人工神经网络

卜晨阳

邮箱：**chenyangbu@hfut.edu.cn**

本章内容

- 监督学习规则
- 神经网络的类型
- 隐层单元的功能
- 集成神经网络

监督学习规则

- 监督学习问题
- 梯度下降优化
- 量化共轭梯度
- 粒子群优化
- 进化策略

监督学习问题

考虑从平稳密度$\Omega(D)$采样的数据集

$$D=\{d_p=(z_p,t_p)\,|\,p=1,\cdots,P\}$$

其中，z_p是输入，t_p是对应的输出。

对于所有$i=1,\cdots,I$，$k=1,\cdots,K$，$z_{i,p}$是模式p对应输入单元z_i的值，$t_{k,p}$是输出单元o_k的目标值，有$z_{i,p},t_{k,p}\in R$。

根据信号加噪声模型，有：

$t_p=\mu(z_p)+\varsigma_p$，其中$\mu(z)$是未知函数；$\varsigma_{k,p}$是以密度$\phi(\varsigma)$采用的独立的、同分布的噪声样本，其均值为0。

- **学习的目标是使用有限集合D中包含的信息来逼近未知函数$\mu(z)$。**

监督学习问题

- 对于神经网络学习而言，通常将数据集D随机划分为训练集合D_T、验证集合D_V以及测试集合D_G，集合之间相互独立。
  - 对$\mu(z)$的逼近由训练集D_T建立（用于调节权值）。
  - 记忆（memorization）由D_V确定（用于最小化过拟合）。
  - 泛化精度则由测试集D_G进行估计（测试最终解证实网络的真实效果）。
- 对于多层神经网络，给定网络结构时，其搜索空间是由权值向量W所描述的搜索空间。
- 通常，关于$\Omega(D)$的先验知识一无所知，因此神经网络学习器会使用非参数回归方法在其假设空间$\mathcal{H}$中搜索能对未知函数$\mu(z)$给出良好估计的函数$f_{NN}(D_T,W)$。

监督学习问题

- **最小化经验误差：**

$$\varepsilon_T(D_T;\mathbf{W})=\frac{1}{P_T}\sum_{p=1}^{P_T}(f_{NN}(z_p,\mathrm{W})-t_p)^2$$

其中，P_T是训练模式的总数，函数F_{NN}：$\mathrm{R}^{\mathrm{I}}\rightarrow\mathrm{R}^{K}$。

期望：低的经验（训练）误差同样会给出低的真实误差，或泛化误差。

真实误差或泛化误差被定义为：

$$\varepsilon_T(\Omega;\mathbf{W})=\int(f_{NN}(z,\mathrm{W})-t)^2\mathrm{d}\Omega(z,t)$$

监督学习问题

- 训练神经网络的优化算法分3类：
  - 局部优化，例如梯度下降法、量化共轭梯度法等。
  - 全局优化，例如模拟退火、进化算法、粒子群优化等。
  - 局部优化和全局优化组合起来的混合训练算法。

- 根据何时更新权值，有两类监督学习算法。
  - 随机/在线学习：每出现一个模式，立即更新权值。为防止模式在训练集中的出现次序可能引起的误差，通常随机选取下一个输入模式。
  - 批/离线学习：仅当使用完所有的训练模式后，才使用累积变化来对权值进行调整。

梯度下降优化

- 从梯度下降优化方法导出了一个最流行的学习方法，即反向传播（**backpropagation**）。

- 一次学习迭代包括两个阶段：
  - 前馈传递：对于每一个训练模式简单地计算输出值。
  - 反向传播：从输出层向输入层反向传播误差信号。权值作为反向传播误差信号的函数被调整。

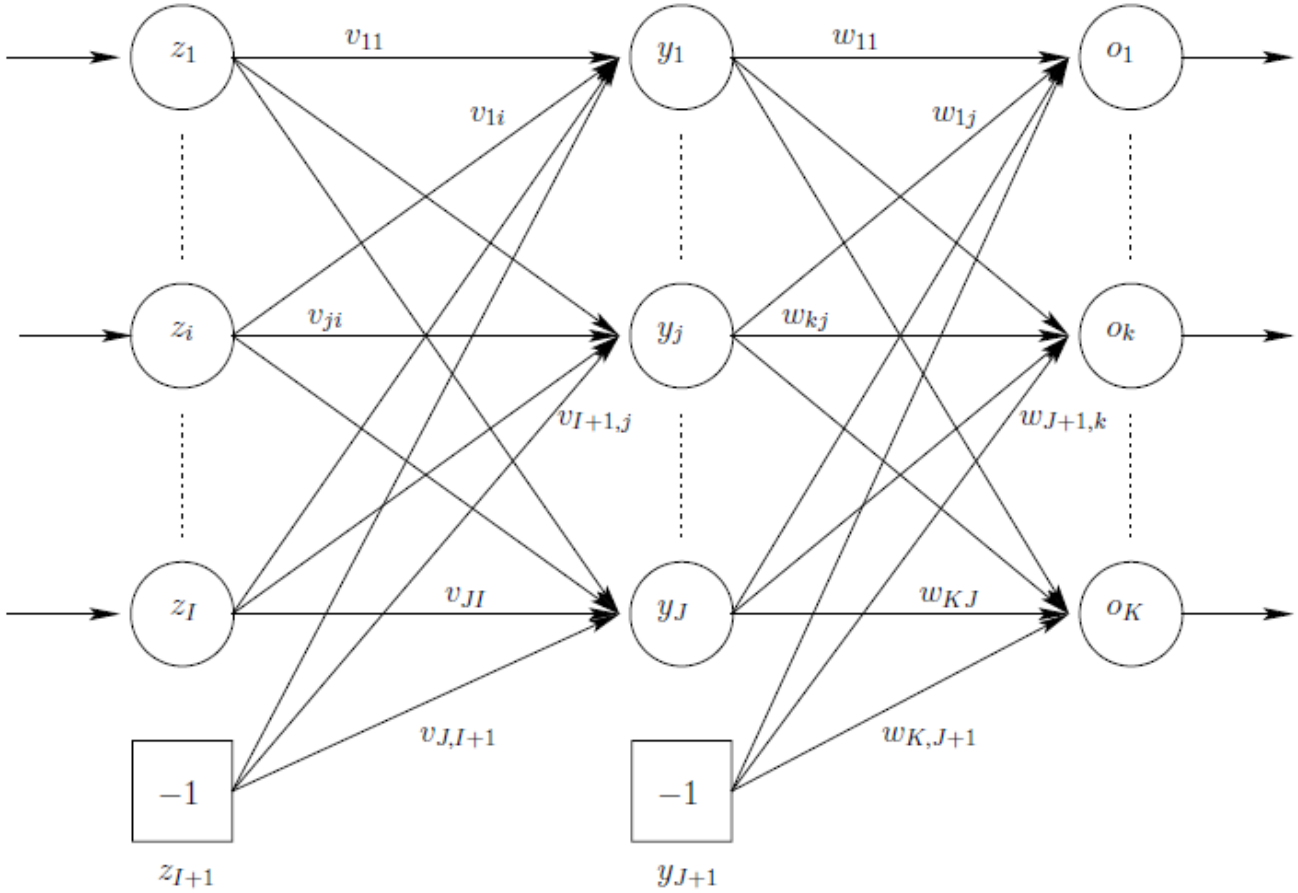
反向传播

- 问题：
- 链式法则更新权重虽然简单，但过于冗长，是否可以减少不必要的重复计算？

- 已经计算完毕的节点我们完全可以直接拿来用
- 先更新后边的权重，之后再在此基础上利用更新后边的权重产生的中间值来更新较靠前的参数

梯度下降优化

- 神经网络假设为：标准的前馈神经网络
  - 一个输入层、一个隐层、一个输出层



梯度下降优化

- 对于任意给定的一个输入模式z_p，前馈神经网络的输出单元o_k的输出为

$$o_{k,p} = f_{o_k}(net_{o_{k,p}})$$

$$= f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}(net_{y_{j,p}})\right)$$

$$= f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}(\sum_{i=1}^{I+1} v_{ji} z_{i,p})\right)$$

f_{o_k}和f_{y_j}分别是输出单元o_k和隐层单元y_j的激活函数。

$z_{i,p}$是输入模式z_p的输入单元z_i的值。第$(I+1)$个输入单元和第$(J+1)$个隐层单元代表下一层中的神经元的阈值的偏置单元。

梯度下降优化—前馈神经网络

- 假定以误差平方和作为目标函数，则对于每一个模式$\mathbf{z_p}$，有

$$\varepsilon_p = \frac{1}{2}\left(\frac{\sum_{k=1}^{K}(t_{k,p}-o_{k,p})^2}{K}\right)$$

其中，K是输出单元的个数，$t_{k,p}$和$o_{k,p}$分别是第k个输出单元的目标输出值和实际输出值。

以下推导均针对单个模式。因此，为了表示上的方便，将模式的下标p省略。同时，假定隐层和输出层均使用Sigmoid激活函数，使用增广向量，所有隐层和输出层单元均使用求和单元。

梯度下降优化—前馈神经网络

- 则：

$$o_k = f_{o_k}(net_{o_k}) = \frac{1}{1+e^{-net_{ok}}}，\ y_j = f_{y_j}(net_{y_j}) = \frac{1}{1+e^{-net_{yj}}}$$

在随机学习中，权值根据下列等式进行更新：

$$w_{kj}(t)+ = \Delta w_{kj}(t) + \alpha \Delta w_{kj}(t-1)$$

$$v_{ji}(t)+ = \Delta v_{ji}(t) + \alpha \Delta v_{ji}(t-1)$$

其中，α是冲量。

下面将推导计算$\triangle w_{kj}(t)$和$\triangle v_{ji}(t)$等式。为了表示上的便捷，推导中把表示时间的符号t省略。

梯度下降优化—前馈神经网络

- 因为

$$o_k = f_{o_k}(net_{o_k}) = \frac{1}{1+e^{-net_{ok}}}$$

有

$$\frac{\partial o_k}{\partial net_{o_k}} = \frac{\partial f_{o_k}}{\partial net_{o_k}} = (1-o_k)o_k = f'_{o_k},$$ （注：f'_{o_k}是对应激活函数的导数） (1)

$$\frac{\partial net_{o_k}}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}}\left(\sum_{j=1}^{J+1} w_{kj} y_j\right) = y_j,$$ (2)

$$\frac{\partial o_k}{\partial w_{kj}} = \frac{\partial o_k}{\partial net_{o_k}} \cdot \frac{\partial net_{o_k}}{\partial w_{kj}} = (1-o_k)o_k y_j = f'_{o_k} y_j$$ (3)，根据公式(1), (2)

梯度下降优化—前馈神经网络

- 又因为

$$\varepsilon_p = \frac{1}{2}\left(\frac{\sum_{k=1}^{K}(t_{k,p}-o_{k,p})^2}{K}\right), \quad 记E = \frac{1}{2}\sum_{k=1}^{K}(t_k-o_k)^2$$

因此有

$$\frac{\partial E}{\partial o_k} = \frac{\partial}{\partial o_k}\left(\frac{1}{2}\sum_{k=1}^{K}(t_k-o_k)^2\right) = -(t_k-o_k) \tag{4}$$

梯度下降优化—前馈神经网络

定义需要被反向传播的输出误差为$\delta_{o_k} = \frac{\partial E}{\partial net_{o_k}}$，则

$$\delta_{o_k} = \frac{\partial E}{\partial net_{o_k}} = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial net_{o_k}}$$

$$= -(t_k - o_k)(1 - o_k)o_k = -(t_k - o_k) f'_{o_k} \quad \textbf{(5)}$$

因此，从隐层到输出层权值的改变可计算为：

$$\Delta w_{kj} = \eta\left(-\frac{\partial E}{\partial w_{kj}}\right) = -\eta \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial w_{kj}} = -\eta \times \left(-(t_k - o_k)\right) \times f'_{o_k} y_j$$

$$= -\eta \delta_{o_k} y_j$$

(6)，根据公式(3),(4)推出

(7)，根据公式(5)推出

梯度下降优化—前馈神经网络

- 进一步，分析输入层到隐层的权值变化量：

$$\frac{\partial y_j}{\partial net_{y_j}} = \frac{\partial f_{y_j}}{\partial net_{y_j}} = (1-y_j)y_j = f'_{y_j} \tag{8}$$

$$\frac{\partial net_{y_j}}{\partial v_{ji}} = \frac{\partial}{\partial v_{ji}}\left(\sum_{i=1}^{I+1} v_{ji}z_i\right) = z_i \tag{9}$$

$$\frac{\partial y_j}{\partial v_{ji}} = \frac{\partial y_j}{\partial net_{y_j}} \cdot \frac{\partial net_{y_j}}{\partial v_{ji}} = (1-y_j)y_j z_i = f'_{y_j} z_i \tag{10}$$，根据公式(8),(9)

$$\frac{\partial net_{o_k}}{\partial y_j} = \frac{\partial}{\partial y_j}\left(\sum_{j=1}^{J+1} w_{kj}y_j\right) = w_{kj}, \tag{11}$$

梯度下降优化—前馈神经网络

$$\frac{\partial E}{\partial y_j}=\frac{\partial}{\partial y_j}\left(\frac{1}{2}\sum\nolimits_{k=1}^{K}(t_k-o_k)^2\right)=\sum_{k=1}^{K}\frac{\partial E}{\partial o_k}\cdot\frac{\partial o_k}{\partial net_{o_k}}\cdot\frac{\partial net_{o_k}}{\partial y_j} \tag{12}$$

$$=\sum_{k=1}^{K}\frac{\partial E}{\partial net_{o_k}}\cdot\frac{\partial net_{o_k}}{\partial y_j}=\sum_{k=1}^{K}\delta_{o_k}w_{kj} \tag{13}$$

(13)，根据公式(5)

- **将需要反向传播的隐层误差定义为：**

$$\delta_{y_j}=\frac{\partial E}{\partial net_{y_j}}=\frac{\partial E}{\partial y_j}\cdot\frac{\partial y_j}{\partial net_{y_j}}=\sum_{k=1}^{K}\delta_{o_k}w_{kj}f'_{y_j} \tag{14}$$

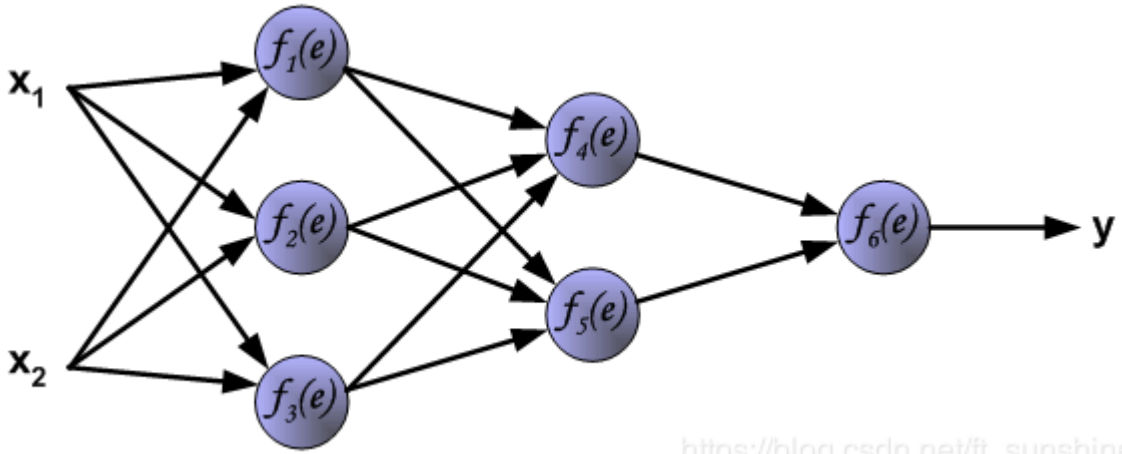
最终，输入层到隐层权值的变化量为：

$$\Delta v_{ji}=\eta\left(-\frac{\partial E}{\partial v_{ji}}\right)=-\eta\frac{\partial E}{\partial y_j}\cdot\frac{\partial y_j}{\partial v_{ji}}=-\eta\delta_{y_j}z_i \tag{15}$$

(15)，根据公式(10),(13),(14)

前向传播与反向传播

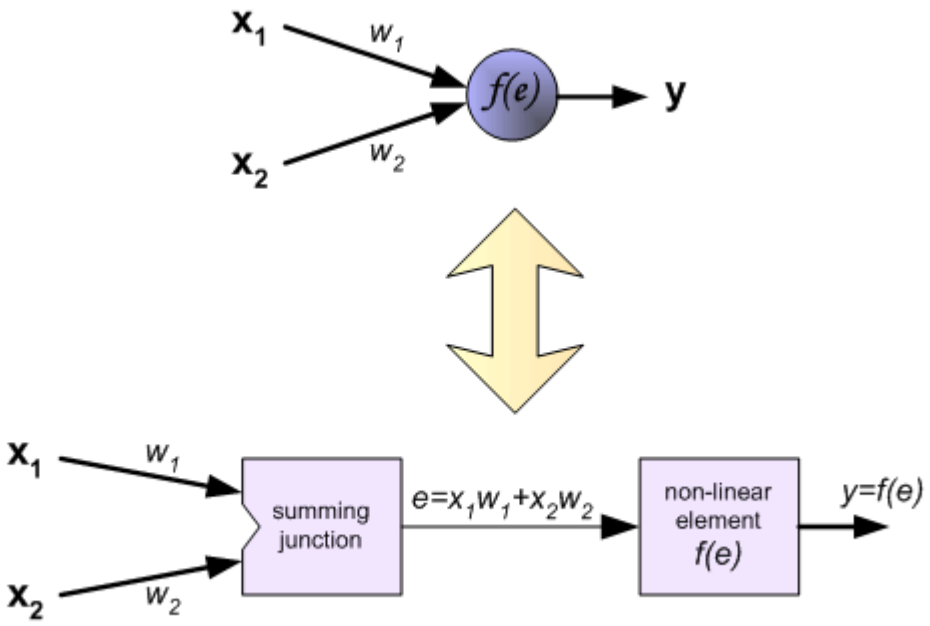
用一个简单的三层神经网络举例[1]



[1] 示例来自于网址：http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html, 文字来源于CSDN: https://blog.csdn.net/ft_sunshine/java/article/details/90221691

前向传播与反向传播

- 每个神经元由两部分组成
  - 第一部分（e）是输入值和权重系数乘积的和
  - 第二部分（f(e)）是一个激活函数（非线性函数）的输出， y=f(e)即为某个神经元的输出



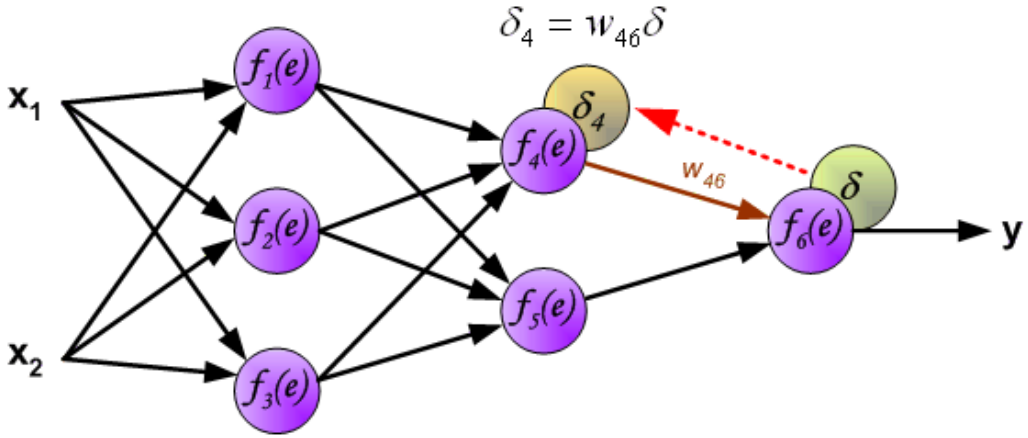
前向传播与反向传播

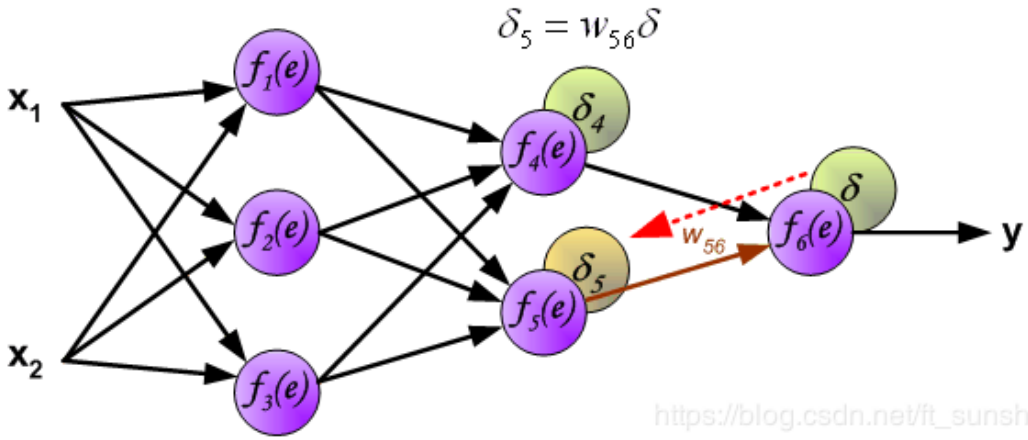
- 每个神经元由两部分组成
  - 第一部分（e）是输入值和权重系数乘积的和
  - 第二部分（f(e)）是一个激活函数（非线性函数）的输出， y=f(e)即为某个神经元的输出

- 基本概念：
  - 正向传播：求损失
  - 反向传播：回传误差（基于链式求导法则）
  - 神经网络每层中每个神经元都可以根据误差信号修正每层的权重

反向传播

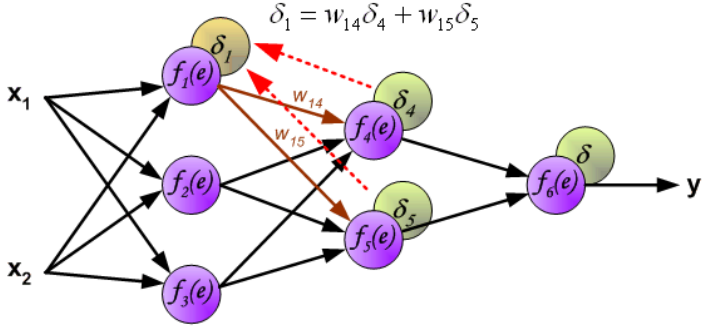
- 下面开始计算每个神经元的误差（δ）



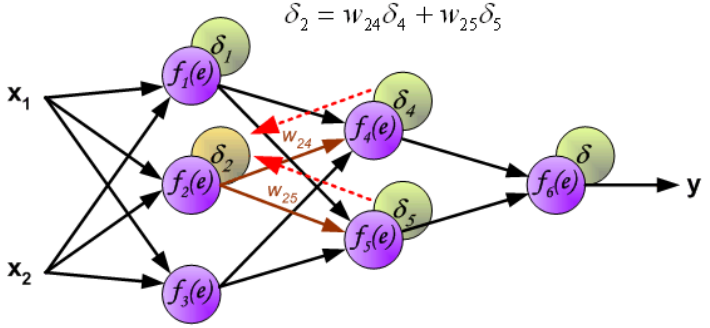


反向传播

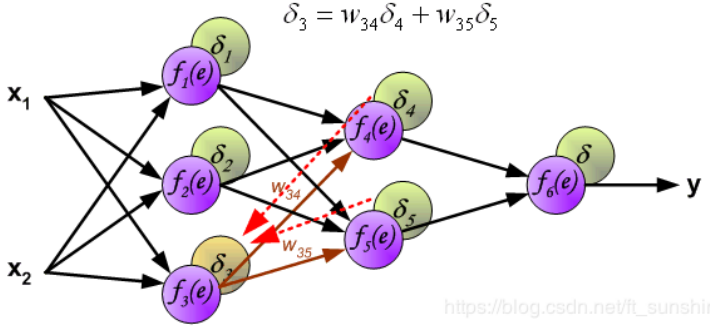
$$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$$



$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$

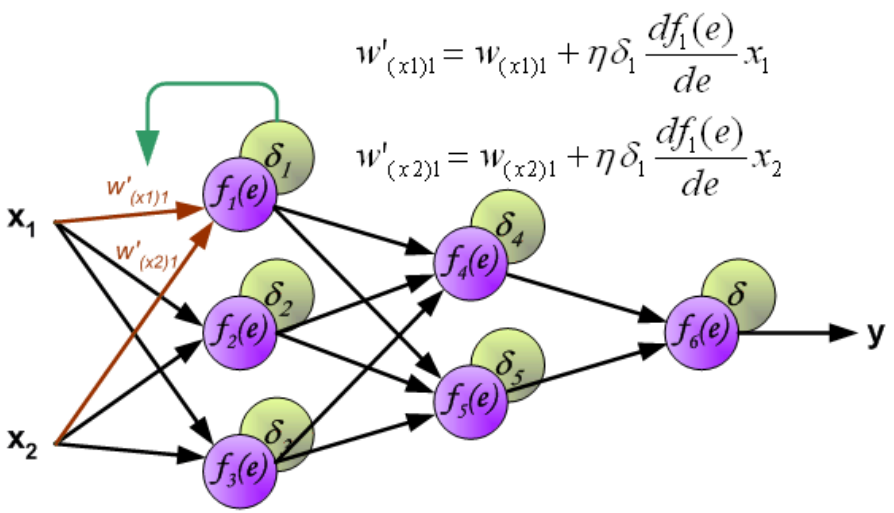


$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$



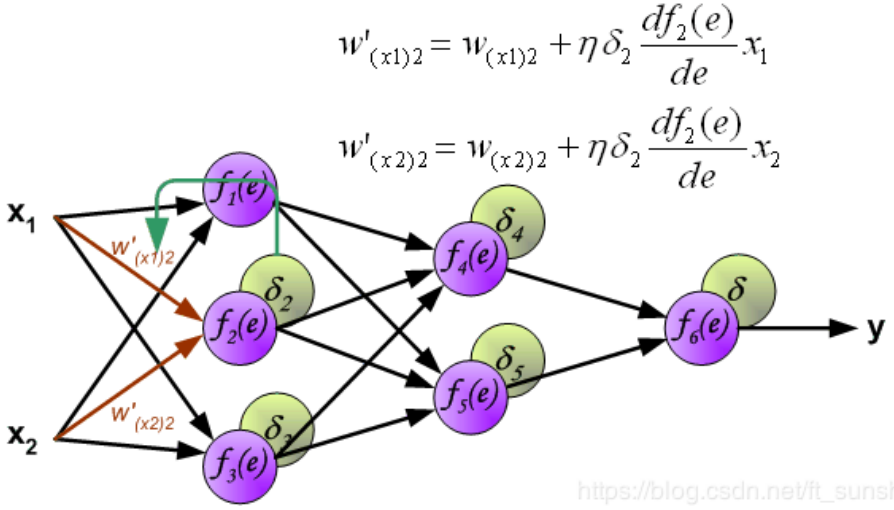
反向传播：链式求导

- 下面开始利用反向传播的误差，计算各个神经元（权重）的导数，开始反向传播修改权重



备注：示意图中的误差$\boldsymbol{\delta_i}$与前面推导过程公式（14）中误差（设为$\boldsymbol{\delta'_i}$）的定义不同。此处

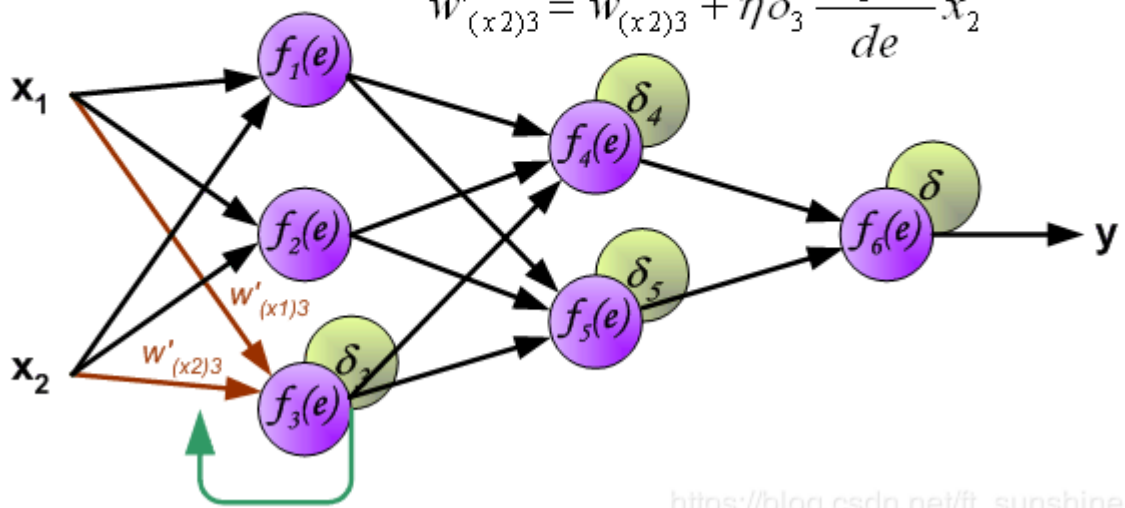
$$\boldsymbol{\delta'_i = \delta_i f'_i}$$



反向传播

$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$

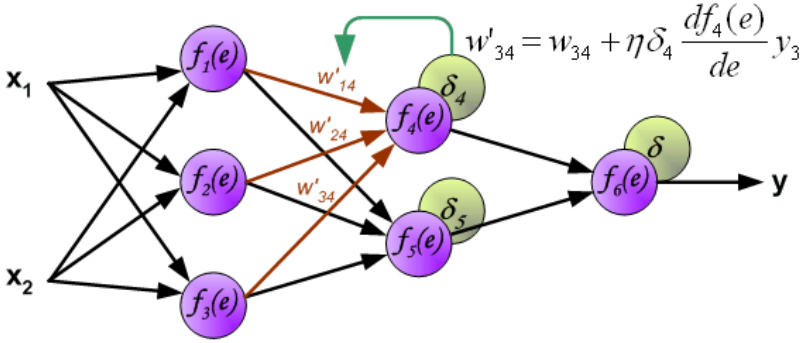


反向传播

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

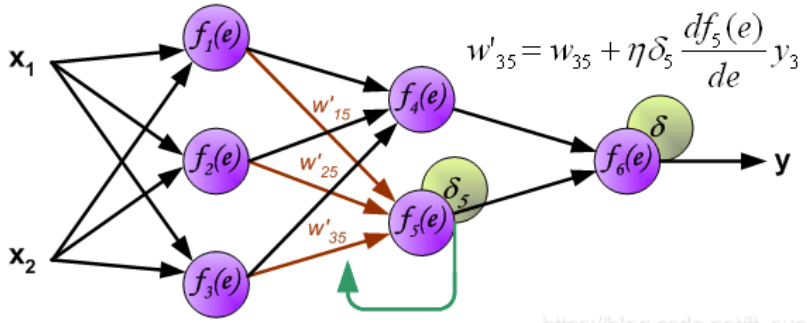
$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$



$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$$

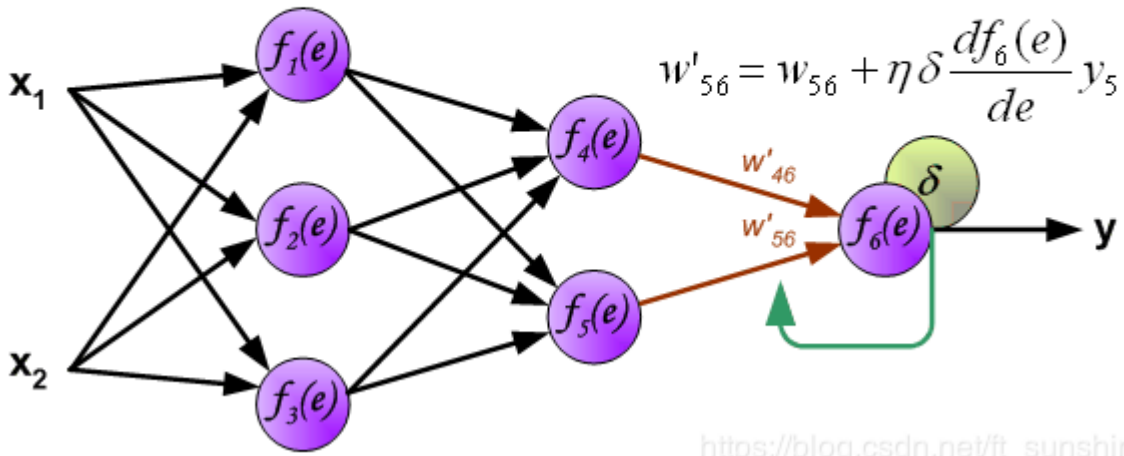
$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$



反向传播

$$w'_{46} = w_{46} + \eta\delta\frac{df_6(e)}{de}y_4$$

$$w'_{56} = w_{56} + \eta\delta\frac{df_6(e)}{de}y_5$$



•到此为止，整个网络的前向，反向传播和权重更新已经完成

反向传播算法-小结

- （1）正向传播阶段：利用输入和当前的权重设置，依次计算新的权值和每个神经元的输入、输出
- （2）反向传播阶段：利用正向传播得到的权值和每个神经元的输入、输出，按照从输出层到输入层的方向，将输出误差在整个网络中进行反向传播。

梯度下降优化—前馈神经网络

//随机梯度下降学习算法

初始化权值、η、α以及学习迭代次数，令t=0；

While 终止条件不满足 **do**

设ε_T=0；

for 每一个训练模式p **do**

在前馈阶段计算$y_{j,p}$和$o_{k,p}$（其中，j=1,…,J, k=1,…,K）

计算输出误差信号$\delta_{o_{k,p}}$和隐层误差信号$\delta_{y_{j,p}}$；

调整权值w_{kj}和v_{ji}（误差反向传播）；

$$\varepsilon_T += [\varepsilon_p = \sum_{k=1}^{K}(t_{k,p} - o_{k,p})^2];$$

end

t=t+1；

end

梯度下降优化—前馈神经网络

- 随机梯度下降学习算法的终止准则包括：
  ① 当超过学习迭代的最大次数时终止。
  ② 当训练集的均方误差MSE足够小时停止（或方根均方误差足够小）。

$$\varepsilon_T = \frac{\sum_{p=1}^{P_T}\sum_{k=1}^{K}(t_{k,p} - o_{k,p})^2}{P_T K}$$

③ 当观察到过拟合时终止，即当训练数据被记忆时终止。

过拟合的一个现象是：$\varepsilon_V > \bar{\varepsilon}_V + \delta_{\varepsilon_V}$，其中是$\bar{\varepsilon}_V$在先前学习迭代中的平均验证误差，$\delta_{\varepsilon_V}$是验证误差的标准差。

梯度下降优化—前馈神经网络

- 问题：

- 可以直接将梯度下降优化应用于其他类型的神经网络。
  - 具体的权值更新公式的推导？

  - 如果有输入层到输出层的直接权值，如何更新

梯度下降优化—乘积单元神经网络

- **推导示例：**仅在隐层中使用乘积单元的学习等式，并假定使用梯度下降优化和线性激活函数。
  - 因为只有**输入层到隐层**的等式改变，因此仅给出这部分权值更新等式的推导。
- 权值v_{ji}的变化量 Δv_{ji}为：

$$\Delta v_{ji} = \eta\left(-\frac{\partial E}{\partial v_{ji}}\right) = -\eta\frac{\partial E}{\partial net_{y_{j,p}}}\frac{\partial net_{y_{j,p}}}{\partial v_{ji}} = -\eta\delta_{y_{j,p}}\frac{\partial net_{y_{j,p}}}{\partial v_{ji}}$$

其中，$\delta_{y_{j,p}}$是误差信号，其计算方式与求和单元相同。

且
$$\frac{\partial net_{y_{j,p}}}{\partial v_{ji}} = \frac{\partial}{\partial v_{ji}}\left(\prod_{i=1}^{I} z_{i,p}^{v_{ji}}\right) = \frac{\partial}{\partial v_{ji}}\left(e^{\rho_{j,p}}\cos(\pi\phi_{j,p})\right)$$

$$= e^{\rho_{j,p}}\left[\ln|z_{i,p}|\cos(\pi\phi_{j,p}) - \Gamma_i\pi\sin(\pi\phi_{j,p})\right]$$

梯度下降优化—乘积单元神经网络

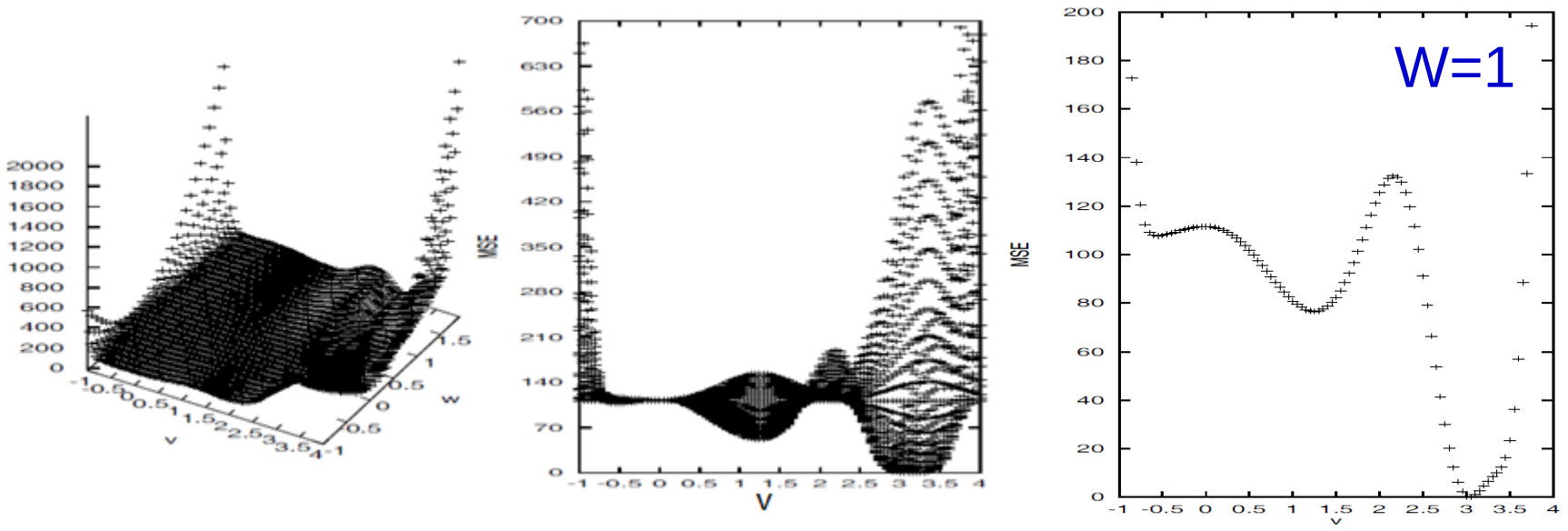
- **Durbin和Rumelhart的工作表明：单个乘积单元的信息容量（ information capacity）近似为3I，而单个求和单元的信息容量为2I，其中I为该单元输入的个数。**
  - **信息容量：能够学习的随机布尔模式数。**

- **更大的容量意味着，与求和单元相比，用乘积单元逼近相同的函数需要更少的处理单元。**

Function	SUs	PUs
$f(z) = z^2$	2	1
$f(z) = z^6$	3	1
$f(z) = z^2 + z^5$	3	2
$f(z_1, z_2) = z_1^3 z_2^7 - 0.5 z_1^6$	8	2

梯度下降优化—乘积单元神经网络

- 以函数$f(z)=z^3$为例。
  - 仅需一个乘积单元，即1-1-1结构的神经网络，输入层到隐层的权值为3，隐层到输出层的权值为1。



需用全局随机优化算法以便能在搜索空间中更广阔的区域进行搜索，而不是过分依赖梯度信息。例如，模拟退火，遗传算法，粒子群优化，LeapFrog等。

梯度下降优化—乘积单元神经网络

- 乘积单元神经网络：能够提供更小的网络结构。
  - 缺点：增加了局部最小值的数量以及深的峡谷。
  - 因此，乘积单元的搜索空间极其复杂。

- 但是，梯度下降在搜索空间相对平滑时效果较好。

- Leerink等人的工作表明了6位奇偶问题不能够使用最基本的梯度下降乘积单元来训练。两个的原因导致失败：
  - 权值初始化
  - 局部最小值的存在

本章内容

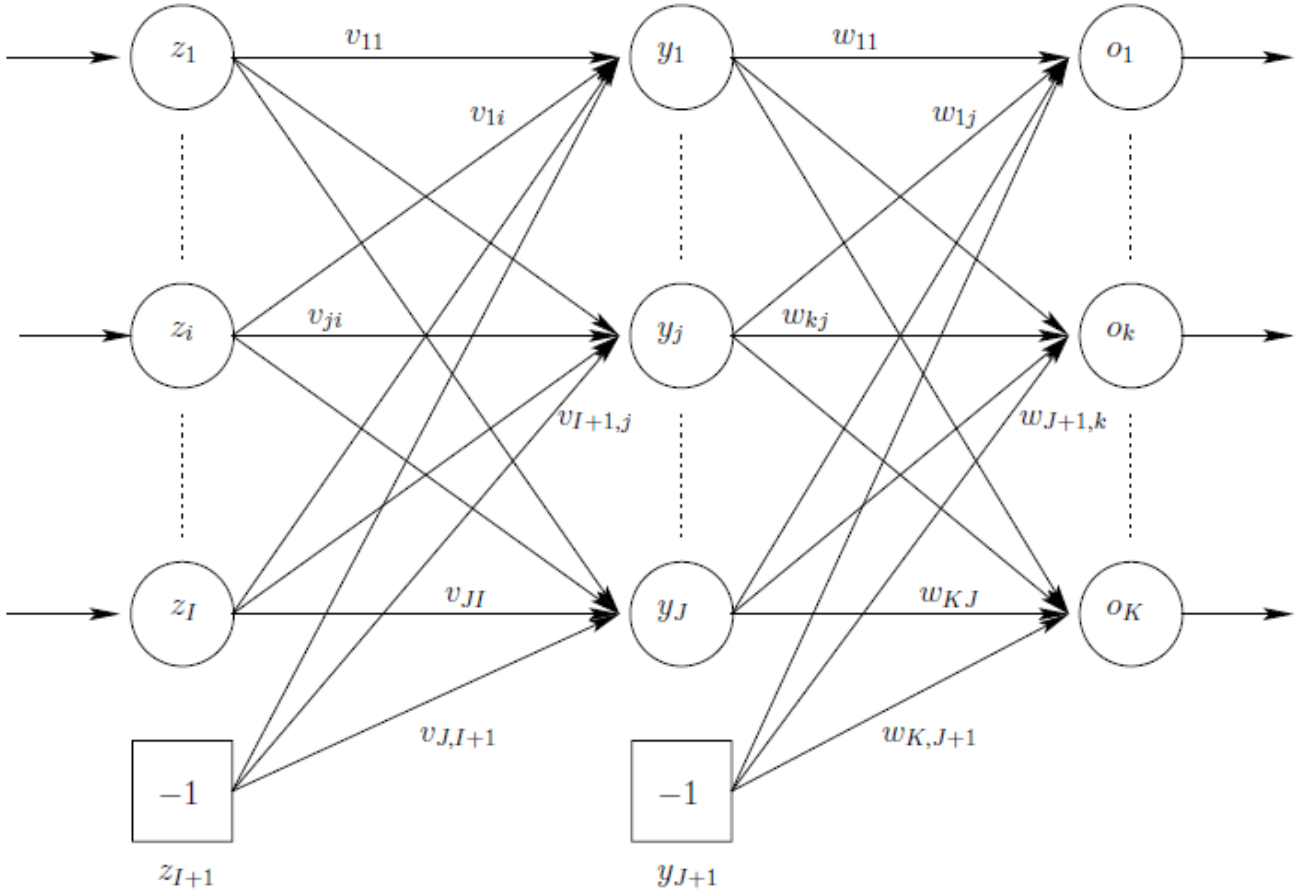
- 监督学习规则
- 神经网络的类型
- 隐层单元的功能
- 集成神经网络

神经网络的类型

1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

前馈神经网络

- 标准的前馈神经网络
  - 一个输入层、一个隐层、一个输出层



前馈神经网络

- 有些有关神经网络的文献，不将输入层记为一层。

- 一个前馈神经网络可以有多个隐层。
  - 已经证明了使用单调递增可微函数的单隐层前馈神经网络能够逼近任意的连续函数，只要隐层具有足够多的隐层神经元。

- 并不需要所有的神经元使用相同的激活函数。每个输入单元也可以实现一个激活函数。

- 一个前馈神经网络也可以在输入层和输出层之间建立直接（线性）连接）。

神经网络的类型

1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

乘积单元神经网络

- 乘积单元神经网络的神经元的网络输入是输入信号的加权乘积而不是加权和。

- 已有多种不同类型的乘积单元神经网络：
  ① 每个输入单元都连接到求和单元和一组乘积单元
  ② 乘积单元层和求和单元层相交替
  ③ ……

乘积单元神经网络

- 本节：隐层仅包括乘积单元，不含求和单元；输出层仅含有求和单元，并且假定网络中的所有神经元均使用线性激活函数。
  - 因此，对于每一个隐层单元y_j，其网络输入为：

不含偏置：

$$net_{y_{j,p}} = \prod_{i=1}^{I} z_{i,p}^{v_{ji}}$$

$$= \prod_{i=1}^{I} e^{v_{ji}\ln(z_{i,p})}$$

$$= e^{\sum_i v_{ji}\ln(z_{i,p})}$$

含失真度（distortion），对所有模式均有$z_{I+1,p}=-1$，$v_{j,I+1}$代表失真度：

$$net_{y_{j,p}} = \prod_{i=1}^{I+1} z_{i,p}^{v_{ji}}$$

引入失真度的目的是在训练中动态调整激活函数以使其更好地逼近训练数据所代表的真实函数。

神经网络的类型

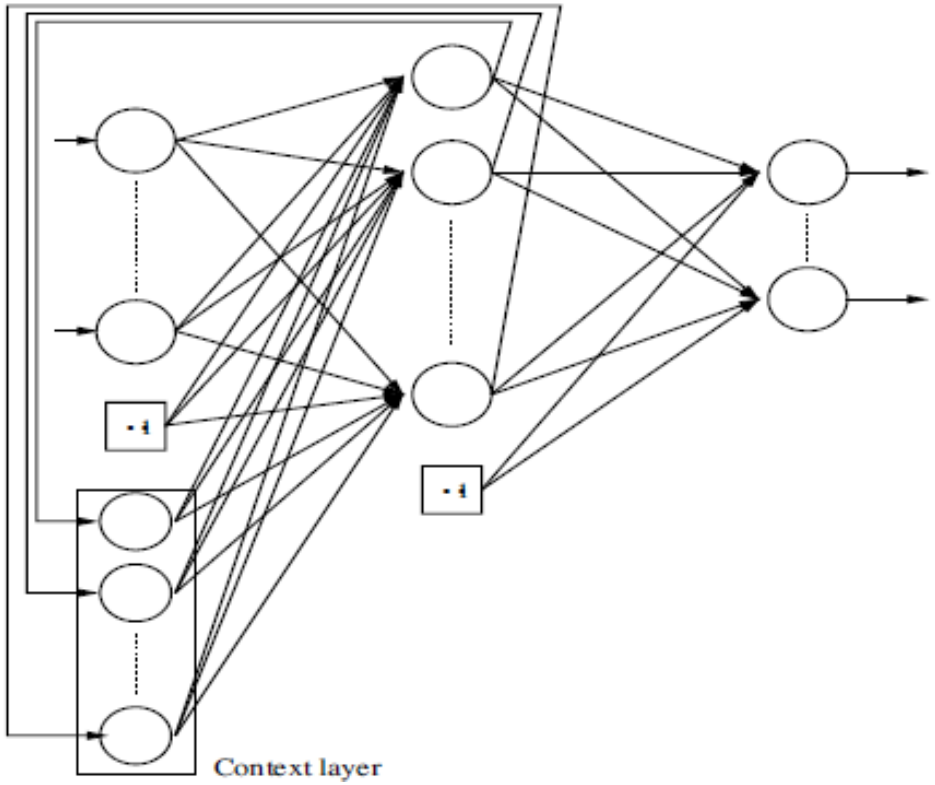
1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

简单反馈神经网络

- 简单反馈神经网络具有反馈连接。

- 已经建立了几个不同类型的简单反馈神经网络。其中，**Elman**和**Jordan**简单反馈神经网络是前馈神经网络的简单扩展。

Elman简单反馈神经网络



- 通过复制隐层得到环境层，而环境层作为输入层的一个扩展，给隐层提供表示先前网络状态的信号。
  - 环境层的目的是存储隐层的先前状态。

Elman简单反馈神经网络

- 环境单元z_{I+2}，…，z_{I+1+J}和所有隐层单元均为全连接。每一个隐层单元y_j（j=1,…,J)到其对应的环境单元z_{I+1+j}的连接权值为1。
  - 每一个输出单元的输出值为：

$$o_{k,p} = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1+J} v_{ji} z_{i,p}\right)\right)$$

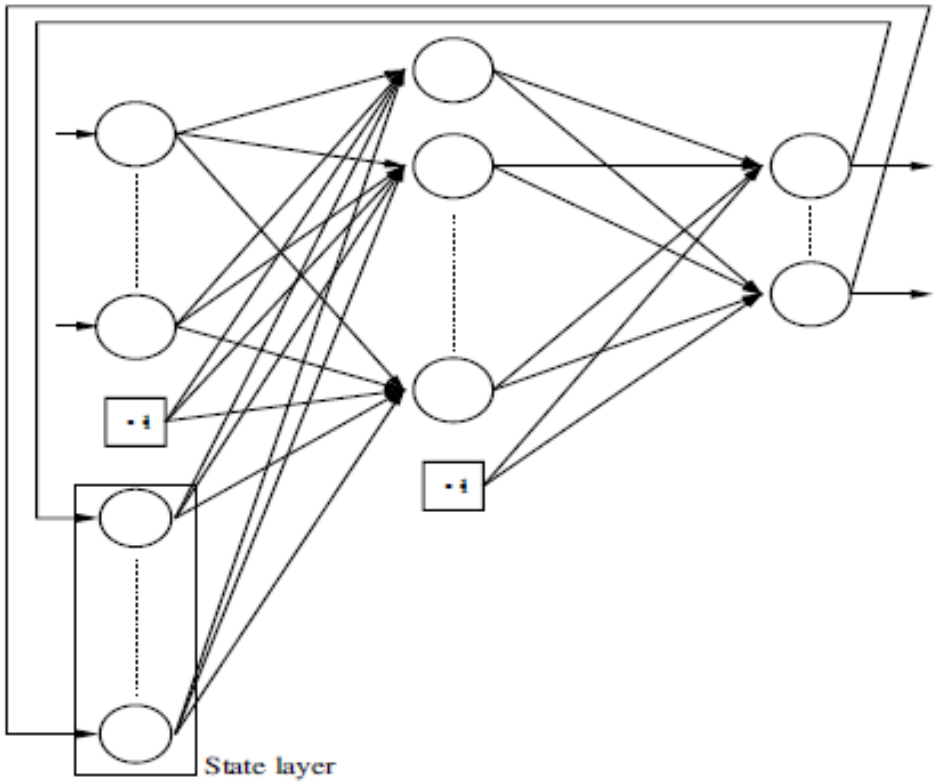
$$\text{其中}(z_{I+2,p},\cdots,z_{I+1+J,p}) = (y_{1,p}(t-1),\cdots,y_{J,p}(t-1))$$

每一个隐层单元y_j（j=1,…,J)到其对应的环境单元z_{I+1+j}的连接权值也可以不为1。

- 先前状态的影响被加权。确定这些权值为训练步骤增加了额外的复杂度。

Jordan简单反馈神经网络

- 对输出层进行复制。
  - 复制的输出层称作状态层。



Jordan简单反馈神经网络

- 输入层扩展为：

$$(\underbrace{z_{1,p},\cdots,z_{I+1,p}}_{\text{实际输入}},\underbrace{z_{I+2,p},\cdots,z_{I+1+K,p}}_{\text{状态单元}})$$

对于每一个输出单元，其输出为：

$$o_{k,p}=f_{o_k}\left(\sum_{j=1}^{J+1}w_{kj}\,f_{y_j}\left(\sum_{i=1}^{I+1+K}v_{ji}z_{i,p}\right)\right)$$

$$\text{其中}(z_{I+2,p},\cdots,z_{I+1+K,p})=(o_{1,p}(t-1),\cdots,o_{K,p}(t-1))$$

神经网络的类型

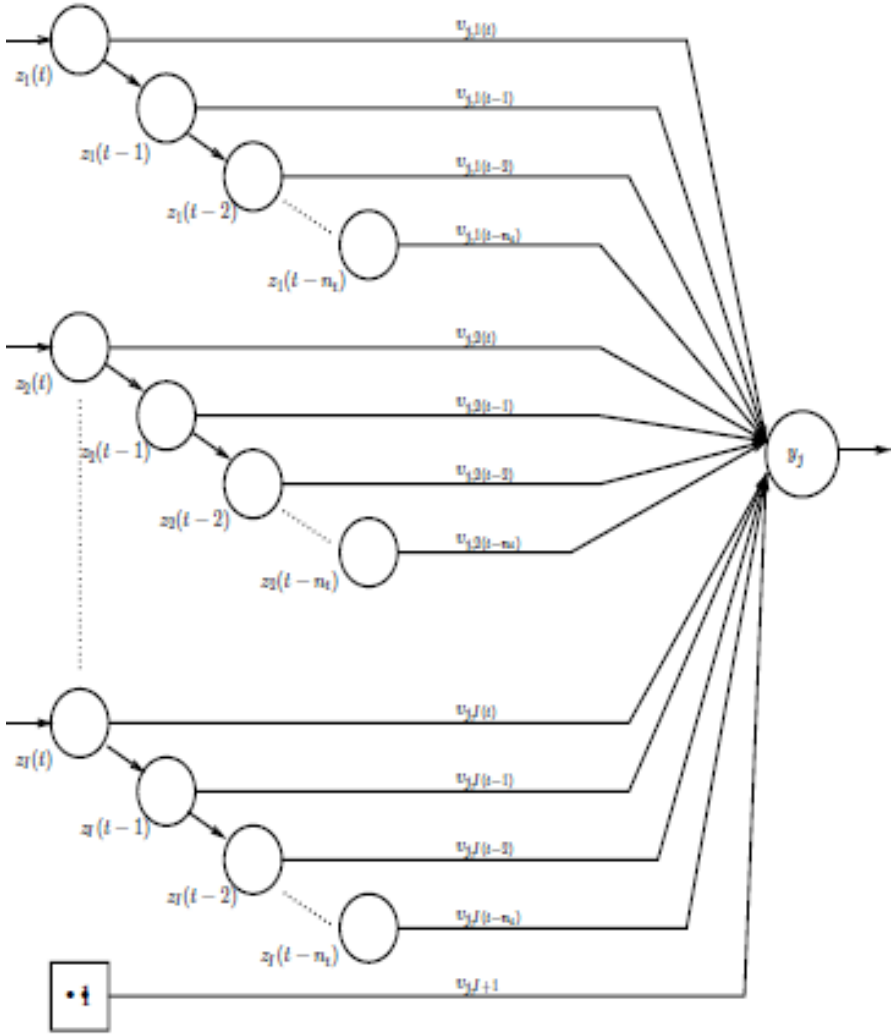
1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

时延神经网络

- 时延神经网络是一个将其输入模式连续地延时的时域神经网络。

单个时延神经元
（每个输入都被n_t时延）

用于构造一个完整的前馈时延神经网络。



时延神经网络

初始时，仅$z_{i,p}(0)$有值，其他所有的$z_{i,p}(t-t')$（$i=1,\cdots,\mathrm{I};t'=1,\cdots,\mathrm{n_t}$）均为0。$\mathrm{n_t}$为时延模式的个数。

在第一个模式出现后并在给出第二个模式前，有$z_{i,p}(t-1)=z_{i,p}(t)$。

在t'个模式出现后并在给出$t'+1$个模式前，对于所有的$t=1,\cdots,t'$，有$z_{i,p}(t-t')=z_{i,p}(t-t'+1)$。

- **如此类推，共有n_t个模式影响权值的更新，从而使得时域特征可以 引导所学习到的函数曲线。**

时延神经网络

- 时延神经网络的输出为：

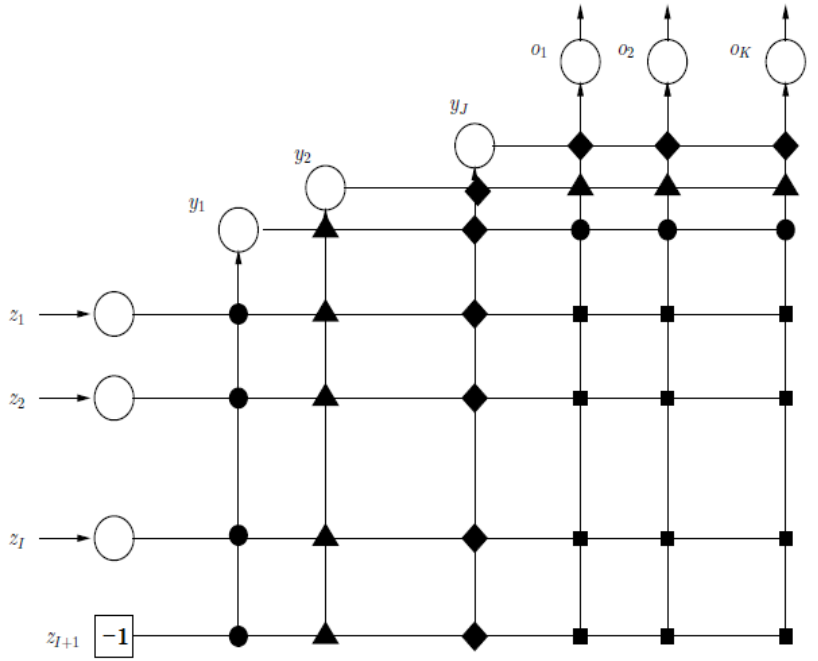
$$o_{k,p} = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I}\sum_{t=0}^{n_t} v_{ji}(t) z_{i,p}(t) + z_{I+1} v_{j,I+1}\right)\right)$$

神经网络的类型

1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

级联神经网络

- 级联神经网络是一个多层的前馈神经网络，并且：
  - 所有的输入单元都对所有的隐层单元和输出单元建立了直接连接。
  - 隐层单元也是级联的，即每一个隐层单元的输出均作为后续所有隐层单元和输出单元的一个输入。



级联神经网络

- 级联神经网络的输出为：

$$o_{k,p} = f_{o_k}\left(\sum_{i=1}^{I+1} u_{ki} z_{i,p} + \sum_{j=1}^{J} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1} v_{ji} z_{i,p} + \sum_{l=1}^{j-1} s_{jl} y_l\right)\right)$$

其中，u_{ki}表示输出单元k和输入单元i之间的权重，s_{jl}是隐层单元j和l之间的权重，y_l是隐层单元l的输出。

本章内容

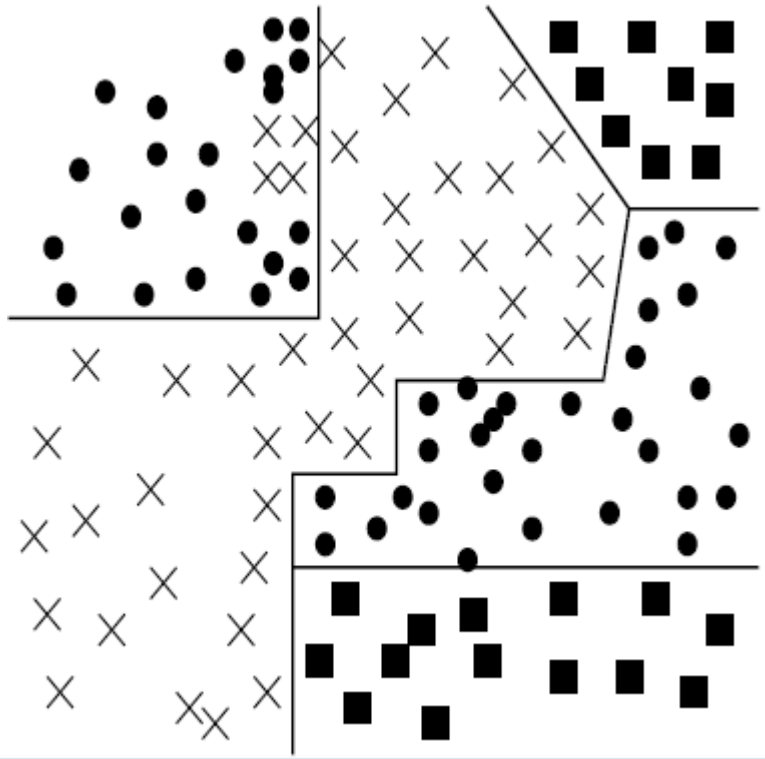
- 监督学习规则
- 神经网络的类型
- 隐层单元的功能
- 集成神经网络

隐层单元的功能

- **分类问题：**假定是二维问题，使用单隐层标准前馈神经网络，隐层单元的任务是形成决策边界来分隔不同的类。

例如，右图所示的三类问题

- 10个边界
- 10个隐层单元
- 左上角有分类误差，需增加3个隐层单元
- 隐层单元减少，误差增加



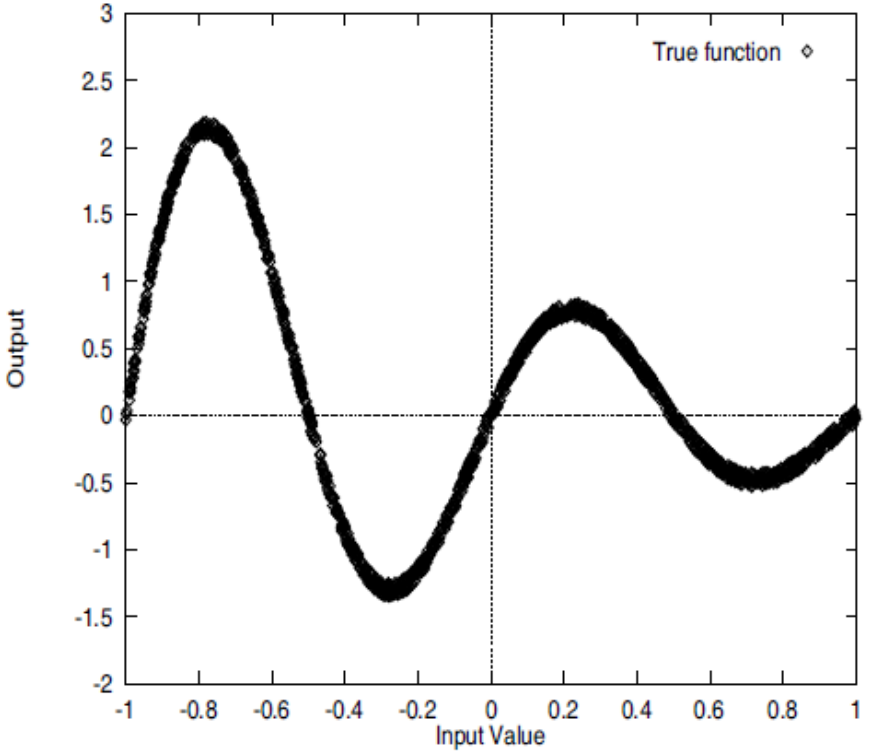
如何能在不使用关于输入空间的任何先验知识的前提下确定隐层单元的个数？

隐层单元的功能

- **函数逼近：**假定是一维函数，使用单隐层标准前馈神经网络。

右图所示的一维函数，需要5个使用Sigmoid激活函数的隐层单元来学习该函数。对于目标函数的每一个拐点，需要一个Sigmoid函数进行拟合。隐层隐层单元的个数是拐点的个数加1.

采用线性激活函数的隐层单元执行同样的任务。为了达到与使用Sigmoid函数一样的正确率，需要更多的线性激活函数来学习该函数。



本章内容

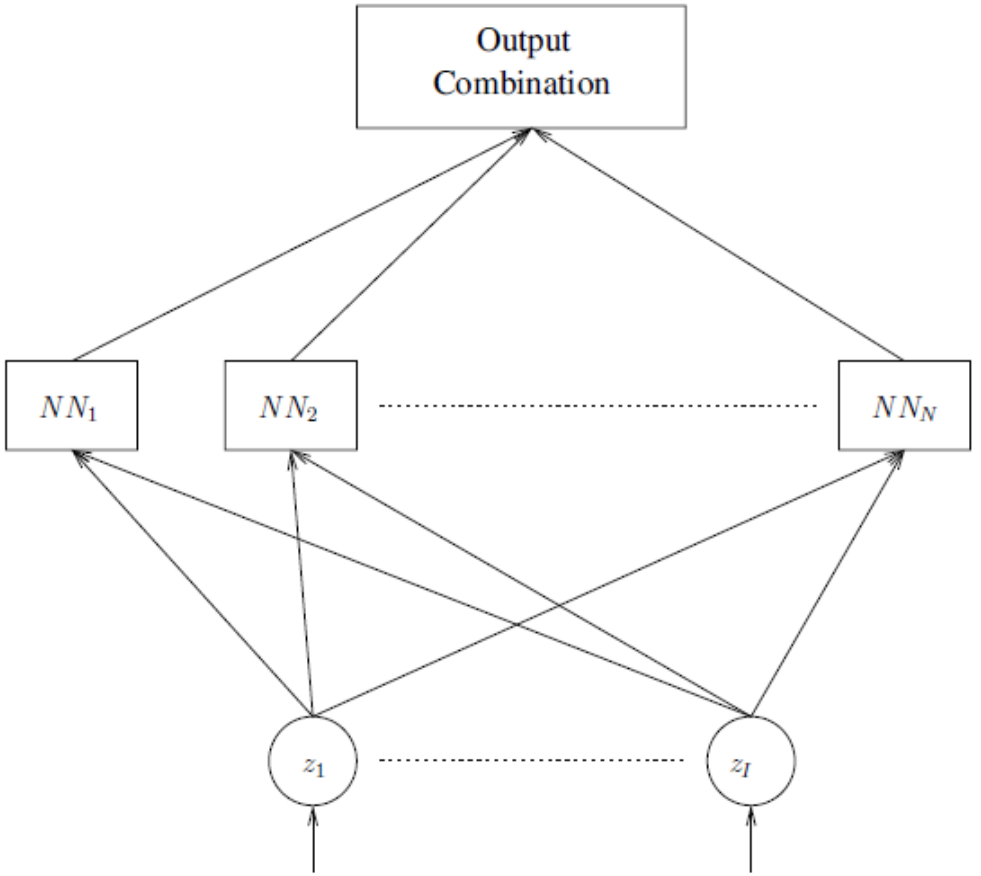
- 监督学习规则
- 神经网络的类型
- 隐层单元的功能
- 集成神经网络

集成神经网络

- 神经网络的训练通过随机选择的初始权值开始。每一次在在同一数据集上重新训练一个神经网络时，都可能得到一个不同的结果。

- 神经网络的这一问题推动了集成神经网络的发展。
  - 集成神经网络的目标是通过一些不同类型网络的组合来优化结果，这些神经网络将在同样的任务上进行训练。

集成神经网络



- 仅当集成成员之间存在不一致，或者成员在搜索空间的不同部分产生误差时，集成神经网络才有意义。

集成神经网络

- 集成网络的最终结果可通过不同的方式计算。

- 例如：
  - 在集成网络中选择给出最佳泛化性能的神经网络。
  - 对集成网络所有成员的输出求平均值。
  - 对集成网络中所有神经网络的输出进行线性组合。在这种情况下，可每一个神经网络分配一个权值作为其可靠性的指示。

集成神经网络

- 以彼此独立的方式训练个体（基本的 神经网络）
  - 例如，每个神经网络使用不同的训练数据集。
  - 典型的方法是**Bagging**（一种**Bootstrap**集成方法）。

- **Bootstrap**集成方法是通过在原始训练集的一个随机分布上训练每一个网络成员，创建起集成网络的个体。如果原始训练集包含P_T个模式，则对于每一个集成神经网络的成员，其由P_T个模式构成的数据集是从原始训练集随机采样的（有可能重复）。

集成神经网络

- 协作式集成策略
  - 不同类型的神经网络在训练阶段交换它们的经验和知识。
  - 例如，**Boosting**策略。

- **Boosting**策略：按次序训练参与集成的神经网络个体。已经训练好的成员将模式分为简单模式和困难模式，未训练的新成员更多地关注于先前训练好的成员所确认的困难样本。

小结

- 神经网络的类型
  - 熟悉典型的神经网络。
- 监督学习规则
  - 了解简单学习问题，熟悉其基本学习方法。
- 隐层单元的功能
  - 对隐层单元的功能有所了解。
- 集成神经网络
  - 了解基本思想。

作业

1. 对于一个在输入层和输出层之间使用直接连接的前馈神经网络，给出$o_{k,p}$的表达式。

2. 假定使用梯度下降作为优化算法，推导Elman简单反馈神经网络的学习等式。