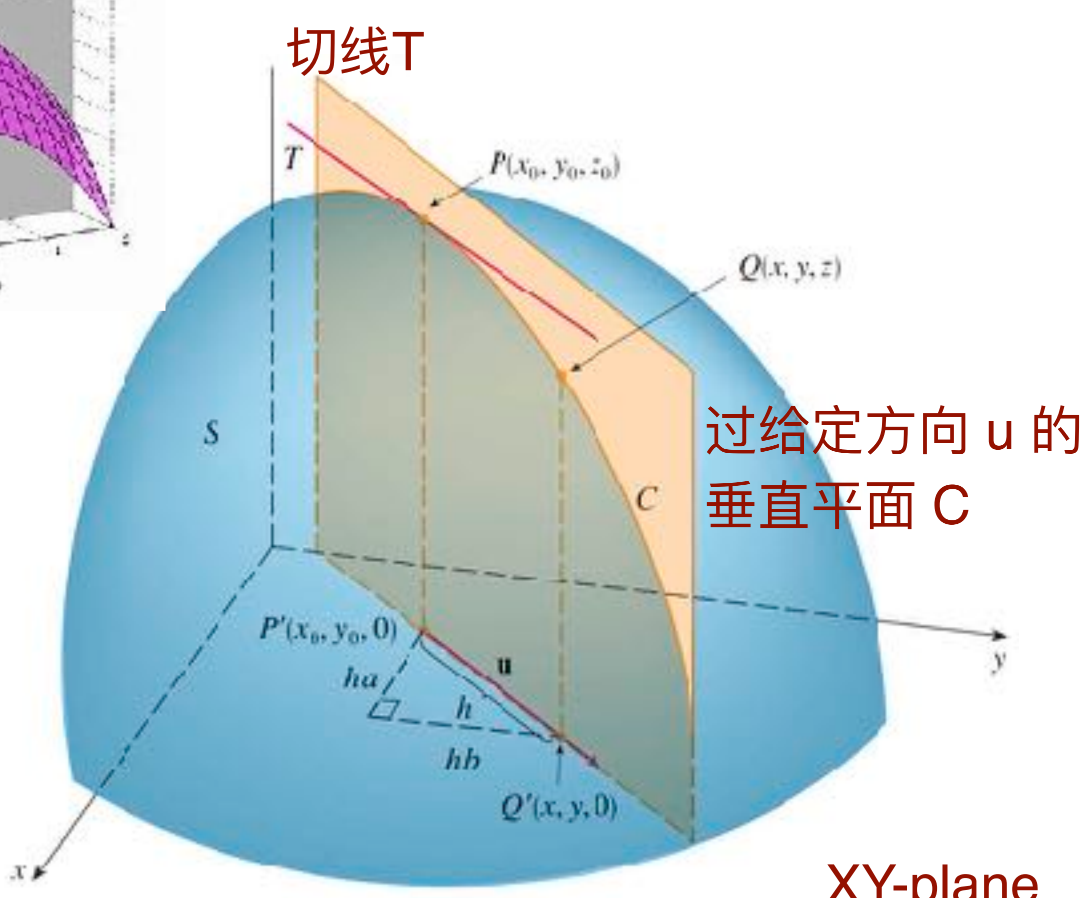
导数就是x的单位长度内切线 *T*(x) 的变化值!



多变量连续曲面上一点的导数(方向导数)

需先在变量平面上指定一个方向 **u**，才能定义给定点在该方向上的导数（方向导数，Directional derivatives):

$$D_{\mathbf{u}}f(x_0,y_0)=\lim_{h\to 0}\frac{f(x_0+h\cos\alpha,y_0+h\sin\alpha)-f(x_0,y_0)}{h}$$



多变量连续曲面上一点的偏导数(partial derivatives)

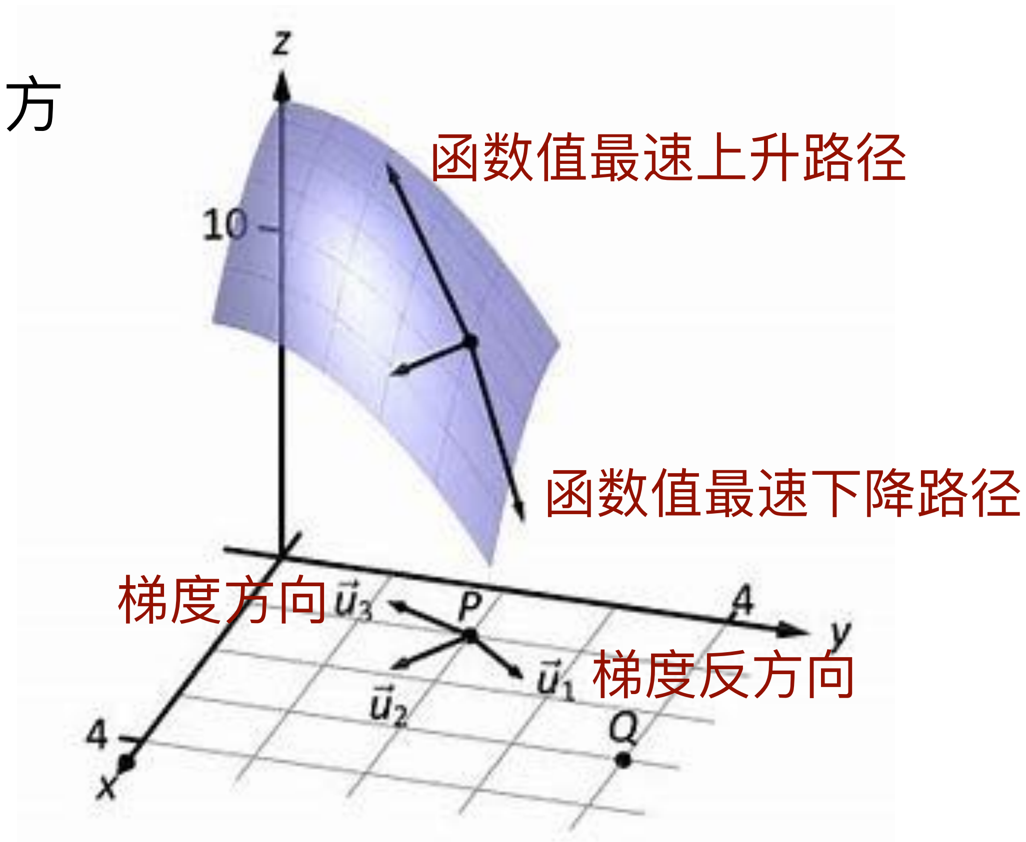
给定坐标系后，偏导数就是在坐标抽方向上的方向导数:

$$\frac{\partial f(x_0,y_0)}{\partial x}=\lim_{h\to 0}\frac{f(x_0+h,y_0)-f(x_0,y_0)}{h}$$

$$\frac{\partial f(x_0,y_0)}{\partial y}=\lim_{h\to 0}\frac{f(x_0,y_0+h)-f(x_0,y_0)}{h}$$

3.7 函数面上一点处方向导数值最大的方向: 梯度方向!

与人下山类似，比较函数面一点 (x0, y0) 处的所有方向导数值，导数值最大的方向就是函数值“最速上升”的方向，反方向就是“最速下降”的方向！



（注意：梯度是一个在XY-plane里的向量！）

方向导数值最大的方向称梯度方向！

指定坐标系后，可求出坐标轴方向的偏导数：

$$\frac{\partial f(x_0,y_0)}{\partial x}=\lim_{h\to 0}\frac{f(x_0+h,y_0)-f(x_0,y_0)}{h}$$

$$\frac{\partial f(x_0,y_0)}{\partial y}=\lim_{h\to 0}\frac{f(x_0,y_0+h)-f(x_0,y_0)}{h}$$

由以下正交的偏导数向量（与坐标轴方向一致）相加：

$$[\frac{\partial f(x_0,y_0)}{\partial x},0] \qquad [0,\frac{\partial f(x_0,y_0)}{\partial y}]$$

所得向量称为梯度（gradient)：

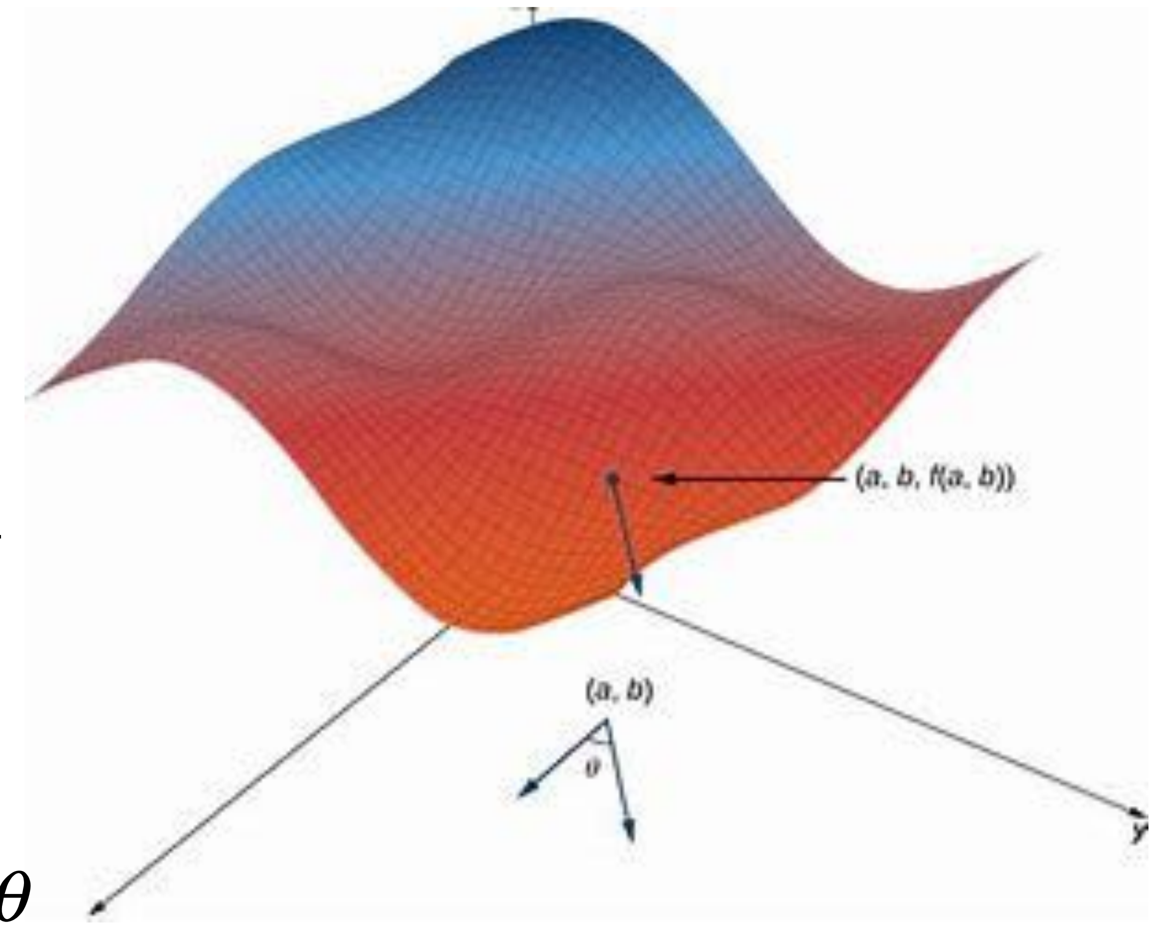
$$\nabla f(x_0,y_0)=[\frac{\partial f(x_0,y_0)}{\partial x},\frac{\partial f(x_0,y_0)}{\partial y}]$$

该向量在XY-plane的方向称梯度方向
向量长度等于函数在梯度方向的导数值。

3.8 梯度向量与方向导数关系

数学上可证明，XY-plane任何方向 **u** 的方向导数：

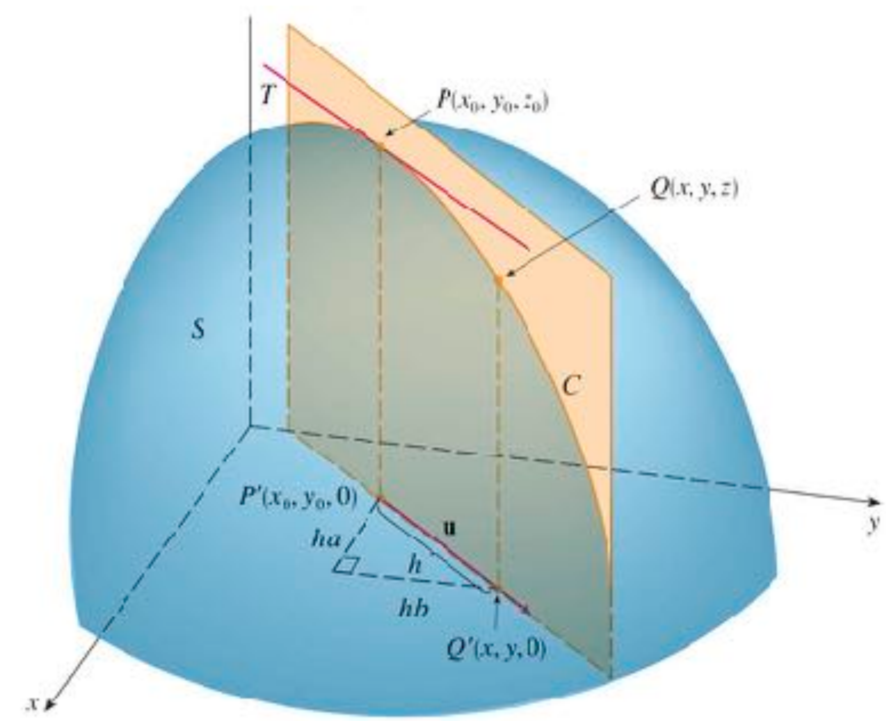
$$D_{\mathbf{u}}f(x_0,y_0)=\lim_{h\to 0}\frac{f(x_0+h\cos\alpha,y_0+h\sin\alpha)-f(x_0,y_0)}{h}$$



与梯度向量有（设两个向量之间的夹角为 θ）：

$$D_{\vec{u}}f(x_0,y_0)=\nabla f(x_0,y_0)\bullet\vec{u}=\left\|\nabla f(x_0,y_0)\right\|\times\left\|\vec{u}\right\|\cos\theta$$

(**u** 单位向量)



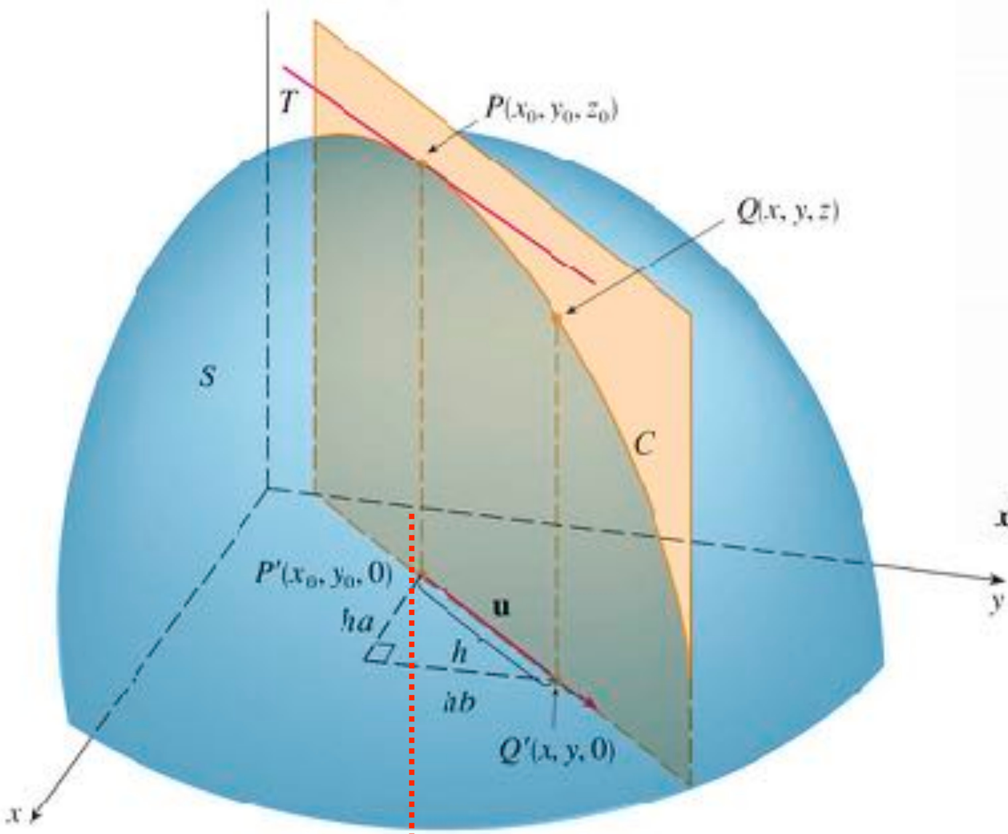
结论：当夹角为0度，方向导数值有最大值，即最大方向就是梯度方向。

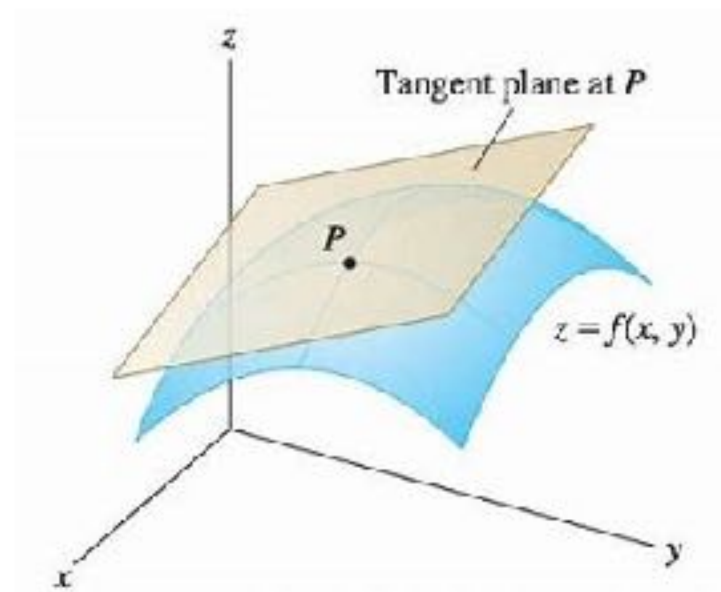
(请自行参考数学书和网上资料推导上述公式！)

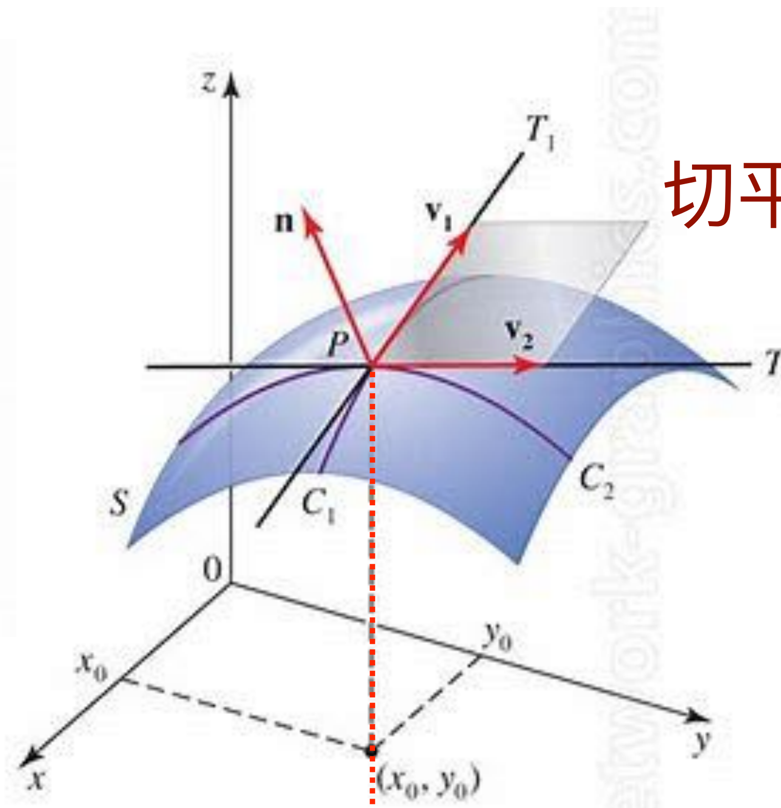
讨论与思考：

为什么连续变化曲面上的点存在梯度，且唯一?

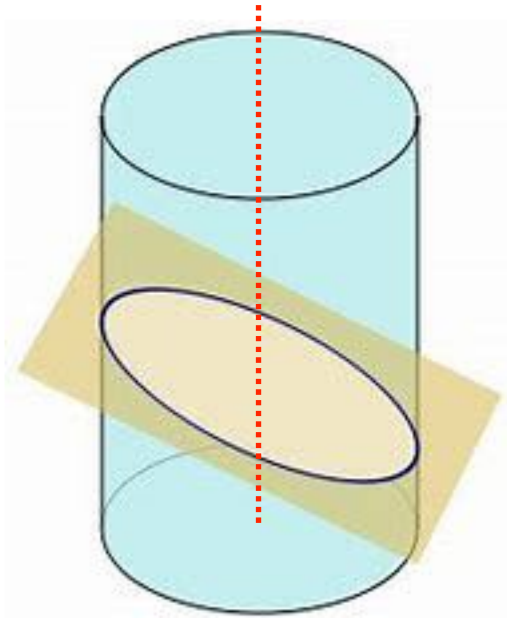
3.9 梯度 (或最大方向导数) 的一种直观几何理解





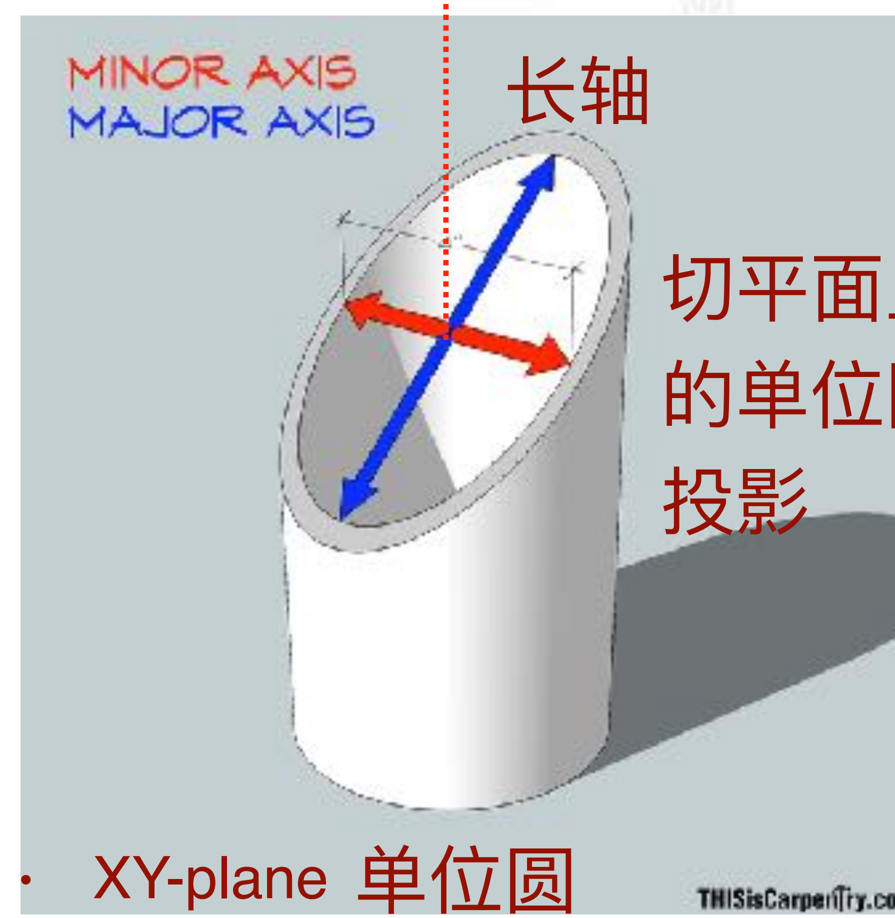


切平面



- XY-plane 单位圆

- 各方向切线簇组成切平面；
- 以(x0, y0)为中心的XY-plane 单位圆在切平面上投影为椭圆；
- 椭圆长轴正反方向与梯度向量正反方向一致，长度与方向导数值对应。



长轴

切平面上的单位圆投影

- XY-plane 单位圆

简单例子演示：

最速下降法

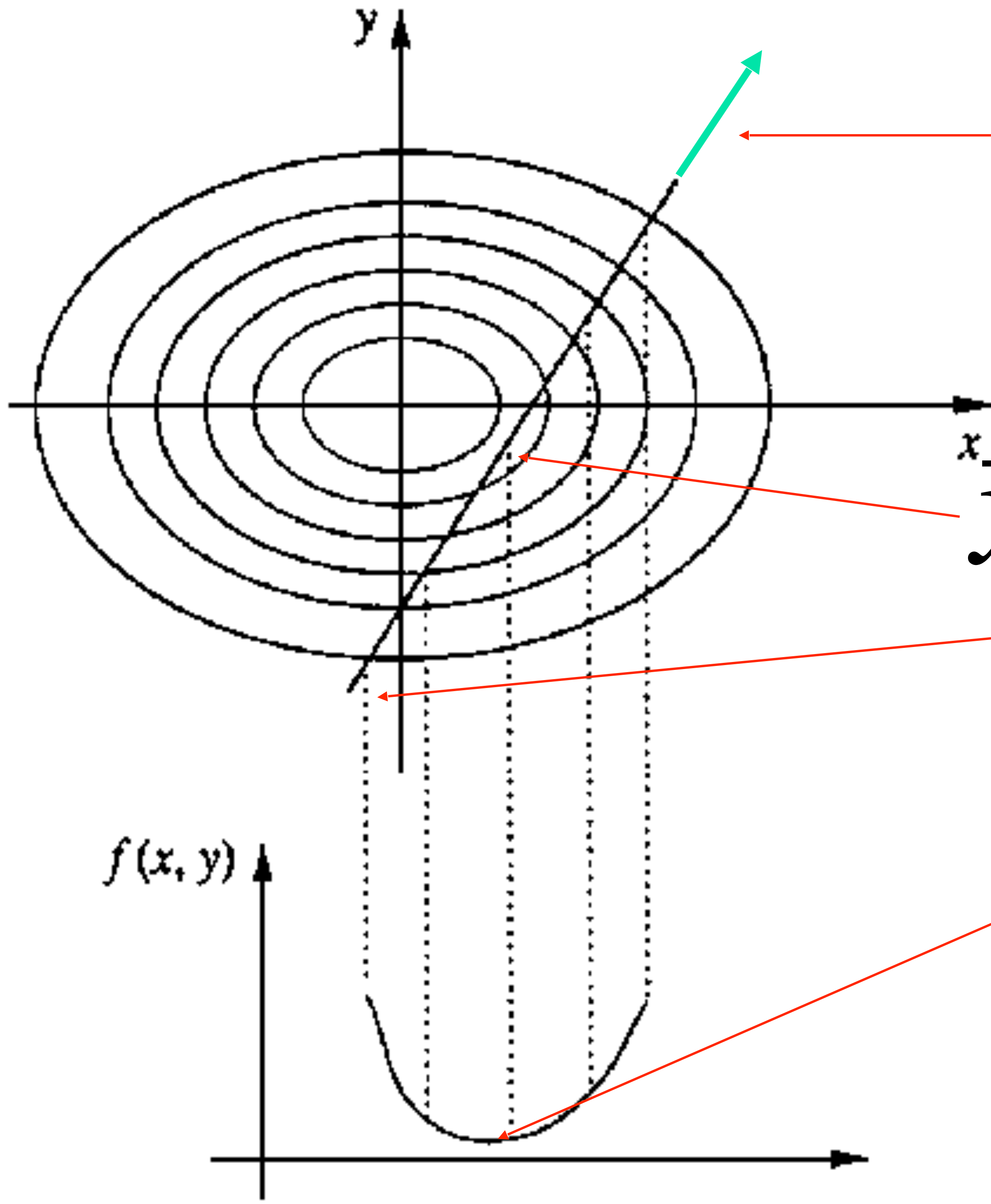
$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{s}_k$$

$$\vec{s}_k = -\frac{\vec{g}_k}{\left|\vec{g}_k\right|} \quad \begin{array}{l} \vec{g}_k \longleftarrow \text{梯度} \\ \left|\vec{g}_k\right| \longleftarrow \text{向量长度} \end{array}$$

用一维搜索算法找步长因子，使得在负梯度方向上有：

$$\min_{\alpha_k > 0} f(\vec{x}_k + \alpha_k \vec{s}_k)$$

第*k*步的一维搜索（Line search)

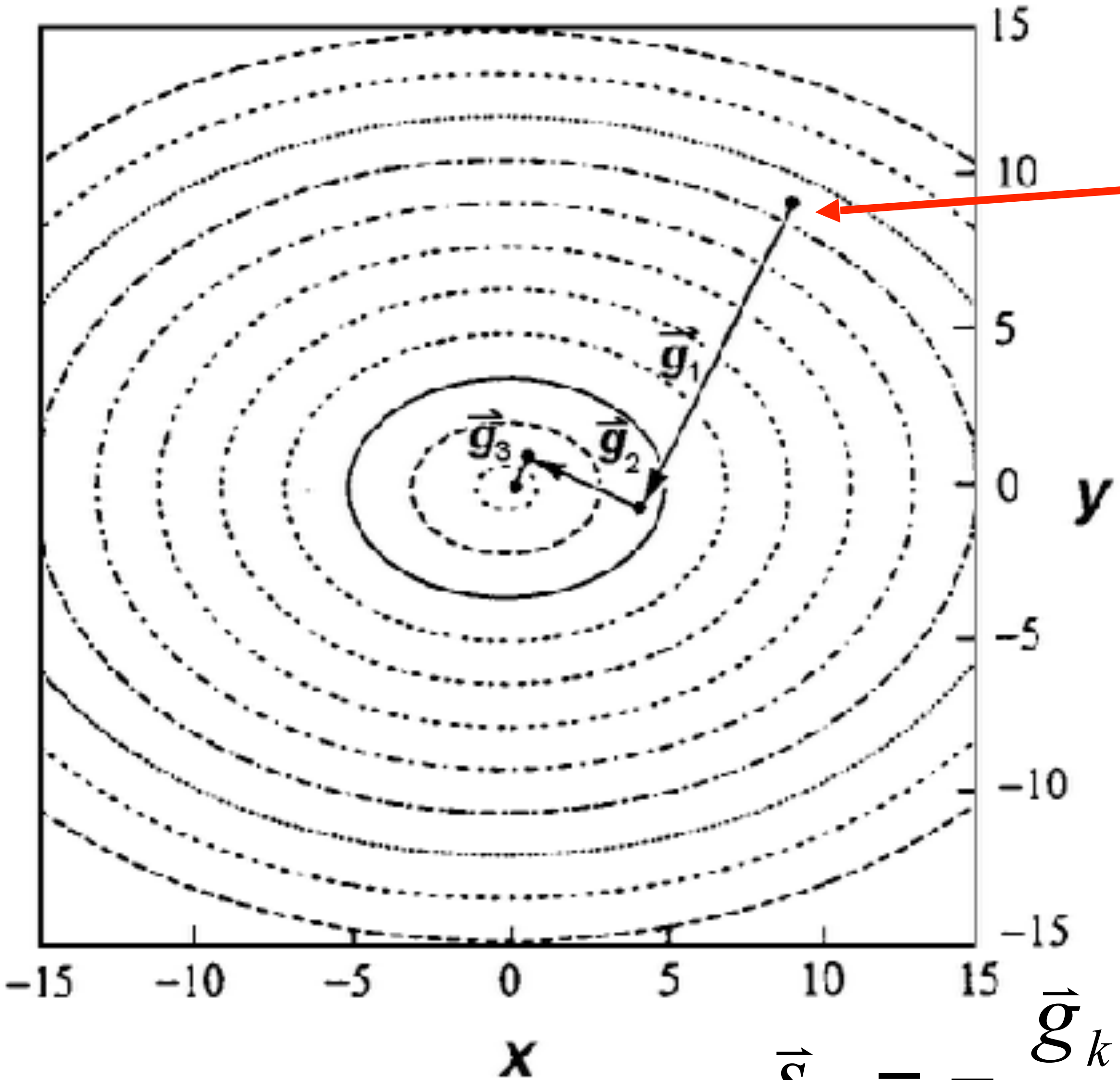


搜索方向

$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{s}_k$$

合适的步长因子
对应给定的搜索
方向上的最小值

例子：求 $f(x,y)=x^2+2y^2$ 的极小值



起始点 (9,9)，k=0

$$f(\vec{x}_k)=(9)^2+2(9)^2=243$$

$$\vec{g}_k=\left(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y}\right)_k$$

$$=(2x,4y)_k=(18,36)$$

$$\vec{s}_k=-\frac{\vec{g}_k}{|\vec{g}_k|}=-\frac{(18,36)}{\sqrt{18^2+36^2}}=(-0.447,-0.894)$$

$$\vec{x}_{k+1}=\vec{x}_k+\alpha_k\vec{s}_k=\begin{pmatrix}9\\9\end{pmatrix}+\alpha_k\begin{pmatrix}-0.447\\-0.894\end{pmatrix}=\begin{pmatrix}9-0.447\alpha_k\\9-0.894\alpha_k\end{pmatrix}$$

一维搜索

α_k	$\vec{x}_{k+1}$	$f(\vec{x}_{k+1})$
1.0	(8.53, 8.11)	204.57
2.0	(8.11, 7.21)	169.73
3.0	(7.66, 6.32)	138.49
4.0	(7.21, 5.42)	110.85
5.0	(6.76, 4.53)	86.754
6.0	(6.32, 3.63)	66.36
7.0	(5,87, 2.79)	49.51
8.0	(5.42, 1.85)	36.25
9.0	(4.98, 0.95)	26.59
10.0	(4.53, 0.06)	20.53
11.0	(4.08, -0.83)	18.06
11.2	(4.00, -1.00)	18.00
12.0	(3.64, -1.73)	19.19

极小

3.10 用损失函数的负梯度向量值迭代修正参数

回顾：多元线性回归神经网络模型的“搜索方向”

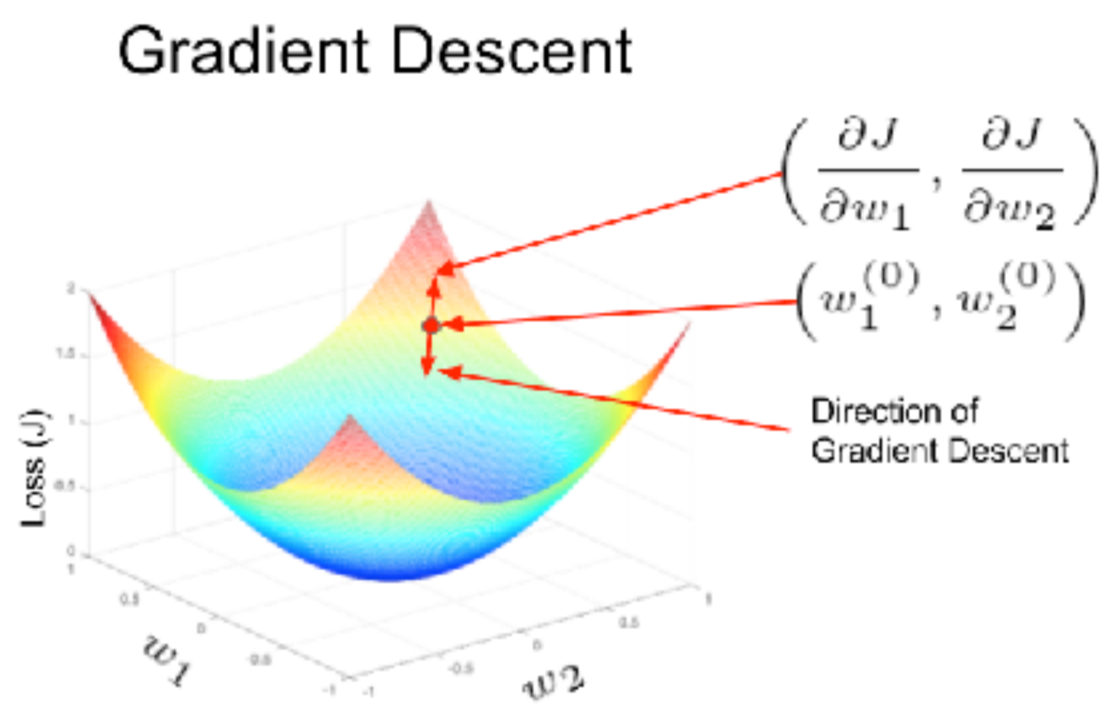
$$
\begin{aligned}
w_1 &\leftarrow w_1 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial w_1} = w_1 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} x_1^{(i)}\left(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)}\right),\\
w_2 &\leftarrow w_2 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial w_2} = w_2 - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} x_2^{(i)}\left(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)}\right),\\
b &\leftarrow b - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial b} = b - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\left(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)}\right).
\end{aligned}
$$

负方向

负梯度

$$\overrightarrow{x}_{k+1} = \overrightarrow{x}_k + \alpha_k \mathbf{P}_k$$

步长因子如何决定?



3.11 步长因子：神经网络训练的学习(速)率 (learning rate)

回顾：多元线性回归神经网络模型的“步长因子”

常用方法之一

小批量随机梯度下降 Mini-batch stochastic gradient descent (SGD)

$$\overrightarrow{x}_{k+1} = \overrightarrow{x}_k + \alpha_k \mathbf{P}_k$$

人为设定正数

$$w_1 \leftarrow w_1 - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial w_1} = w_1 - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} x_1^{(i)} \left(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)} \right),$$

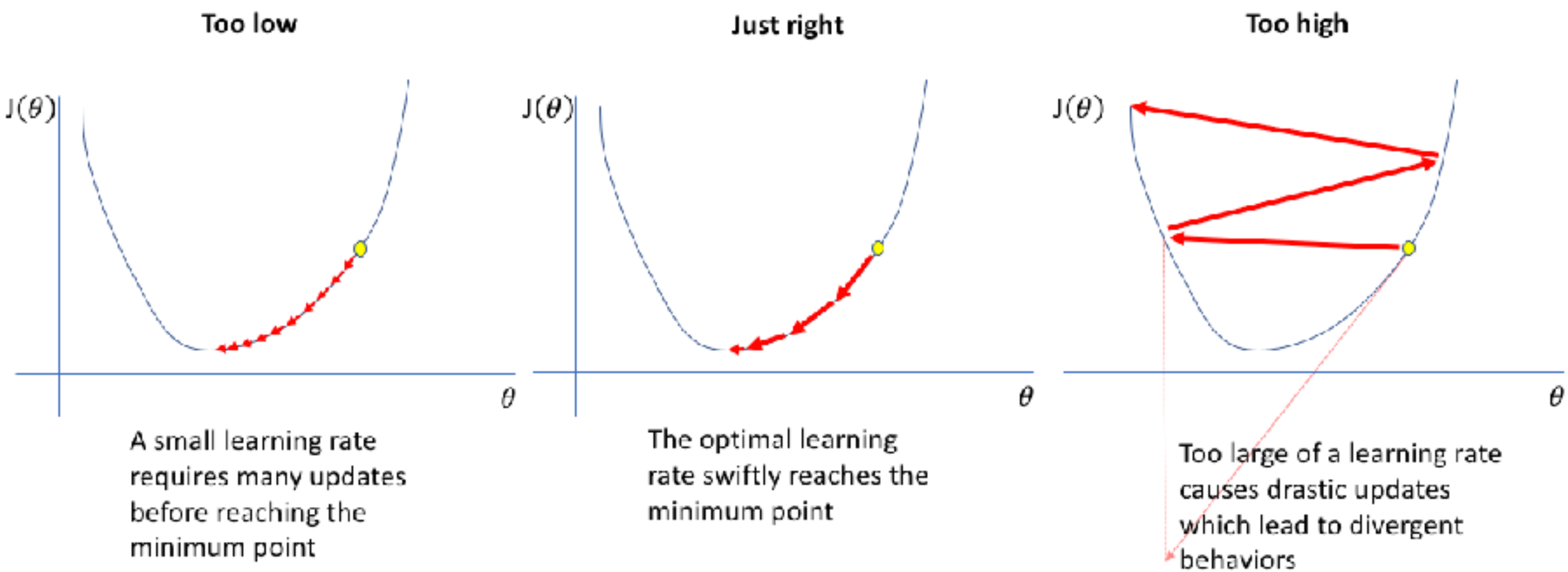
$$w_2 \leftarrow w_2 - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial w_2} = w_2 - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} x_2^{(i)} \left(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)} \right),$$

$$b \leftarrow b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{\partial \ell^{(i)}(w_1, w_2, b)}{\partial b} = b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left(x_1^{(i)} w_1 + x_2^{(i)} w_2 + b - y^{(i)} \right).$$

小批量随机样本数（mini-batch） *B*

3.12 学习率的设置

$$\overrightarrow{x}_{k+1} = \overrightarrow{x}_k + \alpha_k \mathbf{P}_k$$



调参与做生物学实验没有区别，不能太大，也不能太小，需反复试验。

本次课程内容

1 深度学习的基本概况

2 感知机模型与单层人工神经网络

3 多层神经网络与误差反向传播算法

4 深度卷积神经网络与应用

3.13 单层人工神经网络模型的主要局限

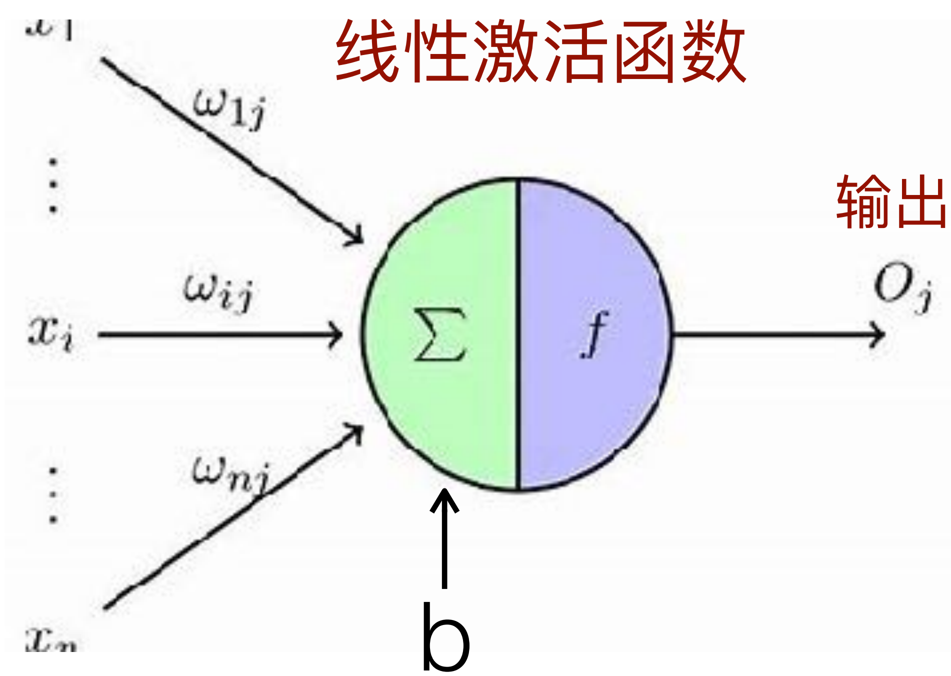
缺乏良好的输入到输出的非线性映射能力（Nonlinear mapping capability)

回归问题用

输入层

线性激活函数

输出



n个输入的单层神经网络

分类问题用

（请参考3.4～3.8节，关注**交叉熵损失函数**的定义。）

softmax 激活函数

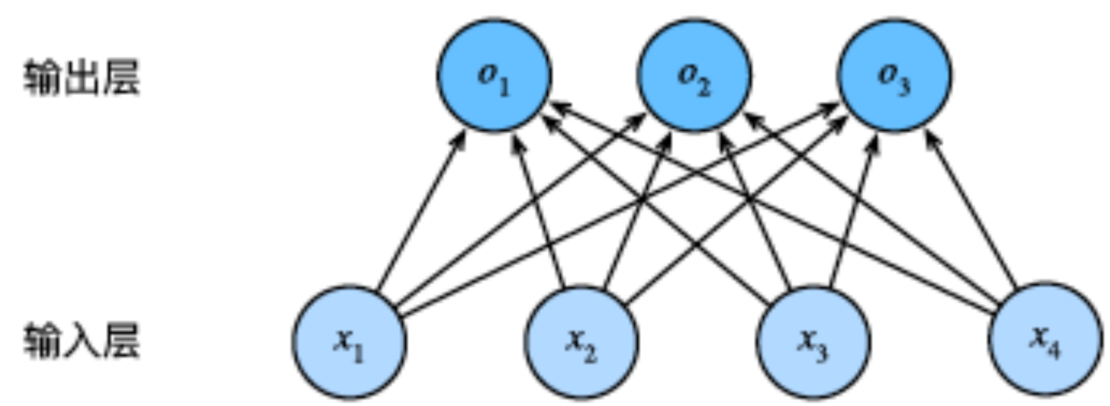


图 3.2: softmax回归是一个单层神经网络

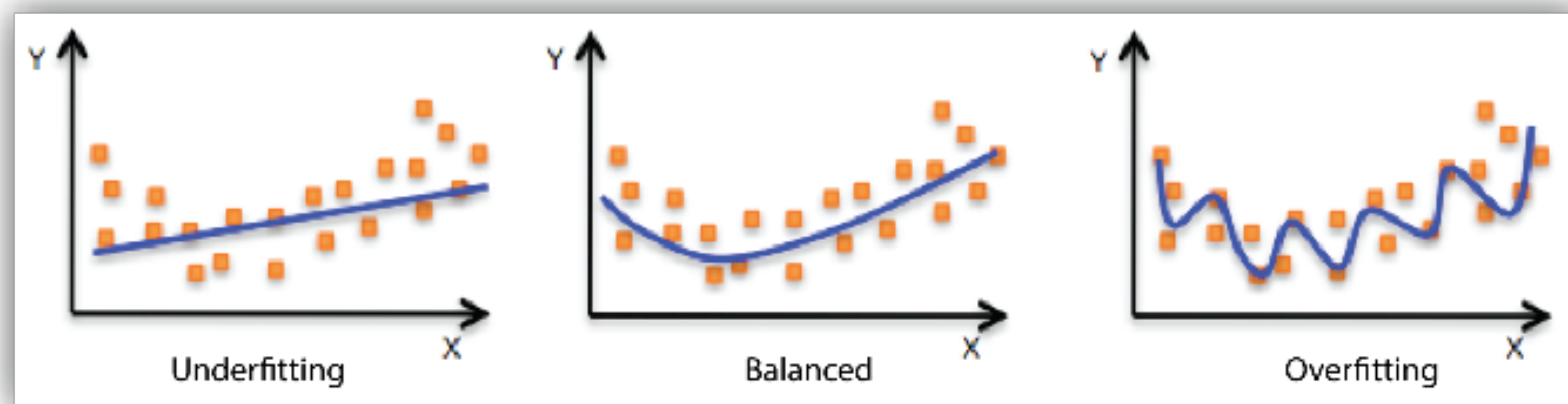
二元线性回归：$y(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$

其它不足：训练数据结果好，但是泛化能力差，易产生过拟合 (overfitting).

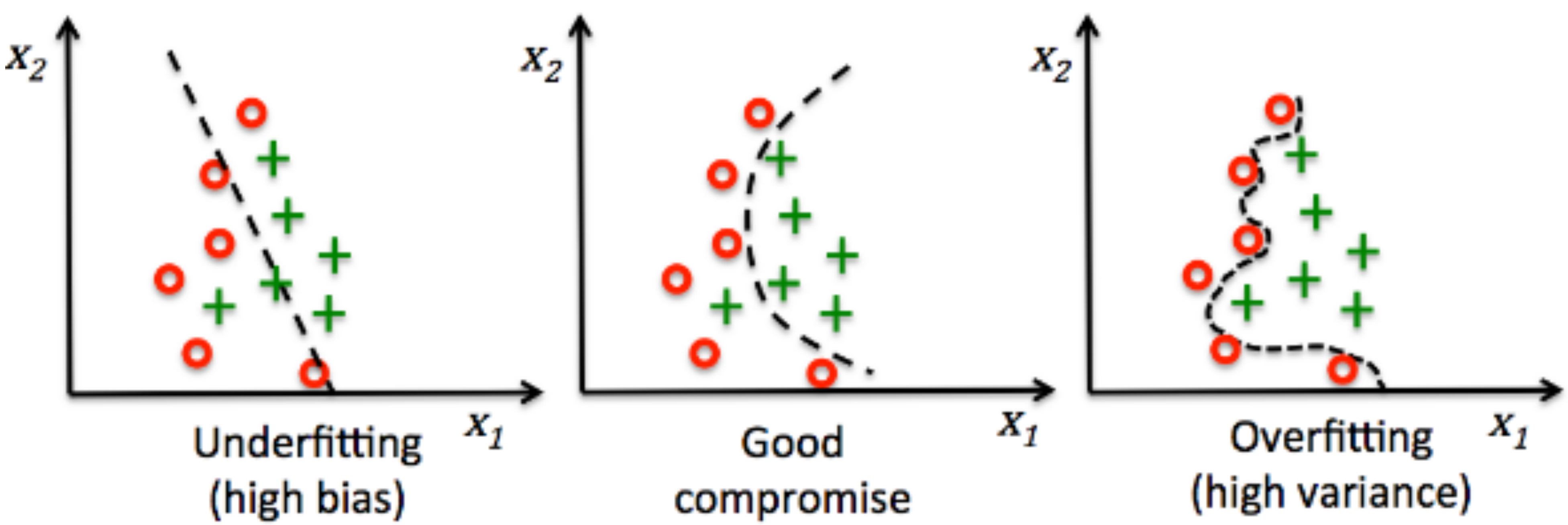
3.14 模型的欠拟合(underfitting)、过拟合 (overfitting)

缺乏良好的输入到输出的非线性映射能力（Nonlinear mapping capability)

回归问题

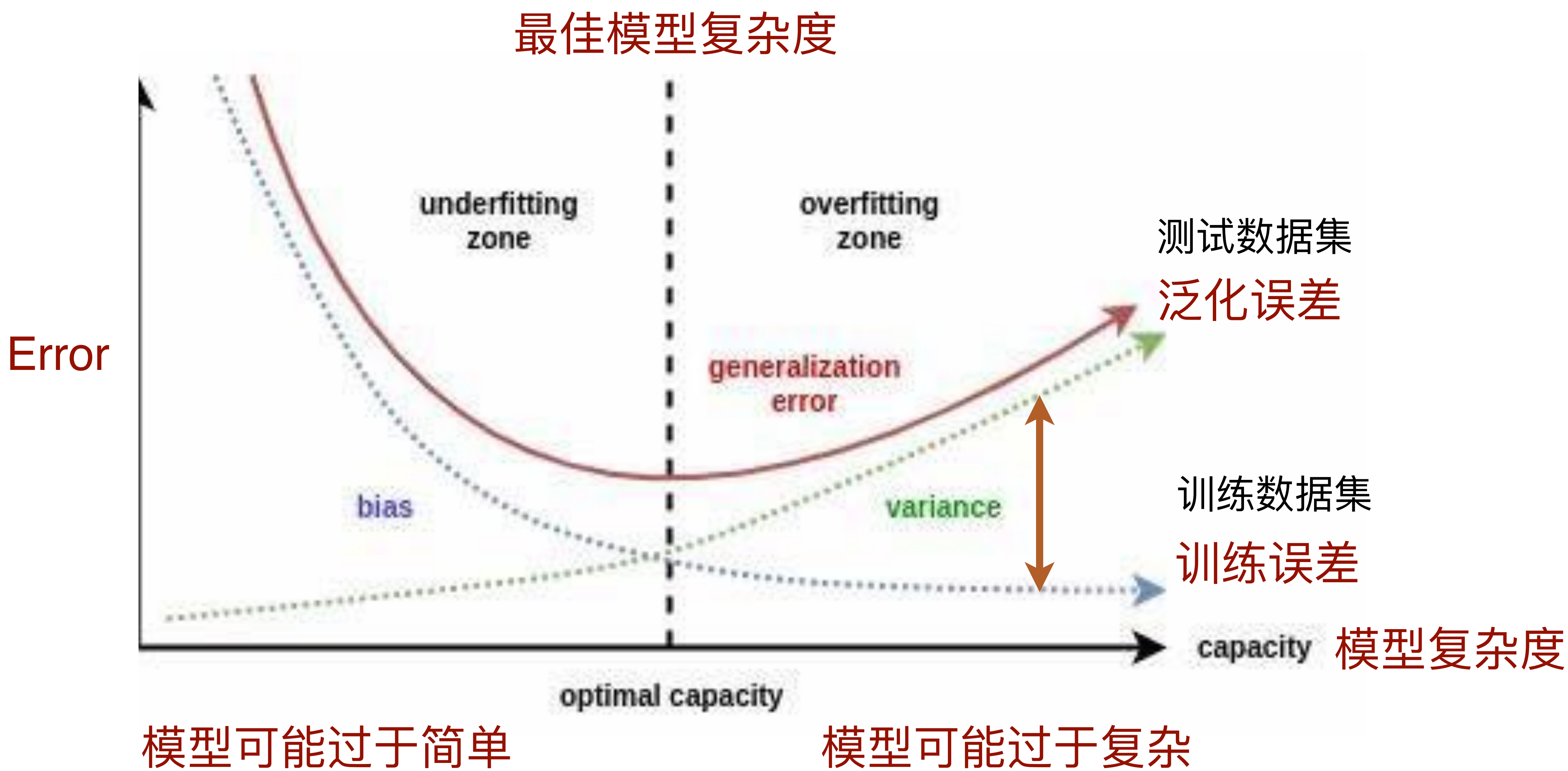


分类问题



3.15 神经网络模型的训练误差与泛化误差

神经网络模型的训练重点要降低 泛化 (generalization) 误差。



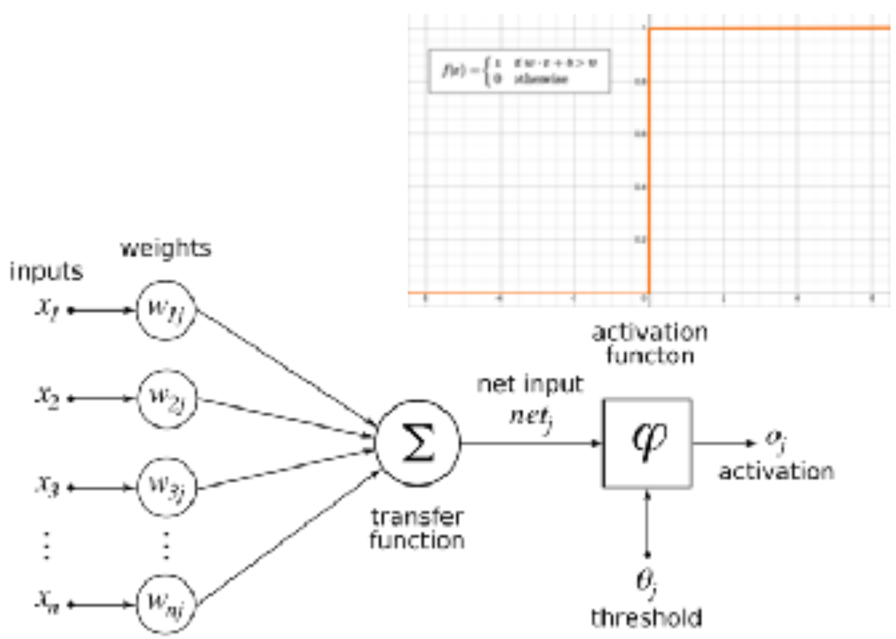
模型选择：实践中，需要根据训练数据集选择复杂度适合的神经网络模型。

3.16 多层人工神经网络的提出及其特点

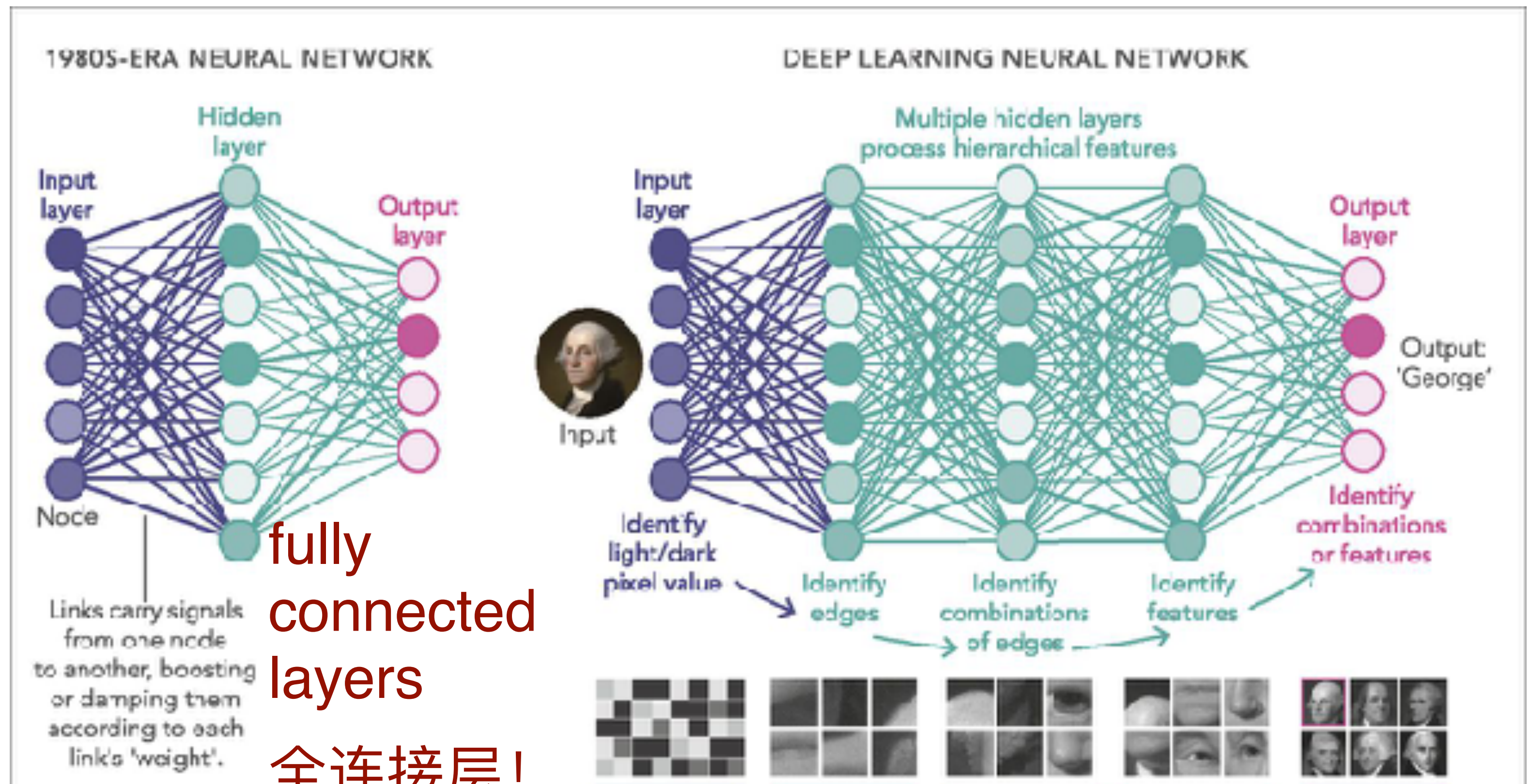
在单层神经网络的基础上引入一个或多个隐层 [hidden layer(s)]

single-layer ⟶ multi-layer artificial neural networks

1950s　　　　1980s ~ 90s　　　　~2010s



Rosenblatt’s perceptron



fully connected layers

全连接层!

(From *PNAS* 116:1074-1077, 2019)

Rumelhart, Hinton & Williams
Nature, 323:533, 1986

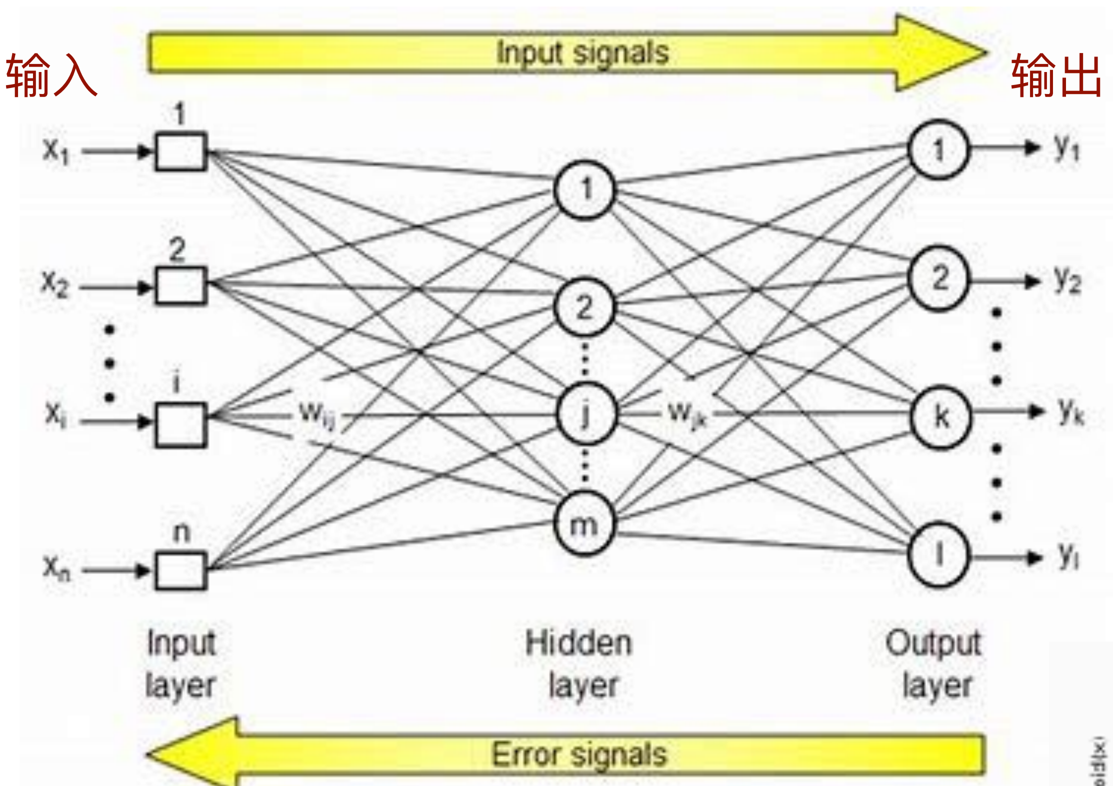
Hinton & Salakhutdinov
Science, 313:504, 2006

- 具有输入到输出的非线性映射能力(nonlinear mapping capability)
- 三层网络可逼近任意函数(universal approximation power, Hornik *et al.*, 1989)

3.17 多层前馈神经网络 (Multi-layer feedforward neural network)

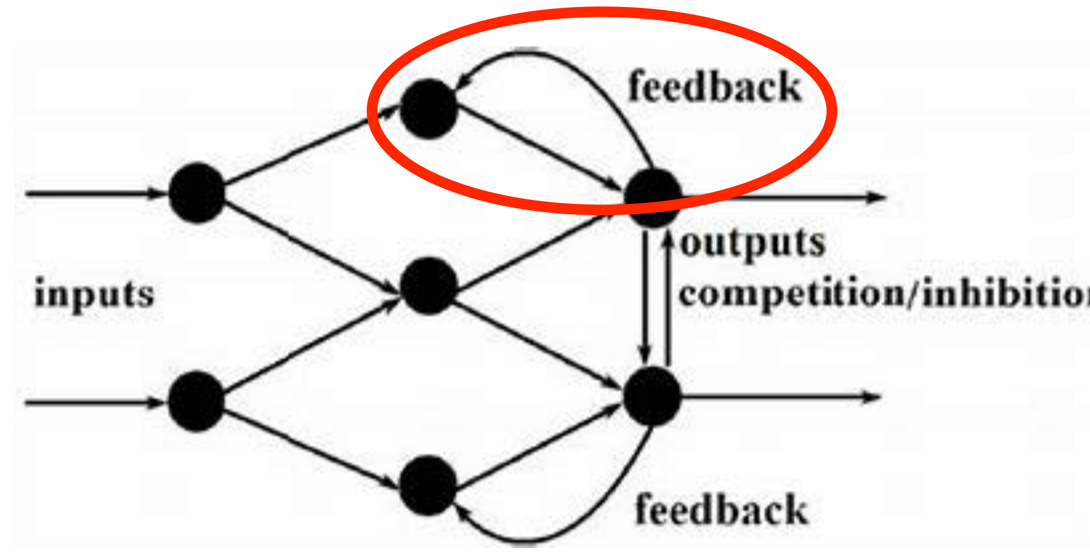
信息由输入层向前传递到一个或多个隐层，然后再传递到输出层，节点无任何 **反馈(feedback)连接**。

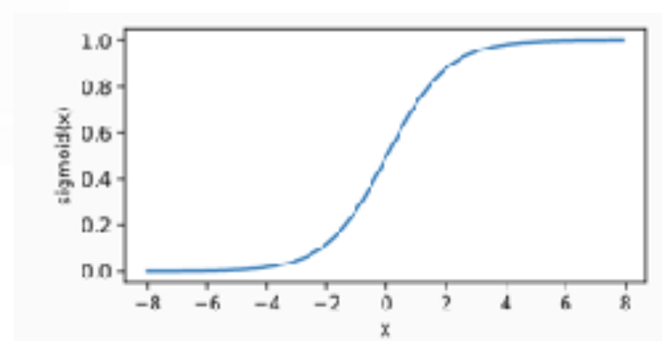
information feedforward

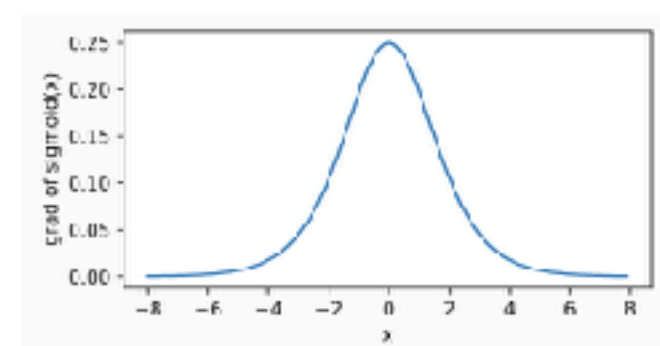


error backpropagation

神经元有反馈 (feedback)连接





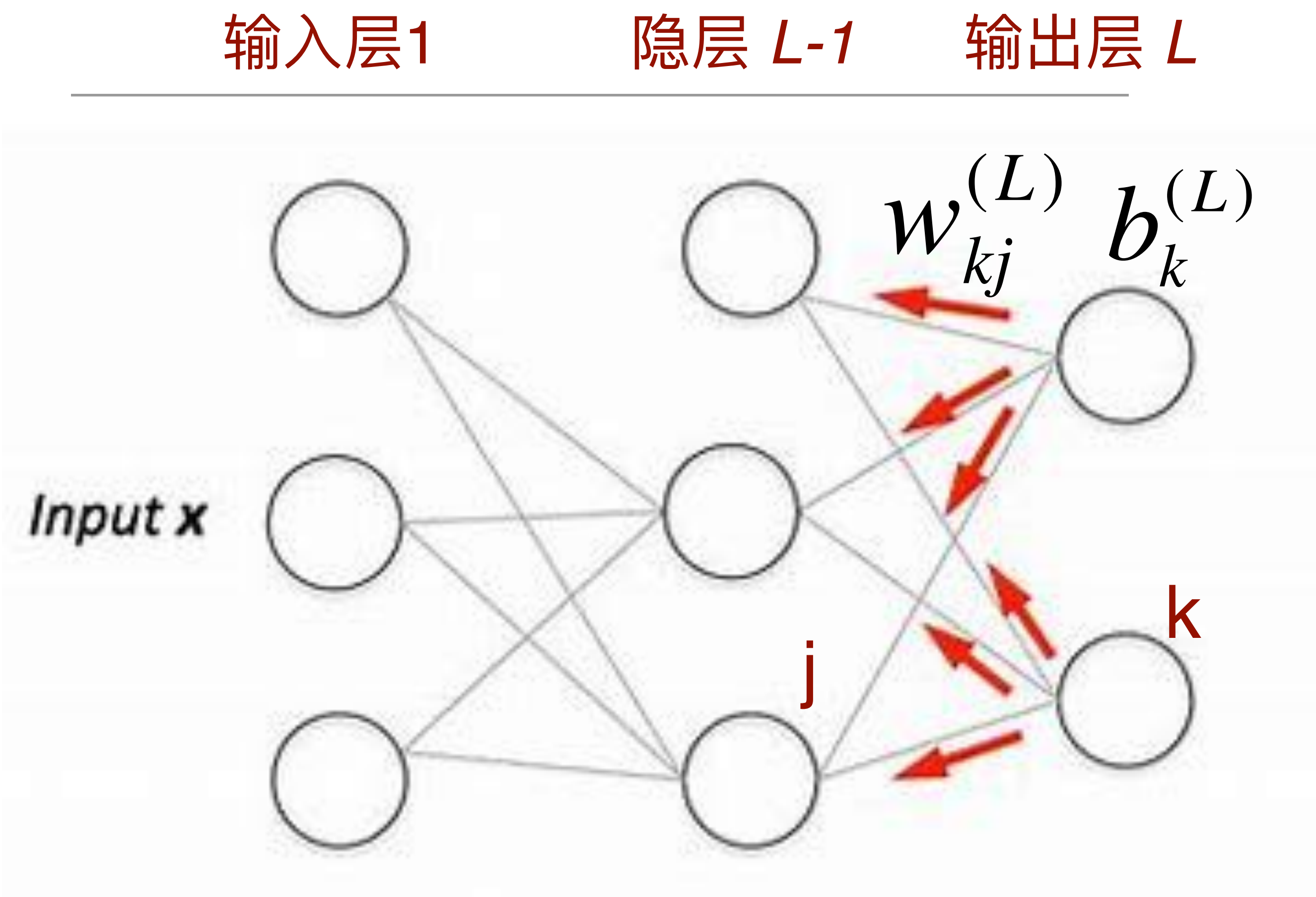


最常用的神经元激活函数 $\mathrm{sigmoid}(x) = \frac{1}{1+\exp(-x)}$ $\mathrm{sigmoid}'(x) = \mathrm{sigmoid}(x)(1-\mathrm{sigmoid}(x))$

万能逼近能力 (Universal approximation capability) Cybenko, *Math. Control Signals Systems*, 2:303,1989.

一个单一隐层的神经网络，如神经元个数足够多，可以拟合任意连续函数。

3.18 多层神经网络模型训练的梯度下降方法



输入

$$Z_k^{(L)} = \sum_{j=1}^{J}(w_{kj}^{(L)}a_j^{(L-1)} + b_k^{(L)})$$

激活函数 $$\sigma[Z] = \frac{1}{1+e^{-Z}}$$

输出

$$\sigma\left[Z_k^{(L)}\right] = \sigma\left[\sum_{j=1}^{J}(w_{kj}^{(L)}a_j^{(L-1)} + b_k^{(L)})\right]$$

训练误差

样本i $$e_i = \tfrac{1}{2}\sum_{k=1}^{K}\left(a_{k,i}^{(L)} - y_i\right)^2$$ 全部样本 $$E = \tfrac{1}{n}\sum_{i=1}^{N} e_i$$

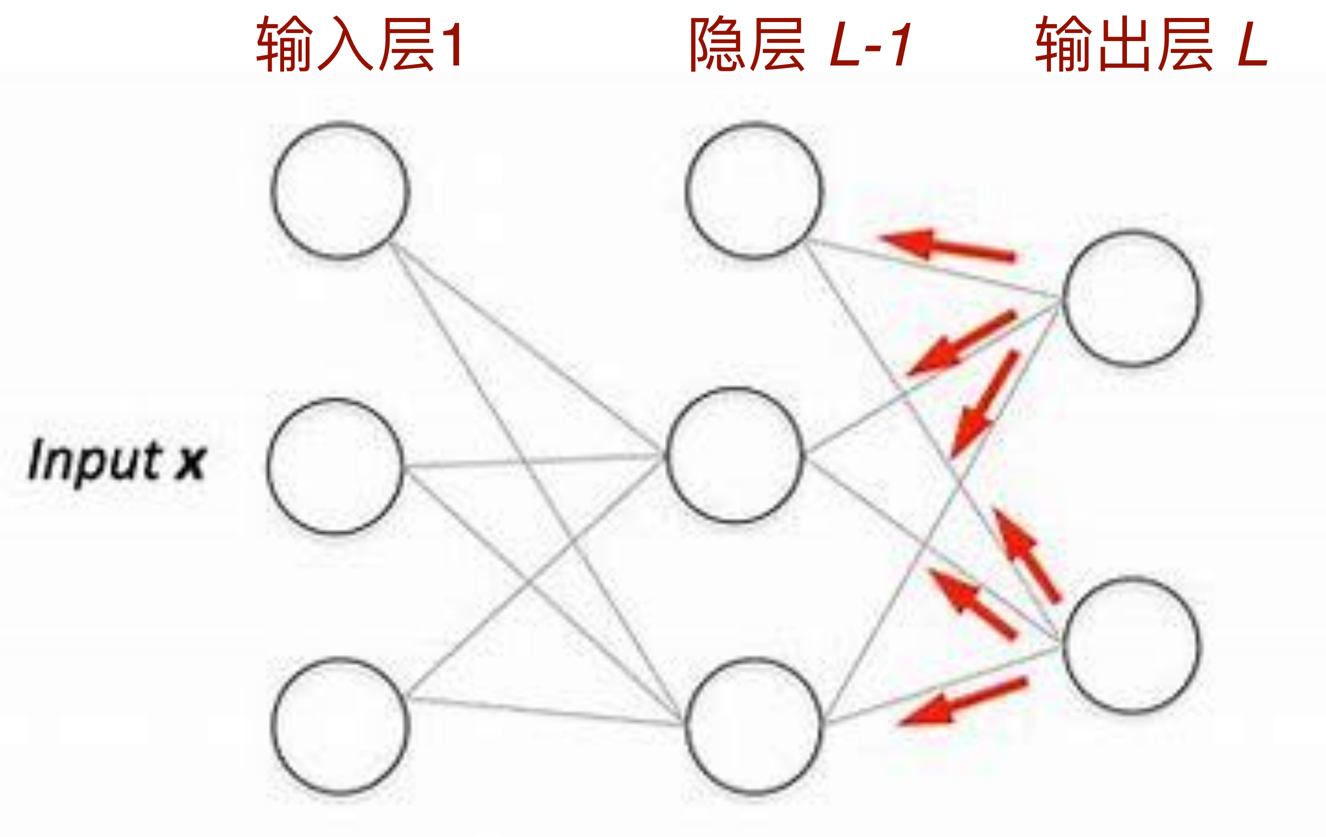
误差函数

$$E = E(W^L, b^L; W^{L-1}, b^{L-1}; \ldots; W^1, b^1)$$

思考：比较单层神经网络模型，如何求L-1层前参数偏导数?

3.19 误差反向传播法 (Error backpropagation algorithm)

误差由输出层反向传递到隐层，逐层用误差函数求该层的偏导数，一直到输出层，不是一次全部求所有层的偏导数。



误差函数

$$C = \frac{1}{2} f\left[a^{(L)} - y\right]^2$$

$$a^{(L)} = \sigma\left[Z^{(L)}\right]$$

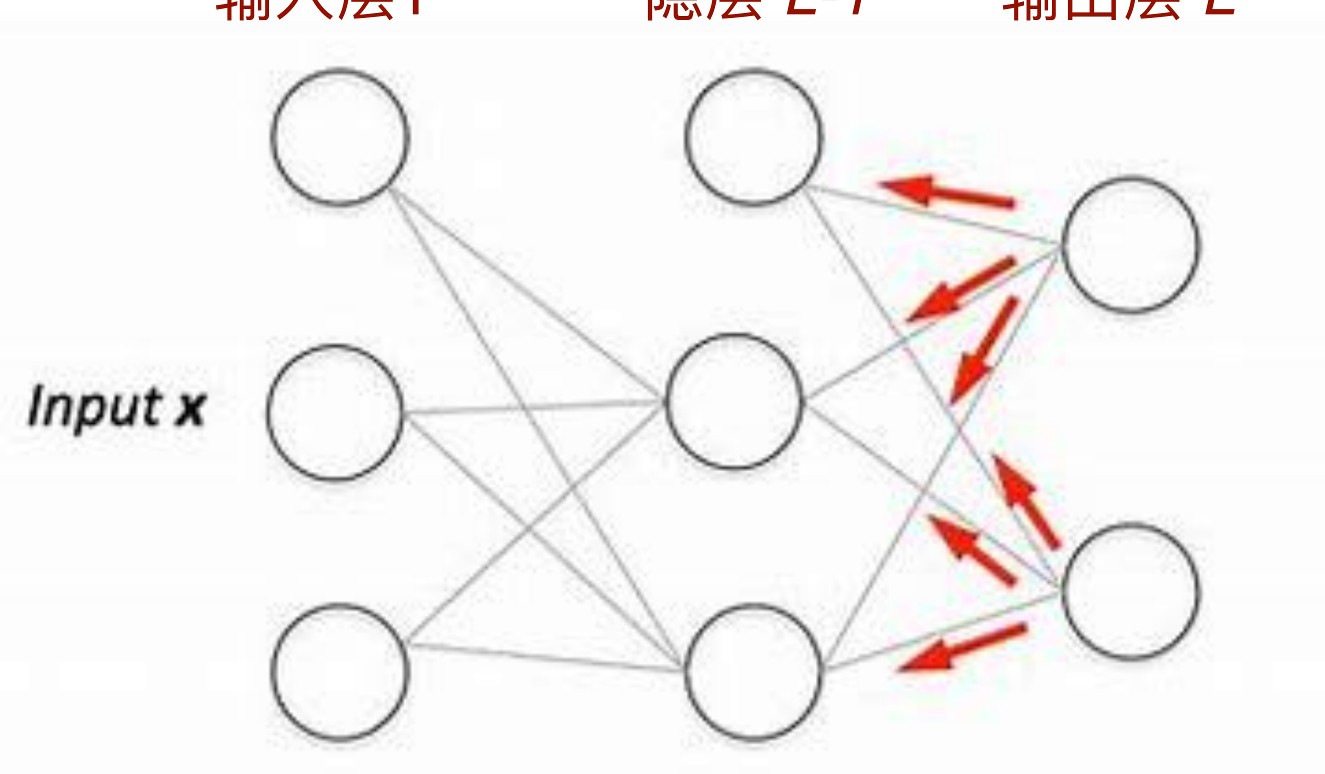
$$Z^{(L)} = W^{(L)} \times a^{(L-1)} + b^{(L)}$$

The chain rule：

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial Z} \cdot \frac{\partial Z}{\partial w}$$

3.20 误差反向传播法

输出层（L层）参数的偏导数



误差函数

$$C = f\left[W^{(L)}, b^{(L)}, a^{(L-1)}\right]$$

$$\frac{\partial C}{\partial W^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial Z^{(L)}} \cdot \frac{\partial Z^{(L)}}{\partial W^{(L)}}$$

$$= \left[a^{(L)} - y\right] \cdot \sigma'\left[Z^{(L)}\right] \cdot a^{(L-1)}$$

$$\frac{\partial C}{\partial b^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial Z^{(L)}} \cdot \frac{\partial Z^{(L)}}{\partial b^{(L)}}$$

$$= \left[a^{(L)} - y\right] \cdot \sigma'\left[Z^{(L)}\right]$$

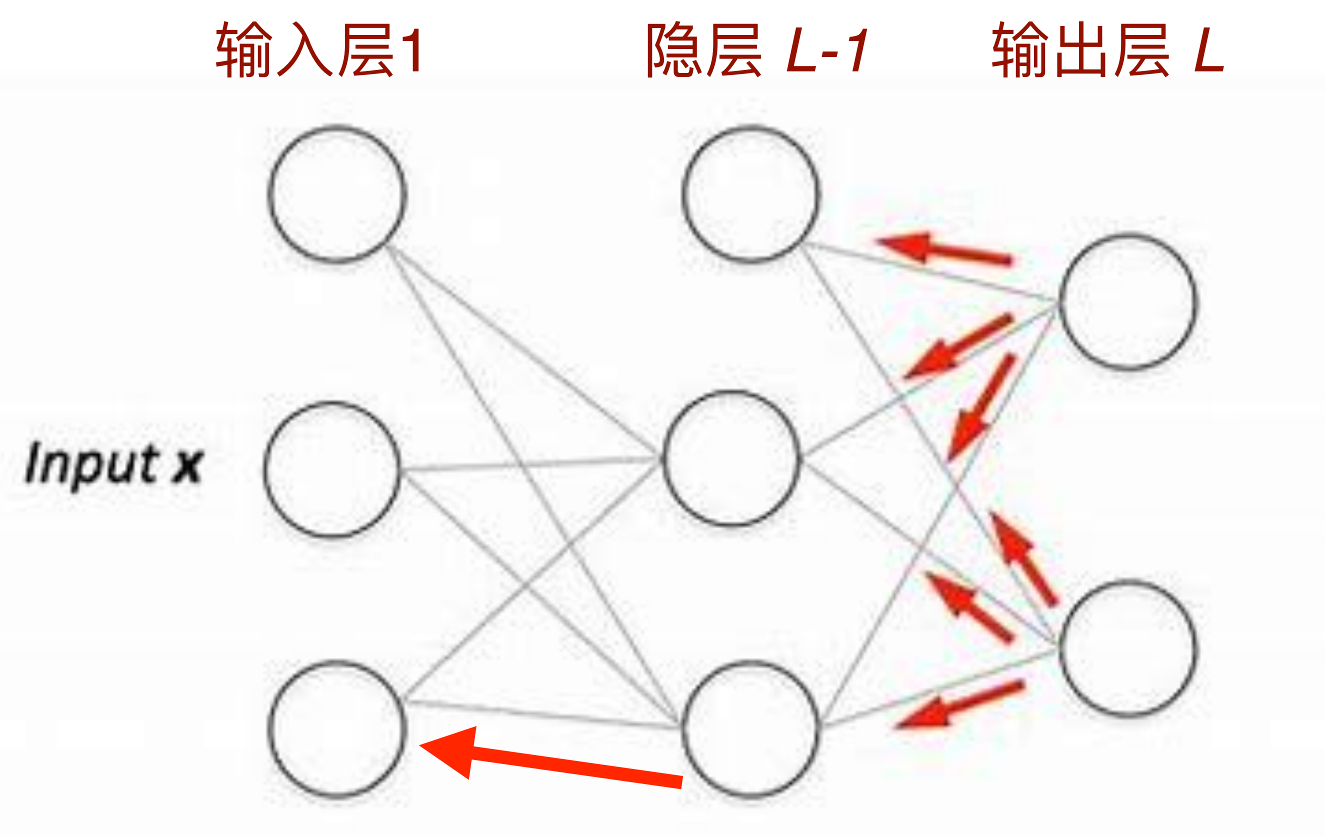
$$\frac{\partial C}{\partial a^{(L-1)}} = \frac{\partial C}{\partial a^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial Z^{(L)}} \cdot \frac{\partial Z^{(L)}}{\partial a^{(L-1)}}$$

$$= \left[a^{(L)} - y\right] \cdot \sigma'\left[Z^{(L)}\right] \cdot W^{(L)}$$

（L-1）层计算用

3.21 误差反向传播法

隐层（L-1层）参数的偏导数



对多隐层网络，
反复计算直到输出层。

L层计算所得

$$\frac{\partial C}{\partial W^{(L-1)}} = \frac{\partial C}{\partial a^{(L-1)}} \cdot \frac{\partial a^{(L-1)}}{\partial Z^{(L-1)}} \cdot \frac{\partial Z^{(L-1)}}{\partial W^{(L-1)}} = \left[a^{(L-1)} - y\right] \cdot \sigma'\left[Z^{(L-1)}\right] \cdot a^{(L-2)}$$

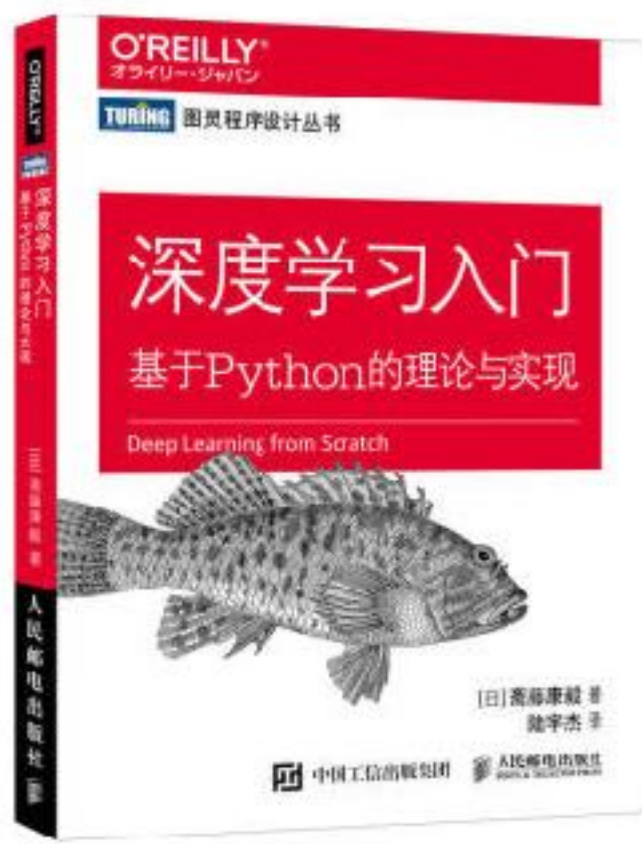
$$\frac{\partial C}{\partial b^{(L-1)}} = \frac{\partial C}{\partial a^{(L-1)}} \cdot \frac{\partial a^{(L-1)}}{\partial Z^{(L-1)}} \cdot \frac{\partial Z^{(L-1)}}{\partial b^{(L-1)}} = \left[a^{(L-1)} - y\right] \cdot \sigma'\left[Z^{(L-1)}\right]$$

$$\frac{\partial C}{\partial a^{(L-2)}} = \frac{\partial C}{\partial a^{(L-1)}} \cdot \frac{\partial a^{(L-1)}}{\partial Z^{(L-1)}} \cdot \frac{\partial Z^{(L-1)}}{\partial a^{(L-2)}} = \left[a^{(L-1)} - y\right] \cdot \sigma'\left[Z^{(L-1)}\right] \cdot W^{(L-1)}$$

（L-2）层计算用

实践练习：误差反向传播算法的Python 程序

中国工信出版集团 人民邮电出版社



http://www.ituring.com.cn/book/1921

请按链接下载参考书全部程序的源代码，并实践第5章5.7节的相关程序。