

Foundation of Artificial Intelligence

Prof. Fangshi Wang

Beijing Jiaotong University

Email: fshwang@bjtu.edu.cn

Image Recognition 图像识别

Video-18mins

观看网易公开课视频：如何教计算机理解图片（李飞飞）

http://open.163.com/movie/2015/3/Q/R/MAKN9A24M_MAKN9QAQR.html

图像识别的理想目标

就是让计算机像人一样理解图像



我们看到的

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

计算机看到的

图像识别的实际目标

让计算机将语义概念相似的图像划分为同一类别。

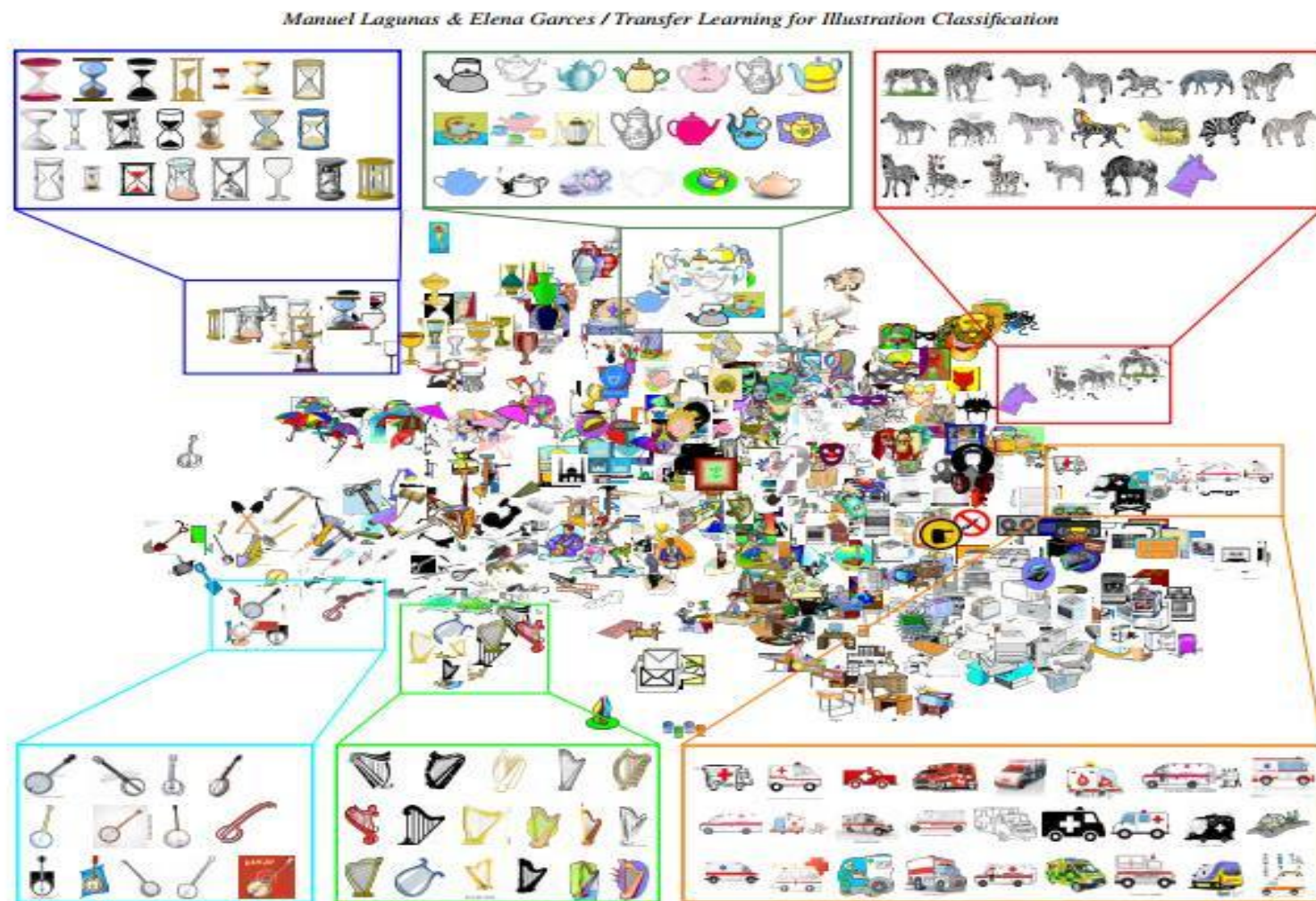


Figure 6: t-SNE algorithm on the image descriptors of the optimized network. The boxes show groups of images of the same class that have been grouped together after applying the algorithm. This shows us that the network is able to understand the low-level image characteristics and, consequently, that these image descriptors can be classified with a support vector machine obtaining great results.

图像识别面临挑战

Semantic Gap (语义鸿沟现象)

- ◆ 图像的**底层视觉特性**和**高层语义概念**之间的鸿沟.
- ◆ 例如: 相似的底层视觉统计特性(color, texture, shape, ...), 却对应于不同的语义概念.



图像识别面临挑战

Semantic Gap (语义鸿沟现象)

又如：不相似的视觉特性，却对应于相同的语义概念

不同视角 view



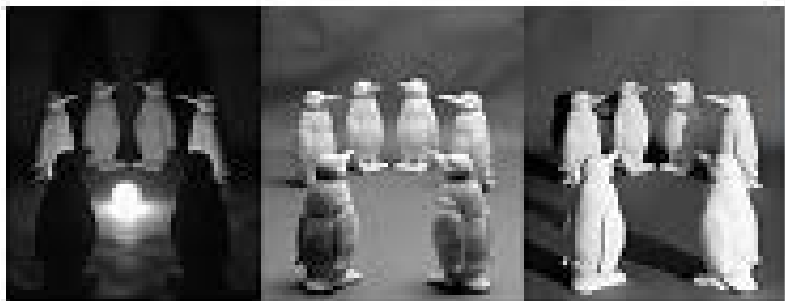
不同大小 size



形变 deformation 遮挡 occlusion



不同光照 illumination



背景干扰



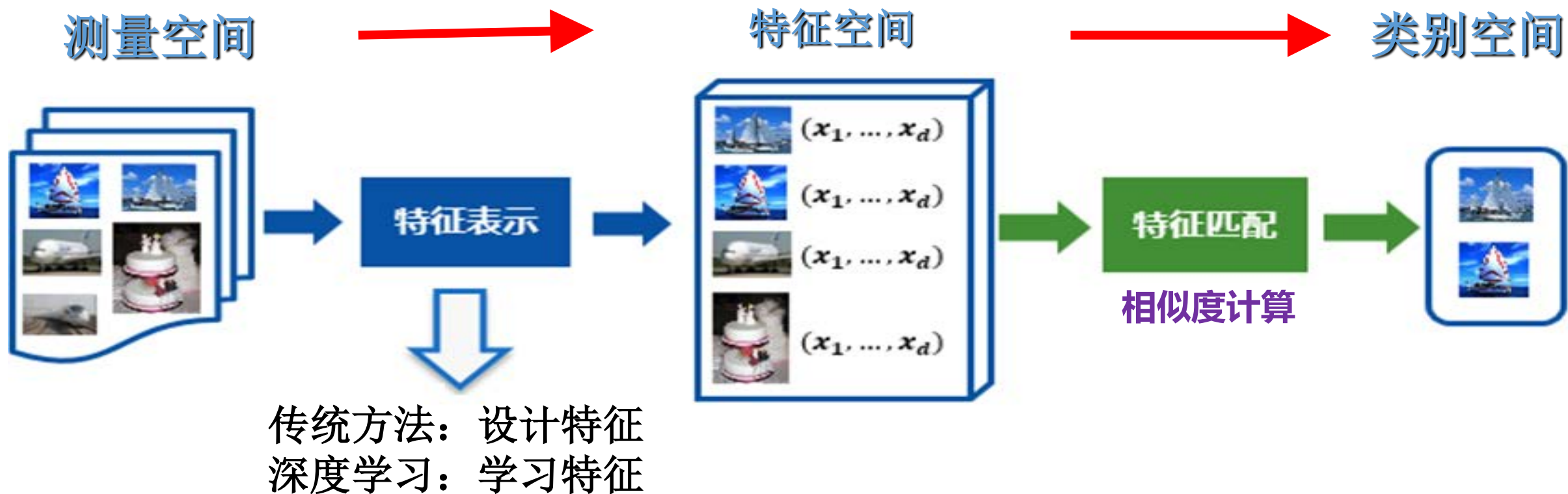
同类异形 furniture



background interference

Different shape, same category

图像识别的基本框架

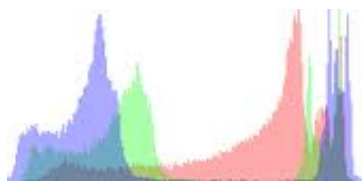


Features (特征): 颜色、形状、纹理.....

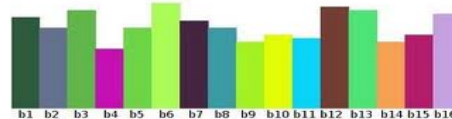
应用场景: 场景识别、目标识别、人脸识别.....

早期图像识别技术

◆全局特征提取：用全局的底层视觉统计特征表示图像



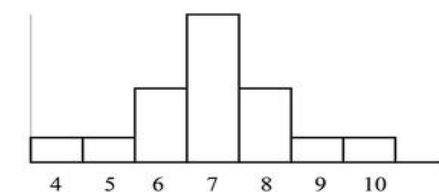
颜色



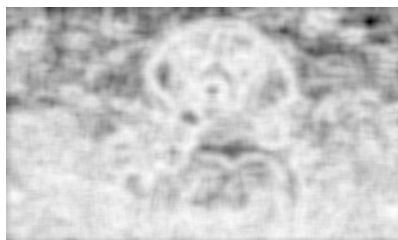
颜色直方图



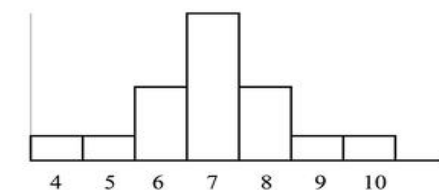
形状



形状直方图



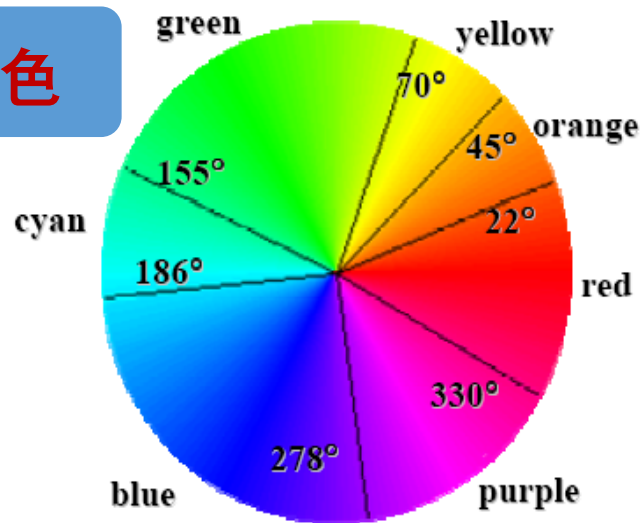
纹理



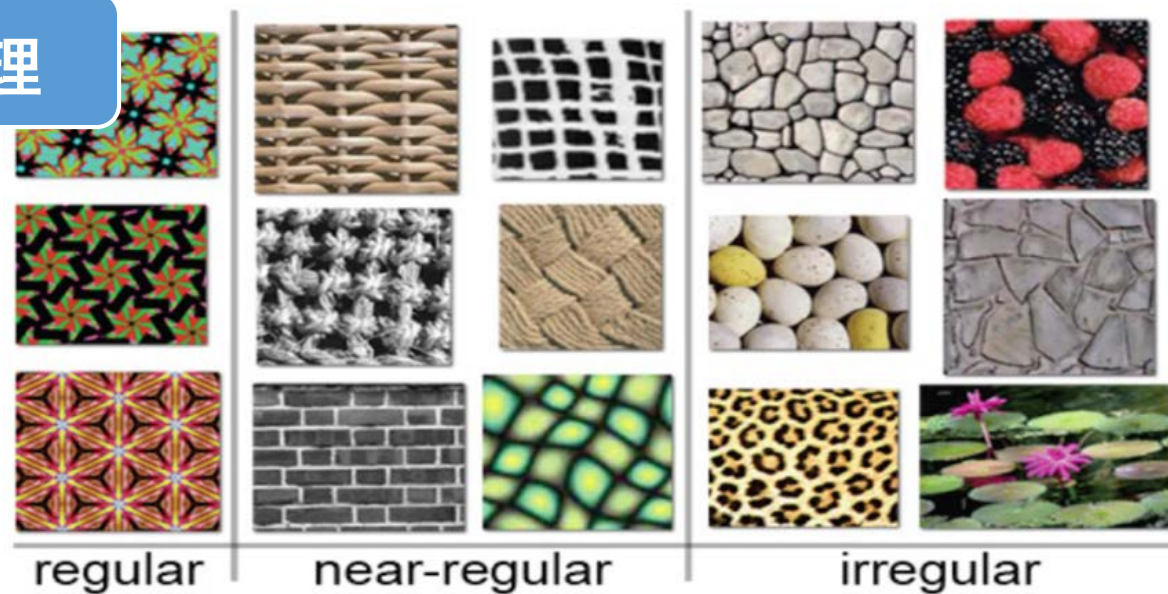
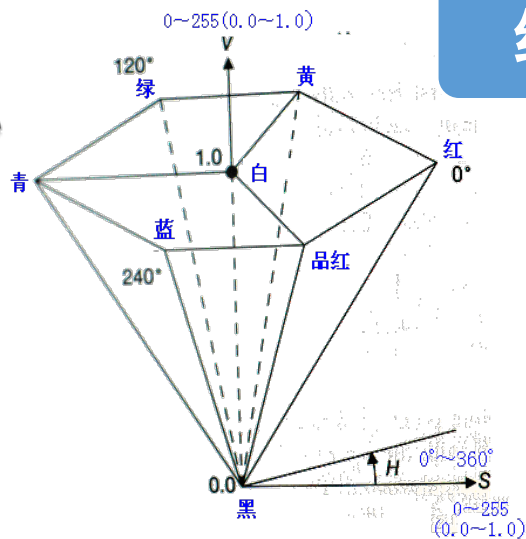
纹理向量

全局特征示例

颜色

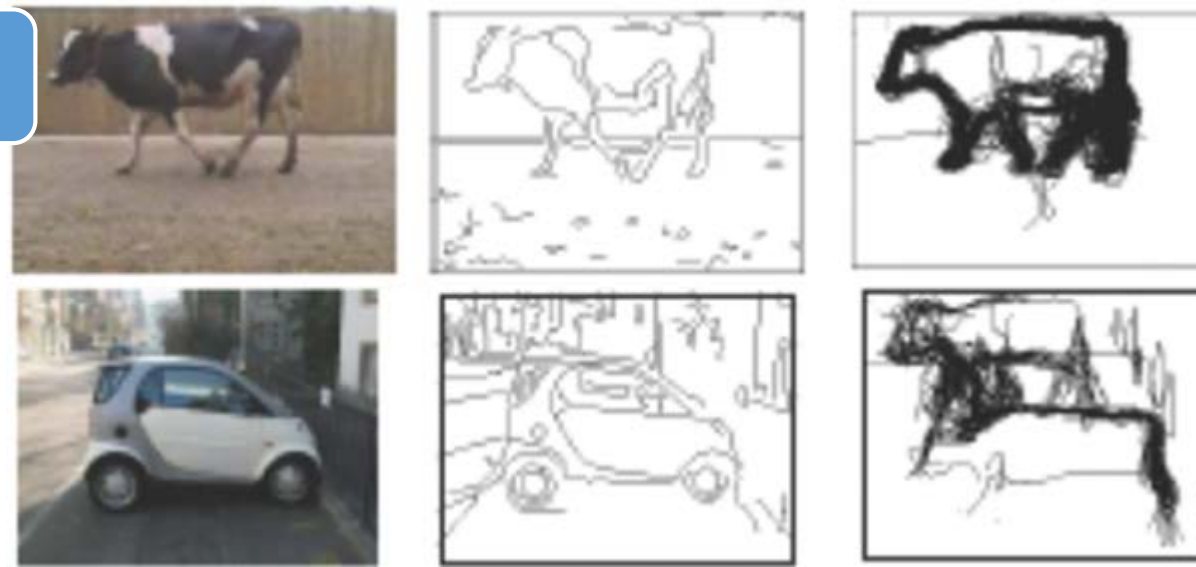
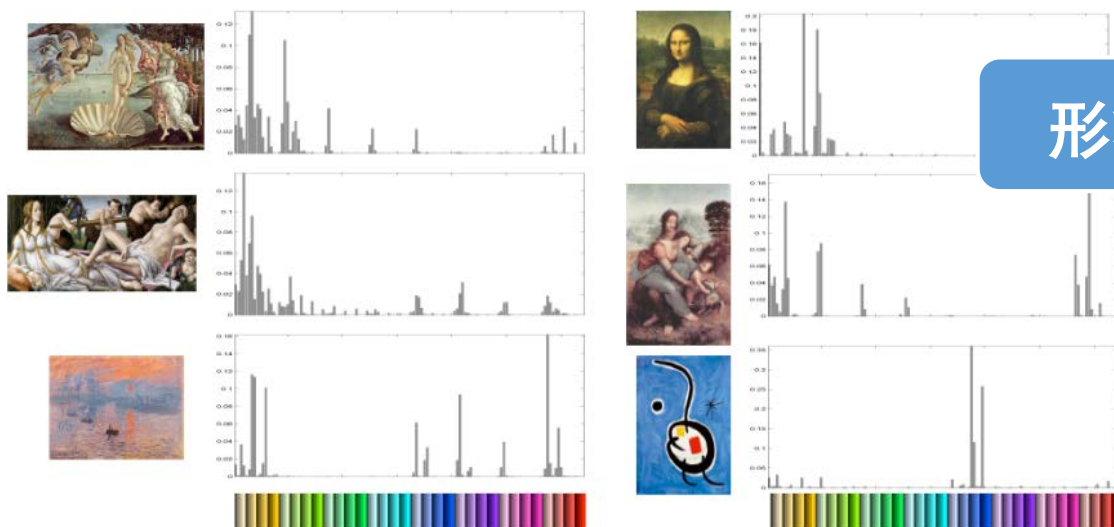


纹理



规则的纹理

形状



早期图像识别技术的问题

问题：全局特征丢掉了图像细节

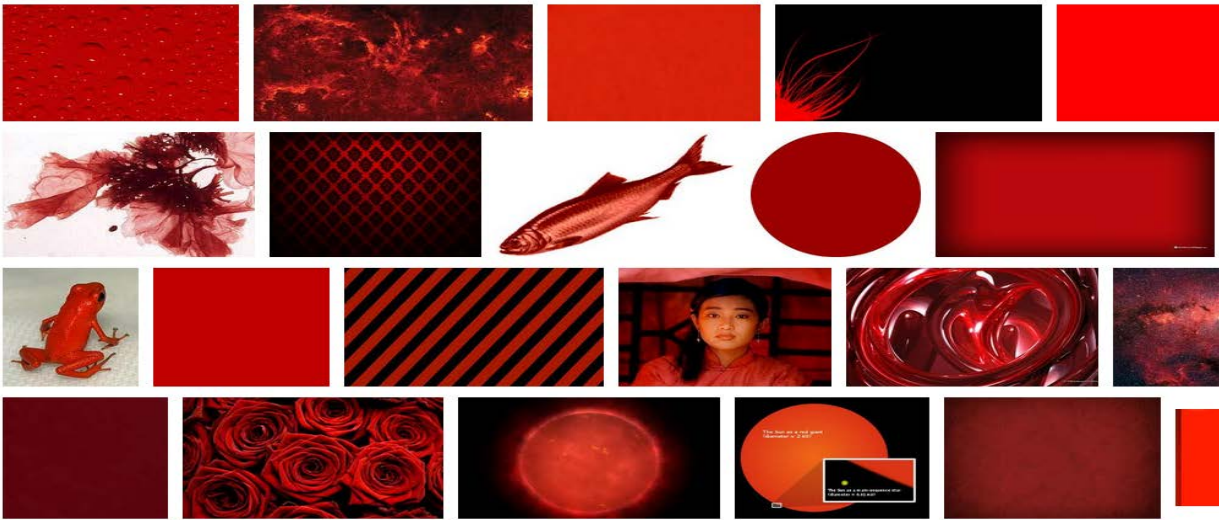
提取颜色特征：颜色直方图



整体偏红色的图片都被找出来了，被认为是日出图像。



正确匹配



错误匹配

深度学习与图像识别

深度学习起源与发展

深度学习在图像处理领域的应用

如何使用深度学习识别图像

深度学习起源与发展



1957 Perceptron

1974 Backpropagation

Convolution Neural Networks for
Handwritten Recognition

1998

Google Brain Project on
16k Cores

2012

awkward silence (AI Winter)

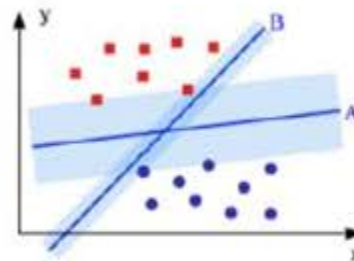
1969

Perceptron criticized



1995

SVM reigns



2006

Restricted
Boltzmann
Machine



2012

AlexNet wins
ImageNet
IMAGENET



深度学习在图像处理领域的应用



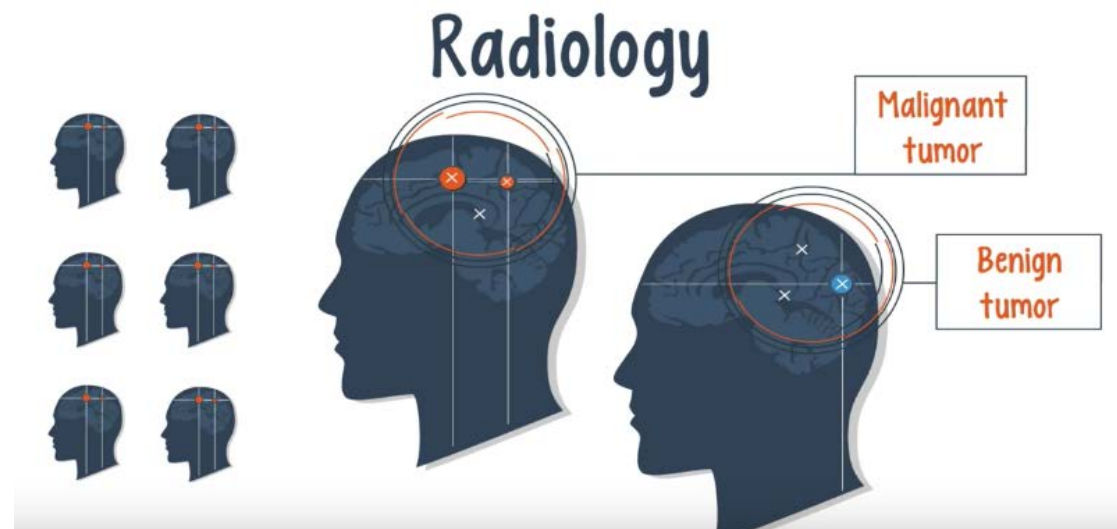
Predicted Tags

no person table elegant
indoors luxury furniture
fashion decoration tableware
party

Similar Images



图片检索



异常肿瘤或癌细胞识别

深度学习在图像处理领域的应用



图像的语义描述



图片着色

深度学习用于图像识别

使用机器学习（深度学习）的目的：寻找一个合适的函数/模型

◆ 数字识别

$f($



$)=$

“3”

◆ 猫狗识别

$f($



$)=$

“cat”

◆ 人脸识别

$f($



$)=$

“Li Chen”

手写体数字识别：学习任务

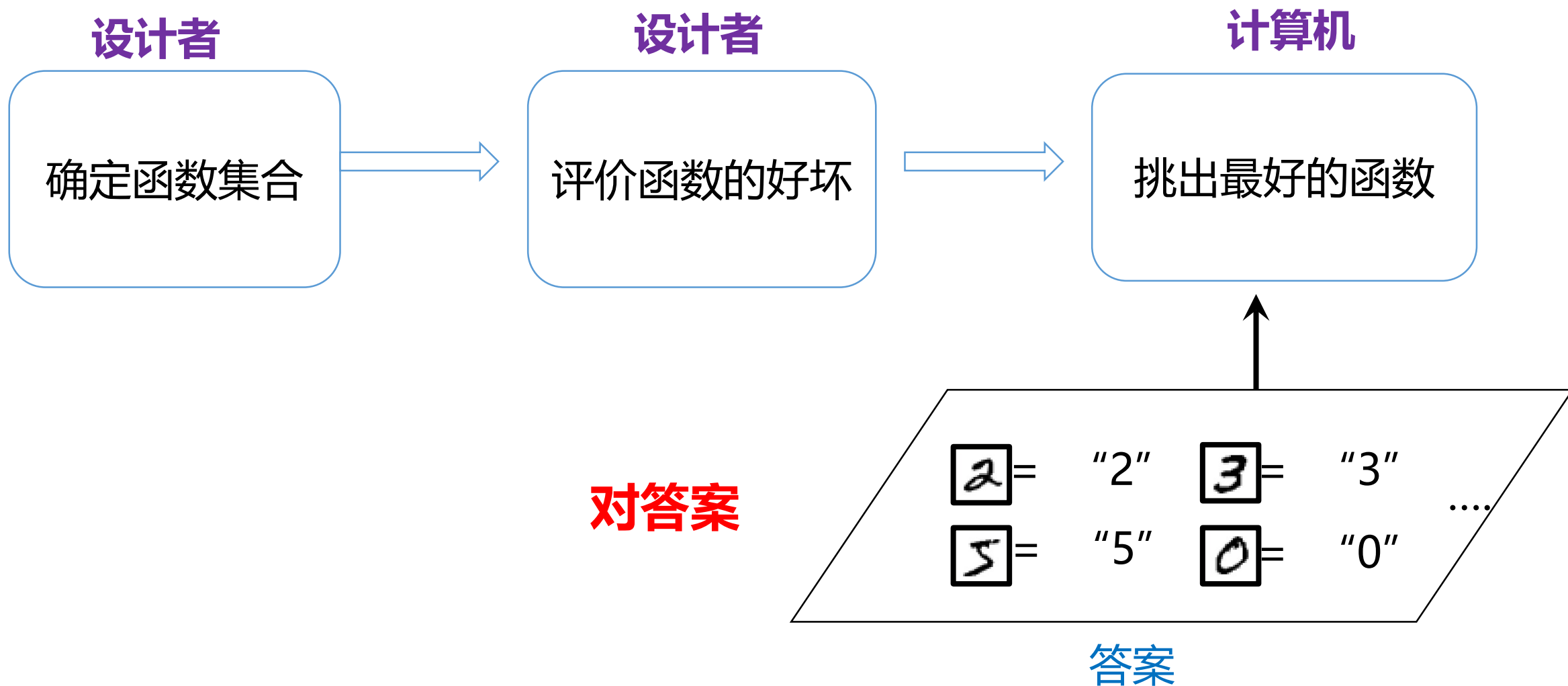
学习任务：寻找手写数字识别函数，候选函数集合为 ($f, g, h \dots\dots$)

$$f\left(\boxed{\text{2}}\right) = \text{"烫烫烫烫"}$$

$$g\left(\boxed{\text{2}}\right) = \text{"8"}$$

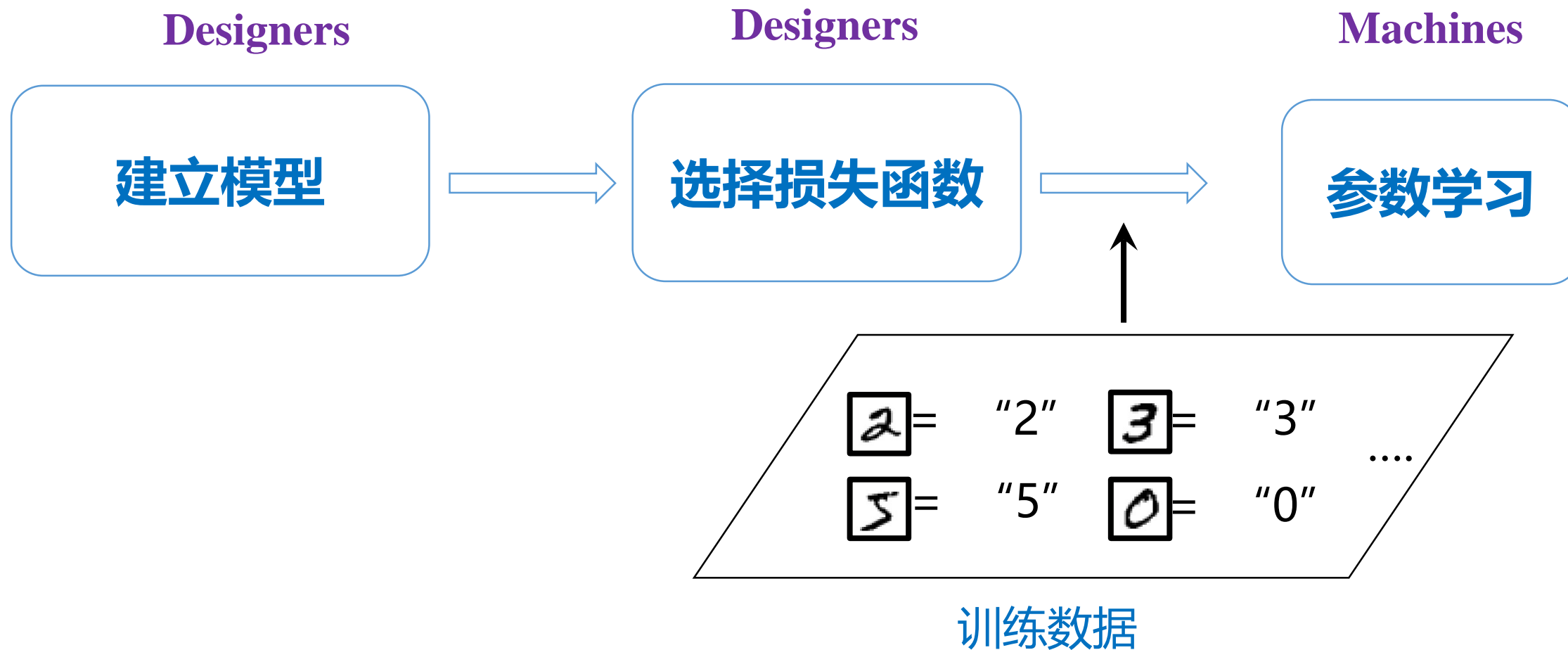
$$h\left(\boxed{\text{2}}\right) = \text{"2"}$$

手写体数字识别：学习过程



手写体数字识别：学习过程

监督学习 (Supervised Learning)



手写体数字识别：学习过程

Designers

建立模型

Designers

选择损失函数

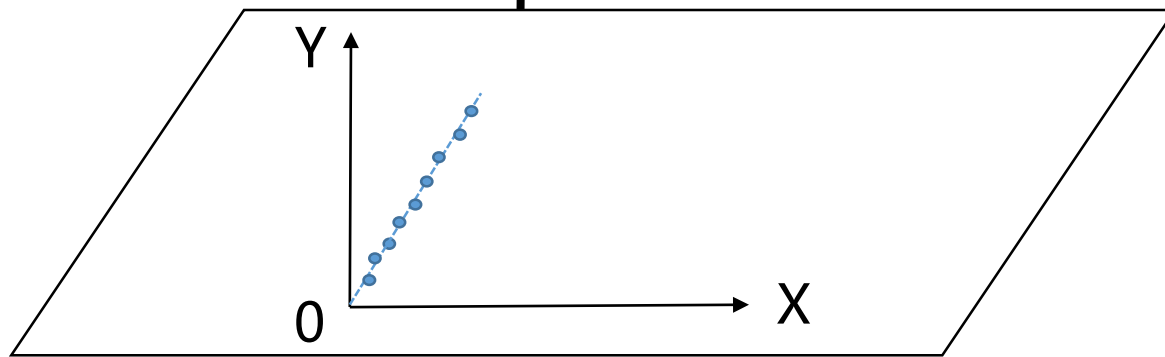
Machines

参数学习

$$\{f(x) = kx \mid k = 1, 2, 3, \dots\}$$

$$\min \sum (y - f(x))^2$$

$$f(x) = 2x$$



答案

深度学习步骤

◆ Build Model （建立模型）

- 确定网络结构
- 确定网络层数、每层的神经元数（即滤波器的个数）

◆ Loss Function （损失函数）

常用损失函数：平方误差，交叉熵

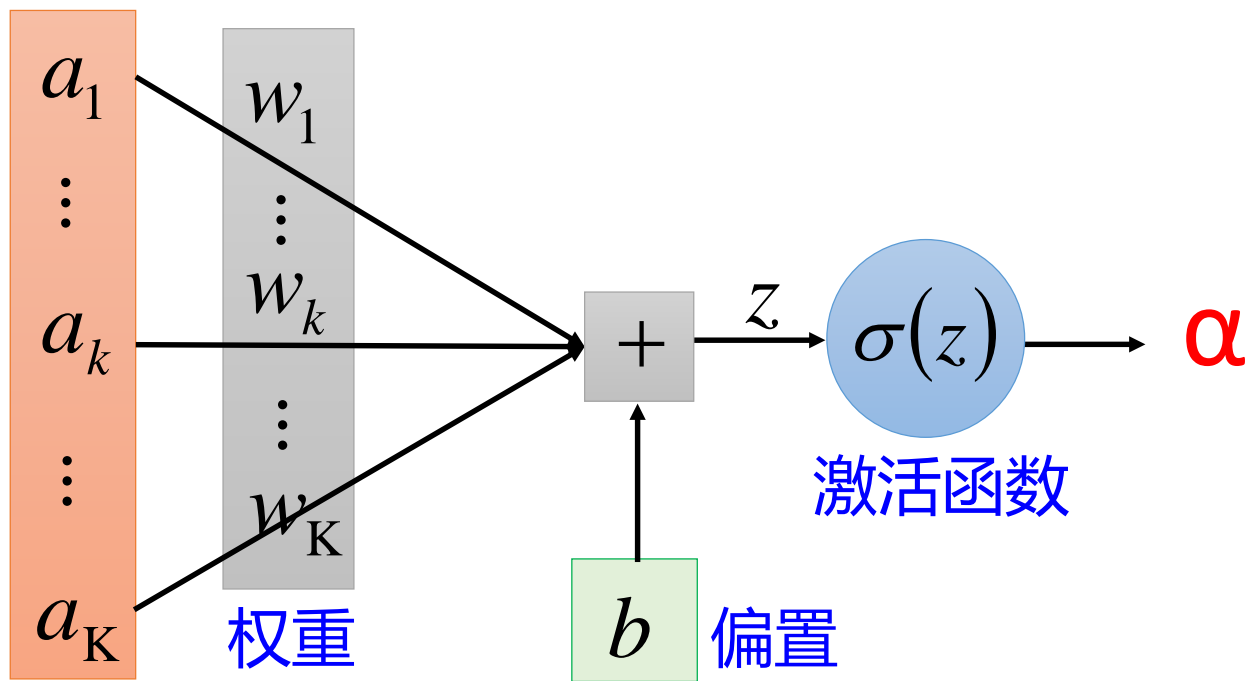
◆ Parameter Learning Algorithm （参数学习方法）

- 梯度下降
- 误差反向传播算法

建立模型

神经元

$$z = a_1w_1 + \dots + a_kw_k + \dots + a_Kw_K + b$$



理解：简单线性函数

$$\mathbf{f}(\mathbf{x}) = \mathbf{k}\mathbf{x} + \mathbf{b}$$

k是斜率，b是截距

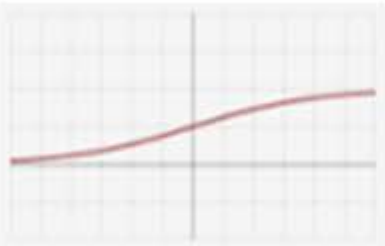
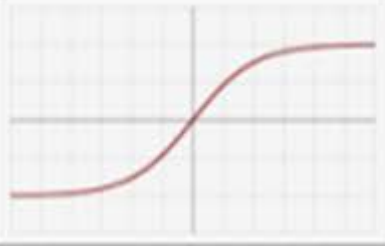
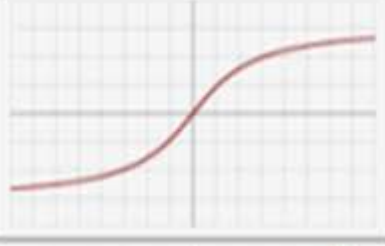


激活函数 (Activation Function)

◆为什么引入激活函数？

- 为增强网络的表达能力，需用激活函数将线性表示转换为非线性表示
- 非线性激活函数需有连续性，因为连续非线性激活函数是可导的，故可用最优化方法来求解

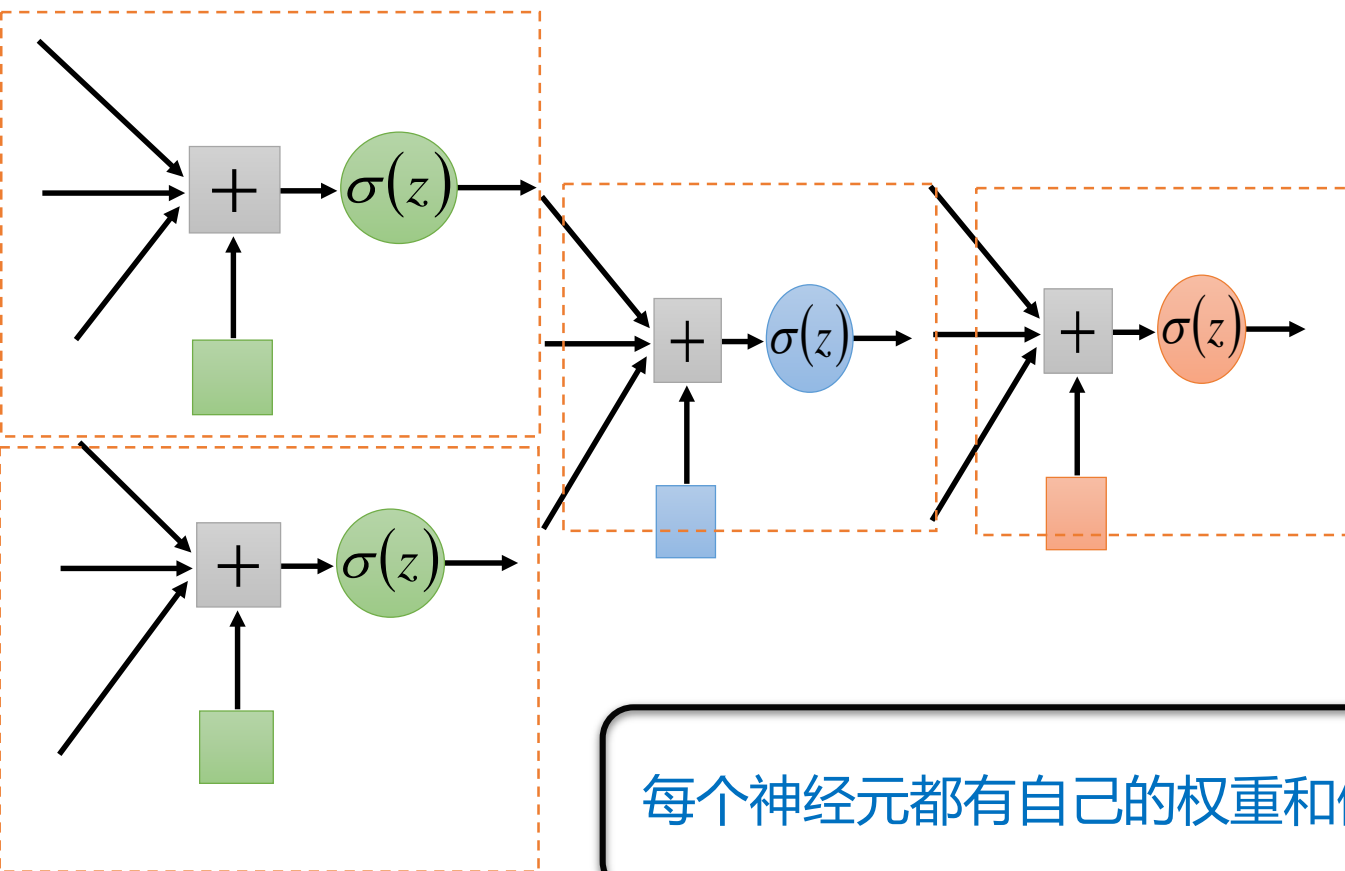
常用的激活函数

求导

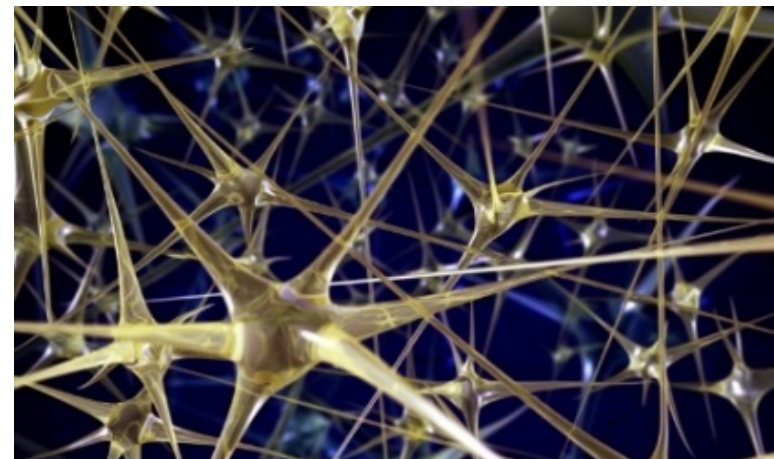
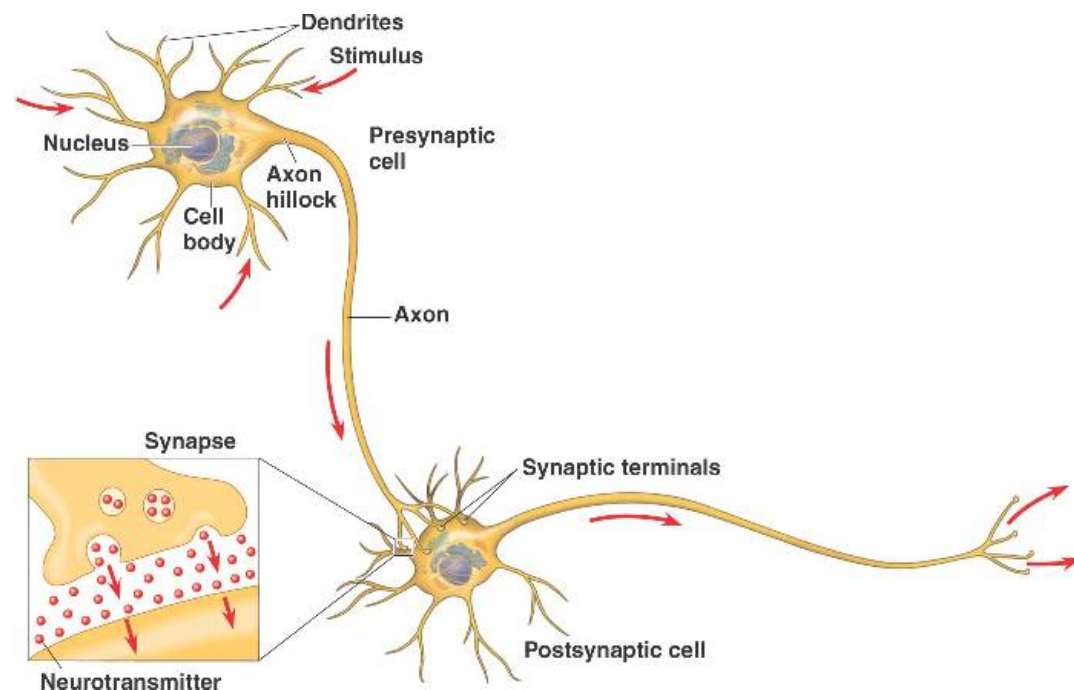
Logistic (a.k.a Soft step) sigmoid		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

建立模型

神经元不同的连接方式构成不同的网络结构.

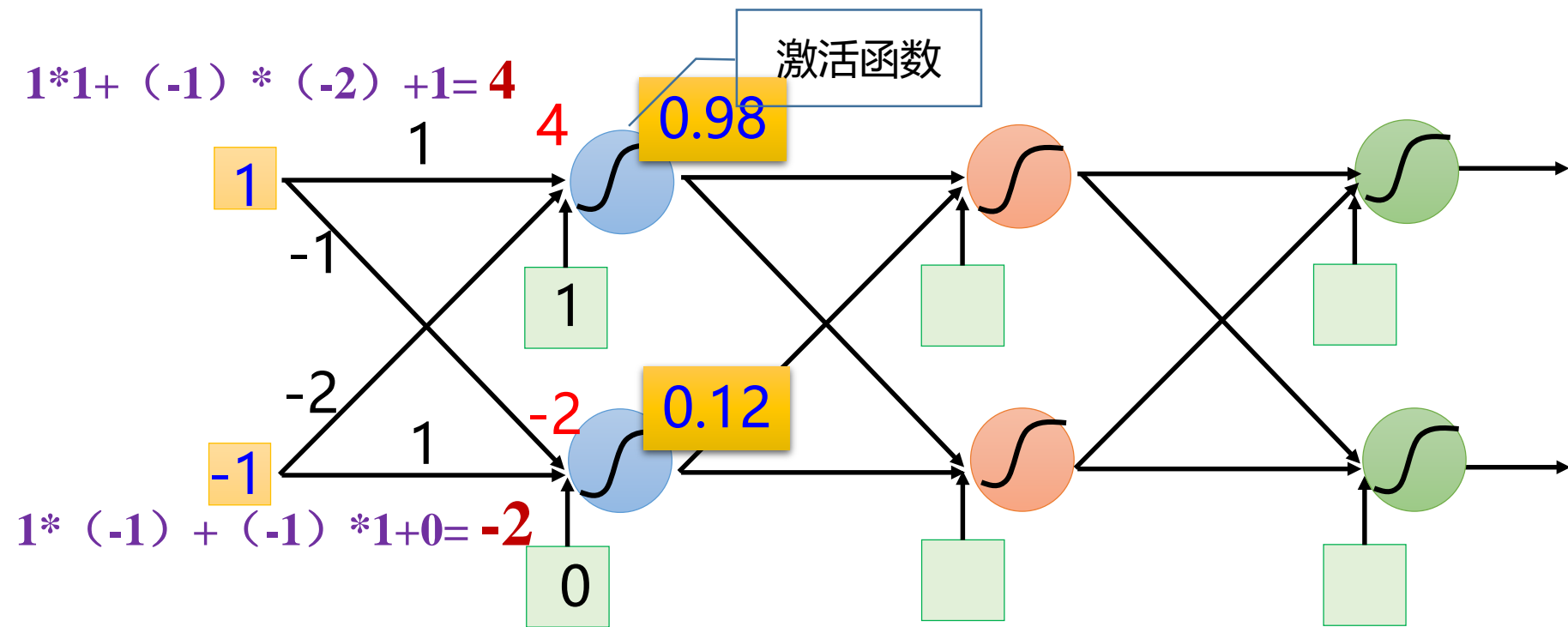


每个神经元都有自己的权重和偏置参数



建立模型

建立前馈神经网络



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

$$\sigma(4) = \frac{1}{1 + e^{-4}} = 0.98$$

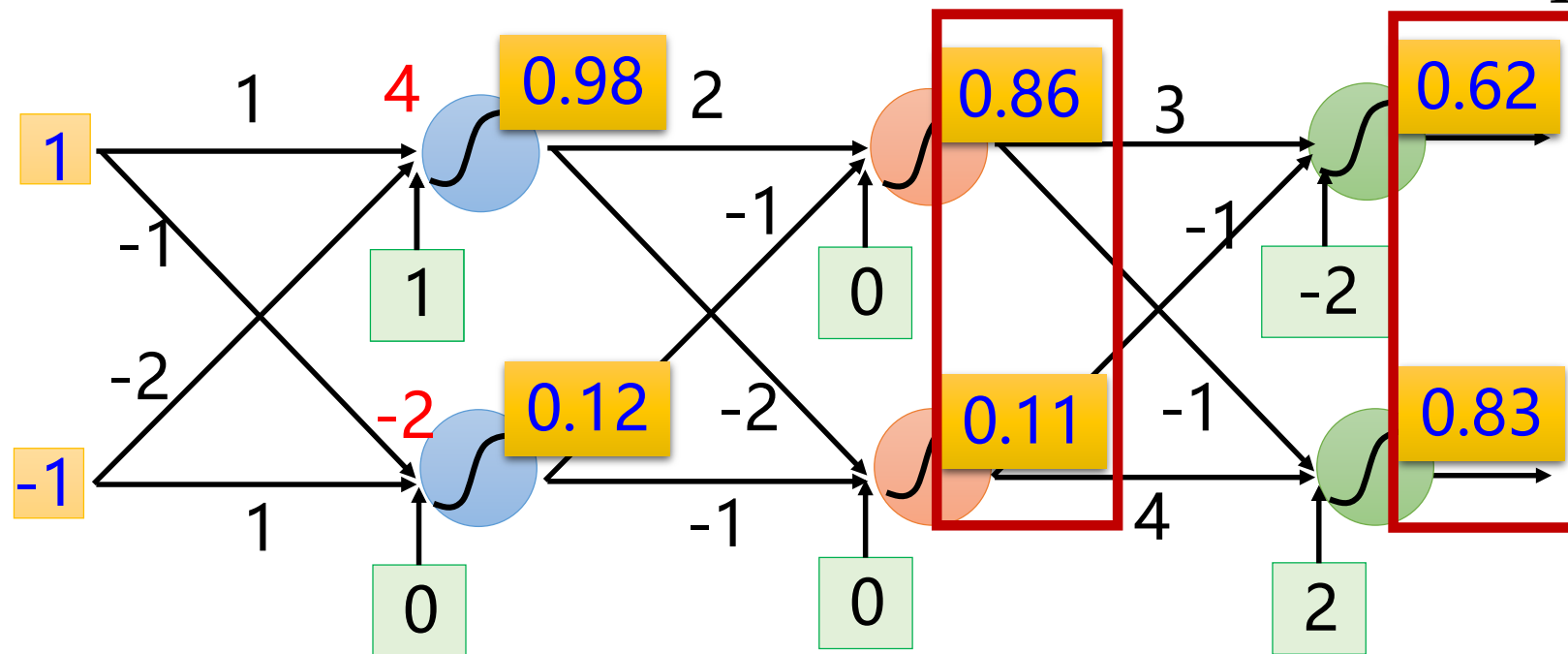
$$\sigma(-2) = \frac{1}{1 + e^2} = 0.12$$

输入信号自左向右依次传播

建立模型

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

$$0.98 * 2 + 0.12 * (-1) + 0 = 1.84 \quad \sigma(1.84) = \frac{1}{1 + e^{-1.84}} = 0.86$$



$$0.86 * 3 + 0.11 * (-1) + (-2) = 0.47$$

$$\sigma(0.47) = \frac{1}{1 + e^{-0.47}} = 0.62$$

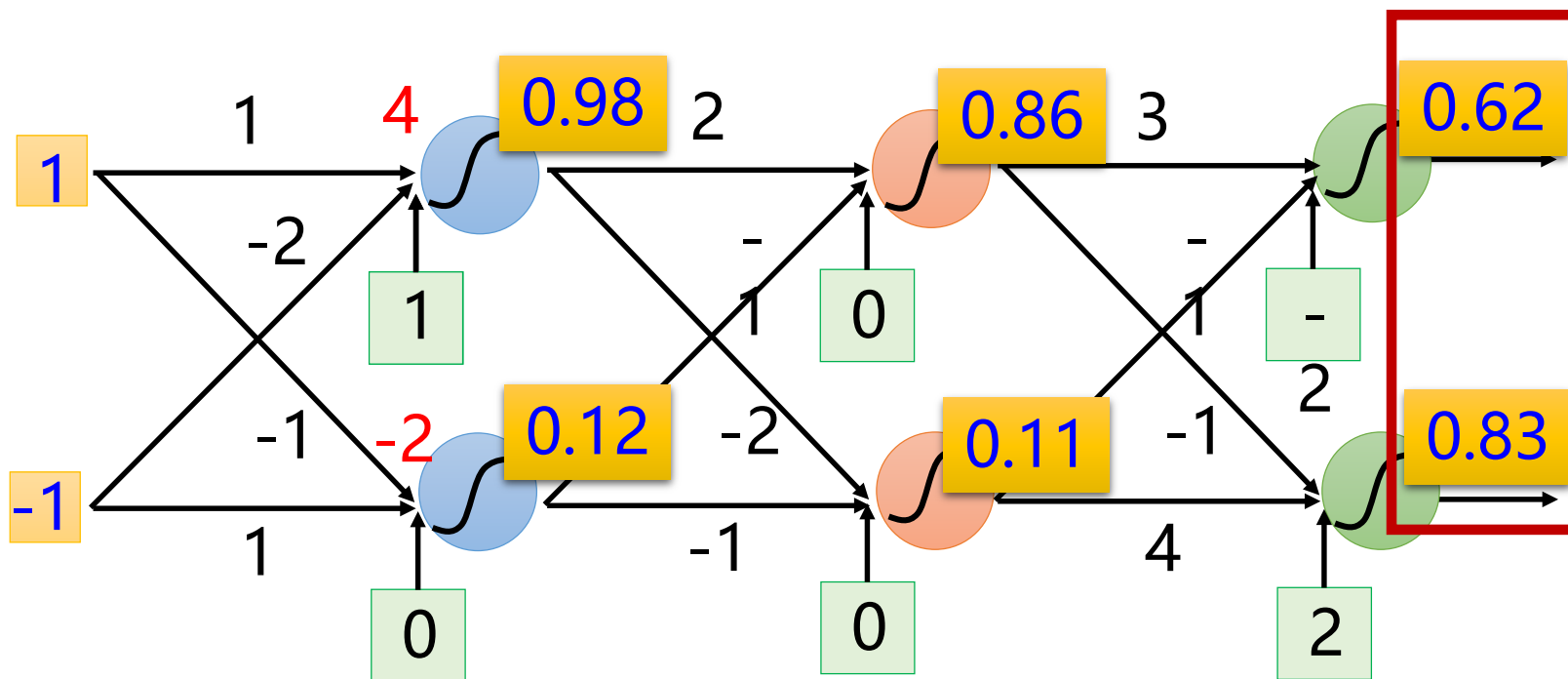
$$0.86 * (-1) + 0.11 * 4 + 2 = 1.58$$

$$\sigma(1.58) = \frac{1}{1 + e^{-1.58}} = 0.83$$

$$0.98 * (-2) + 0.12 * (-1) + 0 = -2.08 \quad \sigma(-2.08) = \frac{1}{1 + e^{2.08}} = 0.11$$

输入信号自左向右依次传播

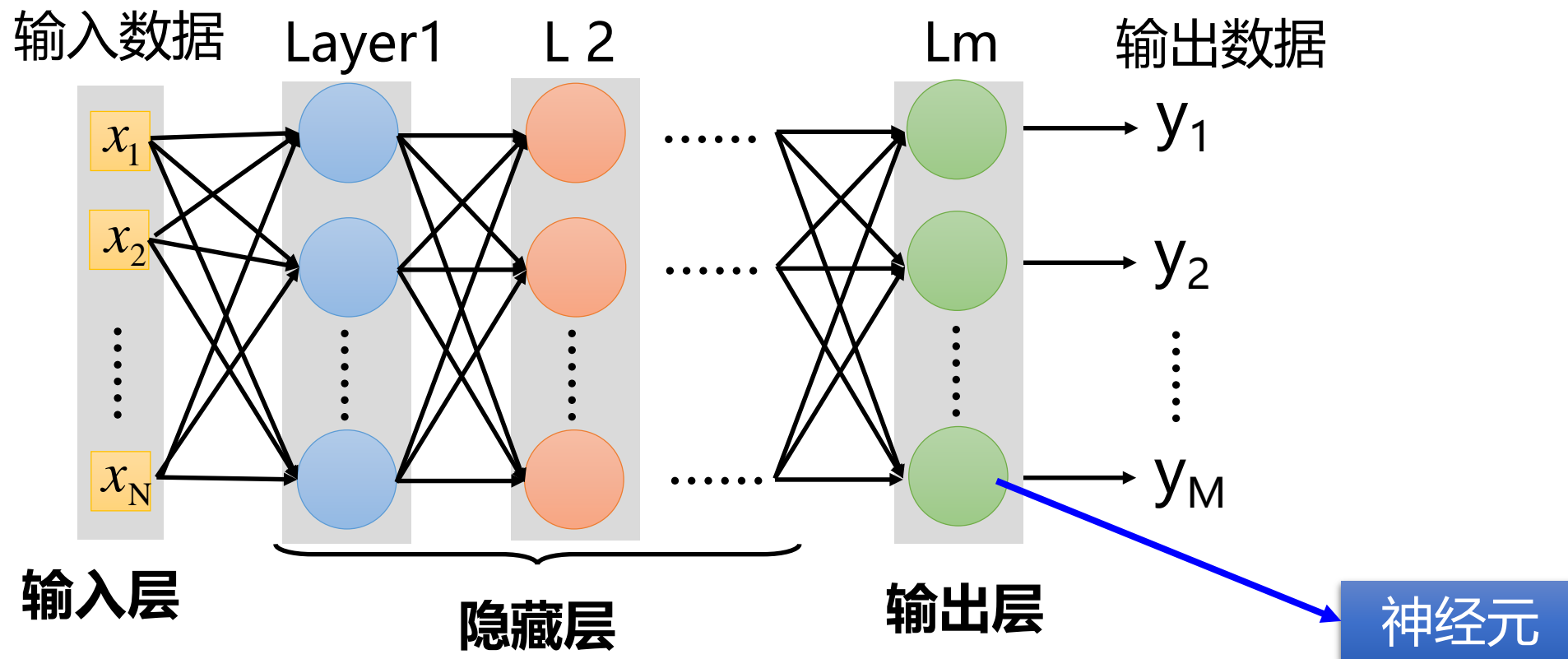
建立模型



这是一个函数模型，如上图确定参数后， $f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix}$

结论：给定一个网络结构，建立了一个模型。

建立模型 (前向NN)



深度意味着隐藏层数很多

NN的层数为 m .

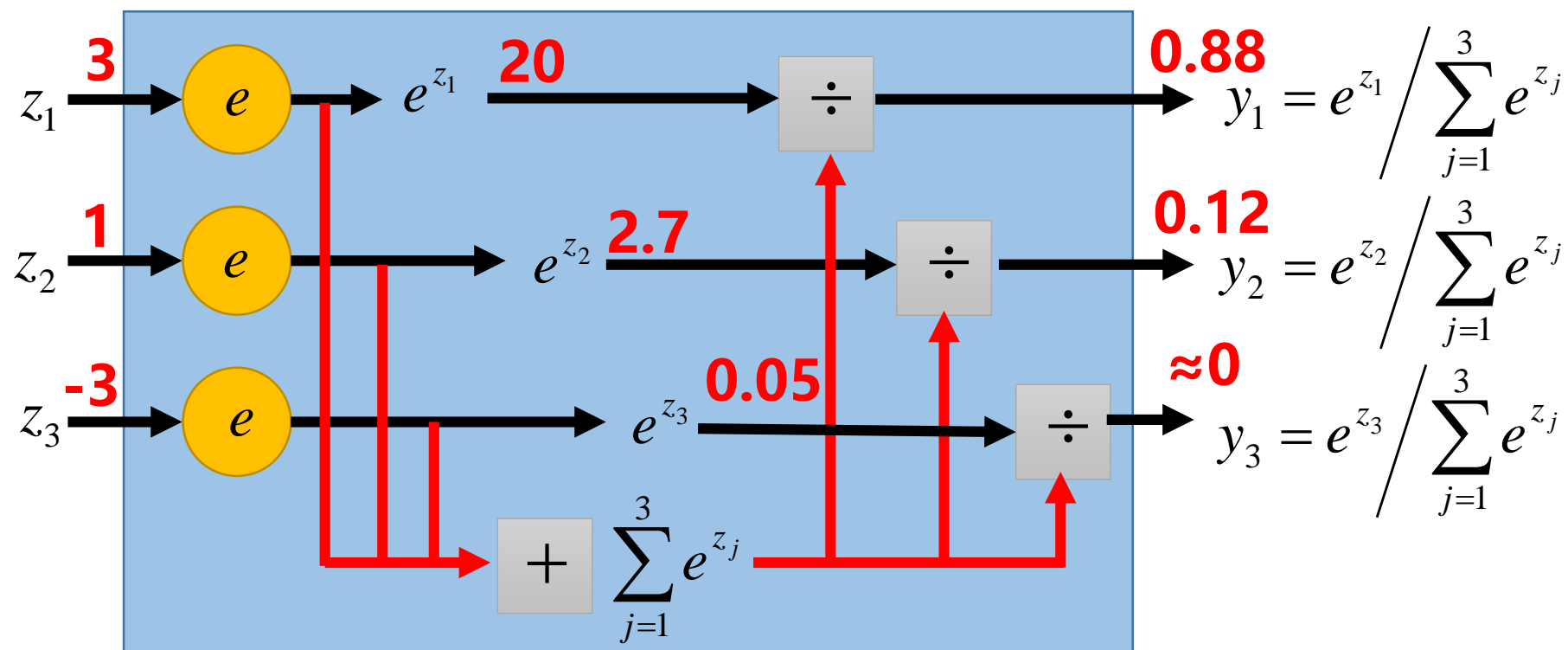
输出层

常用softmax函数作为输出层的激活函数，因为它容易理解、便于计算。

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}, \quad j = 1, \dots, K, \quad K \text{ input}$$

相当于计算概率值：

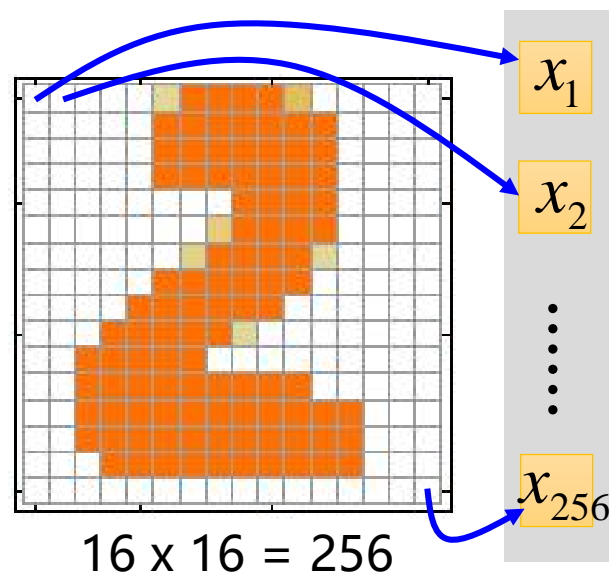
- $1 > y_i > 0$
- $\sum_i y_i = 1$



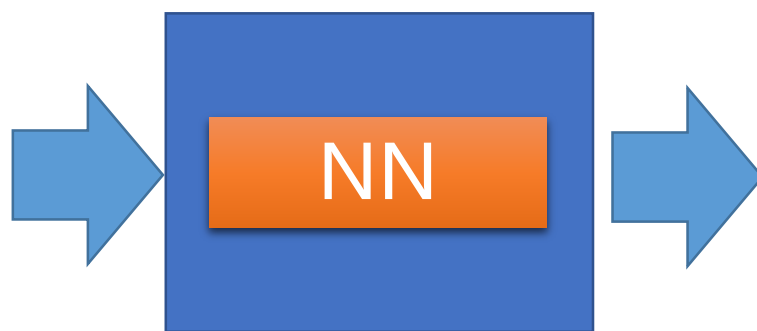
应用示例：手写体数字识别



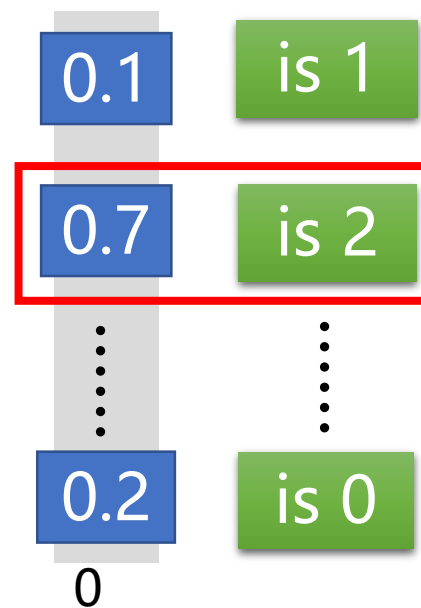
Input



black pixel \rightarrow 1
white pixel \rightarrow 0



Output

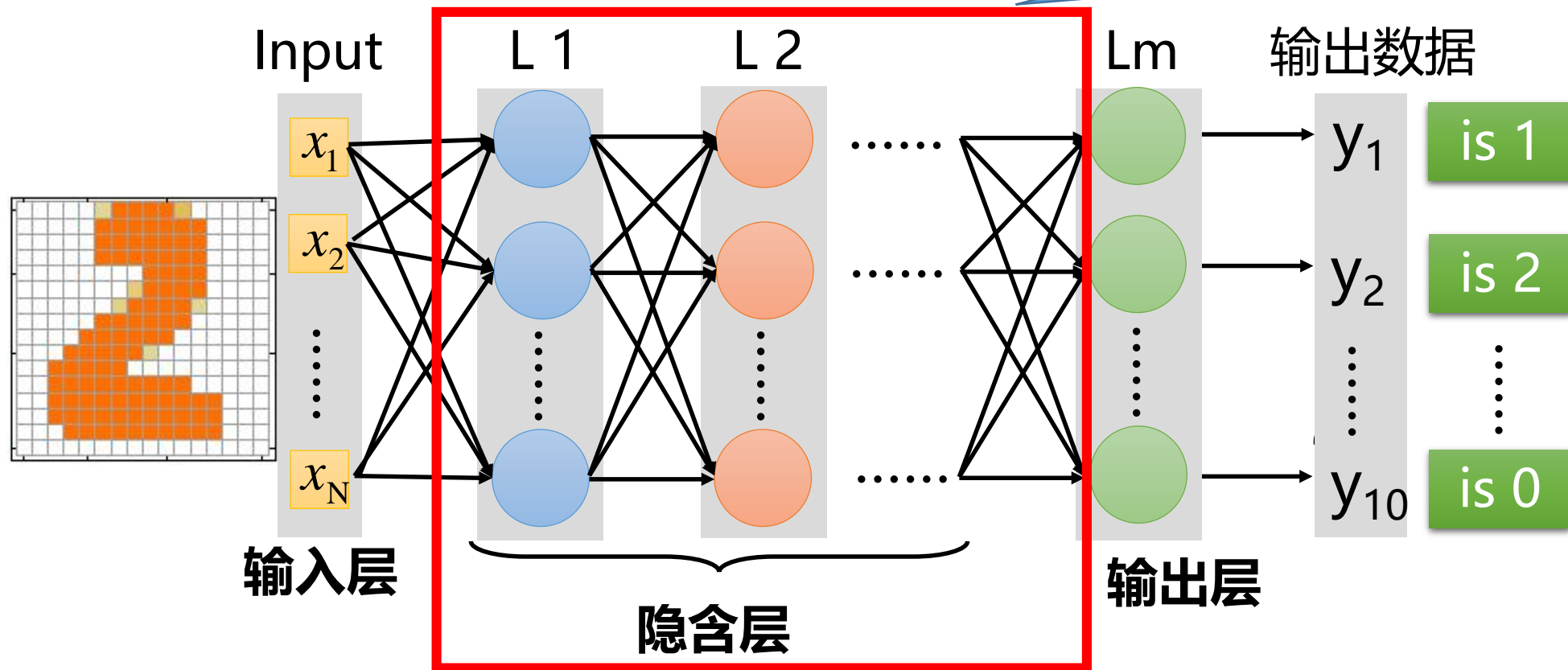


图片被识别成 "2"

每一个输出值代表其对应标签的概率值

应用示例：手写体数字识别

问题：应该设置多少层，多少结点？
是否需要选择其他网络结构？
如CNN/RNN？



人为设置合适的网络结构：层数、每层结点个数、激活函数

深度学习步骤

◆ Build Model （建立模型）

- 确定网络结构
- 确定网络层数、及每层的神经元数

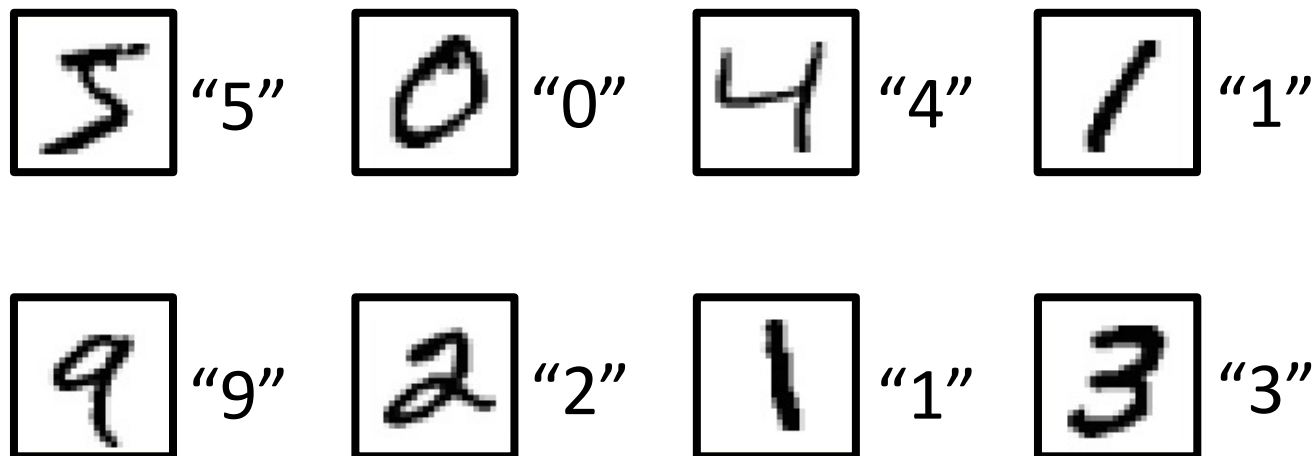
◆ Loss Function （损失函数）

常用损失函数：平方误差，交叉熵

◆ Parameter Learning Algorithm （参数学习方法）

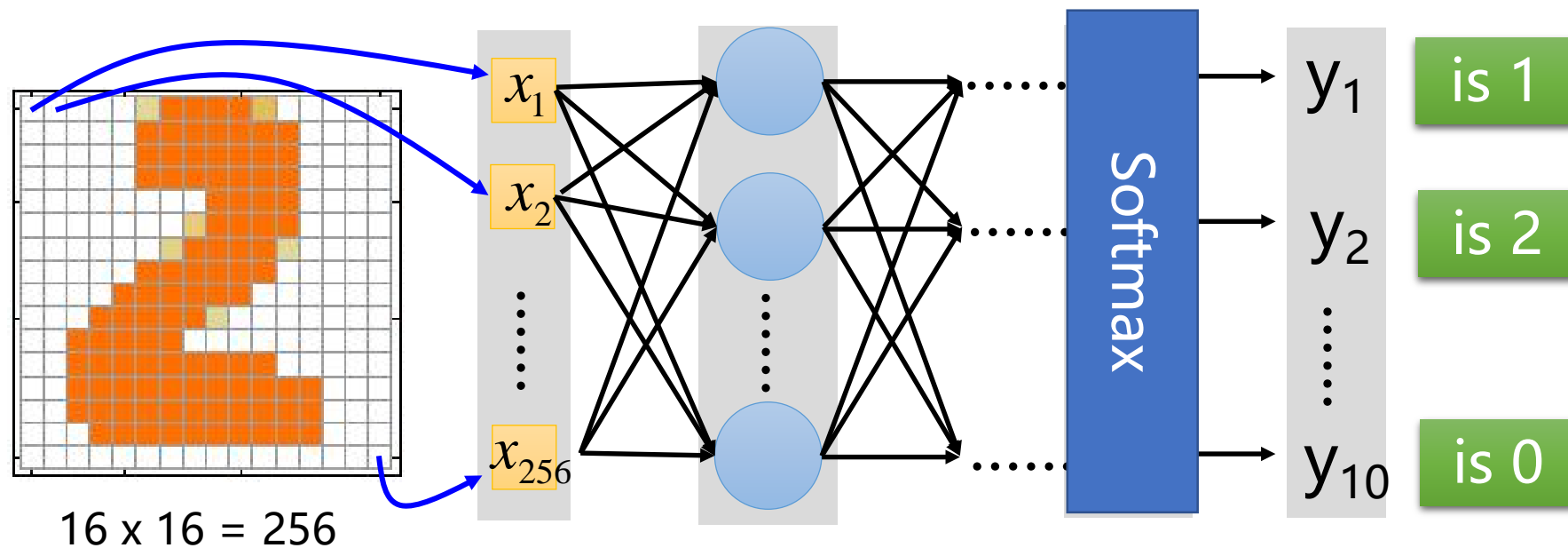
- 梯度下降
- 误差反向传播算法

Loss Function (损失函数)





损失函数依赖于数据集

Loss Function (损失函数)



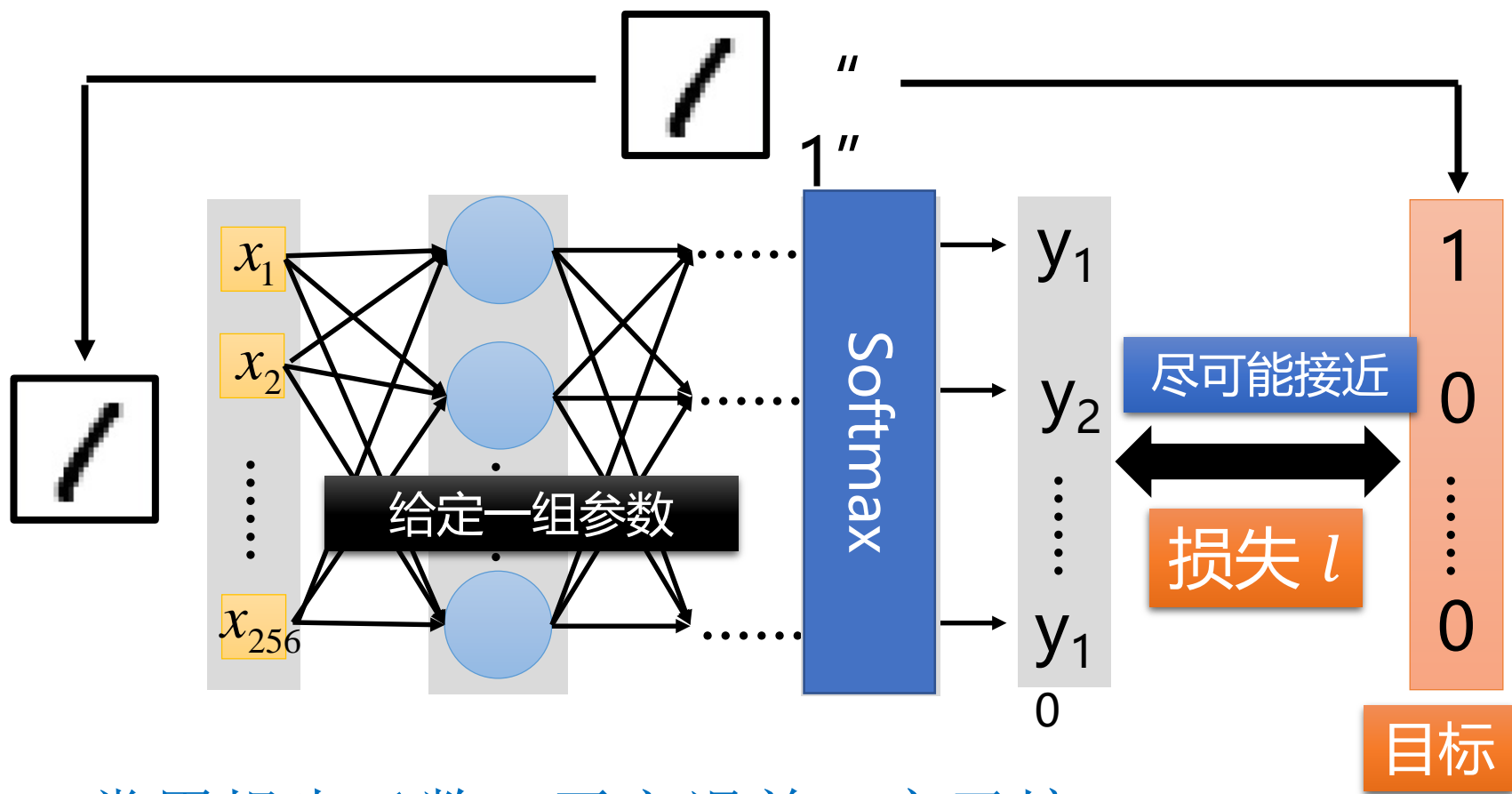
损失函数设计使得:

input:  \rightarrow y_1 值最大

input:  \rightarrow y_2 值最大

Loss Function

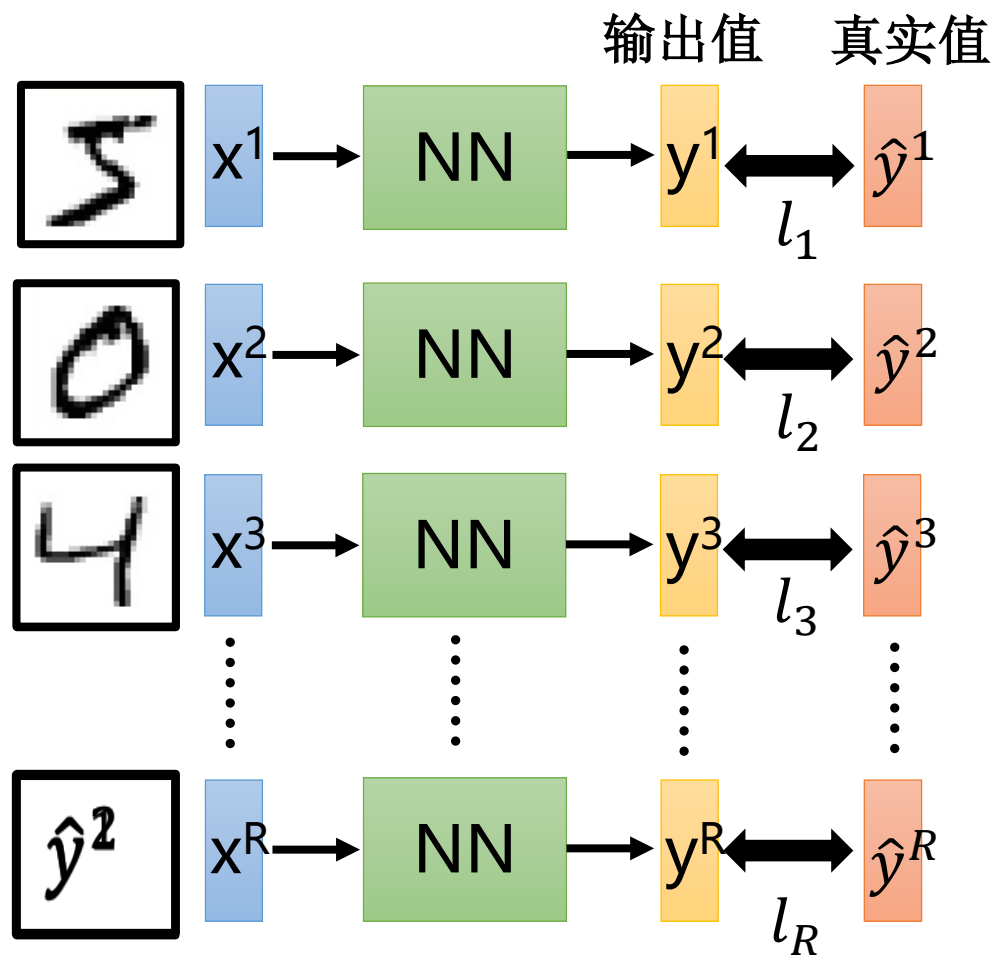
好的参数使得所有训练数据的损失越小越好



常用损失函数：平方误差，交叉熵

深度学习

输入所有训练数据R (样本个数) :



总损失:

$$L = \sum_{r=1}^R l_r$$

尽可能小

训练模型, 使得总损失L最小

即确定参数使得总损失L最小

深度学习步骤

◆ Build Model （建立模型）

- 确定网络结构
- 确定网络层数、及每层的神经元数

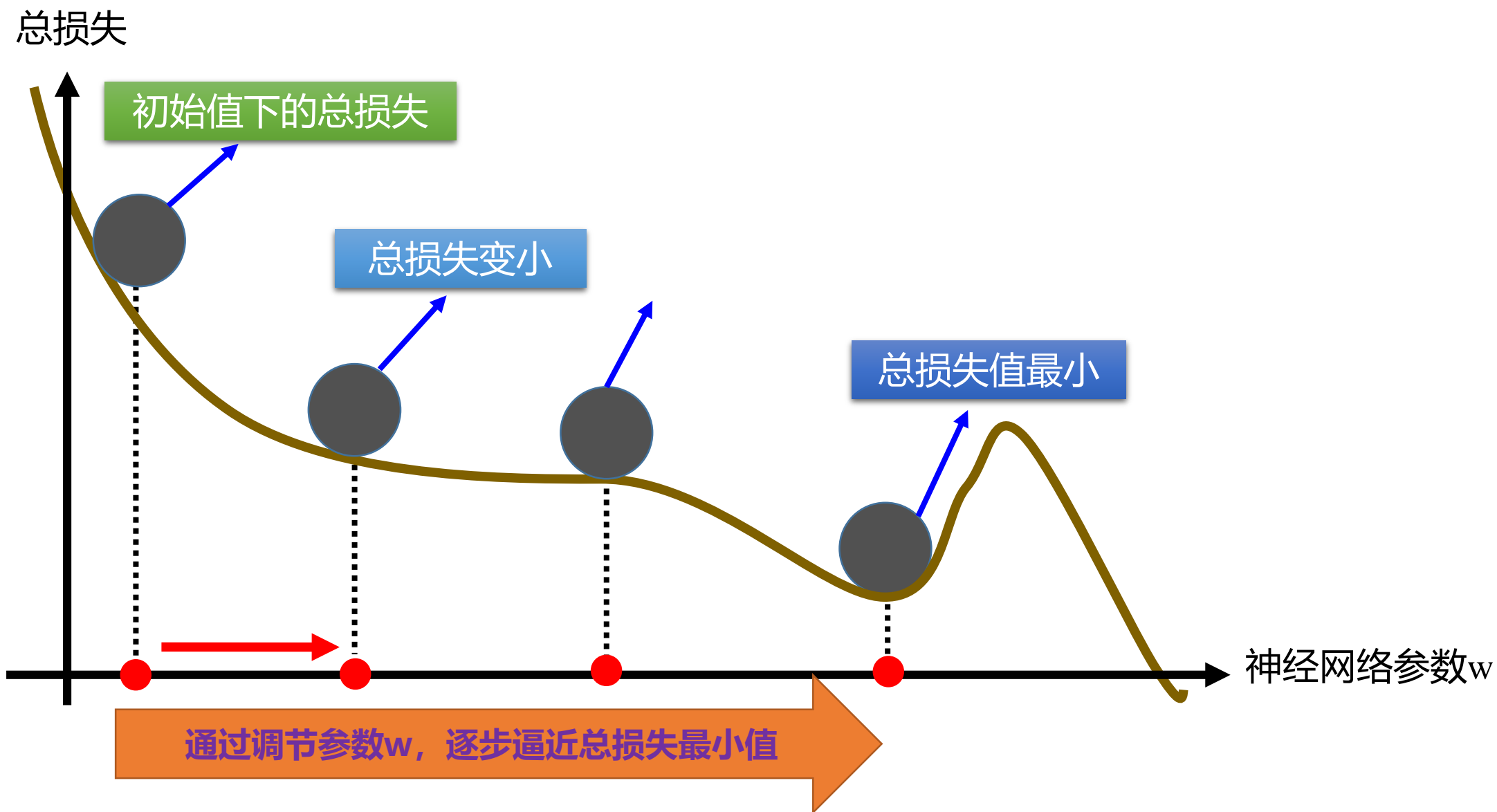
◆ Loss Function （损失函数）

常用损失函数：平方误差，交叉熵

◆ Parameter Learning Algorithm （参数学习方法）

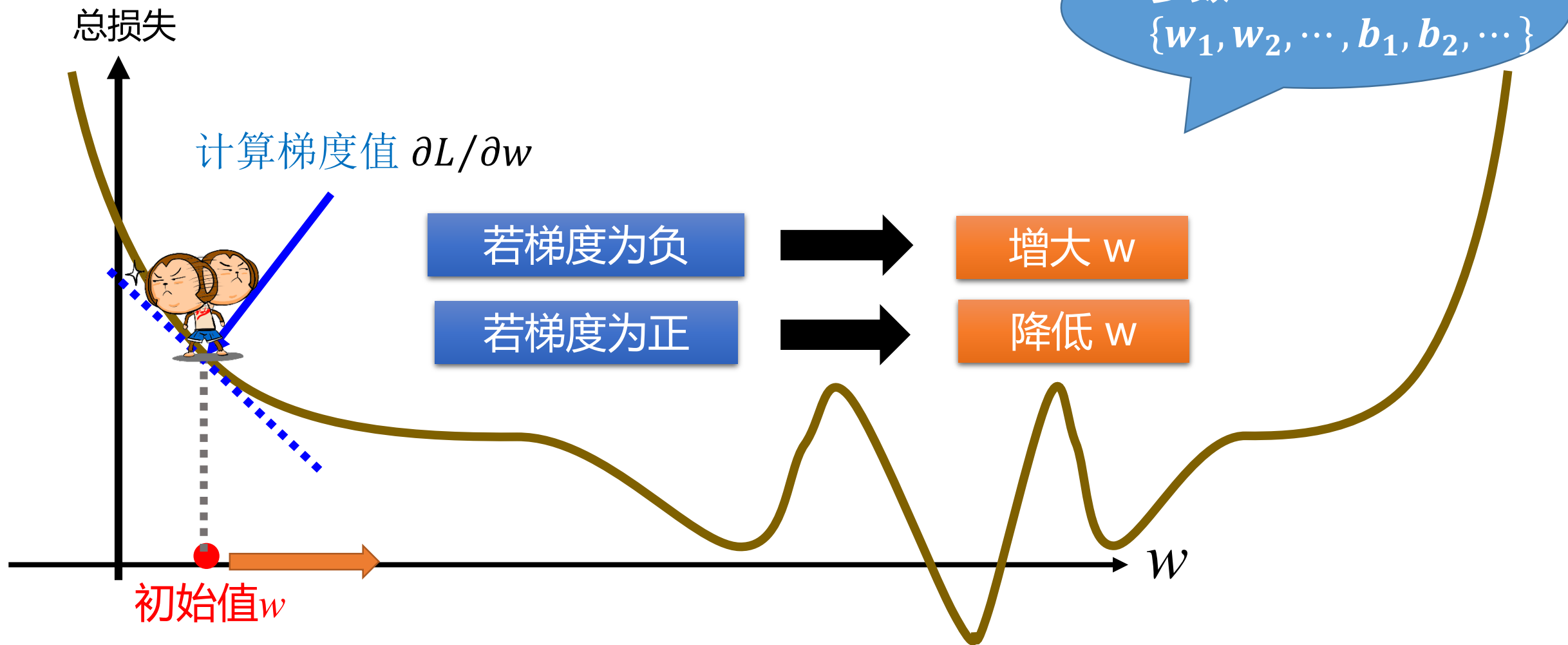
- 梯度下降
- 误差反向传播算法

参数学习—梯度下降法

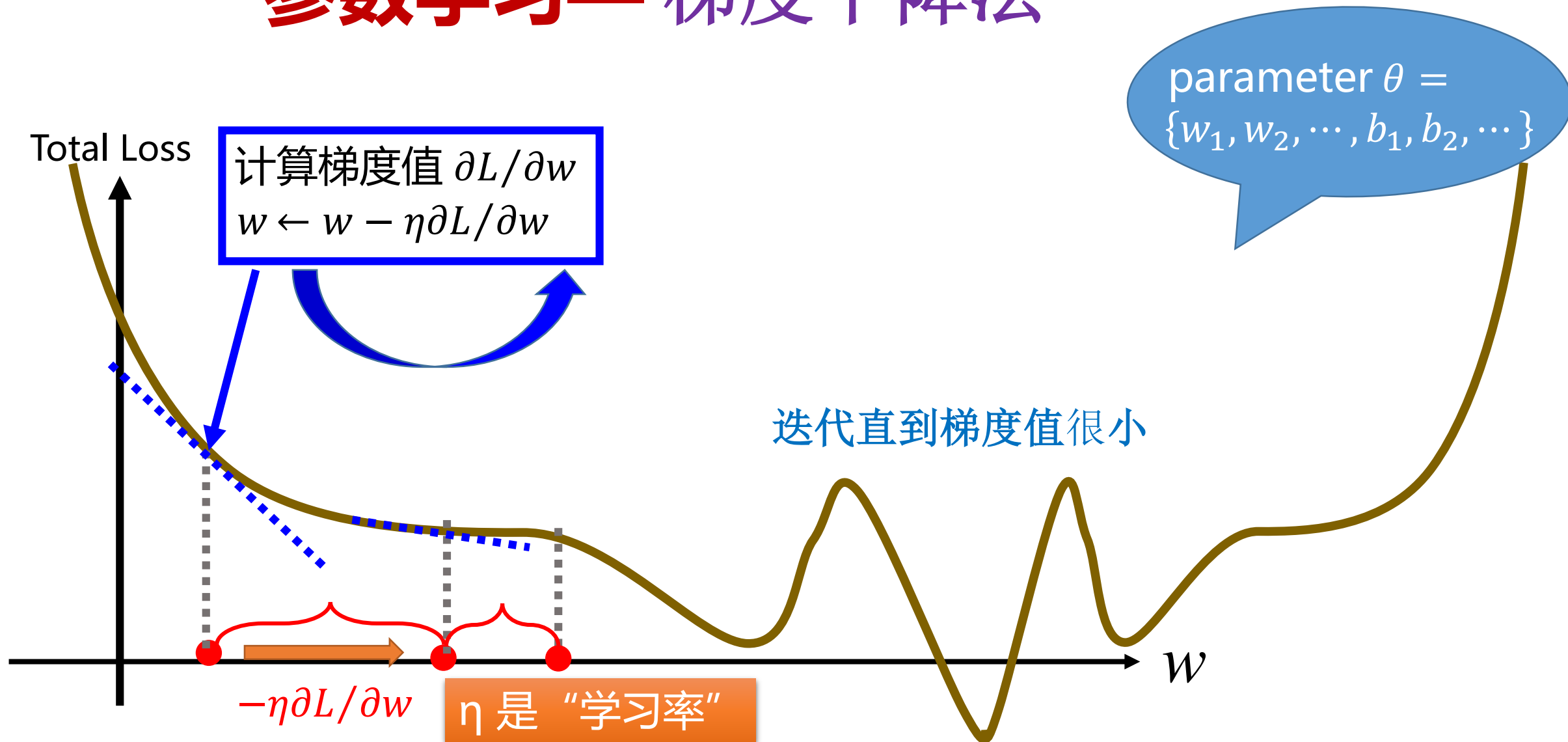


参数学习— 梯度下降法

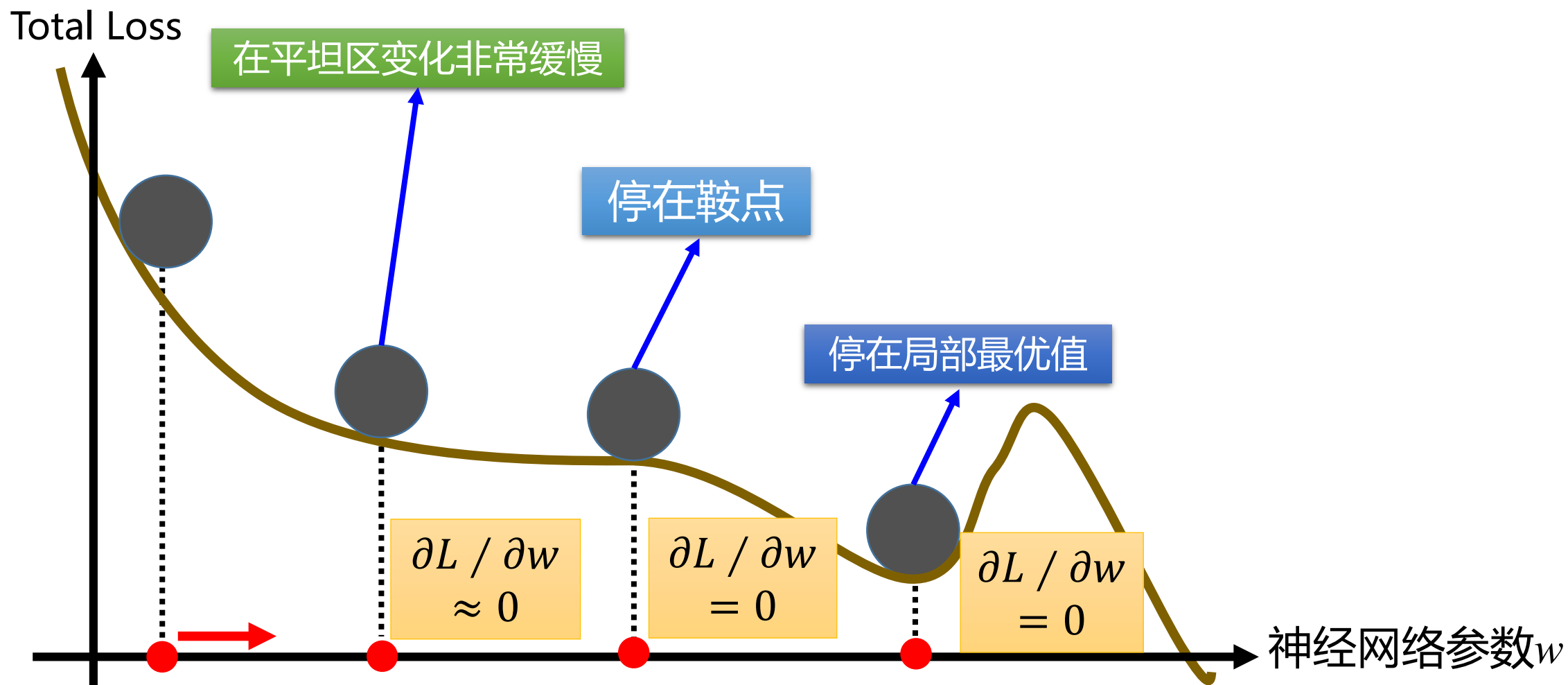
- ◆ 随机选择一个初始的参数值 w ，预训练



参数学习—梯度下降法

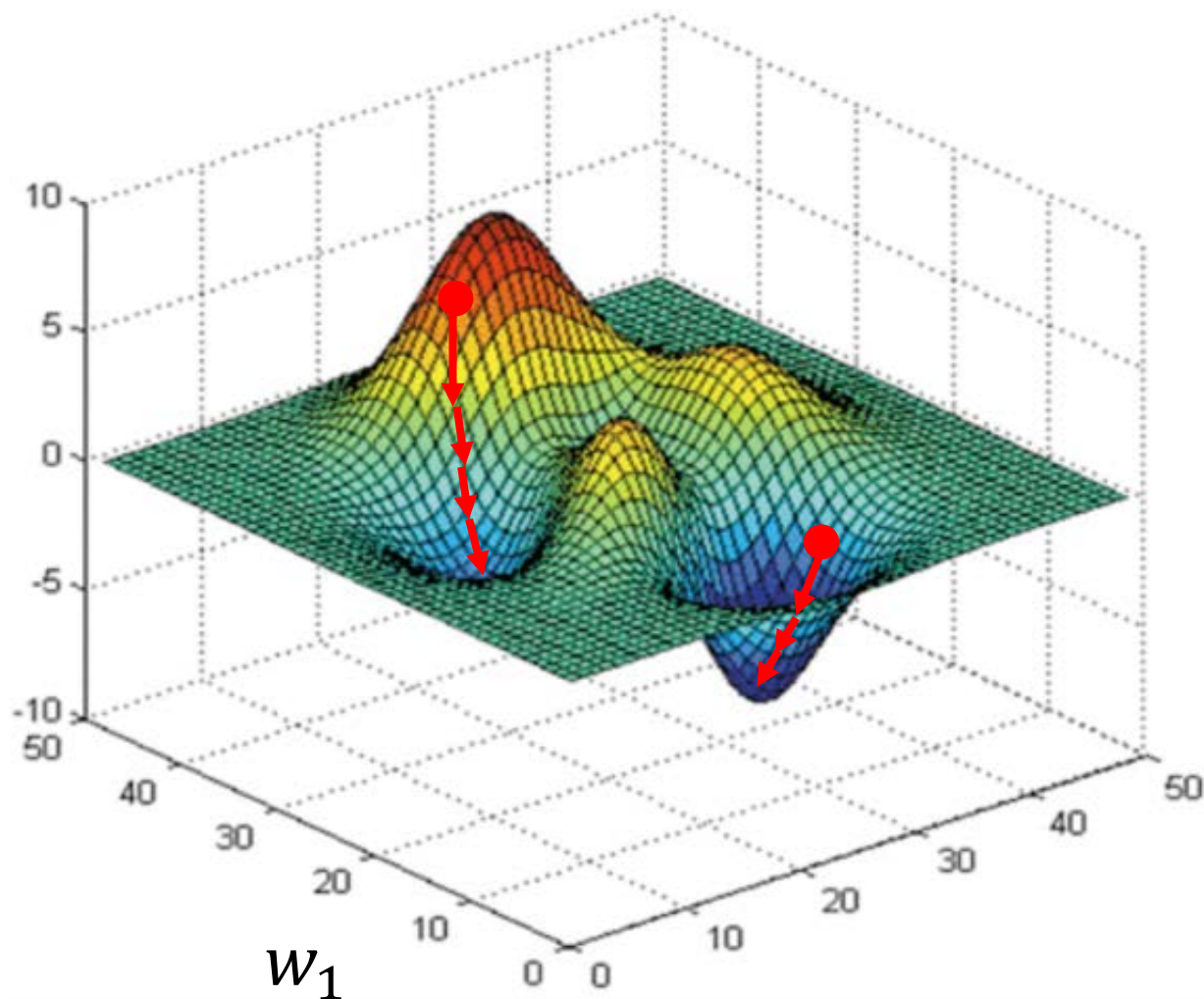


参数学习—梯度下降法

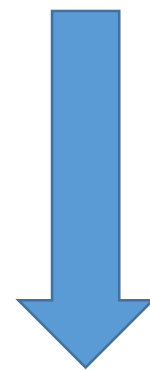


参数学习—梯度下降法

初始值对极值的影响



设置不同的初始值



可能到达不同的局部最小值

深度学习步骤

◆ Build Model （建立模型）

- 确定网络结构
- 确定网络层数、及每层的神经元数

◆ Loss Function （损失函数）

常用损失函数：平方误差，交叉熵

◆ Parameter Learning Algorithm （参数学习方法）

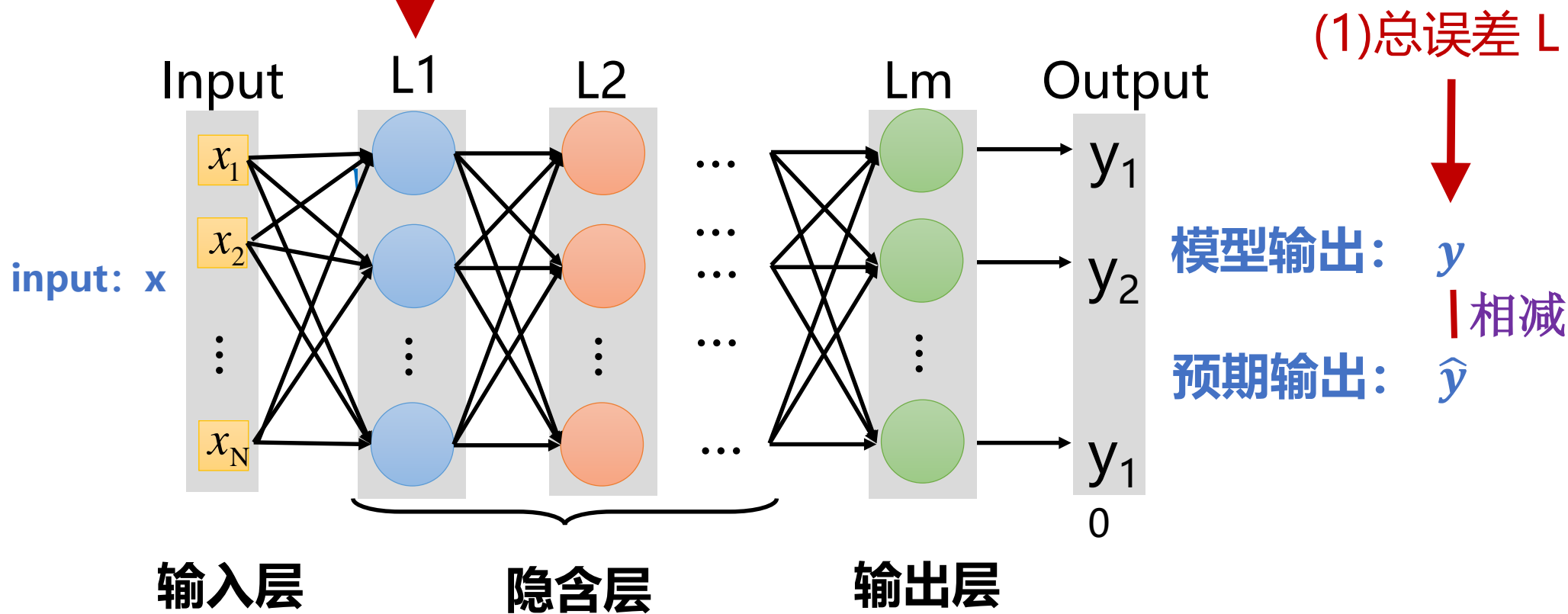
- 梯度下降
- 误差反向传播算法

参数学习—误差反向传播



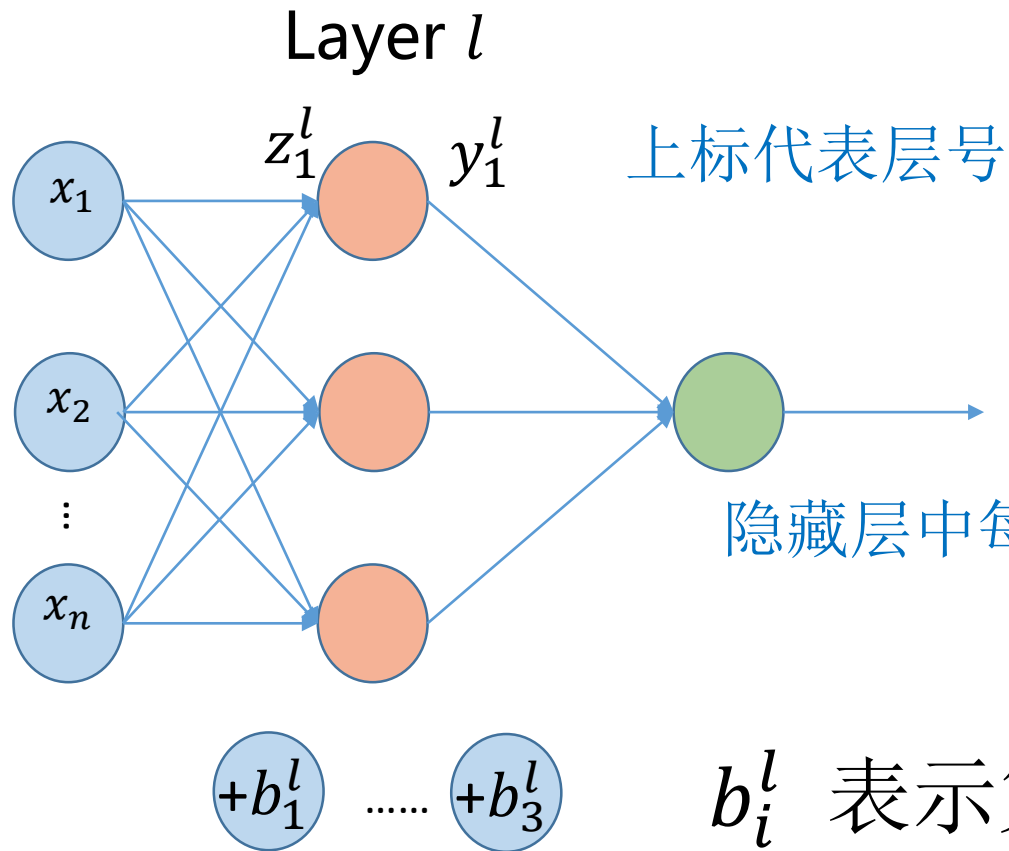
如何计算 $\partial L / \partial w$?

(2) 计算 $\partial L / \partial w_1$
(3) $w_1 - \eta \partial L / \partial w_1$
更新 w_1
以此类推更新 w_2 w_3

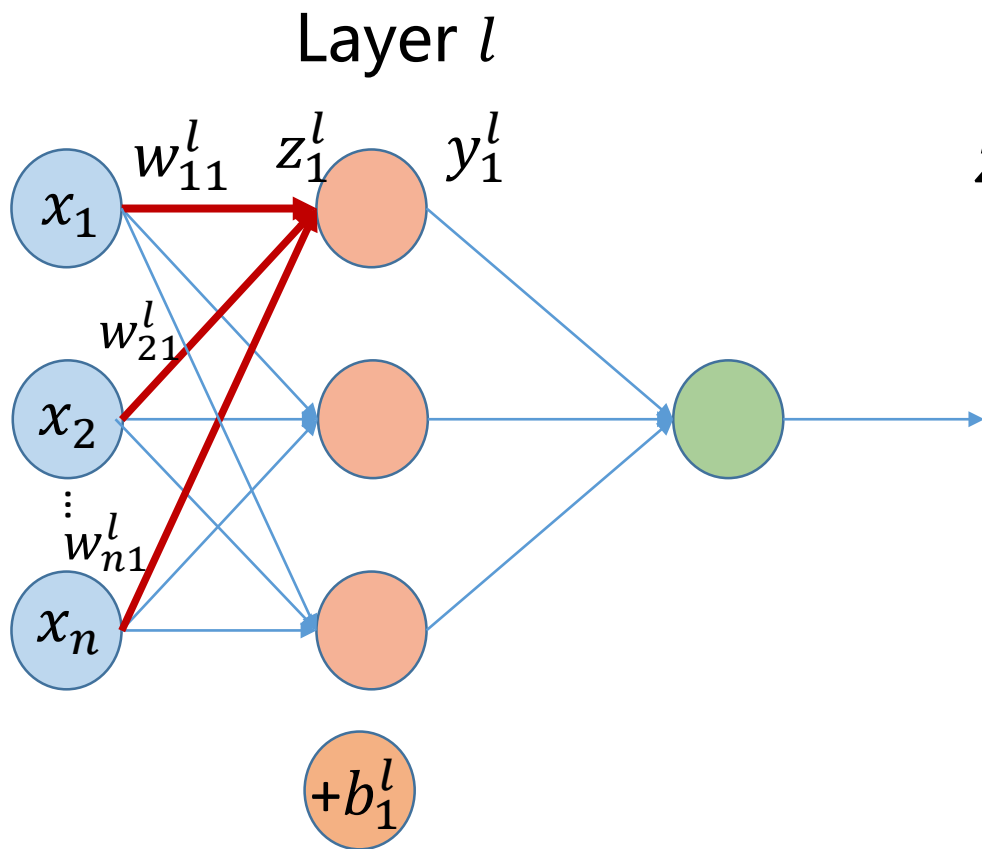


参数学习：信号正向传播

假定第 l 层神经元的输入为 z^l ，经激活函数后的输出值为 y^l



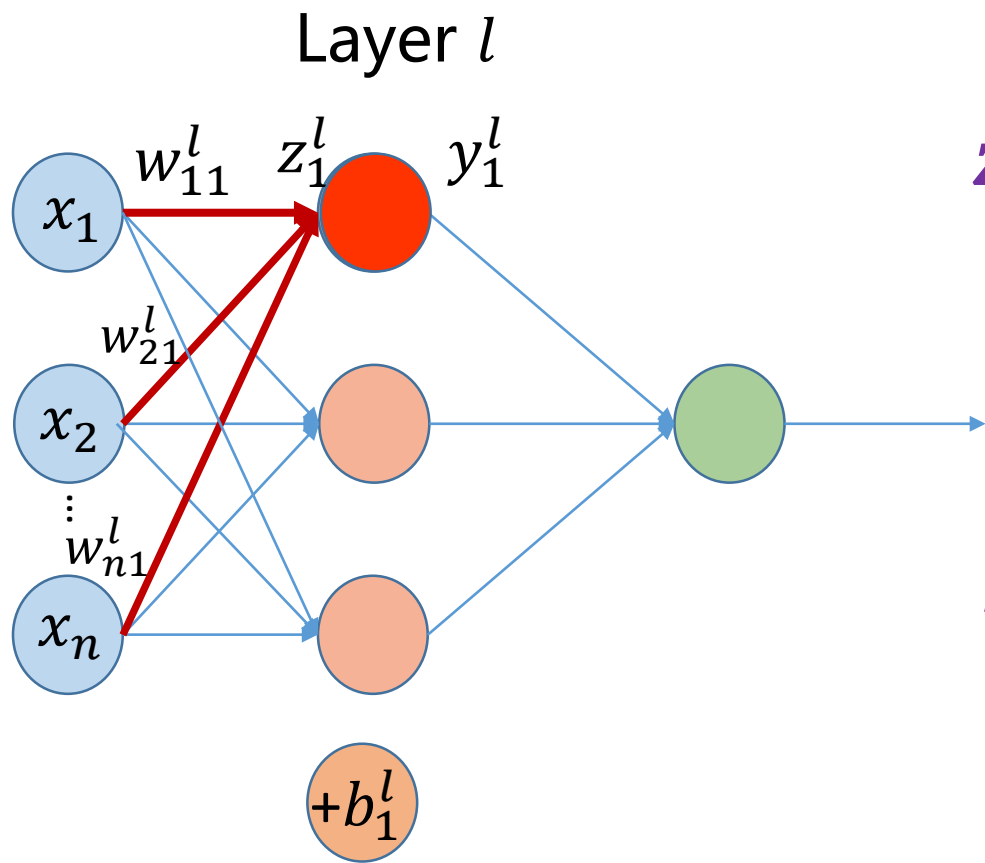
参数学习：信号正向传播



z_i^l 表示第 l 层的第 i 个结点的输入信号

- $$z_1^l = w_{11}^l x_1 + w_{21}^l x_2 + \dots + w_{n1}^l x_n + b_1^l$$

参数学习：信号正向传播



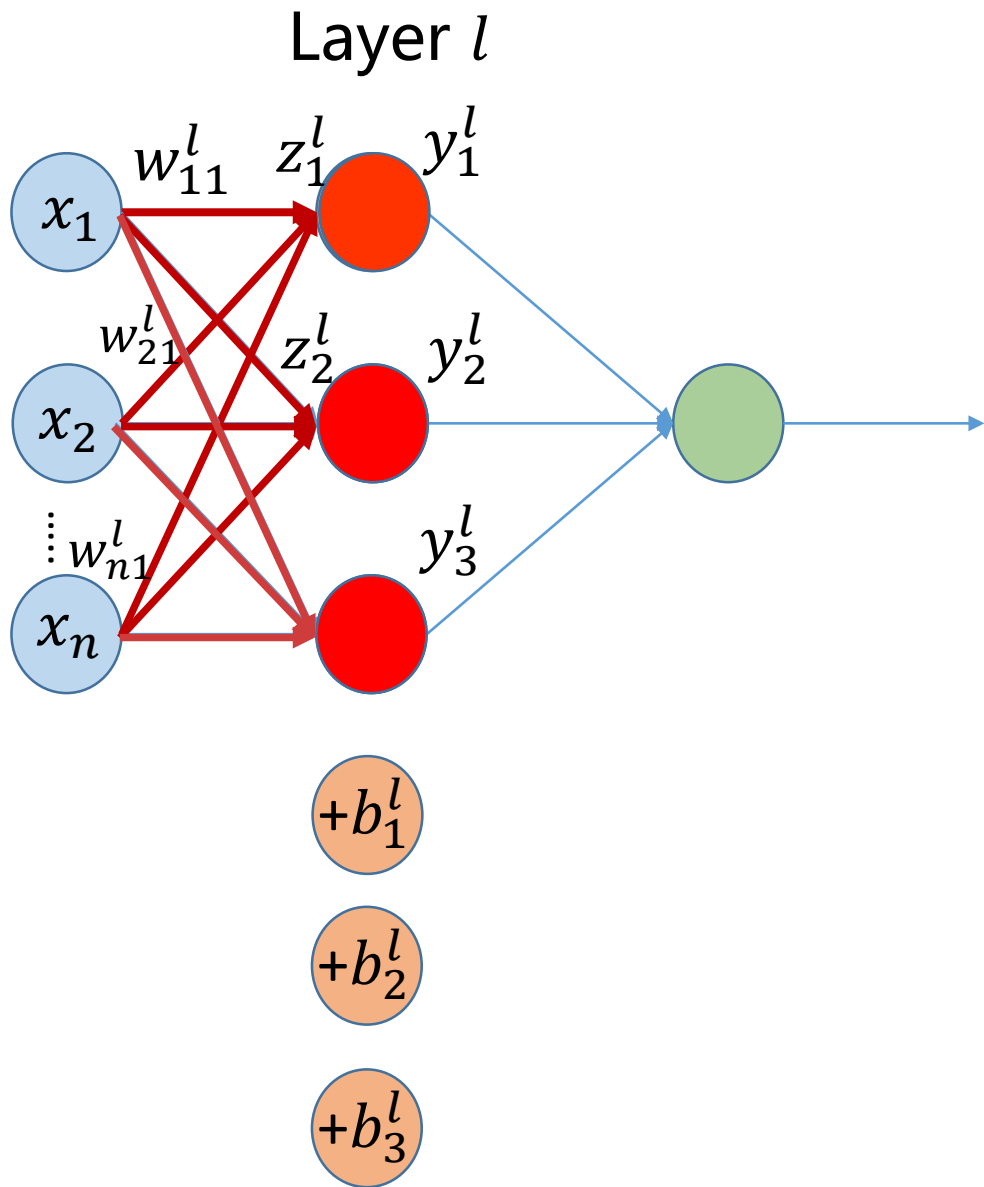
z_i^l 表示第 l 层的第 i 个结点的输入信号

- $z_1^l = w_{11}^l x_1 + w_{21}^l x_2 + \dots + w_{n1}^l x_n + b_1^l$

y_i^l 表示第 l 层的第 i 个结点的输出信号

- $y_1^l = f(z_1^l)$
 $= f(w_{11}^l x_1 + w_{21}^l x_2 + \dots + w_{n1}^l x_n + b_1^l)$

参数学习：信号正向传播



- $z_1^l = w_{11}^l x_1 + w_{21}^l x_2 + \dots + w_{n1}^l x_n + b_1^l$

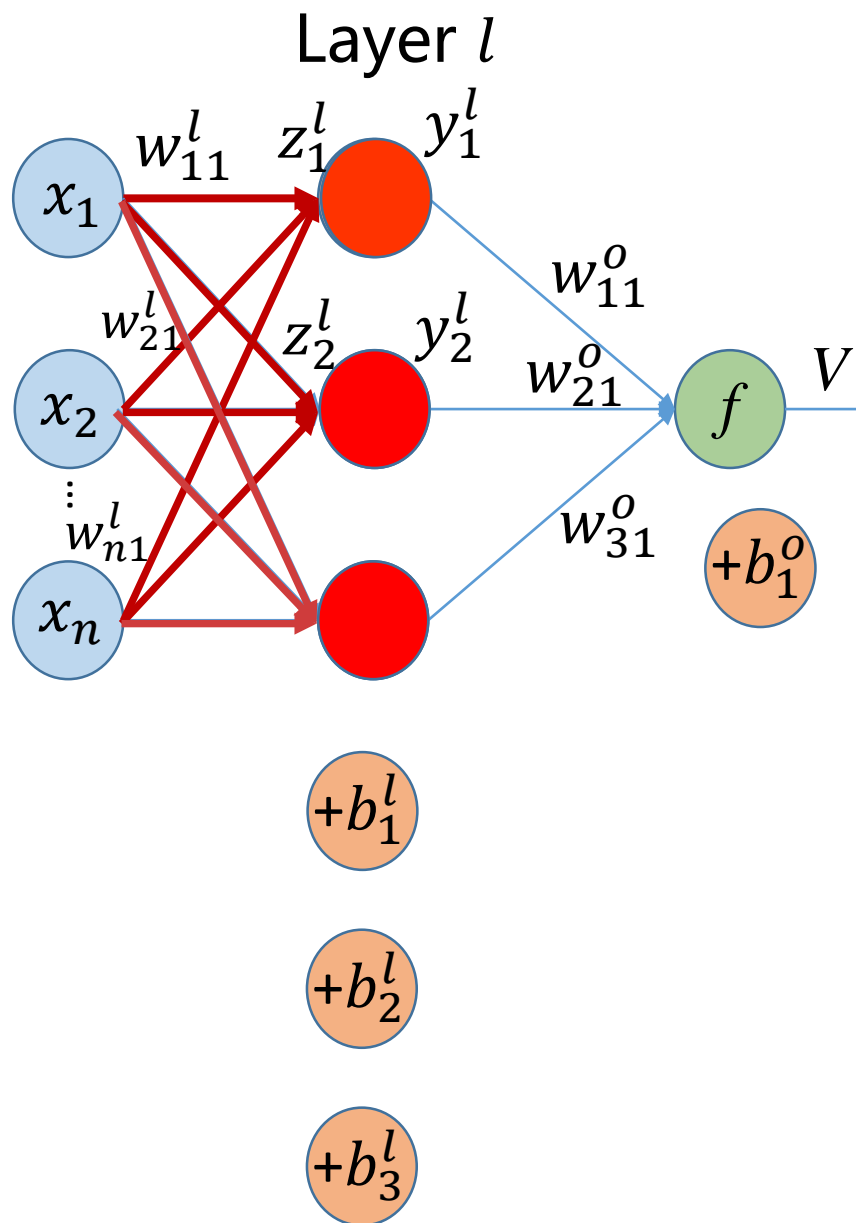
- $y_1^l = f(z_1^l)$
 $= f(w_{11}^l x_1 + w_{21}^l x_2 + \dots + w_{n1}^l x_n + b_1^l)$

相似地,

- $y_2^l = f(z_2^l)$
 $= f(w_{12}^l x_1 + w_{22}^l x_2 + \dots + w_{n2}^l x_n + b_2^l)$

- $y_3^l = f(z_3^l)$
 $= f(w_{13}^l x_1 + w_{23}^l x_2 + \dots + w_{n3}^l x_n + b_3^l)$

参数学习：信号正向传播



假设 l 是最后一个隐含层, 其输出为:

$$V = f(w_{11}^o y_1^l + w_{21}^o y_2^l + w_{31}^o y_3^l + b_1^o)$$

其中:

- $y_1^l = f(z_1^l)$
 $= f(w_{11}^l x_1 + w_{21}^l x_2 + \dots + w_{n1}^l x_n + b_1^l)$
- $y_2^l = f(z_2^l)$
 $= f(w_{12}^l x_1 + w_{22}^l x_2 + \dots + w_{n2}^l x_n + b_2^l)$
- $y_3^l = f(z_3^l)$
 $= f(w_{13}^l x_1 + w_{23}^l x_2 + \dots + w_{n3}^l x_n + b_3^l)$

参数学习：误差反向传播算法

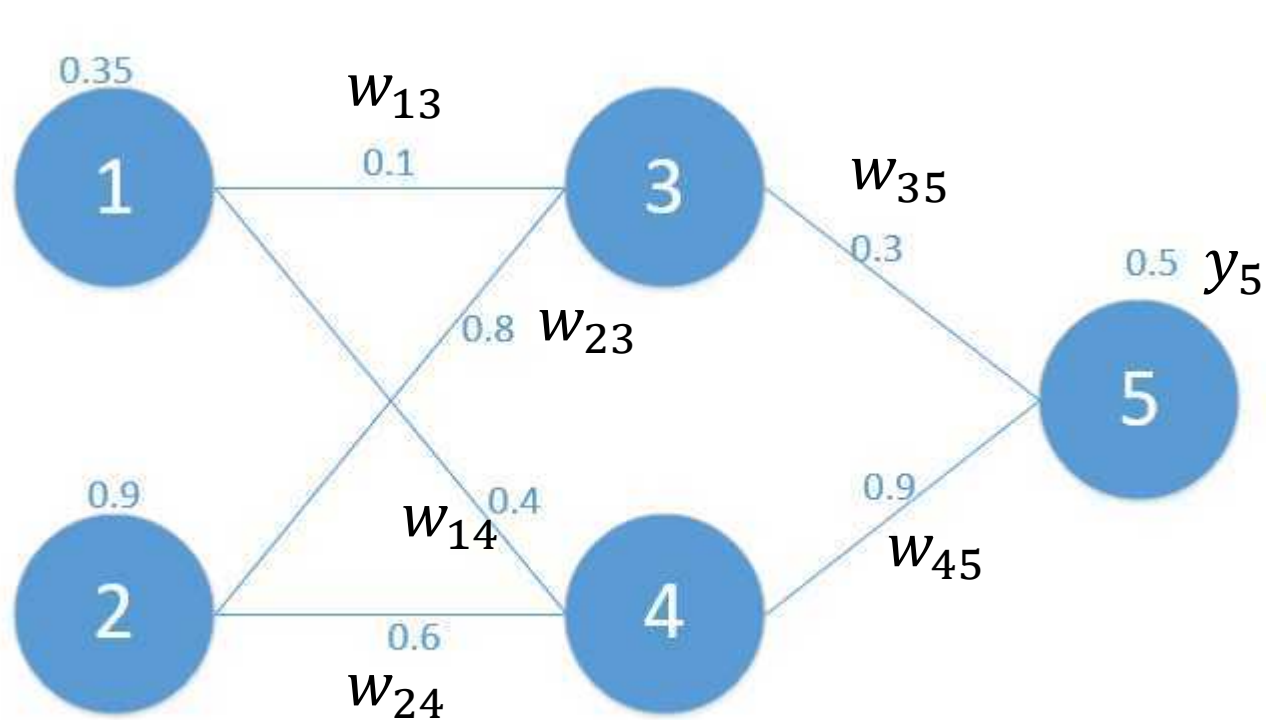
例如： $f = \text{sigmoid}$ 函数

$$x_1 = 0.35, x_2 = 0.9, \hat{y} = 0.5$$

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

w_{ij} 是第 i 个结点与第 j 个结点连接的权值

z_i 是第 i 个结点的输入， y_i 是第 i 个结点的输出



<http://blog.csdn.net/bitcarmanlee>

$$z_3 = 0.35 * 0.1 + 0.9 * 0.8 = 0.755$$

$$z_4 = 0.35 * 0.4 + 0.9 * 0.6 = 0.68$$

$$y_3 = f(z_3) = 0.68$$

$$y_4 = f(z_4) = 0.663$$

$$z_5 = 0.68 * 0.3 + 0.663 * 0.9 = 0.8007$$

$$y_5 = f(z_5) = f(0.8007) = 0.69$$

$$\text{Error} = y_5 - \hat{y} = 0.69 - 0.5 = 0.19$$

$$\text{Loss: } L = \frac{1}{2} (0.69 - 0.5)^2 = 0.01805$$

参数学习：误差反向传播算法

误差越小越好

$$L = \frac{1}{2}(y_5 - \hat{y})^2$$

$$y_5 = f(z_5)$$

$$z_5 = y_3 * w_{35} + y_4 * w_{45}$$

$$\partial L / \partial w = ?$$

$$f'(z) = f(z) * (1 - f(z))$$

$$\frac{\partial L}{\partial w_{35}} = \frac{\partial L}{\partial y_5} * \frac{\partial y_5}{\partial z_5} * \frac{\partial z_5}{\partial w_{35}}$$

$$= (y_5 - \hat{y}) * f(z_5) * (1 - f(z_5)) * y_3$$

$$= (0.69 - 0.5) * (0.69) * (1 - 0.69) * 0.68$$

$$= 0.02763$$

$$\frac{\partial L}{\partial w_{45}} = \frac{\partial L}{\partial y_5} * \frac{\partial y_5}{\partial z_5} * \frac{\partial z_5}{\partial w_{45}}$$

$$= (y_5 - \hat{y}) * f(z_5) * (1 - f(z_5)) * y_4$$

$$= (0.69 - 0.5) * (0.69) * (1 - 0.69) * 0.663$$

$$= 0.02711$$

$$w_{35} = w_{35} - \frac{\partial L}{\partial w_{35}} = 0.3 - 0.02763 = 0.27237$$

$$w_{45} = w_{45} - \frac{\partial L}{\partial w_{45}} = 0.9 - 0.02711 = 0.87289$$

参数学习：误差反向传播算法

仍需计算 w_{13}, w_{23}, w_{14} and w_{24}

$$f'(z) = f(z) * (1 - f(z))$$

$$L = \frac{1}{2}(y_5 - \hat{y})^2$$

$$y_5 = f(z_5)$$

$$z_5 = y_3 * w_{35} + y_4 * w_{45}$$

$$y_3 = f(z_3)$$

$$z_3 = x_1 * w_{13} + x_2 * w_{23}$$

$$w_{13} = w_{13} - \frac{\partial L}{\partial w_{13}} = 0.1 - 0.00093 = 0.09907$$

$$\begin{aligned}\frac{\partial L}{\partial w_{13}} &= \frac{\partial L}{\partial y_5} * \frac{\partial y_5}{\partial z_5} * \frac{\partial z_5}{\partial y_3} * \frac{\partial y_3}{\partial z_3} * \frac{\partial z_3}{\partial w_{13}} \\ &= (y_5 - \hat{y}) * f(z_5) * (1 - f(z_5)) * w_{35} * f(z_3) * (1 - f(z_3)) * x_1 \\ &= (0.69 - 0.5) * (0.69) * (1 - 0.69) * 0.3 * 0.68 * (1 - 0.68) * 0.35 \\ &= 0.00093\end{aligned}$$

参数学习：误差反向传播算法

仍需计算 w_{13}, w_{23}, w_{14} and w_{24}

$$w_{13} = w_{13} - \frac{\partial L}{\partial w_{13}} = 0.09907$$

$$w_{23} = w_{23} - \frac{\partial L}{\partial w_{23}} = 0.78986$$

$$w_{14} = w_{14} - \frac{\partial L}{\partial w_{14}} = 0.39662$$

$$w_{24} = w_{24} - \frac{\partial L}{\partial w_{24}} = 0.58986$$

$$w_{35} = w_{35} - \frac{\partial L}{\partial w_{35}} = 0.27237$$

$$w_{45} = w_{45} - \frac{\partial L}{\partial w_{45}} = 0.87289$$

再重复上述过程 99 次, 得到如下结果:

$$w_{13} = 0.09933$$

$$w_{23} = 0.64253$$

$$w_{14} = 0.39933$$

$$w_{24} = 0.44253$$

$$w_{35} = -0.30032$$

$$w_{45} = 0.31509$$

$$Error = y_5 - \hat{y} = 5.92944818e-07$$

$$\text{重复1次: } Error = y_5 - \hat{y} = 0.165$$

评价指标

预测类别

实际分类	预测类别	
	True	False
	True	False
实际分类	True	False
	False	True

P—Positive, N----Negative

$P=TP+FN$, 为所有正例的个数

$N=FP+TN$, 为所有负例的个数

TP—True positive (正例, 被分为正例)

FN—False Negative (正例, 被分为负例)

FP—False positive (负例, 被分为正例)

TN—True Negative (负例, 被分为负例)

评价指标

◆正确率 (accuracy)

$\text{accuracy} = (\text{TP} + \text{TN}) / (\text{P} + \text{N})$ ，即被分对的样本数除以所有的样本数，正确率越高，分类器越好；

◆错误率 (error rate)

$\text{error rate} = (\text{FP} + \text{FN}) / (\text{P} + \text{N})$ ，对某一个实例来说，分对与分错是互斥事件，所以 $\text{accuracy} = 1 - \text{error rate}$ ；

◆精度 (precision)

表示被分为正例的样例且实际为正例的比例， $\text{precision} = \text{TP} / (\text{TP} + \text{FP})$ ；

◆召回率 (recall)

召回率是覆盖面的度量，度量有多个正例被分为正例， $\text{recall} = \text{TP} / (\text{TP} + \text{FN}) = \text{TP} / \text{P}$ 。

Fei-Fei Li (李飞飞)

- ◆ 1976年出生,16岁随父母移居美国新泽西州。
- ◆ 1999年于普林斯顿大学取得物理学学士。
- ◆ 2005年获得加州理工学院电子工程博士学位。
- ◆ 2017年, 入职谷歌,同时仍是斯坦福大学教授。
- ◆ 她最著名的作品之一是[ImageNet](#)项目, 该项目彻底改变了大规模视觉识别领域



李飞飞

ImageNet

- ◆ ImageNet是一个大型视觉数据库，用于视觉对象识别的研究.
- ◆ 从**2007年**起，李飞飞与团队便从互联网上下载了近10亿幅图片。
- ◆ 手工分类、注释了超过1400万图像，以指示图像中的物体，并且在至少一百万个图像中，还提供了边界框。
- ◆ 到2009年，ImageNet中包括1500万张照片，20000多种物品，如“气球”或“草莓”。

ImageNet

- ◆ ImageNet作为论文发布在CVPR-2009。当时人们还很怀疑通过更多数据就能改进算法的看法。
- ◆ 自 **2010年** 以来，ImageNet项目开展了大规模视觉识别挑战赛（[ILSVRC](#)），其中软件程序竞争以正确分类和检测对象和场景。
- ◆ ImageNet挑战赛的机器错误率，七年来从28%降到了3.6%（人眼识别错误率5.1%）
- ◆ 深度学习发展起来有几个关键的因素，一个就是庞大的数据（比如说ImageNet），一个是GPU的出现。

ImageNet Database

<http://www.image-net.org/about-overview>



n15075141_2837.
JPEG



n15075141_2853.
JPEG



n15075141_2856.
JPEG



n15075141_2879.
JPEG



n15075141_2898.
JPEG



n15075141_2917.
JPEG



n15075141_2919.
JPEG



n15075141_2968.
JPEG



n15075141_2973.
JPEG



n15075141_2998.
JPEG



n15075141_3009.
JPEG



n15075141_3011.
JPEG



n15075141_3031.
JPEG



n15075141_3042.
JPEG



n15075141_3082.
JPEG



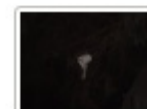
n15075141_3138.
JPEG



n15075141_3141.
JPEG



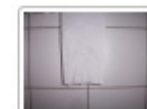
n15075141_3165.
JPEG



n15075141_3171.
JPEG



n15075141_3184.
JPEG



n15075141_3191.
JPEG



n15075141_3263.
JPEG



n15075141_3267.
JPEG



n15075141_3280.
JPEG



n15075141_3296.
JPEG



n15075141_3338.
JPEG



n15075141_3339.
JPEG



n15075141_3343.
JPEG



n15075141_3432.
JPEG



n15075141_3443.
JPEG



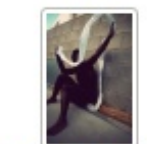
n15075141_3460.
JPEG



n15075141_3490.
JPEG



n15075141_3498.
JPEG



n15075141_3516.
JPEG



n15075141_3524.
JPEG

Winners in ILSVRC

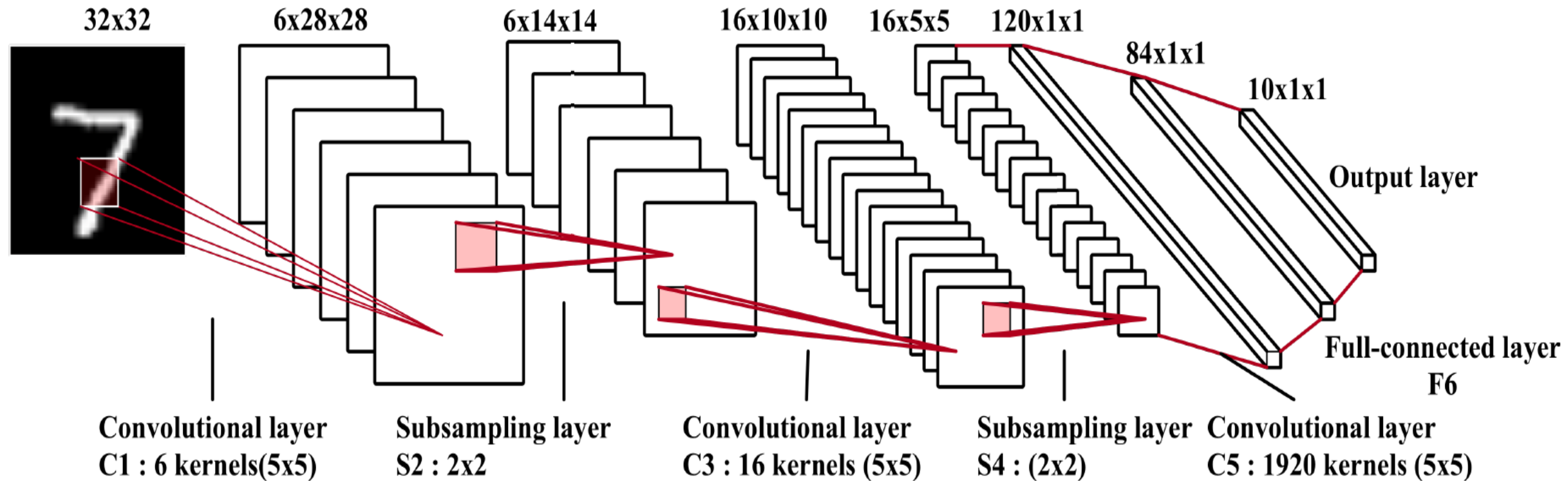
- ◆2012, **AlexNet-8** (Alex Krizhevsky、Ilya Sutskever 和 Geoffrey Hinton)
- ◆2013, **ZFNet-8**
- ◆2014, **GoogLeNet-22** (Inception Modules) and **VGG-19 (runner-up)**
- ◆2015, **ResNet-152**
- ◆2016年, WideResNet, FractalNet, DenseNet, ResNeXt, DPN, SENet。
- ◆ILSVRC end in 2017.

模型	AlexNet	ZF Net	GoogLeNet	ResNet
Year	2012	2013	2014	2015
Number of layers	8	8	22	152
Top 5 Error rate	15.4%	11.2%	6.7%	3.57%
Data enhancement	√	√	√	√
Dropout	√	√		
Normalization				√

LeNet-5的网络结构

其学习目标是从一系列由 $32 \times 32 \times 1$ 灰度图像表示的手写数字中，识别和区分0-9。LeNet-5的隐含层由**2个卷积层、2个池化层、2个全连接层、1个输出层**组成，按如下方式构建：

- ◆ $(3 \times 3) \times 1 \times 6$ 的卷积层（步长为1，无填充）， 2×2 均值池化（步长为2，无填充），tanh激励函数
- ◆ $(5 \times 5) \times 6 \times 16$ 的卷积层（步长为1，无填充）， 2×2 均值池化（步长为2，无填充），tanh激励函数
- ◆ 2个全连接层，神经元数量为120和84
- ◆ LeNet-5使用[双曲正切函数](#)作为激活函数，使用[均方差](#)（Mean Squared Error, MSE）作为[误差函数](#)，并对卷积操作进行了修改以减少计算开销，这些设置在随后的CNN中已被更优化的方法取代，如用[交叉熵](#)（categorical cross-entropy）作为损失函数，**用ReLU做激励函数**。

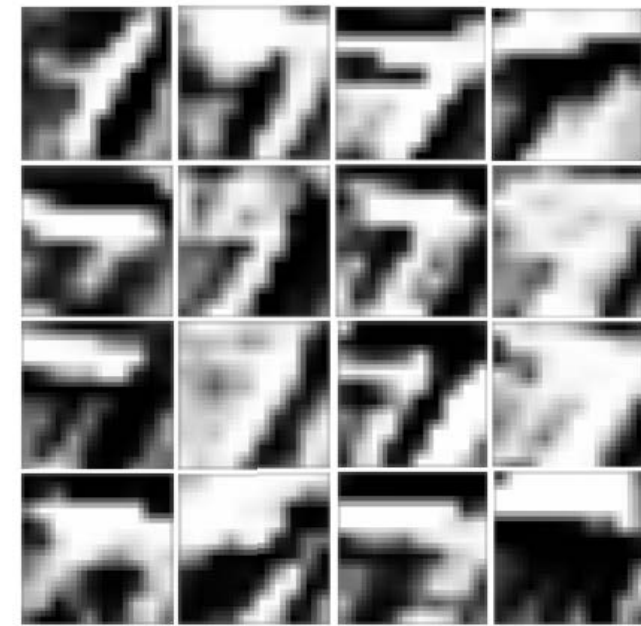


LeNet-5

- ◆ LenNet-5的输入层图像大小是 32×32 （Caffe中Mnist数据库为 28×28 ）
- ◆ LenNet-5共有7层（不包括输入层），分别是C1, S2, C3, S4, F5, F6, output, C是Convolutions, S是SubSampling, F是Full connection; 每层都包含不同数量的训练参数。

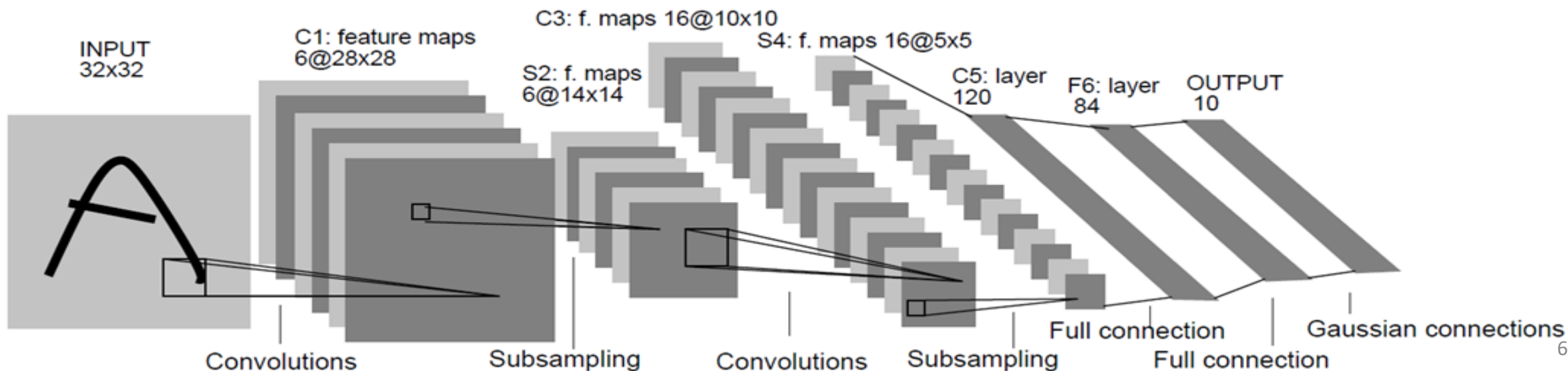


Convolutional layer C1
 $6 \times 28 \times 28$



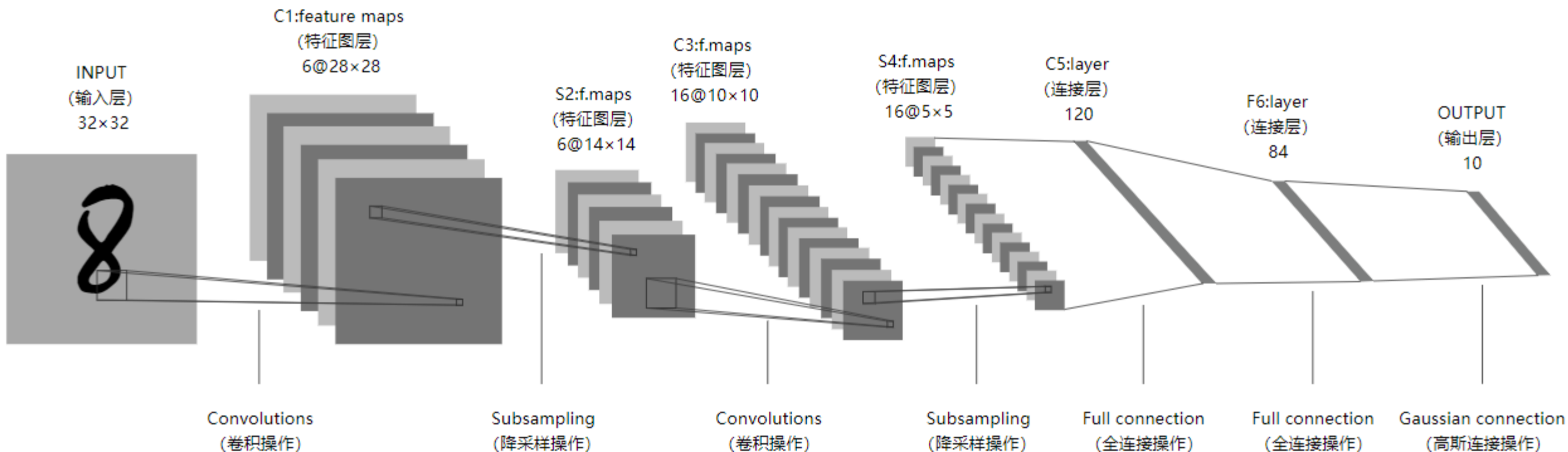
Convolutional layer C3
 $16 \times 10 \times 10$

Learned Features



LeNet-5

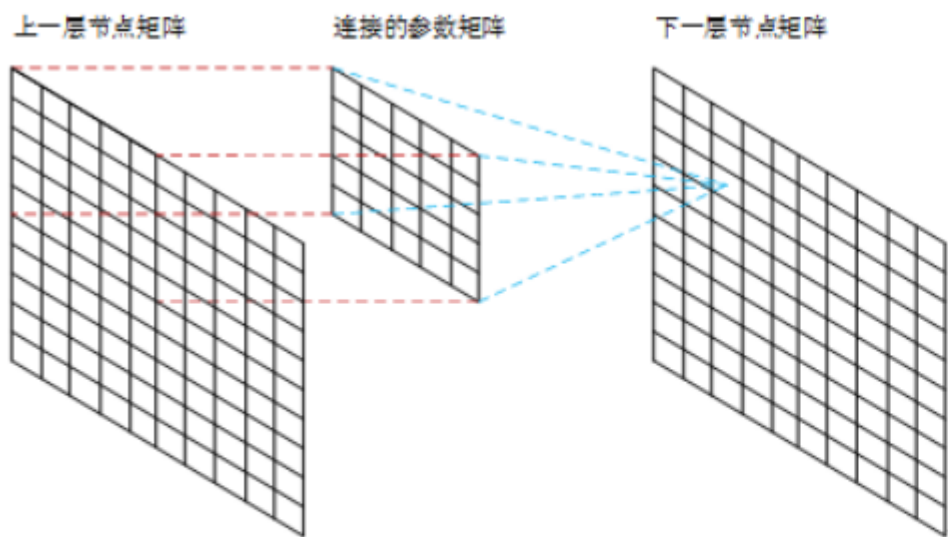
LenNet-5 共有7层（不包括输入层），分别是C1, S2, C3, S4, C5（或记为 F5）,F6,output



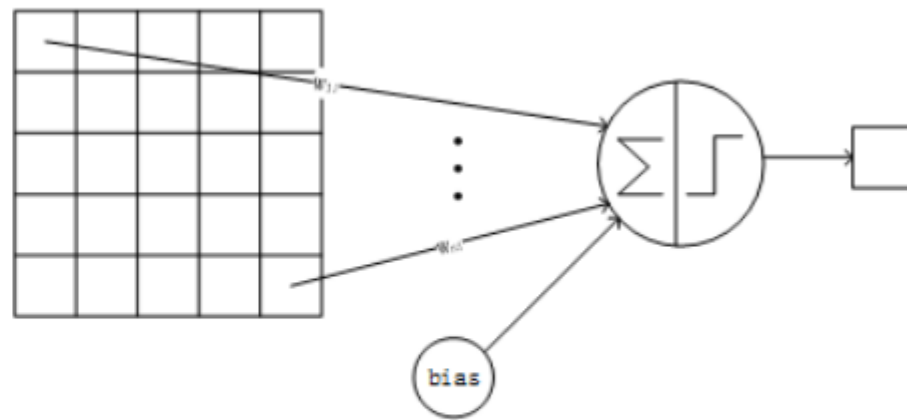
LeNet-5

◆卷积层

卷积层采用的都是5x5大小的卷积核/过滤器（kernel/filter），且卷积核每次滑动一个像素（stride=1），一个特征图谱使用同一个卷积核. 每个上层节点的值乘以连接上的参数，把这些乘积及一个偏置参数相加得到一个和，把该和输入激活函数，激活函数的输出即是下一层节点的值



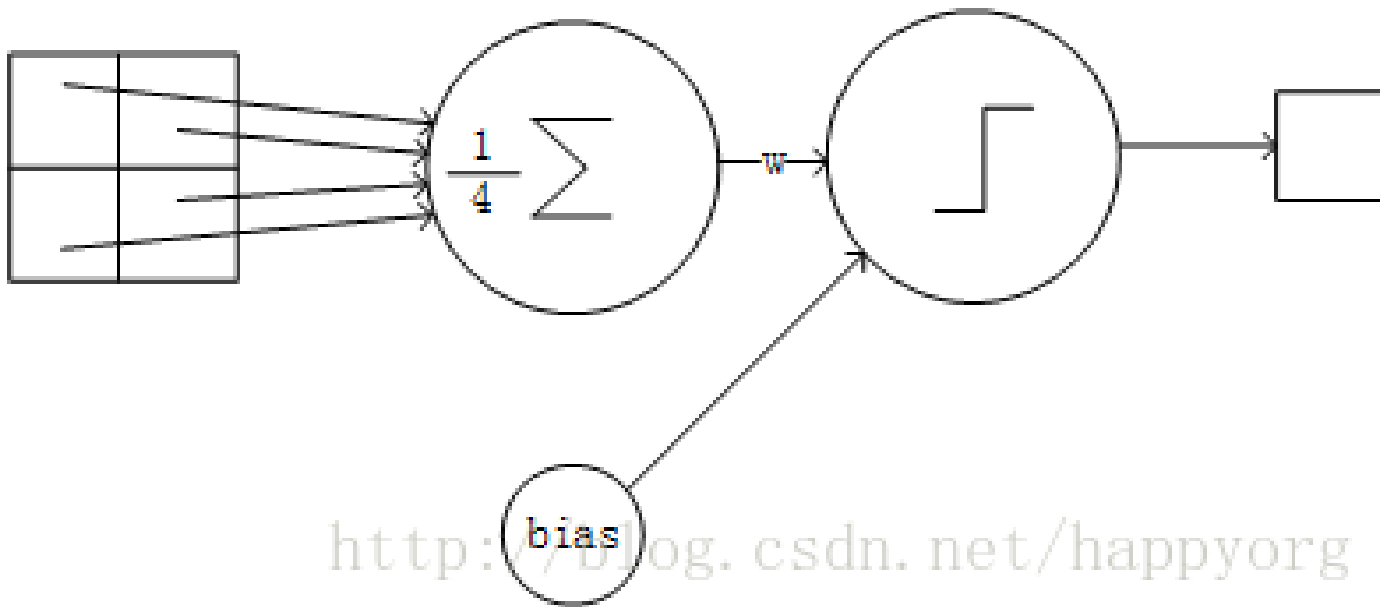
卷积神经网络连接与矩阵卷积的对应关系



LeNet-5

◆ 池化层（pooling）

池化层采用的是**2x2的平均池化**，即上一层的4个节点作为下一层1个节点的输入，且输入域不重叠，即每次滑动2个像素。下采样节点的结构如下：



每个下采样节点的4个输入节点求和后取平均（**平均池化**），均值乘以一个参数，加上一个**偏置参数**，作为激活函数的输入，激活函数的输出即是下一层节点的值。

◆ LeNet-5第一层：卷积层C1

C1层是卷积层，有**6个5x5卷积核**，形成6个特征图谱。

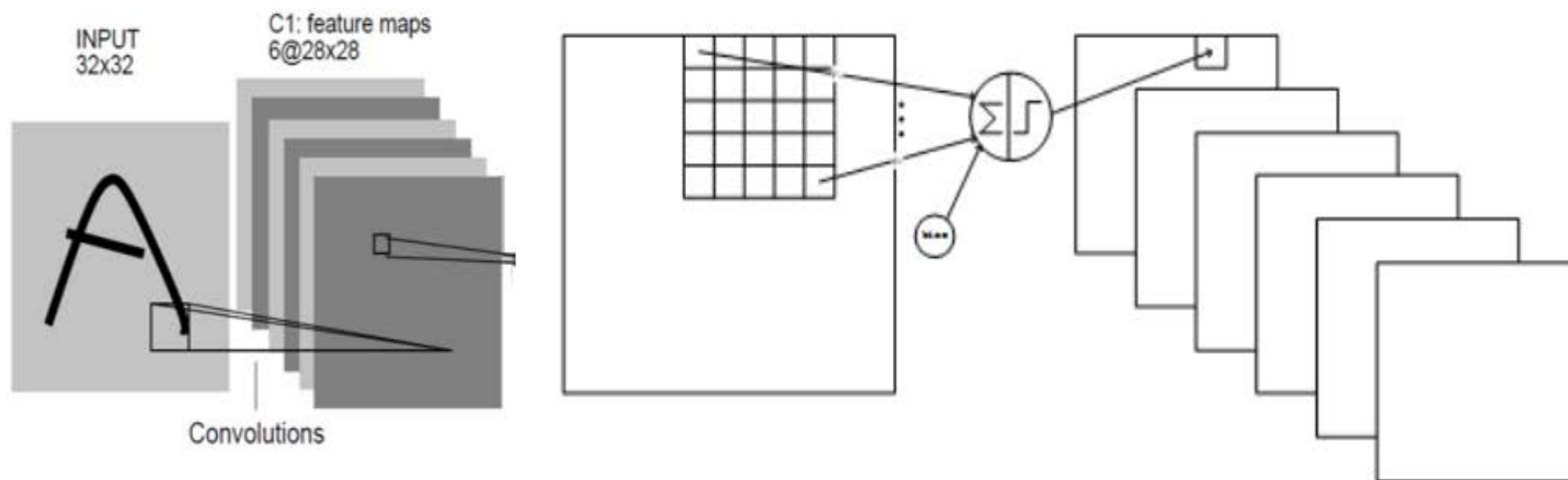
每个特征图谱内参数共享，即每个特征图谱内只使用一个共同卷积核。

卷积核有5x5个连接参数，加上1个偏置，共26个参数。

卷积区域每次滑动一个像素，这样卷积层形成的每个特征图谱大小是 $(32-5)/1+1=28 \times 28$ 。

C1层共有 $26 \times 6 = 156$ 个训练参数，有 $(5 \times 5 + 1) \times 28 \times 28 \times 6 = 122304$ 个连接。

C1层的连接结构如下所示。



$$\text{outLength} = (\text{inLength} - \text{filterLength} + 2 * \text{padding}) / \text{strideLength} + 1$$

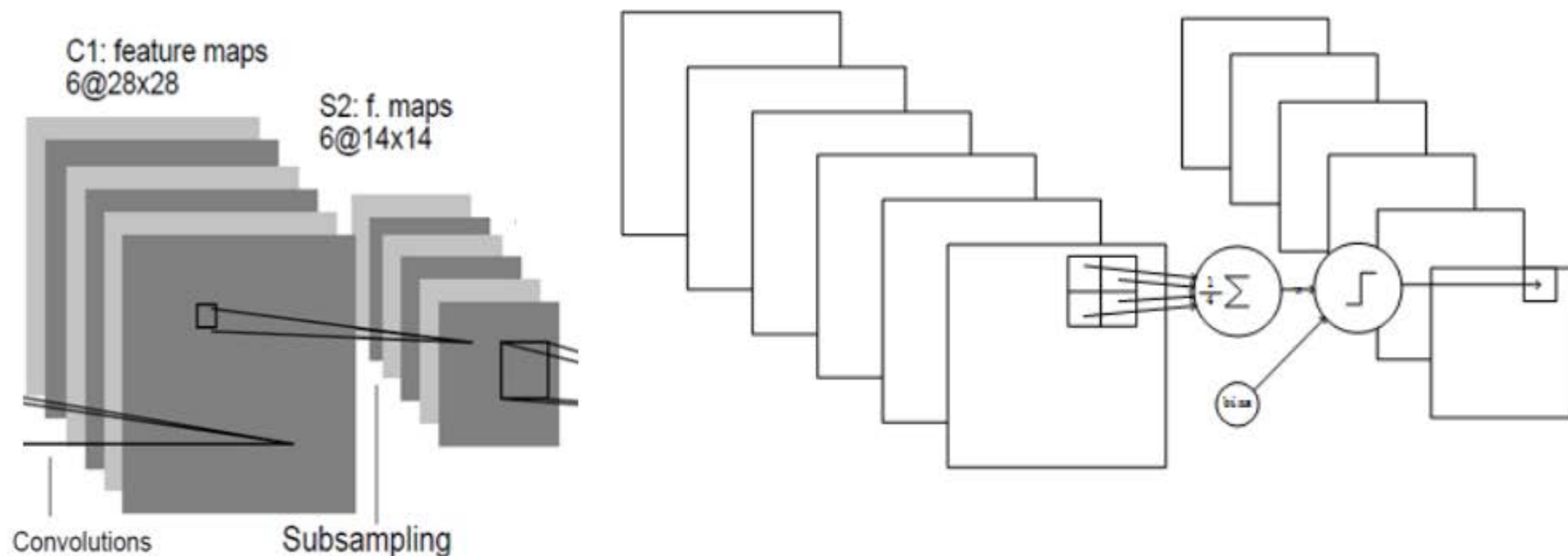
◆ LeNet-5第二层：池化层S2

S2层是一个池化层（为什么要下采样？利用图像局部相关性的原理，对图像进行亚采样，可以减少数据处理量，同时保留有用信息）。

C1层输出的是6个28x28的特征图谱，再分别进行用2x2池化，得到S2层的结果为6个14x14 【 $(28-2)/2+1$ 】 的图。

每个特征图谱使用一个池化核，有 $(2 \times 2 + 1 \text{个偏置}) \times 14 \times 14 \times 6 = 5880$ 个连接。

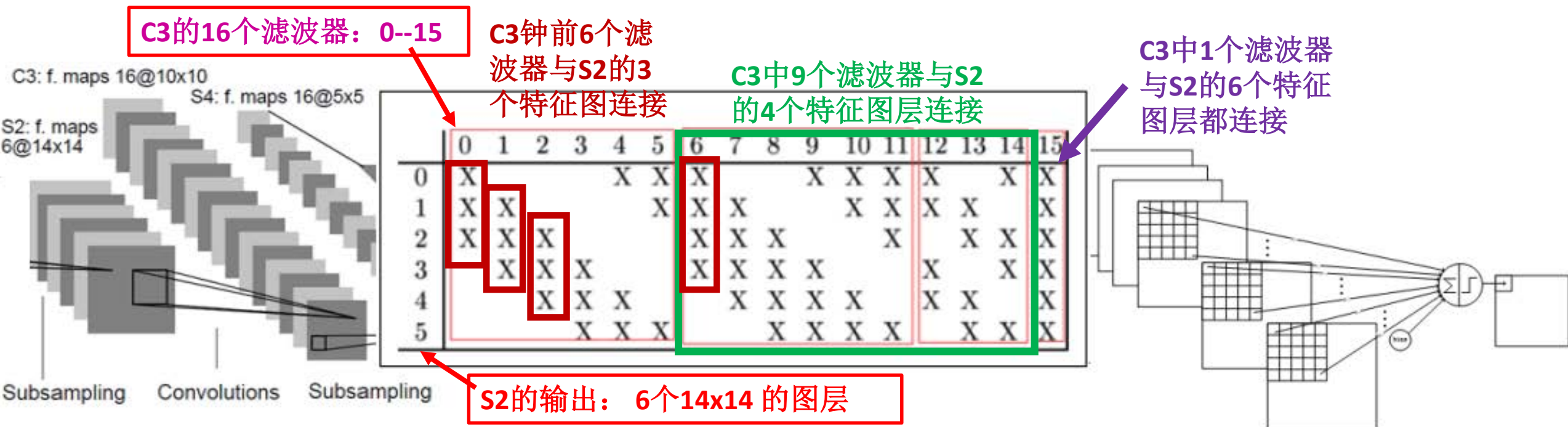
S2层的网络连接结构如下右图



$$\text{outLength} = (\text{inLength} - \text{filterLength} + 2 * \text{padding}) / \text{strideLength} + 1$$

◆ LeNet-5第三层：卷积层C3

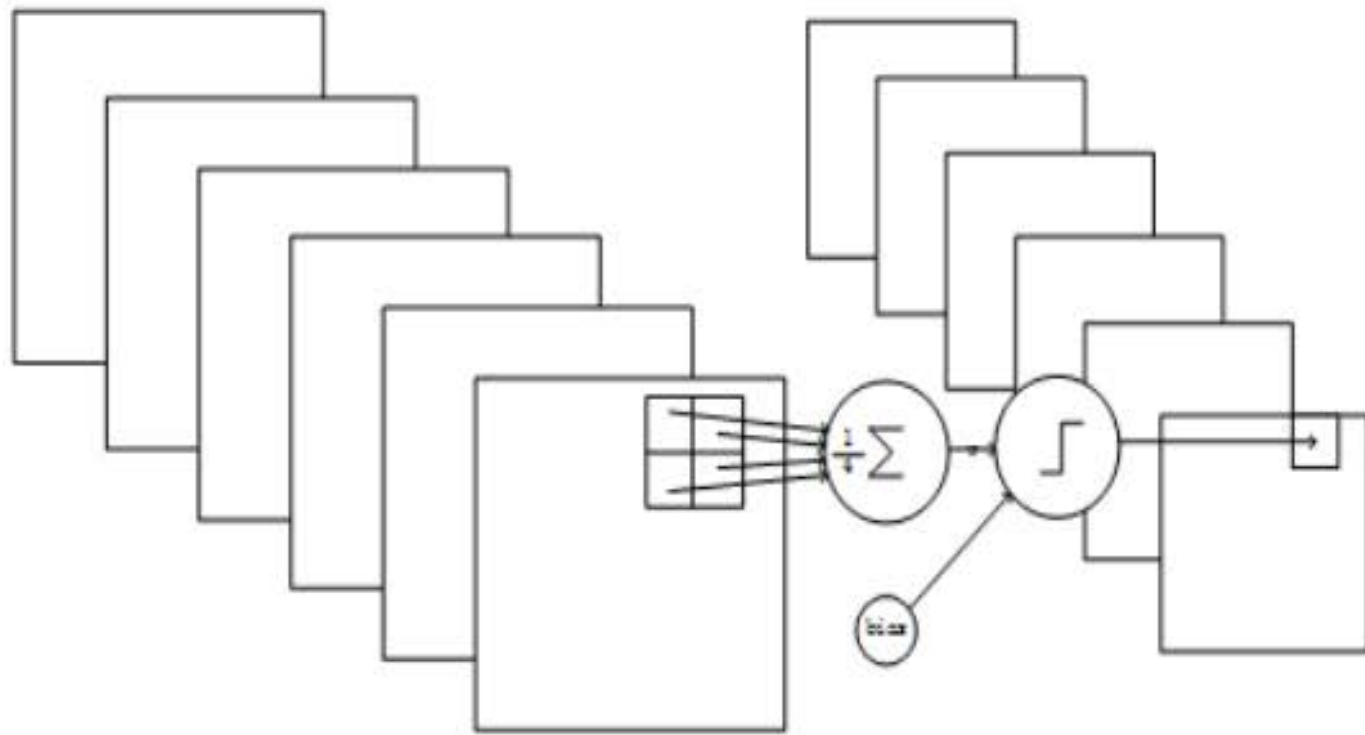
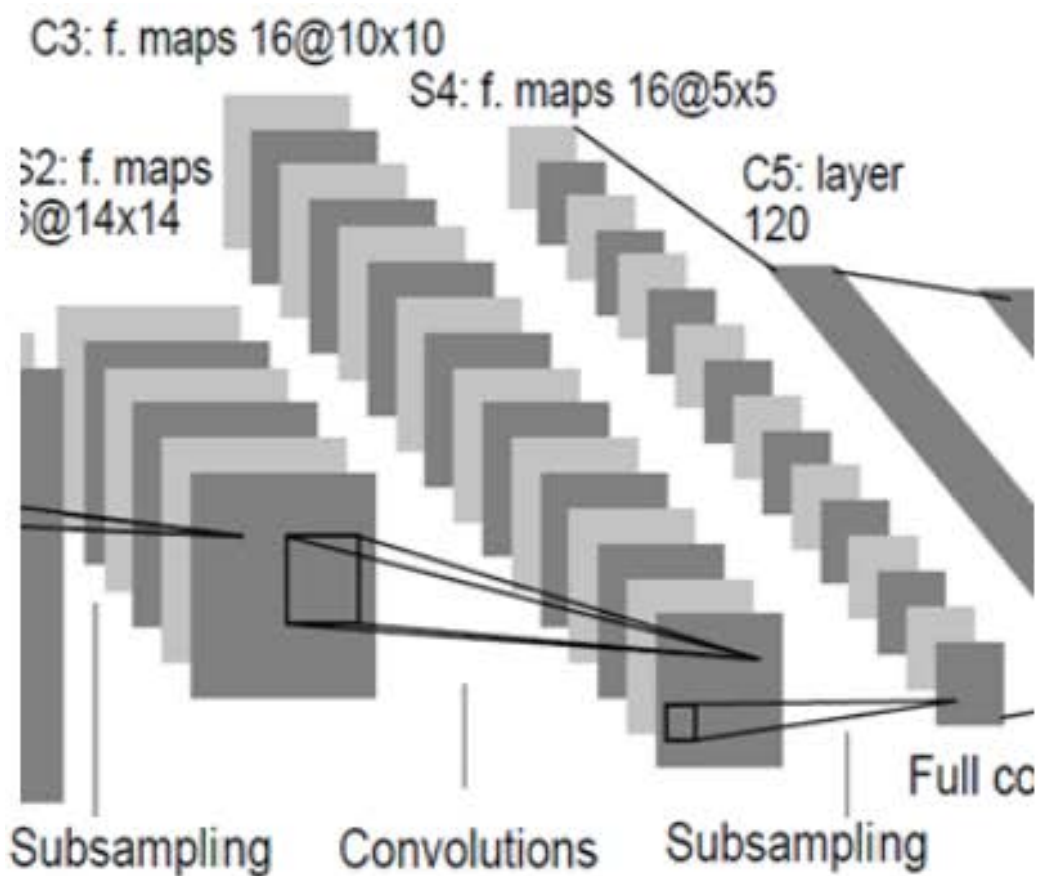
- C3层有**16个5*5的滤波器**，C3的每个滤波器与S2中的多个（不是所有）特征图相连，即**S2（6个14x14的图层）与C3不是全连接**，它们的连接的方式**如下表**所示。
- 这种不对称的组合连接的方式有利于提取多种组合特征。
- C3层输出**16个10x10**（14-5+1）的特征图。
- 该层有 $(5 \times 5 \times 3 + 1) \times 6 + (5 \times 5 \times 4 + 1) \times 6 + (5 \times 5 \times 4 + 1) \times 3 + (5 \times 5 \times 6 + 1) \times 1 = 1516$ 个训练参数，共有 $1516 \times 10 \times 10 = 151600$ 个连接。



$$\text{outLength} = (\text{inLength} - \text{filterLength} + 2 * \text{padding}) / \text{strideLength} + 1$$

◆ LeNet-5第四层：池化层 S4

- C3层的输出是16个10x10的图，再分别进行做2x2的池化，得到16个5x5的图。
- $(2 \times 2 + 1 \text{ 个偏置}) \times 5 \times 5 \times 16 = 2000$ 个连接。连接的方式与S2层类似，如下所示。



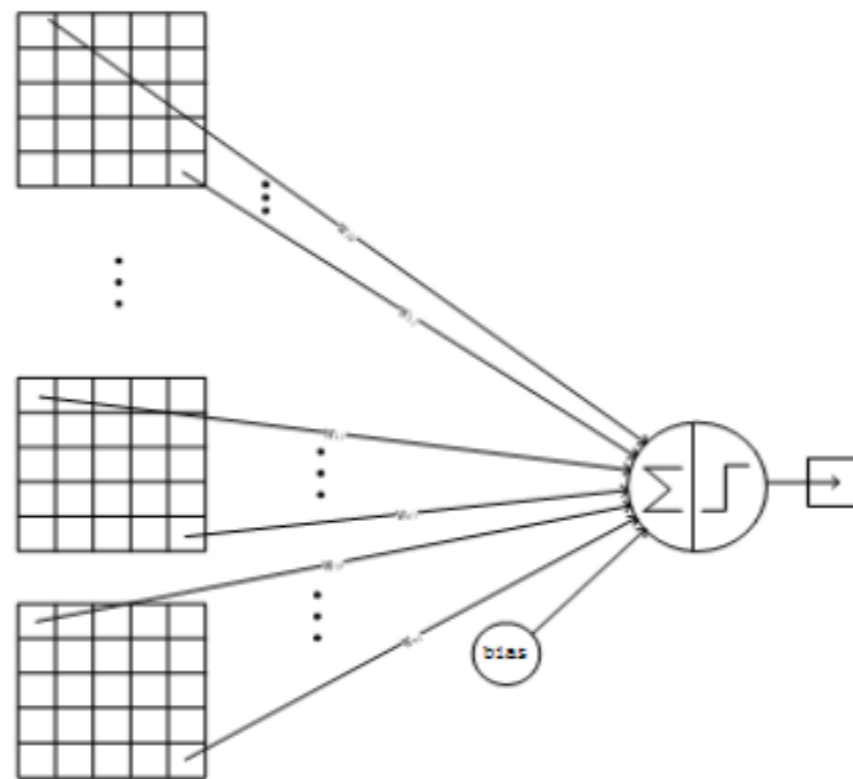
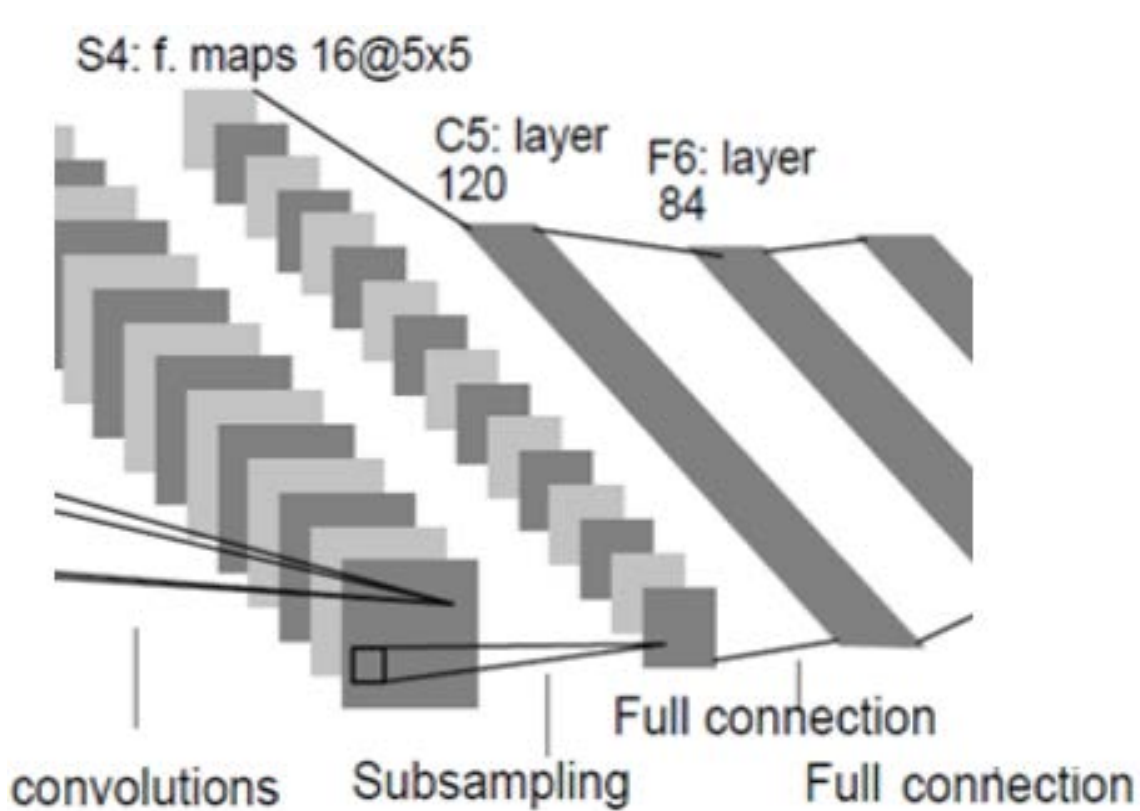
$$\text{outLength} = (\text{inLength} - \text{filterLength} + 2 * \text{padding}) / \text{strideLength} + 1$$

◆ LeNet-5第五层：全连接层C5（应为F5）

C5层是一个全连接层（卷积层），采用120个16x5x5的滤波器。

由于S4层的输出为16个5x5的特征图，与C5的卷积核大小相同，所以卷积后形成的图的大小为1x1。

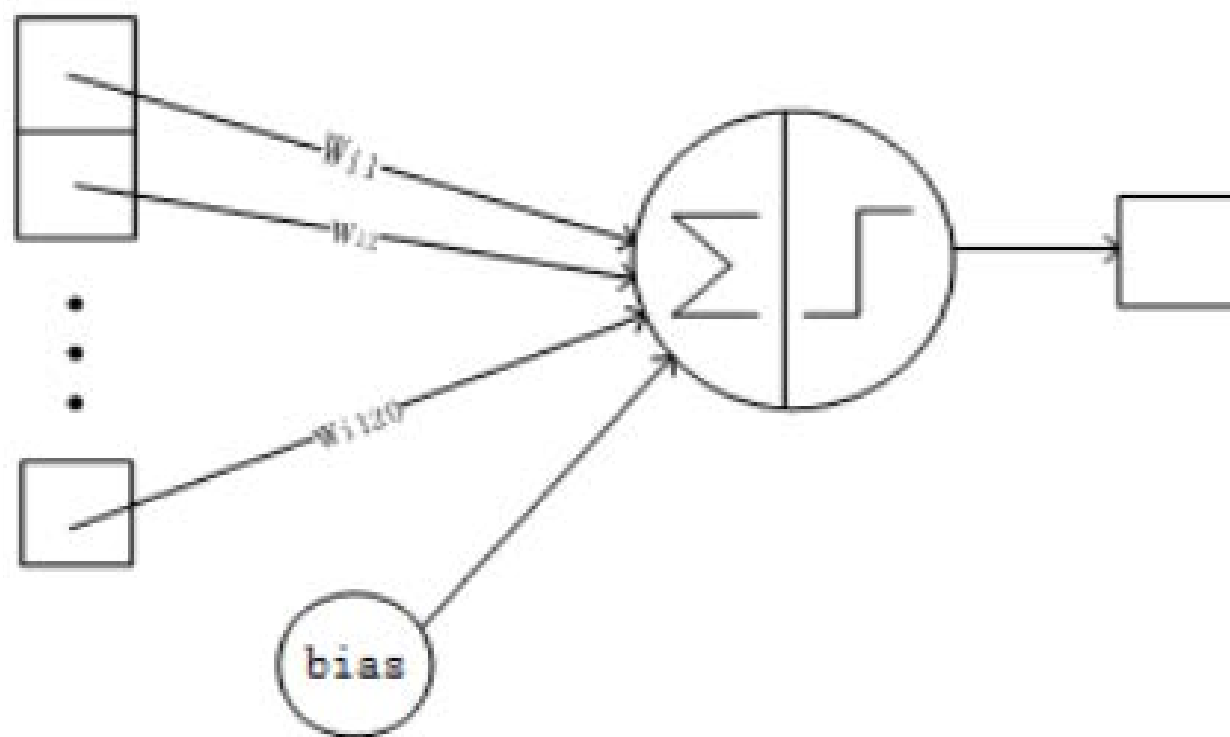
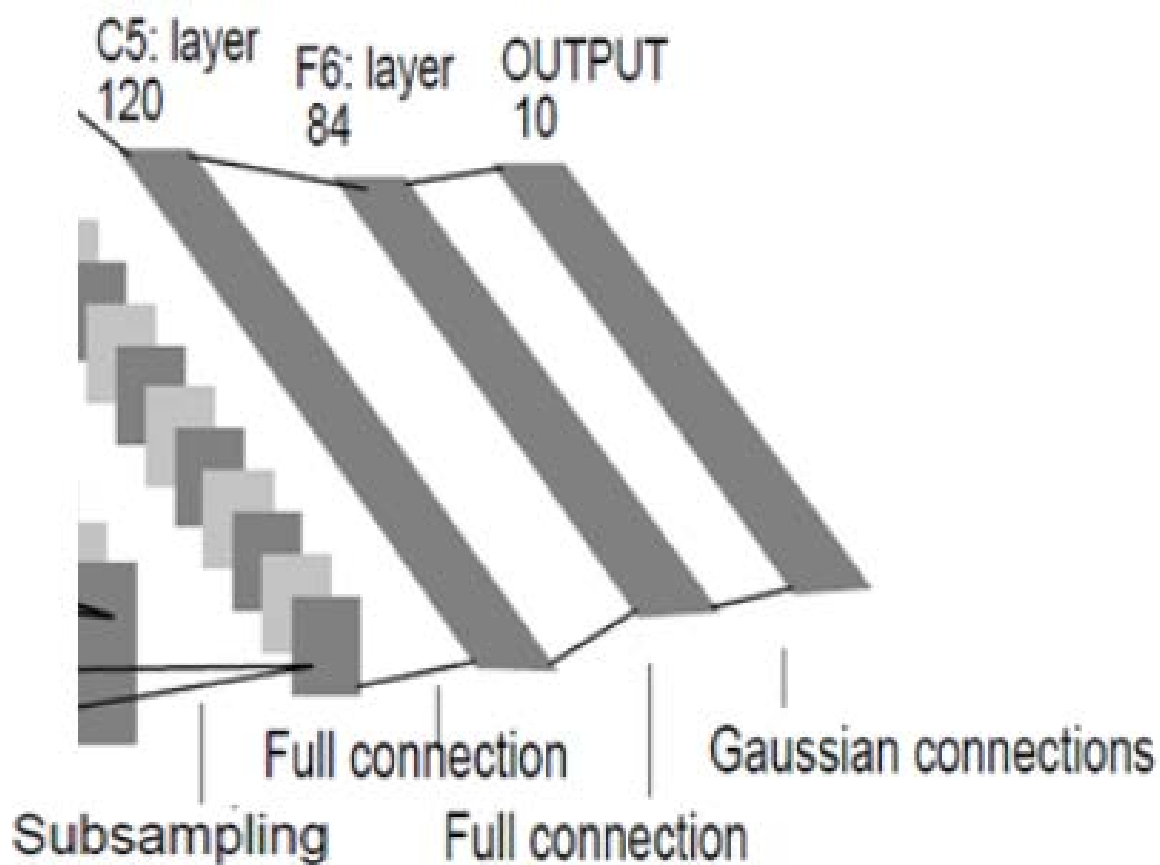
C5的输出为120位的1维向量。每个都与上一层的16个图层相连（所以是全连接）。所以共有 $(5 \times 5 \times 16 + 1) \times 120 = 48120$ 个参数，同样有48120个连接。



$$\text{outLength} = (\text{inLength} - \text{filterLength} + 2 * \text{padding}) / \text{strideLength} + 1$$

◆ LeNet-5第六层：全连接层F6

F6层是全连接层。其输入为C5层的120维向量。F6层有84个节点，对应于一个7x12的比特图，该层的训练参数和连接数都是 $(120 + 1) \times 84 = 10164$ 。



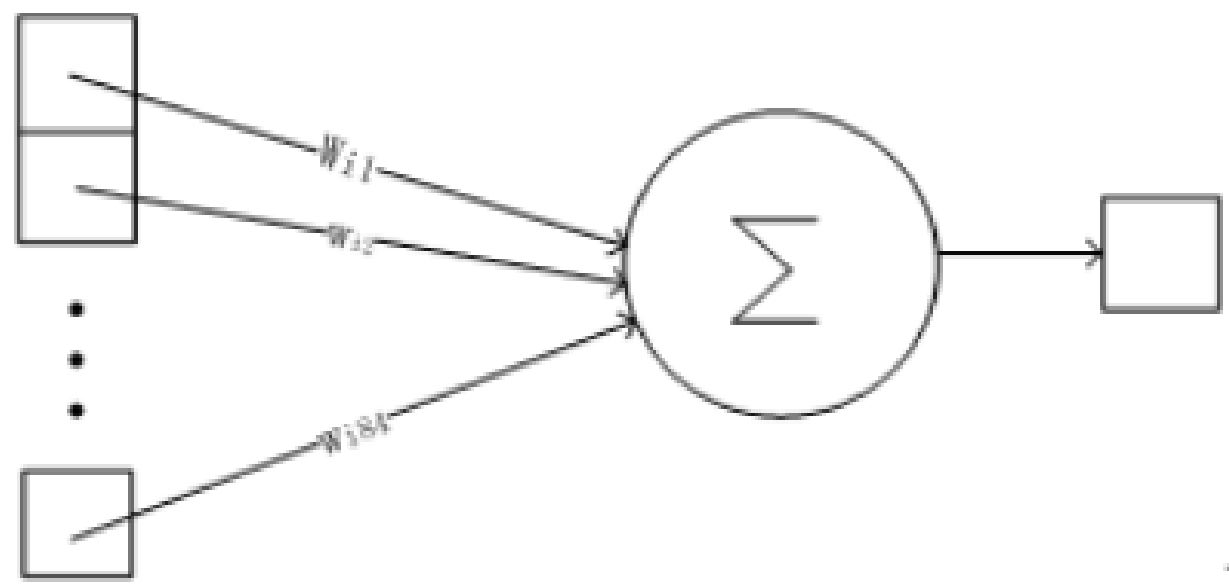
$$\text{outLength} = (\text{inLength} - \text{filterLength} + 2 * \text{padding}) / \text{strideLength} + 1$$

◆ LeNet-5第七层：全连接层Output

Output层也是全连接层，共有10个节点，分别代表数字0到9，如果节点i的输出值为0，则网络识别的结果是数字i。采用的是径向基函数（RBF）的网络连接方式。假设x是上一层的输入，y是RBF的输出，则RBF输出的计算方式是

$$y_i = \sum_j (x_j - w_{ij})^2$$

yi 的值由i的比特图编码（即参数Wij）确定。
yi 越接近于0，则标明输入越接近于i的比特图编码，表示当前网络输入的识别结果是字符i。



该层有84x10=840个设定的参数和连接。连接的方式如上图。
以上是LeNet-5的卷积神经网络的完整结构，共约有60,840个训练参数， 340,908个连接。