模式识别引论

An Introduction to Pattern Recognition

主讲: 李春光

www.pris.net.cn/teacher/lichunguang

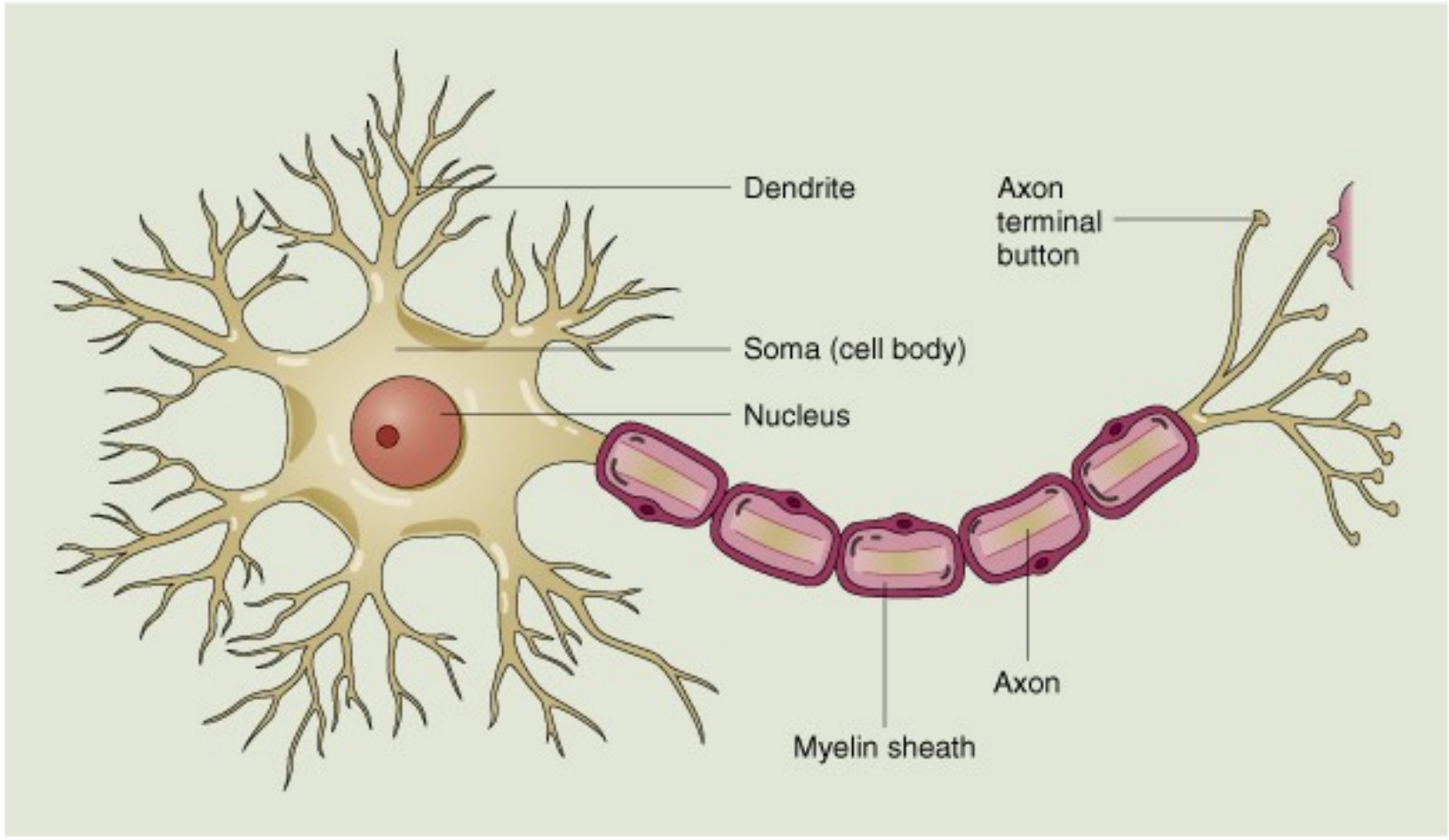
模式识别与智能系统实验室

网络搜索教研中心 信息与通信工程学院 北京邮电大学

神经网络 内容提要

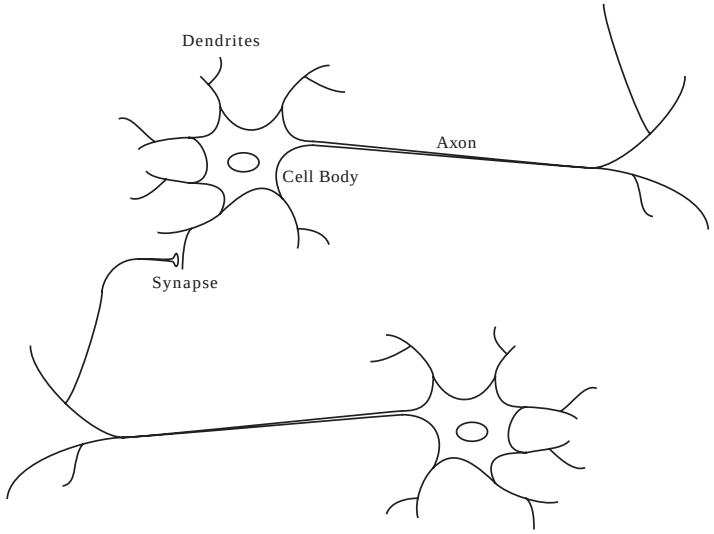
- 生物神经元
- 人工神经元模型
  - 数学表示、激活函数、网络结构
- 感知器(Perceptron)
  - 感知器学习算法(误差修正法则)
- 单层感知网络
  - 误差修正法则
- 多层感知器(Multi-Layer Perceptron)
  - 反传(BP)算法

引子: 生物神经元



生物神经元的结构

- **神经元构成**
  - 树突：接受从其他神经元传入信息的神经元纤维
  - 胞体：接受外来的信息，对各种信息进行汇总，并进行阈值处理，产生神经冲动
  - 轴突：连接其他神经元的树突和细胞体，以及完成神经元之间的信息传递



[美]Dennis Coon, John O. Mitterer 著，郑钢译，心理学导论——
思想与行为的认识之路（Gateways to Mind and Behavior）
(11th Edition)，中国轻工业出版社，2007.06

  - 突触的特点：单方向的传递信息，且强度可变、具有学习功能

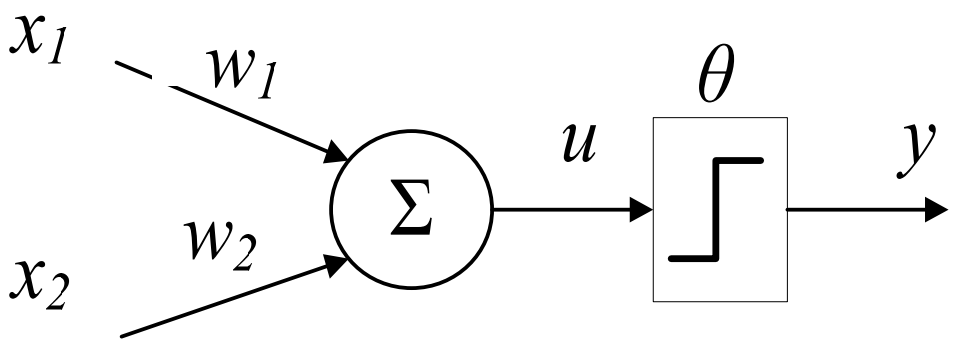
Artificial neuron...

神经网络 内容提要

- 生物神经元
- 人工神经元模型
  - 数学表示、激活函数、网络结构
- 感知器(Perceptron)
  - 感知器学习算法(误差修正法则)
- 单层感知网络
  - 误差修正法则
- 多层感知器(Multi-Layer Perceptron)
  - 反传(BP)算法

McCulloch-Pitts 神经元模型

- 对输入信号加权求和，再与阈值比较以确定输出

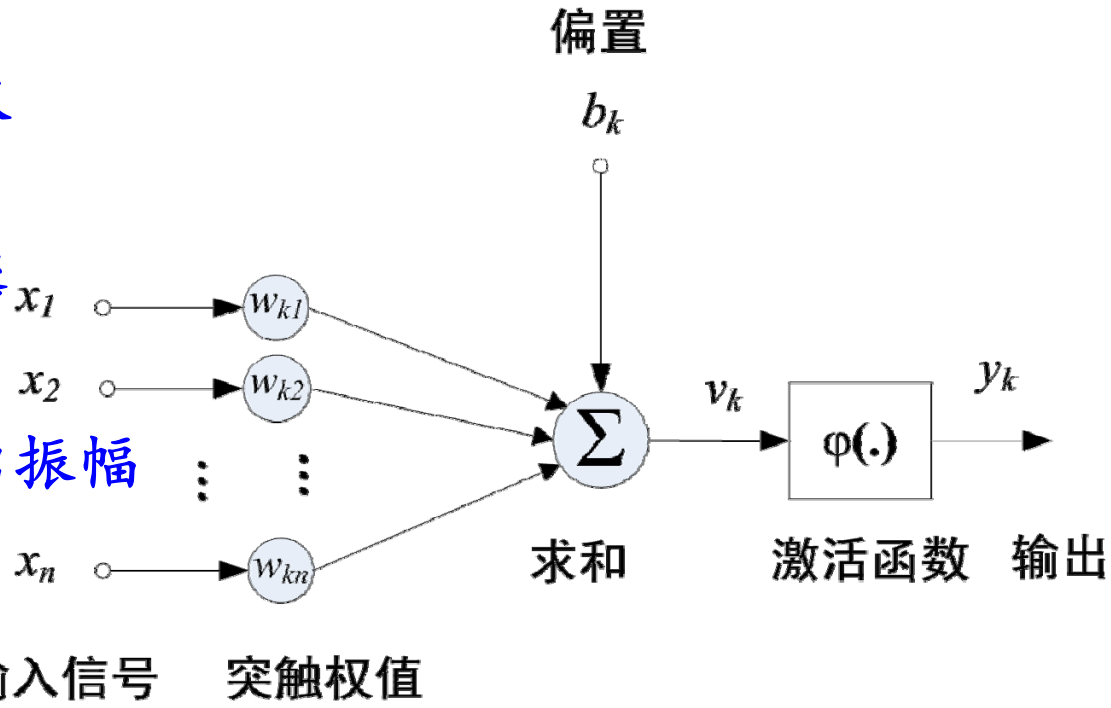


  - **[1] McCulloch W.S. and Pitts W., "A logical calculus of the ideas immanent in nervous activity". Bulletin of Mathematical Biophysics, vol.5, 1943, pp.115-133.**

- 标志神经网络和人工智能学科的诞生

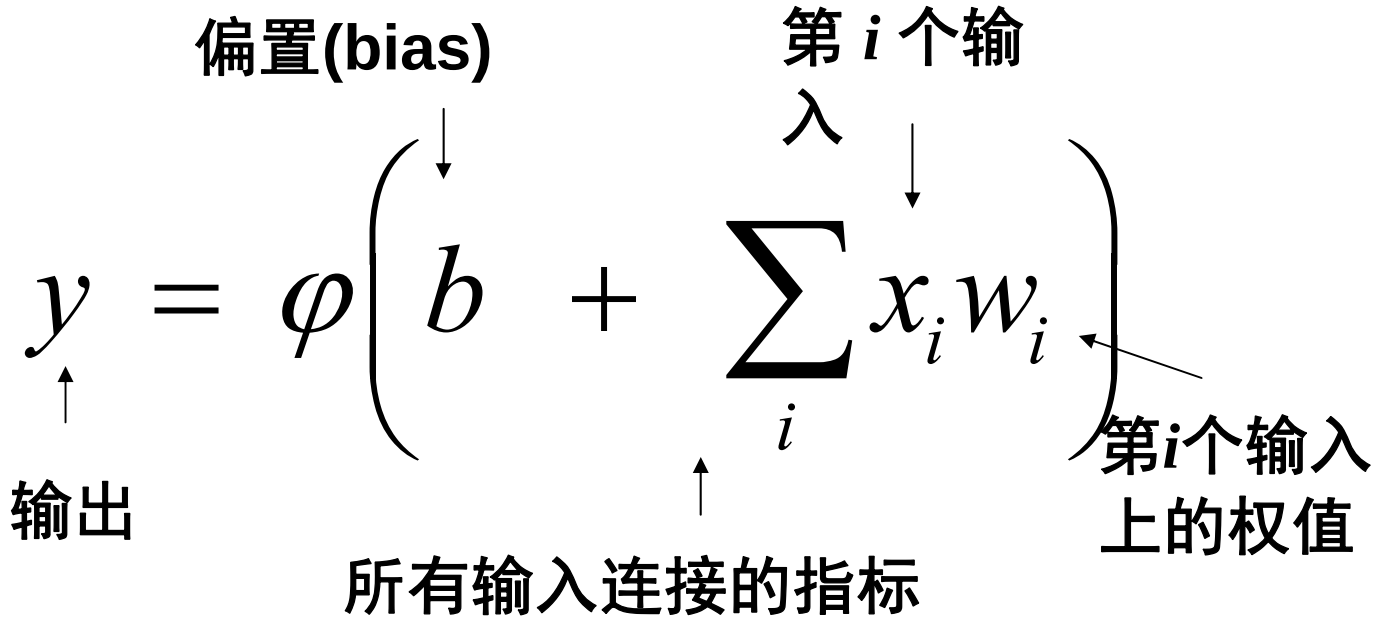
人工神经元模型

- 人工神经元是神经网络的基本信息处理单位
  - 突触权值：
    - 对输入信号加权
  - 加法器
    - 构成线性组合器
  - 激活函数
    - 限制神经元输出振幅



人工神经元模型的数学表达

- **数学表达**



- **神经元的排列和突触的强度确立了神经网络的结构**
  - 突触单方向的传递信息，且强度可变、具有学习功能

人工神经元模型的数学表达

- 数学表达

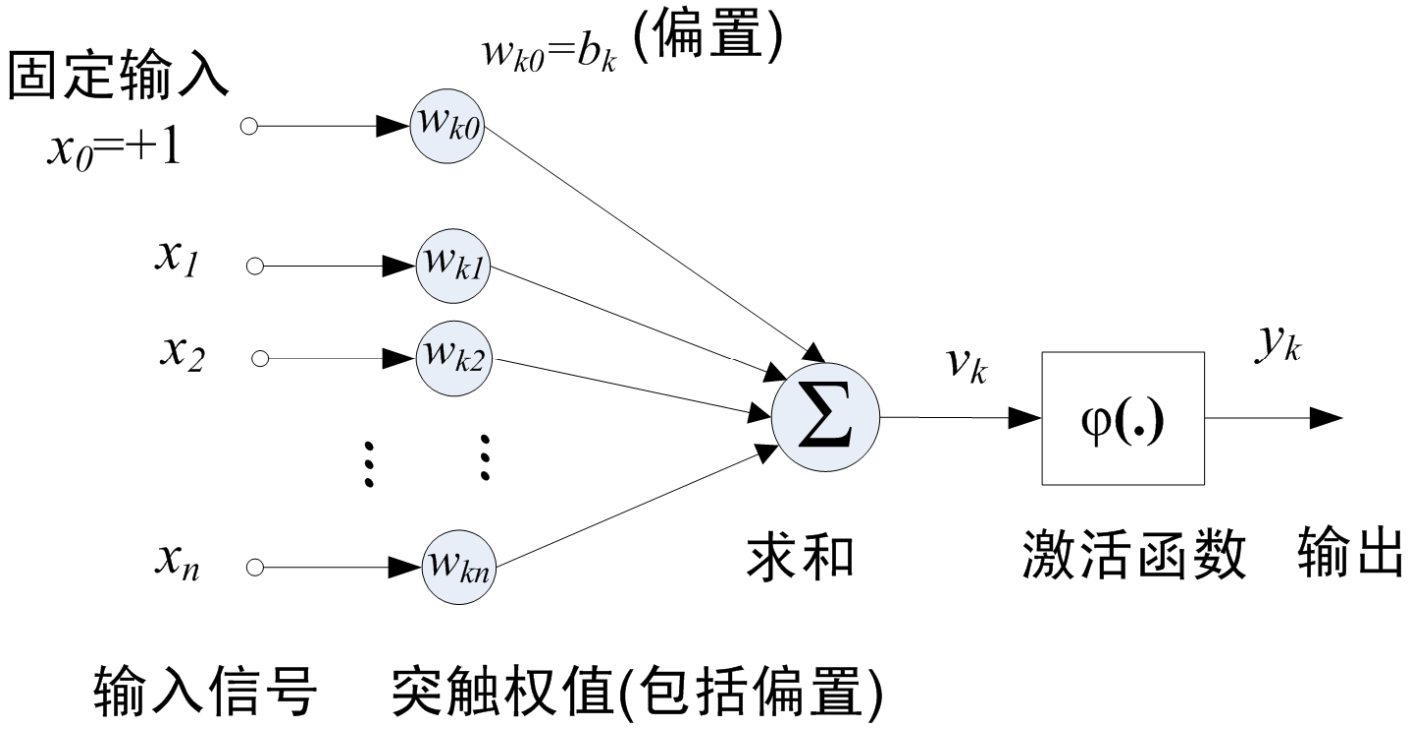
$$y_k = \varphi\left(\sum_{i=0,1,\ldots,n} x_i w_{ki} \right)$$

↑

所有输入连接的指标, 从0开始

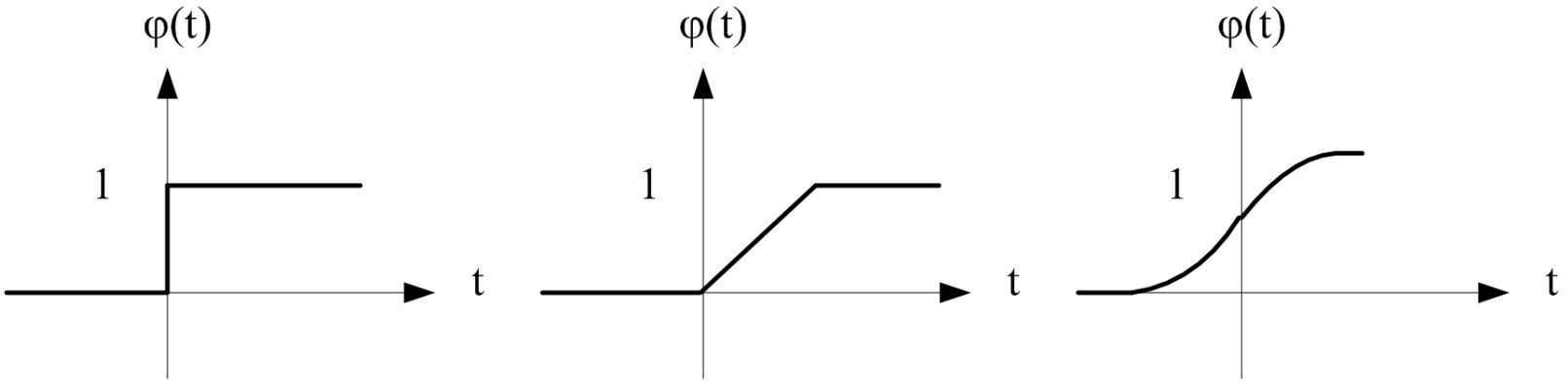
人工神经元模型

- 无外部偏置的：



激活函数

- 阶跃函数或符号函数(sign)
  - **McCulloch-Pitts模型**
- 分段线性函数
- Sigmoid函数



(a) 阶跃函数　(b) 分段线性函数　c) Sigmoid函数

Sigmoid函数的例子

- 严格递增函数
- 典型例子：
  - **logistic函数：** $\varphi(t)=\dfrac{1}{1+\exp(-a\cdot t)}$
  - **a 是sigmoid函数的倾斜参数，原点处斜率为a/4**
  - 可微分， 其导数为：

$$\varphi'(t)=\frac{a\cdot\exp(-a\cdot t)}{\left(1+\exp(-a\cdot t)\right)^2}=a\cdot\varphi(t)\left(1-\varphi(t)\right)$$

Sigmoid函数的双极值形式

- 可以采用双曲正切函数

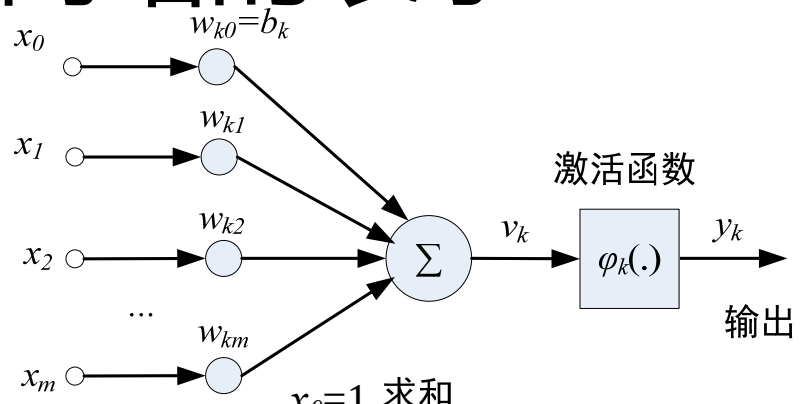
$$\varphi(t)=\tanh(a\cdot t)=\frac{e^{a\cdot t}-e^{-a\cdot t}}{e^{a\cdot t}+e^{-a\cdot t}}$$

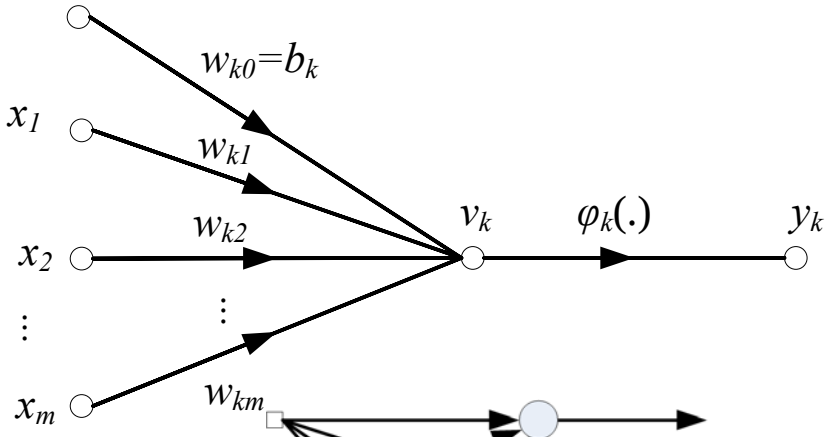
  - **a** 是倾斜参数，原点处斜率为**a**
  - 可微分，其导数为：

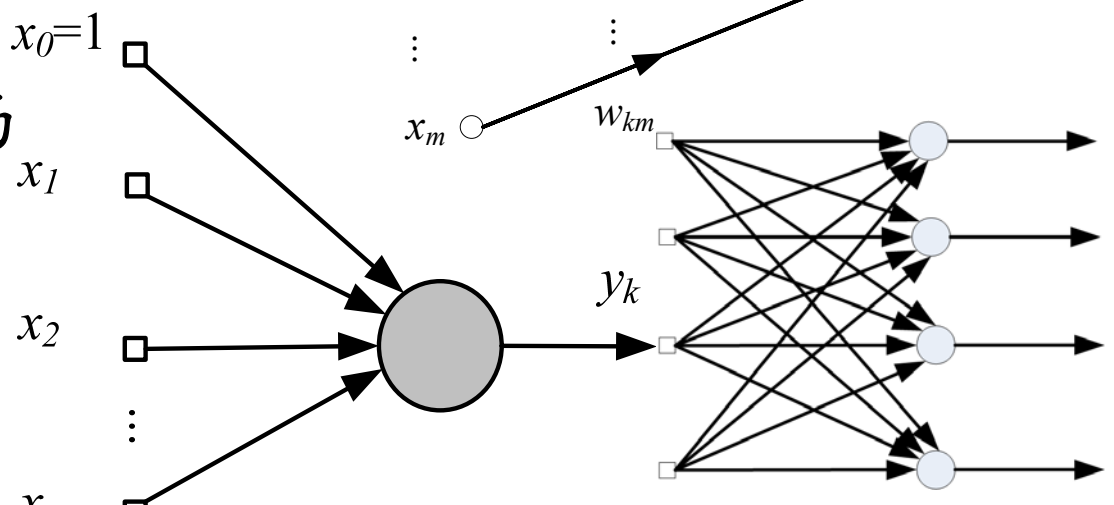
$$\varphi'(t)=a\cdot\left(1-\varphi(t)\right)\left(1+\varphi(t)\right)$$

神经网络的表示

- 框图
  - 提供功能描述
- 信号流图
  - 提供信号流的完备描述
    - 信号沿箭头流动
    - 汇聚，即线性求和
- 体系结构图
  - 描述网络的布局
    - 输入节点
    - 计算节点







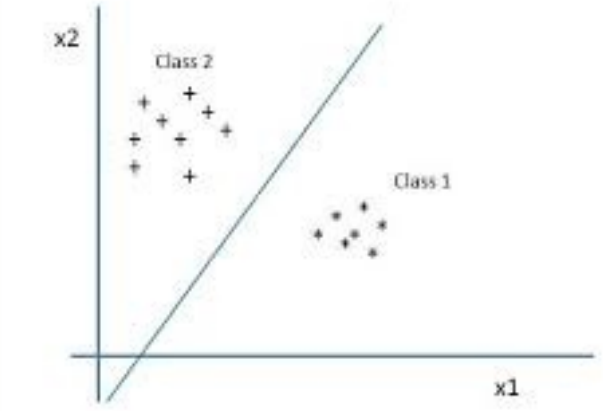
神经网络 内容提要

Frank Rosenblatt

- 生物神经元
- 人工神经元模型
  - 数学表示、激活函数、网络结构
- 感知器(Perceptron)
  - 感知器学习算法(误差修正法则)
- 单层感知网络
  - 误差修正法则
- 多层感知器(Multi-Layer Perceptron)
  - 反传(BP)算法

线性分类

- 寻找一个线性判别函数$g(x) = w^T x + b$，对于来自类别C_1 和 C_2 的样本z
  - 如果 **g(x) > 0,** 我们把**x** 判定为类别**1**
  - 如果 **g(x) < 0,** 我们把**x**判定为类别**2**



感知器 (Perceptron)

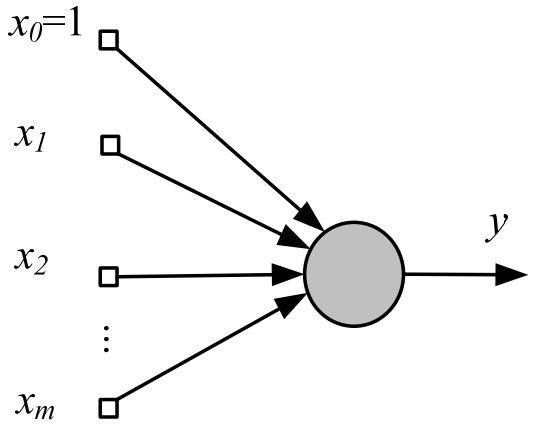
- 建立在1个M-P神经元基础上
  - **Frank Rosenblatt 在1958年第一次给出面向计算的神经网络**
    - **@ Cornell Aeronautical Laboratory (1957-1959)**

  - 主要贡献:
    - 在算法上，提出用于解决模式分类问题的神经网络训练规则(或学习规则)
    - 在理论上，证明: 只要求解问题的权值存在，通常会收敛到正确的权值上

[1] Rosenblatt F. The perceptron: probabilistic model for information storage and organization in the brain [J]. Psychological Review,1958,65(6):386-408.

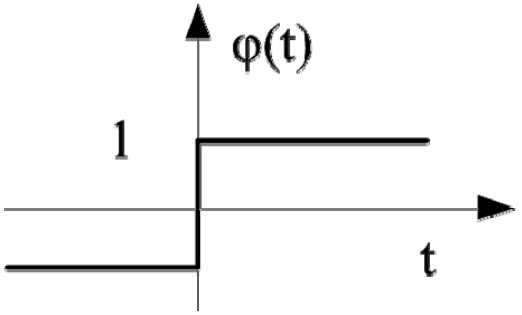
感知器 (Perceptron)

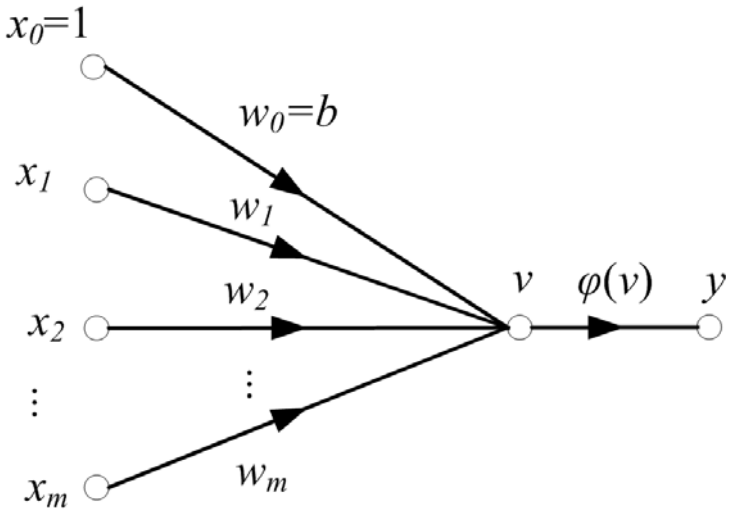
- 即1个MP神经元
  - 非线性神经元



- 数学表达 $y = \varphi\left(\mathbf{w}^T\mathbf{x}\right) = \varphi\left(\sum_{i=0}^{m} w_i x_i\right)$

- 其中 $\varphi(t) = \begin{cases} 1 & t > 0 \\ -1 & t \le 0 \end{cases}$





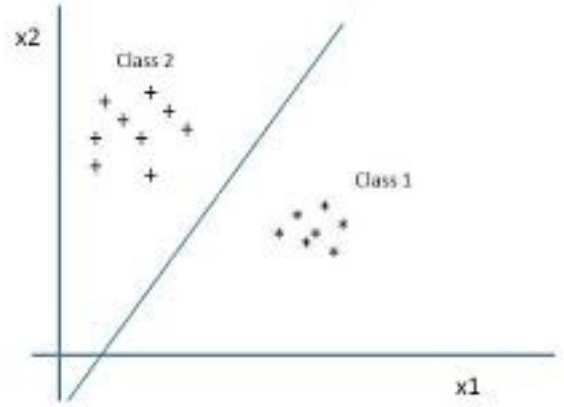
线性可分

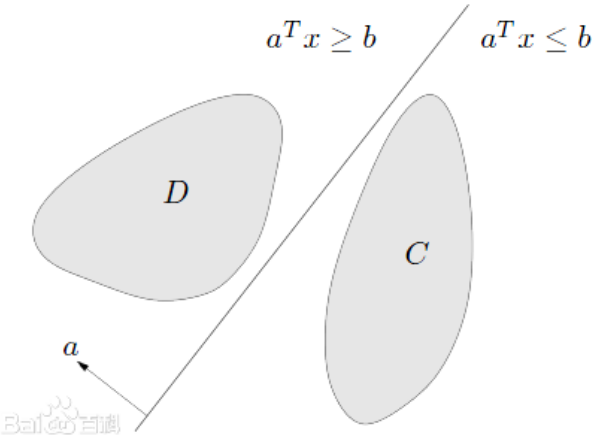
- 两个类别C_1 和 C_2 线性可分
  - 即存在一个权值向量 **w** 满足
    - 对于来自类别 C_1 的输入向量 **x** ：

$$\mathbf{w}^T\mathbf{x} > 0$$

    - 对于来自类别 C_2 的输入向量 **x**：

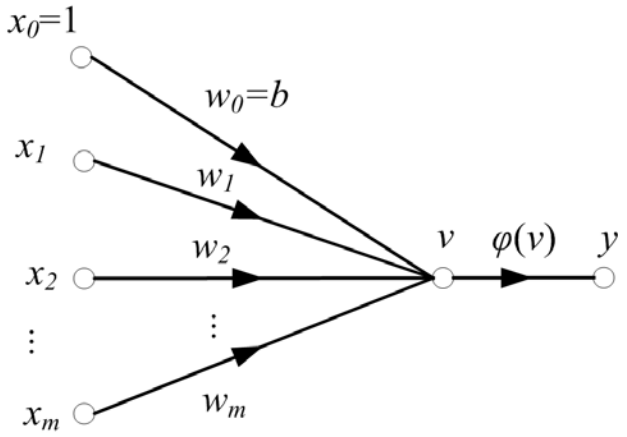
$$\mathbf{w}^T\mathbf{x} \le 0$$





感知器训练算法

- 假设输入数据是线性可分的



- 感知器模型

$$y=\varphi\left(\mathbf{w}^T\mathbf{x}\right)=\operatorname{sgn}\left(\mathbf{w}^T\mathbf{x}\right)$$

- 感知器的训练
  - 更新权值**w**
  - 权值更新规则 $\mathbf{w}\leftarrow\mathbf{w}+\eta\left(y_i-\hat{y}_i\right)\cdot\mathbf{x}_i$
    - 其中

$$\hat{y}_i=\operatorname{sgn}\left(\mathbf{w}^T\mathbf{x}_i\right),\quad \eta>0$$

感知器训练算法

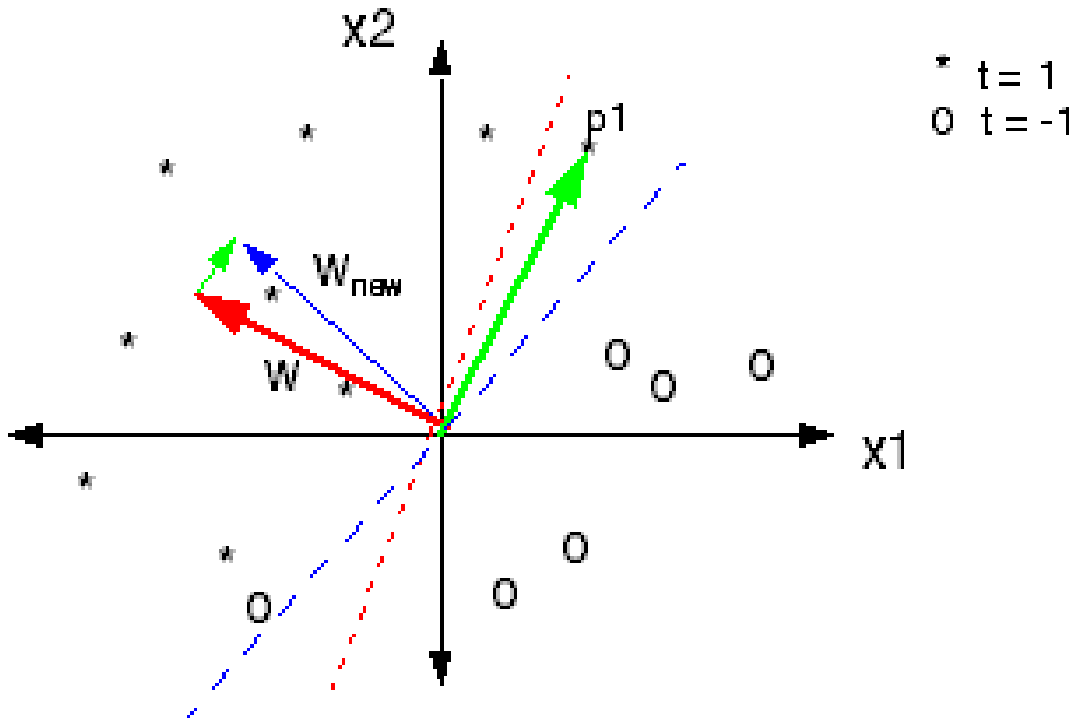
Algorithm 1 Perceptron Algorithm

Input: train data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, learning rate $\eta = 1$
Output: weight vector $\mathbf{w}$
Initiate $\mathbf{w}$ randomly
Repeat
  For $i = 1$ to N
    If $y_i \mathbf{w}^\top \mathbf{x}_i \leq 0$ then
      $\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot y_i \mathbf{x}_i$
    End-If
  End-For
Until no sample is misclassified

感知器训练过程示意图

- 类别1的样本被错分到类别 2： $\mathbf{w} \leftarrow \mathbf{w} + \eta\left(y_i - \hat{y}_i\right) \cdot \mathbf{x}_i$

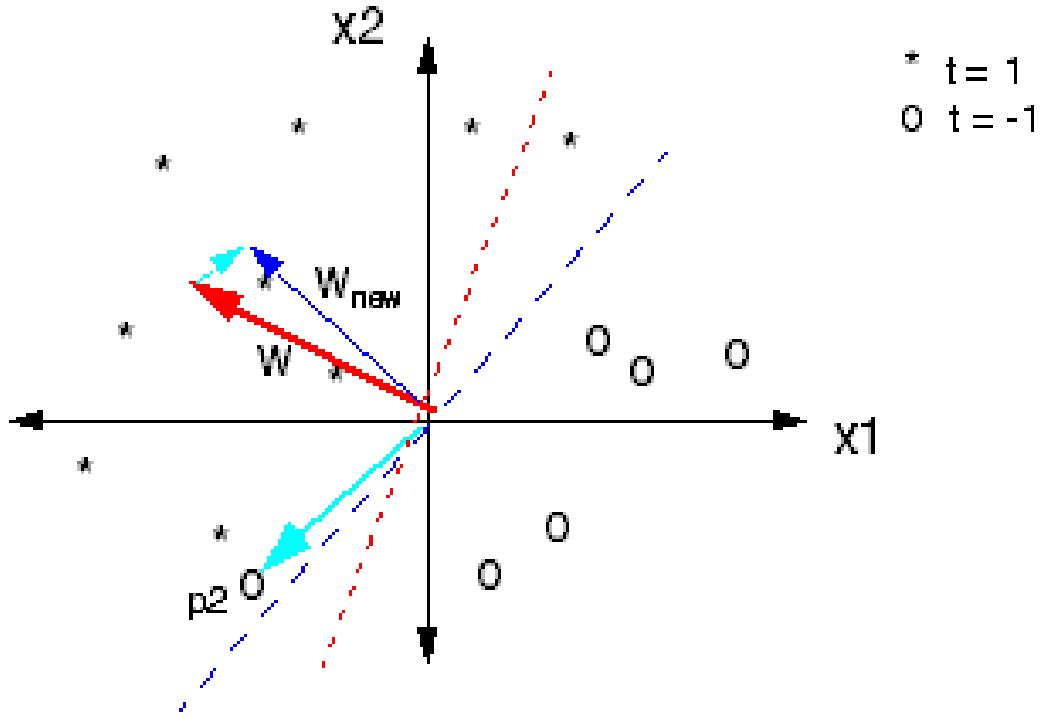
$\eta = 0.5$



感知器训练过程示意图

- 类别2的样本被错分到类别 1： $\mathbf{w} \leftarrow \mathbf{w} + \eta\left(y_i - \hat{y}_i\right) \cdot \mathbf{x}_i$

$\eta = 0.5$



关于感知器的几个问题

- 算法收敛么？
  - 何时收敛？
- 感知器的解唯一么？
  - 最优么？
  - 若不是，那么如何找到最优解？

感知器算法收敛定理

- 定理:
  - 如果训练数据是线性可分的，感知器算法给出的权向量序列必定终止于某个解向量，即

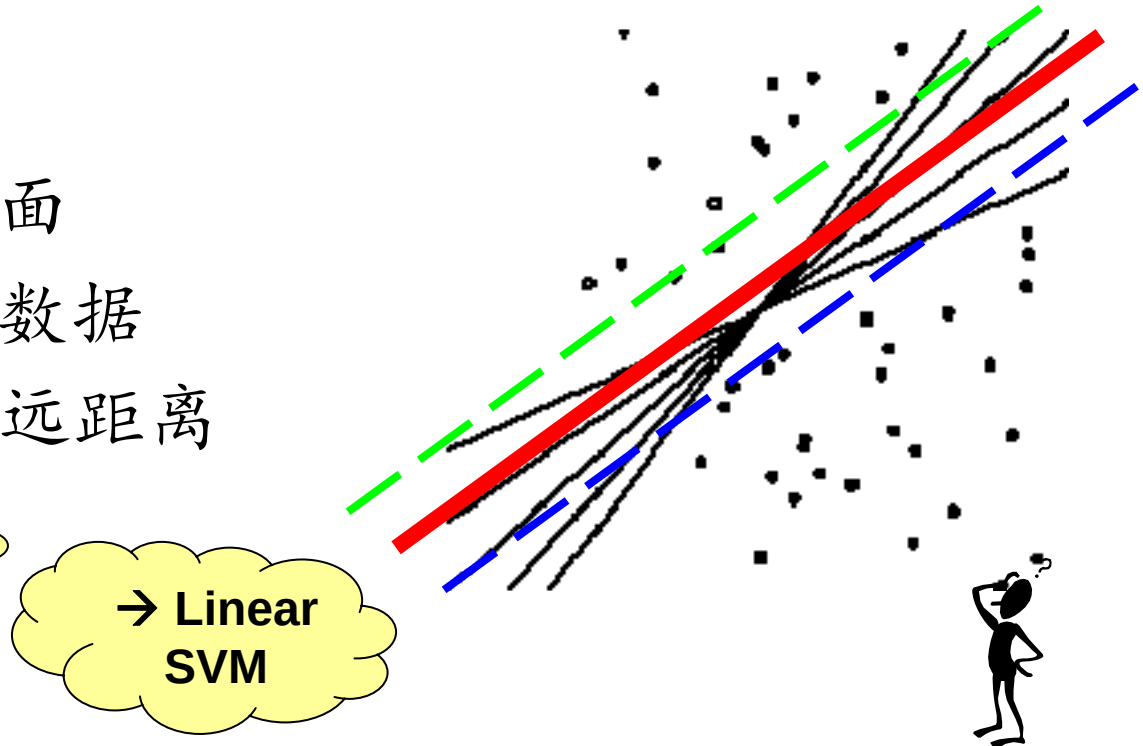
$$\mathbf{w}(n_0)=\mathbf{w}(n_0+1)=\mathbf{w}(n_0+2)=...$$

  - 前提：固定增量，即学习速率参数 $\eta=1$ 固定
    - 证明略.

感知器训练算法得到的解

- 感知器算法得到的解向量w并不唯一
  - 感知器训练算法收敛在任意一个在训练数据能够获得零错误率的向量w

- 最优权值向量
  - 最优决策超平面
  - 与两个类别的数据分布均保持最远距离



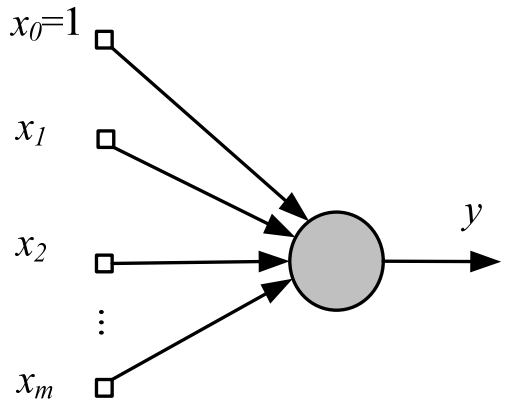
神经网络 内容提要

Frank Rosenblatt

- 生物神经元
- 人工神经元模型
  - 数学表示、激活函数、网络结构
- 感知器(Perceptron)
  - 感知器学习算法(误差修正法则)
- 单层感知网络
  - 误差修正法则
- 多层感知器(Multi-Layer Perceptron)
  - 反传(BP)算法

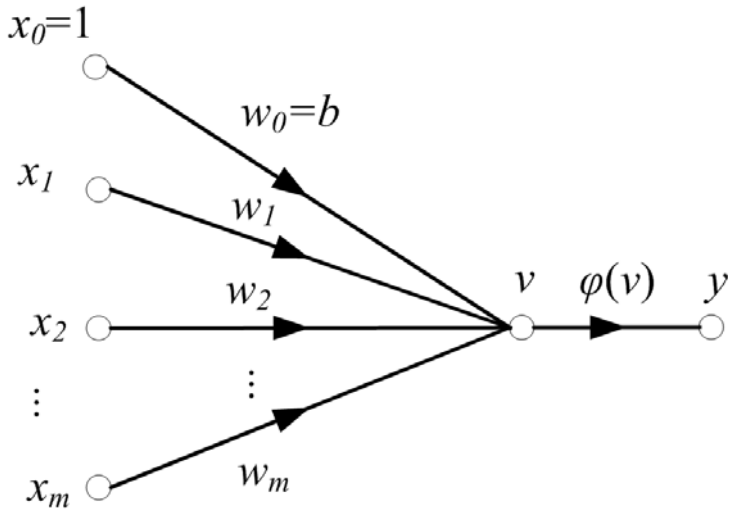
回顾: 感知器模型

- 1个非线性神经元
  - 不可微



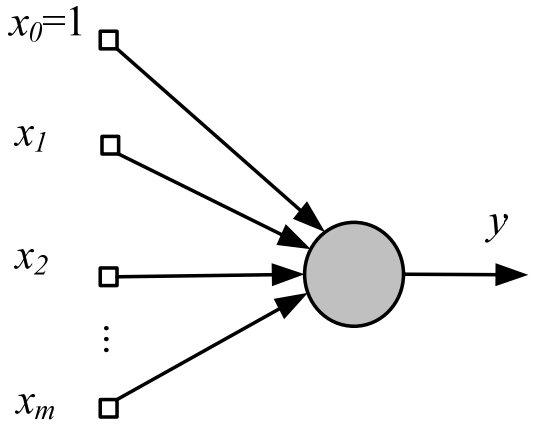
- 数学表达 $y = \varphi\left(\mathbf{w}^T\mathbf{x}\right) = \varphi\left(\sum_{i=0}^{m} w_i x_i\right)$

- 其中 $\varphi(t) = \begin{cases} 1 & t > 0 \\ -1 & t \le 0 \end{cases}$



回顾: 线性神经元模型

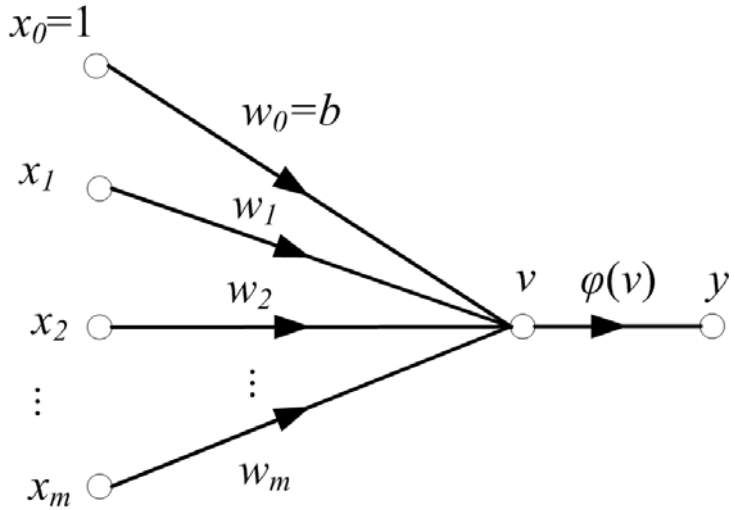
- 1个线性神经元
  - **Adaline**



  - 数学表达

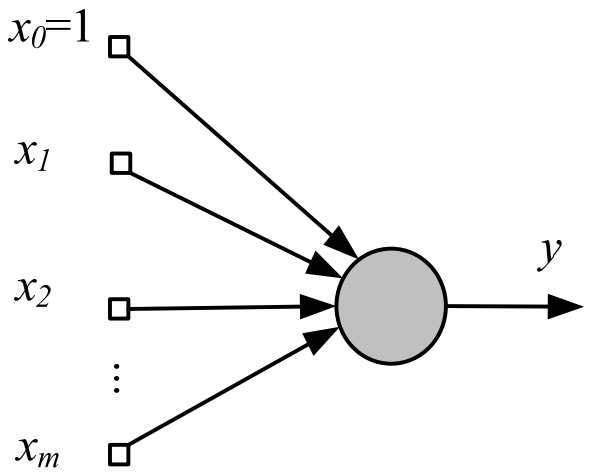
$$y = \varphi\left(\sum_{i=0}^{m} w_i x_i\right) = \varphi\left(\mathbf{w}^T\mathbf{x}\right) = \mathbf{w}^T\mathbf{x}$$

  - 其中 $\varphi(v) = v$



非线性神经元模型

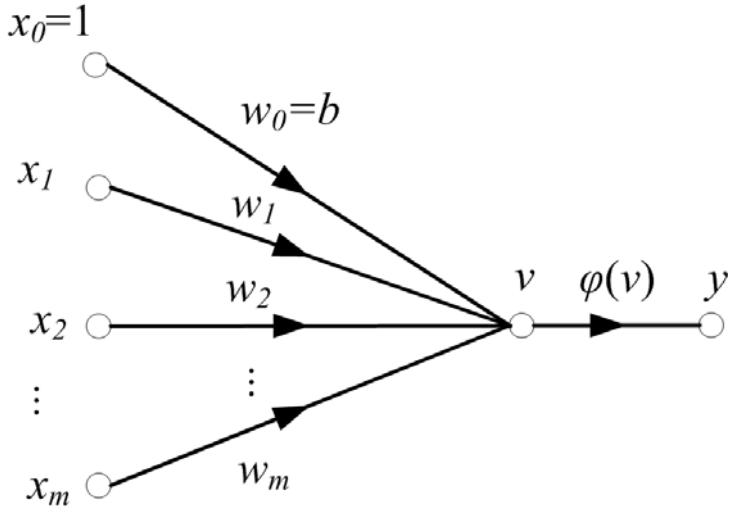
- 1个非线性神经元



– 数学表达

$$y = \varphi\left(\sum_{i=0}^{m} w_i x_i\right) = \varphi\left(\mathbf{w}^T \mathbf{x}\right)$$

- 其中 $\varphi(v) = \dfrac{1}{1+e^{-v}}$



最小均方(LMS)算法

- 最小均方(LMS：Least Mean Square)
  - 定义代价函数为瞬时误差的平方

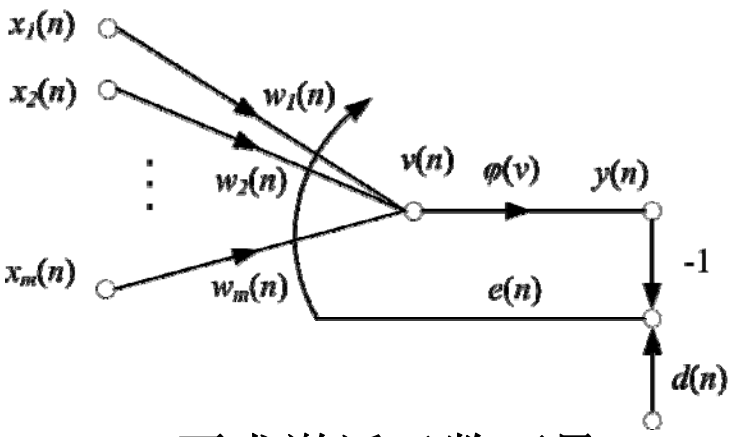
$$\varepsilon(\mathbf{w})=\frac{1}{2}e^2(n)$$ 其中，$e(n)$ 为时刻n 测得的误差

  - 定义权值更新规则：

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\eta\cdot\Delta\mathbf{w}$$

其中，基于最速下降法，则

$$\Delta\mathbf{w}=-\frac{\partial\varepsilon(\mathbf{w})}{\partial\mathbf{w}}=-e(n)\frac{\partial e(n)}{\partial\mathbf{w}}$$

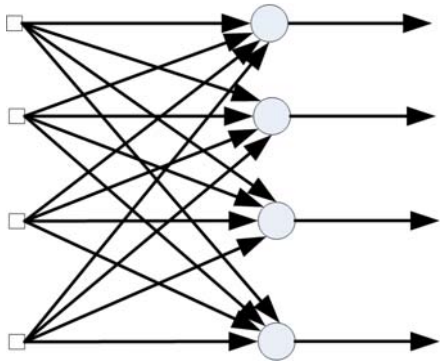


- 要求激活函数可导

- 权值更新特点：
  - 每个训练样本呈现之后，即进行权值更新
    - 称为串行方式、在线方式、随机方式

单层感知器网络

- 网络结构:
  - 基于sigmoid激活函数的非线性神经元构成单层感知器网络
    - 多个非线性神经元



- 训练算法
  - 最小均方算法
    - 基于LMS算法分别给出各个神经元的权值更新规则

$$w_{kj}(n+1) \leftarrow w_{kj}(n) + \Delta w_{kj}(n)$$

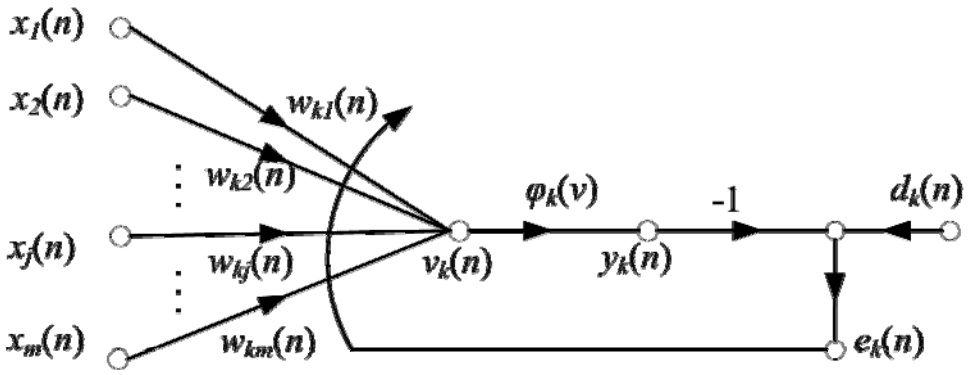
单层感知器网络

- **定义代价函数**
  - 串行方式：$\varepsilon(\mathbf{w}) = \frac{1}{2}\sum_{k\in C} e_k^2(n)$
    - C：输出层的所有神经元

$$\Delta w_{kj}(n) = -\frac{\partial \varepsilon(\mathbf{w})}{\partial w_{kj}(n)} = -\frac{\partial \varepsilon(\mathbf{w})}{\partial e_k(n)}\frac{\partial e_k(n)}{\partial w_{kj}(n)} = -e_k(n)\frac{\partial e_k(n)}{\partial w_{kj}(n)}$$

  - 集中方式：

$$\varepsilon_{av}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\sum_{k\in C} e_k^2(n)$$

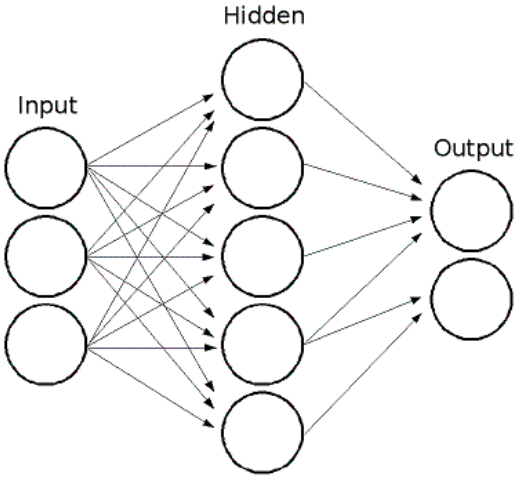


神经网络 内容提要

- 生物神经元
- 人工神经元模型
  - 数学表示、激活函数、网络结构
- 感知器(Perceptron)
  - 感知器学习算法(误差修正法则)
- 单层感知网络
  - 误差修正法则
- 多层感知器(Multi-Layer Perceptron)
  - 反传(BP)算法

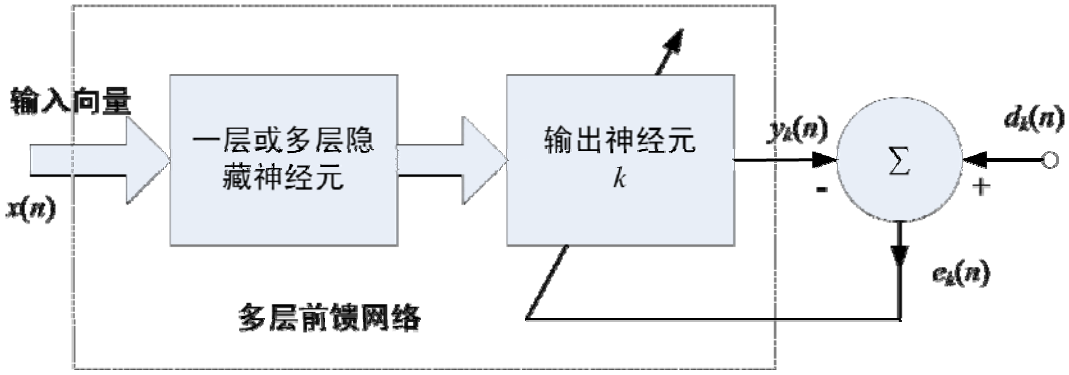
从单层网络到多层网络

- **多层感知器**
  - 如何给出权值更新规则？



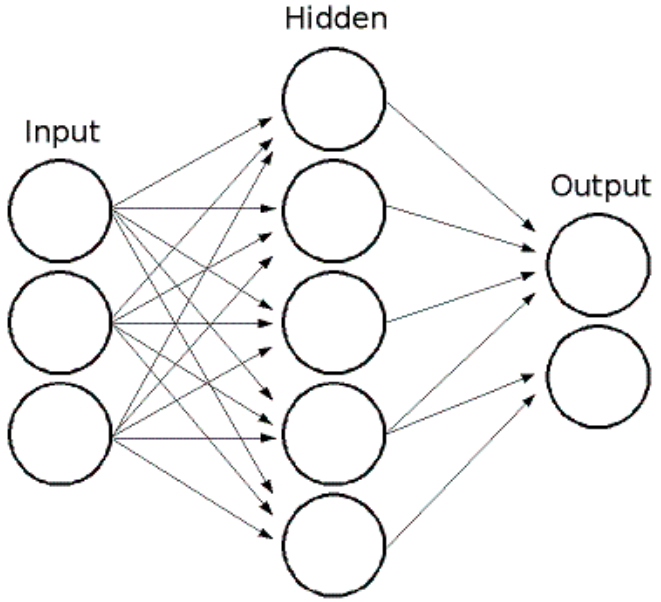
- **误差-修正学习的前提**
  - 误差信号是直接可以测量的
    - 在单层感知器中误差信号是直接可以测量的
    - 在多层感知器中， 隐层神经元是不可见的，其误差信号无法直接测量

误差反向传播算法



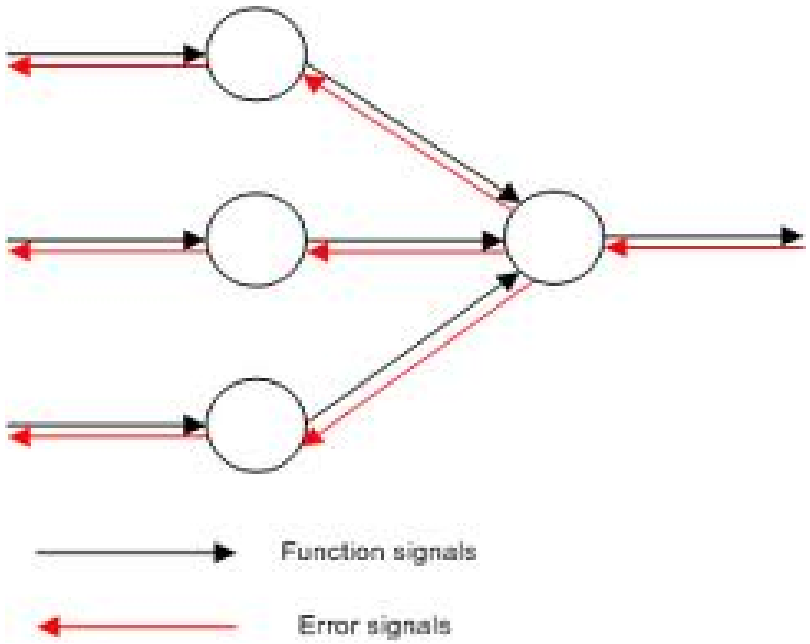
多层感知器

- 多层感知器(MLP)3个特点：
  - 非线性激活函数
    - 非线性且处处可微
      - **Logistic**函数 $\varphi(t)=\dfrac{1}{1+\exp(-a\cdot t)}$
      - 双曲正切 $\varphi(t)=\tanh(a\cdot t)=\dfrac{e^{a\cdot t}-e^{-a\cdot t}}{e^{a\cdot t}+e^{-a\cdot t}}$
  - 包括隐藏层神经元
  - 高度连通性



多层感知器中的两种信号与计算

- 网络中传递的两种信号
  - 函数信号:
    - 是输入和权值的函数
    - 来自于输入端，到达输出端
  - 误差信号
    - 误差以及误差的函数
    - 产生于输出端，反向传播



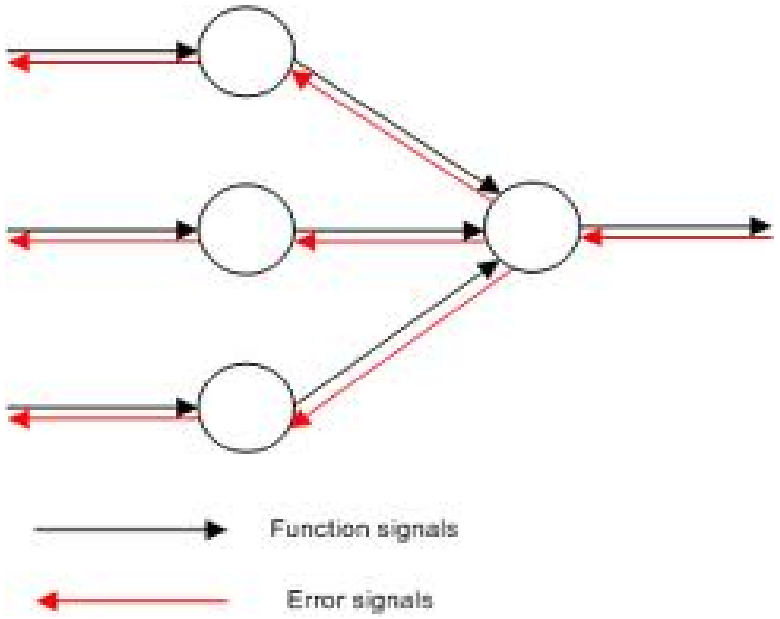
- 每个神经元完成两种计算
  - 计算输出函数信号
    - 输入信号和权值的连续非线性函数
  - 计算梯度向量
    - 误差曲面对于权值的梯度的估计

反向传播算法的基本思想

- **反向传播(BP：back-propagation)**
  - 反向传播的是“误差信号”

Rumelhart, Hinton & Williams, “Learning representations by back-propagating errors”, Nature, 1986

- **两个计算过程：**
  - 信号的前向通过
    - 产生网络的实际响应
    - 权值固定
  - 误差的反向通过
    - 提供修正权值的“误差”信号
    - 权值被调整



反向传播算法的两种工作方式

- 串行方式
  - 定义代价函数：$\varepsilon(n)=\frac{1}{2}\sum_{j\in C}e_j^2(n)$
  - 提交一个样本，进行一次前向运算，再进行一次反向运算
    - 每提交1个样本，更新一次权值
- 集中方式
  - 定义代价函数：$\varepsilon_{av}=\frac{1}{2N}\sum_{n=1}^{N}\varepsilon(n)=\frac{1}{2N}\sum_{n=1}^{N}\sum_{j\in C}e_j^2(n)$
  - 提交一个样本，进行一次前向运算，等全部N个样本都提交完毕再进行反向运算
    - 提交全部N个样本，再更新一次权值

$$\Delta w_{ji}(n)=-\eta\cdot\partial\varepsilon_{av}(n)/\partial w_{ji}(n)=-\frac{\eta}{N}\sum_{i=1}^{N}e_j(n)\frac{\partial e_j(n)}{\partial w_{ji}(n)}$$

网络中几个信号的定义

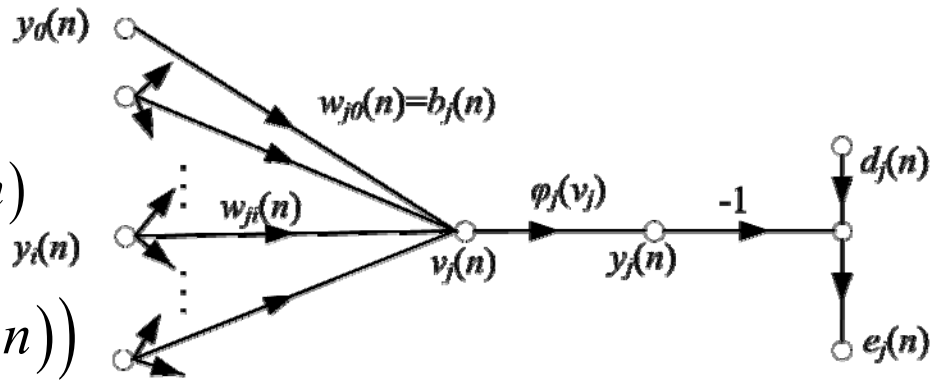
- 误差信号定义： $e_j(n)=d_j(n)-y_j(n)$
  - 如果神经元j是输出节点，则误差可直接测量

$$\varepsilon_j(n)=\frac{1}{2}e_j^2(n)$$

- “可见”误差的瞬时能量 $\varepsilon(n)=\frac{1}{2}\sum_{j\in C}e_j^2(n)$
  - **C:** 输出层的所有神经元

这里介绍串行方式

- 均方误差能量： $\varepsilon_{av}=\frac{1}{N}\sum_{n=1}^{N}\varepsilon(n)$
  - 对所有n，对瞬时能量求和
- 诱导局部域：$v_j(n)=\sum_{i=0}^{m}w_{ji}(n)y_i(n)$
  - 神经元j的激活函数输入处
- 输出的函数信号：$y_j(n)=\varphi_j(v_j(n))$
  - 神经元j输出处的函数信号



- 突触权值的修正：
  - 权值修正量正比于偏导 $w_{ji}(n+1)=w_{ji}(n)+\Delta w_{ji}(n)$

$$\Delta w_{ji}(n)\propto\partial\varepsilon(n)/\partial w_{ji}(n)$$

偏导怎样计算？

串行方式工作的反向传播算法

- 误差信号定义： $e_j(n)=d_j(n)-y_j(n)$ $\varepsilon_j(n)=\frac{1}{2}e_j^2(n)$
  - 神经元j是输出节点
- “可见”误差的瞬时能量 $\varepsilon(n)=\frac{1}{2}\sum_{j\in C}e_j^2(n)$
  - **C:** 输出层的所有神经元
- 诱导局部域：

$$v_j(n)=\sum_{i=0}^{m}w_{ji}(n)y_i(n)$$

- 输出的函数信号：

$$y_j(n)=\varphi_j\left(v_j(n)\right)$$

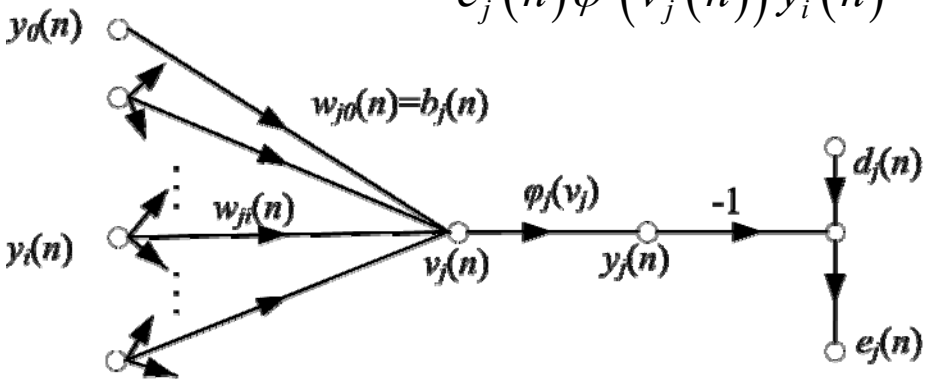
- 突触权值的修正：
  - 权值修正量正比于偏导

$$\Delta w_{ji}(n)\propto\partial\varepsilon(n)/\partial w_{ji}(n)$$

偏导的计算：

$$\frac{\partial\varepsilon(n)}{\partial w_{ji}(n)}=\frac{\partial\varepsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

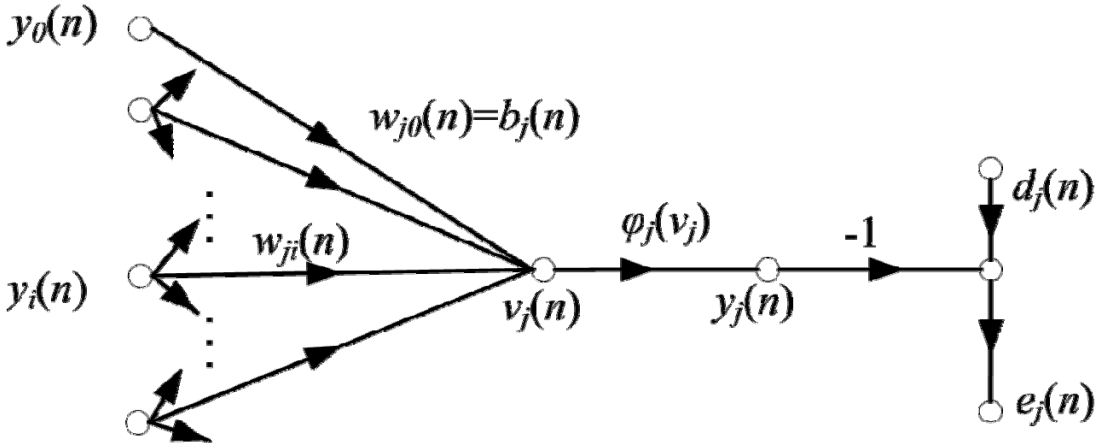
$$=-e_j(n)\varphi'\left(v_j(n)\right)y_i(n)$$



1. 输出层神经元的权值修正法则

- 权值修正的delta法则 $w_{ji}(n+1)=w_{ji}(n)+\Delta w_{ji}(n)$
  - 其中 $\Delta w_{ji}(n)=-\eta\cdot\partial\varepsilon(n)/\partial w_{ji}(n)$
  - 即 $\dfrac{\partial\varepsilon(n)}{\partial w_{ji}(n)}=\dfrac{\partial\varepsilon(n)}{\partial e_j(n)}\dfrac{\partial e_j(n)}{\partial y_j(n)}\dfrac{\partial y_j(n)}{\partial v_j(n)}\dfrac{\partial v_j(n)}{\partial w_{ji}(n)}=-e_j(n)\varphi'\left(v_j(n)\right)y_i(n)$

$$w_{ji}(n+1)=w_{ji}(n)+\eta e_j(n)\varphi_j'\left(v_j(n)\right)y_i(n)$$



反向传播算法？哪里有反向传播的影子。。。

隐藏层的权值如何调整？

其误差信号如何计算？

定义神经元的局部梯度

- 神经元局部梯度的定义： $\delta_j(n) = -\dfrac{\partial \varepsilon(n)}{\partial v_j(n)}$
  - 对于输出层神经元j

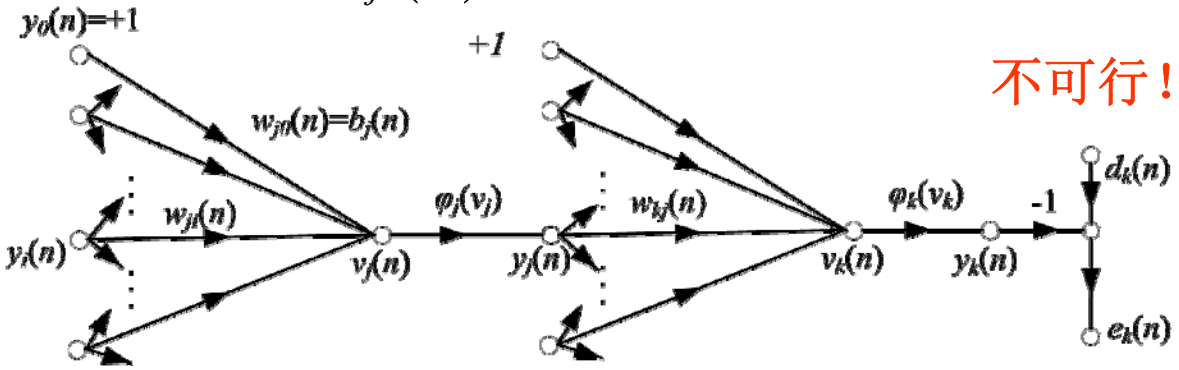
$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial v_j(n)} = -\frac{\partial \varepsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n))$$

$$\Rightarrow \Delta w_{ji}(n) = \eta e_j(n)\varphi_j'(v_j(n))y_i(n) = \eta \delta_j(n) y_i(n)$$

  - 对于隐藏层神经元j

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial v_j(n)} = -\frac{\partial \varepsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}$$

如此链导可行么？



不可行！ 没有$e_j(n)$

2. 隐藏层神经元的局部梯度计算

- 神经元局部梯度的定义
  - 对于隐藏层神经元j $\delta_j(n)=-\frac{\partial\varepsilon(n)}{\partial v_j(n)}=-\frac{\partial\varepsilon(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}=-\frac{\partial\varepsilon(n)}{\partial y_j(n)}\varphi_j'(v_j(n))$
  - 其中 $\varepsilon(n)=\frac{1}{2}\sum_{k\in C}e_k^2(n),\ e_k(n)=d_k(n)-y_k(n),\ y_k(n)=\varphi_k(v_k(n)),$
  - 于是 $v_k(n)=\sum_{j=0}^{m}w_{kj}(n)y_j(n)$

$$\frac{\partial\varepsilon(n)}{\partial y_j(n)}=\sum_{k\in C}e_k(n)\frac{\partial e_k(n)}{\partial y_j(n)}=\sum_{k\in C}e_k(n)\frac{\partial e_k(n)}{\partial y_k(n)}\frac{\partial y_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)}$$

$$=\sum_{k\in C}e_k(n)\left(-\varphi_k'(v_k(n))\right)w_{kj}(n)=-\sum_{k\in C}\delta_k(n)w_{kj}(n)$$

$$=-\tilde{e}_j(n)$$ 神经元j应该承担的“误差信号”

其中：

$$\delta_k(n)=e_k(n)\varphi_k'(v_k(n))$$

$$\delta_j(n)=-\frac{\partial\varepsilon(n)}{\partial v_j(n)}=-\frac{\partial\varepsilon(n)}{\partial y_j(n)}\varphi_j'(v_j(n))=\tilde{e}_j(n)\varphi_j'(v_j(n))$$

误差信号分析与权值更新

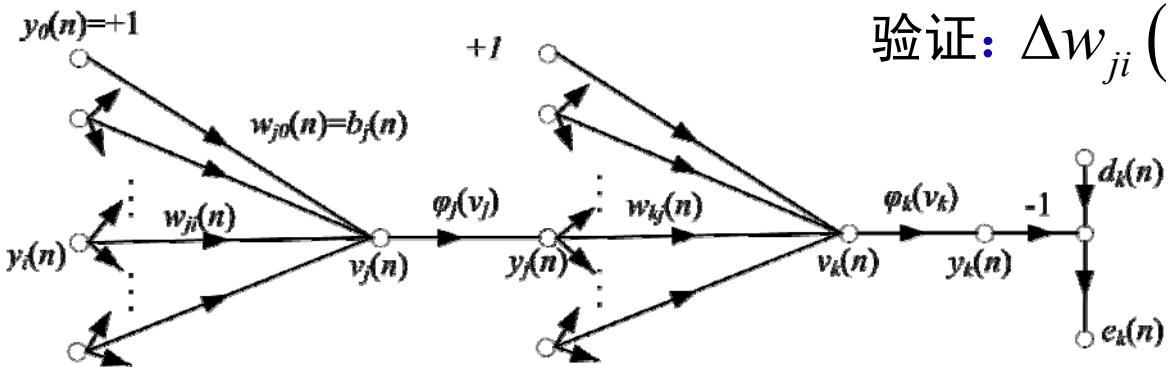
- 误差信号分析与权值更新法则：
  - 输出节点**j**：$e_j(n)$ ➡ $\delta_j(n)=e_j(n)\varphi_j'\left(v_j(n)\right)$

    ➡ $\Delta w_{ji}(n)=\eta e_j(n)\varphi_j'\left(v_j(n)\right)y_i(n)=\eta\delta_j(n)y_i(n)$
  - 隐藏层节点**j**：$\tilde{e}_j(n)=-\sum_{k\in C}\delta_k(n)w_{kj}(n)$

    隐藏层的误差信号需通过所有与隐藏层直接相连的神经元的误差递归计算

    ➡ $\delta_j(n)=\tilde{e}_j(n)\varphi_j'\left(v_j(n)\right)$

    ➡ $\Delta w_{ji}(n)=\eta e_j(n)\varphi_j'\left(v_j(n)\right)y_i(n)=\eta\delta_j(n)y_i(n)$



验证：$\Delta w_{ji}(n)=-\eta\cdot\partial\varepsilon(n)/\partial w_{ji}(n)$

$$=-\eta\frac{\partial\varepsilon(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

权值修正的delta规则

- 由神经元i连接到神经元j的权值的校正值由delta规则定义如下：

学习速率参数

$$\Delta w_{ji}(n) = \eta \cdot \delta_j(n) \cdot y_i(n)$$

权值校正　　局部梯度　　神经元j的输入信号

- 局部梯度取决于**神经元j**是一个输出层节点还是隐藏层节点
  - 神经元j是输出层节点：$\delta_j(n) = \varphi_j'(v_j(n))e_j(n)$
  - 神经元j是隐藏层节点：$\delta_j(n) = \varphi_j'(v_j(n))\sum_{k\in C}\delta_k(n)w_{kj}(n)$

$$= \varphi_j'(v_j(n))\tilde{e}_j(n)$$

误差信号反向传播的过程

- 误差信号反向传播信号流图
  - 输出层 $\delta_k(n)=e_k(n)\left(\varphi_k'\left(v_k(n)\right)\right)$

隐藏层神经元j应该承担的“误差信号”

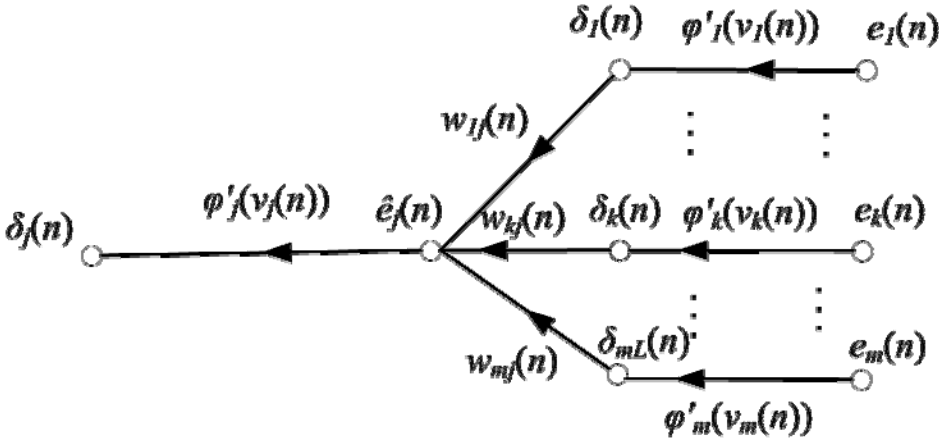
- 隐藏层 $\delta_j(n)=-\frac{\partial\varepsilon(n)}{\partial v_j(n)}=-\frac{\partial\varepsilon(n)}{\partial y_j(n)}\varphi_j'\left(v_j(n)\right)=\varphi_j'\left(v_j(n)\right)\tilde{e}_j(n)$

其中： $\tilde{e}_j(n)=\sum_{k\in C}\delta_k(n)w_{kj}(n)$

隐藏层无法直接测量误差，但可以“承担”输出误差

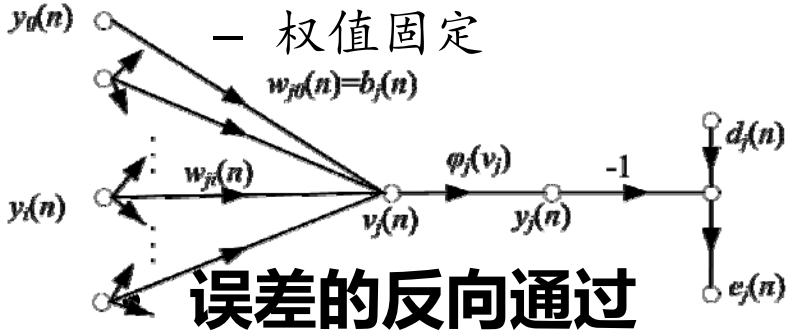
- 隐藏层误差信号涉及：
  - 对所有与隐藏层直接相连的神经元k的局部梯度加权求和

$$\tilde{e}_j(n)=\sum_{k\in C}\delta_k(n)w_{kj}(n)$$



BP算法中的两个计算过程

- **信号的前向通过**
  - 产生网络的实际响应 $y_j(n)=\varphi_j\left(v_j(n)\right)$ $v_j(n)=\sum_{i=0}^{m} w_{ji}(n)\,y_i(n)$
  - 权值固定

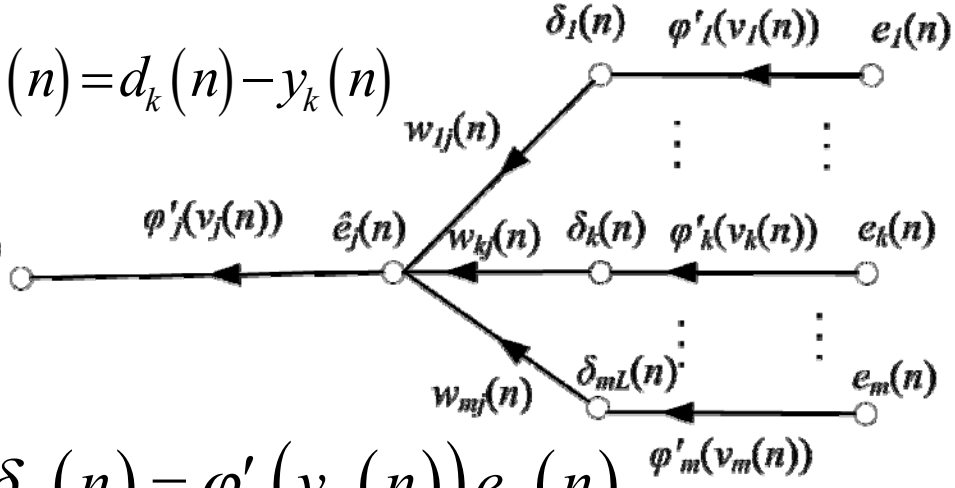


若i在输入层：$y_i(n)=x_i(n)$

若j在输出层：$y_j(n)=o_j(n)$

- **误差的反向通过**
  - 提供修正权值的“误差”信号 $e_k(n)=d_k(n)-y_k(n)$
  - 权值调整

$$\Delta w_{ji}(n)=\eta\cdot\delta_j(n)\cdot y_i(n)$$



局部梯度递归计算

1. 神经元j在输出层：$\delta_j(n)=\varphi'_j\left(v_j(n)\right)e_j(n)$

2. 神经元j在隐藏层：$\delta_j(n)=\varphi'_j\left(v_j(n)\right)\tilde{e}_j(n)$

串行方式BP算法小结

- **给定训练数据集** $\{(\mathbf{x}(n),\mathbf{d}(n))\}_{n=1}^{N}$

1. 初始化：随机选择权值和阈值(均匀分布)
2. 训练样本呈现：每次随机地提交1个训练样本给网络
3. 前向计算：
   - 在第 l 层的神经元 j 的诱导局部域为： $v_j^{(l)}(n)=\sum_{i=0}^{m} w_{ji}^{(l)}(n)\,y_i^{(l-1)}(n)$
   - 神经元 j 的输出信号为：

$$y_j^{(l)}(n)=\varphi_j\left(v_j^{(l)}(n)\right),\ y_j^{(0)}(n)=x_j(n),\ y_j^{(L)}(n)=o_j(n)$$

   - 计算误差信号：

$$e_j(n)=d_j(n)-o_j(n)$$

4. 反向计算：
   - 计算局部梯度：

$$\delta_j^{(l)}(n)=\begin{cases} e_j^{(l)}(n)\varphi_j'\left(v_j^{(L)}(n)\right) & j\in output\ \ L \\ \varphi_j'\left(v_j^{(l)}(n)\right)\sum_{k\in C}\delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n) & j\in hidden\ \ l \end{cases}$$

   - 调整权值： $w_{kj}^{(l)}(n+1)=w_{kj}^{(l)}(n)+\eta\cdot\delta_j^{(l)}(n)\,y_i^{(l-1)}(n)$
5. 若不满足停止条件，呈现训练样本继续前向和反向计算

停止准则

- **收敛准则——1**
  - 梯度向量的欧氏范数足够小
    - 驻点：即误差曲面对权值向量的梯度向量为零向量
- **收敛准则——2**
  - 当每个回合的均方误差的变化的绝对速率足够小
    - 一个回合(epoch): 训练集中N个样本每个都依次呈现一次
- **收敛准则——3**
  - 在每次学习迭代后，检查网络的泛化性能
    - 反向传播学习的本质：把输入/输出映射编码为多层感知器的突触权值和阈值
    - 学习过程：对该数据集合给出网络参数化的一个选择；网络训练问题可以看作候选模型的选择问题
    - 通过交叉确认选择：
      - 把可用数据集随机划分为：训练集与测试集
      - 把训练集划分为：估计子集与确认子集
      - 在估计子集中的数据训练网络，用确认子集评价模型的性能，用测试集估计模型的泛化性能

深度学习(Deep Learning)

- 3篇标志性工作
  - 深度置信网络**(DBNs)**
    - **2006：Hinton’s revolutionary work on Deep Belief Networks (DBNs)**

Hinton et al., “A fast learning algorithm for deep belief nets”, Neural Computation, 18:1527-1554, 2006.

  - 自编解码网络
    - **2006：encoder-decoder**

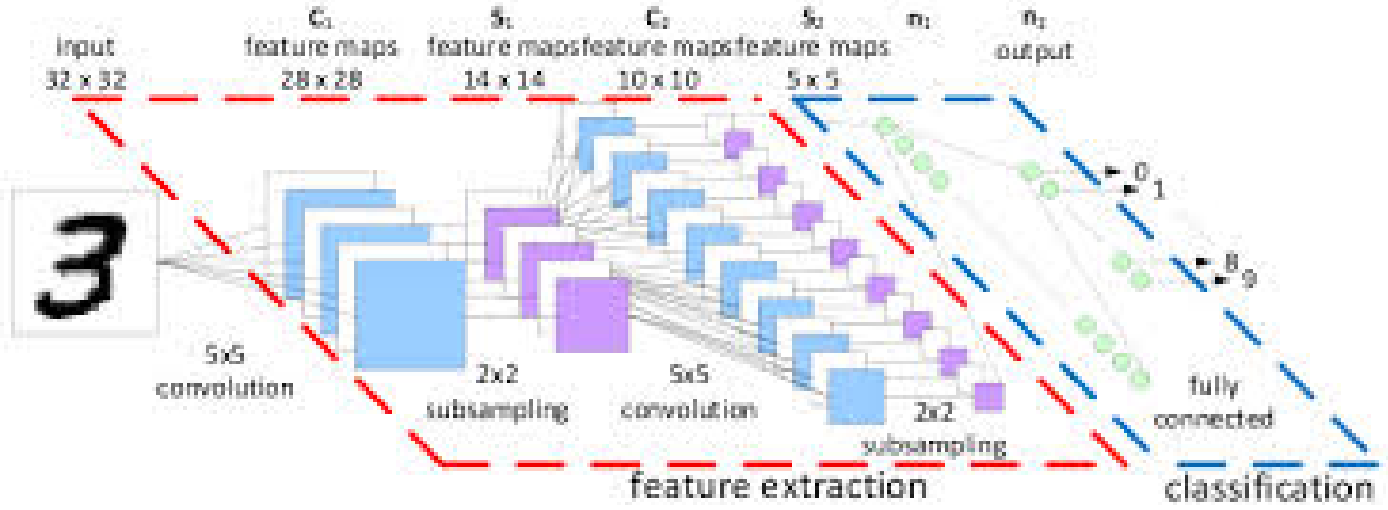
Yoshua Bengio et al., “Greedy Layer-Wise Training of Deep Networks”, NIPS 2006.

  - 稀疏自编解码网络
    - **2006：Sparse encoder-decoder**

Yann LeCun, “Efficient Learning of Sparse Representations with an Energy-Based Model”, NIPS 2006.

其它的深度网络结构

- 卷积神经网络 (Convolutional Neural Networks)



MLP网络训练“新”动向

- 构造大容量多层感知器网络，通过“随机化”策略进行“正则化”
  - 结构特点：使用标准的2-3层感知器网络
  - 举例
    - **DropOut**
    - **DropConnect**

Q / A

- Any Questions...

神经网络 内容小结

- 生物神经元
- 人工神经元模型
  - 数学表示、激活函数、网络结构
- 感知器(Perceptron)
  - 感知器学习算法(误差修正法则)
- 单层感知网络
  - 误差修正法则
- 多层感知器(Multi-Layer Perceptron)
  - 反传(BP)算法

Q / A

- Any Questions...