模式识别 (8)

非线性分类器

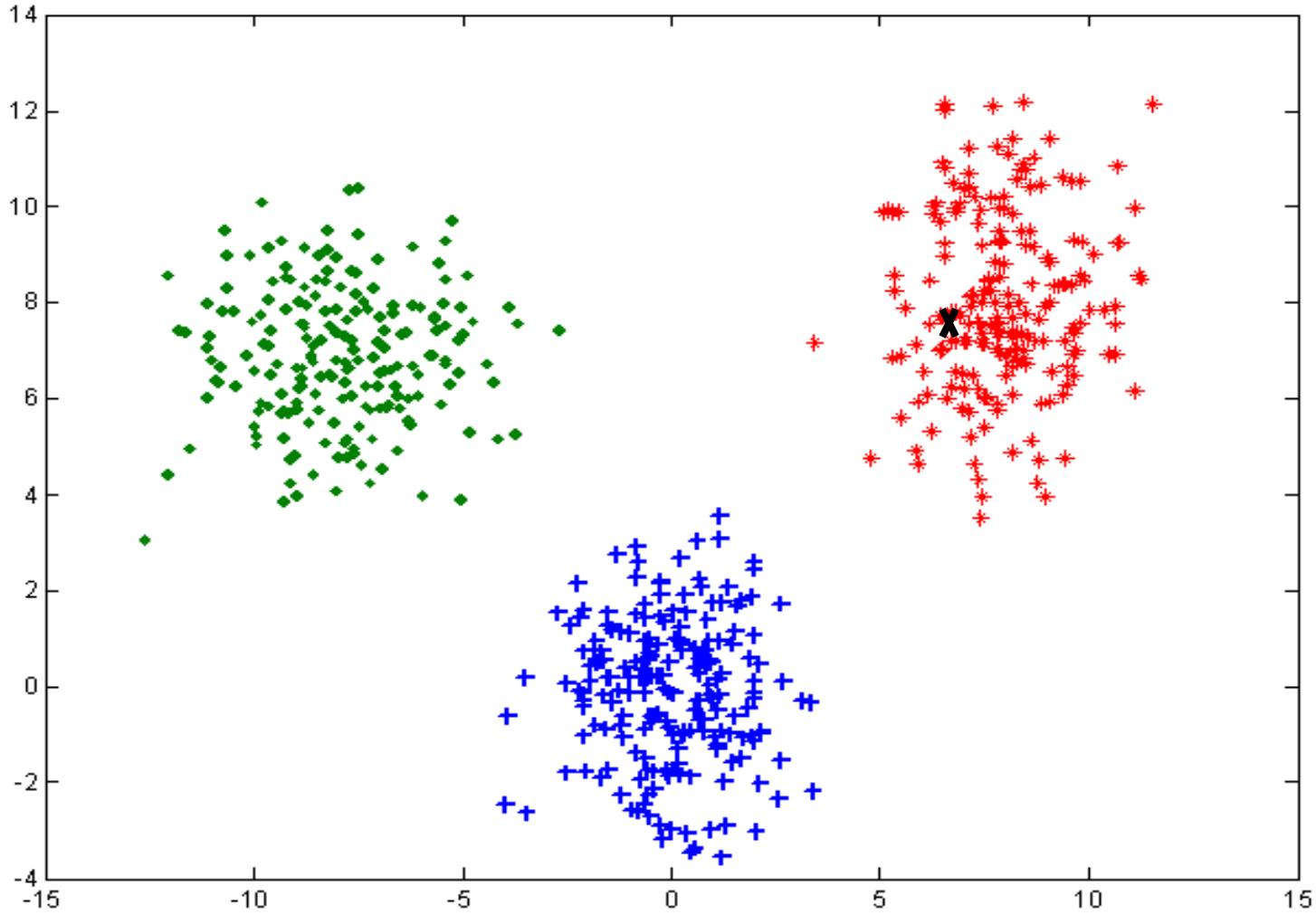
左旺孟

哈尔滨工业大学计算机学院
机器学习研究中心
综合楼712
cswmzuo@gmail.com
13134506692

非线性判别分类器

- 最近邻分类器 (kNN)
- 神经网络
- 核方法

距离分类器的思想



1. 最近邻分类器

- 基本思想
  - 判别学习
- 分类器
  - 最近邻
  - K近邻
- 几点说明
  - 距离
  - 样本

应用

- 信息检索与搜索
- 推荐系统
- 计算机视觉
  - CBIR
  - 图像智能填充
- 个性化广告

- 最小错误率分类准则

$$i = \arg\max_{1 \le j \le c} P\left(\omega_j \middle| \mathbf{x}\right), \text{ 则: } \mathbf{x} \in \omega_i$$

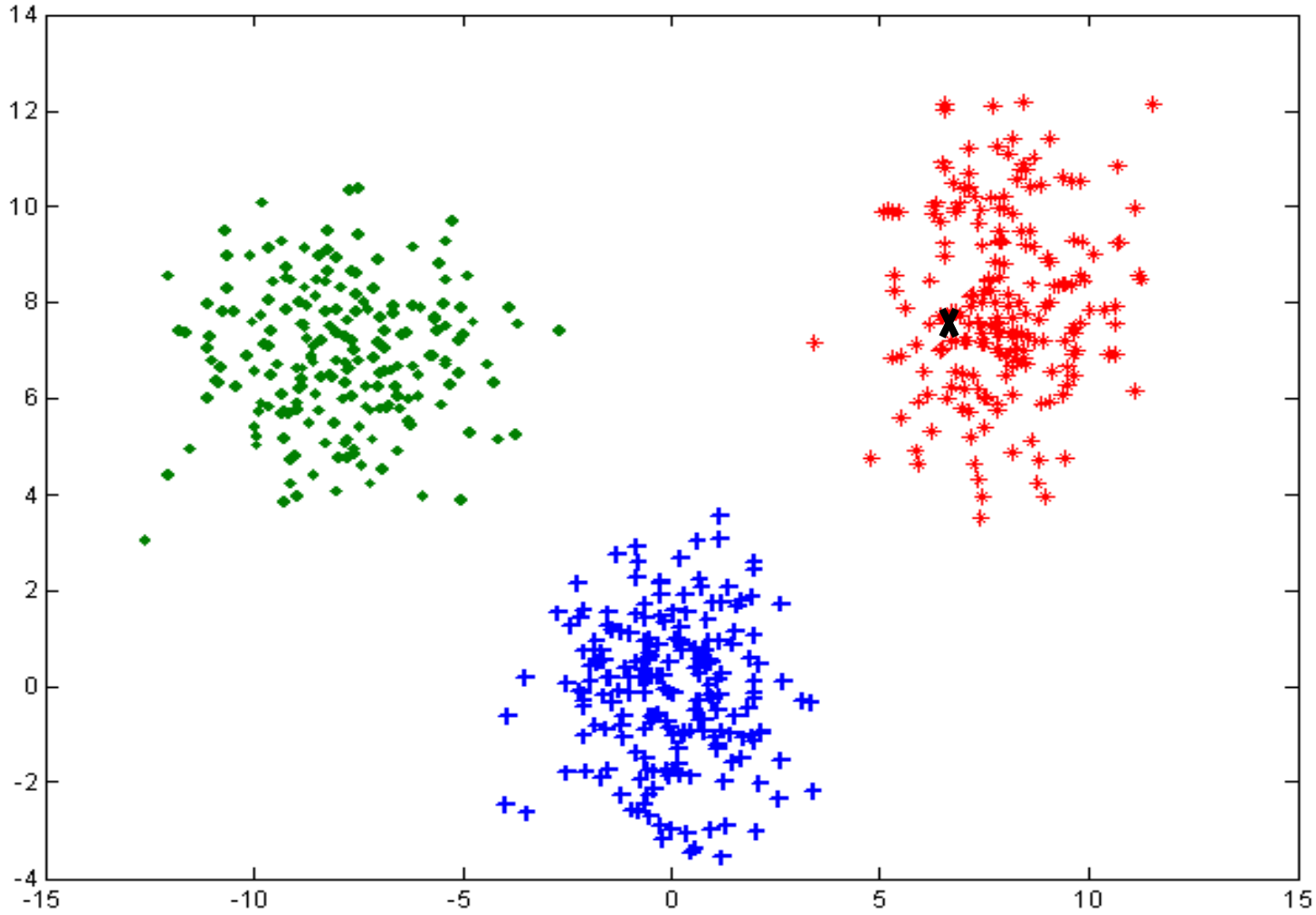
  - 直接求解

$$\arg\max_{1 \le j \le c} P\left(\omega_j \middle| \mathbf{x}\right)$$

  - 求解

$$P\left(\omega_j \middle| \mathbf{x}\right)$$

距离分类器的思想



距离度量

- 距离度量应满足如下四个性质：

1. 非负性： $D(\mathbf{a},\mathbf{b}) \geq 0$

2. 自反性：$D(\mathbf{a},\mathbf{b}) = 0$ 当且仅当 $\mathbf{a} = \mathbf{b}$

3. 对称性： $D(\mathbf{a},\mathbf{b}) = D(\mathbf{b},\mathbf{a})$

4. 三角不等式：$D(\mathbf{a},\mathbf{b}) + D(\mathbf{b},\mathbf{c}) \geq D(\mathbf{a},\mathbf{c})$

常用的距离函数

- **欧几里德距离：(Eucidean Distance)**

$$d(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{n}(x_i - y_i)^2\right]^{\frac{1}{2}}$$

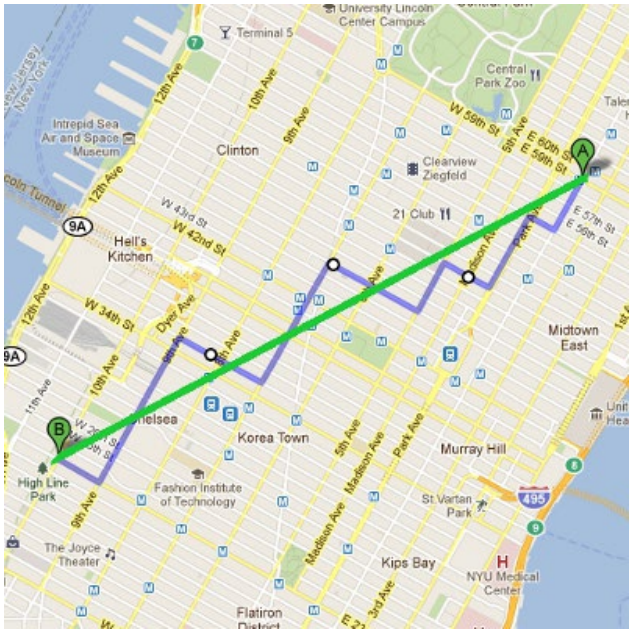
$$= \left[(\mathbf{x}-\mathbf{y})^t(\mathbf{x}-\mathbf{y})\right]^{\frac{1}{2}}$$

$\mathbf{x}^t\mathbf{y} = \sum_{i=1}^{n} x_i y_i$ 是**x**与**y**之间的内积

常用的距离函数

- 街市距离：(Manhattan Distance)

$$d(\mathbf{x},\mathbf{y})=\sum_{i=1}^{n}\left|x_i-y_i\right|$$



常用的距离函数

- **明氏距离：(Minkowski Distance)**

$$d(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{n} |x_i - y_i|^m\right]^{\frac{1}{m}}$$

  - $m >= 1$

常用的距离函数

- **马氏距离：(Mahalanobis Distance)**

$$d(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^t \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\mathbf{y})$$

常用的距离函数

- 海明距离：(Hamming Distance)

  x和**y**为2值特征矢量：

$$\mathbf{x}=\left(x_1,\cdots,x_d\right)^t,\quad \mathbf{y}=\left(y_1,\cdots,y_d\right)^t,\quad x_i,y_i\in\{0,1\}$$

  D(**x**, **y**)定义为**x**, **y**中使得不等式 $x_i \neq y_i$ 成立的i的个数。

常用的距离函数

- **角度相似函数：(Angle Distance)**
  - 相似度、不相似度

$$d(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^t\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$$

$\|\mathbf{x}\|$ 为矢量**x**的长度，也称为范数

单个标准样本距离分类器

c个类别：

$$\omega_1, \omega_2, \cdots, \omega_c$$

每个类别有一个标准样本：

$$\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_c$$

对待识样本x进行分类。

建立分类准则

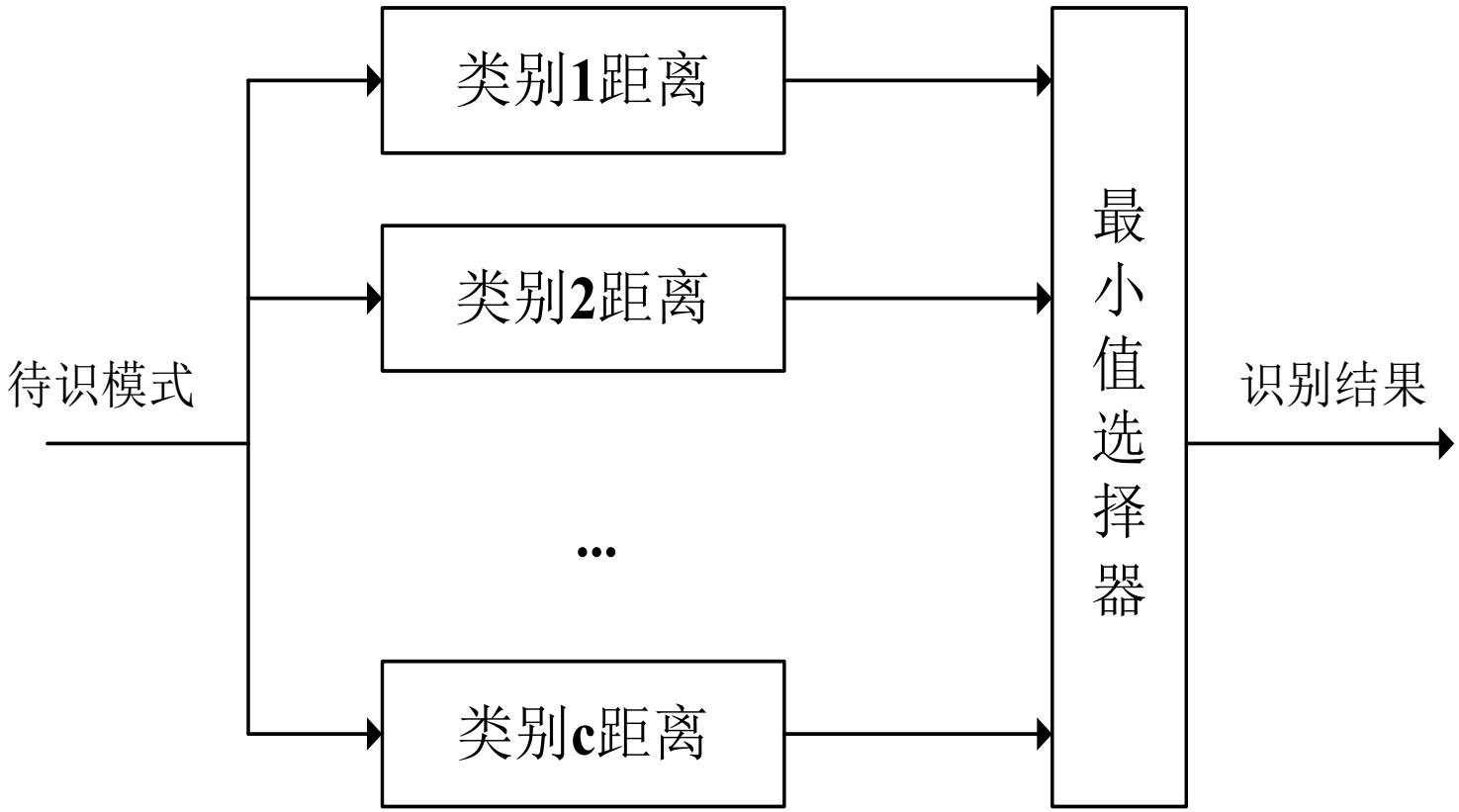
如果有：

$$i_0 = \arg\min_i d\left(\mathbf{x}, \mathbf{T}_i\right)$$

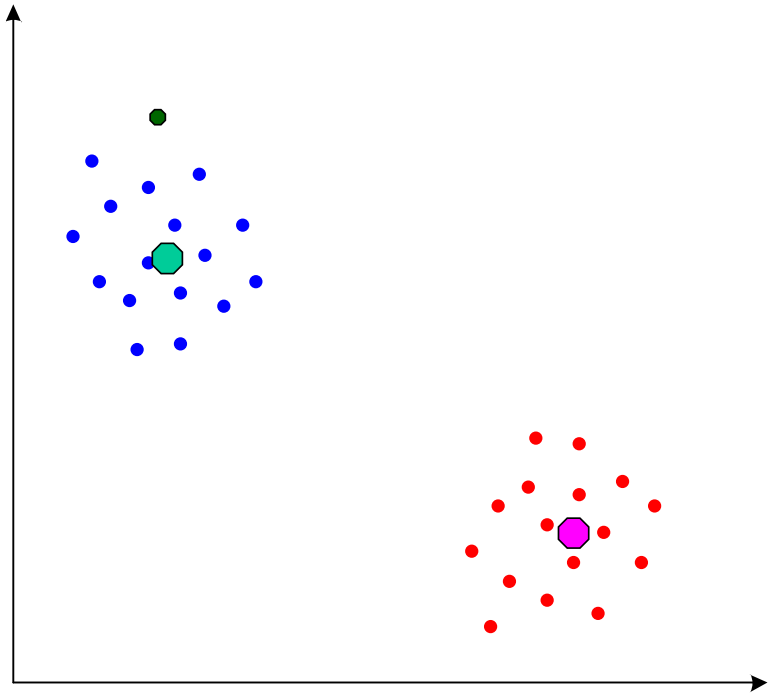
则判别：

$$\mathbf{X} \in \omega_{i_0}$$

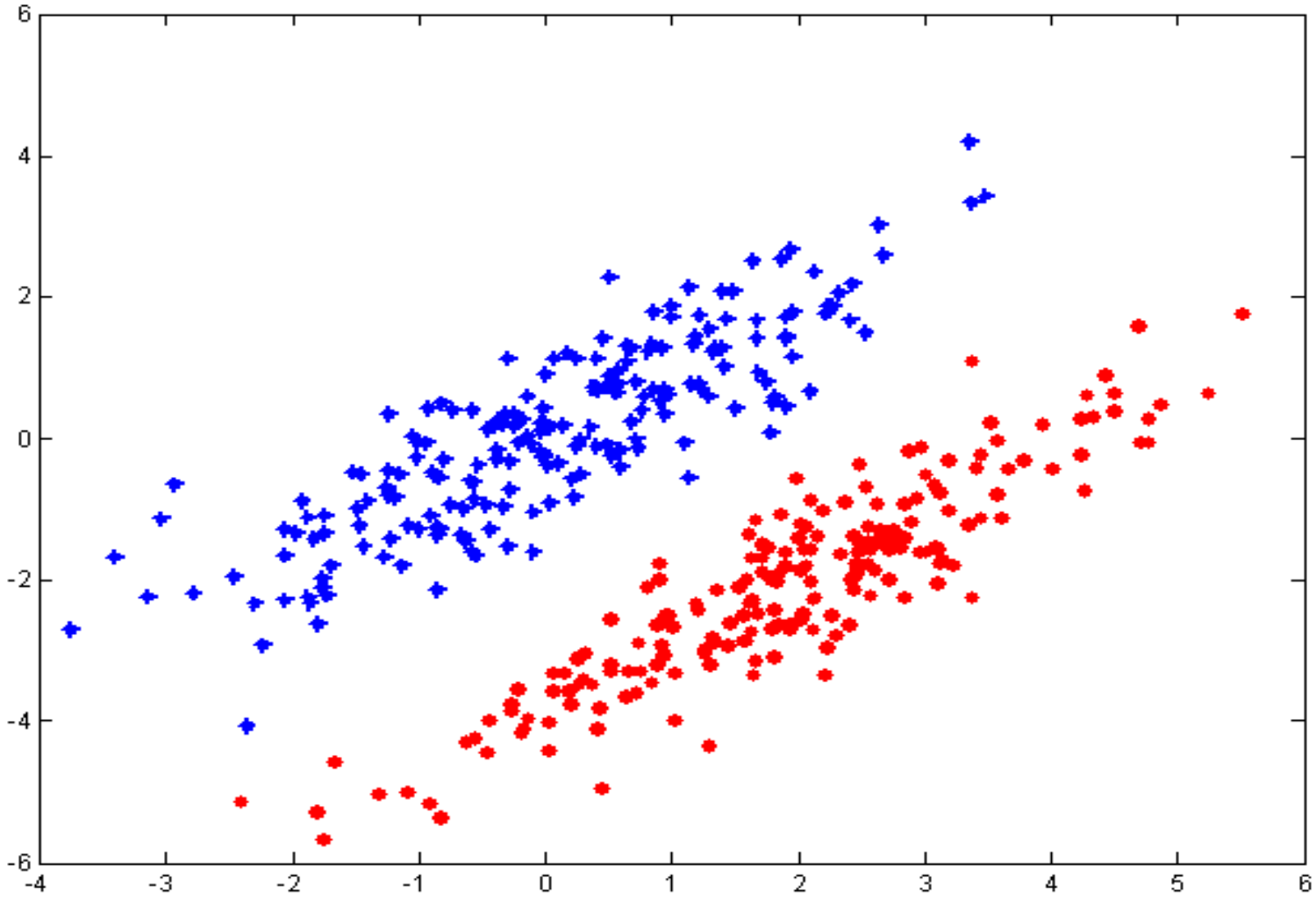
距离分类器：平均样本法



平均样本法示意



单个标准样本存在的问题



最近邻法

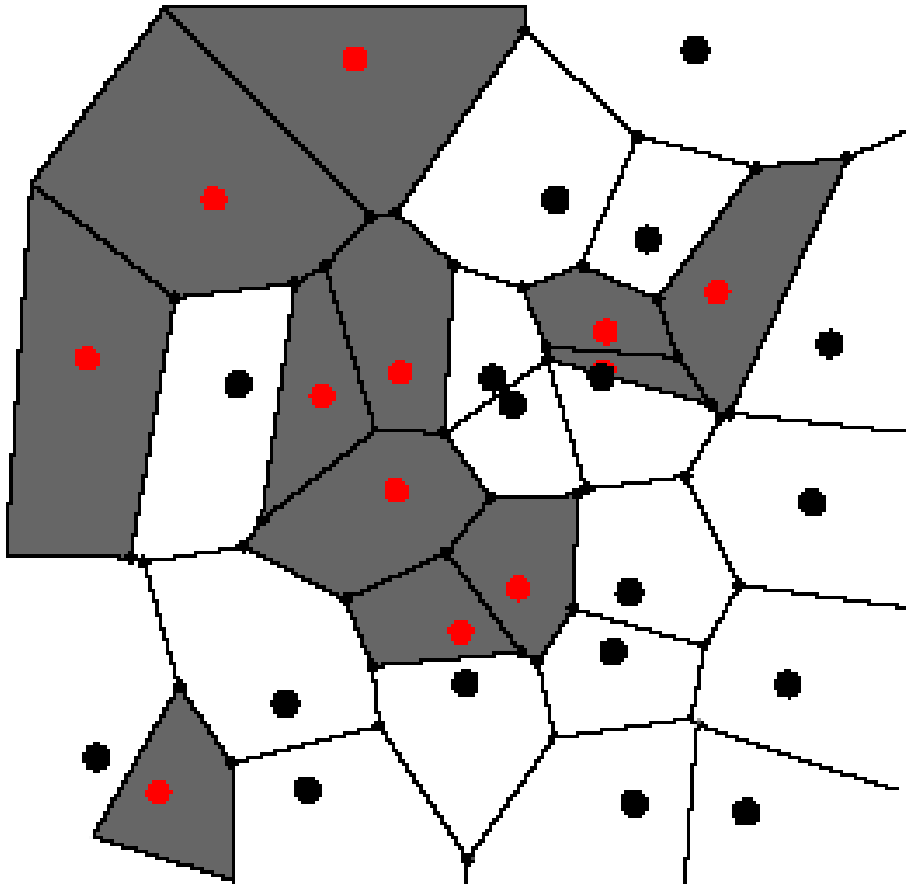
- 计算待识别样本x与每一个训练样本的距离；
- 寻找与x距离最近的训练样本；
- 以这个训练样本的类别作为x所属的类别。

最近邻规则

- 分类规则：在训练样本集中寻找与待识别样本**x**距离最近的样本**x'**，将**x**分类到**x'**所属的类别。

- 最近邻规则相当于k=1的k-近邻分类，其分类界面可以用Voronoi网格表示。

Voronoi网格（欧氏距离）



近邻分类器: 判别学习解释

- **后验概率**的估计

  Parzen窗法估计的是每个类别的类条件概率密度$p(\mathbf{x}|\omega_i)$，而k-近邻法是直接估计每个类别的后验概率$p(\omega_i|\mathbf{x})$

  将一个体积为V的区域放到待识样本点**x**周围，包含k个训练样本点，其中k_i个属于ω_i类，总的训练样本数为n，则有：

$$p_n(\mathbf{x},\omega_i)=\frac{k_i/n}{V}$$

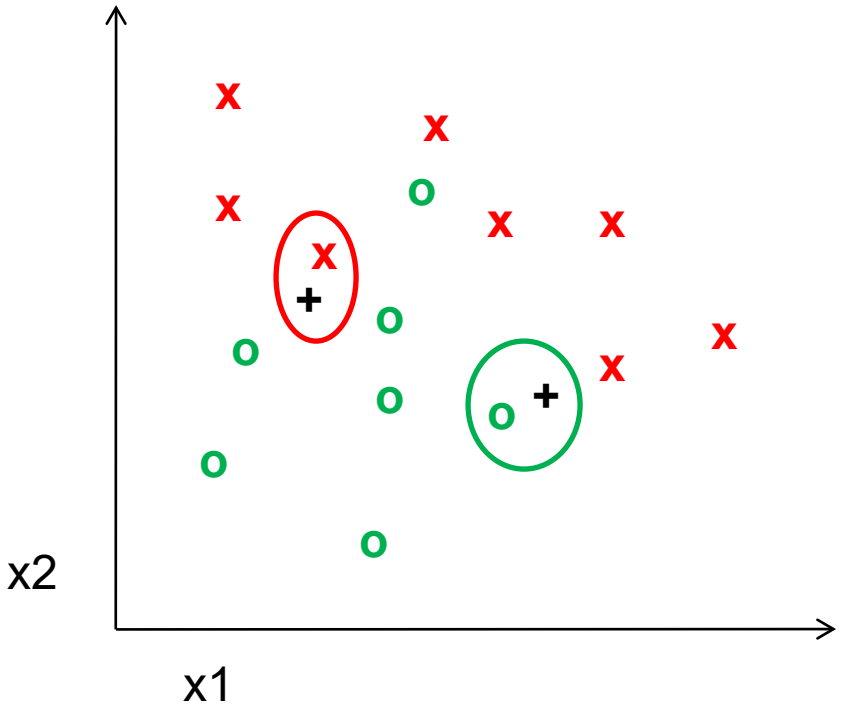
$$p(\omega_i|\mathbf{x})=\frac{p_n(\mathbf{x},\omega_i)}{p_n(\mathbf{x})}=\frac{p_n(\mathbf{x},\omega_i)}{\sum_{j=1}^{c}p_n(\mathbf{x},\omega_j)}=\frac{k_i}{k}$$

理论（Cover & Hart, 1967）

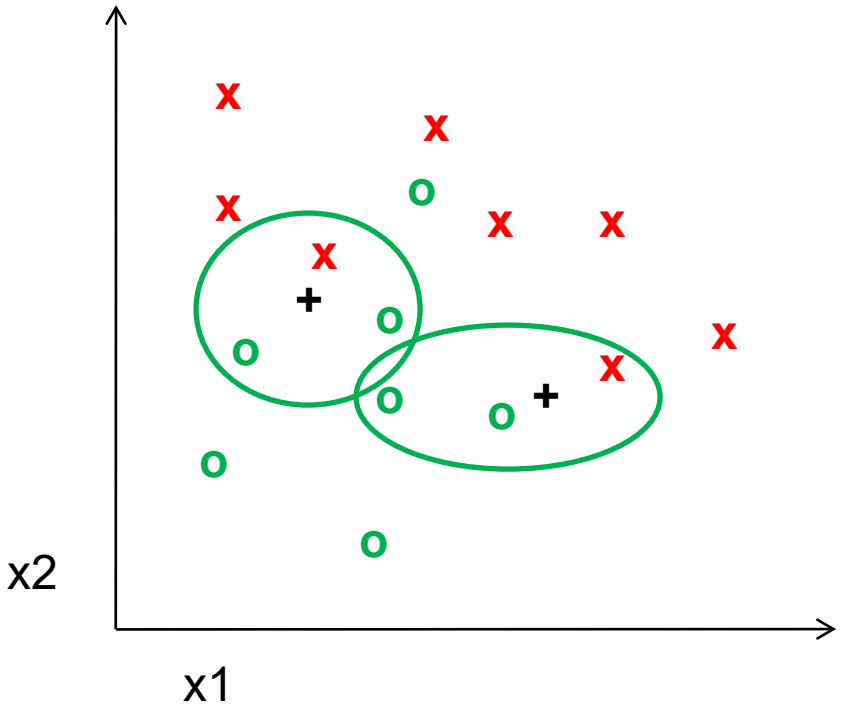
- 在近似意义上，最近邻分类器的误差不会高于贝叶斯误差率的二倍。

- T. Cover, P. Hart, Nearest Neighbor Pattern Classification, IEEE Trans. Information Theory, 11: 21-27, 1967.

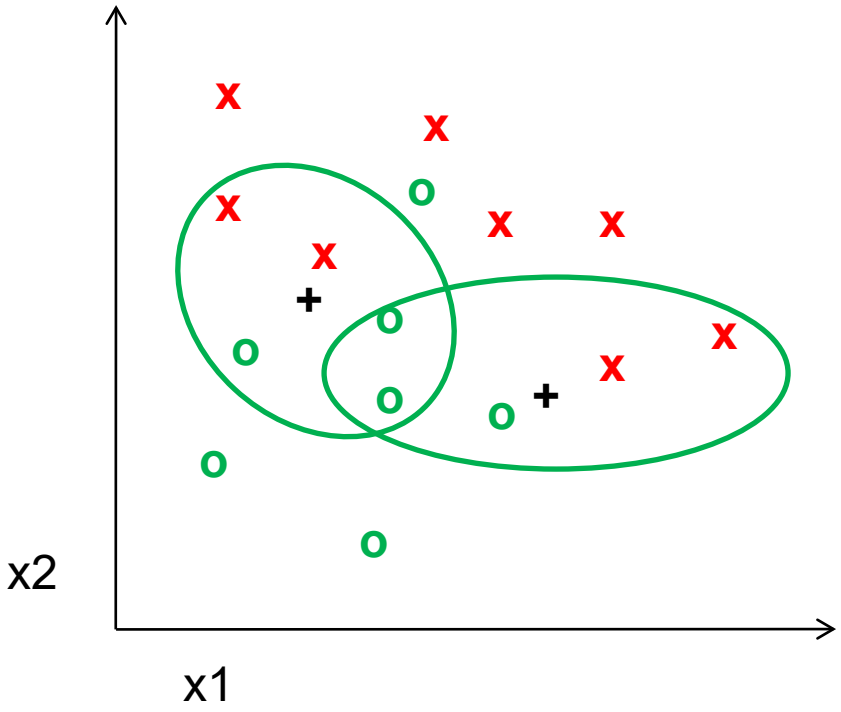
k-近邻分类，k=1



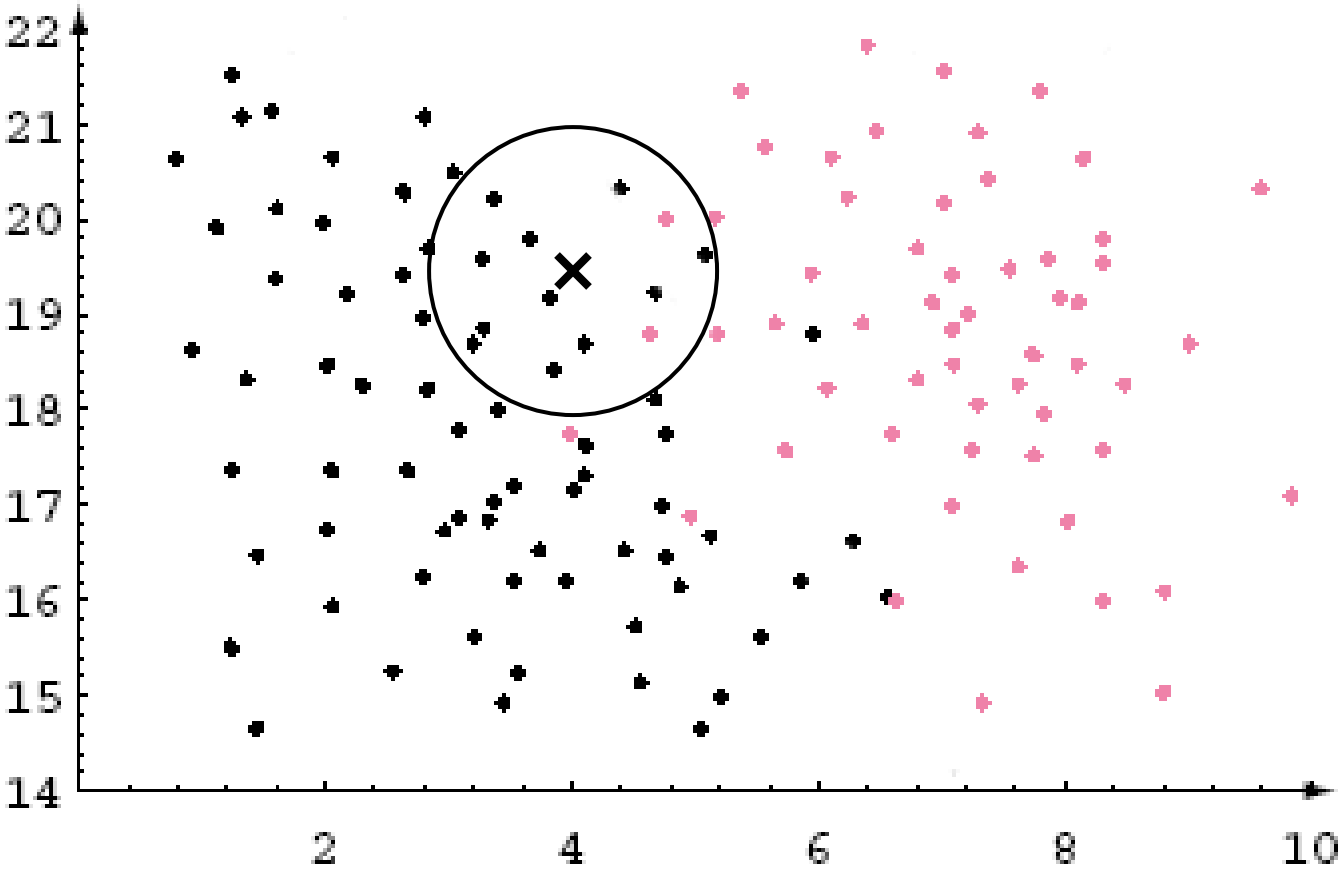
k-近邻分类，k=3



k-近邻分类，k=5



k-近邻分类，k=13



k-近邻法

1. 计算x与所有训练样本的距离；
2. 所计算出的距离从小到大排序；
3. 统计前K个中各类样本的个数N_i；
4. 如果：

$$i_0 = \arg\max_{1 \le i \le C} N_i$$

则判别：

$$\mathbf{X} \in \omega_{i_0}$$

几种方法的比较

- 平均样本法：简单，计算量小，但精度不高；
- 最近邻法：识别率高，计算量大，有时过于极端，抗噪声能力不强；
- K-近邻法：识别率高，计算量大，需要选择一个合适的参数K。

马氏距离测度学习

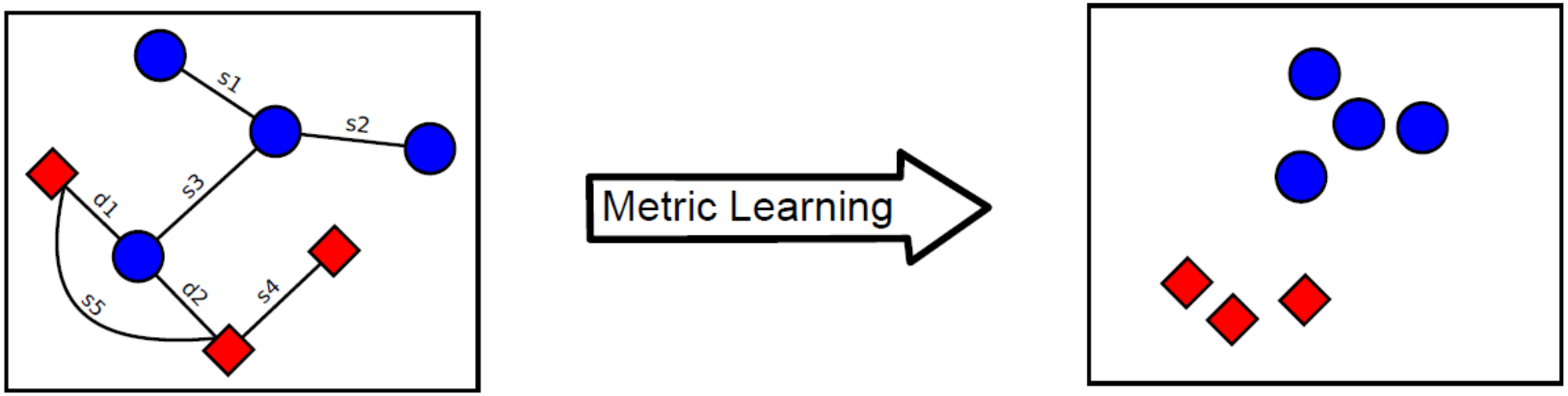
- 马氏距离

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)\|_2^2 = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)$$

- 模型

$$\min_{M} \quad \ell(M, \mathcal{S}, \mathcal{D}, \mathcal{R}) \quad + \quad \lambda R(M)$$

- 直观展示

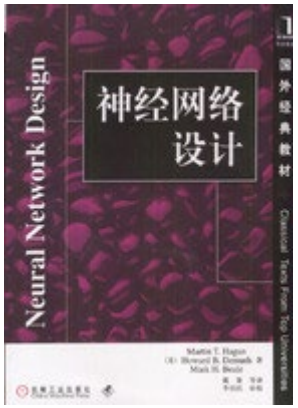


2. 多层感知器网络

- 多层感知器网络（MLP，Multilayer Perceptron）
  - 多层神经网络
  - BP神经网络
- 主要任务
  - P1: 网络设计
  - P2: 训练

参考书

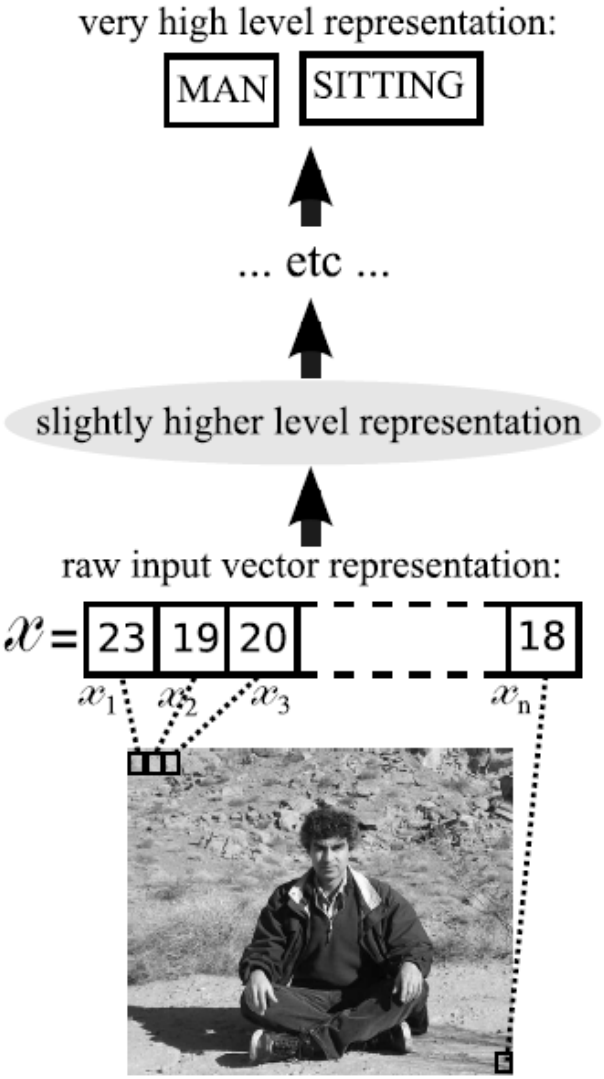
- **《神经网络设计》**，[美]哈根等 著，戴葵等 译，机械工业出版社，2002-9-1

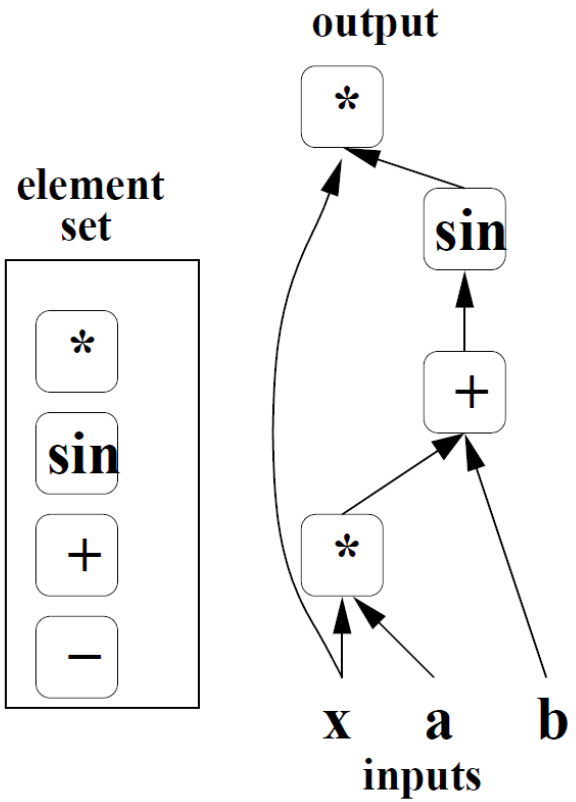


数据的表示

- 单变量: x, y, z, x_1
- 向量: $\mathbf{x}, \mathbf{y}, \mathbf{x}_1$
  - 向量中的组元 $\mathbf{x} = [x_1, \cdots, x_d]^t$
- 矩阵 $\mathbf{A}, \mathbf{X}, \mathbf{M}$

多层的意义

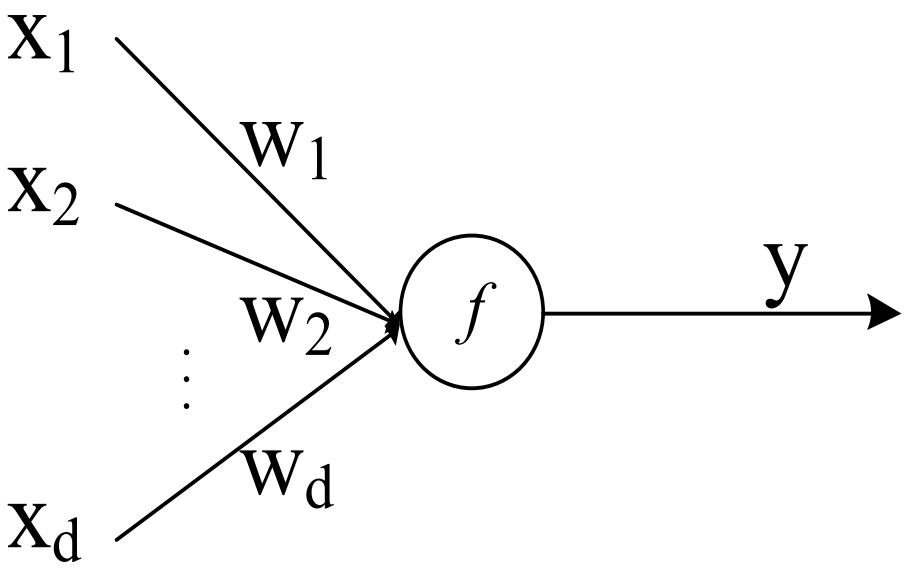




$$x * \sin(a * x + b)$$

2. 多层感知器网络
（MLP，Multilayer Perceptron）

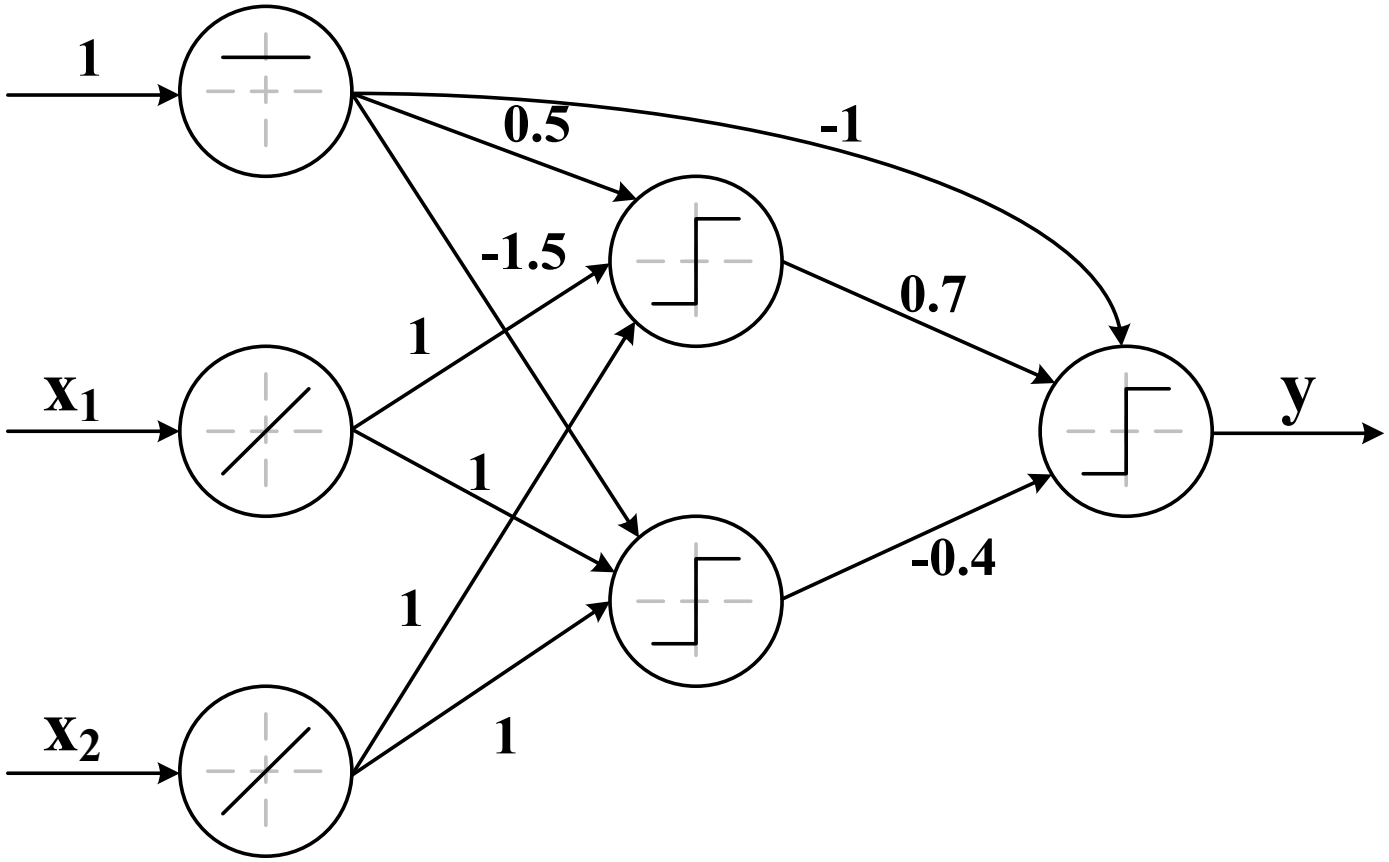
- 神经元模型



$$y = f\left(\mathbf{w}^t\mathbf{x}\right) = f\left(\sum_{i=1}^{d} w_i x_i\right),$$

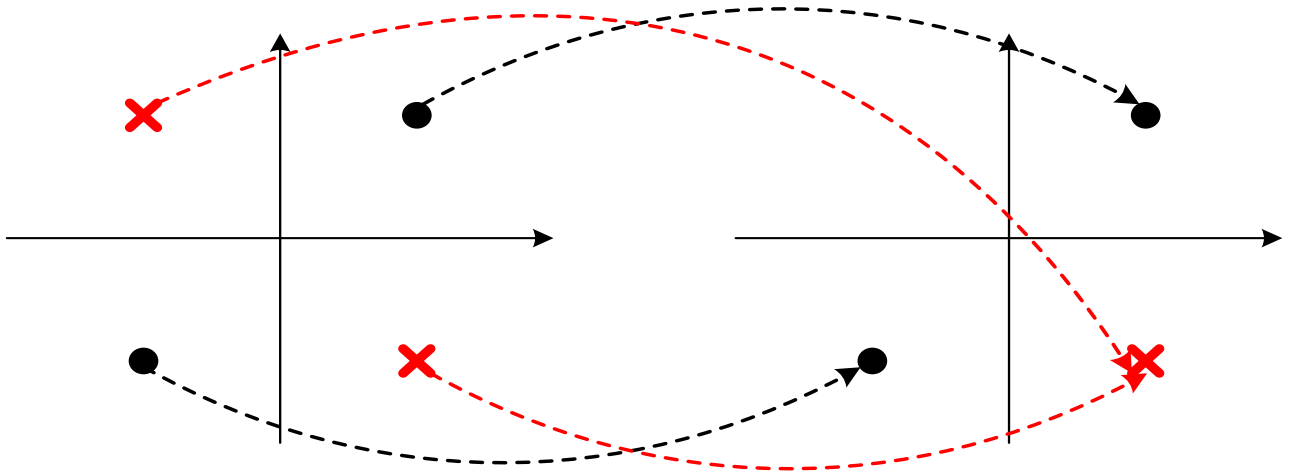
f称为激活函数

解决异或问题的多层感知器

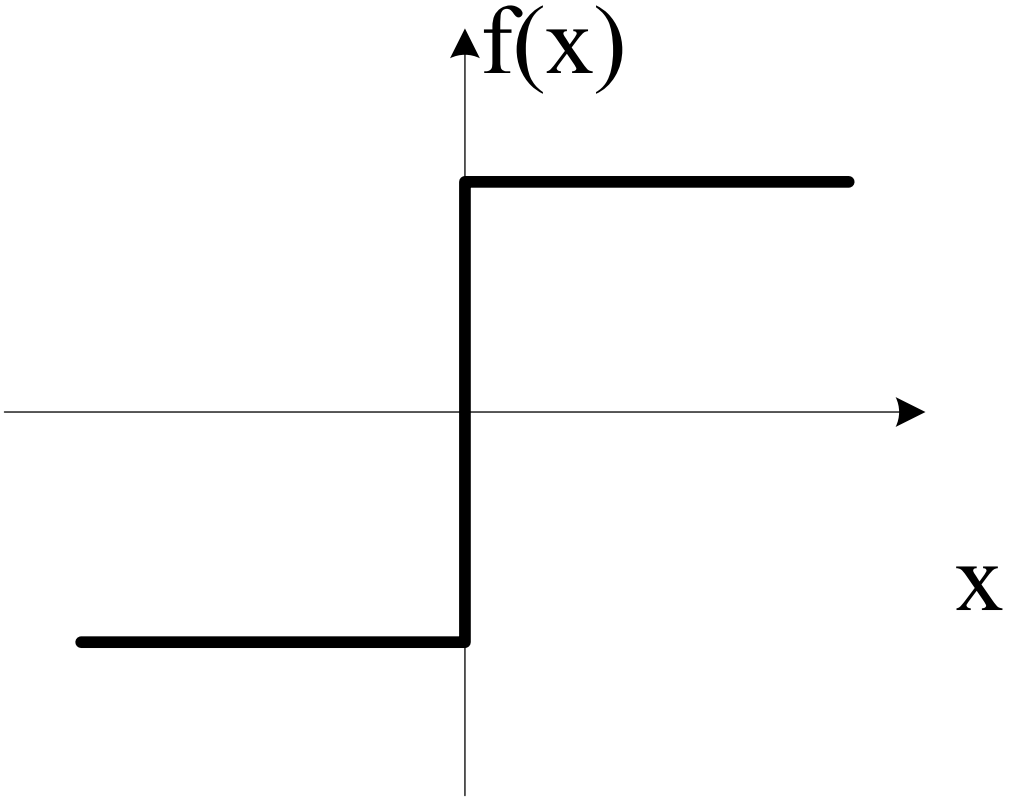


多层感知器的分类原理

- 隐含层实现对输入空间的非线性映射，输出层实现线性分类；
- 非线性映射方式和线性判别函数可以同时学习。

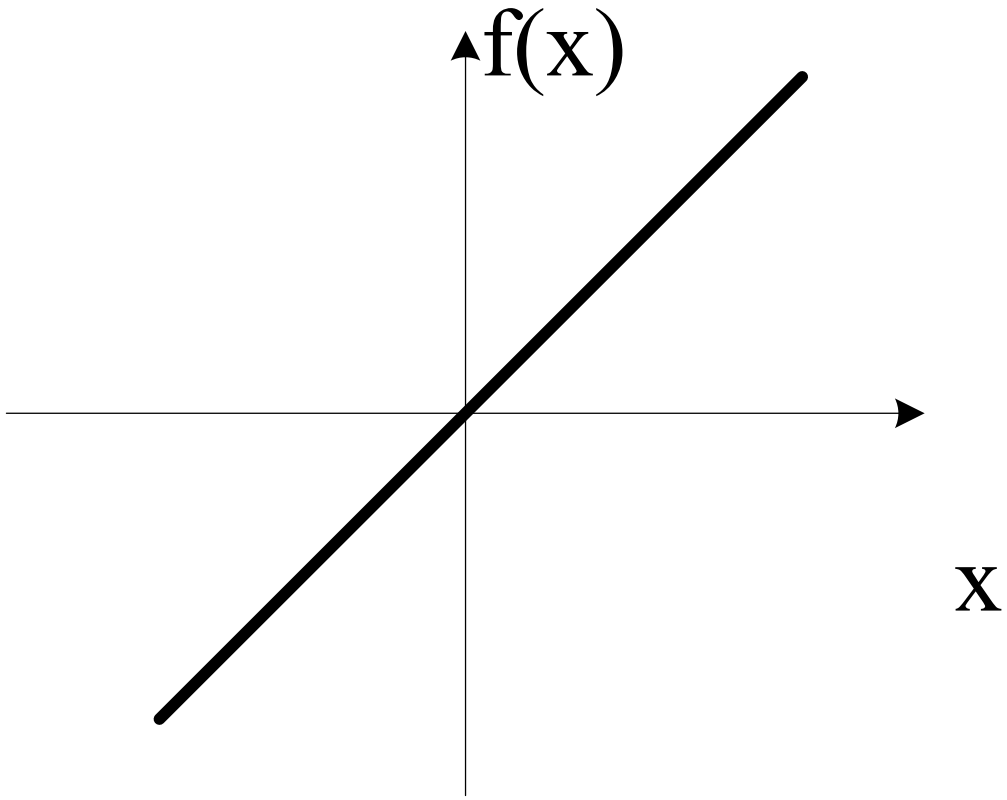


激活函数—阈值函数



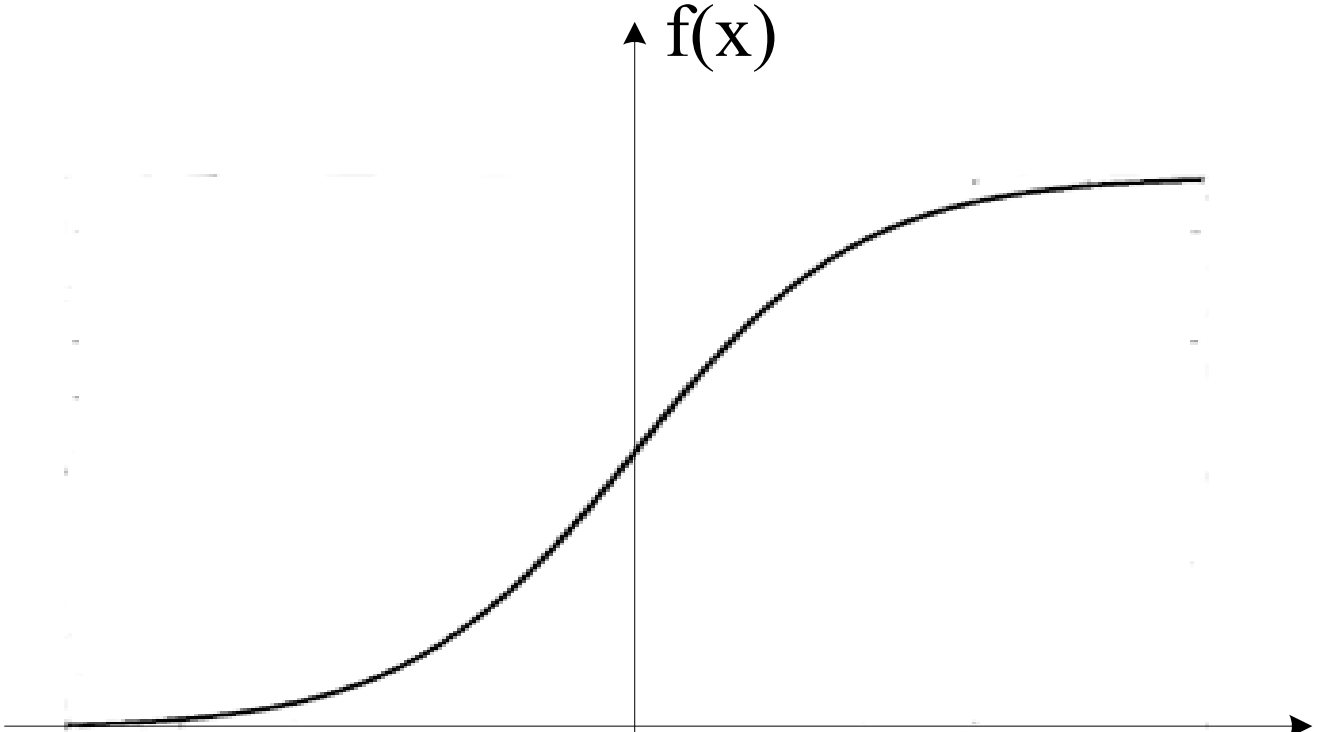
$$f(x)=\begin{cases}1, & x>0\\ -1, & x\le 0\end{cases}$$

激活函数—线性函数



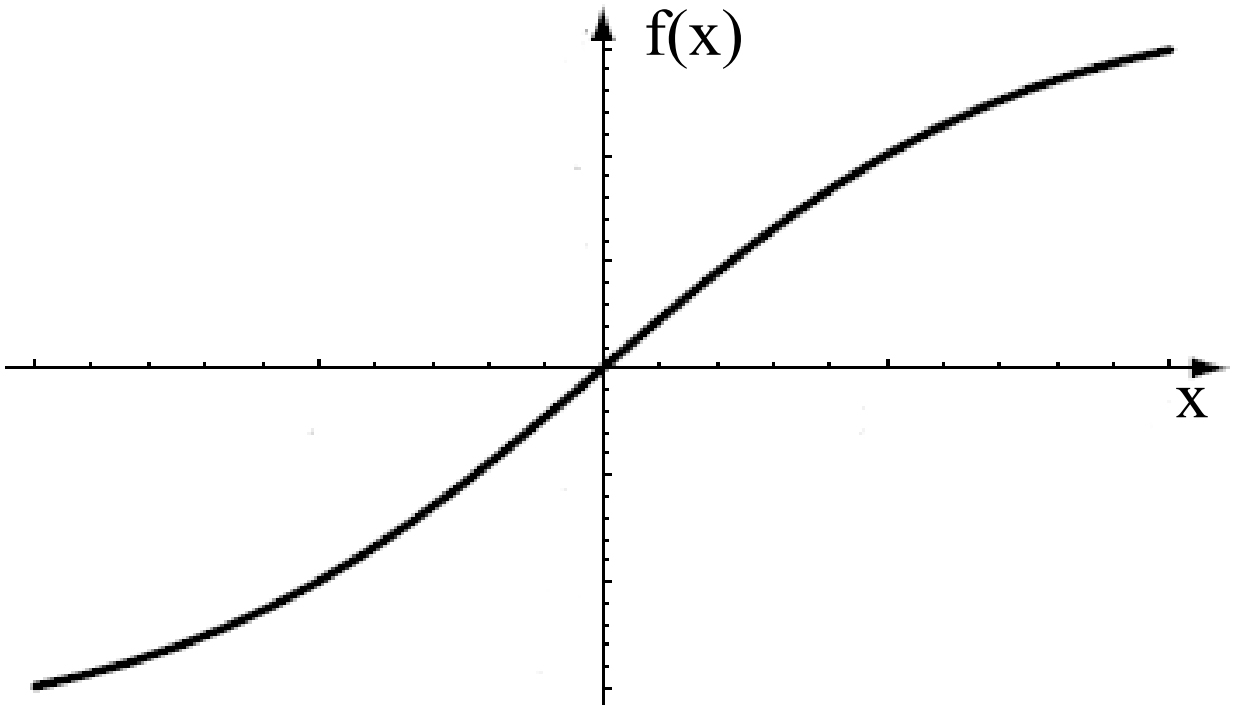
$$f(x) = x$$

激活函数—对数Sigmoid函数



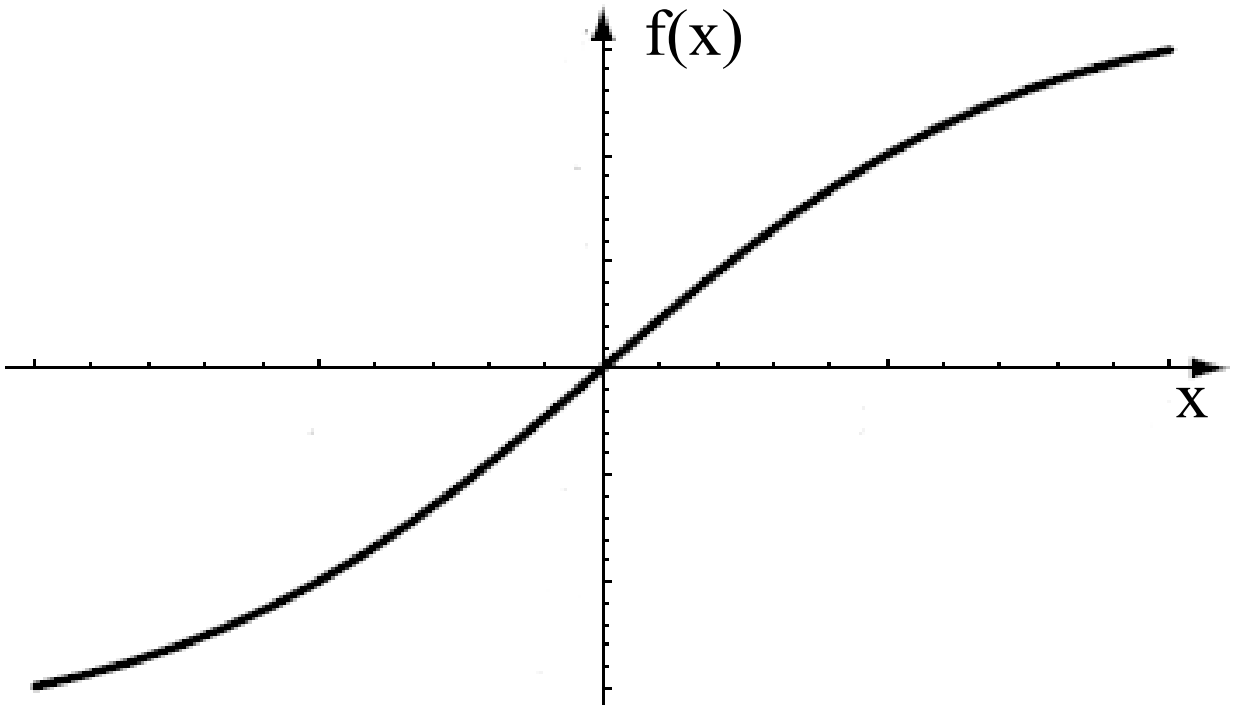
$$f(x)=\frac{1}{1+e^{-x}}$$

激活函数—双曲正切Sigmoid函数



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

激活函数—双曲正切Sigmoid函数

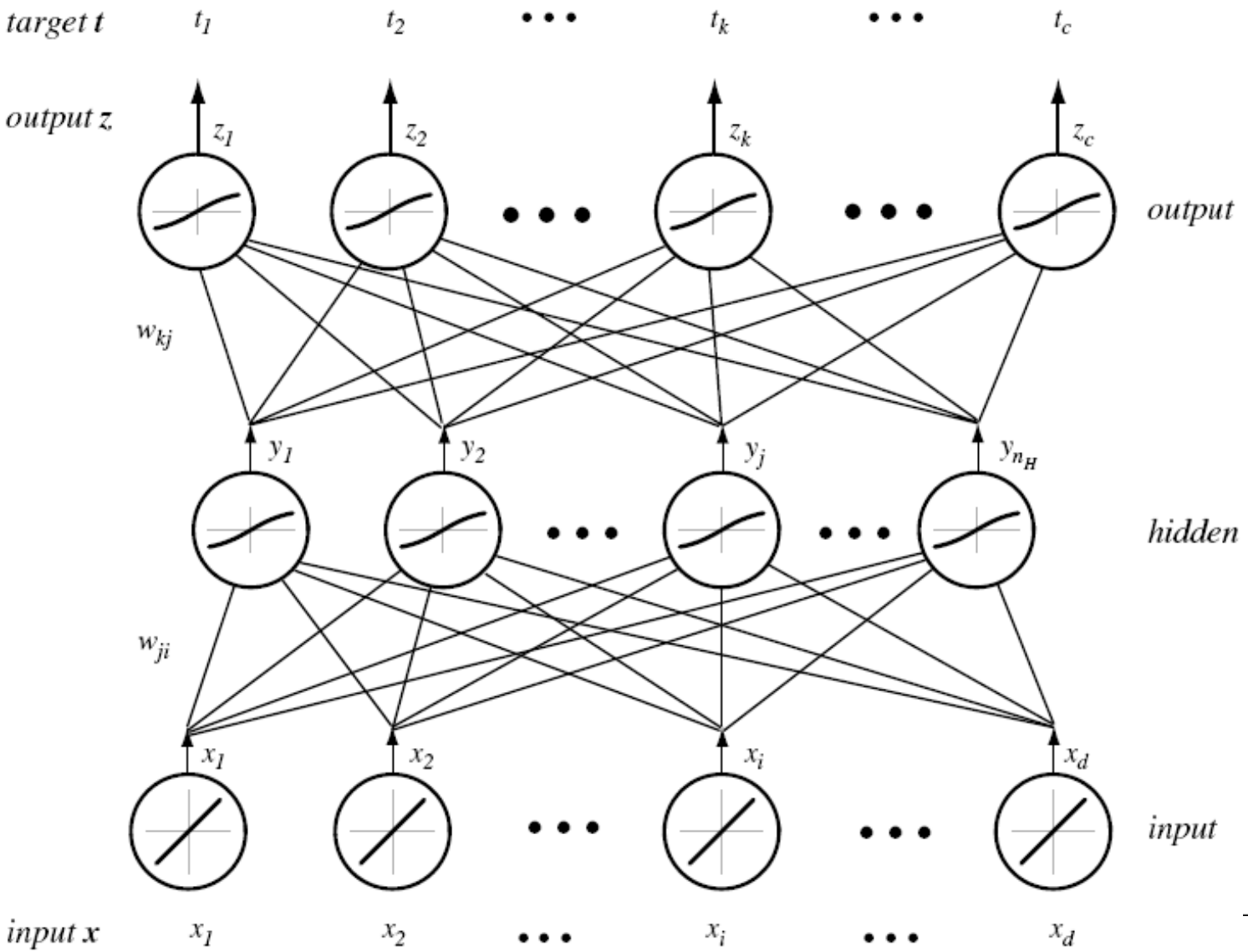


$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

作业

- 分析对数Sigmoid函数和双曲正切Sigmoid函数的关系，并推导对数Sigmoid函数的一阶导数。

标准的三层感知器网络


target t
t_1
t_2
t_k
t_c
output z
z_1
z_2
z_k
z_c
output
w_{kj}
y_1
y_2
y_j
y_{n_H}
hidden
w_{ji}
x_1
x_2
x_i
x_d
input
input x
x_1
x_2
x_i
x_d

多层感知器网络

- 主要任务
  - P1: 网络设计
  - P2: 训练

多层感知器网络的设计

- 选定层数：通常采用三层网络
  - 增加网络层数并不能提高网络的分类能力？
- 输入层：输入层**节点数**为输入特征的维数d，**映射函数**采用线性函数；
- 隐含层：隐含层**节点数**需要设定，一般来说，隐层节点数越多，网络的分类能力越强，**映射函数**一般采用Sigmoid函数；
- 输出层：输出层**节点数**可以等于类别数c，也可以采用编码输出的方式，少于类别数c，**输出函数**可以采用线性函数或Sigmoid函数。

三层网络的判别函数形式

$$g_k(\mathbf{x}) = f_2\left(\sum_{j=1}^{n_H} w_{kj} f_1\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

第k个输出层神经元的输出，其中d为特征维数，n_H为隐层节点数。

2.2 学习机制

- 大脑神经网络的学习过程
  - 环境
  - 网络结构/参数调整
  - 响应、反馈

- 神经网络学习机制
  - 错误校正
  - 记忆学习
  - Hebbian学习
  - 竞争学习

Who Invented Backpropagation

- http://people.idsia.ch/~juergen/who-invented-backpropagation.html

- The first NN-specific application of efficient BP as above was described in 1981 (Werbos, 1981).
- Related work was published several years later (Parker, 1985; LeCun, 1985).
- A paper of 1986 significantly contributed to the popularisation of BP for NNs (Rumelhart et al., 1986).

2.2 MLP的训练--误差反向传播算法

（BP，Backpropagation algorithm）

$$z_k = f_2\left(\sum_{j=1}^{n_H} w_{kj} f_1\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

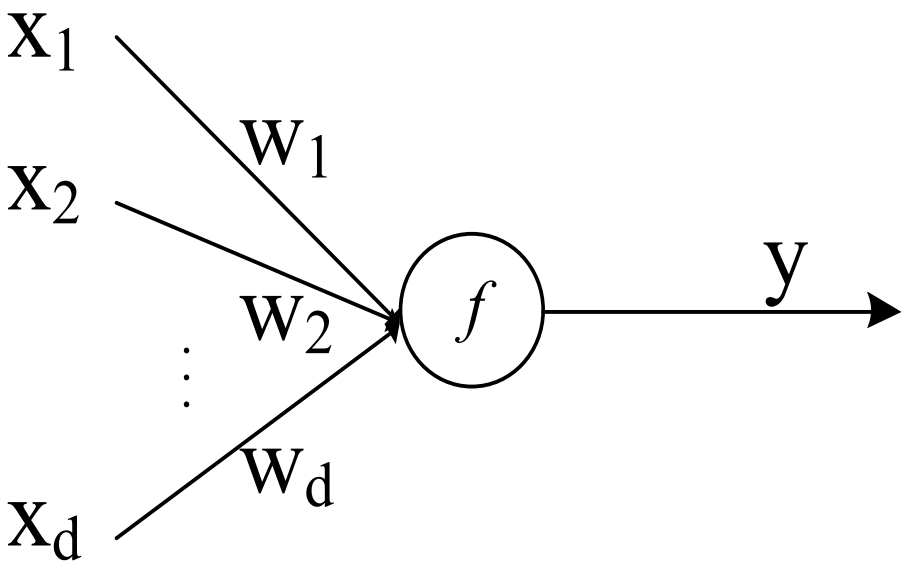
- BP算法的实质是一个**均方误差最小算法** (LMS)

- 符号定义：训练样本x，期望输出$t=(t_1,\ldots, t_c)$，网络实际输出$z=(z_1,\ldots, z_c)$，隐层输出$y=(y_1,\ldots, y_{n_H})$，第k个神经元的净输出net_k。

- 目标函数：$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{t}-\mathbf{z}\|^2 = \frac{1}{2}\sum_{i=1}^{c}(t_i - z_i)^2$

迭代公式： $\mathbf{w}(m+1) = \mathbf{w}(m) + \Delta\mathbf{w}(m) = \mathbf{w}(m) - \eta\frac{\partial J}{\partial \mathbf{w}}$

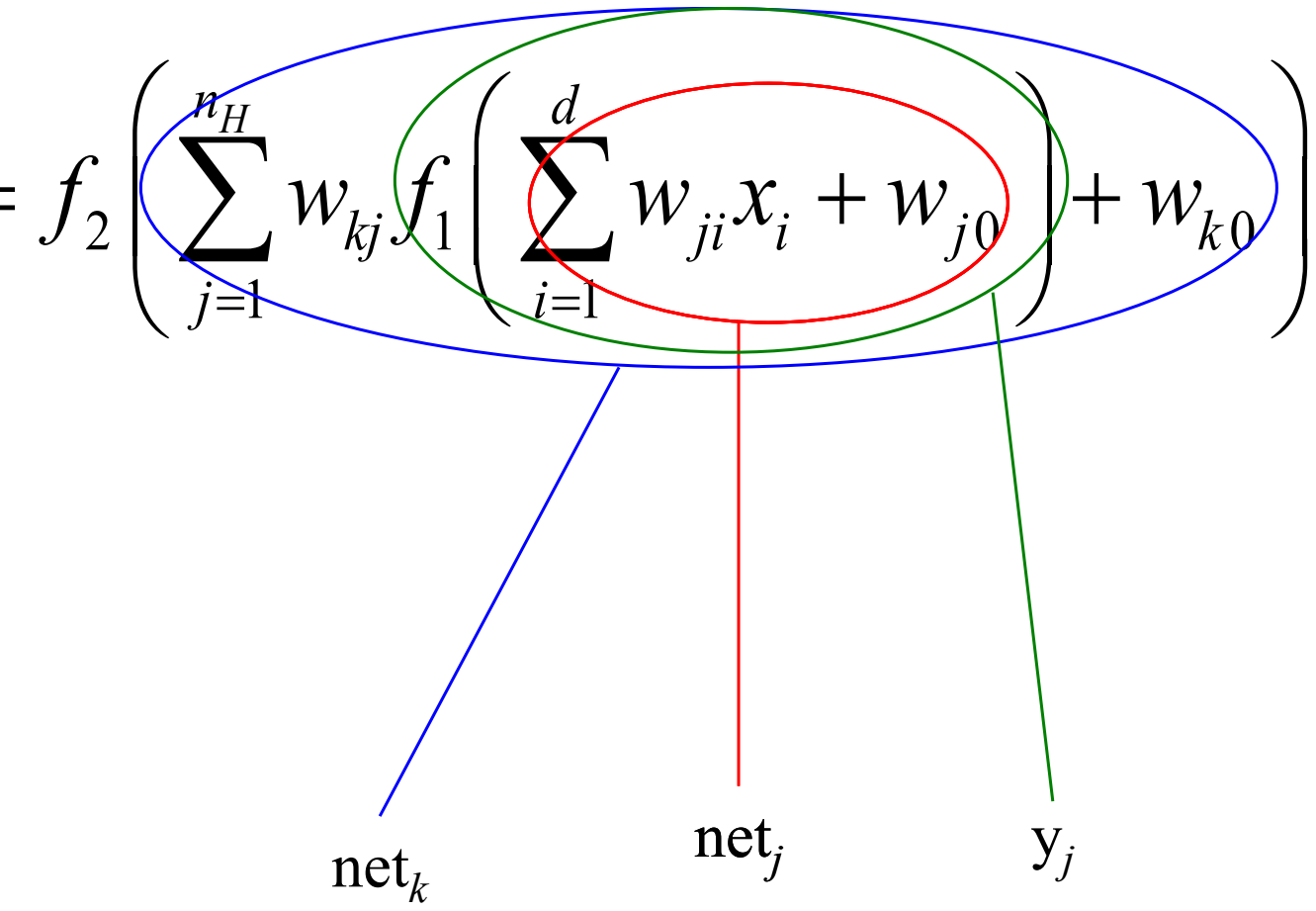
神经元

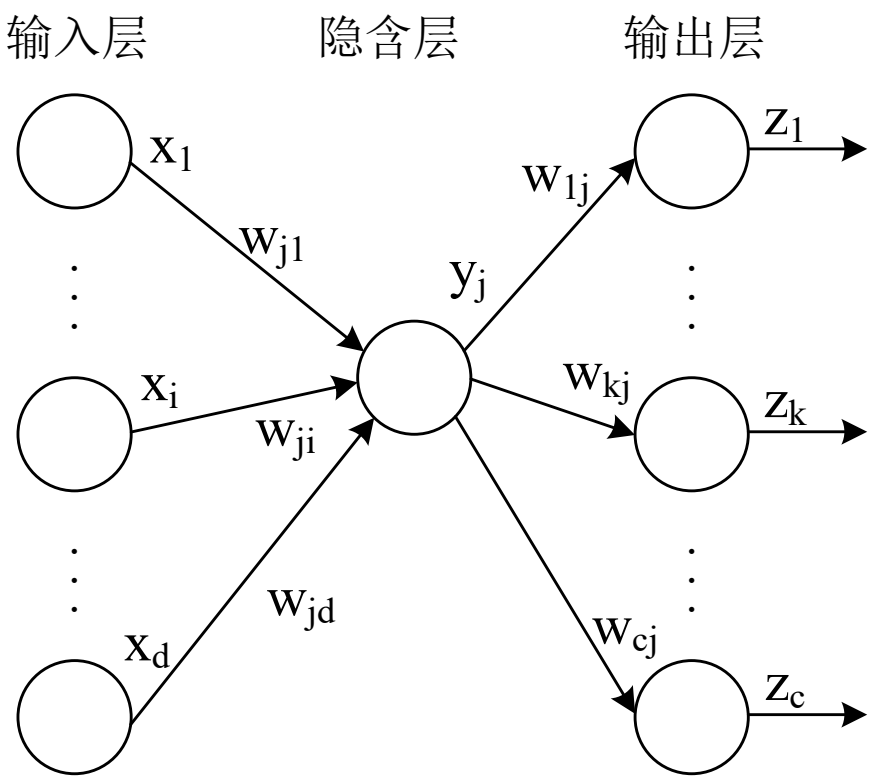


$$y = f\left(\mathbf{w}^t\mathbf{x}\right) = f\left(\sum_{i=1}^{d} w_i x_i\right),$$

f称为激活函数

$$g_k(\mathbf{x}) = f_2\left(\sum_{j=1}^{n_H} w_{kj} f_1\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$




输入层
隐含层
输出层
x_1
x_i
x_d
w_{j1}
w_{ji}
w_{jd}
y_j
w_{1j}
w_{kj}
w_{cj}
z_1
z_k
z_c

输出层

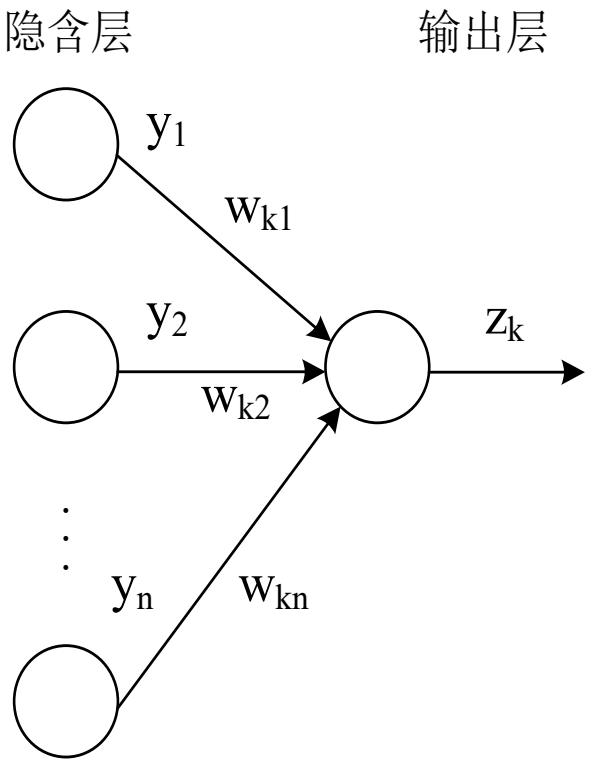
$$\frac{\partial J}{\partial w_{kj}}=\frac{\partial J}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}}=\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}}$$

$$net_k=\sum_{j=0}^{n_H}w_{kj}y_j,\qquad \frac{\partial net_k}{\partial w_{kj}}=y_j$$

$$\frac{\partial J}{\partial net_k}=\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k}=-\left(t_k-z_k\right)f'\left(net_k\right)$$

$$\frac{\partial J}{\partial w_{kj}}=-\left(t_k-z_k\right)f'\left(net_k\right)y_j=-\delta_k y_j$$

$$\delta_k=\left(t_k-z_k\right)f'\left(net_k\right)$$

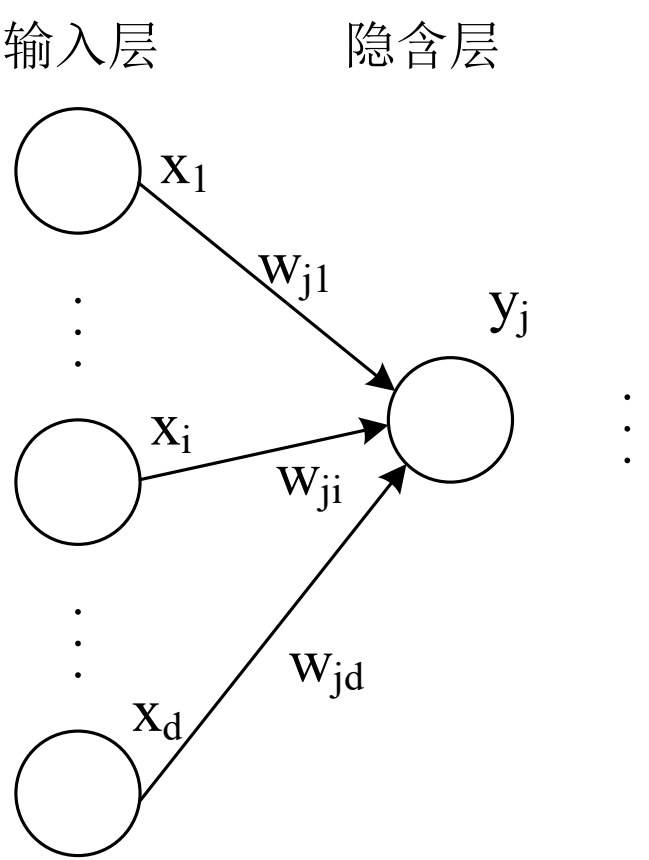


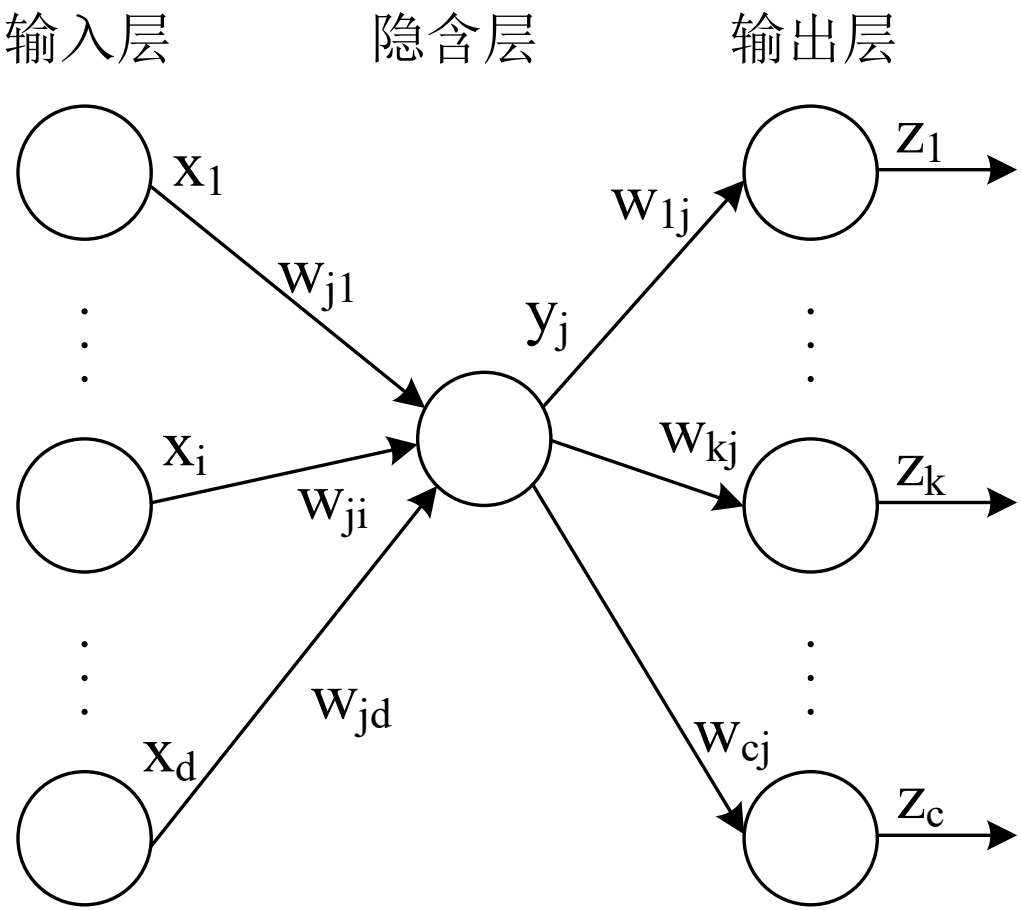
隐含层

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}$$

$$\frac{\partial y_j}{\partial net_j} = f'\left(net_j\right)$$

$$\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\left(\sum_{m=0}^{d} w_{jm}x_m\right) = x_i$$




输入层
隐含层
输出层
x1
xi
xd
wj1
wji
wjd
yj
w1j
wkj
wcj
z1
zk
zc

隐含层

$$\frac{\partial J}{\partial y_j}=\frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}\left(t_k-z_k\right)^2\right]=-\sum_{k=1}^{c}\left(t_k-z_k\right)\frac{\partial z_k}{\partial y_j}$$

$$=-\sum_{k=1}^{c}\left(t_k-z_k\right)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j}=-\sum_{k=1}^{c}\left(t_k-z_k\right)f'\left(net_k\right)w_{kj}$$

$$\frac{\partial J}{\partial w_{ji}}=-\left[\sum_{k=1}^{c}\left(t_k-z_k\right)f'\left(net_k\right)w_{kj}\right]f'\left(net_j\right)x_i$$

$$\frac{\partial J}{\partial w_{ji}}=-\delta_j x_i,\qquad \delta_j=f'\left(net_j\right)\sum_{k=1}^{c}\delta_k w_{kj}$$

迭代公式

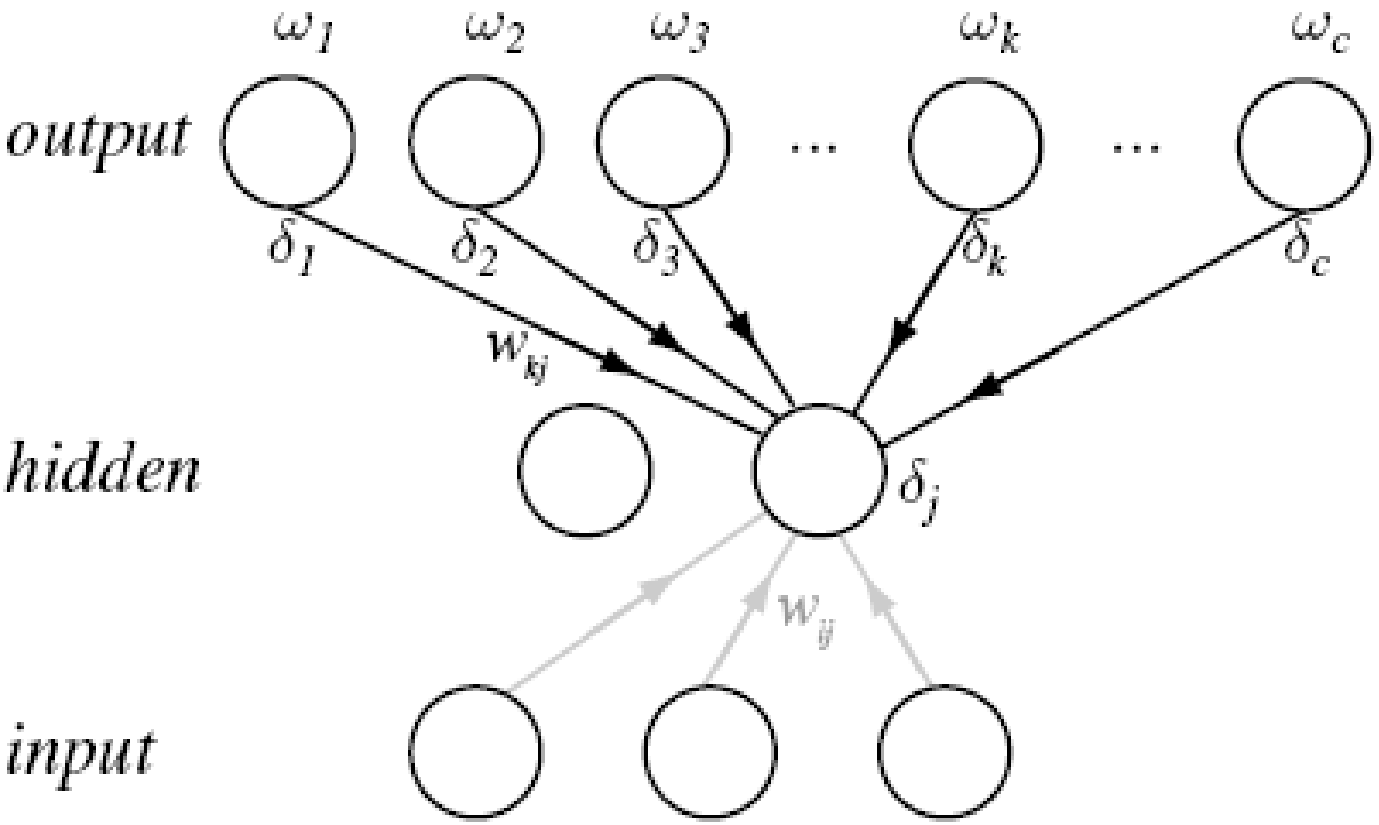
- 输出层：

$$\frac{\partial J}{\partial w_{kj}} = -\delta_k y_j, \qquad \delta_k = \left(t_k - z_k\right) f'\left(net_k\right)$$

- 隐含层：

$$\frac{\partial J}{\partial w_{ji}} = -\delta_j x_i, \qquad \delta_j = f'\left(net_j\right)\sum_{k=1}^{c}\delta_k w_{kj}$$

误差反向传播


ω_1
ω_2
ω_3
ω_k
ω_c
output
δ_1
δ_2
δ_3
δ_k
δ_c
w_{kj}
hidden
δ_j
w_{ij}
input

P2: BP算法—批量修改

1. begin initialize n，w，θ，η，r←0
2. do r←r+1
3. m←0;Δw_{ji}←0;Δw_{kj}←0
4. do m←m+1
5. x_m←select pattern
6. Δw_{kj}←$\Delta w_{kj}+\eta\delta_k y_j$ Δw_{ji}←$\Delta w_{ji}+\eta\delta_j x_i$;
7. until m = n
8. w_{ji}← $w_{ji}+\Delta w_{ji}$; w_{kj}← $w_{kj}+\Delta w_{kj}$
9. until $\|\nabla J(w)\|<\theta$
10. return w
11. end

BP算法的一些实用技术

- **激活函数的选择**：一般可以选择双曲型的Sigmoid函数，最近倾向于使用relu激活函数；
- **目标值**：期望输出一般选择**(-1, +1)**或**(0, 1)**；
- **规格化**：训练样本每个特征一般要规格化为**0均值和标准差**；
- **权值初始化**：期望每个神经元的-1<net<+1，因此权值一般初始化为 $-1/\sqrt{d} < w < 1/\sqrt{d}$ ；
- **学习率的选择**：太大容易发散，太小则收敛较慢；
- **冲量项**：有助于提高收敛速度。

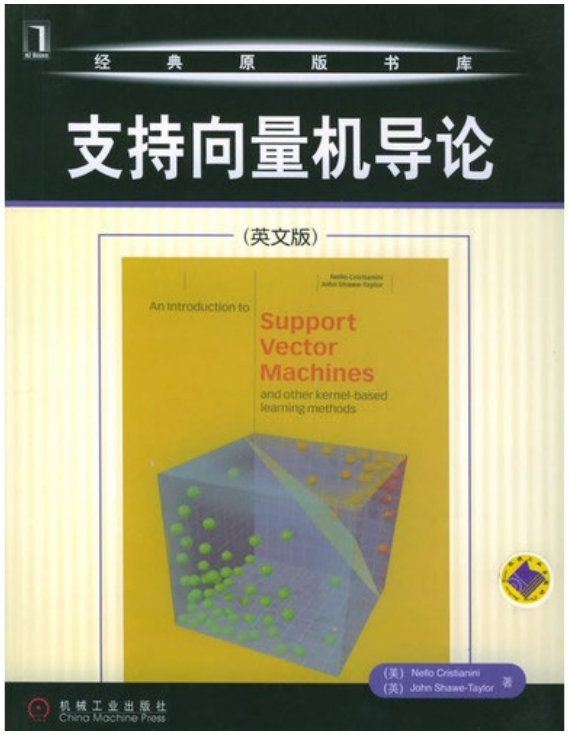
$$\mathbf{w}(m+1) = \mathbf{w}(m) + (1-\alpha)\Delta\mathbf{w}_{bp}(m) + \alpha\Delta\mathbf{w}(m-1)$$

基于核方法的非线性分类器

- **支持向量机**
- Logistic回归

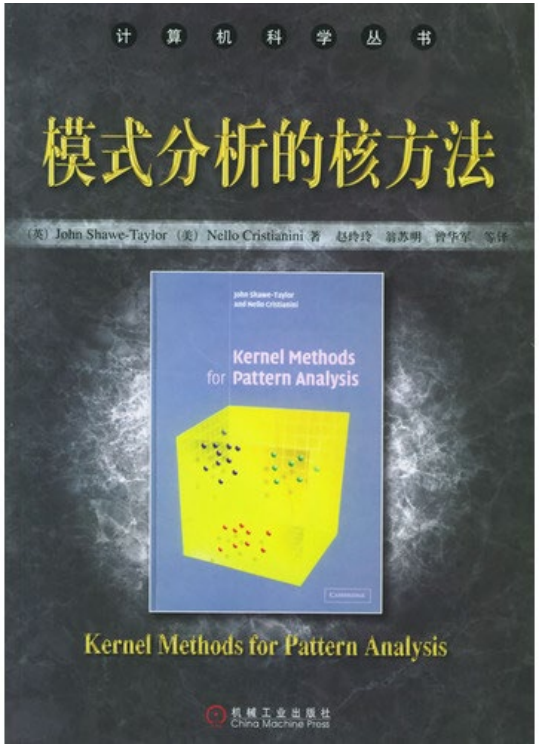
参考文献

- **支持向量机导论（英文版）**
  - 克里斯蒂亚尼尼（Cristianini, N.） 等著
  - 机械工业出版社
  - 2005-7-1



参考文献

- 模式分析的核方法（中文、英文）
  - John Shawe-Taylor, Nello Cristianini
  - 机械工业出版社
  - 2005. 1



温故而知新

- 分类器：$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x}$
- 感知器算法(单样本调整版本)

1. **begin initialize** $\mathbf{w}(0)$, k←0
2. **do** k←(k+1)mod n
3. **if** $\mathbf{x}_k$ is misclassified by w(k) **then**
   $$\mathbf{w}(k+1) \leftarrow \mathbf{w}(k) + \mathbf{x}_k$$
4. **until** all patterns properly classified
5. **return w**
6. **end**

❖ 分类器： $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$ $\quad g(\mathbf{x}) = \left(\sum_{i=1}^{n} \alpha_i \mathbf{x}_i\right)^t \mathbf{x} + w_0$

温故而知新

- 线性支持向量机 $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$
- 对偶问题

Max $\sum_{k=1}^{R}\alpha_k - \frac{1}{2}\sum_{k=1}^{R}\sum_{l=1}^{R}\alpha_k\alpha_l Q_{kl}$ 其中 $Q_{kl} = y_k y_l(\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to: $0 \le \alpha_k \le C \quad \forall k$ $\sum_{k=1}^{R}\alpha_k y_k = 0$

❖性质 $\mathbf{w} = \sum_{k=1}^{R}\alpha_k y_k \mathbf{x}_k$

$\alpha_i = 0$ 非支持向量
$\alpha_i \neq 0$ 支持向量

❖分类器: $g(\mathbf{x}) = \left(\sum_{i=1}^{n}\alpha_i^{SV}\mathbf{x}_i^{SV}\right)^t \mathbf{x} + w_0$

总结与展望

- 许多线性分类器都可以写为：

$$g(\mathbf{x}) = \left(\sum_{i=1}^{n} \alpha_i \mathbf{x}_i \right)^t \mathbf{x} + w_0$$

$$= \sum_{i=1}^{n} \alpha_i \left\langle \mathbf{x}_i, \mathbf{x} \right\rangle + w_0$$

- 内积：

$$\left\langle \mathbf{x}_i, \mathbf{x} \right\rangle = \mathbf{x}_i^t \mathbf{x}$$

- 将原始数据投影到一个高维甚至无穷维特征空间，$\mathbf{x} \to \Phi(\mathbf{x})$　　$\Phi(\mathbf{x})^t \Phi(\mathbf{y}) = K(\mathbf{x}, \mathbf{y})$

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{y}) + w_0$$

- 特征空间下的内积快速计算？

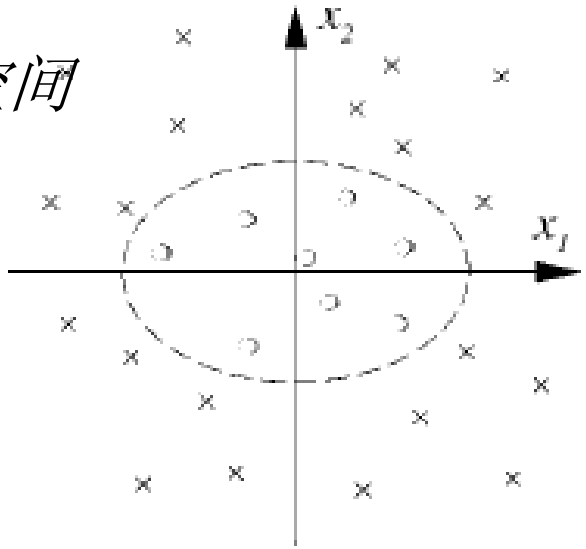
空间的非线性映射

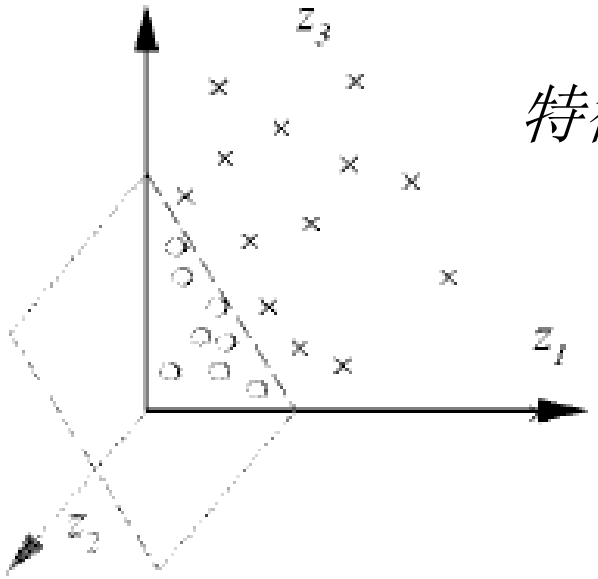
- 建立一个$\mathbf{R^2 \rightarrow R^3}$的非线性映射 $\Phi:(x_1,x_2)^t \rightarrow \left(x_1^2,\sqrt{2}x_1x_2,x_2^2\right)^t$
- 计算$\mathbf{R^3}$中2个矢量的内积：

$$\Phi(\mathbf{x})^t\,\Phi(\mathbf{y})=\left(x_1^2,\sqrt{2}x_1x_2,x_2^2\right)\left(y_1^2,\sqrt{2}y_1y_2,y_2^2\right)^t=\left(\mathbf{x}^t\mathbf{y}\right)^2$$

- 定义核函数：$K(\mathbf{x},\mathbf{y})=\left(\mathbf{x}^t\mathbf{y}\right)^2$，则：$\Phi(\mathbf{x})^t\,\Phi(\mathbf{y})=K(\mathbf{x},\mathbf{y})$

输入空间





特征空间

核函数

- 上个例子说明：特征空间中两个向量之间的内积可以通过定义输入空间中的核函数直接计算得到。

- 这就启示我们可以不必显式计算非线性映射Φ而通过**在输入空间中定义核函数K**来完成非线性映射。

- 这样做的条件是：
  1. 定义的**核函数K**能够对应于特征空间中的内积；
  2. 识别方法中不需要计算特征空间中的向量本身，而只须计算特征空间中两个向量的内积。

Hibert-Schmidt理论

- 作为核函数应满足如下条件：

  $K(\mathbf{x},\mathbf{y})$是$\mathbf{L_2}$下的对称函数，对任意 $g(\mathbf{x})\neq 0$，且

$$\int g^2(\mathbf{x})d\mathbf{x}<\infty$$

  有：

$$\iint K(\mathbf{x},\mathbf{y})g(\mathbf{x})g(\mathbf{y})d\mathbf{x}d\mathbf{y}>0$$

  成立，则$K(\mathbf{x},\mathbf{y})$可以作为核函数。

- 此条件也称为Mercer条件（准则）。

常用的核函数

- **Gaussian RBF**：
  - 无穷维

$$K(\mathbf{x},\mathbf{y})=\exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{c}\right)$$

- **Polynomial**：

$$K(\mathbf{x},\mathbf{y})=\left(\left(\mathbf{x}^t\mathbf{y}\right)+\theta\right)^d$$

- **Sigmoid**：

$$K(\mathbf{x},\mathbf{y})=\tanh\left(\alpha\left(\mathbf{x}^t\mathbf{y}\right)+\theta\right)$$

- **Inv. Multiquardric**：

$$K(\mathbf{x},\mathbf{y})=\frac{1}{\sqrt{\|\mathbf{x}-\mathbf{y}\|^2+c^2}}$$

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

核方法

- 线性支持向量机 → 非线性支持向量机
- 线性Logisitc 回归 → 非线性Logisitc 回归
- 感知器 → 非线性感知器
- ……

核函数应用于线性分类器（非线性SVM）

- **SVM的求解，最后归结为如下目标函数的优化：**

$$L(\boldsymbol{\alpha})=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j z_i z_j \mathbf{y}_i^t\mathbf{y}_j \qquad \alpha_i\geq 0,\quad i=1,\cdots,n$$

- **可以引入非线性映射Φ，则目标函数变为：**

$$L(\boldsymbol{\alpha})=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\left(\alpha_i\alpha_j z_i z_j \Phi^t(\mathbf{y}_i)\Phi(\mathbf{y}_j)\right)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\left(\alpha_i\alpha_j z_i z_j K(\mathbf{y}_i,\mathbf{y}_j)\right)$$

$$\alpha_i\geq 0,\quad i=1,\cdots,n$$

- **而权向量为：**

$$\mathbf{w}=\sum_{i=1}^{n}z_i\alpha_i\Phi(\mathbf{y}_i)$$

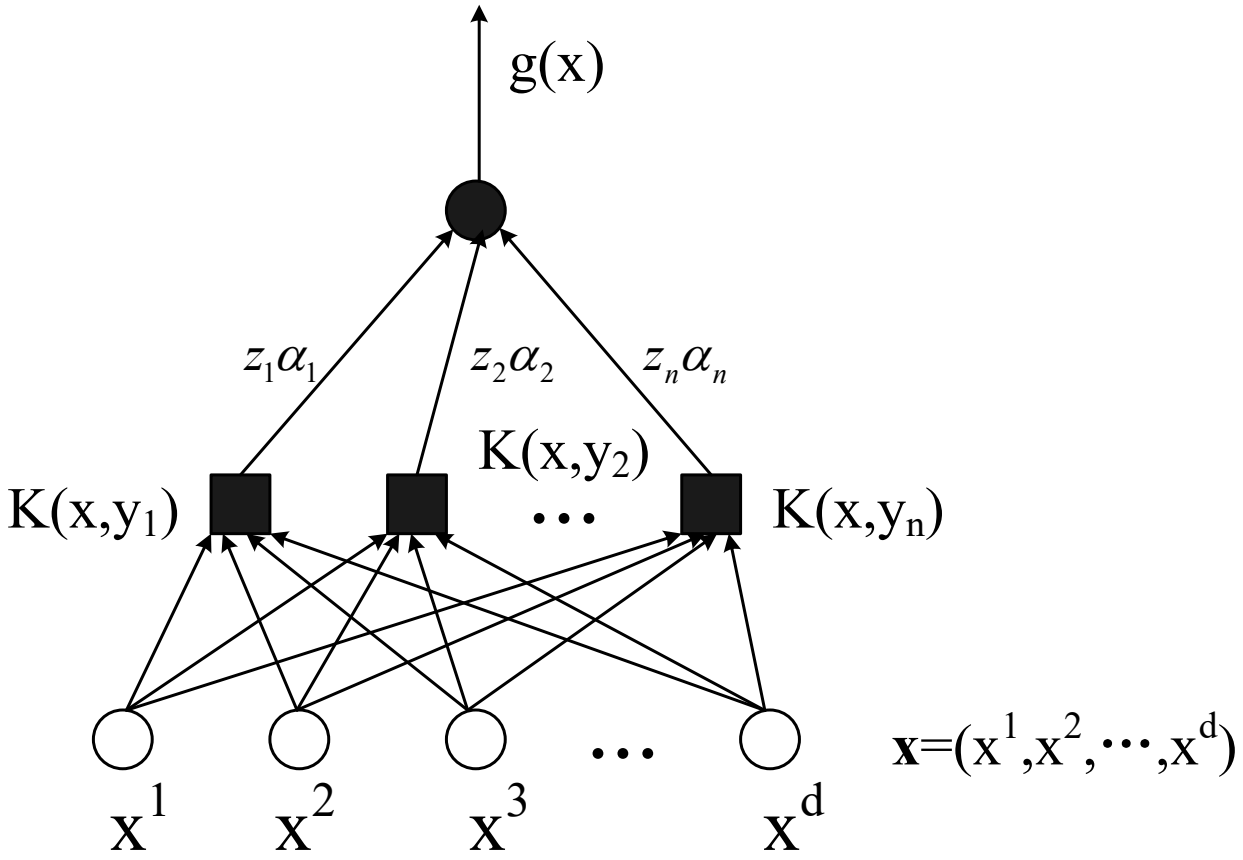
- **判别函数：**

$$g(\mathbf{x})=\mathbf{w}^t\Phi(\mathbf{x})+w_0=\sum_{i=1}^{n}\left(z_i\alpha_i\Phi(\mathbf{y}_i)\right)^t\Phi(\mathbf{x})+w_0=\sum_{i=1}^{n}z_i\alpha_i K(\mathbf{y}_i,\mathbf{x})+w_0$$

解释

- 支持向量机中采用对偶形式的原因
  - 更容易求解
  - 原始数据空间和核特征空间**问题表示的一致性**
- **线性方法 -> 核函数 -> 非线性方法**
  - 分类器的变化
  - 学习和求解过程的变化

支持向量机的神经网络表示



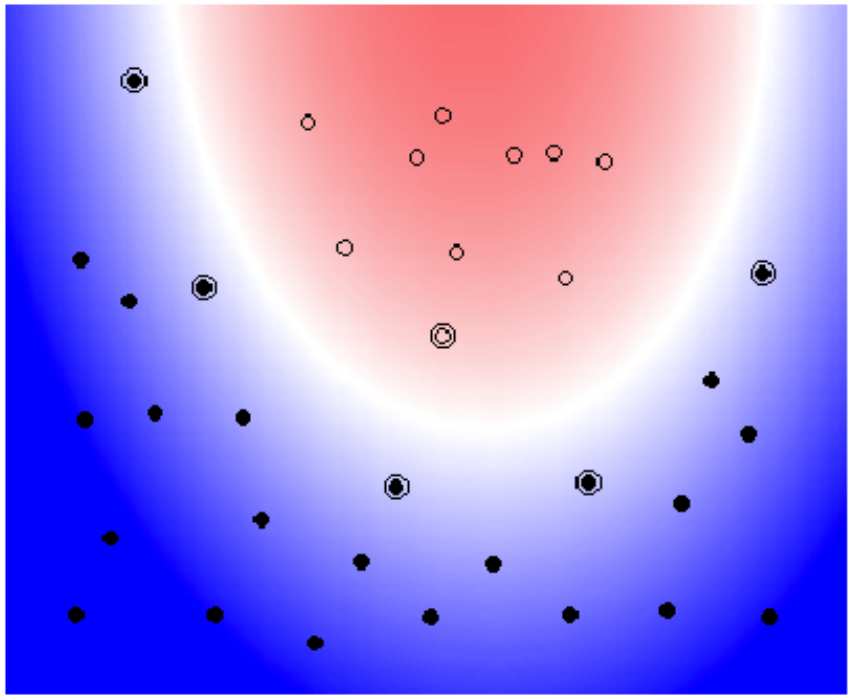
➡ **径向基函数（RBF）神经网络**

分类结果显示

Example: SVM with Polynomial of Degree 2

Kernel: $K(\vec{x}_i,\vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^2$

plot by Bell SVM applet

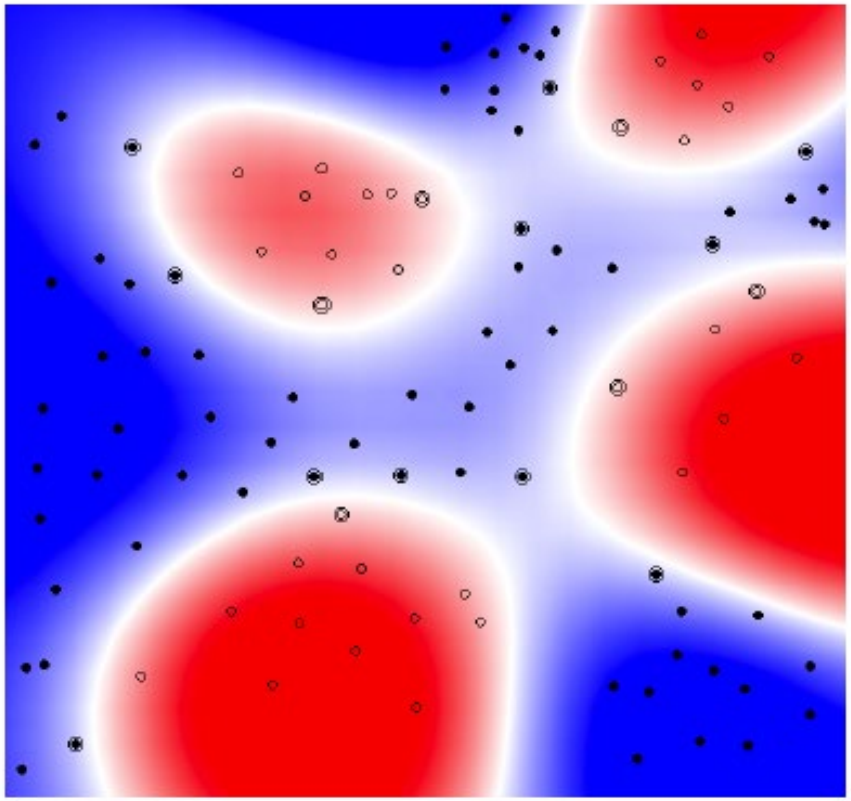


分类结果显示

Example: SVM with RBF-Kernel

Kernel: $K(\vec{x}_i,\vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2/\sigma^2)$

plot by Bell SVM applet



例

- 使用SVM设计手写数字分类器，每个类别100个降维后的训练样本，采用一对一投票的多类别解决方案。
- 核函数可以选择：线性核，多项式核以及径向基函数核，实验不同的参数设置。
- 可以考虑直接利用Kernel Method Toolbox中的SVM分类器，也可以利用matlab中的quadprog函数训练SVM。

基于核方法的 Logistic 回归

$$\begin{aligned}\Pr(G=k|X=x) &= \frac{\exp(\beta_{k0}+\beta_k^T x)}{1+\sum_{\ell=1}^{K-1}\exp(\beta_{\ell 0}+\beta_\ell^T x)}, \quad k=1,\ldots,K-1,\\ \Pr(G=K|X=x) &= \frac{1}{1+\sum_{\ell=1}^{K-1}\exp(\beta_{\ell 0}+\beta_\ell^T x)}, \end{aligned} \tag{4.18}$$

$$\log\frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)} = \beta_{10}+\beta_1^T x$$

如何转换为基于核方法的Logistic 回归?

基于核方法的 Logistic 回归

$$x \to \phi(x),\ \beta_l = \sum_{i=1}^{N} \alpha_i \phi(x_i)$$

$$K(\beta_l, x) = \sum_{i=1}^{N} \alpha_{li} K(x_i, x)$$

$$p(G = K \mid x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp\left\{\beta_{l0} + \sum_{i=1}^{N} \alpha_{li} K(x_i, x))\right\}}$$

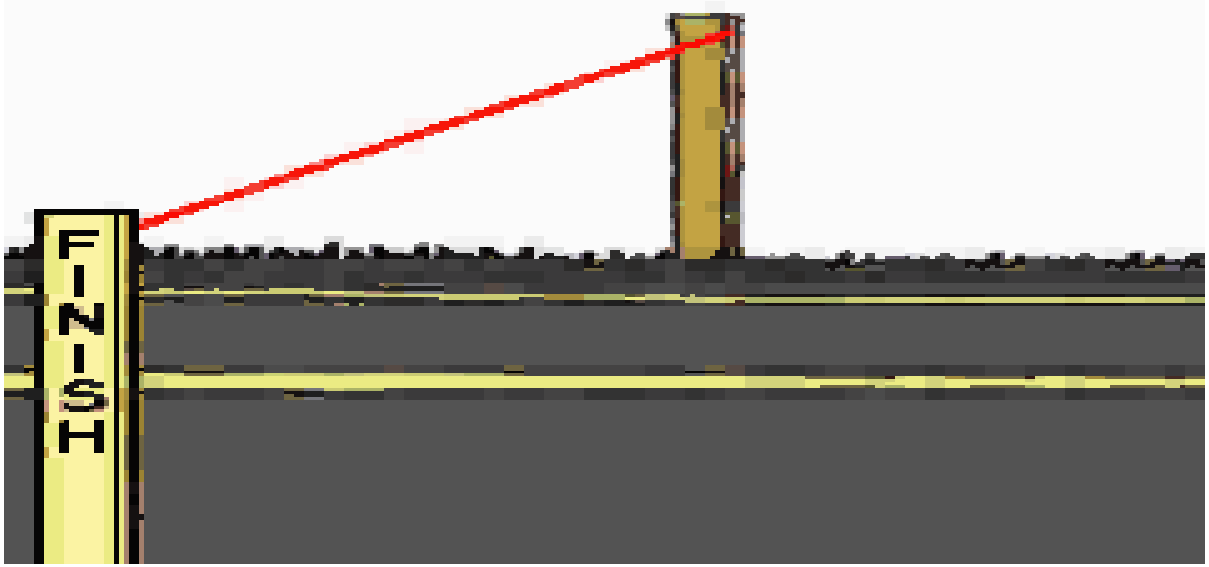
$$p(G = k \mid x) = \frac{\exp\left\{\beta_{k0} + \sum_{i=1}^{N} \alpha_{ki} K(x_i, x))\right\}}{1 + \sum_{l=1}^{K-1} \exp\left\{\beta_{l0} + \sum_{i=1}^{N} \alpha_{li} K(x_i, x))\right\}}$$

学习和训练方法

- 比较复杂，
  - 原始问题
  - 对偶问题
- 感兴趣的学生可阅读以下文献：
  - S. S. Keerthi, K. B. Duan, S. K. Shevade and A. N. Poo; 2005. A fast dual algorithm for kernel logistic regression, Machine Learning, Volume 61, Numbers 1-3, 151-165.

思考：基于核方法的**感知器**学习算法

- 分类器表示
- 学习算法
- **支持向量机导论（英文版）**
  - 克里斯蒂亚尼尼（Cristianini, N.） 等著
  - 机械工业出版社
  - 2005-7-1



END