Machine Learning

第八章 基于实例的学习

8.1 简介

- 基于实例学习特点
  - 训练 : 只存储样本
  - 分类：检索训练集中相似的样本

- 与其它算法的关键差异：
  - 为每个分类实例建立目标函数
  - 优点
    - 用多个简单局部函数逼近复杂的目标函数

简介(2)

- IBL缺点：
  - 分类代价高
    - 高效索引训练样本
  - 相似实例可能会距离远
    - 有时目标概念仅依赖于几个属性

- 本章算法
  - k-近邻（k-Nearest neighbor learning）及变种
  - 局部加权回归（Locally weighted regression）
  - 径向基函数（Radial basis function network）
  - 消极学习（Lazy learning） 积极学习（eager learning）

8.2 k-近邻算法

- 适用问题
  - 实例表示为R^n空间中的点， 特征向量 $\langle a_1(x), ..., a_n(x)\rangle$
  - 任意两个实例之间的距离定义：

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2}$$

  - 学习目标函数f: $R^n \rightarrow V$
    - V： 离散值或实数值

k-近邻算法（2）

- 学习离散值目标函数f: $R^n \rightarrow V$，$V=\{v_1,\ldots,v_s\}$
  - 训练算法

    For each training sample <x, f(x)>

    { add it to the list *training_examples* }

  - 分类算法
    - 待分类（查询）实例 x_q
      - 在training_examples中， 选择K 个 x_q的最近邻x1...xk
      - 返回
      - $$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

$$\delta(a,b)=\begin{cases}1 & a=b \\ 0 & a \neq b\end{cases}$$

k-近邻算法（3）

- 解释：
  - $\hat{f}(x_q)$：k 近邻中最普遍的类别
  - $\hat{f}(x_q)$：结果依赖于近邻数目, 见图.8-1 （下个PPT）
- k-近邻算法学习到的一般函数（隐含）
  - Fig. 8-1，1-近邻决策面.
  - Voronoi 图

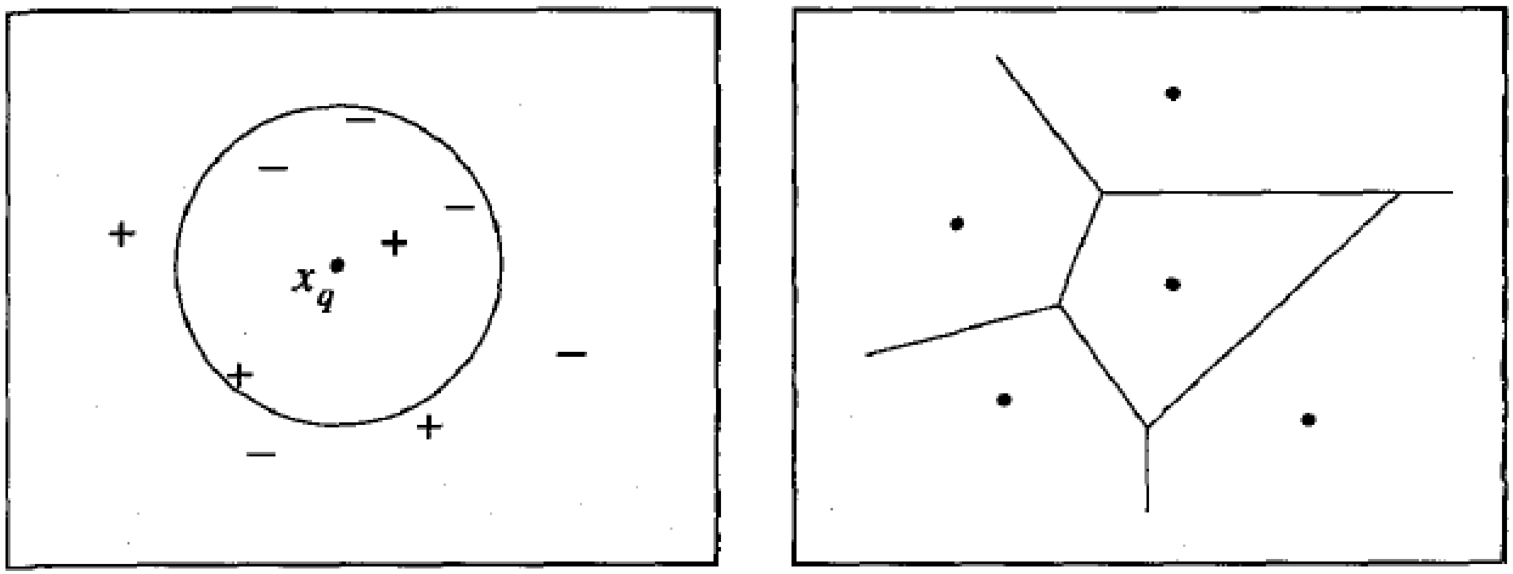


FIGURE 8.1

k-近邻算法（3）

- 学习实值目标函数
  - 用k个最近邻训练样本的均值表示
  - 实函数 f: $R^n \rightarrow R$ 的逼近

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

8.2.1 距离加权最近邻算法

- k-近邻改进
  - 根据距离对k-近邻样本进行加权
- 离散值目标函数：

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i)) \qquad w_i = \frac{1}{d(x_q, x_i)^2}$$

  - 如果 $d(x_q, x_i)^2 = 0$，$\hat{f}(x_q)$ $=f(x_i)$.
  - 如果存在多个零距离样本, $\hat{f}(x_q)$ 为样本数最多的类别
- 实值目标函数：

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i} \qquad (8.4)$$

距离加权最近邻算法（2）

- 改进：用全部训练样本分类查询点
  - 计算全部样本的距离权重
    - 远距离样本对 $\hat{f}(x_q)$ 影响极小
  - 缺点:
    - 分类时间长
  - 全局方法与局部方法

- Shepard 方法
  - 用全局方法逼近实值目标函数（见8.4）

8.2.2 k-近邻算法的说明

- 距离加权k-NN 对噪声有鲁棒性
  - 可以平滑掉孤立的噪声训练样本的影响.
- k-近邻的归纳偏置:
  - x_q 的类别与它附近的实例的类别相似.
- k-近邻实践中的问题: 维度灾难（curse of dimensionality）
  - 基于全部属性计算距离
  - 实际中类别相关属性只占少数.
    - 误导k-近邻分类器

k-近邻算法的说明（2）

- 克服维度灾难的方法:
  - 对属性加权，相当于按比例缩放坐标轴.
    - 伸展第j个坐标轴 z_j倍
- 其它方法
  - 清除最不相关属性（ $Z_j=0$）
    - 用cross-validation 方法 (leave-one-out) 选择属性
  - 用变化的值伸展属性坐标轴

Remarks on K-NN algorithm （3）

- 实践问题: 如何建立搜索近邻样本的高效索引
  - 方法举例
    - KD-tree： 实例存储在叶子节点，其近邻存储在相同或附近的节点内.
    - 通过测试 x_q的选定属性，KD tree 把查询点 x_q 排序到相关叶子节点

术语解释

- 统计模式识别术语
  - 回归（Regression）：逼近实值目标函数
  - 残差(Residual)：逼近目标函数的误差 $\hat{f}(x)-f(x)$
  - 核函数(Kernel function)：
    - 决定训练样例权值的距离函数.
    - K: $w_i=K(d(x_i, x_q))$

8.3 局部加权回归

- 最近邻方法是在单一查询点 x_q逼近目标函数
- 局部加权回归是对k-近邻的推广
  - 可采用线性函数, 二次函数, 神经网络逼近 f
  - 局部：在x_q 的局部区域逼近目标函数*f*
  - 加权：用加权训练样本构造目标函数 *f* 的局部近似
  - 回归

局部加权回归（2）

- 局部加权回归的一般方法：
  - 构造 $\hat{f}$ 拟合 x_q邻域内的训练样本
  - 用 $\hat{f}$ 逼近 $\hat{f}(x_q)$
  - 删除 $\hat{f}$

8.3.1 局部加权线性回归

- 用线性函数逼近 x_q邻域的目标函数 f
  - $a_i(x)$: 实例x的第i个属性

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \ldots + w_n a_n(x)$$

- 定义误差函数

$$E = \frac{1}{2}\sum_{x \in D}(f(x) - \hat{f}(x))^2$$

- 训练规则

$$\Delta w_j = \eta \sum_{x \in D}(f(x) - \hat{f}(x)) a_j(x)$$

局部加权线性回归（2）

- 三种误差函数E(x_q)
  - k-近邻上的误差平方和最小

$$E_1(x_q) = \frac{1}{2} \sum_{x \in \text{k } nearest \text{ neighbors of } x_q} (f(x) - \hat{f}(x))^2$$

  - D上误差平方和最小, 样例的误差加权（距离的递减函数）

$$E_2(x_q) = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

  - 两种结合

$$E_2(x_q) = \frac{1}{2} \sum_{x \in k \text{ nearest neighbors of } x_q} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

局部加权线性回归（3）

- Criterion 2： 也许最好, 计算量大
- Criterion 3： 准则2的近似，计算代价低
- Criterion 3的梯度公式：

$$\Delta w_i = \eta \sum_{x \in k \text{ nearest neighbors of } x_q} K(d(x_q, x))(f(x) - \hat{f}(x))a_j(x)$$

8.3.2 局部加权回归的说明

- 可用常量、线性函数、二次函数局部近似目标函数，更复杂的函数形式不常见，因为：
  - 代价高
  - 简单函数的近似效果已相当好

8.4 径向基函数（Radial Basis Functions）

- 径向基函数
  - 用于函数逼近
  - 类似于距离加权回归、ANN
- 假设表示：

$$\hat{f}(x)=w_0+\sum_{u=1}^{k}w_u K_u(d(x_u,x))$$

- $\hat{f}(x)$ 是f(x)的全局近似函数
  - $K_u(d(x_u,x))$ 是高斯函数

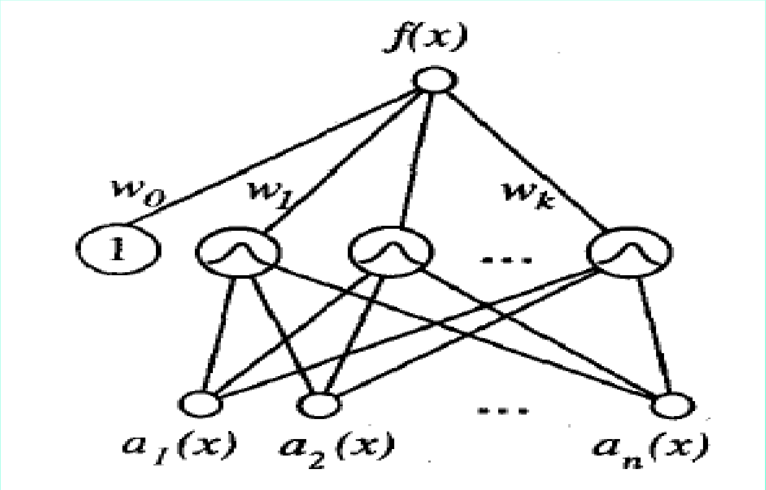
$$K_u(d(x_u,x))=e^{-\frac{1}{2\sigma_u^2}d^2(x_u,x)}$$

  - $K_u(d(x_u,x))$ 有局部化特性

径向基函数（2）

- $\hat{f}(x)$ (Gaussian Kernel) 以任意小误差逼近任意函数的条件
  - k足够大
  - 可分别指定每个核的宽度
- $\hat{f}(x)$ 可表示为两层神经网络
  - 第一层计算$K_u(d(x_u,x))$，
  - 第二层是第一层单元的线性组合
- 两阶段训练RBF
  - 确定k，选择x_u 及σ_u^2
  - 定义全局误差, 学习 权 w_u

$$E=\frac{1}{2}\sum_{x\in D}(f(x)-\hat{f}(x))^2$$



径向基函数（3）

- 如何选择核函数数量
  - 每一训练样本<x_i, f(x_i)>分配一个
    - x_i 为中心，σ_u^2 相同
    - <x_i,f(x_i)> 只对f(x_i)产生局部影响

    - 优点 : RBF 可精确拟合训练数据
      - 对每一样本可学习到权w_i… w_m使得$\hat{f}(x_i) = f(x_i)$

Radial Basis Functions （4）

– 核数目小于训练样本数
  - 提高计算效率
  - 核函数的中心在实例空间均匀分布
  - 或者, 非均匀分布（特别当实例非均匀分布时）
    – 随机选取训练样本作为和函数的中心
    – 聚类，聚类中心作为核函数的中心

径向基函数小结（5）

- 对目标函数的全局逼近
- 输入样本的局部有效性
- 对目标函数进行局部逼近的多个函数的平滑线性组合
- 优点: 训练高效（由于输入层、输出层单独训练）

8.6 消极学习和积极学习评论

- 消极学习：k-NN, locally weighted regression
- 积极学习：RBF
- 两者区别：
  - 计算时间: 训练时间，分类时间
  - 归纳偏置
    - 泛化时是否考虑查询点xq 的信息
- 泛化精度
  - 核心观点：
    - 消极学习器用（查询点附近）多个局部逼近的组合表示目标函数
    - 积极学习器需要在训练时间完成全局逼近函数的学习

消极学习和积极学习评论（2）

- 积极方法能否使用多个局部逼近?
  - 径向基函数网络
- RBF 提供全局逼近
  - 由多个局部核函数的线性组合实现
  - 但是，并非针对查询点的局部逼近

Summary

- IBL：
  - 属消极学习方法
  - 无需构造目标函数的显式假设
  - 为每个查询点构造目标函数的不同逼近
- IBL优点：
  - 用不太复杂的局部逼近函数集构造复杂目标函数
  - 不会损失训练样例的任何信息
- IBL问题：
  - 分类效率低
  - 合适的实例距离度量函数难构造
  - 无关特征对距离的副作用

Summary（2）

- k-Nearest Neighbor

- Locally weighted regression methods
  - k-NN的推广

- Radial basis function networks
  - 局部核函数构成的神经网络
  - 可看作IBL 及ANN的混合