Kubi Code'Blog

The palest ink is better than the best memory.

机器学习常见算法个人总结（面试用）

By Kubi Code

发表于 2015-08-16

> ~~这个总结最初是写在本地的MacDown中的，他的markdown语法解析与hexo的略微有点差异，懒得改了-_-，忍着点看..~~
> 对排版看不下去了，将本文的公式使用`Latex`重写了^_^ .on 2016-04-15

朴素贝叶斯

> 参考[1]

事件`A`和`B`同时发生的概率为`在A发生的情况下发生B`或者`在B发生的情况下发生A`

$$P(A \cap B) = P(A) * P(B|A) = P(B) * P(A|B)$$

所以有：

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

对于给出的待分类项，求解在此项出现的条件下各个目标类别出现的概率，哪个最大，就认为此待分类项属于哪个类别

工作原理

1. 假设现在有样本$x = (a_1, a_2, a_3, \dots a_n)$这个待分类项(并认为$x$里面的特征独立)
2. 再假设现在有分类目标$Y = \{y_1, y_2, y_3, y_4 . . y_n\}$
3. 那么$max(P(y_1|x), P(y_2|x), P(y_3|x) . . P(y_n|x))$就是最终的分类类别
4. 而$P(y_i|x) = \frac{p(x|y_i) * P(yi)}{P(x)}$
5. 因为x对于每个分类目标来说都一样，所以就是求$max(P(x|y_i) * p(y_i))$
6. $P(x|y_i) * p(y_i) = p(y_i) * \prod_i (P(a_i|y_i))$
7. 而具体的$p(a_i|y_i)$和$p(y_i)$都是能从训练样本中统计出来

   $p(a_i|y_i)$表示该类别下该特征出现的概率

   $p(y_i)$表示全部类别中这个这个类别出现的概率
8. 好的，就是这么工作的^_^

工作流程

1. 准备阶段

   确定特征属性，并对每个特征属性进行适当划分，然后由人工对一部分待分类项进行分类，形成训练样本。
2. 训练阶段

   计算每个类别在训练样本中的出现频率及每个特征属性划分对每个类别的条件概率估计
3. 应用阶段

   使用分类器进行分类，输入是分类器和待分类样本，输出是样本属于的分类类别

属性特征

1. 特征为离散值时直接统计即可（表示统计概率）
2. 特征为连续值的时候假定特征符合高斯分布:$g(x, n, u)$

   那么$p(a_k|y_i) = g(x_k, n_i, u_i)$

Laplace校准(拉普拉斯校验)

当某个类别下某个特征划分没有出现时，会有$P(a|y) = 0$，就是导致分类器质量降低，所以此时引入Laplace校验，就是对没类别下所有划分的计数加1。

遇到特征之间不独立问题

参考改进的贝叶斯网络，使用`DAG`来进行概率图的描述

优缺点

朴素贝叶斯的优点：

1. 对小规模的数据表现很好，适合多分类任务，适合增量式训练。

   缺点：

2. 对输入数据的表达形式很敏感（离散、连续，值极大极小之类的）。

逻辑回归和线性回归

> 参考[2,3,4]

`LR`回归是一个线性的二分类模型，主要是`计算在某个样本特征下事件发生的概率`，比如根据用户的浏览购买情况作为特征来计算它是否会购买这个商品，抑或是它是否会点击这个商品。然后`LR`的最终值是根据一个线性和函数再通过一个`sigmoid`函数来求得，这个线性和函数权重与特征值的累加以及加上偏置求出来的，所以在训练`LR`时也就是在训练线性和函数的各个权重值`w`。

$$h_w(x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

关于这个权重值`w`一般使用最大似然法来估计,假设现在有样本$\{x_i, y_i\}$,其中x_i表示样本的特征，$y_i \in \{0, 1\}$表示样本的分类真实值，$y_i = 1$的概率是p_i,则$y_i = 0$的概率是$1 - p_i$，那么观测概率为:

$$p(y_i) = p_i^{y_i} * (1 - p_i)^{1 - y_i}$$

则最大似然函数为:

$$\prod \left(h_w(x_i)^{y_i} * (1 - h_w(x_i))^{1 - y_i}\right)$$

对这个似然函数取对数之后就会得到的表达式

$$L(w) = \sum_i (y_i * log h_w(x_i) + (1 - y_i) * log(1 - h_w(x_i))) = \sum_i \left(y_i * (w^T$$

估计这个$L(w)$的极大值就可以得到w的估计值。

> 实际操作中一般会加个负号 改为求最小

所以求解问题就变成了这个最大似然函数的最优化问题，这里通常会采样随机梯度下降法和拟牛顿迭代法来进行优化

梯度下降法

`LR` 的损失函数为:

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}(y_i * log(h_w(x_i)) + (1-y_i) * log(1-h_w(x_i)))$$

这样就变成了求$min(J(w))$
其更新w的过程为

$$w := w - \alpha * \nabla J(w)$$

$$w := w - \alpha * \frac{1}{N} * \sum_{i=1}^{N}((h_w(x_i) - y_i) * x_i)$$

其中α为步长,直到$J(w)$不能再小时停止

梯度下降法的最大问题就是会陷入局部最优，并且每次在对当前样本计算 `cost` 的时候都需要去遍历全部样本才能得到 `cost` 值，这样计算速度就会慢很多（虽然在计算的时候可以转为矩阵乘法去更新整个 `w` 值）
所以现在好多框架（ `mahout` ）中一般使用随机梯度下降法，它在计算cost的时候只计算当前的代价，最终 `cost` 是在全部样本迭代一遍之求和得出，还有他在更新当前的参数w的时候并不是依次遍历样本，而是从所有的样本中随机选择一条进行计算，它方法收敛速度快（一般是使用最大迭代次数），并且还可以避免局部最优，并且还很容易并行（使用参数服务器的方式进行并行）

$$w := w - \alpha * \left((h_w(x_j) - y_j) * x_i\right); j \in 1\ Nandrandomly$$

这里 `SGD` 可以改进的地方就是使用动态的步长

$$\alpha = 0.04 * (1.0 + n + i) + r$$

其他优化方法

- 拟牛顿法（记得是需要使用Hessian矩阵和cholesky分解）
- BFGS
- L-BFGS

> 优缺点：无需选择学习率α，更快，但是更复杂

关于LR的过拟合问题：

> 如果我们有很多的特性，在训练集上拟合得很好，但是在预测集上却达不到
> 效果

1. 减少feature个数（人工定义留多少个feature、算法选取这些feature）
2. 正则化（为了方便求解，`L2`使用较多）

   添加正则化之后的损失函数为:

   $J(w) = -\frac{1}{N}\sum_{i=1}^{N}(y_i * log(h_w(x_i)) + (1-y_i) * log(1-h_w(x_i))) + \lambda \|w\|_2$

   同时w的更新变为$w := w - \alpha * \left(h_w(x_j) - y_j\right) * x_i\Big) - 2\alpha * w_j$

   注意：这里的w_0不受正则化影响

> `p`范数的求解:
>
> $$\|X\|_p = \begin{cases} Count(x_i \neq 0) & if \quad p = 0 \\ (\sum_i^n |x_i|^p)^{\frac{1}{p}} & if \quad p \neq 0 \end{cases}$$

关于LR的多分类：softmax

假设离散型随机变量Y的取值集合是{1,2,..,k},则多分类的LR为

$$P(Y = a|x) = \frac{exp(w_a * x)}{(\sum_{i=1}^{k}(wi * x))}; 1 < a < k$$

这里会输出当前样本下属于哪一类的概率，并且满足全部概率加起来=1

关于softmax和k个LR的选择

如果类别之间是否互斥（比如音乐只能属于古典音乐、乡村音乐、摇滚月的一种）
就用softmax
否则类别之前有联系（比如一首歌曲可能有影视原声，也可能包含人声，或者是舞曲），这个时候使用k个LR更为合适

优缺点：
Logistic回归优点：

1. 实现简单；
2. 分类时计算量非常小，速度很快，存储资源低；

缺点：

1. 容易欠拟合，一般准确度不太高
2. 只能处理两分类问题（在此基础上衍生出来的softmax可以用于多分类），且必须线性可分；

> ps 另外 `LR` 还可以参考这篇以及多分类可以看这篇,softmax可以看这篇

KNN算法

给一个训练数据集和一个新的实例，在训练数据集中找出与这个新实例最近的k个训练实例，然后统计最近的k个训练实例中所属类别计数最多的那个类，就是新实例的类

三要素：

1. k值的选择
2. 距离的度量（常见的距离度量有欧式距离，马氏距离等）
3. 分类决策规则 （多数表决规则）

k值的选择

1. k值越小表明模型越复杂，更加容易过拟合
2. 但是k值越大，模型越简单，如果k=N的时候就表明无论什么点都是训练集中类别最多的那个类

> 所以一般k会取一个较小的值，然后用过交叉验证来确定
> 这里所谓的交叉验证就是将样本划分一部分出来为预测样本，比如95%训练，5%预测，然后k分别取1，2，3，4，5之类的，进行预测，计算最后的分类误差，选择误差最小的k

KNN的回归

在找到最近的k个实例之后，可以计算这k个实例的平均值作为预测值。或者还可以给这k个实例添加一个权重再求平均值，这个权重与度量距离成反比（越近权重越大）。

优缺点：

KNN算法的优点：

1. 思想简单，理论成熟，既可以用来做分类也可以用来做回归；
2. 可用于非线性分类；
3. 训练时间复杂度为O(n)；
4. 准确度高，对数据没有假设，对outlier不敏感；

缺点：

1. 计算量大；
2. 样本不平衡问题（即有些类别的样本数量很多，而其它样本的数量很少）；
3. 需要大量的内存；

KD树

KD树是一个二叉树，表示对K维空间的一个划分，可以进行快速检索（那KNN计算的时候不需要对全样本进行距离的计算了）

构造KD树

在k维的空间上循环找子区域的中位数进行划分的过程。
假设现在有K维空间的数据集$T = \{x_1, x_2, x_3, \dots x_n\}$,$xi = \{a_1, a_2, a_3 . . a_k\}$

1. 首先构造根节点，以坐标a_1的中位数b为切分点，将根结点对应的矩形局域划分为两个区域，区域1中$a_1 < b$,区域2中$a_1 > b$
2. 构造叶子节点，分别以上面两个区域中a_2的中位数作为切分点，再次将他们两两划分，作为深度1的叶子节点，（如果a2=中位数，则a2的实例落在切分面）
3. 不断重复2的操作，深度为j的叶子节点划分的时候，索取的a_i 的$i = j\%k + 1$，直到两个子区域没有实例时停止

KD树的搜索

1. 首先从根节点开始递归往下找到包含x的叶子节点，每一层都是找对应的xi
2. 将这个叶子节点认为是当前的“近似最近点”
3. 递归向上回退，如果以x圆心，以“近似最近点”为半径的球与根节点的另一半子区域边界相交，则说明另一半子区域中存在与x更近的点，则进入另一个子区域中查找该点并且更新”近似最近点“
4. 重复3的步骤，直到另一子区域与球体不相交或者退回根节点
5. 最后更新的”近似最近点“与x真正的最近点

KD树进行KNN查找

通过KD树的搜索找到与搜索目标最近的点，这样KNN的搜索就可以被限制在空间的局部区域上了，可以大大增加效率。

KD树搜索的复杂度

当实例随机分布的时候，搜索的复杂度为log(N)，N为实例的个数，KD树更加适用于实例数量远大于空间维度的KNN搜索，如果实例的空间维度与实例个数差不多时，它的效率基于等于线性扫描。

> 后来自己有实现过KD树，可以看KNN算法中KD树的应用

SVM、SMO

对于样本点(x_i, y_i)以及svm的超平面：$w^T x_i + b = 0$

- 函数间隔：$y_i(w^T x_i + b)$
- 几何间隔：$\frac{y_i(w^T x_i+b)}{\|w\|}$,其中$\|w\|$为$w$的L2范数，几何间隔不会因为参数比例的改变而改变

> svm的基本想法就是求解能正确划分训练样本并且其几何间隔最大化的超平面。

线性SVM问题

先来看svm的问题:

$$\begin{aligned} & \underset{w,b}{argmax} \quad \gamma \\ st. \quad & \frac{y_i(w^T x_i + b)}{\|w\|} \geq \gamma \end{aligned}$$

那么假设$\hat{\gamma} = \gamma * \|w\|$
则将问题转为：

$$\begin{aligned} & \underset{w,b}{argmax} \quad \frac{\hat{\gamma}}{\|w\|} \\ st. \quad & y_i(w^T x_i + b) \geq 1 \end{aligned}$$

由于$\hat{\gamma}$的成比例增减不会影响实际间距，所以这里的取$\hat{\gamma} = 1$，又因为$max(\frac{1}{\|w\|}) = min(\frac{1}{2} * \|w\|^2)$
所以最终的问题就变为了

$$\underset{w,b}{argmin} \quad \frac{1}{2} * \|w\|^2$$

$$st. \quad y_i(w^T x_i + b) \geq 1$$

这样就变成了一个凸的二次规划化，可以将其转换为拉格朗日函数，然后使用...算法来求解

对偶求解

引进拉格朗日乘子$\alpha = \{\alpha_1, \alpha_2 .. \alpha_n\}$,定义拉格朗日函数：

$$L(w, b, a) = \frac{1}{2} * \|w\|^2 - \sum_{i=1}^{N}(\alpha_i * y_i(w^T x_i + b)) + \sum(\alpha_i)$$

根据对偶性质 原始问题就是求对偶问题的极大极小

$$\underset{\alpha}{max}\underset{w,b}{min}L(w, b, \alpha)$$

先求L对w, b的极小，再求对α的极大。
求$\underset{w,b}{min}L(w, b, \alpha)$，也就是相当于对$w, b$求偏导并且另其等于0

$$\nabla_w L(w, b, \alpha) = w - \sum_{i=1}^{N}(\alpha_i y_i x_i) = 0$$

$$\nabla_b L(w, b, \alpha) = \sum_{i=1}^{N}(a_i y_i) = 0$$

代入后可得

$$\underset{w,b}{min}L(w, b, \alpha) = -\frac{1}{2} * \sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i \alpha_j y_i y_j (x_i \cdot x_j)\right) + \sum_{i=1}^{N}\alpha_i$$

求$\underset{w,b}{min}L(w, b, \alpha)$对$\alpha$的极大，即是对偶问题:

$$\max_{\alpha} -\frac{1}{2} * \sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i\alpha_j y_i y_j (x_i \cdot x_j)\right) + \sum_{i=1}^{N}\alpha_i$$

$$st. \quad \sum_{i=1}^{N}(a_i y_i) = 0$$

$$\alpha \geq 0, i = 1, 2, 3 \ldots N$$

将求最大转为求最小，得到等价的式子为：

$$\min_{\alpha} \frac{1}{2} * \sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i\alpha_j y_i y_j (x_i \cdot x_j)\right) - \sum_{i=1}^{N}\alpha_i$$

$$st. \quad \sum_{i=1}^{N}(a_i y_i) = 0$$

$$\alpha \geq 0, i = 1, 2, 3 \ldots N$$

假如求解出来的α为$\alpha^* = (\alpha_1^*, \alpha_2^*, \ldots \alpha_n^*)$
则得到最优的w, b分别为

$$w^* = \sum_{i=1}^{N}(\alpha_i^* y_i x_i)$$

$$b^* = y_j - \sum_{i=1}^{N}(\alpha_i^* y_i (x_i \cdot x_j))$$

所以，最终的决策分类面为

$$f(x) = sign\left(\sum_{i=1}^{N}(\alpha_i^* y_i (x \cdot x_i) + b^*\right)$$

也就是说，分类决策函数只依赖于输入x与训练样本的输入的内积

> ps：上面介绍的是SVM的硬间距最大化，还有一种是软间距最大化，引用了松弛变量ζ，则次svm问题变为:

$$\underset{w,b}{argmin} \quad \frac{1}{2} * \|w\|^2 + C\sum_{i=1}^{N}\zeta_i$$
$$st. \quad y_i(w^T x_i + b) \geq 1 - \zeta_i$$
$$\zeta_i \geq 0, i = 1, 2 \dots N$$

其余解决是与硬间距的一致~

还有：与分离超平面最近的样本点称为支持向量

损失函数

损失函数为（优化目标）：

$$\sum_{i=1}^{N}[1 - y_i(w^T x_i + b)]_+ + \lambda\|w\|^2$$

其中$[1 - y_i(w^T x_i + b)]_+$称为折页损失函数，因为:

$$[1 - y_i(w^T x_i + b)]_+ = \begin{cases} 0 & if \quad 1 - y_i(w^T x_i + b) \leq 0 \\ 1 - y_i(w^T x_i + b) & otherwise \end{cases}$$

为什么要引入对偶算法

1. 对偶问题往往更加容易求解(结合拉格朗日和kkt条件)
2. 可以很自然的引用核函数（拉格朗日表达式里面有内积，而核函数也是通过内积进行映射的）

核函数

将输入特征x（线性不可分）映射到高维特征R空间，可以在R空间上让SVM进行线性可以变，这就是核函数的作用

- 多项式核函数:$K(x, z) = (x * z + 1)^p$
- 高斯核函数:$K(x, z) = exp(\frac{-(x-z)^2}{\sigma^2})$
- 字符串核函数：貌似用于字符串处理等

SVM优缺点

优点：

1. 使用核函数可以向高维空间进行映射
2. 使用核函数可以解决非线性的分类
3. 分类思想很简单，就是将样本与决策面的间隔最大化
4. 分类效果较好

缺点：

1. 对大规模数据训练比较困难
2. 无法直接支持多分类，但是可以使用间接的方法来做

SMO

SMO是用于快速求解SVM的
它选择凸二次规划的两个变量，其他的变量保持不变，然后根据这两个变量构建一个二次规划问题，这个二次规划关于这两个变量解会更加的接近原始二次规划的解，通过这样的子问题划分可以大大增加整个算法的计算速度，关于这两个变量：

1. 其中一个是严重违反KKT条件的一个变量
2. 另一个变量是根据自由约束确定，好像是求剩余变量的最大化来确定的。

SVM多分类问题

1. 直接法
   直接在目标函数上进行修改，将多个分类面的参数求解合并到一个最优化问题中，通过求解该优化就可以实现多分类（计算复杂度很高，实现起来较为困难）
2. 间接法
   1. 一对多
      其中某个类为一类，其余n-1个类为另一个类，比如A,B,C,D四个类，第一次A为一个类，{B,C,D}为一个类训练一个分类器，第二次B为一个类,{A,C,D}为另一个类,按这方式共需要训练4个分类器，最后在测试的时候将测试样本经过这4个分类器$f_1(x),f_2(x),f_3(x)$和$f_4(x)$,取其最大值为分类器(这种方式由于是1对M分类，会存在偏置，很不实用)
   2. 一对一(libsvm实现的方式)
      任意两个类都训练一个分类器，那么n个类就需要n*(n-1)/2个svm分类器。还是以A,B,C,D为例,那么需要{A,B},{A,C},{A,D},{B,C},{B,D},{C,D}为目标共6个分类器，然后在预测的将测试样本通过这6个分类器之后进行投票选择最

终结果。（这种方法虽好，但是需要n*(n-1)/2个分类器代价太大，不过有好像使用循环图来进行改进）

决策树

决策树是一颗依托决策而建立起来的树。

ID3

1. 首先是针对当前的集合，计算每个特征的信息增益
2. 然后选择信息增益最大的特征作为当前节点的决策决策特征
3. 根据特征不同的类别划分到不同的子节点（比如年龄特征有青年，中年，老年，则划分到3颗子树）
4. 然后继续对子节点进行递归，直到所有特征都被划分

$$S(C, ai) = -\sum_i (p_i * log(p_i))$$

一个属性中某个类别的熵 $p_i = P(y_i|a_i)$, p_i表示a_i情况下发生y_i的概率，也即是统计概率。

$$S(C, A) = \sum_i (P(A = a_i) * S(a_i))$$

整个属性的熵，为各个类别的比例与各自熵的加权求和。

$$Gain(C, A) = S(C) - S(C, A)$$

增益表示分类目标的熵减去当前属性的熵，增益越大，分类能力越强
(这里前者叫做经验熵，表示数据集分类C的不确定性，后者就是经验条件熵，表示在给定A的条件下对数据集分类C的不确定性，两者相减叫做互信息，决策树的增益等价于互信息)。
比如说当前属性是是否拥有房产，分类是是否能偿还债务
现在：

- 有用房产为7个，4个能偿还债务，3个无法偿还债务
- 然后无房产为3个，其中1个能偿还债务，2个无法偿还债务

然后
有房子的熵：$S(have\_house) = -(\frac{4}{7} * log\frac{4}{7} + \frac{3}{7} * log\frac{3}{7})$
无房子的熵：$S(no\_house) = -(\frac{1}{3} * log\frac{1}{3} + \frac{2}{3} * log\frac{2}{3})$

分类的熵：$S(classifier) = -(\frac{5}{10} * log\frac{5}{10} + \frac{5}{10} * log\frac{5}{10})$
最终的增益=$S(classifier) - (\frac{7}{10} * S(have\_house) + \frac{3}{10} * S(no\_house))$ 最大越好

关于损失函数
设树的叶子节点个数为T，t为其中一个叶子节点，该叶子节点有N_t个样本，其中k类的样本有N_{tk}个，$H(t)$为叶子节点上的经验熵，则损失函数定义为

$$C_t(T) = \sum(N_t * H(t)) + \lambda|T|$$

其中

$$H(t) = \sum(\frac{N_{tk}}{N_t} * log(\frac{N_{tk}}{Nt}))$$

代入可以得到

$$C_t(T) = \sum(\sum(N_{tk} * log(N_{tk}/N_t))) + \lambda|T|$$

$\lambda|T|$为正则化项，λ是用于调节比率
决策树的生成只考虑了信息增益

C4.5

它是ID3的一个改进算法，使用信息增益率来进行属性的选择

$$splitInformation(S, A) = -\sum_i(\frac{|S_i|}{|S|} * log2(\frac{|Si|}{|S|}))$$

$$GainRatio(S, A) = \frac{Gain(S, A)}{splitInformation(S, A)}$$

优缺点：
准确率高，但是子构造树的过程中需要进行多次的扫描和排序，所以它的运算效率较低

Cart

分类回归树(Classification And Regression Tree)是一个决策二叉树，在通过递归的方式建立，每个节点在分裂的时候都是希望通过最好的方式将剩余的样本划分成两类，这里的分类指标：

1. 分类树：基尼指数最小化(gini_index)
2. 回归树：平方误差最小化

分类树：

1. 首先是根据当前特征计算他们的基尼增益
2. 选择基尼增益最小的特征作为划分特征
3. 从该特征中查找基尼指数最小的分类类别作为最优划分点
4. 将当前样本划分成两类，一类是划分特征的类别等于最优划分点，另一类就是不等于
5. 针对这两类递归进行上述的划分工作，直达所有叶子指向同一样本目标或者叶子个数小于一定的阈值

gini用来度量分布不均匀性（或者说不纯），总体的类别越杂乱，GINI指数就越大（跟熵的概念很相似）

$$gini(a_i) = 1 - \sum_i (p_i^2)$$

p_i当前数据集中第i类样本的比例
gini越小，表示样本分布越均匀（0的时候就表示只有一类了），越大越不均匀
基尼增益

$$gini\_gain = \sum_i (\frac{N_i}{N} * gini(a_i))$$

表示当前属性的一个混乱 $\frac{N_i}{N}$表示当前类别占所有类别的概率
最终Cart选择GiniGain最小的特征作为划分特征

以ID3中的贷款的那棵树为样例：
基尼指数有房产:$gini(have\_house) = 1 - \left((\frac{3}{7})^2 + (\frac{4}{7})^2\right)$
基尼指数无房产:$gini(no\_house) = 1 - \left((\frac{1}{3})^2 + (\frac{2}{3})^2\right)$
基尼增益为:$gini\_gain = \frac{7}{10} * gini(have\_house) + \frac{3}{10} * gini(no\_house)$

回归树：

> 回归树是以平方误差最小化的准则划分为两块区域

1. 遍历特征计算最优的划分点s，
   使其最小化的平方误差是：
   $min\{min(\sum_i^{R1}((y_i - c_1)^2)) + min(\sum_i^{R2}((y_i - c_2)^2))\}$

计算根据s划分到左侧和右侧子树的目标值与预测值之差的平方和最小，这里的预测值是两个子树上输入xi样本对应y_i的均值

2. 找到最小的划分特征j以及其最优的划分点s,根据特征j以及划分点s将现有的样本划分为两个区域，一个是在特征j上小于等于s，另一个在在特征j上大于s

$$R1(j) = \{x|x(j) \leq s\}$$
$$R2(j) = \{x|x(j) > s\}$$

3. 进入两个子区域按上述方法继续划分，直到到达停止条件

> 这里面的最小化我记得可以使用最小二乘法来求

关于剪枝：用独立的验证数据集对训练集生长的树进行剪枝（事后剪枝）。

停止条件

1. 直到每个叶子节点都只有一种类型的记录时停止，（这种方式很容易过拟合）
2. 另一种时当叶子节点的记录树小于一定的阈值或者节点的信息增益小于一定的阈值时停止

关于特征与目标值

1. 特征离散 目标值离散：可以使用ID3，cart
2. 特征连续 目标值离散：将连续的特征离散化 可以使用ID3，cart
3. 特征离散 目标值连续

决策树的分类与回归

- 分类树
  输出叶子节点中所属类别最多的那一类
- 回归树
  输出叶子节点中各个样本值的平均值

理想的决策树

1. 叶子节点数尽量少
2. 叶子节点的深度尽量小(太深可能会过拟合)

解决决策树的过拟合

1. 剪枝

1. 前置剪枝：在分裂节点的时候设计比较苛刻的条件，如不满足则直接停止分裂（这样干决策树无法到最优，也无法得到比较好的效果）
2. 后置剪枝：在树建立完之后，用单个节点代替子树，节点的分类采用子树中主要的分类（这种方法比较浪费前面的建立过程）

2. 交叉验证
3. 随机森林

优缺点

优点：

1. 计算量简单，可解释性强，比较适合处理有缺失属性值的样本，能够处理不相关的特征；
   缺点：
2. 单颗决策树分类能力弱，并且对连续值变量难以处理；
3. 容易过拟合（后续出现了随机森林，减小了过拟合现象）；

随机森林RF

随机森林是有很多随机得决策树构成，它们之间没有关联。得到RF以后，在预测时分别对每一个决策树进行判断，最后使用Bagging的思想进行结果的输出（也就是投票的思想）

学习过程

1. 现在有N个训练样本，每个样本的特征为M个，需要建K颗树
2. 从N个训练样本中有放回的取N个样本作为一组训练集（其余未取到的样本作为预测分类，评估其误差）
3. 从M个特征中取m个特征左右子集特征(m<<M)
4. 对采样的数据使用完全分裂的方式来建立决策树，这样的决策树每个节点要么无法分裂，要么所有的样本都指向同一个分类
5. 重复2的过程K次，即可建立森林

预测过程

1. 将预测样本输入到K颗树分别进行预测
2. 如果是分类问题，直接使用投票的方式选择分类频次最高的类别

3. 如果是回归问题，使用分类之后的均值作为结果

参数问题

1. 这里的一般取m=sqrt(M)
2. 关于树的个数K，一般都需要成百上千，但是也有具体的样本有关（比如特征数量）
3. 树的最大深度，（太深可能可能导致过拟合？？）
4. 节点上的最小样本数、最小信息增益

泛化误差估计

使用oob（out-of-bag）进行泛化误差的估计，将各个树的未采样样本作为预测样本（大约有36.8%），使用已经建立好的森林对各个预测样本进行预测，预测完之后最后统计误分得个数占总预测样本的比率作为RF的oob误分率。

学习算法

1. ID3算法：处理离散值的量
2. C45算法：处理连续值的量
3. Cart算法：离散和连续 两者都合适?

关于CART

Cart可以通过特征的选择迭代建立一颗分类树，使得每次的分类平面能最好的将剩余数据分为两类

$gini = 1 - \sum(p_i^2)$，表示每个类别出现的概率和与1的差值，
分类问题：$argmax(Gini - GiniLeft - GiniRight)$
回归问题：$argmax(Var - VarLeft - VarRight)$

查找最佳特征f已经最佳属性阈值th 小于th的在左边，大于th的在右边子树

优缺点

1. 能够处理大量特征的分类，并且还不用做特征选择
2. 在训练完成之后能给出哪些feature的比较重要
3. 训练速度很快
4. 很容易并行
5. 实现相对来说较为简单

GBDT

> GBDT的精髓在于训练的时候都是以上一颗树的残差为目标，这个残差就是上一个树的预测值与真实值的差值。

比如，当前样本年龄是18岁，那么第一颗会去按18岁来训练，但是训练完之后预测的年龄为12岁，差值为
所以第二颗树的会以6岁来进行训练，假如训练完之后预测出来的结果为6，那么两棵树累加起来就是真实年
但是假如第二颗树预测出来的结果是5，那么剩余的残差1就会交给第三个树去训练。

Boosting的好处就是每一步的参加就是变相了增加了分错instance的权重，而对已经对的instance趋向于0，这样后面的树就可以更加关注错分的instance的训练了

Shrinkage

Shrinkage认为，每次走一小步逐步逼近的结果要比每次迈一大步逼近结果更加容易避免过拟合。

$$y(1 \sim i) = y(1 \sim i-1) + step * y_i$$

> 就像我们做互联网，总是先解决60%用户的需求凑合着，再解决35%用户的需求，最后才关注那5%人的需求，这样就能逐渐把产品做好.

调参

1. 树的个数 100~10000
2. 叶子的深度 3~8
3. 学习速率 0.01~1
4. 叶子上最大节点树 20
5. 训练采样比例 0.5~1
6. 训练特征采样比例 sqrt(num)

优缺点：

优点：

1. 精度高
2. 能处理非线性数据
3. 能处理多特征类型
4. 适合低维稠密数据

   缺点：
5. 并行麻烦（因为上下两颗树有联系）

6. 多分类的时候 复杂度很大

BP

最小二乘法

最小二乘法是一种数学的优化技术，通过求最小化平方误差来寻找最佳的函数匹配
假设现在有二维的观测数据$(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$，求$y = a + bx$的拟合。

现设$yi = a + b * x_i + k_i$ 如果有a, b能得到$\sum_{i=1}^{N}(|ki|)$最小，则该线比较理想
所以先变为求$min(\sum_{i=1}^{N}(k_i))$ ，这个与$min(\sum_{i=1}^{N}(k_i^2))$等价
而$k_i = y_i - (a + b * x_i)$
那么现设$f = \sum_{i=1} N \left((y_i - (a + b * x_i))^2\right)$求其最小即可

> 上述就是最小二乘原则，估计a, b的方法称为最小二乘法

先求f对a, b的偏导：

$$\nabla_a f = -2 * \sum_{i=1}^{N} (y_i - (a + b * x_i)) = 0$$

$$\nabla_b f = -2 * xi * \sum_{i=1}^{N} (y_i - (a + b * x_i)) = 0$$

现设：

$$X = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$Y = \frac{\sum_{i=1}^{N} y_i}{N}$$

则代入上述偏导：

$$a*N+b*N*X=N*Y$$
$$a*N*X+b*\sum_{i=1}^{N}(x_i^2)=\sum_{i=1}^{N}(x_i*y_i)$$

求该行列式:

$$\begin{vmatrix} N & N*X \\ N*X & \sum_{i=1}^{N} x_i^2 \end{vmatrix} = N*\sum_{i=1}^{N}((x_i-X))! = 0$$

所以有唯一解

最后记：

$$l(xx)=\sum_{i=1}^{N}(x_i-X)^2$$
$$l(yy)=\sum_{i=1}^{N}(y_i-Y)^2$$
$$l(xy)=\sum_{i=1}^{N}((x_i-X)(y_i-Y))$$

则

$$b=\frac{l(xy)}{l(xx)}$$
$$a=Y-b*X$$

百度文库-最小二乘法

EM

EM用于隐含变量的概率模型的极大似然估计，它一般分为两步：第一步求期望(E)，第二步求极大(M)，
如果概率模型的变量都是观测变量，那么给定数据之后就可以直接使用极大似然法或者贝叶斯估计模型参数。
但是当模型含有隐含变量的时候就不能简单的用这些方法来估计，EM就是一种含有隐含变量的概率模型参数的极大似然估计法。

应用到的地方：混合高斯模型、混合朴素贝叶斯模型、因子分析模型

Bagging

1. 从N样本中有放回的采样N个样本
2. 对这N个样本在全属性上建立分类器(CART,SVM)
3. 重复上面的步骤，建立m个分类器
4. 预测的时候使用投票的方法得到结果

Boosting

boosting在训练的时候会给样本加一个权重，然后使loss function尽量去考虑那些分错类的样本（比如给分错类的样本的权重值加大）

凸优化

在机器学习中往往是最终要求解某个函数的最优值，但是一般情况下，任意一个函数的最优值求解比较困难，但是对于凸函数来说就可以有效的求解出全局最优值。

凸集

一个集合C是，当前仅当任意x,y属于C且$0 \leq \Theta \leq 1$，都有
$\Theta * x + (1 - \Theta) * y$属于C
用通俗的话来说C集合线段上的任意两点也在C集合中

凸函数

一个函数f其定义域(D(f))是凸集，并且对任意x,y属于D(f)和$0 \leq \Theta \leq 1$都有

$$f(\Theta * x + (1 - \Theta) * y) \leq \Theta * f(x) + (1 - \Theta) * f(y)$$

用通俗的话来说就是曲线上任意两点的割线都在曲线的上方

常见的凸函数有：

- 指数函数$f(x) = a^x; a > 1$
- 负对数函数$-logax; a > 1, x > 0$

- 开口向上的二次函数等

凸函数的判定：

1. 如果f是一阶可导，对于任意数据域内的x,y满足$f(y) \geq f(x) + f'(x)(y -$
2. 如果f是二阶可导，

凸优化应用举例

- SVM：其中由$max|w|$ 转向$min(\frac{1}{2} * |w|^2)$
- 最小二乘法?
- LR的损失函数$\sum (y_i * log(h_w(x_i)) + (1 - y_i) * (log(1 - h_w(x_i))))$

参考

[1]. http://www.cnblogs.com/leoo2sk/archive/2010/09/17/naive-bayesian-classifier.html
[2]. http://www.cnblogs.com/biyeymyhjob/archive/2012/07/18/2595410.html
[3]. http://blog.csdn.net/abcjennifer/article/details/7716281
[4]. http://ufldl.stanford.edu/wiki/index.php/Softmax%E5%9B%9E%E5%BD%92
[5]. 《统计学习方法》.李航

备注

资料主要来源于网络或者《统计学习方法》，还有自己一小部分的总结，如果错误之处敬请指出

Machine Learning Machine Learning

上一篇：
《台大机器学习基石》Linear Regression

下一篇：

› 机器学习常见面试题整理