第五章：神经网络

主讲：连德富 特任教授 | 博士生导师

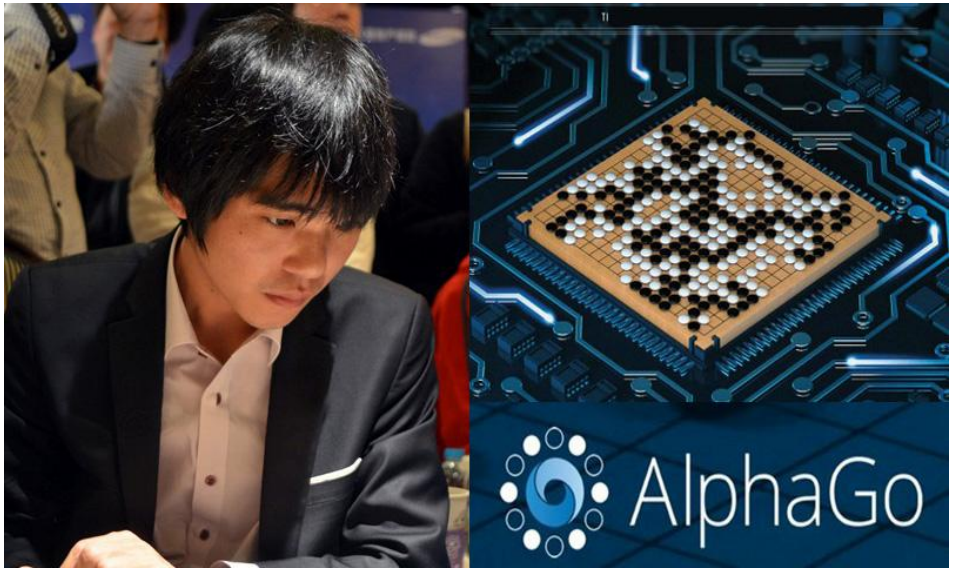
邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：http://staff.ustc.edu.cn/~liandefu

2016年AI技术重大突破

AlphaGo 以4:1比分赢得Lee Sedol



Versions	Hardware	Elo	Date	Results
AlphaGo Fan	176 GPUs, distributed	3,144	Oct-15	5:0 against Fan Hui
AlphaGo Lee	48 TPUs, distributed	3,739	Mar-16	4:1 against Lee Sedol
AlphaGo Master	4 TPUs, single machine	4,858	May-17	60:0 against professional players; Future of Go Summit
AlphaGo Zero (40 block)	4 TPUs, single machine	5,185	Oct-17	100:0 against AlphaGo Lee 89:11 against AlphaGo Master
AlphaZero (20 block)	4 TPUs, single machine	5,018	Dec-17	60:40 against AlphaGo Zero (20 block)

Rank	Name	♂♀	Flag	Elo
1	Shin Jinseo	♂		3800
2	Ke Jie	♂		3726
3	Park Junghwan	♂		3692
4	Gu Zihao	♂		3667
5	Lian Xiao	♂		3596
6	Fan Tingyu	♂		3589
7	Fan Yunruo	♂		3576
8	Jiang Weijie	♂		3574
9	Shin Minjun	♂		3572
10	Yang Dingxin	♂		3570
11	Xie Erhao	♂		3566
12	Mi Yuting	♂		3562
13	Xie Ke	♂		3557
14	Ding Hao	♂		3556
15	Tuo Jiaxi	♂		3555
16	Ichiriki Ryo	♂		3554
17	Chen Yaoye	♂		3550
18	Xu Jiayang	♂		3538
19	Iyama Yuta	♂		3538
20	Tong Mengcheng	♂		3530
21	Byun Sangil	♂		3527
22	Lee Donghoon	♂		3525
23	Tao Xinran	♂		3522
24	Zhao Chenyu	♂		3519
25	Li Weiqing	♂		3518
26	Kang Dongyun	♂		3504
27	Liao Yuanhe	♂		3501
28	Shi Yue	♂		3499

AlphaGo中的机器学习

- 策略网络

监督学习

- 预测人类如何下下一步棋

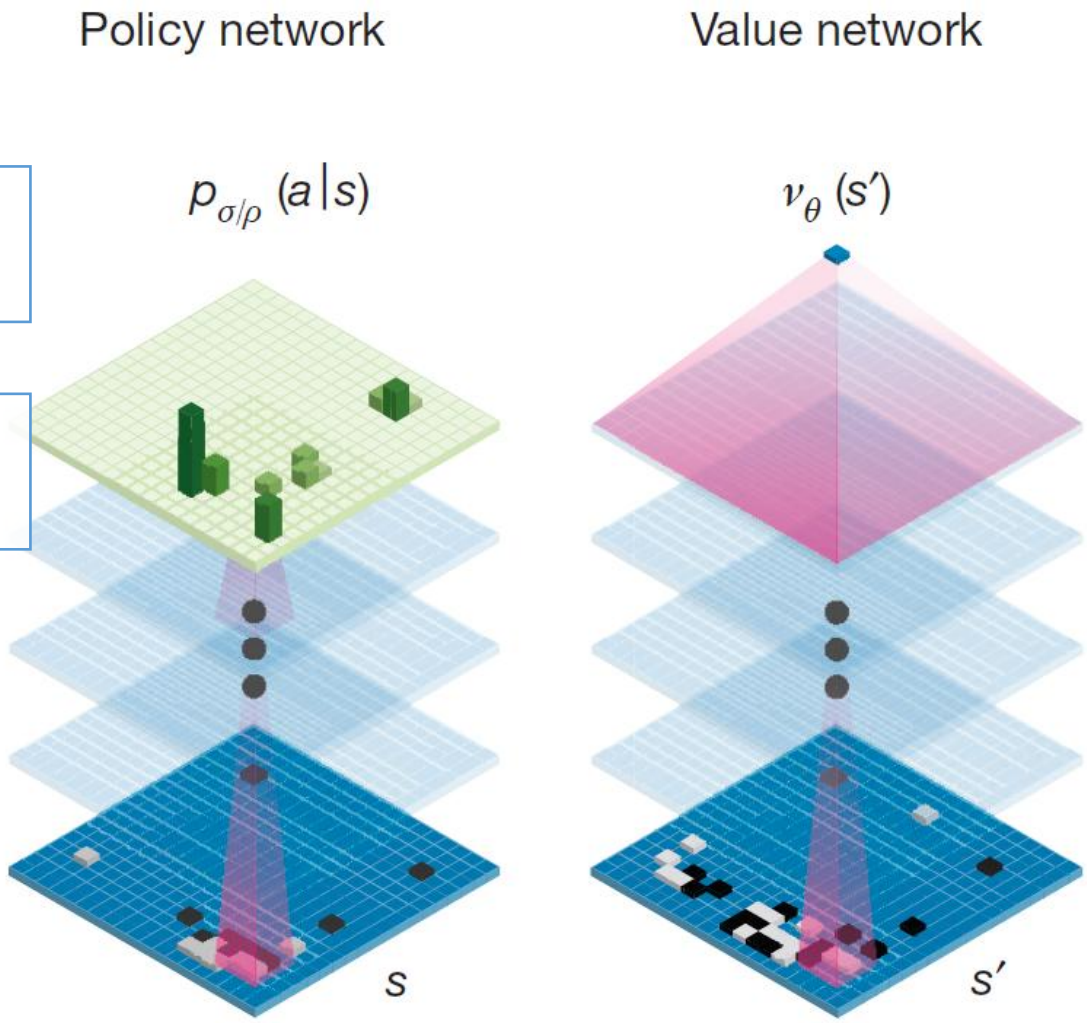
强化学习

- 学习如何下下一步棋以最大化赢率

- 值网络

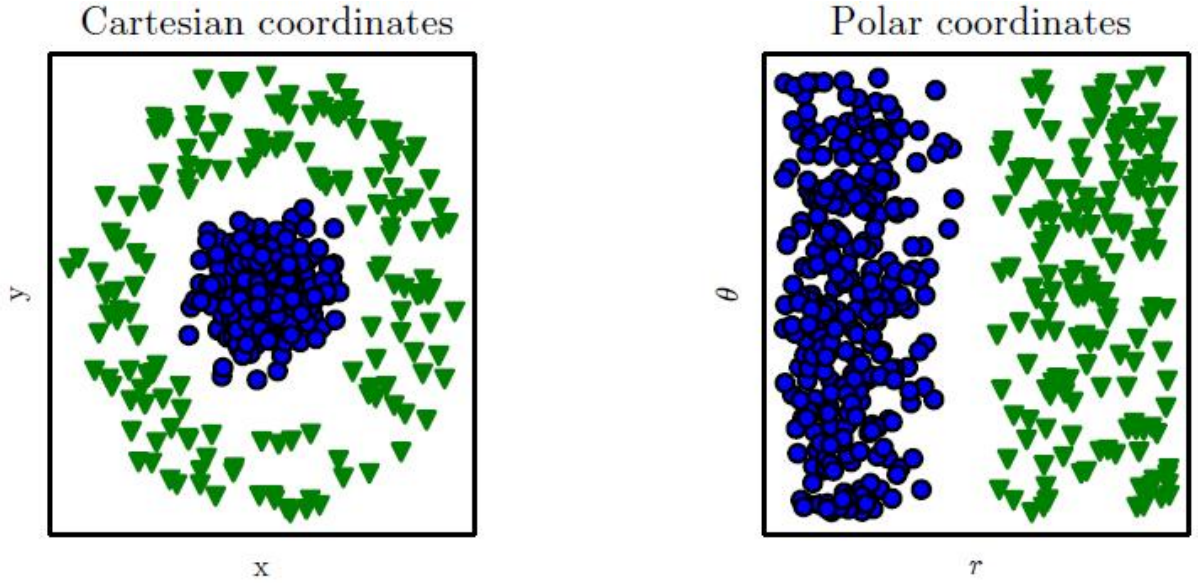
估计给定棋局的赢率

通过（深度）神经网络来实现



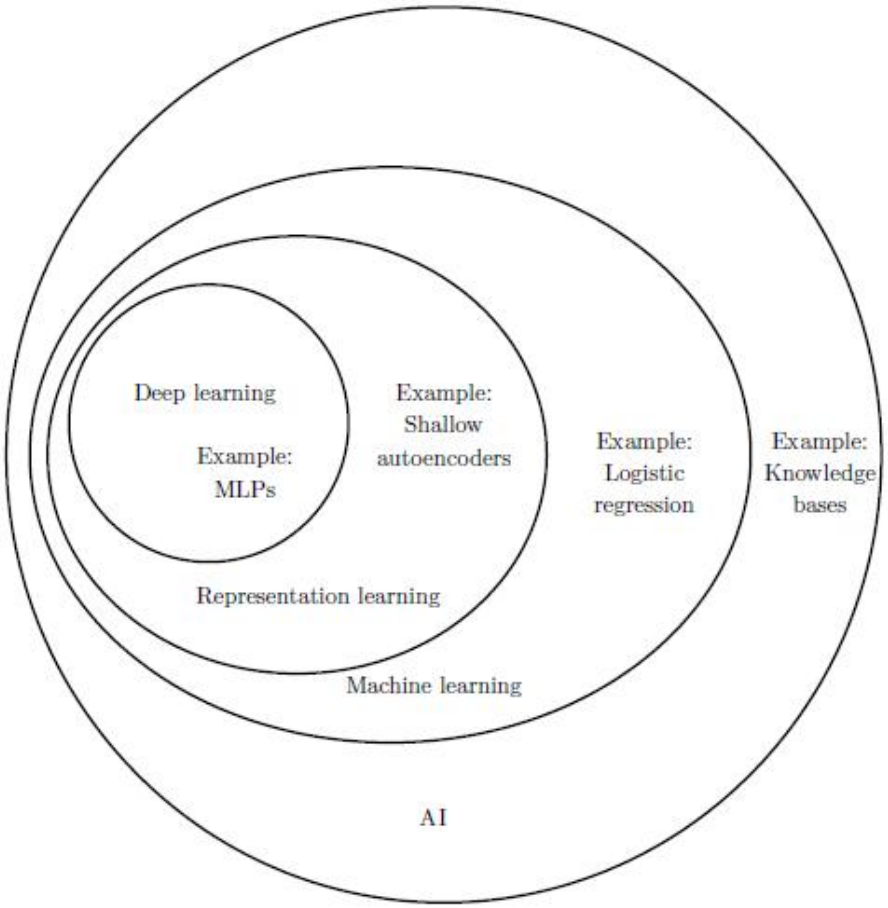
神经网络是什么？

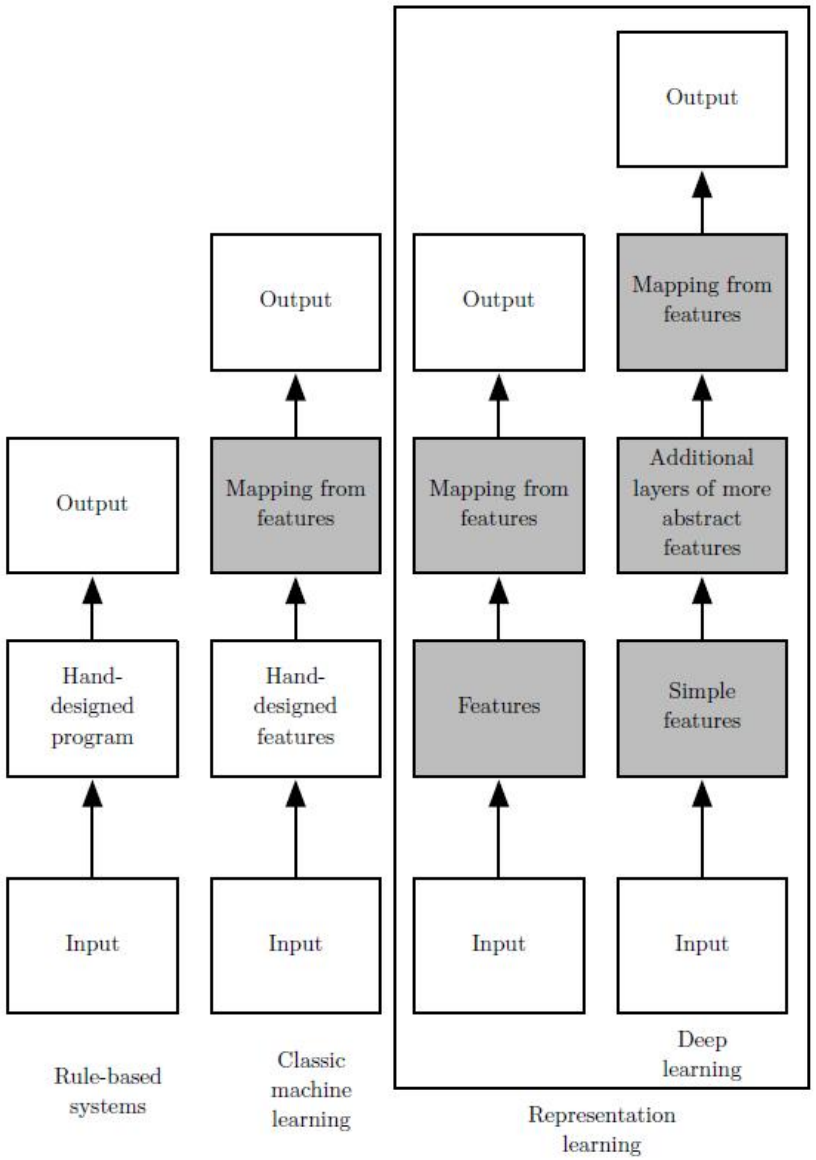
- 线性模型$y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$：无法建模任何两个输入变量之间的相互作用
- 扩展线性模型来表示 $\boldsymbol{x}$ 的非线性函数，$y = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{w}$
- 简单机器学习方法常通过特征工程设计$\boldsymbol{\phi}(\boldsymbol{x})$



- （深度）神经网络直接学习特征表示，即 $y = f(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{w}) = \boldsymbol{\phi}(\boldsymbol{x};\boldsymbol{\theta})^{T}\boldsymbol{w}$

神经网络是什么？


Deep learning
Example: MLPs
Example: Shallow autoencoders
Representation learning
Example: Logistic regression
Machine learning
Example: Knowledge bases
AI


Output
Hand-designed program
Input
Rule-based systems
Output
Mapping from features
Hand-designed features
Input
Classic machine learning
Output
Mapping from features
Features
Input
Output
Mapping from features
Additional layers of more abstract features
Simple features
Input
Deep learning
Representation learning

神经网络发展史—第一阶段

- 1943年, McCulloch和Pitts 提出第一个神经元数学模型, 即M-P模型, 并从原理上证明了人工神经网络能够计算任何算数和逻辑函数
- 1949年, Hebb 发表《The Organization of Behavior》一书, 提出生物神经元学习的机理, 即Hebb学习规则
- 1958年, Rosenblatt 提出感知机网络（Perceptron）模型和其学习规则
- 1960年, Widrow和Hoff提出自适应线性神经元（Adaline）模型和最小均方学习算法
- 1969年, Minsky和Papert 发表《Perceptrons》一书, 指出单层神经网路不能解决非线性问题, 多层网络的训练算法尚无希望. 这个论断导致神经网络进入低谷

神经网络发展史—第二阶段

- 1982年, 物理学家Hopfield提出了一种具有联想记忆、优化计算能力的递归网络模型, 即Hopfield 网络
- 1986年, Rumelhart 等编辑的著作《Parallel Distributed Proceesing: Explorations in the Microstructures of Cognition》报告了反向传播算法
- 1987年, IEEE 在美国加州圣地亚哥召开第一届神经网络国际会议（ICNN）
- 90年代初, 伴随统计学习理论和SVM的兴起, 神经网络由于理论不够清楚, 试错性强, 难以训练, 再次进入低谷

神经网络发展史—第三阶段

- 2006年, Hinton提出了深度信念网络(DBN), 通过“预训练+微调”使得深度模型的最优化变得相对容易
- 2012年, Hinton 组参加ImageNet 竞赛, 使用 CNN 模型以超过第二名10个百分点的成绩夺得当年竞赛的冠军
- 伴随云计算、大数据时代的到来，计算能力的大幅提升，使得深度学习模型在计算机视觉、自然语言处理、语音识别等众多领域都取得了较大的成功

神经元模型

- 神经网络的定义

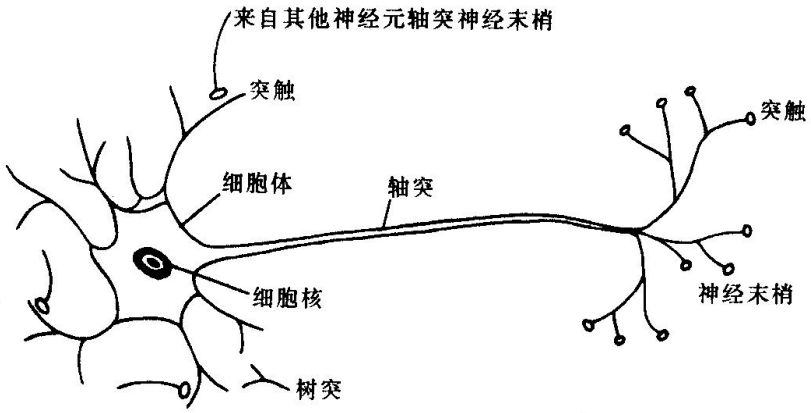
> “神经网络是由具有适应性的简单单元组成的广泛并行互联的网络, 它的组织能够模拟生物神经系统对真实世界物体所作出的反应”

[Kohonen, 1988]

- 机器学习中的神经网络通常是指“神经网络学习” 或者机器学习与神经网络两个学科的交叉部分
- 神经元模型即上述定义中的“简单单元”是神经网络的基本成分

生物神经网络

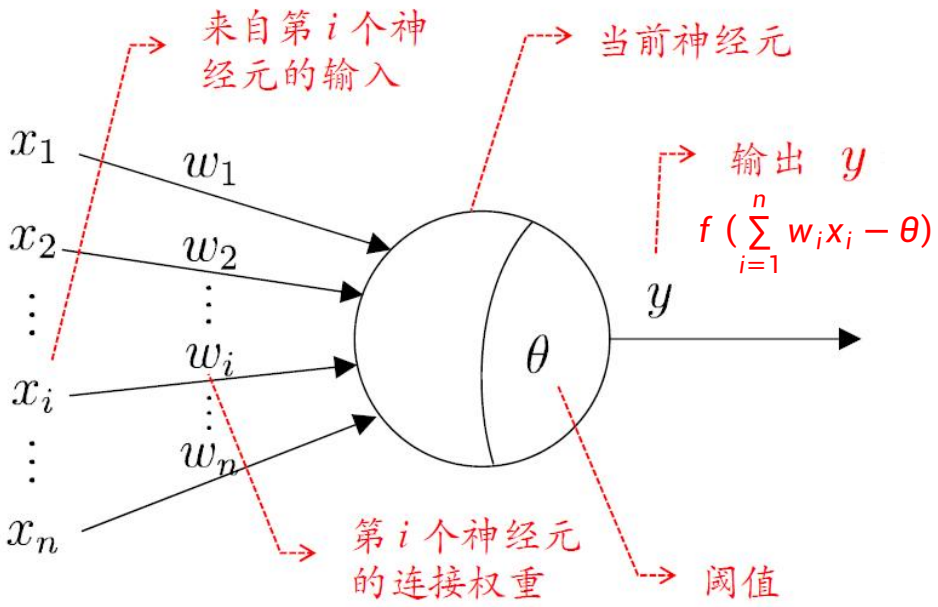
每个神经元与其他神经元相连, 当它“兴奋”时, 就会向相连的神经元发送化学物质, 从而改变这些神经元内的电位; 如果某神经元的电位超过一个“阈值”, 那么它就会被激活, 即“兴奋”起来, 向其它神经元发送化学物质



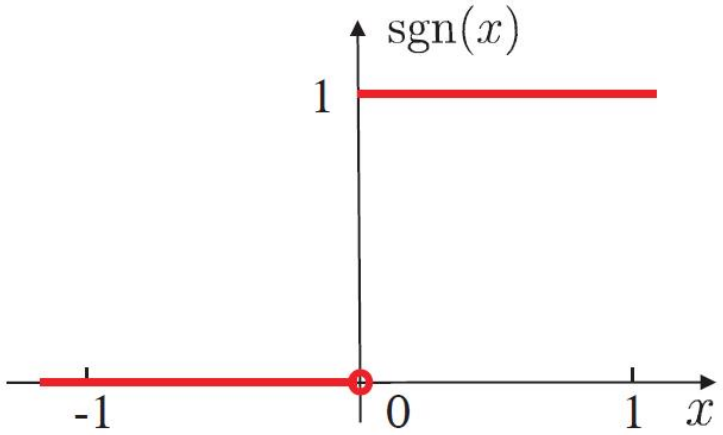
神经元模型

M-P 神经元模型 [McCulloch and Pitts, 1943]

- 输入：来自其他n个神经云传递过来的输入信号
- 处理：输入信号通过带权重的连接进行传递, 神经元接受到总输入值将与神经元的阈值进行比较
- 输出：通过激活函数的处理以得到输出

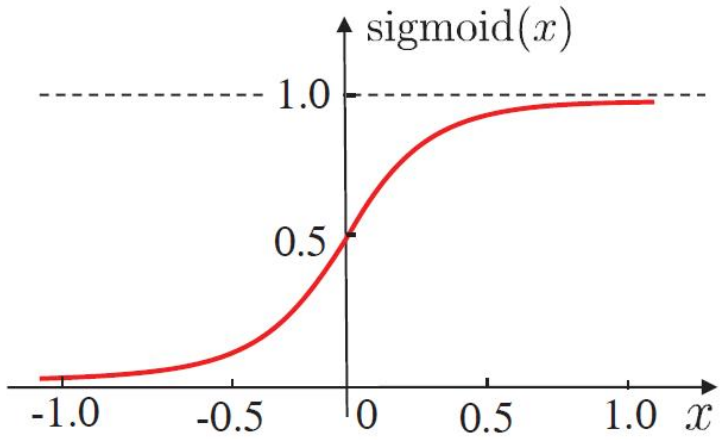


神经元模型—激活函数



$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

(a) 阶跃函数



$$\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

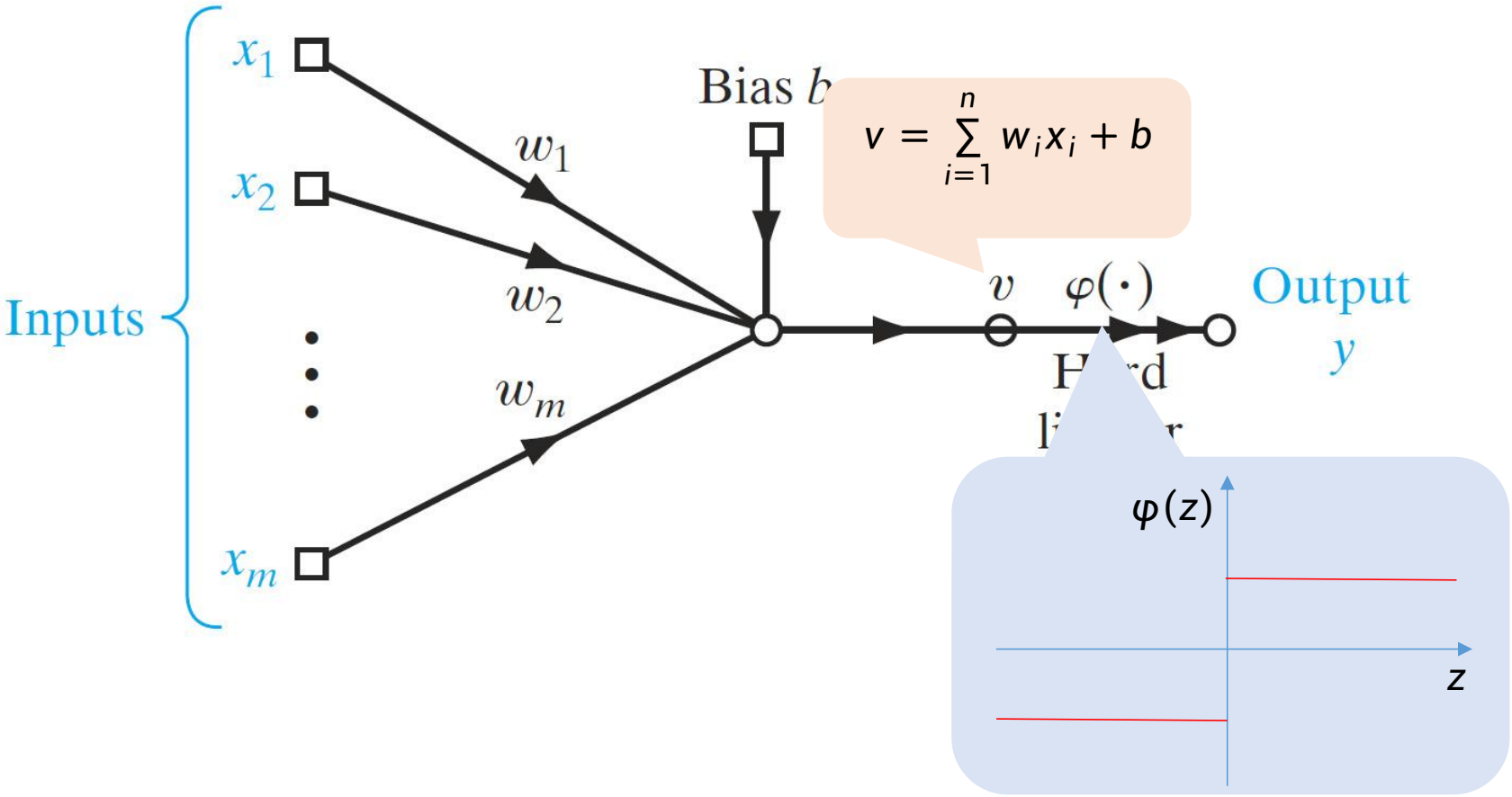
(b) Sigmoid 函数

理想激活函数是阶跃函数
0表示抑制神经元；1表示激活神经元

阶跃函数具有不连续、不光滑等不好的性质, 常用的是 Sigmoid 函数

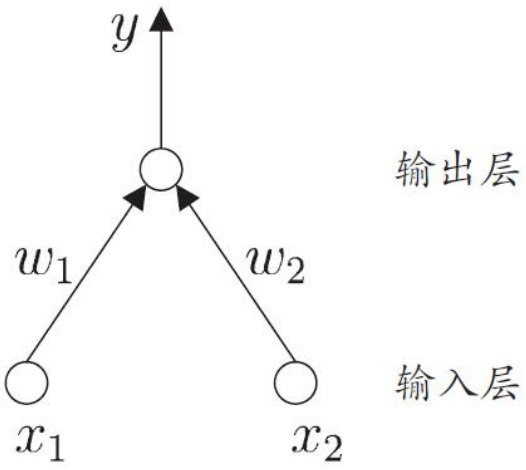
感知机

- 感知机由两层神经元组成, 输入层接受外界输入信号传递给输出层, 输出层是M-P神经元（阈值逻辑单元）



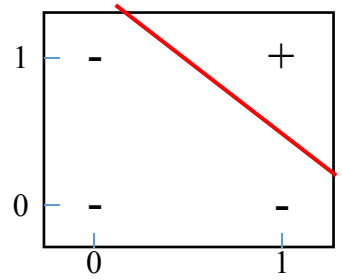
感知机

- 感知机能够容易地实现逻辑与、或、非运算



$x_1 \wedge x_2$：

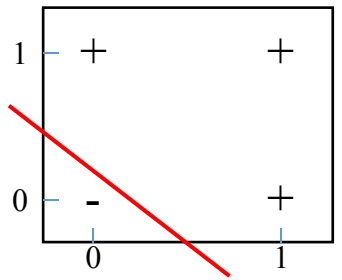
$y = f(1 \cdot x_1 + 1 \cdot x_2 - 1.5)$



与

$x_1 \vee x_2$：

$y = f(1 \cdot x_1 + 1 \cdot x_2 - 0.5)$



或

感知机学习

- 给定训练集, 权重$w_i(i = 1,2,\cdots,n)$与阈值θ可以通过学习得到

- 感知机学习规则

对训练样例$(\boldsymbol{x}, y)$, 若当前感知机的输出为$\hat{y}$, 则感知机权重调整规则为:

$$w_i \leftarrow w_i + \eta(y - \hat{y})x_i$$

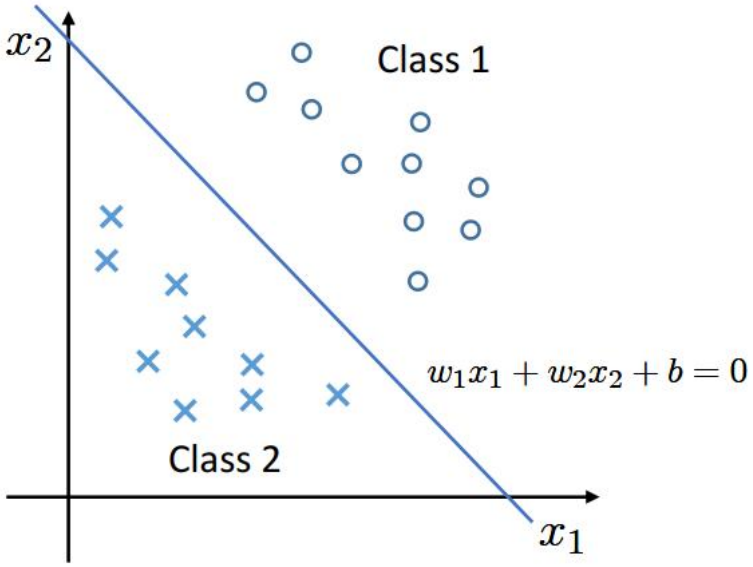
学习率

对应如下更新规则

若对训练样本$(\boldsymbol{x}, y)$预测正确，则感知机不发生变化;
若预测值更大，降低激活输入的权重;
若预测值更小，增加激活输入的权重

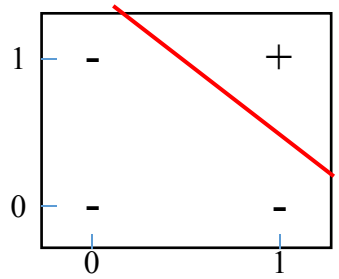
感知机学习

- 若两类模式线性可分, 则感知机的学习过程一定会收敛；否感知机的学习过程将会发生震荡 [Minsky and Papert, 1969]
- 单层感知机的学习能力非常有限, 只能解决线性可分问题

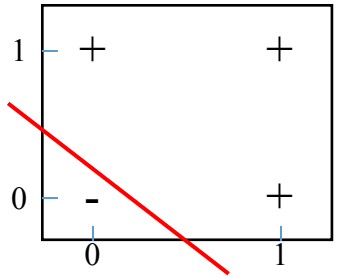


感知机学习

- 与、或、非问题是线性可分的, 因此感知机学习过程能够求得适当的权值向量

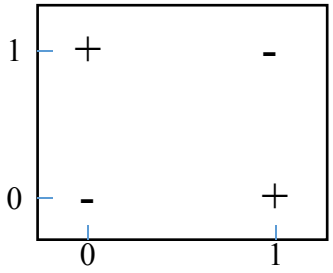


与



或

- 异或问题不是线性可分的, 感知机学习不能求得合适解



异或 $x_1 \oplus x_2$

对于非线性可分问题, 如何求解？ 多层感知机

多层感知机

- 输出层与输入层之间的一层神经元, 被称之为隐层或隐含层, 隐含层和输出层神经元都是具有激活函数的功能神经元

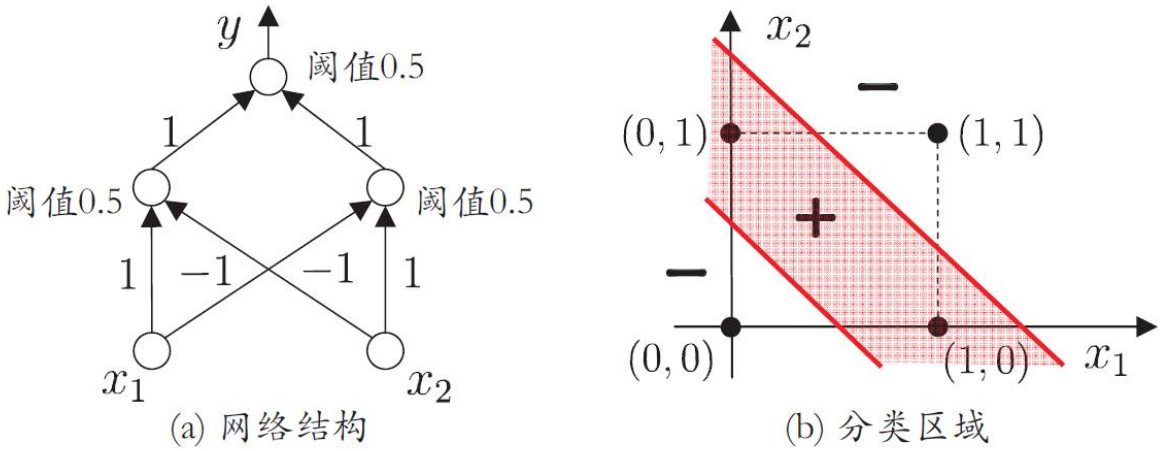
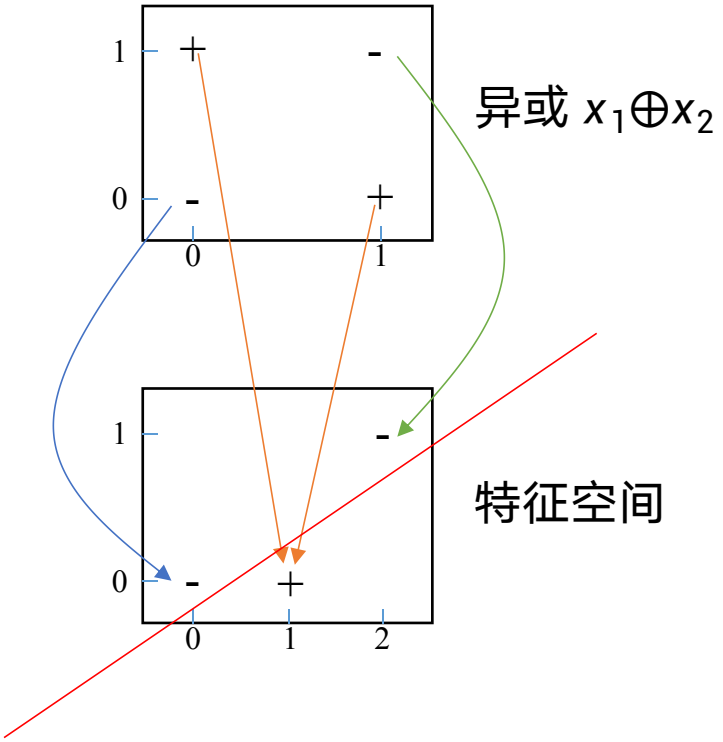


图 5.5 能解决异或问题的两层感知机

多层感知机

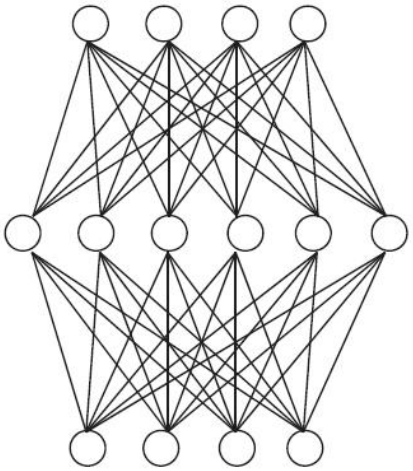
- $\boldsymbol{h} = max\left(0, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right)$

- $y = [1, -2]\,\boldsymbol{h} - 0.5$

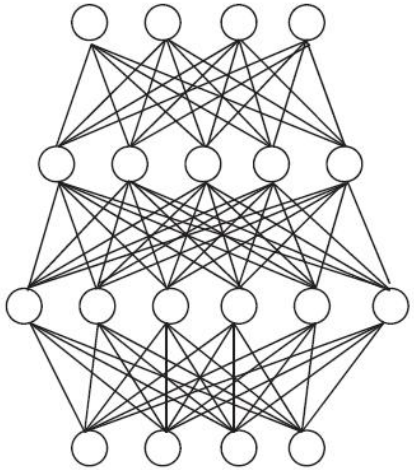


多层前馈神经网络

- 定义：每层神经元与下一层神经元全互联，神经元之间不存在同层连接也不存在跨层连接
- 前馈：输入层接受外界输入，隐含层与输出层神经元对信号进行加工，最终结果由输出层神经元输出
- 学习：根据训练数据来调整神经元之间的“连接权”以及每个功能神经元的“阈值”
- 多层网络：包含隐层的网络



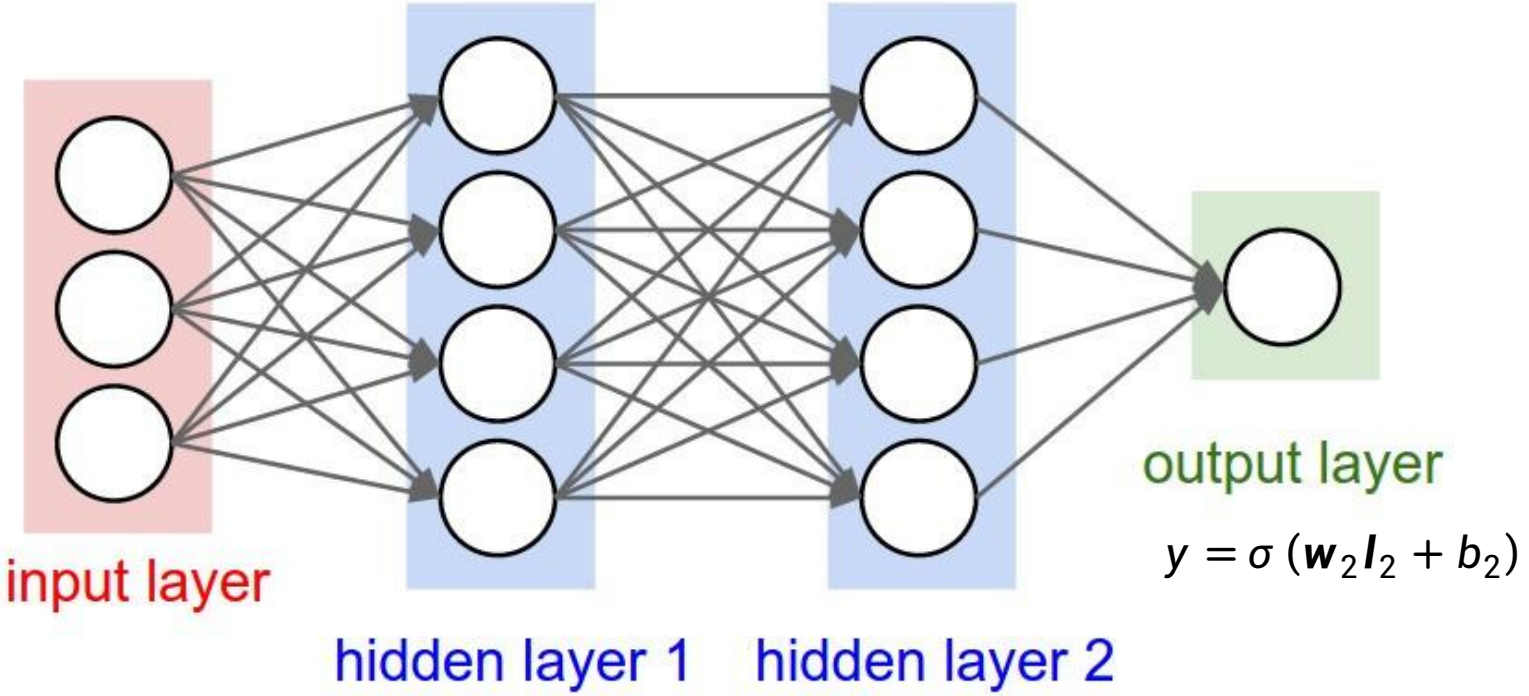
(a) 单隐层前馈网络



(b) 双隐层前馈网络

多层前馈神经网络—表示能力

- 只需要一个包含足够多神经元且具有任何“挤压”性质的激活函数的隐层, 多层前馈神经网络就能以任意精度逼近有界闭集上的任意连续函数



多层前馈神经网络

- 如何学习多层前馈神经网络的参数呢？

误差逆传播算法

误差逆传播算法（Error BackPropagation, 简称BP）是最成功的训练多层前馈神经网络的学习算法

误差逆传播算法

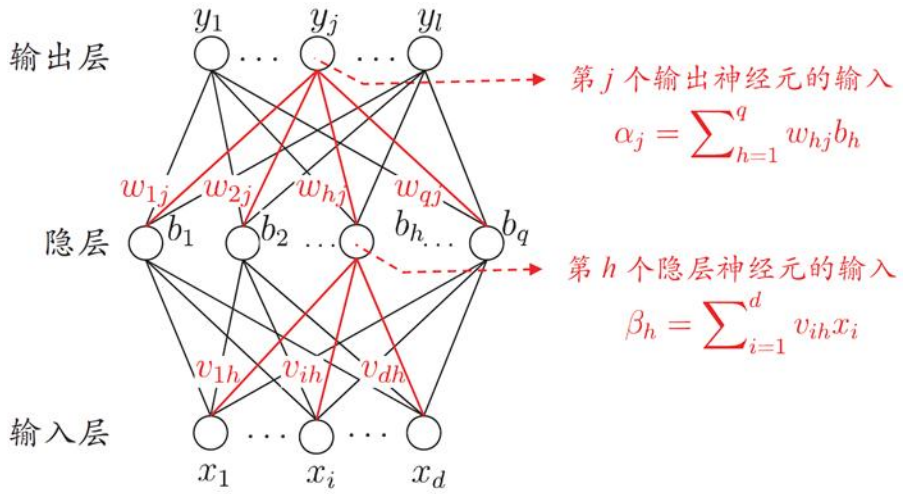
输出l维实值向量$\boldsymbol{y}$

θ_j： 输出层第j个神经元阈值

w_{hj}： 隐层与输出层神经元之间的连接权重

γ_h： 隐含层第h个神经元阈值

v_{ih}： 输入层与隐层神经元之间的连接权重



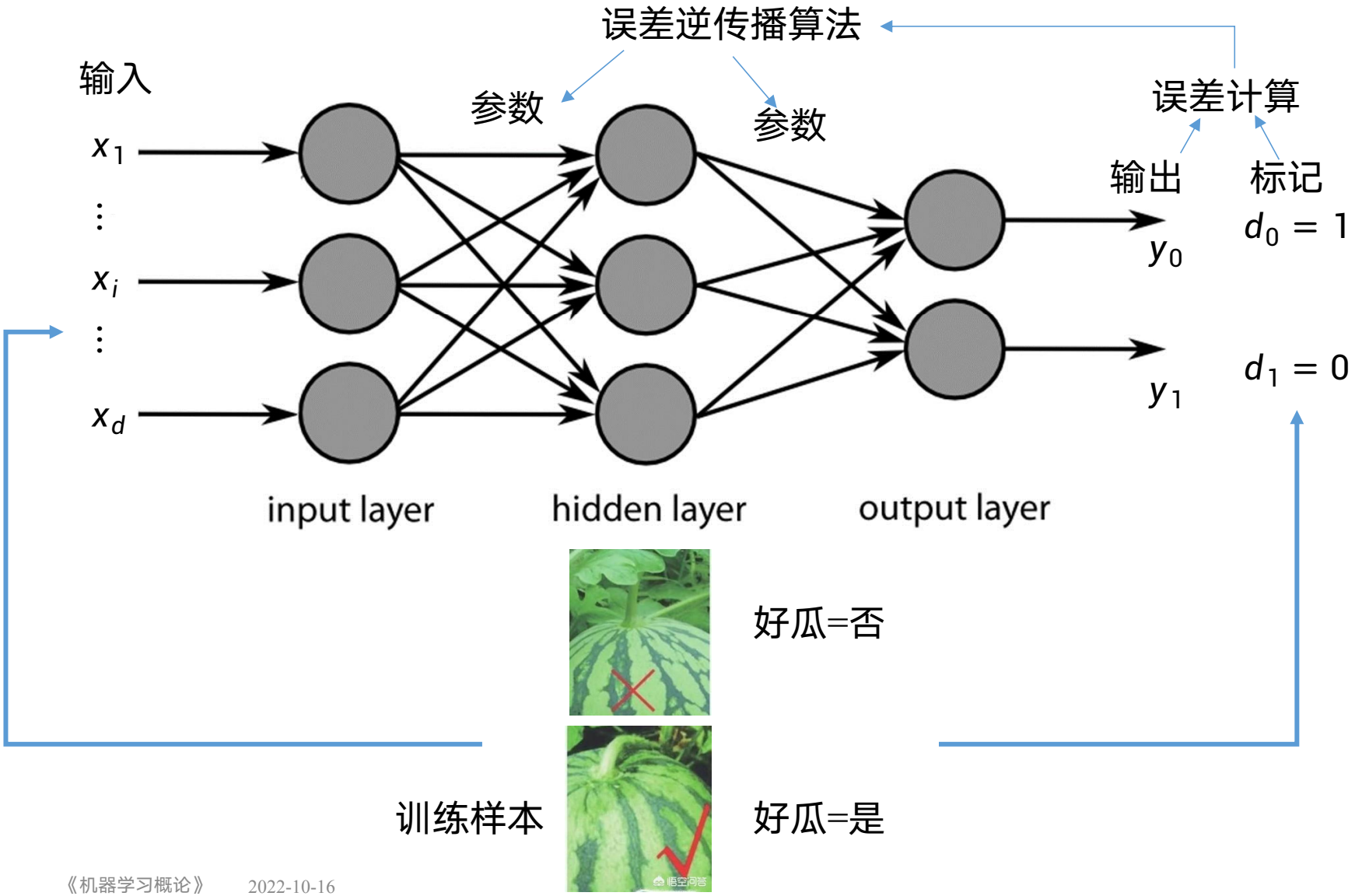
网络中需要 $(d+l+1)q+l$ 个参数需要优化

输入示例$\boldsymbol{x}$由d个属性描述

BP是一个迭代学习算法, 在迭代的每一轮中采用广义的感知机学习规则对参数进行更新估计, 任意的参数v的更新估计式为

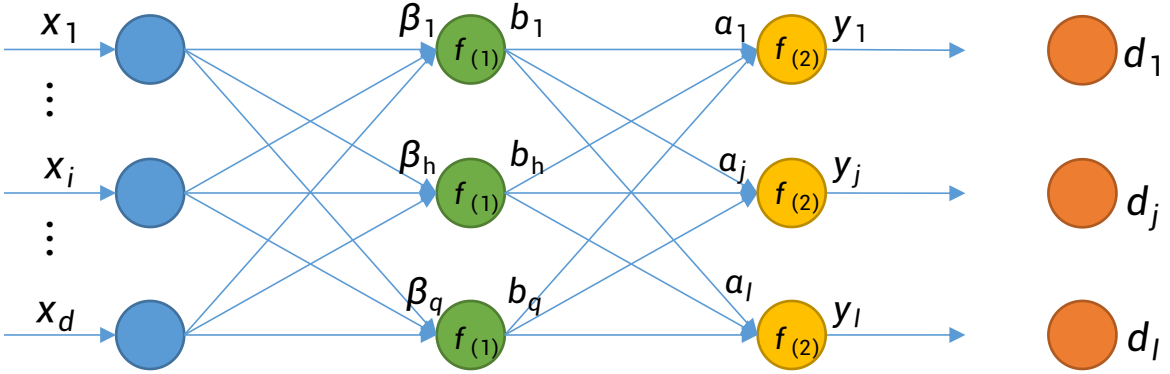
$$v \leftarrow v + \Delta v$$

误差逆传播算法


误差逆传播算法
输入
参数
参数
误差计算
输出
标记
x_1
x_i
x_d
y_0
y_1
$d_0 = 1$
$d_1 = 0$
input layer
hidden layer
output layer
好瓜=否
训练样本
好瓜=是

误差逆传播算法—前向

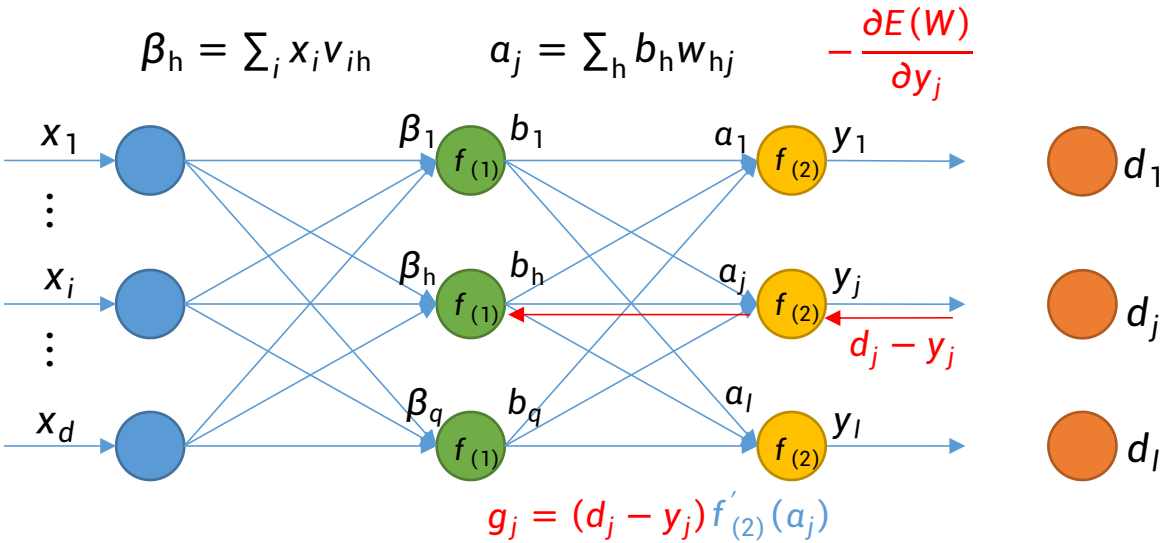
$$\beta_h = \sum_i x_i v_{ih} \qquad a_j = \sum_h b_h w_{hj}$$



前向预测

$$\boldsymbol{x} \xrightarrow[f^{(1)}(\beta_h)]{\beta_h = \sum_i x_i v_{ih}} b_h \xrightarrow[f^{(2)}(a_j)]{a_j = \sum_h b_h w_{hj}} y_j$$

误差逆传播算法—后向

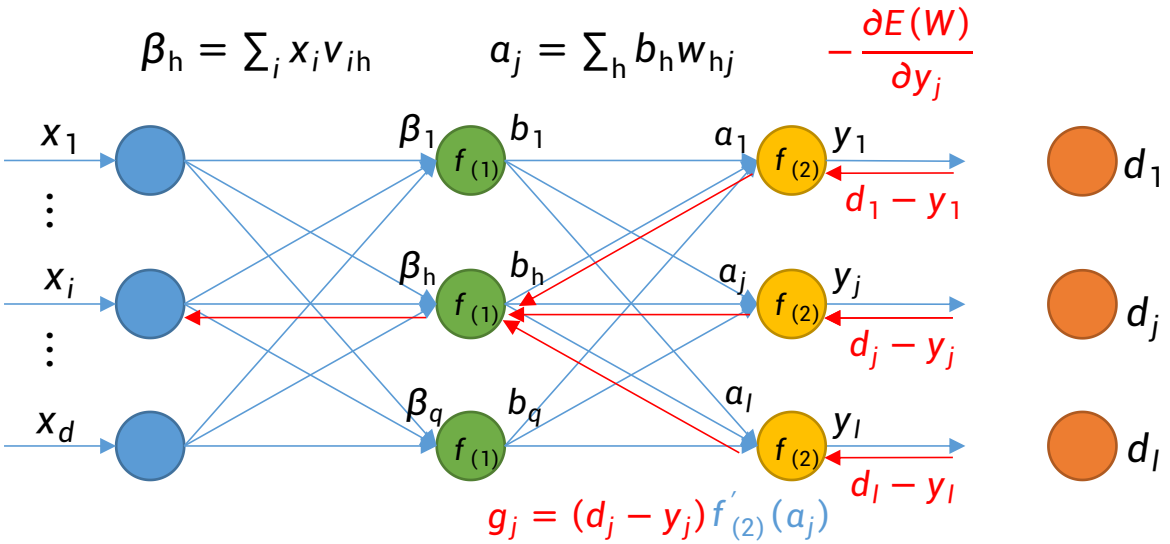


后向传播

$$w_{hj} = w_{hj} + \Delta w_{hj} \quad \Delta w_{hj} = \eta \text{Error}_j \text{Output}_h = \eta g_j b_h \quad E(\boldsymbol{W}) = \frac{1}{2}\sum_{j=1}^{l}(y_j - d_j)^2$$

$$\Delta w_{hj} = -\eta \frac{\partial E(W)}{\partial w_{hj}} = -\eta \frac{\partial E(W)}{\partial y_j}\frac{\partial y_j}{\partial a_j}\frac{\partial a_j}{\partial w_{hj}} = \eta(d_j - y_j) f'_{(2)}(a_j) b_h = \eta g_j b_h$$

误差逆传播算法—后向



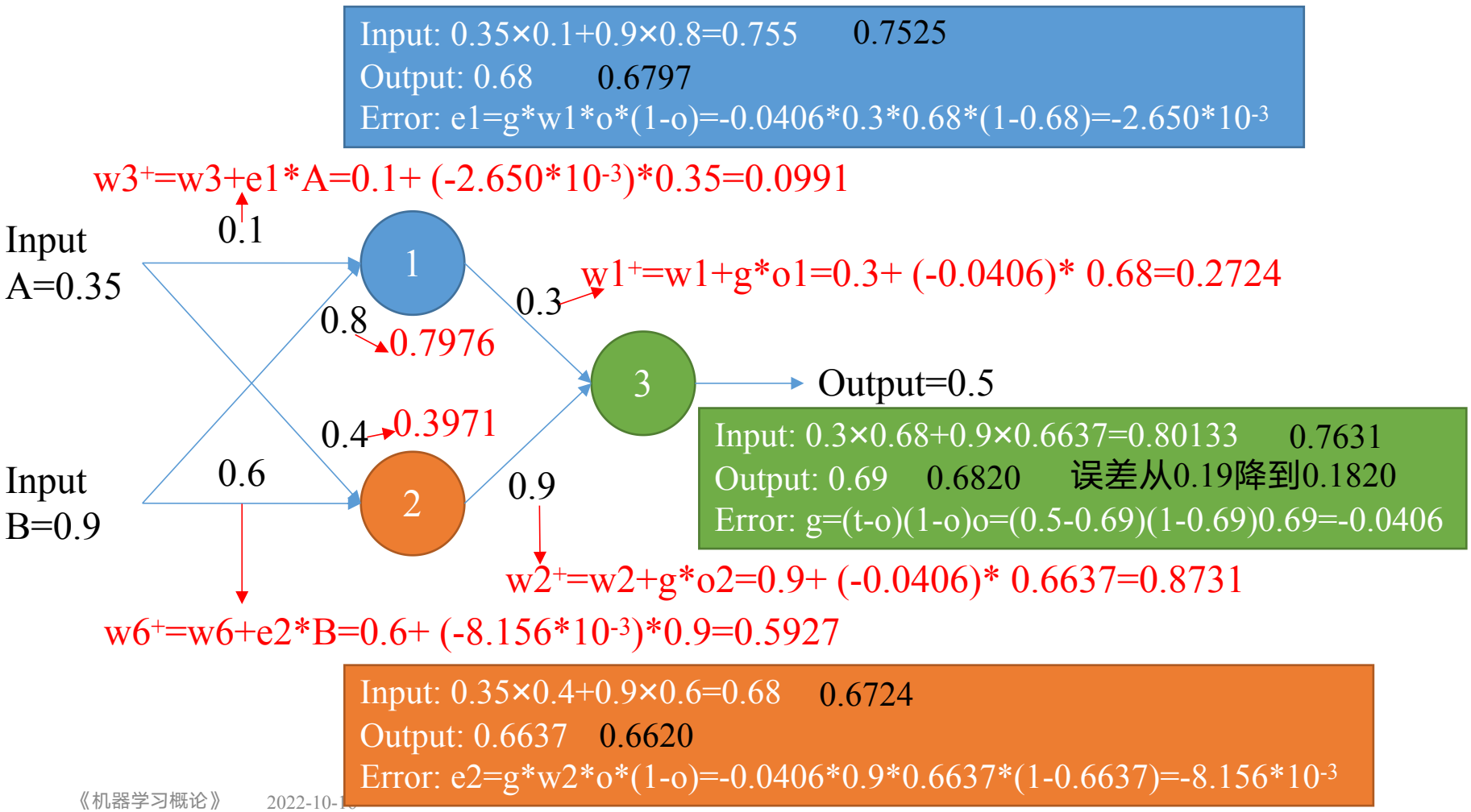
后向传播

$$v_{ih} = v_{ih} + \Delta v_{ih} \qquad \Delta v_{ih} = \eta \text{Error}_h \text{Output}_i = \eta e_h x_i \qquad E(\boldsymbol{W}) = \frac{1}{2}\sum_{j=1}^{l}(y_j - d_j)^2$$

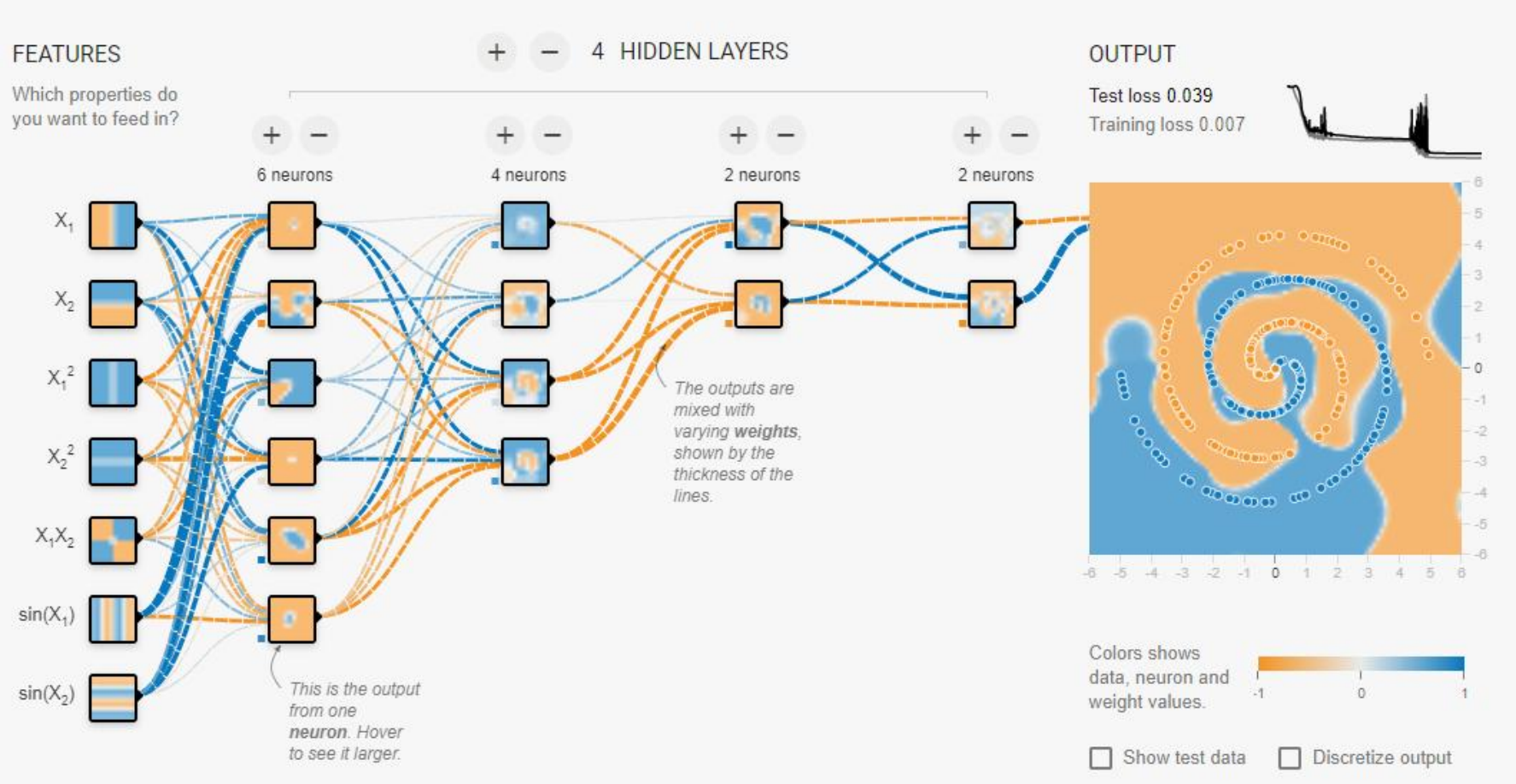
$$\Delta v_{ih} = -\eta \frac{\partial E(W)}{\partial v_{ih}} = -\eta \frac{\partial E(W)}{\partial b_h}\frac{\partial b_h}{\partial \beta_h}\frac{\partial \beta_h}{\partial v_{ih}} = \eta \sum_j g_j w_{hj} f'_{(1)}(\beta_h) x_i = \eta e_h x_i$$

BP算法：简单例子

- 考虑如下简单网络　假设激活函数为Sigmoid函数



BP演示



http://playground.tensorflow.org/

误差逆传播算法

输入: 训练集 $D = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^{m}$;
学习率 η.

过程:

在$(0,1)$范围内随机初始化网络中所有连接权和阈值
repeat
  for all $(\boldsymbol{x}_k, \boldsymbol{y}_k) \in D$ **do**
    根据当前参数和式(5.3) 计算当前样本的输出 $\hat{\boldsymbol{y}}_k$;
    根据式(5.10) 计算输出层神经元的梯度项 g_j;
    根据式(5.15) 计算隐层神经元的梯度项 e_h;
    根据式(5.11)-(5.14) 更新连接权 w_{hj}, v_{ih} 与阈值 θ_j, γ_h
  end for
until 达到停止条件

输出: 连接权与阈值确定的多层前馈神经网络

图 5.8　误差逆传播算法

误差逆传播算法

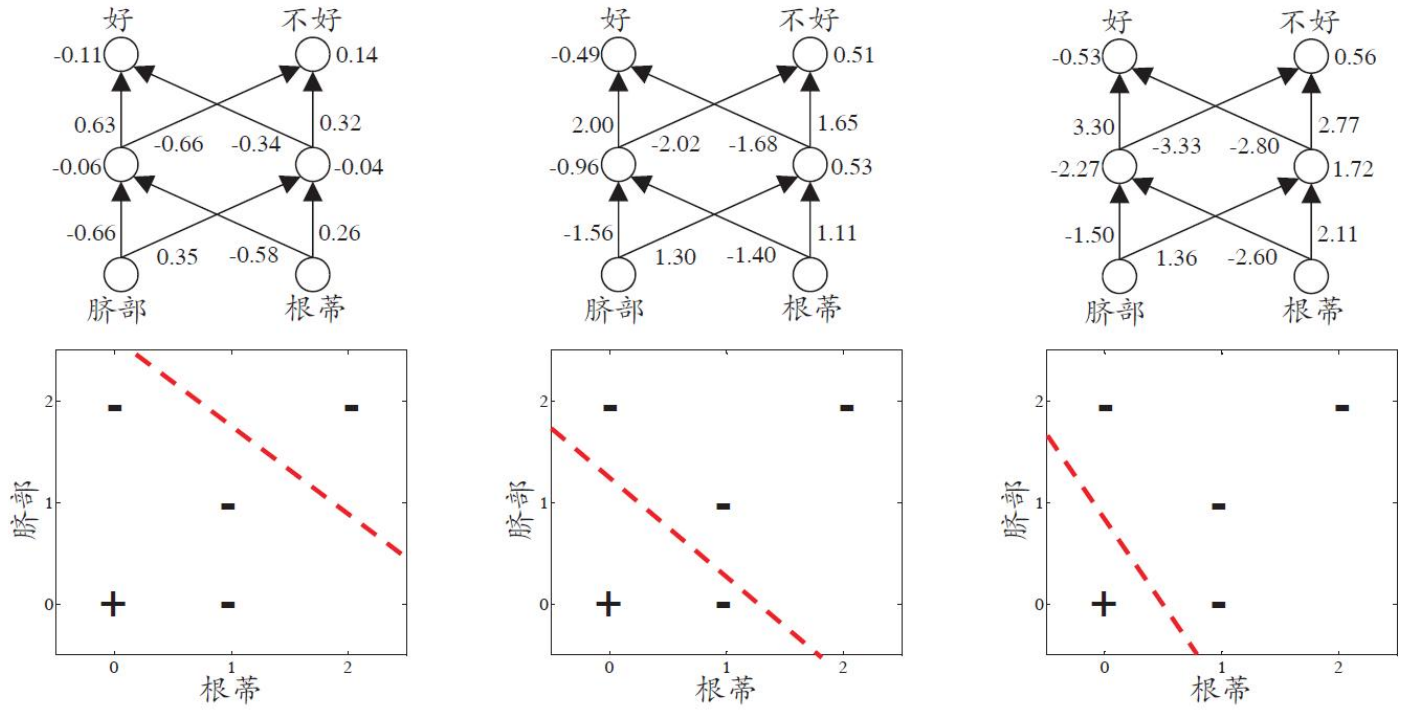


图 5.9 在 2 个属性、5 个样本的西瓜数据上, BP网络参数更新和分类边界的变化情况

误差逆传播算法

- 标准 BP 算法
  - 每次针对单个训练样例更新权值与阈值
  - 参数更新频繁, 不同样例可能抵消, 需要多次迭代.

也称为随机梯度下降

- 累计 BP 算法
  - 优化的目标是最小化整个训练集上的累计误差

$$E = \frac{1}{m}\sum_{k=1}^{m} E_k$$

  - 读取整个训练集一遍才对参数进行更新, 参数更新频率较低.

但在很多任务中, 累计误差下降到一定程度后, 进一步下降会非常缓慢, 这时标准BP算法往往会获得较好的解, 尤其当训练集非常大时效果更明显.

误差逆传播算法

- 标准 BP 算法 Stochastic Gradient Descent
  - 每次针对单个训练样例更新权值与阈值 每读一个数据集更新一次参数
- 累计 BP 算法
  - 优化的目标是最小化整个训练集上的累计误差 $E = \frac{1}{m}\sum_{k=1}^{m} E_k$

读完整个数据集再更新参数（更快地收敛）

一般是先累计BP再标准BP

- 小批量随机梯度下降法 Mini-Batch Stochastic Gradient Descent

$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \frac{1}{m}\sum_{k=1}^{m}\frac{\partial E_k}{\partial \boldsymbol{\theta}} \approx \frac{1}{n}\sum_{i=1}^{n}\frac{\partial E_i}{\partial \boldsymbol{\theta}}$$

$n \ll m$，称为batch size 批大小

随机性bias --> shuffle

需要保证梯度无偏性

误差逆传播算法

- 多层前馈网络局限　数据集太少--->过拟合
  - 神经网络由于强大的表示能力, 经常遭遇过拟合

    表现为：训练误差持续降低, 但测试误差却可能上升

早停

- 在训练过程中, 若训练误差降低, 但验证误差升高, 则停止训练

正则化　保证权重不会太大

- 在误差目标函数中增加一项描述网络复杂程度的部分, 例如连接权值与阈值的平方和

  - 如何设置隐层神经元的个数仍然是个未决问题

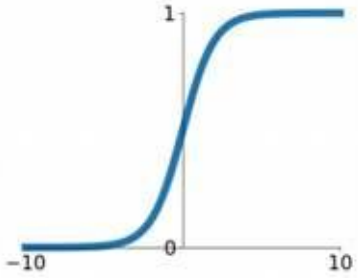
    实际应用中通常使用“试错法”调整

激活函数

这两个函数容易陷入饱和 --> 梯度下降到0

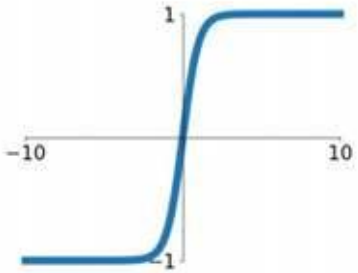
Sigmoid

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



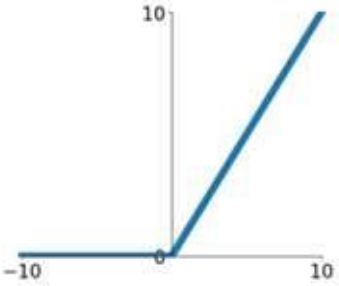
tanh

$$\tanh(x)$$



ReLU

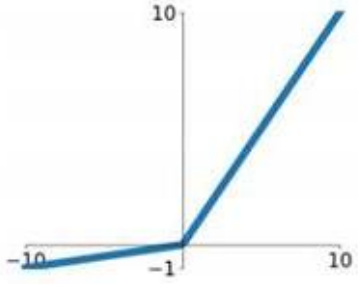
$$\max(0, x)$$



对ReLU改进，在 x < 0 改为梯度很小，可以继续学习

Leaky ReLU

$$\max(0.1x, x)$$

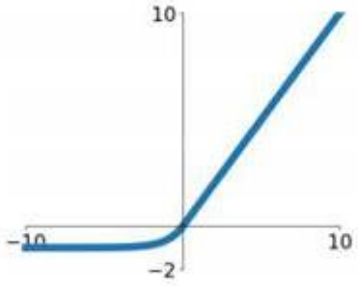


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

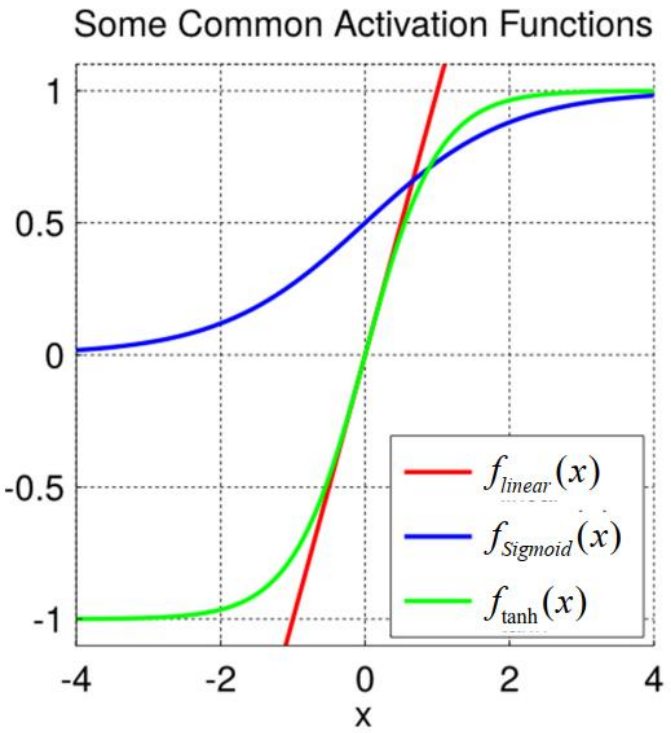
ELU

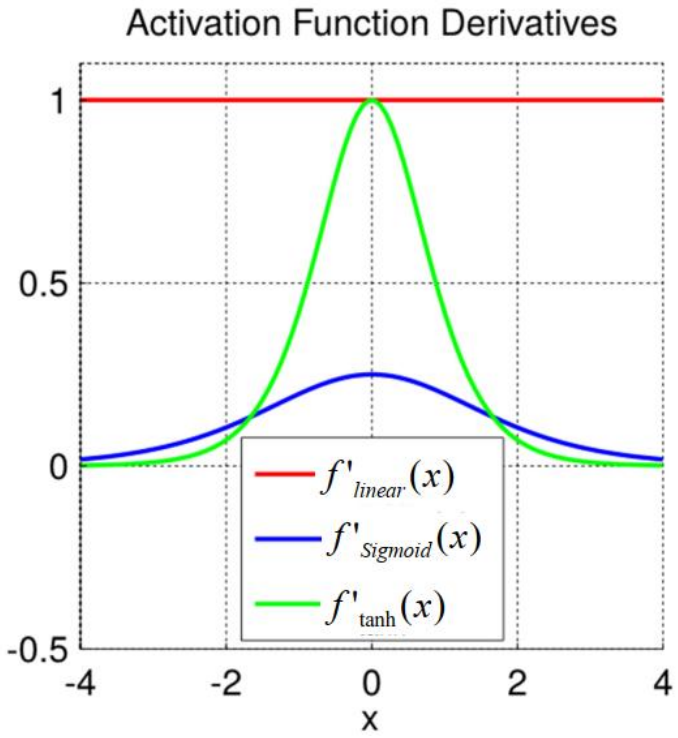
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



x > 0 时梯度恒为1，不会陷入饱和。
但x < 0 后梯度为0，无法继续学习

激活函数


Some Common Activation Functions
1
0.5
0
-0.5
-1
-4
-2
0
2
4
x
$f_{linear}(x)$
$f_{Sigmoid}(x)$
$f_{\tanh}(x)$


Activation Function Derivatives
1
0.5
0
-0.5
-4
-2
0
2
4
x
$f'_{linear}(x)$
$f'_{Sigmoid}(x)$
$f'_{\tanh}(x)$

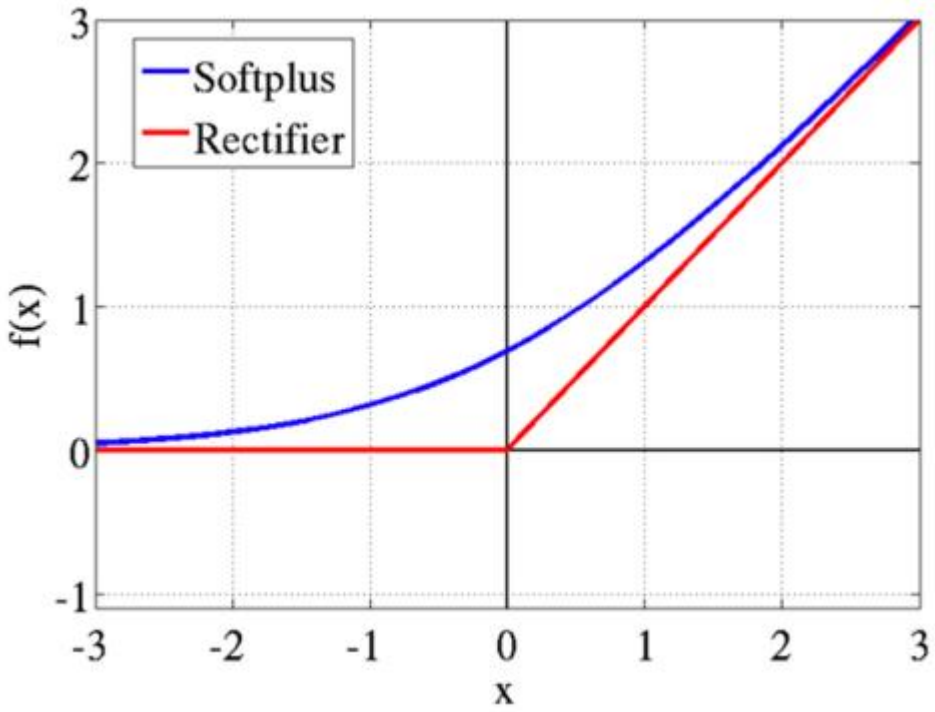
激活函数

- ReLU（整流线性单元）
- 与Softplus函数近似

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

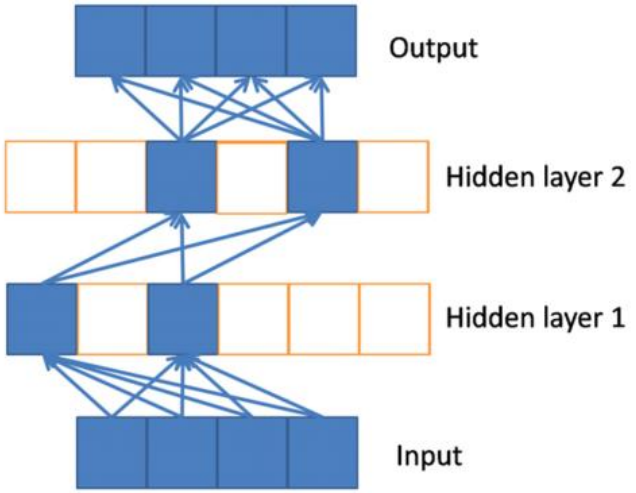
$$f_{\text{Softplus}}(x) = \log(1 + e^{x})$$

蓝色这个激活函数效果不好，导致网络稀疏



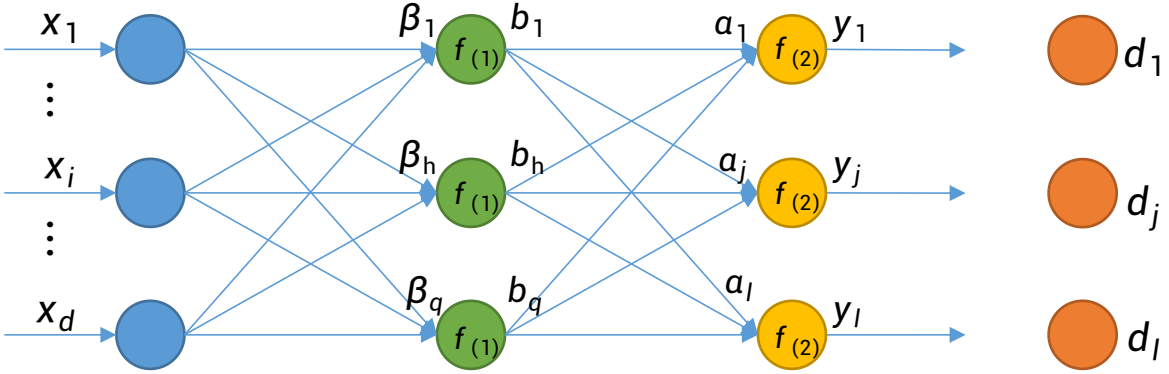
ReLU支持稀疏表示

只有一部分神经元被激活



损失函数

均方误差 $\frac{1}{2}\sum_j (y_j - d_j)^2$



KL散度（描述某种意义上的距离）

one-hot向量

多分类损失 Cross Entropy $-\sum_j d_j \log y_j$

交叉熵

$$H(P,Q) = \sum_x P(x)\log Q(x)$$

$$y_j = \frac{\exp(a_j)}{\sum_{j'}\exp(a_{j'})}$$

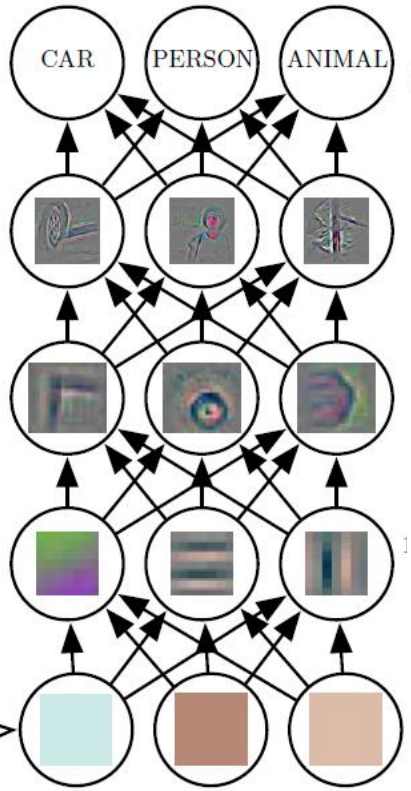
二分类损失 Cross Entropy $l = 1$ $-d\log y - (1-d)\log(1-y)$

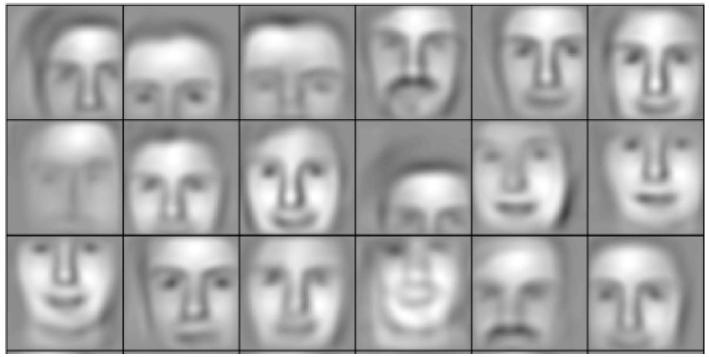
$$y = \frac{1}{1+\exp(-a)}$$

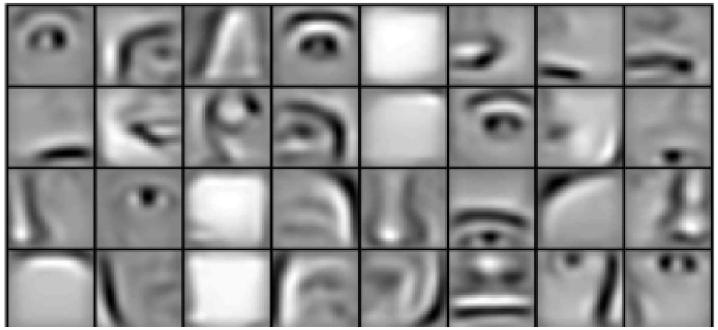
深度学习初探

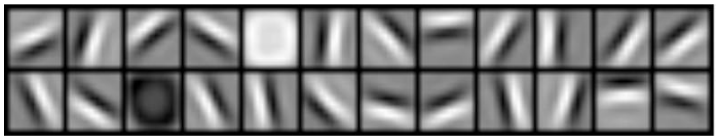
层次性地抽取特征 --> 递归性定义

- 深度学习将大千世界表示为嵌套的层次概念体系
  - 由较简单概念间的联系定义复杂概念
  - 从一般抽象概括到高级抽象表示



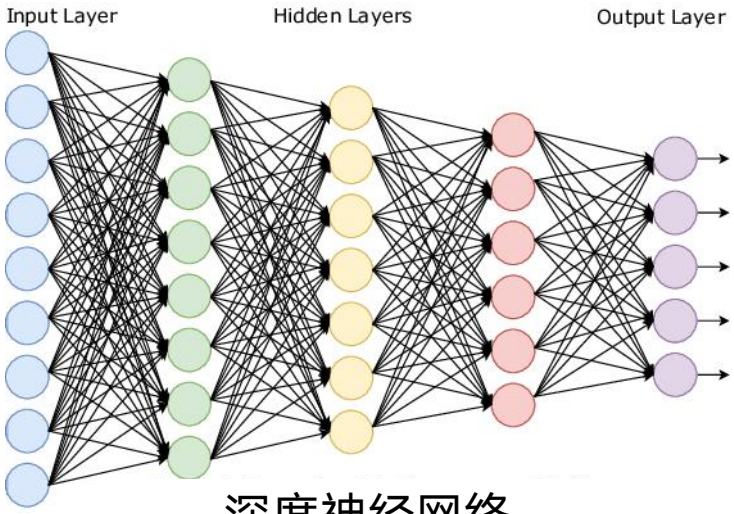






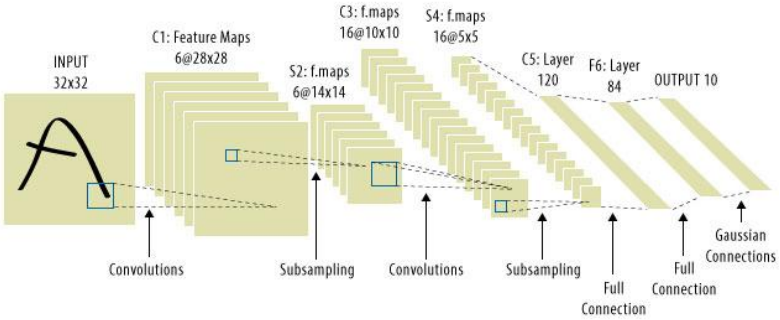
颜色对比 --> 形成轮廓

深度神经网络

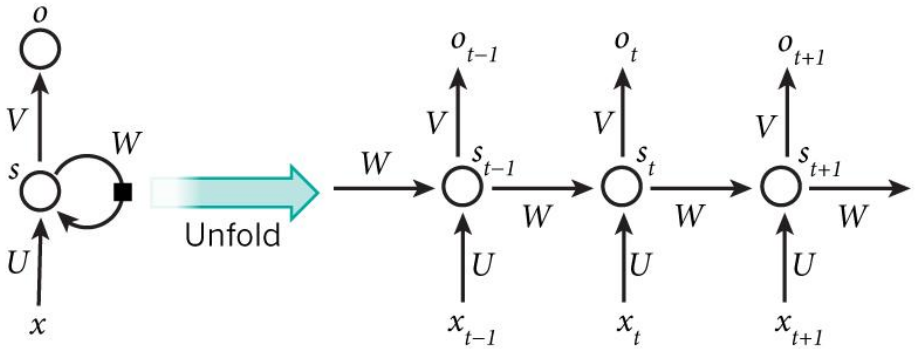


深度神经网络

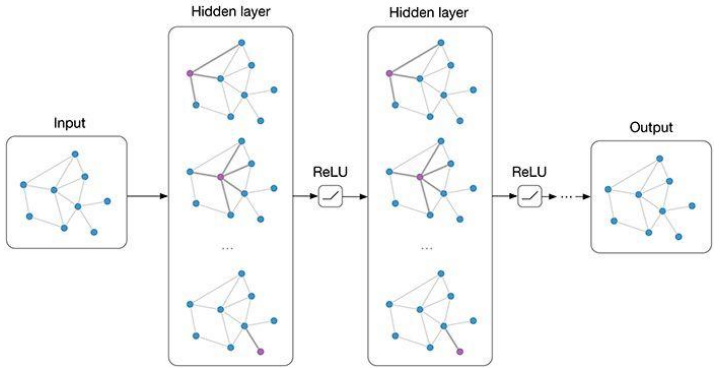
共享参数的前窥神经网络



深度卷积网络



循环神经网络



图卷积网络

深度卷积网络

- 一维卷积

$$s(t) = \int x(a)w(t-a)da$$

$$s(t) = \sum_a x(a)w(t-a)$$

输入

$$s(t) = (x * w)(t)$$

核函数

- 二维卷积

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, i-n)$$

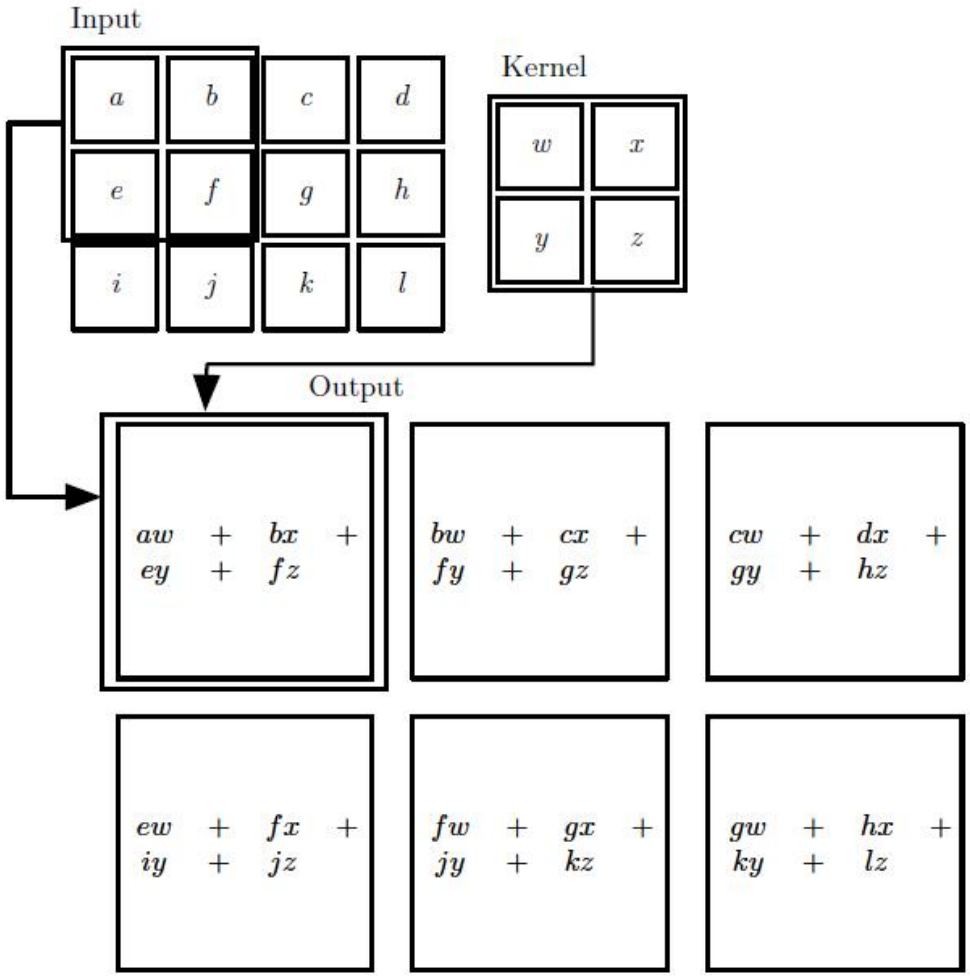
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

- 互相关函数

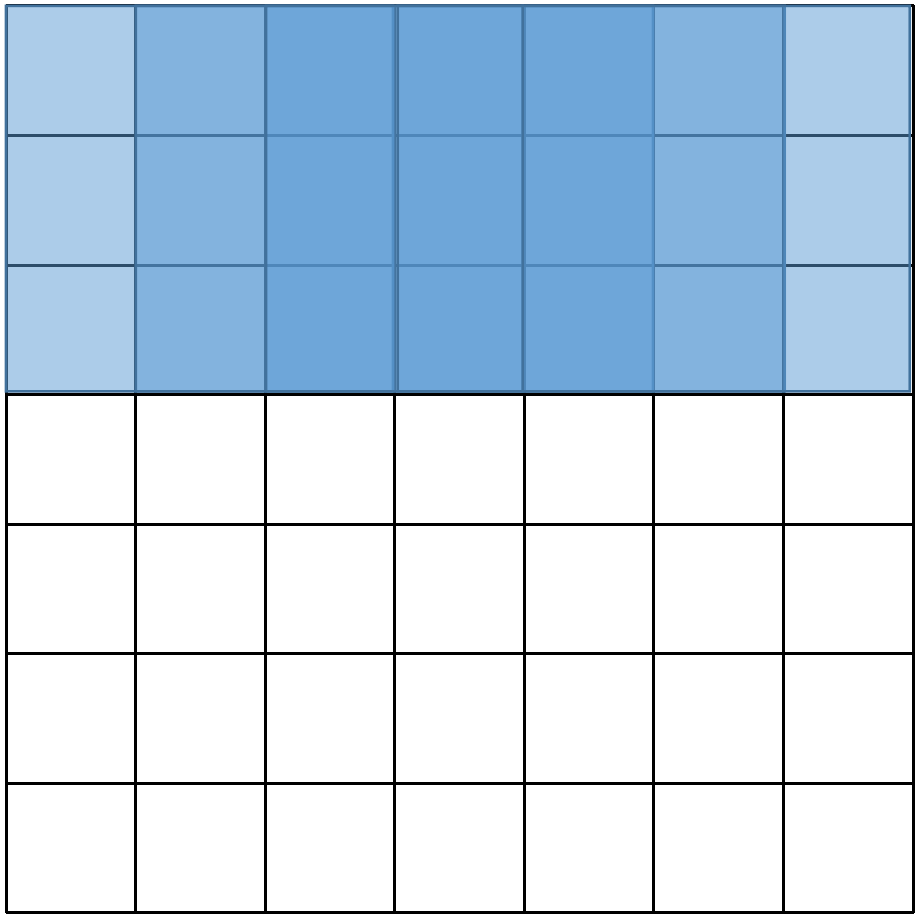
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

卷积网络中的卷积

卷积的例子


Input
a b c d
e f g h
i j k l
Kernel
w x
y z
Output
aw + bx + ey + fz
bw + cx + fy + gz
cw + dx + gy + hz
ew + fx + iy + jz
fw + gx + jy + kz
gw + hx + ky + lz

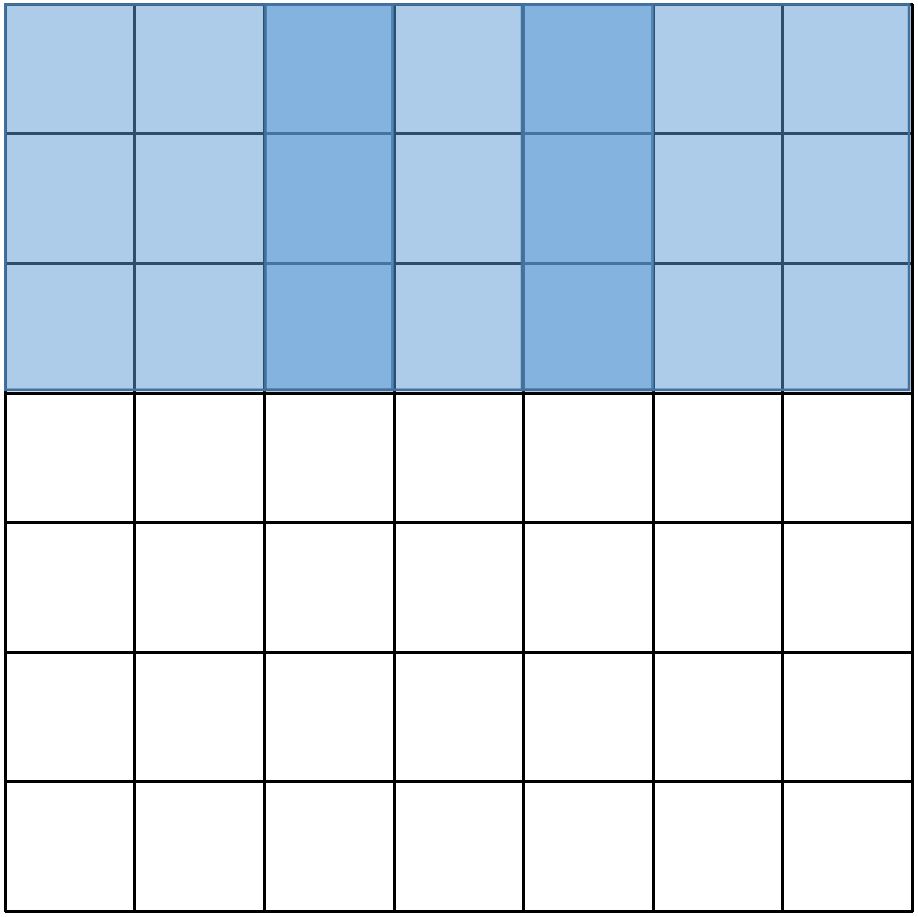
卷积的例子



7x7的输入
3x3的核

5x5的输出

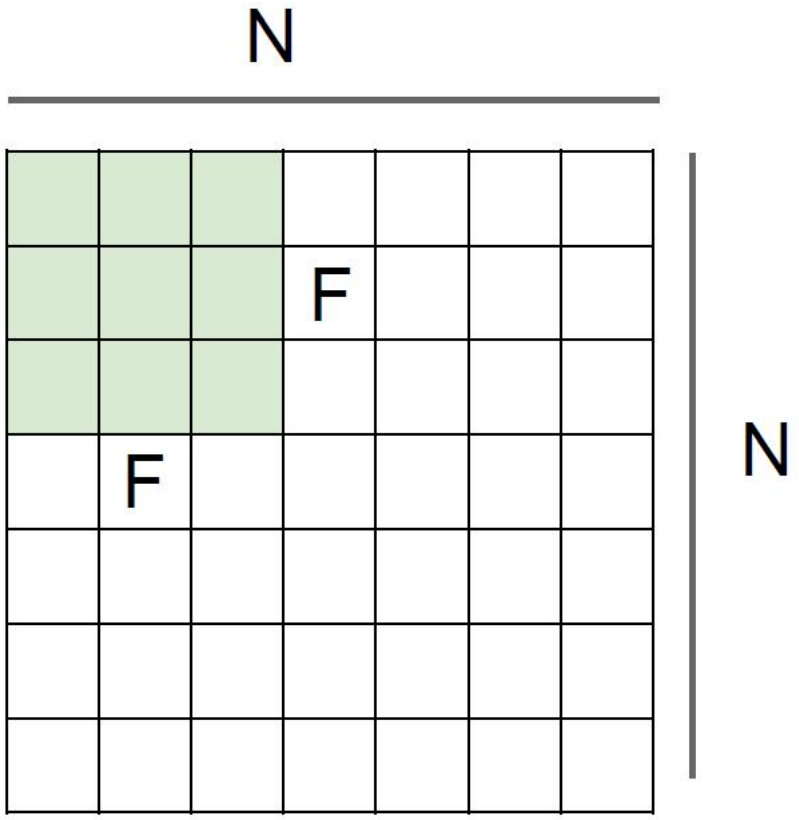
卷积的例子



7x7的输入
3x3的核
步幅为2

3x3的输出

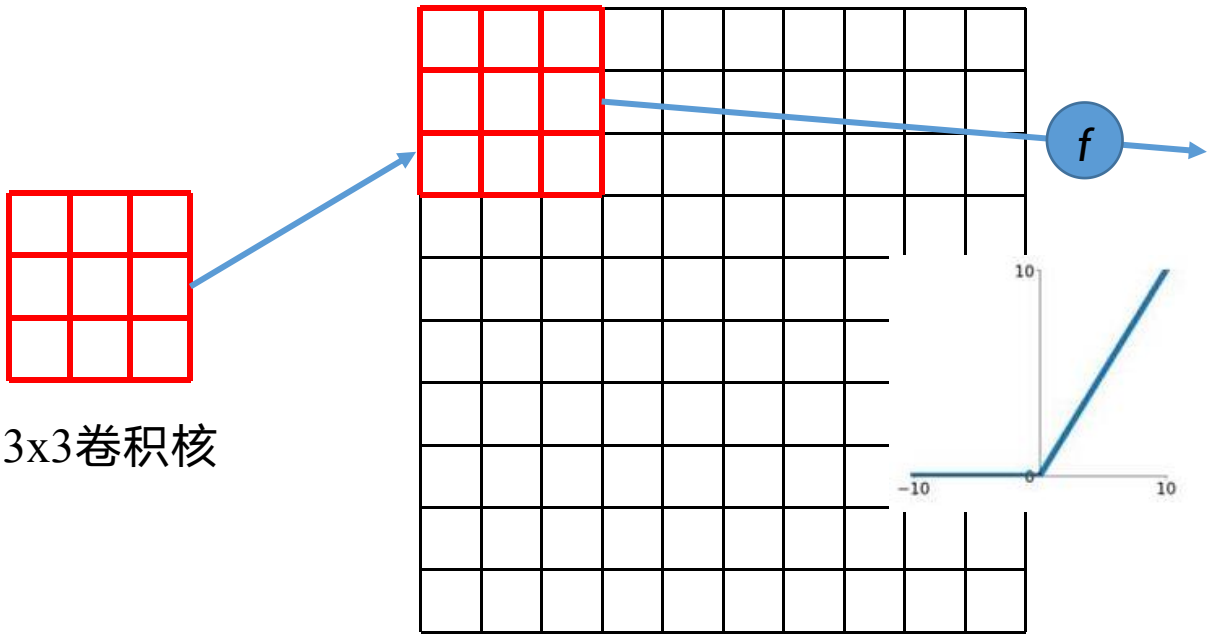
卷积



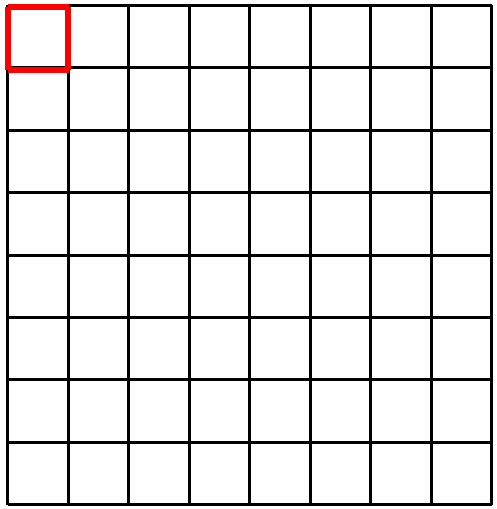
- 输出大小：(N-F)/S+1
  - S为步幅大小
- 比如N=7, F=3
  - 步幅为1，(7-3)/1+1=5
  - 步幅为2，(7-3)/2+1=3
  - 步幅为3，(7-3)/3+1=2.3..

卷积层

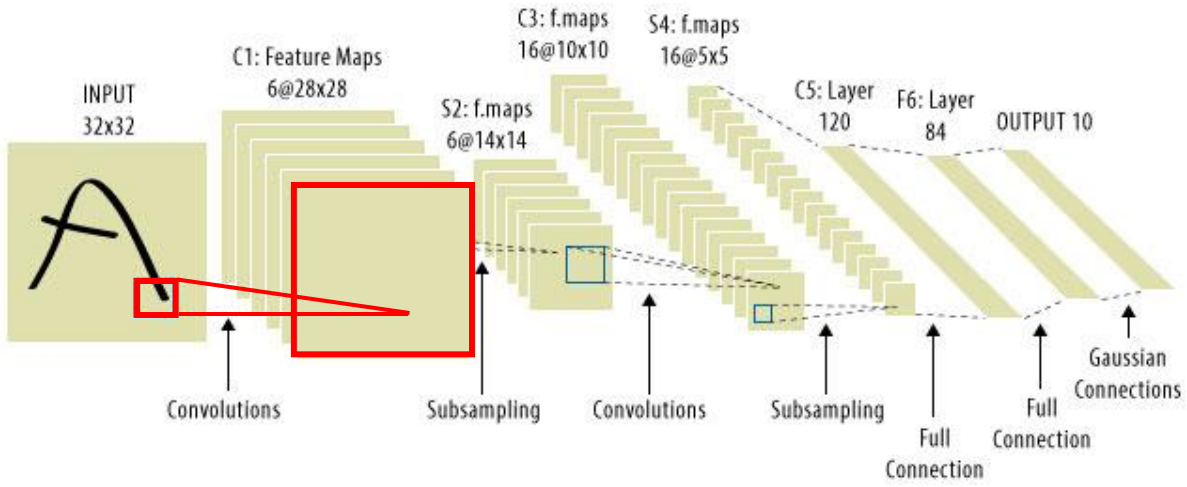
输入：10x10图片



输出：8x8图片



3x3卷积核

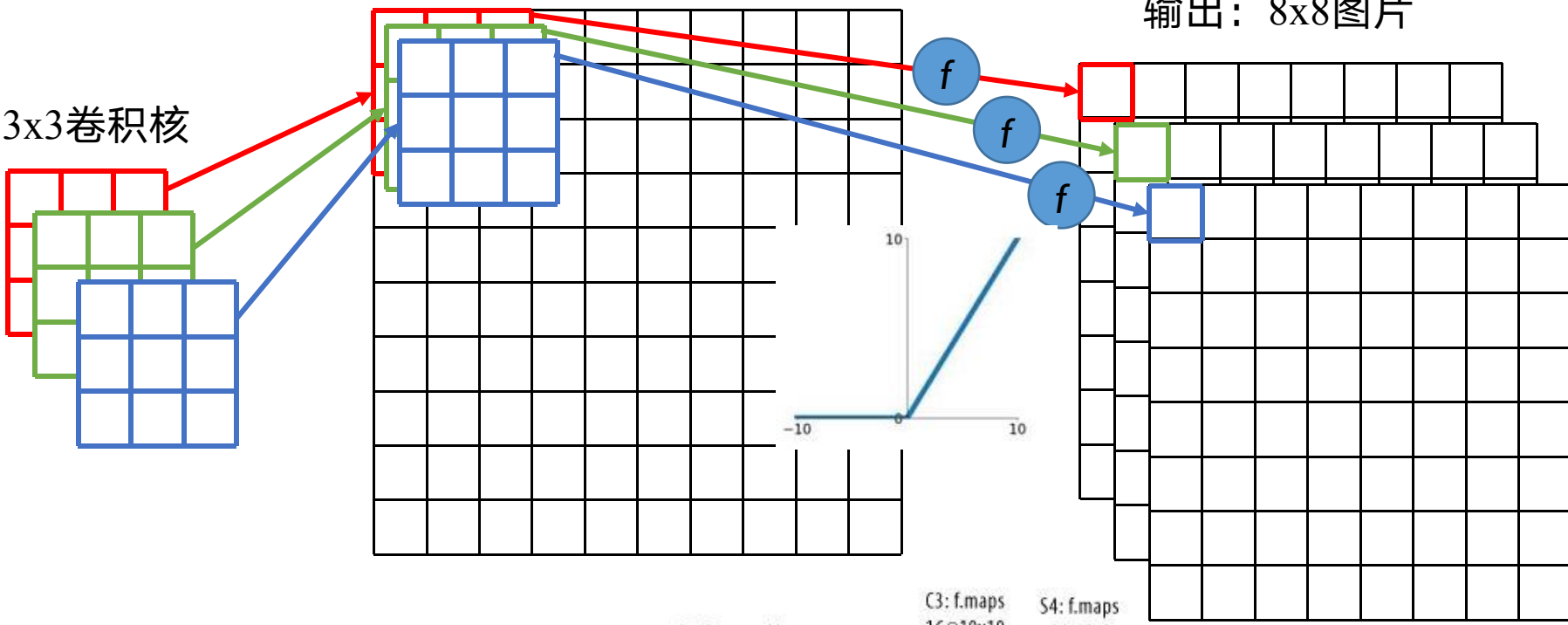


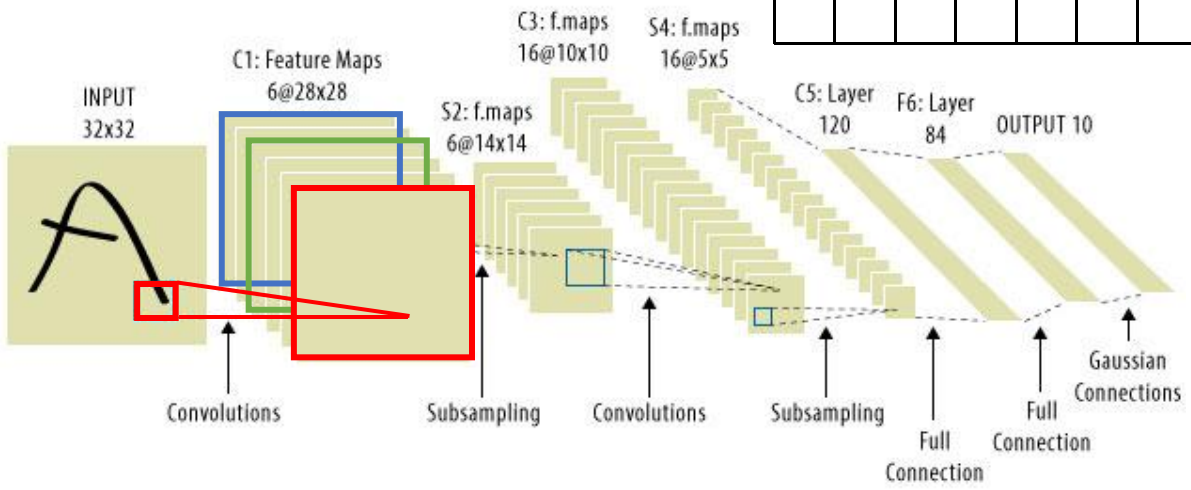
卷积层

输入：10x10图片

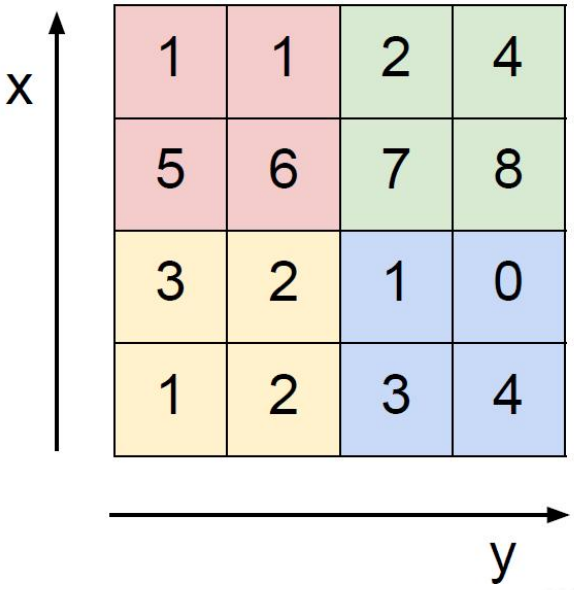
输出：8x8图片

3x3卷积核

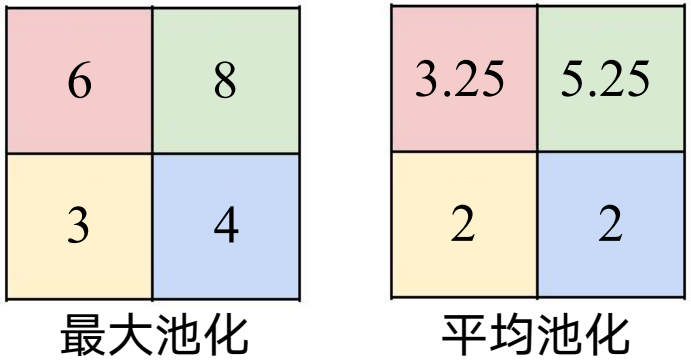




池化层

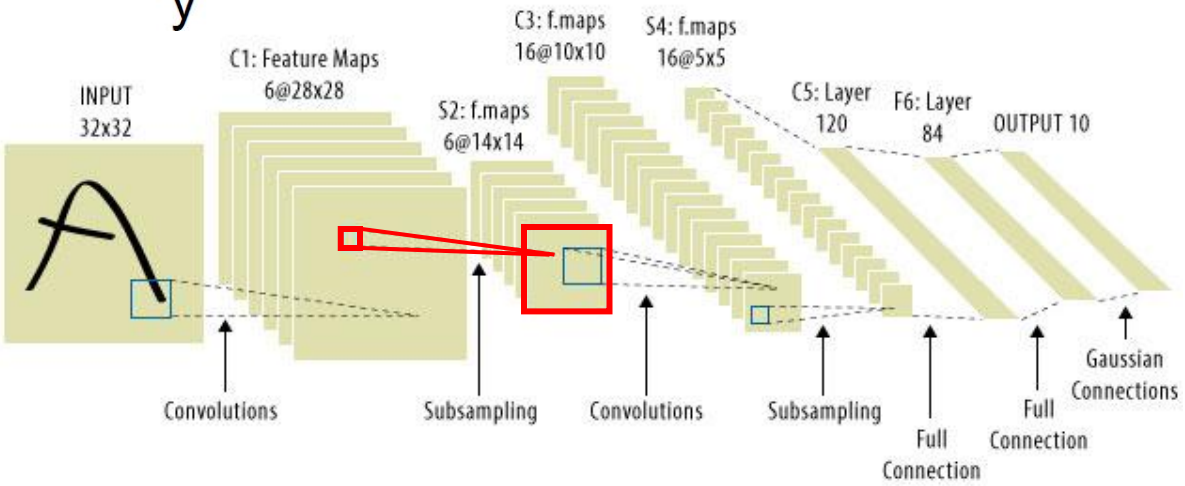


2x2的核，步幅为2



最大池化　　平均池化

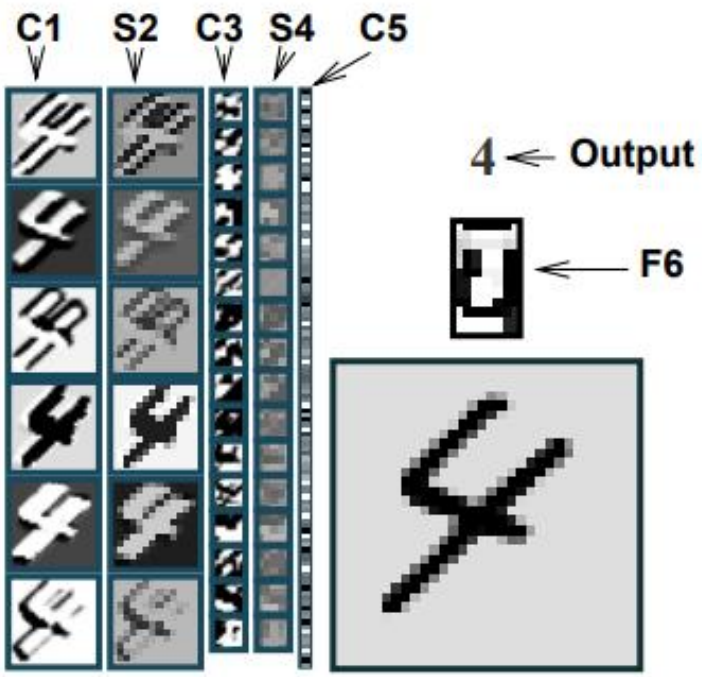
pulling可以带来一定的平移不变性



手写字符识别

- MNIST (handwritten digits) 数据集 http://yann.lecun.com/exdb/mnist/

6万训练样本和1万测试样本

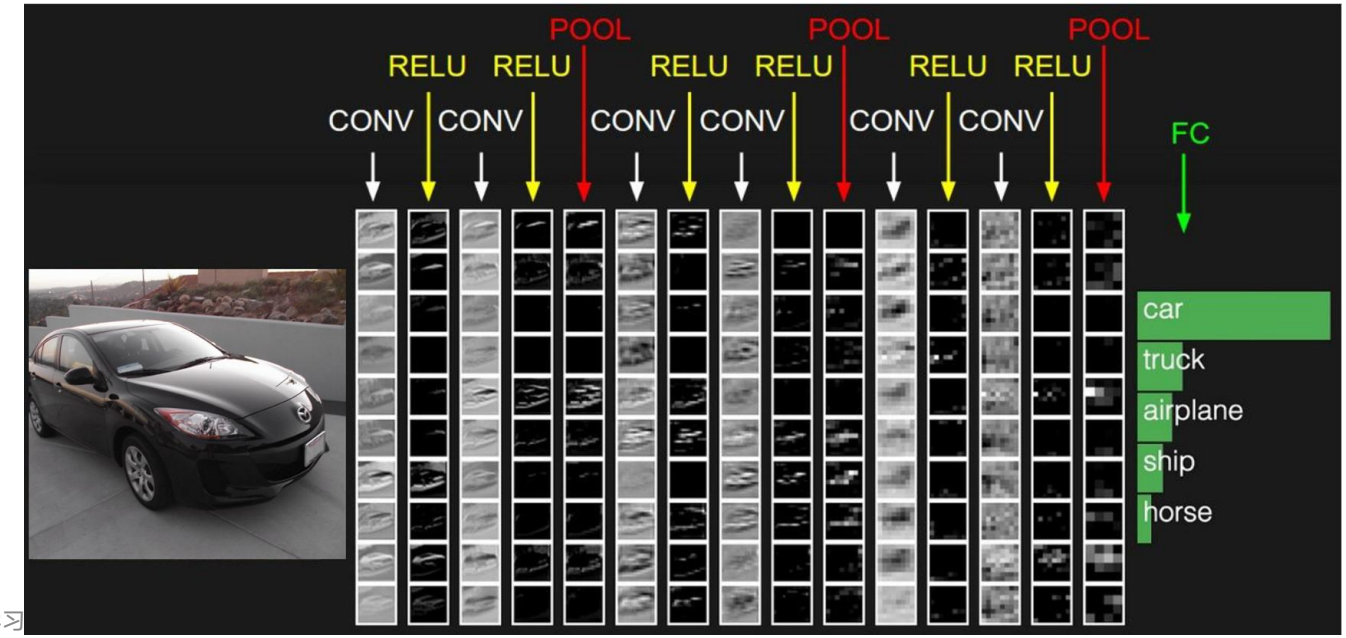


错误案例

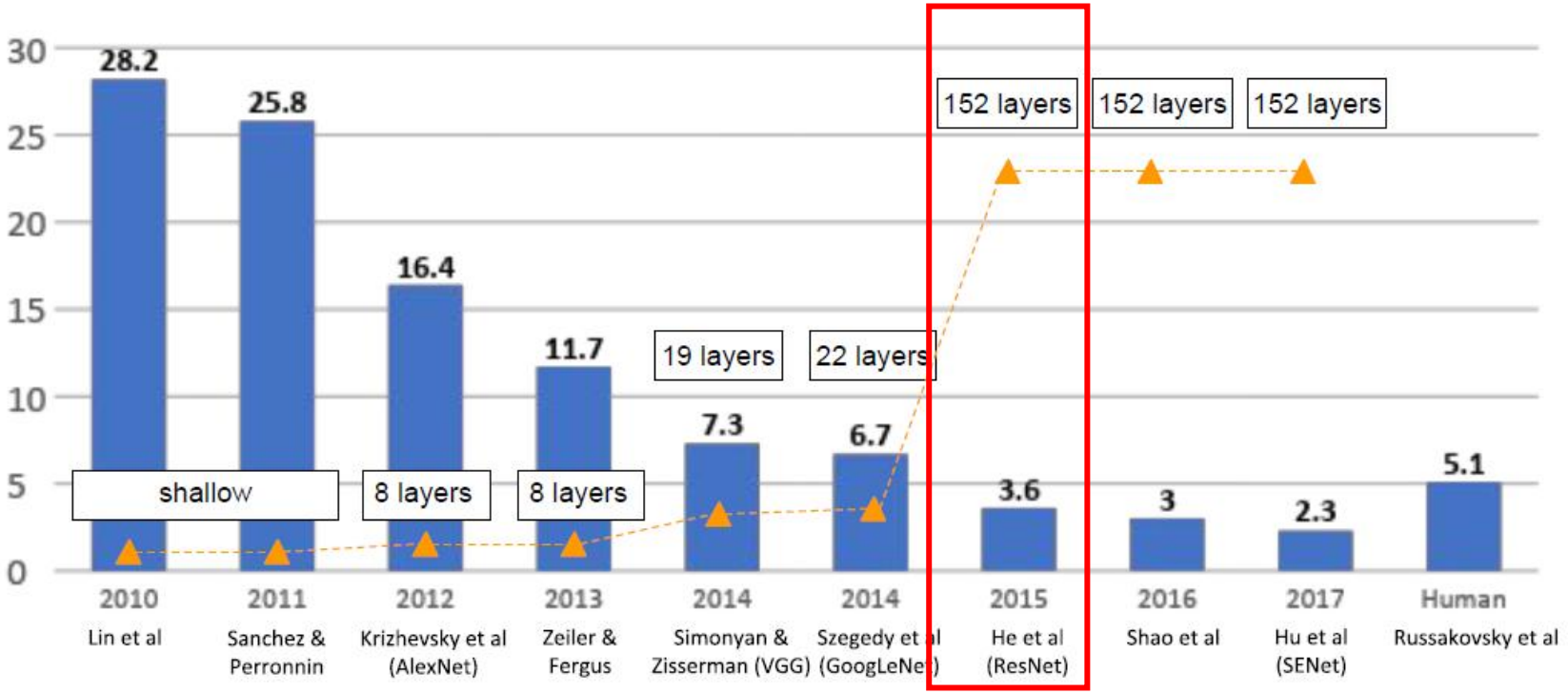
图片分类

ImageNet数据集

大约2万2千个类别，1千5百万张经过Amazon众包平台标注过的图片



图片分类

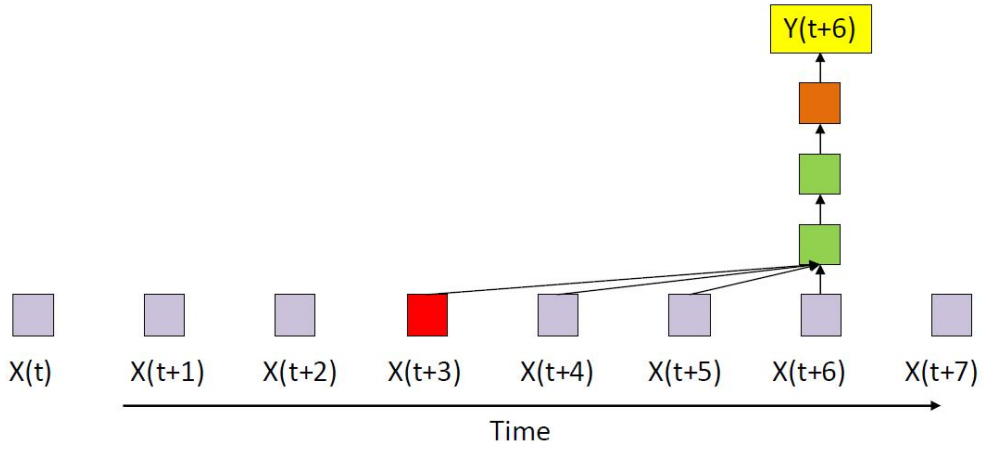


微软ResNet

在ImageNet图像数据库上，达到4.94%的错误率，低于人类5.1%的错误率

循环神经网络

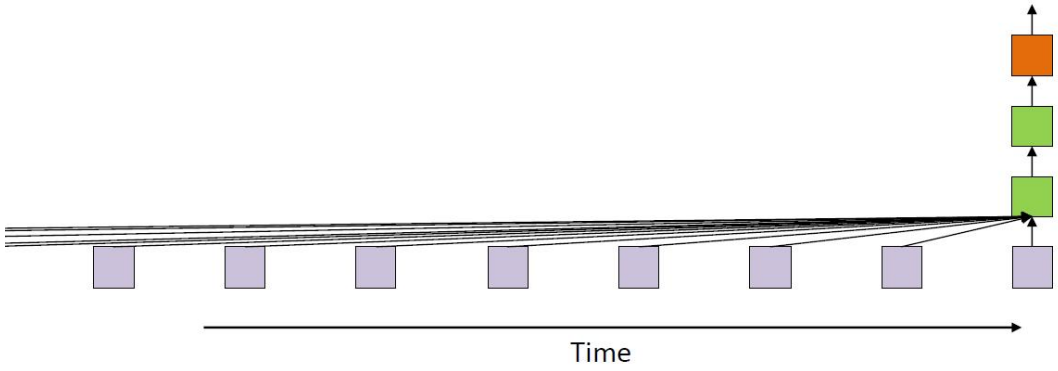
- 建模序列数据，比如文本、时间序列等



有限响应模型

今天的信息只会对未来N天内的预测有用

$$Y_t = f(X_t, X_{t-1}, \cdots, X_{t-N})$$



无限响应模型

今天的信息对未来任何时刻的预测都有用

$$Y_t = f(X_t, X_{t-1}, \cdots, X_{t-\infty})$$

循环神经网络

- 用隐状态变量（状态）$\boldsymbol{h}_t$“存储”直到t时刻的历史数据$\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\}$，并用状态转移方程进行描述

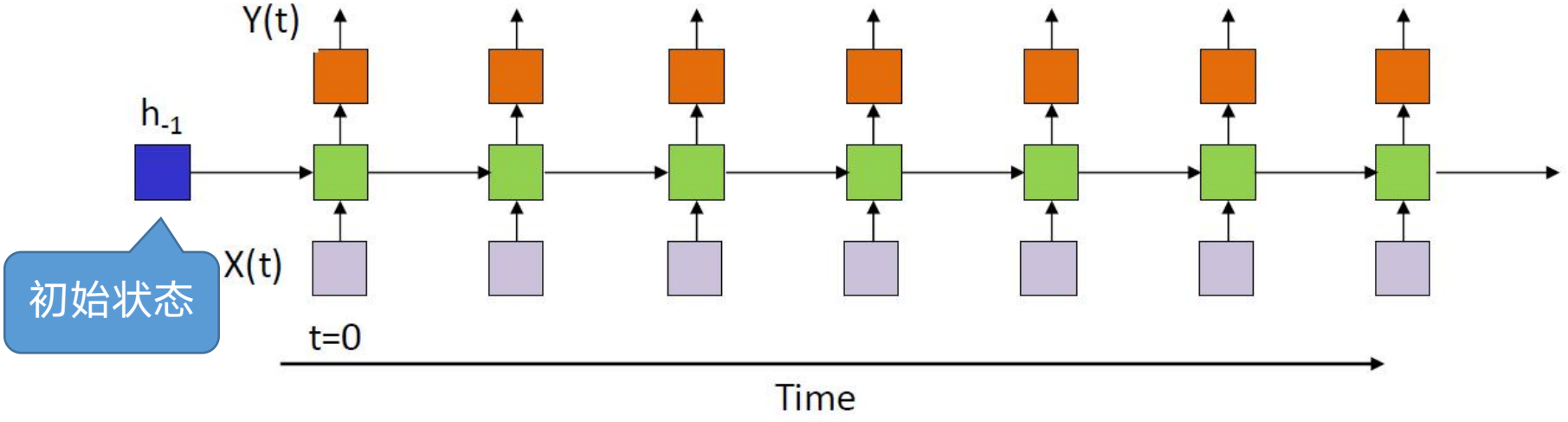
$$\boldsymbol{h}_t = f_W(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

每个时间步上的函数相同

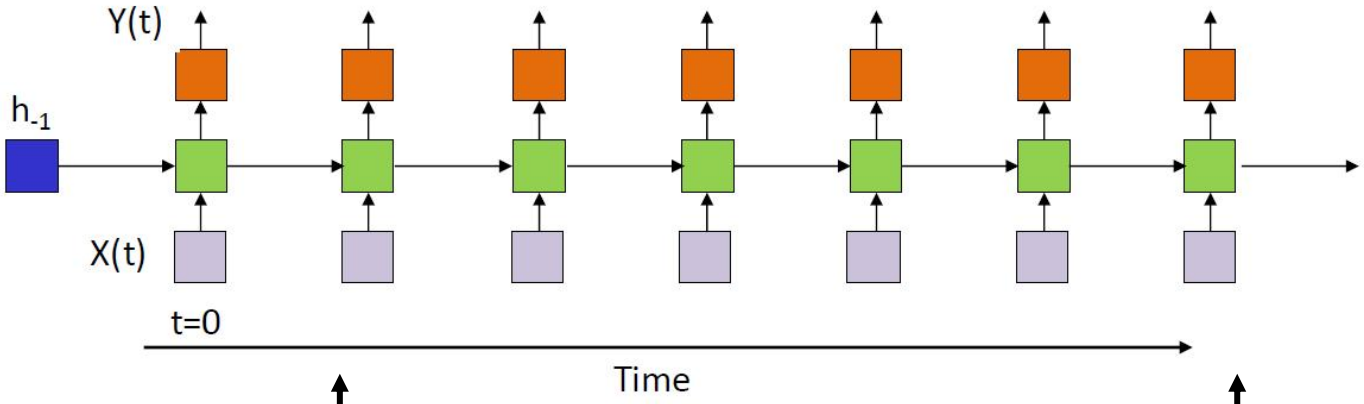
新状态　参数W　旧状态　　t时刻的输入

$$\boldsymbol{h}_t = f_W(f_W(f_W(\boldsymbol{h}_{t-3}, \boldsymbol{x}_{t-2}), \boldsymbol{x}_{t-1}), \boldsymbol{x}_t)$$

$$y_t = g(\boldsymbol{h}_t)$$

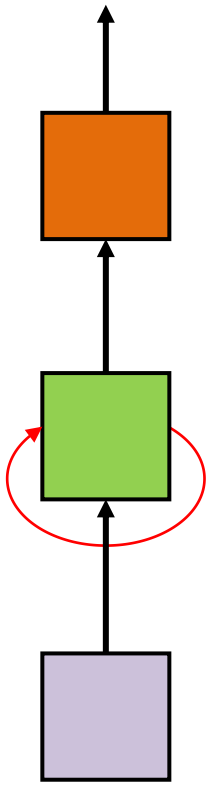


循环神经网络



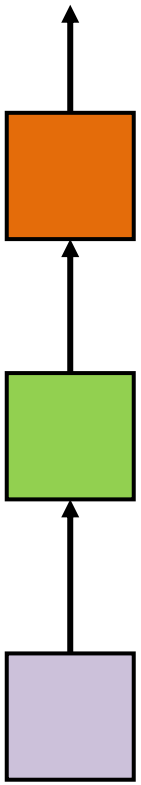
$$y_t = g(\boldsymbol{h}_t)$$

$$\boldsymbol{h}_t = f_{\boldsymbol{W}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

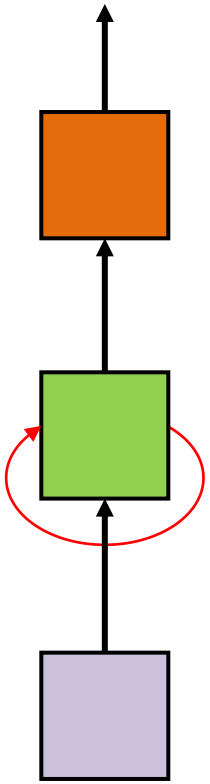


$$y_t = g(\boldsymbol{h}_t)$$

$$\boldsymbol{h}_t = f_{\boldsymbol{W}}(\boldsymbol{x}_t)$$



循环神经网络



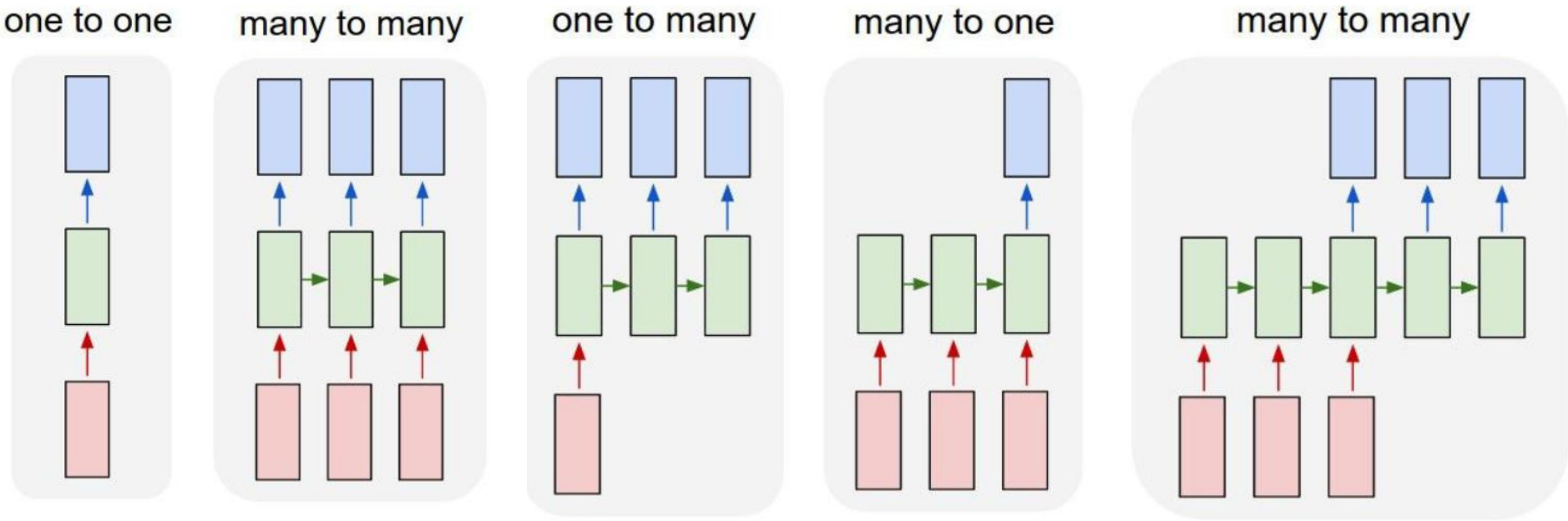
$$y_t = g(\boldsymbol{h}_t) \quad = \boldsymbol{W}_{\text{hy}}\boldsymbol{h}_t$$

$$\boldsymbol{h}_t = f_{\boldsymbol{W}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

$$= \tanh(\boldsymbol{W}_{\text{hh}}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{x\text{h}}\boldsymbol{x}_t)$$

称Vanilla RNN
或Elman RNN

循环神经网络



前向网络

例：视频帧分类
视频帧 -> 类别

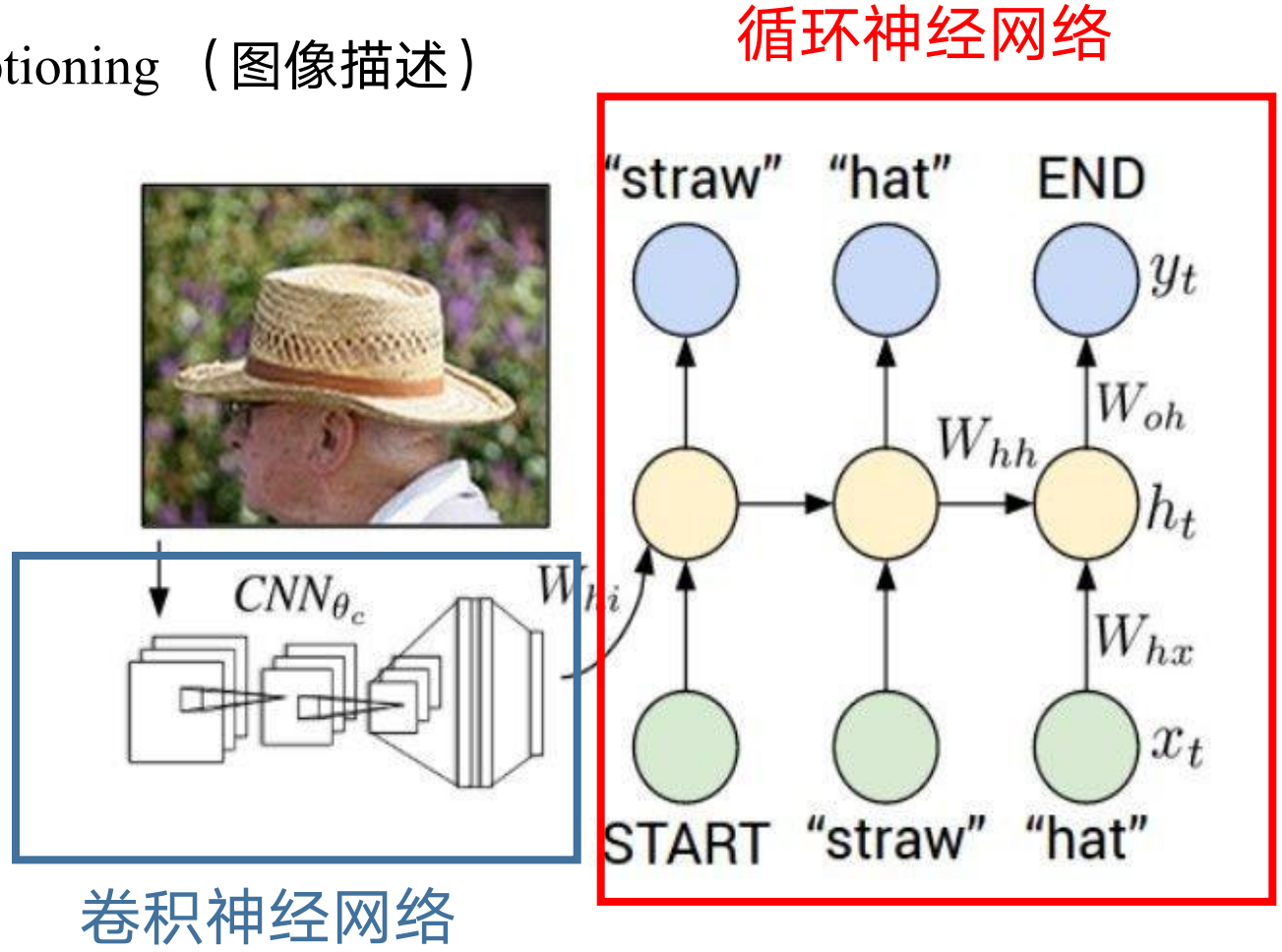
例：图像描述
图像 -> 句子

例：文本分类
文档 -> 类别

例：机器翻译
句子 -> 句子

循环神经网络

Image Captioning （图像描述）



循环神经网络

Image Captioning （图像描述）

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

循环神经网络

Google 机器翻译

翻译

关闭即时翻译

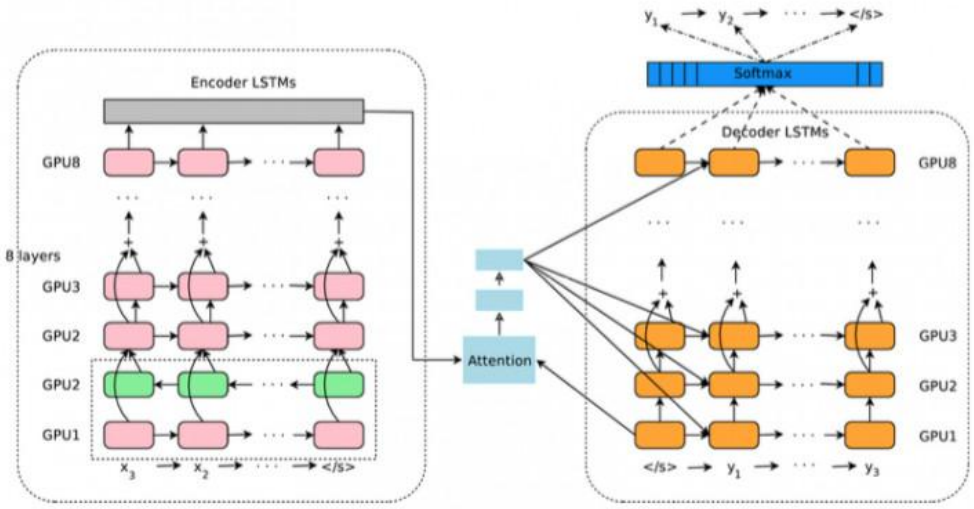
中文 英语 德语 检测语言 ⇄ 英语 中文(简体) 日语 翻译

Google神经机器翻译系统面世。该系统克服在大型数据集上工作的挑战，不再将句子分解为词和短语独立翻译，而是翻译完整的句子，使得误差降低了55%-85%以上。目前这项技术已经运用于Google Translate的汉英翻译。

112/5000

The Google Neuro Machine Translation System is available. The system overcomes the challenge of working on large data sets, no longer decomposing sentences into independent translations of words and phrases, but translating complete sentences, reducing errors by more than 55%-85%. This technology is currently used in Chinese-English translation of Google Translate.

提出修改建议



2016 年，Google官方将全产品线的翻译算法换成了基于神经网络的机器翻译系统

作业

- 5.1
- 讨论$\frac{\exp(x_i)}{\sum_{j=1}^{C}\exp(x_j)}$和$\log\sum_{j=1}^{C}\exp(x_j)$的数值溢出问题
- 计算$\frac{\exp(x_i)}{\sum_{j=1}^{C}\exp(x_j)}$和$\log\frac{\exp(x_i)}{\sum_{j=1}^{C}\exp(x_j)}$关于向量$\boldsymbol{x}=[x_1,\cdots,x_C]$的梯度
- 考虑如下简单网络，假设激活函数为ReLU，用平方损失$\frac{1}{2}(y-\hat{y})^2$计算误差，请用BP算法更新一次所有参数（学习率为1），给出更新后的参数值（给出详细计算过程），并计算给定输入值x=(0.2,0.3)时初始时和更新后的输出值，检查参数更新是否降低了平方损失值.