# 机器学习(ML )

## 概率近似正确 - PAC(Probably Approximately Correct)

最重要理论模型 : $$ P (|f(X)-y| \le \epsilon)\ge 1 - \delta $$ 其中:

- $X : data$
- $f(X): 对X的判断$
- $y : 真实值$
- $\epsilon : 一个趋于零值$
- $P : 概率值$
- $\delta: 一个趋于零值$

NFL定理: 具体问题, 具体分析.

没有最好的算法,只有相对较优的算法

( 马哲贯彻人生🙃 )

## 误差

- Underfitting(欠拟合)
- Overfitting(过拟合)

image-20230925213330855

## 三个关键问题

### 1. 评估方法

- 留出法(hold-out)

  > 将data切分为两个set,训练集和测试集(0.8:0.2 ...)

  - 保证数据分布一致性(分层采样)
  - 多次重复划分(百次测试求均值,去除切分数据的影响)
  - 最终预测模型为全数据训练

- $k$-fold交叉验证法(cross calidation)

  > 进行$k$次划分,去除划分扰动,再根据$k$ 划分 data后,循环训练 这$k$ 个集合

  - 留一法($k =X-1$, 则得到 level-one-out, LOO )

- 自助法(bootstrap)

  > 基于可重复采样, 约有36.8%的数据不会被抽中

  - 训练集与原样本集相同

    - 包外估计(out-of-bag estimation)

$$ \lim_\limits{m\to \infty} {(1-\cfrac{1}{m})}^m = \cfrac{1}{e} \approx 0.368 $$

**参数**

- 算法的参数: 一般由人设定,也称为 “超参数”
- 模型的参数: 一般由学习确定

## 2. 性能度量

**回归任务**

常用均方误差Err : $$ E(f,X) = \cfrac{1}{2m} \displaystyle \sum\limits_{i=1}^{m}(f(x_i)-y_i)^2 $$

**分类任务**

- 错误率Err:

$E(f,X) = \cfrac{1}{m} \displaystyle \sum \limits_{i=1}^{m} I \cdot (f(x_i)\ne y_i)$

- 精度Acc:

$A(f,X) = \cfrac{1}{m}\displaystyle \sum \limits_{i=1}^{m} I \cdot (f(x_i) = y_i)$

$= 1-E(f,X)$

image-20230925224501316

- 查准率: $P=\cfrac{TP}{TP+FP}$
- 查全率: $R=\cfrac{TP}{TP+FN}$
- $F_1$度量: $\cfrac{1}{F_1} = \cfrac{1}{2} \cdot (\cfrac{1}{P} + \cfrac{1}{R})$
- $F_{\beta}$度量:

$\cfrac{1}{F_{\beta}} = \cfrac{1}{1+{\beta}^2} \cdot \left(\cfrac{1}{P} + \cfrac{{\beta}^2}{R}\right)$

$\text{当} \begin{cases} \beta < 1, & \text{查准率影响更大} \ \beta > 1, & \text{查全率影响更大} \end{cases}$

## 3. 比较检验

> 统计假设检验为学习器性能比较提供了重要依据

- 交叉验证t检验(基于成对t检验)
    - k-fold交叉检验; $5 \times 2$ 交叉验证
- McNemar检验(基于列联表,卡方检验)

# 线性模型

**1. 线性回归**

$y \backsimeq \displaystyle \sum_{i=1}^{m} w_ix_i + b = w^Tx+b$

- 离散变量
  - 有序: 0,1,2
  - 无序: 001,010,100 (k维向量)

令均方误差最小化:

$(w^,b^) = \arg\min_\limits{(w,b)} \displaystyle \sum_\limits{i=1}^{m} (f(x_i)-y_i)^2$

$=\arg \min_\limits{(w,b)} \displaystyle \sum_{i=1}^{m} (wx_i + b - y_i)^2$

对$E(w,b)= \displaystyle \sum_{i=1}^{m}(wx_i+b-y_i)^2$

- 最小二乘法估计

$\cfrac {\partialial E(w,b)}{\partialial w} = 2 \left( w\displaystyle\sum_{i=1}^{m} x_i^2 - \displaystyle\sum_{i=1}^{m}(y_i -b)x_i \right)$

$\cfrac {\partialial E(w,b)}{\partialial b} = 2 \left( mb- \displaystyle\sum_{i=1}^{m}(y_i-wx_i) \right)$

令导数为0,得到闭式(closed-form)解:

$w = \cfrac {\displaystyle\sum_{i=1}^{m} y_i(x_i-\overline {x})} {\displaystyle\sum_{i=1}^{m}x_i^2 - \cfrac{1}{m}\left(\displaystyle\sum_{i=1}^{m}x_i \right)^2}$

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

**2. 多元(multi-variate)线性回归**

$y \backsimeq w_0 + w^1x^1+w^2x^2 +\cdot \cdot \cdot+w^nx^n$

$=\displaystyle \sum_{i=0}^{m}w^{i}x^{i} = W^{T}X$

后面我们对观测集,采用下面记号:

$X_{N \times p} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \ x_{21} & x_{22} & \dots & x_{2p} \ \vdots & \vdots & \ddots & \vdots \ x_{N1} & x_{N2} & \dots & x_{Np} \end{bmatrix}$

$= (x_1,x_2,x_3,\cdots,x_N)^T$

其中$x_i = (x_{i1},x_{i2},\cdots,x_{ip})^T \quad (i \in N)$

- 最小二乘估计

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

令$E_{\hat w} = (y-X\hat w)^T(y-X\hat w)$

对$\hat w$求导: $\partialial E_{\hat w} = 2X^T(X_{\hat w}-y)$令其为零可得$\hat w$

- 若$X^TX$ 满秩或正定, 则$\hat w^* = (X^TX)^{-1}X^T y$
- 若$X^TX$不满秩, 则可以解出多个$\hat w$
  - 设置归纳偏好或表达为引入正则化

### 3. 广义线性模型

一般形式: $$ y = g^{-1} (W^TX) $$

例如对于$f(x) = e^{W^TX}$ 可以通过对其求$\ln$ 来降幂从而达到线性拟合,如下图

### 4. 对率回归:

> 对数几率回归(logistic regression) 简称对率回归
>
> $\cfrac{y}{1-y} \longrightarrow \cfrac{P(postive |X)}{P(negetive|X)}$ :几率(odds) 即 log odds $\longrightarrow$ logit

对于线性回归模型产生的实值输出$z = W^TX$和期望输出$y\in {0,1}$

理想函数 $y(z) = \begin{cases} 0,& z<0 \ 0.5, &z =0 \ 1 ,& z>0 \end {cases}$的函数性质较差,因此寻找如下替代函数$y = \cfrac{1}{1+e^{-z}}$

相比之下,替代函数的性质更好

- 对率函数(logistic function)与逻辑没有任何关系
- 实值函数,在$y \in (0,1)$ 连续
- 用回归模型做分类

因此对 $y =\cfrac{1}{1+e^{-z}},\quad z =W^TX$

$\Longrightarrow y = \cfrac{1}{1+e^{-(W^TX)}} \Longrightarrow \ln \cfrac{y}{1-y} = W^TX$

- 无需事先进行假设数据分布
- 可以得到"类别"的近似概率预测
- 可直接应用现有数值优化算法求取最优解

###### 通过极大似然法求解

> 不能通过求梯度为零得极值点，因为目标函数，并不是凸函数

$\max \ln{\left( P(\text{True-Positive})P(\text{Positive}) + P(\text{True-Negative})P(\text{Negative}) \right)}$

即 $\max \ln\left( y \cdot \cfrac{e^{W^TX}}{1+e^{W^TX}} + (1-y)\cfrac{1}{e^{W^TX}}\right)$

简化之后可得

$\max \bigl( \ln ( y \cdot {e^{W^TX}} +1 - y ) - \ln (1+{e^{W^TX}}) \bigr)$

由于 $y=0$ 或 $y=1$

那么 $\max \begin{cases} {W^TX} - \ln(1+{e^{W^TX}} ), & \text{if } y=1 \ \qquad\quad - \ln(1+{e^{W^TX}}), & \text{if } y=0 \end{cases}$

合并上述讨论可得函数 $\max \quad \bigl( y \cdot W^TX - \ln(1+{e^{W^TX}}) \bigr)$

$\to \quad z = \min \quad \bigl( \ln \cfrac{1+{e^{W^TX}}}{{e^{y \cdot W^TX}}} \bigr) \to z= \min\quad \bigl( \ln \cfrac{1+{e^{f(x)}}}{{e^{y \cdot f(x)}}} \bigr)$

一般情况下，到此处使用梯度下降法求解，更适合计算机迭代，可以使用二阶导得到，但并不通用。

- 梯度下降法 如果矩阵不是满秩，没有逆矩阵，就无法使用最小二乘法。

#### 5. 线性鉴别分析

> Linear Discriminant Analysis
>
> 目标：最大化广义瑞利商
>
> 尽可能的**最小化同类之间的距离，最大化异类之间的距离**

$\min (w^T\Sigma_0w + w^T\Sigma_1w)$

$\max(| w^T\mu_0-w^T\mu_1 |_2^2)$

于是 可求解$\max J$

$\max J = \max \bigg ( \cfrac{| w^T\mu_0-w^T\mu_1 | _2^2}{w^T\Sigma_0w + w^T\Sigma_1w} \bigg) = \max \bigg( \cfrac{w^T(\mu_0 - \mu_1)(\mu_0-\mu_1)^T w}{w^T(\Sigma_0+\Sigma_1)w} \bigg)$

> 类内散度矩阵（within-class scatter matrix）

$S_w = \Sigma_0 + \Sigma_1$

$= \sum_\limits{x \in X_0} (x-\mu_0)(x-\mu_0)^T + \sum_\limits{x\in X_1}(x-\mu_1)(x-\mu_1)^T$

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

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

因此就有：

$\max J = \max \bigg( \cfrac{w^TS_bw}{w^TS_ww} \bigg)$

其等价于

$\min_\limits{w} -w^TS_bw\ s.t. \quad w^TS_ww =1$

由拉格朗日子乘法

$g(x) = -w^TS_bw + \lambda (w^TS_ww -1 )$

令$g'(x) = 0$ 且其相关系数矩阵是对称 即得 $-(S_b+S_b^T)w + \lambda (S_w+S_w^T)w = -2S_bw + 2\lambda S_w w = 0$

易得$S_bw= (\mu_0 - \mu_1)(\mu_0 -\mu_1)^T w$

注意到求解的$w$关注的是线性方程的方向，而$(\mu_0- \mu_1)^Tw$为标量

于是可令$\lambda = (\mu_0-\mu_1)^Tw$

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

通常情况下进行奇异值分解更加快速便捷$S_w= U \Sigma V^T$

然后可得：$S_w^{-1} = V\Sigma^{-1}U^T$

#### 6. 多分类问题

> 现实中常使用多分类来解决分类问题

###### OvO & OvR

对于$N$个类别$C_1,C_2,C_3,\dots,C_N$，可以将其折分成二分类问题，这将会产生$\cfrac{N\cdot(N-1)}{2}$次分类

或者拆分成非均衡的二分类，即一对其余（One v.s. Rest），仅需$N$个分类器

虽然OvO进行多次分类，但OvR每个分类器都使用全部的样例，所以两者在多数情况下时间开销相近

###### MvM

> 多对多分类（Many v.s. Many）

OvO & OvR显然是MvM的特例

针对多对多的分类问题，常采用**纠错输出码**（error correction output codes ECOC）

###### ECOC

> ECOC工作过程主要分为编码和解码两步

- 编码 对$N$个类别做$M$次划分，每次划分，将一部分类别化为正类，一部分化为反类，从而形成一个二分类训练机；这样一共有$M$个训练集，可以训练出$M$个分类器
- 解码 $M$个分类器分别对样本进行预测，这些预测标记组成一个编码，将这个预测编码与每个类别各自比较，返回其中距离最小的类别作为最终预测结果

#### 7. 类别不平衡问题

> 之前的分类学习方法都有一个共同基本假设，不同类别的训练样例数目 ***相当***

如果不同类别的训练样例数目稍稍不同，通常影响不大

但若是差别很大，会对学习过程造成困扰，例如十分极端的训练样例$正例:反例=99:1或者 998:2$

模型的训练时，只需一直返回多数方即可达到$99%,99.8%$ 但是这样的学习器没有任何价值，它无法预测任何反例。

**类别不平衡** 就是指在分类任务中不同类别的训练样例数目差别较大的情况。

> 在拿到新的数据进行预测时，预测结果的决策，需要依靠所训练出的分类规则

对于线性模型，当进行决策时，预测结果为正类，就是根据其预测为正类的概率大于负类的概率，换言之：

$\cfrac{y}{1-y} > 1$

然而在类别不平衡的条件下，训练出的分类器并不是这样，如正类小于负类的情况下，当预测的符合$\cfrac{y}{1-y} > ? > 1$ 才会被鉴别为正类，显然这样的分类器并不“公平”

因此在类别不平衡学习中，对于分类器的决策执行时，可以对其进行“再缩放”

即$\cfrac{y'}{1-y'} = \cfrac{y}{1-y} \cdot\cfrac{1}{?}$使之平衡

但实际上实现起来却很难，因为我们认为的训练集是总体的无偏采样，能够代表总体概率。但实际上训练集数据抽取是随机的，它的偏差可能很大。

- 欠（下）采样——减少多的一方
- 过（上）采样——增加少的一方
- 阈值移动——采用缩放使之平衡

## 决策树模型

> 分而治之，对属性进行判断从而划分

停止条件

- 当前结点包含的样本全属于同一类别，无需划分
- 当前节点属性集为空，或者所有样本在属性上取值相同，无法划分
- 当前结点包含的样本集为空，不能划分

> 决策树的核心在于，使用什么划分方式，能使得属性得到最优划分 决策树是从信息论的基础上发展而来

**信息熵**

> Entropy 用于度量信息的混乱和纯净程度

在集合$D$中，第k类样本占比$p_k$,则$D$的信息熵为

$Ent(D) = - \sum\limits_{k=1}^{|y|}p_{k} \cdot log_2{p_k}$

**信息增益**

> 信息增益是进行划分后信息熵减小所获得的收益量

对离散属性a：${a^1,a^2,...,a^V}$ $D_v (a = a^v) \subseteq D$

$Gain(D,a) = Ent(D) - \sum\limits_{v=1}^{V} \cfrac{|D^v|}{|D|}\cdot Ent(D^v)$

**增益率**

> 当编号考虑为属性，那么上述信息增益的划分方式泛化会非常糟糕，因此引入一个分支数目作为分母，抵消分支数目过多的问题

$Gain_ratio(D,a)=\cfrac{Gain(D,a)}{IV(a)}$，$IV(a) = - \sum\limits_{v=1}^{V} \cfrac{|D^v|}{|D|} \cdot log_2 \cfrac{|D^V|}{|D|}$ （C4.5算法)

#### 基尼指数(Gini index)

$Gini(D) = 1 - \sum\limits_{k=1}^{|y|} p_k^2$

属性a的基尼指数：$Gini_index(D,a)= \sum\limits_{v=1}^{V} \cfrac{|D^v|}{|D|} \cdot Gini(D^v)$

#### 剪枝

> pruning 用于对抗过拟合

- 预剪(pre-pruning)：预先设置条件，防止生长
- 后剪枝(post-pruning)：生成后再剪枝

#### 缺失值处理

> 直接丢弃的方式在高维度数据时十分浪费

- 如何进行划分属性选择
- 给定划分属性，若样本在该属性值缺失，如何划分

> 基本思路：样本赋权，权重划分

### 神经网络

> 简单的神经元模型：激活信号达到反应阈值，就产生输出。

$y= f(\sum\limits_{i=1}^{n}w_ix_i - \theta_j)$

> 理想的映射法则是间断的$y = sgn(x)$，然而更长用的是其替代函数：Sigmoid函数

多层网络：包含隐层的网络。前馈网络：神经元之间不存在同层连接、跨层连接。

> 隐层和输出层神经元也称为 功能单元

设置隐层数目需要进行试错

- 万有逼近性 说明了 **神经网络的可行性**

#### BP算法

> 误差逆传播算法（BackPropagation），使用广义感知机学习规则：$v \leftarrow v + \Delta v$ 基于梯度下降的策略，以负方向对参数进行调整

为方便讨论，做如下规定： 给定训练集$D={(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)},x_i\in R^d , y_i \in R^l$ 输入：$d$维向量 输出：$l$个输出值 隐层：$q$个隐层神经元 输入层权值：$v_{1h},v_{2h},\dots,v_{ih}$ 隐层权值：$w_{h1},w_{h2},\dots,w_{hj}$

第$h$个隐层神经元的输入：$\alpha_h = \sum\limits_{i=1}^{d} v_{ih} x_i$ 第$h$个隐层神经元的输出：$b_h$ 第$j$个输出神经元的输入：$\beta_j = \sum\limits_{h=1}^{q}w_{hj}b_h$ 网络实际输出$\hat y_k =

(\hat y_1^k, \hat y_2^k,\dots,\hat y_l^k)$ $\hat y_j^k = f(\beta_j - \theta_j)$ 均方误差：$E_k = \cfrac{1}{2} \sum\limits_{j=1}^{l} (\hat y_j^k - y_j^k)^2$

则共需学习的参数数目为$(d+l+1)q+l$

误差导致$\Delta v$需要进行改变，因此通过梯度下降的方式调整 $\Delta w_{hj} = -\eta \cfrac{\partial E_k}{\partial w_{hj}}$ 其中$\eta \in (0,1)$表示每次进行改变的**幅度**，**不宜过大，否则在后期易发生振荡，也不宜过小，导致迭代次数过多**

$\cfrac{\partial E_k}{\partial w_{hj}} =\cfrac{\partial E_k}{\partial \hat y_{j}^k} \cdot \cfrac{\partial \hat y_j^k}{\partial \beta_{j}} \cdot \cfrac{\partial \beta_j}{\partial w_{hj}}$

注意到

$\cfrac{\partial E_k}{\partial \hat y_j^k} = (\hat y_j^k - y_j^k)$

$\hat y_j^k = f(\beta_j-\theta_j)$

对Sigmoid函数有 $f'(x)=f(x)\cdot\big(1-f(x)\big)$

令 $g_j = - \cfrac{\partial E_k}{\partial \hat y_{j}^k} \cdot \cfrac{\partial \hat y_j^k}{\partial \beta_{j}}=\hat y_j^k(1-\hat y_j^k)(y_j^k -\hat y_j^k)$

于是有 $\Delta w_{hj} = - \eta \cfrac{\partial E_k}{\partial w_{hj}} = \eta g_j b_h$

类似地

$\Delta\theta_j =-\eta g_j$

$e_h&= -\cfrac{\partial E_k}{\partial b_h} \cdot \cfrac{\partial b_h }{\partial \alpha_{h}}\$ $=- \sum\limits_{j=1}^{l} \cfrac{\partial E_k}{\partial \beta_j}\cdot\cfrac{\partial \beta_j}{\partial b_h}\cdot f'(\alpha_h - \gamma_h)$ $=\quad \sum\limits_{j=1}^{l} w_{hj}g_jf'(\alpha_h - \gamma_h)$ $=b_h(1-b_h)\sum\limits_{j=1}^{l}w_{hj}g_j$

$\Delta v_{ih} = \eta e_h x_i$ $\Delta\gamma_h = -\eta e_h$

### 其它算法

- RBF算法
- ART网络
- SOM网络
- 级联相关网络
- Elman网络
- Boltzmann机

支持向量机