机器学习

第一章 绪论

1.1.引言

- 计算机科学是研究关于算法的学问，机器学习是研究关于学习算法的学问

1.2.基本术语

- 数据集(data set) 是一组记录的集合
- 示例(instance)样本(sample) 是对一个事件或对象的描述
- 属性(attribute)特征(feature) 是反映事件或对象在某方面的表现或性质的事项
- 属性值(attribute value) 属性上的取值
- 属性空间(attribute space)样本空间(sample space) 属性张成的空间
- 特征向量(feature vector) 每个示例都可以在样本空间里找到对应的坐标位置，由于空间的每个点也对应一个坐标向量，一个实例也可以叫做一个特征向量
- 维数(dimensionality) 属性的个数
- 学习(learning)训练(training) 从数据中学得模型的过程
- 训练数据(training data) 训练过程中使用的数据
- 训练样本(training sample) 训练数据中的具体一个样本
- 训练集(training set) 训练样本组成的集合
- 假设(hypothesis) 学得模型对应了关于数据的某种潜在规律
- 真相、真实(ground-truth) 潜在规律本身
- 学习器(learner) 具体学习算法在给定数据和参数空间上的实例化
- 预测(prediction) 模型基于训练样本的得到的某种结果
- 标记(label) 关于样本的结果信息
- 样例(example) 拥有标记的样本
- 标记空间、输出空间(label space) 所有标记的集合
- 有监督任务(supervised learning) 有标记信息的数据集
    1. 分类(classification) 预测结果是离散值
    2. 回归(regression) 预测结果是连续值
- 无监督任务(unsupervised learning) 没有标记信息的数据集
    1. 聚类(clustering) 将训练集分组，每组为一个簇(cluster)，簇可能存在潜在的概念划分
- 测试(testing) 学习模型后使用模型预测的过程
- 测试样本(testing sample) 被预测的样本

- 泛化(generalization) 学得模型适用于新样本的能力

1.3.假设空间

- 假设空间(hypothesis space) 所有假设组成的空间
- 版本空间(version space) 与训练集一致的假设集合
    1. 假设空间按照删除与正例不一致（与反例相同）的假设的搜索方式在某个训练集上得到的结果就是版本空间

1.4.归纳偏好

- 归纳偏好(inductive bias) 机器学习算法在学习过程中对某种类型假设的偏好
    1. 例如，在回归学习中，一般认为相似的样本具有相似的输出，我们应该让算法偏向于归纳这种假设
- 没有免费的午餐定理(No Free Lunch Theorem) 所有算法的期望性能和随机胡猜一样

$$\sum_f E_{ote}(\mathcal{E}_a|X,f) = \sum_f E_{ote}(\mathcal{E}_b|X,f)$$

> NFL定理证明

第二章 模型评估与选择

2.1.经验误差与过拟合

- 错误率(error rate) 分类错误的样本占样本总数的比例
- 精度(accuracy) 1-错误率
- 误差(error) 学习器在训练集的实际预测输出与样本的真实输出之间的差异
- 训练误差(training error)经验误差(empirical error) 学习器在训练集上的误差
- 泛化误差(generalization error) 学习器在新样本上的误差
    1. 我们希望得到一个泛化误差小的学习器，然而，我们根据训练样本只能得到一个经验误差很小的学习器
- 过拟合(overfitting) 把训练样本自身的一些特点当作了所有潜在样本具有的某种一般性质，导致泛化性能下降
- 欠拟合(underfitting) 对训练样本的一般性质尚未学习好
    1. 一般情况下，学习能力高低分别会导致过拟合和欠拟合，欠拟合可以通过增加扩展分支（决策树）、训练轮数（神经网络），过拟合则是机器学习的主要障碍，过拟合是无法避免的

2.2.评估方法

- 测试集(testing set) 测试学习器对新样本的判别能力
- 测试误差(testing error) 用作泛化误差的近似
    1. 我们一般假设测试样本是从真实样本分布中独立同分布采样而得。测试集应该和训练集互斥

2.2.1.留出法(hold-out)

- 将数据集D划分为互斥的两个集合，其中一个为训练集S，另一个为测试集T，即$D = S \cup T$, $S \cap T = \varnothing$,基于S训练出的模型，用T来评估测试误差，作为泛化误差的估计
    1. 按照采样(sampling)的角度看待数据集的划分过程，则保留类别比例的采样方式通常为分层采样(stratified sampling)
    2. 训练集S过大会导致接近数据集D、测试集T较小，评估结果不够准确；S过小会导致基于S和D训练出模型的差别过大，降低评估结果的保真性(fidelity)
    3. 通常取2/3~4/5的样本作为训练集

2.2.2.交叉验证法(cross validation)

- 将数据集D划分为k个大小相同的互斥子集，即 $D = D_1 \cup D_2 \cup \ldots \cup D_k, D_i \cup D_j = \varnothing (i \neq j)$ 。每个子集D_i 都保持数据分布的一致性，即按照分层采样得到。每次取k -1个作为训练集，其余作为测试集，进行k次训练和测试，返回k个结果的均值
    1. 交叉验证法评估结果依赖于k的取值，因此也叫做“k折交叉验证”(k-fold cross validation)
    2. k通常取值为10
    3. 若数据集D包含m个样本，令k=m，得到交叉验证法的一个特例：留一法(Leave-One-Out)

2.2.3.自助法(bootstrapping)

- 自助法以自助采样法(bootstrap sampling)为基础，在一个含有m个样本的数据集D中，每次随机抽取一个样本拷贝入D' (D' 中有重复的样本)，重复执行m次，使得D' 中也有m个样本
    1. 样本在m次采样始终采样不到的概率为$(1-\frac{1}{m})^m$，极限是 $\lim\limits_{m \to \infty}(1-\frac{1}{m})^m = \frac{1}{e} \approx 0.368$
    2. 使用始终没有被采样的部分作为测试集
    3. 这样的测试结果，叫包外估计(out-of-bagestimate)
    4. 自助法适用于较小和难以划分的数据集，对集成学习有很大好处

2.2.4.调参与最终模型

- 参数的取值范围是在实数范围，参数的选择一般是按照取值范围和步长
    1. 超参数:数目通常在10以内，采用人工设定
    2. 模型的参数:比如神经网络中的参数，采用学习（比如训练轮数）
- 包含m个样本的数据集D，在模型评估选择时，选用了部分数据做训练，另一部分做评估，在学习算法和参数配置选定以后，需要重新将所有的样本，即数据集D进行训练，得到的模型交给用户
- 模型在实际使用的过程中遇到的数据称为测试数据
- 模型评估与选择中用于评估测试的数据叫做验证集(validation set)，比如，研究算法泛化性能时，我们测试集估计实际使用的泛化能力，而把训练数据划分为训练集和验证集，基于验证集进行模型选择和调餐参数

2.3.性能度量(performance measure)

- 性能度量(performance measure) 衡量模型泛化性能的评价标准

---

1. 回归任务的性能度量是均方误差(mean squared error)

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

2. 对于数据分布$\mathcal{D}$和概率密度函数$p(\cdot)$，均方误差的描述为

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

2.3.1.错误率与精度

- 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

---

1. 样例集D分类错误率定义为

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

精度定义为

$$acc(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(f(\mathbf{x\_i})=y_i)=1-E(f;D)$$

2. 对于数据分布$\mathcal{D}$和概率密度函数$p(\cdot)$，错误率定义为

$$E(f;D)=\int_{\mathbf{x}\sim\mathcal{D}}\mathbb{I}(f(\mathbf{x})\neq y)p(\mathbf{x})d\mathbf{x}$$

精度定义为

$$acc(f;D)=\int_{\mathbf{x}\sim\mathcal{D}}\mathbb{I}(f(\mathbf{x})=y)p(\mathbf{x})d\mathbf{x}=1-E(f;D)$$

2.3.2.查准度(precision)、查全率(recall)与F1

- 样例根据其真实类别和学习器预测类别的组合为真正例(true positive)、假正例(false positive)、真反例(true negative)、假反例(false negative)

---

1. 查准率定义为

$$P=\frac{TP}{TP+FP}$$

2. 查全率定义为

$$R=\frac{TP}{TP+FN}$$

- 查准率P和查全率R是一对矛盾的度量
- 以查准率为纵轴、以查全率为横轴作图，得到查准率-查全率曲线，简称“P-R曲线”，显示曲线的的图称为“P-R图”
- 一个学习器P-R曲线完全“包住”另一个的，则说明前者性能好与后者性能
- 当查准率P=查全率R时的取值，叫做平衡点(Break-Even Point)

- 由于BEP过于简化，更常用的是$F1$度量

---

1. $F1$度量定义为

$$F1=\frac{2\times P\times R}{P+R}=\frac{2\times TP}{\text{样例总数}+TP-TN}$$

2. $F1$是基于查准率和查全率的平均调和平均定义的

$$\frac{1}{F1}=\frac{1}{2}\cdot(\frac{1}{P}+\frac{1}{R})$$

3. F1的一般形式是F_β，F_β定义为

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

4. 当$\beta=1$时，退化为$F1$；$\beta>1$时，查全率有更大的影响；$\beta<1$时，查准率有更大的影响

- 在n个二分类混淆矩阵上宏查准率、宏查全率和宏F1定义为

---

1. 宏查准率

$$macro-P=\frac{1}{n}\sum_{i=1}^{n}P_i$$

2. 宏查全率

$$macro-R=\frac{1}{n}\sum_{i=1}^{n}R_i$$

3. 宏F1

$$macro-F1=\frac{2\times macro-P\times macro-R}{macro-P+macro-R}$$

- 在n个二分类混淆矩阵上将TP、FP、TN、FN进行平均$\overline{TP}$、$\overline{FP}$、$\overline{TN}$、$\overline{FN}$进而得到微查准率、微查全率和微F1

---

1. 微查准率

$$micro-P=\frac{\overline{TP}}{\overline{TP}+\overline{FP}}$$

2. 微查全率

$$micro-R=\frac{\overline{TP}}{\overline{TP}+\overline{FR}}$$

3. 微F1

$$micro-F1=\frac{2\times micro-P\times micro-R}{micro-P+micro-R}$$

2.3.3.ROC与AUC

- ROC曲线是衡量学习器泛化性能的有力工具，其中ROC曲线的纵轴是真正例率$TPR=\frac{TP}{TP+FN}$、横轴是假正例率$FPR=\frac{FP}{TN+FP}$

---

1. AUC(Area Under ROC Curve)面积可以反映学习器性能

$$\begin{aligned}AUC&=\frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1}-x_i)\cdot(y_i+y_{i+1})\\&=1-\ell_{rank}\end{aligned}$$

2. m^+个正例、m^-个反例，D^+为正例集合、D^-为反例集合排序的损失，AUC和Mann-Whitney U检验等价

$$\ell_{rank}=\frac{1}{m^+m^-}\sum_{m^+\in D^+}\sum_{m^-\in D^-}\left(\mathbb{I}(f(x^+)<f(x^-)+\frac{1}{2}\mathbb{I}(f(x^+)=f(x^-))\right)$$

> ROC曲线画法

2.3.4.代价敏感错误率与代价曲线

- 为了权衡不同类型错误所造成的不同损失，可为错误赋予非均等代价(unequal cost)
- 在考虑非均等代价下，我们希望最小化总体代价(total cost)，以二分类问题为例，

  此时错误率为

$$E(f;D;cost)=\frac{1}{m}\left(\sum_{x_i\in D^+}\mathbb{I}(f(\mathbf{x\_i})\neq y_i)\times cost_{01}+\sum_{x_i\in D^-}\mathbb{I}(f(\mathbf{x\_i})\neq y_i)\times cost_{10}\right)$$

- 在非均等代价下，ROC不能直接反映学习器的期望总体代价，而需要通过代价曲线(cost curve)

---

1. 代价曲线横轴是取值[0,1]的正例概率代价

$$P(+)cost=\frac{p\times cost_{01}}{p\times cost_{01}+(1-p)\times cost_{10}}$$

2. 纵轴是取值为[0,1]归一化代价

$$cost_{norm}=\frac{FNR\times p\times cost_{01}+FPR\times(1-p)\times cost_{10}}{p\times cost_{01}+(1-p)\times cost_{10}}$$

- 每个ROC曲线上的一点转化为代价平面上的一条线段，取所有线段的下界围成的面积为期望的总体代价

2.4.比较检验

- 使用统计假设检验(hypotheis test)进行学习器性能的比较

2.4.1.假设检验

---

1. 在包含m个样本的测试集上，泛化错误率为ϵ的学习器被测得测试错误率$\hat{\epsilon}$的概率是

$$P(\hat{\epsilon};\epsilon)=\begin{pmatrix}m\\ \hat{\epsilon}\times m\end{pmatrix}\epsilon^{\hat{\epsilon}\times m}(1-\epsilon)^{m-\hat{\epsilon}\times m}$$

2. 考虑$\epsilon\leq\epsilon_0$，则在$1-\alpha$的概率内所观测到最大错误率

$$\bar{\epsilon}=max\epsilon\ \ s.t.\ \ \sum_{i=\epsilon_0\times m+1}^{m}\binom{m}{i}\epsilon^i(1-\epsilon)^{m-i}<\alpha$$

3. k个测试错误率的平测试错误率μ和方差σ^2分别为

$$\mu=\frac{1}{k}\sum_{i=1}^{k}\hat{\epsilon}_i\qquad\sigma^2=\frac{1}{k-1}\sum_{i=1}^{k}(\hat{\epsilon}_i-\mu)^2$$

4. k个测试错误率看作泛化错误率ϵ_0的独立采样，变量$\tau_t=\frac{\sqrt{k}(\mu-\epsilon_0)}{\sigma}$服从自由度为$k-1$的$t$分布

2.4.2.交叉验证t检验

- 用于比较不同学习器的性能
- 学习器A和学习器B使用k折交叉验证法得到测试错误率分别为$\epsilon_1^A,\epsilon_2^A,.....,\epsilon_k^A$和$\epsilon_1^B,\epsilon_2^B,.....,\epsilon_k^B$，可以使用k折交叉验证“成对t检验”(paired t-tests)进行比较检验，基本思想是如果两个学习器性能相同，则使用相同数据集测试错误率应该相同，即$\epsilon_i^A=\epsilon_i^B$分别对每对测试错误率求差$\Delta_i=\epsilon_i^A-\epsilon_i^B$，（如果两个学习器性能相同它们的差值应该为零），计算均值μ和方差σ^2，在显著度α下，变量$\tau_t=\left|\frac{\sqrt{k}\mu}{\sigma}\right|$小于临界值$t_{\alpha/2,\,k-1}$则假设不能被拒绝，说明学习器A和学习器B的性能相当；如果不一样，则认为平均错误率小的性能更好
- 以上需要满足测试错误率均为泛化错误率的独立采样，但是通常样本有限，训练集会有重叠，即测试错误率不独立，会过高估计假设成立的概率，因此，可以采用“5×2交叉验证”法5×2交叉验证是作5次2折交叉验证，每次2折交叉验证之前随机将数据打乱，使得5次交叉验证数据划分不重复，对学习器A和B的第i次的两个测试错误率求差，计算平均值$\mu=0.5(\Delta_1^1+\Delta_1^2)$，对每次2折实验的结果都计算出其方差$\sigma_i^2=(\Delta_i^1-\frac{\Delta_i^1+\Delta_i^2}{2})^2+(\Delta_i^2-\frac{\Delta_i^1+\Delta_i^2}{2})^2$，变量$\tau_t=\mu/\sqrt{0.2\sum_{i=1}^{5}\sigma_i^2}$

2.4.3.McNemar检验

- 对于二分类问题，使用留出法可以得到分类器分类结果的差别，如果两个学习器性能相同，则$e_{01}=e_{10}$，那么变量$|e_{01}-e_{10}|$应当服从正态分布。McNemar检验考虑变量$\tau_{\chi^2}=\frac{(|e_{01}-e_{10}|-1)^2}{e_{01}+e_{10}}$在给定显著度$\alpha$，如果变量小于临界值$\chi_\alpha^2$则假设不能被拒绝，说明学习器A和学习器B的性能相当；如果不一样，则认为平均错误率小的性能更好

2.4.4.Friedman检验与Nemenyi后续检验

- 在一组数据集上对多个算法性能比较，既可以在每个数据集上分别列出不同算法两两比较，也可以使用基于算法排序的Friedman检验就可以满足我们的要求

---

1. 使用留出法或者交叉验证法得到每个算法在每个数据集上的测试结果，然后根据测试的性能进行排序，并计算平均序值
2. 使用Friedman检验算法性能假设N个数据集，k个算法，令r_i表示第i个算法的平均序值，变量

$$\tau_{\chi^2} = \frac{k-1}{k} \cdot \frac{12N}{k^2-1} \sum_{i=1}^{k} \Big(r_i - \frac{k+1}{2}\Big)^2$$
$$= \frac{12N}{k(k+1)} \Big(\sum_{i=1}^{k} r_i^2 - \frac{k(k+1)^2}{4} \Big)$$

但是“原始Friedman检验”过于保守，通常使用变量

$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1)-\tau_{\chi^2}}$$

3. 若性能显著不同，此时需要进行后续检验(post-hoc test)进行进一步检验，这里我们使用Nemenyi后续检验，Nemenyi检验计算出平均序值差别的临界值域为

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

4. 如果两个算法的平均序值之差超出临界值域CD，则认为两个算法的性能有显著差异

2.5.偏差与方差

- 偏差-方差分解(bias-variance decomposition)是解释学习算法泛化性能的一种工具

---

1. 回归任务的算法期望为

$$\bar{f}(x) = \mathbb{E}_D[f(x;D)]$$

2. 使用样本相同的不同算法产生的方差为

$$var(x) = \mathbb{E}_D\Big[\big(f(x;D) - \bar{f}(x)\big)^2\Big]$$

3. 噪声为

$$\varepsilon^2 = \mathbb{E}_D\Big[(y_D - y)^2\Big]$$

4. 期望输出和真实标记输出的偏差

$$bias^2(x) = (\bar{f}(x) - y)^2$$

5. 算法的期望泛化误差（假定噪声期望为0）

$$E(f;D) = \mathbb{E}_D\big[(f(x;D) - y_D)^2\big] = bias^2(x) + var(x) + \varepsilon^2$$

- 偏差度量了学习算法期望预测与真实结果的偏离程度，即刻画了学习算法本身的拟合能力
- 方差度量了同样大小训练集的变动导致的学习性能的变化，即刻画数据扰动所造成的影响
- 噪声则表达了当前任务学习算法所能达到的期望泛化误差的下界，即刻画问题本身的难度
- 偏差和方差是有冲突的，称为偏差-方差窘境(bias-variance dilemma)

第三章 线性模型

3.1.基本形式

- 给定由d个属性描述的示例$x=(x_i;x_2;\ldots;x_d)$，其中x_i是x在第i个属性的取值，线性模型(linear model)是试图学得一个通过属性的线性组合来预测的函数，即

$$f(x)=w_1x_1+w_2x_2+\ldots+w_dx_d+b$$

- 一般也可用向量形式写成，即

$$f(x)=w^Tx+b$$

其中$w=(w_1;w_2;\ldots;w_d)$，当w和b学得以后就可以确定模型了

- 功能强大的非线性模型(nonlinear model)可以在线性模型的基础上通过引入层级结构或高维映射而得
- w直观表达了各属性在预测中的重要性(权重)，因此线性模型具有很好的可解释性(comprehensibility)

3.2.线性回归

- 给定数据集$D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$，其中$x_i=(x_{i1};x_{i2};\ldots;x_{id}),y_i\in\mathbb{R}$。线性回归(linear regression)试图学习一个线性模型尽可能准确的预测实值输出标记
- 对于离散属性，若属性值间存在“序”(oeder)关系，可以通过连续化将其转化为连续值，例如，二值属性身高的取值高矮可以转化为{1.0,0.0}
- 若属性之间没有序关系，假定有k个属性值，可以转化为k维向量，例如，瓜类的取值有西瓜、黄瓜和南瓜可以转化为(0,0,1),(0,1,0),(1,0,0)
- 线性回归试图学得

$$f(x_i)=wx_i+b,\text{使得}f(x_i)\simeq y_i$$

- 均方误差是回归任务最常用的性能度量

---

1. 令均方误差最小化，即

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

2. 均方误差对应了欧式距离(Euclidean distance)，因此拥有非常好的几何意义
3. 基于均方误差最小化的方法叫做最小二乘法(least square method)，最小二乘法就是试图找到一条直线，使得所有样本到直线上的欧式距离之和最小
4. 求解$E_{(w,b)}=\sum_{i=1}^m(y_i-wx_i-b)^2$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)，可以对w和b分别求偏导，得到

$$\begin{aligned}\frac{\partial E_{(w,b)}}{\partial w}&=2\Big(w\sum_{i=1}^mx_i^2-\sum_{i=1}^m(y_i-b)x_i\Big)\\\frac{\partial E_{(w,b)}}{\partial b}&=2\Big(mb-\sum_{i=1}^m(y_i-wx_i)\Big)\end{aligned}$$

令上式为0，得到w和b的最优解的闭式解

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

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

- 更一般的情况是在数据集D中，样本由d个属性描述，此时多元线性回归(multivariate linear regression)试图学得

$$f(x_i)=w^Tx_i+b,\ 使得\ f(x_i)\simeq y_i$$

- 对于多元线性回归，也可利用最小二乘法对w和b进行估计

---

1. 将w和b吸收入向量形式$\hat{w}=(w,b)$，相应的，把数据集D表示一个$m\times(d+1)$大小的矩阵$\mathbf{X}$，其中的每行对应一个示例，该行的前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

$$\mathbf{X}=\begin{pmatrix}x_{11}&x_{12}&\cdots&x_{1d}&1\\x_{21}&x_{22}&\cdots&x_{2d}&1\\\vdots&\vdots&\ddots&\vdots&\vdots\\x_{m1}&x_{m2}&\cdots&x_{md}&1\end{pmatrix}=\begin{pmatrix}x_1^T&1\\x_2^T&1\\\vdots&\vdots\\x_m^T&1\end{pmatrix}$$

   将标记也写成向量形式$y=(y_1;y_2;\ldots;y_m)$，于是有

$$\hat{w}^*=\arg\min_{\hat{w}}(y-\mathbf{X}\hat{w})^T(y-\mathbf{X}\hat{w})$$

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

$$\frac{\partial E\hat{w}}{\partial\hat{w}}=2\mathbf{X}^T(\mathbf{X}\hat{w}-y)$$

2. 令上式为0，得到$\hat{w}$的最优解的闭式解，当$\mathbf{X}^T\mathbf{X}$为满秩矩阵或者正定矩阵时

$$\hat{w}^*=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^Ty$$

   令$\hat{x}_i=(x_i;1)$，此时学得的线性回归模型是

$$f(\hat{x}_i)=\hat{x}_i^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^Ty$$

   当矩阵不是满秩矩阵的时候，可解出多个解，此时作为输出的将由学习算法的归纳偏好决定，常见的做法是引入正则化(regularization)项

- 当我们认为示例对应的输出标记是在指数尺度上的变化，那就可以将输出标记的对数作为线性模型逼近的目标，即

$$\ln y=w^Tx+b$$

  我们称为对数线性回归(log linear regression)，虽然它实际上是e^{w^Tx+b}去逼近y，但在上式形式上还是线性回归，但实质上已经是求输入到输出的非线性映射了

- 更一般的则是广义线性模型(generalized linear model)

$$y=g^{-1}(w^Tx+b)$$

其中，$g(\cdot)$是单调可微的联系函数(link function)，显然，对数线性回归就是广义线性回归在$g(\cdot)=ln(\cdot)$的特例

3.3.对数几率回归

- 当需要进行分类任务的时候，需要找到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
- 例如二分类任务，我们需要的输出标记$y\in\{0,1\}$，而线性回归模型产生的预测值$z=w^Tx+b$是个实值，因此我们需要将实值z转换为$0/1$值，最理想的是单位阶跃函数(unit-step function)或称Heaviside函数

$$y=\begin{cases}0, z<0;\\0.5, z=0;\\1, z>0;\end{cases}$$

即若预测值z大于0就判别为正例，小于0就判别为反例，预测值为临界值0就任意判别

- 但是由于单位阶跃函数不连续，一次不能直接作为广义线性模型中的联系函数$g^-(\cdot)$，于是需要找到一个一定程度上近似单位阶跃函数的替代函数(surrogate function)，并且这个函数单调可微

---

1. 对数几率函数(logistic function)为常用的替代函数

$$y=\frac{1}{1+e^{-z}}$$

将对数几率函数作为$g^-(\cdot)$代入$y=g^{-1}(w^Tx+b)$得到

$$y=\frac{1}{1+e^{-(w^Tx+b)}}$$

继而可变化为

$$\ln\frac{y}{1-y}=w^Tx+b$$

将其中y作为正例的可能性$1-y$作为反例的可能性，两者的比值称为几率(odds)，反映了x作为正例的相对可能性，对几率取对数则得到对数几率(log odds,logit)

- 由此可以发现，利用线性回归模型的预测结果去逼近真实标记的对数几率，因此模型称为对数几率回归(logistic regression)或者逻辑回归(logistic regression)
- 我们将y视为类后验概率估计$p(y=1|x)$，则$\ln\frac{y}{1-y}=w^Tx+b$可以重写为$\ln\frac{p(y=1|x)}{p(y=0|x)}=w^Tx+b$显然有$p(y=1|x)=\frac{e^{w^Tx+b}}{1+e^{w^Tx+b}}$和$p(y=0|x)=\frac{1}{1+e^{w^Tx+b}}$于是，使用极大似然法(maximum likelihood method)进行估计w和b

---

1. 给定数据集$\{(x_i,y_i)\}_{i=1}^m$对数回归模型最大化对数似然(log-likelihood)

$$\ell(w,b)=\sum_{i=1}^m\ln p(y_i|x_i;w,b)$$

其中似然函数是

$$P(w,b)=\prod_{i=1}^m p(y_i|x_i;w,b)$$

即令每个样本属于真实值的概率越大越好

令$\beta=(w;b)$,$\hat{x}=(x;1)$，则w^Tx+b可以简写为

$$\beta^T\hat{x}$$

令$p_1(\hat{x};\beta)=p(y=1|\hat{x};\beta)$,$p_0(\hat{x};\beta)=p(y=0|\hat{x};\beta)=1-p_1(\hat{x};\beta)$，则似然项可以重写为

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

最大化对数似然，得

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

上式等价于最小化$-\ell(\beta)$，即

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

这是一个关于β的高阶可导连续凸函数，根据凸优化理论、梯度下降法(gradient descent method)、牛顿法(Newton method)都可以求最优解

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

> 对数几率回归推导

- 牛顿法求最优解

---

1. 牛顿法第$t+1$轮迭代公式是

$$\beta^{t+1}=\beta^t-\Big(\frac{\partial^2\ell(\beta)}{\partial\beta\partial\beta^T}\Big)^{-1}\frac{\partial\ell(\beta)}{\partial\beta}$$

2. 其中关于β的一阶导数是

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

3. 关于β的二阶导数是

$$\frac{\partial^2\ell(\beta)}{\partial\beta\partial\beta^T}=\sum_{i=1}^{m}\hat{x}_i\hat{x}_i^Tp_1(\hat{x}_i;\beta)(1-p_1(\hat{x};\beta))$$

3.4.线性判别分析

- 线性判别分析(Linear Discriminant Analysis)是一种经典的线性学习方法，也称为Fisher判别分析
- LDA的思想：给定训练样例集，设法将样例投影在一条直线上，使得同类样本的投影点尽可能接近、异类样本的投影点尽可能远离；当出现新的样本时，将其投影在这条直线上，根据投影点的位置确定类别

---

1. 数据集$D=\{(x_i,y_i)\}_{i=1}^m, y_i\in\{0,1\}$，令$X_i$、$\mu_i$、$\sum_i$分别表示第$i\in\{0,1\}$类示例的集合、均值向量、协方差矩阵
2. 将数据投影到直线w上，则两类样本的中心在直线上的投影分别是$w^T\mu_0$和$w^T\mu_1$；若将所有的样本都投影到直线上，两类样本的协方差是$w^T\sum_0 w$和$w^T\sum_1 w$
3. 我们希望同类样例的投影点接近，可以让同类样例的协方差尽可能小，即$w^T\sum_0 w+w^T\sum_1 w$尽可能小
4. 我们还希望异类样例的投影点尽可能远离，我们可以使它们的类中心之间的距离尽可能大，即$||w^T\mu_0-w^T\mu 1||_2^2=w^T(\mu_0-\mu_1)(\mu_0-\mu_1)^Tw$尽可能大
5. 我们同时考虑同类样例协方差小和异类样例类中心距大的目标，即可最大化

$$\begin{aligned}J&=\frac{||w^T\mu_0-w^T\mu_1||_2^2}{w^T\sum_0 w+w^T\sum_1 w}\\&=\frac{w^T(\mu_0-\mu_1)(\mu_0-\mu_1)^Tw}{w^T\sum_0 w+w^T\sum_1 w}\end{aligned}$$

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

$$\begin{aligned}S_w&=\sum_0+\sum_1\\&=\sum_{x\in X_0}(x-\mu_0)(x-\mu_0)^T+\sum_{x\in X_1}(x-\mu_1)(x-\mu_1)^T\end{aligned}$$

   定义类间散度矩阵(between-class scatter matrix)

$$S_b=(\mu_0-\mu_1)(\mu_0-\mu_1)^T$$

   那么

$$J=\frac{w^TS_bw}{w^TS_ww}$$

   这就是LDA最大化的目标，S_b和S_w的广义瑞利商(generalized Rayleigh quotient)
7. 因为J的分子和分母是关于w的二次项，可以将$w^TS_ww=1$，那么J等价于

$$\begin{aligned}&\max_w w^TS_bw\\&s.t.\ w^TS_ww=1\end{aligned}$$

   即

$$\begin{aligned}&\min_w -w^TS_bw\\&s.t.\ w^TS_ww=1\end{aligned}$$

   使用拉格朗日乘子法得

$$S_bw=\lambda S_ww$$

   其中λ是拉格朗日乘子。注意到S_bw的方向恒为$\mu_0-\mu_1$，令$S_bw=\lambda(\mu_0-\mu_1)$

$$w=S_w^{-1}(\mu_0-\mu_1)$$

8. 可以对S_w进行奇异值分解$S_w=U\sum V^T$再由$S_w^{-1}=V\sum^{-1}U^T$得到S_w^{-1}

> LDA证明

- LDA可以推广到多分类问题

---

1. 定义全局散度矩阵

$$\begin{aligned} S_t &= S_b + S_w \\ &= \sum_{i=1}^{m}(x_i - \mu)(x_i - \mu)^T \end{aligned}$$

其中μ是所有示例的均值

2. 将类内散度矩阵S_w重新定义为每个类别的散度矩阵之和

$$S_w = \sum_{i=1}^{N} S_{w_i}$$

其中

$$S_{w_i} = \sum_{x \in X_i}(x - \mu_i)(x - \mu_i)^T$$

由此可得

$$\begin{aligned} S_b &= S_t - S_w \\ &= \sum_{i=1}^{N} m_i(\mu_i - \mu)(\mu_i - \mu)^T \end{aligned}$$

优化目标是

$$\max_W \frac{tr(W^T S_b W)}{tr(W^T S_w W)}$$

其中$W \in \mathbb{R}^{d \times (N-1)}$，$tr(\cdot)$是矩阵的迹

3. 通过广义特征值求解得

$$S_b W = \lambda S_w W$$

4. W的闭式解是$S_w^{-1} S_b$的d'个最大非零广义特征值所对应的特征向量组成的矩阵，$d' \leq N - 1$

- 将W视为一个投影矩阵，则多分类LDA将样本投影到d'维空间，通常d'小于数据原有的属性数d。因此，LDA也是一种监督降维技术

3.5.多分类学习

- 现实中，很多多分类任务可以直接使用二分类的推广，例如LDA的推广
- 考虑N个类别$C_1, C_2, \ldots, C_N$，多分类任务的基本思路是拆解法，即将多分类任务分为多个二分类任务求解
- 分类学习器也称为分类器(classifier)
- 多分类问题的关键是对多分类任务进行拆分和对多个分类器进行集成
- 对于给定数据集$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}, y_i \in \{C_1, C_2, \ldots, C_N\}$，拆分策略有三种

---

1. 一对一(One vs. One，简称OvO)

将N个类别两两配对，从而产生$N(N-1)/2$个二分类任务，例如为区分C_i和C_j训练一个分类器。在测试阶段，将样本同时提交给所有的份分类器，因此得到$N(N-1)/2$个分类结果。最终结果可以投票产生：即预测最多的类别为最终分类结果

2. 一对其余(One vs. Rest，简称OvR，OvR也叫One vs. All)

   将一个类的样例作为正例，其余类的样例作为反例来训练N个分类器。在预测阶段，如果只有一个分类器预测为正类，则对应类别标记为最终结果；如果多个分类器预测为正类，则通常考虑预测置信度最大的类别标记为分类结果

3. 多对多(Many vs. Many，简称MvM)

   每次将若干个类作为正类，若干个其他类作为反类。显然上述的OvO和OvR都是MvM的特例。MvM的正反类构造需有特殊的设计，例如，纠错输出码(Error Correcting Output Codes，简称ECOC)

OvR需要训练N个分类器，OvO则需要训练$N(N-1)/2$个分类器；OvO的存储开销和测试时间较大于OvR；但是训练的时候OvR的每个分类器需要使用全部样本，OvO只需要使用两个类别的样本，在类别很多的时候OvO比OvR的时间开销小

- ECOC的工作过程

---

1. 编码：对N个类别做M次划分，每次划分一部分类别做为正类，一部分划分为反类，从而设计一个二分类训练集；这样一共产生M个训练集，训练M个分类器
2. 解码：使用M个分类器分别对测试样本进行预测，这些预测会组成一个编码。将这个编码和每个类别各自的编码进行比较，其中距离最小的类别为预测类别
3. 类别划分通过编码矩阵(coding matrix)指定。编码矩阵主要有二元码和三元码。二元码将分别为每个类别指定为正类和反类，三元码还将指定除正反类以外的停用类
4. ECOC便编码的纠错能力来源于如果某个分类器出错，根据距离计算决定结果的方式仍能正确判断。一般来说，同一个学习任务，ECOC编码越长，纠错能力越强。但是，ECOC编码越长意味着分类器的数目增加，计算和存储的开销增大，而且有限的类别，可能的组合数是有限的，码长超过一定范围是没有意义的
5. 同等长度的编码，任意两个类别的编码距离越远，纠错能力越强

3.6.分类不平衡问题

- 之前介绍的分类学习算法都是假设，不同类别的训练样本的数量是相同的。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
- 对于OvR和MvM，由于对每个类进行了相同的处理，拆分出的二分类问题的类别不平很问题会相互抵消
- 这里以线性分类器进行讨论

---

1. 对于$y=w^Tx+b$分类新样本x的时候，实际上是用预测出的y值与一个阈值进行比较，例如通常认为$y>0.5$时判别为正例，反之，则为反例

2. y实际上表达了正例的可能性，几率$\frac{y}{y-1}$则反映了正例可能性和反例可能性的比值，当阈值为0.5时，表明分类器的正、反例可能性相同，即分类器决策规则是

$$若\frac{y}{1-y}>1则\ 预测为正例$$

然而当训练集的正反例数目不同的时候，令m^+表示正例的数目，m^-表示反例的数目，则观测几率是$\frac{m^+}{m^-}$，由于我们假设训练集是真实样本总体的无偏采样，因此观测几率和真实几率是等价的。此时，分类器的决策规则是

$$若\frac{y}{1-y}>\frac{m^+}{m^-}则\ 预测为正例$$

我们需要将分类器的决策模式变更为基于$\frac{y}{1-y}>1$的，因此需令

$$\frac{y'}{1-y'}=\frac{y}{1-y}\times\frac{m^-}{m^+}$$

这就是类别不平衡学习的一个策略----再缩放(rescaling)再平衡(rebalance)

- 在实际操作中再缩放却并不平凡，因为“训练集是真实样本总体的无偏采样”是不成立的，因此不能使用观测几率来代替真实几率
- 现有三种技术的做法分别是（我们假定正类样例较少，反类样例较多）

---

1. 直接对训练集的反类样例进行欠采样(undersampling)下采样(downsampling)，即去除一部分反例使得正反样例数目接近
2. 对训练集里的正类样例进行过采样(oversampling)上采样(upsampling)，即增加一些正例使得正、反例数目接近
3. 直接基于原始训练集进行学习，在用训练好的分类器进行预测时，嵌入$\frac{y'}{1-y'}=\frac{y}{1-y}\times\frac{m^-}{m^+}$到决策过程中，称为阈值移动(threshold-moving)

欠采样法的时间开销小于过采样法，因为前者丢弃了很多反例，使实际使用的训练集小于初始训练集，而过采样法则增加了许多正例，其训练集大于初始训练集

过采样法不能简单地对初始正例样本进行重复采样，否则会导致严重的过拟合

- 使用SMOTE算法对训练集里的正例进行插值来产生额外的正例

欠采样法若随机的丢弃反例，可能的导致重要信息的丢失

- 使用EasyEnsemble算法，利用集成学习的机制，将反例划分为若干个集合供不同学习器使用

- 再缩放也是代价敏感学习(cost-sensitive learning)的基础。将$\frac{y'}{1-y'}=\frac{y}{1-y}\times\frac{m^-}{m^+}$转化为$\frac{y'}{1-y'}=\frac{y}{1-y}\times\frac{cost^+}{cost^-}$，其中$cost^+$、$cost^-$分别是将正例误分为反例的代价和将反例误分为正例的代价

第四章 决策树

4.1.基本流程

- 决策树(decision tree)是一类常见的机器学习方法。以二分类任务为例，我们希望从给定训练数据集学得一个模型用以对新示例进行分类，把这个样本分类任务，可以看作“当前样本属于正类吗？”这个问题的“决策”过程
- 决策树是基于树结构来进行决策的。决策过程中提出的每一个判定问题都是对某个属性的测试；每个测试的结果或是导出最终结论，或是导出进一步的判定问题，其考虑范围是在上次决策结果的限定范围之内
- 一般的，一颗决策树包含一个根结点、若干个内部结点和若干个叶结点；叶结点对应于决策结果，其他每个结点则对应于一个属性测试
- 决策树学习的目的是为了产生一颗泛化能力强，即处理未见示例能力强的决策树，其基本流程遵循简单且直观的分而治之(divide-and-conquer)策略
- 决策树学习的基本算法

输入：训练集$D=\{(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)\}$;

属性集$A=\{a_1,a_2,\dots,a_d\}$.

过程：函数$TreeGenerate(D,A)$

1:生成结点node；

2:if D中的样本全属于同一类别C then

3: 将node标记为C类叶结点；return

4:end if

5:if $A=\varnothing\ OR\ D$中的样本在A上取值相同 then

6: 将node标记为叶结点，其类别标记为D中的样本数最多的类；return

7:end if

8:从A中选择最优划分属性a_*；

9:for a_*的每一个值a_*^v do

10: 为node生成一个分支；令D_v表示D中在a_*上取值为a_*^v的样本子集；

11: if D_v为空 then

12: 将分支结点标记为叶结点，其类型标记为D中样本的最对的分类；return

13: else

14: 以$TreeGenerate(D_v,A\backslash\{a_*\})$为分支结点

15: end if

16:end for

输出：以node为根结点的一棵决策树

- 决策树的基本算法递归返回的情形

---

1. 当前结点包含的样本全属于同一类别，无需划分，递归返回
2. 当前属性集为空，或是所有样本在所有属性上的取值相同，无法划分，递归返回

    - 我们把当前结点标记为叶结点，并将其类别设为该结点所含样本最多的类别
3. 当前结点包含的样本集合为空，不能划分，递归返回
    - 把当前结点标记为叶结点，但将其类别设为其父结点所含样本最多的类别

4.2.划分选择

- 决策树学习的关键是如何选择最优划分属性
    1. 一般而言，随着划分过程不断进行，我们希望决策树的分支结点所包含的样本尽可能属于一个类别，即结点的纯度(purity)越来越高

4.2.1.信息增益(information gain)

- 信息熵(information entropy)是度量样本集合纯度最常用的指标

---

1. 假定当前样本集合D中第k类样本所占比例为$p_k(k=1,2,\dots,|\gamma|)$，则$D$的信息熵定义为

$$Ent(D)=-\sum_{k=1}^{|\gamma|}p_k\log_2 p_k$$

2. 若$p=0$，则$plog_2p=0$
3. $Ent(D)$的最小值为0，最大值为$\log_2|\gamma|$
4. $Ent(D)$的值越小，则D的纯度越高

- 假定离散属性a有V个可能的取值$a^1,a^2,\dots,a^V$，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上的取值为a^v的样本，记为D^v。由此，可以计算D^v的信息熵，由于不同分支结点所包含的样本数不同，给分支结点赋予权重$|D^v|/|D|$，于是可以计算用属性a对样本集D进行划分所获的信息增益(information gain)

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

2. 信息增益越大，使用该属性所获得的纯度提升就越大
3. 选择属性$a_*=\arg\max_{a\in A}Gain(D,a)$

4.2.2.增益率(gain ratio)

- 信息增益准则对可取数值数目较多的属性有偏好
- 减少这种偏好的不利影响，引入了增益率(gain ratio)

---

1. 增益率定义为

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

2. 其中，$IV(a)=-\sum_{v=1}^{V}\frac{|D^v|}{|D|}log_2\frac{|D^v|}{|D|}$，称为属性$a$的固有值(intrinsic value)，属性$a$的可能取值数目越多($V$越大)，则$IV(a)$的值越大

- 增益率准则对可能取值数目较少的属性有偏好
- C4.5算法使用了启发式增益率，从候选划分属性中找出信息增益高于平均水平的属性，在从中选择增益率最高的

4.2.3.基尼指数

- CART(Classification and Regression Tree)决策树使用基尼指数(GIni index)来选择划分属性

---

1. 基尼指数可以度量数据集D的纯度

$$\begin{aligned}Gini(D)&=\sum_{k=1}^{|\gamma|}\sum_{k'\neq k}p_kp'_k\\&=1-\sum_{k=1}^{|\gamma|}p_k^2\end{aligned}$$

- 基尼指数反映了从数据集D中随机抽取两个样本，其类别标记不一致的概率。因此，基尼指数越小，数据集D的纯度越高

---

1. 属性a的基尼指数定义

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

2. 因此，我们候选属性集合A中，选择那个使得划分后基尼指数最小的属性作为最优划分，即

$$a_*=\arg\min_{a\in A}Gini\_index(D,a)$$

4.3.剪枝处理

- 剪枝(pruning)是决策树学习算法应对过拟合的主要手段
- 决策树学习中，为了尽可能正确分类训练样本，结点划分过程不断重复，会造成决策树分支过多，这时就可能导致因训练样本学得“太好”，以致于把训练集自身的一些特点当作所有数据都具有的一般性质而导致过拟合
- 决策树剪枝的基本策略有预剪枝(prepruning)和后剪枝(postpruning)

4.3.1.预剪枝(prepruning)

- 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，如果当前结点的划分不能带来决策树的泛化性能提升，则停止划分并将当前的结点标记为叶结点
- 只有一层划分的决策树，也叫做决策树桩
- 预剪枝显著减少了决策树的训练时间开销和测试时间开销

- 虽然有些分支当前划分不能提高泛化性能，但在其基础上进行后续划分有可能导致性能显著提高
- 预剪枝决策树有欠拟合的风险

4.3.2.后剪枝(postpruning)

- 后剪枝则是先从训练集生成一棵完整的决策树，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结点能带来决策树泛化性能提升，则将该子树替换为叶结点
- 后剪枝决策树的欠拟合风险较小，泛化性能优于预剪枝决策树
- 由于后剪枝过程是在生成完全决策树之后进行的，并且要自底向上地对树中的所有非叶结点进行逐一考察，因此训练时间开销要远大于未剪枝决策树和预剪枝决策树

4.4.连续与缺失值

4.4.1.连续值处理

- 对于连续属性，由于可取数目不再有限，因此，不能直接根据连续属性的可取值来对结点进行划分
- 需要使用连续属性离散化技术
- 最简单的策略是二分法(bi-partition)对连续属性进行处理，这是$C4.5$决策树算法采用的机制

---

1. 在给定样本集D和连续属性a，假定a在D上出现了n个不同的取值，将这些值从小到大进行排序，记为$\{a^1,a^2,\dots,a^n\}$。基于划分点t可将D分为子集D_t^-和D_t^+，其中D_t^-包含那些在属性a上取值不大于t的样本，而D_t^+则包含那些在属性a上取值大于t的样本。显然，对相邻的属性取值a^i与a^{i+1}来说，t在区间$[a^i,a^{i+1})$中取任意值所产生的划分结果相同
2. 因此，对连续属性a，我们可考察包含$n-1$个元素的候选划分点集合为$T_a=\left\{\frac{a^i+a^{i+1}}{2}|1\leq i\leq n-1\right\}$，把区间$[a^i,a^{i+1})$的中位点$\frac{a^i+a^{i+1}}{2}$作为候选划分点
3. 根据信息增益选取最优划分点，通过最优划分点进行样本集合的划分

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

- 与离散属性不同，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

- 现实任务中常会遇到不完整样本，即样本的某些属性值缺失

---

1. 给定训练集D和属性a，令$\tilde{D}$表示D中在属性a上没有缺失值的样本子集，我们可以仅根据$\tilde{D}$来判定属性a的优劣。假定属性a有V个可取值$\{a^1,a^2,\dots,a^V\}$，令$\tilde{D}^v$表示$\tilde{D}$在属性a上的取值为a^v的样本子集，$\tilde{D}^k$表示$\tilde{D}$中属性第k类$(k=1,2,\dots,|\gamma|)$的样本子集，则显然有$\tilde{D}=\bigcup_{k=1}^{|\gamma|}\tilde{D}_k$，$\tilde{D}=\bigcup_{v=1}^{V}\tilde{D}^v$。假定我们为每一个样本$x$赋予一个权重$w_x$，并定义

   ρ无缺失值样本所占比例

$$\rho=\frac{\sum_{x\in\tilde{D}}w_x}{\sum_{x\in D}w_x}$$

$\tilde{p}_k$无缺失值样本中第k类所占比例

$$\tilde{p}_k = \frac{\sum_{x\in\tilde{D}_k} w_x}{\sum_{x\in\tilde{D}} w_x} \quad (a \le k \le |\gamma|)$$

$\tilde{r}_v$无缺失值样本中在属性a中取值a^v的样本所占比例

$$\tilde{r}_v = \frac{\sum_{x\in\tilde{D}^v} w_x}{\sum_{x\in\tilde{D}} w_x} \quad (1 \le v \le V)$$

显然

$$\sum_{k=1}^{|\gamma|} \tilde{p}_k = 1$$

$$\sum_{v=1}^{V} \tilde{r}_v = 1$$

2. 将信息增益的计算式推广为

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

3. 信息熵推广为

$$Ent(\tilde{D}) = -\sum_{k=1}^{|\gamma|} \tilde{p}_k \log_2 \tilde{p}_k$$

- 若样本x在划分属性a上取值已知，则将x划分入其取值对应的子结点，且样本权值在子结点中保持为w_x
- 若样本x在划分属性a上的取值未知，则将x划分入所有的子结点，且样本权值在与属性值a^v对应的子结点中调整为$\tilde{r}_v \cdot w_x$；直观地看，这就是让同一个样本以不同的概率划入不同的子结点中去

4.5.多变量决策树(multivariable decision tree)

- 若把每个属性视为坐标空间的一个坐标轴，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，对样本的分类意味着在坐标空间中寻找不同类样本之间的分类边界。
- 决策树所形成的分类边界有一个明显的特点：轴平行(axis-paraellel)，即它的分类边界由若干个与坐标轴平行的分段组成
- 分类边界的每一段都是与坐标轴平行的，这样的分类边界使得学习结果有较好的可解释性。
- 但是，学习任务的真实分类边界比较复杂时，必须使用很多段划分才能获得较好的近似，此时决策树会相当复杂，由于需要进行大量的属性测试，时间开销比较大
- 如果可以使用斜的划分边界，则决策树模型将大为简化
- 多变量决策树(multivariable decision tree)就是能实现斜划分甚至是更复杂的决策树

---

1. 多变量决策树中，非叶结点不再是仅对某个属性，而是对属性的线性组合进行测试
2. 每个非叶结点是一个形如$\sum_{i=1}^{d} w_i a_i = t$的线性分类器，其中$w_i$是属性$a_i$的权重，$w_i$和$t$可在该结点所含的样本集和属性集上学得

3. 多变量决策树与传统的单变量决策树(univariable decision tree)不同，在多变量决策树中的学习过程是建立一个合适的线性分类器

第五章 神经网络

5.1.神经元模型

- 神经网络(neural networks)是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的交互反应
- 神经网络中最基本的成分是神经元(neuron)模型
  1. 在生物神经网络中，每个神经元与其他神经元相连，当它兴奋时，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位；如果某神经元的电位超过了一个阈值(threshold)，那么它就会被激活，即兴奋起来，向其他神经元发送化学物质
  2. 上述的情景可以抽象为M-P神经元模型，神经元接收到来自n个其他神经元传递过来的输入信号，这些输入信号通过带权重的连接(connection)进行传递，神经元接收到的总输入值将于神经元的阈值进行比较，然后通过激活函数(activation function)处理以产生神经元的输出

     $$y=f\left(\sum_{i=1}^{n}w_ix_i-\theta\right)$$

     其中x_i是第i个神经元的输入，w_i是对应神经元的连接权重，θ是阈值，y是输出
  3. 理想的激活函数是阶跃函数，但是阶跃函数具有不连续、不光滑的性质，因此实际使用的激活函数是Sigmoid函数
- 许多个神经元按一定层次结构连接起来，就得到了神经网络
- 神经网络可以视为包含了许多参数的数学模型，这个模型是若干个函数，例如$y=f(\sum_i w_ix_i-\theta_j)$相互嵌套代入而得
- 有效的神经网络学习算法大多数以数学证明为支撑

5.2.感知机与多层网络

- 感知机(Perceptron)由两层神经元组层，输入层接受外界输入信号后传递给输出层，输出层是M-P神经元，亦称阈值逻辑单元(threshold logic unit)
- 感知机能实现与、或、非逻辑运算，假定f是阶跃函数
  1. 与$(x_1\wedge x_2)$:令$w_1=w_2=1$，$\theta=2$，则$y=f(1\cdot x_1+1\cdot x_2-2)$，仅在$x_1=x_2=1$时，$y=1$
  2. 或$(x_1\vee x_2)$:令$w_1=w_2=1$，$\theta=0.5$，则$y=f(1\cdot x_1+1\cdot x_2-0.5)$，当$x_1=1$或$x_2=1$时，$y=1$
  3. 非$(\neg x_1)$:令$w_1=-0.6$，$w_2=0$，$\theta=-0.5$，则$y=f(-0.6\cdot x_1+0\cdot x_2+0.5)$，当$x_1=1$时，$y=0$；当$x_1=0$时，$y=1$
- 给定训练数据集，权重$w_i(i=1,2,\dots,n)$以及阈值θ可供过学习得到，其中阈值θ可看作一个固定输入为-1.0的哑结点(dummy node)所对应的连接权重w_{n+1}
- 感知机的学习规则是，对训练样例(x,y)，若当前感知机的输出$\hat{y}$，则感知机权重的调整

$$w_i \leftarrow w_i + \Delta w_i$$
$$\Delta w_i = \eta(y - \hat{y})x_i$$

其中$\eta \in (0, 1)$称为学习率(learning rate)

- 感知机对训练样例(x, y)预测正确，即$\hat{y} = y$，则感知机不发生变化
- 感知机只有输出层进行激活函数处理，即只拥有一层功能神经元(functional neuron)
- 只有求解问题是线性可分的，才存在一个线性超平面将它们分开，进而感知机的学习过程是一定会收敛的(converge)，否则感知机的学习过程就会发生振荡(fluctuation)
- 解决非线性可分问题，需要使用多层功能神经元，例如两层感知机
- 输出层和输入层之间的多层神经元，被称作隐层或隐含层(hidden layer)
- 隐含层和输出层的神经元都是拥有激活函数的功能神经元
- 常见的神经网络是层级结构，每层神经元与下一层神经元全互联，神经元之间不存在同层连接，也不存在跨层连接，这样的神经网络叫做多层前馈神经网络(multi-layer feedforward neural networks)
    1. 输入层神经元接受外界输入
    2. 隐层和输出层神经元对信号进行加工
- 神经网络的学习过程，就是根据训练数据来调整神经元之间的连接权(connection weight)以及每个功能神经元的阈值