

深度学习

基本理论

DAY01

深度学习概述

深度学习概述

引入

人工智能划时代事件

深度学习的定义

深度学习巨大影响

深度学习的特点

什么是深度学习

深度神经网络

深度学习与机器学习的关系

深度学习的特点

深度学习的优点

深度学习的缺点

深度学习与机器学习对比

为什么要学习深度学习

深度学习的应用

深度学习的应用

课程内容与特点

课程内容

深度学习发展史

课程特点

学习资源推荐

深度学习发展简史

深度网络演化过程

引入

人工智能划时代事件

知识讲解

- 2016年3月，Google公司研发的AlphaGo以4:1击败世界围棋顶级选手李世石。次年，AlphaGo2.0对战世界最年轻的围棋四冠王柯洁，以3:0击败对方。背后支撑AlphaGo具备如此强大能力的，就是“深度学习”（Deep Learning）。
- 一时间，“深度学习”这个本专属于计算机学科的术语，成为包括学术界、工业界、风险投资界等众多领域的热词。



深度学习巨大影响

- 除了博弈，深度学习在计算机视觉（computer vision）、语音识别、自动驾驶等领域，表现与人类一样好，甚至有些地方超过了人类。2013年，深度学习就被麻省理工学院的《MIT科技评论》评为世界10大突破性技术之一。
- 深度学习不仅是一种算法升级，还是一种全新的思维方式，它的颠覆性在于，将人类过去痴迷的算法问题，演变成数据和计算问题，以前“算法为核心竞争力”正在转换为“数据为核心竞争力”。



深度学习的定义

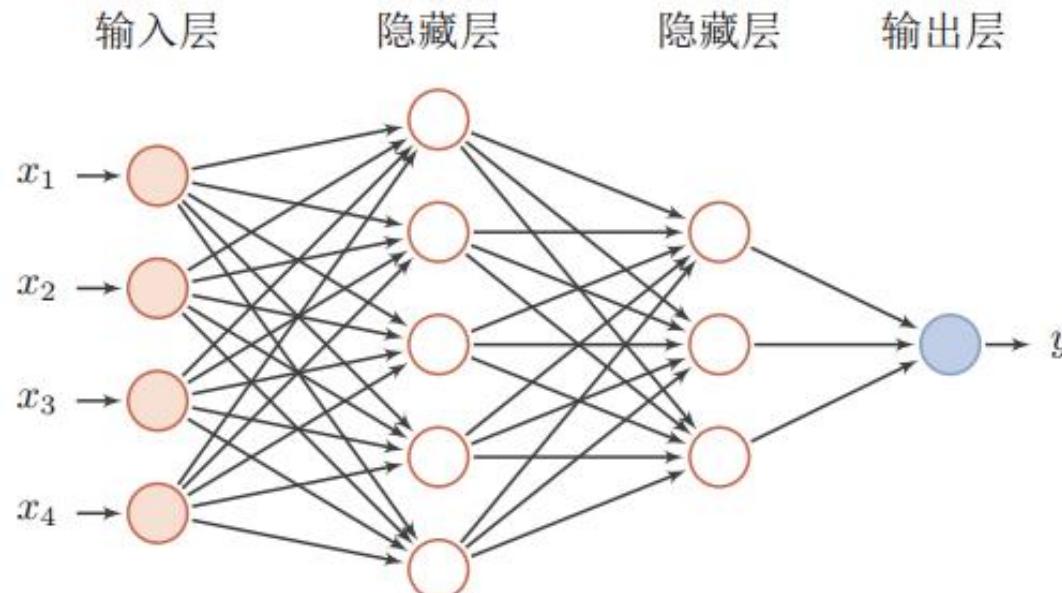
什么是深度学习

- 简单来说，深度学习就是一种包括多个隐含层（越多即为越深）的多层感知机。它通过组合低层特征，形成更为抽象的高层表示，用以描述被识别对象的高级属性类别或特征。能自生成数据的中间表示（虽然这个表示并不能被人类理解），是深度学习区别于其它机器学习算法的独门绝技。
- 所以，深度学习可以总结成：通过加深网络，提取数据深层次特征



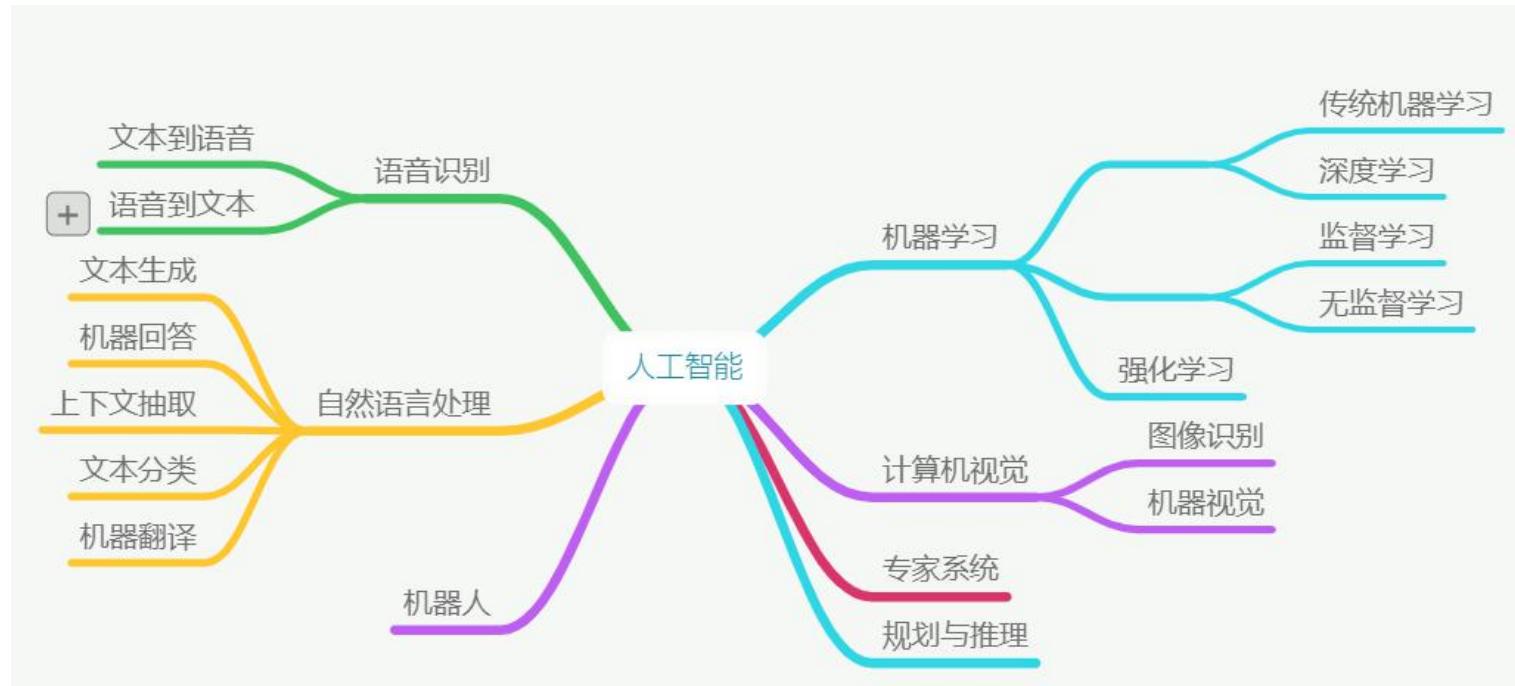
深度神经网络

- 深度神经网络（及其变种）是深度学习中心模型



深度学习与机器学习的关系

- 人工智能学科体系



深度学习与机器学习的关系（续）

- 人工智能、机器学习、深度学习三者的关系，可以认为深度学习是机器学习的“高级阶段”



深度学习的特点

深度学习的特点

• 优点

- 性能更优异
- 不需要特征工程
- 在大数据样本下有更好的性能
- 能解决某些传统机器学习无法解决的问题

• 缺点

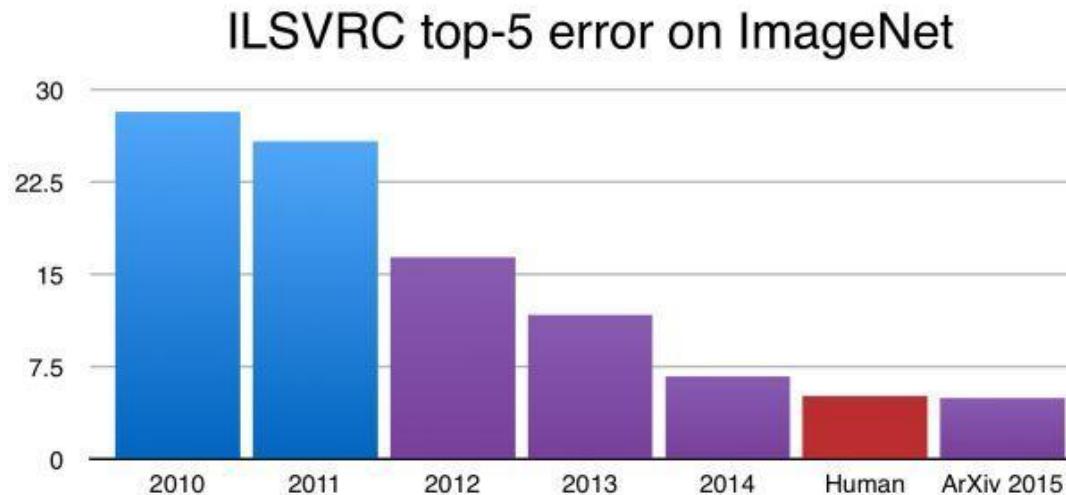
- 小数据样本下性能不如机器学习
- 模型复杂
- 过程不可解释



深度学习的优点

- 性能更优异

下图是历年ImageNet大规模视觉识别挑战 (ILSVRC) 的分类精度，其中蓝色是传统机器学习方法，其它为深度学习方法。2015年比赛成绩，识别率超过了人类



深度学习的优点（续1）

- 性能更优异

在语音识别领域，进入深度学习时代

后，识别率有了明显的提高

- 2015 Google：错误率8%
- IBM Watson智能系统：错误率6.9%
- 2016年9月 微软：错误率6.3%



近20年来语音识别错误率的下降趋势图



深度学习的优点（续2）

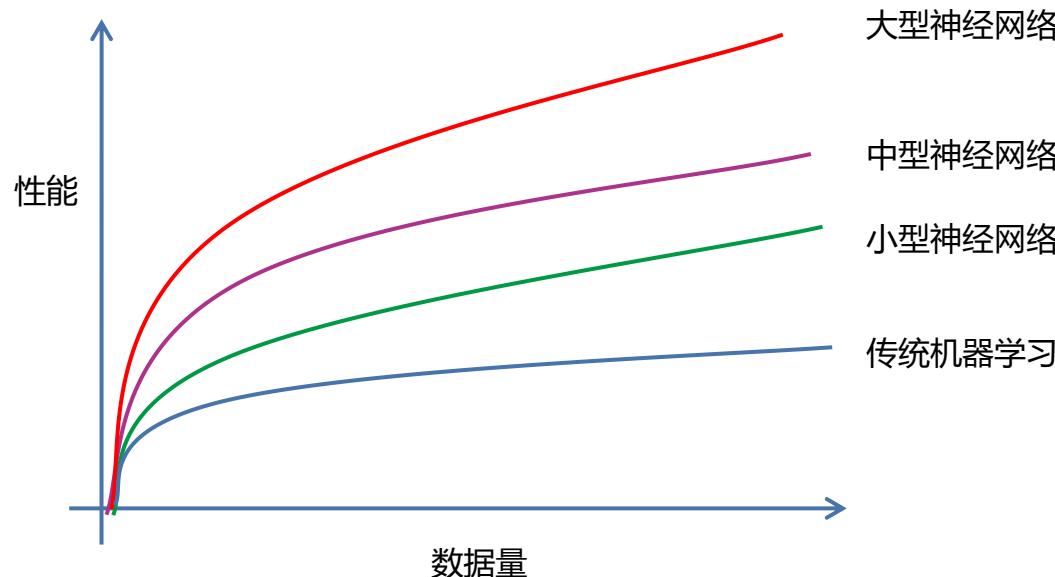
- 优点2：不需要特征工程

传统机器学习需要人进行特征提取（特征工程），机器性能高度依赖于特征工程的质量。在特征很复杂的情况下，人就显得无能为力。而深度学习不需要这样的特征工程，只需将数据直接传递给深度学习网络，由机器完成特征提取。



深度学习的优点（续3）

- 深度学习在大样本数据下有更好的性能和扩展性



深度学习的优点（续4）

- 深度学习能解决传统机器学习无法解决的问题（如深层次特征提取）

知识讲解



上传图一

上传图二

分析结果

相似度31.98%

同一个人的可能性低

说明



上传图一

上传图二

分析结果

相似度93.82%

同一个人的可能性极高

说明



深度学习的缺点

- 深度学习在小数据上性能不如传统机器学习
- 深度学习网络结构复杂、构建成本高
- 相比传统机器学习，深度学习可解释性较差



深度学习与传统机器学习对比

- 相同点

- 目的相同：都是利用机器自我学习能力，解决软件系统的难题
- 基本问题相同：回归问题、分类问题、聚类问题
- 基本流程相同：数据准备 --> 模型选择 --> 模型构建 --> 模型训练 --> 预测
- 问题领域相同
 - ✓ 样本是否有标签：监督学习、非监督学习、半监督学习
 - ✓ 应用领域：推荐引擎、计算机视觉、自然语言处理、强化学习
- 评价标准相同
 - 回归问题：均方误差；R2值
 - 分类问题：交叉熵；查准率、召回率、F1综合系数
 - 模型泛化能力：过拟合、欠拟合



深度学习与传统机器学习对比（续）

- 不同点

比较项	传统机器学习	深度学习
特征提取	人提取特征	模型自己发现特征
性能（准确度/精度）	低	高
可解释性	强	弱
训练数据	小	大
模型结构	简单	复杂



为什么要学习深度学习

- 深度学习具有更强的解决问题能力（例如图像识别准确率明显超过机器学习，甚至超过了人类）
- 掌握深度学习具有更强的职业竞争力
- 深度学习在行业中应用更广泛



深度学习的应用

深度学习的应用

- 照片上色



老照片上色



深度学习的应用（续1）

- 换脸



人脸更换



深度学习的应用（续2）

- 图像风格转换



生成毕加索、梵高、莫奈风格的蒙娜丽莎



深度学习的应用（续3）

- 虚拟主播



采用语音图像合成技术虚拟的新闻主播



深度学习的应用（续4）

- 声音模仿

知识讲解



机器模仿歌星声音



深度学习的应用（续5）



图像分类



人脸识别



图像迁移



语音处理



自动驾驶



机器博弈



机器人



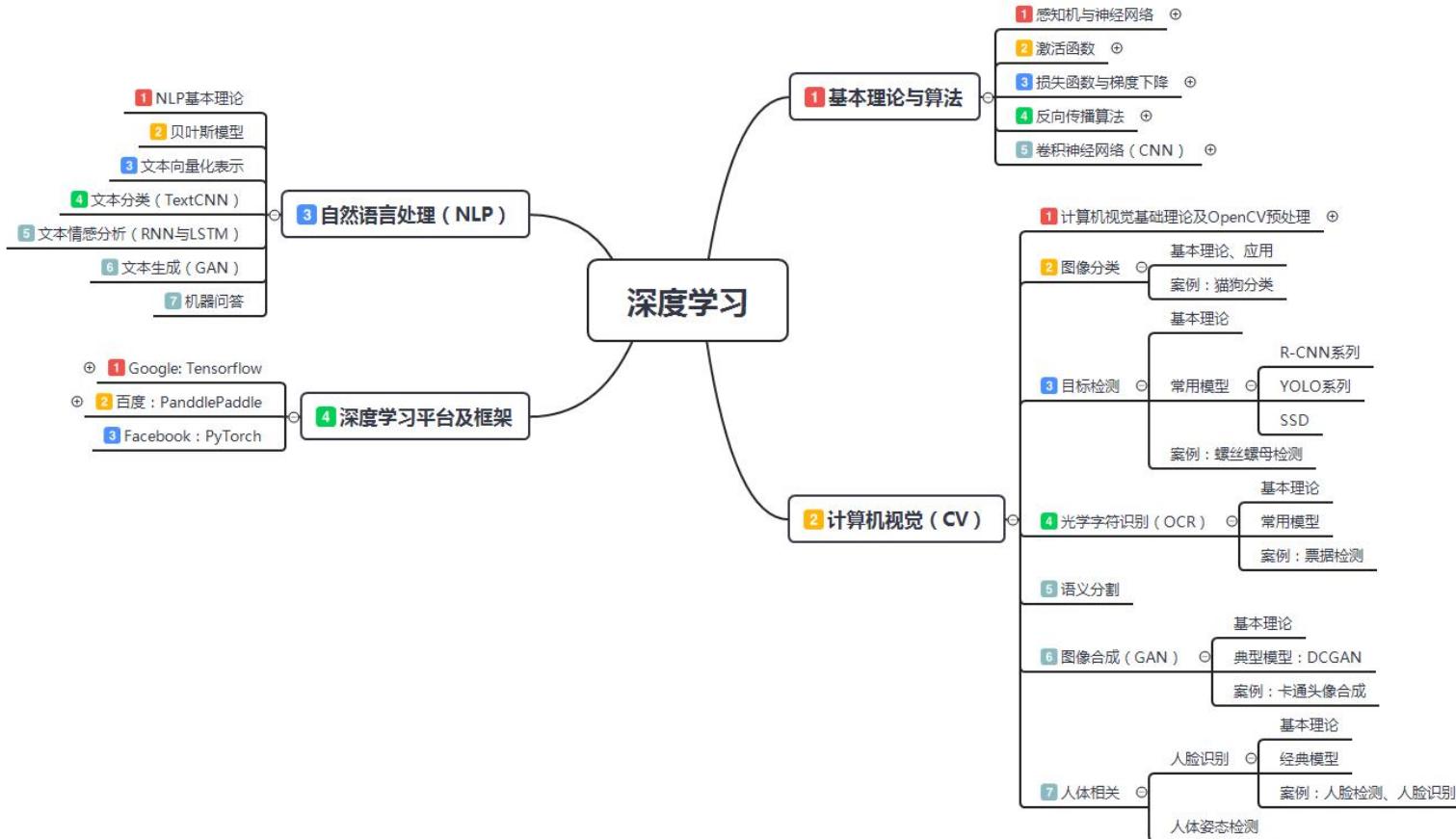
自然语言处理



课程内容与特点

课程内容

知识讲解



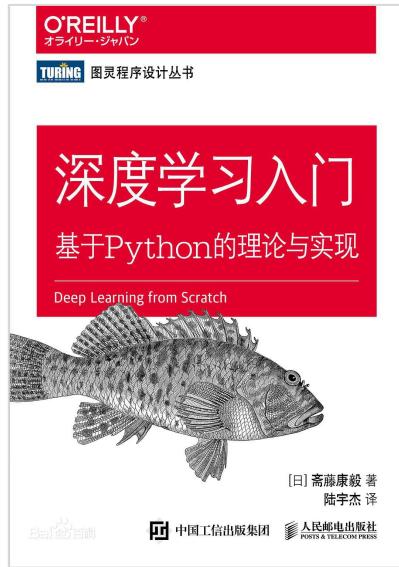
课程特点

- 概念术语多，理论复杂，学习曲线陡峭，需要长期、反复学习、理解、体会、实践
- 需要部分数学知识（记住结论、会使用API、理解公式、推导过程）
- 案例复杂度高
- 与传统程序差异较大



学习资源推荐

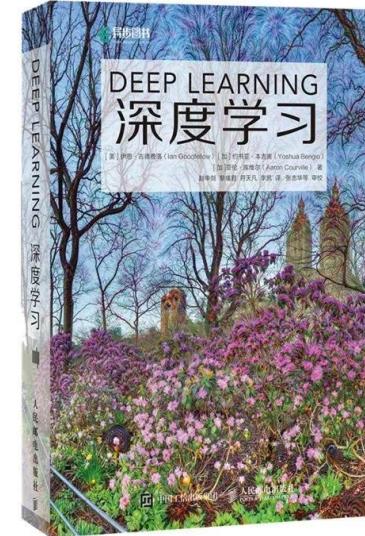
知识讲解



斋藤康毅 工信出版社



张玉宏 电子工业出版社



Ian Goodfellow、Yoshua
Bengio、Aaron Courville
工信出版社



学习资源推荐（续）

- 吴恩达深度学习视频：
<https://www.bilibili.com/video/av49445369?p=1>
- 西瓜视频：搜索“浙江大学研究生机器学习课程”



深度学习发展史

深度学习发展简史

- 从1940年起，首先提出了MP模型Hebb(海布)学习规则.这是神经网络的起源，也奠定了神经网络的基础模型。
- 1960年，提出了感知机模型，感知机模型可以对简单的数据节点进行分类，这个发现引起了第一波的AI浪潮，因为人们认为简单的感知机可以实现分类功能，那通过组合可以实现更复杂的功能，但后面发现感知机无法模拟异或运算，无法处理非线性的问题，第一波浪潮就这样沉入了低谷。



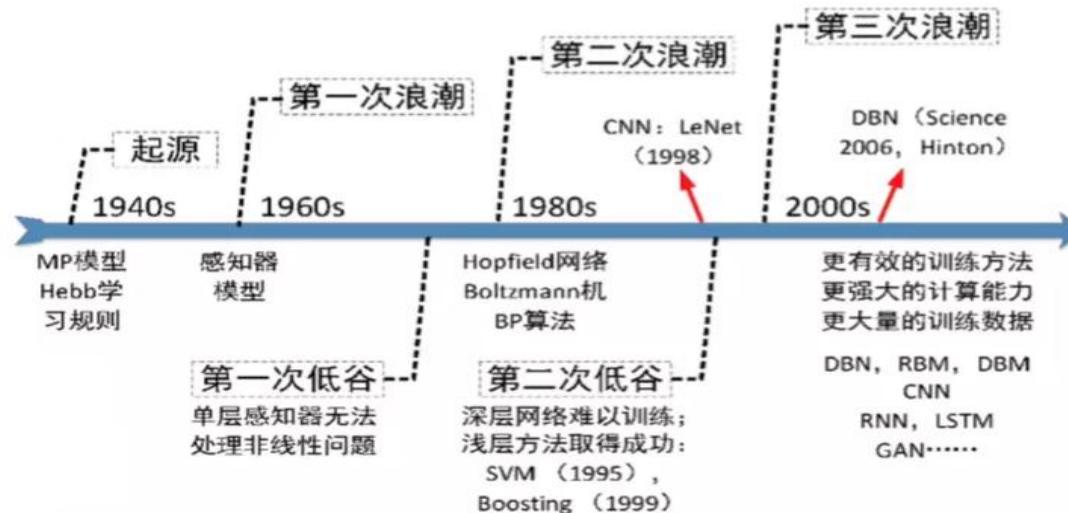
深度学习发展简史（续1）

- 1980年Hopfield网络，Boltzmann机和BP算法的提出，人们发现可以增加网络的深度来实现非线性的功能，所以开始了第二次浪潮。但是在80年代，计算机的计算能力十分有限，很难训练出一个有效的模型来使用，所以导致了这种方式始终处于鸡肋的状态。再加上同一时期浅层方法的成功，如SVM(1995)，使得人们转为研究浅层的方法。
- 1998年CNN被提出，也应用到了邮政局的邮政编码识别，但是因为当时并不重视这种深度网络，导致并没有火起来。



深度学习发展简史（续2）

- 2006年，Hinton提出了DBN（深度信念网络），解决了更深层次的网络是可以通过一些策略更好的训练和实现，所以就引起了现在深度学习的第三次浪潮。



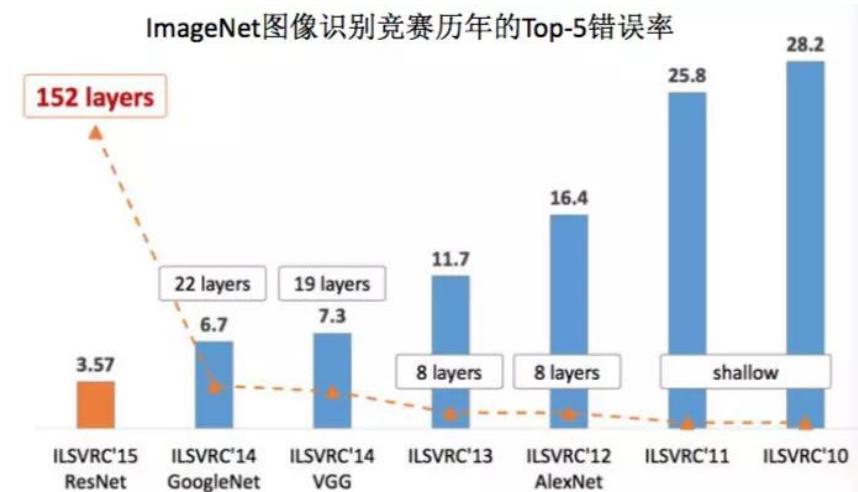
深度学习发展简史（续3）

- 相比而言，区别于传统的浅层学习，深度学习强调模型结构的深度，隐含层远远不止一层。通常来说，层数更多的网络，通常具有更强的抽象能力（即数据表征能力），也就能够产生更好的分类识别的结果。
- 2012年，杰弗里·辛顿（Geoffery Hinton）教授团队在ImageNet中首次使用深度学习完胜其他团队，那时网络层深度只有个位数。2014年，谷歌团队把网络做了22层，问鼎当时的ImageNet冠军。到了2015年，微软研究院团队设计的基于深度学习的图像识别算法ResNet，把网络层做到了152层。很快，在2016年，商汤科技更是叹为观止地把网络层做到了1207层。



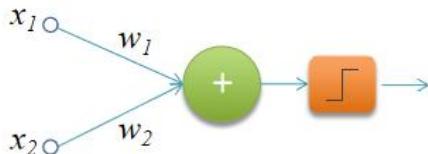
深度学习发展简史（续4）

- ImageNet Top5错误率和网络深度
 - 2012年冠军 ([AlexNet](#), top-5错误率16.4%，使用额外数据可达到15.3%，8层神经网络)
 - 2014年亚军 (VGGNet, top-5错误率7.3%，19层神经网络)，
2014年冠军 (InceptionNet, top-5错误率6.7%，22层神经网络)
 - 2015年的冠军 (ResNet, top-5错误率3.57%，152层神经网络)

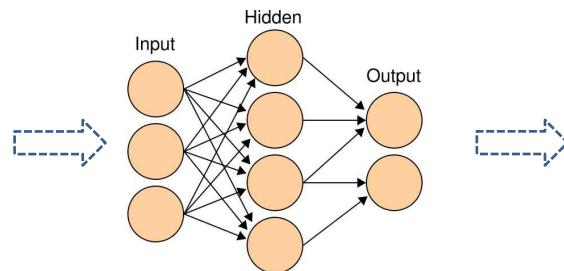


深度网络进化过程

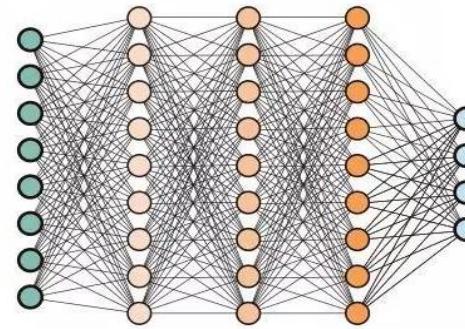
知识讲解



感知机
(神经元)



多层感知机
(神经网络)



深度神经网络



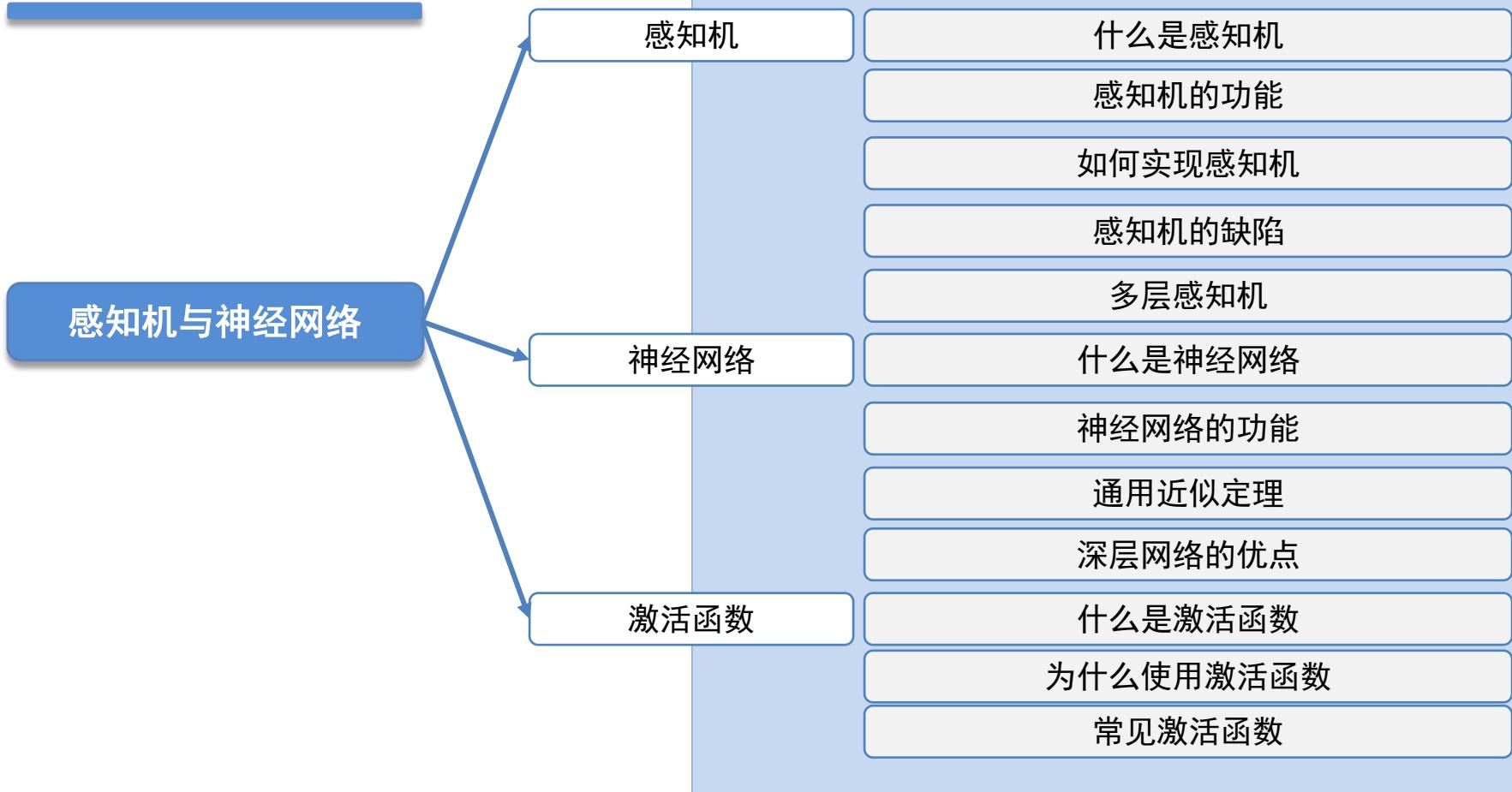
小结

- 时至今日，深度学习网络越来越深，应用越来越广，解决的问题越来越难，扮演的角色越来越重要。但万丈高楼平地起，让我们追根溯源，探索如何深度学习究竟是如何由一个简单的“单细胞”演化成复杂神经网络系统的。



感知机与神经网络

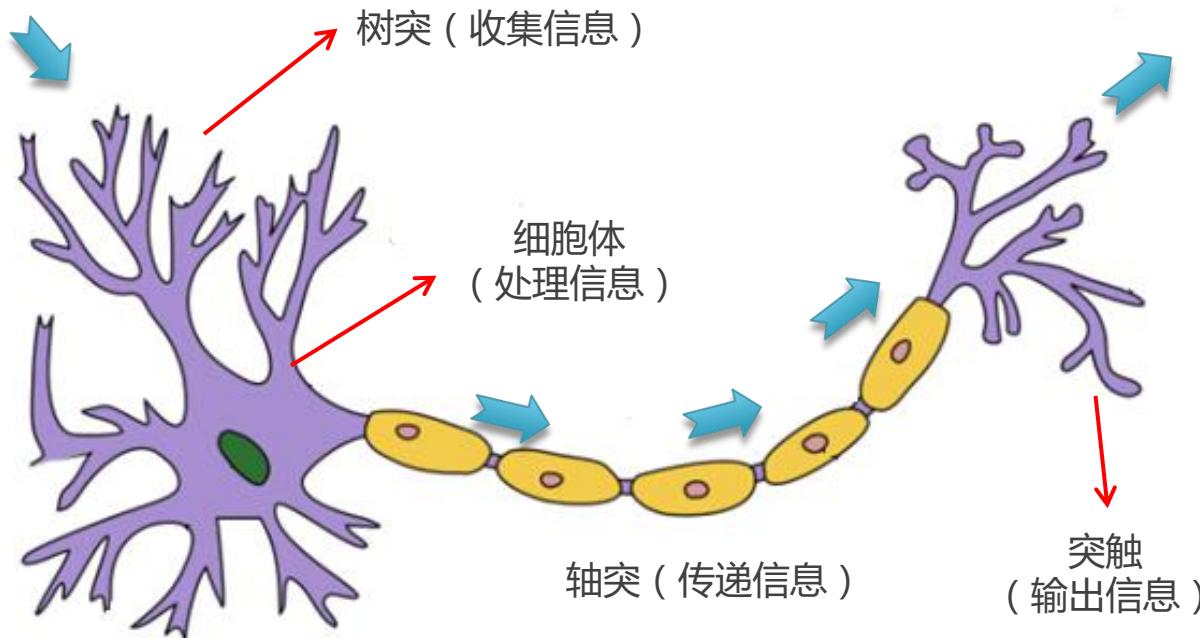
感知机与神经网络



感知机

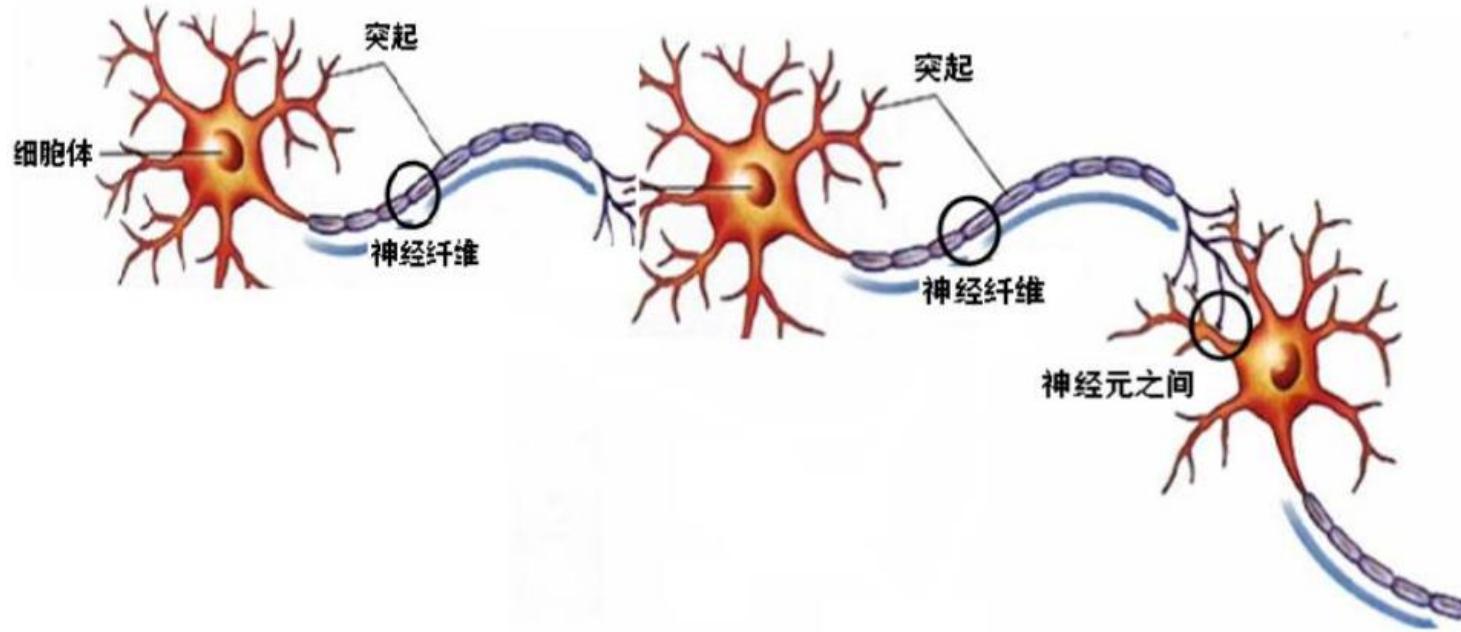
生物神经元

知识讲解



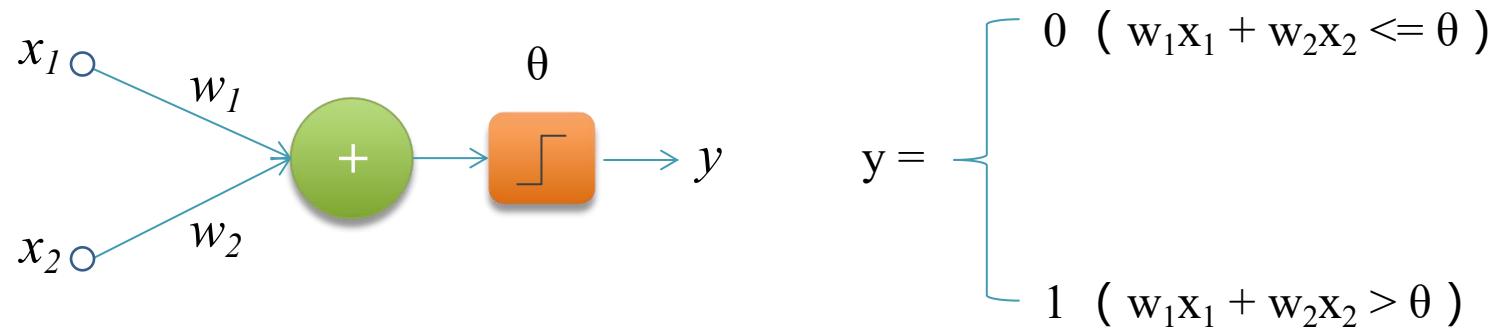
生物神经网络

知识讲解



什么是感知机

- 感知机（ Perceptron ），又称神经元（ Neuron ，对生物神经元进行了模仿）是神经网络（深度学习）的起源算法，1958年由康奈尔大学心理学教授弗兰克·罗森布拉特（ Frank Rosenblatt ）提出，它可以接收多个输入信号，产生一个输出信号。

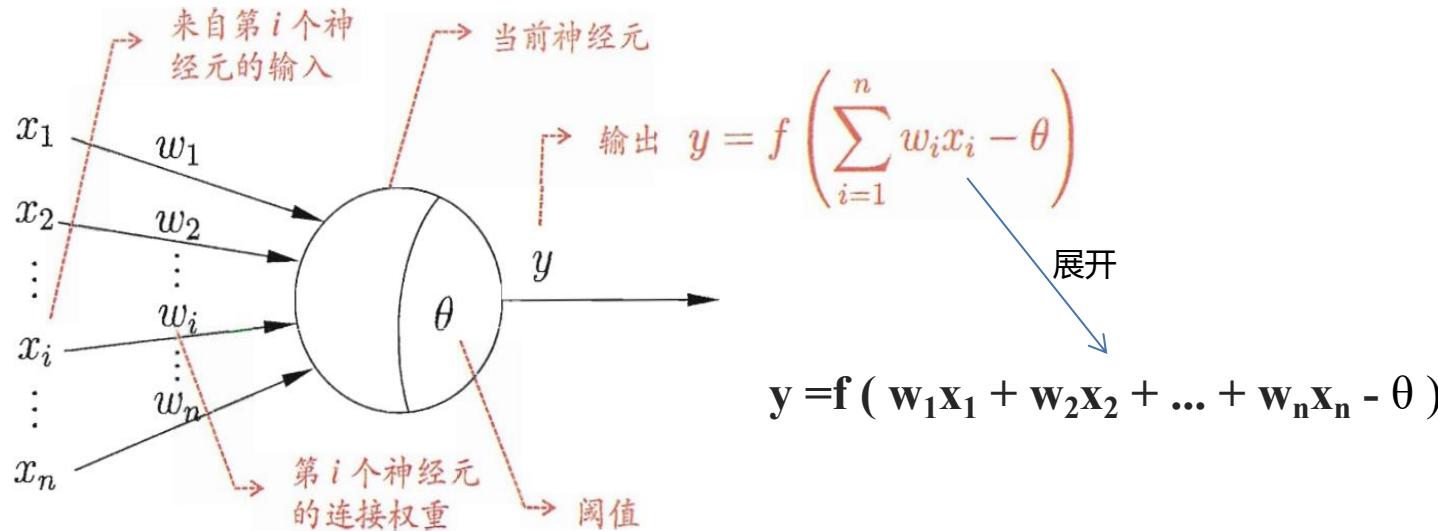


其中， x_1 和 x_2 称为输入， w_1 和 w_2 为权重， θ 为阈值， y 为输出



什么是感知机（续1）

- 神经元更通用的图形表示和表达式



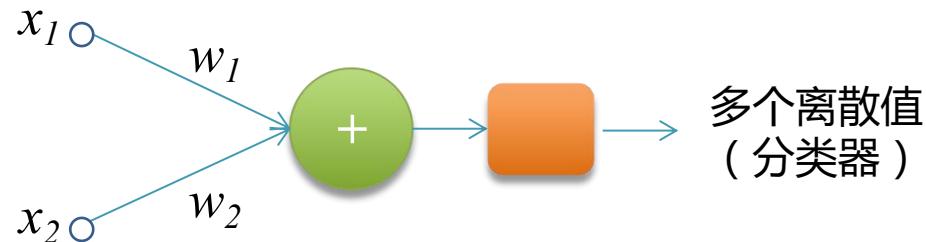
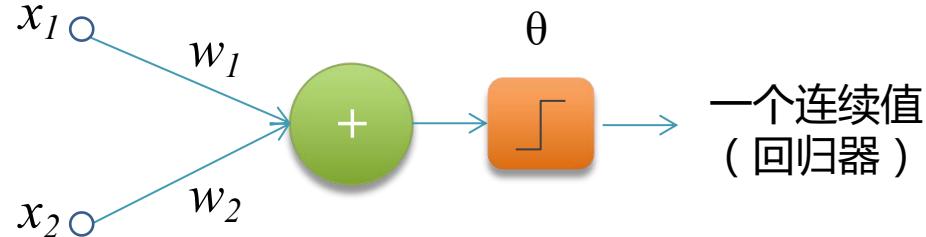
感知机的功能

- 作为分类器/回归器，实现自我学习
- 实现逻辑运算，包括逻辑和（ AND ）、逻辑或（ OR ）
- 组成神经网络



感知机的功能（续1）

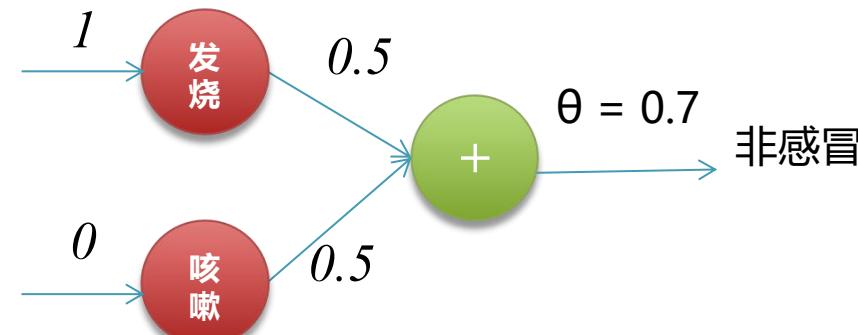
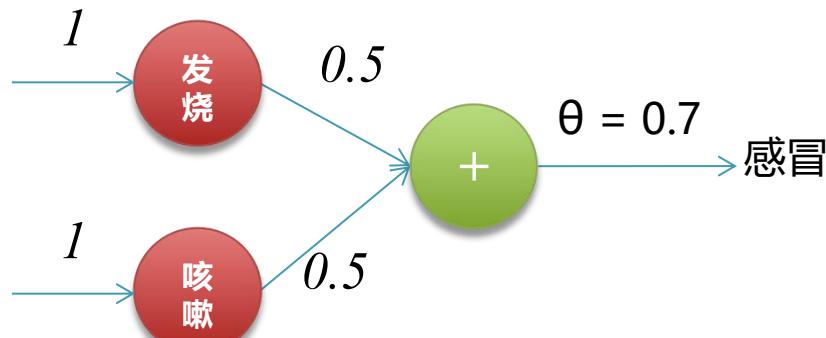
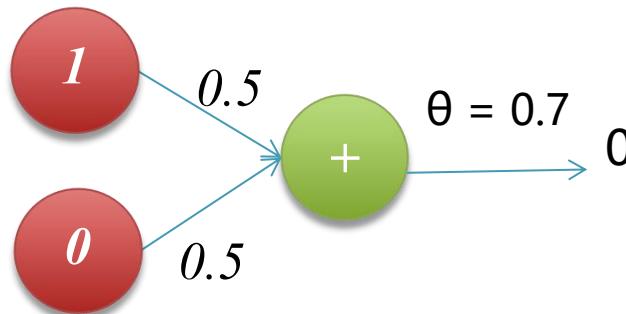
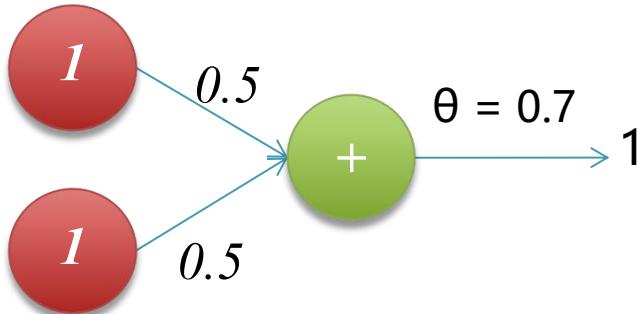
- 神经元作为回归器 / 分类器



感知机功能 (续2)

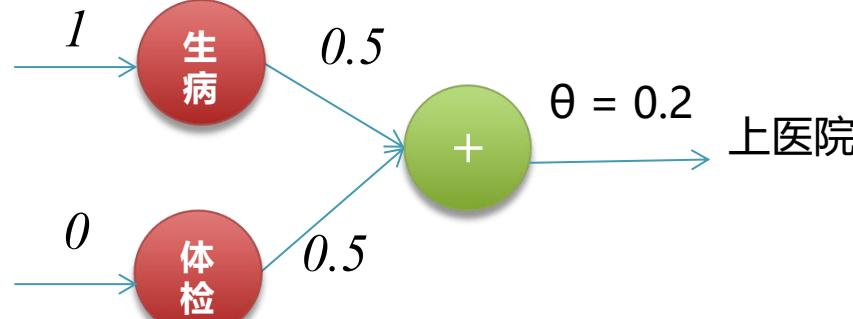
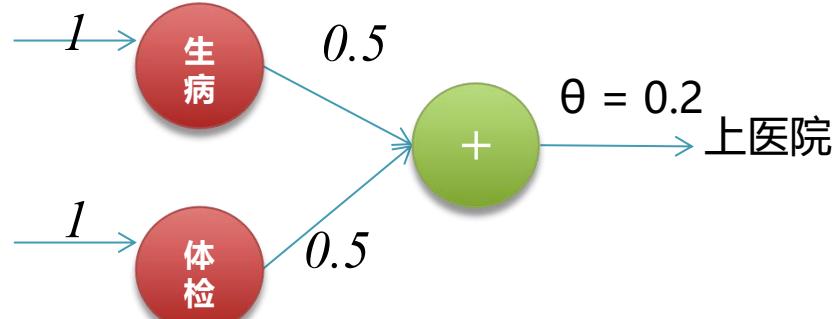
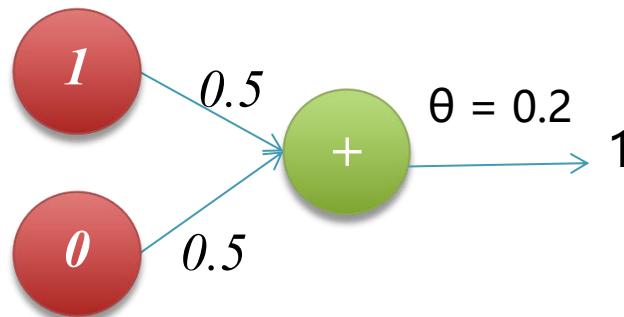
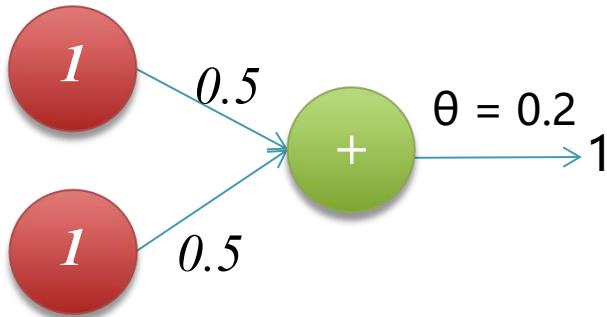
- 实现逻辑和

知识讲解



感知机功能 (续3)

- 实现逻辑或



知识讲解



如何实现感知机

- 实现逻辑和

```
4      # 实现逻辑和
5      def AND(x1, x2):
6          w1, w2, theta = 0.5, 0.5, 0.7
7          tmp = x1 * w1 + x2 * w2
8          if tmp <= theta:
9              return 0
10         else:
11             return 1
12
13     print(AND(1, 1))
14     print(AND(1, 0))
```



如何实现感知机（续）

- 实现逻辑或

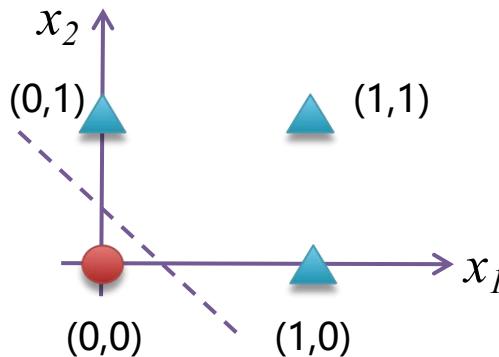
```
17 # 实现逻辑或
18 def OR(x1, x2):
19     w1, w2, theta = 0.5, 0.5, 0.2
20     tmp = x1 * w1 + x2 * w2
21     if tmp <= theta:
22         return 0
23     else:
24         return 1
25
26 print(OR(0, 1))
27 print(OR(0, 0))
```



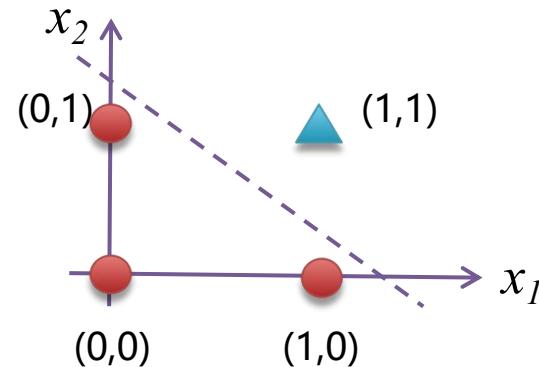
感知机的缺陷

- 感知机的局限在于无法处理“异或”问题

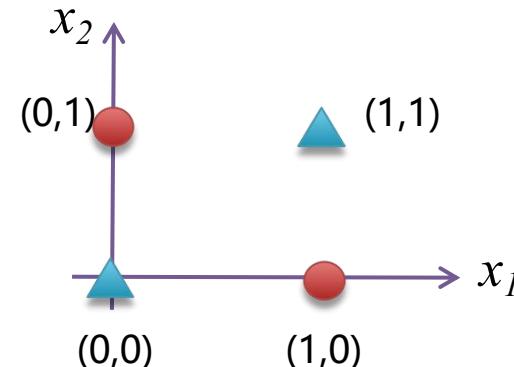
知识讲解



或门



与门

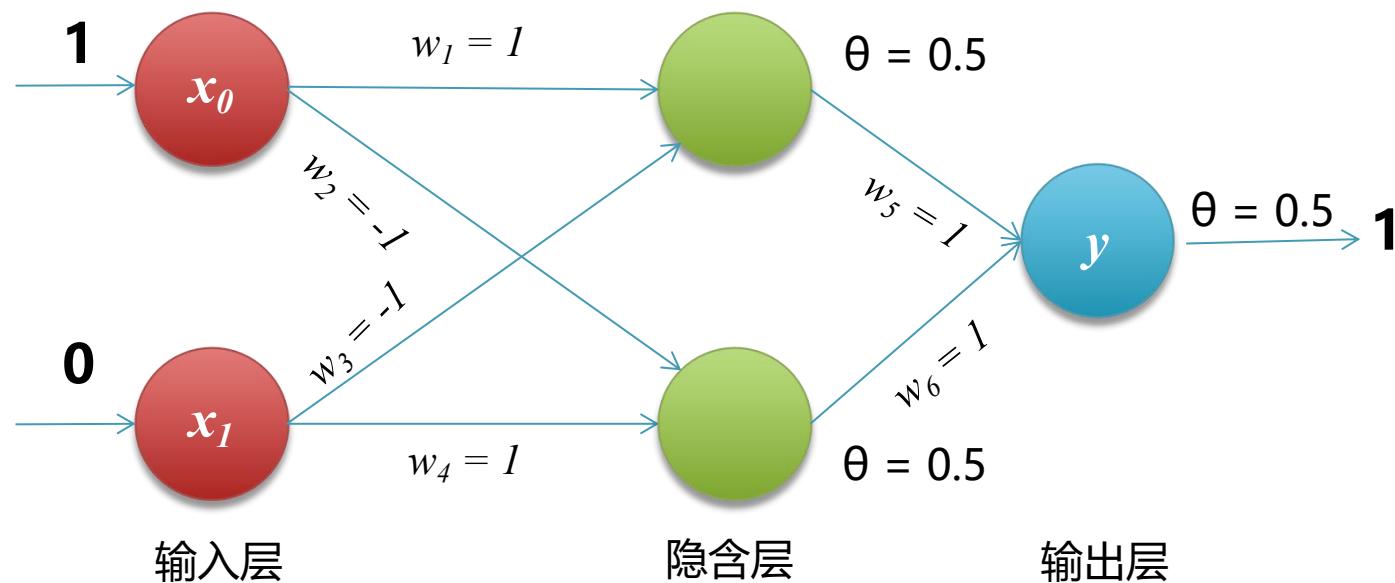


异或门



多层感知机

- 1975年，感知机的“异或”难题才被理论界彻底解决，即通过多个感知机组合来解决该问题，这种模型也叫多层感知机（Multi-Layer Perceptron，MLP）。如下图所示，神经元节点阈值均设置为0.5



多层感知机（续）

- 多层感知机解决异或门实现

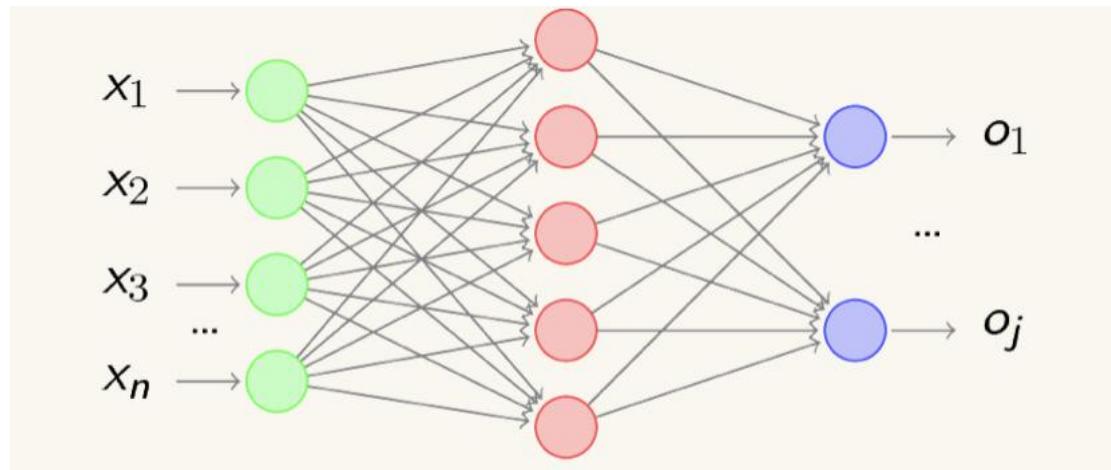
```
29  # 实现异或
30  def XOR(x1, x2):
31      s1 = not AND(x1, x2) # 与非门
32      s2 = OR(x1, x2)
33      y = AND(s1, s2)
34      return y
35
36  print(XOR(1, 0))
37  print(XOR(0, 1))
38  print(XOR(1, 1))
39  print(XOR(0, 0))
```



神经网络

什么是神经网络

- 感知机由于结构简单，完成的功能十分有限。可以将若干个感知机连在一起，形成一个级联网络结构，这个结构称为“多层前馈神经网络”（Multi-layer Feedforward Neural Networks）。所谓“前馈”是指将前一层的输出作为后一层的输入的逻辑结构。每一层神经元仅与下一层的神经元全连接。但在同一层之内，神经元彼此不连接，而且跨层之间的神经元，彼此也不相连。



神经网络的功能

- 1989年，奥地利学者库尔特·霍尼克（Kurt Hornik）等人发表论文证明，对于任意复杂度的连续波莱尔可测函数（Borel Measurable Function） f ，仅需要一个隐含层，只要这个隐含层包括足够多的神经元，前馈神经网络使用挤压函数（Squashing Function）作为激活函数，就可以以任意精度来近似模拟 f 。如果想增加 f 的近似精度，单纯依靠增加神经元的数目即可实现。
- 这个定理也被称为通用近似定理（Universal Approximation Theorem），该定理表明，前馈神经网在理论上可近似解决任何问题。



通用近似定理

现有目标函数 $f(x) = x^3 + x^2 - x - 1$ ，可以使用6个神经元进行模拟：

$$neruon_1(x) = \text{ReLU}(-5x - 7.7)$$

$$neruon_2(x) = \text{ReLU}(-1.2x - 1.3)$$

$$neruon_3(x) = \text{ReLU}(1.2x + 1)$$

$$neruon_4(x) = \text{ReLU}(1.2x - 0.2)$$

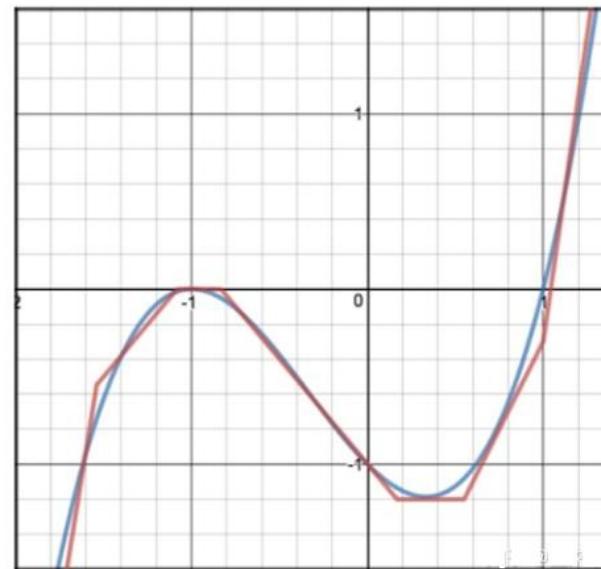
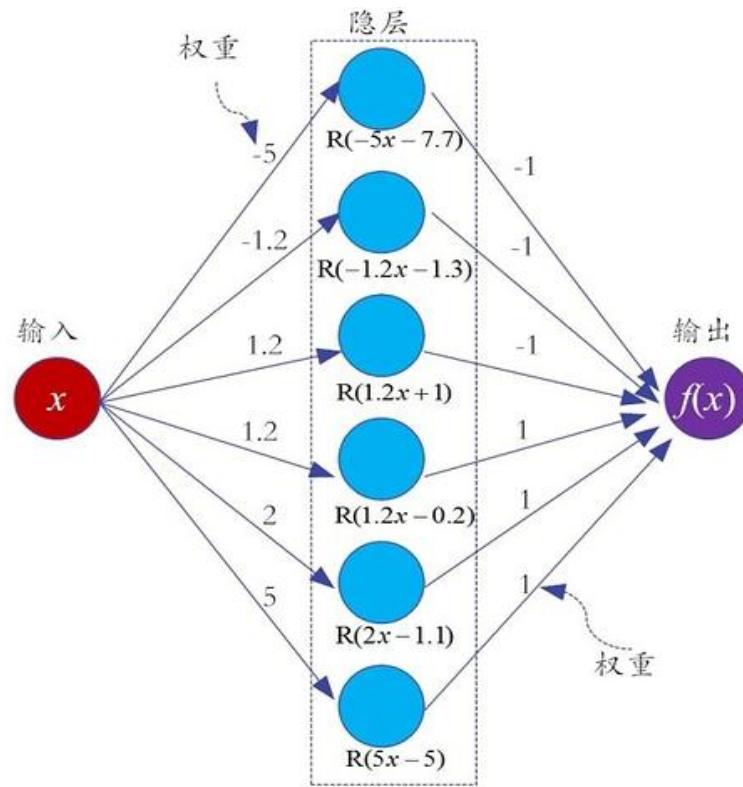
$$neruon_5(x) = \text{ReLU}(2x - 1.1)$$

$$neruon_6(x) = \text{ReLU}(5x - 5)$$



通用近似定理（续1）

知识讲解



通用近似定理（续2）

$f($



)



猫

$f($



)



Hello, deep learning

$f($ “哈哈哈哈” $)$



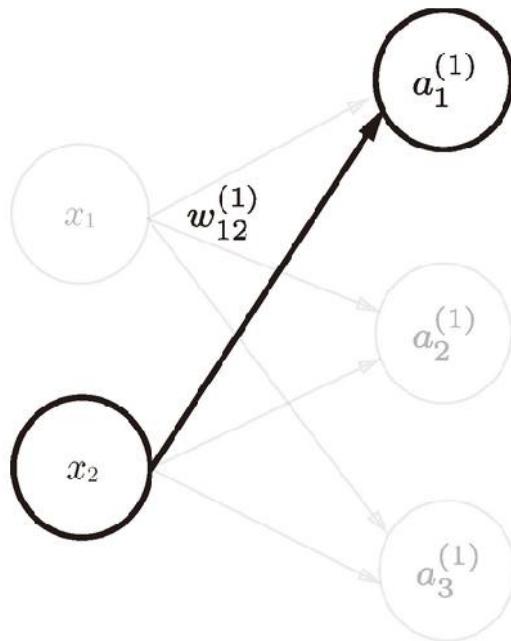
高兴

深层网络的优点

- 其实，神经网络的结构还有另外一个“进化”方向，那就是朝着“纵深”方向发展，也就是说，减少单层的神经元数量，而增加神经网络的层数，也就是“深”而“瘦”的网络模型。
- 微软研究院的科研人员就以上两类网络性能展开了实验，实验结果表明：增加网络的层数会显著提升神经网络系统的学习性能。



多层神经网络计算公式



$w^{(1)}$

第1层的权重

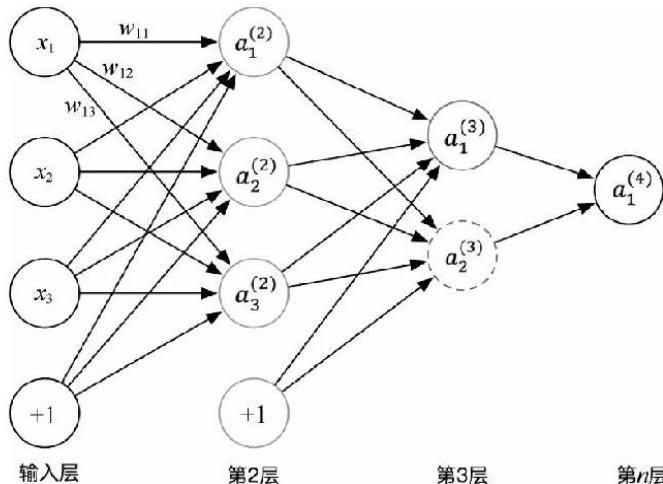
前一层的第2个神经元

后一层的第1个神经元

多层神经网络计算公式(续)

- 以下是一个多层神经网络及其计算公式

知识讲解



$$a_1^{(2)} = f(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3 + b_{11}^{(1)})$$

$$a_2^{(2)} = f(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + b_{12}^{(1)})$$

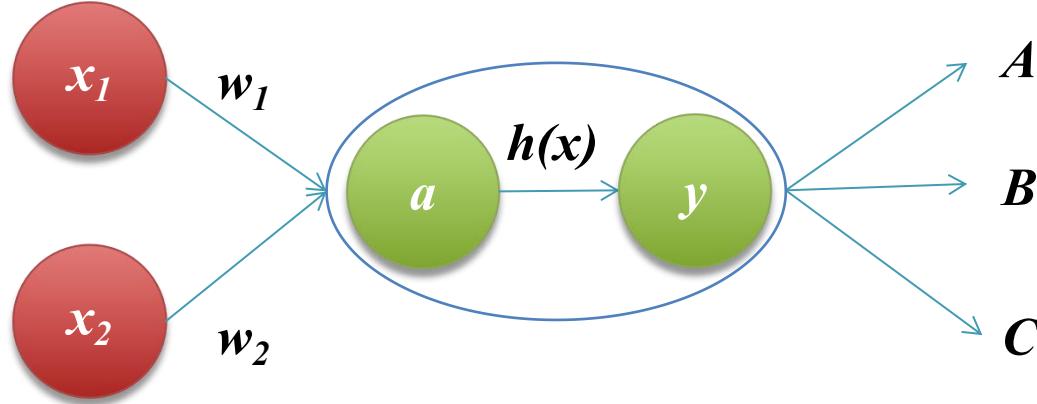
$$a_3^{(2)} = f(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_{13}^{(1)})$$

$$h_{W,b}(x) = a_1^{(3)} = f(w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}a_3^{(2)} + b_{21}^{(2)})$$

激活函数

什么是激活函数

- 在神经网络中，将输入信号的总和转换为输出信号的函数被称为激活函数（activation function）



为什么使用激活函数

- 激活函数将多层感知机输出转换为非线性，使得神经网络可以任意逼近任何非线性函数，这样神经网络就可以应用到众多的非线性模型中。
- 如果一个多层网络，使用连续函数作为激活函数的多层网络，称之为“神经网络”，否则称为“多层感知机”。所以，激活函数是区别多层感知机和神经网络的依据。

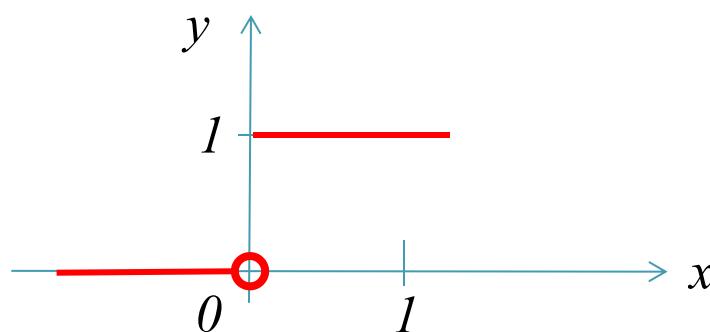


常见激活函数

- 阶跃函数

- 阶跃函数 (Step Function) 是一种特殊的连续时间函数，是一个从0跳变到1的过程，函数形式与图像：

$$f(x) = \begin{cases} 1 & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$

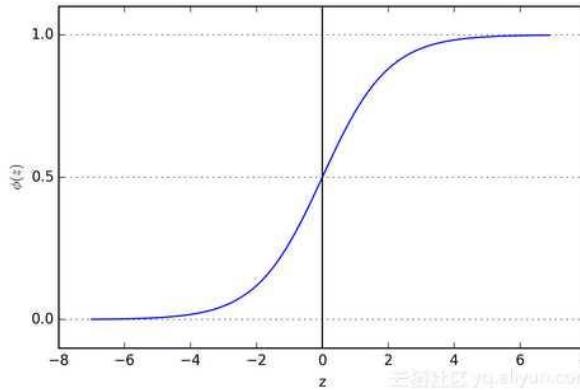


常见激活函数（续1）

- sigmoid函数

- sigmoid函数也叫Logistic函数，用于隐层神经元输出，取值范围为(0,1)，它可以将一个实数映射到(0,1)的区间，可以用来做二分类

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



优点：平滑、易于求导

缺点：激活函数计算量大，反向传播求误差梯度时，求导涉及除法；反向传播时，很容易就会出现梯度消失的情况，从而无法完成深层网络的训练

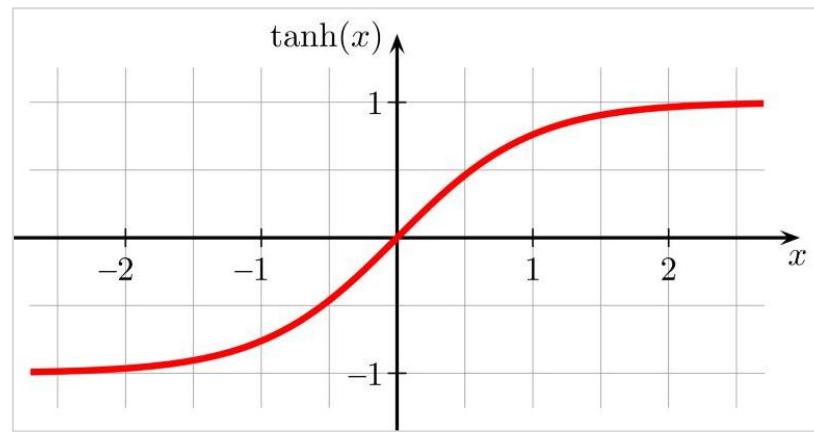


常见激活函数（续2）

- tanh双曲正切函数

知识讲解

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$



优点：平滑、易于求导；输出均值为0，收敛速度要比sigmoid快，从而可以减少迭代次数

缺点：梯度消失

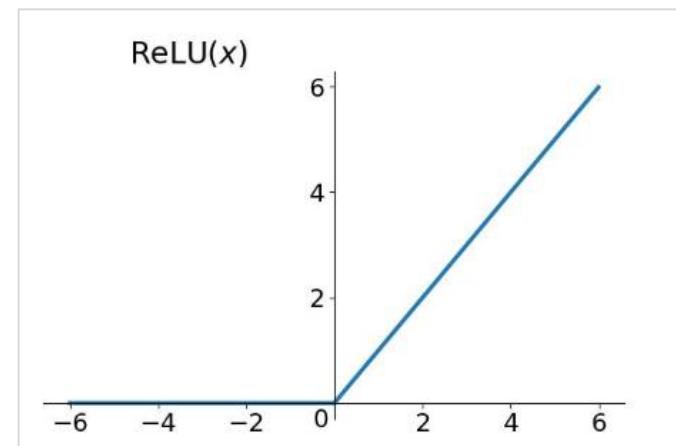


常见激活函数（续3）

- ReLU (Rectified Linear Units, 修正线性单元)

知识讲解

$$f(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$



优点：(1) 更加有效率的梯度下降以及反向传播，避免了梯度爆炸和梯度消失问题
(2) 计算过程简单



常见激活函数（续4）

- Softmax

Softmax函数定义如下，其中 v_i 是分类器前级输出单元的输出。 i 表示类别索引，总的类别个数为 C 。 s_i 表示的是当前元素的指数与所有元素指数和的比值。通过Softmax函数就可以将多分类的输出数值转化为相对概率，而这些值的累和为1。

$$s_i = \frac{e^{v_i}}{\sum_i^C e^{v_i}}$$



小结

- 本章节介绍了深度学习一些重要的基本概念，需要理解并熟练掌握。同时，还介绍了如何从简单的感知机逐步演化到复杂的神经网络。重要的概念有：
 - ✓ **感知机。**接收多个输入信号，产生一个输出信号，无法解决异或问题
 - ✓ **多层感知机。**将多个感知机组合
 - ✓ **多层前馈网络。**若干个感知机组合成若干层的网络，上一层输出作为下一层输入
 - ✓ **激活函数。**将计算结果转换为输出的值，包括阶跃函数、sigmoid、tanh、ReLU



损失函数与梯度下降

损失函数与梯度下降

损失函数

什么是损失函数

损失函数的作用

常用的损失函数

梯度下降

什么是梯度

梯度下降

导数与偏导数

学习率

梯度递减训练法则

梯度下降算法

损失函数

什么是损失函数

- 损失函数 (Loss Function) , 也有称之为代价函数 (Cost Function) , 用
来度量预测值和实际值之间的差异。

$$E = y - y'$$



损失函数的作用

- 度量决策函数 $f(x)$ 和实际值之间的差异。
- 作为模型性能参考。损失函数值越小，说明预测输出和实际结果（也称期望输出）之间的差值就越小，也就说明我们构建的模型越好。学习的过程，就是不断通过训练数据进行预测，不断调整预测输出与实际输出差异，使的损失值最小的过程。

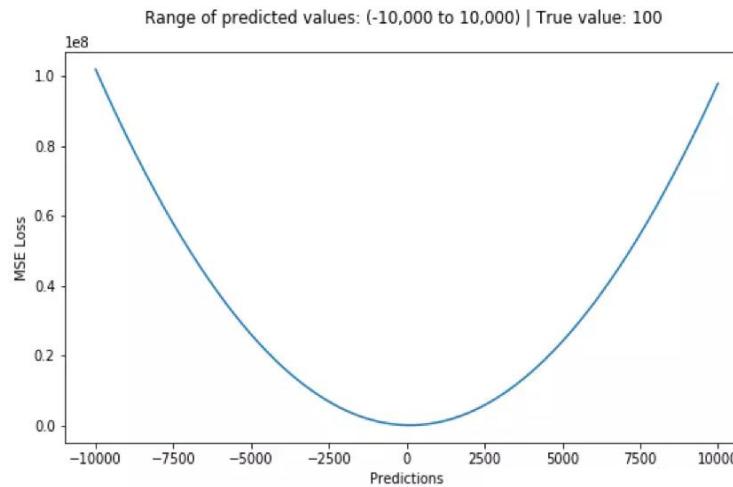


常用损失函数

- 均方误差 (Mean square error) 损失函数。均方误差是回归问题常用的损失函数，它是预测值与目标值之间差值的平方和，其公式和图像如下所示：

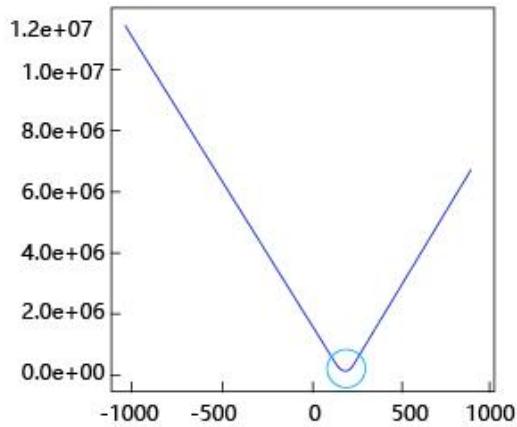
知识讲解

$$MSE = \frac{\sum_{i=1}^n (y_i - y_i^p)^2}{n}$$



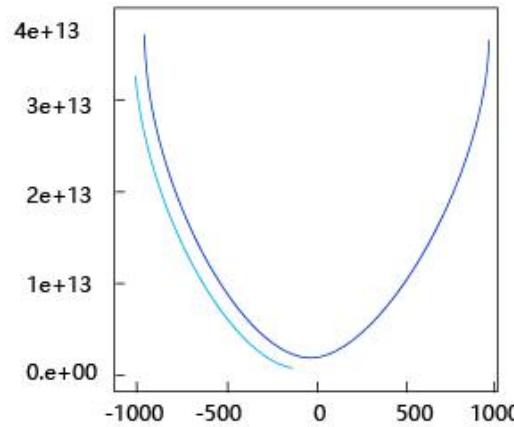
常用损失函数（续1）

绝对值误差



绝对值误差的Loss函数（不可微）

均方误差



均方误差的Loss函数（可微）

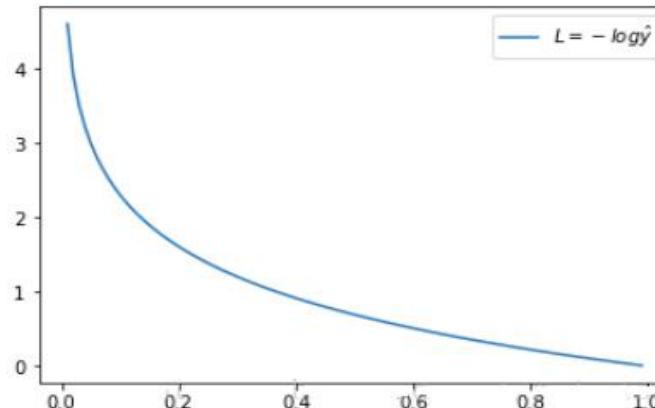
- 曲线的最低点是可导的
- 越接近最低点，曲线的坡度逐渐放缓，有助于通过当前的梯度来判断接近最低点的程度（是否逐渐减少步长，以免错过最低点）



常用损失函数（续2）

- 交叉熵 (Cross Entropy)。交叉熵是Shannon信息论中一个重要概念，主要用于度量两个概率分布间的差异性信息，在机器学习中用来作为分类问题的损失函数。假设有两个概率分布， t_k 与 y_k ，其交叉熵函数公式及图形如下所示：

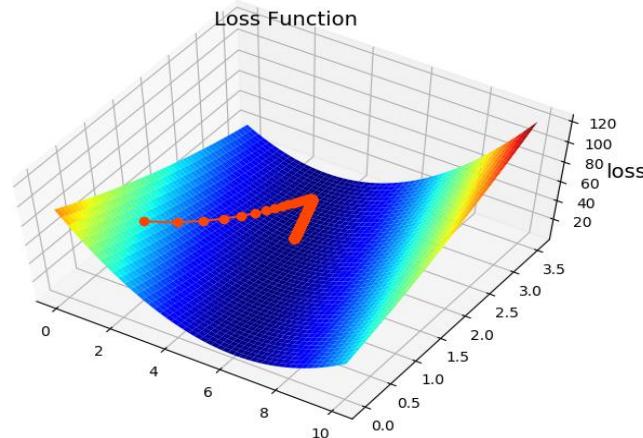
$$E = - \sum_k t_k \log y_k$$



梯度下降

什么是梯度

- 梯度 (gradient) 是一个向量 (矢量, 有方向), 表示某一函数在该点处的方向导数沿着该方向取得最大值, 即函数在该点处沿着该方向 (此梯度的方向) 变化最快, 变化率最大。 损失函数沿梯度相反方向收敛最快 (即能最快找到极值点)。当梯度向量为零 (或接近于零), 说明损失函数到达一个极小值点, 模型准确度达到一个极大值点。



梯度下降

- 通过损失函数，我们将“寻找最优参数”问题，转换为了“寻找损失函数最小值”问题。 寻找步骤：

- (1) 损失是否足够小？如果不是，计算损失函数的梯度。
- (2) 按梯度的反方向走一小步，以缩小损失。
- (3) 循环到(1)。

这种按照负梯度不停地调整函数权值的过程就叫作“梯度下降法”。通过这样的方法，改变每个神经元与其他神经元的连接权重及自身的偏置，让损失函数的值下降得更快，进而将值收敛到损失函数的某个极小值。



导数与偏导数

- 导数的定义

所谓导数，就是用来分析函数“变化率”的一种度量。其公式为：

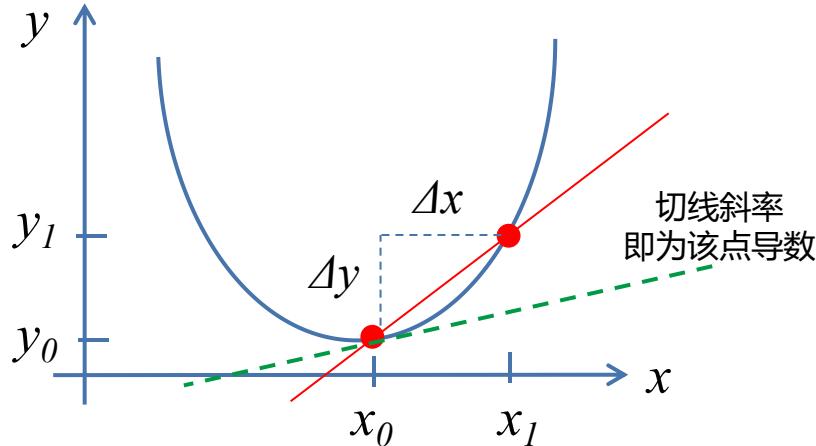
$$f'(x_0) = \lim_{\Delta x \rightarrow 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$



导数与偏导数（续1）

- 导数的含义：反映变化的剧烈程度（变化率）

知识讲解



例如： $y = x^2 + 1$

$$x_0=1 \quad y_0=2$$

$$x_1=1.0001 \quad y_1=2.00020001$$

$$\Delta y / \Delta x \approx 0.0002 / 0.0001 = 2$$

$$x_0=2 \quad y_0=5$$

$$x_1=2.0001 \quad y_1=5.00040001$$

$$\Delta y / \Delta x \approx 0.0004 / 0.0001 = 4$$

导数与偏导数（续2）

• 偏导数

“偏导”的英文本意是“partial derivatives”（表示局部导数）。对于多维变量函数而言，当求某个变量的导数时，就是把其他变量视为常量，然后对整个函数求其导数（相比于全部变量，这里只求一个变量，即为“局部”）。例如有函数：

$$f = x^2 + 3xy + y^2 + z^3$$

则，对x, y, z分别求偏导公式为：

$$\frac{\partial f}{\partial x} = 2x + 3y$$



y, z为常量

$$\frac{\partial f}{\partial y} = 3x + 2y$$



x, z为常量

$$\frac{\partial f}{\partial z} = 3z^2$$

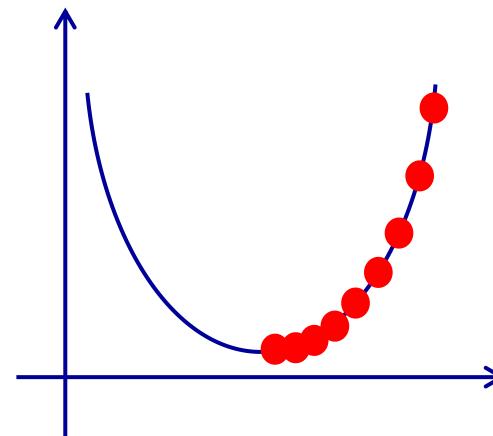
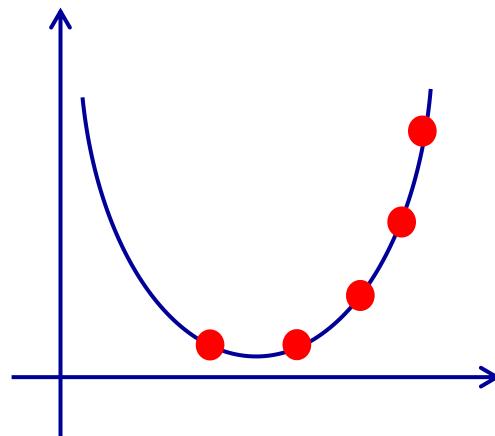


x, y为常量



学习率

- 如果在梯度下降过程中，每次都按照相同的步幅收敛，则可能错过极值点（下图左），所以设置一个比率，用来控制调整的步幅大小，这个比率称之为“学习率”（下图右）。



梯度递减训练法则

- 神经网络中的权值参数是非常多的，因此针对损失函数E的权值向量的梯度如以下公式所示：

$$\nabla E(\vec{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

$\nabla E(\vec{w})$ 表示损失函数E的梯度，它本身也是一个向量，它的多个维度分别由损失函数E对多个权值参数 w_i 求偏导所得。当梯度被解释为权值空间中的一个向量时，它就确定了E陡峭上升的方向，那么梯度递减的训练法则就如下公式所示：

$$w_i \leftarrow w_i + \Delta w_i \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$



梯度下降算法

• 批量梯度下降

批量梯度下降法 (Batch Gradient Descent , BGD) 是最原始的形式，它是指在每一次迭代时使用所有样本来进行梯度的更新。

➤ 优点：

- ✓ 一次迭代是对所有样本进行计算，此时利用矩阵进行操作，实现了并行。
- ✓ 由全数据集确定的方向能够更好地代表样本总体，从而更准确地朝向极值所在的方向。

当目标函数为凸函数时，BGD一定能够得到全局最优。

➤ 缺点：

- ✓ 当样本数目 m 很大时，每迭代一步都需要对所有样本计算，训练过程会很慢。



梯度下降算法（续1）

• 随机梯度下降

随机梯度下降法（Stochastic Gradient Descent，SGD）每次迭代使用一个样本来对参数进行更新，使得训练速度加快。

➤ 优点：

- ✓ 由于不是在全部训练数据上的损失函数，而是在每轮迭代中，随机优化某一条训练数据上的损失函数，这样每一轮参数的更新速度大大加快。

➤ 缺点：

- ✓ 准确度下降。由于即使在目标函数为强凸函数的情况下，SGD仍旧无法做到线性收敛。
- ✓ 可能会收敛到局部最优，由于单个样本并不能代表全体样本的趋势。
- ✓ 不易于并行实现。



梯度下降算法（续2）

• 小批量梯度下降

小批量梯度下降（ Mini-Batch Gradient Descent, MBGD ）是对批量梯度下降以及随机梯度下降的一个折中办法。其思想是：每次迭代 使用指定个（ batch_size ）样本来对参数进行更新。

➤ 优点：

- ✓ 通过矩阵运算，每次在一个batch上优化神经网络参数并不会比单个数据慢太多。
- ✓ 每次使用一个batch可以大大减小收敛所需要的迭代次数，同时可以使收敛到的结果更加接近梯度下降的效果。

➤ 缺点：

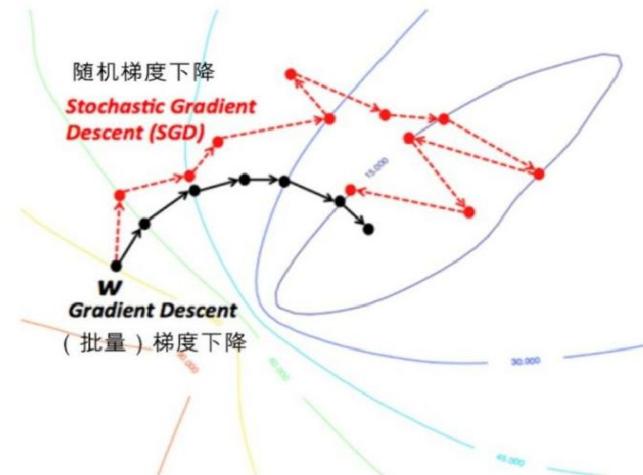
- ✓ batch_size的不当选择可能会带来一些问题。



梯度下降算法（续3）

• 几种梯度下降算法收敛比较

- 批量梯度下降稳健地向着最低点前进的
- 随机梯度下降震荡明显，但总体上向最低点逼近
- 小批量梯度下降位于两者之间



小结

- 本章节介绍了损失函数与梯度下降概念与算法
 - ✓ 损失函数。 用于度量预测值和期望值之间的差异，根据该差异值进行参数调整
 - ✓ 梯度下降。 用于以最快的速度、最少的步骤快速找到损失函数的极小值

反向传播算法

反向传播算法

反向传播算法

什么是正向传播网络

什么是反向传播

为什么需要反向传播

图解反向传播

反向传播计算

链式求导法则

案例1：通过反向传播计算偏导数

什么是正向传播网络

- 前一层的输出作为后一层的输入的逻辑结构，每一层神经元仅与下一层的神经元全连接，通过增加神经网络的层数虽然可为其提供更大的灵活性，让网络具有更强的表征能力，也就是说，能解决的问题更多，但随之而来的数量庞大的网络参数的训练，一直是制约多层神经网络发展的一个重要瓶颈。



什么是反向传播

- 反向传播（ Backpropagation algorithm ）全称“误差反向传播”，是在深度神经网络中，根据输出层输出值，来反向调整隐藏层权重的一种方法。



为什么需要反向传播

- 为什么不直接使用梯度下降而使用反向传播方式更新权重呢？
- 梯度下降应用于有明确求导函数的情况，或者可以求出误差的情况（比如线性回归），我们可以把它看做没有隐藏层的网络。但对于多个隐藏层的神经网络，输出层可以直接求出误差来更新参数，但隐藏层的误差是不存在的，因此不能对它直接应用梯度下降，而是先将误差反向传播至隐藏层，然后再应用梯度下降。



反向传播算法极简史

- 1974年，哈佛大学沃伯斯博士在他的博士论文中，首次提出了通过误差的反向传播来训练人工神经网络，以解决神经网络数量庞大的参数训练问题。但是，沃伯斯的工作并没有得到足够的重视，因为当时神经网络正陷入低潮，可谓“生不逢时”。
- 1986年，由杰弗里·辛顿（Geoffrey Hinton）和大卫·鲁姆哈特（David Rumelhart）等人在著名学术期刊Nature（自然）上发表了论文“借助误差反向传播算法的学习表征（Learning Representations by Back-propagating errors）”，系统而简洁地阐述了反向传播算法在神经网络模型上的应用。反向传播算法非常好使，它直接把纠错的运算量降低到只和神经元数目本身成正比的程度。



反向传播算法极简史（续）

- 后来，沃伯斯得到了IEEE（电气电子工程师学会）神经网络分会的先驱奖；Geoffrey Hinton与Yoshua Bengio、Yann LeCun（合称“深度学习三巨头”）共同获得了2018年的图灵奖

知识讲解



沃伯斯



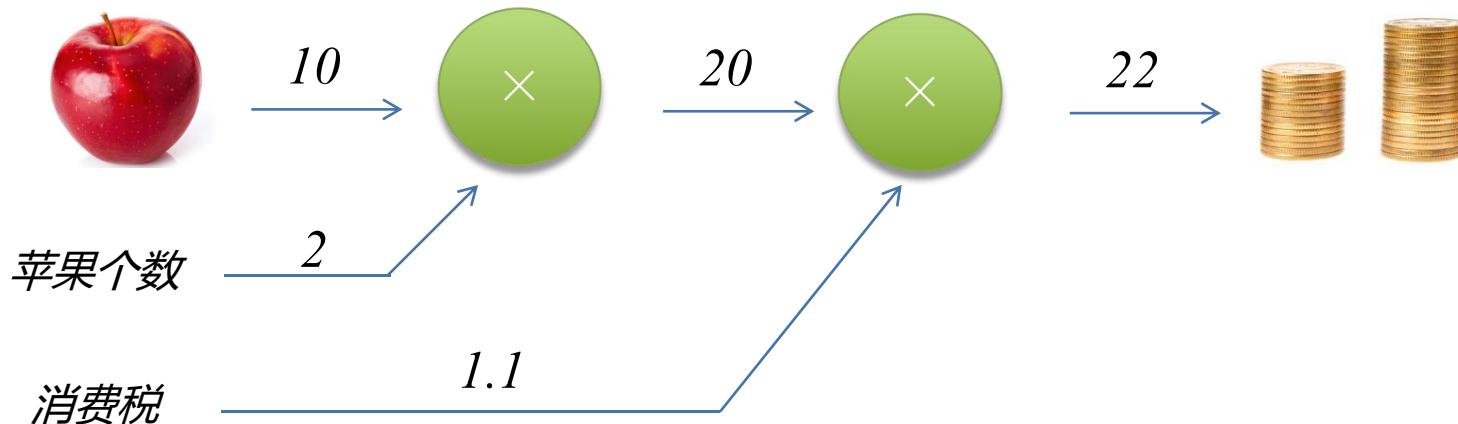
杰弗里·辛顿



图解反向传播

- 问题：Tom在超市买了2个苹果，每个10元，消费税10%，请计算应该支付的金额

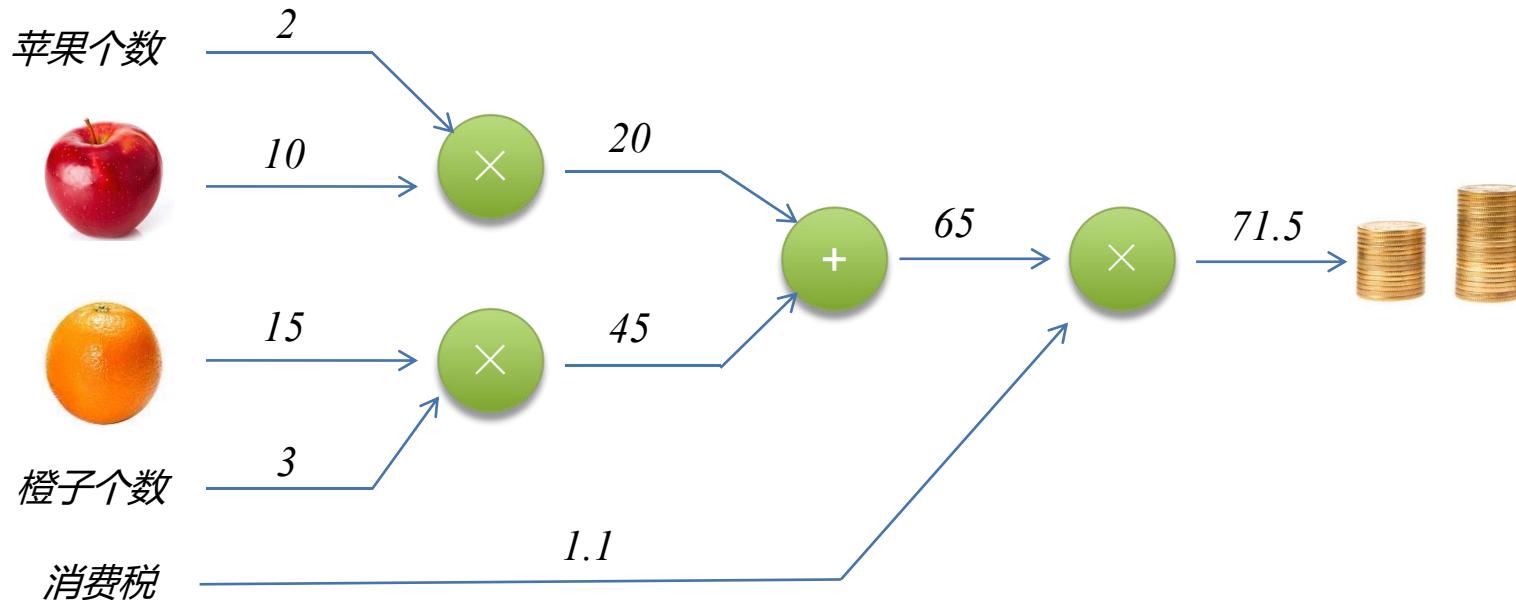
知识讲解



图解反向传播（续1）

- 问题：Tom在超市买了2个苹果，3个橙子，其中苹果每个10元，橙子每个15元，消费税10%，请计算应该支付的金额

知识讲解

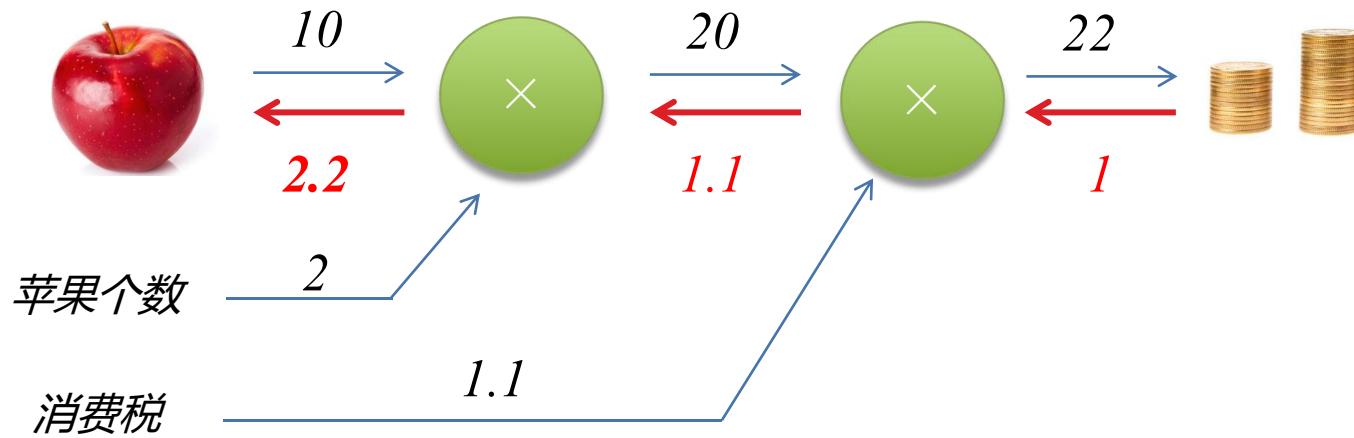


图解反向传播（续2）

- 问题：Tom在超市买了2个苹果，每个10元，消费税10%，请计算苹果价格上涨会在多大程度上影响支付金额（即求“支付金额关于苹果的价格的导数”）。设苹果的价格为x，支付金额为L，则相当于 $\frac{\partial L}{\partial x}$ 。这个导数的值表示当苹果的价格稍微上涨时，支付金额会增加多少。

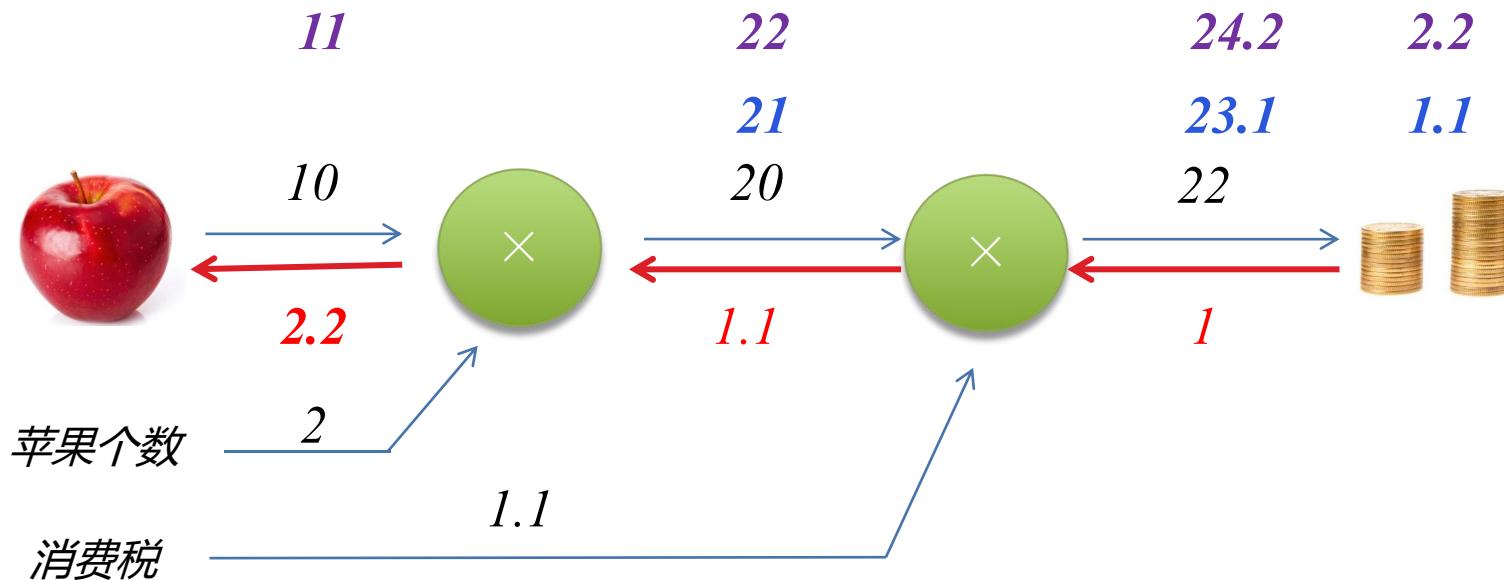
知识讲解

苹果价格增长1
支付金额增加2.2



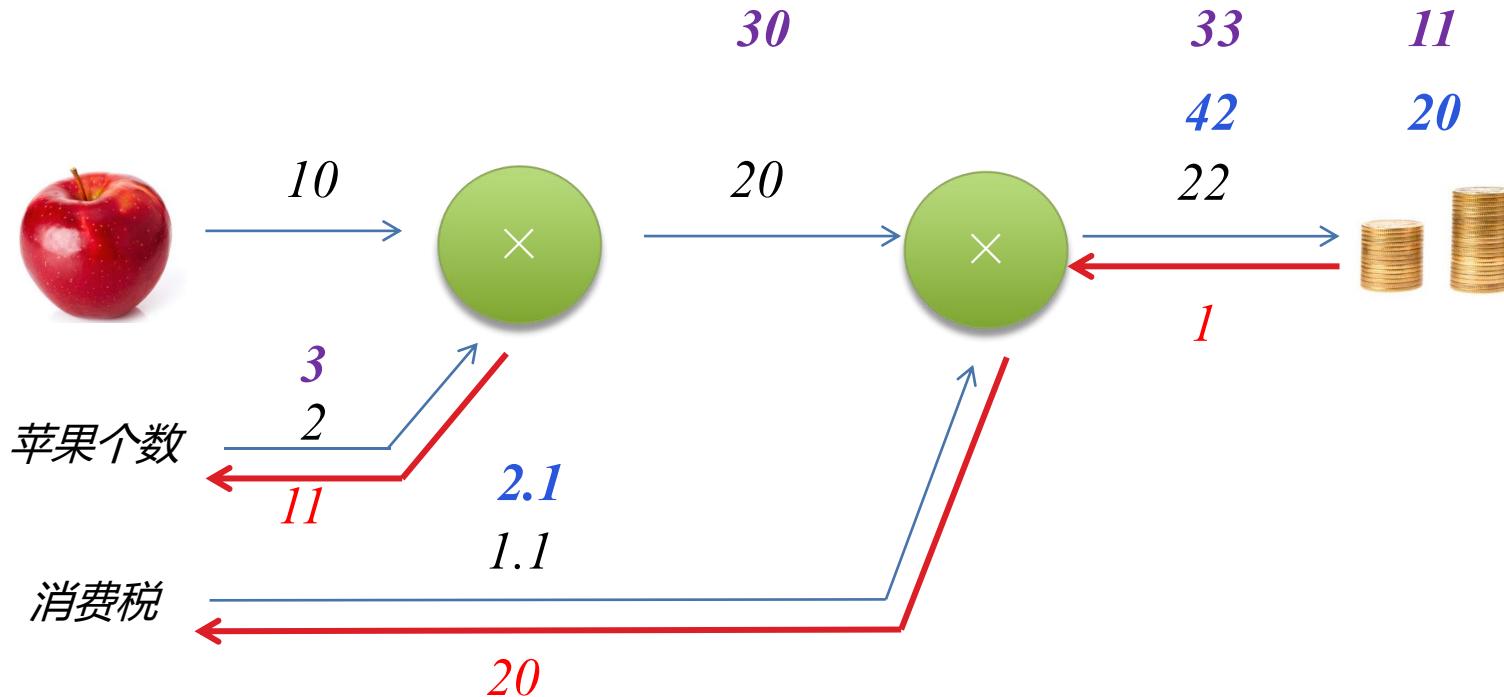
图解反向传播（续3）

知识讲解



图解反向传播（续4）

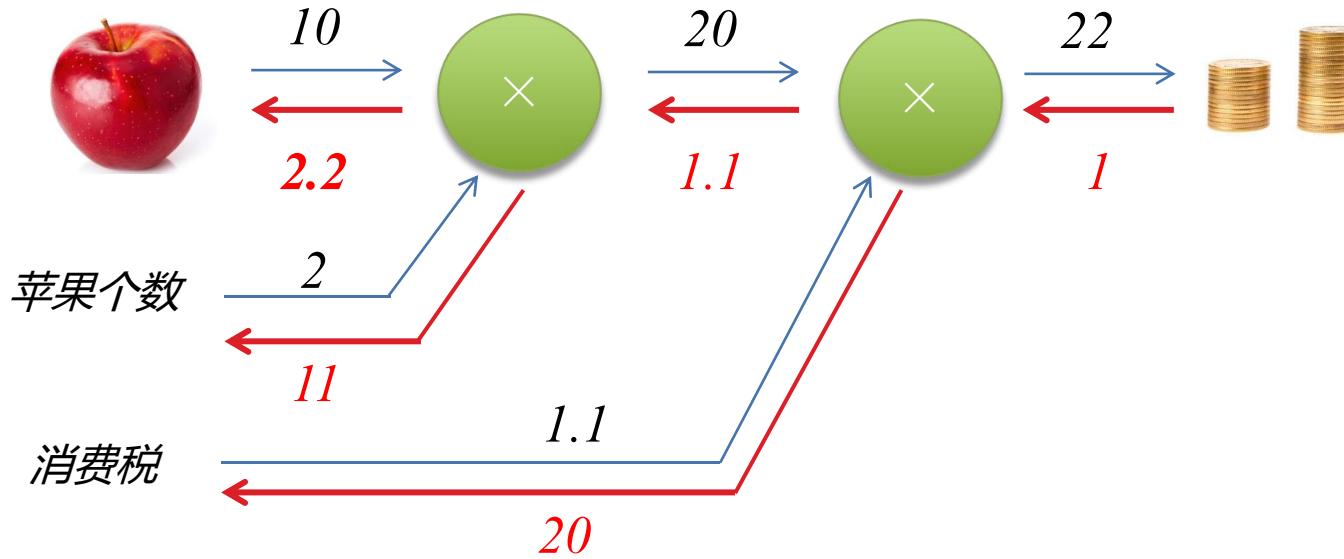
知识讲解



图解反向传播（续5）

苹果价格的导数为2.2，苹果个数导数为11，消费税导数为20，可以解释为：苹果价格、苹果个数或消费税增加相同的值，分别对支付金额产生2.2倍、11倍、20倍的影响

知识讲解



反向传播计算

- 考虑函数 $y = f(x)$, 输出为E, 反向传播的计算顺序是, 将信号E乘以节点的局部导数(偏导数), 传递给前面的节点, 这样可以高效地求出导数的值。

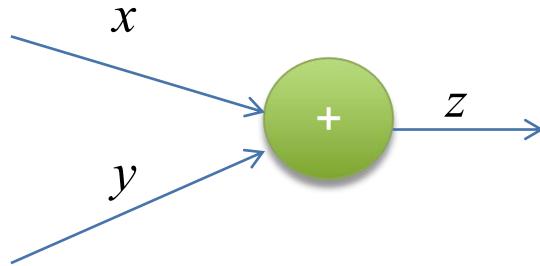
知识讲解



反向传播计算（续1）

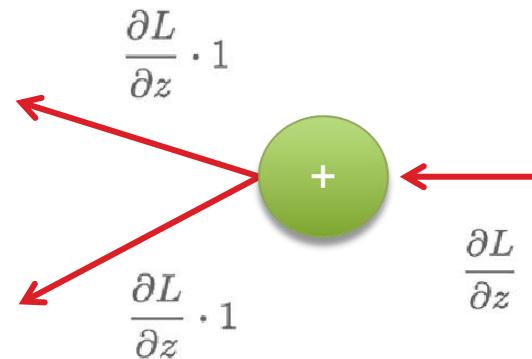
- 加法节点反向传播计算

知识讲解



正向传播

偏导数均为1



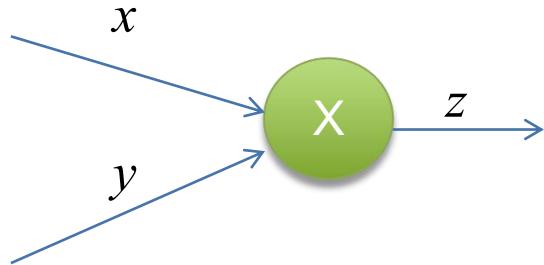
反向传播

将上游值原封不动传递给下游



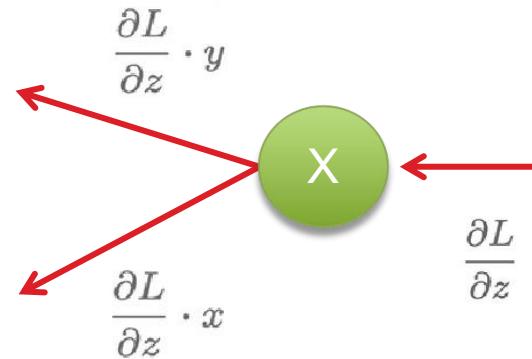
反向传播计算（续2）

- 乘法节点反向传播计算



正向传播

偏导为 x, y



反向传播

将上游信号和正向传播“翻转值”传递给下游



链式求导法则

- 考虑如下复合函数

$$z = t^2$$

$$t = x + y$$

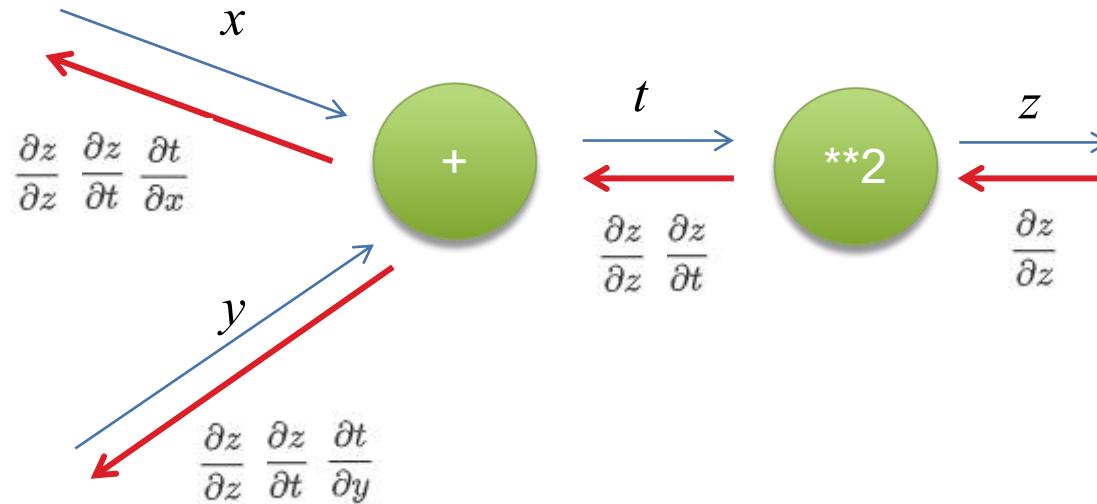
z 关于 x 的导数(x 变化对 z 的影响率)可以表示为：

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \frac{\partial t}{\partial x} \longrightarrow \begin{aligned} \frac{\partial z}{\partial t} &= 2t \\ \frac{\partial t}{\partial x} &= 1 \end{aligned} \longrightarrow \frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x + y)$$

这称之为“链式求导法则”



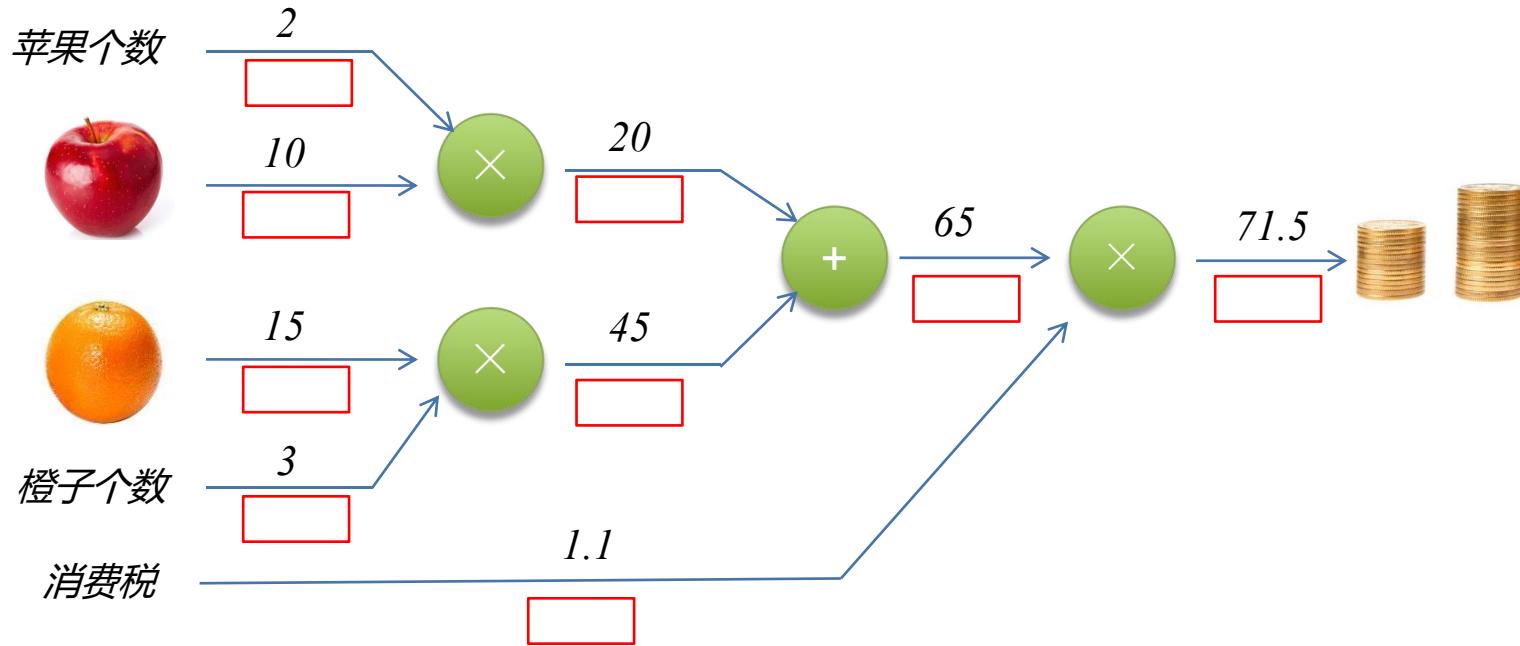
链式求导法则（续）



案例1：通过反向传播计算偏导数

- 问题：苹果、橙子价格和个数以及税率如下图所示，利用反向传播算法，在方框处计算填入导数

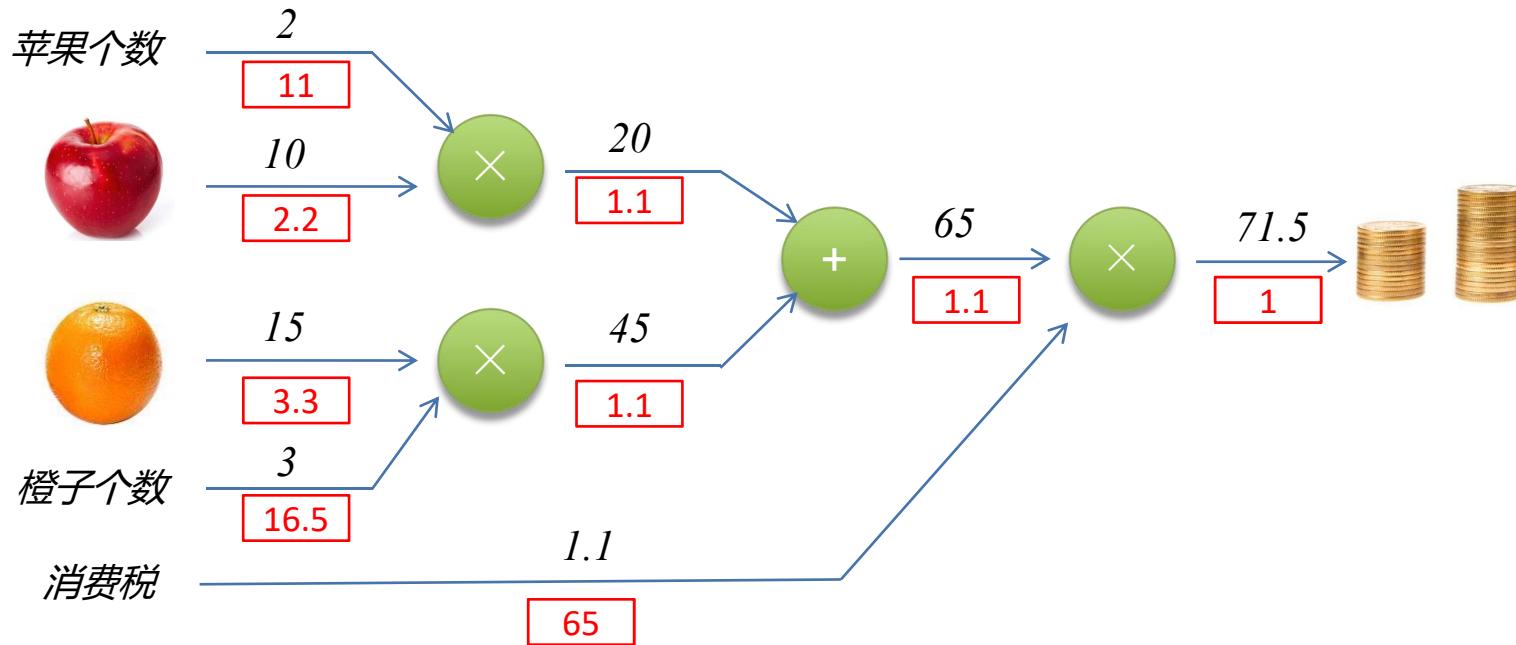
知识讲解



案例1：通过反向传播计算偏导数（续）

- 问题：苹果、橙子价格和个数以及税率如下图所示，利用反向传播算法，在方框处计算填入导数

知识讲解



小结

- 本章节主要介绍了反向传播算法，其目的是根据预测输出，调整权重参数，使得模型更快收敛。

卷积神经网络

卷积神经网络

卷积函数

什么是卷积

离散卷积与多维卷积

生活中的卷积

全连接神经网络的局限

什么是卷积神经网络

卷积神经网络的用途

卷积运算

卷积运算的效果

案例2：图像卷积运算

卷积运算神经网络结构

典型CNN介绍

卷积函数

什么是卷积

- “卷积”其实是一个数学概念，它描述一个函数和另一个函数在某个维度上的加权“叠加”作用。函数定义如下：

$$s(t) = \int_{-\infty}^{\infty} f(a) * g(t-a) da$$

其中，函数 f 和函数 g 是卷积对象， a 为积分变量，星号 “ $*$ ” 表示卷积。公式所示的操作，被称为连续域上的卷积操作。这种操作通常也被简记为如下公式：

$$s(t) = f(t) * g(t)$$



离散卷积与多维卷积

- 一般情况下，我们并不需要记录任意时刻的数据，而是以一定的时间间隔（也即频率）进行采样即可。对于离散信号，卷积操作可用如下表示：

$$s(t) = f(t) \times g(t) = \sum_{a=-\infty}^{\infty} f(a)g(t-a)$$

当然，对于离散卷积的定义可推广到更高维度的空间上。例如，二维的公式可表示为公式：

$$s(i, j) = f(i, j) \times g(i, j) = \sum_m \sum_n f(m, n)g(i-m, j-n)$$



生活中的卷积

- 在一根铁丝某处不停地弯曲，假设发热函数是 $f(t)$ ，散热函数是 $g(t)$ ，此时此刻的温度就是 $f(t)$ 跟 $g(t)$ 的卷积
- 在一个特定环境下，发声体的声源函数是 $f(t)$ ，该环境下对声源的反射效应函数是 $g(t)$ ，那么在这个环境下感受到的声音就是 $f(t)$ 的和 $g(t)$ 的卷积
- 记忆也是一种卷积

$$h_{\text{记忆}}(t)$$

$$= f_{\text{认知}}(t) * g_{\text{遗忘}}(t)$$

$$= \int_0^{+\infty} f_{\text{认知}}(\tau) g_{\text{遗忘}}(t - \tau) d\tau$$



卷积神经网络

全连接神经网络的局限

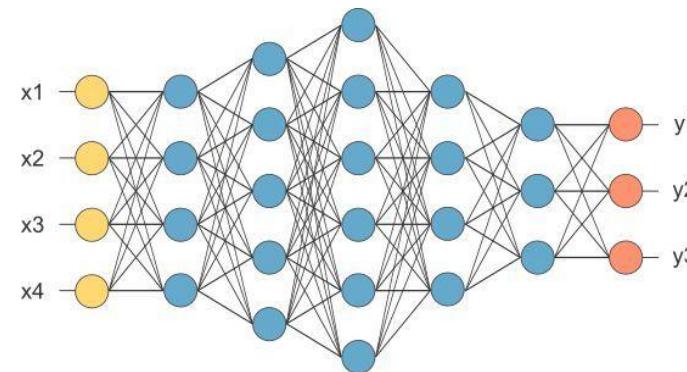
- 之前介绍的神经网络，相邻层所有神经元之间都有连接，这称为全连接（fully-connected）。全连接神经网络有以下几个问题：

1) 未考虑数据的“形状”，会破坏数据空间结构。

例如，输入数据是图像时，图像通常是高长通道方向上的3维形状。但是，向全连接层输入时，需要将3维数据拉平为1维数据。

2) 全连接网络层次深度受限，一般不超过七层。

3) 全连接网络参数量庞大，需要降低参数量。



什么是卷积神经网络

- 卷积神经网络 (Convolutional Neural Network , CNN) 针对全连接网络的局限做出了修正，加入了卷积层 (Convolution 层) 和池化层 (Pooling 层) 。



卷积神经网络的用途

- CNN被广泛应用于图像识别、语音识别等各种场合，在图像识别的比赛中，基于深度学习的方法几乎都以CNN为基础（比如，AlexNet、VGGNet、Google Inception Net及微软的ResNet等）上。近几年深度学习大放异彩，CNN功不可没。



卷积运算

- 单通道、二维卷积运算示例

知识讲解

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

输入数据

×

2	0	1
0	1	2
1	0	2

滤波器（卷积核）



15	

输出数据



卷积运算（续1）

- 单通道、二维卷积运算示例

知识讲解

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

输入数据

×

2	0	1
0	1	2
1	0	2

滤波器（卷积核）



15	16

输出数据



卷积运算（续2）

- 单通道、二维卷积运算示例

知识讲解

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

输入数据

×

2	0	1
0	1	2
1	0	2

滤波器（卷积核）



15	16
6	

输出数据



卷积运算（续3）

- 单通道、二维卷积运算示例

知识讲解

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

输入数据

×

2	0	1
0	1	2
1	0	2

滤波器（卷积核）



15	16
6	15

输出数据

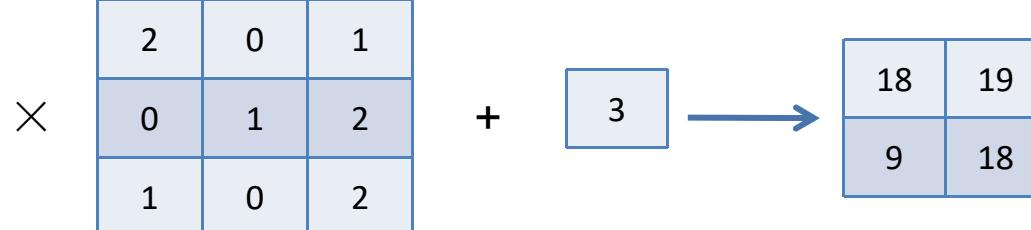


卷积运算（续4）

- 单通道、二维、带偏置的卷积运算示例

知识讲解

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1



输入数据

滤波器（卷积核）

偏置

输出数据



卷积运算（续5）

- 带填充（padding）的单通道、二维卷积运算示例

知识讲解

0	0	0	0	0	0
0	1	2	3	0	0
0	0	1	2	3	0
0	3	0	1	2	0
0	2	3	0	1	0
0	0	0	0	0	0

×

2	0	1
0	1	2
1	0	2



7	12	10	2
4	15	16	10
10	6	15	6
8	10	4	3

输入数据（padding:1）

滤波器（卷积核）

输出数据



卷积运算（续6）

- 步幅（stride）为2的卷积运算示例

知识讲解

1	2	3	0	1
0	1	2	3	0
3	0	1	2	3
2	3	0	1	2
1	2	3	0	1

×

2	0	1
0	1	2
1	0	2



15	

1	2	3	0	1
0	1	2	3	0
3	0	1	2	3
2	3	0	1	2
1	2	3	0	1

×

2	0	1
0	1	2
1	0	2



15	17



卷积运算（续7）

- 卷积运算输出矩阵大小计算公式

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

其中，输入大小为(H, W)，滤波器大小为(FH, FW)，输出大小为(OH, OW)，填充为P，步幅为S。例如：输入大小(28, 31)；填充2；步幅3；滤波器大小(5, 5)，则输出矩阵大小为：

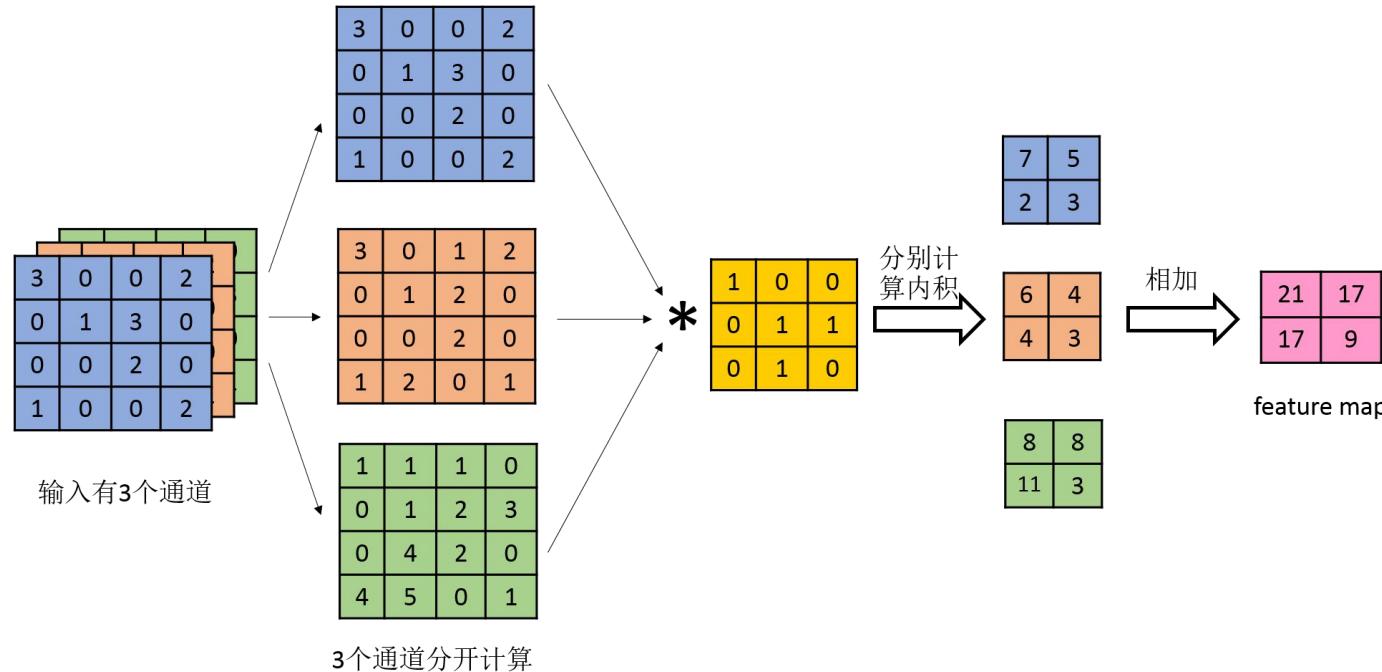
$$OH = \frac{28 + 2 \cdot 2 - 5}{3} + 1 = 10$$

$$OW = \frac{31 + 2 \cdot 2 - 5}{3} + 1 = 11$$



卷积运算 (续8)

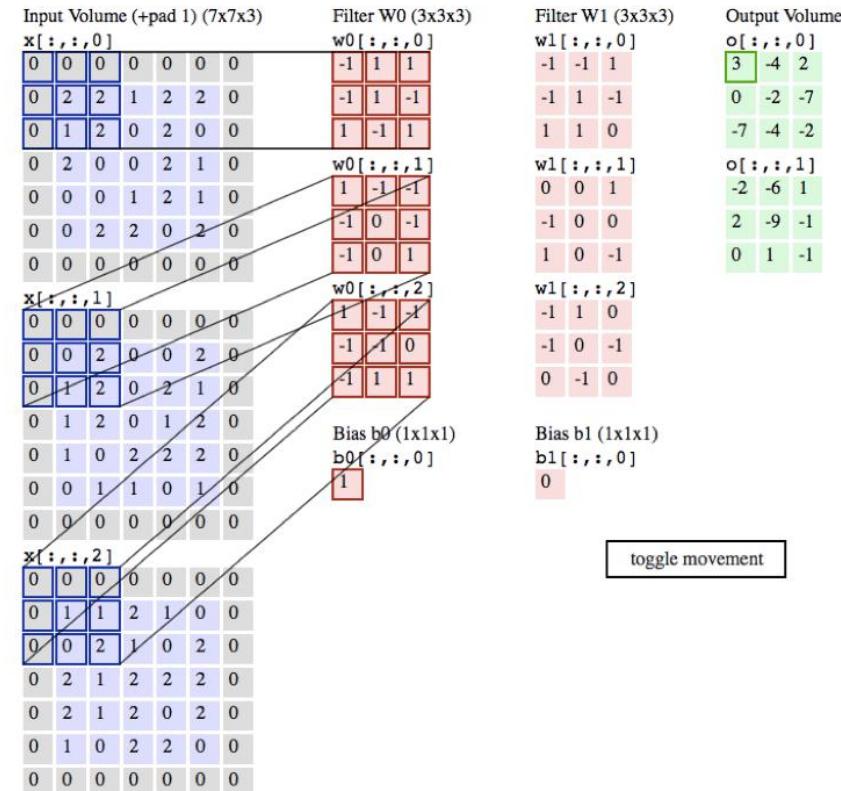
- 多通道卷积会按通道进行输入数据和滤波器的卷积运算，并将结果相加，从而得到输出



卷积运算（续9）

• 多通道、多卷积核卷积

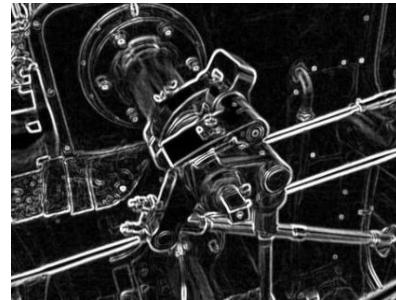
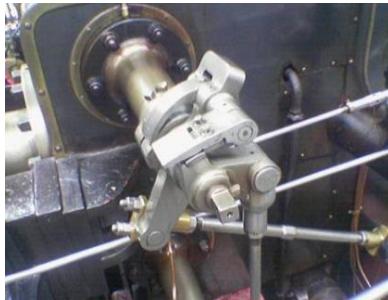
- ✓ 每个通道先与第一组卷积核执行卷积，然后多通道结果叠加，产生一个输出
- ✓ 每个通道与下一组卷积核执行卷积，产生另一个输出
- ✓ 有多少组卷积核，就有多少个通道输出（如右图，两组卷积核，产生两个通道的输出数据）



卷积运算的效果

- 通过卷积运算，能对输入数据起到加强或平滑效果。在图像处理中，通过选取合适的卷积核（或称算子），可以对图像进行锐化、去噪、模糊、加强边沿。

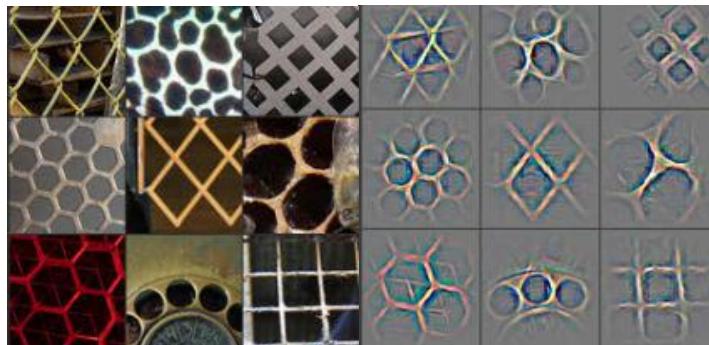
$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$



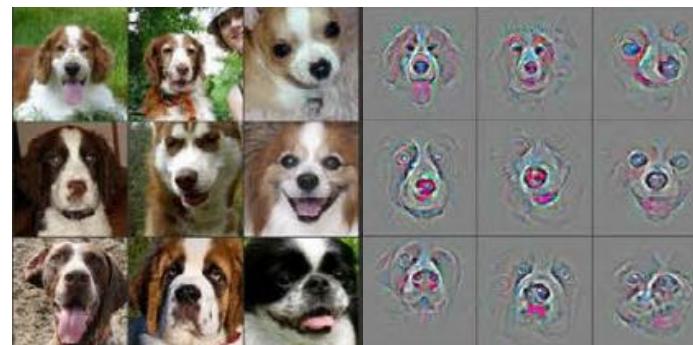
卷积运算的效果（续1）

- 卷积运算能提取深层次复杂特征

知识讲解



纹理相似性



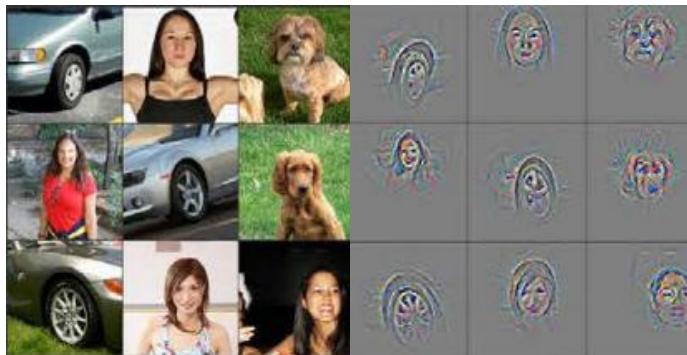
复杂环境下提取主体



卷积运算的效果（续2）

- 卷积运算能提取深层次复杂特征

知识讲解



形态变化



无关场景，卷积效果集中于草地背景



案例2：图像卷积运算

课堂练习

```
1  from scipy import signal
2  from scipy import misc
3  import matplotlib.pyplot as plt
4  import numpy as np
5
6  face = misc.imread("../a.png", flatten=True)
7
8  flt = np.array([[-1, 0, 1],
9                  [-2, 0, 2],
10                 [-1, 0, 1]])
11
12 # flt = np.array([[1, 2, 1],
13 #                   [0, 0, 0],
14 #                   [-1, -2, -1]])
15
16 # 把图像的face数组和设计好的卷积核作二维卷积运算,设计边界处理方式为symm
17 grad = signal.convolve2d(face, flt,
18                          boundary='symm', mode='same').astype("int32")
```



案例2：图像卷积运算（续）

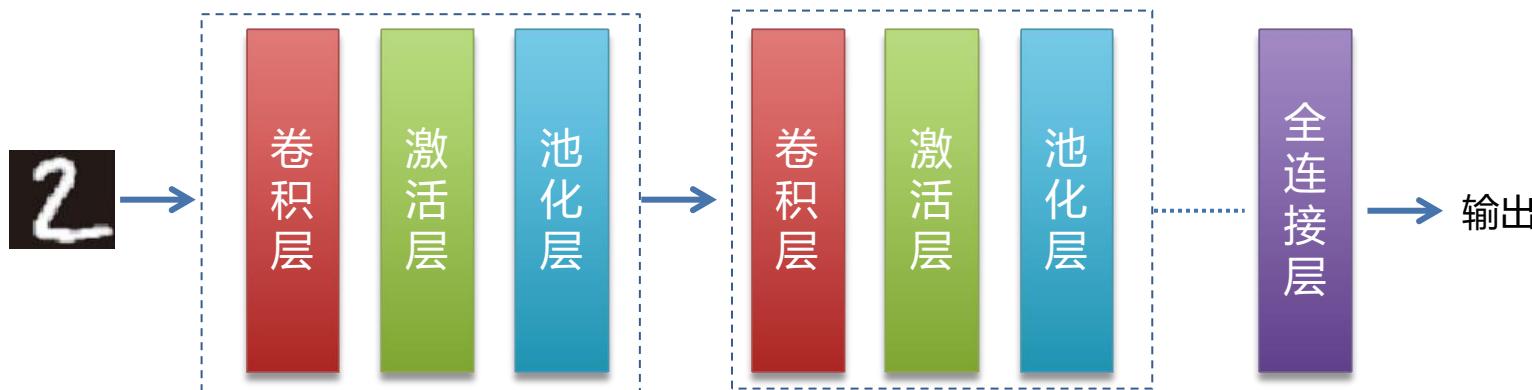
```
20 plt.figure("Conv2D")
21 plt.subplot(121)
22 plt.imshow(face, cmap='gray') # 显示原始的图
23 plt.xticks([])
24 plt.yticks([])
25
26 plt.subplot(122)
27 plt.xticks([])
28 plt.yticks([])
29 plt.imshow(grad, cmap='gray') # 显示卷积后的图
30 plt.show()
```



卷积神经网络结构

- 总体结构

通常情况下，卷积神经网络由若干个卷积层（Convolutional Layer）、激活层（Activation Layer）、池化层（Pooling Layer）及全连接层（Fully Connected Layer）组成。

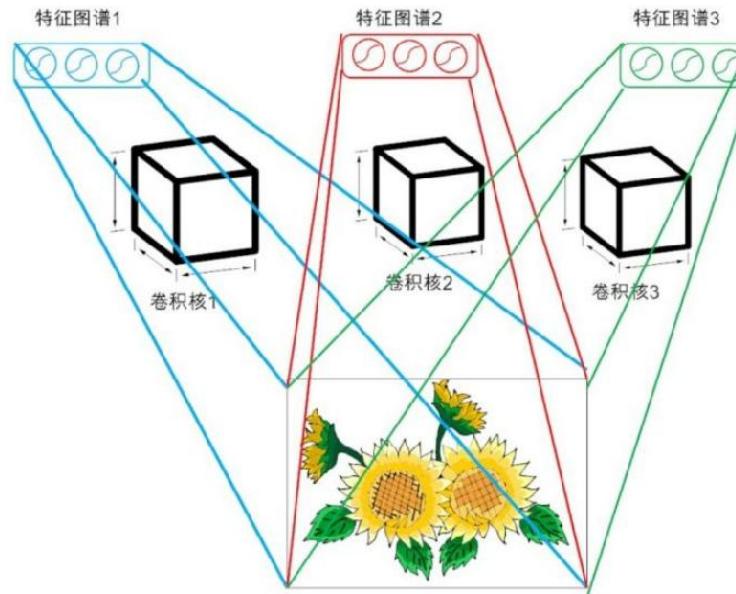


卷积神经网络结构（续1）

- 卷积层

它是卷积神经网络的核心所在，通过卷积运算，达到降维处理和提取特征两个重要目的

知识讲解



卷积神经网络结构（续2）

• 激活层

其作用在于将前一层的线性输出，通过非线性的激活函数进行处理，这样用以模拟任意函数，从而增强网络的表征能力。前面章节中介绍的激活函数，如挤压函数Sigmoid也是可用的，但效果并不好。在深度学习领域，ReLU（Rectified-Linear Unit，修正线性单元）是目前使用较多的激活函数，主要原因是它收敛更快，次要原因在于它部分解决了梯度消失问题。

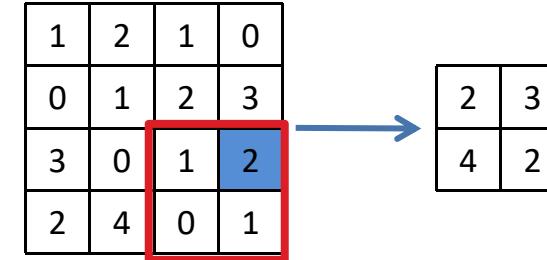
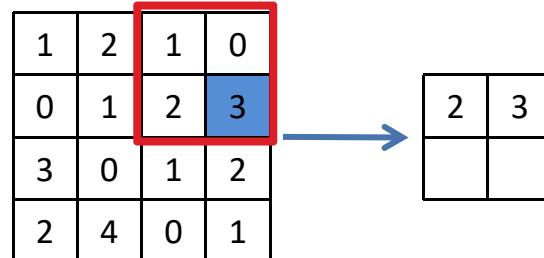
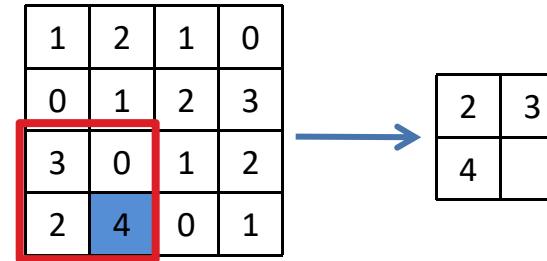
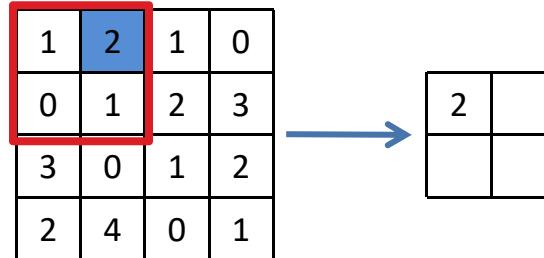


卷积神经网络结构（续3）

- 池化层（Pooling Layer）

也称子采样层或下采样层（Subsampling Layer），目的是缩小高、长方向上的空间的运算，以降低计算量，提高泛化能力。如下的示例，将 4×4 的矩阵缩小成 2×2 的矩阵输出

知识讲解



卷积神经网络结构（续4）

• 池化层计算

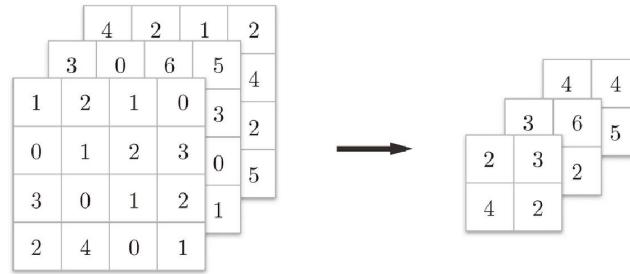
- 对于每个输入矩阵，我们将其切割成若干大小相等的正方形小块，对每一个区块取最大值或者平均值，并将结果组成一个新的矩阵
- Max池化：对各个参与池化计算的区域取最大值，形成的新矩阵。在图像识别领域，主要使用Max池化
- Average池化：对各个参与池化计算的区域计算平均值



卷积神经网络结构（续5）

• 池化层的特征

- **没有要学习的参数。** 池化层和卷积层不同，没有要学习的参数。池化只是从目标区域中取最大值（或者平均值），所以不存在要学习的参数
- **通道数不发生变化。** 经过池化运算，输入数据和输出数据的通道数不会发生变化



- **对微小的位置变化具有鲁棒性（健壮）。** 输入数据发生微小偏差时，池化仍会返回相同的结果



卷积神经网络结构（续6）

• 全连接层

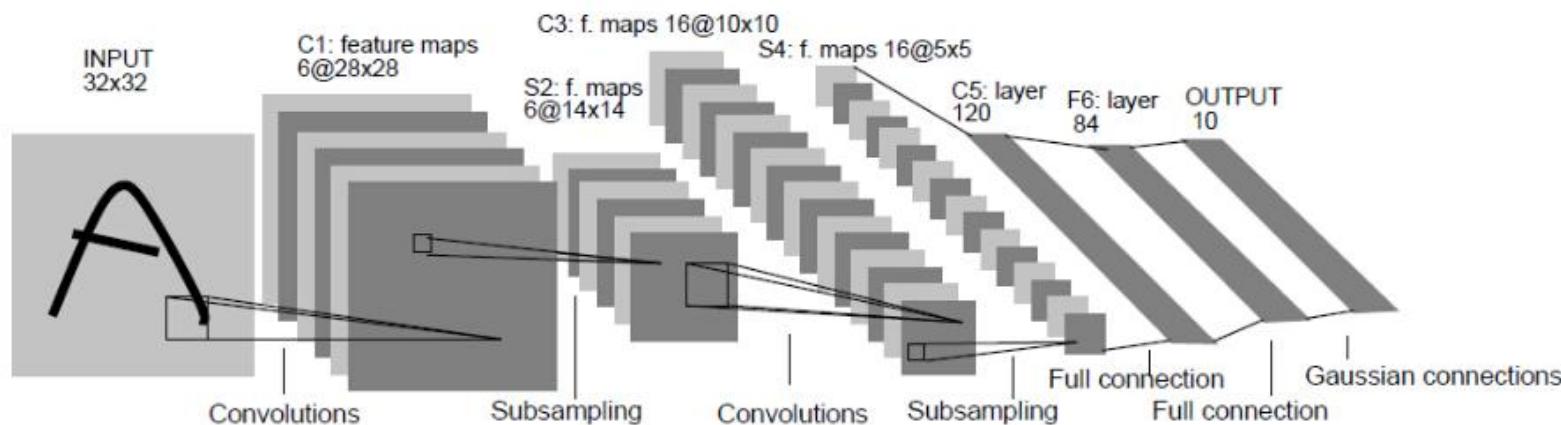
- 这个网络层相当于多层感知机（Multi-Layer Perceptron，简称MLP），其在整个卷积神经网络中起到分类器的作用
- 通过前面多个“卷积-激活-池化”层的反复处理，待处理的数据特性已经有了显著提高：一方面，输入数据的维度已下降到可用传统的前馈全连接网络来处理了；另一方面，此时的全连接层输入的数据已不再是“泥沙俱下、鱼龙混杂”，而是经过反复提纯过的结果，因此输出的分类品质要高得多。



经典CNN介绍

- LeNet：结构

LeNet是Yann LeCun在1998年提出，用于解决手写数字识别的视觉任务。自那时起，CNN的最基本的架构就定下来了：卷积层、池化层、全连接层。



典型CNN介绍（续1）

- LeNet : 主要参数

- 输入：输入 $32*32$ 大小单通道图像
- 两个“卷积-池化层”
- 第一个全连接层神经元数目为500，再接激活函数
- 第二个全连接层神经元数目为10，得到10维的特征向量，用于10个数字的分类训练，送入softmax分类，得到分类结果的概率



典型CNN介绍（续2）

• AlexNet：特点

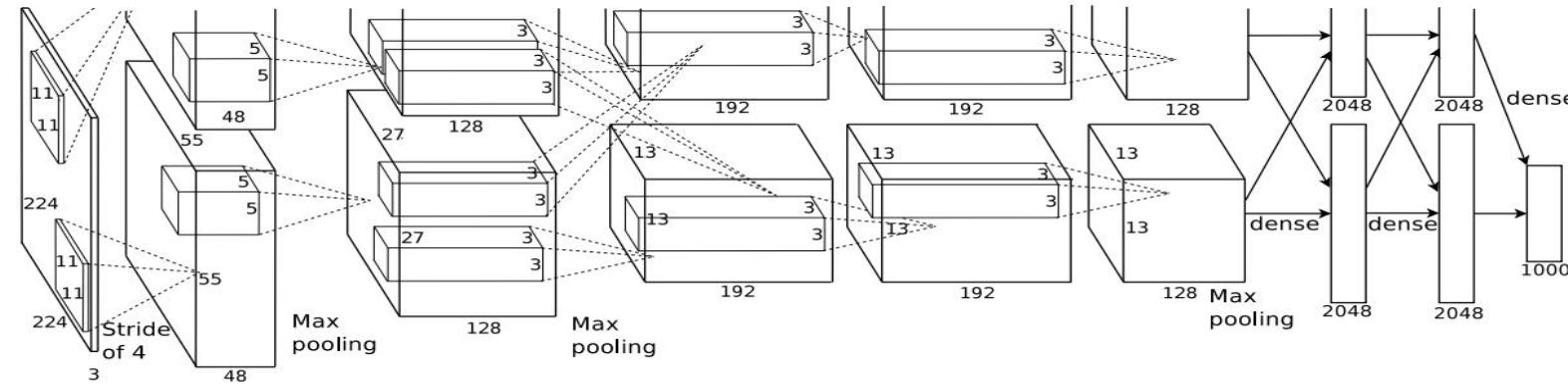
AlexNet是2012年ImageNet竞赛冠军获得者Hinton和他的学生Alex Krizhevsky设计的，把CNN的基本原理应用到了很深很宽的网络中。其特点有：

- ✓ 使用ReLU作为激活函数，并验证其效果在较深的网络超过了Sigmoid，成功解决了Sigmoid在网络较深时的梯度弥散问题
- ✓ 使用Dropout（丢弃学习）随机忽略一部分神经元防止过拟合
- ✓ 在CNN中使用重叠的最大池化。此前CNN中普遍使用平均池化，AlexNet全部使用最大池化，避免平均池化的模糊化效果
- ✓ 提出了LRN（Local Response Normalization，局部正规化）层，对局部神经元的活动创建竞争机制，使得其中响应比较大的值变得相对更大，并抑制其他反馈较小的神经元，增强了模型的泛化能力
- ✓ 使用CUDA加速深度卷积网络的训练，利用GPU强大的并行计算能力，处理神经网络训练时大量的矩阵运算



典型CNN介绍（续3）

- AlexNet：网络结构



典型CNN介绍（续4）

• AlexNet：主要参数

AlexNet网络包含8层，其中前5层为卷积-池化层，后3层为全连接层；输入 $224 \times 224 \times 3$ 的图像，第一卷积层用96个 $11 \times 11 \times 3$ 的卷积核对进行滤波，步幅4像素；全连接的每层有4096个神经元，最后一个完全连接的层的输出被馈送到1000路SoftMax，它产生超过1000个类别标签的分布；整个网络共650000个神经元

参考论文：<http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf>



典型CNN介绍（续5）

- VGG: 概要介绍

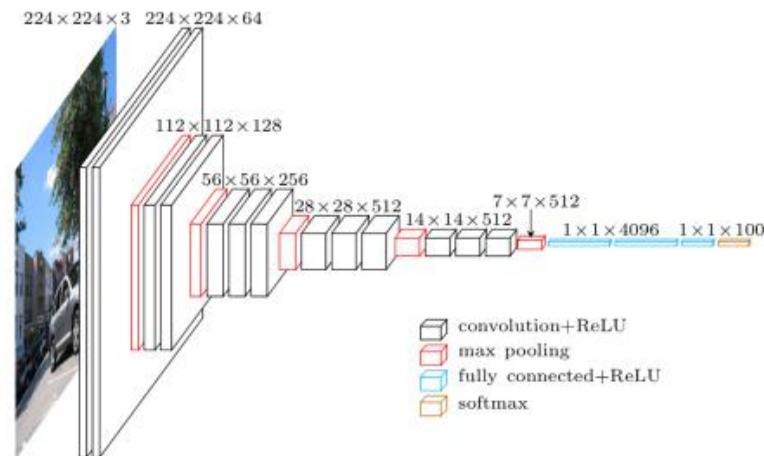
VGG是Visual Geometry Group, Department of Engineering Science, University of Oxford (牛津大学工程科学系视觉几何组) 的缩写，2014年参加ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 2014大赛获得亚军 (当年冠军为GoogLeNet，但因为VGG结构简单，应用性强，所以很多技术人员都喜欢使用基于VGG的网络)



典型CNN介绍（续6）

- VGG: 主要参数

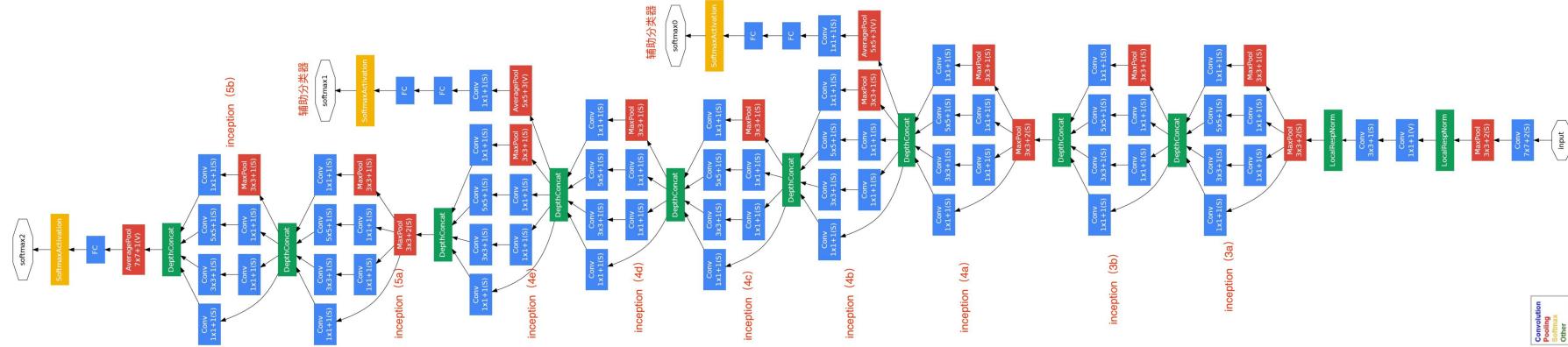
- ✓ 网络深度：16~19层
- ✓ 5组卷积-池化层，3个全连接层
- ✓ 三个全连接层，前两层都有4096通道，第三层共1000路及代表1000个标签类别；最后一层为softmax层
- ✓ 所有卷积层有相同的配置，即卷积核大小为 3×3 ，步长为1，填充为1



典型CNN介绍（续7）

- GoogLeNet

知识讲解



请参考论文：<https://arxiv.org/pdf/1409.4842.pdf>

小结

- 本章节介绍了卷积神经网络（CNN），CNN是深度学习的主要模型，在解决复杂工程问题中表现出了良好的性能。卷积神经网络主要由以下几层构成：
 - ✓ 卷积层。 执行卷积运算
 - ✓ 激活层。 对卷积结果执行激活函数运算
 - ✓ 池化层。 降低数据规模，防止过拟合
 - ✓ 全连接层。 执行输出计算

今日总结

- 深度学习概述
- 感知机、神经网络
- 损失函数与梯度下降
- 反向传播算法
- 卷积神经网络