第6.1节 人工神经网络
Artificial Neural Network（ANN）

中国科学院大学 叶齐祥

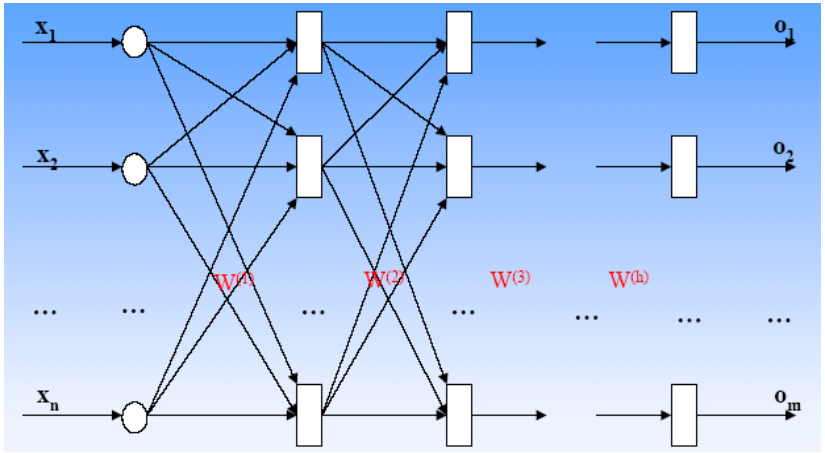
qxye@ucas.ac.cn

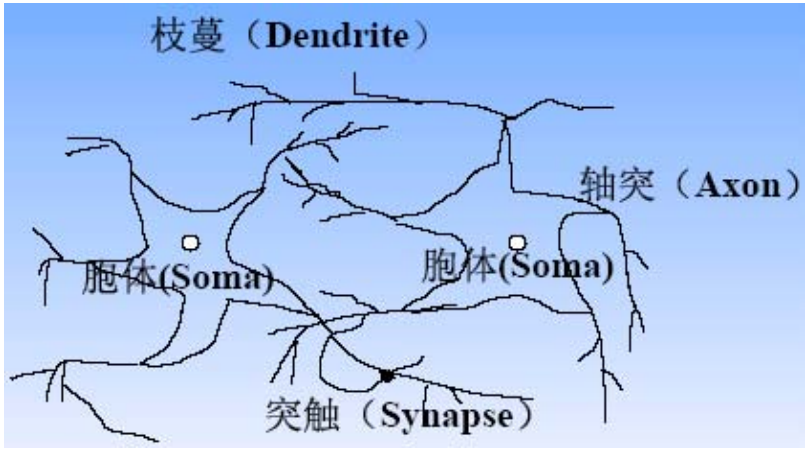
提纲

- 概述
- 单层神经网络
- 多层神经网络
- 设计ANN时需要考虑的问题

概述：仿生物神经系统

- 生物的学习系统是由相互连接的神经元组成的复杂网络。
- 人脑的构成，大约有10^{11}个神经元，平均每一个与其他10^4个相连
- 神经元的活性通常被通向其他神经元的连接激活或抑制
- 最快的神经元转换时间比计算机慢很多，然而人脑能够以惊人的速度做出复杂度惊人的决策
- 很多人推测，生物神经系统的信息处理能力一定得益于对分布在大量神经元上的信息表示的高度并行处理
- ANN受到生物学的启发，由一系列简单的单元相互密集连接构成的，其中每一个单元有一定数量的实值输入，并产生单一的实数值输出





概述：发展

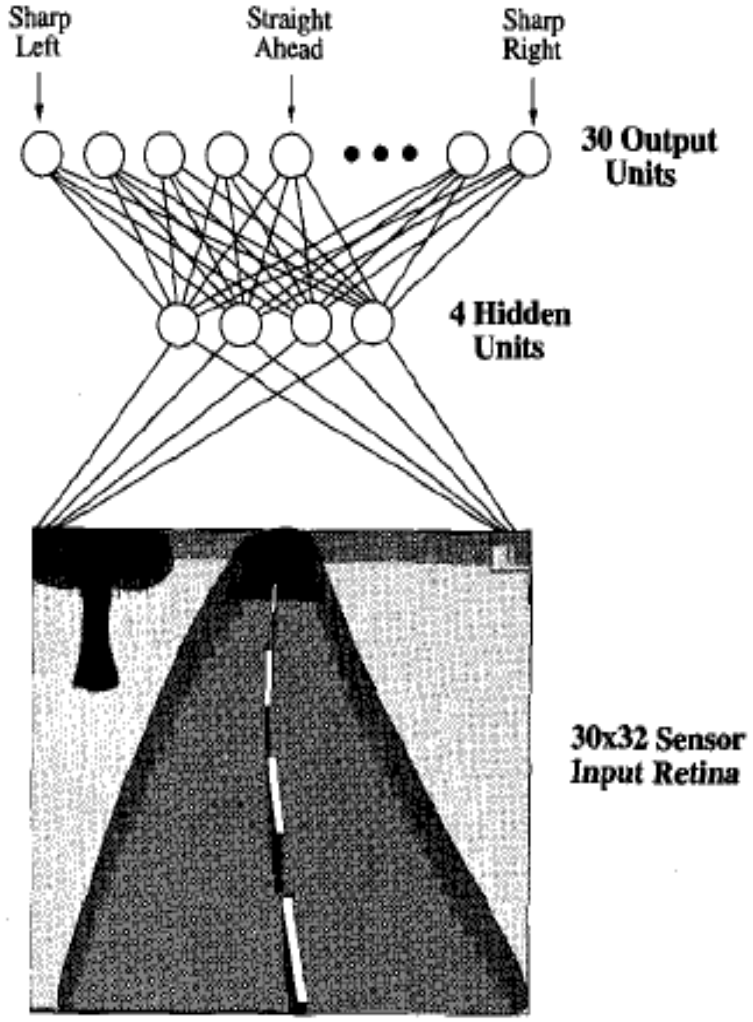
- 第一次热潮(40-60年代末)
  - 1943年,美国心理学家W.McCulloch和数学家W.Pitts在提出了一个简单的神经元模型，即MP模型。1958年，F.Rosenblatt等研制出了感知机(Perceptron）。
- 低潮(70-80年代初):
- 第二次热潮
  - 1982年，美国物理学家J.J. Hopfield提出Hopfield模型，它是一个互联的非线性动力学网络他解决问题的方法是一种反复运算的动态过程,这是符号逻辑处理方法所不具备的性质。1987年首届国际ANN大会在圣地亚哥召开，国际ANN联合会成立，创办了多种ANN国际刊物。1990年12月，北京召开首届学术会议。
- 低潮（1990-2005年）
- 第三次热潮（2006~现在）：
  - 深度神经网络与大规模机器学习算法

概述：应用

- 人工神经网络以其具有自学习、自组织、较好的容错性和优良的非线性逼近能力，受到众多领域学者的关注
- 在实际应用中，80%～90%的人工神经网络模型是采用误差反传算法或其变化形式的网络模型（简称BP网络）
- 早期的ANN应用于模式识别、分类、信号处理或数据挖掘
- 大规模深度神经网的应用扩展到了互联网搜索、智能识别、人机交互等领域

概述：应用

- ALVINN系统(Pomerleau1993)
  - **功能**：使用学习到的ANN以正常的速度在高速公路上驾驶汽车
  - ANN的输入是一个30x32像素的网格，输出是车辆行进的方向
  - 每个输出单元对应一个特定的驾驶方向，这些单元的输出决定哪一个方向是车辆的行驶方向



概述：适合NN处理的问题

- 含有噪声的复杂传感器数据
  - 例如来自摄像机和麦克风的数据
- 无法理解学到的目标函数
- 目标函数的输出是离散值、实数值或者由若干实数属性或离散属性组成的向量
- 训练数据可能包含较多的错误
- 允许长时间训练
- 需要快速的预测

提纲

- 概述
- 单层神经网络
- 多层神经网络
- 设计ANN时需要考虑的问题

单层神经网络：感知器

- 感知器以一个实数值向量作为输入，计算这些输入的线性组合，如果结果大于某个阈值，就输出1，否则输出-1

$$o(x_1,\ldots,x_K)=\begin{cases}1 & w_0+w_1x_1+\ldots+w_Kx_K>0\\ -1 & otherwise\end{cases}$$

其中每个w_i是一个实数，或叫做权值，用来决定输入x_i对感知器输出的贡献率。特别地，$-w_0$是阈值。

单层神经网络：感知器

- 两种简化形式，附加一个常量输入$x_0=1$，不等式写成

$$\sum_{k=0}^{K} w_k \cdot x_k > 0$$

或写成向量形式

$$\vec{w} \cdot \vec{x} > 0$$

- 简短起见，把感知器函数写为

$$o(\vec{x}) = \mathrm{sgn}(\vec{w} \cdot \vec{x})$$

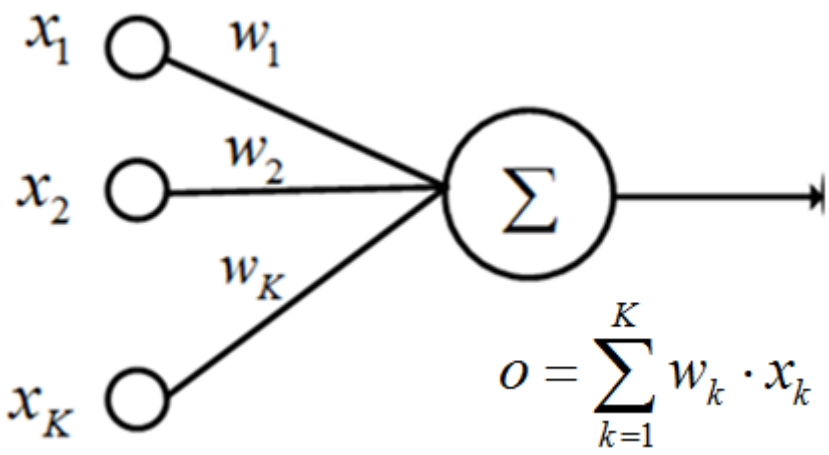
其中，

$$\mathrm{sgn}(y) = \begin{cases} 1 & y > 0 \\ -1 & otherwise \end{cases}$$

单层神经网络：感知器

- 感知器学习的过程是确定权$w_0,\ldots,w_K$的过程。感知器学习要考虑的候选假设空间H就是所有可能的实数值权向量的集合

$$H = \{\vec{w} \mid \vec{w} \in R^{K+1}\}$$



单层神经网络：感知器

- 分类问题
  - 可以把感知器看作是*K*维空间中的超平面形式的决策面
  - 对于超平面一侧的实例，感知器输出1，对于另一侧的实例，输出-1
  - 这个决策超平面方程是 $\vec{w} \cdot \vec{x} = 0$
  - 其中，可以被此平面分割的样本集合，是线性可分的

感知器训练

- 算法过程
  - 从随机的权值开始
  - 循环应用感知器到每个训练样本，对误分的样本修改感知器权值
  - 重复这个过程，直到感知器正确分类所有的训练样例
- 感知器训练法则

$$w_k \leftarrow w_k + \Delta w_k$$

其中 $$\Delta w_k = \eta \cdot (y - o) x_k$$

感知器训练

- 为什么这个算法会收敛到正确的权值呢？
  - 工程实际中验证
  - 证明（Minskey & Papert 1969）
    - 如果训练样例线性可分，并且使用了充分小的η
    - 否则，不能保证

梯度下降和delta法则

- delta法则的目的
  - 在线性不可分的训练样本上，收敛到目标概念的最佳近似
- delta法则的思想
  - 使用梯度下降搜索可能的权向量的假设空间，以找到最佳拟合训练样例的权向量
- delta法则为反向传播算法提供了基础
  - 而反向传播算法能够学习多个单元的互连网络
- **对于包含多种不同类型的连续参数化假设的假设空间，梯度下降是遍历这样的空间的所有算法的基础**

梯度下降和delta法则

- 把delta训练法则理解为训练一个无阈值的感知器

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

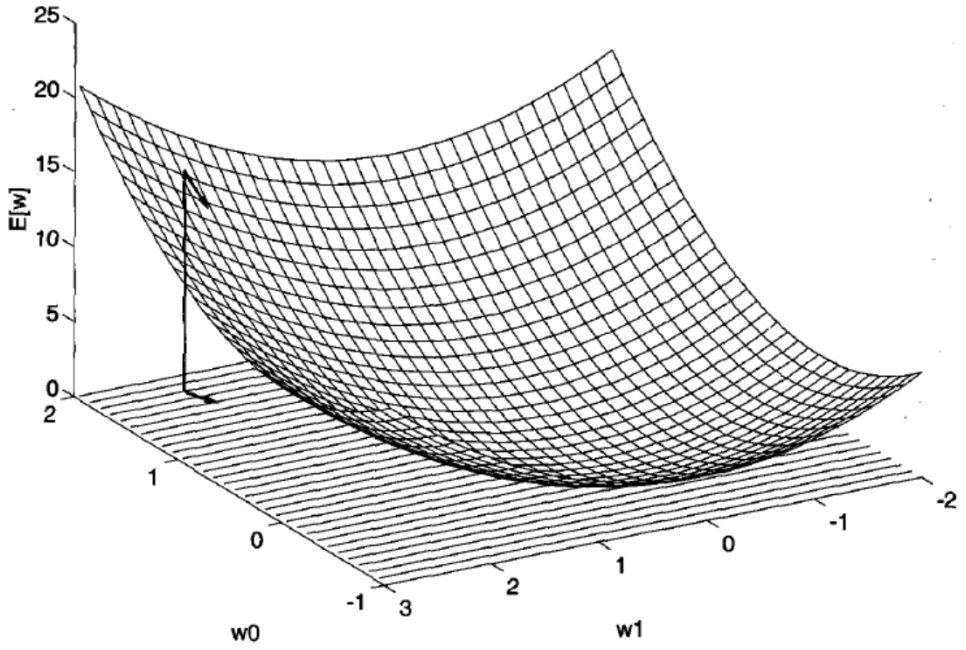
- 定义一个度量标准来衡量假设相对于训练样例的训练误差

$$E(\vec{w}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - o^{(i)})^2$$

- 在一定条件下，使E最小化的假设就是H中最可能的假设

假设空间内的误差函数

- 根据E的定义，误差曲面是抛物面，存在一个全局最小值
- 梯度下降搜索从一个任意的初始权向量开始，然后沿误差曲面最陡峭下降的方向，以很小的步伐反复修改这个向量，直到得到全局的最小误差点



梯度下降算法的推导

- 如何发现沿误差曲面最陡峭下降的方向？
  - 通过计算E相对向量 $\vec{w}$的每个分量的导数，这个向量导数被称为E对于 $\vec{w}$ 的梯度，记作 $\nabla E(\vec{w})$
  - 当梯度被解释为权空间的一个向量时，它确定了使E最陡峭上升的方向，所以这个向量的反方向给出了最陡峭下降的方向
- 梯度算法的迭代

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

其中，

$$\Delta\vec{w} = -\eta\nabla E(\vec{w})$$

梯度下降法则的推导

$$E(\vec{w}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - o^{(i)})^2$$

- 需要在每一步都计算这个梯度

$$\frac{\partial E}{\partial w} = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_K}\right]$$

- 梯度下降权值更新法则

$$\Delta w_k = \eta \frac{\partial E}{\partial w_k}$$

$$\frac{\partial E}{\partial w_k} = \sum_{i=1}^{N}(y^{(i)} - o^{(i)})(-x_k^{(i)})$$

梯度下降训练算法

- 梯度下降算法(gradient-descent)

  输入： 训练样本集合：$X=\{X^{(1)},...,X^{(N)}\,|\,Y^{(1)},...,Y^{(N)}\}$ 学习速率：η

  输出： 权值向量：w

  - **初始化** w 的各个分量 为0-1.0间的随机数
  - 循环更新权值向量：

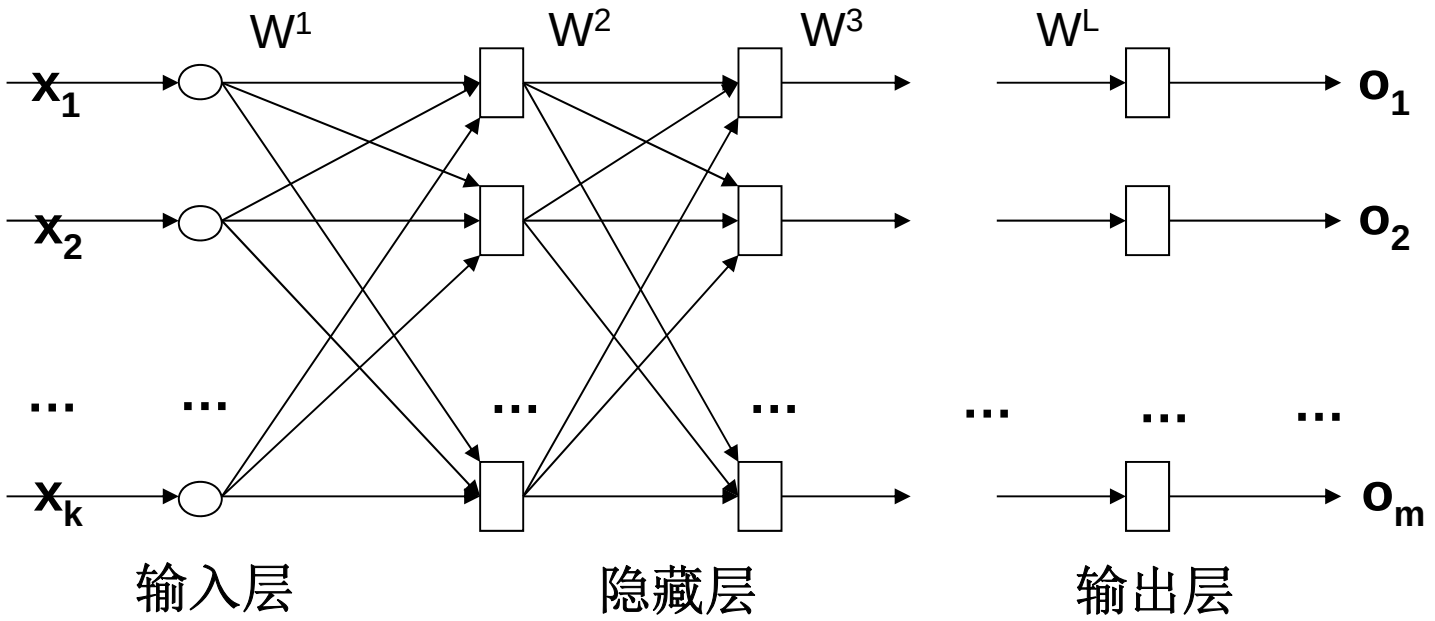
$$\Delta w_k = \eta \frac{\partial E}{\partial w_k}$$

$$w_k \leftarrow w_k + \Delta w_k$$

提纲

- 概述
- 单层神经网络
- 多层神经网络
- 设计ANN时需要考虑的问题

多层网络和反向传播(BP)算法

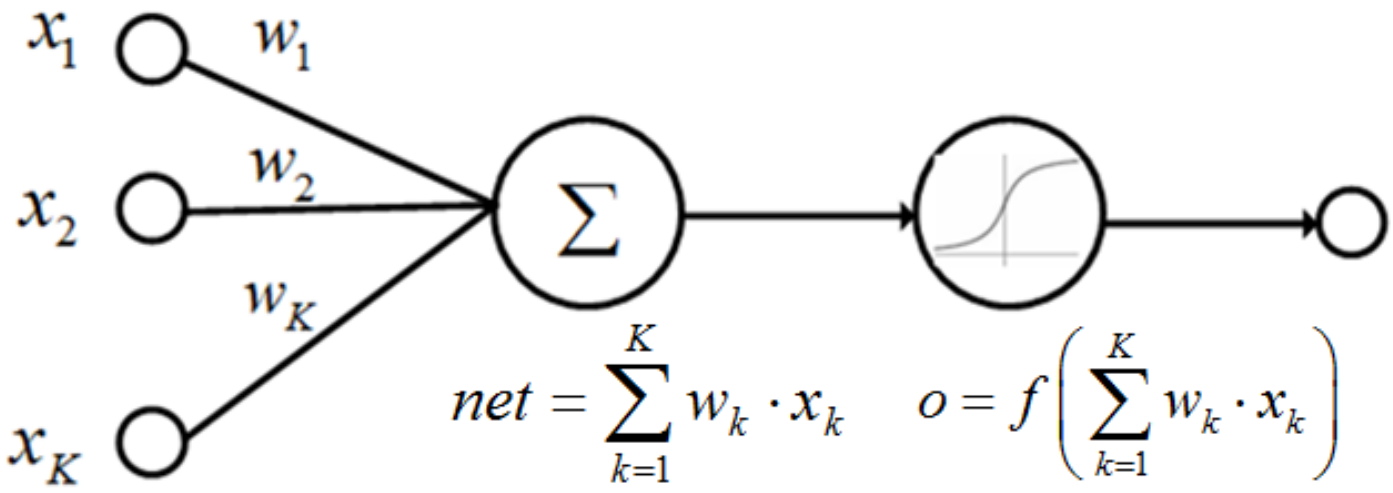


非线性变换

- 先计算输入向量各维的线性组合，然后应用非线性变换

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

$$f'(net)=-\frac{1}{(1+e^{-net})^2}(-e^{-net})=o(1-o)$$



非线性变换

- Logistic函数
  - 输出范围是0到1
  - 单调递增
  - 导数容易用函数本身表示
- Logistic函数的变型
  - 其他易计算导数的可微函数
  - 增加陡峭性
  - 双曲正切函数

反向传播(BP)算法

向前传播阶段：

（1）从样本集中取一个样本($x^{(i)}$，$y^{(i)}$)，将$x^{(i)}$输入网络；

（2）计算相应的实际输出：

$$o^{(n)} = f_L(w^L \ldots \cdot f_2(w^2 \cdot f_1(w^1 \cdot x^{(n)})))$$

反向传播(BP)算法

反向传播阶段—误差传播阶段：

（1）计算实际输出与相应的真实输出的差；

（2）按极小化误差的方式调整权矩阵。

（3）网络关于第i个样本的误差测度：

$$E^{(i)} = \frac{1}{2}\sum_{j=1}^{m}\left(y_j^{(i)} - o_j^{(i)}\right)^2$$

$$E = \sum_{i=1}^{N} E^{(i)}$$

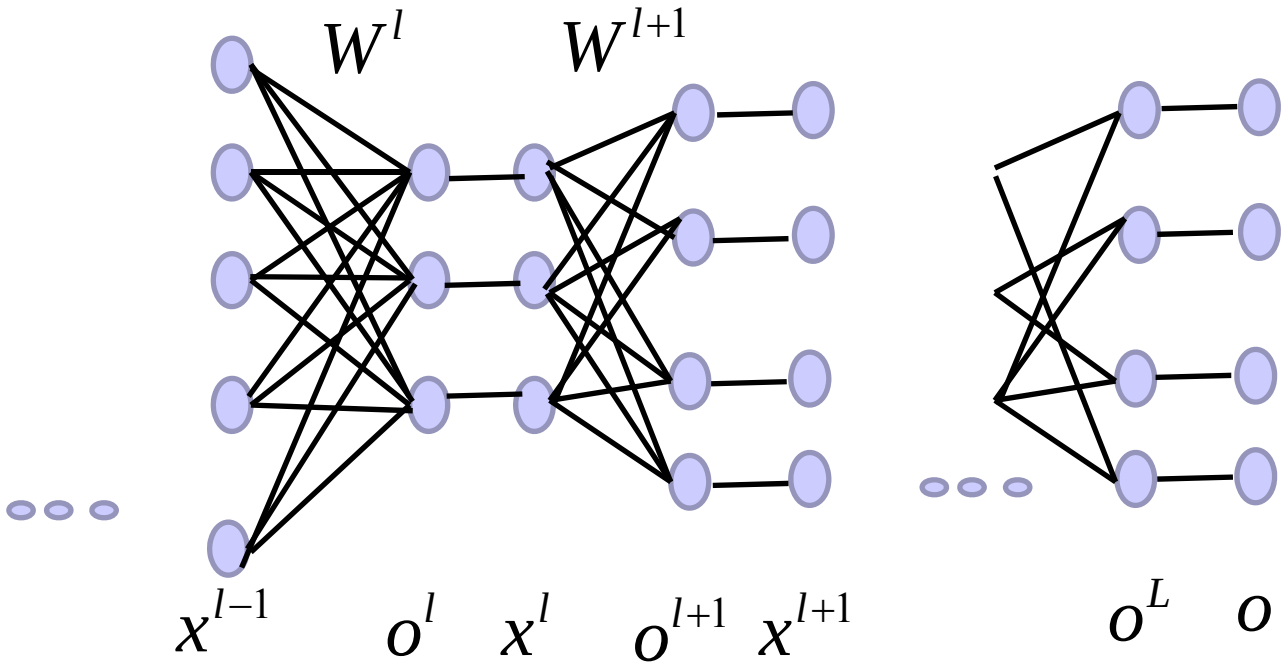
反向传播（BP）算法

- 反向传播算法的学习任务
  - 搜索一个巨大的假设空间，这个空间由网络中所有的单元的所有可能的权值定义，得到误差曲面
  - 在多层网络中，**误差曲面可能有多个局部极小值，梯度下降仅能保证收敛到局部极小值**
  - 尽管有局部极小的问题，对于实践中的很多应用，反向传播算法都产生了出色的结果

反向传播（BP）算法

- 样本集：$X=\{(X^1,Y^1),(X^2,Y^2),\ldots,(X^{(N)},Y^{(N)})\}$
- 基本思想：
  - **给定随机初始化的权值矩阵**
  - **前向计算：**根据样本集中的样本$(X^{(i)},Y^{(i)})$计算实际输出$O^{(i)}$和误差$E^{(i)}$，对W^1，W^2，…，W^L各做一次调整，循环，直到$E<\varepsilon$。
  - **后向计算：**用输出层的误差调整输出层权矩阵，并用此误差估计输出层的直接前导层的误差，再用输出层前导层误差估计更前一层的误差。形成将输出端表现出的误差沿着与输入信号相反的方向逐级向输入端传递的过程

反向传播（BP）算法的权值更新



反向传播（BP）算法的权值更新

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{m}\left(y_j^{(i)} - o_j^{(i)}\right)^2 \tag{1}$$

$$x^l = f\left(o^l\right) \qquad o^l = W^l x^{l-1} + w_0 \tag{2}$$

$$\delta^l = (W^{l+1})^T \cdot \delta^{l+1} \circ f'(\mathrm{o}) \tag{3}$$

$$\delta^L = f'(\mathrm{o}^L) \circ \sum_{i=1}^{N}(\mathrm{y}^{(i)} - \mathrm{o}^{(i)}) \tag{4}$$

$$\frac{\partial E}{\partial w^l} = x^{l-1} \circ (\delta^l)^T \tag{5}$$

$$\Delta W = -\eta \frac{\partial E}{\partial W} \tag{6}$$

反向传播（BP）算法的权值更新

$$\delta^l = (W^{l+1})^T \cdot \delta^{l+1} \circ f'(\mathrm{o}) \qquad \text{(3)的推导}$$

$$\delta^l = \frac{\partial E}{\partial o^l}$$

$$= \frac{\partial E}{\partial o^{l+1}} \circ \frac{\partial o^{l+1}}{\partial o^l}$$

$$= \delta^{l+1} \circ W^{l+1} \cdot f'(\mathrm{o}^l)$$

$$o^l = W^l x^{l-1} + w_0$$

$$o^{l+1} = W^{l+1} x^l + w_0$$

反向传播（BP）算法的权值更新

$$\delta^L = f'(\mathrm{o}^L) \circ \sum_{i=1}^{N} (\mathrm{y}^{(i)} - \mathrm{o}^{(i)})$$

(4)的推导

$$\delta^L = \frac{\partial E}{\partial o^L}$$

$$= \frac{\partial E}{\partial o} \circ \frac{\partial o}{\partial o^L}$$

$$= (y^{(i)} - o^{(i)}) \circ f'(o^L)$$

$$o = f(o^L)$$

反向传播（BP）算法的权值更新

$$\frac{\partial E}{\partial W^l} = x^{l-1} \circ (\delta^l)^T$$

(5)的推导

$$\frac{\partial E}{\partial W^l} = \frac{\partial E}{\partial o^l} \circ \frac{\partial o^l}{\partial W^l}$$

$$= \frac{\partial E}{\partial o^l} \circ \frac{\partial o^l}{\partial W^l}$$

$$= x^{l-1}(\delta^l)^T$$

$$o^l = W^l x^{l-1} + W_0$$

反向传播（BP）算法

反向传播（**BP**）算法伪代码

1 for l=1 to L do

1.1 初始化W^l;

2 初始化精度控制参数ε;

3 E=ε+1;

4 while E>ε do

4.1 权值矩阵更新;

反向传播算法

4.2 对S中的每一个样本（$\mathbf{X^{(i)},Y^{(i)}}$）：

4.2.1 计算出$X^{(i)}$对应的实际输出$O^{(i)}$；

4.2.2 计算出$E^{(i)}$；

4.2.3 $E=E+E^{(i)}$；

4.2.4 根据相应式子调整W^{L}；

4.2.5 $l=L-1$；

4.2.6 while $l\neq 0$ do

4.2.6.1 根据相应式子调整W^{l}；

4.2.6.2 $l=l-1$

4.3 E=E/2.0

提纲

- 概述
- 单层神经网络
- 多层神经网络
- 设计ANN时需要考虑的问题

收敛性和局部极小值

- 网络的权越多，误差曲面的维数越高，也就越可能为梯度下降提供更多的局部极小点
- 迭代过程中网络权值的演化
  - 如果把网络的权值初始化为接近于0的值，那么在早期的梯度下降步骤中，网络将表现为一个非常平滑的函数，近似为输入的线性函数，这是因为Logistic函数本身在权值靠近0时接近线性
  - 仅当权值增长一定时间后，它们才会到达可以表示较高非线性网络函数的程度；
  - 在这个能表示更复杂函数的权空间区域存在更多的局部极小值，但是人们认为当权到达这一点时，它们已经靠近全局最小值

收敛性和局部极小值

- 用来缓解局部极小值问题的启发式规则
  - 为梯度更新法则加一个冲量，可以带动梯度下降过程，冲过狭窄的局部极小值
  - 使用随机的梯度下降：对于每个训练样例沿一个不同的误差曲面有效下降，这些不同的误差曲面通常有不同的局部极小值，这使得下降过程不太可能陷入一个局部极小值
  - 使用同样的数据训练多个网络，但用不同的随机权值初始化每个网络。如果不同的训练产生不同的局部极小值，那么对分离的验证集合性能最好的那个网络将被选中，或者保留所有的网络，输出是所有网络输出的平均值

隐藏层的表示

- 反向传播算法的特性是：它能够在网络内部的隐藏层发现有用的中间表示
- 多层网络在隐藏层自动发现中间层特征的能力是ANN学习的一个关键特性
- 网络中使用的单元层越多，就可以创造出越复杂的中间层特征

拓扑结构的确定

- 隐层数
  - 增加隐层数可以降低网络误差，也使网络复杂化，从而增加了网络的训练时间和出现“过拟合”的倾向。
  - 在设计BP网络时可参考这一点，应优先考虑3层BP网络（即有1个隐层）
  - 一般地，靠增加隐层节点数来获得较低的误差，其训练效果要比增加隐层数更容易实现。

拓扑结构的确定

- 隐层节点数
  - 是训练时出现“过拟合”的直接原因，但是理论上还没有一种普遍的解决办法。
  - 为尽可能避免训练时出现“过拟合”现象，保证足够高的网络性能和泛化能力，确定隐层节点数的最基本原则是：在满足精度要求的前提下取尽可能紧凑的结构，即取尽可能少的隐层节点数。

拓扑结构的确定

- 隐层节点数
  - 隐层节点数须小于*N*-1（其中*N*为训练样本数），否则，网络模型的误差与训练样本的特性无关而趋于零，即建立的网络模型没有泛化能力，也没有任何实用价值。同理可推得：输入层的节点数（变量数）必须小于*N*-1。
  - 训练样本数必须多于网络模型的连接权数，一般为2~10倍。

泛化、过拟合和迭代停止

- 权值更新算法的终止条件
  - 一种策略是，对训练样例的误差降低至某个预先定义的阈值之下
- 泛化精度：网络拟合训练数据外的实例的精度

过拟合

- 过拟合发生在迭代的后期
  - 设想网络的权值是被初始化为小随机值的，使用这些几乎一样的权值仅能描述非常平滑的决策面
  - 随着训练的进行，一些权值开始增长，以降低在训练数据上的误差，同时学习到的决策面的复杂度也在增加
  - 如果权值调整迭代次数足够多，反向传播算法可能会产生过度复杂的决策面，拟合了训练数据中的噪声和训练样例中没有代表性的特征

过拟合解决方法

- 权值衰减
  - 它在每次迭代过程中以某个小因子降低每个权值，这等效于修改E的定义，加入一个与网络权值的总量相应的惩罚项，此方法的动机是保持权值较小，从而使学习过程向着复杂决策面的反方向偏置
- 验证数据
  - 最成功的方法是在训练数据外再为算法提供一套验证数据，应该使用在验证集合上产生最小误差的迭代次数，不是总能明显地确定验证集合何时达到最小误差

过拟合解决方法

- k-fold交叉验证方法
  - 把训练样例分成k份，然后进行k次交叉验证过程，每次使用不同的一份作为验证集合，其余k-1份合并作为训练集合。
  - 每个样例会在一次实验中被用作验证样例，在k-1次实验中被用作训练样例
  - 每次实验中，使用上面讨论的交叉验证过程来决定在验证集合上取得最佳性能的迭代次数，然后计算这些迭代次数的均值
  - 最后，运行一次反向传播算法，训练所有m个实例并迭代