机器学习

基本术语

data set：数据集
sample/instance：样本/示例
attribute/feature：对象某个属性
attribute value：属性值
attribute space/sample space：样本空间/输入空间，属性张成的空间
feature vector：空间中示例对应的特征向量
m 个示例的数据集 $D=\{x_1,x_2,\ldots,x_m\}$, 每个示例有 d 个属性 $x_i=(x_{i1};x_{i2};\ldots;x_{id})$

label：标记，关于示例结果的信息
label space：标记空间/输出空间，所有标记的集合
example：样例，(x_i,y_i) 表示第 i 个样例，y_i 是示例 x_i 的标记

classification：分类，预测离散值
regression：回归，预测连续值
clustering：聚类，将训练集分成若干组，每组称为一个簇(cluster)

supervised learning：监督学习
unsupervised learning：无监督学习（根据是否有label区分）
generalization：泛化，模型适用于新样本的能力，机器学习的目标

模型评估与选择

误差(error)

训练集上的误差：训练误差
新样本上的误差：泛化误差
测试集上的误差：测试误差，作为泛化误差的近似

过拟合（overfitting）
欠拟合（underfitting）

评估方法

- 留出法(hold-out)
  将数据集D划分为训练集S和测试集T，在S上训练模型过后在T上评估测试误差。
- 交叉验证法(cross validation)
  先将D划分成k个大小相似的互斥子集，每次用k-1个训练，用1个测试，重复k次取均值，称为k折交叉验证(k-fold cross validation)
  - 留一法(LOO)：特例，D中有m个样本，令k=m，计算开销更大但更准确

性能度量

- 均方误差(MSE)
  $E(f;D)=1/m\cdot\Sigma_{i=1}^{m}(f(x_i)-y_i)^2$

  更一般的，对于数据分布 $\mathcal{D}$ 和概率密度函数 $p(\cdot)$，均方误差可描述为

  $$E(f;\mathcal{D})=\int_{\boldsymbol{x}\sim\mathcal{D}}(f(\boldsymbol{x})-y)^2p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\ . \tag{2.3}$$

- 错误率

  $$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(f(\boldsymbol{x}_i)\neq y_i)\ . \tag{2.4}$$

  $$E(f;\mathcal{D})=\int_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{I}(f(\boldsymbol{x})\neq y)p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\ , \tag{2.6}$$

- 精度
  $acc(f;D)=1-E(f;D)$
- 混淆矩阵

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP（真正例）	FN（假反例）
反例	FP（假正例）	TN（真反例）

查准率 $P=\frac{TP}{TP+FP}$

查全率 $R=\frac{TP}{TP+FN}$

- P-R曲线：查准率-查全率曲线
  平衡点是 $P=R$ 时的取值
  - F1 score：$F1=2\cdot P\cdot R/(P+R)=2\cdot TP/(\text{样例总数}+TP-TN)$

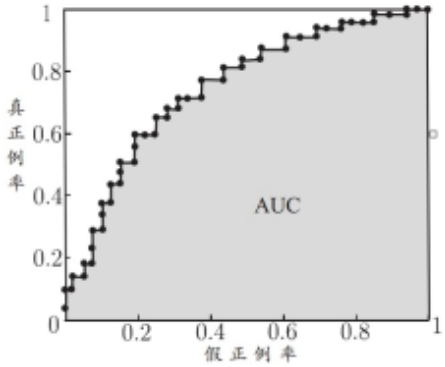
    比F1更一般的形式 F_β ，

    $$F_\beta=\frac{(1+\beta^2)\times P\times R}{(\beta^2\times P)+R}$$

    $\beta=1$：标准F1
    $\beta>1$：偏重查全率（逃犯信息检索）
    $\beta<1$：偏重查准率（商品推荐系统）

- ROC图和AUC值

ROC图的绘制：给定 m^+ 个正例和 m^- 个负例，根据学习器预测结果对样例进行排序，将分类阈值设为每个样例的预测值，当前标记点坐标为 (x,y)，当前若为真正例，则对应标记点的坐标为 $(x,y+\frac{1}{m^+})$；当前若为假正例，则对应标记点的坐标为 $(x+\frac{1}{m^-},y)$，然后用线段连接相邻点.



基于有限样例绘制的 ROC 曲线与 AUC

假设ROC曲线由 $\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$ 的点按序连接而形成，则：AUC可估算为：

$$\mathrm{AUC}=\frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1}-x_i)\cdot(y_i+y_{i+1})$$

AUC衡量了样本预测的排序质量。

$$\mathrm{AUC}=1-\ell_{rank}.$$

线性模型

给出示例 $x=(x_1;x_2;\ldots;x_d)$，线性模型试图通过学习得到一个函数
$f(x)=w_1x_1+w_2x_2+\ldots+w_dx_d+b$

向量形式：$f(x)=w^Tx+b$

线性回归(linear regression)

给定数据集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$，其中 $\boldsymbol{x}_i=(x_{i1};x_{i2};\ldots;x_{id})$，$y_i\in\mathbb{R}$. “线性回归”(linear regression)试图学得一个线性模型以尽可能准确地预测实值输出标记.

目的：确定w和b
方式：最小化均方误差

$$\begin{aligned}(w^*,b^*)&=\arg\min_{(w,b)}\sum_{i=1}^{m}(f(x_i)-y_i)^2\\&=\arg\min_{(w,b)}\sum_{i=1}^{m}(y_i-wx_i-b)^2.\end{aligned}$$

求解 w 和 b 使 $E_{(w,b)} = \sum_{i=1}^{m}(y_i - wx_i - b)^2$ 最小化的过程, 称为线性回归模型的最小二乘“参数估计”(parameter estimation). 我们可将 $E_{(w,b)}$ 分别对 w 和 b 求导, 得到

$$\frac{\partial E_{(w,b)}}{\partial w} = 2\left(w\sum_{i=1}^{m}x_i^2 - \sum_{i=1}^{m}(y_i - b)\,x_i\right) , \tag{3.5}$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2\left(mb - \sum_{i=1}^{m}(y_i - wx_i)\right) , \tag{3.6}$$

然后令式(3.5)和(3.6)为零可得到 w 和 b 最优解的闭式(closed-form)解

$$w = \frac{\sum_{i=1}^{m} y_i(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2} , \tag{3.7}$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i) , \tag{3.8}$$

其中 $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$ 为 x 的均值.

多元线性回归:

类似的, 可利用最小二乘法来对 $\boldsymbol{w}$ 和 b 进行估计. 为便于讨论, 我们把 $\boldsymbol{w}$ 和 b 吸收入向量形式 $\hat{\boldsymbol{w}} = (\boldsymbol{w}; b)$, 相应的, 把数据集 D 表示为一个 $m \times (d+1)$ 大小的矩阵 $\mathbf{X}$, 其中每行对应于一个示例, 该行前 d 个元素对应于示例的 d 个属性值, 最后一个元素恒置为 1, 即

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} & 1 \\ \boldsymbol{x}_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^{\mathrm{T}} & 1 \end{pmatrix} ,$$

再把标记也写成向量形式 $\boldsymbol{y} = (y_1; y_2; \ldots; y_m)$, 则类似于式(3.4), 有

$$\hat{\boldsymbol{w}}^* = \arg\min_{\hat{\boldsymbol{w}}} (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})^{\mathrm{T}} (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}) . \tag{3.9}$$

令 $E_{\hat{\boldsymbol{w}}} = (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})^{\mathrm{T}} (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})$, 对 $\hat{\boldsymbol{w}}$ 求导得到

$$\frac{\partial E_{\hat{\boldsymbol{w}}}}{\partial \hat{\boldsymbol{w}}} = 2\,\mathbf{X}^{\mathrm{T}} (\mathbf{X}\hat{\boldsymbol{w}} - \boldsymbol{y}) . \tag{3.10}$$

令上式为零可得 $\hat{\boldsymbol{w}}$ 最优解的闭式解, 但由于涉及矩阵逆的计算, 比单变量情形要复杂一些. 下面我们做一个简单的讨论.

对数几率回归

对于二分类问题 $y = \{0, 1\}$, 线性回归模型产生的预测值 $w^T x + b$ 为实值，需要将其转化为0/1值

- unit-step function：单位阶跃函数，大于零判为正例，小于零反例；缺点：不连续

- logistic functon：逻辑斯蒂函数，“Sigmoid函数”之一，将 w^Tx+b 转化为接近0或1的 y 值：

$$y=\frac{1}{1+e^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}}.$$

$$\ln\frac{y}{1-y}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b.$$

$$p(y=1\mid\boldsymbol{x})=\frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}},$$

$$p(y=0\mid\boldsymbol{x})=\frac{1}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}.$$

  - 极大似然法（maximum likelihood）
    给定数据集 $\{(x_i,y_i)\}_{i=1}^m$
    最大化对数似然函数

$$\ell(\boldsymbol{w},b)=\sum_{i=1}^{m}\ln p(y_i\mid\boldsymbol{x}_i;\boldsymbol{w}_i,b)$$

    转化为最小化负对数似然函数求解

    - 令 $\boldsymbol{\beta}=(\boldsymbol{w};b)$，$\hat{\boldsymbol{x}}=(\boldsymbol{x};1)$，则 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ 可简写为 $\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}$

    - 再令 $p_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})=p(y=1\mid\hat{\boldsymbol{x}};\boldsymbol{\beta})$

$$p_0(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})=p(y=0\mid\hat{\boldsymbol{x}};\boldsymbol{\beta})=1-p_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})$$

$$p(y_i\mid\boldsymbol{x}_i;\boldsymbol{w}_i,b)=y_ip_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})+(1-y_i)p_0(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})$$

    则似然项可重写为

$$\ell(\boldsymbol{\beta})=\sum_{i=1}^{m}\left(-y_i\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}_i+\ln\left(1+e^{\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}_i}\right)\right)$$

    - 求解得

$$\boldsymbol{\beta}^*=\arg\min_{\boldsymbol{\beta}}\ell(\boldsymbol{\beta})$$

    - 牛顿法第 t+1轮迭代解的更新公式

$$\boldsymbol{\beta}^{t+1}=\boldsymbol{\beta}^t-\left(\frac{\partial^2\ell(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^{\mathrm{T}}}\right)^{-1}\frac{\partial\ell(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}}$$

    其中关于 β的一阶、二阶导数分别为

$$\frac{\partial\ell(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}}=-\sum_{i=1}^{m}\hat{\boldsymbol{x}}_i(y_i-p_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta}))$$

$$\frac{\partial^2\ell(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^{\mathrm{T}}}=\sum_{i=1}^{m}\hat{\boldsymbol{x}}_i\hat{\boldsymbol{x}}_i^{\mathrm{T}}p_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})(1-p_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta}))$$

线性判别分析(LDA)

给定训练样例集，设法将样例投影到一条直线上，使得同类样例的投影点尽可能接近、异类尽可能远离，对新样本分类时，将其投影到同样的这条直线上，根据投影点位置确定新样本类别。

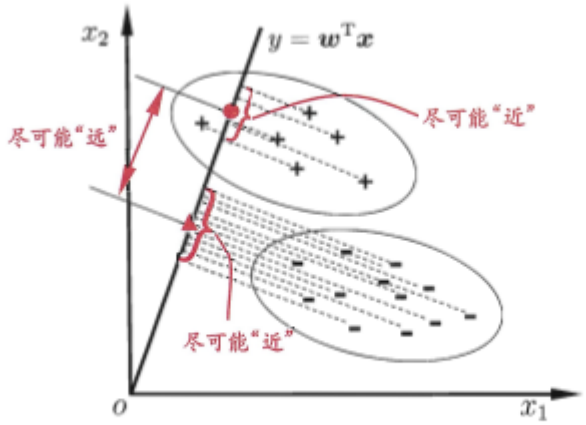


图 3.3 LDA 的二维示意图. "+"、"−" 分别代表正例和反例, 椭圆表示数据簇的外轮廓, 虚线表示投影, 红色实心圆和实心三角形分别表示两类样本投影后的中心点.

欲使同类样例的投影点尽可能接近, 可以让同类样例投影点的协方差尽可能小, 即 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}$ 尽可能小; 而欲使异类样例的投影点尽可能远离, 可以让类中心之间的距离尽可能大, 即 $\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\|_2^2$ 尽可能大. 同时考虑二者, 则可得到欲最大化的目标

$$\begin{aligned} J &= \frac{\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\|_2^2}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}} \\ &= \frac{\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)\boldsymbol{w}} . \end{aligned} \tag{3.32}$$

定义"类内散度矩阵" (within-class scatter matrix)

$$\begin{aligned} \mathbf{S}_w &= \boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1 \\ &= \sum_{\boldsymbol{x}\in X_0} (\boldsymbol{x} - \boldsymbol{\mu}_0)(\boldsymbol{x} - \boldsymbol{\mu}_0)^{\mathrm{T}} + \sum_{\boldsymbol{x}\in X_1} (\boldsymbol{x} - \boldsymbol{\mu}_1)(\boldsymbol{x} - \boldsymbol{\mu}_1)^{\mathrm{T}} \end{aligned} \tag{3.33}$$

以及"类间散度矩阵" (between-class scatter matrix)

$$\mathbf{S}_b = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\mathrm{T}} , \tag{3.34}$$

则式(3.32)可重写为

$$J = \frac{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_b\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w\boldsymbol{w}} . \tag{3.35}$$

这就是 LDA 欲最大化的目标, 即 $\mathbf{S}_b$ 与 $\mathbf{S}_w$ 的"广义瑞利商" (generalized Rayleigh quotient).

多分类学习

拆成若干个二分类问题

给定数据集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$, $y_i\in\{C_1,C_2,\ldots,C_N\}$. OvO 将这 N 个类别两两配对, 从而产生 $N(N-1)/2$ 个二分类任务, 例如 OvO 将为区分类别 C_i 和 C_j 训练一个分类器, 该分类器把 D 中的 C_i 类样例作为正例, C_j 类样例作为反例. 在测试阶段, 新样本将同时提交给所有分类器, 于是我们将得到 $N(N-1)/2$ 个分类结果, 最终结果可通过投票产生: 即把被预测得最多的类别作为最终分类结果. 图 3.4 给出了一个示意图.

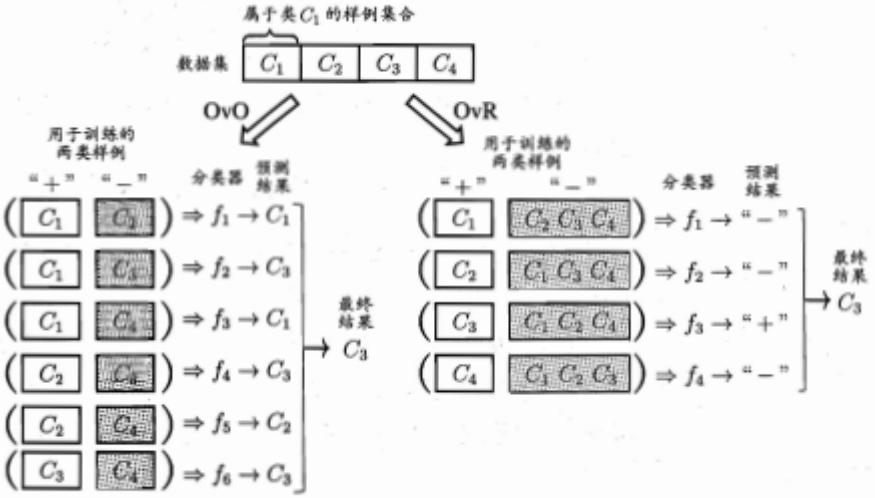
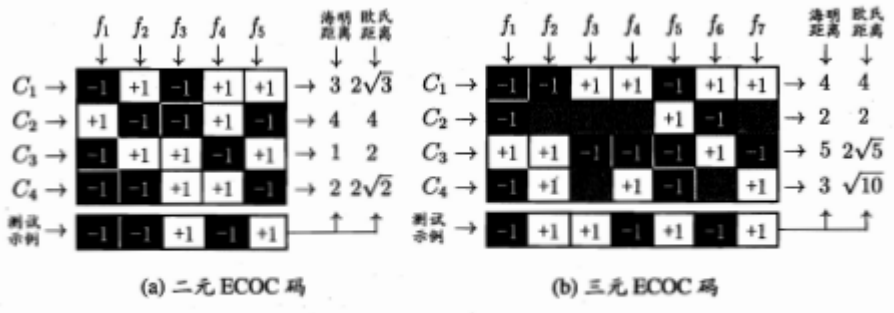


图 3.4 OvO 与 OvR 示意图

MvM 是每次将若干个类作为正类, 若干个其他类作为反类. 显然, OvO 和 OvR 是 MvM 的特例. MvM 的正、反类构造必须有特殊的设计, 不能随意选取. 这里我们介绍一种最常用的 MvM 技术: "纠错输出码" (Error Correcting Output Codes, 简称 ECOC).

ECOC [Dietterich and Bakiri, 1995] 是将编码的思想引入类别拆分, 并尽可能在解码过程中具有容错性. ECOC 工作过程主要分为两步:

- 编码: 对 N 个类别做 M 次划分, 每次划分将一部分类别划为正类, 一部分划为反类, 从而形成一个二分类训练集; 这样一共产生 M 个训练集, 可训练出 M 个分类器.
- 解码: M 个分类器分别对测试样本进行预测, 这些预测标记组成一个编码. 将这个预测编码与每个类别各自的编码进行比较, 返回其中距离最小的类别作为最终预测结果.



(a) 二元 ECOC 码　　(b) 三元 ECOC 码

图 3.5 ECOC 编码示意图. "+1" 、 "−1" 分别表示学习器 f_i 将该类样本作为正、反例; 三元码中 "0" 表示 f_i 不使用该类样本

- 将C1~C4取不同正例/反例，然后训练出对应的 f ，再用 f 预测结果，训练时取的label与预测结果最接近的就是预测的分类。
- 海明距离就是不同位的数量，欧氏距离是看作是坐标，算坐标距离。
    - 海明距离意义下理论最优的ECOC二元码：类与类间最小海明距离最大化
- 三元码就是增加了不使用样本的label

决策树

划分选择

- 引入

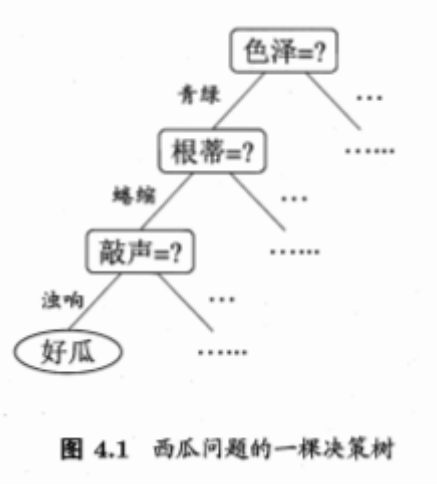


图 4.1 西瓜问题的一棵决策树

- 信息熵(information entropy)

假定当前样本集合 D 中第 k 类样本所占的比例为 p_k $(k = 1, 2, \ldots, |\mathcal{Y}|)$, 则 D 的信息熵定义为

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k \ . \tag{4.1}$$

Ent(D)的值越小，D的纯度越高

- 信息增益

假定离散属性 a 有 V 个可能的取值 $\{a^1, a^2, \ldots, a^V\}$, 若使用 a 来对样本集 D 进行划分, 则会产生 V 个分支结点, 其中第 v 个分支结点包含了 D 中所有在属性 a 上取值为 a^v 的样本, 记为 D^v. 我们可根据式(4.1) 计算出 D^v 的信息熵, 再考虑到不同的分支结点所包含的样本数不同, 给分支结点赋予权重 $|D^v|/|D|$, 即样本数越多的分支结点的影响越大, 于是可计算出用属性 a 对样本集 D 进行划分所获得的“信息增益”(information gain)

$$\mathrm{Gain}(D, a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v) \ . \tag{4.2}$$

一般信息增益越大，使用属性a来进行划分获得的"纯度提升"越大

- 例:

表 4.1 西瓜数据集 2.0

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

以表 4.1 中的西瓜数据集 2.0 为例, 该数据集包含 17 个训练样例, 用以学习一棵能预测没剖开的是不是好瓜的决策树. 显然, $|\mathcal{Y}| = 2$. 在决策树学习开始时, 根结点包含 D 中的所有样例, 其中正例占 $p_1 = \frac{8}{17}$, 反例占 $p_2 = \frac{9}{17}$. 于是, 根据式(4.1)可计算出根结点的信息熵为

$$\mathrm{Ent}(D) = -\sum_{k=1}^{2} p_k \log_2 p_k = -\left(\frac{8}{17}\log_2\frac{8}{17} + \frac{9}{17}\log_2\frac{9}{17}\right) = 0.998 .$$

然后, 我们要计算出当前属性集合 {色泽, 根蒂, 敲声, 纹理, 脐部, 触感} 中每个属性的信息增益. 以属性“色泽”为例, 它有 3 个可能的取值: {青绿, 乌黑, 浅白}. 若使用该属性对 D 进行划分, 则可得到 3 个子集, 分别记为: D^1 (色泽=青绿), D^2 (色泽= 乌黑), D^3 (色泽=浅白).

子集 D^1 包含编号为 {1, 4, 6, 10, 13, 17} 的 6 个样例, 其中正例占 $p_1 = \frac{3}{6}$, 反例占 $p_2 = \frac{3}{6}$; D^2 包含编号为 {2, 3, 7, 8, 9, 15} 的 6 个样例, 其中正、反例分别占 $p_1 = \frac{4}{6}$, $p_2 = \frac{2}{6}$; D^3 包含编号为 {5, 11, 12, 14, 16} 的 5 个样例, 其中正、反例分别占 $p_1 = \frac{1}{5}$, $p_2 = \frac{4}{5}$. 根据式(4.1)可计算出用“色泽”划分之后所获得的 3 个分支结点的信息熵为

$$\mathrm{Ent}(D^1) = -\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) = 1.000 ,$$

$$\mathrm{Ent}(D^2) = -\left(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}\right) = 0.918 ,$$

$$\mathrm{Ent}(D^3) = -\left(\frac{1}{5}\log_2\frac{1}{5} + \frac{4}{5}\log_2\frac{4}{5}\right) = 0.722 ,$$

于是, 根据式(4.2)可计算出属性“色泽”的信息增益为

$$
\begin{aligned}
\text{Gain}(D, \text{色泽}) &= \text{Ent}(D) - \sum_{v=1}^{3} \frac{|D^v|}{|D|} \text{Ent}(D^v) \\
&= 0.998 - \left(\frac{6}{17} \times 1.000 + \frac{6}{17} \times 0.918 + \frac{5}{17} \times 0.722 \right) \\
&= 0.109 .
\end{aligned}
$$

类似的, 我们可计算出其他属性的信息增益:

$$\text{Gain}(D, \text{根蒂}) = 0.143; \quad \text{Gain}(D, \text{敲声}) = 0.141;$$

$$\text{Gain}(D, \text{纹理}) = 0.381; \quad \text{Gain}(D, \text{脐部}) = 0.289;$$

$$\text{Gain}(D, \text{触感}) = 0.006.$$

显然, 属性“纹理”的信息增益最大, 于是它被选为划分属性. 图 4.3 给出了基于“纹理”对根结点进行划分的结果, 各分支结点所包含的样例子集显示在结点中.

图 4.3 基于“纹理”属性对根结点划分

然后, 决策树学习算法将对每个分支结点做进一步划分. 以图 4.3 中第一个分支结点(“纹理=清晰”)为例, 该结点包含的样例集合 D^1 中有编号为 {1, 2, 3, 4, 5, 6, 8, 10, 15} 的 9 个样例, 可用属性集合为 {色泽, 根蒂, 敲声, 脐部, 触感}. 基于 D^1 计算出各属性的信息增益:

$$\text{Gain}(D^1, \text{色泽}) = 0.043; \quad \text{Gain}(D^1, \text{根蒂}) = 0.458;$$

$$\text{Gain}(D^1, \text{敲声}) = 0.331; \quad \text{Gain}(D^1, \text{脐部}) = 0.458;$$

$$\text{Gain}(D^1, \text{触感}) = 0.458.$$

“根蒂”、“脐部”、“触感” 3 个属性均取得了最大的信息增益, 可任选其中之一作为划分属性. 类似的, 对每个分支结点进行上述操作, 最终得到的决策树如图 4.4 所示.

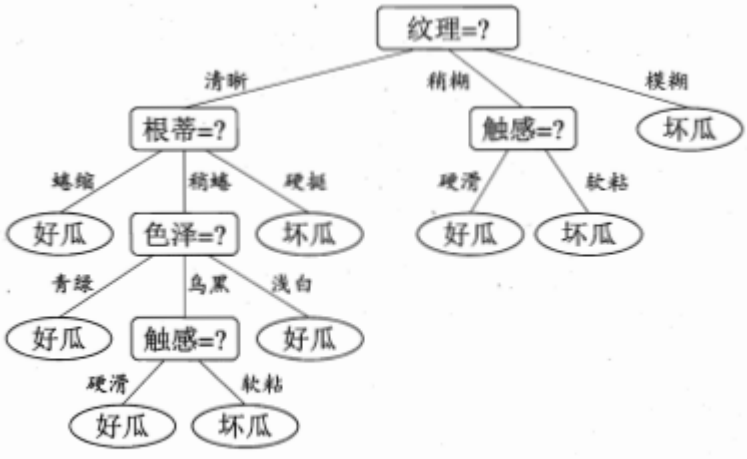


图 4.4 在西瓜数据集 2.0 上基于信息增益生成的决策树

- 增益率(gain ratio)

式(4.2)相同的符号表示, 增益率定义为

$$\text{Gain\_ratio}(D,a)=\frac{\text{Gain}(D,a)}{\text{IV}(a)}, \tag{4.3}$$

其中

$$\text{IV}(a)=-\sum_{v=1}^{V}\frac{|D^v|}{|D|}\log_2\frac{|D^v|}{|D|} \tag{4.4}$$

称为属性 a 的“固有值”(intrinsic value) [Quinlan, 1993]. 属性 a 的可能取值数目越多(即 V 越大), 则 IV(a) 的值通常会越大. 例如, 对表 4.1 的西瓜数据集 2.0, 有 IV(触感) $=0.874$ ($V=2$), IV(色泽) $=1.580$ ($V=3$), IV(编号) $=4.088$ ($V=17$).

- ID3算法构造决策树

```
#假设数据集为D，标签集为A，需要构造的决策树为tree
def ID3(D, A):
    if D中所有的标签都相同:
        return 标签
    if 样本中只有一个特征或者所有样本的特征都一样:
        对D中所有的标签进行计数
        return 计数最高的标签
    计算所有特征的信息增益
    选出增益最大的特征作为最佳特征(best_feature)
    将best_feature作为tree的根结点
    得到best_feature在数据集中所有出现过的值的集合(value_set)
    for value in value_set:
        从D中筛选出best_feature=value的子数据集(sub_feature)
        从A中筛选出best_feature=value的子标签集(sub_label)
        #递归构造tree
        tree[best_feature][value] = ID3(sub_feature, sub_label)
    return tree
```

剪枝处理

> 决策树对付过拟合的主要手段

- 预剪枝：决策树生成时对每个结点在划分前估计
- 后剪枝：训练集生成完整决策树后自底向上对非叶节点考察能否将子树替换成叶子节点

连续与缺失值

- 连续属性离散化：将 D 分为子集 D_t^- 和 D_t^+，D_t^- 表示在属性 a 上取值不大于 t 的样本

$$\begin{aligned}\text{Gain}(D,a)&=\max_{t\in T_a}\ \text{Gain}(D,a,t)\\&=\max_{t\in T_a}\ \text{Ent}(D)-\sum_{\lambda\in\{-,+\}}\frac{|D_t^\lambda|}{|D|}\text{Ent}(D_t^\lambda),\end{aligned}$$

- 缺失值处理：为每个样本 x 赋予一个权重 w_x ，定义

$$\rho = \frac{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in D} w_{\boldsymbol{x}}},$$

$$\tilde{p}_k = \frac{\sum_{\boldsymbol{x} \in \tilde{D}_k} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}} \quad (1 \leqslant k \leqslant |\mathcal{Y}|),$$

$$\tilde{r}_v = \frac{\sum_{\boldsymbol{x} \in \tilde{D}^v} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}} \quad (1 \leqslant v \leqslant V).$$

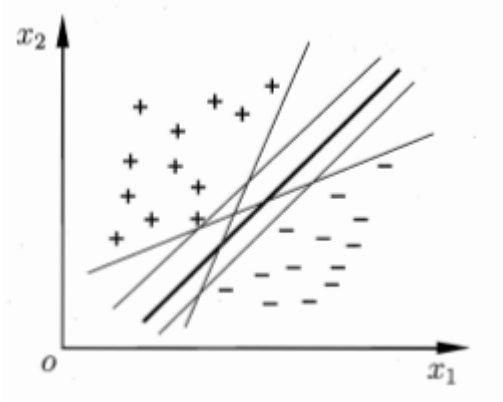
$$\begin{aligned} \operatorname{Gain}(D, a) &= \rho \times \operatorname{Gain}(\tilde{D}, a) \\ &= \rho \times \left(\operatorname{Ent}\left(\tilde{D}\right) - \sum_{v=1}^{V} \tilde{r}_v \operatorname{Ent}\left(\tilde{D}^v\right) \right) \end{aligned}$$

A better tree

- Small trees
  $EPE(f) = (bias)^2 + variance + noise$

支持向量机

对于一个样本集D的分类问题，我们希望找到一个划分超平面，但无法确定哪一个，如下：



划分超平面可表示为 $w^Tx+b=0$，样本空间中任一点 x 到超平面 (w,b) 的距离为 $r=\frac{|w^Tx+b|}{||w||}$

假设超平面 $(\boldsymbol{w},b)$ 能将训练样本正确分类, 即对于 $(\boldsymbol{x}_i,y_i)\in D$, 若 $y_i=+1$, 则有 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b>0$; 若 $y_i=-1$, 则有 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b<0$. 令

$$\begin{cases}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\geqslant+1, & y_i=+1\,;\\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\leqslant-1, & y_i=-1\,.\end{cases}\tag{6.3}$$

如图 6.2 所示, 距离超平面最近的这几个训练样本点使式(6.3)的等号成立, 它们被称为“支持向量”(support vector), 两个异类支持向量到超平面的距离之和为

$$\gamma=\frac{2}{||\boldsymbol{w}||}\,,\tag{6.4}$$

它被称为“间隔”(margin).

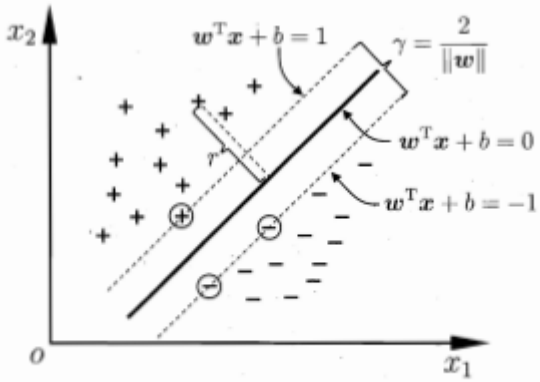


图 6.2 支持向量与间隔

欲找到具有“最大间隔”(maximum margin)的划分超平面, 也就是要找到能满足式(6.3)中约束的参数 $\boldsymbol{w}$ 和 b, 使得 γ 最大, 即

$$\begin{aligned}&\max_{\boldsymbol{w},b}\ \frac{2}{||\boldsymbol{w}||}\\ &\text{s.t. }\ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)\geqslant 1,\quad i=1,2,\ldots,m.\end{aligned}\tag{6.5}$$

显然, 为了最大化间隔, 仅需最大化 $\|\boldsymbol{w}\|^{-1}$, 这等价于最小化 $\|\boldsymbol{w}\|^2$. 于是, 式(6.5)可重写为

$$\begin{aligned}&\min_{\boldsymbol{w},b}\ \frac{1}{2}\|\boldsymbol{w}\|^2\\ &\text{s.t. }\ y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)\geqslant 1,\quad i=1,2,\ldots,m.\end{aligned}\tag{6.6}$$

这就是支持向量机(Support Vector Machine, 简称 SVM)的基本型.

拉格朗日乘子法

对式(6.6)使用拉格朗日乘子法可得到其“对偶问题”(dual problem). 具体来说, 对式(6.6) 的每条约束添加拉格朗日乘子 $\alpha_i \geqslant 0$, 则该问题的拉格朗日函数可写为

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b)\right) , \tag{6.8}$$

其中 $\boldsymbol{\alpha} = (\alpha_1; \alpha_2; \ldots; \alpha_m)$. 令 $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ 对 $\boldsymbol{w}$ 和 b 的偏导为零可得

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i , \tag{6.9}$$

$$0 = \sum_{i=1}^{m} \alpha_i y_i . \tag{6.10}$$

将式(6.9)代入(6.8), 即可将 $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ 中的 $\boldsymbol{w}$ 和 b 消去, 再考虑式(6.10)的约束, 就得到式(6.6)的对偶问题

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j \tag{6.11}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 ,$$

$$\alpha_i \geqslant 0 , \quad i = 1, 2, \ldots, m .$$

解出 $\boldsymbol{\alpha}$ 后, 求出 $\boldsymbol{w}$ 与 b 即可得到模型

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \\ &= \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} + b . \end{aligned} \tag{6.12}$$

如何确定偏移项 b 呢? 注意到对任意支持向量 $(\boldsymbol{x}_s, y_s)$ 都有 $y_s f(\boldsymbol{x}_s) = 1$, 即

$$y_s \left(\sum_{i \in S} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_s + b \right) = 1 , \tag{6.17}$$

其中 $S = \{i \mid \alpha_i > 0, i = 1, 2, \ldots, m\}$ 为所有支持向量的下标集. 理论上, 可选取任意支持向量并通过求解式(6.17)获得 b, 但现实任务中常采用一种更鲁棒的做法: 使用所有支持向量求解的平均值

$$b = \frac{1}{|S|} \sum_{s \in S} \left(y_s - \sum_{i \in S} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_s \right) . \tag{6.18}$$

核函数

得到对偶问题(6.11)后:

求解式(6.21)涉及到计算 $\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$, 这是样本 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 映射到特征空间之后的内积. 由于特征空间维数可能很高, 甚至可能是无穷维, 因此直接计算 $\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$ 通常是困难的. 为了避开这个障碍, 可以设想这样一个函数:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j) \ , \tag{6.22}$$

即 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 在特征空间的内积等于它们在原始样本空间中通过函数 $\kappa(\cdot,\cdot)$ 计算的结果. 有了这样的函数, 我们就不必直接去计算高维甚至无穷维特征空间中的内积, 于是式(6.21)可重写为

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0 \ , \\ & \alpha_i \geqslant 0 \ , \quad i = 1, 2, \ldots, m \ . \end{aligned} \tag{6.23}$$

求解后即可得到

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b \\ &= \sum_{i=1}^{m} \alpha_i y_i \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}) + b \\ &= \sum_{i=1}^{m} \alpha_i y_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i) + b \ . \end{aligned} \tag{6.24}$$

这里的函数 $\kappa(\cdot,\cdot)$ 就是“核函数”(kernel function). 式(6.24) 显示出模型最优解可通过训练样本的核函数展开, 这一展式亦称“支持向量展式”(support vector expansion).

定理 6.1 (核函数) 令 $\mathcal{X}$ 为输入空间, $\kappa(\cdot,\cdot)$ 是定义在 $\mathcal{X}\times\mathcal{X}$ 上的对称函数, 则 κ 是核函数当且仅当对于任意数据 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$, "核矩阵" (kernel matrix) $\mathbf{K}$ 总是半正定的:

$$\mathbf{K}=\begin{bmatrix}\kappa(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_1,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_1,\boldsymbol{x}_m)\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ \kappa(\boldsymbol{x}_i,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_i,\boldsymbol{x}_m)\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ \kappa(\boldsymbol{x}_m,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_m,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_m,\boldsymbol{x}_m)\end{bmatrix}.$$

定理 6.1 表明, 只要一个对称函数所对应的核矩阵半正定, 它就能作为核函数使用. 事实上, 对于一个半正定核矩阵, 总能找到一个与之对应的映射 ϕ. 换言之, 任何一个核函数都隐式地定义了一个称为 "再生核希尔伯特空间" (Reproducing Kernel Hilbert Space, 简称 RKHS)的特征空间.

表 6.1 常用核函数

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j)^d$	$d\geqslant 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{2\sigma^2}\right)$	$\sigma>0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|}{\sigma}\right)$	$\sigma>0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\tanh(\beta\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j+\theta)$	tanh 为双曲正切函数, $\beta>0,\theta<0$