基于Boosting Tree的交通流量预测及可视分析

摘要：交通流预测是智能交通系统中一个热门的研究课题。预测方法主要有统计方法、贝叶斯网络、神经网络模型、混合ARIMA和人工神经网络。广义回归神经网络（GRNN）是用于交通流预测的一个有趣的模型，因为它可以预测性质的动态变化和非线性的数据，这是一般的交通流数据发现。在本研究中，Boosting Tree模型建立过程中的交通流数据，并比较其结果与其他预测方法的结果（ARIMA模型、单指数平滑法、移动平均）。留一交叉验证（LOOCV）是用于检测的交通流数据和平均绝对百分比误差（MAPE）作为测试的评价标准。结果表明，采用广义回归神经网络方法对试验数据进行处理可以提高预测的减少在三其它预测方法的预测值的准确性：有马，单指数平滑法、移动平均。

关键字：

1 引言

交通流数据是分析道路交通状况的主要因素之一。交通流是指在一段道路上通过某一特定点的车辆数。这种流量是用一段时间内的车辆数量来衡量的。短时交通流具有动态性和非线性性特征（1），因此交通流预测在交通管理中起着重要作用。利用神经网络对交通流进行预测（2）。提出了用统计方法预测交通流量的方法（3），将非线性和非平稳方法应用于短时交通流预测。[ 4 ]提出自适应有马预测交通数据。[ 5 ]预测交通数据的贝叶斯模型。[ 6 ] GRNN方法相比，径向基函数神经网络（RBFNN）和反向传播神经网络（BPNN）预测交通流数据。从三种方法，网络提供最高的精度。网络是由[ 7 ] Donald F Specht介绍，属于概率神经网络的分类。网络是一个简单的、稳定的人工神经网络适用于在自然[ 1 ]动态和非线性数据模型。广义回归神经网络的一个特点是在图案层神经元的数目会随着训练数据的增加[ 7 ]，然而太多的神经元会增加过[ 8 ]的机会增多。在本文中，网络是用来预测交通流数据。然后将结果与其他三种预测方法（ARIMA、单指数平滑和滑动平均）进行了比较。

2 数据处理

本文使用的数据是上海市15年的交通数据，其中包括地铁刷卡、公交车刷卡及出租车轨迹等数据。这里选取其中一个月的地铁刷卡数据。原始数据格式如下：

表 地铁原始数据格式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用户ID | 日期 | 时间 | 地名 | 乘车方式 | 费用 | 是否优惠 |
| 2601377141 | 2015-04-01 | 08:23:50 | 1号线莘庄 | 地铁 | 0.00 | 非优惠 |
| 2601377141 | 2015-04-01 | 19:18:30 | 1号线莘庄 | 地铁 | 4.00 | 非优惠 |

数据预处理流程：

（1）每20分钟分为一个时间段

（2）将日期转换为星期

（3）计算相同时间段的人流量

2 方法

（1）提升树模型

提升方法实际采用加法模型（即基函数的线性组合）与前向分步算法，以决策树为基函数的提升方法称为提升树（Boosting Tree）。提升树可以表示为决策树的加法模型：



其中，表示决策树；为决策树的参数；为树的个数。

提升树最基本的组成部分叫做分类与回归树（classification and regression tree，CART），CART会把输入根据输入的属性分配到各个叶子节点，而每个叶子节点上面都会对应一个实数分数。

提升树和决策树有所不同，可以将提升树理解为决策树的一个扩展。从简单的类标到分数之后，我们可以做很多工作，例如概率预测，排序等。

（2）提升树算法

（3）梯度提升

3 实验结果

4 总结

（一种基于贝叶斯超参数优化的提升决策树信用评分方法）

2.1 集合树和学习目标

已知给定数据集如下，

已知一个训练数据集，，为输入空间，，为输出空间。对于一个集合树模型，输出的表达式如下：



是给定第k次CART树时的预测结果。下图1描述了集合树模型。

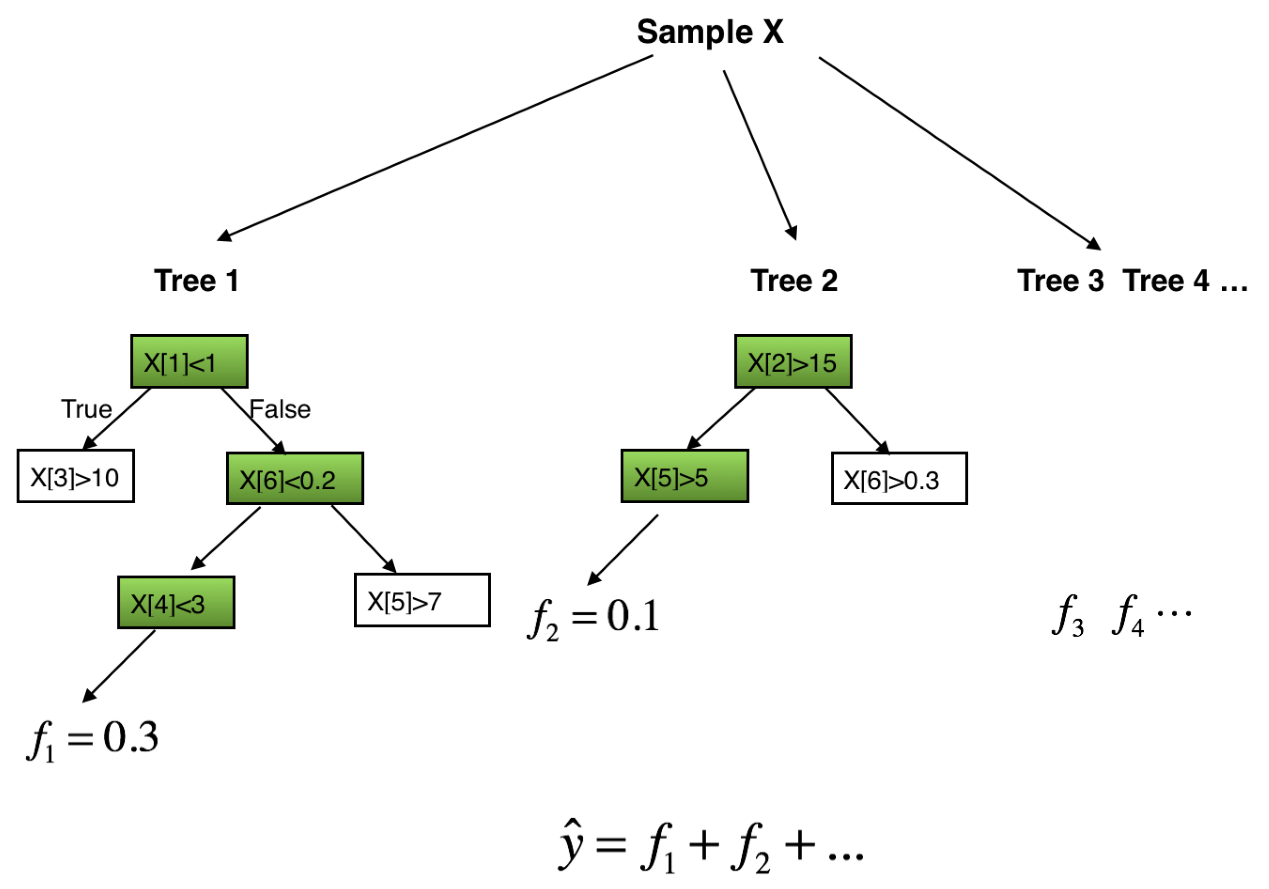


图1 集合树模型。

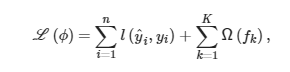
对于给定的xi，最终的预测结果Yˆ是每棵树的预测之和。

将单个CART表示为T，定义实例x对应树中相应的叶子：。根据公式1，对于给定的实例xi，第k颗CART的预测结果可以写成如下格式：



W标识CART中叶子结点的权重，q(xi)表示由树结构定义的映射函数。

树集合模型的学习目标可以被理解为求以下损失函数的最小值：



误差函数l(yi,yi)表示目标结果yi与公式1中预测结果yi的差异性。N表示实例的总数。K表示算法中CART的总数。这里为正则化项，用来惩罚复杂模型，避免模型过度拟合。可以表示为如下公式：

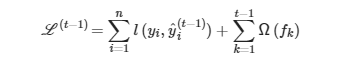




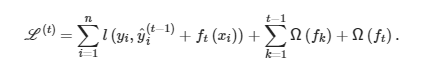
2.2 梯度下降提升树

训练模型的目的是为了最小化总体损失函数。然而传统的优化算法不能在欧式空间使目标函数最小化，因为公式3中的损失函数依赖参数，同样依赖每棵树的结构。为了高效地解决这个问题，梯度下降提升树算法被提出。

在训练模型中，第t-1次迭代的损失函数定义如下：

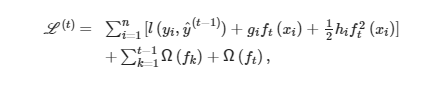


表示第t-1次迭代的预测结果。在下一次迭代也就是第t次迭代中加入ft，目标函数变成如下形式：

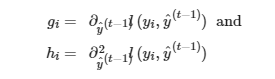


换句话说，根据公式3，贪婪地添加树来改善模型。所以，在第t次迭代，加入第t颗CART，目的是为了是最小。

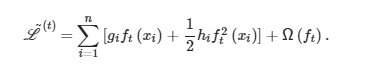
首先，用泰勒表达式展开目标函数：



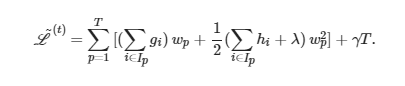
其中：



这个目标函数有一个非常明显的特点，它只依赖每个数据点在误差函数上的一阶导数和二阶导数。在第t次迭代，前一颗树t-1被改善，它的目标函数可以看成一个常量，因此，我们在上式中移除它，得到下面这个 简单的目标函数：



结合公式2和公式4，此表达式等价于下面这个表达式：

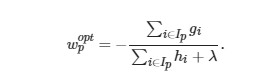


表示叶子p的数据实例集。所有的数据实例都被映射到叶子p上。

因此，对于一个固定的树结构q(x),叶子p的最优权重由下面的最小化方程定义：



由方程10定义，这个函数的结果如下：

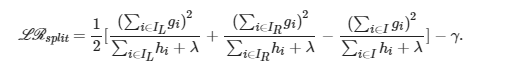


有了这个最优权重，得到如下的最优目标函数方程：



因此，对于每一次迭代，我们的目标变为寻找最优的树q是目标函数最小。

但是，通过枚举所有可能的树是不切实际的。相反，可以贪婪算法解决这个问题。从一颗简单树出发，我们可以迭代地分割树节点并向树中添加分支。对于每次迭代，在分割之前定义数据实例集为I，在分割之后定义数据集的左节点为IL，右节点定义为IR。因为，使用方程13，分解后的目标函数为：



因此，为了找到最优树，该算法通过选择最大化的分裂迭代地增加分支。