第六章 典型的非线性分类器

苏智勇
可视计算研究组
南京理工大学
suzhiyong@njust.edu.cn
https://zhiyongsu.github.io

主要内容

6.1 分段线性判别函数

6.2 二次判别函数

6.3 多层感知器神经网络

6.4 支持向量机

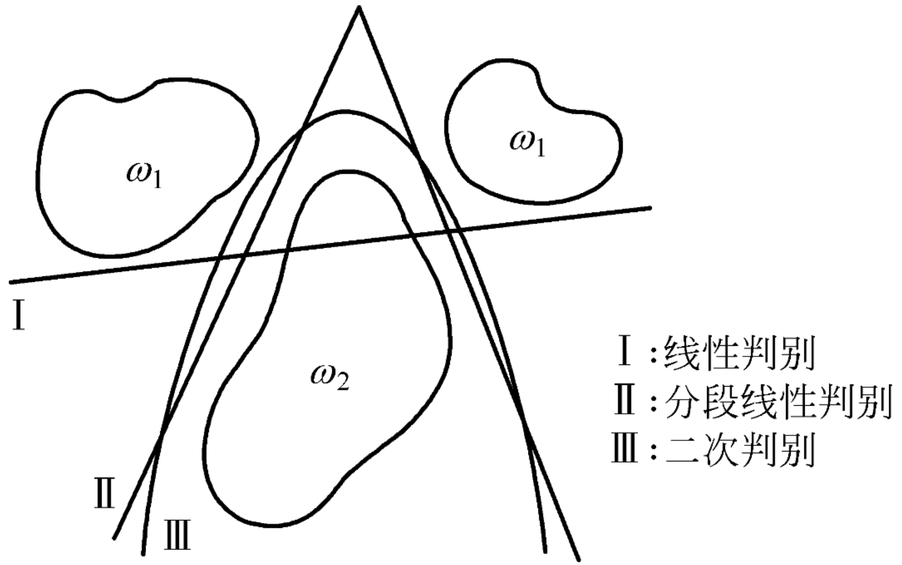
6.5 核函数机器

线性判别函数 VS 非线性判别函数

- **线性判别函数**
  - 简单、实用、经济
  - 线性不可分时错误率可能较大
- **非线性判别函数**
  - 各种函数的集合
  - 分段线性分类器、二次判别函数、多层感知器、支持向量机（非线性）等

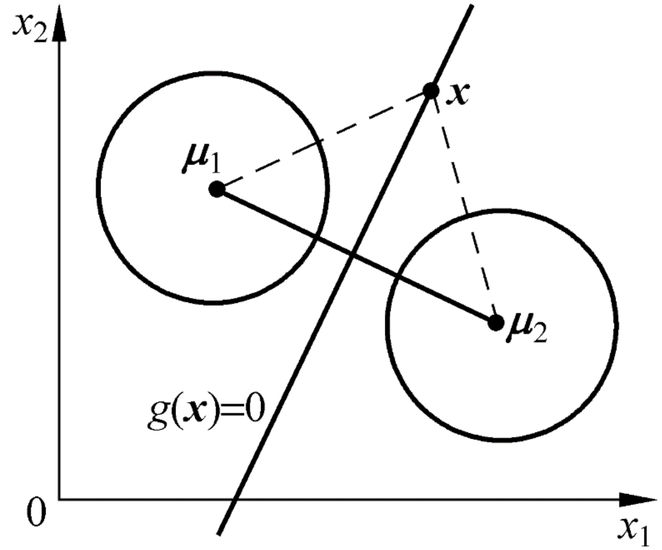
6.1 分段线性判别函数

- **定义**
  - 用多个线性分类器片段来实现非线性分类
  - 用多段线性函数来逼近非线性函数



6.1.1 分段线性距离分类器

- **最小距离分类器**
  - 有$\omega_1, \cdots, \omega_c$个类，均值是$\boldsymbol{\mu}_i (i = 1, \cdots, c)$
  - 对样本$\boldsymbol{x}$，如果$\|\boldsymbol{x} - \boldsymbol{\mu}_k\|^2 = \min\limits_{i=1,\cdots,c} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$，则决策$\boldsymbol{x}$属于$\omega_k$类
  - 两类情况下，最小距离分类器就是两类均值之间连线的垂直平分面（超平面）



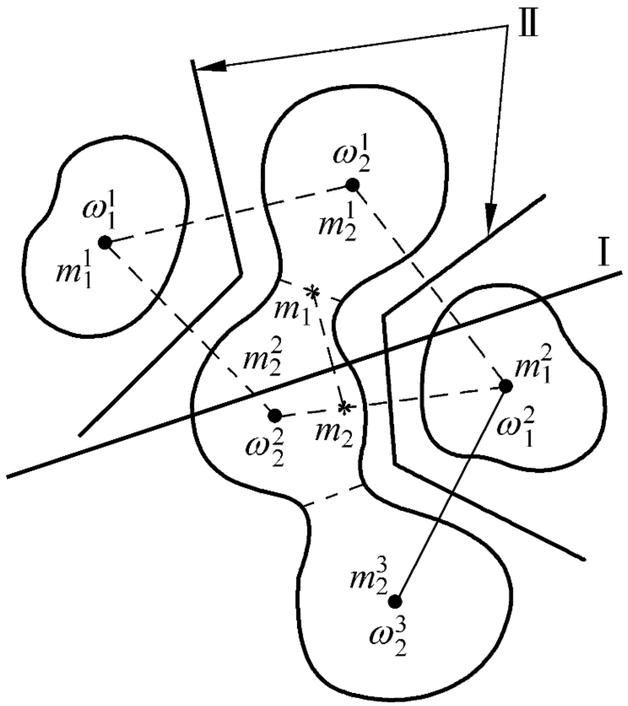
6.1.1 分段线性距离分类器

- **分段线性距离分类器**
  - 把属于ω_i类的样本区域R_i划分（$i = 1,2,\cdots,c$）为l_i个子区域R_i^l，$l = 1,2,\cdots,l_i$，每个子类的均值$\boldsymbol{m}_i^l$，对样本$\boldsymbol{x}$，ω_i类的判别函数定义为

$$g_i(\boldsymbol{x}) = \min_{l=1,\cdots,l_i} \|\boldsymbol{x} - \boldsymbol{m}_i^l\|$$

  - 对一个待分类样本，把它分到距离最近的**子类**所属于的类
  - 决策规则

$$若 g_k(\boldsymbol{x}) = \min_{i=1,\cdots,c} g_i(\boldsymbol{x})，则决策 x \in \omega_k$$



6.1.2 一般的分段线性判别函数

- **对每个子类建立一般形式的线性判别函数**
  - 把每个类别划分成l_i个子类

$$\omega_i = \{\omega_i^1, \omega_i^2, \cdots, \omega_i^{l_i}\},\ i = 1,2,\cdots,c$$

  - 对每个子类定义线性判别函数

$$g_i^l(\boldsymbol{x}) = \boldsymbol{w}_i^l \bullet \boldsymbol{x} + \omega_{i0}^l,\ l = 1,\cdots,l_i,\ i = 1,\cdots,c$$

  - 类ω_i的分段线性判别函数就定义为

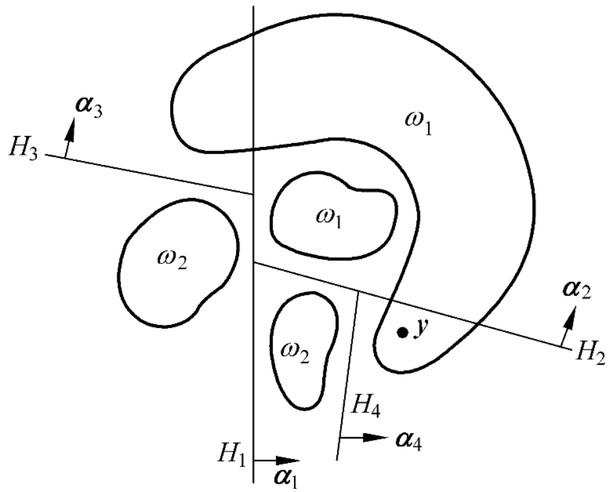
$$g_i(\boldsymbol{x}) = \max_{l=1,\cdots,l_i} g_i^l(\boldsymbol{x}),\ i = 1,\cdots,c$$

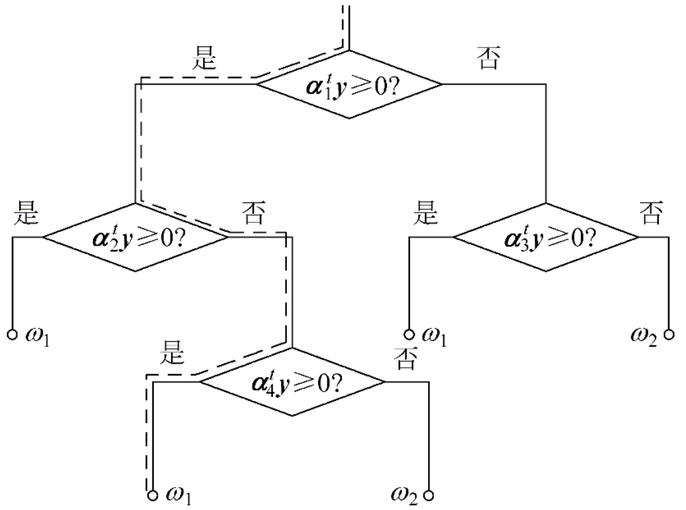
  - 决策规则

$$\text{若} g_k(\boldsymbol{x}) = \min_{i=1,\cdots,c} g_i(\boldsymbol{x}),\ \text{则决策} x \in \omega_k$$

6.1.2 一般的分段线性判别函数

- **子类划分的三种设计方法**
  - 人工确定子类的划分方案：多类分类器投票
  - 已知或可假定子类数目，但不知子类的划分：错误修正法，边划分边分类
  - 子类数目无法事先确定：树状分段线性分类器





6.1.2 一般的分段线性判别函数

- 已知或可以假定子类数目，但不知道子类的划分： ω_i， $i = 1,\cdots,c$， ω_i类中有l_i个子类，$\boldsymbol{\alpha}_i^l(k)$为子类权值

  (a) 初始化权值，$\boldsymbol{\alpha}_i^l(0)$， $i = 1,\cdots,c$， $l = 1,\cdots,l_i$

  (b) 已对某个样本$\boldsymbol{y}_k \in \omega_j$，找出$\omega_j$类的子类中最大的判别函数

$$\boldsymbol{\alpha}_j^m(t)^T\boldsymbol{y}_k = \max_{l=1,\cdots,l_j}\{\boldsymbol{\alpha}_j^l(t)^T\boldsymbol{y}_k\}$$

  ✓是若$\boldsymbol{\alpha}_j^m(t)^T\boldsymbol{y}_k > \boldsymbol{\alpha}_i^l(t)^T\boldsymbol{y}_k$， $\forall i = 1,\cdots,c$， $i \neq j$， $l = 1,\cdots,l_i$, 则$\boldsymbol{\alpha}_i^l(t)$不变

  ✓若对某个$i \neq j$，存在子类l，有$\boldsymbol{\alpha}_j^m(t)^T\boldsymbol{y}_k \leq \boldsymbol{\alpha}_i^l(t)^T\boldsymbol{y}_k$，则$\boldsymbol{y}_k$被错分，对其中最大者（记为$i,n$），修正：

$$\boldsymbol{\alpha}_j^m(t+1) = \boldsymbol{\alpha}_j^m(t) + \rho_k\boldsymbol{y}_k$$

$$\boldsymbol{\alpha}_i^n(t+1) = \boldsymbol{\alpha}_i^n(t) - \rho_k\boldsymbol{y}_k$$

  (c) 对下一个样本重复 (b)，直到收敛

6.2 二次判别函数

- 二次判别函数是一种常用的固定函数类型的分类方法，一般形式

$$g(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 = \sum_{k=1}^{d} w_{kk}x_k^2 + 2\sum_{j=1}^{d-1}\sum_{k=j+1}^{d} w_{jk}x_j x_k + \sum_{j=1}^{d} w_j x_j + w_0$$

  – 参数多、计算复杂；样本数量不够时，可靠性和泛化性难以保证

- 假定每一类数据符合正态分布，可以定义如下二次判别函数

$$g_i(\boldsymbol{x}) = K_i^i - (\boldsymbol{x} - \boldsymbol{m}_i)^{\mathrm{T}}\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x} - \boldsymbol{m}_i)$$

其中，$\hat{\boldsymbol{m}}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} x_j$，$\hat{\boldsymbol{\Sigma}}_i = \frac{1}{N_i - 1}\sum_{j=1}^{N_i}(\boldsymbol{x}_j - \hat{\boldsymbol{m}}_i)(\boldsymbol{x}_j - \hat{\boldsymbol{m}}_i)^{\mathrm{T}}$

- 适用于两类分布较成团的情况，K_i阈值，控制决策椭球大小

6.2 二次判别函数

- 假定一类ω_1近似正态分布，另一类ω_2均匀分布在第一类附近，只要对第一类求解其二次判别函数即可

$$g(\boldsymbol{x}) = K^2 - (\boldsymbol{x} - \hat{\boldsymbol{m}}_1)^{\mathrm{T}} \hat{\sum}_i^{-1} (\boldsymbol{x} - \hat{\boldsymbol{m}}_1)$$

其中，$\hat{\boldsymbol{m}}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} x_j$，$\hat{\boldsymbol{\Sigma}}_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (\boldsymbol{x}_j - \hat{\boldsymbol{m}}_i)(\boldsymbol{x}_j - \hat{\boldsymbol{m}}_i)^{\mathrm{T}}$

6.3 多层感知器神经网络

- **人工神经网络的定义**

  按美国神经网络学者Nielsen的定义

  – 人工神经网络是一个**并行、分布式处理结构**，由处理单元及其称为联接的无向通道互连而成。

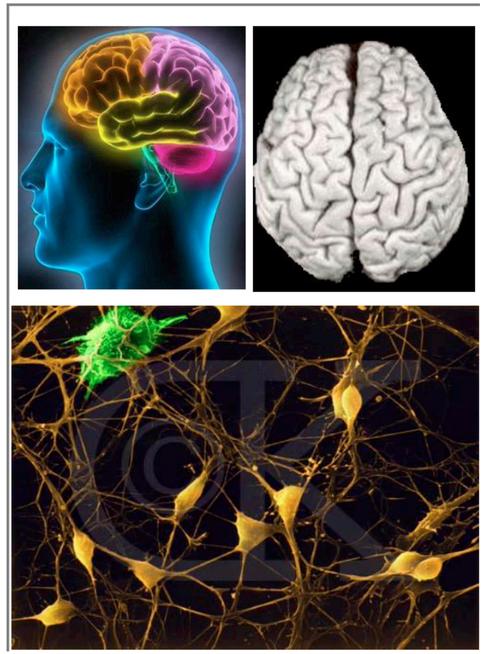
  – 这些处理单元具有局部内存，可以完成**局部操作**。该操作由输入至该单元的信号值和存储在该单元中的信号值来确定。

  – 每个处理单元**有一个单一的输出联接**，输出信号可以是任何需要的数学模型。

  是一种**模仿生物神经网络的结构和功能**的数学模型或计算模型。

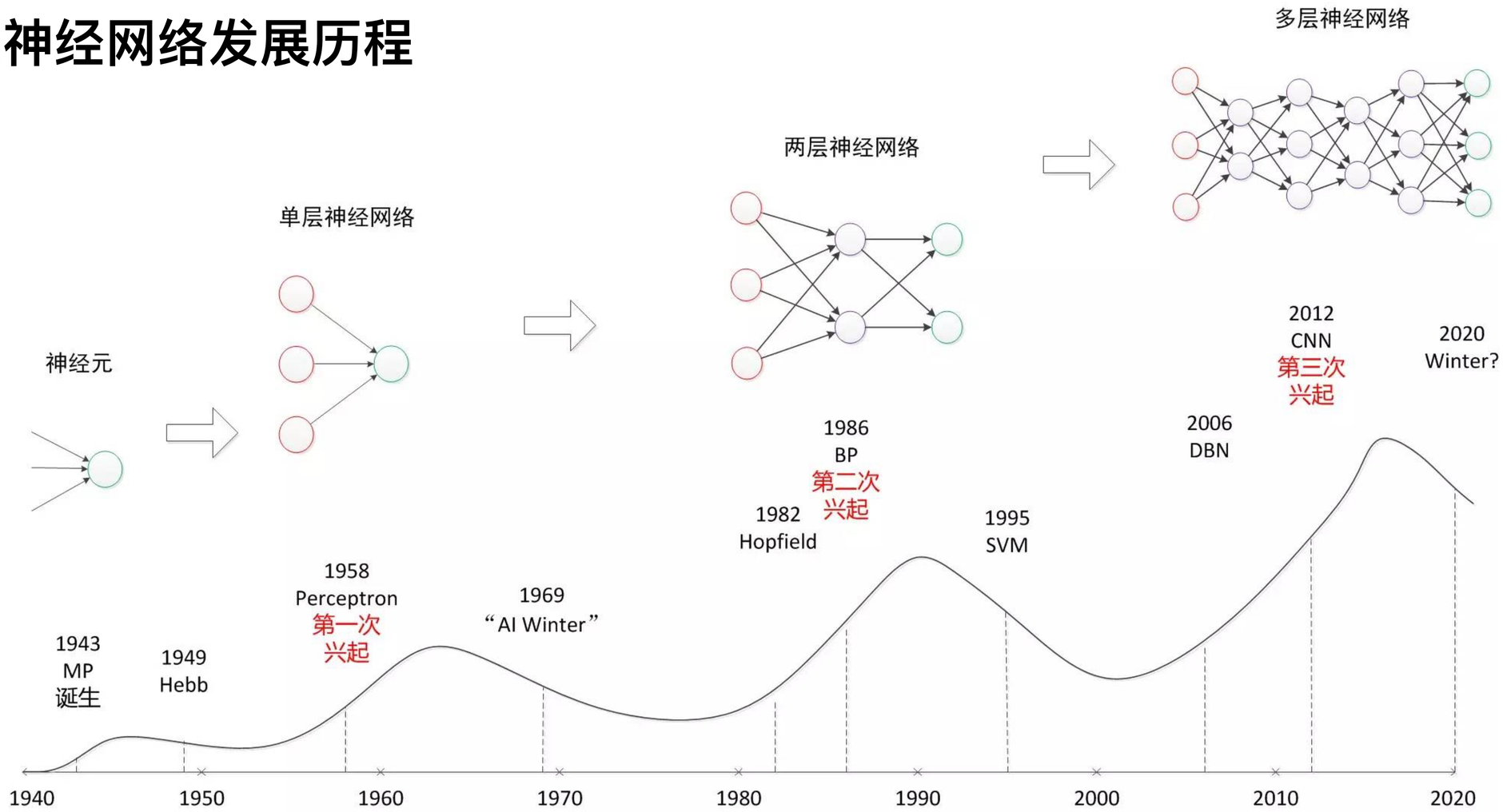
6.3 多层感知器神经网络

- **人脑**
  - 大脑里有许许多多(约800亿) **神经细胞（神经元）**。每个神经细胞由两个部分组成:细胞体和突起
  - 每个神经元可看作是一个小的**处理单元**，这些神经元按某种方式连接起来，形成大脑内部的生理**神经元网络**。
  - 神经元网络中各神经元之间联结的强弱，**按外部的激励信号做自适应变化**，而每个神经元又随着所接收到的多个接收信号的综合大小而**呈现兴奋或抑制状态**。
  - 现已明确**大脑的学习过程**就是神经元之间连接强度随外部激励信息做自适应变化的过程，而大脑处理信息的结果 则由神经元的状态表现出来。



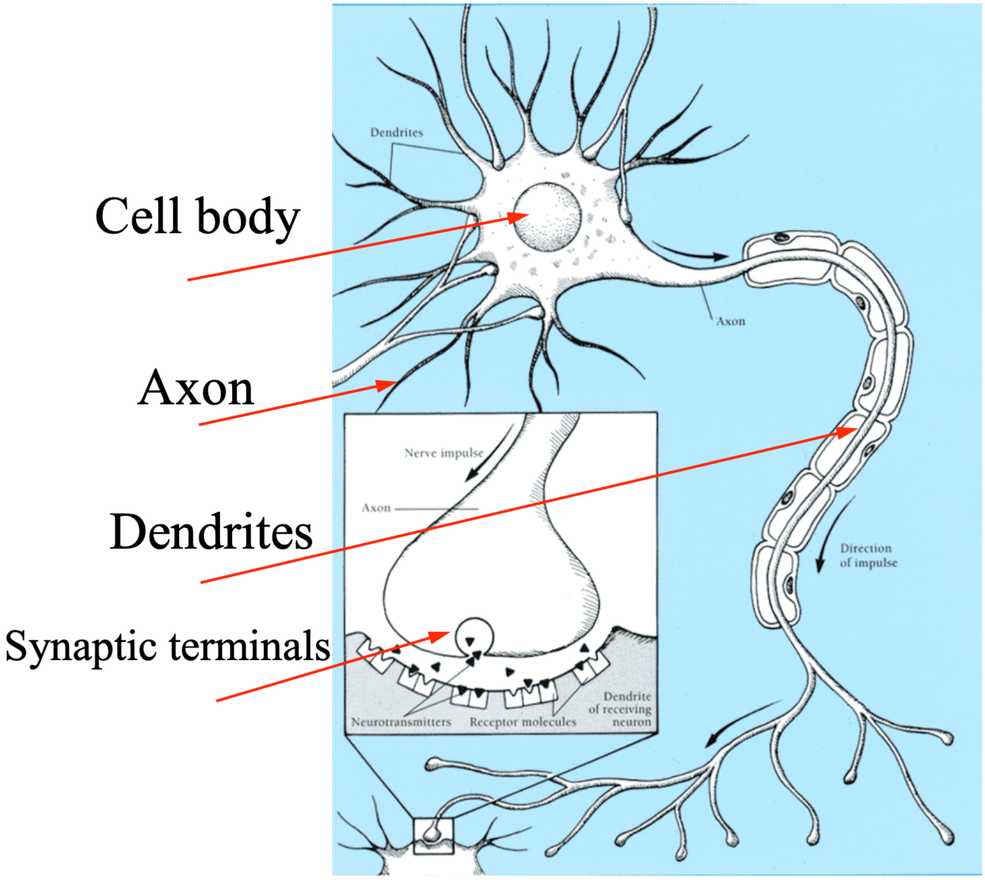
6.3 多层感知器神经网络

- 人工神经网络发展历程



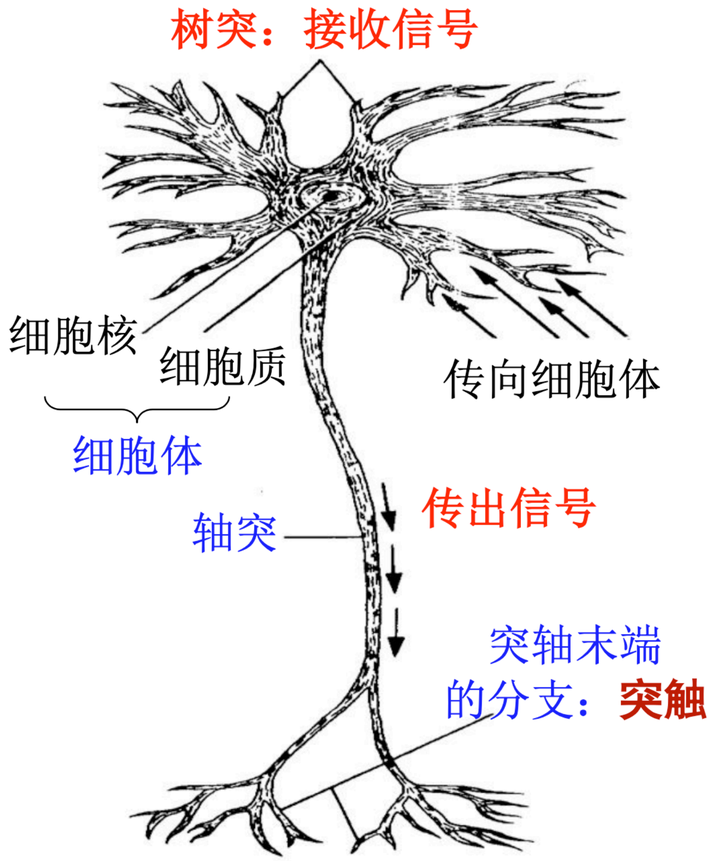
6.3.1 神经元与感知机

- **神经元（neuron）**
  - 由**细胞体**（cell body）和**突起**组成。
  - 突起分为两种：树突（dendrites）、轴突（axon）、突触
    - 短的叫**树突**，从其它神经细胞接收信号传递给细胞体的
    - 长的叫**轴突**，将信号传递给其它神经细胞的树突的。
    - **突触**：突轴末端的分支。



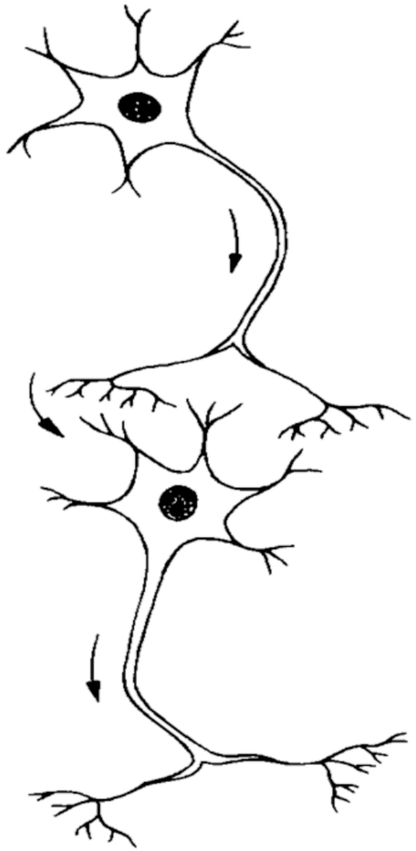
6.3.1 神经元与感知机

- **神经元（neuron）**
  - 神经元之间通过突触两两相连。
  - 轴突记录了神经元间联系的强弱。
  - 只有达到一定的兴奋程度，神经元才向外界传输信息。
  - 每个神经元可抽象成一个激励函数 (非线性处理)。



6.3.1 神经元与感知机

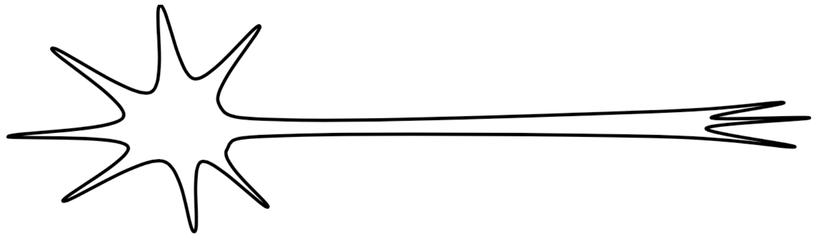
- **神经元的基本工作机制**
  - 一个神经元有两种状态—**兴奋和抑制**
  - 平时处于抑制状态的神经元，当接收到其它神经元经由突触传来的冲击信号时，多个输入在神经元中以代数和的方式叠加。
    - **进入突触的信号会被加权**，起兴奋作用的信号为正， 起抑制作用的信号为负。
  - 如果叠加总量超过某个阈值，神经元就会被激发进入兴奋状态，发出输出脉冲，并由轴突的突触传递给其它神经元。

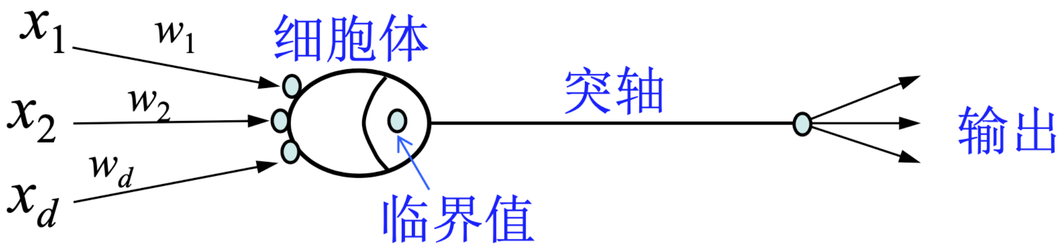


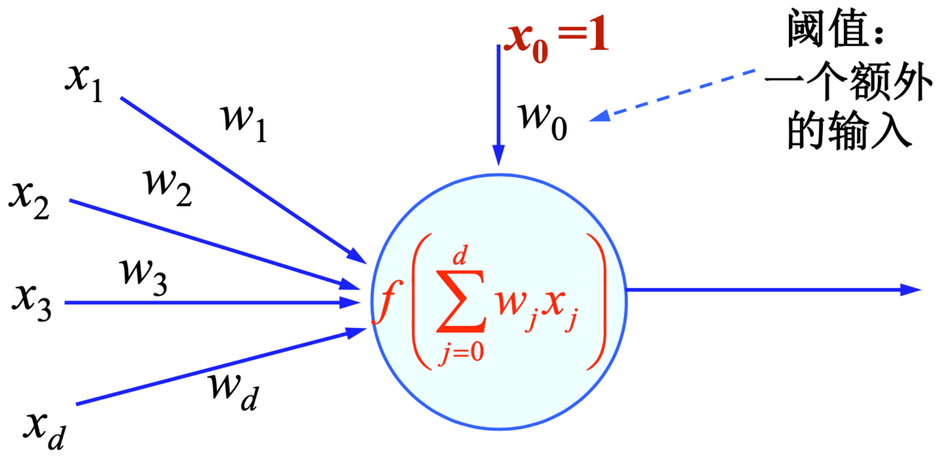
神经元间的信号通过突触传递。

6.3.1 神经元与感知机

- **人工神经网络处理单元**
  - 对每个输入信号进行加权处理(确定其强度)；
  - 确定所有输入信号的组合效果(求和)；
  - 确定其输出(转移特性，即激励特性)。





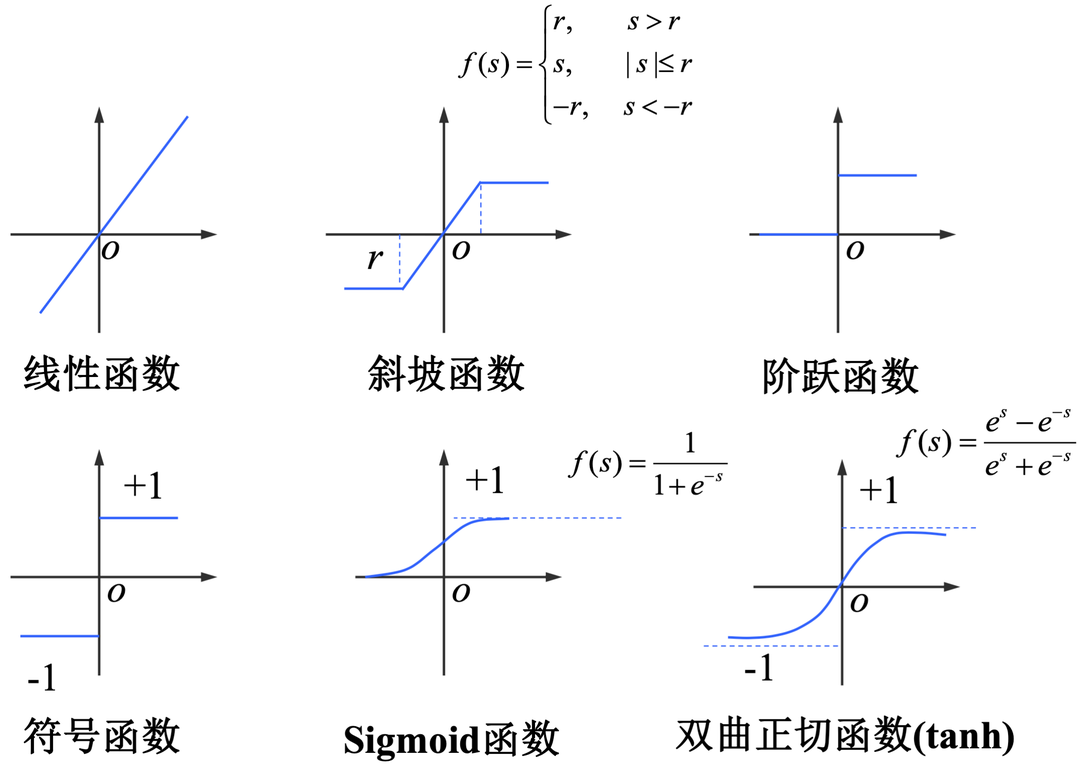


三功能
加权
求和
激励

x: 收到信号　**w**: 信号的权重　$f(.)$: 激励函数

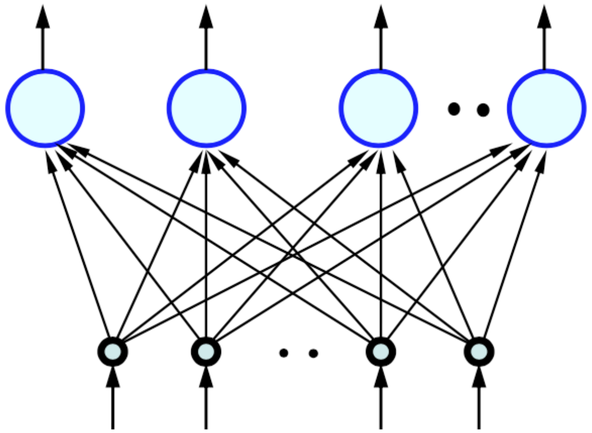
6.3.1 神经元与感知机

- **激励函数**$f(\cdot)$
  - 也称转移函数、传输函数或限幅函数。其作用是将可能的无限域变换到指定的有限范围内进行输出(类似于生物神经元的非线性转移特性)。
  - 常用激励函数
    - Sigmoid函数
    - 符号函数
    - …

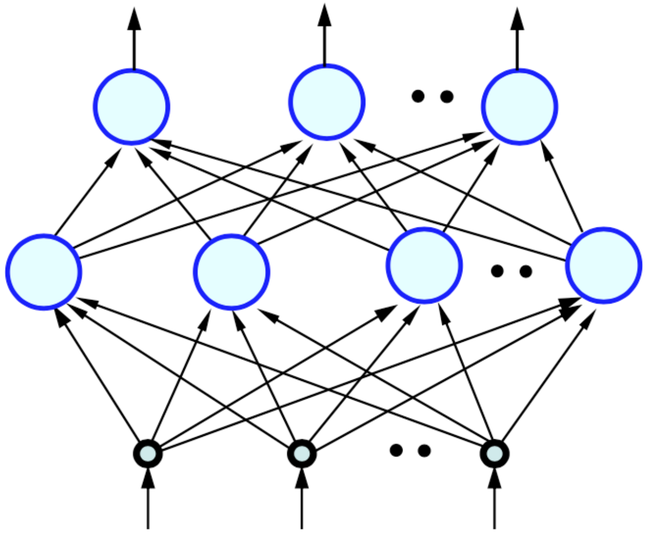


6.3.1 神经元与感知机

- **拓扑结构**
  - 前馈神经网络



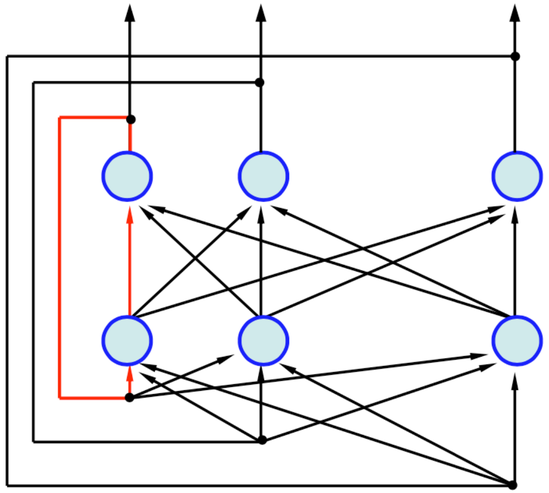
单层感知器



多层感知器

6.3.1 神经元与感知机

- **拓扑结构**
  - 反馈神经网络



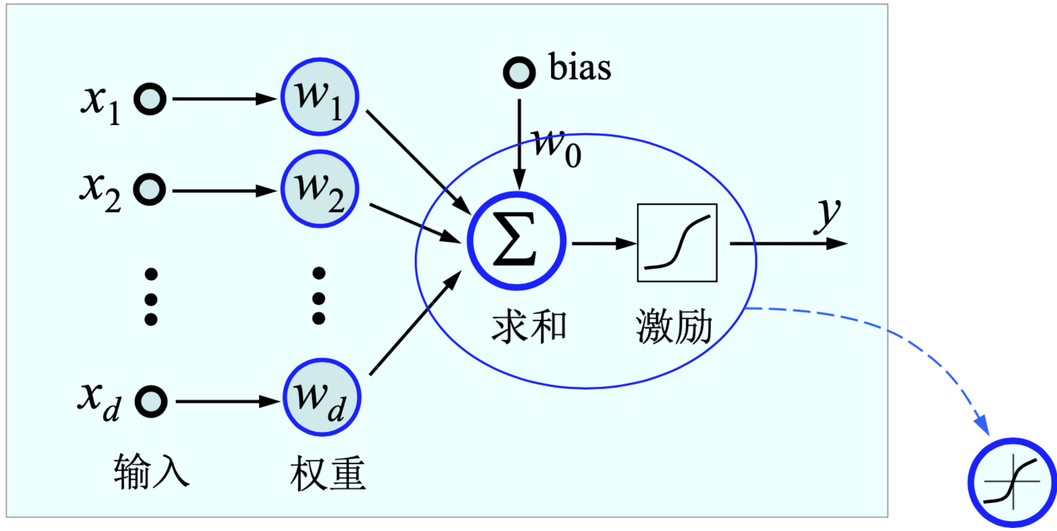
网络有回路，信息可以沿回路流动

结点的输出：依赖于当前的输入，也依赖于自己以前的输出.

6.3.1 神经元与感知机

- **单层网络**
  - McCulloch-Pitts Model (1943)
  - 阈值逻辑单元(Threshold Logic Unit, TLU)

$$y = \theta\left(\sum_{i=1}^{n} w_i x_i + w_0\right)$$



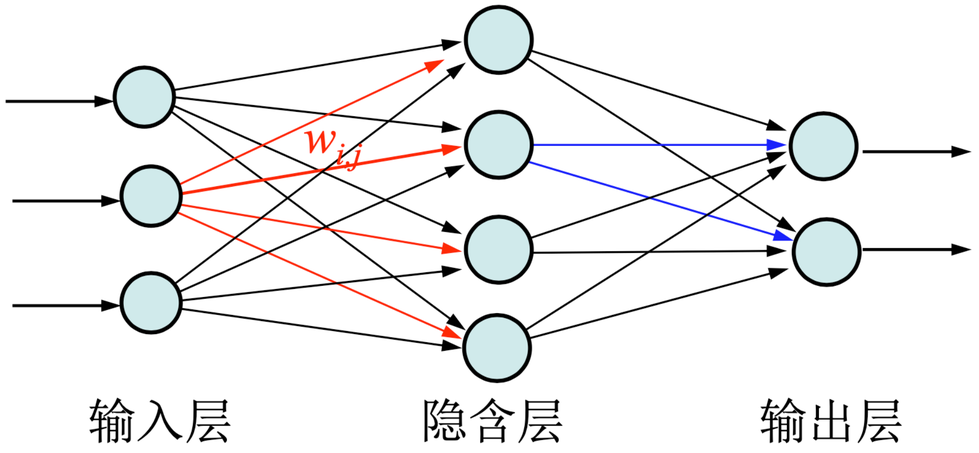
6.3.2 用多个感知器实现非线性分类

- **多层网络**
  - 单个感知器神经元无法解决非线性问题；多个神经元分层组合实现复杂的空间形状分割
  - 虽然目前有很多网络模型，但它们的结点基本上都是按层排列的。这一点模仿了大脑皮层中的网络模块。
  - 多层网络是由单层网络进行级联构成的，即上一层的输出作为下一层的输入。

6.3.2 用多个感知器实现非线性分类

- **多层网络**
  - 对于任意复杂形状的分类区域，总可以用多个神经元组成一定的层次结构来实现分类

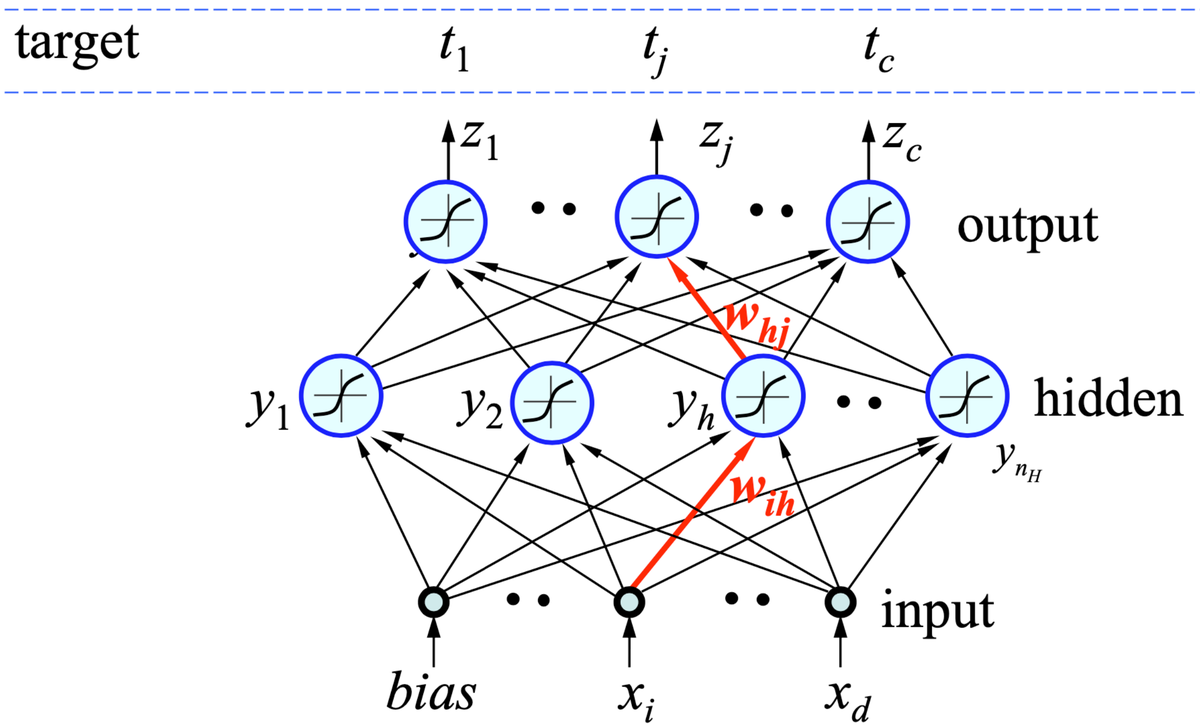
$$y = \theta\left\{\sum_{j=1}^{n} v_j \theta\left(\sum_{i=1}^{m} w_{ij} x_i + w_{0j}\right) + v_0\right\}$$



6.3.3 反向传播算法

- **多层感知器（multi layer perceptron，MLP）**
  - 是一种可普遍适用的非线性学习机器，能够实现任意复杂的函数映射

Hope: $z_1 \approx t_1, \cdots, z_c \approx t_c$, **for all samples:** $J(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{c}(t_j - z_j)^2 \approx 0$



6.3.3 反向传播算法

- 网络描述-每个样本所经历的计算

上标 k 联系
第 k 个样本

对第 k 个样本，隐含层 h 结点的输入加权和为：

$$net_h^k = \sum_i w_{ih} x_i^k$$

经过激励，隐含层 h 结点的输出：

$$y_h^k = f(net_h^k) = f\left(\sum_i w_{ih} x_i^k\right)$$

输出层 j 结点的输入加权和为：

$$net_j^k = \sum_h w_{hj} y_h^k = \sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)$$

经过激励，输出层 j 结点的输出：

$$z_j^k = f(net_j^k) = f\left(\sum_h w_{hj} y_h^k\right) = f\left(\sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)\right)$$

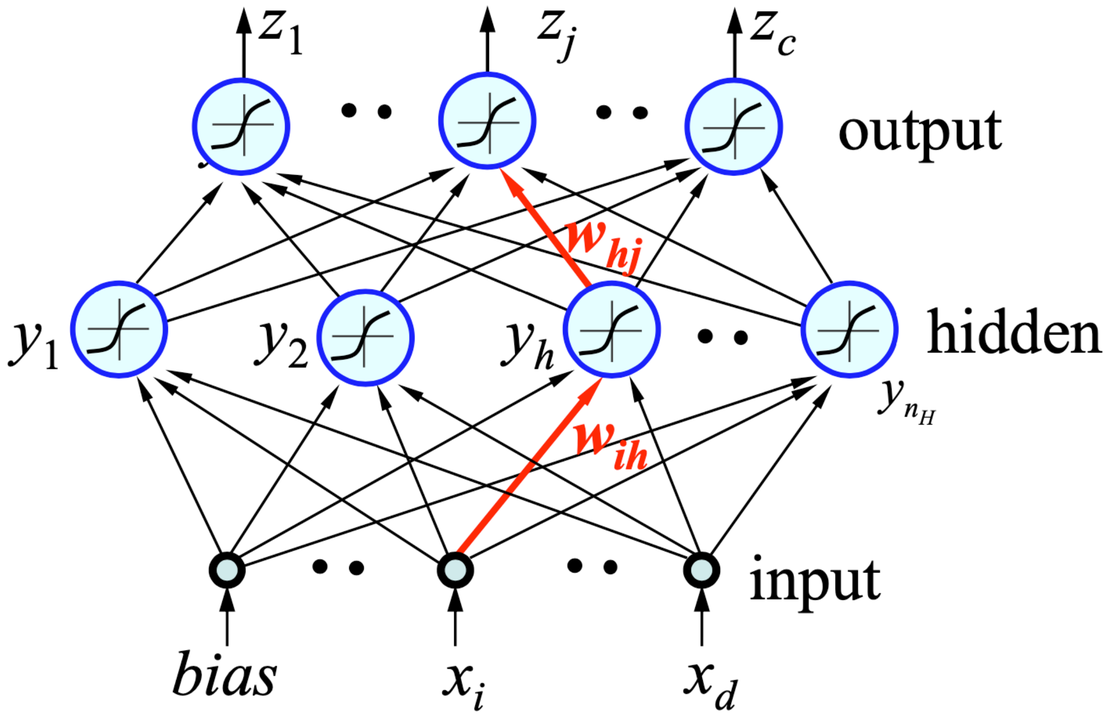
6.3.3 反向传播算法

- 网络描述–每个样本所经历的计算

上标k：第 k 个样本

$$z_j^k = f\left(\sum_h w_{hj} y_h^k\right)$$

$$y_h^k = f\left(\sum_i w_{ih} x_i^k\right)$$

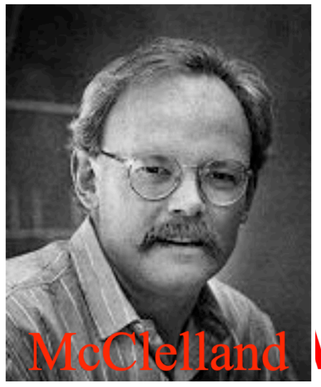


$$z_j^k = f(net_j^k) = f\left(\sum_h w_{hj} y_h^k\right) = f\left(\sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)\right)$$

6.3.3 反向传播算法

- **误差反向传播(BP)算法**
  - D. Rumelhart，J. McClelland于1985年提出了误差反向传播 (Back Propagation, BP)学习算法
  - **基本原理**
    - 利用输出后的误差来估计**输出层的前一层**的误差，再用这个误差估计更前一层的误差，如此一层一层地反传下去，从而获得所有其它各层的误差估计。

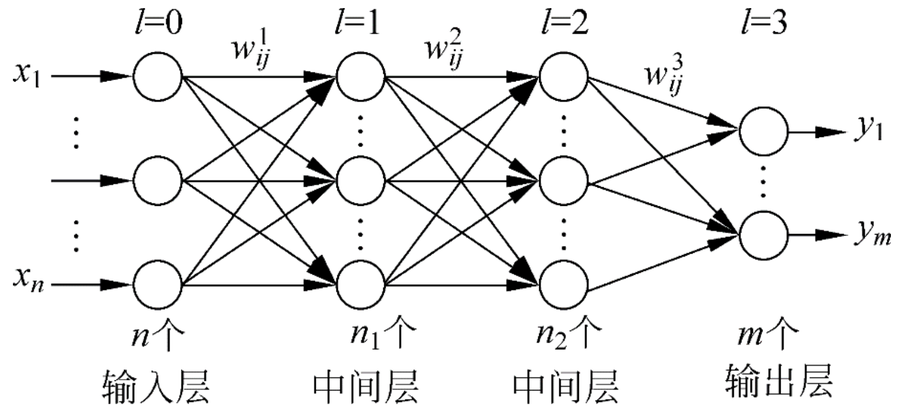

Rumelhart


McClelland

6.3.3 反向传播算法

- **BP算法**

目标函数是**均方误差**，采用**梯度下降法**通过调整各层的权值求目标函数最小化



(1) 用小随机数进行权值初始化，设训练时间$t = 0$

(2) 给出一个训练样本 $\boldsymbol{x} = \left[x_1, \cdots, x_n\right]^T \in R^n$ 和期望输出$\boldsymbol{D} = \left[d_1, \cdots, d_m\right]^T \in R^m$

6.3.3 反向传播算法

- **BP算法**

(3) 计算在 $\boldsymbol{x}$ 输入下的实际输出 $y_r = f(\sum_{s=1}^{n_{L-2}} w_{sr}^{l=L-1} \cdots f\left(\sum_{j=1}^{n_1} w_{jk}^{l=2} f(\sum_{i=1}^{n} w_{ij}^{l=1} x_i)\right)$, $r = 1, \cdots, \mathrm{m}$

(4) 从输出层开始**调整权值**

1)第l层，用下面的公式**修正权值**：$w_{ij}^l(t+1) = w_{ij}^l(t) + \Delta w_{ij}^l(t)$, $j = 1, \cdots, n_l$, $i = 1, \cdots, n_{l-1}$,
$\Delta w_{ij}^l(t) = -\eta \delta_j^l x_i^l$，$\eta$为学习步长

2)输出层：$\delta_j^l = -y_j(1-y_j)(d_j - y_j)$, $j = 1, \cdots, m$

3)对中间层：$\delta_j^l = x_j^l\left(1 - x_j^l\right)\sum_{k=1}^{n_{l+1}} \delta_k^{l+1} w_{jk}^{l+1}(t)$, $j = 1, \cdots, n_l$

6.3.3 反向传播算法

- **BP算法**

(5) 重新计算输出，考查误差指标。如达到终止条件则终止，否则置 $t = t + 1$，返回(2)

注意，上述BP算法是针对sigmoid传递函数：

$$f(\alpha) = 1/(1 + e^{-\alpha})$$

$$f'(\alpha) = f(\alpha)(1 - f(\alpha))$$

当改用其他传递函数时其梯度函数需要相应改变。

6.3.3 反向传播算法

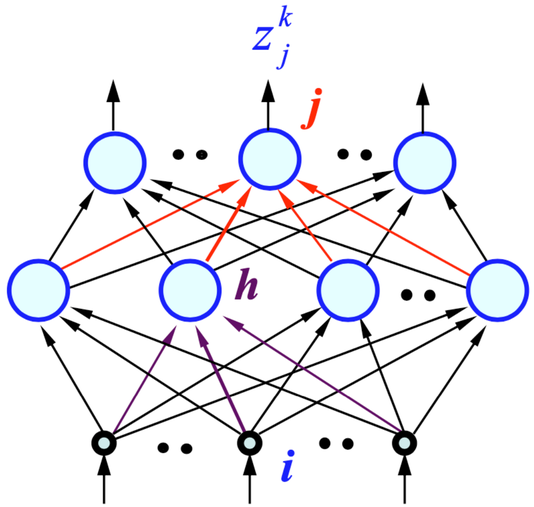
- 误差反向传播训练算法
  - 属于监督学习算法，**通过调节各层的权重，使网络学会**由“输入-输出对”组成的训练组。
  - BP算法核心是梯度下降法。
  - 权重先从输出层开始修正，再依次修正各层权重
    - 首先修正：**“输出层至最后一个隐含层”**的连接权重
    - 再修正：**“最后一个隐含层至倒数第二个隐含层”**的连接权重，….
    - 最后修正：**“第一隐含层至输入层”**的连接权重。

学习的本质：对网络各连接权重作动态调整！

6.3.3 反向传播算法

- 误差函数－单个样本

上标 k 联系
第 k 个样本



$$z_j^k = f\left(net_j^k\right) = f\left(\sum_h w_{hj} y_h^k\right)$$

$$y_h^k = f\left(net_h^k\right) = f\left(\sum_i w_{ih} x_i^k\right)$$

$$E(\mathbf{w})^k = J(\mathbf{w})^k = \frac{1}{2}\sum_j (t_j^k - z_j^k)^2$$

$$= \frac{1}{2}\sum_j \left(t_j^k - f(net_j^k)\right)^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} y_h^k\right)\right\}^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} f\left(net_h^k\right)\right)\right\}^2$$

$$= \frac{1}{2}\sum_j \left\{t_j^k - f\left(\sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)\right)\right\}^2$$

6.3.3 反向传播算法

- 复合函数求导数

设 $f(\mathrm{x})=g_1(g_2(g_3(\cdots g_n(x))))$,

令 $h_2=g_2(g_3(\cdots g_n(x)))$,

$h_3=g_3(g_4(\cdots g_n(x))),\cdots$

$$\frac{\partial f(\mathrm{x})}{\partial \mathrm{x}}$$
$$=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2}{\partial x}$$
$$=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2(h_3)}{\partial h_3}\frac{\partial h_3}{\partial x}$$
$$=\cdots\cdots$$
$$=\frac{\partial g_1(h_2)}{\partial h_2}\frac{\partial h_2(h_3)}{\partial h_3}\cdots\frac{\partial g_n(x)}{\partial x}$$

设 $f(\mathrm{x})=g_1(g_2(g_3(\cdots g_n(x)))+t_2(t_3(\cdots t_m(y))))$,

令 $H_2=g_2(g_3(\cdots g_n(x)))+t_2(t_3(\cdots t_m(y)))$,

$h_2=g_2(g_3(\cdots g_n(x)))$,

$h_3=g_3(g_4(\cdots g_n(x))),\cdots$

$$\frac{\partial f(\mathrm{x})}{\partial \mathrm{x}}=\frac{\partial g_1(H_2)}{\partial H_2}\frac{\partial H_2}{\partial x}$$
$$=\frac{\partial g_1(H_2)}{\partial H_2}\left(\frac{\partial h_2(h_3)}{\partial h_3}\frac{\partial h_3}{\partial x}+0\right)$$
$$=\cdots\cdots$$
$$=\frac{\partial g_1(H_2)}{\partial H_2}\frac{\partial h_2(h_3)}{\partial h_3}\cdots\frac{\partial g_n(x)}{\partial x}$$

6.3.3 反向传播算法

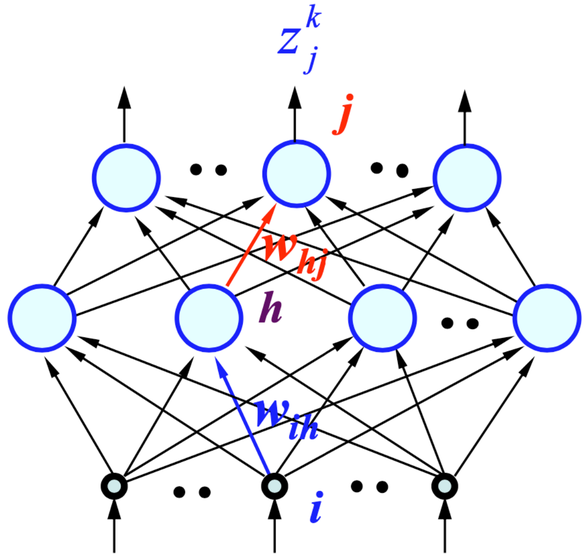
- 网络训练：隐含层－输出层

隐含层到输出层的连接权重调节量：

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}} = -\eta \sum_k \frac{\partial E}{\partial net_j^k} \frac{\partial net_j^k}{\partial w_{hj}}$$

$$= \eta \sum_k (t_j^k - z_j^k) f'(net_j^k) y_h^k$$

$$= \eta \sum_k \delta_j^k y_h^k \qquad \text{(δ规则)}$$



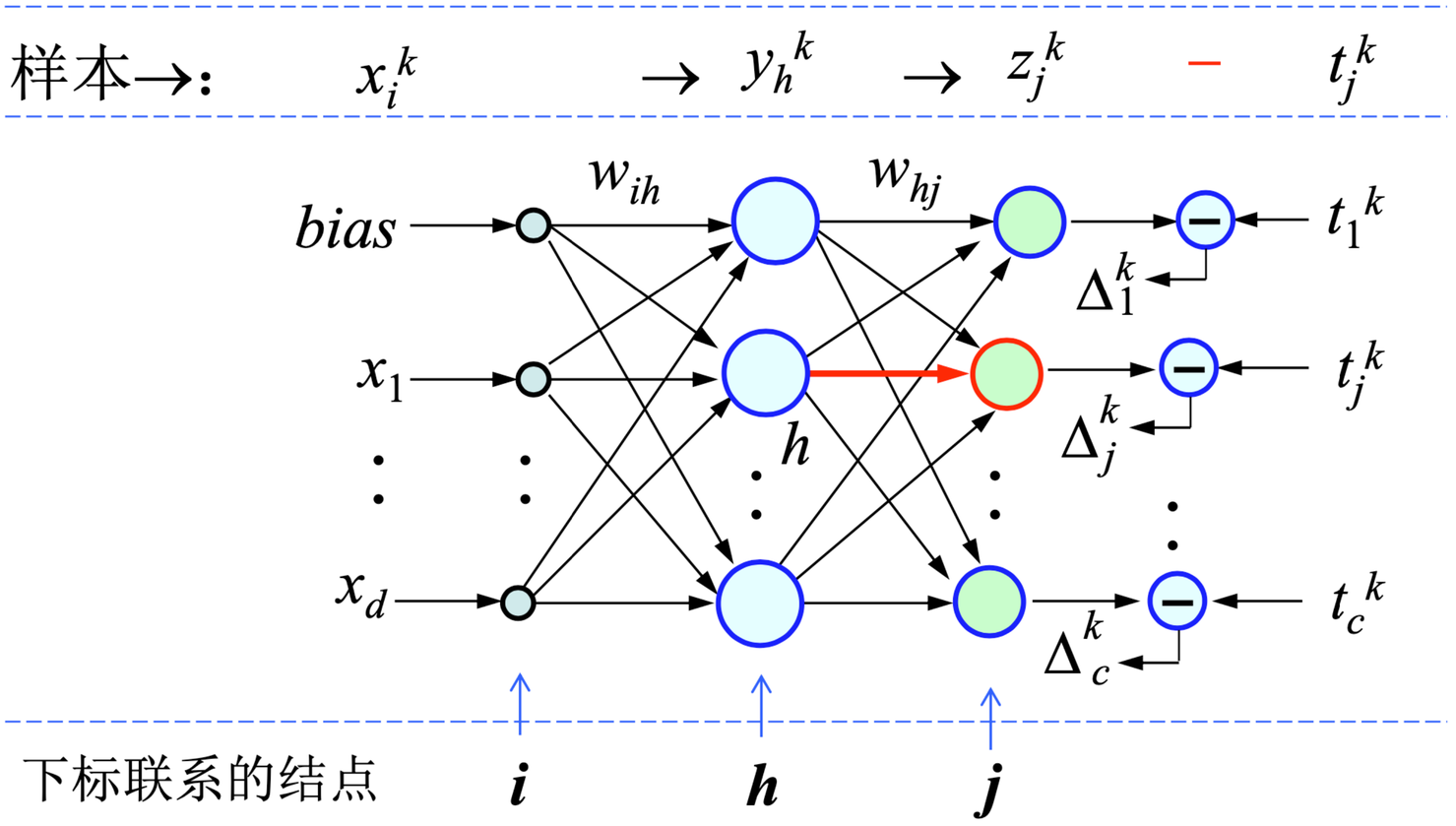
$$\delta_j^k = \frac{\partial E}{\partial net_j^k} = f'(net_j^k)(t_j^k - z_j^k) = f'(net_j^k)\Delta_j^k,$$

δ_j^k：边的指向结点的误差信号（局部梯度）

$$\Delta_j^k = t_j^k - z_j^k$$

6.3.3 反向传播算法

- 隐含层－输出层，**准备好输出层的误差：** $\Delta_j^k = t_j^k - z_j^k$



注：上标 k 联系第 k 个样本

6.3.3 反向传播算法

- 隐含层—输出层：第 k 个训练样本对权重 w_{hj} 的贡献

$h \to j$, for sample k:

δ规则： $\Delta w_{hj}\big|_{\text{sample } k} = \eta \delta_j^k y_h^k$

权重所联边的起始结点（隐含结点 h）的输出

权重所联边的指向结点（输出结点 j）收集到的误差信号

$\delta_j^k = f'(net_j^k)\Delta_j^k, \quad \Delta_j^k = t_j^k - z_j^k$

误差**在权重所联边的指向结点处**计算。

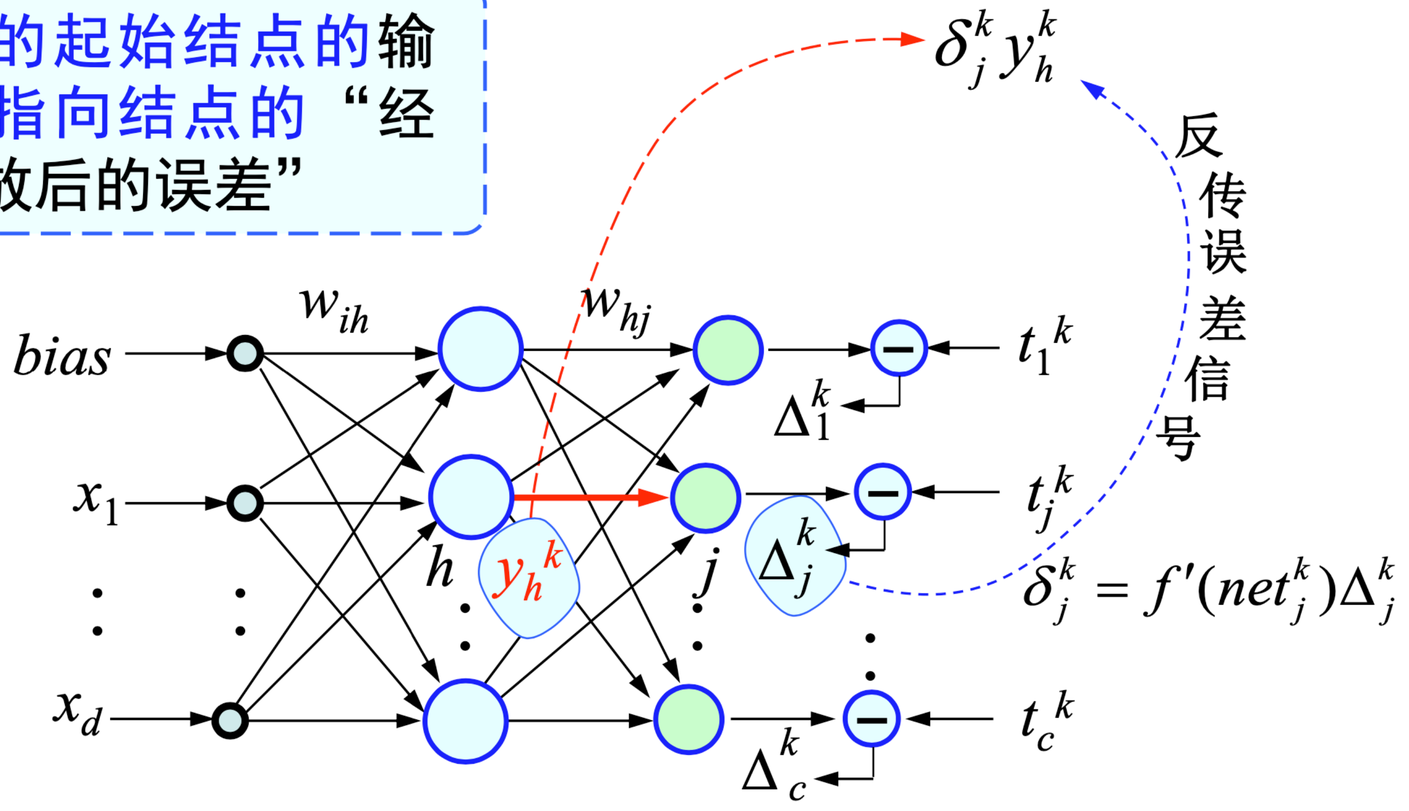
误差大小等于：该结点收集到的误差**乘以**激励函数对“该结点加权和”的导数。

6.3.3 反向传播算法

误差反传与权重更新：

所联边的起始结点的输出乘以指向结点的“经导数缩放后的误差”

第 k 个样本对权重更新的贡献

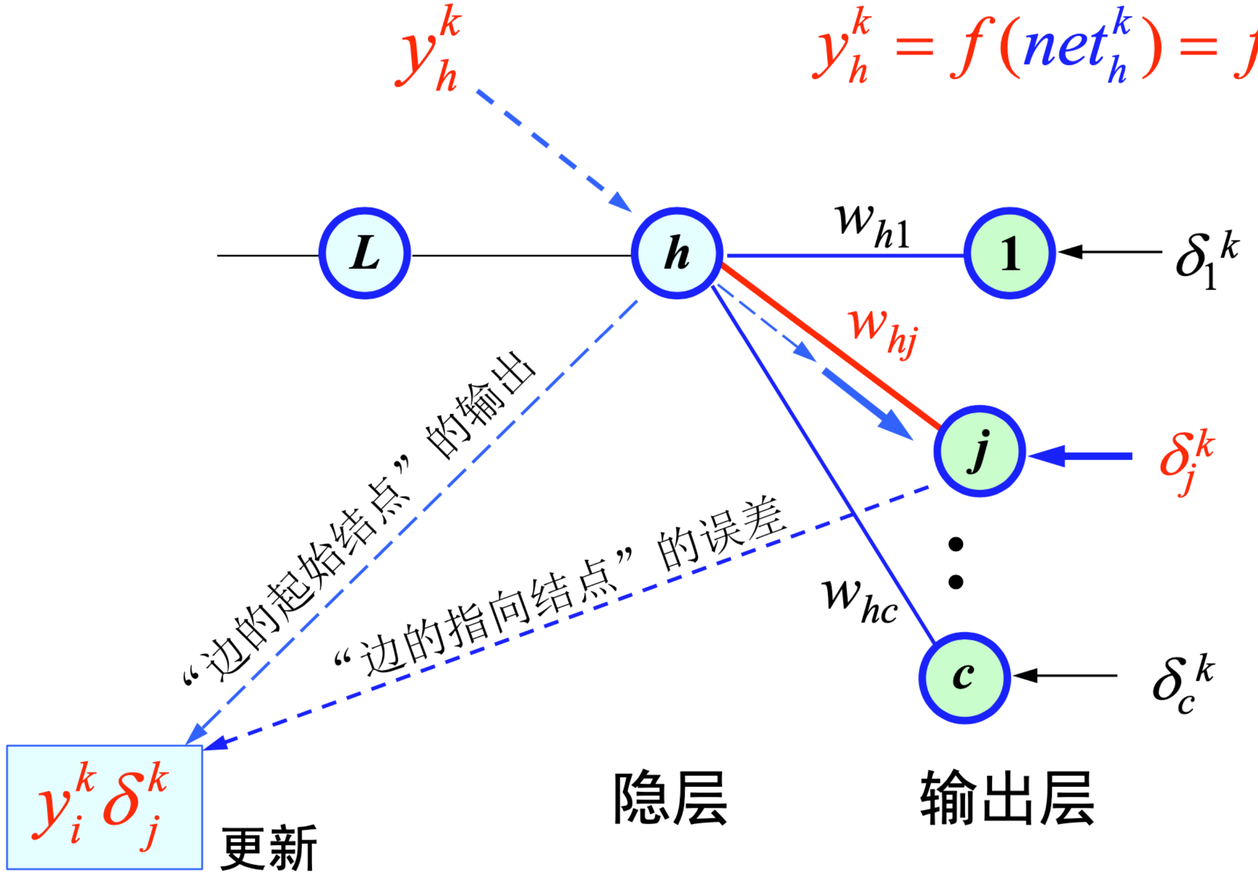


6.3.3 反向传播算法

隐层-输出权重更新示意：

w_{hj}所连接的边的起始结点向外传递的信号值

$$y_h^k = f(net_h^k) = f\left(\sum_i w_{ih} x_i^k\right)$$



w_{hj}所连接的边的指向结点的误差

6.3.3 反向传播算法

- 激励函数采用sigmoid函数(最常用)：

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

$$f'(s)=\frac{e^{-s}}{(1+e^{-s})^2}=\frac{1}{1+e^{-s}}\left(1-\frac{1}{1+e^{-s}}\right)=z(1-z),\quad z=f(s)$$

对输出层结点 j，我们有：

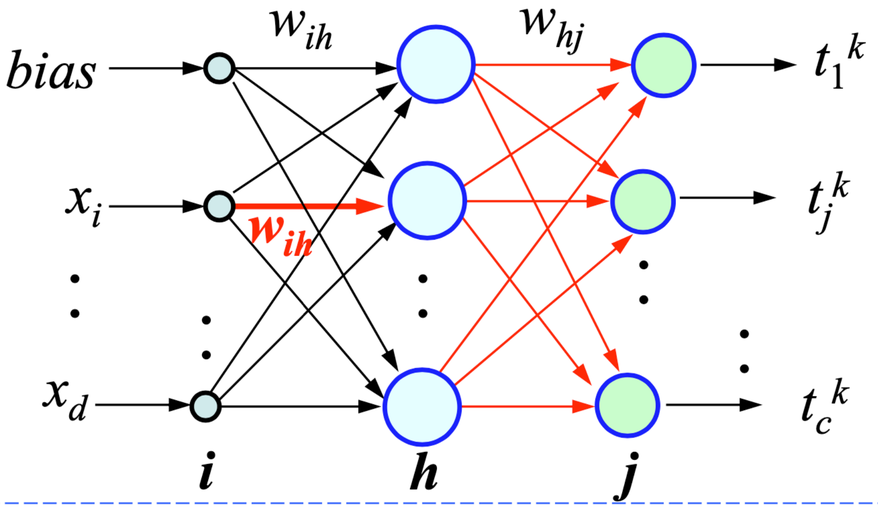
$$\delta_j^k=\frac{\partial E}{\partial net_j^k}=f'\left(net_j^k\right)\left(t_j^k-z_j^k\right)=z_j^k\left(1-z_j^k\right)\left(t_j^k-z_j^k\right)$$

6.3.3 反向传播算法

对于**输入层到隐含层**结点连接的边的权重修正量 Δw_{ih}，必须考虑将 $E(\mathbf{w})$ 对 w_{ih} 求导，需利用**分层链路法**。

输入层至隐含层权重更新：

$$E(\mathbf{w})=\sum_{k}J(\mathbf{w})=\frac{1}{2}\sum_{k,j}(t_j^k-z_j^k)^2=\frac{1}{2}\sum_{k,j}\left\{t_j^k-f\left(\sum_{h}w_{hj}f\left(\sum_{i}w_{ih}x_i^k\right)\right)\right\}^2$$



样本→：　$x_i^k \rightarrow net_h^k \rightarrow y_h^k \rightarrow net_h^k \rightarrow z_j^k$　　t_j^k

6.3.3 反向传播算法

输入层至隐含层权重更新：

用别名$\tilde{h}$代替h

$$E(\mathbf{w})=\frac{1}{2}\sum_{k,j}(t_j^k-z_j^k)^2=\frac{1}{2}\sum_{k,j}\left\{t_j^k-f\left(\sum_{\tilde{h}}w_{\tilde{h}j}f\left(\sum_i w_{i\tilde{h}}x_i^k\right)\right)\right\}^2$$

$$net_{\tilde{h}}^k=\sum_i w_{i\tilde{h}}x_i^k,\quad y_{\tilde{h}}^k=f(net_{\tilde{h}}^k)$$

$$net_j^k=\sum_{\tilde{h}} w_{\tilde{h}j}y_{\tilde{h}}^k,\quad z_j^k=f(net_j^k)$$

z_j^k

net_j^k

$y_{\tilde{h}}^k$

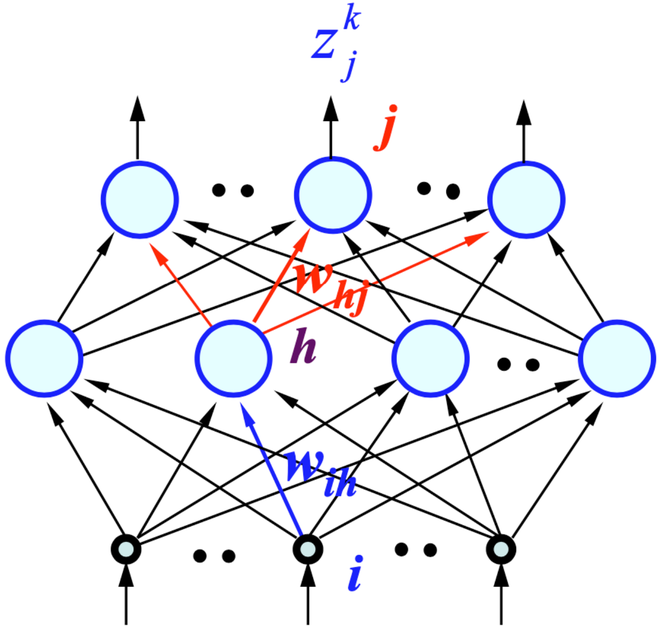
$net_{\tilde{h}}^k$

包含权重w_{ih}

6.3.3 反向传播算法

待更新权重w_{ih}的增量：

$$\Delta w_{ih} = -\eta \frac{\partial E}{\partial w_{ih}} = -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial w_{ih}}$$



$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial w_{ih}}$$

$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial w_{ih}}$$

$$= -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

(链式法则)

$$= -\eta \sum_{k,j} \frac{\partial E}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

6.3.3 反向传播算法

待更新权重的增量（具体化）

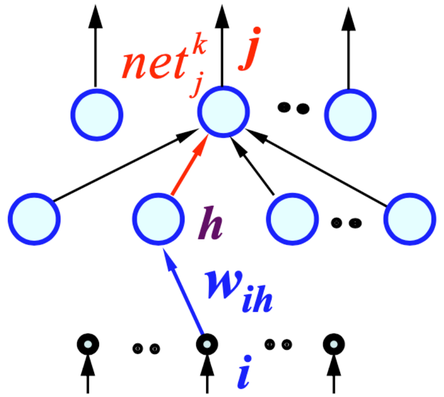
$$\Delta w_{ih} = -\eta \frac{\partial E}{\partial w_{ih}} = -\eta \sum_{k,j} \frac{\partial E}{\partial z_j^k} \cdot \frac{\partial z_j^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) \frac{\partial z_j^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) f'\left(net_j^k \right) \frac{\partial net_j^k}{\partial y_h^k} \frac{\partial y_h^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) f'\left(net_j^k \right) w_{hj} \frac{\partial y_h^k}{\partial w_{ih}}$$



$$net_j^k = \sum_{\tilde{h}} w_{\tilde{h}j} y_{\tilde{h}}^k$$

（只有当$\tilde{h} = h$时 y_h^k 才包含w_{ih}）

6.3.3 反向传播算法

（接前一页）

$$\Delta w_{ih} = \eta \sum_{k,j} \left(t_j^k - z_j^k \right) f'\left(net_j^k \right) w_{hj} \frac{\partial y_h^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) f'\left(net_j^k \right) w_{hj} \frac{\partial y_h^k}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}}$$

$$= \eta \sum_{k,j} \left(t_j^k - z_j^k \right) f'\left(net_j^k \right) w_{hj} f'\left(net_h^k \right) x_i^k$$

$$= \eta \sum_{k,j} \delta_j^k w_{hj} f'(net_h^k) x_i^k$$

$$\delta_j^k = f'\left(net_j^k \right)\left(t_j^k - z_j^k \right)$$

$$= \eta \sum_{k} \left(f'(net_h^k) \sum_j \delta_j^k w_{hj} \right) x_i^k$$

$$= \eta \sum_{k} \delta_h^k x_i^k$$

$$\delta_h^k = \frac{\partial E}{\partial net_h^k} = f'(net_h^k) \sum_j w_{hj} \delta_j^k = f'(net_h^k) \Delta_h^k, \quad \Delta_h^k = \sum_j w_{hj} \delta_j^k$$

6.3.3 反向传播算法

输入-隐层：第 k 个训练样本对权重 w_{ih} 的贡献

$i \to h$, for sample k:

δ规则：

$$\Delta w_{ih}\big|_{sample\ k} = \eta \delta_h^k x_i^k$$

w_{ih}所连接的边的起始结点（输入层结点 i）的输出(此时即为样本第 i 个分量)

w_{ih}所连接的边的指向结点（隐含结点 h）收集到的误差信号

6.3.3 反向传播算法

输入-隐层：第 k 个训练样本对权重 w_{ih} 的贡献

$$\delta_h^k = \frac{\partial E}{\partial net_h^k} = f'(net_h^k) \sum_j w_{hj}\delta_j^k$$

从前一层收集误差：加权和

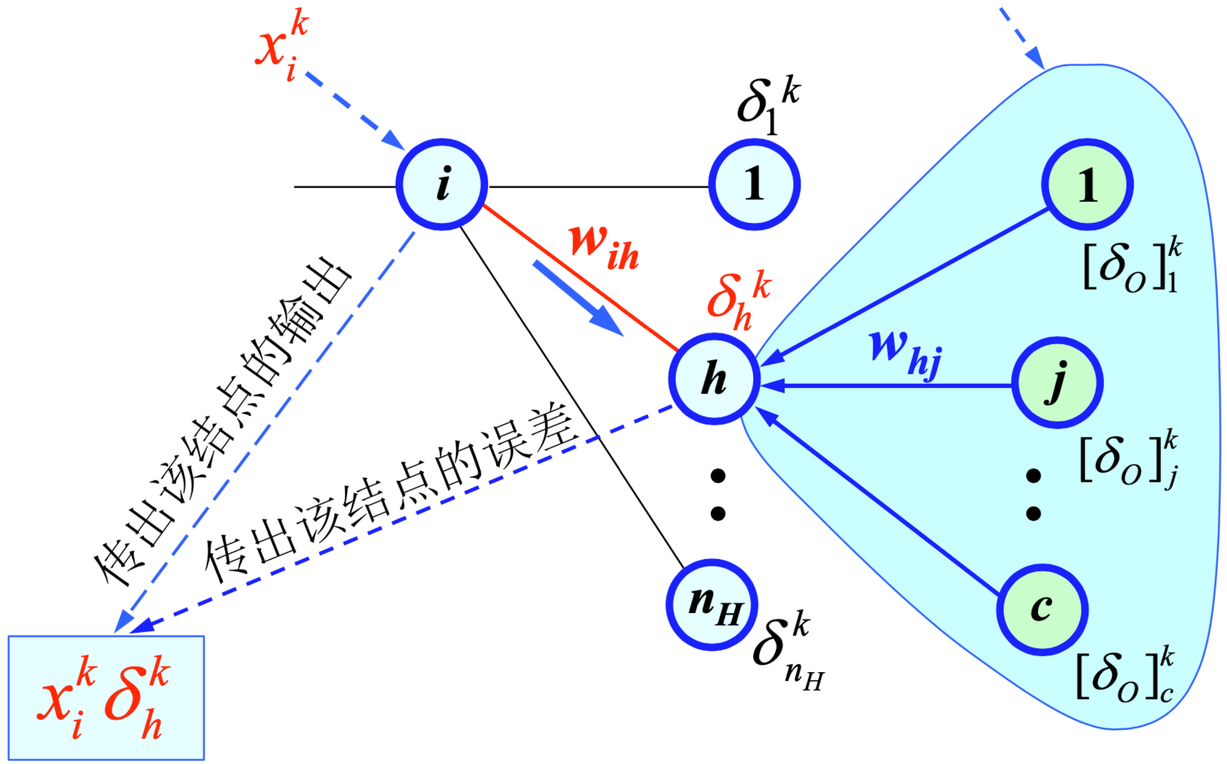
误差**在权重所联边的指向结点处**计算。

误差大小等于：该结点收集到的误差乘以激励函数对“该结点加权和”的导数。

6.3.3 反向传播算法

输入—隐层权重更新示意：

误差收集：$\delta_h^k = f'(net_h^k)\sum_j w_{hj}[\delta o]_j^k$



注：已将上页中的 δ 更为 δ_O (对输出层)

x_i^k：边*i-h* 起点的输出，即向外传递的信号值

6.3.3 反向传播算法

- 网络训练
  - BP算法对任意层的加权修正量的一般形式：

$$\Delta w_{in\to o} = \eta \sum_{all\ samples} \delta_o y_{in}$$

  - 单个训练样本的贡献：

$$\Delta w_{in\to o} = \eta \cdot \delta_o \cdot y_{in} = \eta \cdot \left(\sum_h w_{o\to h} [\delta_o]_h \right) \cdot y_{in}$$

从后一层各结点h收集误差

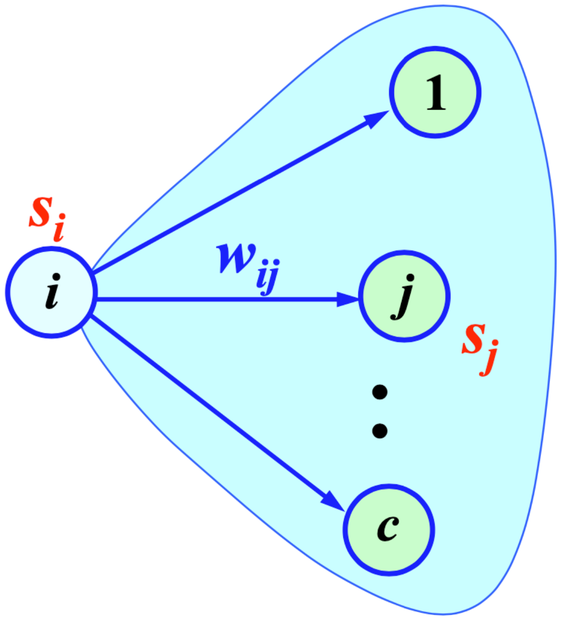
下标 in 和 o 分别指“**待更新权重**”所连边的起始结点和指向结点，y_{in}代表起始结点的实际输出，δ_o表示指向结点的误差 (由后一层收集得到)。

6.3.3 反向传播算法

- 网络训练
  - 从更一般的角度来认识网络
    - 目标函数不是误差平方损失，比如交叉熵、softmax、hinge loss等，对于权重更新是否有上述同样的文字表述？
  - 目标函数对某一层结点 i 的输出 s_i 的梯度（见右上）。
  - 目标函数对权重的梯度(导数)：

$$\nabla_{w_{ij}} E = \frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_j} \frac{\partial s_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}}$$

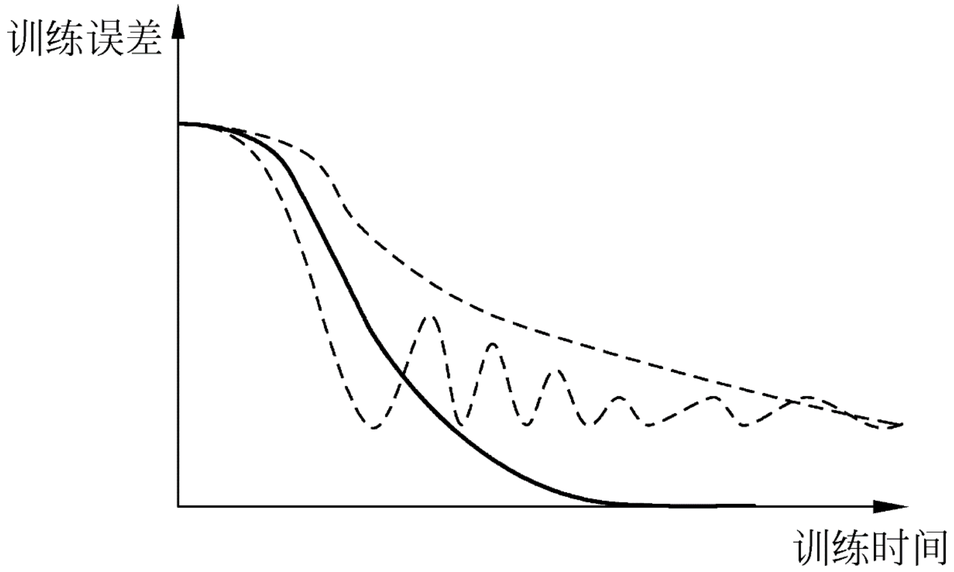
$$\frac{\partial E}{\partial s_i} = \sum_{j \in N(i)} \frac{\partial E}{\partial s_j} \frac{\partial s_j}{s_i}$$



$j \in \{$结点 i 指向的所有结点$\}$

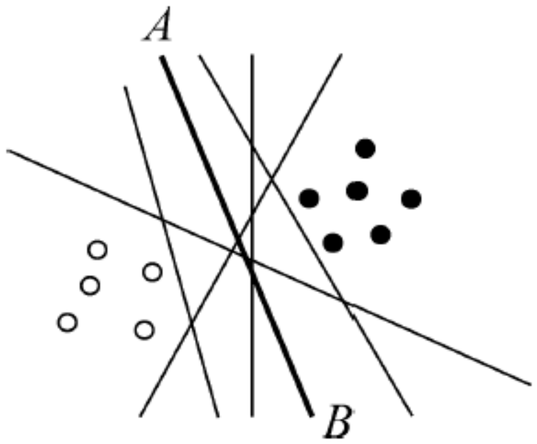
6.3.3 反向传播算法

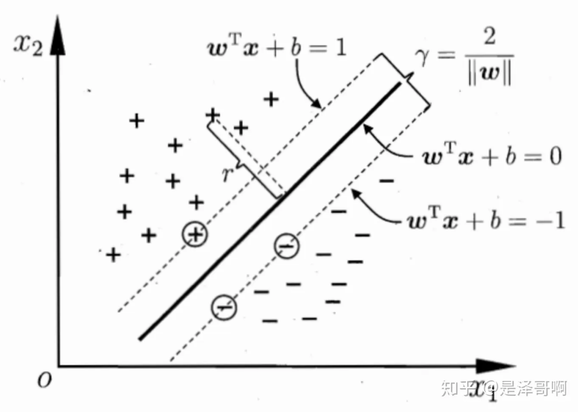
- **BP算法讨论**
  - –准则/损失函数：平方差、交叉熵
  - –激励函数：非线性、连续可导、单调
  - –输入层、隐含层、输出层节点数
  - –初始权重
  - –学习率



6.4 支持向量机

- **线性支持向量机**
  - **分类超平面**
    - $g(\boldsymbol{x}) = (\boldsymbol{w} \bullet \boldsymbol{x}) + b = 0$
    - 点$\boldsymbol{x}$到超平面的距离：$r = \dfrac{\left|\boldsymbol{\omega}^{\top}\boldsymbol{x} + b\right|}{\|\boldsymbol{\omega}\|}$
    - 规范化的分类超平面：$\begin{cases} (\boldsymbol{w} \bullet \boldsymbol{x}_i) + b \geq 1, \ y_i = +1 \\ (\boldsymbol{w} \bullet \boldsymbol{x}_i) + b \leq -1, \ y_i = -1 \end{cases}$
    - **支持向量**：每个样本都是一个向量，距离划分超平面最近的几个样本（使得等号成立）为支持向量

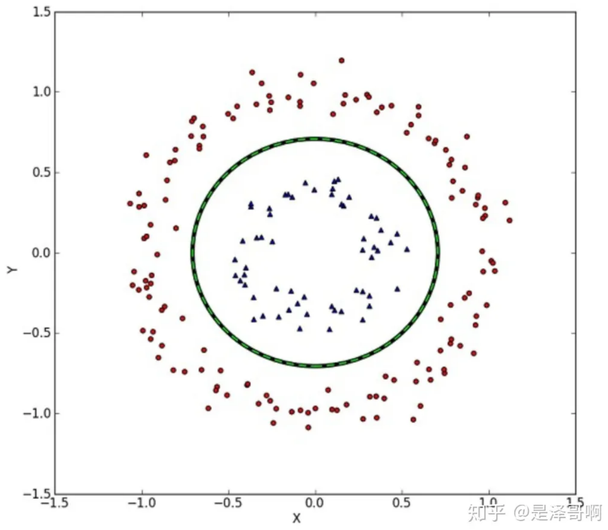


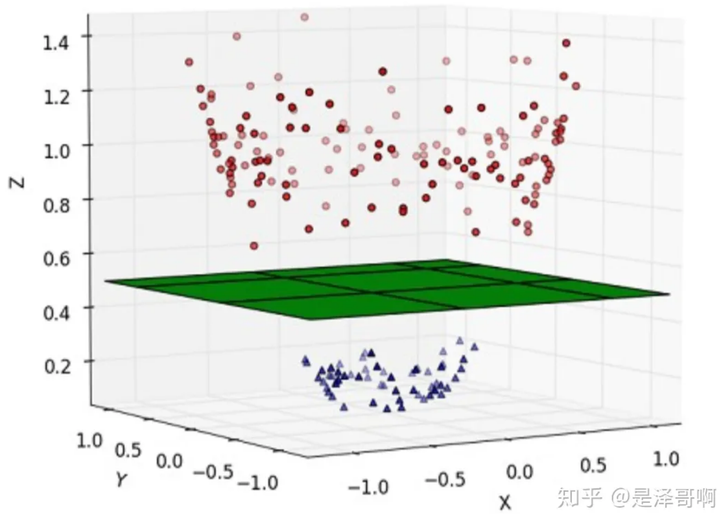


  - **最优分类超平面**
    - 一个分类超平面，如果它能将训练样本没有错误地分开，且两类训练样本中离超平面最近的样本与超平面之间的距离是**最大**的，则这个超平面称作最优分类超平面

6.4 支持向量机

- **非线性支持向量机**
  - **硬间隔和软间隔**：样本**完全**线性可分、**大部分**样本线性可分时适用
  - 样本不是线性可分的情况如何处理





6.4.1 广义线性判别函数

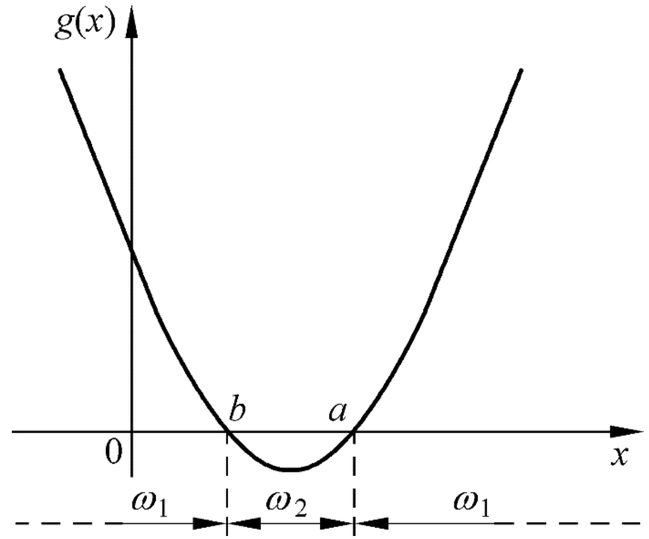
例：右图所示二分类问题。样本特征x是一维；决策规则：如果$x < b$或$x > a$，则$x \in \omega_1$，如果$b < x < a$，则$x \in \omega_2$。

- 建立二次判别函数

$$g(x) = (x-a)(x-b)$$

- 决策规则

$$\text{若} g(x) \gtrless 0, \text{则} x \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$



6.4.1 广义线性判别函数

- 二次判别函数的一般形式

$$g(x) = c_0 + c_1 x + c_2 x^2$$

- 转化为 $\boldsymbol{y}$ 的线性函数

$$g(x) = \boldsymbol{a}^T \boldsymbol{y} = \sum_{i=1}^{s} a_i y_i$$

其中，$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$，$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix}$，$g(x) = \boldsymbol{a}^T \boldsymbol{y}$称为关于$\boldsymbol{y}$的广义线性判别函数，$\boldsymbol{a}$为广义权向量

- 通过低维向高维映射，把非线性转换为线性；维数大大增加，会陷入“**维数灾难**”

$$g(x) = x^T W x + w^T x + \omega_0 = \sum_{i=1}^{d} \omega_{ii} x_i^2 + 2 \sum_{j=1}^{d-1} \sum_{i=j+1}^{d} \omega_{ij} x_i x_j + \sum_{j=1}^{d} \omega_j x_j + \omega_0$$

6.4.2 核函数变换与支持向量机

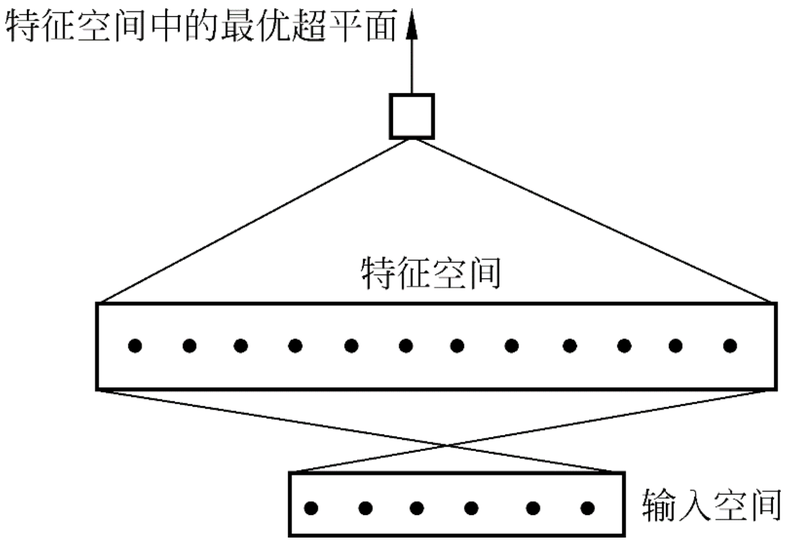
- **支持向量机**
  - 引入特征变换将非线性问题转化成线性问题
- **线性支持向量机的对偶问题**
  - 线性支持向量机的分类器

$$f(x) = \mathrm{sgn}((\boldsymbol{w} \cdot \boldsymbol{x}) + b) = \mathrm{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) + b \right),$$

  - α_i为下列二次优化问题的解

$$\max Q(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j \right)$$

$$\mathrm{s.t.} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \;\; 0 \le \alpha_i \le C, \;\; i = 1, \cdots, N$$



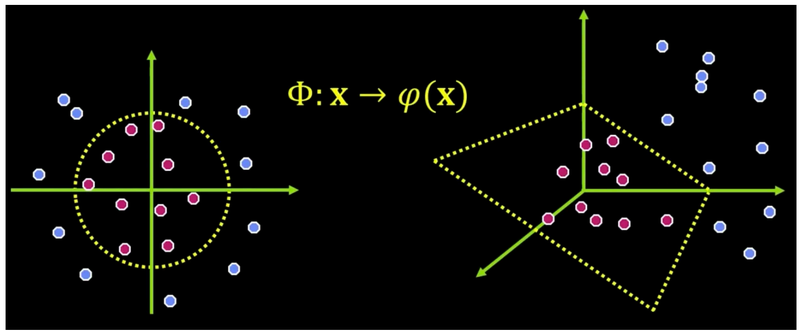
6.4.2 核函数变换与支持向量机

- 判别函数

$$f(x) = \mathrm{sgn}((\boldsymbol{w} \cdot \boldsymbol{x}) + b) = \mathrm{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) + b \right)$$

- 支持向量满足等式

$$y_i \left(\sum_{i=1}^{n} \alpha_i (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) + b \right) - 1 = 0$$



- 特征$\boldsymbol{x}$进行非线性变换为$z = \varphi(\boldsymbol{x})$，则决策函数

$$f(x) = \mathrm{sgn}\left(\boldsymbol{w}^{\varphi} \cdot \boldsymbol{z} + b \right) = \mathrm{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \left(\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}) \right) + b \right)$$

6.4.2 核函数变换与支持向量机

- 优化问题变成

$$\max\ Q(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \Big(\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j) \Big)$$

$$\text{s.t.} \quad \sum_{i=1}^{N} y_i \alpha_i = 0,\ \ 0 \le \alpha_i \le C,\ \ \ i = 1, \cdots, N$$

- 支持向量满足等式

$$y_j \left(\sum_{i=1}^{n} \alpha_i y_i \Big(\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j) \Big) + b \right) - 1 = 0$$

- **核函数**

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) \overset{\text{def}}{=} \Big(\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j) \Big)$$

6.4.2 核函数变换与支持向量机

- 变换空间里的支持向量机

$$f(x) = \mathrm{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \right)$$

- 系数α是下列优化问题的解

$$\max_{\boldsymbol{\alpha}} Q(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$$

$$\mathrm{s.t.} \quad \sum_{i=1}^{n} y_i \alpha_i = 0, \ 0 \le \alpha_i \le C, i = 1, \cdots, n$$

- b通过满足下式的样本（支持向量）求得

$$y_j \left(\sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x_i} \cdot \boldsymbol{x_j}) + b \right) - 1 = 0$$

6.4.2 核函数变换与支持向量机

- **核函数的作用**
  - $K(\boldsymbol{x}_i, \boldsymbol{x}_j) \overset{\text{def}}{=} \left(\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j) \right)$：只是做个内积运算，为什么要有核函数?
  - **降低计算量**

    如果有核函数$k(\boldsymbol{x}, \boldsymbol{y}) = (\phi(\boldsymbol{x}), \phi(\boldsymbol{y}))$，使得$\boldsymbol{x}_i$与$\boldsymbol{x}_j$ 在**特征空间**的内积$\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$等于它们在原始**样本空间**中通过函数$k(\boldsymbol{x}, \boldsymbol{y})$计算的结果，就不需要计算高维甚至无穷维空间的内积了。
  - 直接设计核函数$k(\boldsymbol{x}, \boldsymbol{y})$，不用设计变换$\phi(\boldsymbol{x})$

6.4.2 核函数变换与支持向量机

举个例子：假设我们有一个多项式核函数：

$$k(x,y)=(x\cdot y+1)^2$$

带进样本点的后：

$$k(x,y)=(\sum_{i=1}^{n}(x_i\cdot y_i)+1)^2$$

而它的展开项是：

$$\sum_{i=1}^{n}x_i^2y_i^2+\sum_{i=2}^{n}\sum_{j=1}^{i-1}(\sqrt{2}x_ix_j)(\sqrt{2}y_iy_j)+\sum_{i=1}^{n}n(\sqrt{2}x_i)(\sqrt{2}y_i)+1$$

如果没有核函数，我们则需要把向量映射成：

$$x^{'}=(x_1^2,\ldots,x_n^2,\ldots\sqrt{2}x_1,\ldots,\sqrt{2}x_n,1)$$

然后在进行内积计算，才能与多项式核函数达到相同的效果。

6.4.2 核函数变换与支持向量机

- **定理（Mercer条件）**
  - 对于任意的对称函数$K(\boldsymbol{x},\boldsymbol{x}')$，它是某个特征空间中的内积运算的充分必要条件是，对任意的$\varphi \neq 0$且$\int \varphi^2(\boldsymbol{x})d\boldsymbol{x} < \infty$，有$\iint K(\boldsymbol{x},\boldsymbol{x}')\varphi(\boldsymbol{x})\varphi(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' > 0$
- **正定核（positive definite kernels）**
  - 对$K(\boldsymbol{x}_i,\boldsymbol{x}_j)$是定义在空间$X$上的对称函数，且对任意的训练数据$\boldsymbol{x}_1,\cdots,\boldsymbol{x}_m \in X$和任意的实系数$a_1,\cdots,a_m \in R$，都有$\sum_{i,j} a_i a_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) \geq 0$
- 肯定存在$K(\boldsymbol{x},\boldsymbol{x}') = \left(\varphi(\boldsymbol{x})\varphi(\boldsymbol{x}')\right)$的空间称为可再生核希尔伯特空间**RKHS**（reproducing kernel Hilbert space）

6.4.2 核函数变换与支持向量机

- 常用核函数形式
  - 多项式核函数

$$K(\boldsymbol{x}, \boldsymbol{x}') = \left((\boldsymbol{x} \cdot \boldsymbol{x}') + 1\right)^q$$

  - 径向基（RBF）核函数

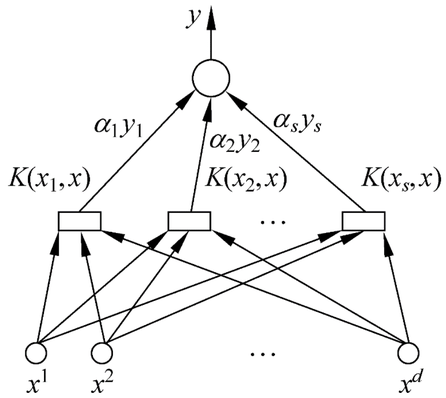
$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\| \boldsymbol{x} - \boldsymbol{x}' \|^2}{\sigma^2}\right)$$

  - Sigmoid函数

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tanh\left(v(\boldsymbol{x} \cdot \boldsymbol{x}') + c\right)$$

6.4.2 核函数变换与支持向量机

- **核函数及其参数的选择**
  - 具体问题具体分析：采用启发式方法或者累试的方法
  - 参数先尝试简单的选择，再考虑非线性核
- **核函数与相似性度量**
  - 不同核函数，可以看作不同相似性度量
- **维数和推广能力**
  - 采用核函数作为内积，避免了高维空间的计算
  - 但核函数仍然是将特征映射到高维空间



输出(决策规则)：

$$y=\mathrm{sgn}(\sum_{i=1}^{s}\alpha_i y_i K(x_i,x)+b)$$

权值$w_i=\alpha_i y_i$

基于s个支持向量$x_1, x_2, \cdots, x_s$的非线性变换(内积)

输入向量$\boldsymbol{x}=[x^1, x^2 \cdots, x^d]$

6.4.4 支持向量机的实现算法

- SVMlight ：

  http://svmlight.joachims.org/

- SVMTorch：

  http://bengio.abracadoudou.com/projects/SVMTorch.html

- LibSVM：

  https://www.csie.ntu.edu.tw/~cjlin/libsvm/

- 更多软件：

  http://www.kernel-machines.org/software

6.5 核函数机器

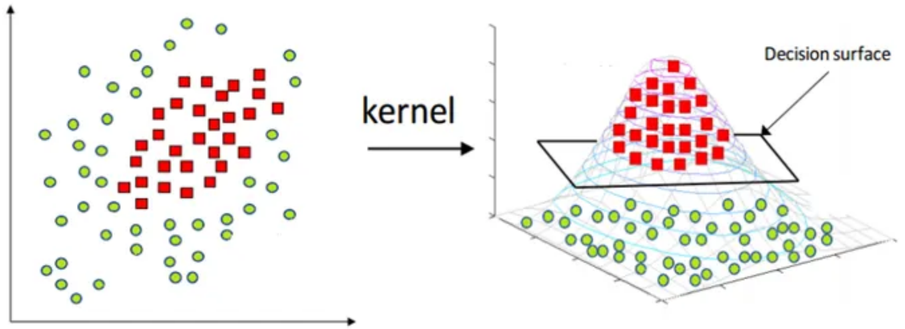
- **核方法（Kernel Method）**
  - 一类把低维空间的非线性可分问题，转化为高维空间的线性可分问题的方法
- **核函数（Kernel Function）**
  - 设$\mathcal{X}$是输入空间（即$x_i \in \mathcal{X}$，$\mathcal{X}$是$\mathbb{R}^n$的子集或离散集合），又设$\mathcal{H}$为特征空间（$\mathcal{H}$是希尔伯特空间），如果存在一个从$\mathcal{X}$到$\mathcal{H}$的映射

$$\phi(x) : \mathcal{X} \to \mathcal{H}$$

使得对所有$x, z \in \mathcal{X}$，函数$K(x, z)$满足条件

$$K(x, z) = \langle \phi(x), \phi(z) \rangle$$

则称K为核函数，其中$\phi(x)$为映射函数，$\langle \cdot, \cdot \rangle$为内积。



- **核技巧（Kernel Trick）**
  - 一种利用核函数直接计算$\langle \phi(x), \phi(z) \rangle$，以避开分别计算$\phi(x)$和 $\phi(z)$，从而加速核方法计算的技巧

6.5.1 大间隔机器与核函数机器

- **非线性变换存在的问题**
  - 计算层面：维数过高，计算复杂度高
  - 概念层面：样本数不变，维数升高，决策函数的参数增加，估计的可信度降低，面对新样本泛化能力差
- **支持向量机核心思想**
  - 大间隔方法：最大化分类间隔保证泛化能力
  - 核函数方法：核函数实现特征的非线性变换，**用变换空间中的线性问题求解原空间中的非线性问题**

6.5.2 核Fisher判别

- Fisher线性判别

$$\max_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_w \boldsymbol{w}}$$

$$S_B = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T$$

$$\boldsymbol{S}_w = \sum_{q=1,2} \sum_{x_i \in \omega_q} (\boldsymbol{x}_i - \boldsymbol{m}_q)(\boldsymbol{x}_i - \boldsymbol{m}_q)^T$$

- 其解为

$$\boldsymbol{w} = \boldsymbol{S}_w^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

6.5.2 核Fisher判别

- 样本x非线性变换至F空间：$x \longrightarrow \Phi(x) \in F$

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B^\Phi \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W^\Phi \boldsymbol{w}}$$

$$S_B^\Phi = (\boldsymbol{m}_1^\Phi - \boldsymbol{m}_2^\Phi)(\boldsymbol{m}_1^\Phi - \boldsymbol{m}_2^\Phi)^T$$

$$S_W^\Phi = \sum_{i=1,2} \sum_{x \in \omega_i} \left(\Phi(x) - \boldsymbol{m}_i^\Phi\right) \left(\Phi(x) - \boldsymbol{m}_i^\Phi\right)^T$$

$$\boldsymbol{m}_i^\Phi = \frac{1}{l_i} \sum_{j=1}^{l_i} \Phi(x_j^i)$$

- **通过核函数求解**：根据可再生希尔伯特空间的有关理论，解$\boldsymbol{w} \in F$满足

$$\boldsymbol{w} = \sum_{j=1}^{l} \alpha_j \Phi(x_j)$$

6.5.2 核Fisher判别

- 可以推出

$$\boldsymbol{w}^T \boldsymbol{m}_i^{\varnothing} = \frac{1}{l_i} \sum_{j=1}^{l} \sum_{k=1}^{l_i} \alpha_j k\left(\boldsymbol{x}_j, \boldsymbol{x}_k^i\right) = \alpha^T M_i$$

- 记

$$M := (M_1 - M_2)(M_1 - M_2)^T$$

- 类间和类内离散度矩阵

$$\boldsymbol{w}^T S_B^{\Phi} \boldsymbol{w} = \boldsymbol{\alpha}^T M \alpha,\ \boldsymbol{w}^T S_W^{\Phi} \boldsymbol{w} = \boldsymbol{\alpha}^T N \alpha$$

$$\boldsymbol{N} := \sum_{j=1,2} \boldsymbol{K}_j (\boldsymbol{I} - \boldsymbol{1}_{l_j}) \boldsymbol{K}_j^T$$

6.5.2 核Fisher判别

- 目标函数成为

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T M \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T N \boldsymbol{\alpha}}$$

- 最大化上式的解是$\boldsymbol{N}^{-1}\boldsymbol{M}$ 的最大本征值的本征向量
- 最优解的方向

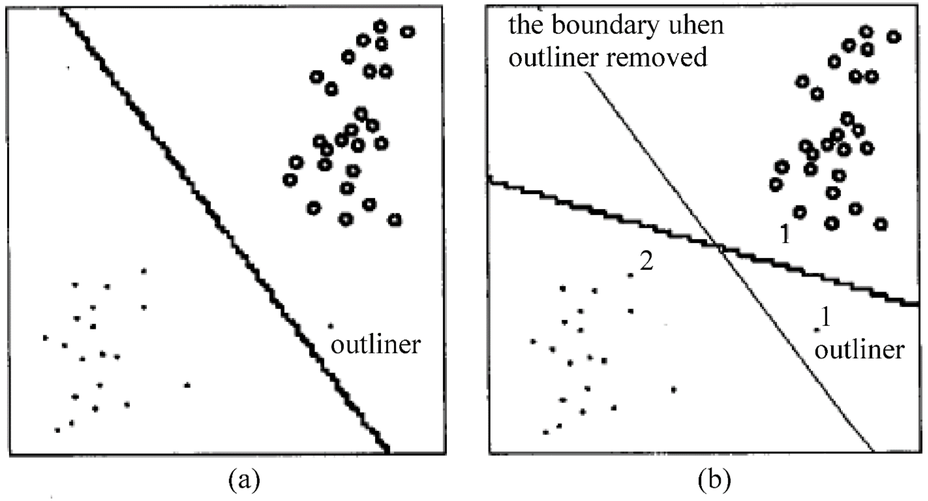
$$\boldsymbol{\alpha} \propto N^{-1}(\boldsymbol{M}_1 - \boldsymbol{M}_2)$$

- 原空间到Fisher判别的投影

$$< \boldsymbol{w}, \Phi(\boldsymbol{x}) > = \sum_{i=1}^{l} \boldsymbol{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x})$$

6.5.3 中心支持向量机

- **支持向量机**
  - **核心思想**：控制**分类间隔**实现分类；通过两类边界上的**支持向量**来定义。
  - 对样本噪声和偏离数据分布的野值非常敏感
  - 样本数非常少时，不确定性很大
- **中心支持向量机（Central SVM）**
  - 对用**中心间隔**代替边界间隔：样本中心
  - 综合基于均值和基于边界样本方法的优势
  - 极少或含野值样本下能够得到更可靠的分类器



(a) (b)

6.5.3 中心支持向量机

线性判别函数 $y = \boldsymbol{w} \cdot \boldsymbol{x} + b, y_i \in \{1, -1\}, i = 1, \cdots, n$

- 全部线性可分

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) > 0,\ i = 1, \cdots, n$$

  – 引入小常数$\varepsilon > 0$:

$$y_i\left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) \geq \varepsilon > 0,\ i = 1, \cdots, n$$

  – 极少或含野值样本下能够得到更可靠的分类器

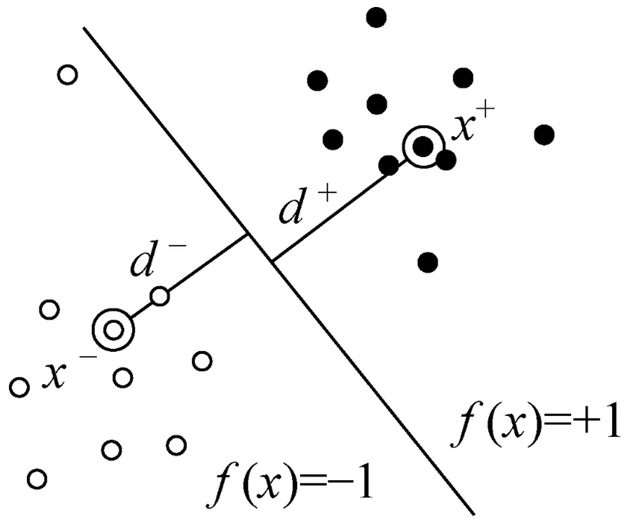
6.5.3 中心支持向量机

线性判别函数 $y = \boldsymbol{w} \cdot \boldsymbol{x} + b, y_i \in \{1, -1\}, i = 1, \cdots, n$

- 全部线性可分
  - 记两类中心为$\boldsymbol{x}^+$和$\boldsymbol{x}^-$，到分类超平面的距离

$$d^+ = \frac{\left| \boldsymbol{w} \cdot \boldsymbol{x}^+ + b \right|}{\| \boldsymbol{w} \|} = \frac{y^+(\boldsymbol{w} \cdot \boldsymbol{x}^+ + b)}{\| \boldsymbol{w} \|}$$

$$d^- = \frac{\left| \boldsymbol{w} \cdot \boldsymbol{x}^- + b \right|}{\| \boldsymbol{w} \|} = \frac{y^-(\boldsymbol{w} \cdot \boldsymbol{x}^- + b)}{\| \boldsymbol{w} \|}$$



  - **中心分离间隔**

$$d = d^+ + d^- = \frac{\sum_{i=1}^{n} l_i y_i (\boldsymbol{w} \cdot \boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|}$$

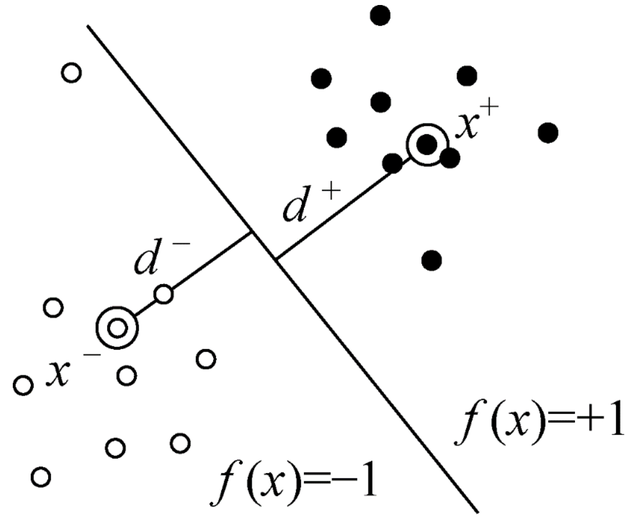
$l_i = 1/n^+$（第一类样本），$l_i = 1/n^-$（第二类样本）；n^+, n^-为两类样本数量

6.5.3 中心支持向量机

线性判别函数 $y = \boldsymbol{w} \cdot \boldsymbol{x} + b, y_i \in \{1, -1\}, i = 1, \cdots, n$

- 全部线性可分
  - 记引入约束条件：两类样本可分

$$\sum_{i=1}^{n} l_i y_i (\boldsymbol{w} \cdot \boldsymbol{x}_i) = 1$$



  - 优化问题

$$\min \frac{1}{2} \|\boldsymbol{w}\|^2$$

6.5.3 中心支持向量机

线性判别函数 $y = \boldsymbol{w} \cdot \boldsymbol{x} + b, y_i \in \{1, -1\}, i = 1, \cdots, n$

- 线性不可分，引入$\xi_i > 0$

$$y_i\left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) + \xi_i \geq \varepsilon > 0,\ i = 1, \cdots, n$$

– 目标函数修正

$$\min \frac{1}{2} \parallel \boldsymbol{w} \parallel^2 + C(\sum_{i=1}^{n} \xi_i)$$

– 对偶问题

$$\max Q\left(\boldsymbol{\alpha} + \boldsymbol{\beta}\right) = \sum_{i=1}^{n} \varepsilon\alpha_i + \boldsymbol{\beta} - \frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i + \beta l_i)(\alpha_j + \beta l_j) y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$$

6.5.3 中心支持向量机

线性判别函数 $y = \boldsymbol{w} \cdot \boldsymbol{x} + b, y_i \in \{1, -1\}, i = 1, \cdots, n$

- 线性不可分，引入$\xi_i > 0$
  - 约束条件

$$\sum_{i=1}^{n} y_i \alpha_i = 0,\ 0 \le \alpha_i \le C, i = 1, \cdots n,\ \beta \ge 0$$

  - 最优解满足

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \left(\alpha_i^* + \beta^* l_i \right) y_i \boldsymbol{x}_i = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i + \beta^* (\boldsymbol{x}^+ - \boldsymbol{x}^-)$$

$$\boldsymbol{w}^{CSVM} = (1 - \lambda) \boldsymbol{w}^{SVM} + \lambda (\boldsymbol{x}^+ - \boldsymbol{x}^-)$$