法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

第5讲　神经网络与BP算法

计算机视觉

主讲人：叶梓

上海交通大学博士
主要研究方向：机器学习、深度学习、人工智能

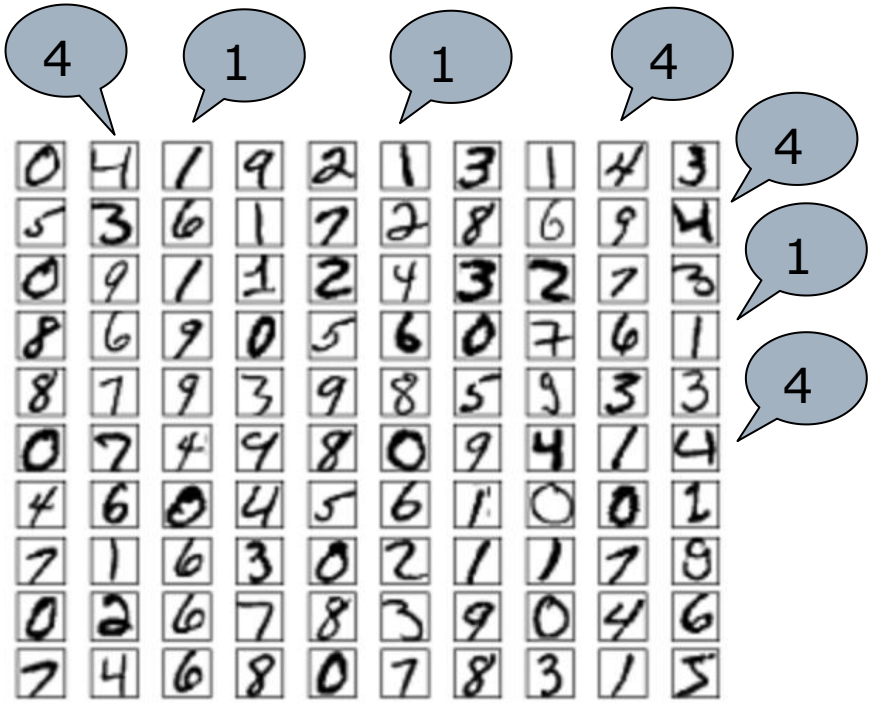
本章内容

- 神经网络
  - 神经元
  - 前馈网络
  - 梯度下降
  - 误差反向传播
- 应用案例：
  - 可以手算的神经网络示例
  - BP神经网络的实现
- 从神经网络到深度学习

神经网络

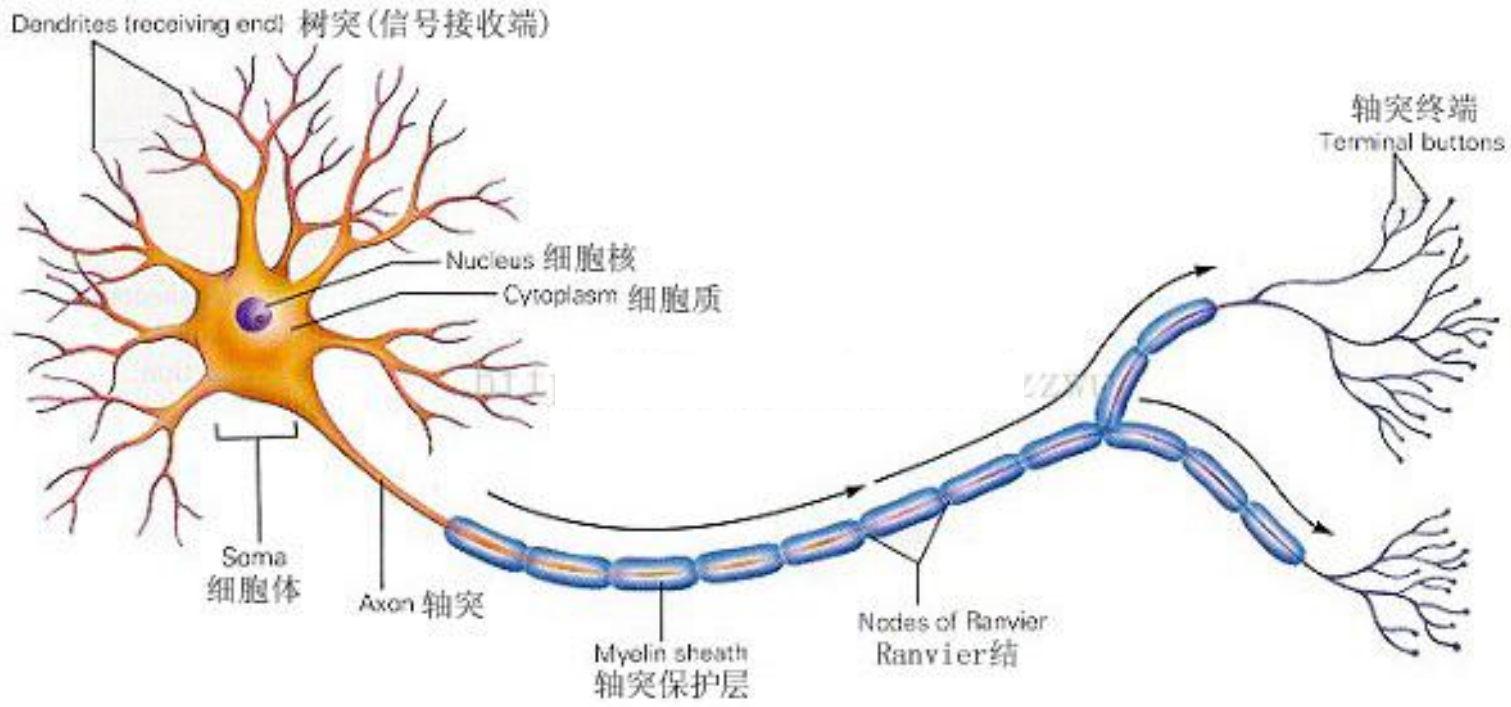
- 神经网络：大量（结构简单的、功能接近的）神经元节点按一定体系架构连接成的网状结构——就是大脑的结构。

- 神经网络的作用
  - 分类、
  - 模式识别
  - 连续值预测
- 建立输入与输出的映射关系



生物神经元

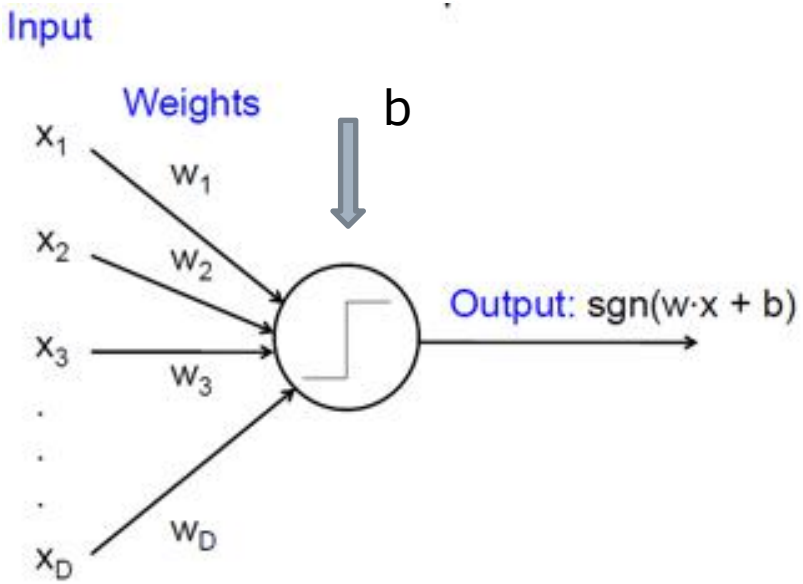
- 生物神经元的结构大致描述如下图所示。
- 生物神经元之间的相互连接，从而让信息传递。



小象学院 ChinaHadoop.cn

人工神经元

神经元模型



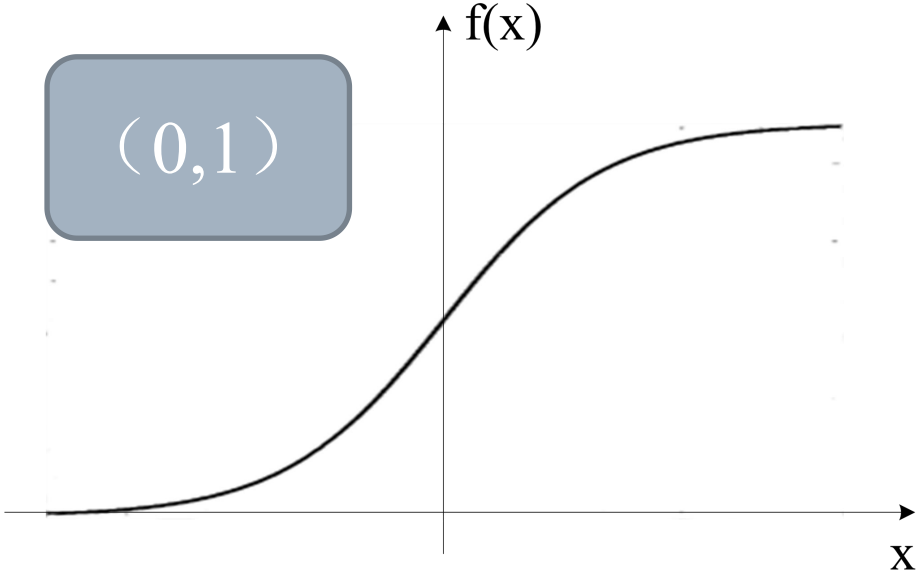
每个神经元都是一个结构相似的独立单元，它接受前一层传来的数据，并将这些数据的加权和输入非线性作用函数中，最后将非线性作用函数的输出结果传递给后一层。

$$y = f\left(\mathbf{w}^t\mathbf{x}\right) = f\left(\sum_{i=1}^{d} w_i x_i\right),$$

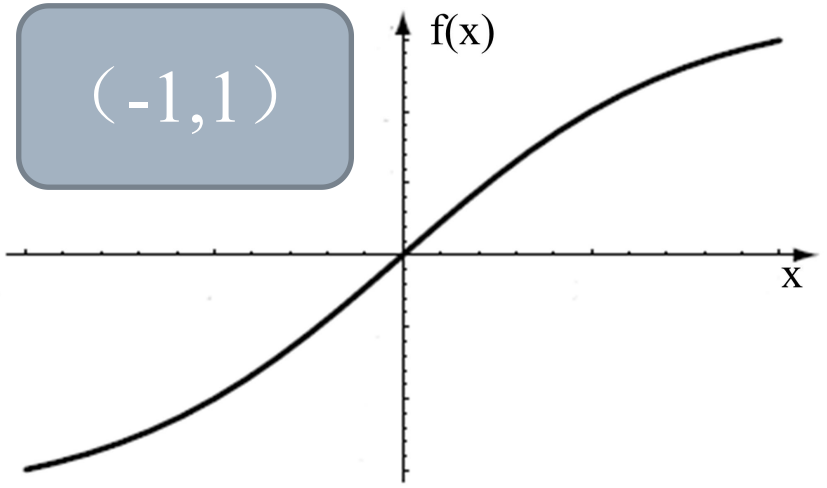
非线性函数f，称为激活函数

激活函数

（0,1）



（-1,1）



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

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

简单说一下导数

- 链式求导：

$$(f(g(x)))' = f'(g(x))g'(x)$$

- 写成另一种形式：

$$\frac{dy}{dx} = \frac{dy}{dz} \bullet \frac{dz}{dx}$$

令：y=f(x)
f’(x)=y(1-y)

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

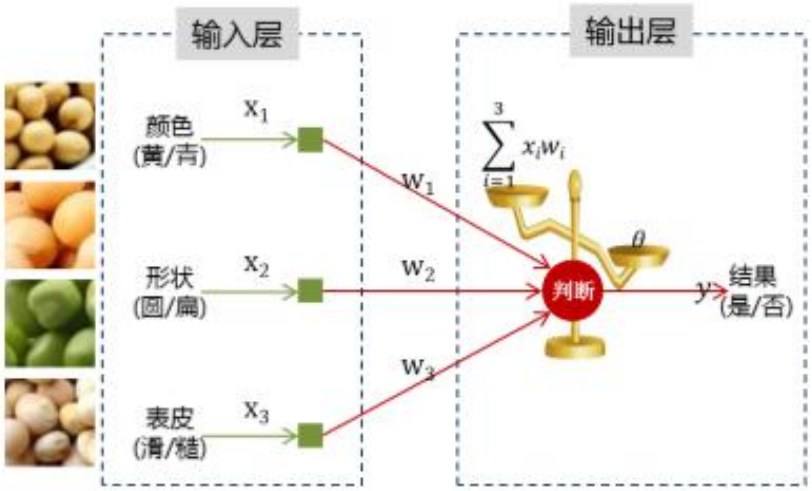
令：y=f(x)
f’(x)=1-y^2

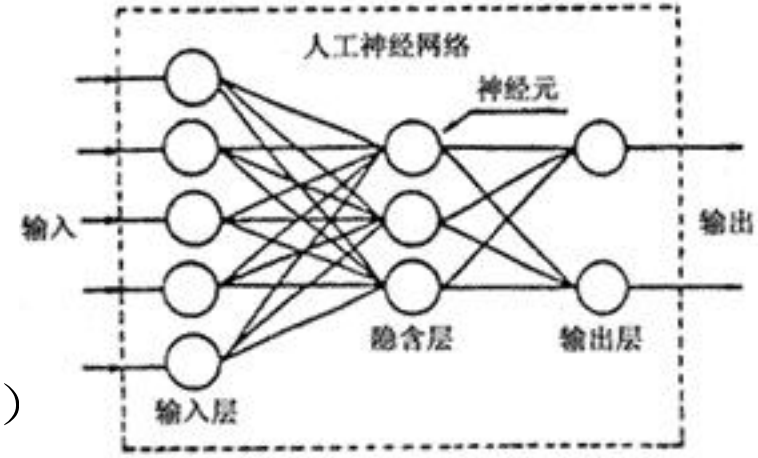
$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

人工神经网络

基础神经网络

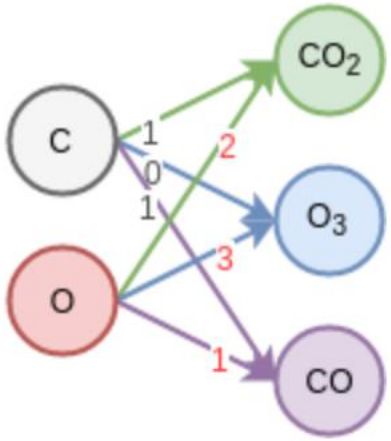
- 神经元
  - 输入向量x
  - 权重向量w
  - 偏置标量b
  - 激活函数sigmoid
- 浅网络
  - 3~5层
  - 优化
    - 梯度下降
    - BP后向传播（链式规则）





“层”的通俗理解

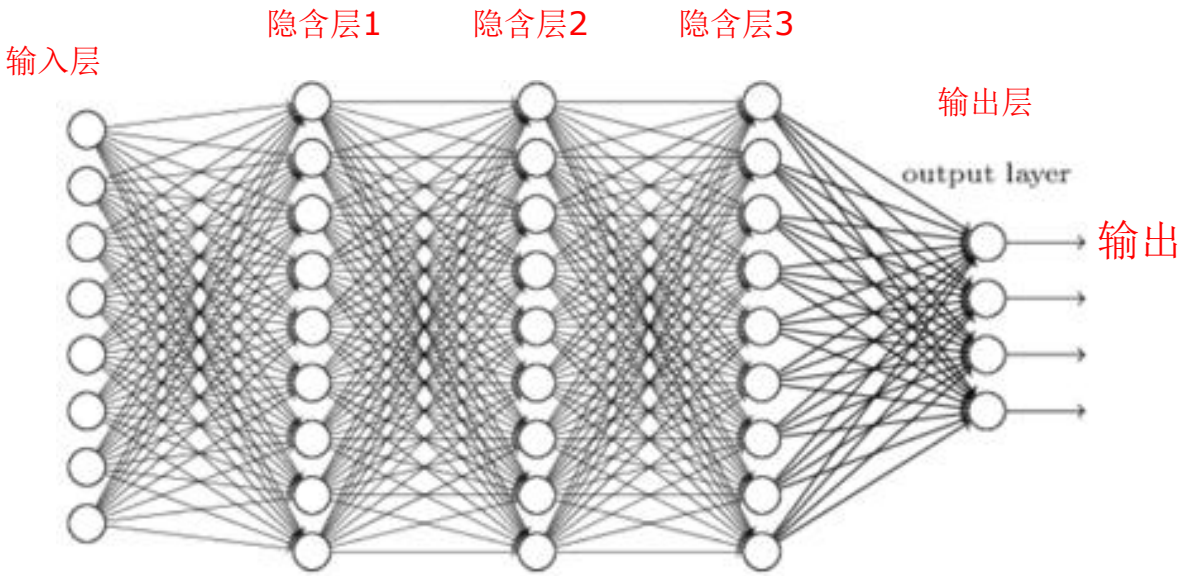
- “层”实现了输入空间到输出空间的线性或非线性变换。
- 假设输入是碳原子和氧原子[C；O]，输出三个变量。
- 通过改变权重的值，可以获得若干个不同物质
  - ——提取不同的特征。
- 右侧的节点数决定了想要获得多少种不同的新 物质
  - ——提取特征的个数。



$$\begin{bmatrix} CO_2 \\ O_3 \\ CO \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} C \\ O \end{bmatrix}$$

人工神经网络

- 神经网络是由大量神经元节点按一定体系架构连接成的网状结构，一般都有输入层，隐含层和输出层。
- 传统的浅层网络，一般有3~5层。



前馈神经网络

- 前馈神经网络，是人工神经网络的一种，各神经元从输入层开始，接收前一级输入，并输出到下一级，直至输出层。整个网络中无反馈，可用一个有向无环图表示。
- 前馈神经网络采用一种单向多层结构。其中每一层包含若干个神经元，同一层的神经元之间没有互相连接，层间信息的传送只沿一个方向进行。
- 其中第一层称为输入层。最后一层为输出层.中间为隐含层。隐含层可以是一层，也可以是多层。

Delta学习规则

- Delta学习规则是一种有监督学习算法，该算法根据神经元的实际输出与期望输出差别来调整连接权，其数学表示如下：

- 即：　　$\Delta w_{ij} = a \cdot (d_i - y_i) x_j(t)$

- 其中：
  - Δw_{ij}表示神经元j到神经元i的连接权重增量，
  - d_i是神经元i的期望输出，
  - y_i是神经元i的实际输出，
  - x_j表示神经元j状态，
  - a是表示学习速度的常数。

前馈神经网络的目标函数

- 对于一系列训练样本**x**，
- 期望输出**t**=(t_1,…, t_c)，网络实际输出**z**=(z_1,…, z_c)，

- 目标函数：

各输出误差的平方的累加

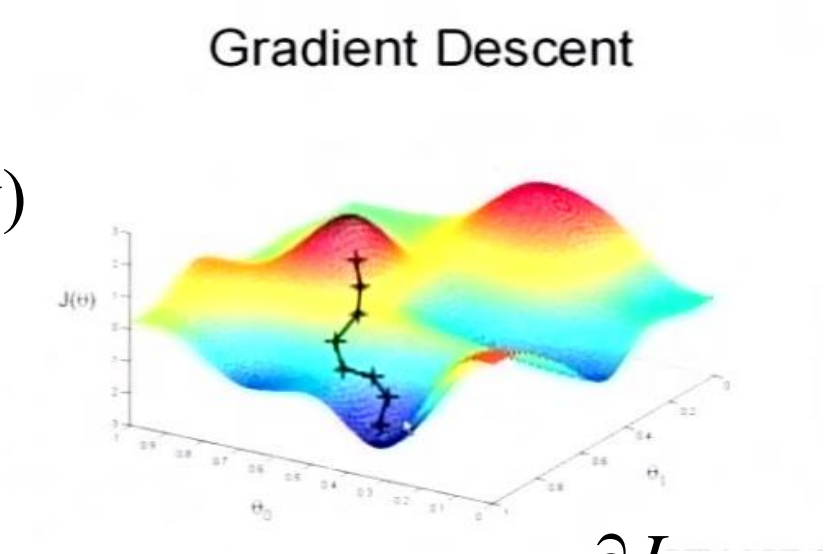
$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{t}-\mathbf{z}\|^2 = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2$$

如何计算目标函数的最小值？

梯度下降

- 这是一个表示参数w与目标函数J(w)的关系图，红色的部分是表示J(w)有着比较高的取值，需要能够让J(w)的值尽量的低。也就是深蓝色的部分。w1，w2表示w向量的两个维度。

- 先确定一个初始点，将w按照梯度下降的方向进行调整，就会使得J(w)往更低的方向进行变化，如图所示，算法的结束将是在w下降到无法继续下降为止。



$$\mathbf{w}(m+1)=\mathbf{w}(m)+\Delta\mathbf{w}(m)=\mathbf{w}(m)-\eta\frac{\partial J}{\partial \mathbf{w}}$$

小象学院 ChinaHadoop.cn

梯度下降示意

- 想象一下，你如何能在被蒙住眼睛的情况下，从山上走下来？
- 先用你灵巧的脚，探一探脚下的山地，哪个方向坡度最陡？（计算梯度方向）
- 朝着这个方向迈一步；（沿梯度方向下降）
- 一大步，还是一小步？（学习速率）
- 持续这个过程直到平地（迭代）
- 可不能有悬崖哦……（目标函数处处可导）



输出层权重改变量

$$J(\mathbf{w})=\frac{1}{2}\|\mathbf{t}-\mathbf{z}\|^2=\frac{1}{2}\sum_{k=1}^{c}(t_k-z_k)^2$$

$$\frac{\partial J}{\partial w_{kj}}=\frac{\partial J}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}}$$

输出单元的总输入

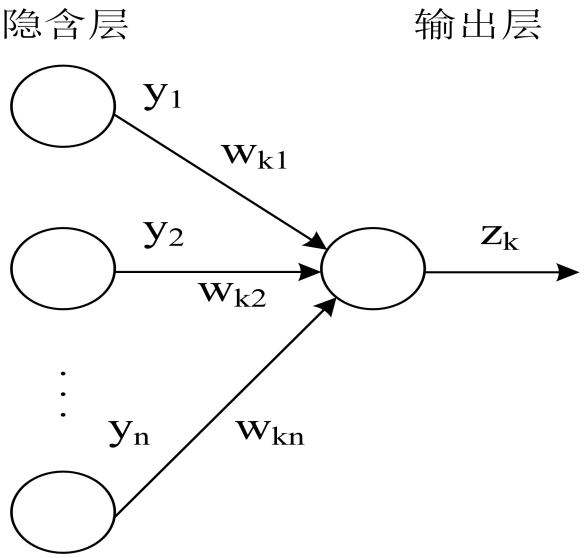
其中：$net_k=\sum_{i=1}^{n_H}w_{ki}y_i$，$\frac{\partial net_k}{\partial w_{kj}}=y_j$

输出

$$\frac{\partial J}{\partial net_k}=\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k}=-(t_k-z_k)f'(net_k)$$

令：$\delta_k=(t_k-z_k)f'(net_k)$

$$\frac{\partial J}{\partial w_{kj}}=-(t_k-z_k)f'(net_k)y_j=-\delta_k y_j$$



隐藏层权重改变量

$$net_j = \sum_{m=1}^{d} W_{jm} X_m$$

隐藏层单元总输入

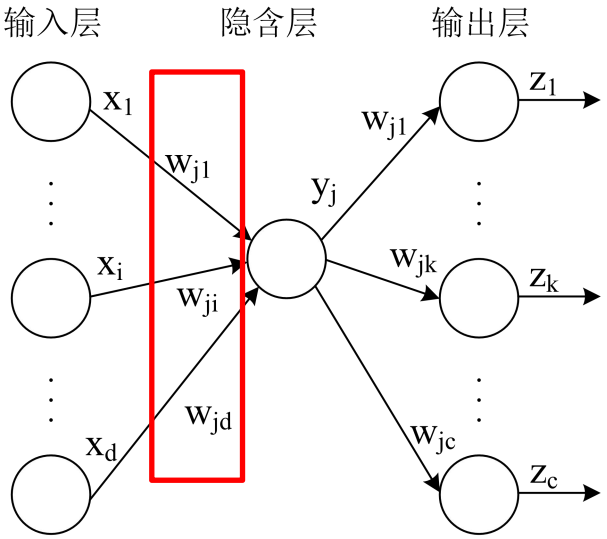
$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}$$

隐藏层单元输出

$$\frac{\partial y_j}{\partial net_j} = f'\left(net_j\right)$$

$$\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\left(\sum_{m=1}^{d} w_{jm} x_m\right) = x_i$$

1、先找w_{ji}与Net_j的关系；
2、然后找Net_j与隐藏层输出y_j的关系



隐含层权重改变量

输出单元的总输入

$$net_k = \sum_{i=1}^{n_H} w_{ki} y_i ,$$

$$\frac{\partial J}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2\right] = -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j}$$

输出

隐藏层单元输出

隐藏层单元输出

$$= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j} = -\sum_{k=1}^{c}(t_k - z_k) f'(net_k) w_{kj}$$

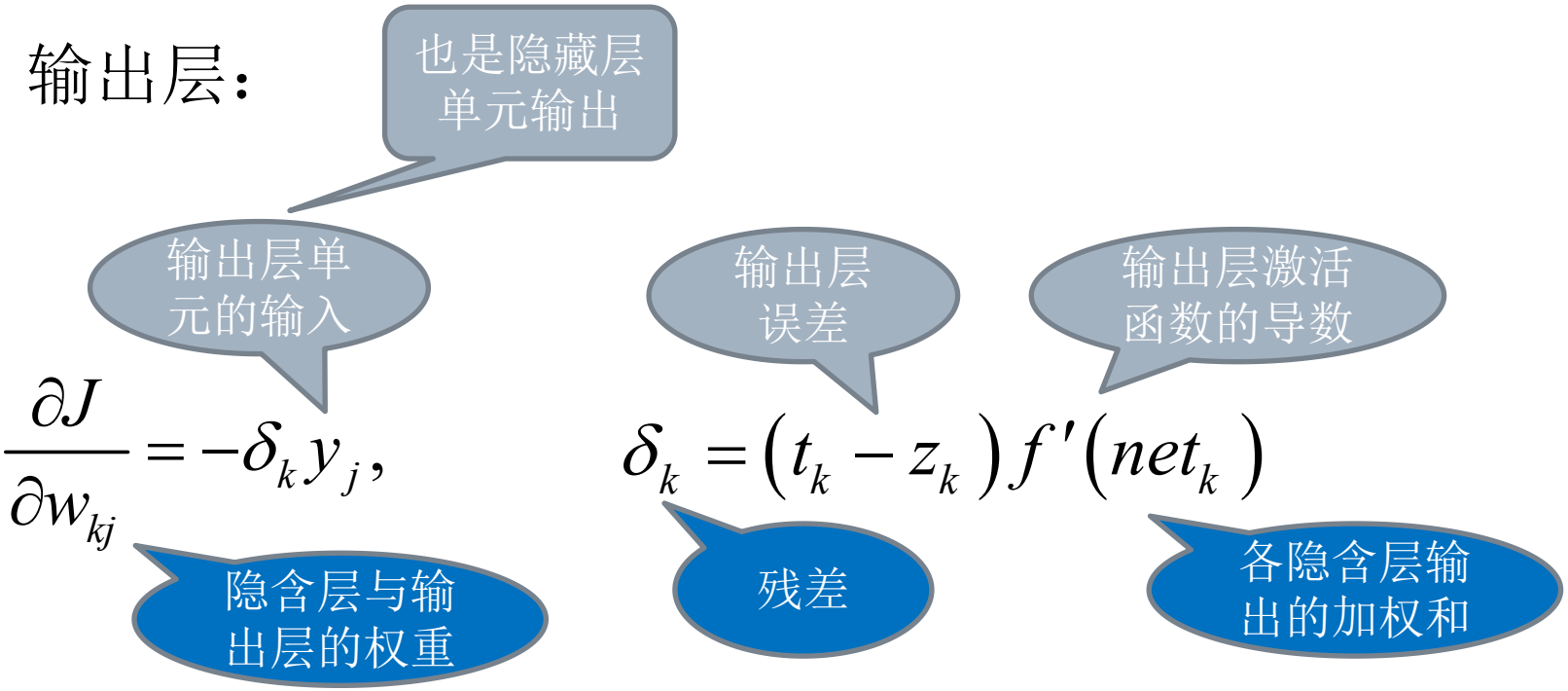
$$\frac{\partial J}{\partial w_{ji}} = -\left[\sum_{k=1}^{c}(t_k - z_k) f'(net_k) w_{kj}\right] f'(net_j)\, x_i$$

令：$\delta_j = f'(net_j)\sum_{k=1}^{c}\delta_k w_{kj}$

$$\frac{\partial J}{\partial w_{ji}} = -\delta_j x_i ,$$

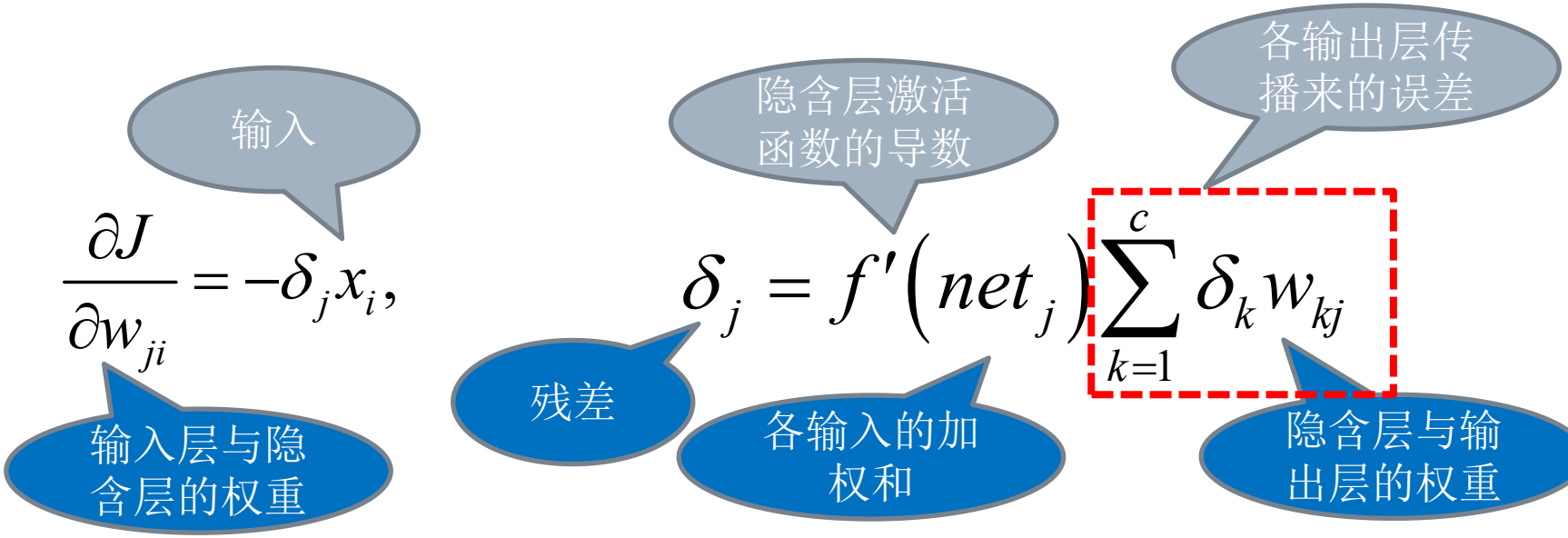
误差传播迭代公式

- 输出层：



误差传播迭代公式

- 隐含层：

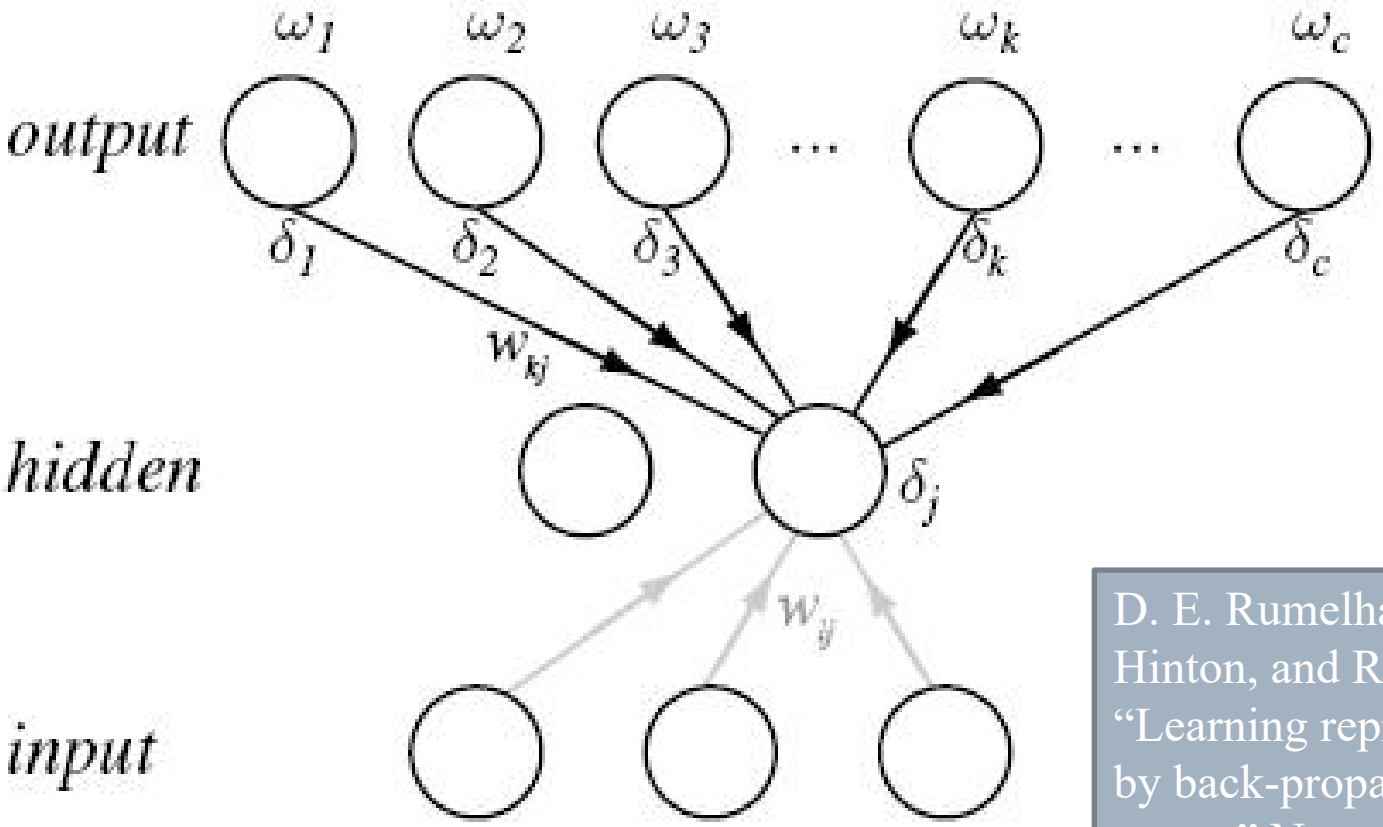


误差传播迭代公式

- 输出层和隐藏层的误差传播公式可统一为：

- 权重增量 = -1*学习步长*目标函数对权重的偏导数
- 目标函数对权重的偏导数 = -1*残差*当前层的输入
- 残差= 当前层激励函数的导数*上层反传来的误差
- 上层反传来的误差 = 上层残差的加权和

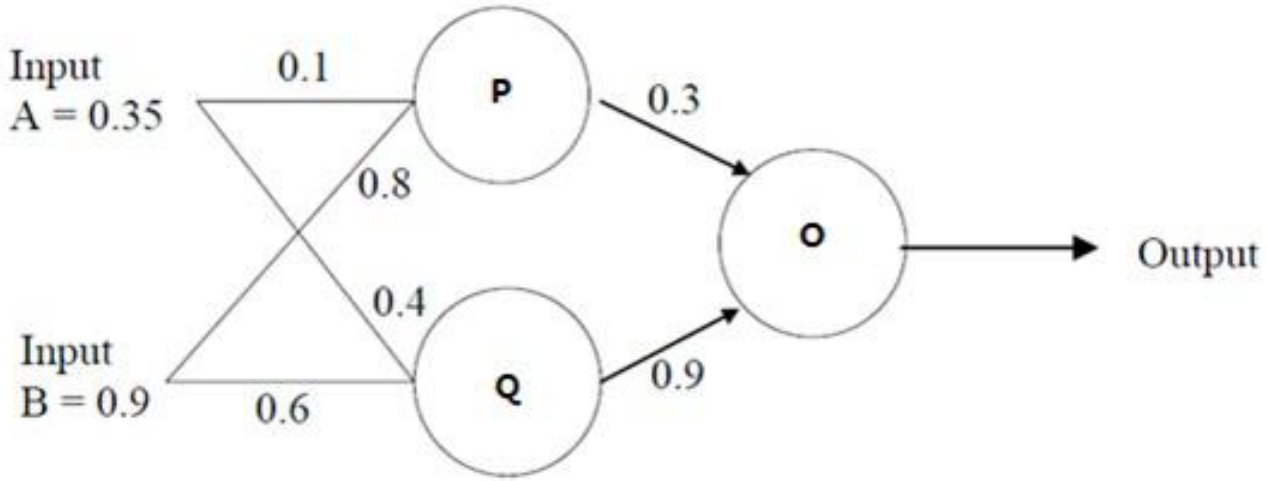
隐藏层误差反向传播示意



D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533–536, 1986.

简单的BP算例

举例



假定输入样本的自变量为(0.35,0.9)，应变量为 0.5，初始的权重函数如图所示

AP=0.1;　AQ=0.4,　PO=0.3

BP=0.8;　BQ=0.6,　QO=0.9

三个神经元的计算公式均为$f(x)=\frac{1}{1+e^{-x}}$，X 等于输入的权重和。

为了以示区别 P 上的函数 p,Q 上的函数记做 q,O 上的函数记做 o。其均为 $f(x)=\frac{1}{1+e^{-x}}$

例如：$P(A,B)=\frac{1}{1+e^{-(AP*A+BP*B)}}=\frac{1}{1+e^{-(0.1*0.35+0.9*0.8)}}=0.68$

简单的BP算例

O 可以看做是 P，Q 的函数 O(P,Q)，也可以看做是 A，B 的函数 O(A,B)。
显然在 A，B 已知的情况下，O 也可以计算出来，计算过程如下：

P(A,B)=0.68;　Q(A,B)=0.6637;　O(P,Q)=**0.69**

计算$\xi = \frac{1}{2}e^2 = \frac{1}{2}(0.69-0.5)^2$=0.01805。我们希望的这个数得到最小值。

很显然ξ是 PO，QO 的函数，怎么调整 PO，QO 才能让其变小呢？
用梯度下降法

$$\frac{\partial \xi}{\partial PO} = \frac{\partial \xi}{\partial e} * \frac{\partial e}{\partial o} * \frac{\partial o}{\partial PO} = \{e\} * \{O * (1-O)\} * \{P\}$$

$$= (0.69-0.5) * \mathbf{0.69} * (1-\mathbf{0.69}) * 0.68 = 0.02763$$

$$\frac{\partial \xi}{\partial QO} = e * f(x)(1-f(x)) * Q = (0.69-0.5) * 0.69 * (1-0.69) * 0.6673 = 0.02711$$

$$PO^* = PO - \frac{\partial \xi}{\partial PO} = 0.2723$$

$$QO^* = QO - \frac{\partial \xi}{\partial QO} = 0.8730$$

简单的BP算例

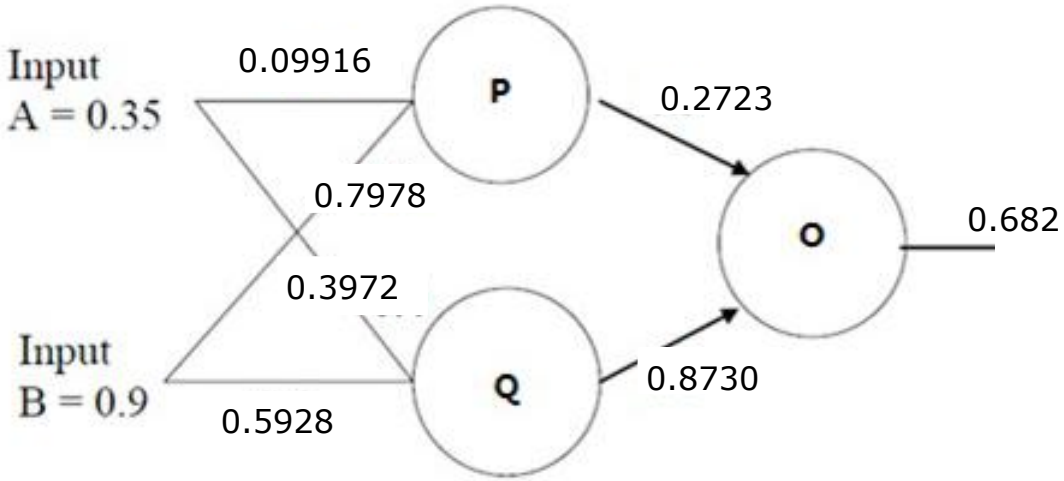
$$\frac{\partial \xi}{\partial AP} = \frac{\partial \xi}{\partial e} * \frac{\partial e}{\partial o} * \frac{\partial o}{\partial p} * \frac{\partial p}{AP} = \{e\} * \{O(1-O)\} * \{PO^*\} * \{(1-P) * P\} * \{A\}$$

$$AP^* = AP - \frac{\partial \xi}{\partial AP} = 0.09916$$

$$BP^* = BP - \frac{\partial \xi}{\partial BP} = 0.7978$$

$$AQ^* = AQ - \frac{\partial \xi}{\partial AQ} = 0.3972$$

$$BQ^* = BQ - \frac{\partial \xi}{\partial BQ} = 0.5928$$



小象学院 ChinaHadoop.cn

Geoffery E. Hinton



(1985)



(2015)

在(AI)这个世界里，他是当之无愧的教父，他的成功绝非一日之功。

可样本显然不是一个……

- **批量梯度下降**
  - 每迭代一步，都要用到训练集所有的数据，它得到的是一个全局最优解，但是如果m很大，那么可想而知这种方法的迭代速度！！
- **随机梯度下降（SGD）**
  - 通过每个样本来迭代更新一次，如果样本量很大的情况，那么可能只用其中部分的样本，就已经迭代到最优解了，
  - 但是，SGD伴随的一个问题是噪音较BGD要多，使得SGD并不是每次迭代都向着整体最优化方向。

Mini-batch Gradient Descent

- mini-batch Gradient Descent的方法是在“批量梯度下降”和“随机梯度下降”两种方法中取折衷，每次从所有训练数据中取一个子集（mini-batch）用于计算梯度。
- 广义条件下，SGD指的就是MBGD。

$$\hat{g} \leftarrow +\frac{1}{m}\nabla_\theta \sum_i L(f(x_i;\theta), y_i)$$

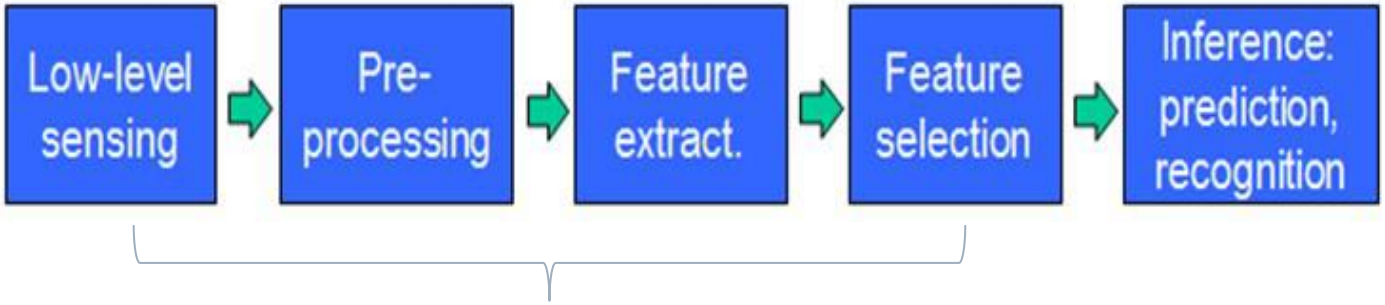
$$\theta \leftarrow \theta - \epsilon\hat{g}$$

学习步长

这里：批容量为m

典型的机器学习步骤

- 典型的机器学习的思路都是这样的：
  - 从开始的通过传感器来获得数据。
  - 然后经过预处理、特征提取、特征选择，再到推理、预测或者识别。
  - 最后一个部分，也就是机器学习的部分，绝大部分的工作是在这方面做的。



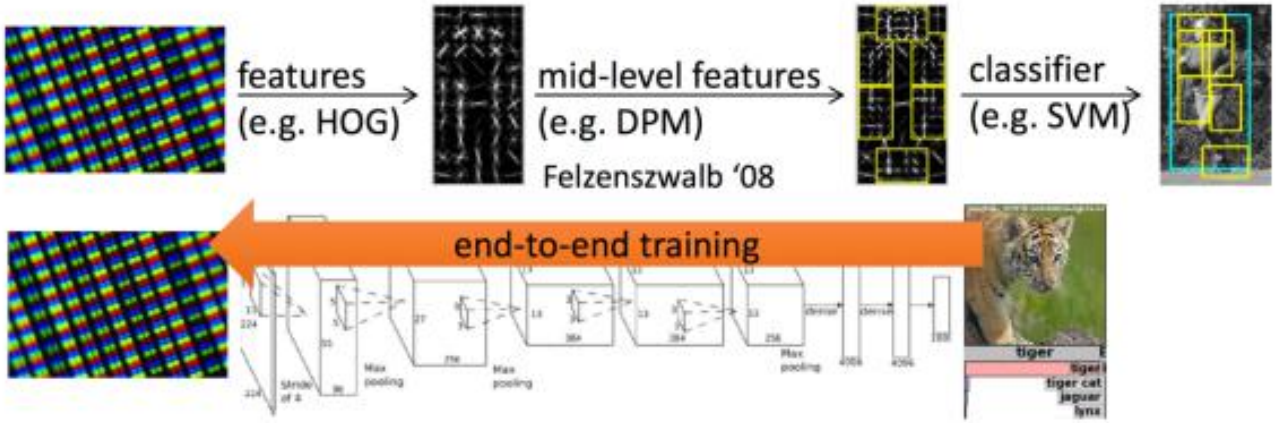
特征对学习的影响

- 一般而言，机器学习中特征越多，给出信息就越多，识别准确性会得到提升；
- 但特征多，计算复杂度增加，探索的空间就大，训练数据在全体特征向量中就会显得稀疏，影响相似性判断；
- 更重要的是，如果有对分类无益的特征，反而可能干扰学习效果。
- 结论：特征不一定越多越好，获得好的特征是识别成功的关键。需要有多少个特征，需要学习问题本身来决定。

深度学习的特征

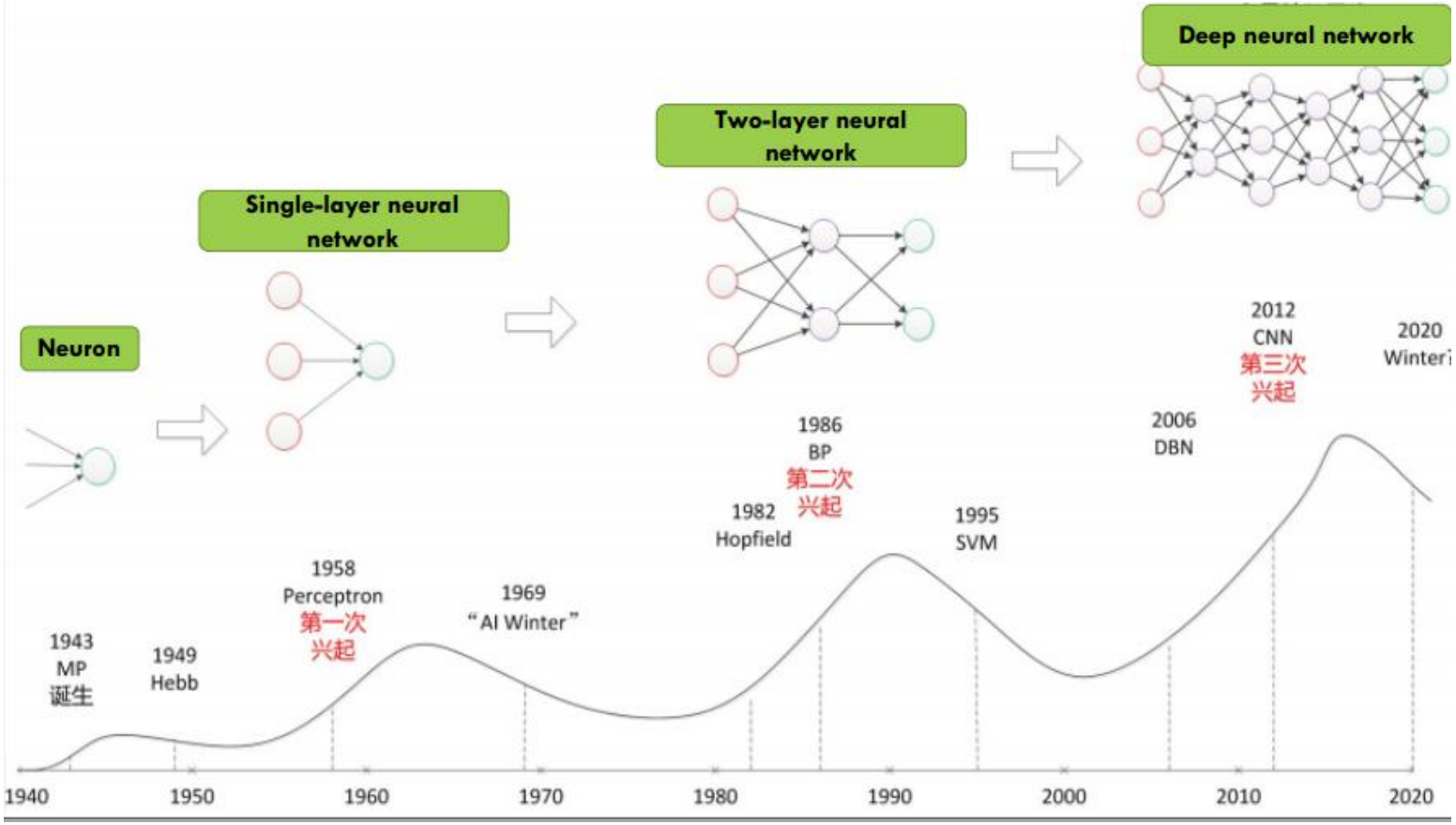
- 深度学习，是一种基于无监督特征学习和特征层次结构的学习模型，其实是对神经网络模型的拓展。

“端到端”的学习	高维度和大数据
输入原始数据，输出任何你想要的	高维的权重参数
利用算法自动提取的特征进行学习	海量的训练数据



“连接主义”的兴衰史

- 某种意义上说，AI的兴衰史，其实就是“连接主义”（感知机→神经网络→深度学习）的兴衰史。



小象学院 ChinaHadoop.cn

深度学习与神经网络的区别

- 网络架构
  - 3层以内→可达上千层
- 层间连接
  - 通常是全连接→形式多样：共享权值、跨层的反馈等
- 目标函数
  - MSE→CE等
- 激活函数
  - Sigmoid等→ReLU等
- 梯度下降的方法
  - GD等→Adam等
- 避免过适应
  - 凭经验→Dropout等

演示环节

- https://github.com/hunkim/DeepLearningZeroToAll/
- https://blog.csdn.net/u010089444/article/details/76725843

参考资料

- 李宏毅：Deep Learning Tutorial
- https://blog.csdn.net/u014595019/article/details/52989301
- https://blog.csdn.net/ccj_ok/article/details/77620738

问答互动

在所报课的课程页面，

1、点击“全部问题”显示本课程所有学员提问的问题。

2、点击“提问”即可向该课程的老师和助教提问问题。



联系我们

小象学院：互联网新技术在线教育领航者

– 微信公众号：**小象学院**

