

**模式识别大作业**

题 目： 汽车投保风险预测

学 院： 信息科学与工程学院

专 业： 控制科学与控制工程

组 员： 陆莉霞 王峰 岳丹阳 吕奕 程奇

指导老师：赵海涛

**完成日期： 2018 年 12 月16日**

**小组分工：**

|  |  |
| --- | --- |
| 论文撰写 | 程奇、吕奕 |
| SMO算法程序实现 | 陆莉霞 |
| CART回归树及剪枝算法程序实现 | 岳丹阳 |
| CART与随机森林算法程序实现 | 王峰 |

第一章 题目分析

**1.1 题目描述**

某保险公司销售一种汽车保险，需要对汽车状态进行评估。现在你需要设计一个算法模型，可以根据汽车的各项指标对汽车的投保风险进行打分。投保风险是从0到70的正整数，数值越大代表风险越高。

**1.2 题目目标**

给定汽车的各项性能指标，设计算法对汽车的投保风险进行打分。

**1.3 题目分析**

汽车投保风险是一个典型的回归问题。先从train.csv中提取每辆车的多项特征（feature）和类别(label)，使用feature和label进行模型训练。

模型考虑使用传统的回归模型，如[SVR(支持向量回归)](http://www.jianshu.com/p/c867e9234035)，[CART(分类与回归树)](http://blog.csdn.net/baimafujinji/article/details/53269040)等。在完成传统回归模型的基础上，使用集成学习的算法将多个简单模型集成为一个复杂模型，本文采用[Random Forest(随机森林)](https://baike.baidu.com/item/%E9%9A%8F%E6%9C%BA%E6%A3%AE%E6%9E%97?fromtitle=Random+forest&fromid=18081353)完成分类。最后使用训练好的模型预测test.csv中每辆车的风险分数。使用RMSE (Root Mean Square Error)作为评价指标，公式如下：

（1-1）

其中N代表测试数据集中汽车的数量，代表其真实的风险值，代表预测的风险值。

第二章 算法介绍

**2.1 CART回归树**

**2.1.1 算法介绍**

CART是在给定输入随机变量X条件下输出随机变量Y的条件概率分布的学习方法。假设决策树是二叉树，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支。这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布。

**2.1.2 CART算法步骤**

CART算法由以下两个步骤组成。决策树生成，基于训练数据集生成决策树，生成的决策树要尽量大；决策树剪枝，用验证数据集对已生成的树进行剪枝并选择最优子树，这时损失函数最小作为剪枝的标准。

**2.1.3 CART生成**

决策树的生成就是递归地构建二叉决策树的过程，对回归树用平方误差最小化准则，对分类树用基尼指数（Gini index）最小化准则，进行特征选择，生成二叉树。本文主要使用了回归树。

假设X与Y分别是输入和输出变量，并且Y是连续变量，给定训练数据集

（2-1）

考虑如何生成回归树。

一个回归树对应着输入空间的一个划分以及在划分的单元上的输出值。假设已经将输入空间划分为M个单元，并且在每个单元上有一个固定的输出值，于是回归树模型可以表示为

（2-2）

当输入空间的划分确定时，可以用平方误差来表示回归树对于训练数据的预测误差，用平方误差最小的准则求解每个单元上的最优输出值。易知，单元上的的最优值是上的所有输入实例对应的输出的均值，即

（2-3）

这里采用启发式的方法，选择第j个变量和它取的值s，作为切分变量（splitting variable）和切分点（splitting point），并定义两个区域：

和 （2-4）

然后寻找最优切分变量j和最优切分点s。具体地，求解

（2-5）

对固定输入变量j可以找到最优切分点s。

和 （2-6）

遍历所有输入变量，找到最优的切分变量j，构成一个对（j,s）。依次将输入空间划分为两个区域。接着，对每个区域重复上述划分过程，知道满足停止条件为止。这样就生成一颗回归树。这样的回归树通常称为最小二乘回归树。

**2.1.4 最小二乘回归树的生成算法**

输入：训练数据集D；

输出：回归树f(x).

在训练数据集所在的输入空间中，递归地将每个区域划分为两个子区域并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量j与切分点s，求解

（2-7）

遍历变量j，对固定的切分变量j扫描切分点s，选择使上式达到最小值的对（j,s）。

（2）用选定的对（j,s）划分区域并决定相应的输出值：

和 （2-8）

（2-9）

（3）继续对两个子区域调用步骤（1），（2），直至满足停止条件。

（4）将输入空间划分为M个区域，生成决策树：

（2-10）

**2.1.5 CART剪枝**

CART剪枝算法由两部分组成：首先从生成算法产生的决策树T0底端开始不断剪枝，直到T0的根节点，形成一个子树序列{ T0，T1，…，Tn}；然后通过交叉验证法在独立的验证数据集上对子树序列进行测试，从中选择最优子树。

1、剪枝形成子树序列

决策树的剪枝往往通过极小化决策树整体的损失函数（loss function）或代价函数（cost function）来实现。设树T的叶节点个数为|T|，t是树T的叶节点，该树叶节点有Nt个样本点，其中k类的样本点有Ntk个，k=1，2，…，K，Ht(T)为叶节点t上的经验熵，为参数，则决策树学习的损失函数定义为：

（2-11）

其中经验熵为

（2-12）

在损失函数中，令

（2-13）

则计算子树的损失函数：

（2-14）

其中为对训练数据的预测误差，为参数是时的子树T的整体损失，树权衡训练数据的拟合程度与模型的复杂度。

对于固定的，一定存在使损失函数最小的子树，将其表示为。在损失函数最小的意义下是最优的。大的时候，最优子树偏小，当时，根节点组成的单结点树是最优的；小的时候，最优子树偏大，当时，整体树是最优的。

Beriman 等人证明：可以用递归方法对树进行剪枝，将从小增大，0 = ,产生一系列的区间；剪枝得到的子树序列对应着区间的最优子树序列{ T0，T1，…，Tn}，序列的子树是嵌套的。

从整体树T0开始剪枝，对T0的任意内部结点t，以t为单结点树的损失函数为

（2-15）

以t为根节点的子树的损失函数是

（2-16）

当及充分小时，有不等式

（2-17）

当增大时，某一有

（2-18）

再继续增大时，上述不等式反向，因此只要

（2-19）

与t有相同的损失函数，而t的结点少，比更可取，故对进行剪枝。因此，对中每一内部结点t，计算

（2-20）

该式表示剪枝后整体损失函数的减小程度。在中剪去最小的，得到的子树作为，同时将最小的设为，为区间的最优子树。如此下去，直到得到根结点。

2、选取最优子树

利用独立的验证数据集，测试子树序列中各棵子树的平方误差或基尼指数。平方误差或基尼指数最小的树为最优的决策树，在子树序列中，每棵子树对应于一个参数，，…，。所有最优子树确定时，对应的也确定了，即得到最优的树。

**2.2 Random Forest(随机森林)**

**2.2.1 算法介绍**

随机森林就是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树，而它的本质属于机器学习的一大分支——集成学习（Ensemble Learning）方法。随机森林的名称中有两个关键词，一个是“随机”，一个就是“森林”。“森林”我们很好理解，一棵叫做树，那么成百上千棵就可以叫做森林了，随机是指每次随机从训练集中又放回地抽取同等数量的训练数据。

从直观角度来解释，每棵决策树都是一个分类器（现在针对的是回归问题），那么对于一个输入样本，N棵树会有N个回归结果。而随机森林集成了所有的回归结果，将所有树的回归结果的平均值指定为最终的输出，这就是一种最简单的 Bagging 思想。

**2.2.2 随机森林的训练过程**

（1）给定训练集S，测试集T，特征维数F。确定参数：使用到的CART的数量t，每棵树的深度d，每个节点使用到的特征数量f，终止条件：节点上最少样本数s，节点上最少的信息增益m。对于第1-t棵树，i=1-t；

（2）从S中有放回的抽取大小和S一样的训练集S(i)，作为根节点的样本，从根节点开始训练；

（3）如果当前节点上达到终止条件，则设置当前节点为叶子节点，如果是分类问题，该叶子节点的预测输出为当前节点样本集合中数量最多的那一类c(j)，概率p为c(j)占当前样本集的比例；如果是回归问题，预测输出为当前节点样本集各个样本值的平均值。然后继续训练其他节点。如果当前节点没有达到终止条件，则从F维特征中无放回的随机选取f维特征。利用这f维特征，寻找分类效果最好的一维特征k及其阈值th，当前节点上样本第k维特征小于th的样本被划分到左节点，其余的被划分到右节点。继续训练其他节点。

1）重复（2)，（3)直到所有节点都训练过了或者被标记为叶子节点。

2）重复（2)，（3)，（4)直到所有CART都被训练过。

（4）利用随机森林的预测过程。

对于第1-t棵树，i=1-t：

1）从当前树的根节点开始，根据当前节点的阈值th，判断是进入左节点(<th)还是进入右节点(>=th)，直到到达，某个叶子节点，并输出预测值。

2）重复执行(1)直到所有t棵树都输出了预测值。如果是分类问题，则输出为所有树中预测概率总和最大的那一个类，即对每个c(j)的p进行累计；如果是回归问题，则输出为所有树的输出的平均值。

**2.2.3 随机森林特点**

（1）在当前所有算法中，具有极好的准确率；

（2）能够有效地运行在大数据集上；

（3）能够处理具有高维特征的输入样本，而且不需要降维；

（4）能够评估各个特征在分类问题上的重要性；

（5）在生成过程中，能够获取到内部生成误差的一种无偏估计；

（6）对于缺省值问题也能够获得很好得结果。

（7）当随机森林中的决策树个数很多时，训练时需要的空间和时间会较大

（8）随机森林模型还有许多不好解释的地方，算黑盒模型

**2.3 SMO（序列最小优化）算法**

我们已经知道了如何去求数据点到超平面的距离，在超平面确定的情况下，我们就能够找出所有支持向量，然后计算出间隔margin。因此用数学语言描述就是确定w、b使得margin最大。这是一个优化问题其目标函数可以写成：、

， （2-21）

为了后面计算的方便，我们将目标函数等价替换为：）（2-22）

这是一个有约束条件的优化问题，通常我们可以用拉格朗日乘子法来求解。应用拉格朗日乘子法如下：

 （2-23）

求L关于求偏导数得：

 （2-24）

将(1.7)代入到(1.6)中化简得：

（2-25）

原问题的对偶问题为：

 （2-26）

其中：

该对偶问题的KKT条件为

 （2-27）

**2.3.1 SMO算法原理**

解SVM问题最终演化为求下列带约束条件的问题：

 （2-28）

 （2-29）



其中：为拉格朗日乘子，一个变量对应一个样本点。总共有N个，问题的解就是找到一组使得目标函数值最小。

1.基本思路

假设将以外的变量固定看作已知，可以在变量的基础上求极值。又因为等式约束条件可知，那么将不在是变量，可以由其他变量表示。因此，我们需要一次性选取两个变量做优化，假设选取和，其中可由表示，则目标函数可以转化为关于的函数。最后可以针对这两个变量构建一个二次规划问题。

2.两个变量二次规划的求解方法

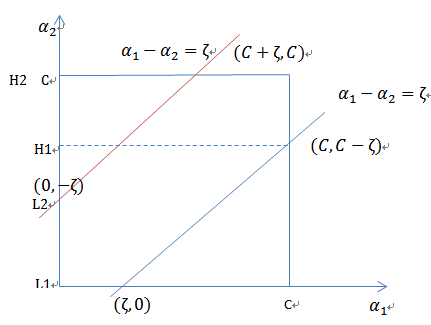
不失一般性，假设选取两个变量为，固定其他变量。则SMO最优化问题式2-1，2-2可以改写成：

（2-30）

 （2-31）

 （2-32）

其中：，是常数，目标函数省略了不含常数项。根据约束条件式2-4，当与不同，即为1，-1时，因此如下图所示，既要在方框内也要在直线上。



**2.1** **二变量优化图示图**

假设初始解为，，最优解为，，假设L和H为对角线上的端点，则根据式2-5，。

当与不同，即为1，-1时，：

 （2-33）

当与相同，即都为1时，

 （2-34）

根据式2-4，得 （2-35）

将式2-6代入目标函数2-3，得到只含变量的目标函数,然后对求导，令其为0，

 （2-36）

 （2-37）

其中,，

 （2-38）

 （2-39）

3.变量的选择方法

SMO算法选取的两个变量，一个是违反KKT条件最严重的一个，一个是根据约束条件自动确定的。在选择两个变量进行优化时，其中至少有一个是违反KKT条件的。

（1）第一个变量的选择

SMO选择第一个变量的过程为外层循环。外层循环选择违反KKT条件最严重的样本点，将其对应的变量定为第一变量。具体步骤，检查样本点是否满足KKT条件。

 （2-40）

 （2-41）

 （2-42）

其中：

该检验是在范围内进行的。再检验过程中，外层循环先遍历所有满足条件的样本点，即在间隔边界上的支持向量点，检查它们是否满足条件KKT条件。如果这些样本都满足KKT条件，那么遍历整个训练集，检验是否满足KKT条件。

（2）第二个变量的选择

SMO称选择第二个变量的过程称为内层循环。假设外层循环已经找到第一个变量，现在要在内层循环中找到第二个变量。第二个变量的选择是希望使有足够大的变化。

由2-10可知，与成正比。要想足够大，只需要足够大，因为与相关且确定，若为正，尽量选择较小的作为，若为负，尽量选择较大的作为。为了节省时间，将所有的值保存在一列表中。

在特殊情况下，如果内层循环通过上述方法选择的不能使目标函数有足够的下降，则采用一下启发式规则继续选择。遍历在间隔边界上的支持矢量点，依次将其对应的变量作为试用，直到目标函数有足够的下降。若找不到合适的，那么遍历训练数据集；依旧找不到，则放弃第一个变量，通过外部循环寻找。依次进行循环往复，直到找到合适的，。

（3）计算阙值b和差值

在每次完成两个变量的优化后，都要重新计算阙值b。当时，由KKT条件式2-14可知：

 （2-43）

于是：

 （2-44）

则：

 （2-45）

上式改写成：

 （2-46）

式2-20带入式2-18得：

 （2-47）

同理可得：

：（2-48）

如果，同时满足条件，那么。若，是0或C，那么和以及它们之间的数都是符合KKT条件的阙值，这时选择它们的中心点作为。在每次完成两个变量的优化之后，还必须更新的值，并把它们保存在列表中。值的更新要用到值，以及所有支持向量对应的：

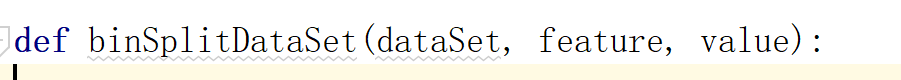
 (2-49)

其中，s是所有支持向量的集合。

第三章 程序实现及仿真

**3.1 CART**

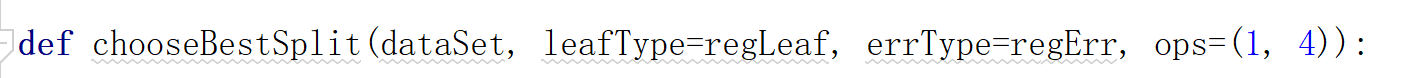
主要的函数作用列举如下：

（1）通过binSplitDataSet函数将数据集按照feature列的value进行二元切分

该函数的作用是在给定特征和特征值的情况下，通过数组过滤方式将上述数据集合切分得到两个子集并返回。其中dataMat 表示数据集，feature表示待切分的特征列，value 表示特征列要比较的值，返回值为mat0和mat1。mat0表示小于等于 value 的数据集在左边，mat1 表示大于 value 的数据集在右边。

（2）计算每一个叶子结点的均值和方差。regLeaf是产生叶节点的函数，就是求均值，即用聚类中心点来代表这类数据。regErr是求这组数据的方差，即通过决策树划分，可以让靠近的数据分到同一类中去。

（3）用最佳方式切分数据集 和 生成相应的叶节点



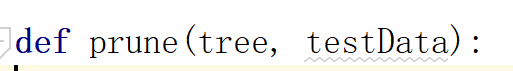
其中dataSet为加载的原始数据集，leafType为建立叶子点的函数。errType为误差计算函数(求总方差)，ops表示[容许误差下降值，切分的最少样本数]。

返回值为bestIndex 表示feature的index坐标和 bestValue 表示切分的最优值。其中，ops=(1,4)，非常重要，因为它决定了决策树划分停止的threshold值，被称为预剪枝（prepruning），其实也就是用于控制函数的停止时机。之所以这样说，是因为它防止决策树的过拟合，所以当误差的下降值小于tolS，或划分后的集合size小于tolN时，选择停止继续划分。

（4）获取回归树

该函数是一个递归函数，即如果构建的是回归树，该模型是一个常数；如果是模型树，其模型是一个线性方程。其中函数的定义与（3）基本相同，返回值为retTree，表示决策树最后的结果

（5）检查是否适合合并分支

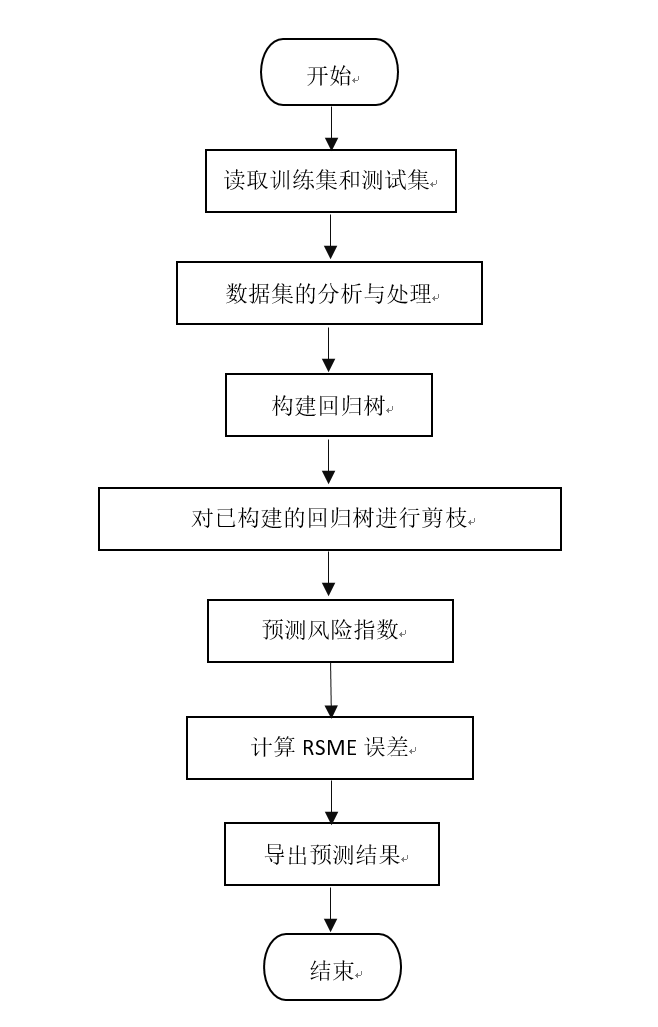


该函数的目的是从上而下找到叶节点，用测试数据集来判断将这些叶节点合并是否能降低测试误差。其中tree表示待剪枝的树，testData表示剪枝所需要的测试数据，返回值为剪枝完成的树tree。

（6）预测结果

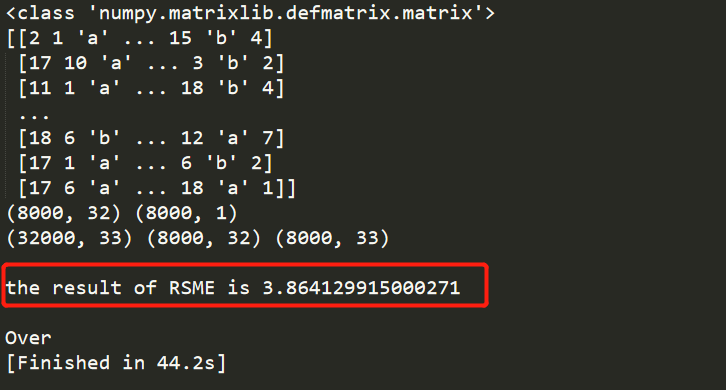
该函数的作用是调用treeForeCast，对特定模型的树进行预测。其中，tree是已经训练好的树的模型，testData是输入的测试数据，modelEval是预测的树的模型类型。返回值为预测值矩阵。

程序流程图如下所示：



**3.1 CART算法流程图**

程序运行结果如下图所示,计算得出的RSME为3.86421。从实验结果来看，CART算法已经能够得到比较好的实验结果。



**3.2 CART算法结果图**

从程序的计算时间来看，CART的运行时间大概为44.2s，用时较短。CART可以将类别特征和数字特征一同处理，例如，投保数据集中有abc类的字母数据，可以与数字特征一起进行构建回归树，而不用单独提出处理。

**3.2 随机森林**

（1）随机森林是建立在CART决策树基础之上的，多个决策树的组合，因此不再详细说明各函数的作用，仅针对森林部分做一个简要介绍。

在原有的训练集中通过不放回抽样的方式，生成50个训练集，每一个训练集训练得出决策树，从而50个决策树形成随机森林。

**for** i **in** range (numtree):

*#森林部分，每个森林训练了50个决策树* a = [int(random.uniform(0, datasettrain.shape[0]-1)) **for** i **in** range(datasettrain.shape[0])]  
 *#产生一个0~32000重复随机数的列表* **for** i **in** range(len(a)):  
 trainset.append(datasettrain[a[i]])

*#随机森林每一棵树的训练集是原训练集有放回抽样得到的* regtree.append(creattree(mat(trainset),regleaf,regerr))

*#将每个决策树放入 regtree列表中* trainset=[]

将测试集中每一条数据经过随机森林中每一棵决策树测试，最终的测试结果为所有决策树回归结果的平均值

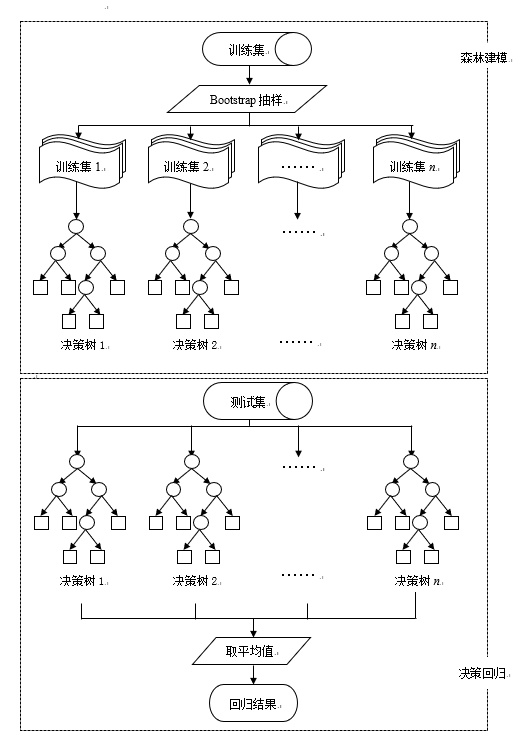
**for** i **in** range(2000):

*#遍历每一个测试集数据* sumtemp=0  
 **for** j **in** range(50):  
 testvector= datasettest[i]  
 temp=classify(regtree[j], testvector)

*#测试集数据在每一棵树下的回归结果* sumtemp+=temp  
 err=(answer[i][0]-sumtemp/50)

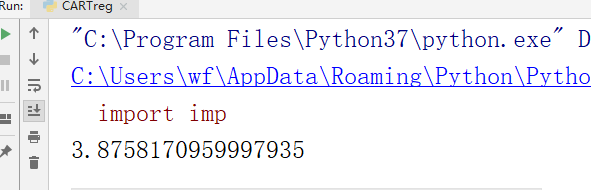
*# 每一个数据测试得分与原数据得分之差*

程序流程图如下图所示：



**3.3 随机森林程序流程图**

程序运行结果如下图所示,计算得出的RSME为3.875817。



**3.4 随机森林程序结果图**

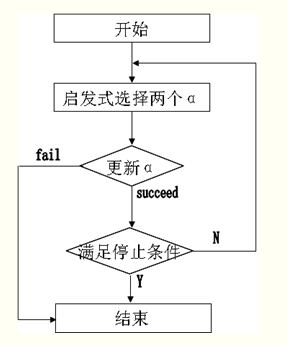
从程序的运行时间来看，随机森林的算法运行时间大概为1h，时间上不占优势。

总结：

随机森林是一个比较优秀的模型，它对于多维特征的数据集分类有很高的效率，还可以做特征重要性的选择。运行效率和准确率较高，实现起来也比较简单。但是在数据噪音比较大的情况下会过拟合，过拟合的缺点对于随机森林来说还是较为致命的。

**3.3 SMO算法流程及仿真结果**

**3.3.1 SMO算法流程**



**3.5 SMO算法流程图**

输入：训练数据集，其中 精度； 输出：近似解

（1）取初值，令k=0；

（2）选取优化变量，解析求解两个变量的最优化问题，求得最优解，更新为；

（3）若在精度范围内，满足停机条件

 (3-1)

其中，

则转（4）；否则令k=k+1，转（2）

（4）

**3.3.2 主程序分析**

（1）优化两个变量。根据KKT条件下，随机选择第一个变量。第二个变量的选择与Ei有关，故先计算Ei，选择合适的Ei对应的变量作为第二个变量。

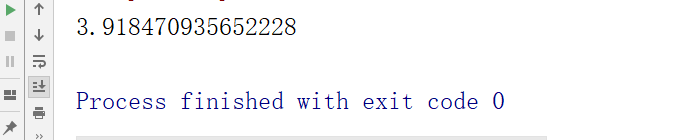
|  |
| --- |
| #步骤1：计算误差  fXi = float(np.multiply(alphas,labelMat).T\*(dataMatrix\*dataMatrix[i,:].T)) + b  Ei = fXi - float(labelMat[i])  #优化alpha，设定一定的容错率。  if ((labelMat[i]\*Ei < -toler) and (alphas[i] < C)) or ((labelMat[i]\*Ei > toler) and (alphas[i] > 0)):  #随机选择另一个与alpha\_i成对优化的alpha\_j  j = selectJrand(i,m)  #步骤1：计算误差Ej  fXj = float(np.multiply(alphas,labelMat).T\*(dataMatrix\*dataMatrix[j,:].T)) + b  Ej = fXj - float(labelMat[j])  #保存更新前的aplpha值  alphaIold = alphas[i].copy(); alphaJold = alphas[j].copy(); |

1. 优化完两个变量后，还有更新阙值b和Ei。

|  |
| --- |
| #步骤7：更新b\_1和b\_2  b1 = b - Ei- labelMat[i]\*(alphas[i]-alphaIold)\*dataMatrix[i,:]\*dataMatrix[i,:].T - labelMat[j]\*(alphas[j]-alphaJold)\*dataMatrix[i,:]\*dataMatrix[j,:].T  b2 = b - Ej- labelMat[i]\*(alphas[i]-alphaIold)\*dataMatrix[i,:]\*dataMatrix[j,:].T - labelMat[j]\*(alphas[j]-alphaJold)\*dataMatrix[j,:]\*dataMatrix[j,:].T |

仿真结果：

通过仿真，运行2次，最后求得RMSE=3.91845，仿真时间t=50.8min



**3.6 SMO算法仿真结果图**

**3.4 结果比较分析**

通过以上仿真实验，得出三种不同算法下汽车投保风险的分析估计值，并绘制如下表格。

**表3.1 汽车投保风险的分析估计值**

|  |  |  |  |
| --- | --- | --- | --- |
| 算法 | 决策树 | 随机森林 | SMO |
| RMSE（结果） | 3.8641 | 3.8758 | 3.9185 |
| T(时间) | 44.2s | 60.0min | 5min |

通过表格，可以观察到，在决策树和随机森林算法下，汽车投保风险的估计值较小，结果最优，但决策树算法下收敛速度最快。相比之下，SMO算法此基本思想是用两个线性优化来取代二次优化，当H=L即线性与二次的分界线，线性无法逾越这一层，SMO只是二次优化的一个近似，因此仿真结果就比较差一些。主要是因为CART算法可以生成理解的规则，计算量相对不大，通过决策树可以清楚的显示哪些此段比较重要，可以处理连续和种类字段。随机森林算法可以处理很高维度的数据，而不用做特征选择，训练可以并行化，训练速度比较快，实现简单。而SMO算法在优化两个变量时，要遍历整个训练样本集，而我们所用的样本数据本身就很庞大，就使得计算量比较大，运行时间较慢。

另外三种方法运行速度各不相同，主要有两个方面的因素：

1. 算法复杂度。CART算法计算复杂程度比SMO小，运行相对较快一些小。
2. 客观因素：电脑运行速度不同。

第四章 结论

通过本次作业，加深了我们对几种算法原理的理解，同时将书本上学到的知识用于实践当中。CART回归树，随机森林和SMO三种算法各有优点和缺点，这次，我们将三者同时用于处理汽车投保风险评估上，使用Python编程测试进行预测，比较和分析了三种算法的优劣处。

随机森通过在每个节点处随机选择特征进行分支，最小化了各棵分类树之间的相关性，提高了分类精度。因为每棵树的生长很快，所以随机森林的分类速度很快，并且很容易实现并行化。

SMO算法是支持矢量机学习的一种快速算法，其特点是不断的将原二次规划问题分解为只有两个变量的二次规划子问题，并对子问题进行解析求解，直到所有变量满足KKT条件为止。这样通过启发式的方法得到原二次规划问题的最优解。因为子问题有解析解，所有每次计算问题都很快，虽然子问题次数很多，但总体上还是高效的。

在这种情况下CART方法有良好的优越性，但是，并不是说在任何情况下CART方法都好。对于许多数据集,CART方法产生的树并不稳定。训练样本集的一点轻微改变都可能完全改变树的结构,这些特点存在于具有显著相关特征的数据集中。在CART中,问题就转换为在单个结点处存在几个分支,而这几个分支在减少子结点的所有复杂度方面几乎是等价的。从而一个特定的分支选择是比较随意的，但是它将导致更多可能不同的树。这种不稳定性意味着我们必须十分清楚由CART产生的树中特定特征的充分解释。另一方面,这一特点暗含着具有相似判别能力的不同树的有用性,它允许通过树的使用改变特征的选择。

三种算法都能有效的提高计算速率，但应对大数据集，计算速度还是有待改善。当然仿真过程中，也会遇到很多问题，但小组成员之间分工合作，互相交流，彼此帮助，最终克服重重困难，交给自己一份满意的答卷。