机器学习
Machine Learning

第8讲：神经网络与深度学习-I
Neural Network and Deep Learning

1.多层感知机（前馈神经网络）

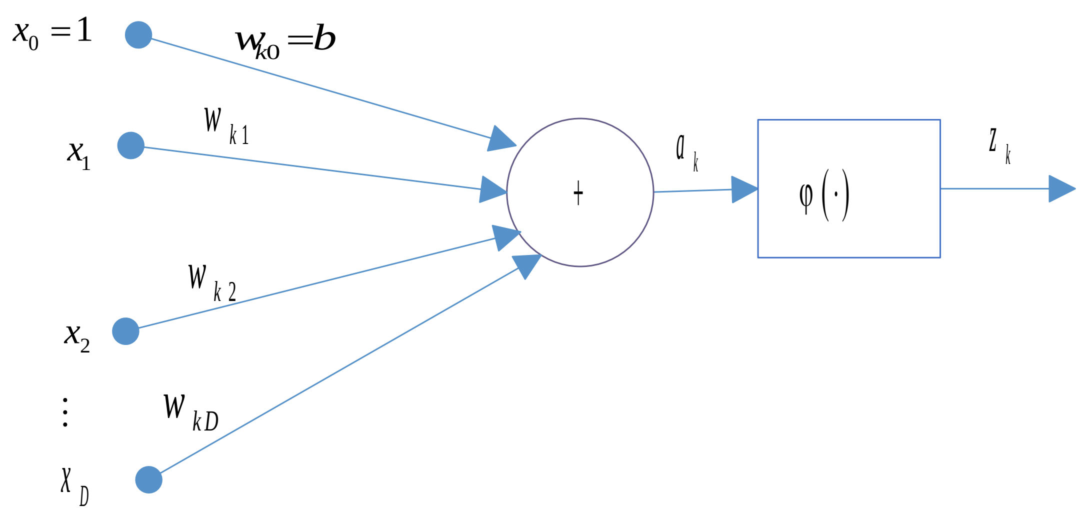
1.1 神经元结构

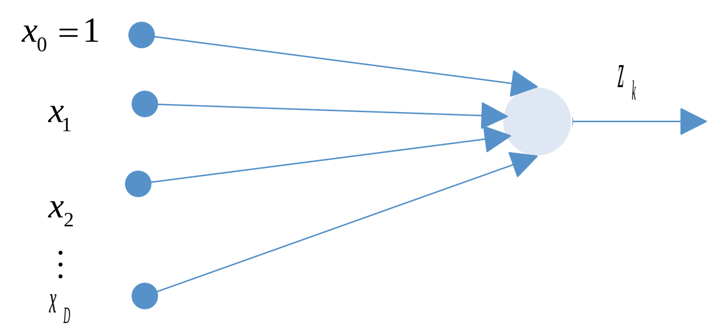
激活值

$$a_k = \sum_{i=1}^{D} w_{ki}x_i + w_{k0} = \boldsymbol{w}_k^T \overline{\boldsymbol{x}}$$

神经元的输出

$$z_k = \varphi(a_k)$$





激活函数例子

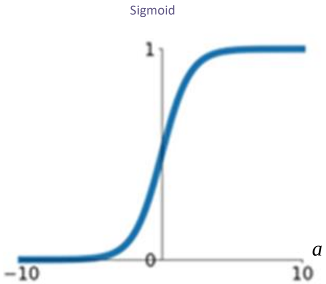
$$\varphi(a) = \mathrm{sgn}(a) = \begin{cases} 1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$

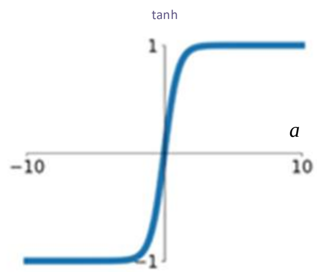
$$\varphi(a) = \sigma(a) = \frac{1}{1 + \mathrm{e}^{-a}}$$

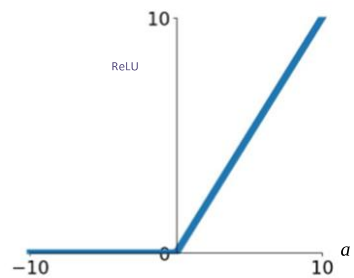
$$\varphi(a) = tanh(a) = \frac{\mathrm{e}^{a} - \mathrm{e}^{-a}}{\mathrm{e}^{a} + \mathrm{e}^{-a}}$$

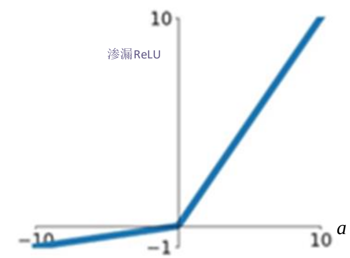
$$\frac{d}{da} tanh(a) = 1 - \tanh^2(a)$$

$$\varphi(a) = \max\{0, a\}$$









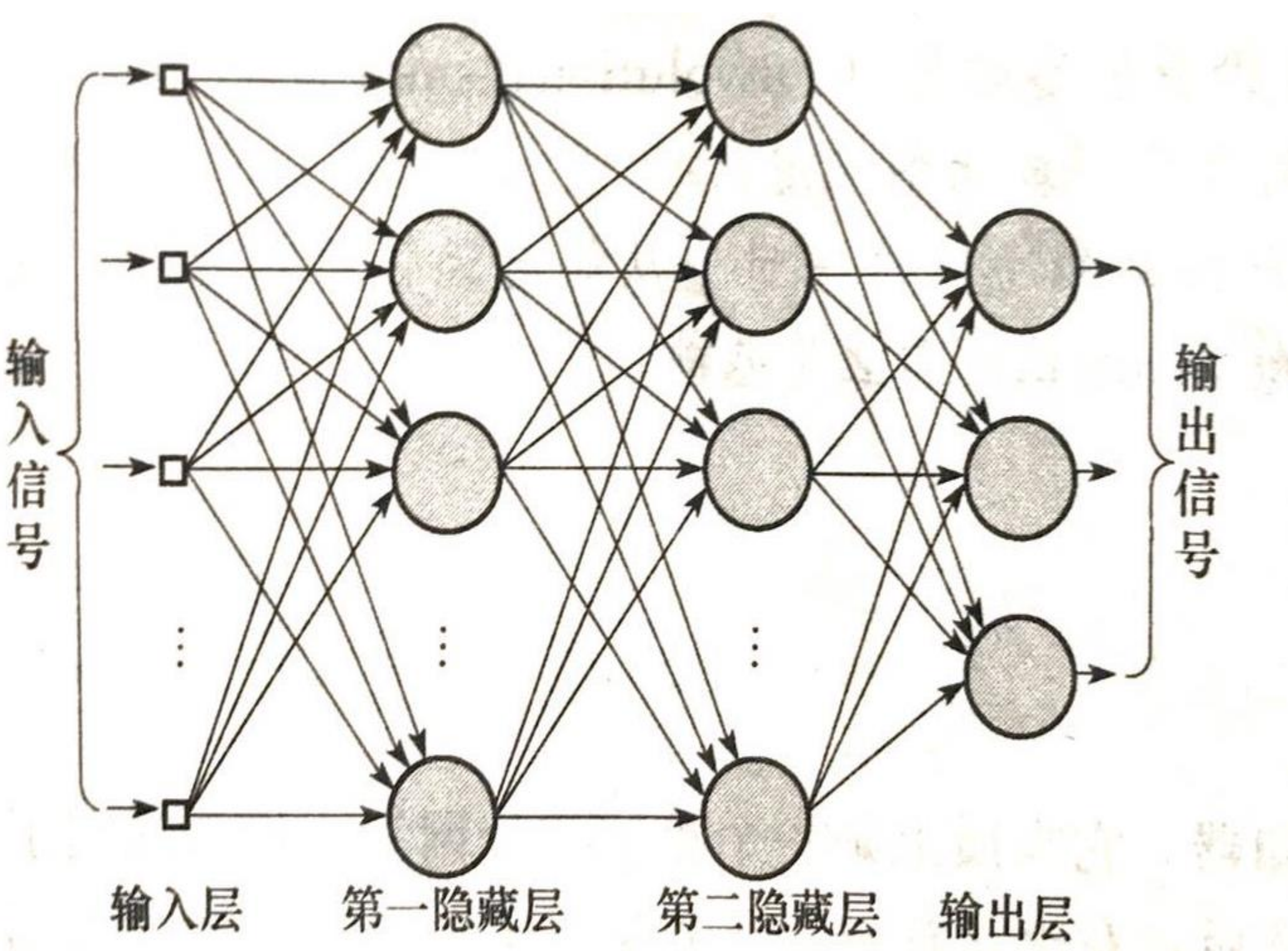
1.2 前馈神经网络

Feed-Forward Neural Network
Multilayer Perceptron：MLP

以一个三层网络为例：二个隐藏层

几个名词：
激活（**activations)** a_j
激活函数（**activation function**） $\varphi(\cdot)$
输入、输出、隐藏单元

一个三层神经网络例子



一个三层神经网络的表示

第一层激活

$$a_k^{(1)} = w_{k0}^{(1)} + \sum_{i=1}^{D} w_{ki}^{(1)} x_i = \sum_{i=0}^{D} w_{ki}^{(1)} x_i$$

$$= \boldsymbol{w}_k^{(1)T} \overline{\boldsymbol{x}} \qquad k = 1,2,\cdots,K_1$$

$w_{ki}^{(1)}$ 权系数　　　$w_{k0}^{(1)}$ 偏置

神经元输出

$$z_k^{(1)} = \varphi_1\left(a_k^{(1)}\right)$$

φ_1

第一层激活函数

一个三层神经网络的表示（续）

第二层单元的激活

$$a_j^{(2)} = w_{j0}^{(2)} + \sum_{k=1}^{K_1} w_{jk}^{(2)} z_k^{(1)} = \sum_{k=0}^{K_1} w_{jk}^{(2)} z_k^{(1)} = \boldsymbol{w}_j^{(2)T} \bar{\boldsymbol{z}}^{(1)}$$

第二层神经元的输出

$$z_j^{(2)} = \varphi_2\left(a_j^{(2)}\right), \qquad j = 1,2,\cdots,K_2$$

一个三层神经网络的表示（续）

第三层：输出层激活

$$a_l^{(3)} = w_{l0}^{(3)} + \sum_{j=1}^{K_2} w_{lj}^{(3)} z_j^{(2)}$$

$$= \boldsymbol{w}_l^{(3)T} \overline{\boldsymbol{z}}^{(2)}$$

输出端的每个输出记为

$$\hat{y}_l = z_l^{(3)} = \varphi_{3l}\left(a_l^{(3)}\right), \quad l = 1,2,\cdots,K_3$$

不同类型输出一般有不同的非线性输出函数

神经网络的输出表示

1. 一般回归输出激活函数是直通函数

$$\hat{y}_k = z_k^{(L)} = a_k^{(L)}$$

2. 一个二分类任务的输出

$$\hat{y}_k = p(C_1|x) = z_k^{(L)} = \sigma\left(a_k^{(L)}\right) = \frac{1}{1+\exp\left(-a_k^{(L)}\right)}$$

3. 一个K类分类器输出用softmax函数

$$\hat{y}_k = p(C_k|x) = z_k^{(L)} = \frac{\exp\left(a_k^{(L)}\right)}{\sum_{j=1}^{K}\exp\left(a_j^{(L)}\right)}, \qquad k = 1,2,\cdots,K$$

$\varphi(\cdot)$激活函数

一般多层感知机的运算关系

$$a_j^{(m)} = w_{j0}^{(m)} + \sum_{k=1}^{K_{m-1}} w_{jk}^{(m)} z_k^{(m-1)} = \boldsymbol{w}_j^{(m)\mathrm{T}} \overline{\boldsymbol{z}}^{(m-1)}$$

$$m = 1, 2, \cdots, L;\ j = 1, 2, \cdots, K_m;\ K_0 = D$$

$$z_j^{(m)} = \varphi_m\left(a_j^{(m)}\right), \quad m = 1, 2, \cdots, L-1;\ j = 1, 2, \cdots, K_m$$

$$\hat{y}_j = z_j^{(L)} = \varphi_{Lj}\left(a_j^{(L)}\right), \quad j = 1, 2, \cdots, K_L$$

输入定义为

$$z_0^{(0)} = 1,\ \ z_i^{(0)} = x_i, \quad i = 1, 2, \cdots, D$$

神经网络的非线性映射

两层回归网络可表示为

$$\hat{y}(\boldsymbol{x},\boldsymbol{W}) = w_0^{(2)} + \sum_{k=1}^{K_1} w_k^{(2)} z_k^{(1)}$$

$$= w_0^{(2)} + \sum_{k=1}^{K_1} w_k^{(2)} \varphi\left(w_{k0}^{(1)} + \sum_{i=1}^{D} w_{ki}^{(1)} x_i\right)$$

表示：$\varphi_0(x) = 1 \qquad \varphi_k(x) = \varphi\left(w_{k0}^{(1)} + \sum_{i=1}^{D} w_{ki}^{(1)} x_i\right)$

有：$\hat{y}(\boldsymbol{x},\boldsymbol{W}) = w_0^{(2)} + \sum_{k=1}^{K_1} w_k^{(2)} \varphi_k(\boldsymbol{x})$

神经网络的非线性映射（续）

两层神经网络二分类输出

$$\hat{y}(\boldsymbol{x}, \boldsymbol{W}) = \sigma\left(w_0^{(2)} + \sum_{k=1}^{K_1} w_k^{(2)} \varphi\left(w_{k0}^{(1)} + \sum_{i=1}^{D} w_{ki}^{(1)} x_i \right)\right)$$

三层神经网络二分类输出：深层嵌套

$$\hat{y}(\boldsymbol{x}, \boldsymbol{W}) =$$

$$\sigma\left(w_0^{(3)} + \sum_{j=1}^{K_2} w_j^{(3)} \varphi_2\left(w_{j0}^{(2)} + \sum_{k=1}^{K_1} w_{jk}^{(2)} \varphi_1\left(w_{k0}^{(1)} + \sum_{i=1}^{D} w_{ki}^{(1)} x_i \right)\right)\right)$$

2. 神经网络通用逼近定理

1. 逼近定理：对于紧支输入空间的任意连续函数，一个线性输出的两层神经网络，只要具有充分大的隐藏单元，则可以以任意精度逼近该函数。

2. 权空间对称性：对于M个隐藏单元的神经网络，其权空间的对称因子为 $M!\times 2^{M}$ （对于tanh类奇函数做激活函数，否则只有 $M!$ 。（假设只有一个隐藏层）

3. 可以构成更多层数的更一般神经网络。可构成深度神经网络（DNN）。

3. 网络训练-目标函数

I.I.D训练数据集

$$\boldsymbol{D}=\left\{\left(\boldsymbol{x}_1,\boldsymbol{y}_1\right),\left(\boldsymbol{x}_2,\boldsymbol{y}_2\right),\cdots,\left(\boldsymbol{x}_N,\boldsymbol{y}_N\right)\right\}=\left\{\left(\boldsymbol{x}_n,\boldsymbol{y}_n\right)\right\}_{n=1}^{N}$$

经验风险

$$J(\boldsymbol{w})=\frac{1}{N}\sum_{n=1}^{N}L\left(\hat{\boldsymbol{y}}(\boldsymbol{x}_n;\boldsymbol{w}),\boldsymbol{y}_n\right)$$

样本损失函数

$$L_n=L\left(\hat{\boldsymbol{y}}_n,\boldsymbol{y}_n\right)$$

3.1 多个回归输出的情况

回归输出

$$\hat{\boldsymbol{y}}_n = \hat{\boldsymbol{y}}(\boldsymbol{x}_n;\mathbf{w}) = \boldsymbol{a}_n^{(L)}$$

逼近误差为 $\boldsymbol{e}_n = \boldsymbol{y}_n - \hat{\boldsymbol{y}}_n$ 满足 $\boldsymbol{e}_n \sim N\left(\boldsymbol{e} \middle| \mathbf{0}, \sigma_e^2 \mathbf{I}\right)$

$\boldsymbol{y}_n$ 满足： $\boldsymbol{y}_n \sim N\left(\boldsymbol{y}_n \middle| \hat{\boldsymbol{y}}_n, \sigma_e^2 \mathbf{I}\right)$

样本似然函数

$$p(\boldsymbol{y}_n|\boldsymbol{w}) = \frac{1}{(2\pi\sigma_e^2)^{K/2}} \exp\left(-\frac{1}{2\sigma_e^2}(\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n)^{\mathrm{T}}(\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n)\right)$$

$$= \frac{1}{(2\pi\sigma_e^2)^{K/2}} \exp\left(-\frac{1}{2\sigma_e^2}\left\|\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n\right\|_2^2\right)$$

3.1 多个回归输出的情况（续）

样本集I.I.D.，样本集似然函数

$$p(\boldsymbol{Y}|\mathbf{w})=\prod_{n=1}^{N}p(\boldsymbol{y}_n|\mathbf{w})=\prod_{n=1}^{N}\frac{1}{(2\pi\sigma_e^2)^{K/2}}\exp\left(-\frac{1}{2\sigma_e^2}\left\|\boldsymbol{y}_n-\hat{\boldsymbol{y}}_n\right\|_2^2\right)$$

令负对数似然函数为损失函数，得

$$J(\mathbf{w})=\frac{1}{2}\sum_{n=1}^{N}\left\|\boldsymbol{y}_n-\hat{\boldsymbol{y}}_n\right\|_2^2$$

3.1 多个回归输出的情况（续）

一个样本的损失函数为

$$L_n = L(\hat{\boldsymbol{y}}_n, \boldsymbol{y}_n) = \frac{1}{2}\|\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n\|_2^2 = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - \hat{y}_{nk})^2$$

损失函数为各样本损失函数之和

$$J(\mathbf{w}) = \sum_{n=1}^{N} L_n = \sum_{n=1}^{N} L(\hat{\boldsymbol{y}}_n, \boldsymbol{y}_n)$$

3.2 K个独立的2元分类

K个独立的二分类器，第k个分类器的输出为

$$\hat{y}_{nk}=\hat{y}_k(\boldsymbol{x}_n;\boldsymbol{w})=\sigma\left(a_{nk}^{(L)}\right)=p\left(C_{k1}\middle|\boldsymbol{x}_n\right)$$

单样本输出标注的似然函数

$$p(\boldsymbol{y}_n|\boldsymbol{w})=\prod_{k=1}^{K}\left(\hat{y}_{nk}\right)^{y_{nk}}\left(1-\hat{y}_{nk}\right)^{1-y_{nk}}$$

样本集似然函数为

$$p(\boldsymbol{Y}|\boldsymbol{w})=\prod_{n=1}^{N}p(\boldsymbol{y}_n|\boldsymbol{w})=\prod_{n=1}^{N}\prod_{k=1}^{K}\left(\hat{y}_{nk}\right)^{y_{nk}}\left(1-\hat{y}_{nk}\right)^{1-y_{nk}}$$

3.2 K个独立的2元分类（续）

样本集负对数似然函数为

$$J(\mathbf{w})=-\sum_{n=1}^{N}\sum_{k=1}^{K}\left[y_{nk}\ln\hat{y}_{nk}+(1-y_{nk})\ln(1-\hat{y}_{nk})\right]$$

样本损失函数：样本交叉熵

$$L_n=L\left(\hat{\boldsymbol{y}}_n,\boldsymbol{y}_n\right)=-\sum_{k=1}^{K}\left[y_{nk}\ln\hat{y}_{nk}+(1-y_{nk})\ln(1-\hat{y}_{nk})\right]$$

损失函数为各样本损失函数之和

$$J(\mathbf{w})=\sum_{n=1}^{N}L_n=\sum_{n=1}^{N}L\left(\hat{\boldsymbol{y}}_n,\boldsymbol{y}_n\right)$$

3.3. K类分类器

y是K维是K-to-1编码输出，只有其中一个元素取1

用softmax表示的输出为

$$\hat{y}_{nk} = p\left(C_k \middle| \boldsymbol{x}_n\right) = \frac{\exp\left(a_{nk}^{(L)}\right)}{\sum_{j=1}^{K} \exp\left(a_{nj}^{(L)}\right)}, \quad k = 1, 2, \cdots, K$$

样本集的联合概率函数

$$p(\boldsymbol{Y}|\boldsymbol{w}) = \prod_{n=1}^{N} p(\boldsymbol{y}_n|\boldsymbol{w}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\left(\hat{y}_{nk}\right)^{y_{nk}}$$

3.3. K类分类器（续）

负对数似然函数为

$$J(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\left[y_{nk}\ln\hat{y}_{nk}\right]$$

样本损失函数（样本交叉熵）

$$L_n = L(\hat{\boldsymbol{y}}_n, \boldsymbol{y}_n) = -\sum_{k=1}^{K}\left[y_{nk}\ln\hat{y}_{nk}\right]$$

损失函数为各样本损失函数之和

$$J(\mathbf{w}) = \sum_{n=1}^{N} L_n = \sum_{n=1}^{N} L(\hat{\boldsymbol{y}}_n, \boldsymbol{y}_n)$$

3.4 样本损失函数对输出激活的导数

三种基本输出类型，回归输出、二分类输出和多分类输出，均满足：

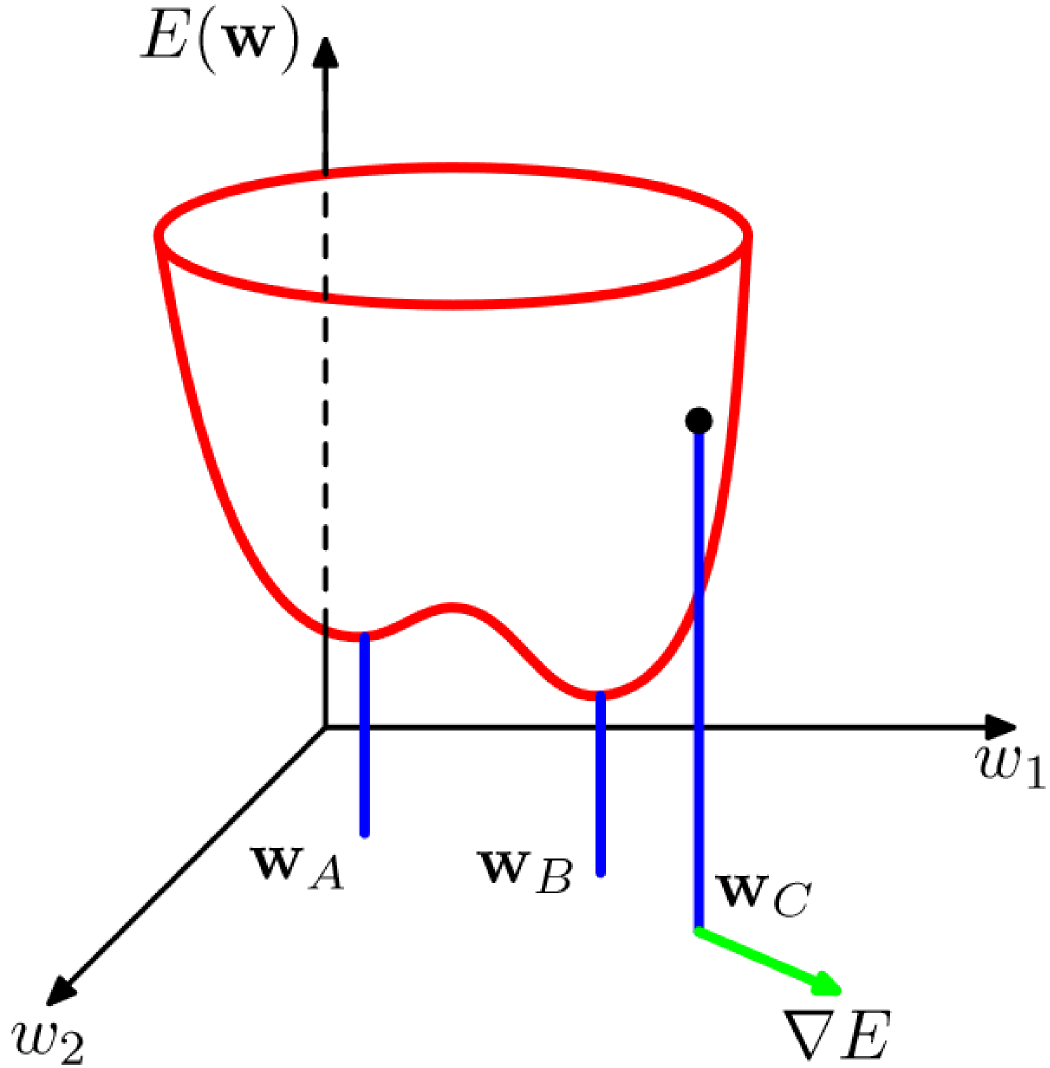
$$\frac{\partial L_n}{\partial \boldsymbol{a}_n^{(L)}} = \frac{\partial L\left(\hat{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)}{\partial \boldsymbol{a}_n^{(L)}} = \hat{\boldsymbol{y}}_n - \boldsymbol{y}_n$$

或

$$\frac{\partial L_n}{\partial a_{nk}^{(L)}} = \frac{\partial L\left(\hat{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)}{\partial a_{nk}^{(L)}} = \hat{y}_{nk} - y_{nk}$$

4. 神经网络的参数优化

神经网络因隐变量的非线性激活函数，其目标函数是多局部极小点，如右图所示



最小化目标函数

$$J(\boldsymbol{w})$$

神经网络的参数优化（续）

最优解满足

$$\nabla_{\mathbf{w}} J(\boldsymbol{w}) = \frac{\partial}{\partial \boldsymbol{w}} J(\boldsymbol{w}) = \mathbf{0}$$

驻点(Stationary point)，与凸目标函数不同，NN的目标函数是高度非线性的，驻点分为：极小点、极大点和鞍点

$$\boldsymbol{H} = \nabla_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

<u>Hessian矩阵</u>。对于极小值点，Hessian是正定矩阵

神经网络的参数优化（续）

梯度优化（批算法）

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta_\tau \nabla J\left(\mathbf{w}^{(\tau)}\right)$$

$\eta > 0$ 学习率

目标函数可分界为按样本

$$J(\mathbf{w}) = \sum_{n=1}^{N} L_n = \sum_{n=1}^{N} L\left(\hat{\mathbf{y}}_n, \mathbf{y}_n\right)$$

则随机梯度下降（SGD）算法为

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta_\tau \nabla L_n \Big|_{\mathbf{w}=\mathbf{w}^{(\tau)}}$$

神经网络的参数优化（续）

小批量梯度算法 $\left\{\left(\boldsymbol{x}_m, \boldsymbol{y}_m\right)\right\}_{m=1}^{N_0}$

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta_\tau \frac{1}{N_0} \sum_{m=1}^{N_0} \nabla L_m \Big|_{\boldsymbol{w}=\boldsymbol{w}^{(\tau)}}$$

其他优化算法包括：
共轭梯度算法，牛顿法，拟牛顿法等。
关键是**<u>有效计算梯度（BP算法）</u>**
和Hessian矩阵（牛顿类算法）

5. 误差反向传播算法（BP）Error Backpropagation

信号在网络中按层交替前向和后向传递

有效计算目标函数对权系数的梯度。

由 $$J(\mathbf{w})=\sum_{n=1}^{N}L_n=\sum_{n=1}^{N}L\left(\hat{\boldsymbol{y}}_n,\boldsymbol{y}_n\right)$$

计算每一个输入样本n，计算导数

$$\frac{\partial L_n}{\partial w_{kj}^{(l)}}$$

若批处理时或小批量批处理需要计算相应求和

$$\frac{\partial J}{\partial w_{kj}^{(l)}}=\sum_n\frac{\partial L_n}{\partial w_{kj}^{(l)}}$$

5.误差反向传播算法（BP）（续）

(1). 前向传播（Forward propagation）

从输入起，依次计算各层，直到输出层

$$a_j^{(l)} = \sum_{i=0}^{K_{l-1}} w_{ji}^{(l)} z_i^{(l-1)}$$

（#1）

$$z_j^{(l)} = \varphi_l\left(a_j^{(l)}\right), \quad j = 1, 2, \cdots, K_l$$

（#2）

$$l = 1, 2, \cdots, L$$

样本输入

$$z_0^{(0)} = 1, z_i^{(0)} = x_i, i = 1, 2, \cdots, D$$

5.误差反向传播算法（BP）（续）

(2). 输出层梯度

输出层传播误差 $\delta_k^{(L)}$

$$\delta_k^{(L)} = \frac{\partial L}{\partial a_k^{(L)}} = \hat{y}_k - y_k\,, \quad k = 1, 2, \cdots, K_L$$

（#3）

对输出层的任一权系数的梯度

$$\frac{\partial L}{\partial w_{kj}^{(L)}} = \frac{\partial L}{\partial a_k^{(L)}} \frac{\partial a_k^{(L)}}{\partial w_{kj}^{(L)}} = \delta_k^{(L)} z_j^{(L-1)}$$

用了（#1）：

$$\frac{\partial a_k^{(L)}}{\partial w_{kj}^{(L)}} = z_j^{(L-1)}$$

5.误差反向传播算法（BP）（续）

(3).计算隐藏层梯度项

由导数链式法则：
$$\frac{\partial L}{\partial w_{ji}^{(l)}} = \frac{\partial L}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}}$$

定义
$$\delta_j^{(l)} = \frac{\partial L}{\partial a_j^{(l)}}$$
传播误差

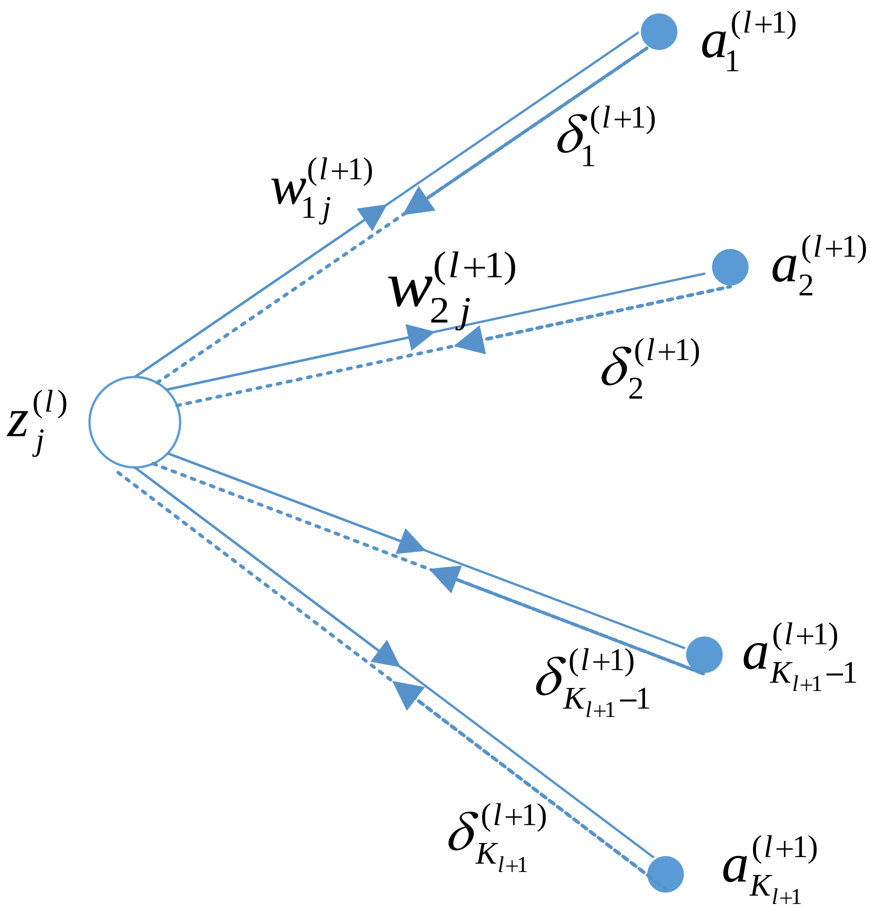
由（#1）
$$\frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}} = z_i^{(l-1)}$$

梯度项表示为
$$\frac{\partial L}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}$$
（#4）

5.误差反向传播算法（BP）（续）

(4).隐藏层反向计算传播误差

再用链式法则



$$\delta_j^{(l)} = \frac{\partial L}{\partial a_j^{(l)}} = \frac{\partial L}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial a_j^{(l)}}$$

使用反向传播机制（右图）计算：

$$\frac{\partial L}{\partial z_j^{(l)}}$$

由以下误差表示

$$\delta_k^{(l+1)} = \frac{\partial L}{\partial a_k^{(l+1)}}$$

5.误差反向传播算法（BP）（续）

(4).隐藏层反向计算传播误差（续）

$$\frac{\partial z_j^{(l)}}{\partial a_j^{(l)}} = \varphi_l'\left(a_j^{(l)}\right)$$

由前向关系：

$$a_k^{(l+1)} = \sum_{m=1}^{K_l} w_{km}^{(l+1)} z_m^{(l)}, \quad k = 1, 2, \cdots, K_{l+1}$$

再用链式法则

$$\frac{\partial L}{\partial z_j^{(l)}} = \sum_{k=1}^{K_{l+1}} \frac{\partial L}{\partial a_k^{(l+1)}} \frac{\partial a_k^{(l+1)}}{\partial z_j^{(l)}} = \sum_{k=1}^{K_{l+1}} \delta_k^{(l+1)} w_{kj}^{(l+1)}$$

得反向传播公式

$$\delta_j^{(l)} = \frac{\partial L}{\partial a_j^{(l)}} = \varphi_l'\left(a_j^{(l)}\right) \sum_{k=1}^{K_{l+1}} w_{kj}^{(l+1)} \delta_k^{(l+1)} \qquad \text{（\#5）}$$

5.误差反向传播算法（BP）（续）

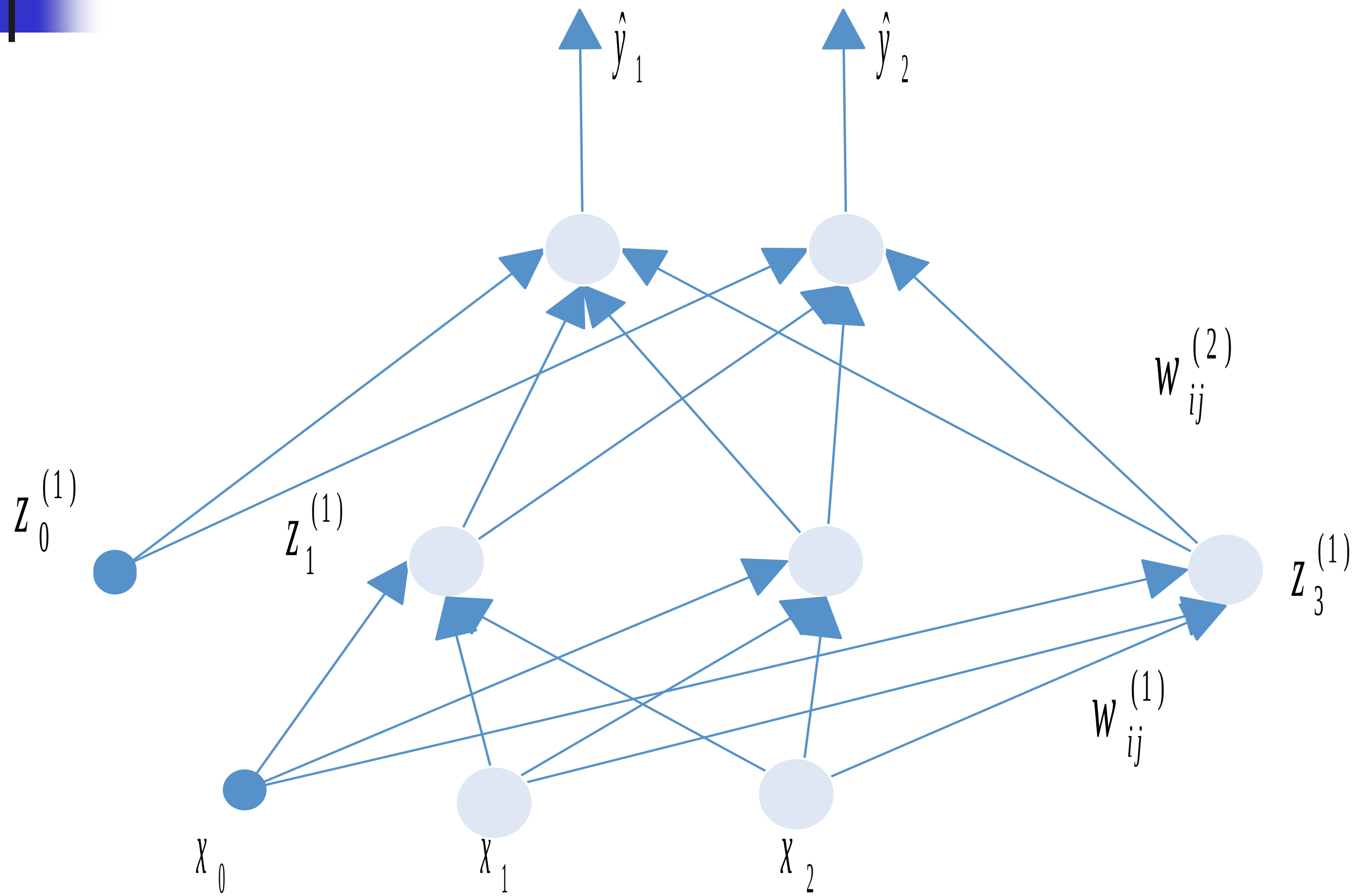
(5). BP算法总结

1. 从输入层起，按层应用（#1）（#2）式，计算各隐藏层和输出层激活值
2. 用（#3）计算输出层传播误差 δ_k
3. 用（#5）反向按层计算各隐藏层传播误差 δ_j
4. 用（#4）式计算梯度中的各项。

注意，在以上表示中，为了简单，激活值输出值等均省略了时间序号n。权系数表示中也省略了迭代序号的标记。

5.6 误差反向传播算法（BP）示例

两层网络例子，一个隐藏层
输出一个回归、一个二分类



例续-前向传播

隐藏层激活和输出计算为

$$a_1^{(1)} = w_{10}^{(1)} + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2$$

$$a_2^{(1)} = w_{20}^{(1)} + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2$$

$$a_3^{(1)} = w_{30}^{(1)} + w_{21}^{(1)} x_1 + w_{32}^{(1)} x_2$$

$$z_1^{(1)} = \tanh\left(a_1^{(1)}\right)$$

$$z_2^{(1)} = \tanh\left(a_2^{(1)}\right)$$

$$z_3^{(1)} = \tanh\left(a_3^{(1)}\right)$$

输出层激活和输出值计算

$$a_1^{(2)} = w_{10}^{(2)} + w_{11}^{(2)} z_1^{(1)} + w_{12}^{(2)} z_2^{(1)} + w_{13}^{(2)} z_3^{(1)}$$

$$a_2^{(2)} = w_{20}^{(2)} + w_{21}^{(2)} z_1^{(1)} + w_{22}^{(2)} z_2^{(1)} + w_{23}^{(2)} z_3^{(1)}$$

输出

$$\hat{y}_1 = z_1^{(2)} = a_1^{(2)}$$

$$\hat{y}_2 = z_2^{(2)} = \sigma\left(a_1^{(2)}\right)$$

例（续）-反向传播

输出层的传播误差

$$\delta_1^{(2)} = \hat{y}_1 - y_1$$

$$\delta_2^{(2)} = \hat{y}_2 - y_2$$

隐藏层反向传播误差

$$\delta_1^{(1)} = \left(1 - \tanh^2\left(a_1^{(1)}\right)\right)\left(w_{11}^{(2)}\delta_1^{(2)} + w_{21}^{(2)}\delta_2^{(2)}\right) = \left(1 - \left(z_1^{(1)}\right)^2\right)\left(w_{11}^{(2)}\delta_1^{(2)} + w_{21}^{(2)}\delta_2^{(2)}\right)$$

$$\delta_2^{(1)} = \left(1 - \tanh^2\left(a_2^{(1)}\right)\right)\left(w_{12}^{(2)}\delta_1^{(2)} + w_{22}^{(2)}\delta_2^{(2)}\right) = \left(1 - \left(z_2^{(1)}\right)^2\right)\left(w_{12}^{(2)}\delta_1^{(2)} + w_{22}^{(2)}\delta_2^{(2)}\right)$$

$$\delta_3^{(1)} = \left(1 - \tanh^2\left(a_2^{(1)}\right)\right)\left(w_{13}^{(2)}\delta_1^{(2)} + w_{23}^{(2)}\delta_2^{(2)}\right) = \left(1 - \left(z_3^{(1)}\right)^2\right)\left(w_{13}^{(2)}\delta_1^{(2)} + w_{23}^{(2)}\delta_2^{(2)}\right)$$

例（续）-梯度计算

计算样本损失函数对各权系数的梯度分量

$$\frac{\partial L}{\partial w_{1j}^{(2)}} = \delta_1^{(2)} z_j^{(1)}$$

$$\frac{\partial L}{\partial w_{2j}^{(2)}} = \delta_2^{(2)} z_j^{(1)}, \quad j = 0,1,2,3$$

$$\frac{\partial L}{\partial w_{1i}^{(1)}} = \delta_1^{(1)} x_i$$

$$\frac{\partial L}{\partial w_{2i}^{(1)}} = \delta_2^{(1)} x_i$$

$$\frac{\partial L}{\partial w_{3i}^{(1)}} = \delta_3^{(1)} x_i, \quad i = 0,1,2$$

6. 多隐藏层前馈网络的一般向量表示

前向传播

对于 $l = 1, 2, \cdots, L-1, L$ $\qquad \boldsymbol{z}^{(0)} = \boldsymbol{x}$

$$\boldsymbol{a}^{(l)} = \mathbf{W}^{(l)} \boldsymbol{z}^{(l-1)} + \boldsymbol{w}_0^{(l)}$$

$$\boldsymbol{z}^{(l)} = \boldsymbol{\varphi}_l \left(\boldsymbol{a}^{(l)} \right)$$

输出层

$$\boldsymbol{y} = \boldsymbol{z}^{(\mathrm{L})} = \boldsymbol{\varphi}_L \left(\boldsymbol{a}^{(\mathrm{L})} \right)$$

多隐藏层前馈网络的一般表示（续）

权系数矩阵

$$\mathbf{W}^{(l)} = \begin{bmatrix} \boldsymbol{w}_1^{(l)\mathrm{T}} \\ \boldsymbol{w}_2^{(l)\mathrm{T}} \\ \vdots \\ \boldsymbol{w}_{\mathrm{K}_l}^{(l)\mathrm{T}} \end{bmatrix} = \begin{bmatrix} w_{11}^{(l)} & w_{12}^{(l)} & \cdots & \cdots & w_{1\mathrm{K}_{l-1}}^{(l)} \\ w_{21}^{(l)} & w_{22}^{(l)} & \cdots & \cdots & w_{2\mathrm{K}_{l-1}}^{(l)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{\mathrm{K}_l 1}^{(l)} & w_{\mathrm{K}_l 2}^{(l)} & \cdots & \cdots & w_{\mathrm{K}_l \mathrm{K}_{l-1}}^{(l)} \end{bmatrix}$$

每一层的偏置系数：

$$\boldsymbol{w}_0^{(l)} = \left[w_{10}^{(l)}, w_{20}^{(l)}, \cdots, w_{K_l 0}^{(l)} \right]^T$$

多隐藏层前馈网络的一般表示（续）

输出层传播误差向量

$$\delta^{(L)} = \hat{y} - y$$

反向传播过程按次序 $l = L-1, L-2, \cdots, 2, 1$

$$\delta^{(l)} = \mathbf{H}'^{(l)} \ \mathbf{W}^{(l+1)\mathrm{T}} \delta^{(l+1)}$$

其中

$$\mathbf{H}'^{(l)} = \mathrm{diag}\left\{\varphi_l'\left(a_1^{(l)}\right), \varphi_l'\left(a_2^{(l)}\right), \cdots, \varphi_l'\left(a_{K_l}^{(l)}\right)\right\}$$

矩阵和向量表示的BP算法

对第 l 层权向量矩阵的梯度为

$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)} \boldsymbol{z}^{(l-1)\mathrm{T}}$$

该层偏置向量的梯度

$$\frac{\partial L}{\partial \boldsymbol{w}_0^{(l)}} = \boldsymbol{\delta}^{(l)}$$

只计算对一个神经元的权系数向量的梯度（包括偏置系数）

$$\frac{\partial L}{\partial \overline{\boldsymbol{w}}_j^{(l)}} = \delta_j^{(l)} \overline{\boldsymbol{z}}^{(l-1)}$$

数值微分求梯度

BP算法的运算复杂度$O(W)$，W是权总数

数值微分也可求梯度，可用于对BP算法进行验证

数值微分运算复杂度 $O(W^2)$

数值微分求梯度项为

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji} - \epsilon)}{2\epsilon} + O(\epsilon^2)$$

数值中心微分精度更高，误差项 $O(\epsilon^2)$

BP算法推广：求Jacobian矩阵

定义Jacobian矩阵项

$$\boldsymbol{J} = \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{x}}$$

Jacobian矩阵有几个应用，其中可评价输出变化

$$\Delta \hat{y}_k \approx \sum_{i=1}^{D} \frac{\partial \hat{y}_k}{\partial x_i} \Delta x_i$$

BP算法推广：求Hessian矩阵

二阶导数 $$\boldsymbol{H} = \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}^2} = \sum_{n=1}^{N} \frac{\partial^2 L_n}{\partial \mathbf{w}^2}$$

把所有权系数（包括偏置）列成向量，分量为 w_i

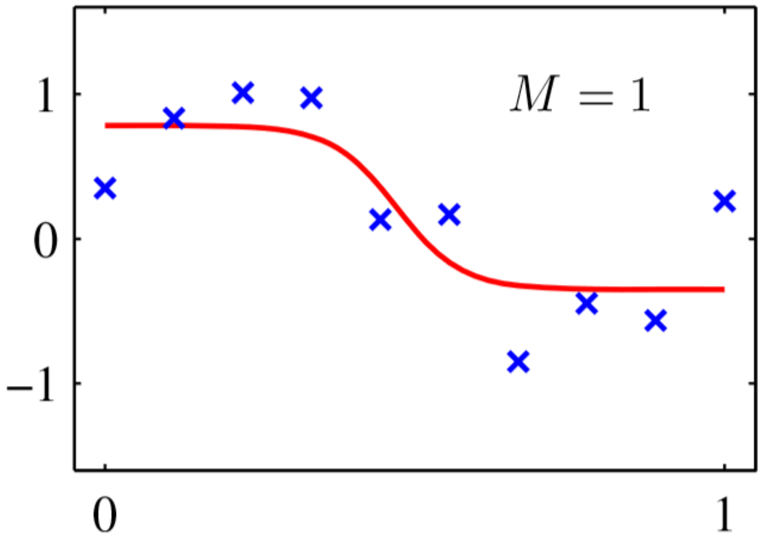
则Hessian矩阵元素为 H_{ij} $\quad i, j \in \{1, \ldots, W\}$

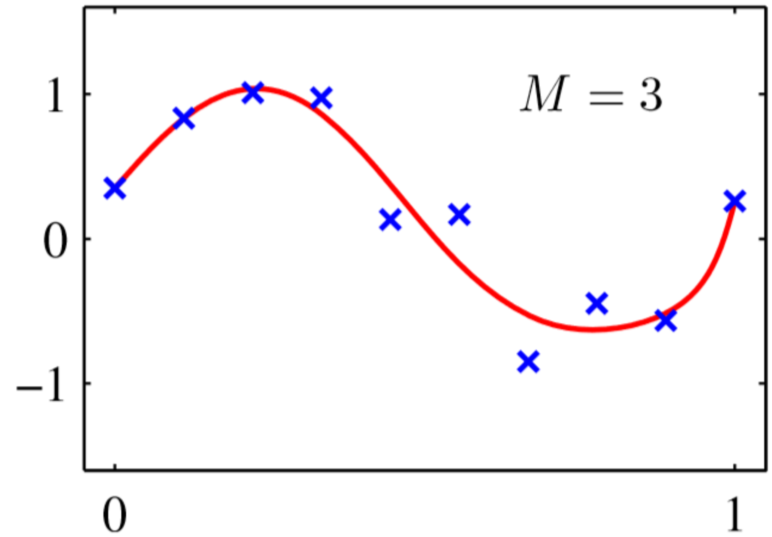
$$H_{ij} = \frac{\partial^2 J(\mathbf{w})}{\partial w_i \partial w_{\mathrm{j}}} = \sum_{n=1}^{N} \frac{\partial^2 L_n}{\partial w_i \partial w_{\mathrm{j}}}$$

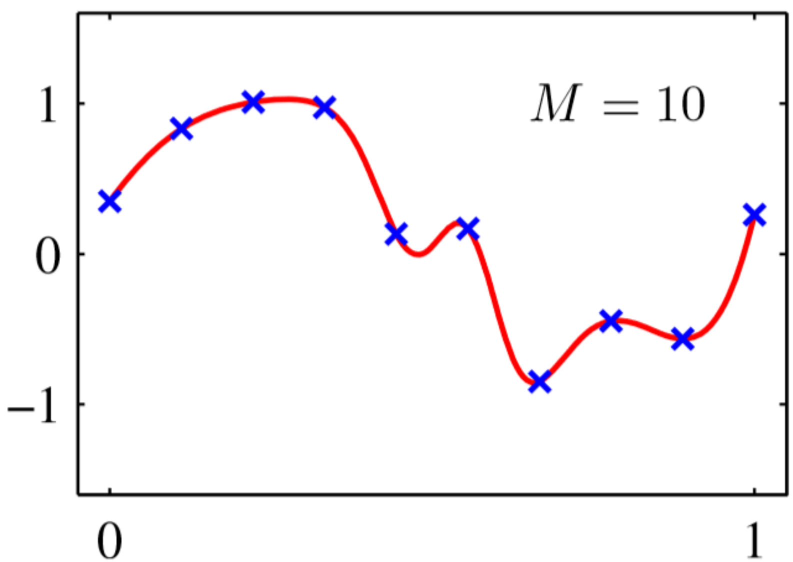
Hessian矩阵为 $\mathbf{H}$

$W \times W$ 维矩阵

神经网络正则化







这里M是二层神经网络的隐节点数。三个图分别欠拟合，拟合合适、过拟合

权衰减正则化
Weight Decay Regularization

$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

可改进BP算法，得到梯度计算

更一般的分组权衰减正则化

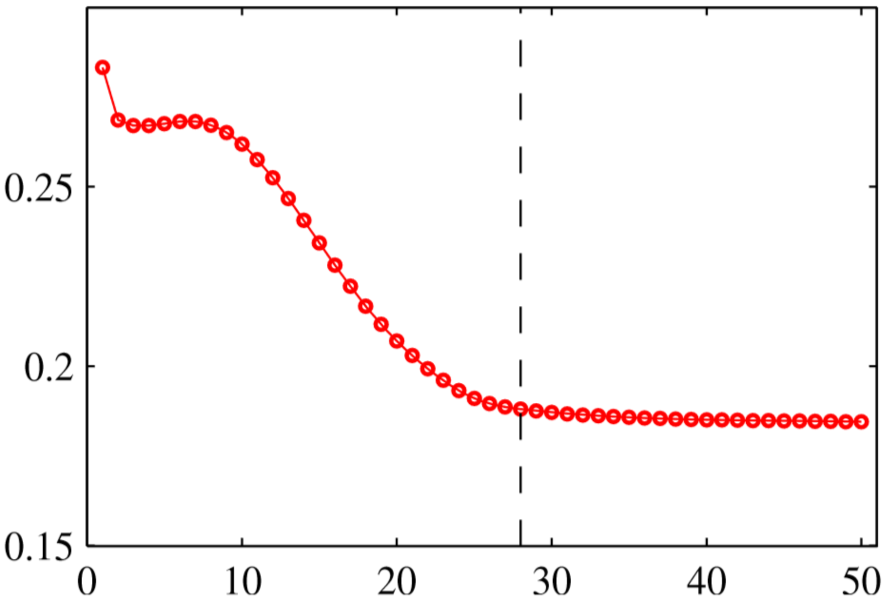
$$p(\mathbf{w}) \propto \exp\left(-\frac{1}{2}\sum_k \alpha_k \|\mathbf{w}\|_k^2\right)$$

这里 $$\|\mathbf{w}\|_k^2 = \sum_{j \in \mathcal{W}_k} w_j^2$$

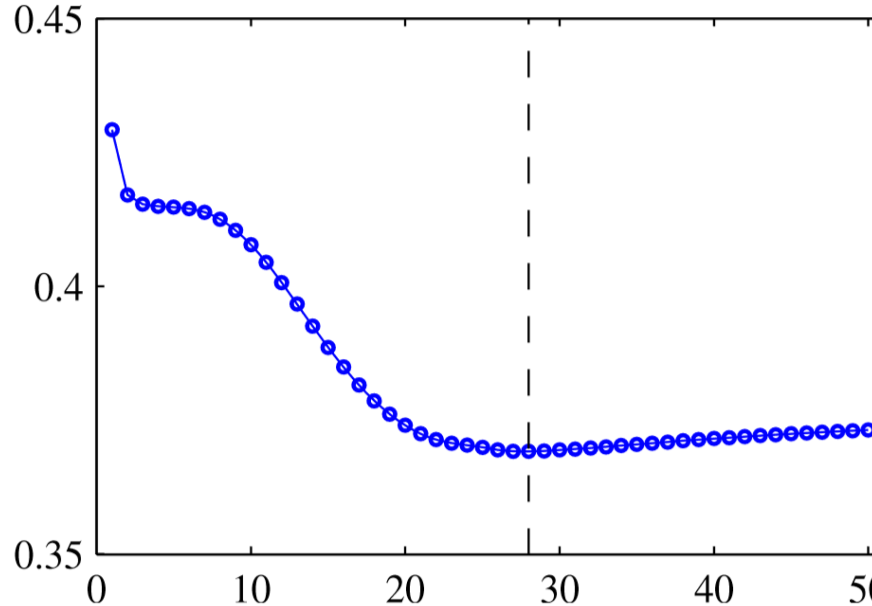
早停止（正则化）Early Stopping Regularization

- 随着迭代进行，训练误差持续减少，直到收敛（可通过随机初始化寻找全局最小）；
- 随着迭代，对于验证集（Validation Set），验证误差首先持续减少，然后开始增加，说明进入过拟合
- 此时停止迭代（尚未达到训练误差最小），早停止相当于限制网络复杂性
- 可以证明，在简化的情况下，早停止与权衰减正则化等价。

早停止（正则化）（续）
Early Stopping Regularization



红色表示随迭代的训练误差，
蓝色表示随迭代的验证误差



可在竖的虚线表示的时刻停止迭代。

$\tau\eta$ 等价于正则系数 λ. 的倒数

τ 表示迭代次数

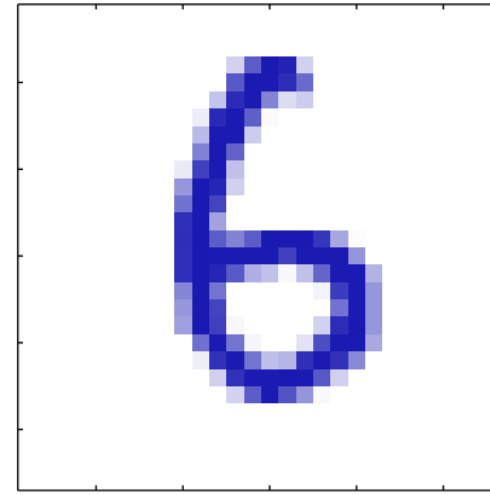
不变性（Invariances）

- **不变性指**：对输入的一些变换得到相同的输出（分类更常有这个要求），例如图像的平移不变性、尺度不变性等，语音的μ律。
- 充分大样本集可逼近于实现不变性
- 有限样本集下逼近不变性有如下几个方法
- （1）增广样本集。对一个样本，对其进行希望的变换后复用。例如，一个代表数字符号的图像可做多次平移，形成多个同标注样本。

不变性（Invariances）（续）

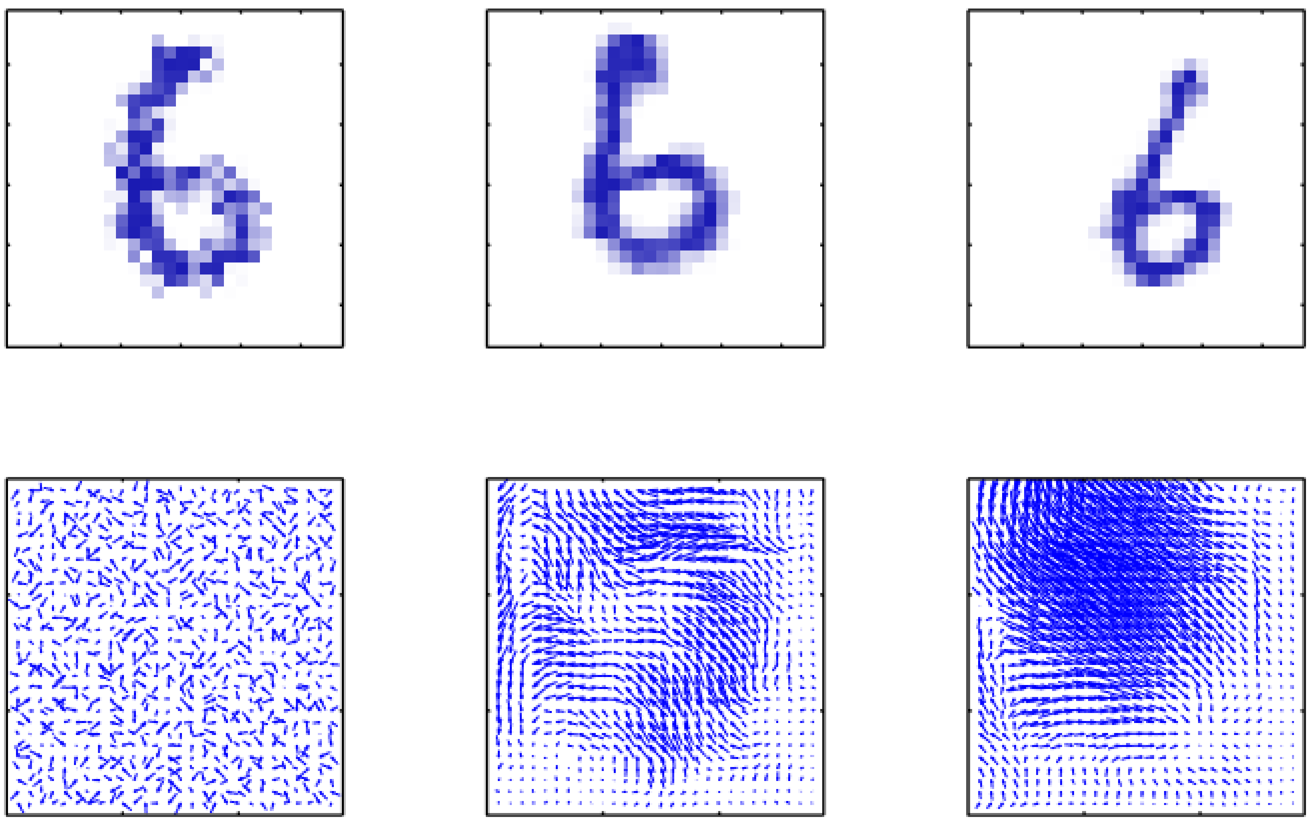
- （2）加入特定的正则化项，惩罚当输入进行某些变换时输出的变化。例如正切传播（Tangent Propagation）。
- （3）预处理。从输入空间抽取特征向量，特征向量对一些变换具有不变性，以特征向量作为NN的输入。
- （4）构造具有一定不变性结构的神经网络。例如采用局部感受野和权共享系统，一个例子是卷积神经网络（Convolutional Neural Networks：CNN）

不变性（Invariances）（续）



第一种：增广训练集的一个例子

一个样本增广了3个不同的变换。



样本加随机噪声（不变性）

样本加入随机噪声

$$\mathbf{x} \rightarrow \mathbf{x} + \boldsymbol{\xi}$$

相当于是一种正则化

$$\widetilde{E} = E + \lambda\Omega$$

正则项为：

$$\Omega = \frac{1}{2}\int \|\nabla y(\mathbf{x})\|^2 \, p(\mathbf{x}) \, \mathrm{d}\mathbf{x}$$

这种正则化称为Tikhonov正则化

卷积神经网络CNN（不变性）

图像和计算机视觉应用最广泛的一种前馈神经网络。
将卷积运算加入到神经网络，并结合应用如下技术
局部感受野
权共享
亚采样

这里只说明其是正则化、不变性的一类，
CNN的更详细讨论见（NN和DNN-II）
权共享实际上限制了系统复杂性，抑制过拟合

卷积神经网络CNN（不变性）（续）

只有一个卷积层的CNN的例子