4 人工神经网络 (Artificial Neural Network)

代启国

大连民族大学
计算机科学与技术系

2018 年 11 月 30 日

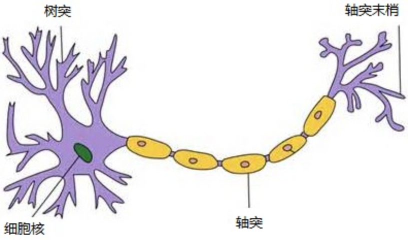
人工神经网络

人工神经网络（Artificial Neural Network）

- 模拟生物神经系统对真实世界物体所做出的交互反应
- 由简单“神经元模型”组成的互联的网络

人工神经网络（Artificial Neural Network）

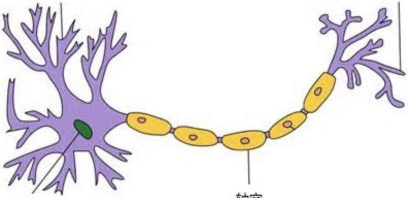
- 模拟生物神经系统对真实世界物体所做出的交互反应
- 由简单“神经元模型”组成的互联的网络



生物神经元

人工神经网络（Artificial Neural Network）

- 模拟生物神经系统对真实世界物体所做出的交互反应
- 由简单“神经元模型”组成的互联的网络

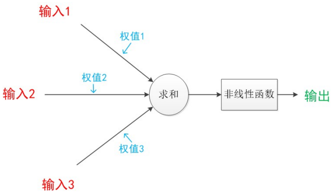


生物神经元

- 在生物神经网络中，每个神经元与其它神经元相连
- 每个神经元通过“树突”接受外界或其它神经元的信号（输入）
- ”轴突”对输入信号进行处理
- 如果信号超过一定阈值，视为”兴奋”
- 通过”轴突末梢”向其它神经元输出信号

人工神经网络（Artificial Neural Network）

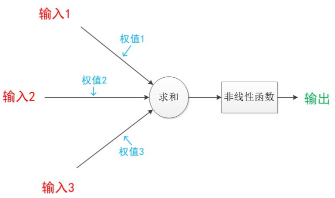
- 模拟生物神经系统对真实世界物体所做出的交互反应
- 由简单“神经元模型”组成的互联的网络



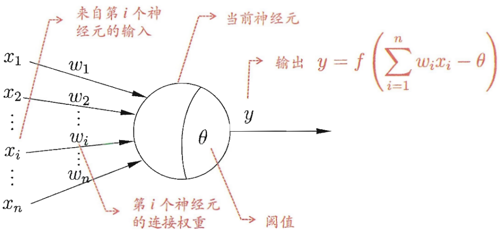
人工神经元模型

人工神经网络（Artificial Neural Network）

- 模拟生物神经系统对真实世界物体所做出的交互反应
- 由简单“神经元模型”组成的互联的网络



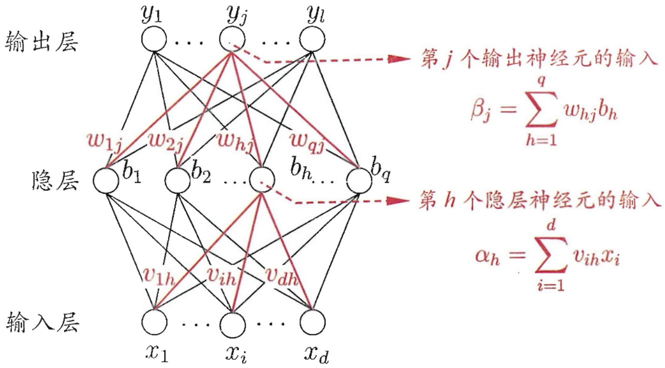
人工神经元模型



人工神经元模型

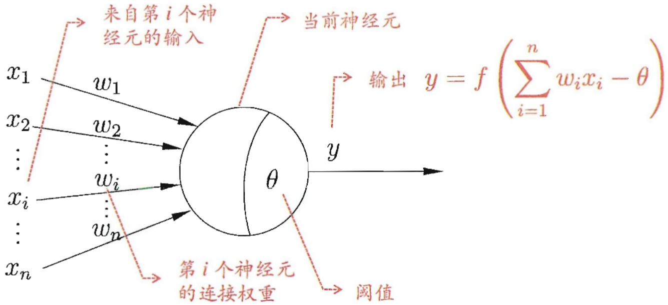
人工神经网络（Artificial Neural Network）

- 把许多简单的神经元模型，按照一定层次结构连接起来



M-P 神经元模型

- 接受来自其他 n 个神经元传递来的带权重的信号
- 将总输入值与自身阈值（偏置）进行比较
- 然后通过激活函数（activation function）$f(\cdot)$产生输出



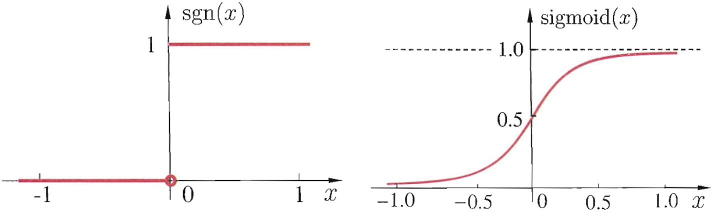
激活函数（activation function）

- 传统的激活函数目标是把输入值映射为 0-1 输出
  - 阶跃函数性质不好：不连续、不光滑
  - Sigmoid 函数可以把输出值挤压到 $(0, 1)$ 区间

人工神经元模型

激活函数（activation function）

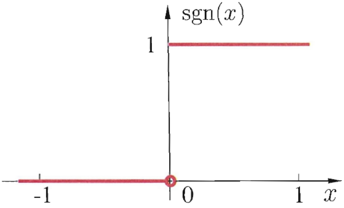
- 传统的激活函数目标是把输入值映射为 0-1 输出
  - 阶跃函数性质不好：不连续、不光滑
  - Sigmoid 函数可以把输出值挤压到 $(0, 1)$ 区间

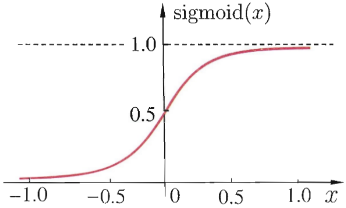


人工神经元模型

激活函数（activation function）

- 传统的激活函数目标是把输入值映射为 0-1 输出
  - 阶跃函数性质不好：不连续、不光滑
  - Sigmoid 函数可以把输出值挤压到 $(0,1)$ 区间



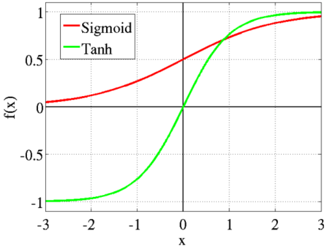


$$\operatorname{sgn}(x) = \begin{cases} 1, & x \geqslant 0; \\ 0, & x < 0. \end{cases}$$

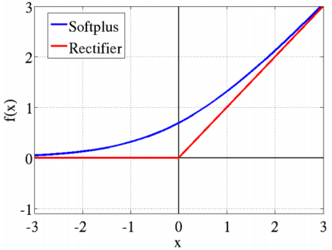
$$\operatorname{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

激活函数（activation function）

- 近年来，出现了一些新的激活函数
  - ReLU 函数、tanh 函数等

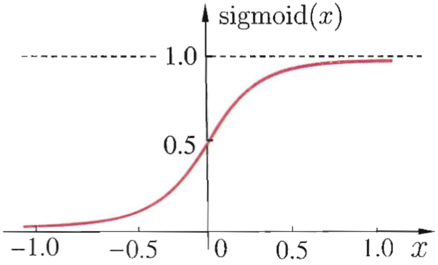


tanh 激活函数



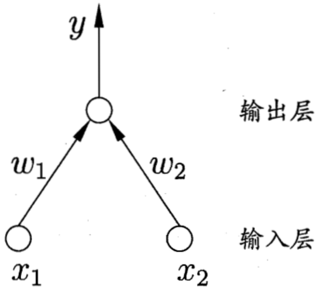
ReLU 激活函数

激活函数（activation function）



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

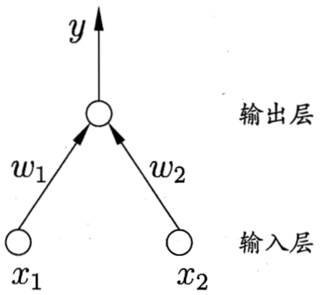
单层感知机网络

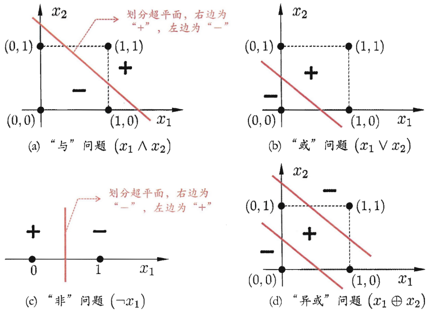


激活函数为阶跃函数
其训练过程的基本思想：

- 比较模型输出值和实际值：$(\hat{y} - y)$
- 利用该差值修改权值 w_1, w_2 等参数
- $w_i \leftarrow w_i + \Delta w_i$
  $\Delta w_i = \eta\,(\hat{y} - y)\,x_i$
  其中，η 为学习率

单层感知机网络

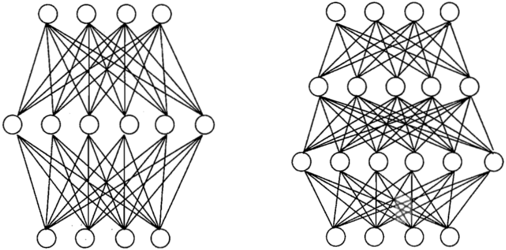




这种单层模型解决不了“线性不可分”问题

多层前馈神经网络 (multi-layer feedforward neural network)

- 也称多层感知机（multi-layer perceptron）
- 输入层（input layer）、隐层（hidden layer）、输出层（output layer）
- 隐层、输出层中的神经元具有激活函数
- 输入层：仅是接受输入，不具有处理功能（对应样本的特征 X)

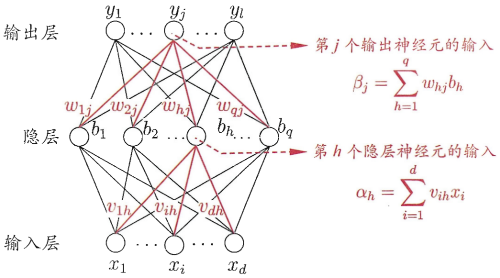


(a) 单隐层前馈网络　(b) 双隐层前馈网络

多层神经网络可以解决“线性不可分问题”

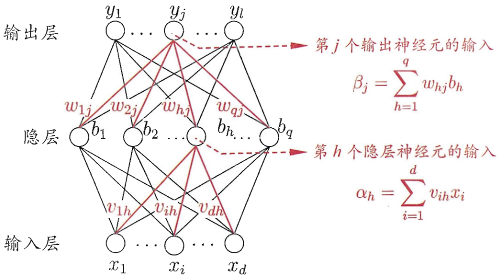
多层前馈神经网络 (multi-layer feedforward neural network)

- 本质上，神经网络是一个由许多参数构成的数学模型
- 这个模型是由多个激活函数 ($y_i = f(\sum_i w_i x_i - \theta_i)$) 相互嵌套代入而得

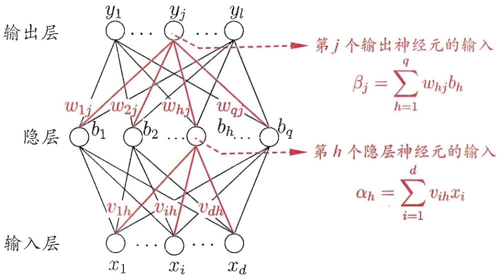


人工神经网络——训练

- 给定样本集 $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \cdots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}$
  其中，$\boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{y}_i \in \mathbb{R}^l$
- 输入样本由 d 个实值属性描述，输出为 l 维实值向量
  - 如果样本包含离散属性，需要进行连续化 (有序) 或向量 (无序)



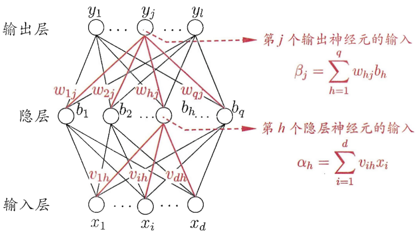
- 给定样本集 $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \cdots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}$
  其中，$\boldsymbol{x}_i \in \mathbb{R}^d$,$\boldsymbol{y}_i \in \mathbb{R}^l$
- 输入样本由 d 个实值属性描述，输出为 l 维实值向量
  - 如果样本包含离散属性，需要进行连续化 (有序) 或向量 (无序)



网络中一共有多少个参数需要学习???

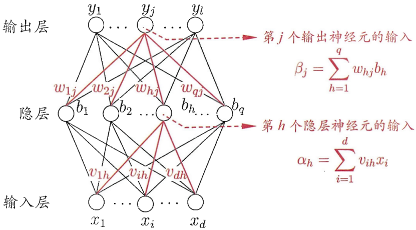
误差逆传播算法（error BackPropagation，BP）

- BP 算法是迄今最成功、应用最广泛的神经网络学习算法



误差逆传播算法（error BackPropagation，BP）

- BP 算法是迄今最成功、应用最广泛的神经网络学习算法



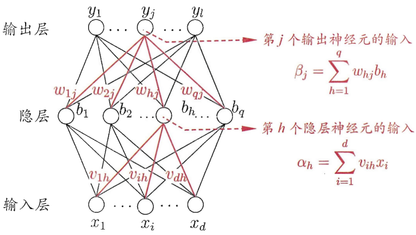
- 对于训练样本 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$，假定神经网络的输出值为

$$\hat{\boldsymbol{y}}_k = \left(\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k\right)$$

其中，$\hat{y}_i^k = f(\beta_j - \theta_j)$

误差逆传播算法（error BackPropagation，BP）

- BP 算法是迄今最成功、应用最广泛的神经网络学习算法



- 对于训练样本 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$，假定神经网络的输出值为

$$\hat{\boldsymbol{y}}_k = \left(\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k\right)$$

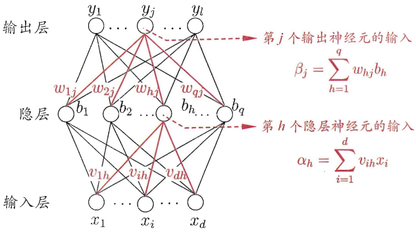
其中，$\hat{y}_i^k = f(\beta_j - \theta_j)$

- 则网络在该样本上的误差为

$$E_k = \frac{1}{2}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2$$

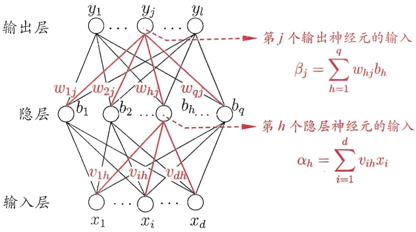
误差逆传播算法（error BackPropagation，BP）

- BP 算法是一种迭代算法，每次迭代中对模型中的参数进行更新



误差逆传播算法（error BackPropagation，BP）

- BP 算法是一种迭代算法，每次迭代中对模型中的参数进行更新

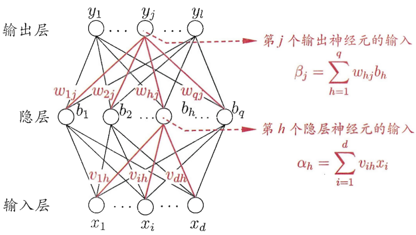


- 给定该样本上的误差为

$$E_k = \frac{1}{2}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2$$

误差逆传播算法（error BackPropagation，BP）

- BP 算法是一种迭代算法，每次迭代中对模型中的参数进行更新



- 给定该样本上的误差为

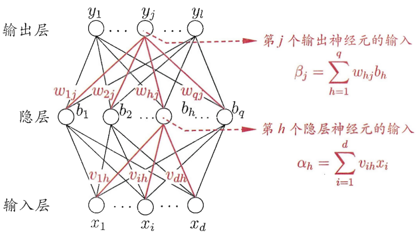
$$E_k = \frac{1}{2} \sum_{j=1}^{l} \left(\hat{y}_j^k - y_j^k \right)^2$$

- 对于任意参数 v 更新可表示为：

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

误差逆传播算法（error BackPropagation，BP）

- BP 算法是一种迭代算法，每次迭代中对模型中的参数进行更新



- 给定该样本上的误差为

$$E_k = \frac{1}{2}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2$$

- 对于任意参数 v 更新可表示为：

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

如何计算 Δv???

误差逆传播算法（error BackPropagation，BP）

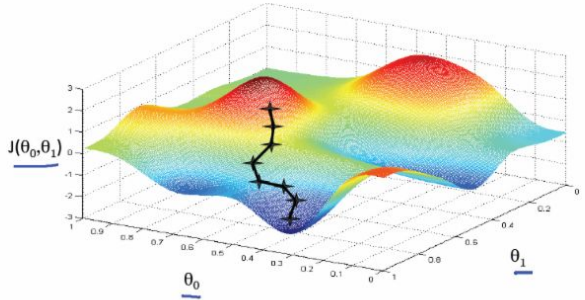
- BP 算法以最小化误差作为目标，以目标的负梯度方向对参数进行调整

误差逆传播算法（error BackPropagation，BP）

- BP 算法以最小化误差作为目标，以目标的负梯度方向对参数进行调整
- 梯度下降（gradient descent）策略

误差逆传播算法（error BackPropagation，BP）

- BP 算法以最小化误差作为目标，以目标的负梯度方向对参数进行调整
- 梯度下降（gradient descent）策略
- 什么是梯度?
  - 梯度是一个向量（矢量），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（为该梯度的模）
  - “水往低处流”

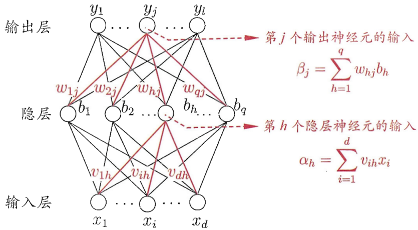


误差逆传播算法（error BackPropagation，BP）

- 某样本在网络上的误差为

$$E_k = \frac{1}{2}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2$$

- 以更新参数 w_{hj} 为例，即 $w_{hj} \leftarrow w_{hj} + \Delta w_{hj}$



人工神经网络——训练

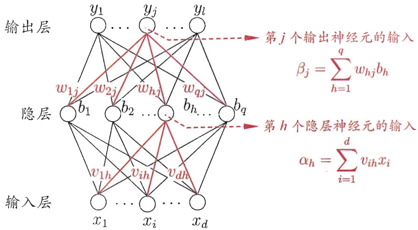
误差逆传播算法（error BackPropagation，BP）

- 某样本在网络上的误差为

$$E_k = \frac{1}{2} \sum_{j=1}^{l} \left(\hat{y}_j^k - y_j^k \right)^2$$

- 以更新参数 w_{hj} 为例，即 $w_{hj} \leftarrow w_{hj} + \Delta w_{hj}$

- $$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$



误差逆传播算法（error BackPropagation，BP）

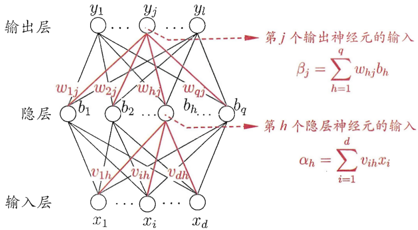
- 某样本在网络上的误差为

$$E_k = \frac{1}{2}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2$$

- 以更新参数 w_{hj} 为例, 即 $w_{hj} \leftarrow w_{hj} + \Delta w_{hj}$

- $$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

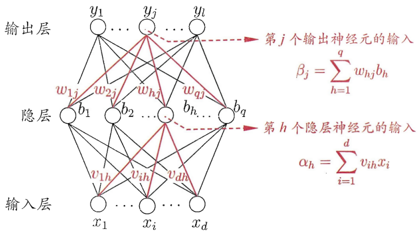
- $$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$



误差逆传播算法（error BackPropagation，BP）

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

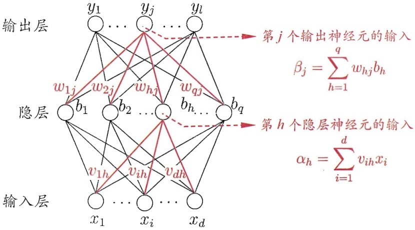


误差逆传播算法（error BackPropagation，BP）

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

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

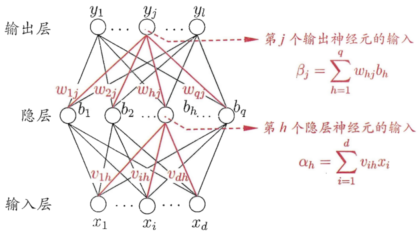


误差逆传播算法（error BackPropagation，BP）

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

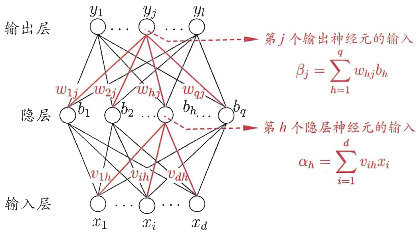
- Sigmoid 性质
  $f'(x) = f(x)(1 - f(x))$
- 令 $g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}$



误差逆传播算法（error BackPropagation，BP）

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$



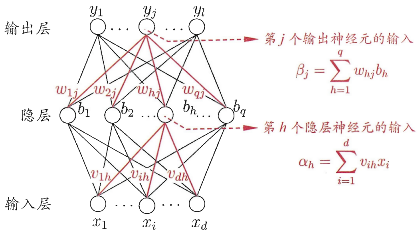
- Sigmoid 性质
  $f'(x) = f(x)(1 - f(x))$
- 令 $g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}$
- 可得 $g_j = \hat{y}_j^k(1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k)$

人工神经网络——训练

误差逆传播算法（error BackPropagation，BP）

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

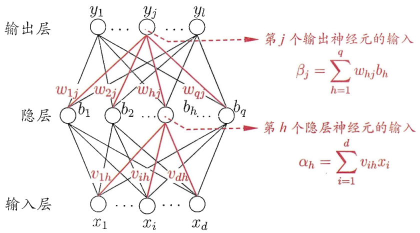


- Sigmoid 性质
  $f'(x) = f(x)(1 - f(x))$
- 令 $g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}$
- 可得 $g_j = \hat{y}_j^k(1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k)$
- 则有 $\Delta w_{hj} = \eta g_j b_h$

误差逆传播算法（error BackPropagation，BP）

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$



- Sigmoid 性质
  $f'(x) = f(x)(1 - f(x))$
- 令 $g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}$
- 可得 $g_j = \hat{y}_j^k(1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k)$
- 则有 $\Delta w_{hj} = \eta g_j b_h$

- 类似地，可以求得
- 第 j 个神经元的阈值：
  $\Delta\theta_j = -\eta g_j$

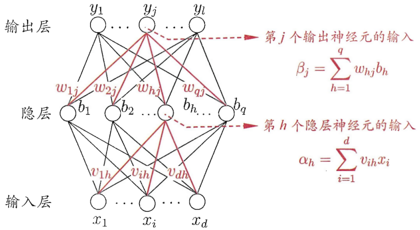
误差逆传播算法（error BackPropagation，BP）

- 将误差 E_k 继续向下层传播，可计算输入层与隐层之间权重的更新值

$$\Delta v_{ih} = \eta e_h x_i$$

- 和隐层中第 h 个神经元阈值的更新值

$$\Delta \gamma_h = -\eta e_h$$



- 上式中

$$e_h = -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} = -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h) = b_h(1-b_h) \sum_{j=1}^{l} w_{hj} g_j$$

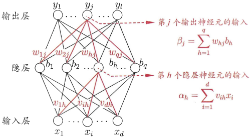
误差逆传播算法（error BackPropagation，BP）

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

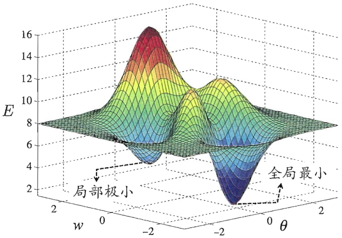
输出: 连接权与阈值确定的多层前馈神经网络



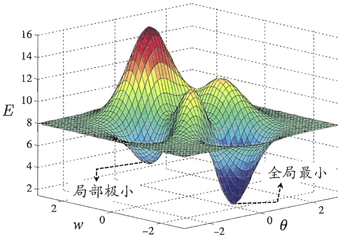
误差逆传播算法（error BackPropagation，BP）

- 上述算法是标准 BP 算法，针对每一个训练样本更新所有参数；
- 批量梯度下降算法：还可根据样本集 D 所有样本的累计误差进行参数更新；
- 小批量梯度下降算法：把 D 划分为若干批次（batch），根据每一批次的累计误差更新参数；
- epoch 概念：一个 epoch 过程 $=$ 正向输出 $+$ 反向传播

全局最小与局部最小

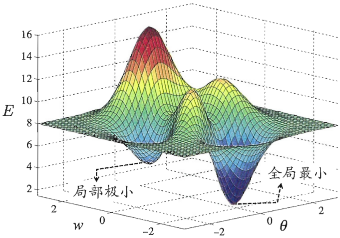


全局最小与局部最小



如何跳出局部最优??

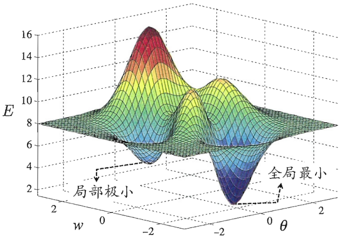
全局最小与局部最小



如何跳出局部最优??

- 以不同参数值初始化多个神经网络，分别训练每个网络，然后取其中精度最高的参数配置为最终参数

全局最小与局部最小

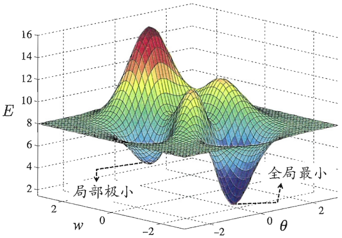


如何跳出局部最优??

- 以不同参数值初始化多个神经网络，分别训练每个网络，然后取其中精度最高的参数配置为最终参数
- 随机梯度下降 (Stochastic gradient descent, SGD)

人工神经网络——训练

全局最小与局部最小



如何跳出局部最优??

- 以不同参数值初始化多个神经网络，分别训练每个网络，然后取其中精度最高的参数配置为最终参数
- 随机梯度下降 (Stochastic gradient descent, SGD)
- 模拟退火算法（Simulated annealing）
- 遗传算法（Genetic algorithm）
- 蚁群算法 (Ant colony algorithm)
- ...