机器学习笔记

作者　：陈浩川 (Holy Chen)
Github　：https://github.com/HolyChen
Email　：chenchen12hao@vip.qq.com
绘图　：Powered by Matplotlib
算法　：Powered by Scikit-learn

2018 年 3 月-2018 年 9 月

目录

1. 简介

1.1. 机器学习定义(Definition of Machine Learning)

Arthur Samuel(1959): 机器学习是一个不用显式编程就能给予计算机学习能力的领域
Machine learning is a field of study that gives computers the ability to learn without being explicitly programmed.
Tom Mitchell(1998): 如果一个程序可以通过对应于任务 T 的经验 E 提升以 P 为为衡量标准的执行 T 的性能，则称其从对应于 T 和 P 的 E 中学习。
Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

1.2. 课程大纲(Outline)

课程由以下四个部分组成
有监督学习(Supervised Learning): 训练数据包含答案。预测连续值称为回归(Regression)，预测离散值称为分类(Classification)。
学习理论(Learning Theory): 学习算法为何有效、需要多少数据等等。(未完成)
无监督学习(Unsupervised Learning): 训练数据集不包含答案。例如，聚类、独立成分分析（鸡尾酒聚会）。(未完成)
强化学习(Reinforcement Learning): 做出一系列的决策从回报函数中学习经验，通过正向激励和反向激励刺激，使其获得更多评价。(未完成)

1.3. 符号定义

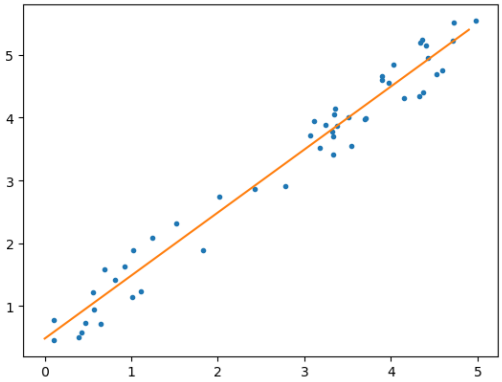
以下为文档中最常见的符号，其他的符号可以通过上下文推断。

$\boldsymbol{m}$	#训练样本（#表示数量）
$\boldsymbol{n}$	#特征
$\boldsymbol{x}$	输入变量、特征
$\boldsymbol{y}$	输出变量、目标特征
$(\boldsymbol{x},\boldsymbol{y})$	训练样本
$(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)})$	第i个训练样本，即训练样本的第i行

文中的大写字母通常表示一个矩阵或者常量，小写字母可能表示矩阵、实数(函数)、向量（函数）。例外情况也是存在的。这些符号通常在上下文中有说明，因此不区分。

2. 线性回归

2.1. 线性回归(Linear Regression)



线性模型(Linear Model)即使用通过仿射变换$Ax+b$将数据$x \in \mathbb{R}^n$投影到目标空间$y \in \mathbb{R}^m$中，然后再对y进行处理，得到目标$g(y)$。预测目标空间为一维连续值时，称为一维线性回归；多维的情况称为多元线性回归(Multivariate Linear Regression)。多元线性回归可以通过组合多个一元线性回归模型得到，因此只讨论一维线性回归。

线性回归假设

$$h_\theta(x) = h(x) = \theta^T x = \sum_{i=0}^{n} \theta_i x_i$$

其中$\theta \in \mathbb{R}^{n+1}$为系数向量。为了方便表示，对$x$进行扩充，假设$x_0 = 0$。
线性回归使用最小二乘函数(Least Square Function)作为待优化的损失函数(Loss Function)，即

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

优化目标为

$$\theta = \underset{\theta}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \underset{\theta}{\operatorname{argmin}} J(\theta)$$

2.2. 梯度下降法(Gradient Descent)

方法

1. 为待优化参数θ设一个初始值；
2. 不断的改变θ，使得J不断减小。

$$\theta_i := \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$$

对J求导可以得：

$$\frac{\partial}{\partial\theta_i}J(\theta)=\sum_{j=1}^{m}\left(h_\theta\left(x^{(j)}\right)-y^{(j)}\right)x_i^{(j)}$$

批梯度下降法(Batch Gradient Descent):

重复直到收敛:

$$\theta_i=\theta_i-\alpha\sum_{j=1}^{m}\left(h_\theta\left(x^{(j)}\right)-y^{(j)}\right)x_i^{(j)}$$

因为平方误差函数是一个凸函数，因此是用梯度下降法配合合理的学习速率，总可以收敛到最小值。

随机梯度下降法(Stochastic Gradient Descent):

重复直到收敛:

for $j=1$ **t** m:

$$\theta_i=\theta_i-\alpha\frac{\partial}{\partial\theta_i}\left(h_\theta\left(x^{(j)}\right)-y^{(j)}\right)x_i^{(j)}$$

大数据样本 BGD 会很慢甚至不可能实现，而 SGD 会快很多但是可能无法在最小值处收敛。

2.3. 正规方程(Normal Equation)

使用正规方程可以求出参数θ的显式解。

将训练样本组成矩阵

$$X=\begin{bmatrix}\left(x^{(1)}\right)^T\\ \cdots\\ \left(x^{(m)}\right)^T\end{bmatrix},y=\begin{bmatrix}y^{(1)}\\ \cdots\\ y^{(m)}\end{bmatrix}$$

$$X\theta=\begin{bmatrix}\left(x^{(1)}\right)^T\theta\\ \cdots\\ \left(x^{(m)}\right)^T\theta\end{bmatrix}=\begin{bmatrix}h_\theta\left(x^{(1)}\right)\\ \cdots\\ h_\theta\left(x^{(m)}\right)\end{bmatrix}$$

优化目标可写成

$$\begin{aligned}J(\theta)&=\frac{1}{2}(X\theta-y)^T(X\theta-y)\\&=\frac{1}{2}tr\left((X\theta-y)^T(X\theta-y)\right)&&\text{(实数迹运算)}\\&=\frac{1}{2}tr(\theta^TX^TX\theta-y^TX\theta-\theta^TX^Ty+y^Ty)\\&=\frac{1}{2}\left(tr(\theta^TX^TX\theta)-tr(y^TX\theta)-tr(\theta^TX^Ty)+tr(y^Ty)\right)\\&=\frac{1}{2}tr(\theta^TX^TX\theta)-tr(y^TX\theta)+\frac{1}{2}tr(y^Ty)&&\text{(迹循环位移)}\end{aligned}$$

对θ求导得

$$\nabla_\theta J(\theta)=\frac{1}{2}(X^TX\theta+X^TX\theta)-X^Ty=X^TX\theta-X^Ty$$

令$\nabla_\theta J(\theta)=0$

$$\theta=(X^TX)^{-1}X^Ty$$

基于数值稳定性考虑，通常使用伪逆(Pseudo Inverse)替代矩阵的逆，即使用$(X^TX)^+$替代$(X^TX)^{-1}$。伪逆的求解需要使用奇异值分解(Singular Value Decomposition, SVD)。矩阵A的伪逆为$A^+=VD^+U^T$，U、D、V分别为A通过 SVD 分解得到的左奇异向量、奇异值、右奇异向

量。因此时间复杂度为$O(mn^2)$，假设$m \geq n$。
在实现时，为了对矩阵X进行增广，即在矩阵的最后加入一列1用于与b相乘，通常使用y相对于原点的偏移量计算b。
通常首先计算X和y的偏移量，使其 0 中心化(Zero Centered)，即首先计算出训练样本x和y的均值，然后对样本已经平移:

$$\mu_x = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$
$$\mu_y = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}$$
$$x^{(i)'} = x^{(i)} - \mu_x$$
$$y^{(i)'} = y^{(i)} - \mu_y$$

即 0 中心化后的特征X为X'，目标y为y'，即

$$X' = X - \mu_x[1 \quad \cdots \quad 1]_{1\times m} \triangleq X - \bar{X}$$
$$y' = y - \mu_y[1 \quad \cdots \quad 1]^T_{1\times m} \triangleq Y - \bar{Y}$$

由于预测目标y已经 0 中心化，因此由X'计算y'的线性模型的截距为0。
有时候，为了提升计算精度，还会对X标准化(normalize)，即使用可逆的对角矩阵W乘 0 中心化的X'作为最终的特征记作$\hat{X}$，即:

$$\hat{X} = WX' = W(X - \bar{X})$$

通常以X各列的L_2范数的倒数作为放缩系数，即

$$W_{i,i} = \frac{1}{\left\|(X - \bar{X})_{:,i}\right\|_2}$$

因此，使用正规方程直接所求解的问题为

$$\hat{X}^T X\hat{\theta} = \hat{X}^T y'$$

对于原始的输入变量$\tilde{x}$，首先也要对其进行平移、放缩，然后再与$\hat{\theta}$相乘，最后与μ_y相加，即

$$\tilde{y} = \hat{\theta}^T W(\tilde{x} - \mu_x) + \mu_y$$

展开可以得到

$$\tilde{y} = \left(W^T\hat{\theta}\right)^T\tilde{x} + \left(\mu_y - \left(W^T\hat{\theta}\right)^T\mu_x\right)$$

因此原问题的系数和截距分别为

$$\theta = W^T\hat{\theta}$$
$$b = \mu_y - \left(W^T\hat{\theta}\right)^T\mu_x$$

通过这种方法，避免了矩阵的增广，因此更加灵活，并且效率更高。

2.4. 线性回归的概率解释(Probability Interpretation)

假设$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$，其中$\varepsilon^{(i)}$为误差项，可以看作一种随机噪声，它代表了未建模的信息。

假设$\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$（中心极限定理），即$P\left(\varepsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{{\varepsilon^{(i)}}^2}{2\sigma^2}\right)$，因此

$$P\left(y^{(i)}|x^{(i)};\theta\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right)$$

即$y^{(i)}|x^{(i)};\theta \sim \mathcal{N}\left(\theta^T x^{(i)}, \sigma^2\right)$。这里的";"表示后面的变量为参数而非随机变量，这里采用的是

贝叶斯学派的观点，将θ视作参数而非随机变量。
并且假设$\varepsilon^{(i)}$为独立同分布(Independently Identically Distributed, i.i.d)。
似然性公式可以写成

$$\begin{aligned} L(\theta) &= P(y|X;\theta) \\ &= \prod_{i=1}^{m} P\left(y^{(i)}|x^{(i)};\theta\right) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)}-\theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \end{aligned}$$

极大似然估计(Maximum Likelihood Estimate)：找到能让似然函数最大化的参数θ。对其取对数得到

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= m\log\frac{1}{\sqrt{2\pi}\sigma} - \sum_{i=1}^{m}\frac{\left(y^{(i)}-\theta^T x^{(i)}\right)^2}{2\sigma^2} \end{aligned}$$

注意到

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)}-\theta^T x^{(i)}\right)^2$$

这与最小二乘法定义的损失函数相同。

2.5. 欠拟合、过拟合与正则化(under/over-fitting & Regularization)

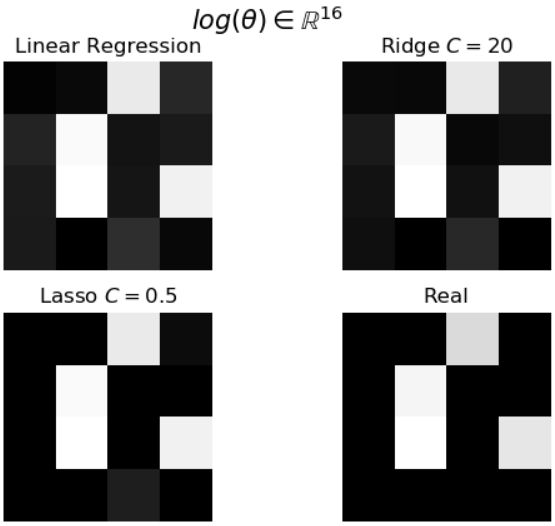
欠拟合(under-fitting)：数据中一些明显的模式没有被捕获出来
过拟合(over-fitting)：模型仅反映出了训练样本的模式，而非本质信息。

为了避免模型过拟合，一种方法是减少模型所利用的特征数量，让模型尽可能只考虑重要的信息，而忽略不重要的噪音，这可以看作是降低了模型的复杂度。
因此考虑将非零系数的数量即θ的 L0 范数作为损失函数的一部分，即：

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)}-\theta^T x\right)^2 + C\sum_{i=1}^{n}1\{\theta_i \neq 0\}$$

其中常数$C \geq 0$是复杂度参数，用于控制模型的收缩(Shrinking)程度，C越大对模型复杂度的惩罚越大，模型收缩越剧烈。过小的C难以修正过拟合，过大的C可能会导致欠拟合。

这种对模型参数的惩罚项称为正则项(Regularization Term)，也称为结构风险(Structural Risk)。描述模型与训练数据差异的一项有时也称为经验风险(Empirical Risk)。



由于 L0 范数不连续，因此通常使用L_1范数或者L_2范数替代。正则项为L_2范数平方的线性回归称为岭回归(Ridge Regression)。

$$L_{Ridge}(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)} - \theta^T x\right)^2 + C\|\theta\|_2^2$$

正则项为L_2范数的称为最小绝对收缩选择算子(Least Absolute Shrinkage and Selection Operator Regression)，通常简称为 LASSO 回归。

$$L_{Lasso}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta^T x\right)^2 + C\|\theta\|_1$$

2.6. 近端梯度下降(Proximal Gradient Descent)

求解L_1正则化，可以采用近端梯度下降(Proximal Gradient Descent, PGD)。
设优化目标为

$$\min_{\theta} f(\theta) + \lambda\|\theta\|_1$$

如果$f(\theta)$可导，并且∇f满足L-Lipschitz 条件，即存在常数$L > 0$使得

$$\|\nabla f(\theta') - \nabla f(\theta)\|_2^2 \le L\|\theta' - \theta\|_2^2 \quad \forall \theta, \theta'$$

则在第k次迭代结果$\theta_{(k)}$附近可以将$f(x)$通过二阶泰勒展开式近似为

$$\hat{f}(\theta) \simeq f\left(\theta_{(k)}\right) + \nabla f\left(\theta_{(k)}\right)^T\left(\theta - \theta_{(k)}\right) + \frac{L}{2}\left\|\theta - \theta_{(k)}\right\|_2^2$$

$$= \frac{L}{2}\left\|\theta - \left(\theta_{(k)} - \frac{1}{L}\nabla f\left(\theta_{(k)}\right)\right)\right\|_2^2 + \text{const}$$

其中的const为与θ无关的常量，显然$\hat{f}(\theta)$的最小值在

$$\theta_{(k+1)} = \theta_{(k)} - \frac{1}{L}\nabla f\left(\theta_{(k)}\right)$$

因此，通过梯度下降法对$f(\theta)$进行最小化，则每一步梯度下降迭代实际上等于最小化二次函数$\hat{f}(\theta)$，将这一思想带入原始的优化目标，可以得到迭代

$$\theta_{(k+1)} = \underset{\theta}{\operatorname{argmin}} \frac{L}{2}\left\|\theta - \left(\theta_{(k)} - \frac{1}{L}\nabla f\left(\theta_{(k)}\right)\right)\right\|_2^2 + \lambda\|\theta\|_1$$

即每一步梯度下降迭代同时考虑L_1范数最小化。首先计算

$$z_{(k)} = \theta_{(k)} - \frac{1}{L}\nabla f\left(\theta_{(k)}\right)$$

然后求解

$$\theta_{(k+1)} = \underset{\theta}{\operatorname{argmin}} \frac{L}{2}\|\theta - z_{(k)}\|_2^2 + \lambda\|\theta\|_1$$

即θ_i为θ的第i个分量，注意到上式中，不存在形如$\theta^i\theta^j(i \neq j)$的交叉项，即$\theta$各项间不相互影响。因此，有闭式解

$$\theta_{(k+1)_i} = \begin{cases} z_{(k)_i} - \lambda/L, & z_{(k)_i} > \lambda/L \quad (x > 0) \\ 0, & \left|z_{(k)_i}\right| \leq \lambda/L \quad (x = 0) \\ z_{(k)_i} + \lambda/L, & z_{(k)_i} < -\lambda/L \quad (x < 0) \end{cases}$$

对于线性回归模型，优化目标中的$f(\theta)$为

$$f(\theta) = (y - \theta^T x)^2$$

显然有

$$\|\nabla f(\theta') - \nabla f(\theta)\|_2^2 = 2\|x(\theta' - \theta)^T x\|_2^2$$

根据内积的关系，

$$\|\nabla f(\theta') - \nabla f(\theta)\|_2^2 = 2\|x(\theta' - \theta)^T x\|_2^2 \leq 2\|\theta' - \theta\|_2^2 \cdot \|x\|_2^4$$

因此，可以取$L = 2\|x\|_2^4$.

对于批梯度下降，$f(\theta)$的梯度为

$$\nabla f(\theta) = 2X^T(y - X\theta)$$

对应的$L = 2\|X^T X\|_F^2 \leq 2\|X\|_F^4$，其中$\|\cdot\|_F$为 Frobenius 范数(Frobenius Norm)。

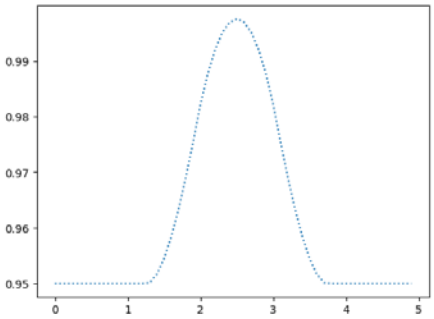
参考：周志华.机器学习.清华大学出版社.2016.252-254

2.7. 局部加权回归(Locally Weighted Regression, Loess/Lowess)

根据模型参数是否随训练样本变动可以将模型分为两类，即

1. 参数学习算法(Parameter Learning Algorithm)：有固定的参数集合，用数据学习参数的算法。
2. 非参数学习算法(Non-parameters Learning Algorithm)：参数的个数随着样本数增长。

一些数据在不同的范围有不同的特征，难以用一个统一的函数进行较好的拟合。对于一个特征向量x，逻辑上只使用它邻域中的数据进行拟合模型。这是一种非参数学习方法，参数依赖于样本。

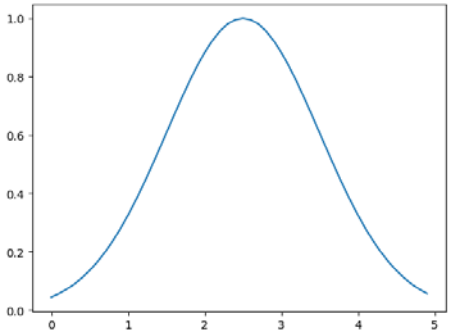


优化目标

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_i w^i \left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

其中，$w^{(i)}$为权值，例如：

$$w^{(i)} = \exp\left(-\frac{\left(x^{(i)} - x\right)^2}{2\tau^2}\right)$$



距离x越近，权重越大，距离x越远，权重越小。参数τ称作波长参数(bandwidth parameter)，它控制函数的下降速率，与正态分布的标准差σ相似，τ越大，下降越平缓，反之越迅速。

补充材料——矩阵导数与迹(Derivative and Trace of Matrix)

矩阵导数(Derivative of Matrix)
向量实值函数$f: \mathbb{R}^N \to \mathbb{R}$:

$$\nabla_\theta J = \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\theta) \end{bmatrix} \in \mathbb{R}^{n+1}$$

矩阵实值函数$f: \mathbb{R}^{M \times N} \to \mathbb{R}$:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

矩阵的迹(Trace of Matrix)
设方阵$A \in \mathbb{R}^{n \times n}$，定义矩阵的迹为

$$\mathrm{tr}A = \sum_{i=1}^{n} A_{ii}$$

迹的性质

$$\begin{aligned}\mathrm{tr}AB &= \mathrm{tr}BA\\ \mathrm{tr}ABC &= \mathrm{tr}BCA = \mathrm{tr}CAB\\ \nabla_A \mathrm{tr}AB &= B^T\\ \mathrm{tr}A &= \mathrm{tr}A^T\\ \mathrm{tr}(a) &= a \quad (a \in \mathbb{R})\\ \nabla_A \mathrm{tr}ABA^TC &= C^TAB^T + CAB\end{aligned}$$

补充材料——频率学派和贝叶斯学派(Frequentist or Bayesian)

频率学派(Frequentist)认为在随机事件的背后存在着一个客观的参数，即世界的本体是真实的，并不改变。
贝叶斯学派(Bayesian)认为我们对随机事件的了解是不完备的，我们对随机事件有一个先验的判断，而随着观察到的事件的增加，我们的知识随之调整改变。
对随机事件进行建模(Modeling)时，其核心就是在求解θ。
方法
频率学派会尝试寻找一个最合理的参数θ，以最好的解释当前发生的事件，例如进行最大似然估计(Maximum Likelihood Estimation, MLE)。

$$\begin{aligned}\hat{\theta}_{MLE} &= \underset{\theta}{\mathrm{argmax}}\, P(X;\theta)\\ &= \underset{\theta}{\mathrm{argmax}} \prod_{i=1}^{m} P\left(x^{(i)};\theta\right)\\ &= \underset{\theta}{\mathrm{argmax}} \sum_{i=1}^{m} \log P\left(x^{(i)};\theta\right)\\ &= \underset{\theta}{\mathrm{argmin}} - \sum_{i=1}^{n} \log P\left(x^{(i)};\theta\right)\end{aligned}$$

最后一行称为 Negative Log Likelihood(NLL)。
对于找到的"最合理"的参数θ，频率学派会寻找一个置信区间(Confidence Interval)$[a,b]$，表示该区间包含了实际的θ的概率。
而贝叶斯学派即将θ视作随机变量，通过一次次的观察对θ的先验概率(Priori)进行纠正，会通过已观察到事件X，计算θ的后验概率$P(\theta|X)$，并选择出具有最大后验概率的θ。此时用到的手段是最大后验估计(Maximum A Posteriori)。

$$\begin{aligned}\hat{\theta}_{MAP} &= \underset{\theta}{\mathrm{argmax}}\, P(\theta|X)\\ &= \underset{\theta}{\mathrm{argmax}} \frac{P(X|\theta)\cdot P(\theta)}{P(X)}\\ &= \underset{\theta}{\mathrm{argmax}}\, P(X|\theta)\cdot P(\theta)\\ &= \underset{\theta}{\mathrm{argmin}} - \log P(X|\theta) - \log P(\theta)\\ &= \underset{\theta}{\mathrm{argmin}} - \sum_{i=1}^{n} \log P\left(x^{(i)}|\theta\right) - \log P(\theta)\end{aligned}$$

如果$\theta \sim \mathcal{N}(0,\sigma^2)$，即$-\log P(\theta) = constant + \frac{\theta^2}{2\sigma^2}$

$$\hat{\theta}_{MAP} = \underset{\theta}{\mathrm{argmin}} - \sum_{i=1}^{n} \log P\left(x^{(i)}|\theta\right) + \theta^2$$

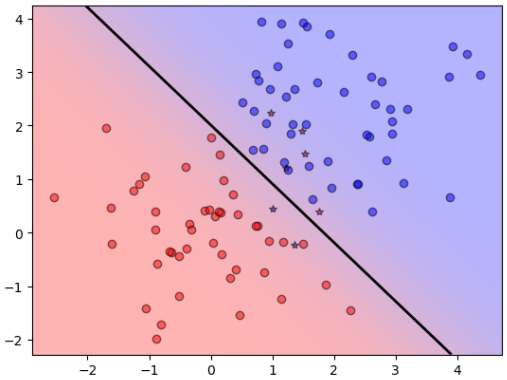
这相当于对 MLE 的 NLL 公式增加了一个L_2正则项，因此正则项从贝叶斯的角度可以解释为模型的先验分布。

参考：http://www.sohu.com/a/215176689_610300

点估计参考：http://staff.ustc.edu.cn/~zwp/teach/Prob-Stat/Lec14_slides.pdf

3. 逻辑回归

3.1. 逻辑回归(Logistic Regression)



分类(Classification)问题需要预测的值为离散值。最常见的二元分类(Binary Classification)问题的预测目标只有两个离散值，如{0,1}。回归模型得到的连续值可以通过一个非线性函数映射为离散值，或者取某个离散值的概率。

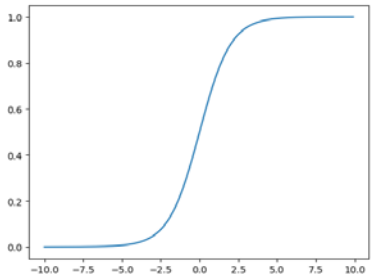
逻辑回归

基于线性模型的逻辑回归模型定义如下

$$h_\theta(x) = g(\theta^T x)$$

函数$g(z)$在神经网络中被称为激活函数(Active Function)。这里取 Sigmoid 函数（也称为 Logistic 函数，或者对数几率函数、对率函数）

$$g(z) = \frac{1}{1+e^{-z}} \triangleq \sigma(z)$$



设预测值$\hat{y} = 1$或$\hat{y} = 0$为

$$\begin{cases} P(y=1|x;\theta) = h_\theta(x) \\ P(y=0|x;\theta) = 1 - h_\theta(x) \end{cases}$$

合并得

$$P(y|x;\theta) = h_\theta(x)^y \left(1 - h_\theta(x)\right)^{1-y}$$

似然函数

$$\begin{aligned} L(\theta) &= P(y|X;\theta) \\ &= \prod_{(i=1)}^{m} P\left(y^{(i)}|x^{(i)};\theta\right) \\ &= \prod_{i=1}^{m} h_\theta\left(x^{(i)}\right)^{y^{(i)}} \left(1-h_\theta\left(x^{(i)}\right)\right)^{1-y^{(i)}} \end{aligned}$$

取对数

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right) \log(1-h_\theta\left(x^{(i)}\right) \end{aligned}$$

这里的而目标是最大化$\ell(\theta)$，可以采用梯度上升法(Gradient Ascend)迭代计算参数的最优值

$$\theta := \theta + \alpha \nabla_\theta \ell(\theta)$$

其中

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{m} (y^{(i)} - h_\theta\left(x^{(i)}\right)x$$

这个梯度在形式上与线性回归相同。

3.2. 指数族分布(Exponential Family Distribution)

形式

$$p(y;\eta) = b(y)\exp\left(\eta^T T(y) - a(\eta)\right)$$

其中，η称作自然参数(Natural Parameter)，$T(y)$为对η的充分统计量(Sufficient Statistic)，通常$T(y)=y$。

常用指数族分布

伯努利分布

$$p(y;\phi) = \phi^y (1-\phi)^{1-y}$$

写成指数族分布

$$p(y;\phi) = \exp\left(\ln\left(\frac{\phi}{1-\phi}\right)y + \ln(1-\phi)\right)$$

其中$\eta^T = \ln\left(\frac{\phi}{1-\phi}\right)$，$-a(\eta) = \ln(1-\phi)$，$b(y)=1$。所以$\phi = \frac{1}{1+e^{-\eta}}$。

高斯分布

$$p(y;\mu,\sigma=1) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2}\right)$$

写成指数族分布

$$p(y;\mu) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y^2\right)\exp\left(\mu y - \frac{1}{2}\mu^2\right)$$

其中$b(y) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y^2\right)$，$\eta=\mu$，$a(\eta) = \frac{1}{2}\mu^2$。

3.3. 广义线性模型(Generalized Linear Model)

模型假设

1. $y|x;\theta \sim \text{ExpFamily}(\eta)$
2. 给定x，目标为输出$E(T(y)|x)$，即学习模型$h(x)=E(T(y)|x)$
3. 令$\eta=\theta^T x$，更一般的对于向量η，令$\eta_i=\theta_i^T x$

伯努利分布

$$h_\theta(x)=E(y|x;\theta)=p(y|x;\theta)=\phi=\frac{1}{1+e^{-\eta}}=\frac{1}{1+e^{-\theta^T x}}$$

称$g(\eta)=E(y;\eta)=\frac{1}{1+e^{-\eta}}$为规范响应函数(Canonical Response Function)，g^{-1}为规范连接函数(Canonical Link Function)。

多项式分布[Multinomial Distribution，**多努利分布(**Multinoulli Distribution)]
离散型随机变量有k个不同的状态，它们的概率分别为$\phi_1,\phi_2,\dots,\phi_k$。

定义向量$T(i)=\begin{bmatrix}0 & \dots & \overset{i-th}{1} & \dots & 0\end{bmatrix}^T$，其中$T(i)_j=1\{j=1\}$，$1\{*\}$为指示函数，大括号内的命题为真则函数值为1，反之为0。
概率质量函数为

$$p(y|\phi)=\prod_{i=1}^{k}\phi_i^{1\{y=i\}}=\prod_{i=1}^{k}\phi_i^{T(y)_i}=\exp\left(\sum_{i=1}^{k}T(y)_i\log\phi_i\right)$$

注意到最后一项$T(y)_k=1-\sum_{i=1}^{k-1}T(y)_i$。

$$p(y|\phi)=\exp\left(\sum_{i=1}^{k-1}\log\left(\frac{\phi_i}{\phi_k}\right)T(y)_i+\log\phi_k\right)=\exp\left(\left[\left(\frac{\phi_i}{\phi_k}\right)_{i=1\dots k-1}\right]^T T(y)+\log\phi_k\right)$$

此时$\eta=\begin{bmatrix}\log\frac{\phi_1}{\phi_k}\\ \vdots \\ \log\frac{\phi_{k-1}}{\phi_k}\end{bmatrix}\in\mathbb{R}^{k-1}$，$a(\eta)=-\log\phi_k$，$b(y)=1$。

可以解出

$$\phi_i=\frac{e^{\eta_i}}{1+\sum_{j=1}^{k-1}e^{\eta_j}}=\frac{e^{\theta_i^T x}}{1+\sum_{j=1}^{k-1}e^{\theta_j^T x}}\qquad i=1\dots k-1$$

令$h_\theta(x)=E(T(y)|x,\theta)=[(\phi_i)_{i=1\dots k-1}]^T$。
这个模型称为 Softmax 回归(Softmax Regression)。优化方法同样为令似然函数最大化。这里的 Softmax 函数可以看做是对第$1\dots k-1$类首先拟合出$n-1$个线性模型，然后对线性模型的输出取指数函数，然后做平均，其中最后一项的权重始终设为0。更一般的，为k生成k个线性模型，然后对这k个输出分别取对数后做平均，即

$$\text{Softmax}(x)_i=\frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}$$

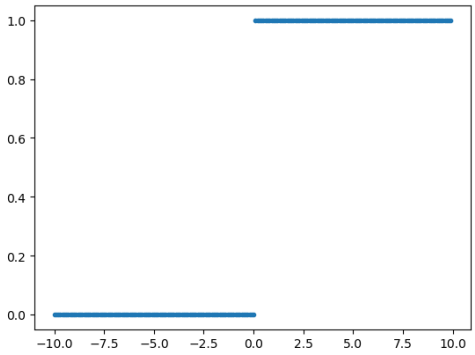
这个函数的一个变形为 Log Softmax，即对 Softmax 的结果再取对数。

$$\text{LogSoftmax}(x)_i=\log(\text{Softmax}(x)_i)=\log\left(\frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}\right)$$

3.4. 感知机(Perceptron)

定义

$$g(z) = \begin{cases} 1, & if\ \ z \geq 0 \\ 0, & otherwise \end{cases}$$



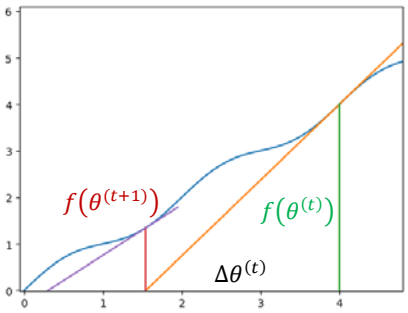
学习规则

$$\theta := \theta + \alpha\left(y^{(i)} - h\left(\theta^T x^{(i)}\right)\right)x^{(i)}$$

其中α为学习速率。

3.5. 牛顿法(Newton's Method)

牛顿法(Newton's Method)是一种寻找零点的迭代计算方法。



目标为寻找$f(\theta)$的零点，每次迭代，变量的更新量为

$$\Delta\theta = \frac{f(\theta)}{f'(\theta)}$$

迭代步骤

$$\theta^{(t+1)} := \theta^{(t)} - \frac{f\left(\theta^{(t)}\right)}{f'(\theta^{(t)})}$$

如果目函数为$f'(\theta)$，例如令$\ell'(\theta) = 0$，对应修改为：

$$\theta^{(t+1)} := \theta^{(t)} - \frac{\ell'\left(\theta^{(t)}\right)}{\ell''(\theta^{(t)})}$$

牛顿迭代的收敛速度为二次收敛，理论上在最优解附近时，每次迭代可以将误差缩小到原上一次的平方。

更一般的，对于多维的情况，假设$f(\theta)$的二阶偏导数连续，对$f(\theta)$进行泰勒展开，可得

$$f\left(\theta + (\Delta\theta)\right) \approx f(\theta) + g^T(\Delta\theta) + \frac{1}{2}(\Delta\theta)^T H(\Delta\theta)$$

其中g为梯度，H为海森矩阵(Hessian Matrix)

$$H_{ij}=\frac{\partial^2 f}{\partial\theta_i\partial\theta_j}$$

假设参数每次更新最优的变化量是$\Delta\theta$

$$f(\theta+\Delta\theta)\approx f(\theta)+(\Delta\theta)^T g+\frac{1}{2}(\Delta\theta)^T H(\Delta\theta)$$

函数取得极小值的必要条件是梯度为0，即令

$$\nabla_{(\Delta\theta)}\left((\Delta\theta)^T g-\frac{1}{2}(\Delta\theta)^T H(\Delta\theta)\right)=0$$

即

$$g+\frac{1}{2}H(\Delta\theta)+\frac{1}{2}H^T(\Delta\theta)=0$$

因为$f(\theta)$的二阶偏导数存在且连续，$H=H^T$，因此

$$(\Delta\theta)=-H^{-1}g$$

由此导出牛顿迭代法为

$$\theta^{(t+1)}\coloneqq\theta^{(t)}-H(f)^{-1}\nabla_\theta f$$

通常牛顿法要比梯度下降法快很多，但是注意海森矩阵是一个$n\times n$的矩阵，计算它的逆时间复杂度为$O(n^3)$，如果参数规模较大，计算海森矩阵的逆会非常慢。此外，当海森矩阵非正定时，所求得的点并不是局部极小值，海森矩阵与极值的关系如下

1) 当H正定时，为极小值；当H负定时，为极大值；
2) 当$|H|=0$时，需要更高阶矩阵判断其类型；
3) 其他情况时，为鞍点。

为了避免震荡，通常在更新时施加一个步长λ，即

$$\theta^{(t+1)}\coloneqq\theta^{(t)}-\lambda H(f)^{-1}\nabla_\theta f$$

求解广义线性模型的损失函数最小化时，可以使用一种牛顿法的变种，称作迭代重加权最小二乘法(Iteratively Reweighted Least Squares, IRLS)。

1) 对于线性回归问题，损失函数为

$$J(\theta)=\frac{1}{2}(X\theta-y)^2$$

梯度为

$$g(\theta)=X^T(X\theta-y)$$

海森矩阵为

$$H(\theta)=X^TX$$

更新公式为

$$\theta^{(t+1)}\coloneqq\theta^{(t)}-(X^TX)^{-1}X^T(X\theta-y)=(X^TX)^{-1}X^Ty$$

即标准的最小二乘法导出的正规方程。

2) 对于逻辑回归问题，损失函数为

$$J(\theta)=\sum_{i=1}^{m}\left(y^{(i)}\ln h_\theta\left(x^{(i)}\right)+\left(1-y^{(i)}\right)\ln\left(1-h_\theta\left(x^{(i)}\right)\right)\right)$$

梯度为

$$g(\theta)=X^T(h_\theta(X)-y)$$

海森矩阵为

$$H(\theta)=\sum_{i=1}^{m}h_\theta\left(x^{(i)}\right)\left(1-h_\theta\left(x^{(i)}\right)\right)x^{(i)}{x^{(i)}}^T=X^TRX$$

其中R为对角矩阵

$$R_{ii}=h_\theta\left(x^{(i)}\right)\left(1-h_\theta\left(x^{(i)}\right)\right)x^{(i)}$$

更新公式为

$$\begin{aligned}\theta^{(t+1)}&\coloneqq\theta^{(t)}-(X^TRX)^{-1}X^T(h_\theta(X)-y)\\&\coloneqq(X^TRX)^{-1}\left(X^TRX\theta^{(t)}-X^T(h_\theta(X)-y)\right)\\&\coloneqq(X^TRX)^{-1}X^TRZ\end{aligned}$$

其中

$$Z=X\theta^{(t)}-R^{-1}(h_\theta(X)-y)$$

注意到，其中的$x^{(i)}{x^{(i)}}^T$为不变量，可以仅计算一次。

3.6. 拟牛顿法(Quasi-Newton Method)

使用牛顿法需要计算目标函数的二阶偏导数，计算复杂度较大，为$O(n^2)$。而且有的时候，目标函数的海森矩阵无法保持正定，从而导致牛顿法失效。为了克服这些问题，人们提出了拟牛顿法(Quasi-Newton Method)，也称准牛顿法。

其核心思想是寻求一个矩阵近似海森矩阵H或者海森矩阵的逆矩阵H^{-1}，分别记为$B\approx H$和$D\approx H^{-1}$，它们应满足如下所示的拟牛顿条件(Quasi-Newton Condition)。

拟牛顿条件

对$f(\theta+\Delta\theta)$做泰勒展开

$$f(\theta+\Delta\theta)\approx f(\theta)+(\Delta\theta)^T\nabla_\theta f(\theta)+\frac{1}{2}(\Delta\theta)^TH(\theta)(\Delta\theta)$$

两边同时对$\Delta\theta$取梯度得

$$\nabla_{\Delta\theta}f(\theta+\Delta\theta)\approx\nabla_\theta f(\theta)+H(\theta)(\Delta\theta)$$

于是

$$\nabla_\theta f(\theta+\Delta\theta)-\nabla_\theta f(\theta)\approx H(\theta)(\Delta\theta)$$

记

$$\begin{aligned}y_k&=g_{k+1}-g_k\\s_k&=\theta_{k+1}-\theta_k=\Delta\theta_k\end{aligned}$$

那么

$$\begin{aligned}y_k&\approx H_{k+1}s_k=B_{k+1}s_k\\s_k&\approx H_{k+1}^{-1}y_k=D_{k+1}y_k\end{aligned}$$

这称之为割线方程(Secant Equation)。

使用矩阵B或D更新自变量的方法与牛顿法相同，即

$$\theta_{k+1}=\theta_k-\lambda B_k^{-1}g_k=\theta_k-\lambda D_kg_k$$

在得到θ_{k+1}后可以计算出y_k和s_k，对矩阵B或D进行更新。拟牛顿法的核心在于如何求解未知的近似矩阵B或D。

DFP 算法

DFP 算法是最早提出的拟牛顿法，核心是通过迭代对H_{k+1}^{-1}进行近似，即

$$D_{k+1} = D_k + \Delta D_k$$

通常D_0取单位矩阵I。

使用待定法求解ΔD_k，设

$$\Delta D_k = \alpha u u^T + \beta v v^T$$

这可以保证ΔD_k为对称矩阵。将D_k带入到s_k中，得

$$s_k = D_k y_k + \alpha u u^T y_k + \beta v v^T y_k = D_k y_k + u(\alpha u^T y_k) + v(\beta v^T y_k)$$

不妨取

$$\alpha u^T y_k = 1$$
$$\beta v^T y_k = -1$$

那么

$$u - v = s_k - D_k y_k$$

不妨直接取

$$u = s_k$$
$$v = D_k y_k$$

可以解出α和β分别为

$$\alpha = \frac{1}{u^T y_k} = \frac{1}{s_k^T y_k}$$
$$\beta = -\frac{1}{v^T y_k} = -\frac{1}{y_k^T D_k y_k}$$

第二个式子利用了$D_k = D_k^T$。

综上，校正矩阵ΔD_k为

$$\Delta D_k = \frac{s_k s_k^T}{s_k^T y_k} - \frac{D_k y_k y_k^T D_k}{y_k^T D_k y_k}$$

BFGS 算法

BFGS 算法直接对海森矩阵H_k进行逼近。与 DFP 算法相比，它的性能更加，并且有着较为完善的局部收敛理论，对其全局收敛性的研究也取得了重要成果，目前已经成为了求解无约束非线性优化问题最常用的算法之一。

对矩阵B的更新公式如下

$$B_{k+1} = B_k + \Delta B_k$$

同样设

$$\Delta B_k = \alpha u u^T + \beta v v^T$$

带入y_k得

$$y_k = B_k s_k + \alpha u u^T s_k + \beta v v^T s_k = B_k s_k + u(\alpha u^T s_k) + v(\beta v^T s_k)$$

不妨取

$$\alpha u^T s_k = 1$$
$$\beta v^T s_k = -1$$

那么

$$y_k - B_k s_k = u - v$$

不妨直接取

$$u = y_k$$
$$v = B_k s_k$$

可以解得

$$\alpha = \frac{1}{u^T s_k} = \frac{1}{y_k^T s_k}$$
$$\beta = -\frac{1}{v^T s_k} = -\frac{1}{s_k^T B_k s_k}$$

因此

$$\Delta B_k = \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$$

使用 BFGS 算法时，更新方向$d_k = B_k^{-1} g_k$通常通过求解线性方程组$B_k d_k = g_k$得到，但更一般的做法是，通过 Sherman-Morrison 公式直接求出B_{k+1}和B_k的递推关系

$$B_{k+1}^{-1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) B_k^{-1} \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}$$

展开得

$$B_{k+1}^{-1} = B_k^{-1} + \left(\frac{1}{s_k^T y_k} + \frac{y_k^T B_k^{-1} y_k}{(s_k^T y_k)^2}\right) s_k s_k^T - \frac{1}{s_k^T y_k}(B_k^{-1} y_k s_k^T + s_k y_k^T B_k^{-1})$$

L-BFGS 算法

在使用 DFP 算法或是 BFGS 算法时，需要储存矩阵D_k或B_k，其内存开销为$O(n^2)$。当特征维度很大时，带来的内存开销也非常大。L-BFGS(Limited-memory BFGS)算法是 BFGS 的一种近似算法，它在迭代过程中，不再储存完整的矩阵B_k^{-1}，而是储存计算过程中的向量序列$\{s_i\}$和$\{y_i\}$，需要矩阵B_k^{-1}时，利用向量序列的进行代替。在 L-BFGS 算法中，只有最新的$l+1$组向量被储存，即$\{s_k, s_{k-1}, \dots, s_{k-l}\}$和$\{y_k, y_{k-1}, \dots, y_{k-l}\}$。

记

$$\rho_k = \frac{1}{y_k^T s_k}$$
$$V_k = I - \rho_k y_k s_k^T$$

矩阵B_{k+1}^{-1}的递推式可以写成

$$B_{k+1}^{-1} = V_k^T B_k^{-1} V_k + \rho_k s_k s_k^T$$

展开得

$$\begin{aligned}
B_{k+1}^{-1} &= V_k^T B_k^{-1} V_k + \rho_k s_k s_k^T \\
&= V_k^T (V_{k-1}^T B_{k-1}^{-1} V_k + \rho_{k-1} s_{k-1} s_{k-1}^T) V_{k-1}^T + \rho_k s_k s_k^T \\
&= \cdots \\
&= \\
&\quad (V_k^T V_{k-1}^T \cdots V_0^T) B_0^{-1} (V_0 \cdots V_{k-1} V_k) \\
&+ (V_k^T V_{k-1}^T \cdots V_1^T)(\rho_0 s_0 s_0^T)(V_1 \cdots V_{k-1} V_k) \\
&+ \cdots \\
&+ V_k^T (\rho_{k-1} s_{k-1} s_{k-1}^T) V_k^T \\
&+ \rho_k s_k s_k^T
\end{aligned}$$

注意到ρ_k和V_k的计算只依赖于s_k和y_k，L-BFGS 只保留最新的$l+1$组向量，因此更新公式中此前的部分都被抛弃，即

$$\begin{aligned}
B_{k+1}^{-1} &= (V_k^T V_{k-1}^T \cdots V_{k-l+1}^T) B_0^{-1} (V_{k-l+1} \cdots V_{k-1} V_k) \\
&+ (V_k^T V_{k-1}^T \cdots V_{k-l+2}^T)(\rho_{k-l+1} s_{k-l+1} s_{k-l+1}^T)(V_{k-l+2} \cdots V_{k-1} V_k) \\
&+ \cdots \\
&+ V_k^T (\rho_{k-1} s_{k-1} s_{k-1}^T) V_k^T \\
&+ \rho_k s_k s_k^T
\end{aligned}$$

更新方向$d_k = B_k^{-1}g_k$通过如下所示的快速方法计算。

```
// 初始化
δ = 0 if k ≤ l else k − l
L = k if k ≤ l else l
// 后向循环
for i = L − 1 to 0
    j = i + δ
    α_i = ρ_j s_j^T q_{i+1}
    q_i = q_{i+1} − α_i y_j
// 前向循环
r_0 = B_0^{-1} q_0
for i = 0 to L − 1
    j = i + δ
    β_j = ρ_j y_j^T r_i
    r_{i+1} = r_i + (α_i − β_i) s_j
输出 r_L = B_k^{-1} g_k
```

考虑到B_0^{-1}只需要计算一次，因此通过上述算法计算$B_k^{-1}g_k$的时间复杂度为$O(ln)$，空间复杂度为$O(ln)$。

参考：https://blog.csdn.net/itplus/article/details/21896453

补充材料——充分统计量(Sufficient Statistic)

对于样本X，如果统计量$T(X)$蕴含了参数θ的全部信息，即

$$P(X|T(X),\theta) = P\big(X\big|T(X)\big)$$

称$T(X)$是对θ的充分统计量(Sufficient Statistic for θ)。

费舍尔分解定理(Fisher Factorization Theorem)

如果$T(X)$是对θ的充分统计量，当且仅当X的概率密度函数能够分解为

$$P(X) = h(X)g\big((T(X),\theta)\big)$$

其中h, g均为非负实值函数。

直观上理解

注意到充分统计量的定义，仅描述了$T(x)$确定后$P\big(X|T(X)\big)$与参数θ无关。而概率密度公式由两部分组成，$T(X)$确定后的部分和确定$T(X)$的部分，即

$$P(X) = P(X,T(X);\theta) = P\big(X|T(X)\big)P(T(X);\theta) = f(X)g_\theta\big(T(X)\big)$$

这里的$g_\theta\big(T(X)\big)$不一定是$T(X)$的概率密度函数，但是可以看出它与$T(X)$概率密度函数的商，一定是关于X的函数。直观上理解，$T(X)$蕴含了θ所能提供的全部信息，那么它的概率密度函数蕴含着全部的θ项使得剩余部分不包含θ。

补充材料——Sherman-Morrison 公式(Sherman-Morrison Equation)

设$A \in \mathbb{R}^{n\times n}$为非奇异方针，$u, v \in \mathbb{R}^n$，若$1 + v^T A^{-1}u \neq 0$，则有

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}$$

4. 生成式算法

4.1. 生成式算法(Generative Algorithm)

对所属不同类别的数据首先分别建模，使用后验概率对数据进行判别。推理被分为两个部分：1. 某类别的先验概率；2. 如果是该类别，产生这样数据的概率。推理过程为

$$\begin{aligned} y &= \underset{y}{\operatorname{argmax}} P(y|x) \\ &= \underset{y}{\operatorname{argmax}} \frac{P(x|y)P(y)}{P(x)} \\ &= \underset{y}{\operatorname{argmax}} P(x|y)P(y) \end{aligned}$$

学习方法

$$\theta = \underset{\theta}{\operatorname{argmax}} P(X,y) = \underset{\theta}{\operatorname{argmax}} P(x|y)P(y) = \underset{\theta}{\operatorname{argmax}} \log P(X|y) + \log P(y)$$

依旧为最大似然估计。

4.2. 高斯判别分析(Gaussian Discriminant Analysis, GDA)

多元高斯分布(Multivariate Gaussian Distribution)

$$x \sim \mathcal{N}(\mu,\Sigma) = (2\pi)^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

其中μ为一个n维向量表示各维度均值，Σ为协方差矩阵。有时候会设为一个对角矩阵，此时假设变量各维度线性无关；更进一步，可以设置Σ为实数乘单位阵I，这种分布称为各向同性(isotropic)的高斯分布。

假设y服从伯努利分布$B(\phi)$，分别假设将两个类别数据的后验分布服从高斯分布，并且具有同样的协方差矩阵为

$$P(x|y=i;\phi,\mu_0,\mu_1,\Sigma) = (2\pi)^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)\right) \quad i = 0,1$$

对联合似然函数(Joint Likelihood Function)取 Log 得

$$\ell(\phi,\mu_0,\mu_1,\Sigma) = \log\prod_{i=1}^{m}P\left(x^{(i)},y^{(i)}\right) = \sum_{i=1}^{m}\log P\left(x^{(i)},y^{(i)}\right) = \sum_{i=1}^{m}\log P\left(x^{(i)}|y^{(i)}\right) + \log P\left(y^{(i)}\right)$$

最大似然估计为

$$\begin{aligned} \phi &= \frac{\sum_i y^{(i)}}{m} \\ \mu_0 &= \frac{\sum_{i=1}^m 1\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^m 1\{y^{(i)}=0\}} \\ \mu_1 &= \frac{\sum_{i=1}^m 1\{y^{(i)}=1\}x^{(i)}}{\sum_{i=1}^m 1\{y^{(i)}=1\}} \\ \Sigma &= \frac{1}{m}\sum_{i=1}^{m}\left(x-\mu_{y^{(i)}}\right)\left(x-\mu_{y^{(i)}}\right)^T \end{aligned}$$

其实这些值就是样本中同样中对应的统计量，注意到协方差矩阵Σ的估计是有偏的。

后验概率$y=1|x$通过如下公式计算

$$P(y=1|x)=\frac{P(x|y=1)P(y=1)}{P(x)}=\frac{P(x|y=1)P(y=1)}{P(x|y=1)P(y=1)+P(x|y=0)P(y=0)}$$

高斯判别分析与逻辑回归对比

高斯判别分析除了逻辑回归的基础假设$y|x\sim B(\phi)$之外，更进一步的假设了$x|y\sim\mathcal{N}(\mu,\Sigma)$。也就是高斯判别分析有着比逻辑回归更强的假设，那么它可以更为有效地利用数据，如果数据的真实分布的确是多元高斯分布，那么模型可以用更少的数据产生更好的效果。相反，如果这一假设是错误的，$x|y$的真实分布并不符合多元高斯分布，那么 GDA 将产生比逻辑回归更差的效果。正是因为逻辑回归对模型假设的正确性不敏感，即鲁棒性更强，因此逻辑回归的使用率更高。

可以证明，如果$x|y=0\sim ExpFamily_0(\eta_0)$，$x|y=1\sim ExpFamily_1(\eta_1)$，即两种情况下$x$均服从任意的指数族分布，最终的$P(y=1|x)$仍为逻辑回归函数。

4.3. 线性判别分析与二次判别分析(Linear/Quadratic Discriminant Analysis)

使用生成式算法解决二元分类问题的最终判别依据为$y=\underset{y}{\operatorname{argmax}}\,P(x|y)P(y)$，可以改写为

$$y=\underset{y}{\operatorname{argmax}}\log\big(P(x|y)P(y)\big)$$

对$y=0$和$y=1$的情况作差

$$D(x)=\log P(x|y=0)P(y=0)-\log P(x|y=1)P(y=1)$$

显然，$D(x)>0$时，取$y=0$，反之取$y=1$，称$D(x)$为判别式。

线性判别分析

贝叶斯解释

上述高斯判别分析中假设$x|y=0$和$x|y=1$具有相同的协方差Σ。将参数带入$\log P(x|y)P(y)$后，包含x的部分为$-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)$，即

$$\begin{aligned}D(x)&=-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)+\log\phi-\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)+\log(1-\phi)\right)\\&=(\mu_0-\mu_1)^T\Sigma^{-1}x-\frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0-\mu_1\Sigma^{-1}\mu_1)+\log\frac{\phi}{1-\phi}\\&=(\mu_0-\mu_1)^T\Sigma^{-1}x-\frac{1}{2}(\mu_0+\mu_1)^T\Sigma^{-1}(\mu_0-\mu_1)+\log\frac{\phi}{1-\phi}\end{aligned}$$

可以看出，由于在判别式中中$\log P(x|y=0)$和$\log P(x|y=1)$两项相减会抵消掉x的二次项，剩余部分关于x的最高次项为一次项。因此，称不同组数据具有相同协方差的高斯判别模型之为线性判别分析。

值得注意的是，因为这是一个二分类问题，我们可以通过平移数据，使得$\mu_0+\mu_1=0$，消去$-\frac{1}{2}(\mu_0+\mu_1)^T\Sigma^{-1}(\mu_0-\mu_1)$。如果两类数据同先验，$\log\frac{\phi}{1-\phi}=\log\frac{0.5}{0.5}=0$。此时最简单的判别式为

$$D^*(x)=(\mu_0-\mu_1)^T\Sigma^{-1}x$$

Fisher 判别分析(Fisher Discriminant Analysis)
线性判别分析的另一种导出方法由 Fisher 提出，又称为 Fisher 判别分析。
对于线性模型$y = w^T x$，本质上是将原始数据投射到实数轴上，然后根据与原点的关系对类别进行判定。投射后，两类均值分别为$w^T \mu_0$和$w^T \mu_1$，两类的协方差分别为$w^T \Sigma_0 w$和$w^T \Sigma_1 w$。希望这个投影变换能够尽可能的将两类数据分开，即使得类内距离（与各类均值的距离）最小化，类间距离（投射后协方差）最大化，即最大化

$$J = \frac{\|w^T \mu_0 - w^T \mu_1\|^2}{w^T \Sigma_0 w + w^T \Sigma_1 w} = \frac{w^T (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T w}{w^T (\Sigma_0 + \Sigma_1) w}$$

其中，类间散度矩阵(Between-class Scatter Matrix)为

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

类内散度矩阵(Within-Class Scatter Matrix)为

$$S_w = \Sigma_0 + \Sigma_1$$

可以将优化目标重写为

$$J = \frac{w^T S_b w}{w^T S_w w}$$

不妨约束$w^T S_w w = 1$，问题改写为

$$\min_w -w^T S_b w$$
$$s.t. \quad w^T S_w w = 1$$

使用拉格朗日乘数法(Lagrangian Multiplier)，上述等式约束问题等价于

$$S_b w = \lambda S_w w$$

观察S_b表达式，显然$S_b w$与$\mu_0 - \mu_1$同方向，因此可以设

$$S_b w = \lambda(\mu_0 - \mu_1)$$

代入得

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

因此模型可以写成

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

如果两类数据具有相同协方差矩阵，并且具有相同先验分布，那么线性判别分析产生了最优的贝叶斯分类器。
对于多分类问题，设定存在N个类，那么可以定义全局散度矩阵为

$$S_t = S_b + S_w = \sum_{i=1}^{m} \left(x^{(i)} - \mu\right)\left(x^{(i)} - \mu\right)^T$$

其中μ是所有输入特征的均值向量。类内散度矩阵依旧为各类散度矩阵之和为

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

类间散度矩阵为

$$S_b = S_t - S_w = \sum_{i=1}^{N} (\mu_i - \mu)(\mu_i - \mu)^T$$

一种常用的模型为，设定一个投影矩阵$W \in \mathbb{R}^{n \times (N-1)}$，将原始的向量投射到$n'$空间，因为$S_b$的秩最多为$N-1$，因此$n' \leq N-1$。优化目标为最大化投影后的各类均值间的距离和，最小化类内的协方差之和，即

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

可以通过定义如下广义特征值问题求解，即

$$S_b W = \lambda S_w W$$

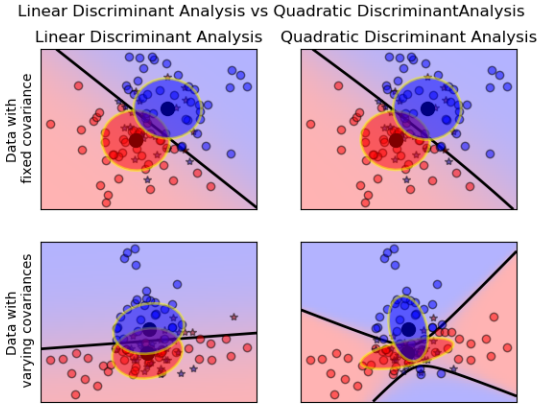
两侧同时乘以S_w，可以写成矩阵乘以特征向量矩阵的形式，即

$$S_w^{-1} S_b W = \lambda W$$

矩阵W的闭式解为$S_w^{-1}S_b$的n'个最大非零广义特征值对应的特征向量组成的矩阵，可以看做是将数据投射到了类内散度最小，类间散度最大的维度。因为W将数据投射到了更低维的空间中，因此线性判别分析常用于有监督降维(Dimension Reduction)。

二次判别分析

如果假设$x|y=0$和$x|y=1$具有不同的协方差Σ_0和Σ_1，判别式中的二次项不会相互抵消，因此判别式的最高次项为二次项。因此，称不同组数据有不同协方差的高斯判别模型为二次判别分析。



4.4. 朴素贝叶斯(Naive Bayes)

朴素贝叶斯假设(Naïve Bayes Assumption)

变量$x_1, x_2, \ldots, x_n$关于y条件独立，即$P(x_1, x_2, \ldots, x_n, y) = \prod_{i=1}^n P(x_i|y)P(y)$。

朴素贝叶斯模型(多变量伯努利事件模型, Multi-variate Bernoulili Event Model)

假设数据具有n个特征$x_1, x_2, \ldots, x_n$，在给定随机变量$y \sim B(\phi_y)$后条件独立，那么y关于x条件概率为

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{\prod_{i=1}^n P(x_i|y)P(y)}{P(x)}$$

假设其中条件分布$x_i|y$均为伯努利分布，即$x_i|y=0 \sim B(\phi_{i|y=0})$，$x_i|y=1 \sim B(\phi_{i|y=1})$

需要学习的参数包括y的先验概率ϕ_y、后验概率$P(x_i|y=1)=\phi_{i|y=0}$、$P(x_i|y=1)=\phi_{i|y=1}$。

联合似然函数

$$\mathcal{L}(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) = \prod_{i=1}^m P(x^{(i)}, y^{(i)}) = \prod_{i=1}^m \prod_{j=1}^n P\left(x_j^{(i)} \middle| y^{(i)}\right) P\left(y^{(i)}\right)$$

最大似然估计为

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$
$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\right\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$
$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$

其实它们就是样本中的对应的统计量。
后验概率$P(y=1|x)$通过如下公式计算

$$P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x)} = \frac{\prod_{i=1}^{n} P(x_i|y=1)P(y=1)}{P(x|y=0)P(y=0) + P(x|y=1)P(y=1)}$$

注意到朴素贝叶斯模型中，$x|y=0 \sim ExpFamily_0(\eta_0)$，$x|y=0 \sim ExpFamily_0(\eta_0)$，因此概率密度函数$P(y=1|x)$同样是一个回归函数。

拉普拉斯平滑(Laplace Smoothing)
如果变量x_i在训练样本中从未出现过，那么$P(x_i=1|y)=0$。如果一个预测样本中出现了x_i，那么$P(y=0|x)$和$P(y=1|x)$的计算都会变成$0/0$。仅因为训练样本中没有出现该变量，就认为该变量永远不可能出现是不合理的。
拉普拉斯平滑为每种可能的状态在统计时额外增加一次，例如对$x \sim B(\phi)$，

$$\phi = \frac{\sum_{i=1}^{m} 1\{x^{(i)} = 1\} + 1}{\left(\sum_{i=1}^{m} 1\{x^{(i)} = 0\} + 1\right) + \left(\sum_{i=1}^{m} 1\{x^{(i)} = 0\} + 1\right)} = \frac{\sum_{i=1}^{m} 1\{x^{(i)} = 1\} + 1}{m+2}$$

对于具有k个不同取值的 Multinoulli 分布，分母应该增加所有可能取值的个数k，即

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{x^{(i)} = j\} + 1}{m+k}$$

4.5. 朴素贝叶斯模型的扩充(Ex. Naive Bayes)

1. 后验概率$P(x|y)$为多项式分布

允许$x|y$有k个取值，并服从多项式分布。常见的情况是，一个连续变量，对它进行离散化使其取k个值，通常$k=10$。

2. 多项式事件模型(Multinomial Event Model)

让x_i表示序列中第i个特征在词典中的索引，即$x_i \in \{1,2,\dots,|V|\}$，其中$|V|$是词典的大小。例如，令$x_i$表示邮件中第$i$个词在词典中的索引。假设词典有$|V|=50000$个词汇，第1000个词汇是“Hello”，$x_1=1000$此时表达的是邮件中第一个词汇是“Hello”。
假设对于所有的j，$P(x_j|y=i)$均服从同一个多项式分布，此时的参数为

$$\phi_{k|y=0} = P(x_j = k|y=0)$$
$$\phi_{k|y=1} = P(x_j = k|y=1)$$
$$\phi_y = P(y=1)$$

联合似然函数为

$$\mathcal{L}(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) = \prod_{i=1}^{m} P(x^{(i)}, y^{(i)})$$
$$= \prod_{i=1}^{m} \prod_{j=1}^{n_i} P\left(x_j^{(i)} \middle| y^{(i)}; \phi_{k|y=0}, \phi_{k|y=1}\right) P(y^{(i)}; \phi_y)$$

通过联合似然函数可以看出，如果一个特征，例如特征k，出现的次数越多，在联合似然函数中，$P(x_j = k|y;\phi_{k|y=0},\phi_{k|y=1})$的幂次越高，即对联合似然函数的影响越大。
举例而言，一个单词出现的次数越多，例如“Viagra”，为垃圾邮件的可能性就越大。
最大似然估计为

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 0\right\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} \cdot n_i}$$
$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} \cdot n_i}$$
$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y = 1\}}{m}$$

其中$\phi_{k|y=0}$的分子是所有$y = 0$时特征k出现的总数，分母是$y = 0$时所有的特征总量。例如，所有垃圾邮件中单词“Viagra”的总数和所有垃圾邮件中总共的单词数量。
对其进行拉普拉斯平滑，得到

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 0\right\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} \cdot n_i + |V|}$$
$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 1\right\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} \cdot n_i + |V|}$$
$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y = 1\} + 1}{m + 2}$$

补充材料——拉格朗日乘数法(Lagrangian Multiplier)

原始问题
对于在极值点ω^*处连续可微的函数f、h_i，约束优化问题为

$$\min_{\omega} f(\omega)$$
$$s.t. \quad h_i(\omega) = 0 \quad i = 1, \dots, m$$

拉格朗日算子
通过定义拉格朗日算子

$$\mathcal{L}(\omega, \beta) = f(\omega) + \sum_i \beta_i h_i(\omega)$$

将原始问题求极值转化为寻找一组ω^*和β^*使得$\frac{\partial \mathcal{L}}{\partial \omega} = 0$且$\frac{\partial \mathcal{L}}{\partial \beta} = 0$，此时得到的$\mathcal{L}$的极值也就是$f$的极值。其中$\beta_i$称为拉格朗日乘数。
证明
为了简化问题，只考虑一个约束条件。当$f(x)$于约束面$h(x) = 0$上取得极值时，应该有$f(x)$沿约束面切平面方向的梯度为0，此时无法再通过$f(x)$的梯度在$h(x) = 0$切平面上的投影移动x，因此达到了极值。此时$\nabla f(x)$与$\nabla h(x)$共线，可以表示为

$$\nabla f(x) = -\beta \nabla h(x)$$

为了方便表达，对两侧同时积分，得到函数$\mathcal{L}(x) = f(x) + \beta h(x)$。对$\mathcal{L}$关于$x$求梯度并令其等于0可以得到上式，关于$\beta$求梯度可以得到原始问题中的约束条件$h(x) = 0$。
对于多个约束条件，若使x不能再移动，应有f沿各约束面的切平面投影的和为零向量，此时$\nabla f(x)$为各约束面的梯度的线性组合，即

$$\nabla f(x) = -\sum_i \beta_i h(x)$$

5. 支持向量机

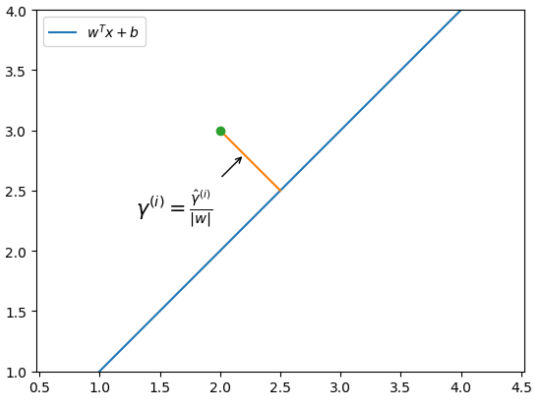
5.1. 支持向量分类(Support Vector Classification)

支持向量机(Support Vector Machine)使用数据集合边界的向量作为预测依据。使用支持向量机处理分类任务称为支持向量分类(Support Vector Classification)。

为了表示方便，对支持向量机中使用的符号规定如下。
类别$y \in \{-1,+1\}$，分割超平面为$z = w^T x + b$，其中x, θ为n维向量，$x = [x_1, x_2, \dots, x_n]^T$。

$$y = \begin{cases} 1, & w^T x \geq 0 \\ -1, & w^T x < 0 \end{cases}$$

函数间隔(Functional Margin)



超平面(w, b)关于数据$\left(x^{(i)}, y^{(i)}\right)$的函数间隔定义为

$$\hat{\gamma}^{(i)} = y^{(i)}\left(w^T x^{(i)} + b\right)$$

显然有当$y^{(i)}$与$w^T x^{(i)} + b$同号时，$\hat{\gamma}^{(i)}$为正数。函数间隔$\hat{\gamma}^{(i)}$越大，表明模型对预测正确的信念越大。
对于整个数据集的函数间隔定义为

$$\hat{\gamma}^{(i)} = \min_i \hat{\gamma}^{(i)}$$

优化目标为使全局函数间隔最大，即令离着分割线最近的数据，也就是最坏情况，能够和超平面分开尽可能大的距离。

几何间隔(Geometric Margin)
对于函数间隔，如果同时扩大w和b常数倍，那么函数间隔也会扩大同样的倍数。因此定义几何间隔为

$$\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\|w\|} = y^{(i)}\left(\frac{w^T}{\|w\|} x^{(i)} + \frac{b}{\|w\|}\right)$$

即对函数间隔标准化(Normalize)。由于$\|w\|$不影响$\gamma^{(i)}$，因此可以对w施加约束，例如令$\|w\| = 1$或是$|w_1| = 1$，或者$w_1^2 + |w_2| = 1$等，这样可以为求解带来便利。
对于整个数据集的几何间隔为

$$\gamma = \min_i \gamma^{(i)}$$

5.2. 最优间隔分类器(Optimal Margin Classifier)

优化目标

最大化对于整个数据集的最小几何间隔

$$\begin{aligned} &\max_{\gamma,w,b} \gamma \\ &s.t. \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq \gamma \quad i = 1, \dots, m \\ &\|w\| = 1 \end{aligned}$$

其中，$\|w\| = 1$使几何间隔等于函数间隔。然而这不是一个凸约束，因为$\|w\| = 1$表达了一个球面。修改为

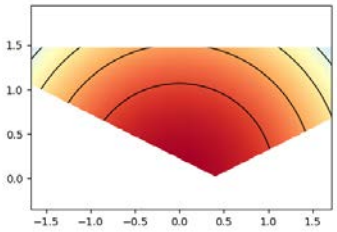
$$\begin{aligned} &\max_{\gamma,w,b} \frac{\hat{\gamma}}{\|w\|} \\ &s.t. \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq \hat{\gamma} \quad i = 1, \dots, m \end{aligned}$$

而这里$\frac{\hat{\gamma}}{\|w\|}$又是非凸的。

令$\hat{\gamma} = 1$，那么$\min_i y^{(i)}\left(w^T b^{(i)} + b\right) = 1$，最终的优化目标为

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ &s.t. \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 \quad i = 1, \dots, m \end{aligned}$$

此时，问题变成了带有线性约束的凸优化问题，此时可以通过凸优化软件求解。



更一般的，通过拉格朗日乘数法，可以将上述问题变换为

$$d^* = \max_{\alpha} \min_{w,b} \mathcal{L}(w, b, \alpha) = \max_{\alpha} \min_{w,b} \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y^{(i)}\left(w^T x^{(i)} + b\right)\right)$$

根据 KKT 条件，对w和b分别求偏导数并令其为0得

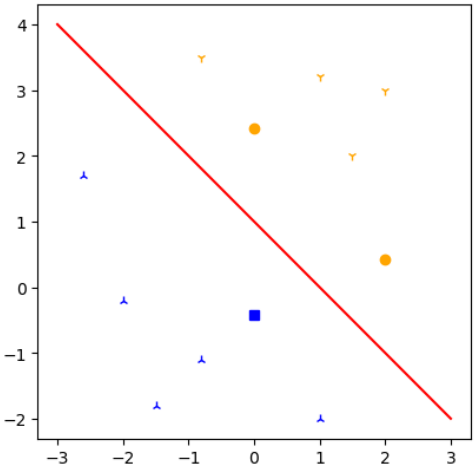
$$\begin{aligned} w &= \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \\ 0 &= \sum_{i=1}^{m} \alpha_i y^{(i)} \end{aligned}$$

带回$\mathcal{L}(w, b, \alpha)$中可以消去$w$和$b$，得到

$$d^* = \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} {x^{(i)}}^T x^{(j)}$$

其中$x^{(i)^T}x^{(j)}$可以写作内积形式$\langle x^{(i)}, x^{(j)}\rangle$。

注意到 KKT 的条件$\alpha_i \cdot \left(1 - y^{(i)}\left(w^T x^{(i)} + b\right)\right) = 0$。如果$\alpha_i = 0$，意味着这一个数据$\left(x^{(i)}, y^{(i)}\right)$没有对模型起作用；如果$\alpha_i \neq 0$，那么对应的约束为活跃约束(Active Constrain)，则说明$\left(x^{(i)}, y^{(i)}\right)$到超平面函数间隔$\gamma = 1$，即距离超平面最近。也就是说，只有距离分割超平面最近的数据对模型有效，这样的数据称为**支持向量**(Support Vector)。最终的模型复杂度仅依赖于极少数的支持向量。



因此，w实质上是支持向量的线性组合，$w^T x$也就是是x关于支持向量的加权内积。而$0 = \sum_{i=1}^{m} \alpha_i y^{(i)}$保证了尽管两侧的支持向量数量可能不同，但是对分割超平面的估计仍然是无偏的，这是线性回归所无法保障的，例如正样本多负样本少，线性回归会使得分割超平面偏向于负样本的一侧。另外，如果$\sum_{i=1}^{m} \alpha_i y^{(i)} \neq 0$，那么$\min_{w,b} \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y^{(i)}\left(w^T x^{(i)} + b\right)\right) = -\infty$，因为可以取绝对值任意大且与$\sum_{i=1}^{m} \alpha_i y^{(i)}$异号的$b$。

最终，假设$h(x)$可以写成

$$h(x) = w^T x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)^T} x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x\rangle + b$$

5.3. 核函数(Kernel Function)

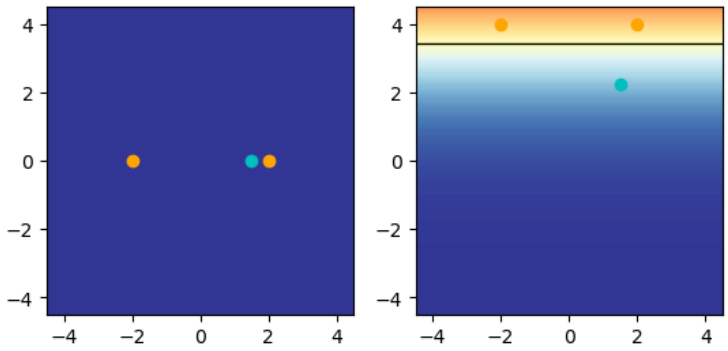
核函数(Kernel Function)

样本并非总是线性可分的，可以通过函数$\Phi(x)$将数据映射到更高维的空间，因此可以用$\Phi(x)$替代x。这样，模型也可以表示为

$$h(x) = g(w^T \Phi(x) + b)$$

在 SVM 算法中，只需要计算向量间的内积$\langle \Phi(x), \Phi(z)\rangle$，不需要用$\Phi(x)$本身，并且高维向量$\Phi(x)$间的内积计算很慢，因此可以使用一个函数$\kappa(x, z) = \langle \Phi(x), \Phi(z)\rangle$替代原始的内积，这成为核技巧(Kernel Trick)，函数κ被称为核函数(Kernel Function)。

例如，使用$\Phi(x) = [x \quad x^2]^T$，将线性不可分的数据$(-2,1)$、$(2,1)$和$(1.5,-1)$投射到二维空间，使其线性可分，对应的核函数$\kappa(x, z) = \langle [x \quad x^2]^T, [z \quad z^2]^T\rangle = xz + x^2 z^2$。



Mercer 定理(Mercer Theorem)

Mercer 定理给出了确定一个函数为核函数的充要条件。

设χ为输入空间，函数$\kappa(x,z)=\Phi(x)\Phi(z)$是定义在$\chi\times\chi$上的对称函数，则$\kappa$是核函数，当且仅当对于任意数据$D=\{x^{(1)},x^{(2)},\dots,x^{(m)}\}$，核矩阵(Kernel Matrix)$K$总是半正定的：

$$K=\begin{bmatrix}\kappa(x^{(1)},x^{(1)}) & \cdots & \kappa(x^{(1)},x^{(m)})\\ \vdots & \ddots & \vdots\\ \kappa(x^{(m)},x^{(1)}) & \cdots & \kappa(x^{(n)},x^{(m)})\end{bmatrix}$$

也就是说，对于一个核函数，只要它所对应的核矩阵是半正定矩阵，它就可以作为核函数。反之，如果核矩阵是半正定的，总能够找到一个与之对应的核函数。定理的必要性可以通过对于任意向量$z\in\mathbb{R}^m$，都有$z^TKz=\sum_{k=1}^m\left(\sum_{i=1}^m z_i\Phi(x^{(i)})_k\right)^2\geq 0$得到。

常用核函数

假设特征$x\in\mathbb{R}^n$。以下为常用的核函数。

线性核

$$\kappa(x,z)=x^Tz$$

取$\Phi(x)=x$时，原有的内积形式保持不变，即线性核。

多项式核

$$\kappa(x,z)=(x^Tz+c)^d\qquad d\in\mathbb{N}^+,c\geq 0$$

$\Phi(x)$有$\binom{n+d}{d}$个线性无关特征，如果$c=0$，则有$\binom{n+d-1}{d}$个线性无关特征。当次数$d=1$时，多项式核退化为线性核。

关于多项式核维度的证明

观察核函数表达式可以发现，括号内部是二次项，因此根据二项式定理，展开后的多项式的每一项都是2^d次项。因为$\kappa(x,z)=\langle\Phi(x),\Phi(z)\rangle$，因此向量$\Phi(x)$的每一项都是一个$d$次项，即$\prod_{i=1}^n x_i^{a(i)}\left(c^{1/d}\right)^b$，其中$a(i),b=0,1\dots,d$且$\sum_{i=1}^n a(i)+b=d$，所有线性无关的项恰好构成了所有的可能组合。这是一个组合问题，即如何将d次幂分配到$n+1$个不同项上。这个问题更形象的表达是：“有$n+1$个不同的杯子，向其中放入总共d个相同的小球，总共有多少种放置方法？”

可以把这个问题转化为一个向长度为$n+1$的队列中插入d个元素的问题，更进一步的，可以

描述为向$n+1$个 0 中，插入d个 1 的问题，其中 0 代表杯子，1 代表小球。对于插入后的队列中，一个 0 前面紧邻的且相连的 1 可以看做是这个杯子中放入的小球。
例如，有4个杯子，3个小球，插入后形成如下的序列：0110100，根据 0 划分为 0/110/10/0，即第1~4个杯子分别放入了0,2,1,0个小球。
因为小球总要放入杯子中，因此序列的最后一位总是 0。由此，问题转化为了：在前面的$n+d$个位置中，有d个 1 和$n-d$个 0，共有多少种不同的序列？这是一个朴素的组合问题，显然为$\binom{n+d}{d}$。
因此对于多项式核$\kappa(x^T z+c)^d$将原始数据映射到了$\binom{n+d}{d}$维空间。

高斯核

$$\kappa(x,z)=\exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)\quad \sigma>0$$

高斯核又称径向基函数核(Radial Basis Function Kernel)，其中σ为高斯核的带宽(width)。
对于高斯核函数，不妨取$\sigma=1$，在$x=z$处泰勒展开，可以写出如下形式的泰勒级数：

$$\begin{aligned}\kappa(x,z)&=\exp\left(-\frac{1}{2}\|x-z\|^2\right)\\&=\exp\left(-\frac{1}{2}(x-z)^T(x-z)\right)\\&=\exp\left(-\frac{1}{2}(x^Tx-z^Tx-x^Tz+z^Tz)\right)\\&=\exp(x^Tz)\exp\left(-\frac{1}{2}\|x\|^2\right)\exp\left(-\frac{1}{2}\|z\|^2\right)\\&=\sum_{i=0}^{\infty}\frac{(x^Tz)^i}{i!}\exp\left(-\frac{1}{2}\|x\|^2\right)\exp\left(-\frac{1}{2}\|z\|^2\right)\end{aligned}$$

从泰勒展开式中可以看出，高斯核函数是无穷次的多项式函数，因此$\Phi(x)$是无穷维向量。因此，高斯核将原始数据映射到了无穷维空间，因此通常将高斯核作为首选核函数。
对于每对(x,z)，高斯核函数的值总是在0到1之间，当x与z的欧氏距离无穷远时为0，相等时为1，因此高斯核函数是一种数据间的相似性度量。

拉普拉斯核

$$\kappa(x,z)=\exp\left(-\frac{\|x-z\|}{\sigma^2}\right)\quad \sigma>0$$

Sigmoid 核

$$\kappa(x,z)=\tanh(\beta x^Tz+\theta)\quad \beta>0,\theta<0$$

组合核函数

如果κ_1、κ_2是核函数，那么对于任意正数γ_1和γ_2的线性组合也是核函数

$$\gamma_1\kappa_1+\gamma_2\kappa_2$$

如果κ_1、κ_2是核函数，那么两函数直积也是核函数

$$\kappa_1\otimes\kappa_2=\kappa_1(x,z)\kappa_2(x,z)$$

如果κ_1是核函数，那么对于任意函数$g(x)$的如下变换也是核函数

$$\kappa(x,z)=g(x)\kappa_1(x,z)g(z)$$

5.4. L_1范数软间隔(L_1 Norm Soft Margin)

软间隔(Soft Margin)
并非所有的数据都是线性可分的，即使通过核函数将这些数据投射到高维空间，也可能因为这些在低维空间线性不可分或者离群的数据点造成过拟合。此前的要求中，每个数据都必须满足硬间隔(Hard Margin)，即

$$y^{(i)}\left(w^T x^{(i)}+b\right)\geq 1$$

现在放松这一条件，允许一些数据不满足约束条件，即它们的函数间隔小于1，甚至在超平面的另一侧，这称为软间隔。同时，不满足约束的数据应该尽可能的少，因此优化目标修改为：

$$\min_{w,b}\frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}1\{y^{(i)}\left(w^T x^{(i)}+b\right)<1\}$$

指示函数$1\{*\}$是非凸、非连续的，因此引入松弛变量(Slack Variables)$\xi_i\geq 0$替换指示函数，通过常量C控制松弛程度，优化目标最终重写为

$$\min_{w,b,\xi_i}\frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}\xi_i$$
$$s.t.\quad y^{(i)}\left(w^T x^{(i)}+b\right)\geq 1-\xi_i$$
$$\xi_i\geq 0,\qquad i=1,2,\dots,m$$

对应的拉格朗日算子为

$$\mathcal{L}(w,b,\xi,\alpha,\mu)=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}\xi_i+\sum_{i=1}^{m}\alpha_i\left(1-\xi_i-y^{(i)}\left(w^T x^{(i)}+b\right)\right)-\sum_{i=1}^{m}\mu_i\xi_i$$

根据 KKT 条件，对w,b,ξ_i分别求偏导并令偏导数为 0 得

$$w=\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}$$
$$0=\sum_{i=1}^{m}\alpha_i y^{(i)}$$
$$C=\alpha_i+\mu_i$$

根据 KKT 条件，应有$\mu_i\xi_i=0$，$\alpha_i\left(1-\xi_i-y^{(i)}\left(w^T x^{(i)}+b\right)\right)=0$，$\alpha_i\geq 0$，$\mu_i=C-\alpha_i\geq 0$。

综合不等式约束，可以得出α_i的取值范围

$$0\leq\alpha_i\leq C$$

将等式带入拉格朗日算子，可以得到优化目标为

$$d^*=\max_{\alpha}\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}{x^{(i)}}^T x^{(j)}$$
$$s.t.\quad 0\leq\alpha_i\leq C$$

可以发现，L_1范数软间隔的最终优化目标与硬间隔相比，只相差了约束条件中α_i的上限为常数C。
根据 KKT 条件，可以有如下结论

$$\begin{aligned} \alpha_i = 0 \quad &\Leftrightarrow \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 \\ \alpha_i = C \quad &\Leftrightarrow \quad y^{(i)}\left(w^T x^{(i)} + b\right) = 1 - \xi_i \leq 1 \\ 0 < \alpha_i < C \quad &\Leftrightarrow \quad y^{(i)}\left(w^T x^{(i)} + b\right) = 1 \end{aligned}$$

上述结论为寻找α_i的最优值提供了保障。

5.5. 序列最小优化算法(Sequential Minimal Optimization，SMO)

优化问题

支持向量机最终的优化问题是一个二次规划(Quadratic Programming, QP)问题，优化目标是最大化关于α的带约束的二次函数，可以使用通用的二次规划算法求解。由于问题的规模正比于训练样本数量，传统算法在实际任务中的开销较大。SMO 算法通过利用 SVM 中代优化目标的特性，解决了这一问题。

坐标上升法(Coordinate Ascent)

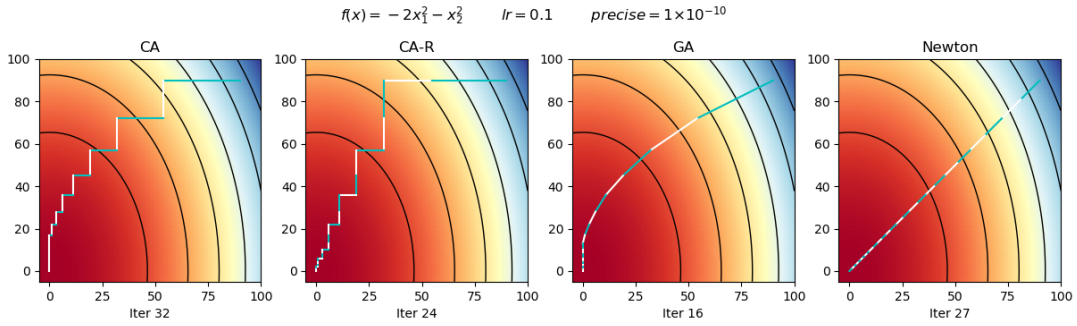
对于待优化的目标

$$\max_x f(\theta)$$

每次固定除了其中的一个变量θ_i外的所有其他变量，仅对θ_i执行梯度上升法。重复地按次序更新每一个变量，直到收敛。算法的一个变种是每次选择一个具有最大梯度变量进行更新，这样可以使算法收敛得更快。

通常，坐标上升法要比牛顿法使用更多的迭代次数，但是对于一些优化目标，固定其他参数，仅对一个参数求梯度代价很小，因此反而执行得很快。

例如优化函数为$f(x) = -2x_1^2 - x_2^2$，初始坐标为$(90,90)$，设定收敛精度为1×10^{-3}。最左侧图为坐标上升法，每次对其中一个分量进行更新，可以看出路径逐段平行于坐标轴；第二张图是每次选择最大梯度方向的梯度上升法，可以看到收敛到同样精度速度更快，每次选择的方向是梯度的方向；第三张图是梯度上升法，可以看出每次沿着局部梯度最大的方向移动，并且两个维度同时更新，速度快于坐标上升法；最右侧图为牛顿法，始终沿着全局梯度最大的方向移动变量，由于设置了较小的学习速率，因此速度反而慢于一次方法。



SMO

由于条件$\sum_{i=1}^m y^{(i)}\alpha_i = 0$的限制，坐标上升法并不能直接应用于求解 SVM 的优化问题。

SMO 的思路如下。

1. 每次启发式的选取两个变量α_i和α_j;
2. 保持其他的变量不变，利用约束$\sum_{i=1}^m y^{(i)}\alpha_i = 0$在可行域内对$\alpha_i$和$\alpha_j$进行更新。

变量选取

选取对应样本间隔最大的两个变量α_i和α_j。直观上看，由于这两个变量对应的样本差距很大，因此对这一对变量进行更新会给目标函数的值带来较大的变化。对应样本间的间隔可以通过构建核矩阵K预先得到。

变量更新

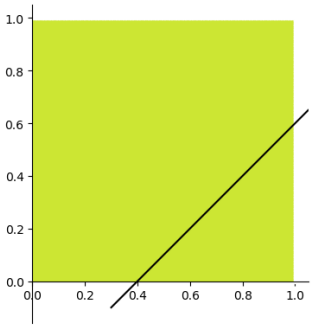
将约束重写为

$$\alpha_i y^{(i)} + \alpha_j y^{(j)} = -\sum_{k \neq i,j} \alpha_k y^{(k)} = \zeta$$

此时，变量α_i和α_j具有如下三个约束

$$0 \leq \alpha_i \leq C$$
$$0 \leq \alpha_j \leq C$$
$$\alpha_i y^{(i)} + \alpha_j y^{(j)} = \zeta$$

前两项构成了一个正方形的可行域，最后一项是一条穿过正方形的直线约束。



将α_i写为

$$\alpha_i = \frac{\zeta - \alpha_j y^{(j)}}{y^{(i)}}$$

带入d^*中，此时d^*变为了关于变量α_j的二次函数求最大值问题，易于求出最大值点，解得当前迭代下最优的α_i和α_j。如果点(α_i, α_j)位于正方形外，可以通过移动到最邻近的、直线与正方形的交点处。这样就可以保证在优化过程中，参数α始终满足约束条件。

截距的求解

求出α后，对于任意的支持向量，即$\alpha_i \neq 0$的向量，都满足

$$y^{(i)}\left(\sum_{j=1}^{m} \alpha_j y^{(j)} {x^{(j)}}^T x^{(i)} + b\right) = 1$$

取任意一个支持向量都可以解出b。基于数值稳定考虑，通常取所有支持向量计算出的结果取平均值。

5.6. 支持向量回归(Support Vector Regression)

依旧沿用线性回归模型

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

线性回归设计损失函数时，要求所有的数据点平均误差最小。而支持向量回归忽略在间隔带内，即与回归线距离小于ϵ的数据的误差。这样仅有一些距离回归线较远的误差会对模型产

生影响，使模型是稀疏(Sparse)的，那些处于隔离带边界的点即支持向量。另一方面，希望模型是尽可能平坦的，这可以通过加入正则项实现。
综合这两方面，可以写出如下的优化目标

$$\min_{w}\frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}\ell_\epsilon\left(f\left(x^{(i)}\right)-y^{(i)}\right)$$

其中$\ell_\epsilon(x)$是ϵ-不敏感损失(ϵ-insensitive loss)函数

$$\ell_\epsilon(x)=\begin{cases}0, & if\ \ |z|\le\epsilon\\ |z|-\epsilon, & otherwise\end{cases}$$

由于函数$\ell_\epsilon(x)$并不容易计算，可以将其替换为不小于0的松弛变量ξ，ξ^*，令它们分别表示间隔带两侧的松弛程度。优化目标修改为

$$\min_{w,b,\xi_i,\xi_i^*}\frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}(\xi_i+\xi_i^*)$$

$$s.t.\quad f\left(x^{(i)}\right)-y^{(i)}\le\epsilon+\xi_i$$
$$y^{(i)}-f\left(x^{(i)}\right)\le\epsilon+\xi_i^*$$
$$\xi_i\ge0$$
$$\xi_i^*\ge0$$

利用拉格朗日对偶法，可得对偶问题

$$d^*=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}(\xi_i+\xi_i^*)+\sum_{i=1}^{m}\mu_i(-\xi_i)+\sum_{i=1}^{m}\mu_i^*(-\xi_i^*)+\sum_{i=1}^{m}\alpha_i\left(f\left(x^{(i)}\right)-y^{(i)}-\epsilon-\xi_i\right)$$

$$+\sum_{i=1}^{m}\alpha_i^*\left(y^{(i)}-f\left(x^{(i)}\right)-\epsilon-\xi_i^*\right)$$

根据 KKT 条件，可以得出

$$w=\sum_{i=1}^{m}(\alpha_i^*-\alpha_i)x^{(i)}$$
$$\sum_{i=1}^{m}\alpha_i=\sum_{i=1}^{m}\alpha_i^*$$
$$C=\alpha_i+\mu_i$$
$$C=\alpha_i^*+\mu_i^*$$

带回原式得

$$\sum_{i=1}^{m}(\alpha_i^*-\alpha_i)y^{(i)}-\sum_{i=1}^{m}(\alpha_i^*+\alpha_i)\epsilon-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\alpha_i^*-\alpha_i)\left(\alpha_j^*-\alpha_j\right){x^{(i)}}^T x^{(j)}$$

$$s.t.\quad\sum_{i=1}^{m}\alpha_i-\alpha_i^*=0$$
$$0\le\alpha_i,\alpha_i^*\le C$$

上式还应满足 KKT 的其他条件，即不等式约束项与拉格朗日乘数之积为0，所有的拉格朗日乘数不小于0。
另一方面，注意到ξ_i和ξ_i^*分别表示两侧的松弛度，当$y^{(i)}$落在回归线的一侧时，不可能落在另一侧，因此ξ_i和ξ_i^*至少有一个为0；同样的，对于α_i和α_i^*，当其中一个不为0时，表示对应不等式的边界条件成立，那么另一个不等式的边界条件必然不成立，因此α_i和α_i^*至少有一个为

0。综上

$$\begin{aligned}\alpha_i\left(f\left(x^{(i)}\right)-y^{(i)}-\epsilon-\xi_i\right)&=0\\ \alpha_i^*\left(y^{(i)}-f\left(x^{(i)}\right)-\epsilon-\xi_i^*\right)&=0\\ \alpha_i\alpha_i^*&=0\\ \xi_i\xi_i^*&=0\\ (C-\alpha_i)\xi_i&=0\\ (C-\alpha_i^*)\xi_i^*&=0\end{aligned}$$

可以分析出如下结果

$$\begin{aligned}\alpha_i=0,\quad \alpha_i^*=0 &\Leftrightarrow \left|f\left(x^{(i)}\right)-y^{(i)}\right|\le\epsilon\\ \alpha_i=0,\ 0<\alpha_i^*<C &\Leftrightarrow y^{(i)}-f\left(x^{(i)}\right)=\epsilon\\ \alpha_i=0,\quad \alpha_i^*=C &\Leftrightarrow y^{(i)}-f\left(x^{(i)}\right)\ge\epsilon\\ 0<\alpha_i<C,\quad \alpha_i^*=0 &\Leftrightarrow f\left(x^{(i)}\right)-y^{(i)}=\epsilon\\ \alpha_i=C,\quad \alpha_i^*=0 &\Leftrightarrow f\left(x^{(i)}\right)-y^{(i)}\ge\epsilon\end{aligned}$$

使用 SMO 可以求出α_i和α_i^*，截距b可以通过恰位于隔离带边界上的所有数据的均值求出。

补充材料——拉格朗日对偶法(Lagrange Dual)

原始问题

$$\begin{aligned}\min_{\omega} & f(\omega)\\ s.t.\quad & g_i(\omega)\le 0 \quad i=1,\dots,k\\ & h_j(\omega)=0 \quad j=1,\dots,l\end{aligned}$$

拉格朗日算子

$$\mathcal{L}(\omega,\alpha,\beta)=f(\omega)+\sum_{i=1}^{k}\alpha_i g_i(\omega)+\sum_{i=1}^{l}\beta_i h_i(\omega)$$

定义

$$\Theta_p(\omega)=\max_{\substack{\alpha,\beta\\ \alpha_i\ge 0}}\mathcal{L}(\omega,\alpha,\beta)$$

显然，如果$g_i(\omega)>0$，则可以通过取任意大的α_i使得$\Theta_P(\omega)=\infty$；同样的，如果$h_i(\omega)\neq 0$，也可以取相同符号的任意大的β_i，使得$\Theta_p(\omega)=\infty$。总之如果违背约束，$\Theta_p=\infty$，否则$\Theta_p=f$，即

$$\Theta_p(\omega)=\begin{cases}f(\omega), & \text{if coincide with } (g,h)\\ \infty, & \text{otherwise}\end{cases}$$

因此

$$\min_{\omega}\Theta_p(\omega)\Rightarrow\min_{\omega}f(\omega)$$

原始问题可以写作

$$p=\min_{\omega}\max_{\substack{\alpha,\beta\\ \alpha_i\ge 0}}\mathcal{L}(\omega,\alpha,\beta)=\min_{\omega}\Theta_P(\omega)$$

对偶问题

定义

$$\Theta_d(\alpha,\beta)=\min_{\omega}\mathcal{L}(\omega,\alpha,\beta)$$

对偶问题可以写作

$$d^* = \max_{\substack{\alpha\geq 0\\ \beta}} \min_{\omega} \mathcal{L}(\omega,\alpha,\beta) = \max_{\substack{\alpha\geq 0\\ \beta}} \Theta_d(\alpha,\beta)$$

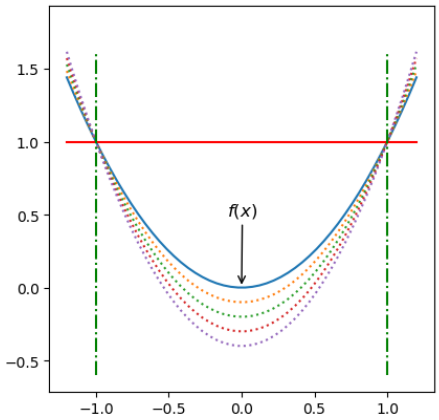
即交换了原始问题中min和max的顺序。显然，d^*是关于(α,β)逐点下确界。因为$\max\min\mathcal{L} \leq \min\max\mathcal{L}$，因此$d^* \leq p$。注意到$\sum_{i=1}^{k}\alpha_i g_i(\omega) + \sum_{i=1}^{l}\beta_i h_i(\omega) \leq 0$，当$\alpha_i = 0$或者$g_i(\omega) = 0$时，$d^* = f$。

KKT 条件(KKT Conditions)

当满足 KKT 条件时，$\inf\mathcal{L} = \sup\mathcal{L}$，即对偶问题与原问题的解间隔为0。KKT 条件为

$$\begin{cases} \nabla_\omega f(\omega) + \sum_{i=1}^{k}\alpha_i\nabla_\omega g_i(\omega) + \sum_{i=1}^{l}\beta_i\nabla_\omega h_i(\omega) = 0 & (1)\\ \alpha_i g_i(\omega) = 0 & (2)\\ g_i(\omega) \leq 0 & (3)\\ h_i(\omega) = 0 & (4)\\ \alpha_i \geq 0 & (5) \end{cases}$$

以单个约束条件$g(x) \leq 0$为例。优化目标函数为$f(x) = x^2$，约束条件为$g(x) = x^2 - 1 \leq 0$。点线表示了α不同取值时的拉格朗日算子$\mathcal{L} = f(x) + \alpha g(x)$，两条绿色的竖线表示了可行域的边界。当处于可行域边界$g(x) = 0$或者$\alpha = 0$时，$\mathcal{L}(x,\alpha,\beta) = f(x)$，否则$\mathcal{L}(x,\alpha,\beta) < f(x)$。从图中可以看出，拉格朗日对偶是通过一组小于原函数$f(x)$的函数的下确界逼近$f(x)$的下确界。这解释了条件(2) 和条件(5)。



当$f(x)$于可行域取极值时，由于x已经确定，$g(x)$是一个确定的数值，此时变成了一个等式优化问题。根据拉格朗日乘数法，当$f(x)$垂直各约束面法线方向的梯度的线性组合为0时，或者其梯度与各约束面的法线的线性组合平行时，此时x的选取已经稳定，沿着任何约束面移动最终都会被移动回这一点上，因此在此时$f(x)$取得极值。这导出了条件(1)。

从$\nabla_\omega\mathcal{L}$的形式上看，如果$\alpha_i = 0$，说明约束$g_i(x)$没有影响到$f(x)$极值点的选取；如果$\alpha_i \neq 0$，约束$g_i(x)$的梯度为原始的$\nabla_\omega f(x)$增加了一个偏移量，将x的极值点“拉进”了可行域。

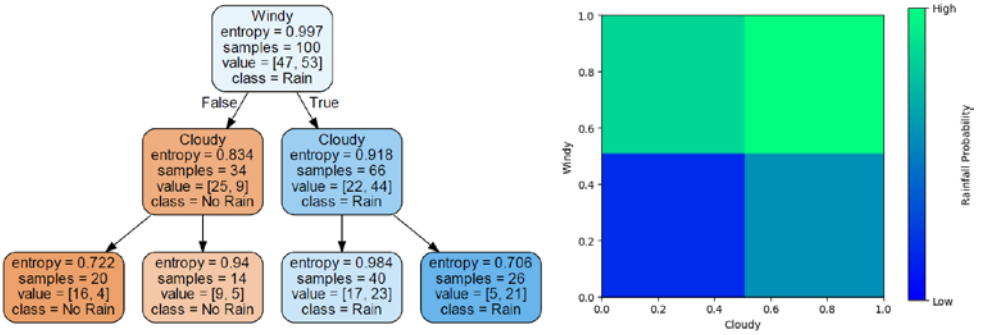
6. 决策树

6.1. 决策树分类(Decision Tree Classification)

决策树分类(Decision Tree Classification)是一种非参数化的有监督学习方法，它通过层次化的二分判断，对特征空间进行划分，以确定特征对应的类型。

为了便于讨论，设特征全部为离散型。训练样本为$D=\{(x^{(1)},y^{(1)}),\dots,(x^{(m)},y^{(m)})\}$，特征集合为$A=\{a_1,\dots,a_n\}$，类型集合为$\mathcal{Y}=\{c_1,\dots,c_{|\mathcal{Y}|}\}$。

决策树的根节点包含全部的训练样本D。内部节点表示待划分的特征，其分支为该特征的不同取值，样本根据其特征的值的不同，被划分到子节点中。叶节点表示最终的判定结果，即类型$\hat{c}$。从根节点到叶节点的路径对应着一个判定序列，即如果样本$x=(x_1,x_2,\dots,x_n)$符合从根节点到叶节点上每一个分支的判定条件，它的类型判定为$y=\hat{c}$。决策树整体的误差为所有样本真实类别与判定类别的误差之和。



构建决策树的流程如下：

初始情况下，决策树的根节点包含全部样本D。每次从特征集合A中选择一个未划分过的特征a对该节点进行划分，将该特征取值相同的样本划分到同一个子树中，重复这一过程，直到以下三种情况：

1) 新节点中所有的样本类型相同为c，即**无需划分**，此时生成叶节点$y=\hat{c}$；
2) 没有还未划分过的特征，或者所有样本其余特征的取值均相同，即**无法划分**，此时生成叶节点$y=\hat{c}$，其中$\hat{c}$为新节点中样本数最多的类型；
3) 新节点不包含任何样本，即**不能划分**，此时生成叶节点$y=\hat{c}$，其中c为父节点中样本数最多类型。

第 2)种情况使用了样本的后验分布，而第 3)种情况使用了父节点的分布作为叶节点的先验分布。

6.2. 特征选取标准(Criterion)

决策树的核心在于划分属性的选取，采用不同的选取顺序，会得到不同的决策树。构建一棵最优决策树是一个 NP 难问题，因此通常采用贪心的策略选取划分属性。

设属性a有V个取值，分别为$a^1,a^2,\dots,a^V$，数据集D中$a=a^v$的子集记为D^v。定义以属性a划分后样本集D产生的每个子树损失函数为$\mathcal{L}(D^v)$，使用该属性划分后总的损失为每个子树损失函数的期望，即

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

最优的划分属性应该能够最小化划分后的的加权损失，即

$$\hat{a}=\underset{a}{\operatorname{argmin}}\,\mathbb{E}_{a^v\sim a|D}\big(L(D^v)\big)$$

对于分类问题，设计损失函数的核心思想是，每一次划分后，子集的类型更加容易判断，即纯度(Purity)更高，或者不确定性即后验分布的熵(Entropy)更小。前者对应的选取标准为基尼指数(Gini Index)或基尼不纯度(Gini Impurity)，后者对应信息增益(Information Gain)也称互信息(Mutual Information)。

基尼指数(Gini Index)

数据集D的纯度定义为在D中随机抽取两个样本，它们类型不相同的概率，即

$$Gini(D)=\sum_{i=1}^{|\mathcal{Y}|}\sum_{j\neq i}p_ip_j=1-\sum_{i=1}^{|\mathcal{Y}|}p_i^2$$

基尼指数的最大值在$p_1=p_2=\cdots=p_{|\mathcal{Y}|}$时取得，基尼不纯度越高，表明数据集越不纯，因此也被称为基尼不纯度。当数据集中只有一个类型时，基尼指数为0。

数据D关于属性a的基尼指数定义为

$$Gini(D,a)=\sum_{v=1}^{V}\frac{|D^v|}{|D|}Gini(D^v)=\mathbb{E}_{a^v\sim a|D}\big(Gini(D^v)\big)$$

即以属性a划分时，产生子节点的基尼指数的期望。

使用基尼不纯度作为特征选取标准时，优先选择基尼不纯度较小的特征。

信息增益(Information Gain)

信息的不确定性使用香农熵(Shannon Entropy)衡量，通常直接称为熵(Entropy)。对于概率分布X，它的熵定义为

$$H(X)=-\sum_{x}p(x)\log_2 p(x)=-\mathbb{E}_{x\sim X}(\log_2 p(x))$$

因为$\lim\limits_{x\to 0^+}x\log x=0$，因此约定在计算熵时，如果$p=0$，$p(x)\log_2 p(x)=0$。如无特殊说明$\log(\cdot)$表示$\log_2(\cdot)$。

对于两个变量的联合概率分布(X,Y)，其联合熵为

$$H(X,Y)=-\mathbb{E}_{(x,y)\sim(X,Y)}(\log p(x,y))$$

根据贝叶斯公式得

$$\begin{aligned}H(X,Y)&=-\sum_{(x,y)}p(x,y)\log p(x,y)\\&=-\sum_{x}\sum_{y}p(y|x)p(x)\log p(y|x)p(x)\\&=-\sum_{x}p(x)\log p(x)\sum_{y}p(y|x)-\sum_{x}p(x)\sum_{y}p(y|x)\log p(y|x)\\&=H(X)+\sum_{x}p(x)H(Y|X=x)\end{aligned}$$

定义条件熵(Conditional Entropy)$H(Y|X)$为

$$H(Y|X)=\sum_x p(x)H(Y|X=x)$$

联合熵可以进一步写为

$$H(X,Y)=H(X)+H(Y|X)=H(Y)+H(X|Y)$$

移项可得

$$H(X)=H(X|Y)+\big(H(Y)-H(Y|X)\big)$$
$$H(Y)=H(Y|X)+\big(H(X)-H(X|Y)\big)$$

注意到

$$\begin{aligned}H(Y)-H(Y|X)&=-\sum_y p(y)\log p(y)+\sum_x p(x)\sum_y p(y|x)\log p(y|x)\\&=-\sum_y\sum_x p(x|y)p(y)\log p(y)+\sum_x\sum_y p(x)p(y|x)\log p(y|x)\\&=\sum_x\sum_y p(x,y)\log\frac{p(y|x)}{p(y)}\\&=\sum_x\sum_y p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\\&=H(X)-H(X|Y)\end{aligned}$$

定义上式为互信息(Mutual Information)，即

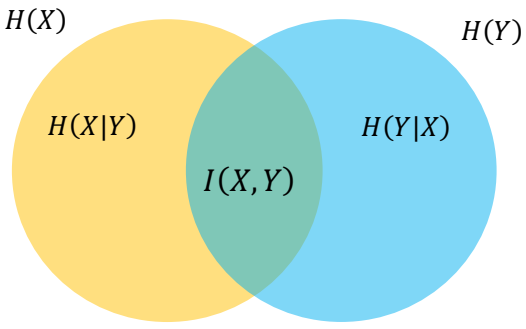
$$I(X,Y)=H(Y)-H(Y|X)=H(X)-H(Y|X)=H(X,Y)-H(X|Y)-H(Y|X)$$

根据熵的含义，$H(Y)-H(Y|X)$可以看作是已知X后对Y消除的信息不确定性，因此也称互信息为信息增益(Information Gain)。

在逻辑上，互信息中的X和Y是对等的，而信息增益中，两者具有先后顺序。例如，X对Y的信息增益$H(Y)-H(Y|X)$表示在确定了X后，通过X消除的Y的不确定性，记作

$$Gain(Y,X)=H(Y)-H(Y|X)$$

熵、条件熵和互信息的关系可以用韦恩图表示。



在构建决策树时，每次的选择要尽可能的消除样本所属类别的不确定性，因此每次选择对数据集D信息增益最大的属性$\hat{a}$，即

$$\hat{a}=\underset{a}{\operatorname{argmax}}\,Gain(D,a)=\underset{a}{\operatorname{argmax}}\,H(D)-H(D|a)$$

由于$H(D)$是一个与划分无关的常量，因此，损失函数也可以写为

$$\hat{a}=\underset{a}{\operatorname{argmin}}\,H(a|D)=\underset{a}{\operatorname{argmin}}\,\mathbb{E}_{a^v\sim a|D}\big(H(D^v)\big)$$

注意到，可取值较多的特征容易产生更大的信息增益，因此信息增益准则对可取值较多的属

性有所偏好。为了避免这一点，C4.5 决策树算法使用信息增益率(Gain Ratio)来选择最优划分属性，增益率定义为

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

其中

$$\mathrm{IV}(a)=-\sum_{v=1}^{V}\frac{|D^v|}{|D|}\log\frac{|D^v|}{|D|}=H\big((a|D)\big)$$

称作属性a的固有值(Intrinsic Value)。
注意到可取值较少的属性固有值通常较小，因此采用信息增益率作为标准时，会倾向于可取值较少的属性。因此 C4.5 算法实际上采用了一个启发式的方式选择属性：首先从候选划分属性中找出信息增益高于平均水平的属性，再从中选择增益率最高的。

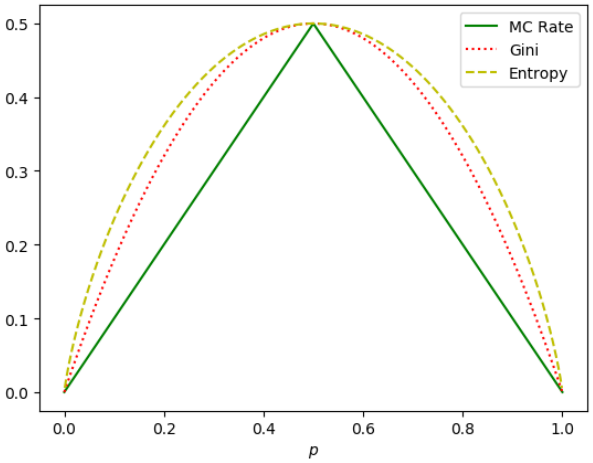
误分类率(Misclassification Rate)
另一种常用的损失函数是误分类率，定义为不同于该集合中占比最大类别的样本比例，即

$$MR(D^v)=1-\frac{1\{(x,y)\in D^v\wedge y\neq\widehat{c_{D^v}}\}}{|D^v|}$$

总的误差为

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

对比
以二分类为例，上述三种误差函数关于正例所占样本比例p的函数图像如下，其中，熵除以2放缩到[0,0.5]之间。



从上图中可以看出，如果父节点分类的纯度已经很高，即p处于两端的位置，当纯度进一步提高时，基尼指数和相对熵的要比误分类率更敏感，因此分出的节点也更纯；反之，如果p处于中间位置，误分类率的变化则更敏感。

6.3. 连续值(Continuous Value)

上述讨论的都是离散情况，对于连续值，通常采用二分的策略进行离散化。对于连续值属性

a，可以考察包含$m-1$个元素的候选划分点集合

$$T_a=\left\{\frac{a^{(i)}+a^{(i+1)}}{2}\middle|1\le i\le m-1\right\}$$

即训练样本中出现过的数值的中点。可以采用不大于中位点的最大值，这样可以保证最终的决策树中的划分点都在训练样本中出现过。
以信息增益做选取标准为例，由于有$m-1$个点可作为划分点，因此对于属性a，同时考察这$m-1$种划分方法，取其中信息增益最大者作为属性a对D的信息增益，即

$$\begin{aligned}\text{Gain}&=\max_{t\in T_a}\text{Gain}(D,a,t)\\&=\max_{t\in T_a}H(D)-H(a,t|D)\end{aligned}$$

需要注意，与离散值不同，在决策树从根节点到叶节点的路径中，同一连续值属性可以出现多次。

6.4. 缺失值(Missing Value)

有的时候，一些训练样本的属性并不完整，存在缺失。如果直接将存在缺失值的样本丢弃，则会造成样本数的锐减。C4.5 采用的方法是，对于缺失的属性值，使用未缺失该属性的样本的分布作为它的先验分布，在划分时，依概率划分入子节点中。
对于训练集D和属性a，令$\widetilde{D}$表示D中在属性a上没有缺失值的样本子集，$\widetilde{D}$在属性a上取值为v的子集记为$\widetilde{D^v}$。我们为每一个样本赋予一个权值w_x，用它来表达这个样本可信度或贡献值，在根节点处，所有样本的权值均为1。
定义ρ_a为属性a对D信息增益的贡献值，即

$$\rho_a=\frac{\sum_{x\in\widetilde{D}}w_x}{\sum_{x\in D}w_x}$$

显然，训练集在属性a上的缺失越少，属性a的信息增益就越可信。由此，将信息增益的计算调整为

$$\text{Gain}(D,a)=\rho_a\times Gain\left(\widetilde{D},a\right)=\rho_a\times\left(H\left(\widetilde{D}\right)-H\left(a\middle|\widetilde{D}\right)\right)$$

在划分时，以未缺失属性样本$\widetilde{D}$在属性a上的分布作为缺失值样本的先验分布，将缺失值样本依概率划分入子节点中。由于是依概率划分，因此样本在子节点的中的权重调整为

$$w_{x|a=a^v}=\begin{cases}w_x, & x|a=a^v\\ w_x\times\dfrac{\sum_{x\in\widetilde{D^v}}w_x}{\sum_{x\in\widetilde{D}}w_x}, & x|a=\emptyset\end{cases}$$

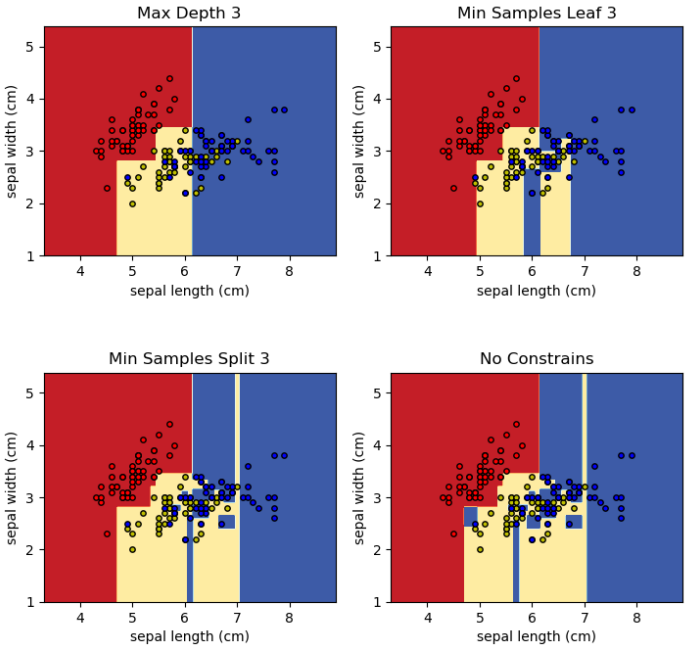
形象地看，对于缺失值的样本，在划分时，将它切成了V份，每个子节点都依概率得到了它的一部分。

6.5. 剪枝(Pruning)

决策树很容易发生过拟合的问题，特别是当样本空间较小、特征空间较大时，过拟合现象更为明显。解决决策树过拟合的方法是剪枝(Pruning)。依据剪枝的时机不同，可以分为两种，一种是在对训练集划分之前进行剪枝，称为预剪枝(Pre-pruning)，另一种是在构建完决策树后再进行剪枝，称为后剪枝(Post-pruning)。

预剪枝

在划分样本时，如果当前节点不满足划分条件，则终止划分，即提前终止(Early Termination)。一种方法是，使用一个验证集对性能进行衡量，如果划分后的决策树在验证集上的性能下降，那么禁止这次划分。这是一种典型的贪心策略，可能会导致欠拟合的问题。
另外一种方法基于节点包含的样本数，例如，限制决策树的最大深度；限制划分时样本的最小个数；限制叶节点包含样本的最小个数等。
下图为采用了三种不同的预剪枝策略与不进行剪枝的决策树决策边界的对比图。



后剪枝

在决策树构建后，从最下层的非叶子节点向根节点回溯。如果一个节点在删除后可以提升验证集的性能，则删除该节点。后剪枝策略从下向上进行剪枝，相比预剪枝可以保留更多的分支，因此欠拟合风险很小，泛化能力往往由于预剪枝决策树。但是由于决策树下层的内部节点要远多于上层节点，因此使用后剪枝的训练时间开销要比预剪枝大得多。

6.6. 决策树回归(Decision Tree Regression)

在决策树分类器中，每个叶节点保存的是该节点中所有样本的类别的分布，即将各类别样本数量的均值作为节点的输出。对于回归任务，树的叶节点保存和输出也是该节点中所有样本目标值的均值，即

$$\hat{y} = \bar{y} = \frac{1}{|D|}\sum_{D} y$$

内部节点的划分标准调整为选取划分后均方误差(Mean Squared Error)期望最小的属性，即

$$\min_{a} MSE(D,a) = \min_{a} \mathbb{E}_{a^v \sim a|D}\left(\frac{1}{|D^v|}\sum_{(x,y)\in D^v}(\hat{y}-y)^2\right)$$

也可以采用平均绝对误差(Mean Absolute Error) 的期望，即

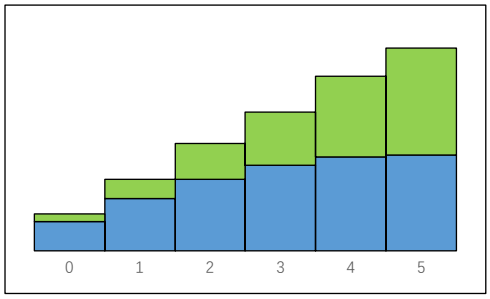
$$\min_{a} MAE(D,a) = \min_{a} \mathbb{E}_{a^v \sim a|D}\left(\frac{1}{|D^v|}\sum_{(x,y)\in D^v}|\hat{y}-y|\right)$$

6.7. 复杂度(Complexity)

在划分决策树的内部节点时，为了寻找最优属性的最优化分点，需要对每一种划分方式进行考察，计算划分后的误差。最坏情况下，对于n种属性中的每一种属性，所有的样本的属性值均不相同，这意味着有m种划分方式。如果不加以优化，每一种属性的每一个划分点，都需要对所有样本进行一次遍历，那么每次划分的时间复杂度为$O(m^2n)$。未经剪枝的决策树有$O(m)$个叶节点，也就意味着有$O(m)$个内部节点，因此构建一棵决策树的时间复杂度为$O(m^3n)$。

考虑在一次划分过程中，首先对所有样本以该属性为标准进行排序，排序后，通过两次遍历，就可以求出不同划分点产生的子树的损失值，这样，每次划分的时间复杂度降低到$O(mn\log m)$，构建整棵树的时间复杂度为$O(m^2n\log m)$。(Hunt 模型)

这个排序和统计的过程，可以看作是构建了以该属性为横轴的类别累积分布直方图，或者划分点左右两侧均值的直方图，如下图所示。通过直方图，可以快速的读出每个划分点划分后两侧的类别比例，因此可以直接计算出误差。



上述算法的排序是运行时进行的，即 on-fly 的，每个节点都需要对所有特征重新排序。进一步的改进是，在构建决策树前，分别对每一种特征进行排序，构建出n个有序链表。在每次划分时，遍历所有的样本，寻找出其中损失最小的划分，这个过程的时间复杂度是$O(mn)$。然后对于每个链表，根据划分结果进行拆分，由于拆分需要对链表进行遍历，因此时间复杂度也为$O(mn)$。总计时间复杂度为$O(m^2n)$。(arXiv:1712.06989)

Sklearn 用$O(mn\log m)$的时间复杂度实现了决策树，为理论最低下界。

参考： *https://zhuanlan.zhihu.com/p/32076887*
http://scikit-learn.org/stable/modules/tree.html

6.8. 随机森林(Random Forest)

随机森林(Random Forest)是一种基于决策树的集成学习方法，它同时训练N棵决策树，以这些决策树的投票结果作为最终的输出。

在随机森林中，每个决策树仅采用全部属性中的随机的k个属性作为决策依据，通常推荐$k=\log_2|A|$，这相当于为每一个决策树引入了属性扰动。训练每棵随机数的样本也是全体样本的随机子集，这相当于为每一个决策树引入了样本扰动。

通常，随机森林中的每一个决策树性能相对于传统决策树都有所降低，但是随着决策树个数的增多，性能会显著优于传统决策树。

7. 集成学习

7.1. 集成学习(Ensemble Learning)

集成学习(Ensemble Learning)通过构建并结合多个学习器来完成学习任务。首先构建出T个弱学习器(Weakly Learner)h_i，也称基学习器(Base Learner)，然后通过一定的策略，将它们组合成一个强学习器(Strong Learner)$H(x)$，使用强学习器的输出作为模型的最终输出。
弱学习器可以是同质的(homogeneous)，例如均采用决策树、线性回归等；也可以是异质的(heterogeneous)，例如同时采用决策树、线性回归等多种模型。
使用集成学习可以从三个方面带来好处：

1) 从统计的角度上，对于同一个训练集，假设空间中可能会有多个假设，即学习器，达到同等性能。单个学习器可能会产生错误预测，而多个学习器的组合会降低这种可能性；
2) 从计算的角度上，单个学习器在训练过程中很可能陷入局部极小值，而多个学习器的组合可以降低最终预测陷入局部极小值的风险；
3) 从表示的角度上，可能真实的假设不在单个学习器的假设空间中，此时使用单个学习器是无效的，总会造成偏差，组合多个学习器，其假设空间会有所扩大，有可能会降低偏差。

常见的集成学习方法有平均法(Averaging)、投票法(Voting)和学习法(Learning)。

平均法

强学习器的输出由一系列基学习器的线性组合表示，即

$$H(x) = \sum_{i=1}^{T} w_i h_i(x)$$

其中w_i为第i个学习器的权重，通常要求$w_i \geq 0$且$\sum_{i=1}^{T} w_i = 1$。这个模型也称作自适应基函数模型(Adaptive Basis-Function Model, ABM)，也称为加性模型(Additive Model)。
当$w_i = 1/T$时，强学习器为T个基学习器的简单平均，称为简单平均法(Simple Averaging)；当w_i不全相等时，称为加权平均法(Weighted Averaging)。
更一般的，这些权重从训练数据中学习得到。但由于训练样本并不充分并且存在噪声，学习出的权重并不一定能够对应数据的真实分布，容易造成过拟合。通常，对于性能相近的学习器，采用简单平均法，对于性能差距较大学习器采用加权平均法。然而，实际中往往并不采用这样的策略，而且得到的模型泛化性能也很优秀，例如 Boost。

投票法

投票法通常用于分类问题。强学习器的输出表示为基学习器中最多的输出，依照投票方式，可以分为绝对多数投票(Majority Voting)和相对多数投票(Plurality Voting)。
假设对于类别集合$\mathcal{Y} = \{c_1, \dots, c_{|\mathcal{Y}|}\}$，基学习器的输出为一个$|\mathcal{Y}|$维向量，它是各类别的概率分布，或者是表示预测类别的指示向量$T(i) = [0 \quad \dots \quad \overset{i-th}{1} \quad \dots \quad 0]^T$。如果基学习器的输出是前者，称为软投票(Soft Voting)，反之称为硬投票(Hard Voting)。
绝对多数法选取超过半数的类别作为最终输出，即

$$H(x)=\begin{cases} c_j, & \sum_{i=1}^{T} w_i h_i^j(x) > 0.5 \sum_{k=1}^{|y|} \sum_{i=1}^{T} w_i h_i^k(x) \\ reject, & otherwise \end{cases}$$

其中w_i为权重，且$\sum_{i=1}^{T} w_i = 1$。
相对多数法仅简单选取占多数的类别，即

$$H(x) = c_{\underset{j}{\operatorname{argmax}} \sum_{i=1}^{T} w_i h_i^j(x)}$$

注意到，绝对多数法提供了拒绝预测这个选项，这对于可靠性较高的学习任务而言是一个较好的选项。
对于软投票而言，一些基学习器甚至无法直接输出预测的概率，例如 SVM 的直接输出是样本与分割超平面的间隔，这些输出自需要经过一些规范化方法进行校准才可以使用。另一方面，不同类型的基学习器所输出的概率不能混用，因此当采用异质基学习器时，应该将概率分布转化为指示向量后再进行投票。

学习法
学习法将基学习器的输出作为另一个学习器的输入，也就是利用基学习器的输出生成了新的训练样本集合，例如 Stacking。

7.2. Boosting(Boosting)

Boosting 是一种用于拟合自适应基函数模型的贪婪算法，也是一种平均模型。它依次训练多个基学习器，每一个学习器都试图减小之前的预测误差。形式上，可以将第k步训练得到的强学习器分解为

$$H(x) = H_{k-1}(x) + w_k h_k(x)$$

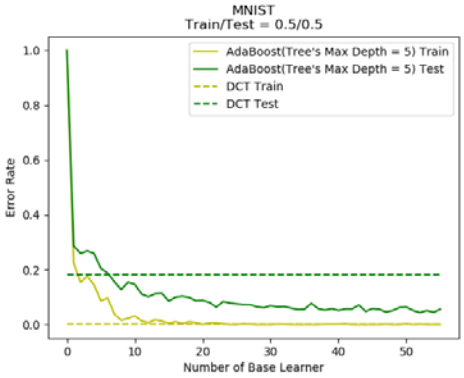
记误差函数为$\mathcal{L}(H, D)$，其中$D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$为训练样本集合，第$k$步的训练目标为

$$w_k, h_k = \underset{w,h}{\operatorname{argmin}}\, \mathcal{L}(H_k, D) = \underset{w,h}{\operatorname{argmin}}\, \mathcal{L}(H_{k-1}(x) + wh(x), D)$$

每一步训练都基于前面的训练结果，但是不会对此前训练出的基学习器进行修改，因此这种训练方法也称作前向逐阶段加性建模(Forward Stagewise Additive Modeling)。
在实践中，将新的学习器与既有结果相加时，通常乘以一个学习步长$0 < \nu \leq 1$进行部分更新，称之为收缩(Shrinkage)，这个学习步长可以取一个较小的固定值例如0.1，也可以基于一个测试集通过线性搜索寻取最优值。此时，更新公式为为

$$H(x) = H_{k-1}(x) + \nu_k w_k h_k(x)$$

大量的实践表明，Boosting 方法不仅可能有效降低训练误差，并且还能防止过拟合。



如图所示，随着基学习器的增加，训练样本的错误率和测试样本的错误率都得到有效降低。而未经剪枝的决策树尽管对可以将训练误差降至0，但是有着较大的泛化误差。

7.3. L_2Boosting(L_2Boosting)

对于回归任务，最常用的损失函数为平方误差，即

$$\mathcal{L}(H_k, D) = \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - H_k\left(x^{(i)}\right)\right)^2$$

记$r_k^{(i)} = y^{(i)} - H_{k-1}(x)$，则

$$\mathcal{L}(H_k, D) = \frac{1}{2}\sum_{i=1}^{n}\left(r_k^{(i)} - w_k h_k\left(x^{(i)}\right)\right)^2$$

不失一般性的，设$w_k = 1$，显然，最小化损失函数等同于令h_k预测第H_{k-1}的残差r_k，也就是基学习器h_k的训练样本为$\left\{\left(x^{(i)}, r_k^{(i)}\right)_{i=1,\dots,m}\right\}$。

7.4. AdaBoost(AdaBoost)

对于二分类任务，不妨设类别集合为$\mathcal{Y} = \{-1,1\}$。常用的损失函数为指数损失函数(Exponential Loss Function)

$$\mathcal{L}(H_k, D) = \sum_{j=1}^{m}\exp\left(-y^{(i)}H_k\left(x^{(i)}\right)\right)$$

可以证明，H_k的最优值为

$$H_k(x) = \frac{1}{2}\log\frac{p(y=1|x)}{p(y=-1|x)}$$

代入$H_k = H_{k-1} + w_k h_k$到损失函数中，可得

$$\mathcal{L}(H_k, D) = \sum_{i=1}^{m}\exp\left(-y^{(i)}\left(H_{k-1}\left(x^{(i)}\right) + w_k h_k\right)\right) = \sum_{i=1}^{m}\alpha_{k,i}\exp\left(-y^{(i)}w_k h_k\left(x^{(i)}\right)\right)$$

其中

$$\alpha_{k,i} = \exp\left(-y^{(i)}H_{k-1}\left(x^{(i)}\right)\right)$$

即前一步的强学习器在样本i上的损失，这可以看做是对数据集中样本施加的权重。假设基分类器的输出即类别$\{-1,1\}$，损失函数可以分解为

$$\begin{aligned}\mathcal{L}(H_k,D)&=\sum_{i=1}^{m}\alpha_{k,i}\exp\left(-w_k y^{(i)}h_k\left(x^{(i)}\right)\right)\\&=e^{-w_k}\sum_{h_k\left(x^{(i)}\right)=y^{(i)}}\alpha_{k,i}+e^{w_k}\sum_{h_k\left(x^{(i)}\right)\neq y^{(i)}}\alpha_{k,i}\\&=\left(e^{w_k}-e^{-w_k}\right)\sum_{h_k\left(x^{(i)}\right)\neq y^{(i)}}\alpha_{k,i}+e^{-w_k}\sum_{i=1}^{m}\alpha_{k,i}\end{aligned}$$

利用指示函数，

$$\sum_{h_k\left(x^{(i)}\right)\neq y^{(i)}}\alpha_{k,i}=\sum_{i=1}^{m}\alpha_{k,i}1\left\{h_k\left(x^{(i)}\right)\neq y^{(i)}\right\}$$

注意到，e^{w_k}为常数且$e^{w_k}-e^{-w_k}>0$，因此最小化$\mathcal{L}(H_k,D)$与w_k无关，这等同于最小化

$$\sum_{i=1}^{m}\alpha_{k,i}1\left\{h_k\left(x^{(i)}\right)\neq y^{(i)}\right\}$$

显然，最优的h_k应与加权的样本分类一致，即h_k的训练样本为以$\alpha_{k,i}$为权重的样本。
在训练出h_k后，代回损失函数中，可以解得权重

$$w_k=\frac{1}{2}\log\frac{1-err_k}{err_k}$$

其中

$$err_k=\frac{\sum_{i=1}^{m}\alpha_{k,i}1\left\{h_k\left(x^{(i)}\right)\neq y^{(i)}\right\}}{\sum_{i=1}^{m}\alpha_{k,i}}$$

即这一步的加权错误率。其中，函数

$$\frac{1}{2}\log\frac{p(y=1|x)}{1-p(y=1|x)}$$

称作对数优势比(Log-Odds Ratio)。
注意到，

$$\begin{aligned}\alpha_{k+1,i}&=\exp\left(-y^{(i)}H_k\left(x^{(i)}\right)\right)\\&=\exp\left(-y^{(i)}\left(H_{k-1}\left(x^{(i)}\right)\right)\exp\left(-y^{(i)}w_k h_k\left(x^{(i)}\right)\right)\\&=\alpha_{k,i}\exp\left(-y^{(i)}h_k\left(x^{(i)}\right)w_k\right)\end{aligned}$$

其中e^{w_k}在计算出err_k后是常量，可以在计算err_{k+1}时上下同时消去。
综上所述，AdaBoost 算法如下

$\alpha_{1,i}=1/m$
for $k=1$ **to** T:
　　训练基学习器$h_k(D;\alpha_k)$
　　计算基学习器误差$err_k=\frac{\sum_{i=1}^{m}\alpha_{k,i}1\{h_k(x^{(i)})\neq y^{(i)}\}}{\sum_{i=1}^{m}\alpha_{k,i}}$ **if** $err_k\geq 0.5$ 抛弃这个弱学习器
　　计算基学习器权重$w_k=\frac{1}{2}\log\frac{1-err_k}{err_k}$
　　更新样本权重$\alpha_{k+1,i}=\alpha_{k,i}\exp\left(-y^{(i)}h_k\left(x^{(i)}\right)w_k\right)$
输出$\mathrm{sgn}(\sum_{k=1}^{T}w_k h_k(x))$

AdaBoost 可以看做是将弱学习器的正确率的对数优势比，也可以看做是置信度，作为该项的权重。在训练过程中，对于预测错误的样本，样本的权重$\alpha_{k+1,i}$会增大，反之则减小，即给予预测错误的样本能更多的强调。由于样本的先验分布发生了改变，下一个学习器会更倾向于提升先验概率较大的样本的查全率(Recall)，因为它们在损失函数中的占比更大。通过这样一系列的对之前错误预测的更正，模型最终达到较小的训练误差。

上述 AdaBoost 模型中，基分类器的输出是离散值$\{-1,1\}$，因此也称作离散 AdaBoost(Discrete AdaBoost)，对应的，基学习器输出实数的 AdaBoost 模型称作实值 AdaBoost(Real AdaBoost)。

实值 AdaBoost 的损失函数无法做与离散 AdaBoost 相同的分解，因此权重w_k没有闭式解。

在实值 AdaBoost 中，w_k始终为1，其输出为$p(y=1|x)$的对数优势比，因此可以用来作为类别概率使用。

对任意特征变量x的损失函数$\mathcal{L}(H_k,x,y)$，它等于在后验分布$y|x$下y不同取值下损失函数值的数学期望，即

$$\begin{aligned}\mathcal{L}(H_k,x,y)&=E_{y|x}\left(e^{-yH_{k-1}(x)}e^{-yh_k(x)}\right)\\&=e^{-h_k(x)}E_{y|x}\left(e^{-yH_{k-1}(x)}1\{y=1\}\right)+e^{h_k(x)}E_{y|x}\left(e^{-yH_{k-1}(x)}1\{y=-1\}\right)\\&=e^{-h_k(x)}e^{-H_{k-1}(x)}p(y=1|x)+e^{h_k(x)}e^{H_{k-1}(x)}p(y=-1|x)\end{aligned}$$

等式右侧除以常数$E_{y|x}\left(e^{-yH_{k-1}(x)}\right)$，并令$\mathcal{L}$对$h_k$的导数为$0$得

$$h_k(x)=\frac{1}{2}\log\frac{E_{y|x}\left(e^{-yH_{k-1}(x)}1\{y=1\}\right)\Big/E_{y|x}\left(e^{-yH_{k-1}(x)}\right)}{E_{y|x}\left(e^{-yH_{k-1}(x)}1\{y=-1\}\right)\Big/E_{y|x}\left(e^{-yH_{k-1}(x)}\middle|x\right)}\triangleq\frac{1}{2}\log\frac{p_\alpha(y=1|x)}{p_\alpha(y=-1|x)}$$

即通过加权的样本训练基学习器，以预测出的$y(x)$分别为1和-1的概率，然后计算对数优势比作为$h_k(x)$。

通过上式进行更新，理论上可以一次性直接达到最小值点。但是实践中，由于$p(y=1|x)$是由模型拟合出的近似值，例如在决策树中，$p(y=1|x)$为叶节点中$y=1$的样本个数权重和与总权重和之比，因此常常需要多次迭代才能收敛。

实值 AdaBoost 的基学习器需要计算对数优势比，然而当$p_\alpha(y=1|x)\ll p_\alpha(y=-1|x)$时，对数值的绝对值会非常大，因此并不稳定。一种使用牛顿法求解的改进版本称作 Gentle AdaBoost。

将$H_k=H_{k-1}+h_k$带入损失函数中，得到$\mathcal{L}(H_k,x,y)=\mathcal{L}(H_{k-1}+h_k,x,y)$，函数$h_k$是函数$H_{k-1}$的变化量，可以使用牛顿法求解其最优值。求在$h_k=0$处对$h_k$的偏导数得

$$\frac{\partial\mathcal{L}(H_{k-1}+h_k,x,y)}{\partial h_k}|_{h_k=0}=-E_{y|x}\left(ye^{-yH_{k-1}(x)}\right)$$

二阶偏导数为

$$\frac{\partial^2\mathcal{L}(H_{k-1}+h_k,x,y)}{\partial h_k^2}|_{h_k=0}=E_{y|x}\left(e^{-yH_{k-1}(x)}\right)$$

牛顿法更新步骤为

$$H_k=H_{k-1}+\frac{E_{y|x}\left(ye^{-yH_{k-1}(x)}\right)}{E_{y|x}\left(e^{-yH_{k-1}(x)}\right)}\triangleq H_{k-1}+\mathbb{E}_\alpha(y)$$

加权期望$\mathbb{E}_\alpha$可以通过使用加权的数据训练得到，例如在决策树中，$\mathbb{E}_\alpha$为叶节点中元素的加权和。

7.5. LogitBoost(LogitBoost)

使用指数损失函数的 AdaBoost 的显著缺点是$H(x)$的输出不能作为类别的概率使用。如果使

用二项概率(Binomial Probability)作为将x预测为正类的概率

$$p'(x)=\frac{e^{H(x)}}{e^{-H(x)}+e^{H(x)}}=\frac{1}{1+e^{-2H(x)}}=\sigma\big(2H(x)\big)$$

记$y^*=(y+1)/2$，最大似然函数为

$$\mathrm{MLL}(H,D)=\prod_{i=1}^{m}p'(x)^{y^{*(i)}}\big(1-p'(x)\big)^{\left(1-y^{*(i)}\right)}$$

取对数得

$$\begin{aligned}\ell(H,D)&=\sum_{i=1}^{m}y^{*(i)}\log p'(x)+\big(1-y^{*(i)}\big)\log\big(1-p'(x)\big)\\&=-\sum_{i=1}^{m}\log\left(1+e^{-2y^{(i)}H\left(x^{(i)}\right)}\right)\end{aligned}$$

损失函数可以写作

$$\mathcal{L}(H,D)=\sum_{i=1}^{m}\log\left(1+\exp\left(-2y^{(i)}H\big(x^{(i)}\big)\right)\right)$$

称作为负二元对数似然函数(Negative Binomial Log Likelihood)。由于无法直接得到显式解，因此采用牛顿法进行迭代求解。

将任意特征x的损失函数写作关于样本实际的的后验分布$y|x$期望的形式，即

$$\begin{aligned}\mathcal{L}(H_k,x,y)&=E_{y|x}\big(\log\big(1+e^{-2yH_k(x)}\big)\big)\\&=-E_{y|x}\left(y^*\log\frac{e^{H_k(x)}}{e^{-H_k(x)}+e^{H_k(x)}}+(1-y^*)\log\frac{e^{-H_k(x)}}{e^{-H_k(x)}+e^{H_k(x)}}\right)\\&=-E_{y|x}\left((2y^*-1)H_k(x)+\log\frac{e^{H_k(x)}}{(e^{-H_k(x)+H_k(x)})e^{H_k(x)}}\right)\\&=-E_{y|x}(2y^*H_k-\log\big(1+e^{2H_k(x)}\big)\end{aligned}$$

带入$H_k=H_{k-1}+h_k$得

$$\mathcal{L}(H_{k-1}+h_k,x,y)=-E_{y|x}\big(2y^*\big(H_{k-1}(x)+h_k(x)\big)-\log\big(1+e^{2\left(H_{k-1}(x)+h_k(x)\right)}\big)\big)$$

为了使用牛顿法对H_{k-1}进行更新，分别求$\mathcal{L}(H_{k-1}+h_k,x,y)$在$h_k=0$处的一阶、二阶偏导数，可得

$$\begin{aligned}\frac{\partial\mathcal{L}(H_{k-1}+h_k,x,y)}{\partial h_k}|_{h_k=0}&=-2E_{y|x}\big(y^*-p'_{k-1}(x)\big)\\\frac{\partial^2\mathcal{L}(H_{k-1}+h_k,x,y)}{\partial h_k^2}|_{h_k=0}&=4E_{y|x}\big(p'_{k-1}(x)(1-p'_{k-1}(x)\big)\end{aligned}$$

因此更新步骤为

$$
\begin{aligned}
H_k(x) &= H_{k-1}(x) + \frac{1}{2}\frac{E_{y|x}\left(y^* - p'_{k-1}(x)\right)}{E_{y|x}\left(p'_{k-1}(x)\left(1-p'_{k-1}(x)\right)\right)} \\
&= H_{k-1}(x) + \frac{1}{2}\frac{E_{y|x}\left(\frac{y^* - p'_{k-1}(x)}{p'_{k-1}(x)(1-p'_{k-1}(x)}p'_{k-1}(x)\left(1-p'_{k-1}(x)\right)\right)}{E_{y|x}\left(p'_{k-1}(x)\left(1-p'_{k-1}(x)\right)\right)} \\
&= H_{k-1}(x) + \frac{1}{2}E_{\alpha}\left(\frac{y^* - p'_{k-1}(x)}{p'_{k-1}(x)\left(1-p'_{k-1}(x)\right)}\right)
\end{aligned}
$$

其中$\alpha_{k,i} = p'\left(x^{(i)}\right)\left(1-p'\left(x^{(i)}\right)\right)$。

从上式中可以看出，基学习器的权重恒定为$1/2$。

记

$$
z_{k,i} = \frac{y^{*(i)} - p'_{k-1}\left(x^{(i)}\right)}{p'_{k-1}\left(x^{(i)}\right)\left(1-p'_{k-1}\left(x^{(i)}\right)\right)}
$$

在训练时，如果直接令$h_k = z_k$则会导致过拟合。因此改为求解h_k和z_k的最小二乘误差。

综上所述，LogitBoost 算法如下

$\alpha_{1,i} = 1/m$

$\pi_{1,i} = 1/2$ // $\pi_i = p\left(y=1\middle|x^{(i)}\right)$

for $k = 1$ **to** T:

计算工作响应$z_i = \frac{y^{*(i)} - \pi_{k,i}}{\pi_{k,i}\left(1-\pi_{k,i}\right)}$

计算样本权重$\alpha_{k,i} = \pi_{k,i}\left(1-\pi_{k,i}\right)$

训练基学习器$h_k(\alpha_k) = \underset{h}{\operatorname{argmin}} \sum_{i=1}^{m} \alpha_{k,i}\left(z_i - h\left(x^{(i)}\right)\right)^2$

更新强学习器$H_k(x) = H_{k-1}(x) + \frac{1}{2}h_k(x)$

计算$\pi_{k,i} = \sigma\left(2H_k\left(x^{(i)}\right)\right)$

输出$\operatorname{sgn}\left(\sum_{k=1}^{T} h_k(x)\right)$

参考：Friedman J, Hastie T, Tibshirani R. Special Invited Paper. Additive Logistic Regression: A Statistical View of Boosting[J]. Annals of Statistics, 2000, 28(2):337-374.

Robert C. Machine Learning, a Probabilistic Perspective[M]. MIT Press, 2012.

7.6. 梯度 Boosting(Gradient Boosting)

为每一种损失函数都求解一种 Boost 算法是不太现实的，而且上述算法在学习过程中需要对样本进行加权采样，在实现时相对复杂。梯度 Boosting(Gradient Boosting)为任意可微的损失函数都提供了统一的框架。考虑到函数沿着梯度方向下降最快，在训练新的基学习器时，总是令强学习器沿着其梯度下降方向移动，即使用梯度下降法更新强学习器$H(x)$。

取$g_k(x)$为损失函数逐样本的对$H_{k-1}\left(x^{(i)}\right)$梯度，即

$$
g_k\left(x^{(i)}\right) = \frac{\partial \mathcal{L}\left(H_{k-1}\left(x^{(i)}\right), x^{(i)}, y^{(i)}\right)}{\partial H_{k-1}\left(x^{(i)}\right)}
$$

更新公式为

$$H_k(x) = H_{k-1}(x) - v_k g_k(x)$$

由于直接使用逐样本的梯度会导致过拟合，因此使用一个弱学习器逼近负梯度，即

$$h_k\left(x^{(i)}\right) = \min_h \sum_{i=1}^{m} \left(-g_k\left(x^{(i)}\right) - h\left(x^{(i)}\right)\right)^2$$

也就是说，将$H_{k-1}(x)$逐样本的负梯度作为下一个弱学习器的训练数据的目标值。

例如：

1) 当损失函数为均方误差时

$$g_k\left(x^{(i)}\right) = 2\left(H_{k-1}\left(x^{(i)}\right) - y^{(i)}\right) = 2r_k^{(i)}$$

这等同于L_2Boosting。

2) 当损失函数为绝对值误差时

$$g_k\left(x^{(i)}\right) = \nabla_{H_{k-1}}\left(\left|y^{(i)} - H_{k-1}\left(x^{(i)}\right)\right|\right) = \operatorname{sgn}\left(y^{(i)} - H_{k-1}\left(x^{(i)}\right)\right)$$

3) 当损失函数为指数损失函数时

$$g_k\left(x^{(i)}\right) = -y^{(i)} \exp\left(-y^{(i)} H_{k-1}\left(x^{(i)}\right)\right)$$

应该注意到尽管与 AdaBoost 很相似，但是 AdaBoost 的样本的目标值本身没有改变，只是修改了分布，而梯度 Boost 修改了目标值。

4) 当损失函数为对数损失函数时

$$g_k\left(x^{(i)}\right) = -2y^{(i)} \frac{\exp\left(-2y^{(i)} H_k\left(x^{(i)}\right)\right)}{1 + \exp\left(-2y^{(i)} H_k\left(x^{(i)}\right)\right)}$$

这个模型称作 BinomialBoost。

使用梯度 Boost 的好处是，不必在每次训练新的弱学习器时调整样本的权重，只需要将上一步得到的强学习器的负梯度作为训练样本的目标值即可。

7.7. GBDT(Gradient Boosting Decision Trees)

GBDT(Gradient Boosting Decision Trees, GBDT)使用决策树作为模型的基学习器，它不仅使用了梯度 Boosting 基本的框架，还对叶节点的输出进行调整，使损失函数进一步减小。

对于决策树h_k，假设它共有J_k个叶节点，记它的第j个叶节点为$R_{k,j}$，对应的叶节点输出为$v_{k,j}$，强学习器H_k可以写作

$$H_k(x) = H_{k-1}(x) + \sum_{j=1}^{J_k} v_{k,j} 1\{x \in R_{k,j}\}$$

这可以看作是将h_k分拆成了J_k个函数，分别进行优化，因此

$$\{v_{k,j}\} = \underset{\{v_j\}}{\operatorname{argmin}} \mathcal{L}\left(H_{k-1}(x) + \sum_{j=1}^{J_k} v_j 1\{x \in R_{k,j}\}, D\right)$$

因为决策树的叶节点是不重叠的，即$v_{k,j}$是彼此独立的，因此可以分别进行优化

$$v_{k,j} = \underset{v}{\operatorname{argmin}} \mathcal{L}(H_{k-1}(x) + v, y) \qquad s.t. \quad x \in R_{k,j}$$

应该注意到，在GBDT算法中，训练新的决策树时，使用的训练数据仍为梯度信息$(x, g_k(x))$，也就是说，梯度信息决定了树的结构，而v在确定了树的结构之后才能得到。

最小二乘误差

$$\mathcal{L}(v, X_R, Y_R) = \sum_{x \in R} \Big(\big(y - H_{k-1}(x)\big) - v \Big)^2$$

输出v的最优值为

$$v^* = \frac{1}{|R|} \sum_{x \in R} \big(y - H_{k-1}(x)\big)$$

即上一步残差的均值。

绝对值误差

$$\mathcal{L}(v, X_R, Y_R) = \sum_{x \in R} \big| \big(y - H_{k-1}(x)\big) - v \big|$$

对v的导数为残差与v相减的符号之和
输出v的最优值为

$$v^* = \frac{1}{|R|} \sum_{x \in R} \operatorname{sgn} \Big(\big(y - H_{k-1}(x)\big) - v \Big)$$

即上一步残差的均值。

对数误差

$$\mathcal{L}(v, X_R, Y_R) = \sum_{x \in R} \log\big(1 + e^{-2y(H_{k-1}(x)+v)}\big)$$

记伪响应(Pseudo-response)为

$$\tilde{y} = \frac{2y}{1 + e^{2yH_{k-1}(x)}}$$

使用单步伪牛顿法得到的最优γ为

$$v = \sum_{x \in R} \frac{\tilde{y}}{\sum_{i \in R} |\tilde{y}|(2 - |\tilde{y}|)}$$

参考： *https://mlnote.com/2016/10/02/gradient-boosting-decision-tree-2*
Friedman J H. Greedy Function Approximation: A Gradient Boosting Machine[J]. Annals of Statistics, 2000, 29(5):1189–1232.

7.8. XGBoost(Extreme Gradient Boosting)

XGBoost(Extreme Gradient Boosting)是对GBDT的进一步改进，它除了在算法上的不同外，更重要的是它深度考虑了系统优化和机器学习的原理。
将总的损失函数重新定义为

$$\mathcal{L}(H, D) = \sum_{i=1}^{m} \mathcal{L}\big(H(x^{(i)}), y^{(i)}\big) + \sum_{i=1}^{T} \Omega(h_i)$$

其中Ω为正则项。带入$H_k = H_{k-1} + h_k$得到$\mathcal{L}(H_{k-1} + h_k, X, Y)$。因为很多损失函数都难以求出显式的最小解，因此用$\mathcal{L}$对$h_k$的泰勒展开作为$\mathcal{L}$的近似

$$\mathcal{L}(H_{k-1}+h_k,D)=\sum_{i=1}^{m}\left(\mathcal{L}\left(H_{k-1},x^{(i)},y^{(i)}\right)+g^{(1)}h_k+\frac{1}{2}g^{(2)}h_k^2\right)+\sum_{i=1}^{T}\Omega(h_i)$$

其中

$$g_k^{(1)}=\frac{\partial\mathcal{L}(H_{k-1}+h_k,x,y)}{\partial h_k}|_{h_k=0}=\frac{\partial\mathcal{L}(H_{k-1},x,y)}{\partial H_{k-1}}$$
$$g_k^{(2)}=\frac{\partial^2\mathcal{L}(H_{k-1}+h_k,x,y)}{\partial h_k^2}|_{h_k=0}=\frac{\partial^2\mathcal{L}(H_{k-1},x,y)}{\partial H_{k-1}^2}$$

因为 Boost 是前馈学习，在训练h_k时不需要考虑此前的基学习器，因此h_k的优化目标为

$$h_k=\underset{h}{\operatorname{argmin}}\sum_{i=1}^{m}\left(g^{(1)}h+\frac{1}{2}g^{(2)}h^2\right)+\Omega(h)$$

XGBoost 使用决策树做基学习器，记决策树h_k共有J_k个叶节点$\{R_{k,1},\dots,R_{k,J_k}\}$，每个叶节点$R_{k,j}$对应的输出为$v_{k,j}$，决策树可以写为

$$h_k(x)=\sum_{j=1}^{J_k}v_{k,j}\,1\{x\in R_{k,j}\}$$

在 XGBoost 中，决策树h的正则项定义为

$$\Omega(h)=\gamma J+\frac{1}{2}\lambda\sum_{j=1}^{J}v_j^2$$

其中γ和λ都是常数。从正则项中可以看出，XGBoost 通过两个方面防止过拟合。其一是减少叶节点的个数，避免过度分割；其二是减少基学习器对总体的贡献，这避决策树对较小的残差进行拟合。

把h_k的优化目标写成以叶节点为单位的求和形式

$$\sum_{j=1}^{J}\left(\left(\sum_{x\in R_j}g_k^{(1)}\right)v_{k,j}+\frac{1}{2}\left(\sum_{x\in R_j}g_k^{(2)}\right)v_{k,j}^2\right)+\gamma J+\frac{1}{2}\lambda\sum_{j=1}^{J}v_{k,j}^2$$
$$=\sum_{j=1}^{J}\left(\left(\sum_{x\in R_j}g_k^{(1)}\right)v_{k,j}+\frac{1}{2}\left(\sum_{x\in R_j}g_k^{(2)}+\lambda\right)v_{k,j}^2\right)+\gamma J$$

记$G_{k,j}^{(1)}=\left(\sum_{x\in R_j}g^{(1)}\right)$，$G_{k,j}^{(2)}=\left(\sum_{x\in R_j}g^{(2)}\right)$。

令其对h_k的偏导数等于0得到叶节点数出的最优值为

$$v_{k,j}^*=-\frac{G_{k,j}^{(1)}}{G_{k,j}^{(2)}+\lambda}$$

带入可得对单个节点，其损失函数的最小值

$$\mathcal{L}^*=-\frac{1}{2}\frac{{G_{k,j}^{(1)}}^2}{G_{k,j}^{(2)}+\lambda}+\gamma$$

与 GDBT 不同，在单个决策树中，损失函数不再是对H_{k-1}梯度的最小二乘误差，而是修改为这里的$\mathcal{L}^*$，因此，分裂的增益为

$$\text{Gain} = \mathcal{L} - \mathcal{L}_L - \mathcal{L}_R = \frac{1}{2}\left(\frac{{G_L^{(1)}}^2}{G_L^{(2)}+\lambda} + \frac{{G_R^{(1)}}^2}{G_R^{(2)}+\lambda} - \frac{\left(G_L^{(1)}+G_R^{(1)}\right)^2}{\left(G_L^{(2)}+G_R^{(2)}\right)+\lambda}\right) - \gamma$$

其中$G_L^{(l)}$、$G_R^{(l)}$分别为分裂后的左右子树的l阶偏导数。这里就体现出了正则项对叶节点个数的控制，如果任何分裂方式减少的损失都少于γ，分裂不会发生，此时决策树被剪枝，降低了决策树过拟合的可能。

参考： *https://xgboost.readthedocs.io/en/latest/tutorials/model.html*

7.9. Bagging(Bagging)

Bagging(Bootstrap Aggregating)与 Boosting 类的算法不同，它是一种并行式的集成学习算法。

在 Bagging 算法中，每一个基学习器的训练样本都通过*自助采样法*(Bootstrap Sampling)构建。给定具有m个样本的样本集合D，每次从中有放回地抽取一个样本加入到训练集D'中，重复m次，形成最终的训练集。一个样本在选取m次后依旧未被选取到的概率是$(1-1/m)^m$，取$m \to \infty$的极限，可得

$$\lim_{m\to\infty} p(x \notin D') = \lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m = \frac{1}{e} \approx 0.3679$$

这意味着，使用自助采样法构建训练集，最后会有大约$1/3$的样本没有在训练集中用到，这些样本将被作为测试集使用，称作*包外估计*(Out-of-bag Estimate)。通常，自助采样法可能会由于采样导致样本的分布发生变化，为模型引入方差。但是在 Bagging 算法中，T个基学习器的训练样本的采样彼此独立，可以允许有交集，因此方差被降低。

对于特征x，它采用 Bagging 模型的预测结果为所有基学习器预测结果中出现次数最多类别，即采用投票法对基学习器的结果进行综合，即

$$y^* = \underset{y\in\mathcal{Y}}{\text{argmax}} \sum_{i=1}^{T} 1\{h_i(x) = y\}$$

记D_t为第t个基学习器的训练样本集。为了考察 Bagging 的泛化性能，对于每一个样本，仅考虑那些未用它做训练样本的基学习器的预测结果。用y^{oob}表示对样本x的包外估计，有

$$y^{oob}(x) = \underset{y\in\mathcal{Y}}{\text{argmax}} \sum_{i=1}^{T} 1\{h_i(x) = y \wedge x \notin D_t\}$$

Bagging 的泛化误差定义为

$$\epsilon^{oob} = \frac{1}{|D|}\sum_{(x,y)\in D} 1\{y^{oob}(x) \neq y\}$$

除了用于估计泛化误差外，包外样本还可以用于辅助基学习器的训练，例如决策树剪枝、神经网络的提前终止等。

7.10. Stacking(Stacking)

Stacking 是学习法的代表，它首选训练出一组*初级学习器*，然后用初级学习器的输出构建出新的训练集，用于训练*次级学习器*h'，也就是将模型级联在一起。

在训练初级学习器时，样本的形式为$(x,y)=(x_1,x_2,\dots,x_n,y)$。训练次级学习器时，样本的形式为$(z,y)=(h_1(x),h_2(x),\dots,h_T(x),y)$。最终的强学习器可以表达为

$$H(x)=h'\big(h_1(x),h_2(x),\dots,h_T(x)\big)$$

这里的基学习器h_i可以是同质的，也可以是异质的。从特征工程的角度看，基学习器是对原始特征的映射，它们将原始的特征向量映射到另一个与目标值紧密联系的新的空间中。

在训练次级学习器h'时，其训练集由初级学习器产生。如果直接使用训练初级学习器的样本集合生成训练次基学习器的样本集合，很容易导致过拟合。一般采用交叉检验或者留一法的方式，选取训练初级学习器未使用的样本生成次级学习器的训练集。

次级学习器的输入表示和次级学习器的算法都对 Stacking 的泛化性能有较大影响。研究表明，将初级学习器输出的类别概率作为次级学习器的输入、用多响应线性回归(Multi-response Linear Regression)模型作为次级学习器的算法效果较好。