个人资料

帛逸TB

GBDT源码剖析

2012-12-04 22:22 8000人阅读 评论(2) 收藏 举报

分类： C and C++（20） 互联网（2） 算法学习（15）

如今，GBDT被广泛运用于互联网行业，他的原理与优点这里就不细说了，网上google一大把。但是，我自认为自己不是一个理论牛人，对GBDT的理论理解之后也做不到从理论举一反三得到更深入的结果。但是学习一个算法，务必要深入细致才能领会到这个算法的精髓。因此，在了解了足够的GBDT理论之后，就需要通过去阅读其源码来深入学习GBDT了。但是，网上有关这类资料甚少，因此，我不得不自己亲自抄刀，索性自己从头学习了一下GBDT源码。幸好，这个算法在机器学习领域中的其它算法还是非常简单的。这里将心得简单分享，欢迎指正。源码可以去GBDT源码下载。

首先，这里需要介绍一下程序中用到的结构体，具体的每一个结构体的内容这里就不再赘述了，源码里面都有。这里只再细说一下每个结构体的作用，当然一些重要的结构体会详细解释。

struct gbdt_model_t:GBDT模型的结构体，也就是最终我们训练得到的由很多棵决策树组成的模型。

```
typedef struct {
    int* nodestatus;    //!<
    int* depth;         //
    int* splitid;       //!<
    double* splitvalue; //!<
    int* ndstart;       //!< 节点对应于 Index 的开始位置
    int* ndcount;       //!< 节点内元素的个数
    double* ndavg;      //!< 节点内元素的均值
   //double* vpredict;
    int* lson;          //!< 左子树
    int* rson;          //!< 右子树
    int nodesize;       //!< 树的节点个数
}gbdt_tree_t;
```

struct gbdt_tree_t:当然就代表模型中的一棵树的各种信息了。为了后面能理解，这里需要详细解释一下这个结构体。splitid[k]保存该棵树的第k个结点分裂的feature下标，splitvalue[k]保存该棵树第k个结点的分裂值，nodestatus[k]代表该棵树的第k个结点的状态，如果为GBDT_INTERIOR，代表该结点已分裂，如果为GBDT_TOSPLIT，代表该结点需分裂，如果为GBDT_TERMINAL表示该结点不需再分裂，一般是由于该结点的样本数ndcount[k]少于等于一阈值gbdt_min_node_size；depth[ncur+1]代表左子树的深度，depth[ncur+2]表示右子树的深度，其中ncur的增长步长为2，表示每次+2都相关于跳过当前结点的左子树和右子树，到达下一个结点。ndstart[ncur+1]代表划分到左子树开始样本的下标，ndstart[ncur+2]代表划分到右子树开始样本的下标，其中到底这个下标是代表第几个样本是由index的一个结构保存。ndcount[ncur + 1]代表划分到左

让编辑状态下的UITable\ (6639)
objective-c delegate (5560)
ios中的代理与回调函数 (5456)
Perl SIG信号处理 (4612)
scrollView实现无限快速 (4005)
GBDT理解二三事 (2957)
Gradient Boost 算法流程 (2256)
谈谈分类算法的选择 (2160)

评论排行

GBDT（MART） 迭代决 (6)
GBDT理解二三事 (4)
objective-c delegate (2)
scrollView实现无限快速 (2)
GBDT源码剖析 (2)
String常用用法总结 (1)
谈谈分类算法的选择 (1)
pid match算法思想 (1)
（EM算法）The EM Alg (1)
Cookie、 Session、 Ca (0)

推荐文章

*机器学习与数据挖掘网上资源搜罗——良心推荐
*架构设计：系统间通信（17）——服务治理与Dubbo 中篇（分析）
*数据库性能优化之SQL语句优化
*Android应用开发allowBackup敏感信息泄露的一点反思
*Linux多线程实践（四 ）线程的特定数据
*Android点击Button水波纹效果

最新评论

GBDT（MART） 迭代决策树入门
atomlion: 非常感谢哦！我在实践中发现GBDT的预测效果非常好，在数据质量较差、复杂度高的大样本集中，几乎是效果...

GBDT（MART） 迭代决策树入门
大本_daben: @liufeng_cp:那到底怎么去理解boost呢？它的英文解释就是“推进、提升”的意思，我可以理...

GBDT理解二三事
march_on: 请问“损失函数可以定义为负的log似然”,这里损失函数为什么是负的log似然，这里有什么推导吗

GBDT理解二三事
qq_18247987: 你好，我想请问一下，文章末尾，说在节点分裂的时候，使用直方图采样去，优化效率，不用遍历所有特征值，想...

GBDT（MART） 迭代决策树入门
keepreder: logistic regression能用于非线性回归

谈谈分类算法的选择
easonlv: 图片标签都不能显示，麻烦楼主调整一下，谢谢

GBDT理解二三事
帛逸TB: @yangxudong:不行

GBDT理解二三事
yangxudong: GBDT能不能自动组合特征？

GBDT（MART） 迭代决策树入门
zhaonvsen: 对c4.5的理解有点出入：c4.5是取信息增益率最大的

子树的样本数量，ndcount[ncur + 2]代表划分到右子树的样本数量。ndavg[ncur+1]代表左子树样本的均值，同理是右子树样本的均值。nodestatus[ncur+1] = GBDT_TOSPLIT表示左子树可分裂。lson[k]=ncur+1表示第k个结点的左子树，同理表示第k个结点的右子树。

gbdt_info_t保存模型配置参数。

```
typedef struct
{
    int* fea_pool; //!< 随机 feature 候选池
    double* fvalue_list; //!< 以feature i 为拉链的特征值 x_i
    double* fv; //!< 特征值排序用的buffer版本
    double* y_list; //!< 回归的y值集合
    int* order_i; //!< 排序的标号
} bufset; //!< 训练数据池
```

bufset代表训练数据池，它保存了训练当前一棵树所用到的一些数据。fea_pool保存了训练数据的特征的下标，循环rand_fea_num（feature随机采样量）次，随机地从fea_pool中选取特征来计算分裂的损失函数（先过的feature不会再选）。fvalue_list保存在当前选择特征fid时，所有采样的样本特征fid对应的值。fv与favlue_list一样。y_list表示采样样本的y值。order_i保存左子树与右子树结点下标。

nodeinfo代表节点的信息。

```
typedef struct
{
    int bestid; //!< 分裂使用的Feature ID
    double bestsplit; //!< 分裂边界的x值
    int pivot; //!< 分裂边界的数据标号
} splitinfo; //!< 分裂的信息
```

splitinfo代表分裂的信息。pivot代表分裂点在order_i中的下标。bestsplit表示分裂值。bestid表示分裂的feature。

好了，解释完关键的一些结构体，下面要看懂整个gbdt的流程就非常简单了。这里我就简单的从头至尾叙述一下整个训练的流程。

首先申请分配模型空间gbdt_model，并且计算所有样本在每一维特征上的平均值。假如我们需要训练 _num棵树，每一棵的训练流程为：从x_fea_value中采样gbdt_inf.sample_r··· 个样本， index[i]记录了第i个结点所对应的样本集合x_fea_value中的下标，其始终保存着训练本棵树的所有采样样本对应样本空间的下标值，同时，结点的顺序是按该棵 按广度优先遍历算法遍历的结果的。即当前树gbdt_single_tree只有一个根结点0，其中gbdt_single_tree->nodestatus为GBDT_TOSPLIT，ndstart[0]=0，ndcount[0]=sample_num，ndavg为所有采样样本的y的梯度值均值。下面就是对这个结点进行分裂的过程：首先nodeinfo ninf这个结构体保存了当前分裂结点的一些信息，比如结点中样本开始的下标（指相对于index的下标值，index指向的值才是样本空间中该样本的下标），样本结束下标（同上），样本结点数，样本结点的y的梯度之和等。循环rand_fea_num次，随机采样feature，来计算在该feature分裂的信息增益,计算方式为（左子树样子目标值和的平方均值+右子树目标值和的平方均值-父结点所有样本和的平方均值）。选过的feature就不会再选中来计算信息增益了。利用data_set来保存当前分裂过程所用到的一些信息，包括候选feature池，选中feature对应的采样样本的特征值及其y值。data_set->order_i保存了左右子树对应结点在样本集合中的下标。计算每个feature的信息增益，并取最大的，保存分点信息到spinf中，包括最优分裂值，最优分裂feature。然后，将该结点小于分裂值的结点样本下标与大于分裂值的结点样本下标都保存在data_set->order_i中，nl记录了order_i中右子树开始的位置。更新index数组，将order_i中copy到index中。将nl更新到spinf中。注意index数组从左至右保存了最终分裂的左子树与右子树样本对应在样本空间的下标。

至此，我们找到了这个结点的最优分裂点。gbdt_single_tree->ndstart[1]保存了左孩子的开始下标（指相对于index的下标值，index指向的值才是样本下标），gbdt_single_tree->ndstart[2]保存了右孩子的开始下标，即nl的值。同理，ndcount，depth等也是对就保存了左右孩子信息。gbdt_single_tree->lson[0]=1，gbdt_single_tree->lson[0]=2即表示当前结点0的左子树是1，右子树是2。当前结点分裂

属性为分类属性，而不是熵。另外，c3.0采用的...

GBDT（MART） 迭代决策树入门
liufeng_cp: 多谢总结，赞，但Boost与迭代不是相同的概念，迭代只是boost的一种具体操作形式

完了之后，下一次就同理广度优先算法，对该结点的孩子继续上述步骤。

该棵树分裂完成之后，对每一个样本，都用目前模型（加上分裂完成的这棵树）计算预测值，并且更新每一个样本的残差y_gradient。计算过程：选取当前结点的分裂feature以及分裂值，小于则走左子树，大于则走右子树，直到叶子结点。预测值为shrink*该叶子结点的样本目标值的均值。

训练第二棵树同理，只是训练的样本的目标值变成了前面模型预测结果的残差了。这点就体现在梯度下降的寻优过程。

好了，这里只是简单的对gbdt代码做了说明，当然如果没有看过本文引用的源码，是不怎么能看懂的，如果结合源码来看，就很容易看懂了。总之，个人感觉，只有结合原码来学习gbdt，才真正能体会到事个模型的学习以及树的生成过程。

上一篇　GBDT（MART）迭代决策树入门教程 | 简介
下一篇　（EM算法）The EM Algorithm

我的同类文章

C and C++（20）　互联网（2）　算法学习（15）

- 变长数组和alloca
- 字符数组、字符指针和sizeof值得注意的地方。。
- C语言函数入栈顺序与可变参数函数
- C++ 虚函数表解析
- C++经典面试
- String常用用法总结
- RTTI 运行时类型识别
- 纯虚函数能为private吗？
- c++异常处理机制
- C++字符串函数

更多

主题推荐　源码

猜你在找

有趣的算法（数据结构）
数据结构和算法
数据结构基础系列(1)：数据结构和算法
数据结构基础系列(9)：排序
数据结构基础系列(5)：数组与广义表
STL源码剖析之哈希表 hashtable20131206
spark-080源码剖析-stage的建立--宽依赖和窄依赖
Chrome源码剖析上--多线程模型进程通信进程模型
Android屏幕分辨率获取方法--源码剖析
《STL 源码剖析读书笔记一》 - 迭代器概念与trais编


cms系统

美国计算机排名

美国大学排名表

量化投资

世界大学排

查看评论

2楼 AN_GZJ 2014-10-21 23:46发表

LZ能否把GBDT的源码发我一份学习一下，下载分太高了。251775119@qq.com感激不尽

1楼 poson 2014-01-16 09:35发表

选择分裂的节点不是用信息增益的吧。C4.5 才是信息增益。

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap