机器学习

第 6 章人工神经网络

欧阳毅

浙江工商大学
管理工程与电子商务学院

2023 年 3 月 12 日

目录

单层感知器 I

- 前面讨论了单层的感知器网络，指出了单层感知神经网络的局限性，如果要实现非线性数据分类，有必要构造多层感知器网络。
- 在输入与输出层之间加上隐含层，从而构成多层感知器(Multilayer Perceptrons, MLP)。这种由输入层、隐含层(一层或者多层）和输出层构成的神经网络称为多层前向神经网络。

单层感知器的局限性

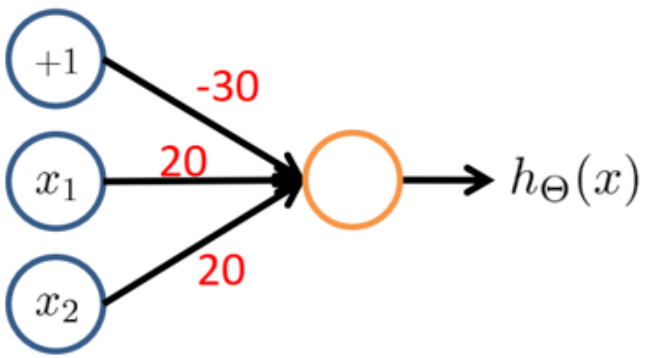
- 产生“与”函数

例一: 逻辑与（ AND ）

Example: AND

$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$



x_1 AND x_2

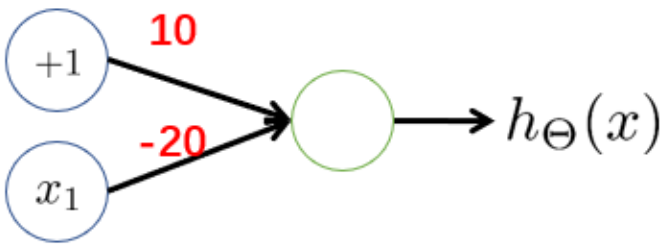
x_1	x_2	x1 AND x2
0	0	0
0	1	0
1	0	0
1	1	1

expected output

单层感知器的局限性

- 产生“非函数”

非函数: 一个输入，输出为相反的数。



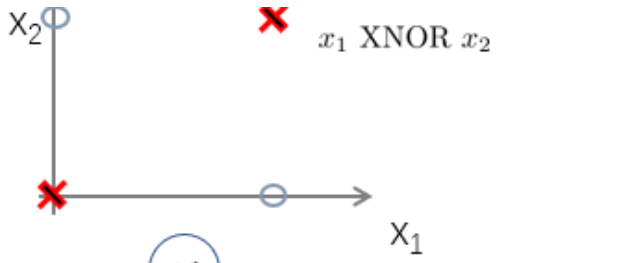
x_1	$h_\Theta(x)$
0	
1	

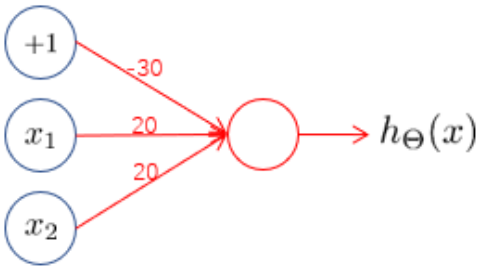
$$h_\Theta(x) = g(10 - 20x_1)$$

单层感知器的局限性

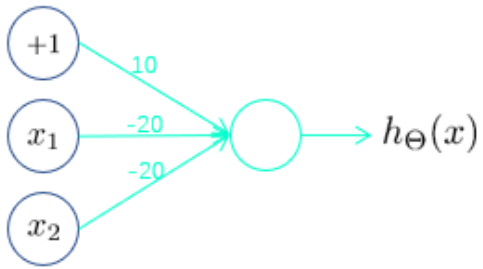
- 产生“异或非”函数

将之前的神经元组合起来，
得到非线性分类函数 x_1 XNOR x_2

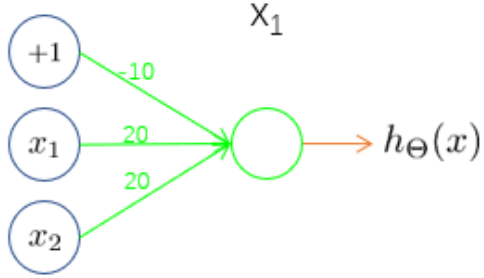




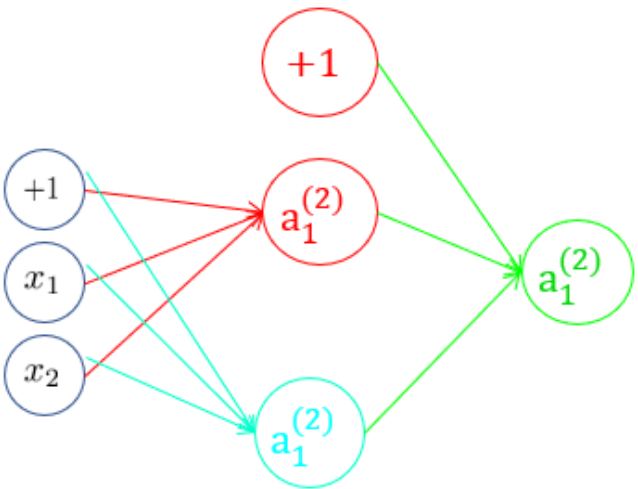
x_1 AND x_2



(NOT x_1) AND (NOT x_2)



x_1 OR x_2



$h_\theta(x)$

x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

单层感知器的局限性

- 设计“异或”函数
- 非线性分类例子:

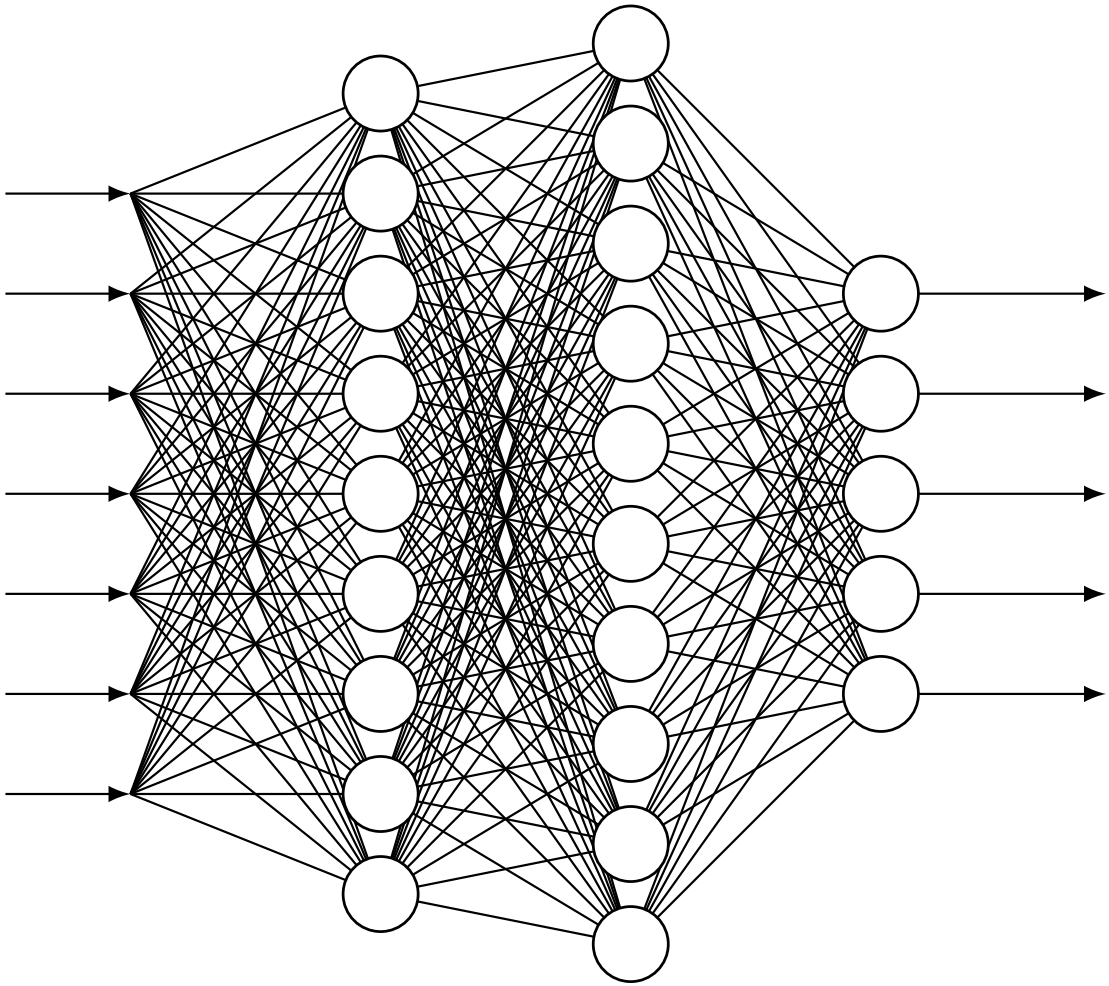
$$XOR = a \bigoplus b = (\neg a \wedge b) \vee a \wedge \neg b)$$

a	b	$a \bigoplus b$
1	0	1
1	1	0
0	0	0
0	1	1

目录

BP 神经网络-多层感知器

(1) 除了输入输出层，多层感知器含有一层或多层隐单元;
(2) 多层感知器具有独特的学习算法，该学习算法就是著名的后向传播算法 (Back-Propagation algorithm, BP)

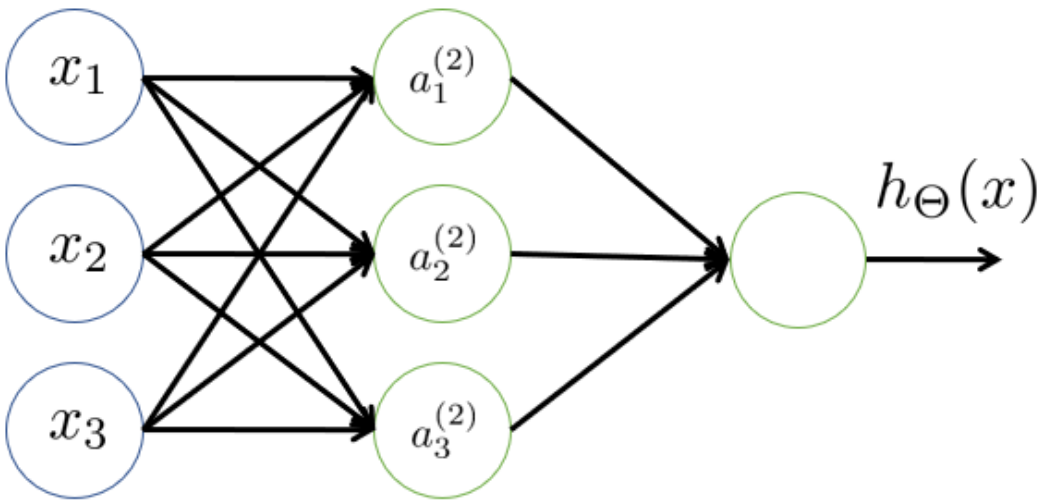


反向传播算法 (Back-Propagation algorithm, BP)

正向传播：输入信号从输入层经隐单元，传向输出层，在输出端产生输出信号，这是工作信号的正向传播。在信号的向前传递过程中网络的权值是固定不变的，每一层神经元的状态只影响下一层神经元的状态。

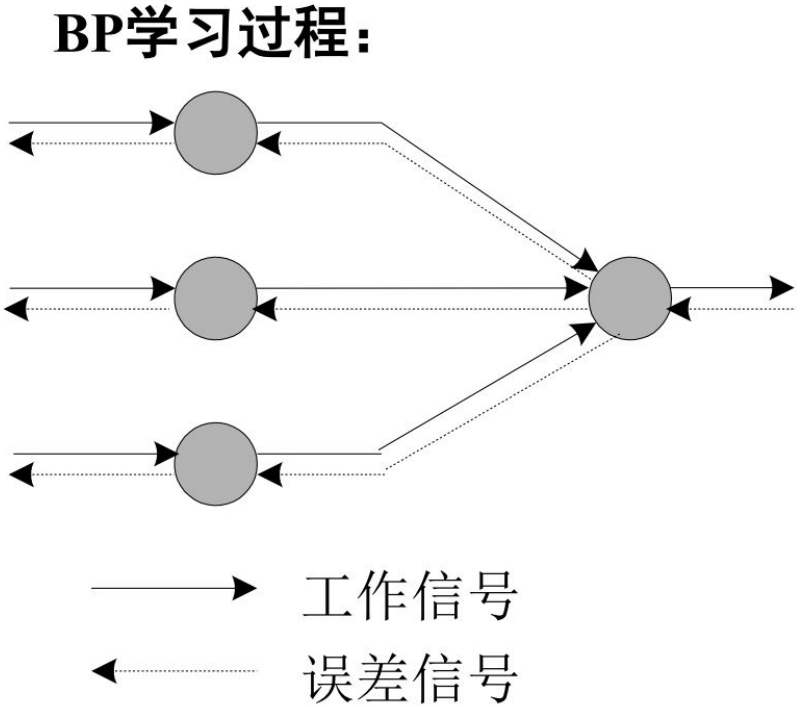
Neural Network

$a_i^{(j)}$ = 第j层的第i个单元的输出（activation）



反向传播算法 (Back-Propagation algorithm, BP)

反向传播：网络的实际输出与期望输出之间差值即为误差信号，误差信号由输出端开始逐层向后传播，这是误差信号的反向传播。在误差信号反向传播的过程中，网络的权值由误差反馈进行调节。通过权值的不断修正使网络的实际输出更接近期望输出。



反向传播算法 (Back-Propagation algorithm, BP)

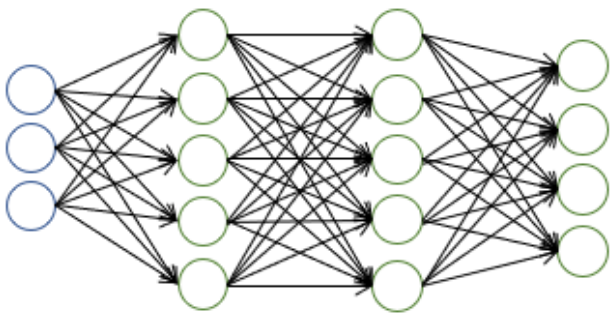
Multiple output units: One-vs-all.


行人


汽车


摩托车


卡车



$h_\Theta(x) \in \mathbb{R}^4$

我们希望得到 $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

当样本是行人 当样本是汽车 当样本是摩托车

反向传播算法 (Back-Propagation algorithm, BP)

神经单元的选择

- 1) 感知器训练中的输出

$$O(x) = sign(w \cdot x)$$

由于 sign 函数时非连续函数，这使得它不可微，因而不能使用梯度下降算法来最小化损失函数。

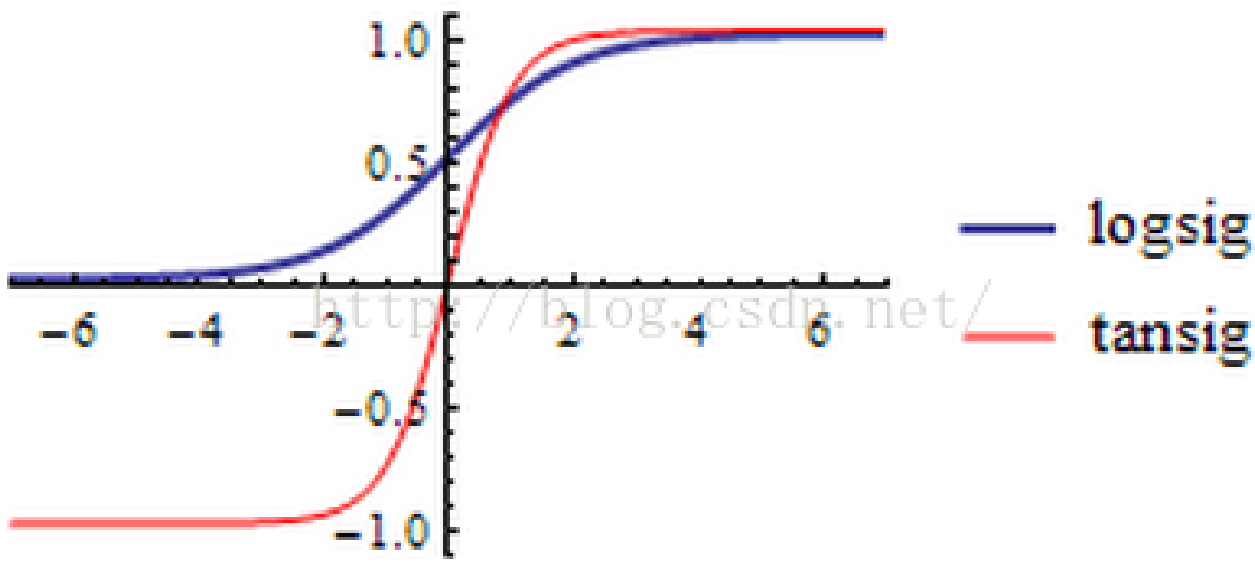
- 2) 每个输出都是输入的线性组合，这样当多个线性单元连接在一起后最终也只能得到输入的线性组合，这和只有一个感知器单元节点没有很大不同。

反向传播算法 (Back-Propagation algorithm, BP)

为了解决上面存在的问题，一方面，我们不能直接使用线性组合的方式直接输出，需要在输出的时候添加一个处理函数；另一方面，添加的处理函数一定要是可微的，这样我们才能使用梯度下降算法。

sigmoid 函数

- $f(x) = logsig(x) = \frac{1}{1+e^{-x}}, f'(x) = f(x) * (1 - f(x))$
- $g(x) = tansig(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, g'(x) = 1 - g(x) * g(x)$



反向传播

1

[1] $f(x) = logsig(x) = \frac{1}{1+e^{-x}}, f'(x) = f(x) * (1 - f(x))$

BP 网络的计算

设：(1) 有 N 个输入节点, 输入层节点的输出等于其输入;

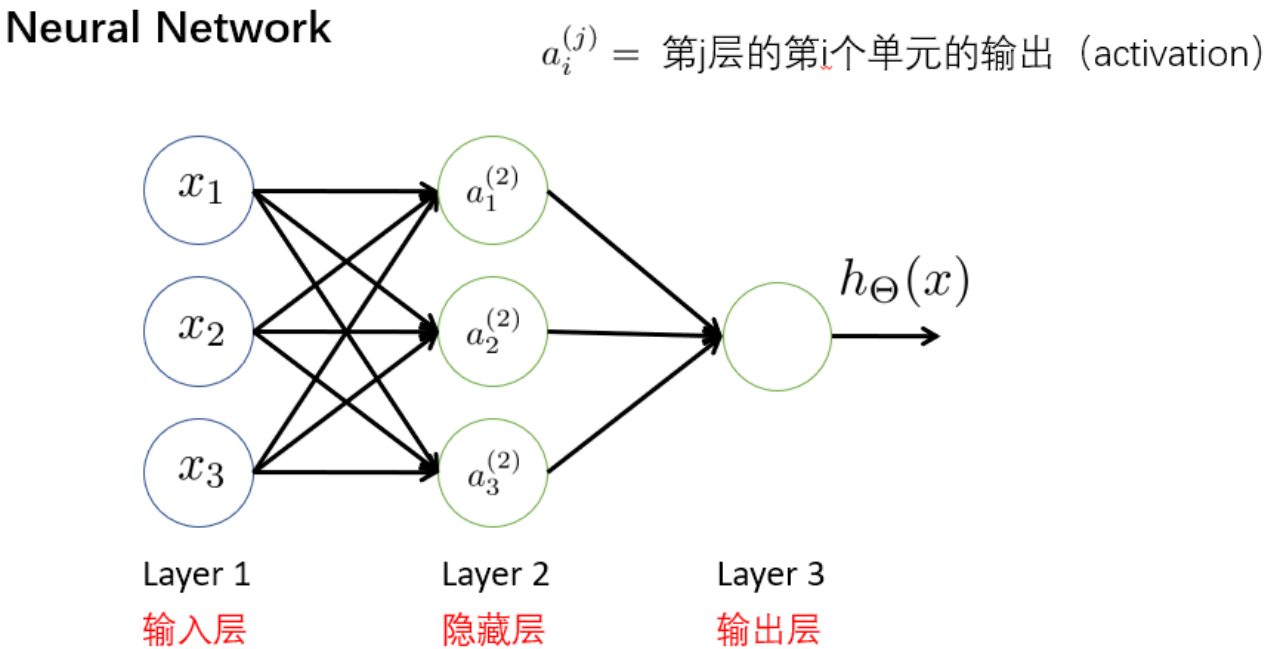
- (2) 网络的隐含层有 M 个节点,w_{ij} 是输入层和隐含层节点之间的连接权值;
- (3) 输出层有 L 个输出节点,w_{jk} 是隐含层和输出层节点之间的连接权值。

BP 神经网络的计算过程

- 当激发函数可微时，可采用 BP 算法进行学习。BP 算法分正向传播过程和反向传播过程。

1 在正向传播过程中，输入信息从输入层经隐含层逐层处理，并传向输出层。

2 如果在输出层不能得到期望的输出，将误差信号沿原来的连接通路返回，并修改各层神经元的权值，即转入反向传播计算



目录

BP 网络的前馈计算 I

- 输入层第 i 个节点的输出为：

$$O_i = x_i$$

- 隐含层第 j 个节点的输入和输出为：

$$I_j = \sum_{i=1}^{N} w_{ij} O_i$$

$$O_j = f(I_j)$$

- 输出层第 k 个节点的输入和输出为：

$$I_k = \sum_{j=1}^{M} w_{jk} O_j$$

$$O_k = f(I_k)$$

BP 学习算法

BP 学习算法的计算步骤如下:

- Python BP 前向传播代码分析

BP 学习算法-习题

- 设所有 w 的初值均为 0.1，x=[[1,2,3],[4,5,6]] ,y=[1,0]，求通过前向传播后的输出（$f(x) = logsig(x)$, 隐含层为 2 个节点）
- 要求：给出每个节点的输入和输出（只计算一趟）

目录

BP 神经网络的计算过程

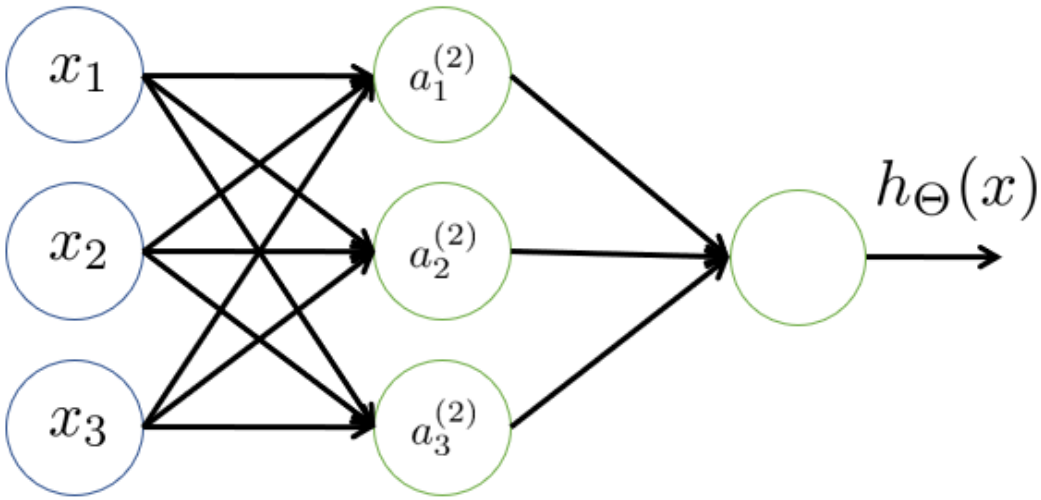
- 当激发函数可微时，可采用 BP 算法进行学习。BP 算法分正向传播过程和反向传播过程。

1 在正向传播过程中，输入信息从输入层经隐含层逐层处理，并传向输出层。

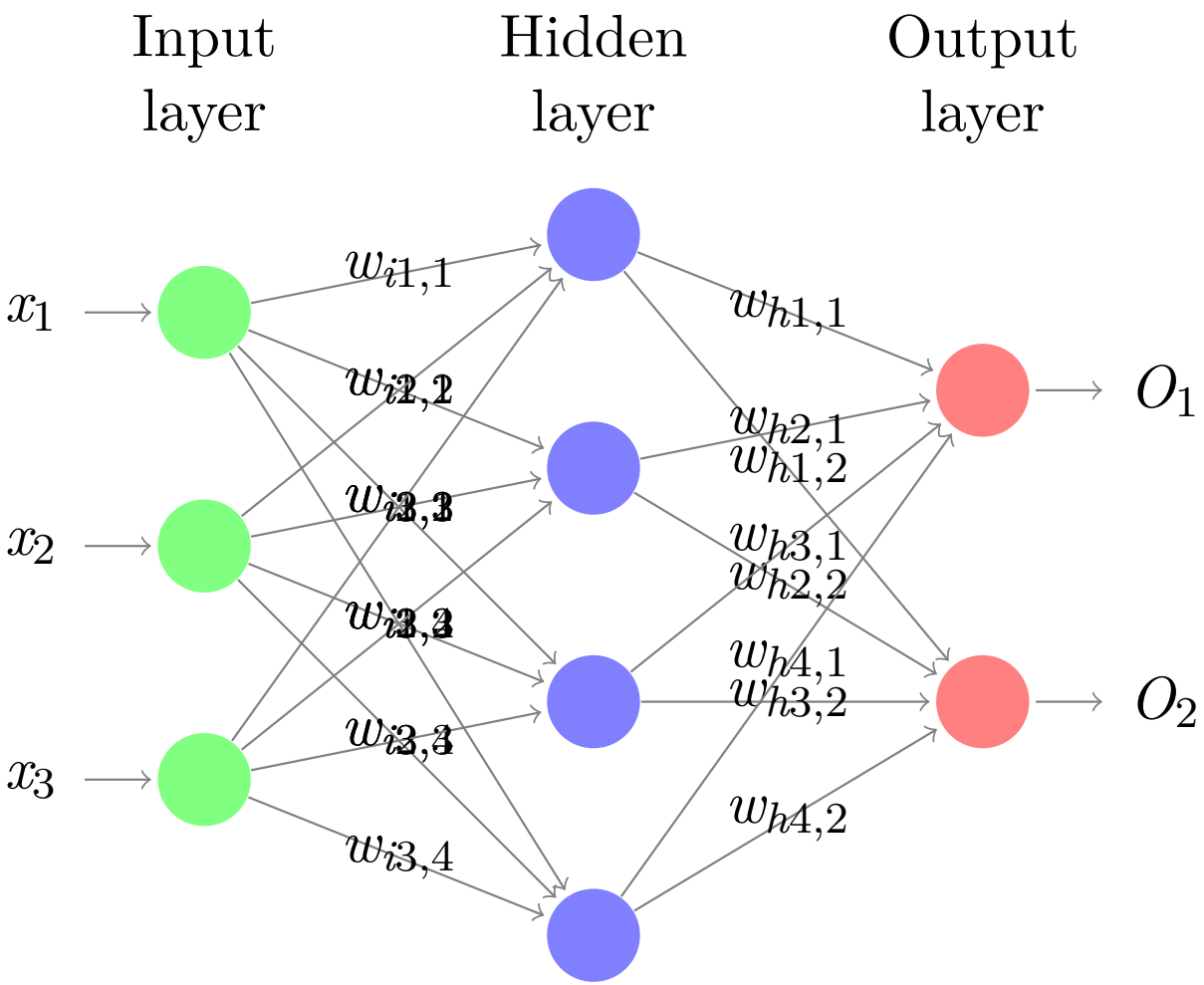
2 如果在输出层不能得到期望的输出，将误差信号沿原来的连接通路返回，并修改各层神经元的权值，即转入反向传播计算

Neural Network

$a_i^{(j)}$ = 第j层的第i个单元的输出（activation）



BP 网络的前馈计算 I


Input layer
Hidden layer
Output layer
x_1
x_2
x_3
$w_{i1,1}$
$w_{i1,2}$
$w_{i2,1}$
$w_{i2,2}$
$w_{i3,1}$
$w_{i3,2}$
$w_{i4,1}$
$w_{i4,2}$
$w_{i3,3}$
$w_{i3,4}$
$w_{h1,1}$
$w_{h1,2}$
$w_{h2,1}$
$w_{h2,2}$
$w_{h3,1}$
$w_{h3,2}$
$w_{h4,1}$
$w_{h4,2}$
O_1
O_2

BP 网络的前馈计算 II

- 输入层第 i 个节点的输出为：

$$O_i = x_i$$

- 隐含层第 j 个节点的输入和输出为：

$$I_j = \sum_{i=1}^{N} w_{ij} O_i$$

$$O_j = f(I_j)$$

- 输出层第 k 个节点的输入和输出为：

$$I_k = \sum_{j=1}^{M} w_{jk} O_j$$

$$O_k = f(I_k)$$

反向传播

$$M = 1 + e^{-x} \tag{1}$$
$$M' = -e^{-x} \tag{2}$$
$$f'(x) = -\frac{1}{M^2} * M' \tag{3}$$
$$= \frac{e^{-x}}{(1+e^{-x})^2} \tag{4}$$
$$= \frac{1+e^{-x}-1}{(1+e^{-x})^2} \tag{5}$$
$$= f(x) - (f(x))^2 \tag{6}$$

2

[2] $f(x) = logsig(x) = \frac{1}{1+e^{-x}}, f'(x) = f(x) * (1 - f(x))$

BP 网络权值得调整规划

- 设激发函数为 f(x)=sigmoid(x) 函数, 激发函数的微分可用激发函数表示为:

$$f'(x) = f(x)(1 - f(x))$$

- 当有 P 个样本模式时，系统的误差为:

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{L}(y_{pk} - O_{pk})^2 = \sum_{p=1}^{P} E_p \tag{7}$$

式中: y_{pk} 是第 p 个样本第 k 个输出的期望值; O_{pk} 是第 p 个样本第 k 个输出的当前值;

- 当样本确定之后，误差 E 仅与连接权值有关。为减少误差 E，连接权值 W 的增减应按照其负梯度方向改变，即:

$$\Delta W = -\eta \frac{\partial E}{\partial W} \tag{8}$$

BP 网络权值的调整规划 I

为简便起见，先考虑一个样本的情况，此时，误差函数可写为：

$$E = \frac{1}{2}\sum_{k=1}^{L}(y_k - O_k)^2$$

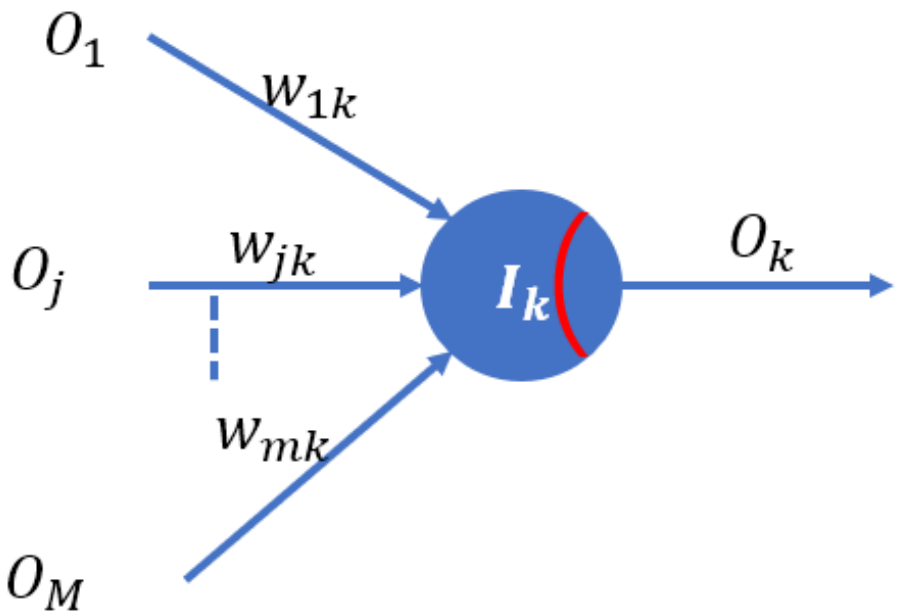
- 输出层权值的调整
  根据公式8, 隐含层第 j 个节点到输出层第 k 个节点的连接权值增量可表示为：

$$\Delta w_{jk} = -\eta\frac{\partial E}{\partial w_{jk}} = -\eta\frac{\partial E}{\partial I_k}\frac{\partial I_k}{\partial w_{jk}} = -\eta\frac{\partial E}{\partial I_k}O_j = \eta\delta_k O_j$$

$$\delta_k = -\frac{\partial E}{\partial I_k}$$

$$\because I_k = \sum_{j=1}^{M} w_{jk}O_j, \therefore \frac{\partial I_k}{\partial w_{jk}} = O_j$$

BP 学习算法



BP 网络权值的调整规划 I

又 $\delta_k = -\frac{\partial E}{\partial O_k}\frac{\partial O_k}{\partial I_k}$，考虑到输出层的第 k 个节点的输入和输出的关系为：$O_k = f(I_k)$。

$$\because \frac{\partial E}{\partial O_k} = -(y_k - O_k), \frac{\partial O_k}{\partial I_k} = f'(I_k) = f(I_k)(1 - f(I_k)) = O_k(1 - O_k)$$

$$\therefore \delta_k = O_k(1 - O_k)(y_k - O_k)$$

因此，隐含层到输出层的连接权值的修正公式为：

$$\Delta w_{jk} = \eta \delta_k O_j = \eta O_j O_k(1 - O_k)(y_k - O_k)$$

BP 学习算法

- 注意：输出层节点是有激活函数的

BP 网络权值的调整规划 I

- 隐含层权值的调整
  输入层第 i 个节点到隐含层第 j 个节点的连接权值增量可表示为:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial I_j} \frac{\partial I_j}{\partial w_{ij}} = \eta \delta_j O_i$$

★ $\delta_j = -\frac{\partial E}{\partial I_j} = -\frac{\partial E}{\partial O_j} \frac{\partial O_j}{\partial I_j}$

$$\because \frac{\partial E}{\partial O_j} = \sum_{k=1}^{L} \frac{\partial E}{\partial I_k} \frac{\partial I_k}{\partial O_j} = \sum_{k=1}^{L} \frac{\partial E}{\partial I_k} \frac{\partial}{\partial O_j} (\sum_{j=0}^{M} (w_{jk} O_j)) = -\sum_{k=1}^{L} \delta_k w_{jk}$$

- E 到 O_j 中间经过 I_k

目录

BP 学习算法

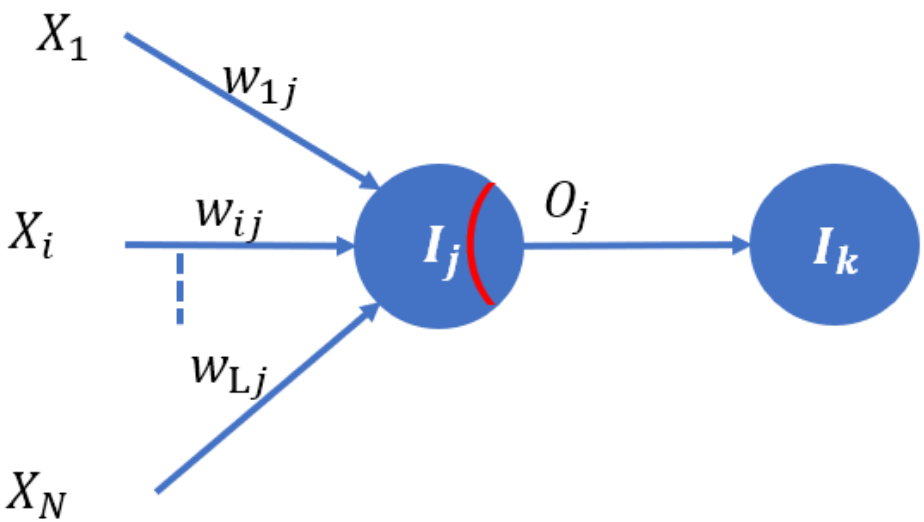
- 输入层第 i 个节点的输出为：

$$O_i = x_i$$

- 隐含层第 j 个节点的输入和输出为：

$$I_j = \sum_{i=1}^{N} w_{ij} O_i$$

$$O_j = f(I_j)$$



BP 网络权值的调整规划 I

- 隐含层权值的调整

$$\frac{\partial O_j}{\partial I_j} = f'(I_j) = f(I_j)(1 - f(I_j)) = O_j(1 - O_j)$$

$$\therefore \delta_j = O_j(1 - O_j) \sum_{k=1}^{L} \delta_k w_{jk}$$

因此，输入层到隐含层的连接权值的修正公式为：

$$\Delta w_{ij} = \eta \delta_j O_i = \eta O_i O_j(1 - O_j) \sum_{k=1}^{L} \delta_k w_{jk}$$

- 无论是输出层还是隐含层，神经元连接权值的修正公式都为：

$$\Delta w_{xy} = \eta \delta_y O_x$$

BP 网络权值的调整规划 II

- 当考虑多个样本时，修正公式为：

$$\Delta w_{xy} = \eta \sum_{p=1}^{P} \delta_{py} O_{px}$$

BP 学习算法

BP 学习算法的计算步骤如下:

- Python BP 反向传播代码分析

BP 学习算法

BP 学习算法的计算步骤如下:

1 连接权值初始化;

2 给定 P 个训练样本;

前向 3 计算各层各神经元的输出;

后向 4 计算各神经元连接权值的修正增量;

5 对 P 个训练样本都执行 3-4 步骤

6 计算出总的误差和总的修正增量;

7 检查总的误差是否满足精度要求, 不满足重复 3-6 步骤。

BP 学习算法-习题

- 以公式 $g(x) = tansig(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, g'(x) = 1 - g(x) * g(x)$ 为激活函数，求 BP 网络输出层和隐含层的权重调整公式。