机器学习（本科生公选课）GEC6531

第11节 神经网络 Neural Networks

计算机科学与技术学院

张瑞 教授

邮箱：ruizhang6@hust.edu.cn

签到 & 思考

■ 微助教签到（学校要求）

1. 加入课堂：微信扫码或者通过微助教公众号

2. 微信扫码签到

回顾感知机

今天的目录

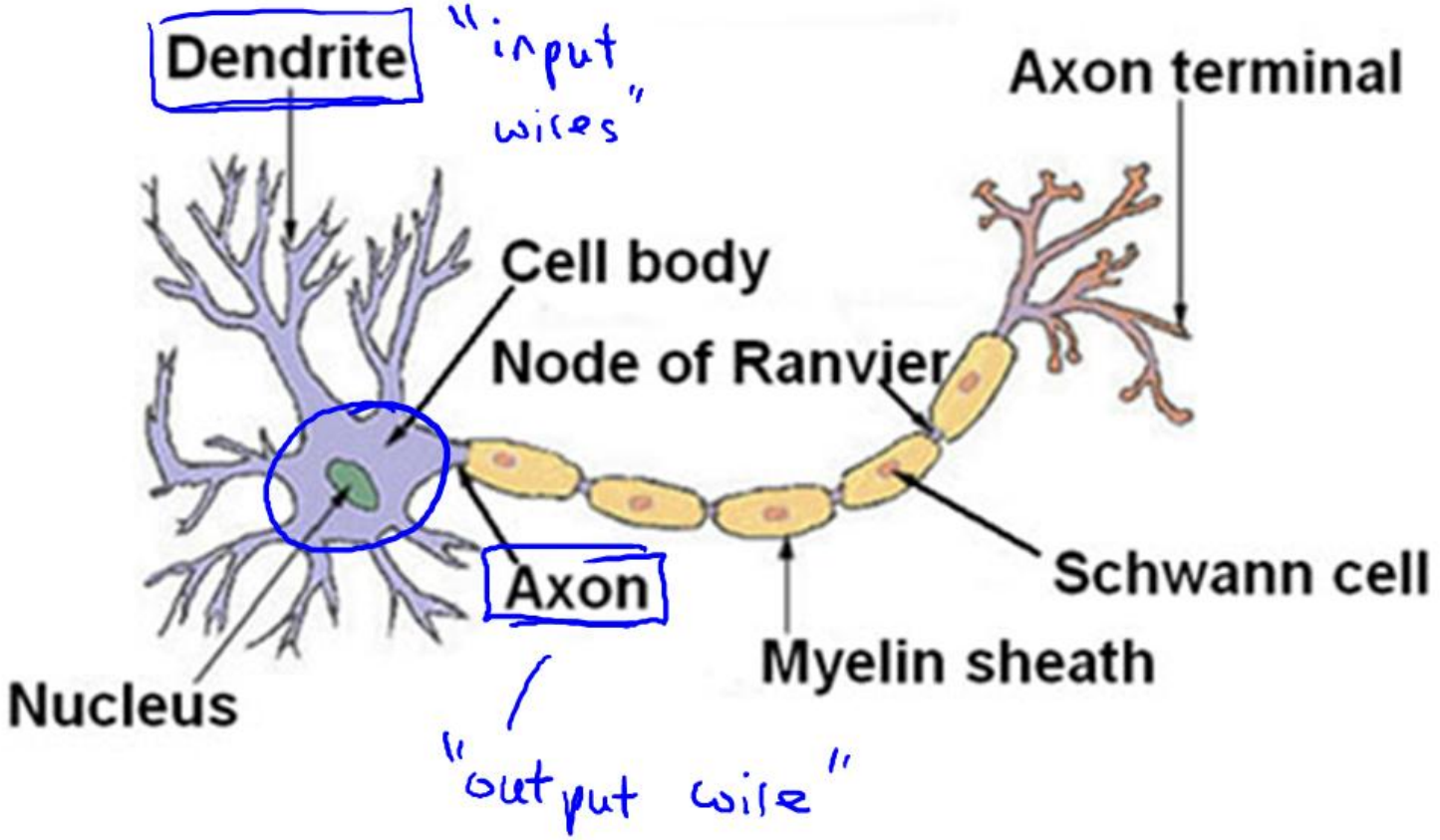
- **回顾感知机**
  - 结构
  - 实际中的分类问题
- **神经网络的表达**
- **神经元和激活函数**
  - Logistic 函数
  - Tanh 函数
  - ReLU 函数
- **前馈神经网络**
- **卷积神经网络 (CNN)**
  - 卷积层
  - 池化层
- **典型的卷积神经网络**

今天的目录

- **回顾感知机**
  - 结构
  - 实际中的分类问题
- **神经网络的表达**
- **神经元和激活函数**
  - Logistic 函数
  - Tanh 函数
  - ReLU 函数
- **前馈神经网络**
- **卷积神经网络 (CNN)**
  - 卷积层
  - 池化层
- **典型的卷积神经网络**

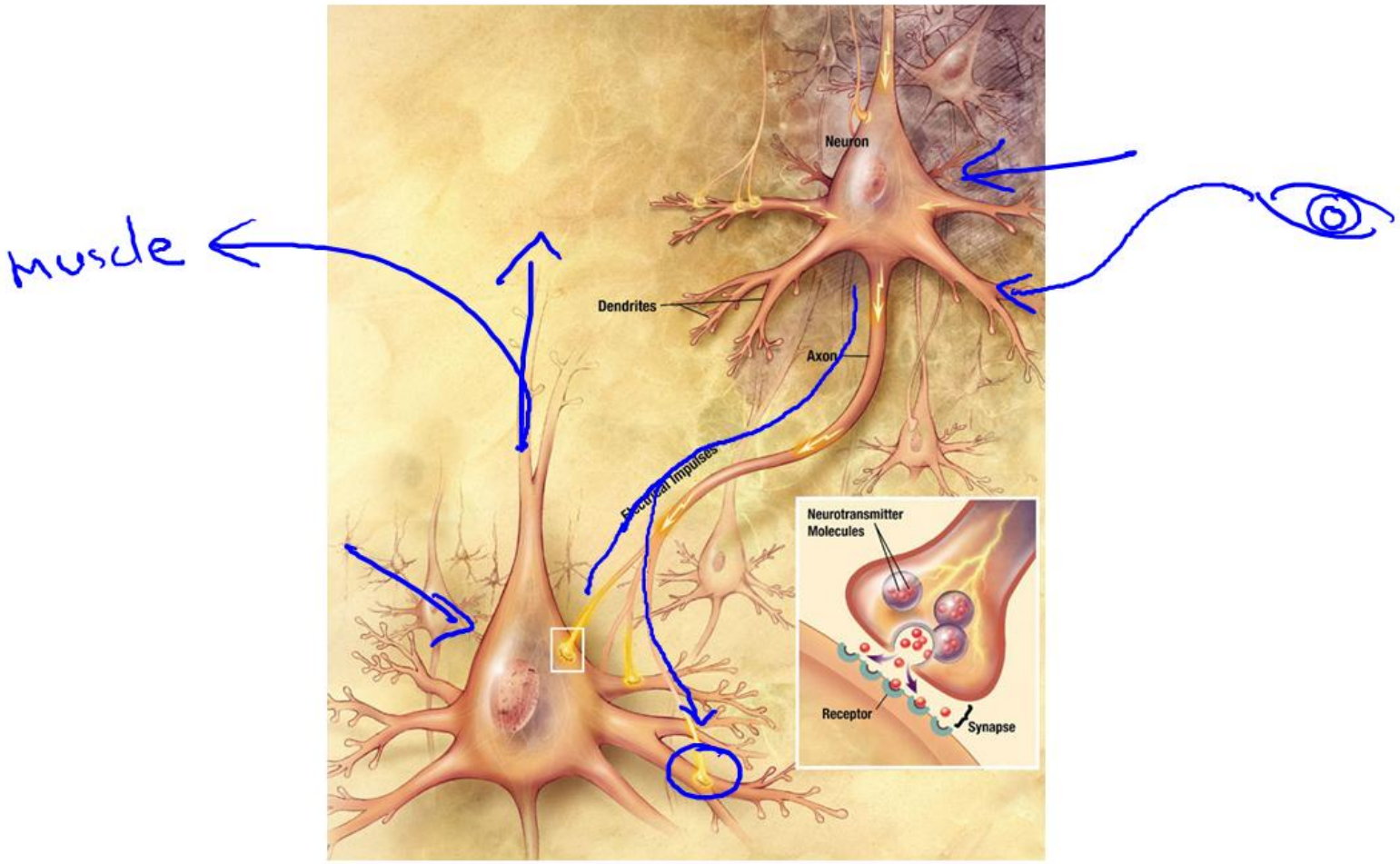
与脑神经类比

Neuron in the brain



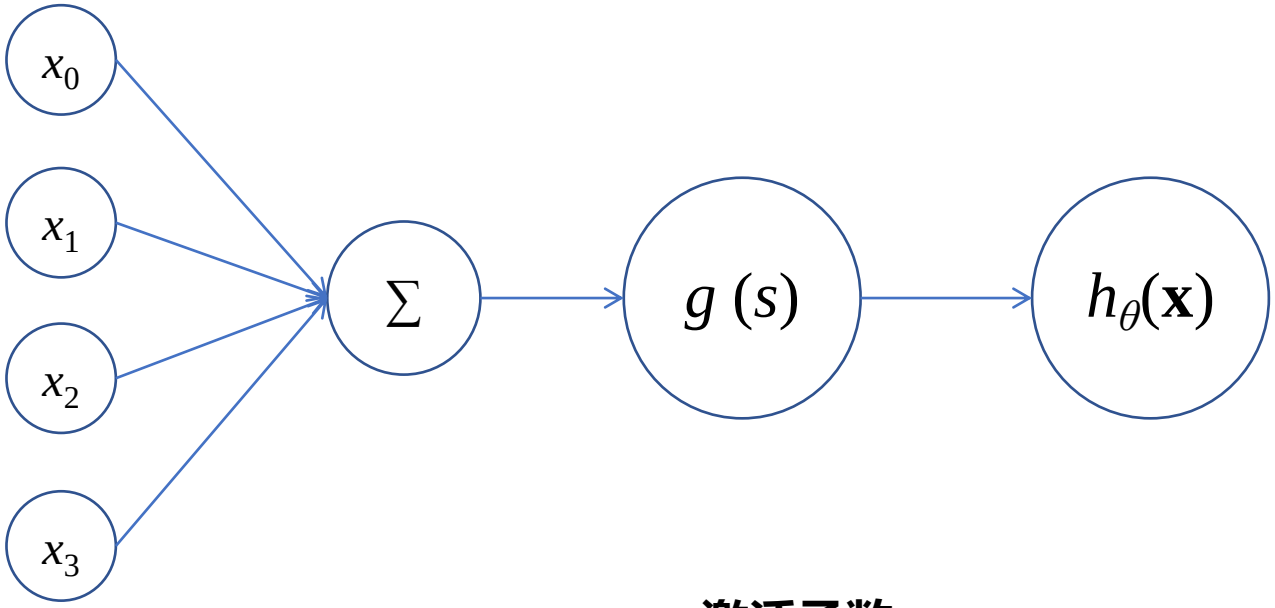
与脑神经类比

Neurons in the brain



[Credit: US National Institutes of Health, National Institute on Aging]

感知机 Perceptron



如果 $s \geq 0$, $h_\theta(\mathbf{x})$ 预测为正类

如果 $s < 0$, $h_\theta(\mathbf{x})$ 预测为负类

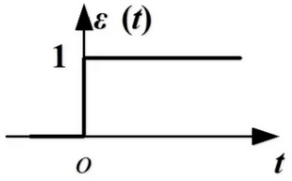
$$s = \sum_{i=0}^{m} x_i w_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

激活函数

Step function

阶跃函数

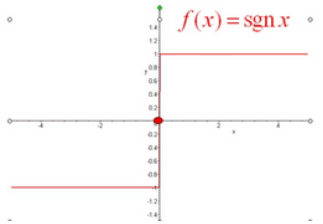
$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$$



Sign function

符号函数

$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$

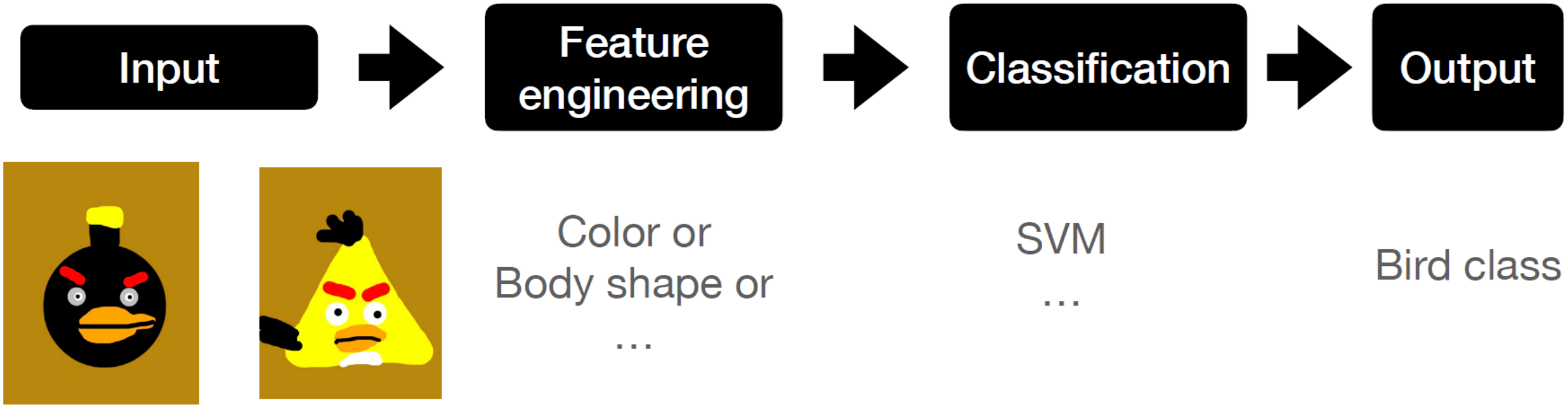


给鸟分类

VS

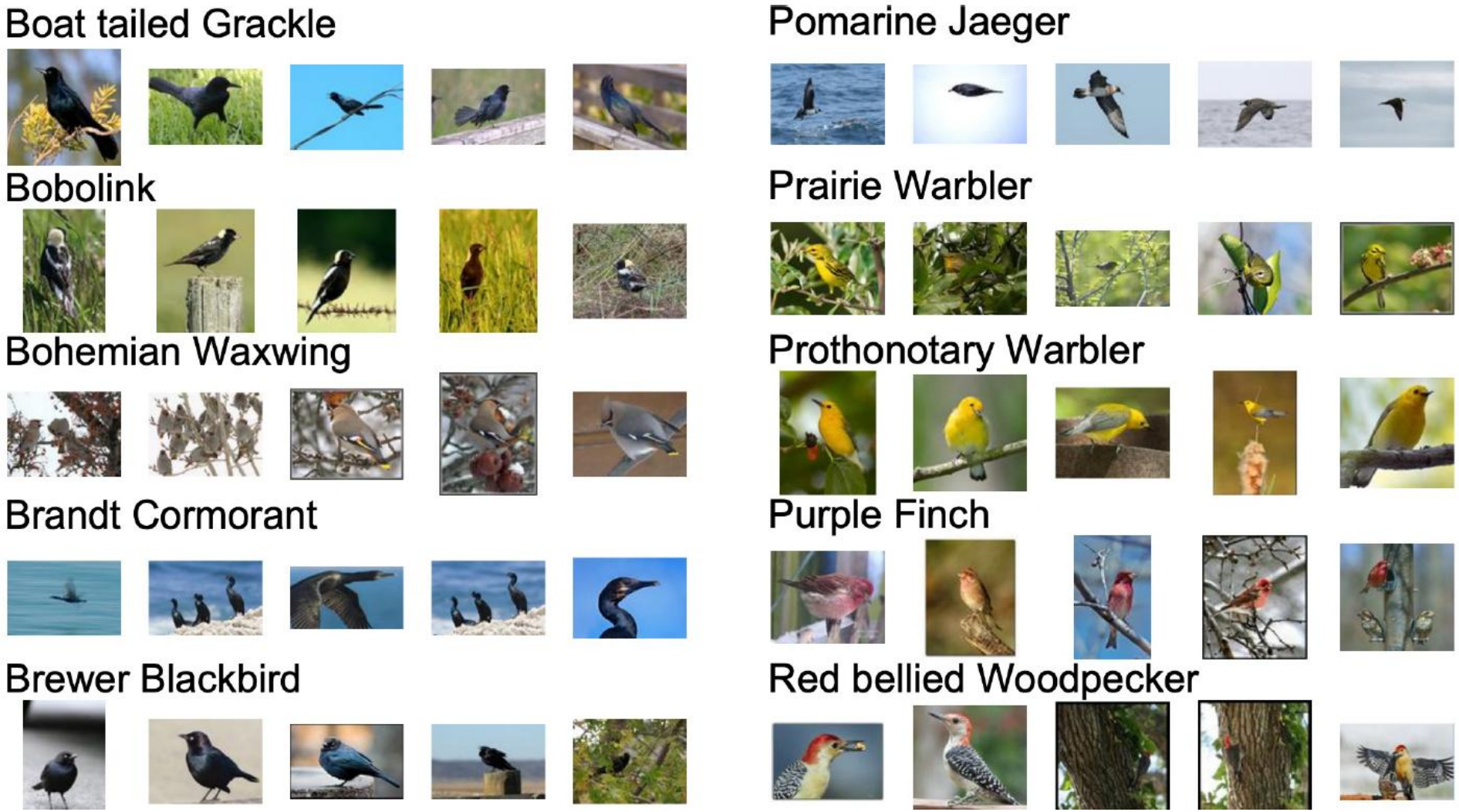
Source: by Papachan (CC BY)
https://www.sketchport.com/drawing/4524248689278976/bomb-and-chuck

给鸟分类



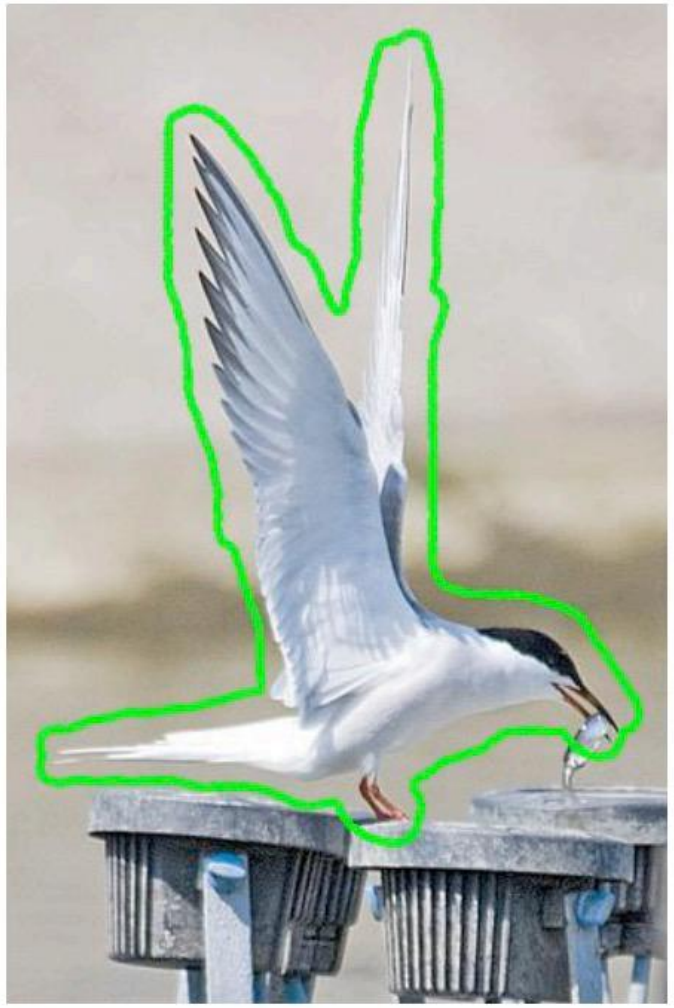
Source: by Papachan (CC BY)
https://www.sketchport.com/drawing/4524248689278976/bomb-and-chuck

实际中给鸟分类



Source: Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).

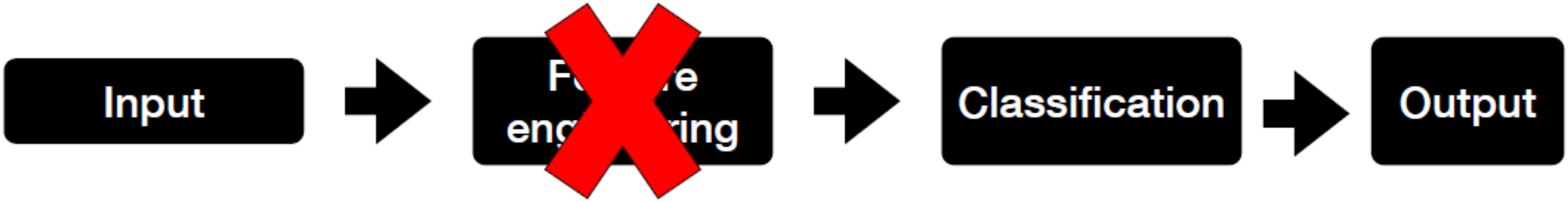
实际中给鸟分类

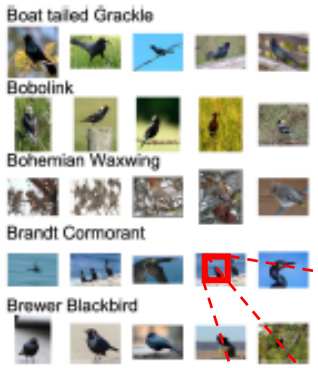


forehead_color	**black**	**black**	**black**
breast_pattern	**solid**	**solid**	**solid**
breast_color	**white**	**white**	**white**
head_pattern	**plain**	**capped**	plain
back_color	**white**	white	**black**
wing_color	grey/white	**grey**	**white**
leg_color	**orange**	orange	**orange**
size	**medium**	large	**medium**
bill_shape	needle	dagger	**dagger**
wing_shape	**pointed**	**tapered**	**long**
...	...	...	...
primary_color	white	white	white

Source: Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).

实际中给鸟分类





Use deep learning models!

Bird class

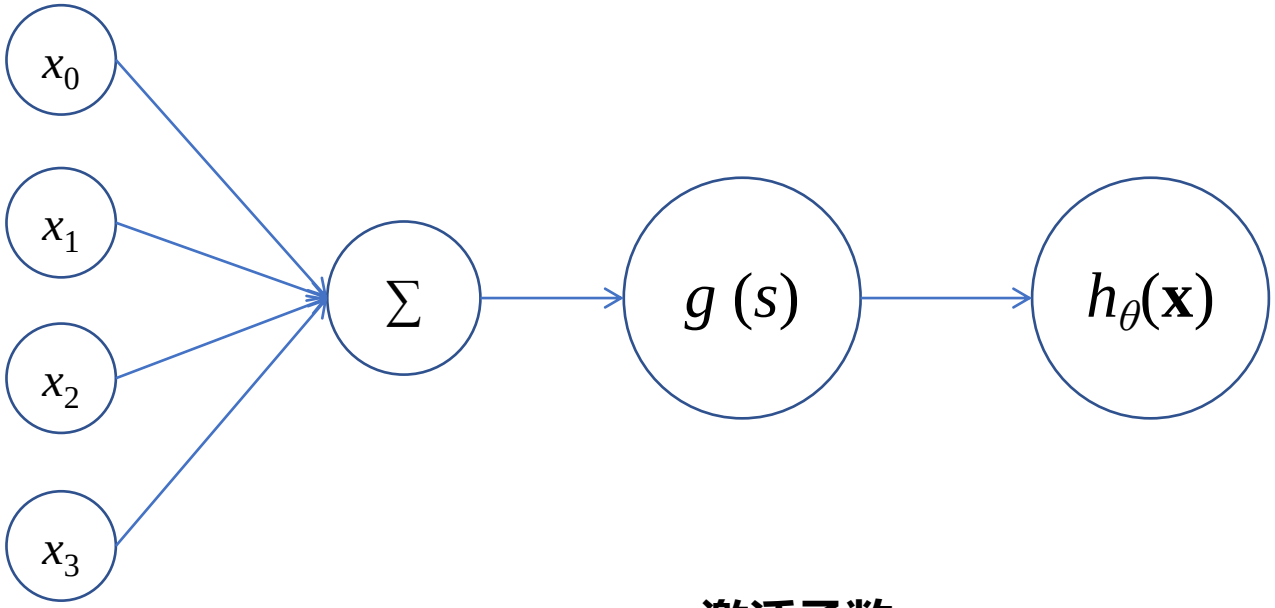
图片中的表达

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

今天的目录

- **回顾感知机**
  - 结构
  - 实际中的分类问题
- **神经网络的表达**
- **神经元和激活函数**
  - Logistic 函数
  - Tanh 函数
  - ReLU 函数
- **前馈神经网络**
- **卷积神经网络 (CNN)**
  - 卷积层
  - 池化层
- **典型的卷积神经网络**

感知机 Perceptron



如果 $s \geq 0$, $h_\theta(\mathbf{x})$ 预测为正类

如果 $s < 0$, $h_\theta(\mathbf{x})$ 预测为负类

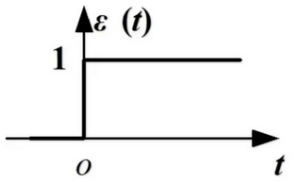
$$s = \sum_{i=0}^{m} x_i w_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

激活函数

Step function

阶跃函数

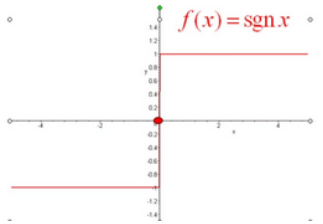
$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$$



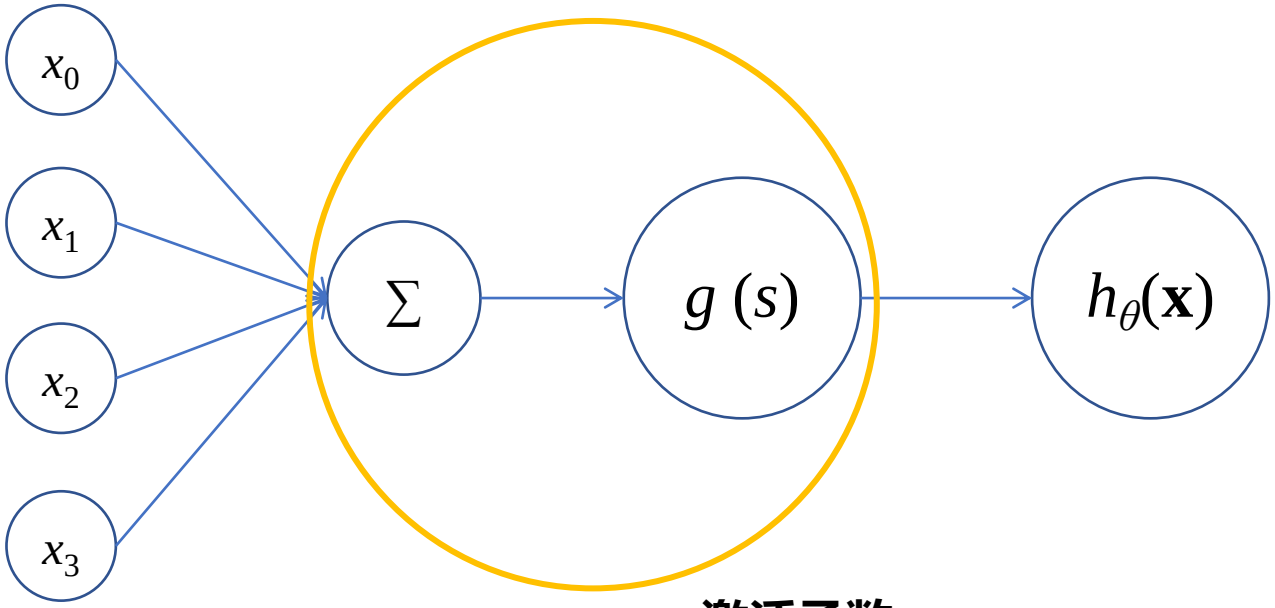
Sign function

符号函数

$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$



感知机 Perceptron：符号简化



如果 $s \geq 0$, $h_\theta(\mathbf{x})$ 预测为正类

如果 $s < 0$, $h_\theta(\mathbf{x})$ 预测为负类

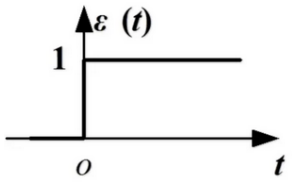
$$s = \sum_{i=0}^{m} x_i w_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

激活函数

Step function

阶跃函数

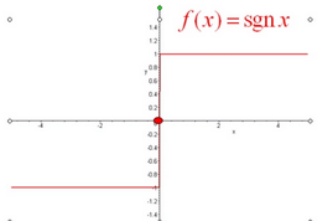
$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$$



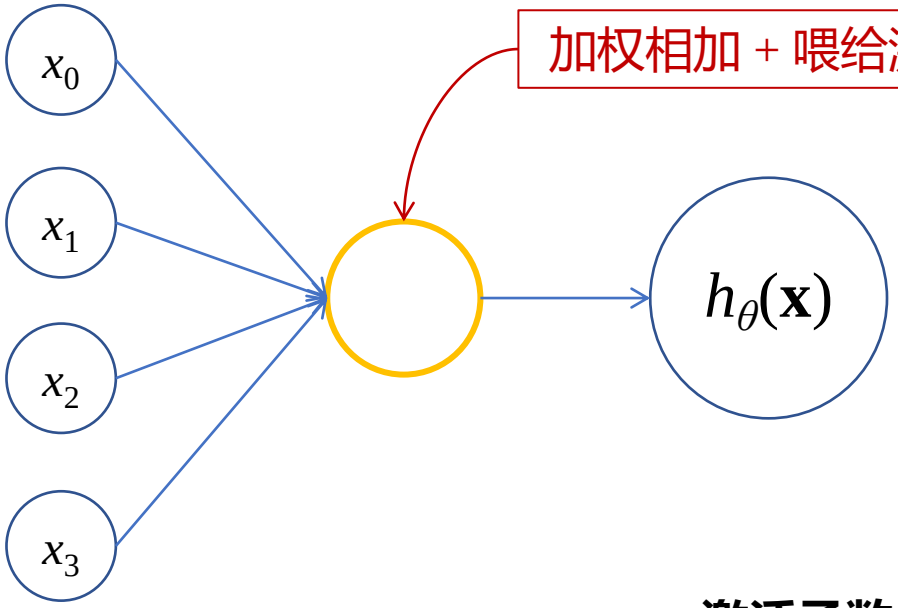
Sign function

符号函数

$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$



感知机 Perceptron：符号简化



加权相加 + 喂给激活函数并输出：$g\,(\mathbf{w}^{\mathrm{T}}\mathbf{x})$

如果 $s \geq 0$, $h_\theta(\mathbf{x})$ 预测为正类

如果 $s < 0$, $h_\theta(\mathbf{x})$ 预测为负类

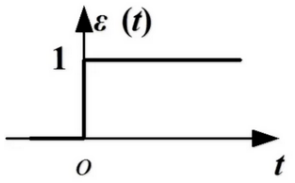
$$s = \sum_{i=0}^{m} x_i w_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

激活函数

Step function

阶跃函数

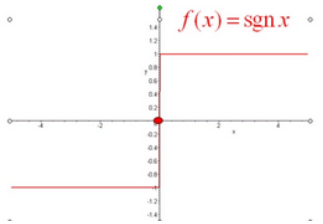
$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$$



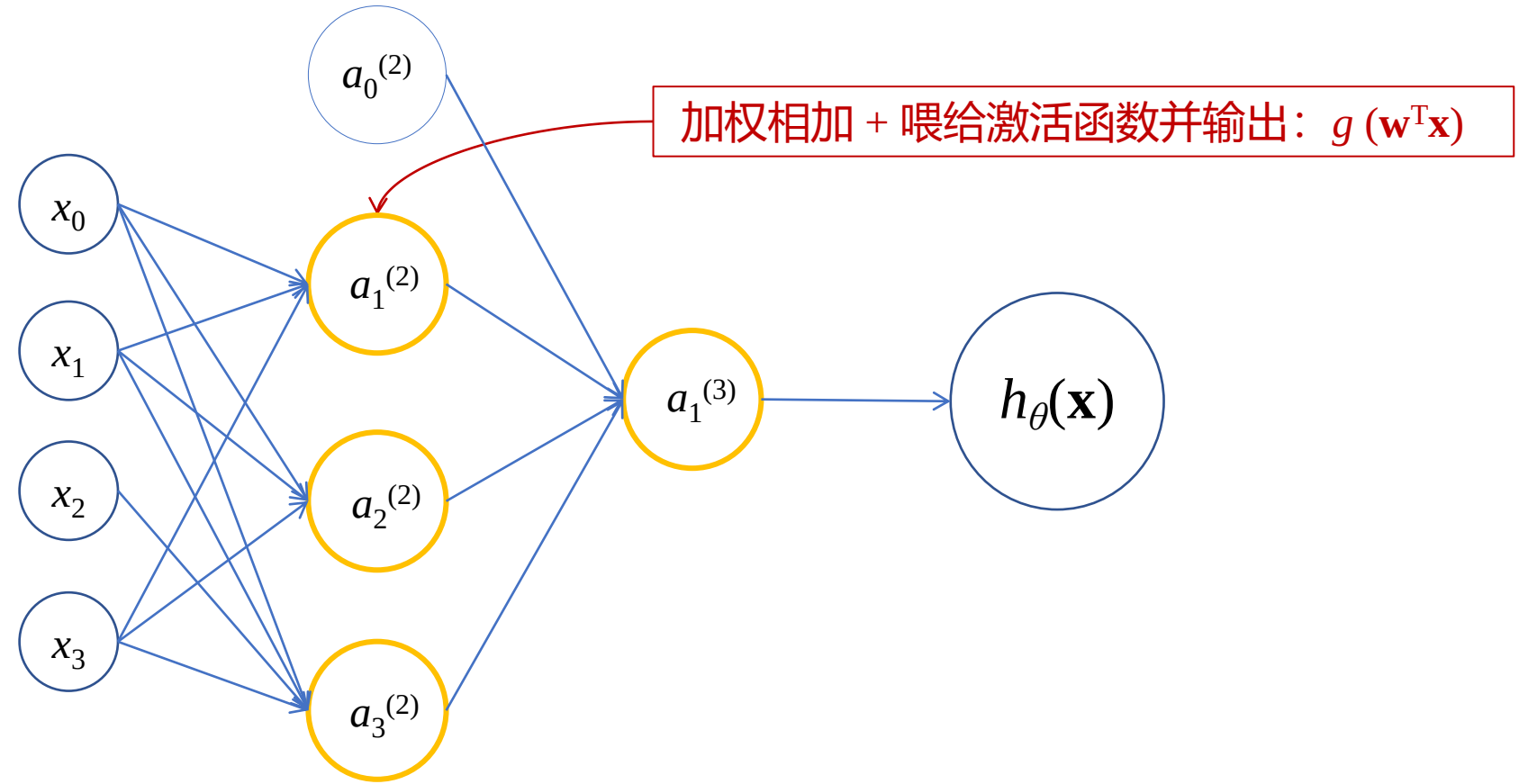
Sign function

符号函数

$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$

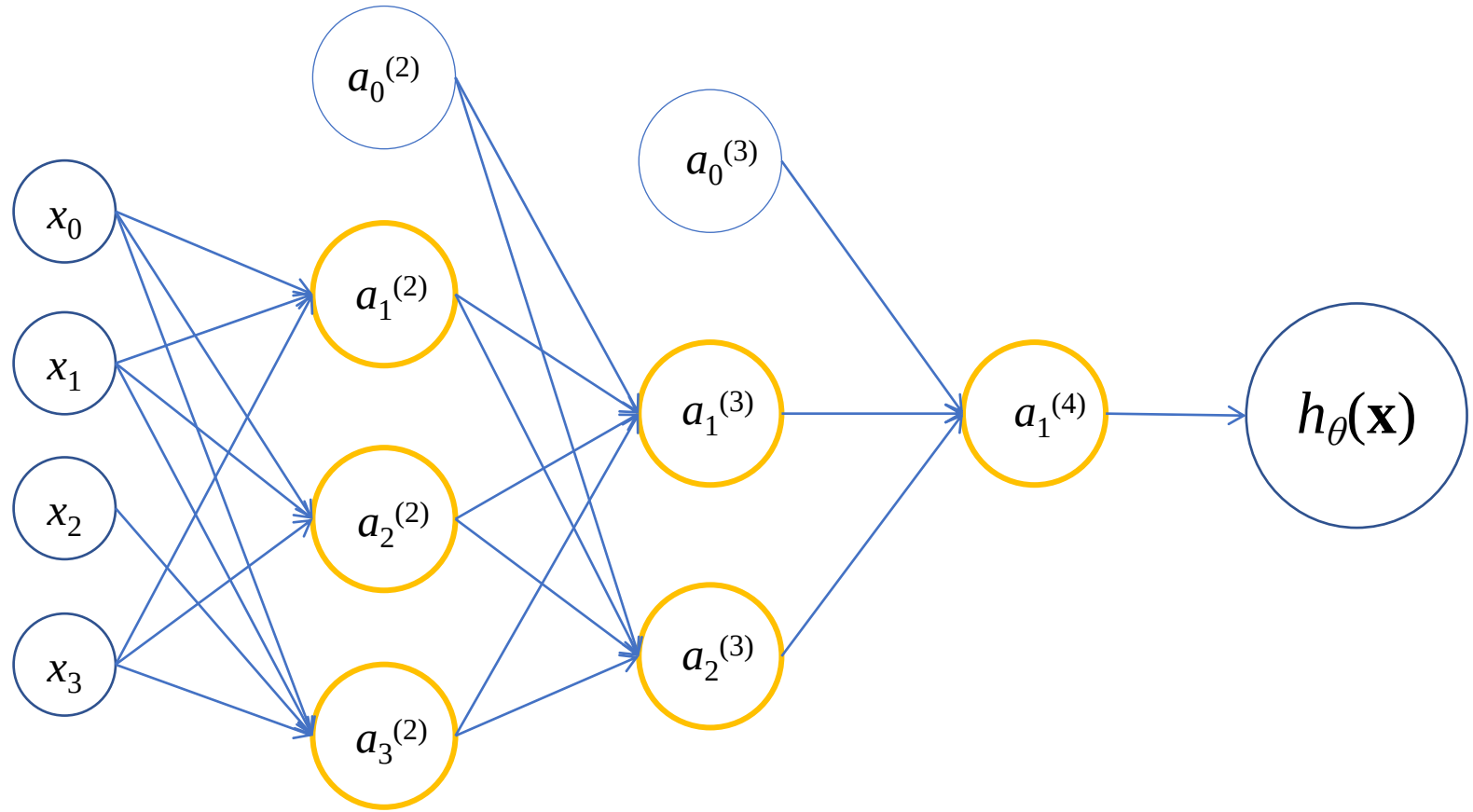


神经网络



第一层　　第二层　　第三层

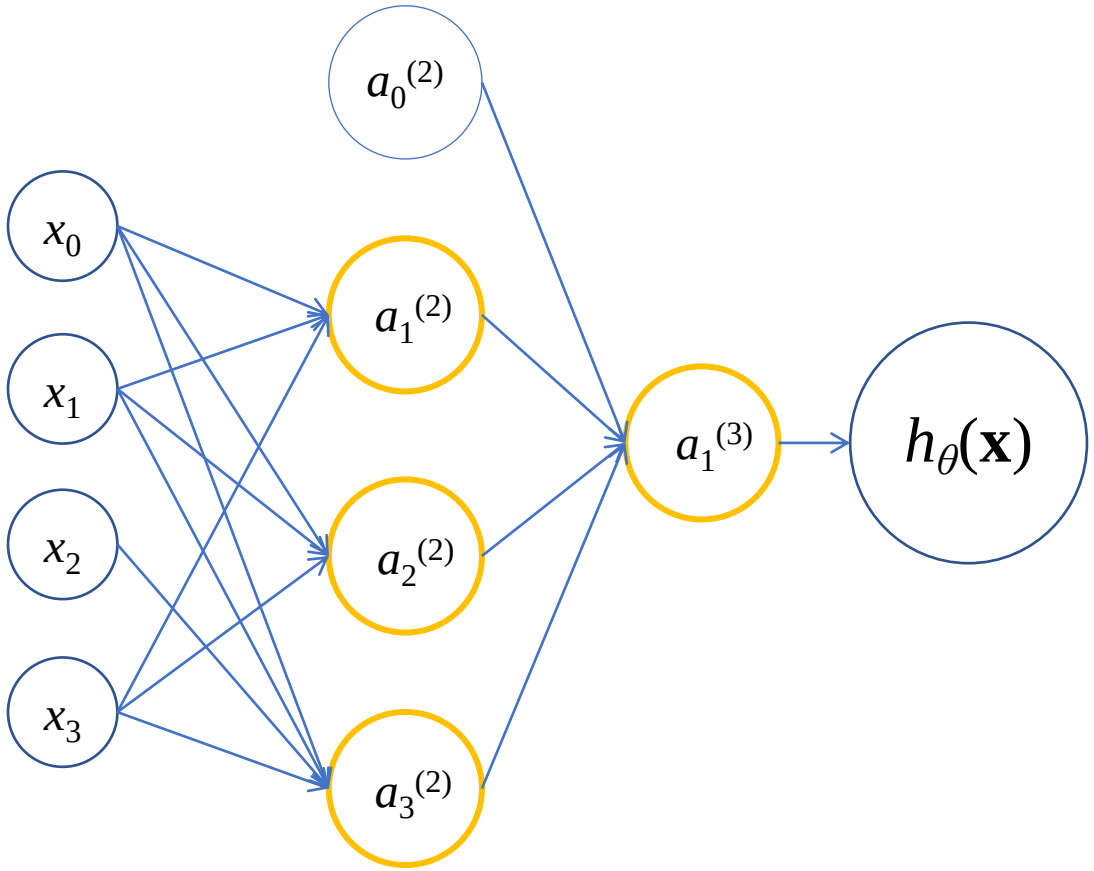
神经网络



第一层　　第二层　　第三层　　第四层

输入层　　隐藏层 (hidden layers)　　输出层

神经网络的表达



第一层　　第二层　　第三层

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_\Theta(x) = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

今天的目录

- **回顾感知机**
  - 结构
  - 实际中的分类问题
- **神经网络的表达**
- **神经元和激活函数**
  - Logistic 函数
  - Tanh 函数
  - ReLU 函数
- **前馈神经网络**
- **卷积神经网络 (CNN)**
  - 卷积层
  - 池化层
- **典型的卷积神经网络**

神经元

- 人工神经元（Artificial Neuron），简称神经元（Neuron），是构成神经网络的基本单元，其主要是模拟生物神经元的结构和特性，接收一组输入信号并产生输出
- 假设一个神经元接收 D 个输入 $x_1, x_2, \ldots, x_D$，令向量 $\mathbf{x} = [x_1; x_2; \ldots; x_D]$ 来表示这组输入，并用净输入（Net Input）$z \in \mathbb{R}$ 表示一个神经元所获得的输入信号 $\mathbf{x}$ 的加权和，

$$\begin{aligned} z &= \sum_{d=1}^{D} w_d x_d + b \\ &= \mathbf{w}^T \mathbf{x}, \end{aligned} \tag{1}$$

其中 $\mathbf{w} = [w_1; w_2; \ldots; w_D] \in \mathbb{R}^D$ 是 D 维的权重向量，$b \in \mathbb{R}$ 是偏置。
净输入 z 在经过一个非线性函数 $f(\cdot)$ 后，得到神经元的活性值（Activation）a，

$$a = f(z),$$

其中非线性函数 $f(\cdot)$ 称为激活函数（Activation Function）。

激活函数

激活函数在神经元中非常重要的。为了增强网络的表示能力和学习能力，激活函数需要具备以下几点性质：

- **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。

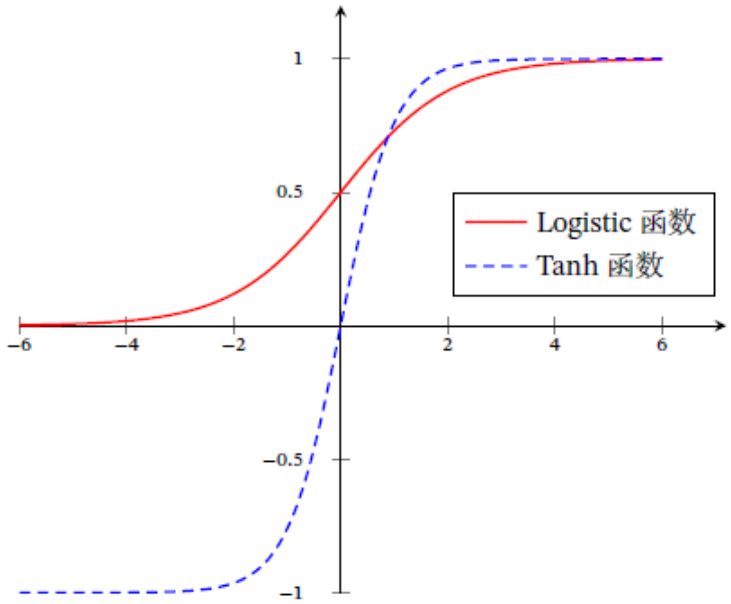
几种在神经网络中常用的激活函数：

- Sigmoid 型函数：Logistic 函数和 Tanh 函数
- ReLU 函数

Sigmoid 型函数: Logistic 函数

Logistic 函数定义为：

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



图：Logistic 函数和 Tanh 函数

- 当输入值在 0 附近时，Sigmoid 型函数近似为线性函数；当输入值靠近两端时，对输入进行抑制。输入越小，越接近于 0；输入越大，越接近于 1。

Sigmod 型函数：Tanh 函数

Tanh 函数也是一种 Sigmoid 型函数. 其定义为：

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

Tanh 函数可以看作放大并平移的 Logistic 函数，其值域是 $(-1, 1)$。

$$\tanh(x) = 2\sigma(2x) - 1$$

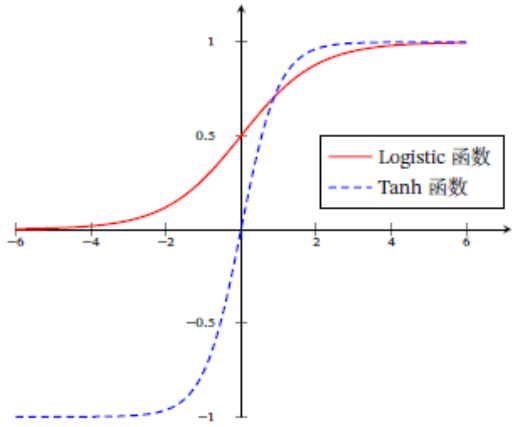


图: Logistic 函数和 Tanh 函数

- Tanh 函数的输出是零中心化的（Zero-Centered），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。

ReLU 函数

ReLU（Rectified Linear Unit，修正线性单元），是目前深度神经网络中经常使用的激活函数。ReLU 实际上是一个斜坡 (ramp) 函数，定义为

$$\text{ReLU}(x) = \max(0, x)$$

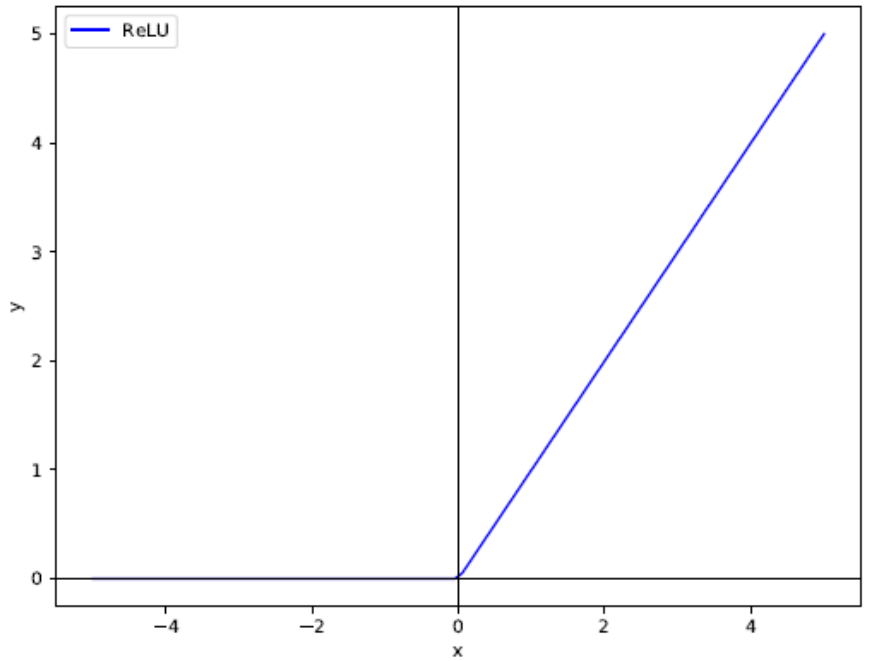


图: ReLU 函数

ReLU 函数

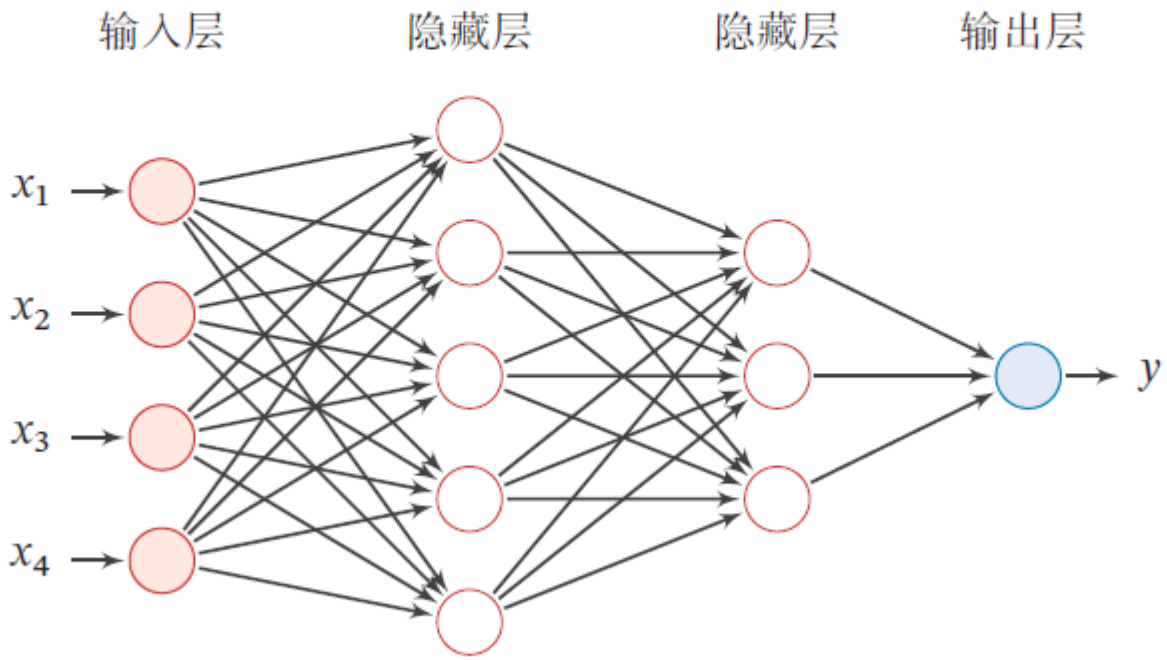
- **优点**：在优化方面，相比于 Sigmoid 型函数的两端饱和，ReLU 函数为左饱和函数，且在 $x > 0$ 时导数为 1，在一定程度上缓解了神经网络的梯度消失问题，加速梯度下降的收敛速度。
- **缺点**：ReLU 函数的输出是非零中心化的，给后一层的神经网络引入偏置偏移，会影响梯度下降的效率。此外，ReLU 神经元在训练时比较容易“消亡”。在训练时，如果参数在一次不恰当的更新后，第一个隐藏层中的某个 ReLU 神经元在所有的训练数据上都不能被激活，那么这个神经元自身参数的梯度永远都会是 0，在以后的训练过程中永远不能被激活。这种现象称为 ReLU 消亡问题（Dying ReLU Problem），并且也有可能会发生在其他隐藏层。

今天的目录

- **回顾感知机**
  - 结构
  - 实际中的分类问题
- **神经网络的表达**
- **神经元和激活函数**
  - Logistic 函数
  - Tanh 函数
  - ReLU 函数
- **前馈神经网络**
- **卷积神经网络 (CNN)**
  - 卷积层
  - 池化层
- **典型的卷积神经网络**

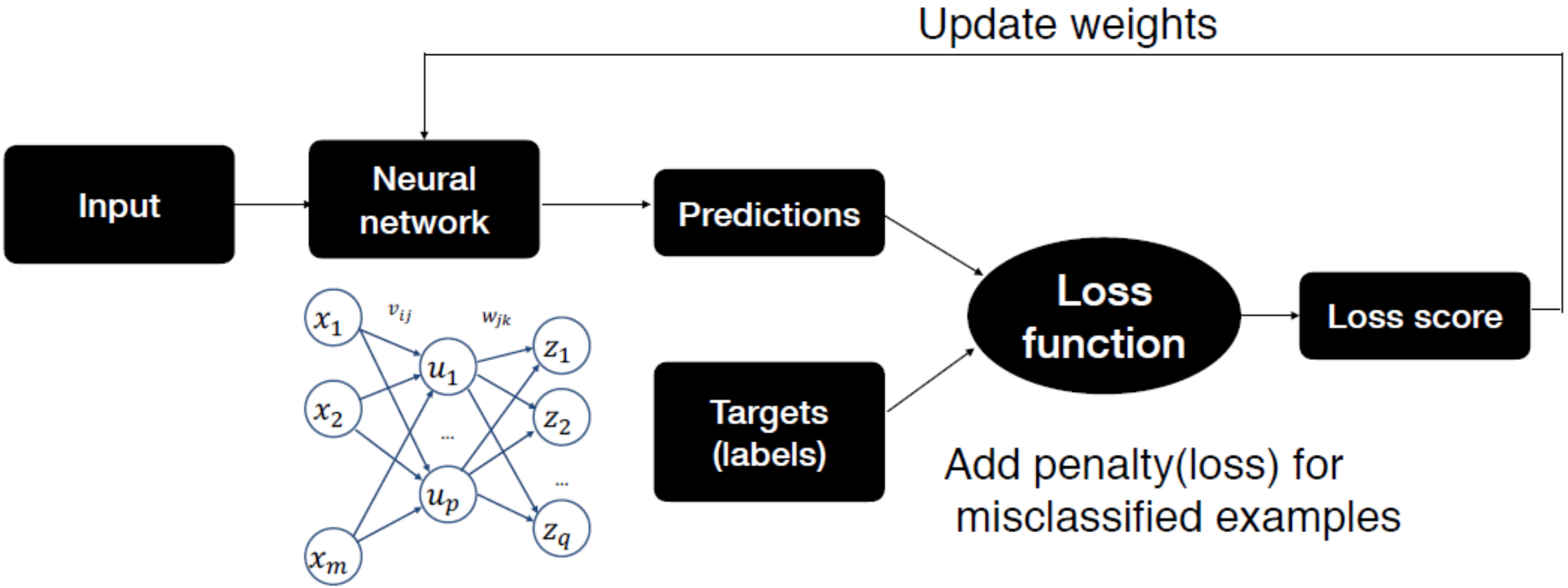
前馈神经网络

在前馈神经网络中，各神经元分别属于不同的层。每一层的神经元可以接收前一层神经元的信号，并产生信号输出到下一层。第 0 层称为输入层，最后一层称为输出层，其他中间层称为隐藏层。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。



图：多层前馈神经网络

前馈神经网络：损失函数计算



今天的目录

- **回顾感知机**
  - 结构
  - 实际中的分类问题
- **神经网络的表达**
- **神经元和激活函数**
  - Logistic 函数
  - Tanh 函数
  - ReLU 函数
- **前馈神经网络**
- **卷积神经网络 (CNN)**
  - 卷积层
  - 池化层
- **典型的卷积神经网络**

卷积神经网络 (Convolutional Neural Network, CNN)

卷积神经网络（Convolutional Neural Network，CNN）是一种具有局部连接、权重共享等特性的深层前馈神经网络。

CNN

目前的卷积神经网络一般是由卷积层、池化层和全连接层交叉堆叠而成的前馈神经网络。
卷积神经网络有三个结构上的特性：局部连接、权重共享、池化。
这些特性使得卷积神经网络具有一定程度上的平移、缩放和旋转不变性。

卷积层

卷积层的作用是提取一个局部区域的特征，不同的卷积核相当于不同的特征提取器。
首先，我们暂时忽略通道（第三维）这一情况，看看如何处理二维图像数据和隐藏表示。
在图中，输入是高度为 3、宽度为 3 的二维张量。卷积核的高度和宽度都是 2 ，而卷积核窗口（或卷积窗口）的形状由内核的高度和宽度决定（即 2×2）。

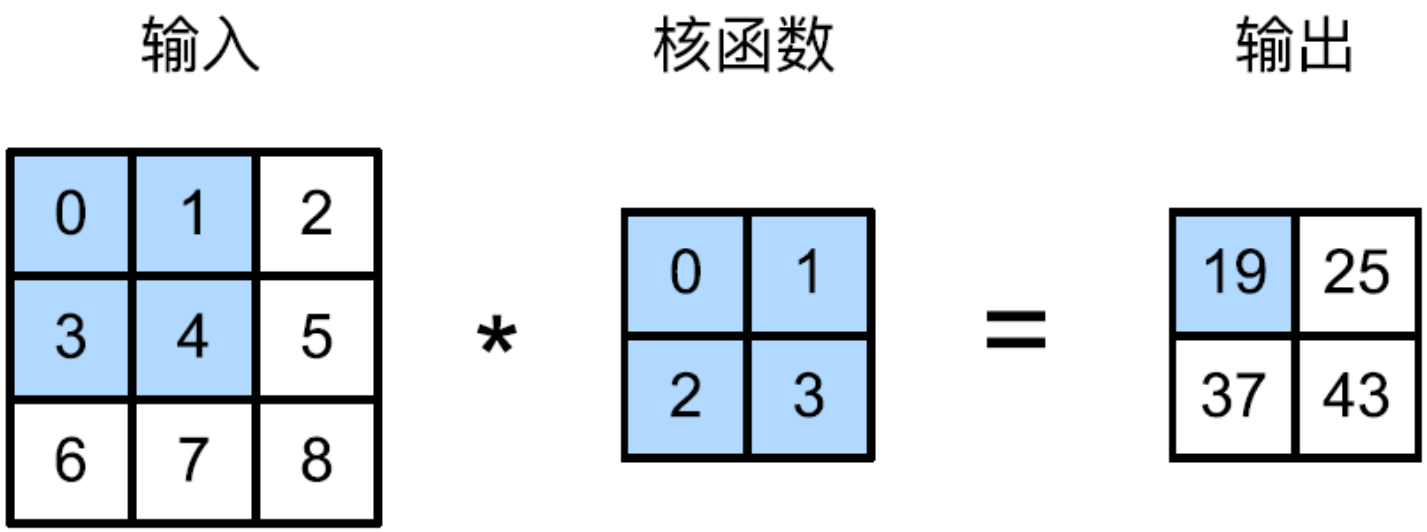


图: 图像卷积运算过程

阴影部分是第一个输出元素，以及用于计算输出的输入张量元素和核张量元素：

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

卷积层

在二维互相关运算中，卷积窗口从输入张量的左上角开始，从左到右、从上到下滑动。当卷积窗口滑动到新一个位置时，包含在该窗口中的部分张量与卷积核张量进行按元素相乘，得到的张量再求和得到一个单一的标量值，由此我们得出了这一位置的输出张量值。在如上例子中，输出张量的四个元素由二维互相关运算得到，这个输出高度为 2 、宽度为 2，如下所示：

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$

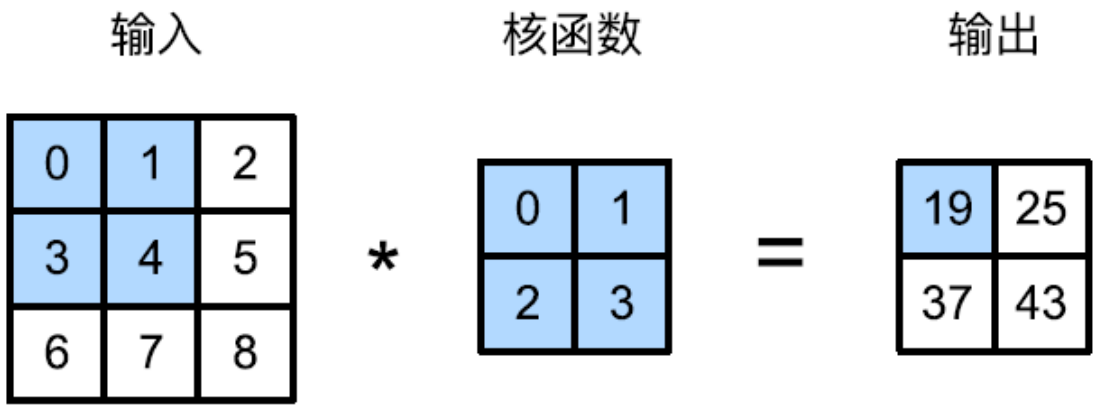


图: 图像卷积运算过程

卷积层

注意，输出大小略小于输入大小。这是因为卷积核的宽度和高度大于 1，而卷积核只与图像中每个大小完全适合的位置进行互相关运算。所以，输出大小等于输入大小 $n_h \times n_w$ 减去卷积核大小 $k_h \times k_w$，即：

$$(n_h - k_h + 1) \times (n_w - k_w + 1)$$

由于卷积网络主要应用在图像处理上，而图像为二维结构，为了更充分地利用图像的局部信息，通常将神经元组织为三维结构的神经层，其大小为高度 H 宽度 W 深度 C，由 C 个 $H \times W$ 大小的特征映射构成。

卷积层

卷积层

不失一般性，假设一个卷积层的结构如下：
(1) 输入特征映射组：$x \in R^{H \times W \times C}$ 为三维张量（Tensor），其中每个切片（Slice）矩阵 $X^c \in R^{H \times W}$ 为一个输入特征映射，$1 \leq c \leq C$；
(2) 输出特征映射组：$y \in R^{H' \times W' \times P}$ 为三维张量，其中每个切片矩阵 $Y^p \in R^{H' \times W'}$ 为一个输出特征映射，$1 \leq p \leq P$。

池化层

池化层

池化层（Pooling Layer）也叫下采样层（Subsampling Layer），其作用是进行特征选择，降低特征数量，从而减少参数数量。

卷积层虽然可以显著减少网络中连接的数量，但特征映射组中的神经元个数并没有显著减少。

如果后面接一个分类器，分类器的输入维数依然很高，很容易出现过拟合。

为了解决这个问题，可以在卷积层之后加上一个池化层，从而降低特征维数，避免过拟合。

池化层

假设池化层的输入特征映射组为 $x \in R^{H \times W \times C}$，对于其中每一个特征映射 $X^c \in R^{H \times W}, 1 \leq c \leq C$，将其划分为很多区域 $R^c_{m,n}$，$1 \leq m \leq M'$，$1 \leq n \leq N'$，这些区域可以重叠，也可以不重叠．池化（Pooling）是指对每个区域进行下采样（Down Sampling）得到一个值，作为这个区域的概括。

常用的池化函数有两种：

(1) 最大池化（Maximum Pooling 或 Max Pooling）

(2) 平均池化（Mean Pooling）

池化层：Max Pooling

Max Pooling

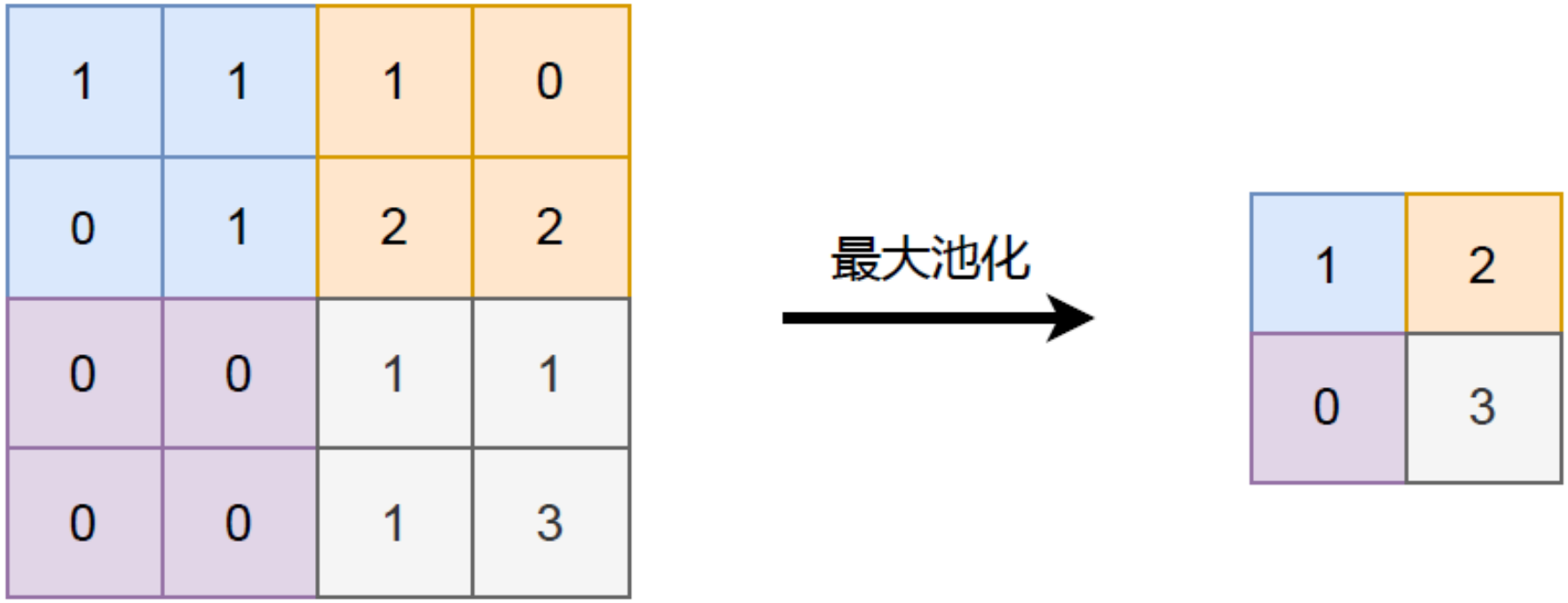
最大池化（Maximum Pooling 或 Max Pooling）：对于一个区域 $R^c_{m,n}$，选择这个区域内所有神经元的最大活性值作为这个区域的表示，即

$$y^c_{m,n} = \max_{i \in R^c_{h,w}} x_i$$

其中 x_i 为区域 R^c_k 内每个神经元的活性值。

池化层：Max Pooling

下图给出了采样最大池化进行子采样操作的示例。可以看出，池化层不但可以有效地减少神经元的数量，还可以使得网络对一些小的局部形态改变保持不变性，并拥有更大的感受野。



图：最大池化过程示例

池化层：Mean Pooling

Mean Pooling

平均池化（Mean Pooling）：一般是取区域内所有神经元活性值的平均值，即

$$y_{m,n}^{c} = \frac{1}{\left|R_{h,w}^{d}\right|} \sum_{i \in R_{h,w}^{d}} x_i.$$

对每一个输入特征映射 x_i 的 $H' \times W'$ 个区域进行子采样，得到池化层的输出特征映射 $Y^c = y_{h,w}^c, 1 \leq h \leq H, 1 \leq w \leq W$。

今天的目录

- **回顾感知机**
  - 结构
  - 实际中的分类问题
- **神经网络的表达**
- **神经元和激活函数**
  - Logistic 函数
  - Tanh 函数
  - ReLU 函数
- **前馈神经网络**
- **卷积神经网络 (CNN)**
  - 卷积层
  - 池化层
- **典型的卷积神经网络**

典型的卷积神经网络

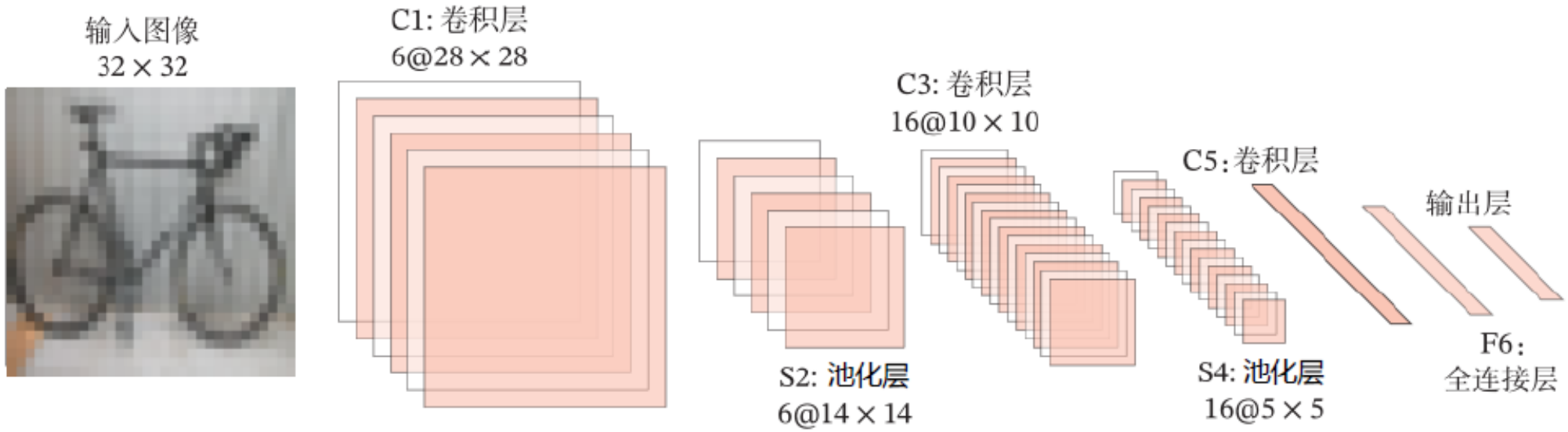
一个典型的卷积网络是由卷积层、池化层、全连接层交叉堆叠而成。
目前常用的卷积网络整体结构：一个卷积块为连续 M 个卷积层和 b 个池化层（M 通常设置为 $2 \sim 5$，b 为 0 或 1）。
一个卷积网络中可以堆叠 N 个连续的卷积块，然后在后面接着 K 个全连接层（N 的取值区间比较大，比如 $1 \sim 100$ 或者更大；K 一般为 $0 \sim 2$。

几种广泛使用的典型深层卷积神经网络：
$LeNet-5$、$AlexNet$、$Inception$ 和 $ResNet$。

LeNet – 5

$LeNet-5$（LeCun 等于 1998 提出]）虽然提出的时间比较早，但它是一个非常成功的神经网络模型。基于 LeNet-5 的手写数字识别系统在 20 世纪 90 年代被美国很多银行使用，用来识别支票上面的手写数字。

$LeNet-5$ 共有 7 层，接受输入图像大小为 $32 \times 32 = 1\ 024$，输出对应 10 个类别的得分。$LeNet-5$ 中的每一层结构如下：



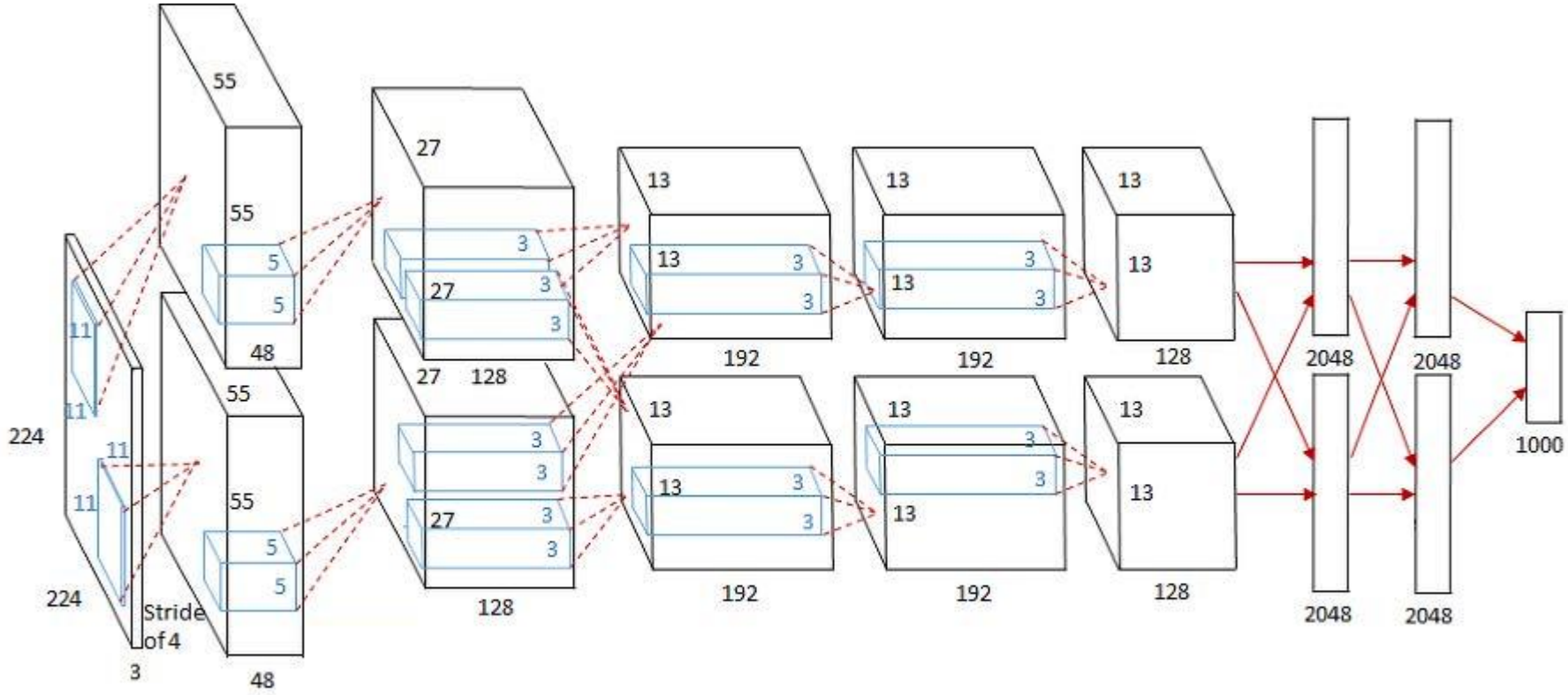
图：LeNet-5 网络结构

AlexNet

AlexNet（Krizhevsky 等于 2012 年提出）是第一个现代深度卷积网络模型。
它首次使用了很多现代深度卷积网络的技术方法，比如使用 *GPU* 进行并行训练，采用了 *ReLU* 作为非线性激活函数，使用 *Dropout* 防止过拟合，使用数据增强来提高模型准确率等。
AlexNet 赢得了 2012 年 ImageNet 图像分类竞赛的冠军。

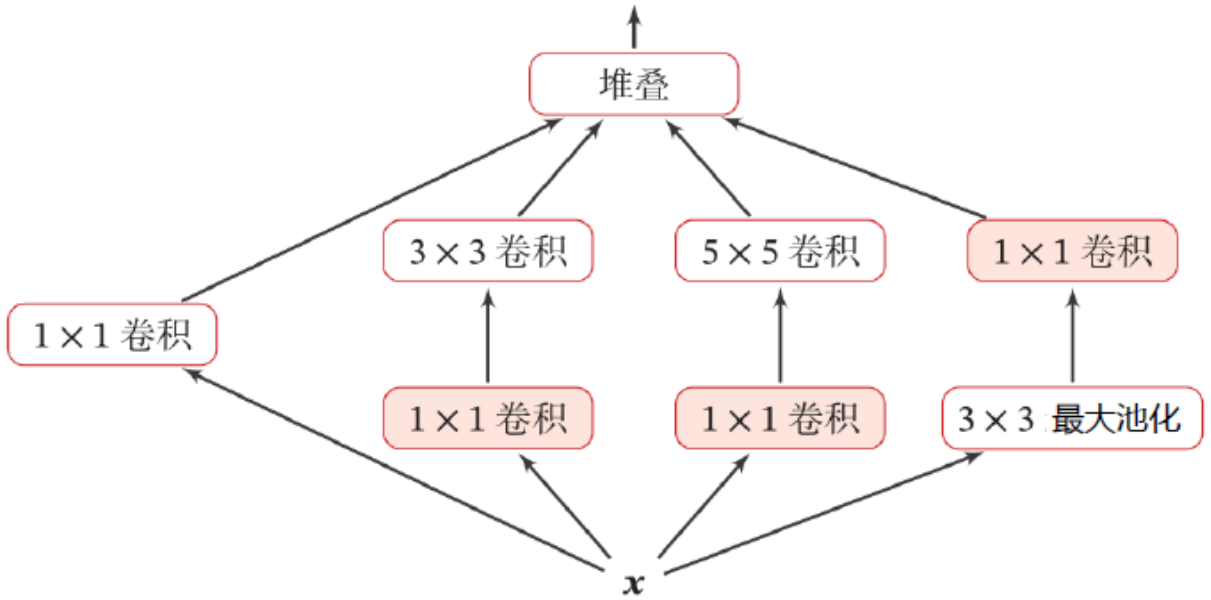
AlexNet

AlexNet 的结构如图所示，包括 5 个卷积层、3 个池化层和 3 个全连接层（其中最后一层是使用 *Softmax* 函数的输出层）。因为网络规模超出了当时的单个 *GPU* 的内存限制，*AlexNet* 将网络拆为两半，分别放在两个 *GPU* 上，*GPU* 间只在某些层（比如第 3 层）进行通信。



Inception

Inception 模块同时使用 1×1、3×3、5×5 等不同大小的卷积核，并将得到的特征映射在深度上拼接（堆叠）起来作为输出特征映射。下图给出了 v1 版本的 Inception 模块结构，采用了 4 组平行的特征抽取方式，分别为 1×1、3×3、5×5 的卷积和 3×3 的最大池化。同时，为了提高计算效率，减少参数数量，*Inception* 模块在进行 3×3、5×5 的卷积之前、3×3 的最大池化之后，进行一次 1×1 的卷积来减少特征映射的深度. 如果输入特征映射之间存在冗余信息，1×1 的卷积相当于先进行一次特征抽取。



图：*Inception* 模块

Inception: GoogLeNet

Inception 网络有多个版本，其中最早的 *Inceptionv1* 版本就是非常著名的 *GoogLeNet* (Szegedy 等于 2015 年提出)。*GoogLeNet* 赢得了 2014 年 *ImageNet* 图像分类竞赛的冠军。

图: GoogLeNet 网络结构

GoogLeNet 由 9 个 *Inceptionv1* 模块和 5 个池化层以及其他一些卷积层和全连接层构成，总共为 22 层网络。

ResNet

残差网络（Residual Network，ResNet）通过给非线性的卷积层增加直连边（Shortcut Connection）（也称为残差连接（Residual Connection））的方式来提高信息的传播效率。

假设在一个深度网络中，拟用一个非线性单元（可以为一层或多层的卷积层）$f(x;\theta)$ 去逼近一个目标函数为 $h(x)$。
如果将目标函数拆分成两部分：恒等函数（Identity Function）x 和残差函数（Residue Function）$h(x)-x$

根据通用近似定理，一个由神经网络构成的非线性单元有足够的能力来近似逼近原始目标函数或残差函数，但实际中后者更容易学习。因此，原来的优化问题可以转换为：让非线性单元 $f(x;\theta)$ 去近似残差函数 $h(x)-x$，并用 $f(x;\theta)+x$ 去逼近 $h(x)$。

ResNet

下图给出了一个典型的残差单元示例。残差单元由多个级联的（等宽）卷积层和一个跨层的直连边组成，再经过 *ReLU* 激活后得到输出。

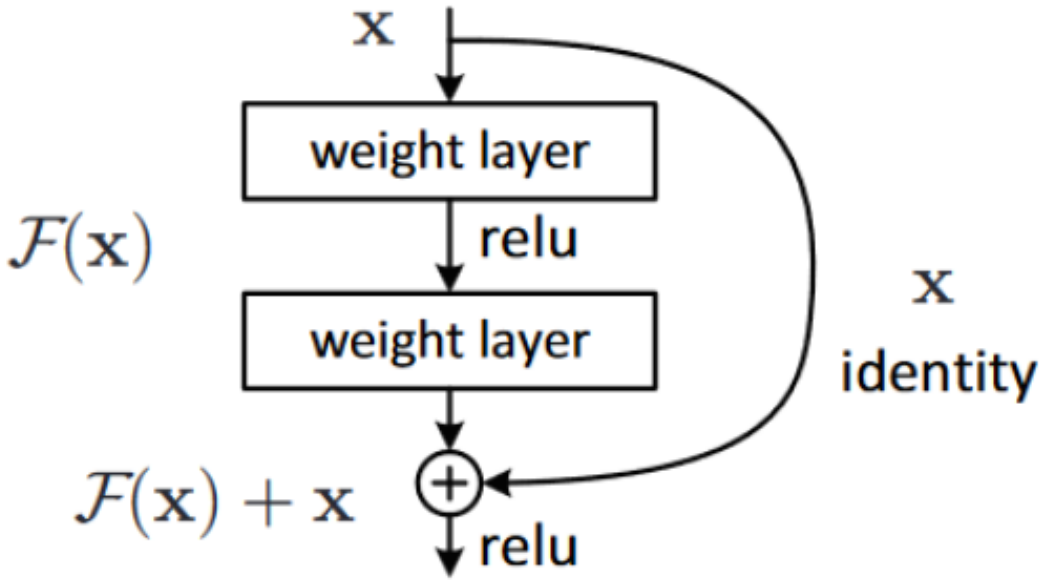


图: 一个残差单元结构

残差网络就是将很多个残差单元串联起来构成的一个非常深的网络。

总结

- **实际中神经网络是更常用的通用机器学习方法**
  - 神经网络的表达
- **激活函数**
  - Logistic, Tanh, ReLU
- **前馈神经网络**
- **卷积神经网络**
  - 卷积层、池化层
- **典型的卷积神经网络**
  - AlexNet, ResNet

朴素贝叶斯分类作业结果

Score	5	4	0	Avg.	Standard Dev.
# of Students	94	3	3	4.82	0.8690
Percentage	94%	3%	3%		