人工智能与机器学习

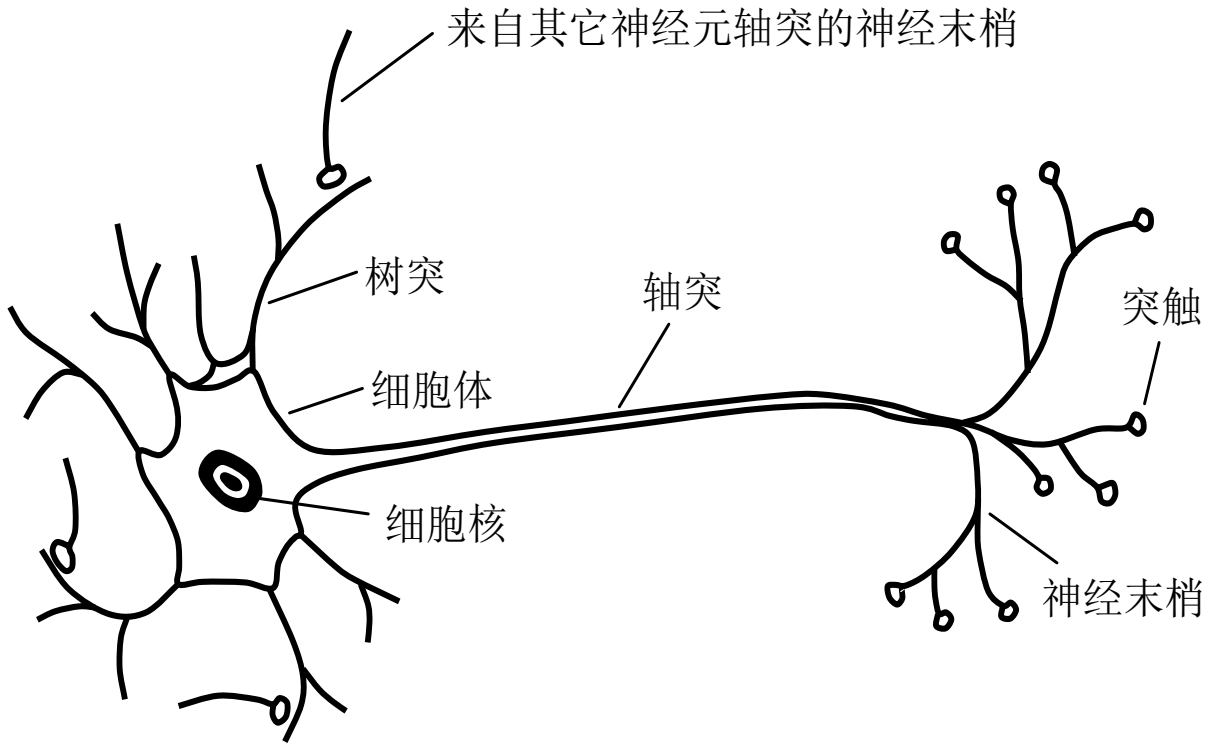
第十一讲 神经网络（上）

倪东 叶琦
工业控制研究所

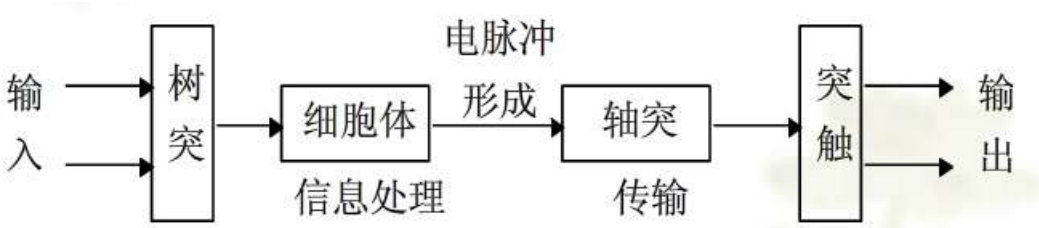
杭州 • 浙江大学 • 2022

生物神经元

生物神经元的结构



神经细胞是构成神经系统的基本单元，称之为生物神经元，简称神经元。神经元主要构成：细胞体、轴突、树突和突触。



生物神经元的工作机制

兴奋和抑制两种状态

抑制状态的神经元
由树突和细胞体
接收传来的兴奋电位

输入兴奋总量超过阈值

神经元被激发
进入兴奋状态

不应期

产生输出脉冲

由突触传递给其它神经元

生物神经元的六个基本特征

1）神经元及其联接；

2）神经元之间的联接强度决定信号传递的强弱；

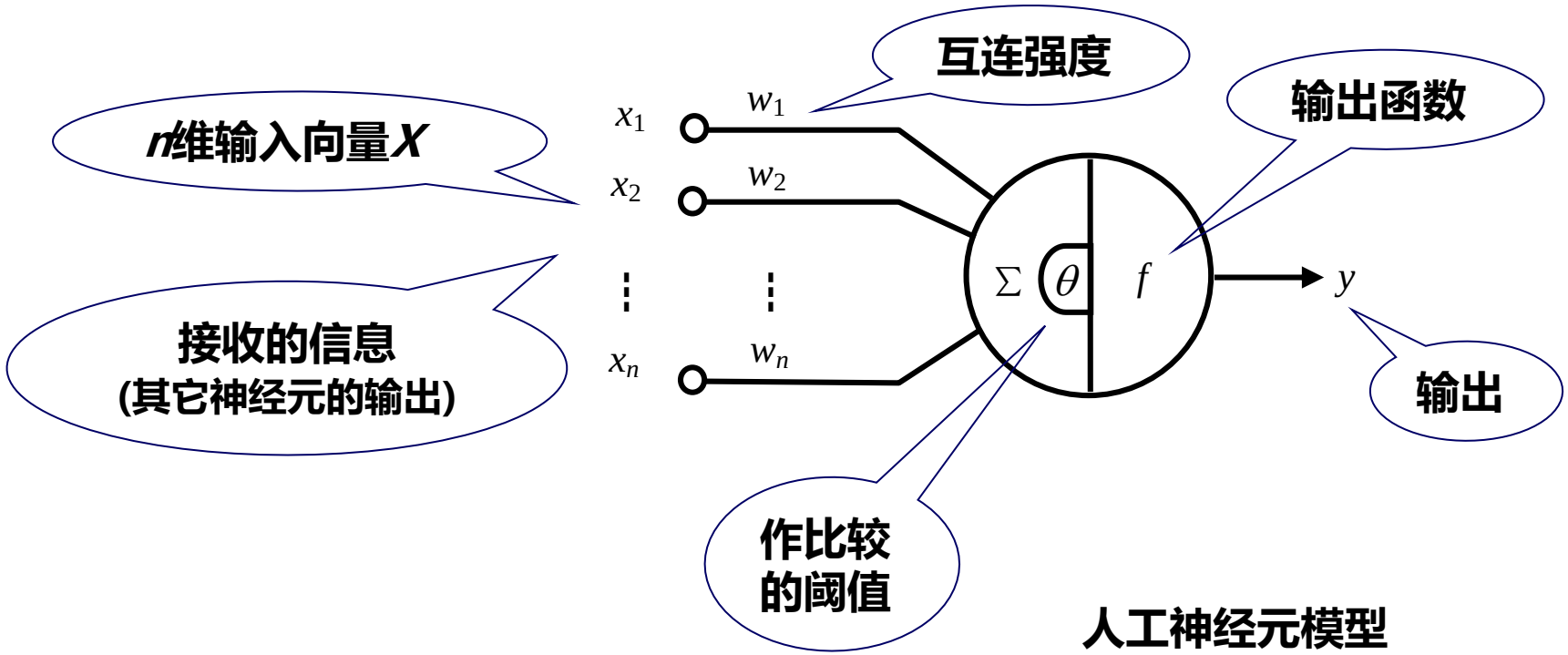
3）神经元之间的联接强度是可以随训练改变的；

4）信号可以是起刺激作用的，也可以是起抑制作用的；

5）一个神经元接受的信号的累积效果决定该神经元的状态；

6）每个神经元可以有一个“阈值”。

人工神经元：生物神经元的简化模拟。



人工神经元模型

人工神经元间的互连：信息传递路径轴突-突触-树突的简化；

连接的权值：两个互连的神经元之间相互作用的强弱。

人工神经网络概述

- 人工神经网络提供了一种普遍且实用的方法从样例中学习值为实数、离散值或向量的函数
- 反向传播算法，使用梯度下降来调节网络参数以最佳拟合由输入—输出对组成的训练集合
- 人工神经网络对于训练数据中的错误健壮性很好
- 人工神经网络已被成功应用到很多领域，例如视觉场景分析，语音识别，机器人控制，工业过程控制

生物学动机

- ANN受到生物学的启发，生物的学习系统是由相互连接的神经元组成的异常复杂的网络。
- ANN系统的一个动机就是获得这种基于分布表示的高度并行算法
- ANN并未模拟生物神经系统中的很多复杂特征
- ANN的研究分为两个方向
  - 使用ANN研究和模拟生物学习过程
  - 获得高效的机器学习算法，不管这种算法是否反映了生物过程
- 属于后一个研究方向

适合神经网络学习的问题

- 训练集合为含有噪声的复杂传感器数据，例如来自摄像机和麦克风，工业过程各类传感器数据
- 需要较多符号表示的问题，例如决策树学习的任务，能够取得和决策树学习大体相当的结果
- 反向传播算法是最常用的ANN学习技术

反向传播算法适合问题的特征

- 实例是用很多“属性-值”对表示的
- 目标函数的输出可能是离散值、实数值或者由若干实数属性或离散属性组成的向量
- 训练数据可能包含错误
- 可容忍长时间的训练
- 可能需要快速求出目标函数值
- 人类能否理解学到的目标函数是不重要的

提纲

- 讨论训练单个单元的学习算法
- 介绍组成神经网络的几种主要单元
  - 感知器（perceptron）
  - 线性单元（linear unit）
  - sigmoid单元（sigmoid unit）
- 给出训练多层网络的反向传播算法
- 讨论几个一般性问题
  - ANN的表征能力
  - 假设空间搜索的本质特征
  - 过度拟合问题
  - 反向传播算法的变体

感知器

- 一种类型的ANN系统是以感知器为基础
- 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

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

其中每个w_i是一个实数常量，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。

感知器（2）

- 两种简化形式，附加一个常量输入$x_0=1$，前面的不等式写成 $\sum_{i=0}^{n} w_i x_i > 0$

  或写成向量形式 $\vec{w} \cdot \vec{x} > 0$

- 为了简短起见，把感知器函数写为 $o(\vec{x}) = \mathrm{sgn}(\vec{w} \cdot \vec{x})$

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

感知器（3）

- 学习一个感知器意味着选择权$w_0,\ldots,w_n$的值。所以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

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

感知器的表征能力

- 可以把感知器看作是n维实例空间（即点空间）中的超平面决策面
- 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 $\vec{w} \cdot \vec{x} = 0$
- 可以被某个超平面分割的样例集合，称为线性可分样例集合

感知器的表征能力（2）

- 单独的感知器可以用来表示很多布尔函数
- 感知器可以表示所有的原子布尔函数：与、或、与非、或非
- 然而，一些布尔函数无法用单一的感知器表示，例如异或

感知器的表征能力（3）

- 因为所有的布尔函数都可表示为基于原子函数的互连单元的某个网络，因此，感知器网络可以表示所有的布尔函数。事实上，只需要两层深度的网络，比如表示析取范式
- 注意，要把一个AND感知器的输入求反只要简单地改变相应输入权的符号
- 因为感知器网络可以表示大量的函数，而单独的单元不能做到这一点，所以感兴趣的是学习感知器组成的多层网络

感知器训练法则

- 虽然目的是学习由多个单元互连的网络，但还是要从如何学习单个感知器的权值开始
- 单个感知器的学习任务，决定一个权向量，它可以使感知器对于给定的训练样例输出正确的1或-1
- 主要考虑两种算法
  - 感知器法则
  - delta法则
- 这两种算法保证收敛到可接受的假设，在不同的条件下收敛到的假设略有不同
- 这两种算法提供了学习多个单元构成的网络的基础

感知器法则

- 算法过程
  - 从随机的权值开始
  - 反复应用这个感知器到每个训练样例，只要它误分类样例就修改感知器的权值
  - 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 $w_i \leftarrow w_i + \Delta w_i$

  其中 $\Delta w_i = \eta(t - o)x_i$

感知器法则（2）

- 为什么这个更新法则会成功收敛到正确的权值呢？
  - 一些例子能说明收敛过程 $x_i(w_i + \Delta w_i) = x_i w_i + \eta(t - o)x_i^2$
  - 可以证明（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**
- 收敛的前提条件：训练样例线性可分

梯度下降和delta法则

- delta法则克服感知器法则的不足，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法则的关键思想是: 使用梯度下降来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，梯度下降是必须遍历这样的空间的所有算法的基础

梯度下降和delta法则（2）

- 把delta训练法则理解为训练一个无阈值的感知器

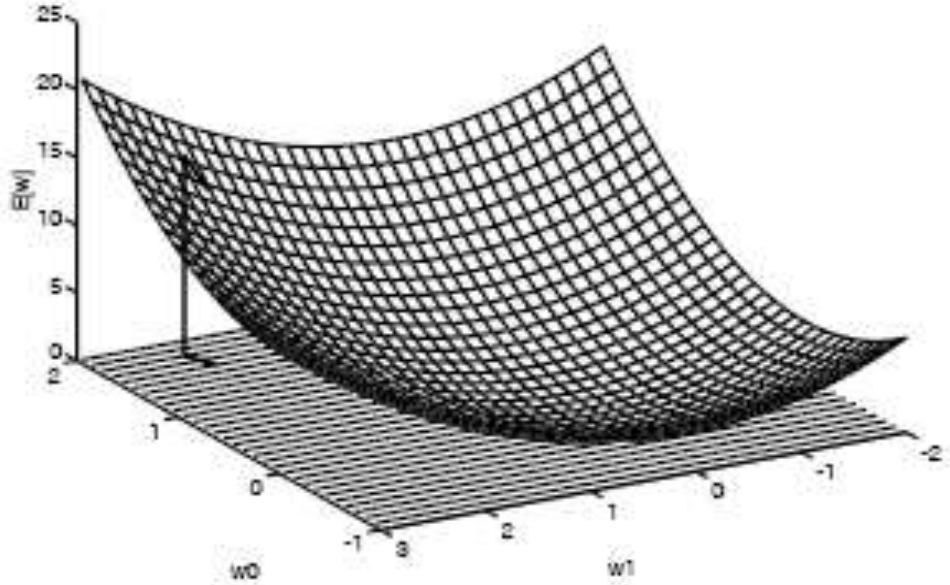
$$O(\vec{X}) = \vec{W} \cdot \vec{X}$$

- 指定一个度量标准来衡量假设相对于训练样例的训练误差

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- 第6章给出了选择这种E定义的一种贝叶斯论证，在一定条件下，使E最小化的假设就是H中最可能的假设

可视化假设空间



- 根据E的定义，误差曲面是一个抛物面，存在一个单一全局最小值

- 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

梯度下降法则的推导

- 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
    $$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$
  - 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
- 梯度训练法则 $\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$

  其中，$\Delta\vec{w} = -\eta\nabla E(\vec{w})$
- 也可写成分量形式：
  $$\Delta w_i = -\eta\frac{\partial E}{\partial w_i}$$

梯度下降法则的推导（2）

- 需要一个高效的方法在每一步都计算这个梯度

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} = \frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2$$

$$= \frac{1}{2}\sum_{d\in D}2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d)$$

$$= \sum_{d\in D}(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x}_d)$$

- 梯度下降权值更新法则

$$\Delta w_i = \eta\sum_{d\in D}(t_d - o_d)x_{id} \quad (式4.7)$$

梯度下降法则的推导（3）

- 训练线性单元的梯度下降算法

Gradient-Descent(training_examples, η)

training_examples中每个训练样例形式为序偶< $\vec{x}$,t> , $\vec{x}$ 是输入值向量，t是目标输出值，η是学习速率

- 初始化每个w_i为某个小的随机值
- 遇到终止条件之前，做以下操作
  - 初始化每个Δw_i为0
  - 对于训练样例training_examples中的每个<$\vec{x}$,t>，做
    - **把实例$\vec{x}$ 输入到此单元，计算输出o**
    - **对于线性单元的每个权增量Δw_i，做**

      $\Delta w_i \leftarrow \Delta w_i + \eta(t-o)x_i$
  - 对于线性单元的每个权w_i，做

    $w_i \leftarrow w_i + \Delta w_i$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

梯度下降法则的推导（4）

- 梯度下降算法如下
  - 选取一个初始的随机权向量
  - 应用线性单元到所有的训练样例，根据公式4.7计算每个权值的$\vec{w}$更新权值
- 因为误差曲面仅包含一个全局的最小值，所以无论训练样例是否线性可分，算法都会收敛到具有最小误差的权向量，条件是使用足够小的学习速率
- 算法的一种常用改进方法是随着梯度下降步数的增加逐渐减小学习速率（变学习率）

梯度下降的随机近似

- 梯度下降是一种重要的通用学习范型，它是搜索庞大假设空间或无限假设空间一种策略
- 梯度下降应用于满足以下条件的任何情况
  - 假设空间包含连续参数化的假设
  - 误差对于这些假设参数可微
- 梯度下降的主要实践问题
  - 有时收敛过程可能非常慢
  - 如果在误差曲面上有多个局部极小值，那么不能保证找到全局最小值

梯度下降的随机近似（2）

- 随机梯度下降（或称增量梯度下降）
  - 根据某个单独样例的误差增量计算权值更新，得到近似的梯度下降搜索（随机取一个样例）
  - 对梯度下降算法的修改

$$\Delta w_i \leftarrow \eta(t-o)x_i, \quad w_i \leftarrow w_i + \Delta w_i$$

  - 可以看作为每个单独的训练样例定义不同的误差函数
  - 在迭代所有训练样例时，这些权值更新的序列给出了对于原来误差函数的梯度下降的一个合理近似
  - 通过使下降速率的值足够小，可以使随机梯度下降以任意程度接近于真实梯度下降

梯度下降的随机近似（3）

- 标准梯度下降和随机梯度下降之间的关键区别
  - 标准梯度下降是在权值更新前对所有样例汇总误差，而随机梯度下降的权值是通过考查每个训练样例来更新的
  - 在标准梯度下降中，权值更新的每一步对多个样例求和，需要更多的计算
  - 标准梯度下降，由于使用真正的梯度，标准梯度下降对于每一次权值更新经常使用比随机梯度下降大的步长
  - 如果标准误差曲面有多个局部极小值，随机梯度下降有时可能避免陷入这些局部极小值
- 实践中，标准和随机梯度下降方法都被广泛应用

梯度下降的随机近似（4）

- delta法则（增量法则），又称LMS法则、Adaline法则、Windrow-Hoff法则
- 增量法则与感知器法则的相似和区别 $o(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$ $o(\vec{x}) = \vec{w} \cdot \vec{x}$
- delta法则可以学习非阈值线性单元的权，也可以用来训练有阈值的感知器单元。
- 如果非阈值输出能够被训练到完美拟合这些值，那么阈值输出也会完美拟合它们
- 即使不能完美地拟合目标值，只要线性单元的输出具有正确的符号，阈值输出就会正确拟合目标值
- 尽管这个过程会得到使线性单元输出的误差最小化的权值，但这些权值不能保证阈值输出的误差最小化

感知器学习小结

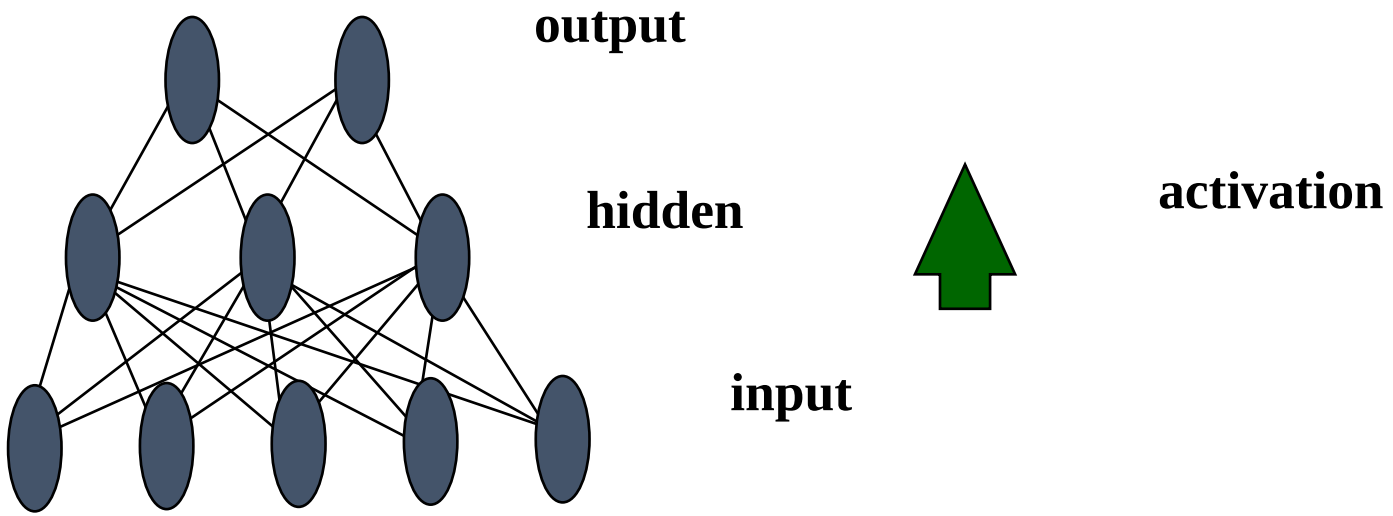
- 感知器法则和delta法则的关键差异
  - 前者根据阈值化的感知器输出的误差更新权值
  - 后者根据输入的非阈值化线性组合的误差来更新权值
- 这个差异带来不同的收敛特性
  - 前者经过有限次的迭代收敛到一个能理想分类训练数据的假设，条件是训练样例线性可分
  - 后者可能经过极长的时间，渐近收敛到最小误差假设，但无论训练样例是否线性可分都会收敛

感知器学习小结（2）

- 学习权向量的第3种方法是线性规划
- 线性规划是解线性不等式方程组的一种通用的有效方法
- 这种方法仅当训练样例线性可分时有解
- Duda和Hart给出了一种更巧妙的适合非线性可分的情况的方法
- 更大的问题是，无法扩展到训练多层网络，而delta法则可以很容易扩展到多层网络

多层网络和反向传播算法

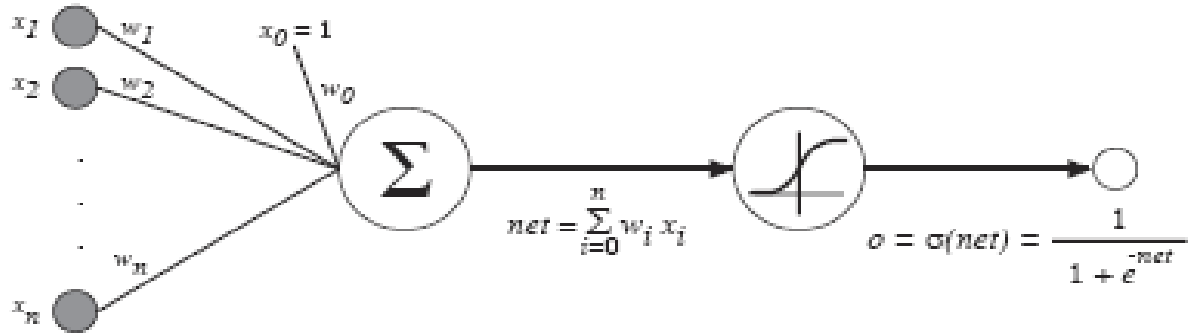
- 多层网络能够表示种类繁多的非线性曲面
- 典型的多层网络



可微阈值单元

- 使用什么类型的单元来构建多层网络?
- 多个线性单元的连接仍产生线性函数，而我们希望构建表征非线性函数的网络
- 感知器单元可以构建非线性函数，但它的不连续阈值使它不可微，不适合梯度下降算法
- 我们需要的单元满足的条件
  - 输出是输入的非线性函数
  - 输出是输入的可微函数
- Sigmoid单元，类似于感知器单元，但基于一个平滑的可微阈值函数

可微阈值单元（2）



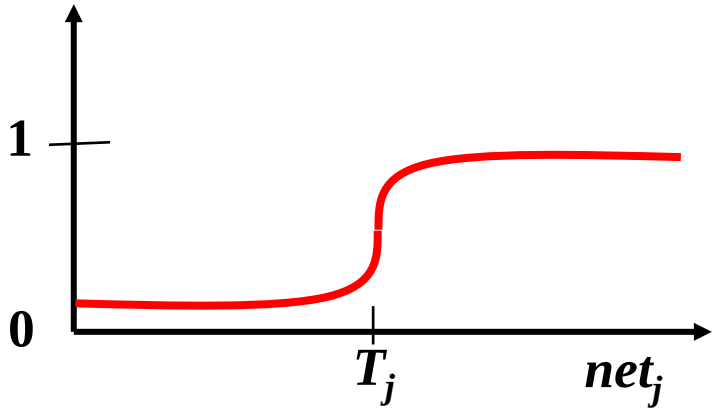
- sigmoid单元先计算它的输入的线性组合，然后应用到一个阈值上，阈值输出是输入的连续函数

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

其中 $\sigma(y) = \dfrac{1}{1+e^{-y}}$

可微阈值单元（3）

- sigmoid函数
  - 也称logistic函数
  - 挤压函数
  - 输出范围是0到1
  - 单调递增
  - 导数很容易用函数本身表示
- sigmoid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

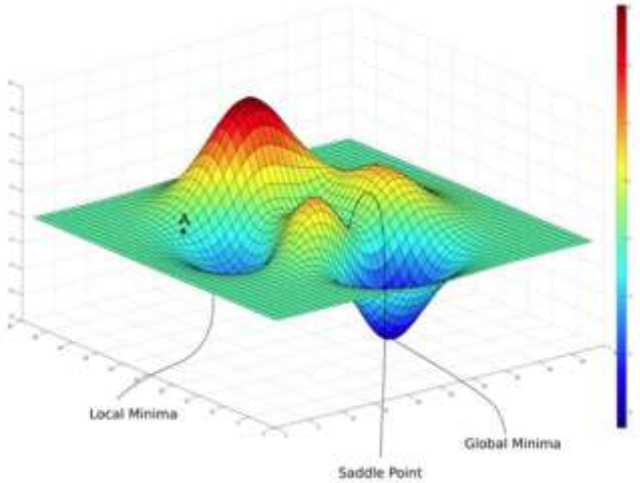


反向传播算法

- 用来学习多层网络的权值
- 采用梯度下降方法试图最小化网络输出值和目标值之间的误差平方
- 网络的误差定义公式，对所有网络输出的误差求和

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outpus} (t_{kd} - o_{kd})^2$$

反向传播算法（2）



- 反向传播算法面临的学习任务
  - 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，得到与前面类似的误差曲面
  - 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
  - 尽管有这个障碍，已经发现对于实践中很多应用，反向传播算法都产生了出色的结果

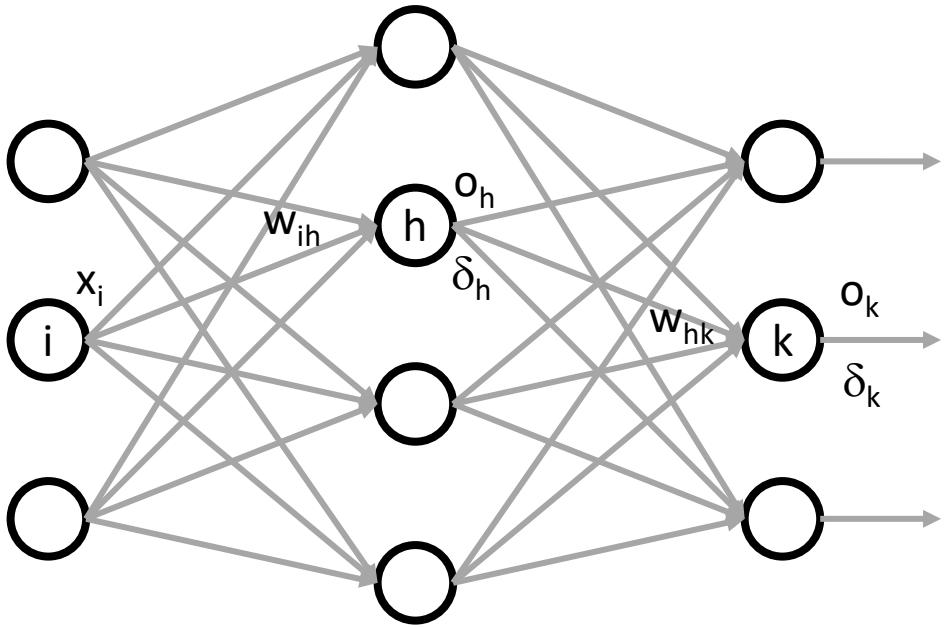
反向传播算法（3）

- 包含两层sigmoid单元的前馈网络的反向传播算法
- BackPropagation(training_examples, η, n_{in}, n_{out}, n_{hidden})
- training_examples是序偶$<\vec{x},\vec{t}>$的集合，$\vec{x}$是网络输入值向量，$\vec{t}$是目标输出值。η是学习速率，n_{in}是网络输入的数量，n_{hidden}是隐藏层单元数，n_{out}是输出单元数。从单元i到单元j的输入表示为x_{ji}，单元i到单元j的权值表示为w_{ji}。
- 创建具有n_{in}个输入，n_{hidden}个隐层，n_{out}个输出单元的网络
- 初始化所有的网络权值为小的随机值
- 在遇到终止条件前
  - 对于训练样例training_examples中的每个 $<\vec{x}, \vec{t}>$：
    - 把输入沿网络前向传播
      - 把实例$\vec{x}$输入网络，并计算网络中每个单元u的输出o_u
    - 使误差沿网络反向传播
      - 对于网络的每个输出单元k，计算它的误差项$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$
      - 对于网络的每个隐层单元h，计算它的误差项 $\delta_h \leftarrow o_h(1-o_h) \sum_{k \in outputs} w_{kh}\ \delta_k$
      - 更新每个网络权值$w_{ji} \leftarrow w_{ji}+\Delta w_{ji}$，其中$\Delta w_{ji}=\eta \delta_j x_{ji}$

反向传播算法（4）

- 前面给出的反向传播算法适用于包含两层sigmoid单元的分层前馈网络，并且每一层的单元与前一层的所有单元相连。
- 是反向传播算法的增量梯度下降（或随机梯度下降）版本
- 使用的符号做了如下扩展
  - 网络中每个节点被赋予一个序号，这里的节点要么是网络的输入，要么是网络中某个单元的输出
  - x_{ji}表示节点i到单元j的输入，w_{ji}表示对应的权值
  - δ_n表示与单元n相关联的误差项。

反向传播算法（5）



- **输出节点**
  1. $\delta_k = o_k(1-o_k)(t_k-o_k)$
  2. $\Delta w_{hk} = \eta \delta_k o_h$
  3. $w_{hk} \leftarrow w_{hk} + \Delta w_{hk}$

- **隐藏节点**
  1. $\delta_h = o_h(1-o_h) \sum_{k \in Downstream(h)} w_{hk}\ \delta_k$
  2. $\Delta w_{ih} = \eta \delta_h o_i$
  3. $w_{ih} \leftarrow w_{ih} + \Delta w_{ih}$

算法解释

- 从建立一个具有期望数量的隐层单元和输出单元的网络并初始化所有的网络的权值为小的随机数开始
- 给定一个固定的网络结构，算法的主循环就对训练样例进行反复的迭代
- 对于每一个训练样例，它应用目前的网络到这个样例，计算出对这个样例网络输出的误差，然后更新网络中所有的权值
- 对这样的梯度下降步骤进行迭代，直到网络的性能达到可接受的精度为止

反向传播算法的梯度下降法则

- 算法中的梯度下降权更新法则与delta训练法则相似
- 类似delta法则，依照以下三者来更新每一个权
  - 学习速率η
  - 该权值涉及的输入值x_{ji}
  - 该单元的输出误差δ
- 不同于delta法则的地方
  - delta法则中的误差项被替换成一个更复杂的误差项δ_j

反向传播算法的误差项

- 输出单元k的误差项
  - δ_k与delta法则中的(t_k-o_k)相似，但乘上了sigmoid挤压函数的导数$o_k(1-o_k)$。
- 隐层单元h的误差项
  - 因为训练样例仅对网络的输出提供了目标值t_k，所以缺少直接的目标值来计算隐层单元的误差值
  - 采取以下的间接方法计算隐层单元的误差项：对受隐层单元h影响的每一个单元的误差δ_k进行加权求和，每个误差δ_k权值为w_{kh}，w_{kh}就是从隐层单元h到输出单元k的权值。这个权值刻画了隐层单元h对于输出单元k的误差应负责的程度。

算法解释（2）

- 算法随着每个训练样例的出现而递增地更新权，这一点与梯度下降的随机近似算法一致
- 要取得误差E的真实梯度，需要在修改权值之前对所有训练样例的$\delta_j x_{ji}$值求和
- 在典型的应用中，权值的更新迭代会被重复上千次
- 有很多终止条件可以用来停止这个过程
  - 迭代的次数到了一个固定值时停止
  - 当在训练样例上的误差降到某个阈值以下
  - 在分离的验证样例集合上的误差符合某个标准
- 终止条件很重要，太少的迭代无法有效地降低误差，太多的迭代会导致对训练数据的过度拟合

增加冲量项

- 因为反向传播算法的应用如此广泛，所以已经开发出了很多反向传播算法的变体
- 修改权值更新法则，使第n次迭代时的权值的更新部分地依赖于发生在第n-1次迭代时的更新，比如
  - $\Delta w_{ji}(n)=\eta\delta_j x_{ji}+\alpha\Delta w_{ji}(n-1)$
- 右侧第一项就是前面算法中的权值更新法则，第二项被称为冲量项
- 梯度下降的搜索轨迹就像一个球沿误差曲面滚下，冲量使球从一次迭代到下一次迭代时以同样的方向滚动
- 冲量有时会使这个球滚过误差曲面的局部极小值或平坦区域
- 冲量也具有在梯度不变的区域逐渐增大搜索步长的效果，从而加快收敛

学习任意的无环网络

- 算法可以简单地推广到任意深度的前馈网络 $\delta_r = o_r(1-o_r)\sum_{s\in m+1层} w_{sr}\delta_s$
- 第m层的单元r的δ_r值由更深的第m+1层δ值根据下式计算
- 将这个算法推广到任何有向无环结构也同样简单，而不论网络中的单元是否被排列在统一的层上，计算任意内部单元的δ的法则是：$\delta_r = o_r(1-o_r)\sum_{s\in Downstream(r)} w_{sr}\delta_s$，Downstream(r)是在网络中单元r的直接下游单元的集合，即输入中包括r的输出的所有单元

反向传播法则的推导

- 随机梯度下降算法迭代处理训练样例，每次处理一个，对于每个训练样例d，利用关于这个样例的误差E_d的梯度修改权值

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$E_d(\vec{w}) = \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

符号说明

- x_{ji}，单元j的第i个输入
- w_{ji}，与x_{ji}相关联的权值
- net_j，单元j的输入的加权和
- o_j，单元j计算出的输出
- t_j，单元j的目标输出
- σ，sigmoid函数
- outputs，网络最后一层的输出单元的集合
- Downstream(j)，单元j的输出到达的单元的集合

随机梯度下降法则的推导

分情况讨论$\frac{\partial E_d}{\partial net_j}$的推导

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} x_{ji}$$

- 输出单元

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j}$$

$$\begin{aligned}\frac{\partial E_d}{\partial o_j} &= \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2 \\ &= \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2 \\ &= \frac{1}{2}2(t_j - o_j)\frac{\partial(t_j - o_j)}{\partial o_j} \\ &= -(t_j - o_j)\end{aligned}$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1 - o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)o_j(1 - o_j)$$

$$\Delta w_{ji} = -\eta\frac{\partial E_d}{\partial w_{ji}} = \eta(t_j - o_j)o_j(1 - o_j)x_{ji}$$

随机梯度下降法则的推导（2）

- 隐层单元

$$
\begin{aligned}
\frac{\partial E_d}{\partial net_j} &= \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k w_{kj} \frac{\partial o_j}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k w_{kj} o_j (1 - o_j) \\
&= -o_j (1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}
\end{aligned}
$$

$$
\Delta w_{ji} = -\eta x_{ji} o_j (1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}
$$

小结

- 人工神经网络为学习实数值和向量值函数提供了一种实际的方法，对于连续值和离散值的属性都可以使用，并且对训练数据中的噪声具有很好的健壮性。
- 反向传播算法是最常见的网络学习算法
- 反向传播算法考虑的假设空间是固定连接的有权网络所能表示的所有函数的空间
- 包含3层单元的前馈网络能够以任意精度逼近任意函数，只要每一层有足够数量的单元。即使是一个实际大小的网络也能够表示很大范围的高度非线性函数
- 反向传播算法使用梯度下降方法搜索可能假设的空间，迭代减小网络的误差以拟合训练数据