河北师范大学软件学院
Software College of Hebei Normal University

分类模型

第二部分 非线性分类模型

BP神经网络

张朝晖

2018-2019学年 20181009

河北师范大学软件学院
Software College of Hebei Normal University

1.分类模块

产生式分类模型 A.贝叶斯分类模型

判别式分类模型 **线性分类模型**
- B. Fisher判别分类
- C. 感知器分类模型
- D. 大间隔分类模型（线性*SVM*）

非线性分类模型
- E. 核*SVM*（非线性*SVM*）
- F. 核Fisher判别分类
- G. 神经网络

其它分类模型
- *H*.KNN分类模型
- *I*.决策树分类模型
- *J*.*Logistic*回归
- *K*.*Softmax*回归

2.聚类模块
- *L*.K-均值聚类
- *M*.高斯混合聚类
- *N*.DBSCAN聚类
- *O*.层次聚类

3.回归模块
- *P*.KNN回归
- *Q*.回归树
- *R*.最小二乘线性回归
- *S*.岭回归
- *T*.LASSO回归

4.集成学习
- *U*.*Bagging*
- *V*.随机森林
- *W*.*Boosting*

5.特征工程
- *X*.主成分分析*(PCA)*
- …

6.评价模块 混淆矩阵（及其相关指标）、ROC曲线、交叉验证

人工神经网络

--前馈神经网络与BP算法

关键词：

人工神经网络

前馈神经网络、人工神经元、激活函数

BP神经网络：梯度下降法、导数链式法则、误差反向传播

自编码器

监督式学习、非监督式学习

分类、回归、表示学习

多层BP神经网络学习

第一阶段：基于逐层表示学习的参数初始化；

第二阶段：基于BP算法的参数精调

主要内容

1. 人工神经网络基本知识、神经元与感知器

生物神经网络、生物神经元

人工神经网络、人工神经网络的基本模型、人工神经元

2. 前馈神经网络、多层感知器、及非线性分类

单层感知器、多层感知器、基于多层前馈神经网络的函数逼近

3. BP神经网络(可用于分类、回归、监督式特征学习)

监督式学习+BP算法(梯度下降法+导数链式法则)

3.1 BP网络的参数学习

3.2 BP网络的应用

4. 基于前馈神经网络的自编码器(Autoencoders)

逐层学习、非监督式学习+BP算法

已知结构的前馈神经网络的学习

利用给定训练样本集，估计相邻层节点之间连接参数、各神经元节点的偏置量。

基于阈值神经元的前馈神经网络的不足

- 隐含层不直接与外界连接，误差无法直接估计
- 神经元节点的激活函数为阈值函数(或阶跃函数)时，无法采用梯度下降法学习神经网络的有关参数。

BP网络：基于BP(Backward Propagation)算法进行参数学习的前馈神经网络

- 神经元节点传递函数：以连续(可微)函数代替阈值函数

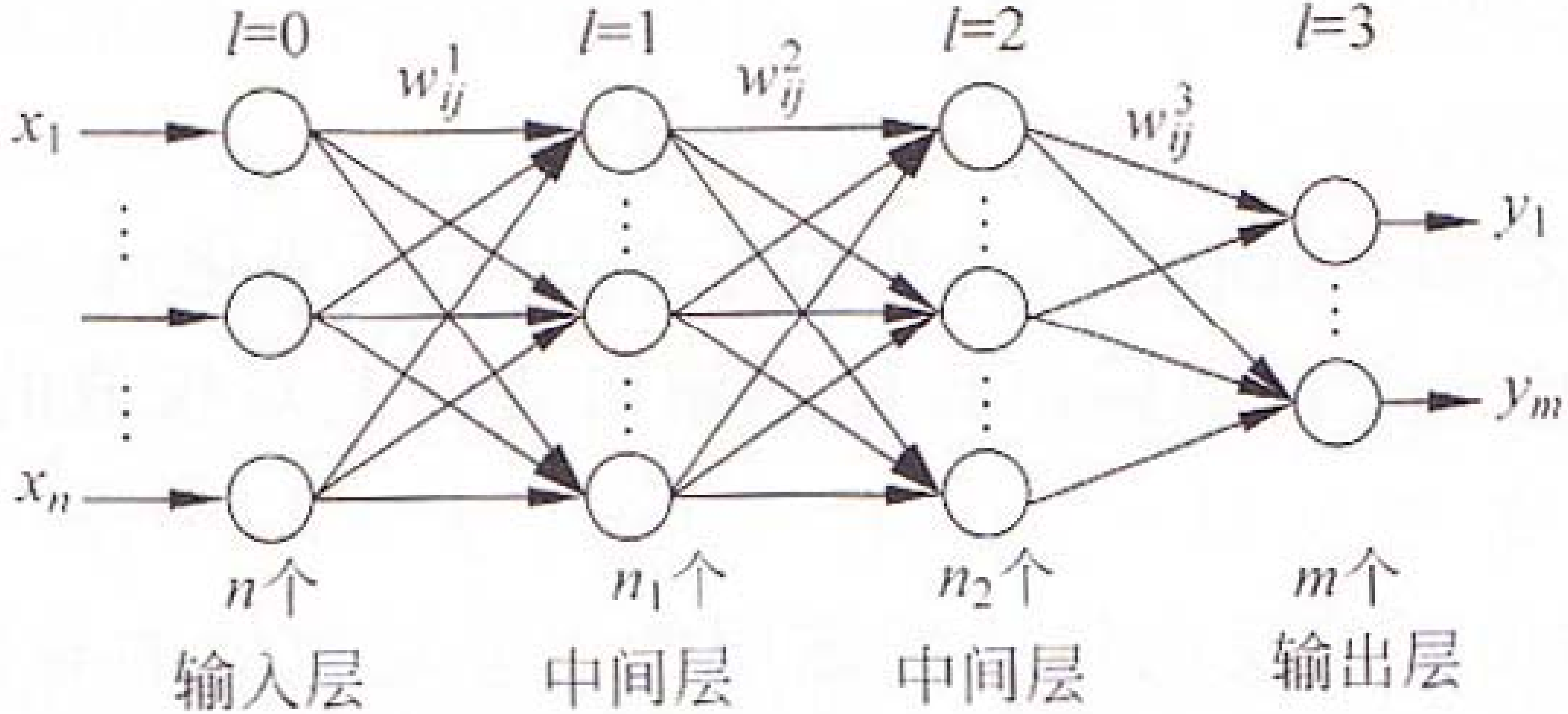
- BP网络的学习：　监督式学习

- 目标函数的构造：基于训练样本集的预测损失函数

- 目标函数的参数寻优：

  BP算法＝梯度下降法＋导数链式法则

- BP网络的用途：实值函数的回归、分类、(特征提取)



多层感知器的例子和符号约定

神经网络模型训练目的在于：

基于给定的样本集$X=\{(x_i, \mathrm{D}_i), i=1,\cdots,N\}$

$$\text{学习参数}\begin{cases}\text{网络连接权} & \boldsymbol{W}^l_{n_{l-1}\times n_l}=\left[\omega^l_{ij}\right]_{n_{l-1}\times n_l} \\ \text{神经元节点偏置量} & \boldsymbol{b}^l=\left[b^l_j\right]_{1\times n_l} \\ l=1,\ldots,L-1 \quad i=1,\ldots,n_{l-1} & j=1,\ldots,n_l\end{cases}$$

$$\Rightarrow\begin{cases}\boldsymbol{W}=\left\{\boldsymbol{W}^l_{n_{l-1}\times n_l}=\left[\omega^l_{ij}\right]_{n_{l-1}\times n_l}, l=2,\ldots,L-1\right\} \\ \boldsymbol{b}=\left\{\boldsymbol{b}^l=\left[b^l_j\right]_{1\times n_l}, l=2,\ldots,L-1\right\}\end{cases}$$

目标函数的构造：

$$E(\boldsymbol{W},\boldsymbol{b}) = L(\boldsymbol{W},\boldsymbol{b}) + \lambda\Omega(\boldsymbol{W})$$

$$\begin{cases} L(\boldsymbol{W},\boldsymbol{b})\text{--关于训练样本集的学习能力} \\ \Omega(\boldsymbol{W}) \quad \text{--正则项} \end{cases}$$

例：$L(\boldsymbol{W},\boldsymbol{b})$的形式

T

(1)**预测残差的平均平方和(MSE)**——回归，分类：

$$L(\boldsymbol{W},\boldsymbol{b})=\frac{1}{2N}\sum_{i=1}^{N}\left\|\boldsymbol{Y}(\boldsymbol{x}_i;\boldsymbol{W},\boldsymbol{b})-\boldsymbol{D}_i\right\|^2$$

(2)**交叉熵(CrossEntropy)**——两类别分类：

$$L(\boldsymbol{W},\boldsymbol{b})=-\frac{1}{N}\sum_{i=1}^{N}\left[d_i\ln y_i+(1-d_\mathrm{i})\ln(1-y_i)\right]$$

输出层只含一个输出节点,目标输出值$d_i\in\{0,1\}$，预测输出$y_i\in(0,1)$

(3)**交叉熵(CrossEntropy)**——C类别分类： $L(\boldsymbol{W},\boldsymbol{b})=-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\left(d_{ij}\ln y_{ij}\right)$

$\boldsymbol{Y}_i=\begin{bmatrix}y_{i1} & \cdots & y_{ic}\end{bmatrix}^T$为输出层关于样本$\boldsymbol{x}_i$的$C$类**预测输出(概率)**.

$\boldsymbol{D}_i=\begin{bmatrix}d_{i1} & \cdots & d_{ic}\end{bmatrix}^T$为输出层关于样本$\boldsymbol{x}_i$的$C$类**目标输出(概率)**.

$d_{ij}\in\{0,1\}$，$y_{ij}\in(0,1)$

若训练样本$\boldsymbol{x}_i$来自第1类，则$\boldsymbol{D}_i=\begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}$

$\Omega(\boldsymbol{W})$的形式：$\Omega(\boldsymbol{W})=\|\boldsymbol{W}\|_F^2=\sum_{l,i,j}\left(\omega_{ij}^l\right)^2$

$$E(\boldsymbol{W},\boldsymbol{b}) = L(\boldsymbol{W},\boldsymbol{b}) + \lambda\Omega(\boldsymbol{W})$$

神经网络的学习： $\left[\widehat{\boldsymbol{W}},\widehat{b}\right] = \arg\min_{W,b} E(\boldsymbol{W},\boldsymbol{b}|\boldsymbol{X})$

梯度下降法的参数迭代估计 $\left(i = 1,\ldots,n_{l-1} \qquad j = 1,\ldots,n_l\right)$

$$\begin{cases} \omega_{ij}^l(t+1) = \omega_{ij}^l(t) + \Delta\omega_{ij}^l(t) & \Delta\omega_{ij}^l(t) = -\eta\dfrac{\partial E}{\partial \omega_{ij}^l(t)} \\ b_j^l(t+1) = b_j^l(t) + \Delta b_j^l(t) & \Delta b_j^l(t) = -\eta\dfrac{\partial E}{\partial b_j^l(t)} \end{cases}$$

梯度下降法

$$\begin{cases} \text{随机梯度下降法}(Stochastic\ Gradient\ Descent) \\ --(\text{也称：单样本法、在线梯度下降法}) \\ \text{小批量样本法}(Mini-Batch\ Gradient\ Descent) \\ \text{批量样本法}(Batch\ Gradient\ Descent) \end{cases}$$

$$\begin{cases} \text{固定学习步长的梯度下降} \\ \text{变学习步长的梯度下降} \end{cases}$$

$$\begin{cases} \text{普通梯度下降法}(\text{不带冲量}) \\ \text{带}\textbf{冲量}(\text{惯性项}, Momentum) \end{cases}$$

$$\text{模型参数初始化}\begin{cases} \text{参数取值应具有差异性}(\text{高斯分布、均匀分布}) \\ \text{取值不能太大} \end{cases}$$

给定训练样本集 $\{(x_i, D_i), i=1,\cdots,N\}$ ， BP 算法进行参数学习的两个阶段:

(1)输入信号的正向传递过程

训练样本的输入部分从输入层经隐层逐层计算、正向传递，直至到达输出层，得到预测输出

(2)预测误差的反向传播过程

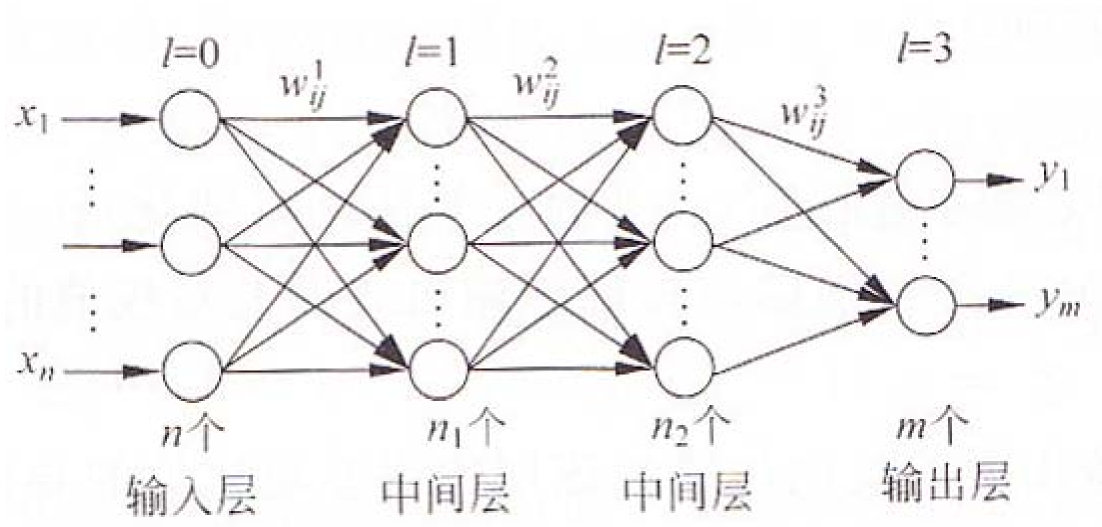
预测输出与目标输出比较，得到预测误差；预测误差逐层、反向传播，可间接计算隐层各单元的误差，并用此误差修正网络有关参数.

BP算法训练过程描述

约定：

(1) $\boldsymbol{n}$维输入向量

$$\boldsymbol{x}=[\boldsymbol{x}_1,\cdots,\boldsymbol{x}_n]^T$$



多层感知器的例子和符号约定

(2) $\boldsymbol{L}$层神经网络

层号$\boldsymbol{l}=0$　　输入层

层号$\boldsymbol{l}=1,\ldots,\boldsymbol{L}-2$　隐含层

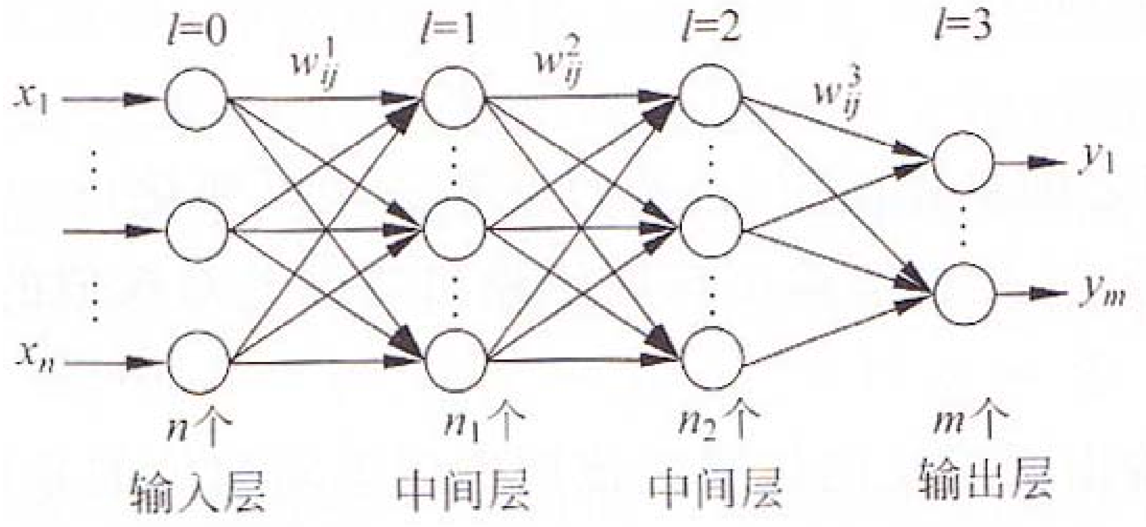
层号$\boldsymbol{l}=\boldsymbol{L}-1$　　输出层

(3) 各层节点数目$\boldsymbol{n}_{\boldsymbol{l}}$，$\boldsymbol{l}=0,1,\ldots,\boldsymbol{L}-1$

输入节点 $\boldsymbol{n}_0=\boldsymbol{n}$

输出节点 $\boldsymbol{n}_{L-1}=\boldsymbol{m}$

(例：$\boldsymbol{m}$个实值函数的回归,$\boldsymbol{m}$个类别的分类)



多层感知器的例子和符号约定

(4)相邻层连接权值

ω_{ij}^{l} −−第**l-1**层的节点***i***与第**l**层节点***j***的连接权值

$$\boldsymbol{W}_{n_{l-1}\times n_l}^{l} = \left[\omega_{ij}^{l}\right]_{n_{l-1}\times n_l} \qquad \begin{cases} \boldsymbol{i} = 1,\ldots,\boldsymbol{n}_{l-1} \\ \boldsymbol{j} = 1,\ldots,\boldsymbol{n}_{l} \end{cases}$$

(5)各计算节点处的偏置量

b_j^l −−第**l**层节点***j***处的偏置量

$$\boldsymbol{b}^{l} = \left[b_j^{l}\right]_{1\times n_l} \qquad \boldsymbol{j} = 1,\ldots,\boldsymbol{n}_l$$

BP算法训练过程描述

(6)**假定**：第l层为**当前处理层**；

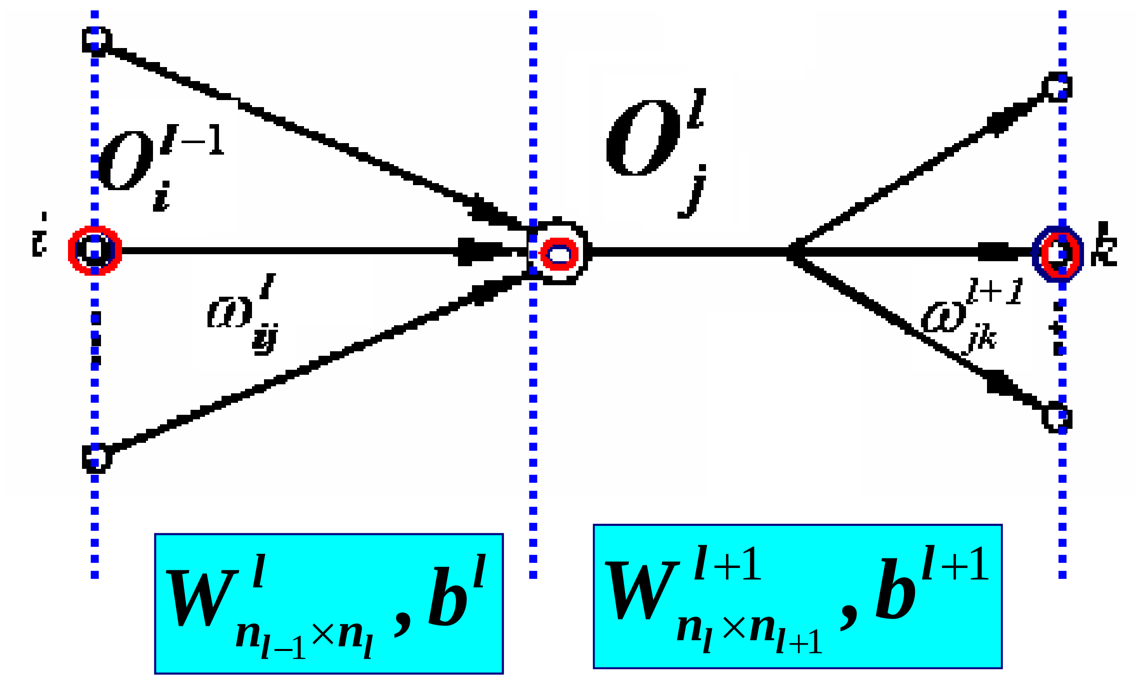
其前一层$l-1$、当前层l、后一层$l+1$的计算单元序号为i, j, k；

位于当前层第j个计算单元的预测输出为O_j^l，$j=1,\dots,n_l$

前层第i个单元到本层第j个单元的连接权值为$\omega_{ij}^l, i=1,\dots,n_{l-1}$

本层第j个单元到后层第k个单元的连接权值为$\omega_{jk}^{l+1}, k=1,\dots,n_{l+1}$

注：采用**梯度法**修正权值，**计算单元输出函数**应连续可微，如：*sigmoid*函数。



第一阶段：输入信号的正向传递过程

固定步长、单样本的梯度下降法

$$x \to \boxed{\begin{matrix} net_j^1 \\ j=1,\dots,n_1 \end{matrix}} \to \boxed{\begin{matrix} O_j^1 = f\left(net_j^1\right) \\ j=1,\dots,n_1 \end{matrix}} \to \cdots \to \boxed{\begin{matrix} net_k^{L-1} \\ k=1,\dots,n_{L-1} \end{matrix}} \to \boxed{\begin{matrix} y_k = O_k^{L-1} = f\left(net_k^{L-1}\right) \\ k=1,\dots,n_{L-1} \end{matrix}}$$

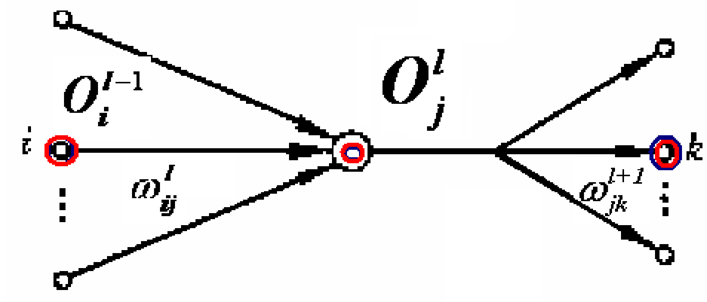
从样本集内取出一个样本(x, D), 将x各分量从输入层输入至网络, 由前向后逐层传递， **当前层**l 的第j个计算节点$(l=1,\dots,L-1 \quad j=1,\dots,n_l)$

$$\begin{cases} \textbf{净输入} \quad net_j^l = \sum_{i=1}^{n_{l-1}} \omega_{ij}^l O_i^{l-1} + b_j^l \\ \textbf{实际输出} \quad O_j^l = f\left(net_j^l\right) = \dfrac{1}{1+e^{-net_j^l}} \end{cases} \Rightarrow \begin{cases} \textbf{矩阵形式：} \\ net^l = \left(W^l\right)^T O^{l-1} + b^l \\ O^l = f\left(net^l\right) \end{cases}$$

若当前层为输出层$(l=L-1)$，则

$$\text{计算单元}j \begin{cases} \textbf{实际输出} \quad y_j = O_j^l \\ \textbf{理想输出} \quad d_j \end{cases} \qquad \text{输出层} \begin{cases} y = \left(y_1, \cdots, y_m\right)^T \\ D = \left(d_1, \dots, d_m\right)^T \end{cases}$$

第二阶段：误差反向传播过程



准则函数 -- 最小误差平方和。

(I)输出误差

样本$\boldsymbol{x}=(x_1,\ldots,x_n)^T$在网络**输出层(各节点)**因**输出$y$**与**目标输出$D$不一致**而导致**输出误差**:

$$E=\frac{1}{2}\|\boldsymbol{D}-\boldsymbol{y}\|^2=\frac{1}{2}(\boldsymbol{D}-\boldsymbol{y})^T(\boldsymbol{D}-\boldsymbol{y})=\frac{1}{2}\left(\boldsymbol{D}^T\boldsymbol{D}-2\boldsymbol{y}^T\boldsymbol{D}+\boldsymbol{y}^T\boldsymbol{y}\right)$$

$$=\frac{1}{2}\left\{\boldsymbol{D}^T\boldsymbol{D}-2\left[\boldsymbol{f}\left(\boldsymbol{net}^{L-1}\right)\right]^T\boldsymbol{D}+\left[\boldsymbol{f}\left(\boldsymbol{net}^{L-1}\right)\right]^T\boldsymbol{f}\left(\boldsymbol{net}^{L-1}\right)\right\}$$

$$=\frac{1}{2}\sum_{j=1}^{m}\left(d_j-y_j\right)^2=\frac{1}{2}\sum_{j=1}^{m}\left(d_j-O_j^{L-1}\right)^2=\frac{1}{2}\sum_{j=1}^{m}\left(d_j-f\left(net_j^{L-1}\right)\right)^2$$

批量样本集关于当前网络的**总输出误差** $E_{total}=\sum E$

(II) 输出层各个权值 ω_{ij}^{L-1} 调整 $\begin{cases} i = 1, \ldots, n_{L-2} \\ j = 1, \ldots, n_{L-1} (m) \end{cases}$

$$\cdots \rightarrow \boxed{\begin{matrix} \left(\omega_{ij}^{L-1}\right)_{n^{L-2} \times n^{L-1}} \\ \left(b_j^{L-1}\right)_{1 \times n^{L-1}} \end{matrix}} \rightarrow \boxed{net_j^{L-1} = \sum_{i=1}^{n_{L-2}} \omega_{ij}^{L-1} O_i^{L-2} + b_j^{L-1}} \rightarrow \boxed{y_j = f\left(net_j^{L-1}\right)} \rightarrow E$$

输出层的**实际输出向量y:**

$$O^{L-1} = y = f\left(net^{L-1}\right) = f\left(\left(W^{L-1}\right)^T O^{L-2} + b^{L-1}\right)$$

其中 $\begin{cases} W^{L-1} = \left(\omega_{ij}^{L-1}\right)_{n^{L-2} \times n^{L-1}} \qquad b^{L-1} = \left[b_1^{L-1} \quad \cdots \quad b_{n_{L-1}}^{L-1}\right]^T \\ net^{L-1} = \left[net_1^{L-1} \quad \cdots \quad net_{n_{L-1}}^{L-1}\right]^T = \left(W^{L-1}\right)^T O^{L-2} + b^{L-1} \\ \boxed{net_j^{L-1} = \sum_{i=1}^{n_{L-2}} \omega_{ij}^{L-1} O_i^{L-2} + b_j^{L-1}} \\ O^{L-1} = \left[O_1^{L-1} \quad \cdots \quad O_{n_{L-1}}^{L-1}\right]^T \qquad m = L-1 \\ y = \left[y_1 \quad \cdots \quad y_m\right]^T \end{cases}$

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

$$f'(x) = \frac{e^{-x}}{\left(1+e^{-x}\right)^2} = f(x)\left[1 - f(x)\right]$$

输出端 $y_j = O_j^{L-1} = f\left(net_j^{L-1}\right)$

(II)输出层各个权值ω_{ij}^{L-1}调整 $\begin{cases} i=1,\ldots,n_{L-2} \\ j=1,\ldots,n_{L-1}(m) \end{cases}$ --续1

输出误差 $$E=\frac{1}{2}\|D-y\|^2=\frac{1}{2}\left(D^TD-2y^TD+y^Ty\right)$$

$$=\frac{1}{2}\left\{D^TD-2\left[f\left(net^{L-1}\right)\right]^TD+\left[f\left(net^{L-1}\right)\right]^Tf\left(net^{L-1}\right)\right\}$$

$$=\frac{1}{2}\sum_{j=1}^{m}\left(d_j-y_j\right)^2=\frac{1}{2}\sum_{j=1}^{m}\left(d_j-O_j^{L-1}\right)^2=\frac{1}{2}\sum_{j=1}^{m}\left(d_j-f\left(net_j^{L-1}\right)\right)^2$$

其中： $y=f\left(net^{L-1}\right)=\dfrac{1}{1+exp\left[-net^{L-1}\right]}$ $\qquad net^{L-1}=\left(W^{L-1}\right)^TO^{L-2}+b^{L-1}$

输出层误差的局部梯度 $$\boxed{\frac{\partial E}{\partial W^{L-1}}=\frac{\partial E}{\partial y}\cdot\frac{\partial y}{\partial net^{L-1}}\cdot\frac{\partial net^{L-1}}{\partial W^{L-1}}=\left(y-D\right)\left[y.*\left(1-y\right)\right]O^{L-2}}$$

其中 $$\begin{cases} \dfrac{\partial E}{\partial y}=y-D & \dfrac{\partial net^{L-1}}{\partial W^{L-1}}=O^{L-2} \\ \dfrac{\partial y}{\partial net^{L-1}}=\dfrac{e^{-net^{L-1}}}{\left(1+e^{-net^{L-1}}\right)^2}=\left[y.*\left(1-y\right)\right] & \end{cases}$$

$$\boxed{\begin{aligned} f(x)&=\frac{1}{1+e^{-x}} \\ f'(x)&=\frac{e^{-x}}{\left(1+e^{-x}\right)^2}=f(x)\left[1-f(x)\right] \end{aligned}}$$

$$\cdots \to \boxed{\begin{array}{l}\left(\omega_{ij}^{L-1}\right)_{n^{L-2}\times n^{L-1}} \\ \left(b_j^{L-1}\right)_{1\times n^{L-1}}\end{array}} \to \boxed{net_j^{L-1} = \sum_{i=1}^{n_{L-2}} \omega_{ij}^{L-1} O_i^{L-2} + b_j^{L-1}} \to \boxed{y_j = f\left(net_j^{L-1}\right)} \to E$$

输出层误差关于ω_{ij}^{L-1}**的偏导数：**

$$\frac{\partial E}{\partial \omega_{ij}^{L-1}} = \frac{\partial E}{\partial net_j^{L-1}} \cdot \frac{\partial net_j^{L-1}}{\partial \omega_{ij}^{L-1}} = \boxed{\delta_j^{L-1}} \cdot \frac{\partial net_j^{L-1}}{\partial \omega_{ij}^{L-1}},$$

$$\boxed{\delta_j^{L-1} = \frac{\partial E}{\partial net_j^{L-1}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j^{L-1}}}$$

$$\begin{cases} E = \dfrac{1}{2}\sum_{j=1}^{m}\left(d_j - y_j\right)^2 \\ y_j = f\left(net_j^{L-1}\right) = \dfrac{1}{1+e^{-net_j^{L-1}}} \\ net_j^{L-1} = \sum_{i=1}^{n_{L-2}} \omega_{ij}^{L-1} O_i^{L-2} + b_j^{L-1} \end{cases} \Rightarrow \begin{cases} \dfrac{\partial E}{\partial y_j} = y_j - d_j \\ \dfrac{\partial y_j}{\partial net_j^{L-1}} = f'\left(net_j^{L-1}\right) = y_j\left(1-y_j\right) \\ \dfrac{\partial net_j^{L-1}}{\partial \omega_{ij}^{L-1}} = O_i^{L-2} \end{cases}$$

$$\Rightarrow \boxed{\frac{\partial E}{\partial \omega_{ij}^{L-1}} = \delta_j^{L-1} \cdot \frac{\partial net_j^{L-1}}{\partial \omega_{ij}^{L-1}} = O_i^{L-2} y_j\left(1-y_j\right)\left(y_j - d_j\right)}$$

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

$$f'(x) = f(x)\left[1 - f(x)\right]$$

(II)输出层各个权值ω_{ij}^{L-1}调整 $\begin{cases} i=1,\ldots,n_{L-2} \\ j=1,\ldots,n_{L-1}(m) \end{cases}$ --续3

基于梯度下降法的连接权值迭代估计

ω_{ij}^{L-1}**修正方式**： $\omega_{ij}^{L-1}(t+1)=\omega_{ij}^{L-1}(t)+\Delta\omega_{ij}^{L-1}(t)$

ω_{ij}^{L-1}**修正量** $\Delta\omega_{ij}^{L-1}(t)=-\eta\dfrac{\partial E}{\partial\omega_{ij}^{L-1}(t)}$

其中 $$\frac{\partial E}{\partial\omega_{ij}^{L-1}}=\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial net_j^{L-1}}\cdot\frac{\partial net_j^{L-1}}{\partial\omega_{ij}^{L-1}}=y_j(1-y_j)(y_j-d_j)O_i^{L-2}$$

矩阵形式：

W^{L-1}**修正方式**： $W^{L-1}(t+1)=W^{L-1}(t)+\Delta W^{L-1}(t)$

W^{L-1}**修正量** $\Delta W^{L-1}(t)=-\eta\dfrac{\partial E}{\partial W^{L-1}(t)}$

其中: $$\frac{\partial E}{\partial W^{L-1}}=\frac{\partial E}{\partial y}\frac{\partial y}{\partial net^{L-1}}\cdot\frac{\partial net^{L-1}}{\partial W^{L-1}}=(y-D)[y.*(1-y)]O^{L-2}$$

(III) 输出层的偏置量b_j^{L-1}调整 $j=1,\ldots,n_{L-1}(m)$

输出端j的净输入 $net_j^{L-1}=\sum_{i=1}^{n_{L-2}}\omega_{ij}^{L-1}O_i^{L-2}+b_j^{L-1}$

输出端j的实际输出

$$O_j^{L-1}=y_j=f\left(net_j^{L-1}\right)=f\left(\sum_{i=1}^{n_{L-2}}\omega_{ij}^{L-1}O_i^{L-2}+b_j^{L-1}\right)$$

输出误差 $E=\frac{1}{2}\sum_{j=1}^{m}\left(d_j-y_j\right)^2=\frac{1}{2}\sum_{j=1}^{m}\left(d_j-f\left(net_j^{L-1}\right)\right)^2$

偏置量b_j^{L-1}对误差影响

$$\frac{\partial E}{\partial b_j^{L-1}}=\frac{\partial E}{\partial y_j}\cdot\frac{\partial y_j}{\partial net_j^{L-1}}\cdot\frac{\partial net_j^{L-1}}{\partial b_j^{L-1}}=\left(y_j-d_j\right)y_j\cdot\left(1-y_j\right)\cdot 1$$

（III）输出层的偏置量b_j^{L-1}调整（续1）

偏置量b_j^{L-1}对误差的影响

$$\frac{\partial E}{\partial b_j^{L-1}} = y_j\left(y_j - d_j\right)\cdot\left(1 - y_j\right)$$

输出层的偏置量b_j^{L-1}调整

b_j^{L-1}修正方式　$b_j^{L-1}\left(t+1\right) = b_j^{L-1}\left(t\right) + \Delta b_j^{L-1}\left(t\right)$

b_j^{L-1}修正量　$\Delta b_j^{L-1} = -\eta\frac{\partial E}{\partial b_j^{L-1}} = \eta y_j\left(d_j - y_j\right)\cdot\left(1 - y_j\right)$

(IV) 隐含层(当前层l)的权值ω_{ij}^{l}调整 $\begin{cases} i = 1,\ldots,n_{l-1} \\ j = 1,\ldots,n_{l} \end{cases}$

当前层l的节点j对后层$(l+1)$全部节点k均有影响

后层$(l+1)$的所有节点均对网络输出误差产生影响

$$\cdots \to \begin{cases} \omega_{ij}^{l} \\ b_{j}^{l} \end{cases} \to net_{j}^{l} \to O_{j}^{l} \to \boxed{\begin{matrix} \text{对于}k = 1,\ldots,n_{l+1} \\ net_{k}^{l+1} \to O_{k}^{l+1} \end{matrix}} \to \cdots \to E$$

$$\frac{\partial E}{\partial \omega_{ij}^{l}} = \boxed{\frac{\partial E}{\partial net_{j}^{l}}} \cdot \frac{\partial net_{j}^{l}}{\partial \omega_{ij}^{l}} = \boxed{\delta_{j}^{l}} \frac{\partial net_{j}^{l}}{\partial \omega_{ij}^{l}}$$

$$\boxed{\delta_{j}^{l}} = \frac{\partial E}{\partial O_{j}^{l}} \cdot \frac{\partial O_{j}^{l}}{\partial net_{j}^{l}} = \left[\sum_{k=1}^{n_{l+1}} \left(\frac{\partial E}{\partial O_{k}^{l+1}} \frac{\partial O_{k}^{l+1}}{\partial net_{k}^{l+1}} \frac{\partial net_{k}^{l+1}}{\partial O_{j}^{l}} \right) \right] \frac{\partial O_{j}^{l}}{\partial net_{j}^{l}}$$

由前面知：$$\frac{\partial E}{\partial \omega_{ij}^{l}}=\boxed{\delta_j^l}\boxed{\frac{\partial net_j^l}{\partial \omega_{ij}^{l}}}$$

所以

$$\boxed{\delta_j^l}=\frac{\partial E}{\partial net_j^l}=\frac{\partial E}{\partial O_j^l}\frac{\partial O_j^l}{\partial net_j^l}=\sum_{k=1}^{n_{l+1}}\left(\frac{\partial E}{\partial O_k^{l+1}}\frac{\partial O_k^{l+1}}{\partial net_k^{l+1}}\frac{\partial net_k^{l+1}}{\partial O_j^l}\right)\frac{\partial O_j^l}{\partial net_j^l}$$

$$=\left[\sum_{k=1}^{n_{l+1}}\delta_k^{l+1}\omega_{jk}^{l+1}\right]f'\left(net_j^l\right)=\left(\sum_{k=1}^{n_{l+1}}\delta_k^{l+1}\omega_{jk}^{l+1}\right)O_j^l\left(1-O_j^l\right)$$

又 $$net_j^l=\sum_{i=1}^{n_{l-1}}\omega_{ij}^l O_i^{l-1}+b_j^l$$

所以 $$\boxed{\frac{\partial net_j^l}{\partial \omega_{ij}^{l}}=O_i^{l-1}}$$

$$\Rightarrow \boxed{\frac{\partial E}{\partial \omega_{ij}^{l}} = \delta_j^l \frac{\partial net_j^l}{\partial \omega_{ij}^l} = O_i^{l-1} O_j^l \left(1 - O_j^l\right) \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \omega_{jk}^{l+1}}$$

隐含层节点与其前层节点连接权值ω_{ij}^l迭代估计

$$\left(l = L-2, L-3, \ldots\right)$$

ω_{ij}^l**修正方式：** $\omega_{ij}^l(t+1) = \omega_{ij}^l(t) + \Delta\omega_{ij}^l(t)$

ω_{ij}^l**修正量** $\Delta\omega_{ij}^l(t) = -\eta \dfrac{\partial E}{\partial \omega_{ij}^l(t)}$

其中 $$\frac{\partial E}{\partial \omega_{ij}^{l}} = \frac{\partial E}{\partial net_j^l} \cdot \frac{\partial net_j^l}{\partial \omega_{ij}^l} = O_i^{l-1} O_j^l \left(1 - O_j^l\right) \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \omega_{jk}^{l+1}$$

(V) 隐含层(当前层l)各节点的偏置量b_j^l调整

$$j=1,\dots,n_l$$

由前面分析知：$\delta_j^l=\dfrac{\partial E}{\partial net_j^l}=O_j^l\left(1-O_j^l\right)\sum_{k=1}^{n_{l+1}}\delta_k^{l+1}\omega_{jk}^{l+1}$

$$net_j^l=\sum_{i=1}^{n_{l-1}}\omega_{ij}^l O_i^{l-1}+b_j^l \quad\Rightarrow\quad \frac{\partial net_j^l}{\partial b_j^l}=1$$

所以：$\dfrac{\partial E}{\partial b_j^l}=\dfrac{\partial E}{\partial net_j^l}\cdot\dfrac{\partial net_j^l}{\partial b_j^l}=\delta_j^l\cdot\dfrac{\partial net_j^l}{\partial b_j^l}=\delta_j^l$

偏置量b_j^l修正

b_j^l修正方式	$b_j^l(t+1)=b_j^l(t)+\Delta b_j^l(t)$
b_j^l修正量	$\Delta b_j^l=-\eta\dfrac{\partial E}{\partial b_j^l}=-\eta\delta_j^l=-\eta O_j^l\left(1-O_j^l\right)\sum_{k=1}^{n_{l+1}}\delta_k^{l+1}\omega_{jk}^{l+1}$

(2)误差反向传播过程(续)

引入惯性项：为加快收敛速度，往往在权值及偏置量修正过程中加上前次的权值修正量，称为**惯性项**。

权值修正

$$\boxed{\begin{aligned}\omega_{ij}(t+1)&=\omega_{ij}(t)+\Delta\omega_{ij}(t)\\ \Delta\omega_{ij}(t)&=-\eta\delta_j O_i+\alpha\Delta\omega_{ij}(t-1)\end{aligned}}$$

偏置量修正

$$\boxed{\begin{aligned}b_j(t+1)&=b_j(t)+\Delta b_j(t)\\ \Delta b_j(t)&=-\eta\delta_j+\alpha\Delta b_j(t-1)\end{aligned}}$$

BP算法步骤

首先明确

- **训练样本**　(输入向量, 期望输出)，如$\left(\boldsymbol{x}_p, \boldsymbol{D}_p\right)$
- **节点参数初始化**　"小随机数"

  例：输入层与第一隐含层间：$\omega_{ij}^{1} \in \left(-\frac{1}{\sqrt{\boldsymbol{d}}}, \frac{1}{\sqrt{\boldsymbol{d}}}\right)$
- **学习步长**η　通常固定$\mathbf{0.1} \sim \mathbf{0.3}$之间；或动态调整
- **惯性(冲量)项系数**α　通常$\mathbf{0.9} \sim 1$之间

$STEP1$

(1)**确定神经网络结构，**包括

输入层节点数；隐含层数目；各隐含层节点数目；

输出层节点数；各神经元节点的传递函数

(2)**样本集的标准化处理以及输出端编码**

(3)**设定终止条件：**最大可允许训练轮数(硬条件)；

训练精度(软条件)

(4)**以小随机数初始化网络权值；**

记训练时间$\boldsymbol{t} = 0$

BP算法步骤(续)

***STEP*2：重复如下过程直至满足算法终止条件**

(1)按随机或任意顺序从训练集中抽取1个训练样本$(\boldsymbol{x},\boldsymbol{D})$

样本输入 $\boldsymbol{x}=\left[x_1,\cdots,x_n\right]^T\in \boldsymbol{R}^n$

期望输出 $\mathbf{D}=\left[d_1,\cdots,d_m\right]^T\in \boldsymbol{R}^m$

(2)计算输入$\boldsymbol{x}$时当前网络的**实际输出** $y=\left[y_1,\quad\cdots,\quad y_m\right]^T$

$$\boldsymbol{y}_r=\boldsymbol{f}\left(\sum_{s=1}^{n_{L-2}}\omega_{sr}^{l=L-1}\cdots\boldsymbol{f}\left(\sum_{j=1}^{n_1}\omega_{jk}^{l=2}f\left(\sum_{i=1}^{n}\omega_{ij}^{l=1}\boldsymbol{x}_i+\boldsymbol{b}_j^{l=1}\right)+\boldsymbol{b}_k^{l=2}\right)\cdots+\boldsymbol{b}_r^{l=L-1}\right)$$

$\boldsymbol{r}=1,\ldots,\boldsymbol{m}$

其中
$$\begin{cases}\text{激活函数 } \boldsymbol{f}(\alpha)=\dfrac{1}{1+\boldsymbol{e}^{-\alpha}}\\[2ex] \boldsymbol{f}'(\alpha)=-\dfrac{-\boldsymbol{e}^{-\alpha}}{\left(1+\boldsymbol{e}^{-\alpha}\right)^2}=\boldsymbol{f}(\alpha)\left(1-\boldsymbol{f}(\alpha)\right)\end{cases}$$

BP算法步骤

(3) 从输出层开始**调整权值**及节点**偏置量**，具体为：

$$\text{对于第}\boldsymbol{l}\text{层，修正}\begin{cases}\text{权值} & \omega_{ij}^{l}(\boldsymbol{t}+1)=\omega_{ij}^{l}(\boldsymbol{t})+\Delta\omega_{ij}^{l}(\boldsymbol{t}) \\ \text{偏置量} & \boldsymbol{b}_{j}^{l}(\boldsymbol{t}+1)=\boldsymbol{b}_{j}^{l}(\boldsymbol{t})+\Delta\boldsymbol{b}_{j}^{l}(\boldsymbol{t})\end{cases}$$

$$\boldsymbol{i}=1,\ldots,\boldsymbol{n}_{l-1} \qquad \boldsymbol{j}=1,\ldots,\boldsymbol{n}_{l}$$

其中 $\Delta\omega_{ij}^{l}(\boldsymbol{t})=-\eta\delta_{j}^{l}\boldsymbol{x}_{i}^{l-1}$ $\qquad \Delta\boldsymbol{b}_{j}^{l}(\boldsymbol{t})=-\eta\delta_{j}^{l}$

$$\begin{cases}\text{输出层}(\boldsymbol{l}=\boldsymbol{L}-1) & \delta_{j}^{l}=\boldsymbol{y}_{j}(1-\boldsymbol{y}_{j})(\boldsymbol{d}_{j}-\boldsymbol{y}_{j}) \\ & \boldsymbol{j}=1,\ldots,\boldsymbol{m} \\ \text{中间层}(\boldsymbol{1}\le\boldsymbol{l}<\boldsymbol{L}-1) & \delta_{j}^{l}=\boldsymbol{x}_{j}^{l}(1-\boldsymbol{x}_{j}^{l})\sum_{k=1}^{n_{l+1}}\delta_{k}^{l+1}\omega_{jk}^{l+1}(\boldsymbol{t}) \\ & \boldsymbol{j}=1,\ldots,\boldsymbol{n}_{l}\end{cases}$$

BP算法步骤

(4) **更新全部权值及偏置量**，对所有训练样本

重新计算输出；估计更新后网络输出误差；

检查算法**终止条件**

$$\begin{cases} \text{若不满足条件，则 } \boldsymbol{t \leftarrow t+1}\text{, 转向(1)} \\ \text{若条件件满足，则终止，转向}\boldsymbol{STEP3} \end{cases}$$

***STEP3*：** 算法结束. **保存各层连接权值及偏置量**。

终止条件可以是如下之一：

$$\begin{cases} \text{(1)网络实际输出与期望输出}\textbf{总误差} < \textbf{阈值1} \\ \text{(2)最近1轮训练中所有权值及偏置量变化}\textbf{最大值} < \textbf{阈值2} \\ \text{(3)算法达到最大允许的}\textbf{总训练次数} = \textbf{阈值3} \end{cases}$$