《机器学习》读书笔记

黄帅 [1] [2] [3]

March 24, 2018

[1] https://github.com/ShuaiHuang/MachineLearningNotes
[2] http://shuaihuang.github.io/
[3] shuaihuang.sjtu@gmail.com

Contents

第 1 章　绪论

1.1　引言

机器学习致力于研究如何通过计算的手段, 利用经验来改善系统自身的性能.

1.2　基本术语

本节用西瓜举了一些机器学习相关的基本例子, 引入一些机器学习的术语, 在此不做展开.

1.3　假设空间

归纳 (induction) 是从特殊到一般的 “泛化”(generalize) 过程, 即从具体的事实归结出一般规律;“演绎”(deduction) 则是从一般到特殊的 “特化”(specialization) 过程, 即从基础原理推演出具体情况. 从样例中学习显然是一个归纳过程, 因此亦称 “归纳学习”(inductive learning).

学习过程实质上就是在所有假设组成的空间中进行搜索的过程. 可能有多个假设与训练集一致, 这些假设的集合称之为 “版本空间”.

1.4　归纳偏好

归纳偏好是本章的重要内容, 也是机器学习能够解决实际问题的理论基础.

在上一节版本空间的举例中, 有三种假设满足判定条件, 但是无法进一步判定这三个假设哪一个更好. 但是对于一个具体的机器学习模型而言, 它

必须要产生一个模型, 此时学习算法本身的归纳偏好就会起到重要作用. 任何一个有效的机器学习算法必有其归纳偏好, 否则就会被版本空间中的等效假设所迷惑, 无法产生确定的学习结果.

归纳偏好可看作学习算法在庞大的假设空间中对假设所进行选择的启发式或价值观."奥卡姆剃刀"(Occam's razor) 是一种一般性的原则来引导建立 "正确的" 偏好, 其内容为 "若有多个假设与观察一致, 则选择最简单的那个". 归纳偏好的原则也不能一概而论, 算法的归纳偏好是否与问题本身匹配, 大多数时候决定了算法能否取得好的性能.

假设样本空间 $\mathcal{X}$ 和假设空间 $\mathcal{H}$ 都是分散的, $P(h|\mathbf{X},\mathcal{L}_a)$ 代表算法 $\mathcal{L}_a$ 基于训练数据 $\mathbf{X}$ 产生假设 h 的概率, 再令 f 代表希望学习的真实函数.$\mathcal{L}_a$ 在**训练集外所有样本上产生的误差**为

$$E_{ote}(\mathcal{L}_a|\mathbf{X},f)=\sum_h\sum_{x\in\mathcal{X}-X}P(\mathbf{x})\mathrm{I}(h(\mathbf{x})\neq f(\mathbf{x}))P(h|\mathbf{X},\mathcal{L}_a) \tag{1.1}$$

其中 $\mathrm{I}(\cdot)$ 是指示函数.

考虑二分类问题, 对所有可能的 f 按均匀分布对误差求和, 有

$$\begin{aligned}\sum_f E_{ote}(\mathcal{L}_a|\mathbf{X},f)&=\sum_f\sum_h\sum_{\mathbf{x}\in\mathcal{X}-X}P(\mathbf{x})\mathrm{I}(h(\mathbf{x})\neq f(\mathbf{x}))P(h|X,\mathcal{L}_a)\\&=\sum_{\mathbf{x}\in\mathcal{X}-X}P(\mathbf{x})\sum_h P(h|X,\mathcal{L}_a)\sum_f\mathrm{I}(h(\mathbf{x})\neq f(\mathbf{x}))\\&=\sum_{\mathbf{x}\in\mathcal{X}-X}P(\mathbf{x})\sum_h P(h|X,\mathcal{L}_a)\frac{1}{2}2^{|\mathcal{X}|}\\&=\frac{1}{2}2^{|\mathcal{X}|}\sum_{x\in\mathcal{X}-X}P(\mathbf{x})\sum_h P(h|X,\mathcal{L})\\&=2^{|\mathcal{X}|-1}\sum_{\mathbf{x}\in\mathcal{X}-X}P(\mathbf{x})\cdot 1\end{aligned} \tag{1.2}$$

误差与学习算法无关, 这就是 "没有免费午餐"(No Free Lunch Theorem, NFL 定理). 第二行到第三行推导成立的原因是若 f 均匀分布, 则有一半的 f 对 x 的预测与 $h(x)$ 不一致.

但是需要注意的是,NFL 引入了一个重要前提, 所有 "问题" 出现的机会相同, 或所有问题同等重要, 即所有可能的 f 按均匀分布. 因此必须清楚地认识到, 脱离具体问题空谈 "什么算法更好" 毫无意义. 学习算法自身的归纳偏好与问题是否相匹配, 起到决定性作用.

第 2 章　模型评估与选择

2.1　经验误差与过拟合

“训练误差”(training error) 或 “经验误差”(empirical error) 是指学习器在训练集上的误差;“泛化误差”(generalization error) 是指学习器在新样本上的误差.

过拟合是机器学习面临的关键障碍, 是无法避免的. 从另一个角度来看, 机器学习问题是 NP 难的, 而有效的机器学习算法必然是在多项式时间内运行完成. 若可避免过拟合, 则通过经验误差最小化就能获得最优解, 这就意味这我们构造性的证明了 $P = NP$, 因此, 我们只要相信 $P \neq NP$, 过拟合就不可避免.

对于一个问题, 往往有不同的模型可供选择, 理想的解决方案是对候选模型的泛化误差进行评估, 然后选择泛化误差最小的那个模型.

2.2　评估方法

泛化误差很难直接度量得到, 通常使用在测试集上的测试误差作为泛化误差的近似. 对样本集划分出测试集的方法如下:

2.2.1　留出法

留出法将数据集划分为两个互斥的集合, 其中一个集合作为训练集, 另外一个作为测试集.

需要注意的是, 训练集和测试集的划分要尽可能保持数据分布的一致性. 另外, 一般采用若干次随机划分, 重复进行实验评估后取平均值作为留出法的评估结果.

2.2.2 交叉验证法

交叉验证法将数据集划分为 k 个大小相似的互斥子集, 并且保持每个子集上的数据一致性. 进行 k 轮迭代, 每次用 $k-1$ 个子集的并集作为训练集, 剩余的 1 个子集作为测试集.

如果 k 取值与训练集样本数目相等, 则称为 “留一法”(Leave-One-Out, LOO). 但是留一法未必永远比其他模型准确,NFL 定理对实验评估方法同样适用[1].

2.2.3 自助法

前面所述的留一法受训练样本规模影响较小, 但是计算复杂度较高; 其他方法会受到样本规模不同的影响.“自助法”(boostrapping) 是一种既不受样本规模影响又能高效进行实验估计的方法.

自助法通过对样本进行有放回采样, 对数据集进行划分. 但是这样会改变原始的样本分布, 会引入估计偏差. 在样本数量较多时, 一般使用留出法和交叉验证法多一些.

2.2.4 调参与最终模型

一般模型是在训练集上进行训练, 测试集上进行测试. 如果通过训练过程选取到了最优的参数, 则应将训练出的模型放到全部数据集上进行训练, 得到最终的模型.

通常我们把学得模型在实际使用过程中遇到的数据集称为测试数据集, 模型评估与选择中用于评估测试的数据集称为验证集. 也就是说, 测试集的功能是估计不同算法模型的泛化能力; 验证集的作用是调参和模型选择.

[1]如果训练集和测试集的采样不考虑样本原本分布，所有样本划分做等价考虑，则最终的平均评价指标不会相差太多.

2.3 性能度量

性能度量的目的是对模型的泛化能力有一个评价标准. 在对比不同模型的能力时, 使用不同的性能度量往往会导致不同的评判结果.

回归任务最常用的性能度量是 “均方误差”(mean square error)

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}(f(\mathbf{x}_i)-y_i)^2 \tag{2.1}$$

更一般的情形下, 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

$$E(f;D)=\int_{\mathbf{x}\sim\mathcal{D}}(f(\mathbf{x})-y)^2p(\mathbf{x})d\mathbf{x} \tag{2.2}$$

2.3.1 错误率与精度

错误率和精度是分类任务中最常用的两种性能度量. 错误率是分类错误的样本数占样本总数的比例, 精度是分类正确的样本数占样本总数的比例.

2.3.2 查准率, 查全率与 $F1$

混淆矩阵包含真正例 (true positive), 假正例 (false positive), 真反例 (true negative), 假反例 (false positive), 分别对应为 TP, FP, TN, FP.

查准率 P 与查全率 R 分别定义为

$$\begin{aligned} P&=\frac{TP}{TP+FP}\\ R&=\frac{TP}{TP+FN} \end{aligned} \tag{2.3}$$

查准率和查全率是一对矛盾的度量.

如果根据学习期的预测结果对样例进行排序, 按此顺序逐个把样本作为正例进行预测, 可以计算出当前的查全率, 查准率. 以查准率为纵轴, 以查全率作为横轴, 就得到了查准率-查全率曲线, 简称 P-R 曲线.

笼统地说, 如果一条曲线 A 能完全包住曲线 B, 则称 A 对应的学习器性能优于 B 对应的学习器. 但是如果两条曲线有交叉, 则很难断定那条曲线对应的学习器性能较好. 因此, 有必要对 P-R 曲线进行量化评判.

“平衡点”(Break-Event Point, BEP) 度量的是查准率 = 查全率时的取值.

更常用的是 $F1$ 度量

$$F1 = \frac{2 \times P \times R}{P + R} \tag{2.4}$$

如果对查全率和查准率的关注程度不同, 可用

$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{(\beta^2 \times P) + R} \tag{2.5}$$

其中,$\beta > 0$ 度量了查全率对查准率的相对重要性;$\beta > 1$ 时查全率有更大影响;$\beta < 1$ 时查准率有更大影响.

如果有多个二分类混淆矩阵, 可以分别计算每个混淆矩阵上的查全率与查准率, 然后就可得到对应的宏查准率, 宏查全率, 宏 $F1$

$$\begin{aligned} marco - P &= \frac{1}{n}\sum_{i=1}^{n} P_i \\ marco - R &= \frac{1}{n}\sum_{i=1}^{n} R_i \\ marco - F1 &= \frac{2 \times marco - P \times marco - R}{marco - P + marco - R} \end{aligned} \tag{2.6}$$

如果对这些混淆矩阵元素分别求平均值, 得到 $\overline{TP}, \overline{FP}, \overline{TN}, \overline{FN}$, 可得到微查准率, 微查全率, 微 $F1$

$$\begin{aligned} micro - P &= \frac{\overline{TP}}{\overline{TP} + \overline{FP}} \\ micro - R &= \frac{\overline{TP}}{\overline{TP} + \overline{FN}} \\ micro - F1 &= \frac{2 \times micro - P \times micro - R}{micro - P + micro - R} \end{aligned} \tag{2.7}$$

2.3.3 ROC 与 AUC

ROC 曲线的纵轴是 “真正例率”[2](True Positive Rate, TPR), 横轴是 “假正例率”[3](False Positive Rate), 分别定义为

$$\begin{aligned} TPR &= \frac{TP}{TP+FN} \\ FPR &= \frac{FP}{TN+FP} \end{aligned} \tag{2.8}$$

点 $(0,1)$ 对应于将所有正例排在所有所有负例之前的理想模型; 对角线对应于随机猜测的模型, 即 TPR 与 FPR 取值相同.

笼统比较方式来看, 曲线越靠近左上角表明学习器的性能越好. 量化比较方式来看, 比较合理的通过 ROC 度量学习器性能的方式为比较 ROC 曲线下的面积, 即 AUC(Area Under ROC Curve).

2.3.4 代价敏感错误率与代价曲线

现实任务中, 不同的错误类型所造成的后果不同, 可以认为是 “非均等代价”(unequal cost). 前面的几种度量方式都隐式地假设了均等代价. 可通过 “代价矩阵”(cost matrix) 对代价度量进行扩充.(见 P35)

在非均等代价下,ROC 曲线不能直接反映出学习器的期望总体代价, 而代价曲线 (cost curve) 则可达到该目的.

ROC 曲线上的每一点对应了代价平面上的一条线段. 设 ROC 曲线上一点坐标 (TPR, FPR), 则计算出 FNR, 然后代价平面上有一条从 $(0, FPR)$ 到 $(1, FNR)$ 的线段. 所有线段围成的面积即为在所有条件下学习器的期望总体代价.

2.4 比较检验

性能度量还存在几个重要问题:

1. 我们希望比较的是泛化性能, 不是测试集上的误差

[2]判断为真的正样本占所有正样本的比例.

[3]判断为真的负样本占所有负样本的比例.

2. 测试集上的性能与测试集本身有很大关系

3. 机器学习的算法具有随机性

本章将上述三点纳入考虑范围, 阐述了如何比较模型的性能.

首先对于**单个学习器**的泛化性能的假设提出假设检验. 即考虑 $\epsilon < \epsilon_0$(即误差 ϵ 小于某个常量 ϵ_0), 则在置信度 $1-\alpha$ 的概率内所能观测到的最大错误率. 常用的检验方法有

- 二项检验
- t-检验

比较检验是对**多个学习器性能**进行比较. 常用的检验方法有

- 交叉验证 t-检验
- McNemar 检验
- Fried 检验
- Nemenyi 检验

2.5 偏差与方差

“偏差-方差分解”(bias-variance decomposition) 是解释学习器泛化性能的一种重要工具. 偏差-方差分解试图对学习算法的期望泛化错误率进行拆解. 拆解过程见 P45 推导过程, 最终结果为

$$E(f;D) = bias^2(\mathbf{x}) + var(\mathbf{x}) + \varepsilon^2 \tag{2.9}$$

即泛化误差可以分解为偏差, 方差和噪声之和.

偏差度量了学习算法的期望预测与真实结果的偏离程度; 方差度量了同样大小的训练集的变动所导致的学习器性能的变化; 噪声表达了在当前任务上任何学习算法所能达到的期望泛化误差的下界.

一般来说, 偏差方差是有冲突的, 一般称为 “偏差-方差窘境”(bias-variance dilemma). 训练不足时, 训练数据的扰动不足以使学习器产生显著变化; 训练充足时, 学习器拟合能力非常强, 训练数据的轻微扰动都会使学习器产生显著变化.

第 3 章　线性模型

3.1　基本形式

线性模型 (Linear Model) 本质上就是一个试图学习出属性线性组合来进行预测的函数. 一般向量形式为

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b \tag{3.1}$$

学得 $\boldsymbol{w}$ 和 b 后, 模型就能被确定.

3.2　线性回归

“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. 一般使用最小二乘法找到一条直线, 使得各个样本到直线上的距离之和最小. 用矩阵形式表示为

$$\hat{\boldsymbol{w}} = \arg\min_{\hat{\boldsymbol{w}}}(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}})^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}) \tag{3.2}$$

当 $\boldsymbol{X}^T\boldsymbol{X}$ 为满秩矩阵或正定矩阵时, 有闭式解

$$\hat{\boldsymbol{w}}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3.3}$$

当 $\boldsymbol{X}^T\boldsymbol{X}$ 不是满秩矩阵或正定矩阵时, 有多个解, 选择哪个解作为输出由算法的归纳偏好确定, 常见的做法是引入正则化 (regularization) 项.

同样也可以让线性模型逼近 y 的衍生物, 如

$$\ln y = \boldsymbol{w}^T\boldsymbol{x} + b \tag{3.4}$$

这就是对数线性回归 (log-linear regression). 更一般的, 考虑单调可微函数 $g(\cdot)$, 令

$$y = g^{-1}(\boldsymbol{w}^T\boldsymbol{x} + b) \tag{3.5}$$

这样得到的模型称为 “广义线性模型”(generalized linear model).

3.3 对数几率回归

对数几率回归又称 *Logistic Regression*, 虽然名字中带着回归两个字, 但它是一个分类模型。如果要做的是分类任务, 只需要找一个单调可微函数将分类任务的真实标记 y 与线性回归模型的预测值联系起来. 最理想的是 “单位阶跃函数”(unit-step function), 但是由于其不是连续的, 所以需要找到一个 “替代函数”(surrogate function). 对数几率函数 (logistic function) 正是这样一个函数.

$$y = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x} + b)}} \tag{3.6}$$

估计 $\boldsymbol{w}, b$ 的过程见 P59.

3.4 线性判别分析

线性判别分析 (Linear Discriminant Analysis, LDA) 的思想为: 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, 异类样例投影点尽可能远离.

令 X_i, μ_i, Σ_i 分别表示第 i 类示例的集合, 均值向量, 协方差矩阵. 若将数据投影到直线 $\boldsymbol{w}$ 上, 则两类样本的中心在直线上的投影分别为 $\boldsymbol{w}^T\boldsymbol{\mu}_0$ 和 $\boldsymbol{w}^T\boldsymbol{\mu}_1$; 若将所有样本点都投影到直线上, 则两类样本的协方差分别为 $\boldsymbol{w}^T\boldsymbol{\Sigma}_0\boldsymbol{w}$ 和 $\boldsymbol{w}^T\boldsymbol{\Sigma}_1\boldsymbol{w}$. 则可得到最大化的目标

$$\begin{aligned} J &= \frac{\|\boldsymbol{w}^T\boldsymbol{\mu}_0 - \boldsymbol{w}^T\boldsymbol{\mu}_1\|_2^2}{\boldsymbol{w}^T\boldsymbol{\Sigma}_0\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{\Sigma}_1\boldsymbol{w}} \\ &= \frac{\boldsymbol{w}^T(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^T\boldsymbol{w}}{\boldsymbol{w}^T(\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)\boldsymbol{w}} \end{aligned} \tag{3.7}$$

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

$$S_w = \Sigma_0 + \Sigma_1 \tag{3.8}$$

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

$$S_b = (\boldsymbol{\mu_0} - \boldsymbol{\mu_1})(\boldsymbol{\mu_0} - \boldsymbol{\mu_1})^T \tag{3.9}$$

求解过程见 P61-62.

3.5 多分类学习

在实际中, 有些二分类学习方法可以直接推广到多分类, 但是在大部分情形下, 我们利用二分类学习器来解决多分类问题. 通常采用的方法是 “拆解法”, 即将多分类任务拆解成二分类任务. 常用的拆解策略有:

- 一对一 (One vs. One, OvO): 训练 $N(N-1)/2$ 个分类器将 N 个类别两两分开;
- 一对其余 (One vs. Rest, OvR): 训练 N 个分类器, 将每个样本与其余的样本分开;
- 多对多 (Many vs. Many, MvM): 常见的有 “纠错输出编码”(Error Correcting Output Codes, ECOC) 技术. 见 P65.

ECOC 工作过程分为两步: 编码和解码. 在编码过程中, 对 N 个类别做 M 次划分, 每次划分将一部分类别划为正类, 一部分划分为反类; 解码过程中, 使用 M 个分类器对测试样本进行预测, 这些预测标记组成一个编码, 然后将预测编码与每个类别各自的编码进行比较, 返回距离最小的类别作为最终的预测结果.

ECOC 在码长较小时可以根据这个原则计算出理论最优编码. 然而码长较长时就难以有效地确定最优编码, 这也是个 NP 难问题.

3.6 类别不平衡问题

类别不平衡 (class-imbalance) 就是指分类任务中不同类别的训练样例数目差别很大的情况. 从线性分类器角度来看, 用 $y = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b$ 对样本 $\boldsymbol{x}$ 进行分类时, 实际上是用 y 与某一个阈值进行比较. 通常用 0.5 作为阈值. 但是由于我们通常假设训练集是真实样本总体的无偏采样, 因此有如下的 “再平衡” 策略:

观测几率就代表了真实几率, 设 m^+ 和 m^- 分别为正例数目和反例数目, 判别形式为

$$若\frac{y}{1-y} > \frac{m^+}{m^-}时, 预测为正例 \tag{3.10}$$

由于我们未必能基于训练集观测几率来推断出真实的几率, 现在有三种做法:

- 直接对训练集里的反类样例进行欠采样 (undersampling)
- 对训练集里的正类样例进行过采样 (oversampling)
- 直接基于原始训练集进行学习, 将式 (3.10) 所代表的再平衡策略纳入到决策过程中, 称为 “阈值移动”(shreshold-moving).

总结

机器学习有四要素: 数据集上特征提取方式, 损失函数, 优化过程和模型. 从本章开始, 注意总结各种模型的损失函数和优化过程.

Table 3.1: 模型总结

模型	**损失函数**	**优化过程**	**判别函数**
线性模型	均方误差 MSE	最小二乘法，有闭式解	$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$
Logistic 回归	最大化对数似然	梯度下降法, 牛顿法等	
线性判别分析	广义瑞利商	拉格朗日乘子法	

第 4 章　决策树

4.1　基本流程

决策树 (decision tree) 生成过程是一个递归过程, 即从当前结点寻找最优属性, 根据属性的不同取值对当前节点对应的样本集进行划分. 有三种情形会导致递归返回:

- 当前结点包含的样本全属于同一个类别, 无需划分;
- 当前属性集为空, 或是所有样本在所有属性上取值相同, 无法划分;
- 当前结点包含的样本集合为空, 不能划分.

4.2　划分选择

决策树生成过程中最重要的一个环节是如何选取最优属性在当前结点对应的样本集上进行划分, 我们希望决策树的分支节点所包含的样本尽可能属于同一个类别, 即节点的 “纯度”(purity) 越来越高.

4.2.1　信息增益

“信息熵”(information entropy) 是度量样本集合纯度最常用的一种指标. 设当前样本集合 $\mathcal{D}$ 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

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

$\mathrm{Ent}(D)$ 的值越小,D 的纯度越高.

设 D^v 表示 D 中所有在属性 a 上取值为 v 的样本, 记为 D^v, 则有 “信息增益”(information gain)

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

信息增益对取值数目较多的属性有偏好, 通常使用 “增益率”(gain ratio) 来选择最优划分属性.

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

其中,

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

称为属性 a 的 “固有值”(intrinsic value).

增益率准则可对取值数目较少的属性有所偏好.

4.2.2 基尼指数

CART 决策树使用 “基尼指数”(Gini index) 来选择划分属性, 数据集 D 纯度度量方式为

$$\mathrm{Gini}(D)=\sum_{k=1}^{|\mathcal{Y}|}\sum_{k'\neq k}p_kp_{k'}=1-\sum_{k=1}^{|\mathcal{Y}|}p_k^2 \tag{4.5}$$

属性 a 的基尼指数为

$$\mathrm{Gini\_index}(D,a)=\sum_{v=1}^{V}\frac{|D^v|}{|D|}\mathrm{Gini}(D^v) \tag{4.6}$$

4.3 剪枝处理

决策树剪枝的策略有 “预剪枝”(prepruning) 和 “后剪枝”(postpruning). 预剪枝是在决策树生成过程中对每个结点在划分前进行估计, 若当前结点的划分不能带来泛化性能的提升, 则停止划分并将当前结点标记为叶节点; 后剪枝则是先从训练集生成一颗完整的决策树, 然后自底向上地对非叶结

点进行考察, 若将该结点对应的子树替换为叶节点能提升泛化性能, 则将该子树替换为叶结点.

通常将数据集划分为训练集和验证集. 采用预剪枝方法时, 根据训练集选取最优属性, 如果选取的最优属性可以提升验证集的精度时, 则进行划分, 否则不进行划分. 采用后剪枝方法时, 自底向上选择非叶结点, 依次替换为叶结点, 如果精度提升, 则进行替换, 否则不进行替换.

4.4 连续与缺失值

4.4.1 连续值处理

在 C4.5 决策树算法中, 对于属性取值为连续值, 最简单的处理方式采用二分法 (bi-partition) 对连续属性进行处理. 设连续属性 a 在样本集 D 上出现了 n 个不同的取值 $\{a^1, a^2, \ldots, a^n\}$, 基于划分点 t 可将 D 分为子集 D_t^- 和 D_t^+, 则划分点集合为 $T_a = \{\frac{a^i+a^{i+1}}{2} | 1 \leq i \leq n-1\}$, 可得信息增益

$$\begin{aligned} \mathrm{Gain}(D, a) &= \max_{t \in T_a} \mathrm{Gain}(D, a, t) \\ &= \max_{t \in T_a} \mathrm{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \mathrm{Ent}(D_t^\lambda) \end{aligned} \tag{4.7}$$

其中 $\max_{t \in T_a} \mathrm{Gain}(D, a, t)$ 是样本集 D 基于划分点 t 二分后的信息增益.

与离散属性不同, 若当前结点划分属性为连续属性, 该属性还可作为其后代结点的划分属性.

4.4.2 缺失值处理

缺失值处理主要解决以下两个问题:

1. 如何在属性值缺失的情况下进行划分属性选择;
2. 给定划分属性, 若样本在该属性上的值缺失, 如何对样本进行划分.

对于问题 1, 将信息增益推广为

$$\mathrm{Gain}(D, a) = \rho \times \mathrm{Gain}(\tilde{D}, a) \tag{4.8}$$

其中 ρ 为没有属性缺失值样本占所有样本的比例,$\tilde{D}$ 为无属性缺失值的样本集合.

对于问题 2, 需要将缺失属性值样本同时划分到不同的结点中, 并调整 P86 定义的权重值, 即让同一个样本以不同的概率概率划分到不同的子结点中去.

4.5 多变量决策树

若把每个属性视为坐标空间中的一个坐标轴, 则 d 个属性描述的样本就对应了 d 维空间中的一个数据点, 决策树就是在这些数据点集合中寻找一个分类边界. 决策树所形成的边界有一个特点即分类边界平行于坐标轴.

如果样本集合分布较为复杂, 则需要较多的分类边界去拟合真实的分类边界. 若能使用斜的分类边界, 则决策树模型将大为简化.“多变量决策树”(multivariate decision tree) 就是能实现这样的 “斜划分” 甚至更复杂划分的决策树.

总结

决策树的生成过程是一个递归的过程, 每一次都需要从当前的属性集中选取出一个属性 (也有选取多个属性的算法, 这里不做展开.), 按照不同的属性值将对应的样本划分到对应的子节点中. 一般我们希望划分出的子节点的熵要比父节点的熵低, 这里就涉及到一个属性选取标准.

Table 4.1: 子节点属性划分标准

决策树生成算法	属性划分标准	备注
ID3	信息增益 $\mathrm{Gain}(D,a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{\lvert D^v\rvert}{\lvert D\rvert}\mathrm{Ent}(D^v)$	对取值数目较多的属性有偏好
C4.5	增益率 $\mathrm{Gain\_ratio}(D,a) = \frac{\mathrm{Gain}(D,a)}{\mathrm{IV}(a)}$，其中 $\mathrm{IV}(a) = -\sum_{v=1}^{V} \frac{\lvert D^v\rvert}{\lvert D\rvert} \log_2 \frac{\lvert D^v\rvert}{\lvert D\rvert}$	对取值数目较少的属性有所偏好
CART	基尼指数 $\mathrm{Gini\_index}(D,a) = \sum_{v=1} V\frac{D^v}{D}\mathrm{Gini}(D,a)$，其中 $\mathrm{Gini}(D) = \sum_{k=1}^{\lvert\mathcal{Y}\rvert}\sum_{k'\neq k} p_k p_{k'} = 1 - \sum_{k=1}^{\lvert\mathcal{Y}\rvert} p_k^2$	分类和回归任务都适用

第 5 章　神经网络

5.1　神经元模型

神经网络的基本单元为如 P97 底部图 5.1 所示的 M-P 神经元模型, 其数学模型可以抽象为

$$y = f\Big(\sum_{i=1}^{n} w_i x_i - \theta\Big) \tag{5.1}$$

其中,x_i 是当前神经元的输入信号;w_i 是上一层第 i 个神经元与当前神经元的连接权重;θ 为当前神经元的激活阈值;$f(\cdot)$ 是激活函数. 理论上最优的激活函数是阶跃函数, 但是由于其数学性质并不是特别好, 所以一般使用 Sigmoid 函数作为最优近似.

从数学角度上看, 神经网络就是由大量的 $f(\sum_i w_i x_i - \theta)$ 嵌套而得.

5.2　感知机与多层网络

感知机 (Perceptron) 由两层神经元组成, 输入层接收外界输入信号传递给输出层, 输出层是 M-P 神经元. 需要从训练数据集中训练得到输出层权重 w_i 以及阈值 θ. 可以将阈值 θ 看做一个输入权重固定为 -1 的 “哑结点”(dummy node), 这样权重和阈值的学习过程就可以归一化为权重的学习. 对于训练样例 $(\mathbf{x}, y)$, 当前感知机的输出为 $\hat{y}$, 则感知机权重调整策略为[1]

$$\begin{aligned} w_i &\leftarrow w_i + \triangle w_i \\ \triangle w_i &= \eta(y - \hat{y}) x_i \end{aligned} \tag{5.2}$$

其中 $\eta \in (0, 1)$ 称为学习率,x_i 是 $\boldsymbol{x}$ 的对应于第 i 个输入神经元的分量[2].

[1]这里是感知机权重调整策略

[2]感知器学习策略见总结部分

感知机只有输出层神经元进行激活函数处理, 学习能力非常有限. 要解决非线性可分问题, 就要使用多层功能神经元, 通常使用的是 “多层前馈神经网络”.

5.3 误差逆传播算法

要训练多层神经网络, 最常用的是学习算法是误差逆传播 (error Back-Propagation, BP) 算法. 误差逆传播算法对**均方误差**求取目标的负梯度方向对参数进行调整[3], 调整方向为从输出层开始逐步调整到输入层[4].

标准 *BP* 算法每次针对一个训练样本进行参数优化, 更新频率较快. 为了解决这一问题, 使用累积 *BP* 算法, 即对所有样本求取累积均方误差, 然后更新一次参数.

BP 神经网络经常遭遇过拟合, 有两种策略可以解决过拟合问题:

- “早停”(early stopping): 将数据分为训练集和验证集, 若训练集误差降低但是验证集误差升高, 则停止训练;
- “正则化”(regularization): 将误差目标函数中增加一个用于描述网络复杂度的部分.

5.4 全局最小与局部极小

基于梯度的搜索是使用最为广泛的参数寻优方法. 如果误差函数具有多个局部极小, 则不能保证找到的解是全局最小. 通常采用的 “跳出” 局部极小的方法有:

- 以多组不同参数值初始化多个神经网络, 从中选取更有可能获得更接近全局极小值的结果;
- 使用 “模拟退火”(stimulated annealing) 技术, 在每一步以一定的概率接受比当前解更差的结果;

[3]广义感知器学习规则

[4]链式求导法则

- 使用随机梯度下降.

遗传算法 (genetic algorithms) 也常用来训练神经网络以更好地逼近全局最小.

但是上述方法大多是启发式, 缺乏理论保障.

5.5 其他常见的神经网络

5.5.1 RBF 网络

RBF(Radial Basis Function, 径向基函数) 是一种单隐层前馈神经网络, 它使用径向基函数 $\rho(\boldsymbol{x}, \boldsymbol{c}_i)$ 作为隐层神经元激活函数. 具有足够多隐层神经元的 RBF 网络能以任意精度逼近任意连续函数. 常用的高斯径向基函数形如

$$\rho(\boldsymbol{x}, \boldsymbol{c}_i) = e^{-\beta_i \|\boldsymbol{x}-\boldsymbol{c}_i\|^2} \tag{5.3}$$

5.5.2 ART 网络

竞争型学习 (competitive learning) 是神经网络中一种常用的无监督学习策略. 网络的输出神经元相互竞争, 每一时刻仅有一个竞争获胜的神经元被激活, 其他神经元状态被抑制.

ART 网络比较好地缓解了竞争型学习中的 “可塑性-稳定性窘境”(stability-plasticity dilemma), 这就使得 ART 网络具有一个很重要的优点: 可进行增量学习 (incremental learning) 或在线学习 (online learning).

5.5.3 SOM 网络

SOM(Self-Organizing Map, 自组织映射) 网络是一种竞争学习型无监督神经网络, 它能将高维输入数据映射到低维空间, 同时保持输入数据在高维空间的拓扑结构.

5.5.4 级联相关网络

机构自适应网络将网络结构也作为学习的目标之一, 级联相关 (Cascade-Correlation) 网络是结构自适应网络的重要代表.

5.5.5 Elman 网络

“递归神经网络”(recurrent neural networks) 允许网络中出现环形结构, 从而能处理与时间有关的动态变化.

5.5.6 Boltzmann 机

Boltzmann 机是一种 “基于能量的模型”(energy-based model): 它将网络状态定义一个 “能量”, 能量最小化时网络达到理想状态, 而网络的训练就是在最小化这个能量函数.

Boltzmann 机中的神经元都是布尔型的, 状态 1 表示激活, 状态 0 表示抑制. 令向量 $\mathbf{s} \in \{0,1\}^n$ 表示 n 个神经元的状态,w_{ij} 表示神经元 i 与 j 之间的连接权,θ_i 表示神经元 i 的阈值, 则状态 $\mathbf{s}$ 所对应的 Boltzmann 机能量定义为

$$E(\mathbf{s}) = -\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}s_i s_j - \sum_{i=1}^{n}\theta_i s_i \tag{5.4}$$

标准 Boltzmann 机是一个全连接图, 难以解决实际任务, 通常采用受限 Boltzmann 机解决实际问题. 受限 Boltzmann 机仅保留了显层与隐层之间的连接, 从而将 Boltzmann 机结构由完全图简化为二部图.

受限 Boltzmann 机常用 “对比散度”(Contrastive Divergence) 算法来进行训练.

5.6 深度学习

多隐层神经网络难以使用经典算法进行训练, 因为误差在多隐层内逆传播时, 往往会发散而不能收敛到稳定状态.

无监督逐层训练 (unsupervised layer-wise training) 是多隐层网络训练的有效手段, 其基本思想是每次训练一层隐结点, 而本层隐结点的输出作为

下一层隐结点的输入, 这称为 “预训练”(pre-training); 在预训练全部完成后, 再对整个网络进行 “微调”(fine-tuning) 训练.

另一种节省训练开销的策略是 “权共享”(weight sharing), 即让一组神经元使用相同的连接权.

总结

感知器学习策略推导

感知器的损失函数的一种选择是**所有误分类点到判别平面的总距离**, 即有

$$-\frac{1}{\|w\|}\sum_{x_i\in M} y_i(w\cdot x_i+b) \tag{5.5}$$

其中 M 是所有误分类点的集合. 通常不考虑 $\frac{1}{\|w\|}$, 通过使得损失函数 (式5.6) 最小化获得参数 w, b:

$$\min_{w,b} L(w,b) = -\sum_{x_i\in M} y_i(w\cdot x_i+b) \tag{5.6}$$

则有

$$\begin{aligned}\nabla_w L(w,b) &= -\sum_{x_i\in M} y_i x_i \\ \nabla_b L(w,b) &= -\sum_{x_i\in M} y_i\end{aligned} \tag{5.7}$$

根据上式中梯度可以对 w, b 进行更新

$$\begin{aligned}w &\leftarrow w+\eta y_i x_i \\ b &\leftarrow b+\eta y_i\end{aligned} \tag{5.8}$$

第 6 章　支持向量机

本章的内容从逻辑上来说就是支持向量机的推导过程, 其中涉及部分关于最优化理论的知识点, 需要掌握: 拉格朗日乘子法, 对偶问题,KKT 条件, 松弛变量 (见附录 B).

6.1　间隔与支持向量

给定训练样本集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}, y_i \in \{-1, +1\}$, 从直观上看, 应该去找位于两类样本间的超平面去划分两类样本, 而位于两类样本 "正中间" 的划分超平面泛化性能最好[1]. 这就是支持向量机的最核心思想.

空间中超平面可以通过

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0 \tag{6.1}$$

来描述, 其中 $\mathbf{w} = (w_1; w_2; \cdots, w_d)$ 为法向量;b 为位移项; 支持向量机的目标就是为了寻找参数 $(\mathbf{w}, b)$.

假设超平面 $(\mathbf{w}, b)$ 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 $\boldsymbol{w^T x_i} + b > 0$; 若 $y_i = -1$ 则有 $\boldsymbol{w^T x_i} + b < 0$. 令

$$\begin{cases} \boldsymbol{w^T x_i} + b \geq +1, y_i = +1 \\ \boldsymbol{w^T x_i} + b \leq -1, y_i = -1 \end{cases} \tag{6.2}$$

距离超平面最近的样本点 (即支持向量) 使得上式中等号成立. 两个异类支持向量到超平面的距离值和为

$$\gamma = \frac{2}{\|\boldsymbol{w}\|} \tag{6.3}$$

[1]首先要保证特征空间选取合适，即正负样本间要存在分类间隔.

γ 也被称为间隔.

因此 SVM 基本型可以被定义为

$$\begin{aligned} &\max_{\boldsymbol{x},b} \frac{2}{\|\boldsymbol{w}\|} \\ &\text{s.t. } y_i(\boldsymbol{w_i^T x_i} + b) \geq 1, \quad i = 1, 2, \cdots, m. \end{aligned} \tag{6.4}$$

上述问题等价为

$$\begin{aligned} &\min_{\boldsymbol{x},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{s.t. } y_i(\boldsymbol{w_i^T x_i} + b) \geq 1, \quad i = 1, 2, \cdots, m. \end{aligned} \tag{6.5}$$

这就是 SVM 的基本型.

6.2 对偶问题

SVM 基本型本身一个凸二次规划问题, 可以用现成的优化计算包求解. 但是还有效率提升的空间. 基本思路就是**通过拉格朗日乘子法求得其 "对偶问题"(dual problem)**[2].

该问题的拉格朗日函数可以写为

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\mathbf{w^T x_i} + \mathbf{b})) \tag{6.6}$$

令 L 对 $\boldsymbol{w}$ 和 b 的偏导为 0, 并代回拉格朗日函数消去 $\boldsymbol{w}$ 和 b, 得到对偶问题

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^T x_j} \\ &\text{s.t. } \sum_{i=1}^{m} \alpha_i y_i = 0, \\ &\qquad \alpha_i \geq 0, \quad i = 1, 2, \cdots, m. \end{aligned} \tag{6.7}$$

[2]为什么可以这样做? 见附录 B.1, 在推导拉对偶问题时, 常通过将拉格朗日乘子 $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ 对 $\boldsymbol{x}$ 求导并令导数为 0, 来获得对偶函数的表达形式. 值得注意的是, 在强对偶性成立时 (即主问题为凸优化问题), 将拉格朗日函数分别对原变量和对偶变量求导, 再并令导数等于 0, 即可得到原变量与对偶变量的数值关系.

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

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b \\ &= \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{T}} \boldsymbol{x} + b \end{aligned} \tag{6.8}$$

α_i 是拉格朗日乘子, 它恰好对应着训练样本 $(\boldsymbol{x}_i, y_i)$, 注意到对偶问题中有不等式约束, 因此上述推导过程需要满足 KKT(Karush-Kuhn-Tucker) 条件,

$$\begin{cases} \alpha_i \geq 0 \\ y_i f(\boldsymbol{x}_i) - 1 \geq 0 \\ \alpha_i (y_i f(\boldsymbol{x}_i) - 1) = 0 \end{cases} \tag{6.9}$$

训练完成后, 模型仅与支持向量有关.

上述问题可以通过 SMO(Sequential Minimal Optimization) 方法求解, 其基本思路是固定除 α_i 之外的所有参数, 然后求 α_i 上的极值, 则 α_i 可由其他变量导出. 于是,SMO 每次选择两个变量 α_i 和 α_j, 并固定其他参数. 这样在参数初始化后,SMO 不断执行如下两个步骤直至收敛:

- 选取一对需要更新的变量 α_i 和 α_j;
- 固定 α_i 和 α_j 以外的参数, 求解对偶问题获得更新后的 α_i 和 α_j.

6.3 核函数

前两节中, 暗含有假设两类样本是线性可分的. 但是在现实中, 这样的条件很难被满足. 对于这种情况, 可以将样本从原始空间映射到一个更高维的特征空间, 使得样本在这个特征空间内线性可分.

原问题可以推广为

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x_i})^T \phi(\boldsymbol{x_j}) \\ &\text{s.t.} \sum_{i=1}^{m} \alpha_i y_i = 0, \\ &\quad \alpha_i \geq 0, \quad i = 1, 2, \cdots, m. \end{aligned} \tag{6.10}$$

上式涉及到计算 $\phi(\boldsymbol{x_i})^T\phi(\boldsymbol{x_j})$, 由于特征空间的维数很高, 因此直接计算很困难. 可以通过设置

$$\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) =< \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}) >= \phi(\boldsymbol{x_i})^T\phi(\boldsymbol{x_j}) \tag{6.11}$$

只需要 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 在原始空间中计算函数 $\kappa(\cdot,\cdot)$ 就可以得到在特征空间中的内积.

因此可得

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

上式显示出模型最优解可以通过训练样本的核函数展开, 这一展式又被称为 “支持向量展式”(support vector expansion).

一些核函数性质可见 P128-P129.

6.4 软间隔与正则化

另外一种解决训练样本线性不可分的方法是 “软间隔”(soft margin), 即允许支持向量机在一些样本上出错. 因此, 优化目标可以总结为

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} l_{0/1}(y_i(\boldsymbol{w^T x_i} + b) - 1) \tag{6.13}$$

其中 $C > 0$ 是一个常数,$l_{0/1}$ 是 “0/1” 损失函数

$$l_{0/1} = \begin{cases} 1, \text{ if } \quad z < 0 \\ 0, otherwise \end{cases} \tag{6.14}$$

常见的损失函数有 hinge 损失, 指数损失, 对率损失 (见 P130).

采用 hinge 损失, 引入 “松弛变量”(slack variables), 上式可重写为

$$\begin{aligned} &\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m} \xi_i \\ &\text{s.t. } y_i(\boldsymbol{w^T x_i} + b) \geq 1 - \xi_i \\ &\quad \xi_i \geq 0,\ i = 1, 2, \cdots, m. \end{aligned} \tag{6.15}$$

对偶问题推导以及解稀疏性证明见 P132.

6.5 支持向量回归

支持向量引申到回归问题上即可得到支持向量回归 (Support Vector Regression, SVR).SVR 假设我们能容忍 $f(\boldsymbol{x})$ 与 y 之间最多有 ϵ 的偏差, 即仅当 $f(\boldsymbol{x})$ 与 y 之间的差别绝对值大于 ϵ 时才计算损失.SVR 问题可化为

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m} l_\epsilon(f(\boldsymbol{x}_i) - y_i) \tag{6.16}$$

其中 C 为正则化常数,l_ϵ 为 ϵ-不敏感损失函数

$$l_\epsilon = \begin{cases} 0, & \text{if} |z| \leq \epsilon \\ |z| - \epsilon, otherwise \end{cases} \tag{6.17}$$

对偶问题推导以及 KKT 条件证明见 P135-137.

6.6 核方法

给定训练样本, 不考虑偏移项 b, 则无论 SVM 还是 SVR, 学得的模型总能表示成核函数 $\kappa(\boldsymbol{x_i}, \boldsymbol{x_j})$ 的线性组合. 因此有更一般的 “表示定理”(representer theorem).P137

一般人们通过 “核化”(引入核函数) 来将线性学习器拓展为非线性学习器.P137-139 即以线性判别分析为例, 将其进行非线性拓展, 得到 “核线性判别分析”(Kernelized Linear Discriminant Analysis).

第 7 章　贝叶斯分类器

7.1　贝叶斯判别核心

最小化错误率贝叶斯分类器定义为

$$h^*(\boldsymbol{x}) = argmax_{c\in\mathcal{Y}} P(c|\boldsymbol{x}) \tag{7.1}$$

即选择能使后验概率最大的类别标记.

- 后验概率两种不同的生成方式
  - 判别式模型 discriminative model 直接生成 $P(c|\boldsymbol{x})$
  - 生成式模型 generative model 先生成 $P(\boldsymbol{x}, c)$ 再由之推导出 $P(c|\boldsymbol{x})$
- 进一步分解

$$P(c|\boldsymbol{x}) = \frac{P(\boldsymbol{x}, c)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|c)P(c)}{P(\boldsymbol{x})} \tag{7.2}$$

- 其中，$P(\boldsymbol{x})$ 和 $P(c)$ 可以从抽样样本中估计得出
- 问题是如何得出 $P(\boldsymbol{x}|c)$，进而推导出后验概率 $P(c|\boldsymbol{x})$

7.2　极大似然估计

频率主义学派

估计条件概率的一种常用策略是先假定具有某种确定的概率分布形式，然后基于训练样本对概率分布的参数进行估计.

频率主义学派与贝叶斯学派之争

- 频率主义学派观点：模型参数是一个客观固定值
- 贝叶斯学派观点：模型参数是一个未观察到的随机变量，服从一定的分布规律
- 因此，求取 $P(\boldsymbol{x}|c)$ 的过程进一步被归结为求取 $P(\boldsymbol{x}|\boldsymbol{\theta}_c)$ 的过程

极大似然估计

- 最优模型参数 $\boldsymbol{\theta_c}$ 一定是能够让当前观察值以最大概率出现的参数
- 似然定义 $P(\mathcal{D}_c|\boldsymbol{\theta_c}) = \prod_{\boldsymbol{x}\in\mathcal{D}_c} P(\boldsymbol{x}|\boldsymbol{\theta_c})$
- 防止下溢，取对数 $\log P(\mathcal{D}_c|\boldsymbol{\theta_c}) = \sum_{\boldsymbol{x}\in\mathcal{D}_c} \log P(\boldsymbol{x}|\boldsymbol{\theta_c})$
- 极大似然估计缺点模型必须准确，如果模型不准确，即使参数准确，也有可能出现偏差。即数据潜在分布规律必须要与估计分布相一致

7.3 朴素贝叶斯分类器

贝叶斯学派

- $P(\boldsymbol{x}|c)$ 是所有属性值上的联合分布，对其进行估计存在一定的困难
- 引入前提假设, 对模型进行简化：所有的属性值 x_i 相互独立

$$P(c|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|c)P(c)}{P(\boldsymbol{x})} = \frac{P(c)\prod_{i=1}^{d} P(x_i|c)}{P(\boldsymbol{x})}$$

- 由样本集 $\mathcal{D}$ 得出：每个类别 c 上属性值 x_i 的条件概率 $P(x_i|c)$
- 为防止某个类别上属性值样本数为 0，对样本进行平滑，常用拉普拉斯修正
  - $\widehat{P}(c) = \frac{|D_c|+1}{|D|+N}$
  - $\widehat{P}(x_i|c) = \frac{|D_{c_i x_i}|+1}{|D_c|+N_i}$

– 拉普拉斯修正在样本数量较少时避免了因为某些属性上样本数量为 0 而产生估计偏差；在样本量较大时，引入先验的影响可以忽略不计

7.4 半朴素贝叶斯分类器

- 朴素贝叶斯分类器的前提条件为：所有属性之间相互独立。然而这一条件在实际中很难被满足，由此，半朴素贝叶斯分类器（semi-naïve bayesian classifiers）被提出，用以改进朴素贝叶斯前提条件中的限制性因素。半朴素贝叶斯分类器适当考虑一部分属性间的依赖信息.

- One Dependent Estimator(ODE) 即独依赖估计是半朴素贝叶斯分类器的一个常用策略。独依赖估计是指每一个因素在类别之外至多仅依赖一项其他因素。

$$P(c|\boldsymbol{x}) \propto P(c)\prod_{i=1}^{d} P(x_i|c, pa_i) \tag{7.3}$$

- Super Parent ODE(SPODE) 即超父独依赖估计假定所有的属性都依赖于同一个父属性

- Tree Augmented naïve bayesian(TAN) 通过最大带权生成树算法的基础上，通过以下步骤对属性间依赖关系进行简化

  1. 计算两两属性间的条件互信息

$$I(x_i, x_j|y) = \sum_{x_i,x_j,c\in\mathcal{Y}} P(x_i, x_j|c)\log\frac{P(x_i, x_j|c)}{P(x_i|c)P(x_j|c)} \tag{7.4}$$

  2. 生成全连接图，以互信息为两两结点间的权重
  3. 由此构建最大带权生成树需要进一步学习，挑选根向量，将边置为有向
  4. 增加类别节点 y，增加从类别节点到各个属性节点的有向边

- Averaged ODE(AODE) 是一种基于集成学习机制，更为强大的独依赖分类器

1. AODE 尝试将每一个节点定义成父节点，构建 SPODE
2. 将那些有足够的数据支撑的 SPODE 集成起来作为最终结果

$$P(c|\boldsymbol{x}) \propto \sum_{i=1,|D_{x_i}|\geq m'}^{d} P(c,x_i)\prod_{j=1}^{d} P(x_j|c,x_i) \tag{7.5}$$

3. AODE 无需模型选择

- 如果引入高阶属性依赖?
  - 如果训练样本足够多，则泛化能力可以增强
  - 如果训练样本不足，则陷入高阶联合概率的泥沼

7.5 贝叶斯网

- 贝叶斯网亦称信念网（belief network），使用有向非环图（Directed Acyclic Graph）来说明各个属性之间的依赖关系。
- 有向非环图由结构 G 和参数 Θ 两部分构成，即

$$G =< B|\Theta > \tag{7.6}$$

7.5.1 结构

- 贝叶斯网有效表达出了属性间的条件独立性（给定父节点集，每个节点与其非后裔节点相互独立）

$$P_B(x_1,x_2,\ldots,x_n) = \prod_{i=1}^{d} P_B(x_i|\pi_i) = \prod_{i=1}^{d} \theta_{x_i|\pi_i} \tag{7.7}$$

- 常见的网络结构有
  - 同父结构
  - V 型结构
  - 顺序结构

- V 型结构又称冲撞结构，具有边际条件独立性。
  - 当 x_1 同时依赖于 x_2, x_3 时，若 x_1 值固定，则 x_2 与 x_3 不相互独立，若 x_1 值不固定，则 x_2 与 x_3 相互独立
  - 证明过程

$$P(x_2, x_3) = \sum_{x_1} P(x_2, x_3, x_1) = \sum_{x_1} P(x_1|x_2, x_3)P(x_2)P(x_3) = P(x_2)P(x_3) \tag{7.8}$$

- 条件独立性判别方式：
  1. 生成道德图
     (a) 将所有 V 型结构父节点用无向边进行连接
     (b) 将所有有向边变成无向边
  2. 给定属性 x, y 和属性集 $\boldsymbol{z}$，若在属性集中去除 $\boldsymbol{z}$ 属性后，x, y 节点没有通路进行连接，则 x, y 关于属性集 $\boldsymbol{z}$ 条件独立。

7.5.2 学习

- 学习过程包括学习贝叶斯网络结构 B 和网络参数 Θ，在 B 确定的情况下，Θ 可以通过计算样本频率进行确定。所以首要目的是学习网络结构 B。
- 评分搜索是最常用的方法。通过评分函数 (core function) 来评价网络结构和数据集的契合程度，并且其中包含着归纳偏好。
- 学习准则最小描述长度 (Minimal Description Length)。找到一个最短编码长度来对模型和数据集中的数据分布来进行描述。其中包含了描述该模型自身所需要的字节长度和使用该模型描述数据所需要的字节长度。即

$$s(B|D) = f(\theta)|B| - LL(B|D) \tag{7.9}$$

，其中，$f(\theta)$ 是描述每个属性所用的字节长度，|B| 是网络结构中包含的属性数目，

$$LL(B|D) = \sum_{i=1}^{m} \log P_B(x_i) \tag{7.10}$$

  - $f(\theta) = 1$ 时，为 Akaike Information Criterion 评分函数
  - $f(\theta) = \frac{1}{2}\log m$ 时，为 Bayesian Information Criterion 评分函数
  - $f(\theta) = 0$ 时，退化为极大似然估计

- 从所有可能的网络结构中进行遍历搜索得到最优网络结构是一个 NP 难问题，一般通过启发式搜索得到次优网络结构
  - 贪心法从某个网络结构出发，每次调整一条边，直到评分函数稳定为止
  - 通过给网络结构施加约束来削减搜索空间

7.5.3 推断

- 在得出网络结构之后，最精确的做法是求出联合概率分布，进而得出后验概率。但是在网络结构比较复杂的情况下，计算量非常大，不容易实现。
- 这种情况下就要使用近似推断在有限时间内取得近似解。
- 吉布斯采样 (Gibbs sampling) 原理：
  - 假设 $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_n\}$ 是待查询变量，$\mathbf{E} = \{E_1, E_2, \ldots, E_k\}$ 是证据变量，目标是计算后验概率 $P(\mathbf{Q} = \boldsymbol{q}|\mathbf{E} = \boldsymbol{e})$。
  - 随机产生一个符合 $\mathbf{E} = \boldsymbol{e}$ 的样本 $\boldsymbol{q}_0$ 作为初始起点
  - 对非证据变量 (待查询变量) 进行逐个采样改变其取值，采样概率根据贝叶斯网和其他变量的当前取值计算获得[1]。
  - 假定经过 T 次采样得到的与 $\boldsymbol{q}$ 一致的样本有 n_q 个，则可近似估算出后验概率

$$P(\mathbf{Q} = \boldsymbol{q}|\mathbf{E} = \boldsymbol{e}) \simeq \frac{n_q}{T} \tag{7.11}$$

- 吉布斯采样实质上是在 $\mathbf{E} = \boldsymbol{e}$ 的子空间上进行的随机漫步，每一步依赖于前一步的状态，所以这是一个马尔科夫链

[1]从当前样本出发产生下一个样本, 见图 7.5

- 需要较长时间才能收敛
- 如果网络中存在有 0 或 1 这样的极端概率，则可能收敛于一个错误的状态

7.6 EM 算法

- 在某些变量未被观测到的情况下，直接做最大似然估计，则需要做边际最大似然

$$L(\boldsymbol{\Theta}|\mathbf{X}) = \ln P(\mathbf{X}|\boldsymbol{\Theta}) = \ln \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z}|\boldsymbol{\Theta}) \tag{7.12}$$

- EM 算法原型
  - E 步求出关于 $P(\mathbf{Z}|\mathbf{X}, \boldsymbol{\Theta}^t)$ 的分布，并计算对数似然关于 $\boldsymbol{Z}$ 的期望
  - M 步寻找最大化期望似然
- 隐变量估计也可以通过梯度下降法求解，但是求和项数将会随着隐变量的数目以指数级上升
- EM 算法是一种非梯度求解方法

第 8 章　集成学习

8.1　个体与集成

集成学习通过构建多个分类器并集成多个分类器来完成学习任务。具体步骤为:

1. 通过学习产生多个分类器;

2. 通过某些策略将这些分类器进行结合。

需要注意的是，如果分类器种类相同，则集成是同质的 homogeneous，学习器称为基学习器，相应的学习算法称为基学习算法；如果分类器种类不相同，则集成是异质的 heterogenous，学习器称为组件学习器。

集成学习的目的是通过组合多个分类器来获得更好的泛化性能。但是从常识上来看，好的东西和坏的东西掺杂起来通常会获得不好不坏的东西。集成学习为什么能通过集成来获得泛化能力更强的分类器呢? 原因在于，在理想情况下，学习器器选取的准则为多样性和准确性（具体可见图 8.2 的举例)。用公式描述为: 对二分类问题，$y \in \{-1, +1\}$，真实函数 f，假定分类器的错误率为 ϵ，即对每个分类器有

$$P\Big(h(\mathbf{x}) \neq f(\mathbf{x})\Big) = \epsilon \tag{8.1}$$

，若有超过半数的分类器分类正确，则集成分类就正确

$$H(\mathbf{x}) = sign\Big(\sum_{i=1}^{T} h_i(\mathbf{x})\Big) \tag{8.2}$$

，假设分类器的错误率相互独立，则由 Hoeffding 不等式可知，集成的错误

率为

$$
\begin{aligned}
P\Big(H(\mathbf{x}) \neq f(\mathbf{x})\Big) &= \sum_{k=0}^{\lfloor T/2 \rfloor} C_T^k (1-\epsilon)^k \epsilon^{T-k} \\
&\leq \exp\Big(-\frac{1}{2}T(1-2\epsilon)^2\Big)
\end{aligned}
\tag{8.3}
$$

。当 T 不断增大时，错误率指数级减小。

但是需要指出的一个重要前提条件是，假设分类器的错误率相互独立。这个条件在实际中很难成立，所以学习器的 “准确性” 和 “多样性” 本身就存在冲突。如何产生并结合好而不同的学习器是集成学习的一个核心研究内容。

一般情况下，集成学习分为可以序列化生成学习器的 Boosting 方法和并行生成学习器的 Bagging 方法、随机森林 Random Forest 方法。

8.2 Boosting

本节的 Boosting 方法以 AdaBoosting 为例，虽然之前阅读过 AdaBoosting 的相关文献，但是本节的内容还是花费了一个晚上的时间去梳理。从整体上来看，AdaBoosting 每一轮迭代是以上一轮迭代的结果为基础，重点针对上一轮迭代得到的学习器的错误分类样本进行训练。这就和之前提到的学习器的选取准则准确性和多样性形成了对应关系。图 8.3 是 AdaBoosting 的算法，容易产生疑问的是第 6 行的学习器权重更新公式和第 7 行的样本分布更新公式。本节重点说明并进行证明的也正是这两处。在证明上述两个公式之前，需要对 AdaBoosting 所使用的指数损失函数做一个说明，因为指数损失函数是之后证明过程的基础。

$$
\mathcal{L}_{exp} = E_{\mathbf{x}\sim\mathcal{D}}\Big(e^{-f(\mathbf{x})H(\mathbf{x})}\Big) \tag{8.4}
$$

对 $H(\mathbf{x})$ 偏导，得

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{exp}}{\partial H(\mathbf{x})} &= e^{-f(\mathbf{x})H(\mathbf{x})}(-f(\mathbf{x})) \\
&= -e^{-H(\mathbf{x})}P(f(\mathbf{x}=1|\mathbf{x})) + e^{H(\mathbf{x})}P(f(\mathbf{x})=-1|\mathbf{x})
\end{aligned}
\tag{8.5}
$$

，令上式为 0，得

$$H(\mathbf{x}) = \frac{1}{2}\ln\frac{P(f(\mathbf{x})=1|\mathbf{x})}{P(f(\mathbf{x})=-1|\mathbf{x})} \tag{8.6}$$

。即有

$$sign(H(\mathbf{x})) = \arg\min_{y\in\{-1,1\}} P(f(\mathbf{x})=y|\mathbf{x}) \tag{8.7}$$

，$sign(H(\mathbf{x}))$ 达到了贝叶斯最优错误率，可以用来替代 0/1 损失函数，并且具有连续、可导的优点。

学习器权重更新公式证明

假设在第 t 轮训练中得到的学习器为 $h_t(\mathbf{x})$，则当前的分类器权重 α_t 应使

$$\mathcal{L}_{exp}(\alpha_t h_t(\mathbf{x})|\mathcal{D}_{\sqcup}) = E_{\mathbf{x}\sim\mathcal{D}_{\sqcup}}[e^{-f(\mathbf{x})h_t(\mathbf{x})\alpha_t}] \tag{8.8}$$

最小。将上式对 α_t 求偏导得

$$\frac{\partial\mathcal{L}_{exp}(\alpha_t h_t(\mathbf{x})|\mathcal{D}_{\sqcup})}{\partial\alpha_t} = -e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t \tag{8.9}$$

。令上式为 0，得

$$\alpha_t = \ln\frac{1-\epsilon_t}{\epsilon_t} \tag{8.10}$$

，即为第 6 行中的学习器权重公式。

样本分布更新公式证明

在获取 H_{t-1} 之后，AdaBoosting 对样本分布进行一些调整，使得在第 t 轮训练中得到的学习器能够针对 $t-1$ 轮训练中的错分样本进行正确分类。即最小化

$$\mathcal{L}_{exp}(H_{t-1}+h_t|\mathcal{D}) = E_{\mathbf{x}\sim\mathcal{D}}\left[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}e^{-f(\mathbf{x})h_t(\mathbf{x})}\right] \tag{8.11}$$

。由于有 $f(\mathbf{x})^2 = h_t(\mathbf{x})^2 = 1$，则上式泰勒展开式近似为

$$\mathcal{L}_{exp}(H_{t-1}+h_t|\mathcal{D}) \simeq E_{\mathbf{x}\sim\mathcal{D}}\left[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}\left(1-f(\mathbf{x})h_t(\mathbf{x})+\frac{1}{2}\right)\right] \tag{8.12}$$

。理想的学习器

$$\begin{aligned} h_t(\mathbf{x}) &= \arg\max_h E_{\mathbf{x}\sim\mathcal{D}}\left[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}f(\mathbf{x})h(\mathbf{x})\right] \\ &= \arg\max_h E_{\mathbf{x}\sim\mathcal{D}}\left[\frac{e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}}{E_{\mathbf{x}\sim\mathcal{D}}[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}]}f(\mathbf{x})h(\mathbf{x})\right] \end{aligned} \tag{8.13}$$

将在分布 $\mathcal{D}_t$ 下最小化分类误差。令 $\mathcal{D}_t$ 表示一个分布，

$$\begin{aligned} \mathcal{D}_{t+1}(\mathbf{x}) &= \frac{\mathcal{D}(\mathbf{x})e^{-f(\mathbf{x})H_t(\mathbf{x})}}{E_{\mathbf{x}\sim\mathcal{D}}[e^{-f(\mathbf{x})H_t(\mathbf{x})}]} \\ &= \mathcal{D}_t e^{e^{-f(\mathbf{x})\alpha_t h_t(\mathbf{x})}}\frac{E_{\mathbf{x}\sim\mathcal{D}}[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}]}{E_{\mathbf{x}\sim\mathcal{D}}[e^{-f(\mathbf{x})H_t(\mathbf{x})}]} \end{aligned} \tag{8.14}$$

，第 7 行样本分布更新公式得证。

8.3 Bagging 与随机森林

8.3.1 Bagging

Bagging 是并行集成学习方法最著名的代表。Bagging 方法首先采样出 T 个含有 m 个样本的样本子集，然后在每个样本子集上分别训练出一个个体学习器，然后将这些学习器进行结合。如果采用自助采样法获得 m 个样本子集，则大约有 63.2% 的样本包含在样本子集中，剩余的 36.8% 的样本可以用来对基学习器进行包外估计 (out-of-bag estimation)。从偏差-方差分解的角度来看，Bagging 重点关注降低方差，因此它不在易受样本扰动的决策树、神经网络等基学习器上效果更加明显。

8.3.2 随机森林

随机森林在 Bagging 基础上引入了随机属性选择。但是与 Bagging 中基学习器“多样性”仅仅依靠样本扰动不同，随机森林的“多样性”还来自于属性扰动。这就使得最终集成的泛化性能通过个体学习器之间的差异性的增加而进一步提升。

但是同时需要注意的是，随机森林的起始性能较差，随着个体学习器的数量增加，随机森林会收敛到更低的泛化误差。

8.4 结合策略

8.4.1 平均法

平均法分为简单平均法和加权平均法。其中，加权平均法的权重是从训练数据中学习而得，现实中由于数据存在误差和噪声，这将使得学出的权重不完全可靠。一般而言，在个体学习器性能差异较大时使用加权平均法，而在个体学习器性能相近时使用简单平均法。

8.4.2 投票法

投票发分为绝对多数投票法、相对多数投票法和加权投票法。

在一些能够同时输出分类标签和分类置信度的学习器，将置信度转化为类概率值虽然不太准确，但是能够取得比直接使用分类标签更好的结合性能。

8.4.3 学习法

学习法是指使用学习方法得到各个学习器的权重并进行结合。

Stacking 方法为了避免过拟合，通常使用交叉验证法或者留一法，用训练初级学习器未使用到的数据来训练次级学习器。次级学习器的输入属性和次级学习器算法对 Stacking 集成的泛化性能有很大影响。有研究表明，将初级学习器的输出类概率作为次级学习器的输入属性，用多响应线性回归 (Multi-response Liner Regression, MLR) 作为次级学习算法效果较好，在 MLR 中使用不同的属性集更佳。

贝叶斯模型平均基于后验概率来为不同模型赋予权重，可视为加权平均法的一种特殊实现。

8.5 多样性

8.5.1 误差-分歧分解

8.2 节中提到过，个体学习器应该 “好而不同”，本节主要针对这一点做一个简要的分析.

对于示例 $\mathbf{x}$，定义学习器 h_i 的 “分歧”(ambuiguity) 为

$$A(h_i|\mathbf{x}) = (h_i(\mathbf{x}) - H(\mathbf{x}))^2 \tag{8.15}$$

则集成的分歧为

$$A(H(\mathbf{x})|\mathbf{x}) = \sum_{i=1}^{T} w_i (h_i(\mathbf{x}) - H(\mathbf{x}))^2 \tag{8.16}$$

上述两项分歧主要定义了个体学习器在样本 $\mathbf{x}$ 上的不一致性.

个体学习器 h_i 和集成学习器 H 在 $\mathbf{x}$ 上的误差定义分别为

$$E(h_i(\mathbf{x})|\mathbf{x}) = (h_i(\mathbf{x}) - f(\mathbf{x}))^2 \tag{8.17}$$

$$E(H(\mathbf{x})|\mathbf{x}) = (H(\mathbf{x}) - f(\mathbf{x}))^2 \tag{8.18}$$

则有

$$\bar{A}(h|\mathbf{x}) = \sum_{i=1}^{T} w_i E(h_i|\mathbf{x}) - E(H|\mathbf{x}) \tag{8.19}$$

将上式推广到全样本控件, 设样本概率分布为 $p(\mathbf{x})$, 则有

$$\sum_{i=1}^{T} w_i \int A(h_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^{T} w_i \int E(h_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} - \int E(H|x) p(\mathbf{x}) d\mathbf{x} \tag{8.20}$$

令个体误差和分歧项为

$$E_i = \int E(h_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{8.21}$$

$$A_i = \int A(h_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{8.22}$$

令集成的泛化误差为

$$E = \int E(H|\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{8.23}$$

再令 $\bar{E}$ 和 $\bar{A}$ 分别为个体学习器的加权误差值和加权分歧值, 则有

$$E = \bar{E} - \bar{A} \tag{8.24}$$

从上式可以看出, 个体学习器准确性越高 ($\bar{E}$ 越大), 多样性越大 ($\bar{A}$ 越大), 则集成越好.

但是在现实中很难针对 $\bar{E} - \bar{A}$ 进行优化, 不仅因为其是在全样本空间上的定义, 还由于 $\bar{A}$ 是一个不可直接操作的准确性度量.

另外需要注意的是, 本节中的推导过程是针对回归问题的, 不能推广到分类问题上去.

8.5.2 多样性度量

多样性度量是用于度量集成中个体分类器的多样性, 通常考虑分类器间两两不相似性. 常见度量准则有 (公式略)

- 不合度量 (disagreement measure)
- 相关系数 (correlation coefficient)
- Q-统计量 (Q-statistic)
- κ-统计量 (κ-statistic)

由于上述统计量都是成对学习器的度量值, 可以通过二位坐标平面进行展示. 其中 x 轴是这对分类器的多样性度量值, 纵坐标轴是他们的平均误差. 通常情况下, 点云位置越高, 个体学习器的准确性越低, 点云位置越靠右, 个体学习器多样性越小.

8.5.3 多样性增强

为了增强多样性, 一般思路是在学习过程中增加扰动, 即进入随机性. 常见做法有对数据样本, 输入属性, 输出表示, 算法参数进行扰动.

数据样本扰动

给定初始训练样本集, 从中产生不同的数据子集, 在子集上对学习器进行训练. 需要注意的是, 有些类型的学习器比较容易受到训练样本的扰动, 如决策树, 神经网络等; 有些类型的学习器不容易受到训练样本的扰动, 如线性学习器, 支持向量机, 朴素贝叶斯,k 近邻学习器等, 这样的学习器称为稳定基学习器 (stable base learner). 对于稳定基学习器往往需要使用其他类型的扰动.

输入属性扰动

训练样本通常使用一组属性进行描述, 属性扰动是从这一组属性中随机采样出属性子集对学习器进行训练. 如果属性类别比较多, 则可以再一定程度上降低运算复杂度; 如果属性类别较少, 则不宜使用输入属性扰动.

输出表示扰动

对输出表示进行操纵以增强多样性. 举例有翻转法, 输出调制法,ECOC 法.

算法参数扰动

基学习算法一般都有参数需要进行设置, 通过随机设置不同的参数 (如神经网络隐神经元数目), 往往可以产生差异较大的个体学习器.

第 9 章　聚类

9.1　聚类任务

无监督学习的目标是通过学习未标记样本来发现样本间的内在规律, 为进一步的数据分析提供基础. 聚类是无监督学习中研究最多, 应用最广的任务. 聚类既能作为一个单独过程, 研究数据内在规律, 也可为后续的分析过程提供数据支撑.

9.2　性能度量

与有监督学习过程一样, 聚类也需要一个性能度量指标来度量其学习效果的好坏. 从直观上看, 一个好的聚类模型应该使属性相似的样本归到一簇中去, 并且簇与簇之间的差别应该尽可能大, 即簇内相似度高且簇间相似度低. 通常度量聚类性能的方法有两类, 一类是与某个参考模型作对比, 称为外部指标, 另一类是直接考察聚类结果而不与任何参考模型作对比, 称为内部指标.

外部指标

首先定义数据集 $D = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}\}$, 聚类给出的簇划分为 $C = \{C_1, C_2, \ldots, C_k\}$, 参考模型给出的簇划分为 $C^* = \{C_1^*, C_2^*, \ldots, C_s^*\}$. 令 λ 与 λ^* 分别对应聚类与参考模型的标记向量. 定义

$$a = |SS|, SS = \{(\mathbf{x_i}, \mathbf{x_j}) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\} \tag{9.1}$$

$$b = |SD|, SD = \{(\mathbf{x_i}, \mathbf{x_j}) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\} \tag{9.2}$$

$$c = |DS|, DS = \{(\mathbf{x_i}, \mathbf{x_j})|\lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\} \tag{9.3}$$

$$d = |DD|, DD = \{(\mathbf{x_i}, \mathbf{x_j})|\lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\} \tag{9.4}$$

由于 $i < j$ 且每个样本 $(\mathbf{x_i}, \mathbf{x_j})$ 仅能出现在一个集合中, 则有

$$a + b + c + d = m(m-1)/2 \tag{9.5}$$

基于以上, 常见外部指标有

- Jaccard 系数, Jaccard Coefficient, 简称 JC

$$JC = \frac{a}{a+b+c} \tag{9.6}$$

- FM 指数, Fowlkes and Mallows Index, 简称 FMI

$$FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}} \tag{9.7}$$

- Rand 指数, Rand Index, 简称 RI

$$RI = \frac{2(a+d)}{m(m-1)} \tag{9.8}$$

以上结果值区间在 $[0, 1]$ 之间, 越大越好

内部指标

考虑聚类结果簇划分 $C = \{C_1, C_2, \ldots, C_k\}$, 定义

$$avg(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \leq i < j \leq |C|} dist(\mathbf{x_i}, \mathbf{x_j}) \tag{9.9}$$

$$diam(C) = \max_{1 \leq i < j \leq |C|} dist(\mathbf{x_i}, \mathbf{x_j}) \tag{9.10}$$

$$d_{min}(C_i, C_j) = \min_{\mathbf{x_i} \in C_i, \mathbf{x_j} \ in C_j} dist(\mathbf{x_i}, \mathbf{x_j}) \tag{9.11}$$

$$d_{cen}(C_i, C_j) = dist(\mu_{\mathbf{i}}, \mu_{\mathbf{j}}) \tag{9.12}$$

$avg(C)$: 簇内样本间平均距离 $diam(C)$: 簇内样本间最远距离 $d_{min}(C_i, C_j)$: 簇 C_i 与簇 C_j 间最近样本距离 $d_{cen}(C_i, C_j)$: 簇 C_i 与簇 C_j 中心点间距离根据以上定义, 常见内部指标有

- DB 指数, Davis-Bouldin Inex, DBI

$$DBI = \frac{1}{k}\max_{j\neq i}\left(\frac{avg(C_i) + avg(C_j)}{d_{cen}(\mu_i, \mu_j)}\right) \tag{9.13}$$

- Dunn 指数, Dunn Index, DI

$$DI = \min_{1\leq i\leq k}\left\{\min_{j\neq i}\left(\frac{d_{min}(C_i, C_j)}{\max_{1\leq l\leq k} diam(C_l)}\right)\right\} \tag{9.14}$$

DBI 越小越好,DI 越大越好

9.3 距离计算

函数 $dist(\cdot, \cdot)$ 是一个距离度量, 需要满足下述属性:

- 非负性: $dist(\mathbf{x_i}, \mathbf{x_j}) \geq 0$
- 同一性: $dist(\mathbf{x_i}, \mathbf{x_j}) = 0$ 当且仅当 $\mathbf{x_i} = \mathbf{x_j}$
- 对称性: $dist(\mathbf{x_i}, \mathbf{x_j}) = dist(\mathbf{x_j}, \mathbf{x_i})$
- 直递性: $dist(\mathbf{x_i}, \mathbf{x_j}) \leq dist(\mathbf{x_i}, \mathbf{x_k}) + dist(\mathbf{x_k}, \mathbf{x_j})$

通常我们将属性划分为连续属性与离散属性, 但是在距离计算时, 序的关系更为重要. 区分有序属性与无序属性的标准是能否直接在属性值上计算距离.

有序属性

给定样本 $\mathbf{x_i} = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ 与 $\mathbf{x_j} = \{x_{j1}, x_{j2}, \ldots, x_{jn}\}$, 常见距离定义有:

- 闵可夫斯基距离, Minkowski Distance

$$dist_{mk} = \Big(\sum_{u=1}^{n} |x_{iu} - x_{ju}|^p \Big)^{\frac{1}{p}} \tag{9.15}$$

对于 $p = 1$ 和 $p = 2$, 分别有特例

- 欧氏距离, Euclidean Distance

$$dist_{ed}(\mathbf{x_i}, \mathbf{x_j}) = \|(x_i - x_j)\|_2 = \sqrt{\sum_{u=1}^{n} |x_{iu} - x_{ju}|^2} \tag{9.16}$$

- 曼哈顿距离, Manhattan Distance

$$dist_{man}(\mathbf{x_i}, \mathbf{x_j}) = \|(x_i - x_j)\|_1 = \sum_{u=1}^{n} |x_{iu} - x_{ju}| \tag{9.17}$$

- 如果不同属性重要程度不同, 有

$$dist_{wmk} = \Big(w_1 |x_{i1} - x_{j1}|^p + \cdots + w_n |x_{in} - x_{jn}|^p \Big)^{\frac{1}{p}} \tag{9.18}$$

无序属性

- VDM, Value Difference Metric
  令 $m_{u,a}$ 表示属性 u 上取值为 a 的样本数,$m_{u,a,i}$ 表示在第 i 个样本簇中, 在第 u 个属性上取值为 a 的样本数, 则在属性 u 上两个离散值 a 与 b 的 VDM 值为

$$VDM_p(a, b) = \sum_{i=1}^{k} \Big| \frac{m_{u,a,i}}{m_{u,a}} - \frac{m_{u,b,i}}{m_{u,b}} \Big|^p \tag{9.19}$$

混合属性

以闵可夫斯基距离和 VDM 距离结合为例

$$MinkovVDM_p(\mathbf{x_i}, \mathbf{x_j}) = \left(\sum_{u=1}^{n_c} |x_{iu} - x_{ju}|^p + \sum_{u=n_c+1}^{n} VDM_p(x_{iu}, x_{ju}) \right)^{\frac{1}{p}} \quad (9.20)$$

在现实任务中, 有必要基于数据样本来确定合适的距离公式, 这可以通过距离度量学习 (distance metric learning) 来实现.

9.4 原型聚类

此类聚类假设聚类结构能够通过一组原型来刻画.

9.4.1 k 均值算法

k 均值算法针对聚类所得簇最小化平均误差作为优化目标

$$E = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mu_\mathbf{i}\|_2^2 \quad (9.21)$$

其中,μ_i 是 C_i 簇的均值向量. 上式是一个 NP 难的问题, 常采用贪心算法求解出次优值.

算法描述:

1. 随机初始化聚类中心
2. 选取每个样本, 计算每个样本与簇中心距离, 归到距离最短簇中
3. 更新簇中心位置
4. 重复以上步骤, 并判断是否达到迭代标准 (次数, 簇中心位移)

9.4.2 学习向量量化

本节学习的内容从最后的距离来看较为简单, 但是如果从本节开头的定义以及之后的推导过程来看则相对复杂难以理解. 因此有必要对本节开头的定义作补充说明:

1. 样本集 $D = \{(\mathbf{x_1}, y_1), \mathbf{x_2}, y_2), \ldots, \mathbf{x_m}, y_m)\}$, 每个样本 $\mathbf{x_i}$ 是由 n 个样本属性 $(x_{i1}, x_{i2}, \ldots, x_{in})$ 描述的特征向量,$y_i \in$ † 是样本 x_i 对应的类别标记.

2. LVQ 的目标是学习一组 n 维原型向量 $\{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_q}\}$, 每个原型向量代表一个聚类簇.

注:

1. 原型向量与 k 均值聚类中的聚类中心概念比较接近

2. 簇标记与类别标记的概念并不等价, 一个类别标记可能对应多个簇标记, 即多个簇可能对应一个类别.

算法描述:

1. 原型向量初始化. 对于第 q 个簇, 可以从类别标记为 t_q 的样本中随机选取一个座位原型向量;

2. 便利样本, 计算每个样本与原型向量距离, 并确定与之距离最近的原型向量 $\mathbf{p_{i^*}}$. 若 $\mathbf{x_j}$ 与 $\mathbf{p_{i^*}}$ 类别相同, 则 $\mathbf{p_{i^*}}$ 向 $\mathbf{x_j}$ 方向靠拢, 否则 $\mathbf{p_{i^*}}$ 远离 $\mathbf{x_j}$ 的方向.

注:$\mathbf{p_{i^*}}$ 向 $\mathbf{x_j}$ 方向靠拢公式推导过程

$$\mathbf{p'} = \mathbf{p_{i^*}} + \eta(\mathbf{x_j} - \mathbf{p_{i^*}}) \tag{9.22}$$

$$\begin{aligned} \|\mathbf{p'} - \mathbf{x_j}\|_2 &= \|\mathbf{p_{i^*}} + \eta(\mathbf{x_j} - \mathbf{p_{i^*}}) - \mathbf{x_j}\|_2 \\ &= (1-\eta)\|\mathbf{p_{i^*}} - \mathbf{x_j}\|_2 \end{aligned} \tag{9.23}$$

9.4.3 高斯混合聚类

高斯混合聚类采用概率分布作为聚类原型, 因此为了能够理解高斯混合聚类, 必须放下之前的比较直观的 k 均值聚类与 LVQ. 高斯混合聚类首先初始化初始高斯分布, 然后计算各个样本在各个高斯混合分布成分上的后验概率, 之后根据这个后验概率更新高斯分布参数. 重复上述过程并确定样本的簇标记. 首先定义

- n 维高斯分布

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)} \tag{9.24}$$

其中 μ:n 维均值向量 Σ:$n \times n$ 维协方差矩阵

- 高斯混合分布

$$p_M(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i p(\mathbf{x}|\mu_i, \Sigma_i) \tag{9.25}$$

算法描述:

1. (E 步) 假设 $D = \{(\mathbf{x_1}, y_1), \mathbf{x_2}, y_2), \ldots, \mathbf{x_m}, y_m)\}$ 由上述过程生成, 令 $z_j \in \{1, 2, \ldots, k\}$ 表示生成样本 x_j 的高斯混合成分, 显然 z_j 的先验概率对应于 α_i, 后验概率对应于

$$\begin{aligned} p_M(z_j = i|\mathbf{x_j}) &= \frac{p(z_j = 1) \cdot p_M(\mathbf{x_j}|z_j = i)}{p_M(\mathbf{x_j})} \\ &= \frac{\alpha_i p(\mathbf{x_i}|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_i p(\mathbf{x_j}|\mu_l, \Sigma_l)} \end{aligned} \tag{9.26}$$

上式给出了 x_j 由第 i 个高斯混合成分生成的后验概率记 $p_M(z_j = i|\mathbf{x_j}) = \gamma_{ji}$ 则簇标记 $\lambda = \arg\max_{i \in \{1,2,\ldots,k\}} \gamma_{ji}$

2. (M 步) 模型参数如何更新? 采用极大化似然估计.

$$LL(D) = \ln\Big(\prod_{j=1}^{m} p_M(\mathbf{x_j})\Big) = \sum_{j=1}^{m} \ln\Big(\sum_{i=1}^{n} \alpha_i p(\mathbf{x_j}|\mu_i, \Sigma_i)\Big) \tag{9.27}$$

由 $\frac{\partial LL(D)}{\partial \mu_i} = 0$ 有,$\mu_i = \frac{\sum_{j=1}^{m} \gamma_{ji}\mathbf{x_j}}{\sum_{j=1}^{m} \gamma_{ji}}$. 由 $\frac{\partial LL(D)}{\Sigma_i} = 0$ 有,$\Sigma_i = \frac{\sum_{j=1}^{m} \gamma_{ji}(\mathbf{x_j}-\mu_i)(\mathbf{x_j}-\mu_i)^T}{\sum_{j=1}^{m} \gamma_{ji}}$
对于 α_i, 除了最大化 $LL(D)$, 还要满足 $\alpha_i \geq 0, \sum_{i=1}^{k} \alpha_i = 1$ 对 $LL(D) + \lambda(\sum_{i=1}^{k} \alpha_i - 1)$ 求 α_i 求偏导, 有

$$\sum_{j=1}^{m} \frac{p(\mathbf{x_j}|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l p(\mathbf{x_j}|\mu_l, \Sigma_l)} + \lambda = 0 \tag{9.28}$$

得 $\alpha_i = \frac{1}{m}\sum_{j=1}^{m} \gamma_{ji}$.

9.5 密度聚类 (DBSCAN)

顾名思义, 密度聚类是指根据样本分布的密度来对样本进行聚类. 但是, 如何定义样本的稠密程度, 如何根据样本分布稠密程度对样本簇进行划分是密度聚类要明确的定义. 基于此, 需要明确以下定义:

- ϵ-邻域对 $\mathbf{x_j} \in D$, 其 ϵ-邻域包含样本集 D 中与 $\mathbf{x_j}$ 的距离不大于 ϵ 的样本, 即 $N_\epsilon(\mathbf{x_j}) = \{\mathbf{x_j} \in D | dist(\mathbf{x_i}, \mathbf{x_j}) \leq \epsilon\}$
- 核心对象 (core object) 若 $\mathbf{x_j}$ 的 ϵ 邻域至少包含 $MinPts$ 个样本, 即 $|N_\epsilon(\mathbf{x_j})| \geq MinPts$, 则 $\mathbf{x_j}$ 是一个核心对象.
- 密度直达 (directly density-reachable) 若 $\mathbf{x_j}$ 位于 $\mathbf{x_i}$ 的 ϵ-邻域中, 且 $\mathbf{x_i}$ 是核心对象, 则称 $\mathbf{x_j}$ 由 $\mathbf{x_i}$ 密度直达.
- 密度可达 (density-reachable) 对 $\mathbf{x_i}$ 与 $\mathbf{x_j}$, 若存在序列样本 $\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}$, 其中 $\mathbf{p_1} = \mathbf{x_1}, \mathbf{p_n} = \mathbf{x_j}$, 且 $\mathbf{p_{i+1}}$ 由 $\mathbf{p_i}$ 密度直达, 则称 $\mathbf{x_j}$ 由 $\mathbf{x_i}$ 密度可达.
- 密度相连 (density-connected) 对 $\mathbf{x_i}$ 与 $\mathbf{x_j}$, 若存在 $\mathbf{x_k}$ 使得 $\mathbf{x_i}$ 与 $\mathbf{x_j}$ 均可由 $\mathbf{x_k}$ 密度可达, 则称 $\mathbf{x_i}$ 与 $\mathbf{x_j}$ 密度相连.

基于以上定义, 簇的定义为, 由密度可达关系导出的最大的样本相连密度集合. 形式化地说, 给定邻域参数 $(\epsilon, MinPts)$, 簇 $C \subseteq D$ 是满足以下性质的非空样本子集:

- 连接性:$\mathbf{x_i} \in C, \mathbf{x_j} \in C \Rightarrow \mathbf{x_i}$ 与 $\mathbf{x_j}$ 密度相连
- 最大性:$\mathbf{x_i} \in C, \mathbf{x_j}$ 由 $\mathbf{x_i}$ 密度可达 $\Rightarrow \mathbf{x_j} \in C$

算法描述

1. 找出数据集中所有的核心对象
2. 以任意核心对象为出发点, 找出由其密度可达的样本生成簇
3. 遍历所有的核心对象, 直至所有核心对象均被访问过为止

9.6 层次聚类 (AGNES)

层次聚类试图在不同层次对数据进行划分, 从而形成树形的聚类结构. 数据集的划分可采用 “自底向上” 的聚合策略, 也可采用 “自顶向下” 的分拆策略. AGNES 算法首先将每个样本看做一个初始聚类簇, 然后在算法进行的每一步中找出距离最近的两个聚类簇进行合并, 该过程不断重复, 直至达到预设的聚类簇个数.

聚类簇距离定义:

- 最小距离 $d_{min}(C_i, C_j) = \min_{\mathbf{x}\in C_i, \mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$
- 最大距离 $d_{max}(C_i, C_j) = \max_{\mathbf{x}\in C_i, \mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$
- 平均距离 $d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x}\in C_i} \sum_{\mathbf{z}\in C_j} dist(\mathbf{x}, \mathbf{z})$

使用不同的聚类簇距离定义, 算法的名称也不相同:

- d_{min} 单链接 single-linkage
- d_{max} 全链接 complete-linkage
- d_{avg} 均链接 average-linkage

第 10 章　降维与度量学习

10.1　k 近邻学习

本章首先介绍了 k 近邻学习, 但是并未介绍 k 近邻学习与降维或度量学习之间的关系. 从逻辑上来说,k 近邻学习只是为了从其缺点入手引入降维的必要性.

工作机制: 给定测试样本, 基于某种距离度量找出训练集中与其最近的 k 个样本, 然后基于这 k 个 “邻居” 的信息来进行预测.k 近邻学习是 “懒惰学习”(lazy learning) 的代表, 训练过程仅仅是将训练样本进行存储, 在收到测试样本后再进行处理; 与之相对应的是 “急切学习”, 其在训练阶段就对样本进行处理.

给定测试样本 $\mathbf{x}$, 若其最近邻样本为 $\mathbf{z}$, 则最近邻分类器出错的概率就是 $\mathbf{x}$ 与 $\mathbf{z}$ 类别标记不同的概率.

$$
\begin{aligned}
P(err) &= 1 - \sum_{C \in \mathcal{Y}} P(c|\mathbf{x}) P(c|\mathbf{z}) \\
&\simeq 1 - \sum_{C \in \mathcal{Y}} P^2(c|\mathbf{x}) \\
&\leq 1 - P^2(c^*|\mathbf{x}) \\
&= (1 + P(c^*|\mathbf{x}))(1 - P(c^*|\mathbf{x})) \\
&\leq 2 \cdot (1 - P(c^*|\mathbf{x}))
\end{aligned}
\tag{10.1}
$$

其中,$c^* = \arg\max_{c \in \mathcal{Y}} P(c|\mathbf{x})$ 表示贝叶斯最优分类器分类结果.

10.2 低维嵌入

本节首先明确了上一节中的一个重要前提, 即 k 近邻成立的前提是训练样本集是经过 “密采样”(dense sample) 得到的. 但是在实际中 “密采样” 会产生计算方面的问题. 因此降维的必要性就凸显出来.

上一节的重要假设: 任意测试样本 $\mathbf{x}$ 附近任意小的 δ 距离范围内总能找到一个训练样本, 即训练样本的采样密度足够大.

在高维情况下出现的数据样本稀疏, 距离计算等问题, 是所有机器学习方法面临的严重障碍, 被称为 “维数灾难”(curse of dimensionality).

若在降维的过程中要求原始样本空间中的距离在低维空间中得以保存, 就得到了 “多维缩放”(Multiple Dimensional Scaling) 降维方法.

定义: $\mathbf{D} \in \mathbb{R}^{m\times m}$:$m$ 个样本原始空间距离矩阵; $\mathbf{Z} \in \mathbb{R}^{d'\times m}$: 降维后中心化的样本; $\mathbf{B} = \mathbf{Z^T Z} \in \mathbb{R}^{m\times m}$: 降维后的内积矩阵.

样本 i 与样本 j 之间距离的平方 (降维前后相等)

$$\begin{aligned} dist_{ij}^2 &= \|\mathbf{z_i}\|^2 + \|\mathbf{z_j}\|^2 - 2\mathbf{z_i^T z_j} \\ &= b_{ii} + b_{jj} - 2b_{ij} \end{aligned} \tag{10.2}$$

经过代换得 $b_{ij} = -\frac{1}{2}(dist_{ij}^2 - dist_{i\cdot}^2 - dist_{\cdot j}^2 + dist_{\cdot\cdot}^2)$, 由此求得 $\mathbf{B}$, 对 $\mathbf{B}$ 做特征值分解 $\mathbf{B} = \mathbf{V\Lambda V^T}$, 选取出其中的 d^* 个非零特征值有

$$\mathbf{Z} = \mathbf{\Lambda_*^{\frac{1}{2}} V_*^T} \in \mathbb{R}^{d^*\times m} \tag{10.3}$$

其中,$\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_d)$ 是特征对角矩阵;$\mathbf{V}$ 是特征向量矩阵.

一般来说, 欲获得低维子空间, 最简单的是对原始高维空间进行线性变换, 又称线性降维. 它们都符合 $\mathbf{Z} = \mathbf{W^T X}$ 的形式, 不同之处是对低维子空间的性质有不同要求, 相当于对变换矩阵 $\mathbf{W}$ 加上了不同约束.

评估方式通常是比较降维前后学习器性能, 性能提升则认为降维起了作用.

10.3 主成分分析

主成分分析 (Principal Component Analysis) 是最常用的一种降维方法.

正交空间降维到超平面应该满足:

- 最近重构性: 样本点到这个超平面的距离足够近;
- 最大可分性: 样本点在这个超平面上的投影能尽可能分开.

定义: 投影变换后, 新坐标系为 $\{\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_d}\}$, 其中 $\mathbf{w_i}$ 是标准正交基向量; 降维的目的是丢弃部分基向量, 降低维度至 $d' < d$; $\mathbf{z_i}$ 是 $\mathbf{x_i}$ 在低维空间中投影.

根据最近重构性, 有

$$\begin{aligned}\sum_{i=1}^{m} \| \sum_{j=1}^{d'} z_{ij}\mathbf{w}_j - \mathbf{x}_i\|_2^2 &= \sum_{i=1}^{m} \mathbf{z}_i^T\mathbf{z}_i - 2\sum_{i=1}^{m} \mathbf{z}_i^T\mathbf{W}^T\mathbf{x}_i + const \\ &\propto -tr\Big(\mathbf{W}^T(\sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T)\mathbf{W}\Big)\end{aligned} \tag{10.4}$$

将上式最小化, 并考虑到 $\mathbf{w}_j$ 是标准正交基,$\sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T$ 是协方差矩阵, 有

$$\begin{aligned}&\min_{\mathbf{W}} -tr\Big(\mathbf{W^T X X^T W}\Big) \\ &s.t. \mathbf{W^T W} = \mathbf{I}\end{aligned} \tag{10.5}$$

根据最大可分性,$\mathbf{x}_i$ 在超平面上投影 $\mathbf{W}^T\mathbf{x}_i$, 投影后样本点方差为 $\sum_i \mathbf{W}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{W}$, 于是有

$$\begin{aligned}&\max_{\mathbf{W}} tr(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}) \\ &s.t. \mathbf{W}^T\mathbf{W} = \mathbf{I}\end{aligned} \tag{10.6}$$

由此可见, 最近重构性与最大可分性二者等价.

PCA 舍弃 $d - d'$ 个特征值是降维结果. 一方面, 采样密度增加; 另一方面, 最小特征值往往与噪声有关, 舍弃他们能在一定程度上起到去噪的效果.

10.4 核化线性降维

前一节中提到的降维方式是线性降维, 即假设从高维空间到低维空间的函数映射是线性的, 然而这与实际情况的差别很大, 现实中有不少情况需要非线性降维, 这样可以保持原采样空间中的低维结构.

注:

- 对样本空间进行采样会造成维度增加;
- 原本采样的低维空间称为 “本真”(intrinsic) 低维空间.

非线性降维的一种常用方法, 是基于核技巧对线性降维方法进行 “核化”(kernelized).

通常采用的 PCA 方法对数据降维的原理是

$$\Big(\sum_{i=1}^{m}\mathbf{z_i}\mathbf{z_i}^T\Big)\mathbf{W}=\lambda\mathbf{W} \tag{10.7}$$

其中 $\mathbf{W}$ 是数据降维后投影的超平面;$\mathbf{z_i}$ 是样本点 $\mathbf{x}_i$ 在高维特征空间中通过映射 ϕ 产生的. 因此有

$$\Big(\sum_{i=1}^{m}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T\Big)\mathbf{W}=\lambda\mathbf{W} \tag{10.8}$$

但是由于 $\mathbf{x}_i$ 在一般情况下是未知的, 所以 ϕ 的具体形式也并不清晰. 由此, 引入核函数 $\kappa(\mathbf{x}_i,\mathbf{x}_j)=\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$, 代入上式有

$$\mathbf{KA}=\lambda\mathbf{A} \tag{10.9}$$

其中,$\mathbf{K}$ 为 κ 对应的核矩阵;$\mathbf{A}=(\alpha_1,\alpha_2,\ldots,\alpha_m)$.

为获得投影后的坐标 KPCA 需对所有样本求和, 因此它的计算开销较大.

10.5 流形学习

流形是在局部与欧式空间同胚的空间, 它在局部具有欧式空间的性质, 能用欧氏距离来进行计算.

10.5.1 等度量映射 (Isometric Mapping)

等度量映射的基本出发点, 是认为低维流形嵌入到高维空间后, 直接在高维空间中计算直线距离具有误导性. 因为高维空间中的直线距离在低维嵌入的流形上是不可达的. 通常在流形上使用 “测地线”(geodesic) 距离.

这样距离计算就转化为计算邻阶图上两点间的最短距离. 可以使用经典的 Djikstra 算法或者 Floyd 算法.

新样本映射到低维空间中的方法: 降训练样本中的高维空间坐标作为输入, 低维空间中的坐标作为输出, 训练一个回归学习器来对新样本的低维空间坐标进行预测.

近邻图构造方法有两种: 指定近邻点个数进行构造; 指定距离阈值进行构造.

10.5.2 局部线性嵌入 (Locally Linear Embedding)

LLE 试图保持邻域样本间的线性关系. 其步骤为:

1. 求取重构系数 $\mathbf{w}_i$

$$\begin{aligned} &\min_{\mathbf{w}} \sum_{i=1}^{m} \|\mathbf{x}_i - \sum_{j\in Q} w_{ij}\mathbf{x}_j\|_2^2 \\ &s.t. \sum_{j\in Q} w_{ij} = 1 \end{aligned} \tag{10.10}$$

2. 求取低维坐标 $\mathbf{z}_i$

$$\min_{\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_m} \sum_{i=1}^{m} \|\mathbf{z}_i - \sum_{j\in Q} w_{ij}\mathbf{z}_j\|_2^2 \tag{10.11}$$

10.6 度量学习

高维数据降维的主要目的是通过找到一个合适的低维空间, 在此空间中的学习器性能比原始空间中学习器的性能要好. 实质上, 就是寻找合适的空间, 寻找一个合适的距离度量.

马氏距离 (Mahalanobis Distance) 的定义为:

$$\begin{aligned} dist_{mah}(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) \\ &= \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{M}}^2 \end{aligned} \tag{10.12}$$

其中 $\mathbf{M}$ 是度量矩阵, 是度量学习的目标. 为保持距离非负性和对称性,$\mathbf{M}$ 必须是 (半) 正定.

除了把错误率作为度量学习的目标, 还能在度量学习中引入领域知识, 如定义 “必连”(must-link) 约束集合与 “勿连”(cannot-link) 约束集合.

若学习得到的 $\mathbf{M}$ 是一个低秩矩阵, 通过对 $\mathbf{M}$ 进行特征值分解, 找到一组正交基, 数目为 $rank(\mathbf{M})$, 小于属性数目 d. 于是度量学习的结果可以衍生出一个降维矩阵 $P \in \mathbb{R}^{d \times Rank(\mathbf{M})}$ 用于降维.

第 11 章　特征选择与稀疏学习

11.1　子集搜索与评价

我们将属性称为 “特征”(feature), 对当前学习任务有用的属性称为 “相关特征”(relevant feature), 没什么用的属性称为 “无关特征”(irrelevant feature). 从给定的特征集合中选择出相关特征子集的过程叫做 “特征选择”(feature selection).

特征选择是一个重要的 “数据预处理”(data preprocessing) 过程. 进行特征选择的原因有: 第一, 避免维数灾难; 第二, 去除不相关的特征降低学习任务难度. 需要取出的特征有两种: 第一种是无关特征, 第二种是冗余特征 (需要视情况而定).

选取特征子集的问题是一个 NP 难问题, 因此不能通过随机搜索方法来求得最优解. 为了通过启发式搜索来获得次优解, 需要解决两个问题: 第一, 如何根据评价结果选取下一个特征子集; 第二, 如何评价候选特征子集.

对于第一个 “子集搜索”(subset search) 问题, 采用贪心算法进行特征子集的搜索, 通常使用的策略有:“前向”(foward) 搜索,“后向”(backward) 搜索和 “双向”(bidirection) 搜索, 但是这些搜索策略并未考虑特征之间的关联.

对于第二个 “子集评价”(subset evalution) 问题, 只要评估特征子集划分的信息增益即可. 设 $\mathcal{D}$ 是样本集合;p_i 是 $\mathcal{D}$ 中第 i 类样本所占比例. 对于属性子集 A, 根据取值将 $\mathcal{D}$ 划分为 $\{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^V\}$, 则属性子集增益为

$$Gain(A) = Ent(\mathcal{D}) - \sum_{i=1}^{V} \frac{|\mathcal{D}^i|}{|\mathcal{D}|} Ent(\mathcal{D}^i) \tag{11.1}$$

其中,$Ent(\mathcal{D}) = -\sum_{i=1}^{|\mathcal{Y}|} p_k \log_2 p_k$.

一般的,A 确定了对 $\mathcal{D}$ 的一个划分;$\mathcal{Y}$ 是样本标记, 确定了对样本集 $\mathcal{D}$ 的真实划分, 只要能量化这两个划分的差异, 就能对 A 进行评价.

子集搜索和子集评价结合起来就是特征选择. 常见的特征选择策略有: 过滤式 (filter), 包裹式 (wrapper) 和嵌入式 (embedding).

11.2 过滤式选择

过滤式特征选择先对特征集合进行特征选择, 然后再训练学习器, 特征选择过程与后续学习器训练过程无关.

Relief 特征选择方法是过滤式的特征选择方法, 该方法使用 "相关统计量" 来度量特征的重要性. 假设有训练集 $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$, 对于每个实例 $\mathbf{x}_i$,Relief 先在 $\mathbf{x}_i$ 的同类样本中寻找其最近邻 $\mathbf{x}_{i,nh}$, 称为 "猜中近邻"(near-hit); 再从异类中寻找其最近邻 $\mathbf{x}_{i,nm}$, 称为 "猜错近邻"(near-miss), 则相关统计量对应于属性 j 的分量为

$$\delta^j = \sum_i -\mathrm{diff}(x_i^j, x_{i,nh}^j)^2 + \mathrm{diff}(x_i^j, x_{i,nm}^j)^2 \tag{11.2}$$

相关统计量分量值越大, 对应属性的分类能力越强.

Relief-F 是 Relief 的多类变体, 能处理多类问题.

$$\delta^j = \sum_i -\mathrm{diff}(x_i^j, x_{i,nh}^j)^2 + \sum_{l \neq k} \left(p_l \times \mathrm{diff}(x_l^j, x_{l,nm}^j)^2 \right) \tag{11.3}$$

其中,p_l 为第 l 类样本在数据集 $\mathcal{D}$ 中所占的比例.

11.3 包裹式选择

包裹式特征选择直接把最终将要使用的学习器性能作为特征子集的评价准则. 包裹式特征选择的目的就是为给定学习器选择最有利于其性能的特征子集.

从学习器性能来看, 包裹式比过滤式要好; 从计算开销来看, 由于需要多次训练学习器, 包裹式的开销比过滤式要大.

LVW(Las Vegas Wrapper) 是在拉斯维加斯框架下使用随机策略进行搜索. 若有时间限制, 则不一定能给出满足要求的解. 与之相对应的, 蒙特卡罗方法一定会给出解.

11.4 嵌入式选择与 L1 正则化

嵌入式特征选择是将特征选择过程与学习器训练过程融为一体, 即在学习器训练过程中自动地进行特征选择.

对于线性回归模型, 岭回归 (Ridge Regression) 的优化目标为

$$\min_{w} \sum_{i=1}^{m} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2 \tag{11.4}$$

如果将正则化项中 $2-$ 范数替换为 $p-$ 范数, 特别的, 当 $p = 1$ 时, 有 LASSO(Least Absolute Shrinkage and Selection Operator)

$$\min_{\mathbf{w}} \sum_{i=1}^{m} (y_i - \mathbf{w}^{\mathbf{T}} \mathbf{x_i})^2 + \lambda \|\mathbf{w}\|_1 \tag{11.5}$$

相比于岭回归,LASSO 更容易获得 “稀疏”(sparse) 解. 这意味着 $\mathbf{w}$ 的非零分量对应的特征最终才在模型中得以体现, 即进行了特征选择.

求解 $L1$ 正则化问题可使用近端梯度下降 (Proximal Gradient Descent, PGD) 方法. 对优化目标

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda \|\mathbf{x}\|_1 \tag{11.6}$$

若 $f(\mathbf{x})$ 可导且 $\nabla f(\mathbf{x})$ 满足 L-Lipschitz 条件则可求解出次优解. 其中,L-Lipschitz 条件为存在 $L > 0$, 使得

$$\|\nabla f(\mathbf{x}') - \nabla f(\mathbf{x})\|_2^2 \leq L \|\mathbf{x}' - \mathbf{x}\|_2^2, \forall \mathbf{x}, \mathbf{x}' \tag{11.7}$$

求解过程迭代公式为

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} \frac{L}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda \|\mathbf{x}\|_1 \tag{11.8}$$

其中 $\mathbf{z} = \mathbf{x}_k - \frac{1}{L} \nabla f(\mathbf{x}_k)$

易知 $\mathbf{x}$ 各个分量互不影响, 则有闭式解

$$x_{k+1}^i = \begin{cases} z^i - \lambda/L, & \lambda/L < z^i \\ 0, & |z^i| \leq \lambda/L \\ z^i + \lambda/L, & z^i < -\lambda/L \end{cases} \tag{11.9}$$

推导闭式解的过程:

$$\begin{aligned} x_{k+1}^i &= \arg\min_{x^i} \frac{L}{2}(x^{i2} - 2x^i z^i + z^{i2}) + \lambda|x^i| \\ &= \begin{cases} \frac{L}{2}(x^{i2} - 2x^i z^i + z^{i2}) + \lambda|x^i|, & x^i \geq 0 \\ \frac{L}{2}(x^{i2} - 2x^i z^i + z^{i2}) - \lambda|x^i|, & x^i < 0 \end{cases} \end{aligned} \tag{11.10}$$

对于上述二次曲线 (抛物线), 取得最小值点, 并与条件 $x^i \geq 0$ 或 $x^i < 0$ 进行综合判断.

11.5 稀疏表示与字典学习

特征选择所考虑的问题是特征具有稀疏性, 即从特征集中选择出与学习任务相关的特征子集. 但是从另外一个角度来看, 如果训练样本对应的特征向量具有稀疏性, 那么在高维空间中样本分布就有可能线性可分, 学习器可能会取得较好的性能. 此外, 由于样本是稀疏的, 计算和存储的开销会比较小.

由于稀疏特征具有以上的优点, 因此, 如何学习出一个 “字典”, 为稠密表达的样本转化为稀疏形式, 是 “字典学习”(directoinary learning), 又称 “稀疏编码”(sparse coding) 所要解决的问题.

给定数据集 $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, 稀疏编码的模型为

$$\min_{\mathbf{B}, \alpha_i} \sum_{i=1}^{m} \|\mathbf{x}_i - \mathbf{B}\alpha_i\|_2^2 + \lambda\|\alpha_i\|_1 \tag{11.11}$$

上述问题可通过交替变量优化法求解:

1. 固定 $\mathbf{B}$, 为每个样本 $\mathbf{x}_i$ 求解对应 α_i

$$\min_{\alpha_i} \|\mathbf{x}_i - \mathbf{B}\alpha_i\|_2^2 + \lambda\|\alpha_i\|_1 \tag{11.12}$$

2. 以 α_i 为初值更新 $\mathbf{B}$

$$\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{B}\mathbf{A}\|_F^2 \tag{11.13}$$

其中 $\|\cdot\|_F$ 是矩阵的 Frobenius 范数, 上述优化问题常用于基于逐列更新策略的 KSVD 方法求解.

11.6 压缩感知

压缩感知所要解决的问题是从部分信息中恢复出全部的信号.

原问题可描述为: 假定有长度为 m 的离散信号 $\mathbf{x}$, 不放假定以远小于奈奎斯特采样定理要求的采样率进行采样, 得到长度为 n 的采样后信号 $\mathbf{y}$,$n \ll m$, 即

$$\mathbf{y} = \mathbf{\Phi x} \tag{11.14}$$

其中 $\Phi \in \mathbb{R}^{n\times m}$ 是对信号 $\mathbf{x}$ 的测量矩阵.

不妨假定存在某个线性变换 $\Psi \in \mathbb{R}^{m\times m}$, 使得 $\mathbf{x}$ 可表示为

$$\mathbf{y} = \mathbf{\Phi\Psi s} = \mathbf{As} \tag{11.15}$$

其中 $\mathbf{A} = \mathbf{\Phi\Psi} \in \mathbb{R}^{n\times m}$. 于是, 若能根据 $\mathbf{y}$ 恢复出 $\mathbf{s}$, 则可通过 $\mathbf{x} = \mathbf{\Psi s}$ 来恢复出信号 $\mathbf{x}$.

更进一步, 若 $\mathbf{s}$ 具有稀疏性, 这个问题就很好解. 实际上, 获得稀疏解 $\mathbf{s}$ 的途径有很多.

通常认为, 压缩感知分为 “感知测量” 和 “重构恢复” 两个阶段. 前者是将原始信号进行处理以获得稀疏表示的过程; 后者是从稀疏表示中恢复出原始信号的过程.

压缩感知中一个重要的原理是 “限定等距性”(Restricted Isometry Property, RIP): 对于 $n \times m(n \ll m)$ 的矩阵 $\mathbf{A}$, 若存在常数 $\delta_k \in (0, 1)$, 使得对于任意向量 $\mathbf{s}$ 和 $\mathbf{A}_k \in \mathbb{R}^{n\times k}$ 有

$$(1 - \delta_k)\|\mathbf{s}\|_2^2 \leq \|\mathbf{A}_k\mathbf{s}\|_2^2 \leq (1 + \delta_k)\|\mathbf{s}\|_2^2 \tag{11.16}$$

则称 $\mathbf{A}$ 满足 $k-$RIP. 进而可从下列问题中求解出 $\mathbf{s}$, 并恢复出 $\mathbf{x}$

$$\begin{aligned} &\min_{\mathbf{s}} \|\mathbf{s}\|_0 \\ &s.t. \mathbf{y} = \mathbf{As} \end{aligned} \tag{11.17}$$

上述问题涉及到 0 范数, 是 NP 难问题, 但该问题与

$$\begin{aligned} &\min_{\mathbf{s}} \|\mathbf{s}\|_1 \\ &s.t. \mathbf{y} = \mathbf{As} \end{aligned} \tag{11.18}$$

问题共解.

另外, 本节中所举的例子还涉及到矩阵补全 (matrix completion) 技术, 即通过解

$$\begin{aligned} &\min_{\mathbf{X}} rank(\mathbf{X}) \\ &s.t. (\mathbf{X})_{ij} = (\mathbf{A})_{ij}, (i,j) \in \Omega \end{aligned} \tag{11.19}$$

其中 Ω 是已观测值下标集合;$\mathbf{X}$ 是需要恢复的信号;$\mathbf{A}$ 是包含缺失值的观测信号. 该问题也是 NP 难问题, 但与下列问题同解

$$\begin{aligned} &\min_{\mathbf{X}} \|\mathbf{X}\|_* \\ &s.t. (\mathbf{X})_{ij} = (\mathbf{A})_{ij}, (i,j) \in \Omega \end{aligned} \tag{11.20}$$

其中 $\|\mathbf{X}\|_* = \sum_{j=1}^{\min\{m,n\}} \sigma_j(\mathbf{X})$. 其中,$\sigma_j(\mathbf{X})$ 是 $\mathbf{X}$ 的奇异值. 可通过半正定规划 (Semi-Definite Programming, SPD) 求解.

第 12 章　计算学习理论

本章主要从 “计算” 的角度来分析机器学习算法, 并为机器学习算法提供理论支撑. 由于牵涉到大量的概念定义, 本章的笔记不做过多重复, 仅仅从思路上对各个概念做分析串联.

12.1　基础知识

本节首先重新强调了泛化误差与经验误差这两个概念. 这两个概念在本章第 2 章中就已经被引入过. 大致可以理解为, 泛化误差是训练出的模型在除训练样本外的新样本集上的误差; 经验误差是训练出的模型在训练集上的误差. 本章后续的部分重点关注经验误差与泛化误差之间的逼近程度.

在本章后面的部分, 经常要用到下列几个不等式:

- Jensen 不等式
- Hoeffding 不等式
- McDiarmid 不等式

12.2　PAC 学习

概念这个定义可以理解为, 从样本空间到标记空间的映射, 而我们所关注的本质上的规律为目标概念, 所有目标概念的集合为概念类. 从学习算法较多来看, 他所考虑的所有可能概念的集合称为假设空间. 通常情况下, 假设空间和概念空间通常是不同的. 若目标概念被包含在假设空间中, 则称该假设空间对应的学习算法是 “可分的”(separable) 或者 “一致的”(consistent). 反之, 则称学习算法是 “不可分的”(non-separable) 或者 “不一致的”(non-consistent).

我们通常所期望的是通过学习算法学得的假设尽可能接近目标概念. 但是由于受到训练数据集采样的因素制约, 可能会存在若干个等效的假设, 如何从这些等效假设中以较大的概率选取出最接近目标概念的假设, 是 PAC 学习的目的.

PAC 学习包含:PAC 辨识,PAC 可学习,PAC 学习算法, 样本复杂度等定义. 如果在 PAC 学习中假设空间和概念类完全相同, 则称 “恰 PAC 可学习”(properly PAC learnable). 但是实际情况中研究的重点是假设空间与概念类不同的情况, 一般而言, 假设空间越大, 包含目标概念的可能性越大, 但从中找到目标概念的难度也就越大. 当假设空间规模有限时, 则称为有限假设空间, 否则称为无限假设空间.

12.3 有限假设空间

12.3.1 可分情形

可分情形是目标概念包含在假设空间中的情况, 如果训练集足够大, 则可通过不断去除假设集合中不满足训练集样本标记的假设, 直到假设集合中仅剩下一个集合为止. 但是如果假设空间中剩余多个假设时, 就无法区分剩余假设的优劣了.

从另外一个角度看, 到底需要多少样本才能区分学得目标概念的有效近似呢? 对于 PAC 学习来说, 只要训练集合 D 的规模能使学习算法 $\mathcal{L}$ 能以概率 $1-\delta$ 找到目标概念的 ϵ 近似即可.

1. 首先估计泛化误差大于 ϵ 但是对于分布 $\mathcal{D}$ 上任何样例 $(\mathbf{x}, y)$ 有

$$\begin{aligned} P\Big(h(\mathbf{x}) = y\Big) &= 1 - P\Big(h(\mathbf{x}) \neq y\Big) \\ &= 1 - E(h) \\ &< 1 - \epsilon \end{aligned} \tag{12.1}$$

2. 同样有经验误差

$$P\Big((h(\mathbf{x}_1)=y_1)\wedge(h(\mathbf{x}_2)=y_2)\wedge\cdots\wedge(h(\mathbf{x}_m)=y_m)\Big)=(1-P(h(\mathbf{x})\neq y)^m)$$
$$<(1-\epsilon)^m \tag{12.2}$$

3. 泛化误差大于 ϵ 且在训练集上表现完美的所有假设出现概率不大于 δ

$$\begin{aligned} P\Big(h\in\mathcal{H}:E(h>\epsilon)\wedge\widehat{E}(h)=0\Big) &< |\mathcal{H}|(1-\epsilon)^m \\ &< |\mathcal{H}|e^{-m\epsilon} \end{aligned} \tag{12.3}$$

4. 令上式不大于 δ, 即可导出

$$m\geq\frac{1}{\epsilon}(\ln|\mathcal{H}|+\ln\frac{1}{\delta}) \tag{12.4}$$

从以上可以看出, 所有有限假设空间都是 PAC 可学习的.

但是大部分模型的参数选择范围都在无限区间上, 即假设空间是无限的. 是否可对参数选择范围作出限制将其转化为有限假设空间?(具体可看 12.4 节.)

12.3.2 不可分情形

对于较难的学习问题, 或者说大部分情形来说, 目标概念往往不在假设空间中. 这时可以衡量经验误差 $\widehat{E}(h)$ 与泛化误差 $E(h)$ 之间的差异, 有

$$P\Big(|\widehat{E}(h)-E(h)|\geq\epsilon\Big)\leq 2\exp\Big(-2m\epsilon^2\Big) \tag{12.5}$$

.

在此情况下, 可以从假设空间中找出一个泛化误差最小的假设, 这个 ϵ 近似也是一个比较好的学习目标. 基于此可以引出不可知学习 (agnostic PAC learnable) 的定义.

12.4 VC 维

现实学习任务所面临的通常是假设空间, 欲对此种情形的可学习情况进行研究, 需度量假设空间的复杂度.

增长函数 $\Pi_{\mathcal{H}}(m)$ 表示假设空间 $\mathcal{H}$ 对 m 个示例所能赋予标记的最大可能结果数. 对二分类问题来说,$\mathcal{H}$ 中的假设对 D 中示例赋予标记的每种可能结果称为对 D 的一种 “对分”. 若假设空间 $\mathcal{H}$ 能实现示例集 D 上的所有对分, 即 $\Pi_{\mathcal{H}}(m) = 2^m$, 则称示例集 D 能被假设空间 $\mathcal{H}$“打散”.

假设空间 $\mathcal{H}$ 的 VC 维是能被 $\mathcal{H}$ 打散的最大示例集的大小.(结合例 12.1 和例 12.2, 首先从示例集入手, 逐渐增加示例集中样本的个数, 进而确定 VC 维的大小.)

基于 VC 维的泛化误差界是分布无关, 数据独立的. 任何 VC 维有限的假设空间 $\mathcal{H}$ 都是 (不可知)PAC 可学习的.

12.5 Rademacher 复杂度

由于 VC 维泛化误差界是分布无关, 数据独立的, 因此具有 “普适性”. 但是从另一方面看, 这样的泛化误差界通常也比较 “松”.Rademacher 复杂度 (Rademacher complexity) 是另一种刻画假设空间复杂度的途径, 它在一定程度上考虑了数据分布.

为了计算泛化误差界, 本节首先定义了经验 Radematcher 复杂度和 Rademacher 复杂度. 最后计算出针对回归问题和二分类问题的泛化误差界. 这些 Rademacher 的泛化误差界与分布 $\mathcal{D}$ 或数据 D 有关.

12.6 稳定性

无论是基于 VC 维或者是 Rademacher 复杂度来推导泛化误差界, 所得到的结果均与具体学习算法无关, 所所有的学习算法均适用. 稳定性分析是针对学习算法考察的一种重要方式. 算法的稳定性是指算法在输入发生变化时, 输出是否会随之发生较大的变化.

输入变化即指样例集变化: 移除第 i 个样例得到的集合; 替换第 i 个样本得到的集合.

损失函数即预测标记与真实标记之间的差别: 泛化损失; 经验损失; 留一损失.

定义 12.10 定义了 $\mathcal{L}$ 关于损失函数 $\updownarrow$ 的 β-均匀稳定性, 需要注意的是, 移除示例的稳定性包含替换示例的稳定性.

学习算法的稳定性分析关注的是 $|\hat{l}(\mathcal{L}, D) - l(\mathcal{L}, D)|$, 而假设空间复杂度分析关注的是 $\sup_{h\in\mathcal{H}}(\widehat{E}(h) - E(h))$. 从以上可以看出, 稳定性假设不用分析所有可能的假设, 只需要根据算法本身的特性来讨论输出假设的泛化误差界.

定理 12.9 表明了学习算法稳定性与假设空间复杂度之间的关系. 学习算法的稳定性和假设空间的复杂度并非无关, 由稳定性的定义可知, 两者通过损失函数 $\updownarrow$ 联系起来.

第 13 章　半监督学习

13.1　未标记样本

如果我们有训练样本集 $D_l = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$, 这 l 个样本类别标记已知, 则称为 “有标记”(labeled) 样本; 此外还有 $D_u = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}\}, l \ll u$, 这 u 个样本类别标记未知, 称为 “未标记”(unlabeled) 样本. 如果直接使用有标记样本, 则 D_l 用来建模,D_u 中的信息被浪费; 另一方面, 如果 D_l 较小, 则训练样本不足, 学得模型的泛化能力往往不佳. 除了将 D_u 中样本全部进行标记外, 还有几种代价较小的方法.

如果我们每次都能挑出对改善模型性能帮助较大的未标记样本, 则能构建出较强的模型. 这样的标记成本较小. 这样的学习方式称为 “主动学习”(active learning), 其目的是通过使用尽可能少的 “查询”(query) 来获得尽量好的性能.

如果完全不使用查询这种主动干预的方式进行学习, 只利用未标记样本的分布信息以及标记样本的信息进行学习, 这样的方式仍然是可行的.

假设**未标记样本与有标记样本是从同样的数据源独立同分布采样而来**, 则它们所包含的数据分布信息对于建模将会有很大帮助. 这样就必然要做一些将未标记样本所揭示的数据分布信息与类别标记想联系的假设. 常见的假设有 “聚类假设”(cluster assumption) 和 “流形假设”(manifold assumption), 这些假设的本质都是 “相似的样本拥有相似的输出”.

半监督学习可进一步划分为纯 (pure) 半监督学习和直推学习 (transductive learning). 纯半监督学习假定训练数据中的未标记样本并非待预测的数据, 而直推学习假定学习过程中的未标记样本就是待预测数据.

13.2 生成式方法

生成式方法 (generative methods) 是直接基于生成式模型的方法. 此类方法假设所有数据 (无论是否有标记) 都是由同一个潜在的模型 “生成” 的. 生成式方法有好多种不同的类别, 但是不同类别的主要区别在于生成式模型的假设.

以高斯混合模型为例, 假设样本由

$$p(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \cdot p(\mathbf{x}|\mu_i, \mathbf{\Sigma}_i) \tag{13.1}$$

生成, 其中,$\alpha_i \geqslant 0, \sum_{i=1}^{N} \alpha_i = 1$;$p(\mathbf{x}|\mu_i, \mathbf{\Sigma}_i)$ 是样本 $\mathbf{x}$ 属于第 i 个高斯混合成分的概率;μ_i 和 $\mathbf{\Sigma}_i$ 为该高斯混合成分的参数.

令 $f(\mathbf{x}) \in \mathcal{Y}$ 表示模型 f 对 $\mathbf{x}$ 的预测标记,$\Theta \in \{1, 2, \ldots, N\}$ 表示样本 $\mathbf{x}$ 隶属的高斯混合成分. 由最大化后验概率可知,

$$\begin{aligned} f(\mathbf{x}) &= \arg\max_{j \in \mathcal{Y}} p(y = j|\mathbf{x}) \\ &= \arg\max_{j \in \mathcal{Y}} \sum_{i=1}^{N} p(y = j, \Theta = i|\mathbf{x}) \\ &= \arg\max_{j \in \mathcal{Y}} \sum_{i=1}^{N} p(y = j|\Theta = i, \mathbf{x}) \cdot p(\Theta = i|\mathbf{x}) \end{aligned} \tag{13.2}$$

其中,$p(\Theta = i|\mathbf{x}) = \frac{\alpha_i \cdot p(\mathbf{x}|\mu_i, \Sigma_i)}{\sum_{i=1}^{N} \alpha_i \cdot p(\mathbf{x}|\mu_i, \Sigma_i)}$ 为样本 $\mathbf{x}$ 由第 i 个高斯混合成分生成的后验概率,$p(y = j|\Theta = i, \mathbf{x})$ 为 $\mathbf{x}$ 为 $\mathbf{x}$ 由第 i 个高斯混合成分生成且其类别为 j 的概率. 不难发现,$p(y = j|\Theta = i, \mathbf{x})$ 该项需要知道样本的类别标记, 而 $p(\Theta = i|\mathbf{x})$ 不需要知道样本的类别标记, 通过引入大量的未标记数据, 提升分类器的性能.

高斯混合模型参数可以通过 EM 算法求得.

将上述高斯混合模型替换成其他模型 (如混合专家模型, 朴素贝叶斯模型) 即可得到其他半监督学习方法. 但是需要注意的是, 模型假设必须准确, 否则利用未标记样本可能会降低学习器的泛化性能.

13.3 半监督 SVM

半监督支持向量机 (Semi-Supervised Support Vector Machine, S3VM) 是支持向量机在半监督学习上的推广. 在考虑未标记样本后,S3VM 试图找到能将两类有标记样本区分开并且穿过数据低密度区域的划分超平面.

半监督支持向量机中最著名的是 TSVM(Transductive Support Vector Machine).TSVM 试图考虑对未标记样本进行各种可能的标记指派 (label assignment), 即尝试将每个未标记样本分别作为正样本或者负样本, 然后在所有这些结果中, 寻找一个在所有样本上间隔最大化的划分超平面. 即

$$\begin{aligned} &\min_{w,b,\hat{y},\xi} \frac{1}{2}\|w\|_2^2 + C_l \sum_{i=1}^{l} \xi_i + C_u \sum_{i=l+1}^{m} \xi_i \\ &s.t. y_i(w^T \mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, l, \\ &\quad \hat{y}_i(w^T \mathbf{x}_i + b) \geq 1 - \xi_i, i = l+1, l+2, \ldots, m, \\ &\quad \xi_i \geq 0, i = 1, 2, \ldots, m \end{aligned} \tag{13.3}$$

其中 ξ_i 为松弛向量;C_l 与 C_u 是由用户指定的用于平衡模型复杂度, 有标记样本与未标记样本重要程度的折中参数.

但是, 尝试所有未标记样本的标记指派是一个穷举过程, 为提升算法效率, 需要进行一定的优化:

1. 利用有标记样本学习得到一个 SVM;
2. 利用这个 SVM 对未标记样本进行标记指派, 需要注意的是,C_u 要设置为比 C_l 小的值;
3. TSVM 找出两个标记指派为异类并且很可能发生错误的未标记样本, 交换其标记, 更新划分超平面和松弛向量;
4. 重复上一步骤, 并且在这个过程中逐渐增大 C_u 知道 $C_l = C_u$ 为止.

如果在训练过程中出现类别不平衡问题, 需要将优化目标中的 C_u 拆分为 C_u^+ 和 C_u^- 两项, 分别进行优化.

由以上可以看出, 半监督 SVM 研究的一个重点是如何设计出高效的优化求解策略.

13.4 图半监督学习

给定一个数据集, 我们将其映射为一个图, 数据集中每个样本对应于图中一个结点, 若两个样本间的相似度很高, 则对应的结点之间存在一条边, 边的 “强度”(strength) 正比于样本之间的相似度 (或相关性). 通过建立起相似度矩阵, 我们利用已标记样本的信息对未标记样本进行标记指派, 在得到所有或者大部分未标记样本的标记后, 再利用有监督学习的方法进行学习.

首先基于 $D_l \cup D_u$ 构建一个图 $G = (V, E)$, 边集 E 可表示为一个亲和矩阵 (affinity matrix):

$$(\mathbf{W})_{ij} = \begin{cases} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}), & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases} \tag{13.4}$$

其中 $i, j \in \{1, 2, \ldots, m\}$,$\sigma > 0$ 是用户指定的高斯函数带宽参数.

加入从图 G 学得一个实值函数 $f : V \to \mathbb{R}$, 则可定义关于 f 的 “能量函数”(energy function):

$$E(f) = \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} (\mathbf{W})_{ij} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 \tag{13.5}$$

具有最小能量的函数 f 在有标记样本上满足 $f(\mathbf{x}_i) = y_i (i = 1, 2, \ldots, l)$, 在未标记样本上满足 $\boldsymbol{\Delta}\mathbf{f} = 0$, 其中,$\boldsymbol{\Delta} = \mathbf{D} - \mathbf{W}$ 为拉普拉斯矩阵.

在此定义下, 可以推导得到二分类问题和多类问题的标记传播方法 (见 P302-303). 同时可以看出, 此类算法的缺陷很明显: 存储开销为 $O(m^2)$, 无法处理大规模数据; 构图过程仅能考虑训练样本集, 难以判断新样本在图中的位置.

13.5 基于分歧的方法

基于分歧的方法 (disagreement-based methods) 使用多学习器, 而学习器之间的 “分歧”(disagreement) 对未标记数据的利用至关重要.

“协同训练”(co-training) 是此类方法的重要代表. 它最初是针对 “多视图”(multi-view) 数据设计的, 因此也被看做 “多视图学习”(multi-view learning) 的代表. 一个数据对象往往拥有多个 “属性集”(attribute set), 每个属性

集就构成了一个 “视图”(view). 假设不同视图具有 “相容性”(compatibility), 即其所包含的关于输出空间 $\mathcal{Y}$ 的信息是一致的. 在此假设下, 显式地考虑多视图有很多好处. 在相容性基础上, 不同视图信息的 “互补性” 会给机器学习的构建带来很多便利.

协同训练正是很好地利用了多视图的 “相容互补性”. 其利用未标记数据的步骤为:

1. 在每个视图上基于有标记样本分别训练出一个分类器;
2. 让每个分类器分别挑选出置信度较高的未标记样本进行标记指派;
3. 将上一步中进行标记指派的样本提供给另外一个分类器进行训练更新, 再将后一分类器中标记指派的样本及其指派标记输送回前一分类器进行训练更新;
4. 不断重复前两步, 直至算法收敛.

视图的条件独立在现实任务中很难被满足, 因此提升的幅度不会那么大. 但即便是在更弱的条件下, 协同训练仍可有效地提升若分类器的性能.

为了使用此方法, 需能生成具有显著分歧, 性能尚可的多个学习器, 但是当有标记样本很少, 尤其是数据不具有多个视图时, 要做到这一点并不容易, 需有巧妙地设计.

13.6 半监督聚类

聚类是一种无监督学习, 在实际中我们通常可以获得一些额外的监督信息, 因此可以通过半监督聚类 (semi-supervised clustering) 来利用监督信息以获得更好的聚类效果.

聚类任务中获得的监督信息可以分为两种:

1. “必连” 与 “勿连” 信息;
2. 少量有标记样本.

对于第一种使用约束 k 均值聚类算法进行聚类; 对于第二类, 直接将它们作为 “种子”, 用它们初始化 k 均值算法的 k 个聚类中心, 并且在聚类簇迭代过程中不改变种子样本的簇隶属关系.

第 14 章　概率图模型

本章简要介绍了概率图模型的体系, 为了便于理解本章内容, 需要首先明确以下几个概念:

- 概率模型: 提供了一种描述框架, 将学习任务归结为计算变量的概率分布.
- 推断: 利用已知变量推测出未知变量分布.

假设关心变量集合为 Y, 可观测变量集合为 O, 其他变量集合为 R.“生成式”(generative) 模型考虑 $P(Y, R, O)$;“判别式”(discriminative) 模型考虑条件分布 $P(Y, R|O)$. 给定一组观测变量值, **推断就是要由** $P(Y, R, O)$ **或者** $P(Y, R|O)$ **得到条件概率分布** $P(Y|O)$.

利用概率求和规则直接消去变量 R 复杂度太高, 显然不可行. 此外, 属性变量间还存在着复杂的联系. 概率图模型 (Probabilistic Graphical Model) 正是用图来表达变量相关关系的概率模型. 概率图模型可以分为两类:

- 有向图模型或者贝叶斯网: 使用有向无环图表示变量间的依赖关系 (casual relationship).
- 无向图模型或者马尔可夫网: 使用无向图表示变量间的相关关系 (soft constraints between random variables).

14.1　隐马尔可夫模型

有向图模型 | 生成式模型

隐马尔可夫模型 (Hidden Markov Mode) 是结构最简单的动态贝叶斯网.

隐马尔可夫模型中状态分为两组:

- 状态变量 (隐变量)$\{y_1, y_2, \ldots, y_n\}$, 其中 $y_i \in \mathcal{Y}$ 表示第 i 时刻系统的状态;
- 观测变量 $\{x_1, x_2, \ldots, x_n\}$, 其中 $x_i \in \mathcal{X}$ 表示第 i 时刻的观测值.

在任一时刻, 观测变量的取值仅依赖于状态变量;t 时刻的状态变量仅依赖于 $t-1$ 时刻的状态. 由此, 可得所有概率的联合概率分布即结构信息

$$P(x_1, y_1, \ldots, x_n, y_n) = P(y_1)P(x_1|y_1)\prod_{i=2}^{n} P(y_i|y_{i-1})P(x_i|y_i) \tag{14.1}$$

此外, 还需要以下参数才能确定一个完整的马尔科夫模型:

- 状态转移概率
$$a_{ij} = P(y_{t+1} = s_j|y_t = s_i) \tag{14.2}$$
- 输出观测概率
$$b_{ij} = P(x_t = o_j|y_t = s_j) \tag{14.3}$$
- 初始状态概率
$$\pi_i = P(y_1 = s_1) \tag{14.4}$$

在实际中, 人们常关注三个基本问题:

- 如何评估模型与观测序列的匹配程度?
- 如何根据观测序列推断出隐藏的模型状态?
- 如何训练模型使其能最好地描述观测数据?

14.2 马尔可夫随机场

无向图模型 | 生成式模型

马尔可夫随机场 (Markov Random File, MRF) 是典型的马尔可夫网.

- 马尔可夫随机场有一组势函数 (potential functions), 亦称 “因子”(factor), 这是定义在变量子集上的非负实函数, 主要用于定义概率分布函数.

- 如果结点子集中任意两个结点间都有边连接, 则称该结点子集为 “团”(clique). 如果一个团中再加入任何一个结点都不能再形成团, 则称该团为 “极大团”(maximal clique).

马尔可夫随机场中, 多个变量之间的联合概率分布能基于团分解为多个因子的乘机, 每个因子仅与一个团相关.

$$P(\boldsymbol{x}) = \frac{1}{Z} \prod_{Q \in \mathcal{C}} \psi_Q(\boldsymbol{x}_Q) \tag{14.5}$$

其中,ψ_Q 为与团 Q 对应的势函数;$Z = \sum_{\boldsymbol{x}} \prod_{Q \in \mathcal{C}} \psi_Q(\boldsymbol{x}_Q)$ 为规范化因子. 实际中, 通常用极大团以及对应的势函数来代替上述公式里的团.

分离集 (seprating set): 若从结点集 A 中的结点到 B 中的结点都必须经过结点集 C 中的结点, 则称结点集 A 和 B 被结点集 C 分离.C 称为分离集.

基于分离集定义, 可得 “全局马尔科夫性”(global Markov property) 定义, 即给定两个变量子集的分离集, 则这两个变量子集条件独立. 同时还可以得到 “局部马尔可夫性”(local Markov property) 和 “成对马尔可夫性”(pairwise Markov property) 两个推论.

- 局部马尔可夫性: 给定某变量的邻接变量, 则该变量条件独立于其他变量.

- 成对马尔可夫性: 给定所有其他变量, 两个非邻接变量条件独立.

势函数 $\psi_Q(\boldsymbol{x}_Q)$ 作为连接图和概率分布的桥梁, 它的作用是刻画变量集 $\boldsymbol{x}_Q$ 中变量之间的相关关系. 它应该是非负函数, 并且在所偏好的变量取值上有较大函数值.

14.3 条件随机场

无向图模型 | 判别式模型

条件随机场 (Conditional Random Field,CRF) 是一种判别式无向图模型. 条件随机场视图对多个变量在给定观测值后的条件概率进行建模.

令 $G =< V, E >$ 表示结点与标记变量 $\boldsymbol{y}$ 中元素一一对应的无向图,y_v 表示与结点 v 对应的标记变量,$n(v)$ 表示结点 v 的邻接节点, 若图 G 的每个变量 y_v 都满足马尔可夫性, 即

$$P(y_v|\boldsymbol{x}, \boldsymbol{y}_{V\setminus v}) = P(y_v|\boldsymbol{x}, \boldsymbol{y}_{n(v)}) \tag{14.6}$$

则 $(\boldsymbol{y}, \boldsymbol{x})$ 构成一个条件随机场. 理论上来说 G 可以具有任何结构, 只要能表示标记变量 $\boldsymbol{y}$ 之间的**条件独立性关系**即可.

条件随机场使用势函数和图结构上的团来定义条件概率 $P(\boldsymbol{y}|\boldsymbol{x})$. 以链式条件随机场为例, 通过选用指数势函数并引入特征函数, 条件概率被定义为

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z}\exp\left(\sum_j\sum_{i=1}^{n-1}\lambda_j t_j(y_{i+1}, y_i, \boldsymbol{x}, i) + \sum_k\sum_{i=1}^{n}\mu_k s_k(y_i, \boldsymbol{x}, i)\right) \tag{14.7}$$

其中, 转移特征函数 $t_j(y_{i+1}, y_i, \boldsymbol{x}, i)$ 用以刻画相邻标记变量之间的相关关系以及观测序列对它们的影响; 状态特征函数 $s_k(y_i, \boldsymbol{x}, i)$ 用于刻画观测序列对标记变量的影响;λ_j 和 μ_k 为参数;Z 为规范化因子.

条件随机场和马尔可夫随机场均使用团上的势函数来定义概率; 但是条件随机场处理的是条件概率, 马尔科夫随机场处理的是联合概率.

14.4 学习与推断

精确推断方法

对于概率图模型, 为求得**目标变量的边际分布**或者**以某些可观测变量为条件的条件分布**, 还需要确定具体分布的参数. 若将参数视为待推测的变量, 则参数估计过程和推断十分类似,[1]. 推断问题的关键在于如何高效地计算边际分布[2]. 概率图模型的推断方法可以分为两类, 一类是精确推断方法, 另一类是近似推断方法.

[1]因此将参数估计 “吸收” 至推断过程中.

[2]联合概率可由概率图模型获得; 条件概率可归结为计算边际分布.

14.4.1 变量消去

精确推断的**实质**是一类动态规划算法, 它利用图模型所描述的条件独立性来削减计算目标概率值所需的计算量.

通过利用乘法对加法的分配率, 变量消去法[3]**把多个变量的积的求和问题, 转化为对部分变量交替进行求积与求和的问题**. 但是缺点也很明显, 如果需要计算多个边际分布, 则使用变量消去法会造成大量的冗余计算.

14.4.2 信念传播

信念传播 (Belief Propagation) 算法将变量消去法中的求和操作看做一个消息传递过程, 较好地解决了求解多个边际分布时的重复计算问题.

变量消去法通过求和操作

$$m_{ij}(x_j) = \sum_{x_i} \psi(x_i, x_j) \prod_{k \in n(i) \setminus j} m_{ki}(x_i) \tag{14.8}$$

消去变量 x_i. 在信念传播算法中, 这个操作视为从结点 x_i 向 x_j 传递了一个消息 $m_{ij}(x_j)$.

一个结点尽在接收到来自其它所有邻接结点的消息后才能向另一个结点发送消息, 且结点的边际分布正比于它所接收的消息的乘积

$$P(x_i) \propto \prod_{k \in n(i)} m_{ki}(x_i) \tag{14.9}$$

如果图中没有环, 则信念传播法经过两个步骤即可完成所有的消息传播:

1. 指定一个根节点, 从所有的叶节点开始向根节点传递消息, 直到根节点收到所有邻接结点的消息;

2. 从根节点开始向叶节点传递消息, 直到所有叶节点均接收到消息.

精确推断的缺点也很明显: 计算开销随着极大团规模的增长呈指数增长.

[3]最直观的推断算法.

14.5 近似推断

精确推断方法需要很大的计算开销, 因此实际中常使用近似推断方法, 在较低时间复杂度情况下获得原问题近似解. 近似推断方法可以分为两类:

- 采样 (sampling), 通过使用随机化方法完成近似;
- 使用确定性近似完成近似推断.

14.5.1 MCMC 采样

在实际中, 我们关心某些分布并不是对这些分布的本身感兴趣, 而是基于这些分布计算某些期望, 并且还有可能基于这些期望做决策.

具体来说, 假定我们的目标是计算函数 $f(x)$ 在概率密度函数 $p(x)$ 下的期望

$$\mathbb{E}_p[f] = \int f(x)p(x)dx \tag{14.10}$$

则可根据 $p(x)$ 抽取一组样本 $\{x_1, x_2, \ldots, x_N\}$, 然后计算 $f(x)$ 在这些样本上的均值

$$\hat{f} = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{14.11}$$

近似目标期望 $\mathbb{E}[f]$.

对于概率图模型来说, 就是如何高效地基于图模型所描述的概率分布来获取样本, 从而绕开概率分布.

概率图模型中最常用的采样技术就是马尔科夫链蒙特卡洛 (Markov Chain Monte Carlo, MCMC) 方法.

举例来说,

$$P(A) = \int_A p(x)dx \tag{14.12}$$

若有函数 $f : X \to \mathbb{R}$, 则可计算 $f(x)$ 的期望

$$p(f) = \mathbb{E}_p[f(X)] = \int_x f(x)p(x)dx \tag{14.13}$$

MCMC 先构造出服从 p 分布的独立同分布随机变量 $x_i, x_2, \ldots, x_N$, 再得到无偏估计

$$\tilde{p}(f) = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{14.14}$$

.

若概率密度函数 $p(x)$ 很复杂, 则构造服从 p 分布的独立同分布样本很困难.**MCMC 方法的关键就在于通过构造“平稳分布为 p 的马尔科夫链”来产生样本**. 马尔科夫链的平稳条件为

$$p(x^t)T(x^{t-1}|x^t) = p(x^{t-1})T(x^t|x^{t-1}) \tag{14.15}$$

则 $p(x)$ 是该马尔科夫链的平稳分布, 且马尔可夫链在满足该条件时已收敛到平稳状态.

综上所述,MCMC 先构造一条马尔可夫链, 使其收敛至平稳分布恰为待估计参数的后验分布, 然后通过这条马尔可夫链来产生符合后验分布的样本, 并基于这些样本来进行估计.

转移概率的构造至关重要, 不同的转移概率构造产生不同的 MCMC 算法 (见 P333).

14.5.2 变分推断

变分推断通过使用已知简单分布来逼近需推断的复杂分布, 并通过限制近似分布的类型, 从而得到一种局部最优, 但具有确定解的**近似后验分布**.

首先引入概率模型参数估计的例子. 通常以最大化对数似然函数为手段. 同时可使用 EM 算法:

1. E 步基于模型参数求出对隐变量的期望
2. M 步基于 E 步结果最大化模型参数

其中,E 步对 $p(\boldsymbol{z}|\boldsymbol{x}, \Theta^t)$ 的推断很可能因为 $\boldsymbol{z}$ 模型复杂而难以进行. 此时引入变分推断. 通常假设 $\boldsymbol{z}$ 服从分布

$$q(\boldsymbol{z}) = \prod_{i=1}^{M} q_i(\boldsymbol{z}_i) \tag{14.16}$$

即假设 $\boldsymbol{z}$ 可以拆解为一系列互相独立的多变量 $\boldsymbol{z}_i$.(通过平均场方法对隐变量 $\boldsymbol{z}$ 进行推断的方法见 P336). 并可以令 q_i 分布相对简单或有良好的结构.

在实践中使用变分法时, 最重要的是考虑如何对隐变量进行拆解, 以及假设各变量子集服从何种分布. 再结合 EM 算法即可进行概率图模型的推断的参数估计.

14.6 话题模型

有向图模型 | 生成式模型

隐狄利克雷分配模型 (Latent Dirichlet Allocation, LDA) 是话题模型的典型代表.

生成文档 t 的方法为:

1. 根据参数为 α 的狄利克雷分布随机采样一个话题分布 Θ_t;
2. 按照如下步骤生成文档中的 N 个词:
   (a) 根据 Θ_t 进行话题指派, 得到文档 t 中词 n 的话题 $z_{t,n}$
   (b) 根据指派的话题所对应的词频分布 β_k 随机采样生成词

推断参数的过程可见 P339.

总结

D-Seperation 中关于阻塞 (Block) 的定义

看到 PRML 中概率图模型一章, 不太确定 block 的含义, 所以从知乎[4]上摘抄下相关定义.

对于有向无环图, 如果 A, B, C 是三个集合 (可以是单独的节点或者是节点的集合), 为了判断 A 和 B 是否是 C 条件独立的 (也就是 C 发生的时候,A 和 B 是否独立), 我们考虑图中所有 A 和 B 之间的无向路径 (不管箭头朝向, 只要是把 A 和 B 通过几个点最终连接到一起的). 对于其中一条路径, 如果它满足以下两个条件中的任意一条, 则称这条路径是阻塞 (Block) 的:

- 路径中存在某个节点 X 是 *head-to-tail* 或者 *tail-to-tail* 节点, 并且 X 是包含在 C 中的;

[4]https://zhuanlan.zhihu.com/p/22751416

- 路径中存在的某个节点 X 是 *head-to-head* 节点, 并且 X 或者 X 的子节点是不包含在 C 中的;

如果 A,B 间所有的路径都是关于 C 阻塞的, 那么 A,B 就是关于 C 条件独立的; 否则 A,B 不是关于 C 条件独立的.

第 15 章　规则学习

机器学习中的规则学习更像是用形式化的语言归纳出传统的依靠人类经验去对未知样本进行判别的过程. 因此本章内容学习起来更加直观易懂.

15.1　基本概念

- 规则 (rule): 语义明确, 能描述数据分布所隐含的客观规律或领域概念.
- 规则学习 (rule learning): 从训练数据集中学习出一组能用于对未见示例进行判别的规则.

规则学习具有更好的可解释性, 能够更自然地在学习过程中引入领域知识.

规则集合中的每条规则都可看做一个子模型, 规则集合是这些子模型的一个集成. 如果一个示例被多条判别结果不同的规则覆盖时, 则发生了“冲突”(conflict). 解决冲突的过程称为 “冲突消解”(conflict resolution), 常用的方法有投票法, 排序法, 元规则法等.

如果有示例未能被规则所覆盖, 则设置一条 “默认规则”(default rule).

从形式语言表达能力而言, 规则分为两类:

- 命题规则 (Propositional rule): 由源自命题 (propositional atom) 和逻辑连接词与, 或, 非和蕴含构成的简单陈述句.
- 一阶规则 (first-order rule): 基本成分是能描述事物的属性或者关系的“原子公式”(atomic formula).

显然, 一阶规则能够表达更复杂的关系, 因此被称为 “关系型规则”(relational rule).

15.2 序贯覆盖

规则学习的目标是产生一个覆盖尽可能多的样例的规则集. 最直接的做法是 “序贯覆盖”(sequential convering). 形式化地说, 给定正例集合与反例集合, 学习任务是基于候选文字集合 $\mathcal{F} = \{\mathbf{f}_k\}$ 来生成最优规则 $\mathbf{r}$.

穷尽搜索方法在属性和候选值较多时会由于组合爆炸而不可行. 现实任务中一般有两种策略来产生规则:

- “自顶向下”(top-down): 从比较一般的规则开始, 逐渐添加新文字以缩小规则覆盖范围, 知道满足预定条件为止.
- “自底向上”(bottom-up): 从比较特殊的规则开始, 逐渐删除文字以扩大规则覆盖范围, 直到满足条件为止.

15.3 剪枝优化

与决策树剪枝类似, 规则生成剪枝的目的是为了缓解过拟合的风险. 同时需要考虑两个问题:

- 规则生成本质上是一个贪心搜索过程, 是否贪心搜索过程都有陷入过拟合的风险?
- 决策树生成的过程是不是一个贪心搜索过程?

剪枝过程如果发生在规则生长过程中, 则被称作 “预剪枝”, 如果发生在规则产生后, 则被称为 “后剪枝”. 剪枝通常是基于某种性能度量指标来评估增或删除逻辑文字前后的规则性能, 或增或删除规则前后的规则集性能, 从而判断是否需要剪枝.

剪枝还可借助统计显著性检验[1]来进行. 书中举了一个例子 CN2 算法 (见 P352).

后剪枝常用的策略是 “减错剪枝”(Reduced Error Pruning, REP). 基本流程是:

[1]统计显著性检验概念是什么?

1. 将样本集分为训练集和测试集

2. 在训练集上学习得到规则集 $\mathcal{R}$

3. 在每轮迭代中穷举 $\mathcal{R}$ 中所有剪枝可能

4. 在测试集上进行验证选出性能最好的测试集 $\mathcal{R}'$

5. 在 $\mathcal{R}'$ 基础上进行下一轮剪枝, 直到性能不再提升为止.

REP 性能复杂度为 $O(m^4)$.IREP(Incremental REP) 算法将复杂度降到 $O(m \log^2 m)$, 其做法为:

1. 在生成每条规则前, 将样本集分为训练集和测试集

2. 在训练集上生成一条规则 $\mathbf{r}$

3. 立即在验证集上对 $\mathbf{r}$ 进行 REP 剪枝, 得到 $\mathbf{r}'$

4. 将 $\mathbf{r}'$ 覆盖的样例去除, 在更新的样例集上重复上述过程.

若将剪枝机制与其他一些后处理手段结合起来对规则集进行优化, 则往往能获得更好的效果. 以 RIPPER 为例 (具体算法流程见 P353), 其泛化性能超过很多决策树算法, 而且学习速度也比大多数决策树算法要快, 其根源正是在此.

15.4 一阶规则学习

命题规则学习难以处理对象之间的 “关系”(relation), 而关系信息在很多任务中非常重要. 如果将样例进行一一比较, 并且将比较的结果进行记录, 则生成的记录称为 “关系数据”(relational data), 其中由原样本属性转化而来的原子公式称为 “背景知识”(background knowledge).

- 一阶逻辑表示的样本标记有更高的空间复杂度.
- 如果通过比较关系会不会使得学习结果陷入局部最优?

通常在一阶规则中所有出现的变量都被全称量词限定, 但是在不影响理解的情况下可以进行省略. 此外, 一阶规则学习的优势在于:

1. 具有强大的表达能力

2. 容易地引入领域知识

通常机器学习引入领域知识的方法为: 在现有的属性集上基于领域知识构造出新属性; 基于领域知识设计某种函数来对假设空间加以约束.

FOIL(First-Order Inductive Learner) 是著名的一阶规则, 它是一种遵循序贯覆盖框架且采用自顶向下的规则归纳策略. 采用文字的准则为 FOIL 增益 (FOIL gain)

$$F_Gain = \hat{m}_+ \times (\log_2 \frac{\hat{m}_+}{\hat{m}_+\hat{m}_-} - \log_2 \frac{m_+}{m_+ + m_+}) \tag{15.1}$$

其中 $\hat{m}_+$ 和 $\hat{m}_-$ 分别为增加候选文字后新规则所覆盖的正, 反例数;m_+ 和 m_- 为原规则覆盖的正, 反例数.

采用上述增益的原因为, 关系数据中正例数往往少于反例数, 因此通常对正例应该赋予更多的关注[2].

FOIL 可大致看做命题规则学习与归纳逻辑程序设计之间的过渡, 其自顶向下的规则生成过程不能支持函数和逻辑表达能力嵌套, 但是它是把命题规则学习通过变量替换等操作直接转化为一阶规则学习, 因此比一般归纳逻辑程序设计技术更高效.

15.5 归纳逻辑程序设计 (Inductive Logic Programming, ILP)

IPL 在一阶规则学习中引入了函数和逻辑表达式嵌套. 一方面, 其使得机器学习系统具备了更为强大的表达能力; 另一方面,IPL 可看作使用机器学习技术来解决基于背景知识的逻辑程序归纳.

[2]imbalanced-data 是否可以借鉴这种思想?

15.5.1 最小一般泛化

归纳逻辑程序设计采用自底向上的规则生成策略, 直接将一个或者多个正例所对应的具体事实作为初始规则, 再对规则逐步进行泛化以增加其对样例的覆盖率. 目前的基础技术是 “最小一般泛化”(Least General Generalization, LGG).

给定一阶公式 $\mathbf{r}_1$ 和 $\mathbf{r}_2$,LGG 先找出涉及相同谓词的文字, 然后对文字中每个位置的常量逐一进行考察, 若常量在两个文字中相同则保持不变, 记为 $LGG(t,t) = t$; 否则将它们替换为同一个新变量, 并将该替换应用于公式的所有其他位置.

在归纳逻辑程序设计中, 获得 LGG 后, 可将其看做单条规则加入规则集, 最后再用前几节介绍的技术进一步优化 (如对规则集后剪枝).

15.5.2 逆归结

归结原理与逆归结分别与实际生活中的演绎和归纳相对应. 基于归结原理, 我们可将貌似复杂的逻辑规则与背景知识联系起来化繁为简; 而基于逆归结, 我们可基于背景知识来发明新的概念和关系.

基于以下四种完备的逆归结操作, 可以实现逆归结.

- 吸收 (absorption)
- 辨识 (identification)
- 内构 (intra-construction)
- 互构 (inter-construction)

归结, 逆归结都能容易地扩展为一阶逻辑形式; 与命题逻辑的主要不同之处在于, 一阶逻辑的归结, 逆归结通常需要进行合一置换操作.

- “置换”(substitution) 是用某些项来替换逻辑表达式中的变量.
- “合一”(unification) 是用一种变量置换令两个或者多个逻辑表达式相等.

逆归结的一大特点是能自动发明新谓词, 这些新谓词可能对应于样例属性和背景知识中不存在的新知识, 对知识发现与精化有重要意义.

第 16 章　强化学习

16.1　任务与奖赏

在对机器做操作的过程中, 仅仅能通过操作获得当前反馈, 而难以判断最终奖赏的结果, 对这个过程进行抽象, 便得到了 “强化学习” 的模型. 强化学习任务通常用马尔可夫决策过程 (Markov Decision Process, MDP) 来描述: 机器处于环境 E 中, 状态空间为 X, 每个状态 $x \in X$ 为机器对当前环境的描述; 机器所采取的动作空间为 A, 每个动作 $a \in A$ 作用在当前状态 x 上, 使得状态以概率 P 转移到另一个状态, 环境根据 “奖赏”(reward) 函数 R 反馈给机器一个奖赏. 综上, 强化学习对应四元组 $E =< X, A, P, R >$. 需要注意的是, 在环境中状态的转移, 奖赏的返回是不受机器控制的, 机器只能通过选择要执行的动作来影响环境, 也只能通过观察转移后的状态和返回的奖赏来感知环境.

机器要做的事情就是在环境中不断尝试而学得一个 “策略”(policy)π, 根据这个策略, 在状态 x 下就能得知要执行的动作 $a = \pi(x)$. 一种策略表示是函数 $\pi : X \mapsto A$; 另一种策略表示是 $\pi : X \times A \mapsto \mathbb{R}$. 策略的优劣取决于长期执行这一策略后得到的累积奖赏. 在强化学习任务中, 学习的目的就是要找到能够使长期累积奖赏最大化的策略.

强化学习与有监督学习不同之处在于, 在强化学习中并没有监督学习中的有标记样本, 只有在学习过程最后得到最终结果. 因此, 强化学习在一定程度上可以看做 “延迟标记信息” 的监督学习问题.

16.2 K-摇臂赌博机

16.2.1 探索与利用

欲最大化单步奖赏需要考虑两个方面: 一是需要知道每个动作带来的奖赏, 二是要执行奖赏最大的动作.

对于单步强化学习任务对应的 "K-摇臂赌博机"(K-armed bandit), 有两种策略:"仅探索"(expliration-only) 法, 将所有的尝试机会平均分给每个摇臂, 最后以每个摇臂各自的平均吐币概率作为其奖赏期望的近似估计."仅利用"(exploitation-only) 法, 按下目前最优的摇臂, 若有多个摇臂同为最优, 则从中随机选取一个.

"探索" 和 "利用" 这两者是互相矛盾的, 这就是强化学习所面临的 "探索-利用窘境"(Exploration-Exploitation dilemma). 欲使得累计奖赏最大, 必须在探索与利用之间达成较好的折中.

16.2.2 ϵ-贪心

ϵ-贪心基于一个概率来对探索和利用进行折中: 每次尝试时, 以 ϵ 的概率进行探索, 以 $1-\epsilon$ 的概率进行利用.

16.2.3 Softmax

Softmax 算法基于当前已知的摇臂平均奖赏来对探索和利用进行折中. 若各摇臂的平均奖赏相当, 则选取各摇臂的概率也相当; 若某些摇臂的奖赏明显高于其他摇臂, 则它们被选取的概率也明显要高. 概率分配是基于 Boltzmann 分布:

$$P(k)=\frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^{K}e^{\frac{Q(i)}{\tau}}}$$

其中,$Q(i)$ 记录当前摇臂的平均奖赏;$\tau>0$ 称为 "温度",τ 越小则平均奖赏高的摇臂被选取的概率越高.

上述的 ϵ-贪心和 Softmax 算法都是针对单步强化学习问题的解决方案. 如果简单地将多步强化学习看做多个单步强化学习的叠加, 则会有很多局限性. 因为它没有考虑强化学习任务中马尔可夫决策过程的结构.

16.3 有模型学习

如果任务对应的马尔科夫决策过程四元组 $E =< X, A, P, R >$ 均已知，则称为“模型已知”，模型已知环境中的学习称为“有模型学习”（model-based learning）。

16.3.1 策略评估

状态值函数（state value function）$V(\cdot)$ 评估的是从状态值 x 出发，使用策略 π 所带来的累积奖赏。

状态-动作值函数 (state-action value function)$Q(\cdot)$ 评估的是从状态值 x 出发，执行动作 a 后再使用策略 π 所带来的累积奖赏。

由于 MDP 具有马尔可夫性质，即系统下一时刻的状态仅由当前时刻的状态决定，于是值函数有很简单的递归形式。

16.3.2 策略改进

对策略进行评估的最终目的是为了判定其是否是最优策略，如果不是最优策略则需要对策略进行改进。理想的策略应该能最大化累积奖赏。

$$\pi^* = \arg\max_{\pi} \sum_{x \in X} V^{\pi}(x) \tag{16.1}$$

一个强化学习任务可能有多个最优策略，最优策略对应的值函数 V^* 称为最优值函数，即

$$\forall x \in X V^*(x) = V^{\pi^*}(x) \tag{16.2}$$

当策略空间无约束时 V^* 才是最优策略对应的值函数。如果策略空间有约束，则违背约束的策略是不合法的。

如果当前动作不是最优策略，则改进方式为：将策略选择的动作改变为当前最优的动作。

16.3.3 策略迭代与值迭代

求解最优解的方法：从一个初始策略出发，先进行策略评估，然后改进策略，评估改进的策略，再进一步改进策略……不断迭代改进和评估，直到

算法收敛为止。

策略迭代在每次改进策略后都需要重新进行策略评估，这比较耗费时间。

16.4 免模型学习

在现实的学习任务中，环境的转移概率、奖赏函数往往很难得知，如果算法不依赖于环境建模，则称为“免模型学习”（model-free learning)。

16.4.1 蒙特卡罗强化学习

在免模型学习情形下，最大的问题就是策略无法评估。此时，只能通过在环境中执行不同的动作，来观察转移的状态和得到的奖赏。一种直接的策略评估替代方法是多次“采样”，然后求取平均累积奖赏来作为期望累积奖赏的近似，这称为蒙特卡罗强化学习。

策略迭代算法估计的是状态值函数 V，而最终的策略是通过状态-动作值函数 Q 来获得。于是，需要将估计对象从 V 转变为 Q，即估计每一对“状态-动作”的值函数。

由于在免模型学习情况下，只能从一个起始状态开始探索环境，因此只能在探索的过程中逐渐发现各个状态并估计各状态-动作对的值函数。

如果较好地获得值函数的估计，就需要多条不同的采样轨迹。获取不同采样轨迹的方法为（以 $\epsilon-$ 贪心算法为例）：以 ϵ 的概率从所有动作中均匀随机选取一个，以 $1-\epsilon$ 概率选取当前最优动作。

如果被评估的策略与被改进的策略是同一个策略，则称之为“同策略”（on-policy）蒙特卡罗强化学习算法。然而，引入 $\epsilon-$ 贪心算法是为了便于策略评估，实际上我们需要改进的策略是原始策略。为达到这一目的，可以借鉴以下思想：

一般的，函数 f 在概率分布 p 下的期望

$$\mathbb{E}[f] = \int_x p(x)f(x)dx \tag{16.3}$$

可通过从概率分布 p 上的采样 $\{x_1, x_2, \ldots, x_m\}$ 来估计 f 的期望，即

$$\hat{\mathbb{E}}[f] = \frac{1}{m}\sum_{i=1}^{m} f(x_i) \tag{16.4}$$

若引入另一个分布 q，则函数 f 在概率分布 p 下的期望也可等价地写为

$$\mathbb{E} = \int_x q(x)\frac{p(x)}{q(x)}f(x)dx \tag{16.5}$$

上式可以通过在 q 上的采样 $\{x'_1, x'_2, \ldots, x'_m\}$ 估计为

$$\hat{\mathbb{E}}[f] = \frac{1}{m}\sum_{i=1}^{m} \frac{p(x'_i)}{q(x'_i)}f(x'_i) \tag{16.6}$$

.

同样的，将上述分布 p, q 替换为对应的策略，就可以得到“异策略”(off-policy）蒙特卡罗强化学习算法。(见 P386)

16.4.2 时序差分学习

蒙特卡罗强化学习算法通过考虑采样轨迹，克服了模型未知给策略估计造成的困难。但是蒙特卡洛学习算法没有充分利用强化学习任务的 MDP 结构。时序差分（Temporal Difference, TD）学习则结合了动态规划与蒙特卡罗方法的思想，能做到更搞笑的学习。其本质就是在于将状态动作对的更新变成增量式的方式进行。(见 P387)

典型代表算法有 Sarsa 算法，Q-学习算法。

16.5 值函数近似

前面所述的算法都是针对有限状态空间，现实的强化学习任务所面临的状态空间往往是连续的，有无穷多个状态。一个直接的想法就是对状态空间进行离散化，将连续状态空间转化为有限离散状态空间，但是这仍然是一个很难的问题。

事实上，不妨对连续状态空间的值函数进行学习。先 ** 假定状态空间为 n 维实数空间 $X = \mathbb{R}^n$, 并且值函数能表达为状态的线性函数 **

$$V_\theta(\mathbf{x}) = \theta^T\mathbf{x} \tag{16.7}$$

其中 $\mathbf{x}$ 为状态向量, θ 为参数向量。由于此时的值函数难以像有限状态那样精确记录每个状态的值，因此这样值函数的求解被称为值函数近似（value function approximation）。

16.6 模仿学习

强化学习的经典任务设置中，机器所能获得的反馈信息仅有多步决策后的累积奖赏。如果在决策过程中参考决策范例进行决策的方式称为“模仿学习”（imitation learning）。

16.6.1 直接模仿学习

直接模仿学习通过将状态动作轨迹上的状态动作对抽取出来，构造出一个新的数据集合，然后使用分类或者回归算法学习得到新的模型，这种方式称为直接模仿学习。

16.6.2 逆模仿学习

通常设计奖赏函数非常困难，从人类专家提供的范例数据中反推出奖赏函数有助于解决该问题，这就是“逆强化学习”（inverse reinforcement learning）。

逆强化学习的基本思想是：欲使机器做出与范例一致的行为，等价于在某个奖赏函数的环境中求解最优策略，该最优策略所产生的轨迹与范例数据一致。

总结

本章第一节首先介绍了强化学习系统的主要构成: 环境 E; 状态集合 X, 其中的每一个状态 $x \in X$ 都是对环境 E 的抽象; 动作集合 A, 其中 $a \in A$ 是在所采取的动作; 奖励 R 是采取某一个动作后所获得的奖赏; 转移函数 P 描述了从当前状态转移到另一个状态的概率分布. 从整体上来看, 整个强化学习系统由四元组 $E =< X, A, P, R >$ 组成. 从学习过程上来看, 由于在初

始时刻很难确定样本所对应的标签, 只能根据当前的环境和状态确定动作获取环境的反馈, 经过若干轮反馈后才能获取到最终的标签. 所以强化学习又被看做有延迟标记的有监督学习. 学习的结果是输出策略 π, 根据这个策略就能得到下一步动作 $a = \pi(x)$. 评价策略好坏的指标是累计奖赏, 能使长期累积奖赏最大化的策略就是我们的目标策略.

本章第二节将强化学习限定到单步强化学习的情形, 并借助于 K-摇臂赌博机这一理论模型引出了探索-利用窘境. 为了能在探索和利用之间取得一个较好的折中, 常用 ϵ-贪心法和 Softmax 算法.ϵ-贪心算法以一定的概率 ϵ 进行探索, 以概率 $1-\epsilon$ 进行利用;Softmax 算法基于当前已知的摇臂平均奖赏对探索利用进行折中. 两种算法的优劣主要取决于具体应用. 如果要拓展到多步强化学习的情形, 可以基于单步强化学习的情形进行简单叠加, 将每一步的动作所对应的奖赏进行叠加得到长期累计奖赏. 但是这样做并没有考虑到强化学习中马尔科夫决策过程的结果, 具有很大的局限性.

第三节主要介绍了多步强化学习的情形, 并假设四元组 $E =< X, A, P, R >$ 均已知即*模型已知*情形. 这种学习被称为*有模型学习*. 对于学习到的策略 π, 需要有一个评价标准来判定学习结果的好坏. 通常使用状态值函数 $V(\cdot)$ 评价对当前状态使用策略所得到的累计奖赏; 使用状态-动作函数 $Q(\cdot)$ 评价对当前状态 x 使用动作 a 之后得到的累计奖赏. 由于模型已知, 因此可以对状态函数和状态-动作函数执行全概率展开[1]. 在有了策略评价标注以后, 就可以使用优化算法对策略进行改进.[2]有了优化算法以后, 就可以对策略进行迭代优化.

第四节将上一节中的模型已知情形进行拓展到*免模型学习*情形. 首先遇到的问题就是策略无法评估, 进而导致无法做全概率展开. 另一个问题是策略迭代算法估计的是状态值函数 V, 而最终的策略是通过状态-动作函数 Q 来获得. 因此在这种情况下, 就直接对状态-动作函数 Q 进行估计. 此外, 还需要在采样过程中采样出不同的状态轨迹, 以学习到不同状态-动作对. 如果被评估与被改进的是同一个策略, 则被称为通策略蒙特卡罗强化学习算法; 借助于重要性采样原理, 还可以使得被评估和被采样策略不同, 这种算法被称为异策略蒙特卡罗学习算法. 蒙特卡罗学习算法是在采样得到整个

[1]TODO: 此处待展开描述

[2]TODO:Bella 等式? 最优 Bella 等式?

采样轨迹后进行学习, 时序差分学习算法则是使用增量式方法进行更新.

第五节将有限状态空间中的强化学习拓展到无线状态空间中, 并提出了函数值金丝方法.

第六节则考虑借助于人类的专家知识进行模仿学习. 一种方法是直接模仿学习, 即借助于人类专家的决策轨迹数据进行训练. 另一种方法是从人类专家的决策轨迹数据中反推出奖赏函数.