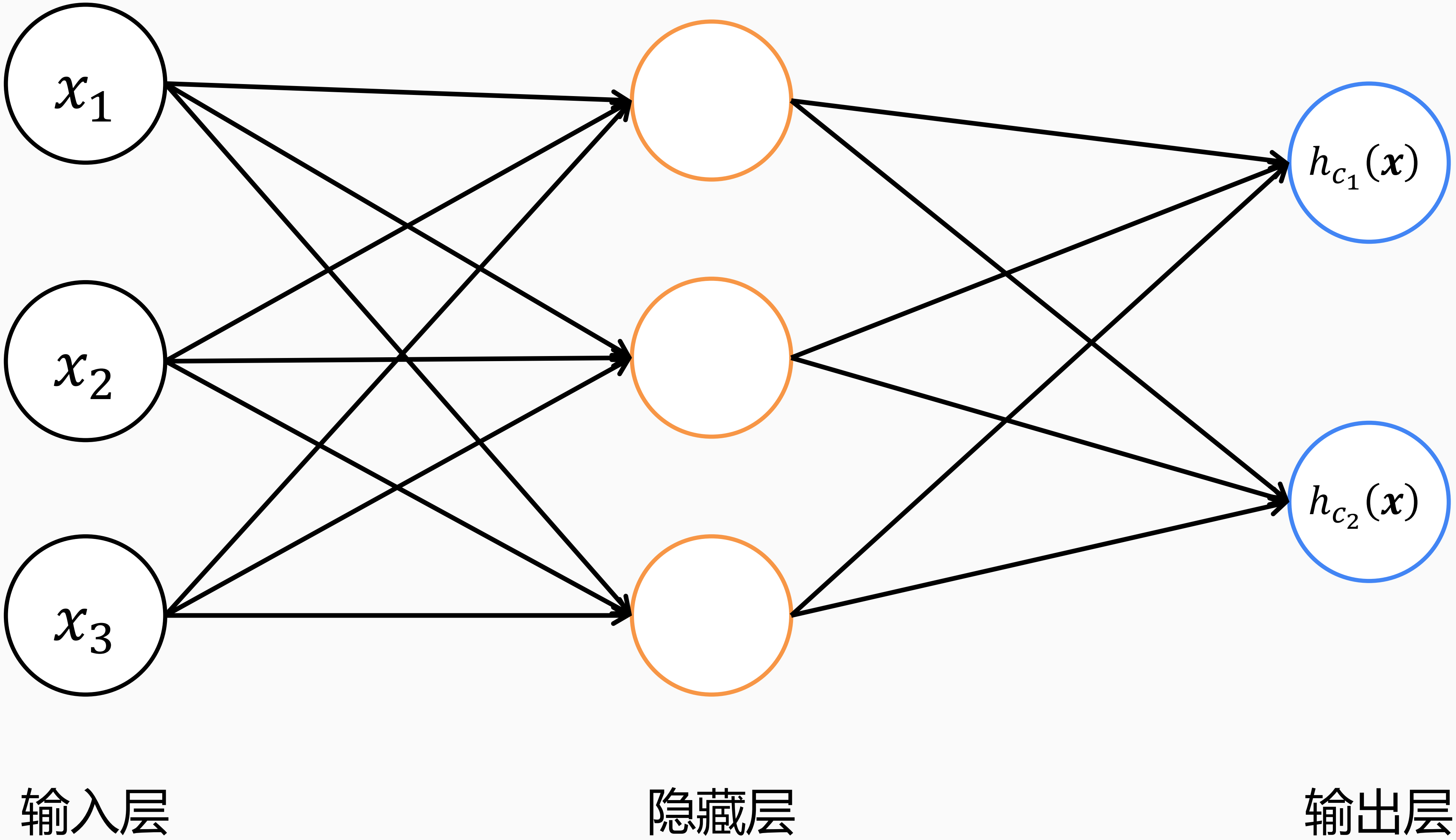
第六章
神经网络

李旻先
智能科学与技术系
计算机科学与工程学院
南京理工大学
minxianli@njust.edu.cn

神经网络

网络结构

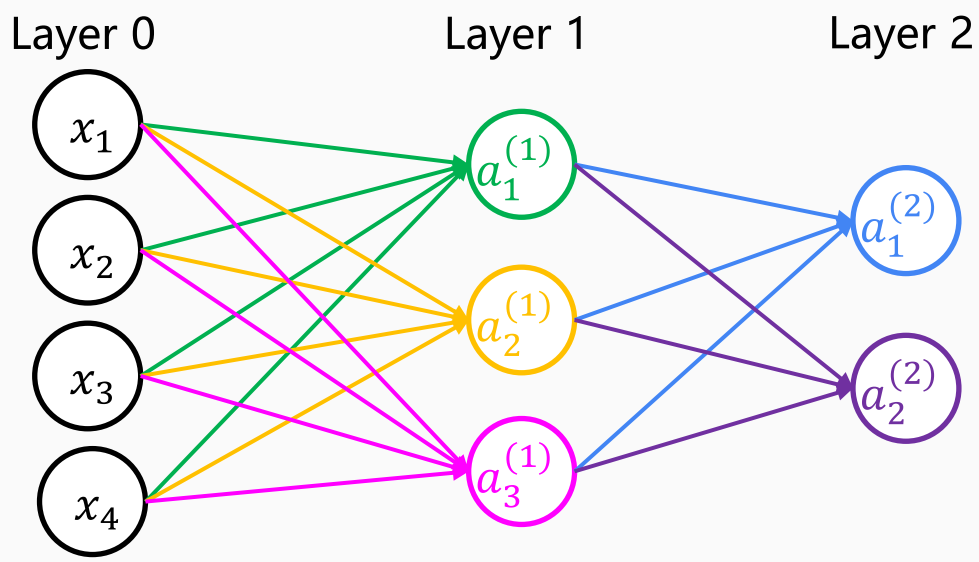
多类神经网络结构



神经网络

假设函数

神经网络的模型表示



L：神经网络的层数（一般只考虑隐藏层和输出层）

s_l：第l层的神经元数量

$\boldsymbol{W}^{(l)} \in \mathbb{R}^{s_{l-1} \times s_l}$: 第$l-1$层向第$l$层传播的**权重矩阵**

$\boldsymbol{b}^{(l)} \in \mathbb{R}^{s_l}$: 第$l-1$层向第$l$层传播的**偏置**

$\boldsymbol{z}^{(l)} \in \mathbb{R}^{s_l}$: 第$l$层神经元的**净输入值**

$\boldsymbol{a}^{(l)} \in \mathbb{R}^{s_l}$: 第$l$层神经元的输出值**（激活值）**

$g_l(\cdot)$: 第l层的神经元的**激活函数**

传播规则：

$$\boldsymbol{a}^{(l-1)} \in \mathbb{R}^{s_{l-1} \times 1} \xrightarrow[\boldsymbol{a}^{(l)} = g_l(\boldsymbol{z}^{(l)})]{\boldsymbol{z}^{(l)} = \left(\boldsymbol{W}^{(l)}\right)^T \boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}} \boldsymbol{a}^{(l)} \in \mathbb{R}^{s_l \times 1}$$

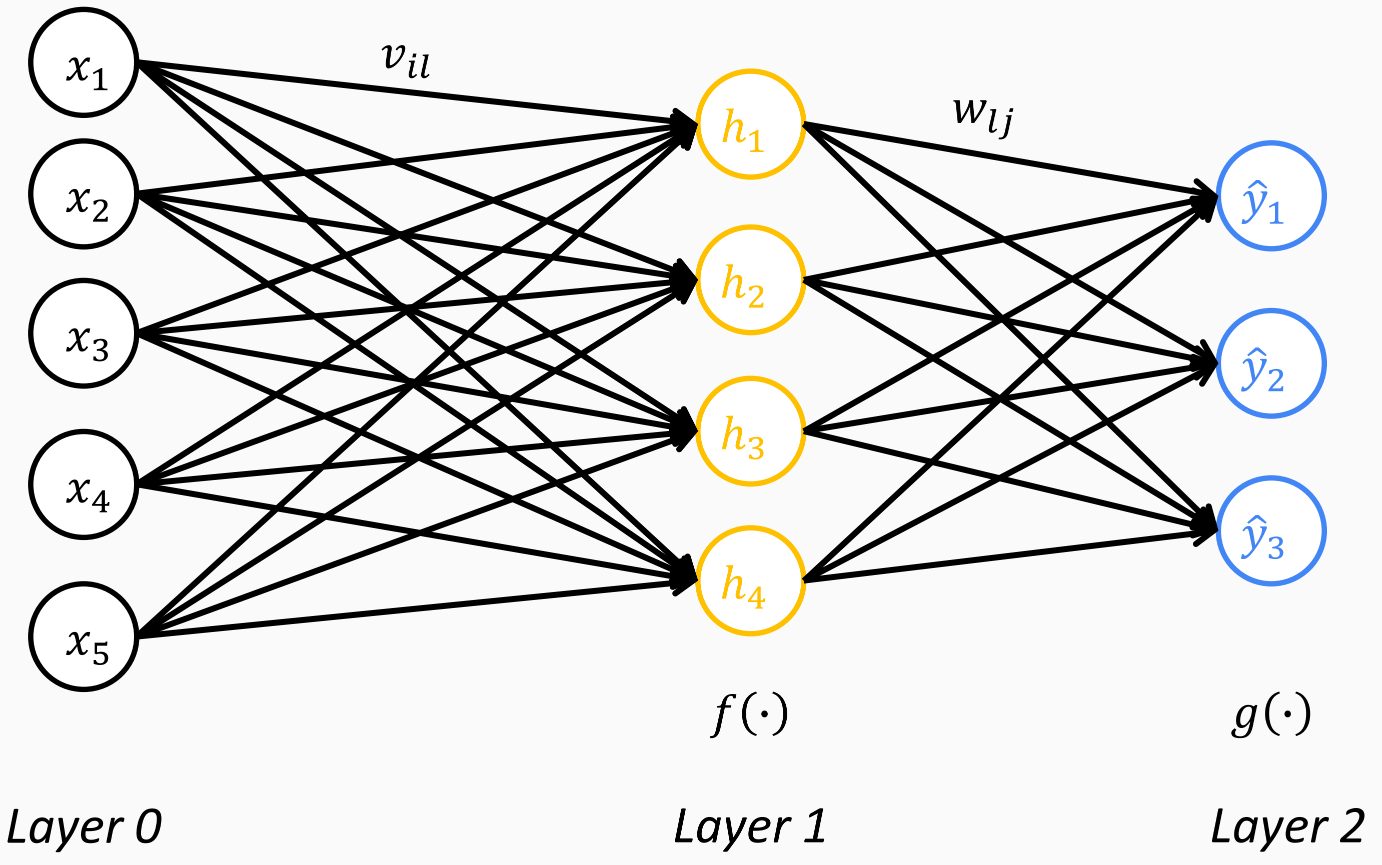
前向传播：

$$\boldsymbol{x} = \boldsymbol{a}^{(0)} \to (\boldsymbol{z}^{(1)} \to \boldsymbol{a}^{(1)}) \to \cdots \to (\boldsymbol{z}^{(L)} \to \boldsymbol{a}^{(L)} = h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}))$$

$\boldsymbol{W}, \boldsymbol{b}$表示所有层的权重矩阵和偏置

例题

三层前馈神经网络进行三分类问题建模，其中$\boldsymbol{x} \in \mathbb{R}^5, \boldsymbol{h} \in \mathbb{R}^4, \hat{\boldsymbol{y}} \in \mathbb{R}^3$分别是输入层、隐藏层和输出层的向量表示。假设输入层到隐藏层为全连接，$h_l = f(p_l), p_l = \sum_{i=1}^{5} x_i v_{il}$，$f$为sigmoid函数；隐藏层到输出层为全连接，$\hat{y}_j = g(o_j), o_j = \sum_{l=1}^{4} h_l w_{lj}$，$g$为softmax函数。求该网络的模型假设（即输入$\boldsymbol{x}$和输出$\hat{\boldsymbol{y}}$之间的关系）。



$$p_l = \sum_{i=1}^{5} x_i v_{il}$$

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

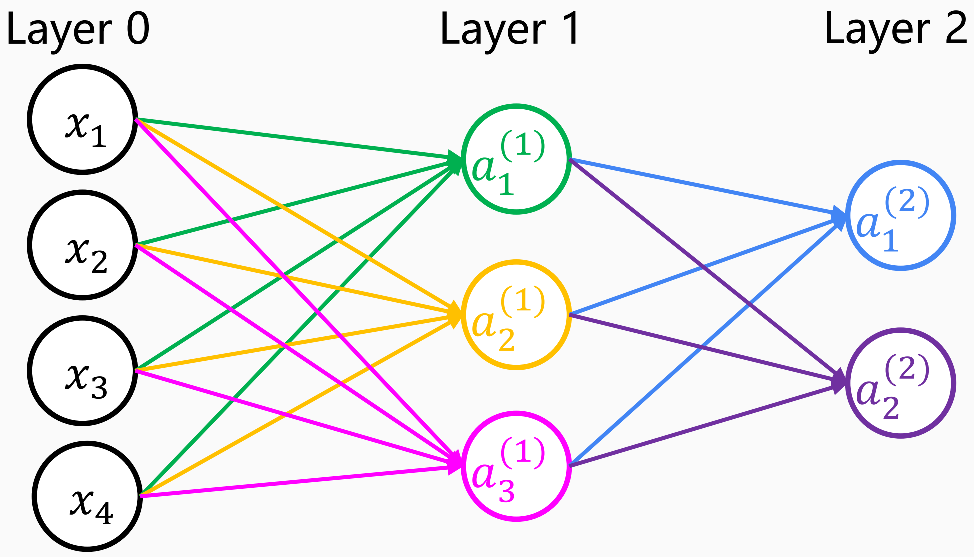
$$o_j = \sum_{l=1}^{4} h_l w_{lj}$$

$$\hat{y}_j = g(o_j) = \frac{e^{o_j}}{\sum_{c=1}^{C} e^{o_c}}$$

神经网络

损失函数

多分类



L : 神经网络的层数（一般只考虑隐藏层和输出层）

s_l : 第l层的神经元数量

$\boldsymbol{W}^{(l)} \in \mathbb{R}^{s_{l-1} \times s_l}$: 第$l-1$层向第$l$层传播的**权重矩阵**

$\boldsymbol{b}^{(l)} \in \mathbb{R}^{s_l}$: 第$l-1$层向第$l$层传播的**偏置**

$\boldsymbol{z}^{(l)} \in \mathbb{R}^{s_l}$: 第$l$层神经元的**净输入值**

$\boldsymbol{a}^{(l)} \in \mathbb{R}^{s_l}$: 第$l$层神经元的输出值**（激活值）**

$g_l(\cdot)$: 第l层的神经元的**激活函数**

传播规则：

$$\boldsymbol{a}^{(l-1)} \in \mathbb{R}^{s_{l-1} \times 1} \xrightarrow[\boldsymbol{a}^{(l)} = g_l(\boldsymbol{z}^{(l)})]{\boldsymbol{z}^{(l)} = \left(\boldsymbol{W}^{(l)}\right)^T \boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}} \boldsymbol{a}^{(l)} \in \mathbb{R}^{s_l \times 1}$$

前向传播：

$$\boldsymbol{x} = \boldsymbol{a}^{(0)} \to (\boldsymbol{z}^{(1)} \to \boldsymbol{a}^{(1)}) \to \cdots \to \left(\boldsymbol{z}^{(L)} \to \boldsymbol{a}^{(L)} = softmax\left(\boldsymbol{z}^{(l)}\right) = h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x})\right)$$

$\boldsymbol{W}, \boldsymbol{b}$表示所有层的权重矩阵和偏置

多分类的损失函数——交叉熵

$$a_c^{(L)} = \left(h_{\boldsymbol{W},\boldsymbol{b}}\left(\boldsymbol{x}^{(i)}\right)\right)_c = \frac{e^{z_c^{(L)}}}{\sum_{c'=1}^{C} e^{z_{c'}^{(L)}}}$$

假设神经网络有C个输出，则$h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}) \in \mathbb{R}^C$，$\left(h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x})\right)_c$表示第$c$个输出神经元的预测概率值。

$$J_{\boldsymbol{W},\boldsymbol{b}}\left(h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}), \boldsymbol{y}\right) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{c=1}^{C}\left[\boldsymbol{y}^{(i)}\right]_c \log\left(h_{\boldsymbol{W},\boldsymbol{b}}\left(\boldsymbol{x}^{(i)}\right)\right)_c$$

根据任务和学习准则的不同，可以设计不同的损失函数

神经网络

优化方法

梯度下降法

损失函数 $$J_{\boldsymbol{W},\boldsymbol{b}}(h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}),\boldsymbol{y})$$

梯度下降法

$$\boldsymbol{W}^{(l)} := \boldsymbol{W}^{(l)} - \alpha \boxed{\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(l)}}}$$

$$\boldsymbol{b}^{(l)} := \boldsymbol{b}^{(l)} - \alpha \boxed{\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{b}^{(l)}}}$$

梯度

因为$\boldsymbol{z}^{(l-1)}$与$\boldsymbol{z}^{(l)}$有这样的关系： $\boldsymbol{a}^{(l-1)} = g(\boldsymbol{z}^{(l-1)})$

$$\boldsymbol{z}^{(l)} = \left(\boldsymbol{W}^{(l)}\right)^T \boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$$

$$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(l)}} = \frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{W}^{(l)}} \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}} = \boldsymbol{a}^{(l-1)} \left(\boxed{\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}}}\right)^T$$

直接求导，繁琐低效

$$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{b}^{(l)}} = \frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{b}^{(l)}} \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}} = \boldsymbol{I} \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}} = \boxed{\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}}}$$

怎么求?

误差

将$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}}$记为$\boldsymbol{\delta}^{(l)}$，又因为$\boldsymbol{z}^{(l)}$与$\boldsymbol{z}^{(l+1)}$有这样的关系：

$$\boldsymbol{a}^{(l)} = g(\boldsymbol{z}^{(l)})$$

$$\boldsymbol{z}^{(l+1)} = \left(\boldsymbol{W}^{(l+1)}\right)^{T} \boldsymbol{a}^{(l)} + \boldsymbol{b}^{(l+1)}$$

当前层的误差　　　　　已知　　　　　后一层的误差

$$\boxed{\boldsymbol{\delta}^{(l)}} = \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}} = \frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}} \frac{\partial \boldsymbol{z}^{(l+1)}}{\partial \boldsymbol{a}^{(l)}} \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l+1)}} = \boxed{diag(g'(\boldsymbol{z}^{(l)}))\boldsymbol{W}^{(l+1)}}\boxed{\boldsymbol{\delta}^{(l+1)}}$$

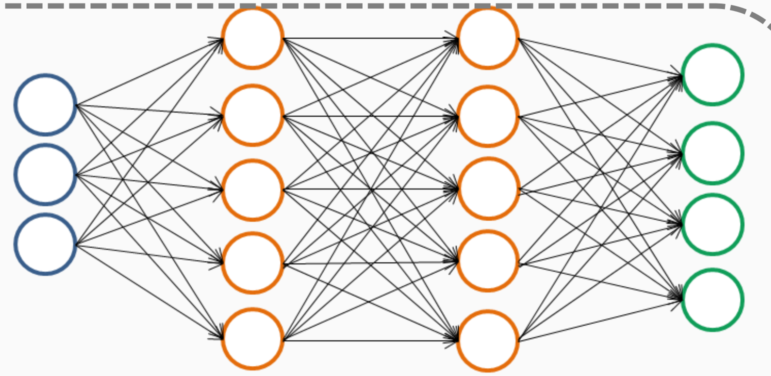
$$= g'(\boldsymbol{z}^{(l)}) \odot (\boldsymbol{W}^{(l+1)} \boldsymbol{\delta}^{(l+1)})$$

⊙为Hadamard乘积

当前层的误差$\delta^{(l)}$可由后一层误差$\delta^{(l+1)}$计算得到，则可求当前层的梯度

反向（误差）传播算法

① 利用损失函数求得模型的最终误差$\boldsymbol{\delta}^{(L)} = \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(L)}}$，接着再将误差自后向前层层传递，获取每层神经元的误差$\boldsymbol{\delta}^{(l)} = g'\left(\boldsymbol{z}^{(l)}\right) \odot \left(\boldsymbol{W}^{(l+1)} \boldsymbol{\delta}^{(l+1)}\right)$。



$\boldsymbol{\delta}^{(1)} \longleftarrow \boldsymbol{\delta}^{(l)} \longleftarrow \boldsymbol{\delta}^{(L)}$

② 根据每层神经元的误差$\boldsymbol{\delta}^{(l)}$对$\boldsymbol{W}^{(l)}$和$\boldsymbol{b}^{(l)}$求偏导，求得梯度$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(l)}}$和$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{b}^{(l)}}$。

$$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(l)}} = \boldsymbol{a}^{(l-1)}\left(\boldsymbol{\delta}^{(l)}\right)^{T}$$

$$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{b}^{(l)}} = \boldsymbol{\delta}^{(l)}$$

③ 根据梯度，更新参数$\boldsymbol{W}^{(l)}$和$\boldsymbol{b}^{(l)}$。

$$\boldsymbol{W}^{(l)} := \boldsymbol{W}^{(l)} - \alpha \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(l)}}$$

$$\boldsymbol{b}^{(l)} := \boldsymbol{b}^{(l)} - \alpha \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{b}^{(l)}}$$

损失函数——基于Softmax函数的交叉熵

$$J_{\boldsymbol{W},\boldsymbol{b}}(h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}),\boldsymbol{y}) = \sum_{c=1}^{C} -[\boldsymbol{y}]_c \log\left(h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x})\right)_c = \sum_{c=1}^{C} -[\boldsymbol{y}]_c \log a_c^{(L)} = -\log\left(a_y^{(L)}\right)$$

$$\boldsymbol{a}^{(L)} = g(\boldsymbol{z}^{(L)}) = softmax(\boldsymbol{z}^{(L)})$$

$$\delta_c^{(L)} = \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial z_c^{(L)}} = \frac{\partial a_y^{(L)}}{\partial z_c^{(L)}} \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial a_y^{(L)}} = -\frac{\partial a_y^{(L)}}{\partial z_c^{(L)}} \frac{1}{a_y^{(L)}}$$

Softmax函数的导数 $\frac{\partial g(\boldsymbol{z})_i}{\partial z_j} = \begin{cases} g(\boldsymbol{z})_i(1 - g(\boldsymbol{z})_i) & i = j \\ -g(\boldsymbol{z})_i g(\boldsymbol{z})_j & i \neq j \end{cases}$

损失函数——交叉熵

Softmax函数的导数 $$\frac{\partial g(\mathbf{z})_i}{\partial z_j} = \begin{cases} g(\mathbf{z})_i(1 - g(\mathbf{z})_i) & i = j \\ -g(\mathbf{z})_i g(\mathbf{z})_j & i \neq j \end{cases}$$

$$\delta_c^{(L)} = \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial z_c^{(L)}} = -\frac{\partial a_y^{(L)}}{\partial z_c^{(L)}} \frac{1}{a_y^{(L)}} = \begin{cases} a_y^{(L)}\left(a_y^{(L)} - 1\right)\frac{1}{a_y^{(L)}} = a_c^{(L)} - 1 & y = c \\ a_y^{(L)} a_c^{(L)} \frac{1}{a_y^{(L)}} = a_c^{(L)} & y \neq c \end{cases}$$

$$= a_c^{(L)} - [\boldsymbol{y}]_c$$

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

损失函数——平方误差

$$J_{\boldsymbol{W},\boldsymbol{b}}(h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}),\boldsymbol{y})=\frac{1}{2}\left\|h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x})-\boldsymbol{y}\right\|_2^2=\frac{1}{2}\sum_{c=1}^{C}(a_c^{(L)}-[\boldsymbol{y}]_c)^2$$

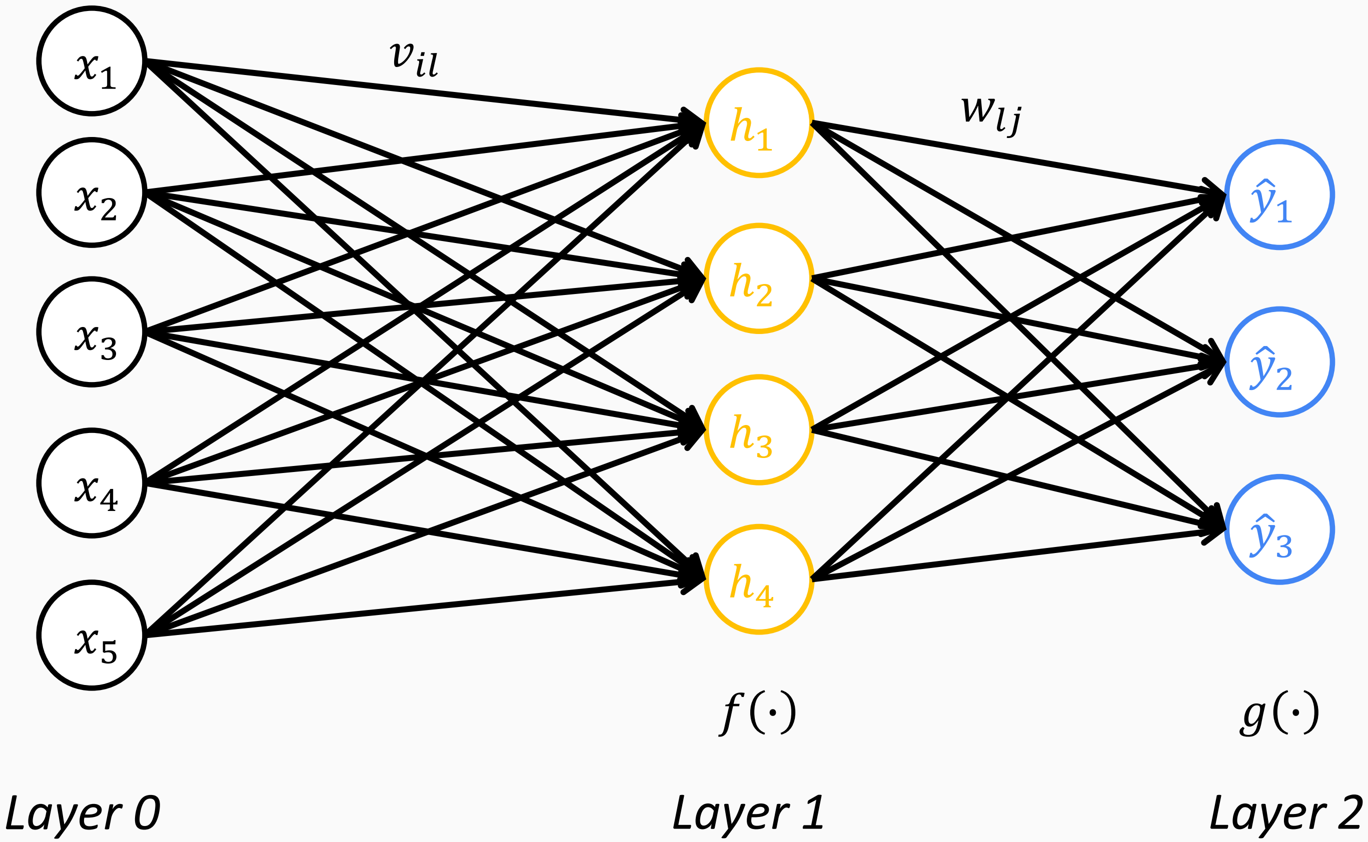
$$\boldsymbol{a}^{(L)}=g(\boldsymbol{z}^{(L)})=sigmoid(\boldsymbol{z}^{(L)})$$

$$\delta_c^{(L)}=\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial z_c^{(L)}}=\frac{\partial a_c^{(L)}}{\partial z_c^{(L)}}\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial a_c^{(L)}}=a_c^{(L)}\left(1-a_c^{(L)}\right)\left[a_c^{(L)}-[\boldsymbol{y}]_c\right]$$

$$\boldsymbol{\delta}^{(L)}=\mathrm{diag}\left(a_c^{(L)}\left(1-a_c^{(L)}\right)\right)\left[\boldsymbol{a}^{(L)}-\boldsymbol{y}\right]$$

例题

三层前馈神经网络进行三分类问题建模，其中$\boldsymbol{x} \in \mathbb{R}^5, \boldsymbol{h} \in \mathbb{R}^4, \hat{\boldsymbol{y}} \in \mathbb{R}^3$分别是输入层、隐藏层和输出层的向量表示。假设输入层到隐藏层为全连接，$h_l = f(p_l), p_l = \sum_{i=1}^{5} x_i v_{il}$，$f$为sigmoid函数；隐藏层到输入层为全连接，$\hat{y}_j = g(o_j), o_j = \sum_{l=1}^{4} h_l w_{lj}$，$g$为softmax函数。若损失函数为交叉熵损失，求$w_{32}$和$v_{43}$的更新规则。



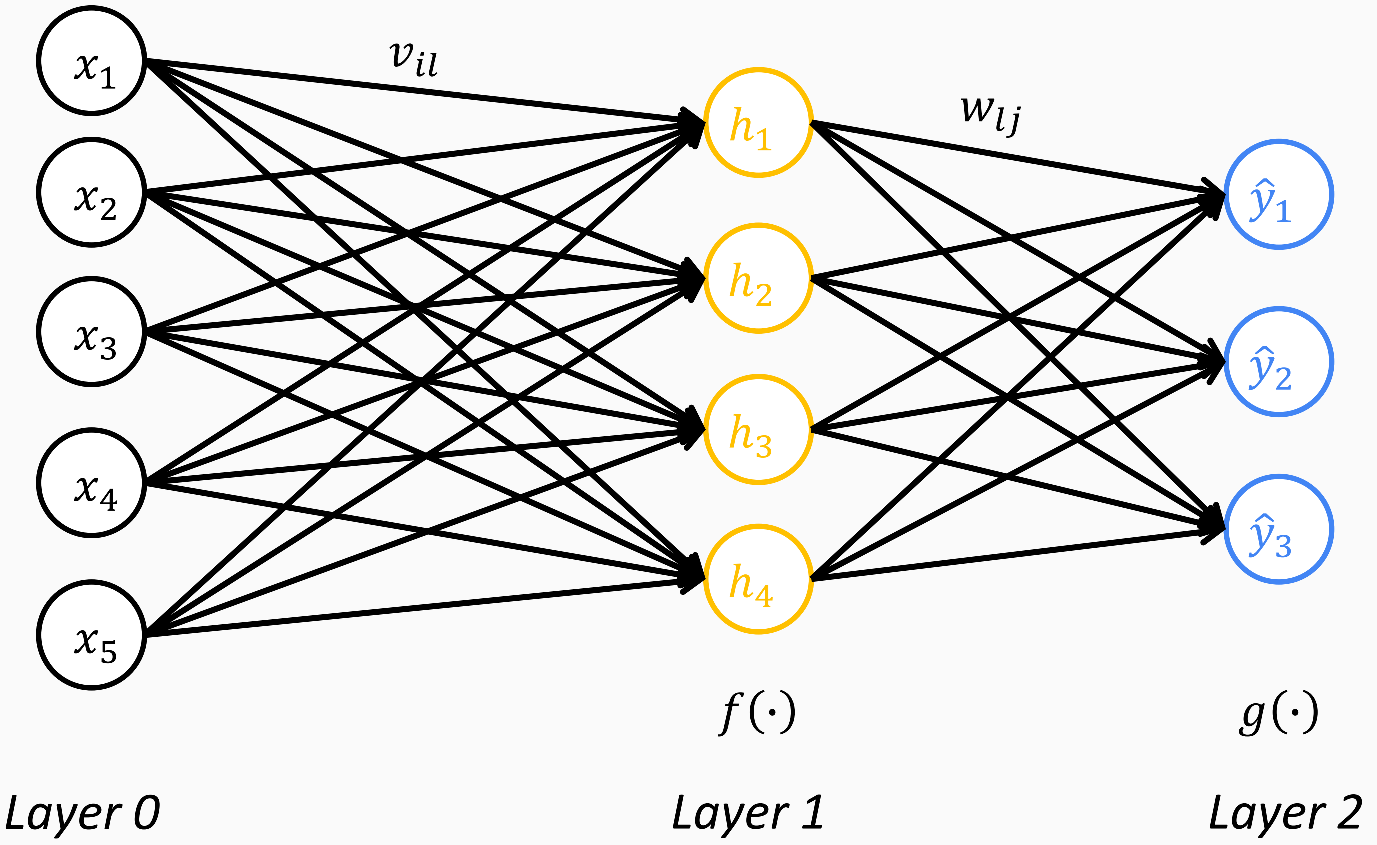
$\boldsymbol{W}^{(l)} \in \mathbb{R}^{s_{l-1} \times s_l}$: 第$l-1$层向第$l$层传播的**权重矩阵**

①$v_{43} \in \boldsymbol{W}^{(1)}, \boldsymbol{W}^{(1)} \in \mathbb{R}^{5\times4}$

②$w_{32} \in \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(2)} \in \mathbb{R}^{4\times3}$

例题

三层前馈神经网络进行三分类问题建模，其中$\boldsymbol{x} \in \mathbb{R}^5, \boldsymbol{h} \in \mathbb{R}^4, \hat{\boldsymbol{y}} \in \mathbb{R}^3$分别是输入层、隐藏层和输出层的向量表示。假设输入层到隐藏层为全连接，$h_l = f(p_l), p_l = \sum_{i=1}^{5} x_i v_{il}$，$f$为sigmoid函数；隐藏层到输入层为全连接，$\hat{y}_j = g(o_j), o_j = \sum_{l=1}^{4} h_l w_{lj}$，$g$为softmax函数。若损失函数为交叉熵损失，求$w_{32}$和$v_{43}$的更新规则。



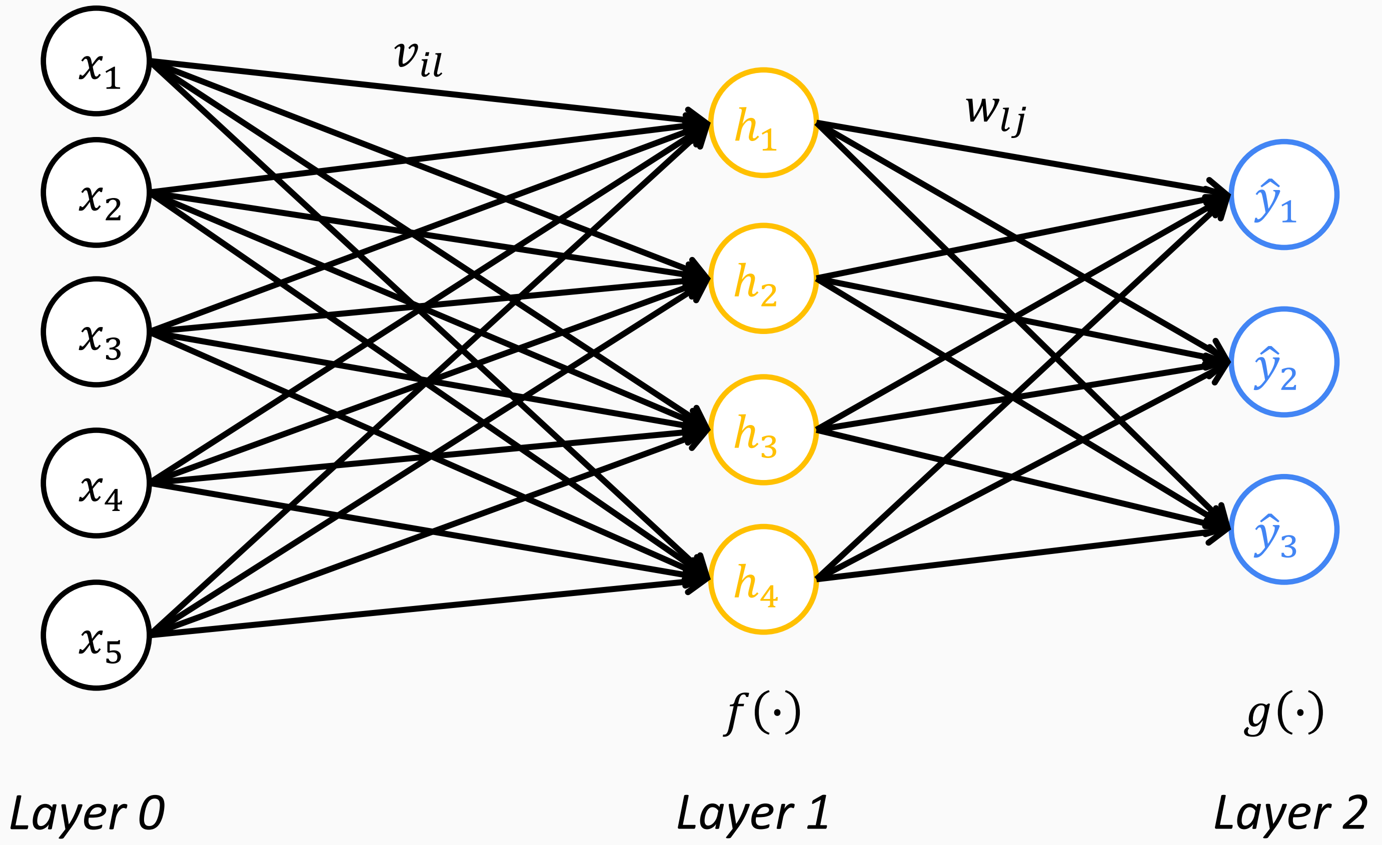
① $\boldsymbol{W}^{(l)} := \boldsymbol{W}^{(l)} - \alpha \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(l)}}$

② $\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(l)}} = \boldsymbol{a}^{(l-1)} \left(\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}}\right)^T$

③ $\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(l)}} = \boldsymbol{\delta}^{(l)} = g'\left(\boldsymbol{z}^{(l)}\right) \odot \boldsymbol{W}^{(l+1)} \boldsymbol{\delta}^{(l+1)}$

例题

三层前馈神经网络进行三分类问题建模，其中$\boldsymbol{x} \in \mathbb{R}^5, \boldsymbol{h} \in \mathbb{R}^4, \hat{\boldsymbol{y}} \in \mathbb{R}^3$分别是输入层、隐藏层和输出层的向量表示。假设输入层到隐藏层为全连接，$h_l = f(p_l), p_l = \sum_{i=1}^{5} x_i v_{il}$，$f$为sigmoid函数；隐藏层到输入层为全连接，$\hat{y}_j = g(o_j), o_j = \sum_{l=1}^{4} h_l w_{lj}$，$g$为softmax函数。若损失函数为交叉熵损失，求$w_{32}$和$v_{43}$的更新规则。



损失函数 $J_{\boldsymbol{W},\boldsymbol{b}} = -\log\left(a_y^{(2)}\right)$

$$\boldsymbol{\delta}^{(2)} = \frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{z}^{(2)}} = \boldsymbol{a}^{(2)} - \boldsymbol{y}$$

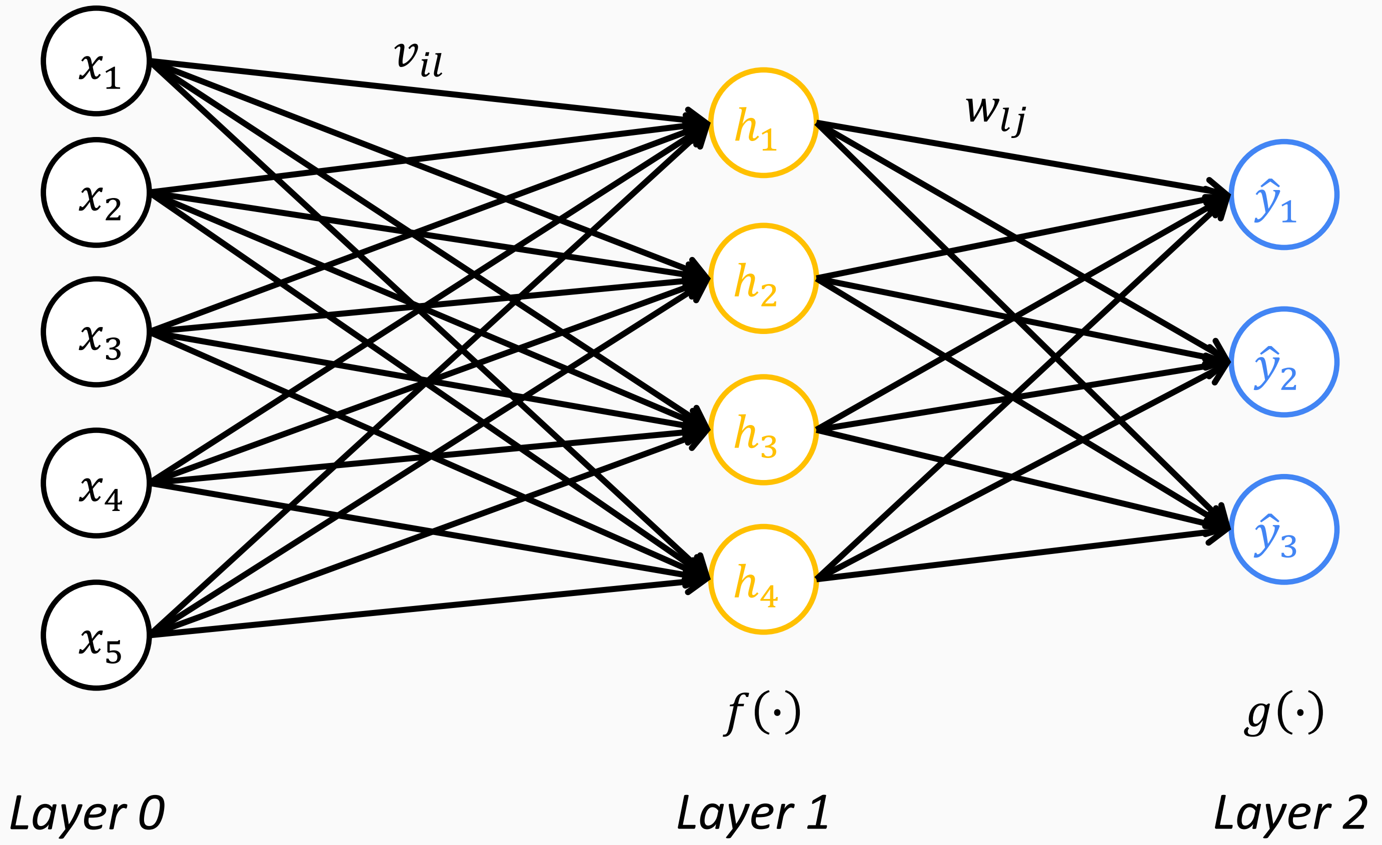
$$= \hat{\boldsymbol{y}} - \boldsymbol{y}$$

$$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(2)}} = \boldsymbol{a}^{(1)}\left(\boldsymbol{\delta}^{(2)}\right)^T = \boldsymbol{h}(\hat{\boldsymbol{y}} - \boldsymbol{y})^T$$

$$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial w_{32}} = h_3(\hat{\boldsymbol{y}}_2 - \boldsymbol{y}_2)$$

例题

三层前馈神经网络进行三分类问题建模，其中$\boldsymbol{x} \in \mathbb{R}^5, \boldsymbol{h} \in \mathbb{R}^4, \hat{\boldsymbol{y}} \in \mathbb{R}^3$分别是输入层、隐藏层和输出层的向量表示。假设输入层到隐藏层为全连接，$h_l = f(p_l), p_l = \sum_{i=1}^{5} x_i v_{il}$，$f$为sigmoid函数；隐藏层到输入层为全连接，$\hat{y}_j = g(o_j), o_j = \sum_{l=1}^{4} h_l w_{lj}$，$g$为softmax函数。若损失函数为交叉熵损失，求$w_{32}$和$v_{43}$的更新规则。



$$\boldsymbol{\delta}^{(l)} = diag(g'(\boldsymbol{z}^{(l)}))\boldsymbol{W}^{(l+1)}\boldsymbol{\delta}^{(l+1)}$$

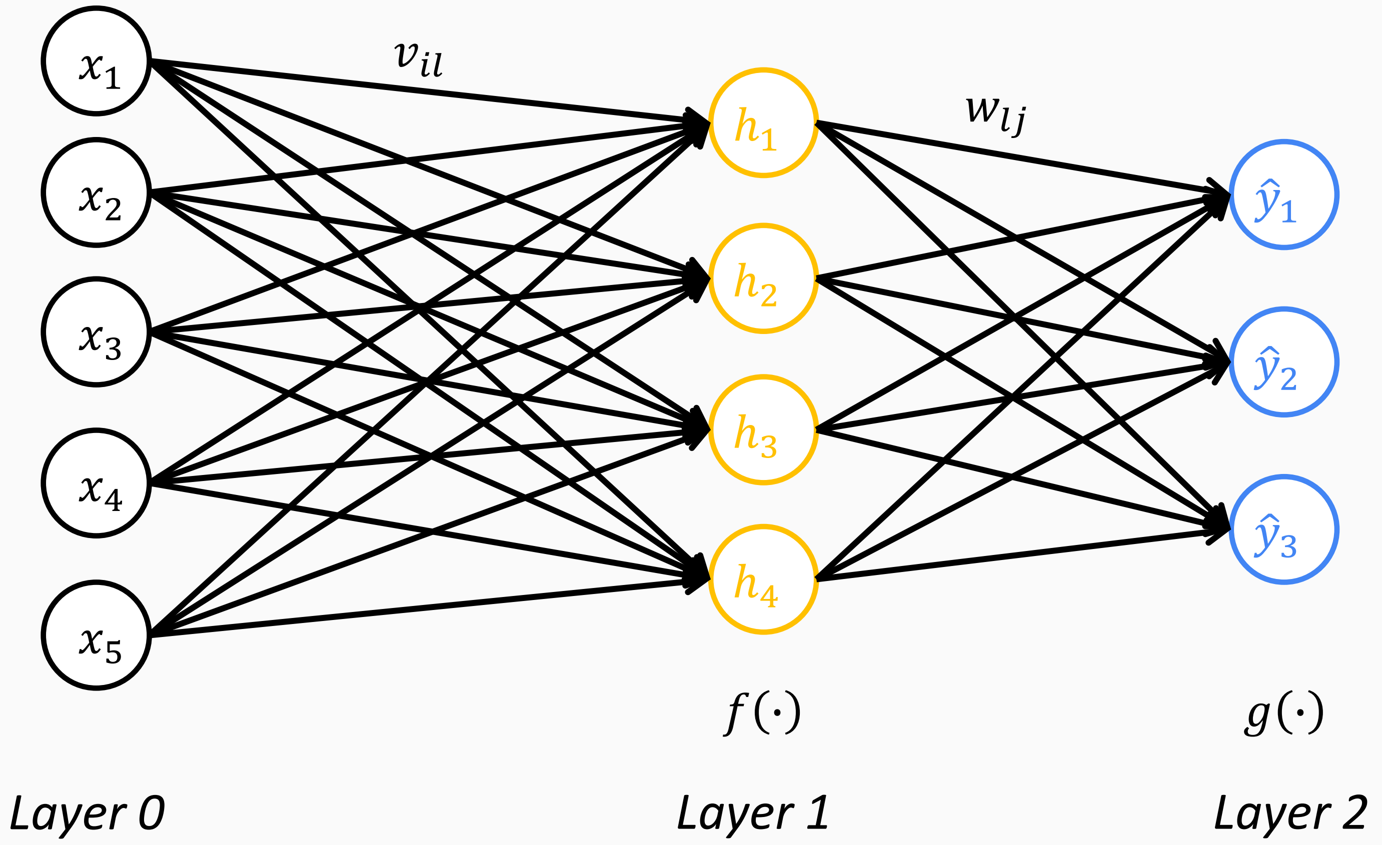
$$\boldsymbol{\delta}^{(1)} = diag(f'(\boldsymbol{p}))\boldsymbol{W}^{(2)}\boldsymbol{\delta}^{(2)}$$

$$= \begin{bmatrix} f'(p_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f'(p_4) \end{bmatrix} \begin{bmatrix} w_{11} & \cdots & w_{13} \\ \vdots & \ddots & \vdots \\ w_{41} & \cdots & w_{43} \end{bmatrix} (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

$$= \begin{bmatrix} f'(p_1) \sum_{j=1}^{3} w_{1j}(\hat{y}_j - y_j) \\ f'(p_2) \sum_{j=1}^{3} w_{2j}(\hat{y}_j - y_j) \\ f'(p_3) \sum_{j=1}^{3} w_{3j}(\hat{y}_j - y_j) \\ f'(p_4) \sum_{j=1}^{3} w_{4j}(\hat{y}_j - y_j) \end{bmatrix}$$

例题

三层前馈神经网络进行三分类问题建模，其中$\boldsymbol{x} \in \mathbb{R}^5, \boldsymbol{h} \in \mathbb{R}^4, \hat{\boldsymbol{y}} \in \mathbb{R}^3$分别是输入层、隐藏层和输出层的向量表示。假设输入层到隐藏层为全连接，$h_l = f(p_l), p_l = \sum_{i=1}^{5} x_i v_{il}$，$f$为sigmoid函数；隐藏层到输入层为全连接，$\hat{y}_j = g(o_j), o_j = \sum_{l=1}^{4} h_l w_{lj}$，$g$为softmax函数。若损失函数为交叉熵损失，求$w_{32}$和$v_{43}$的更新规则。

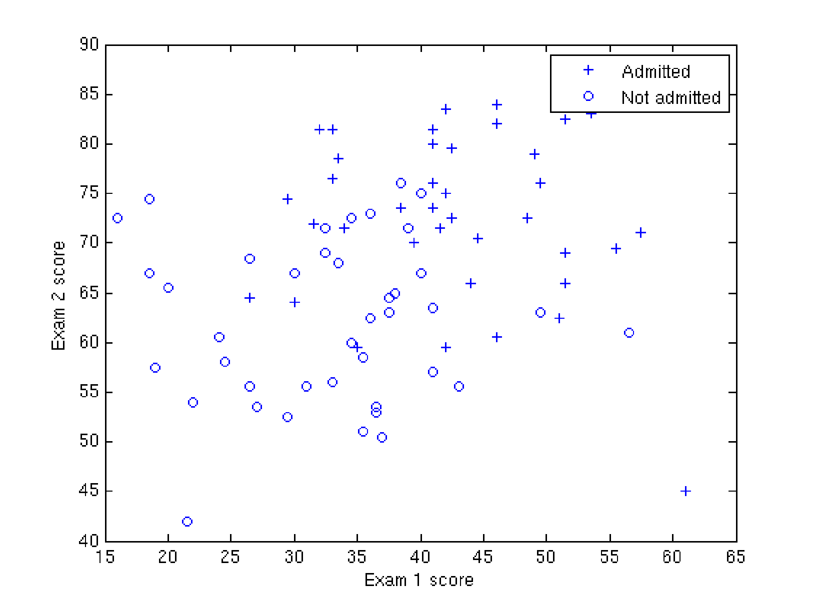


$$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial \boldsymbol{W}^{(1)}} = \boldsymbol{a}^{(0)}\left(\boldsymbol{\delta}^{(1)}\right)^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \begin{bmatrix} f'(p_1) \sum_{j=1}^{3} w_{1j}(\hat{y}_j - y_j) \\ f'(p_2) \sum_{j=1}^{3} w_{2j}(\hat{y}_j - y_j) \\ f'(p_3) \sum_{j=1}^{3} w_{3j}(\hat{y}_j - y_j) \\ f'(p_4) \sum_{j=1}^{3} w_{4j}(\hat{y}_j - y_j) \end{bmatrix}^T$$

$$\frac{\partial J_{\boldsymbol{W},\boldsymbol{b}}}{\partial v_{43}} = x_4 h_3(1 - h_3) \sum_{j=1}^{3} w_{3j}(\hat{y}_j - y_j)$$

作业5: 基于BP算法的三层前向神经网络

- 给出训练数据:



http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=DeepLearning&doc=exercises/ex4/ex4.html

- 使用BP算法实现三层前向神经网络 (自己编码，不要使用Tensorflow/Pytorch等框架)，并对结果进行5倍交叉验证；
- 将其与Logistic回归和Softmax回归进行比较。