



# MLlib中的Random Forests和Boosting

发表于 2015-03-11 15:49 | 1246次阅读 | 来源 Databricks | 3 条评论 | 作者 Joseph Bradley, Manish Amde

机器学习 Spark 分布式 云计算

**摘要：**本文介绍了Random Forests和Gradient-Boosted Trees（GBTs）算法和他们在MLlib中的分布式实现，以及展示一些简单的例子并建议该从何处上手。

【编者按】本文来自Databricks公司网站的一篇博客文章，由Joseph Bradley和Manish Amde撰写。此外，Databricks是由Apache Spark的创始人建立的，成立于2013年年中，目前团队人员均是开源圈子内的重量级人物，他们都热衷于“增值开源软件”：

- 任职CEO的Ion Stoica是UC Berkeley计算机教授、AMPLab联合创始人，同时也是Conviva公司的联合创始人。
- CTO Matei Zaharia是Apache Spark的创作者，同时也是麻省理工学院计算机科学系的助理教授。
- UC Berkeley计算机科学教授Scott Shenker，同时也是知名SDN公司Nicira的联合创始人及前CEO。
- 值得一提的是联合创始人辛湜先生（英文名Reynold Xin，新浪微博为@hashjoin）还是一名中国人。

以下为博文的译文：

在Spark 1.2中，MLlib引入了Random Forests和Gradient-Boosted Trees（GBTs）。在分类和回归处理上，这两个算法久经验证，同时也是部署最广泛的两个方法。Random Forests和GBTs属于ensemble learning algorithms（集成学习算法），通过组合多个决策树来建立更为强大的模型。在本文，我们将介绍这两个模型和他们在MLlib中的分布式实现。同时，我们还会展示一些简单的例子并建议该从何处上手。

## Ensemble Methods

简言之，集成学习算法（Ensemble Learning Algorithms）是对已有的机器学习算法进行组合。组合后的模型将比原有的任意一个子模型更加的强大和精确。

在MLlib 1.2中，我们使用 Decision Trees（决策树）作为基础模型，同时还提供了两个集成方法：Random Forests与 Gradient-Boosted Trees（GBTs）。两个算法的主要区别在于各个部件树（component tree）的训练顺序。

在Random Forests中，各个部件树会使用数据的随机样本进行独立地训练。对比只使用单棵决策树，这种随机性可以帮助训练出一个更健壮模型，同时也能避免造成在训练数据上的过拟合。

GBTs一次训练一棵树，每次加入的新树用于纠正已训练的模型误差。因此，随着越来越多树被添加，模型变得越来越有表现力。

总而言之，两种方法都是多个决策树的加权集合。集成模型基于多个树给出的结果进行结合来做出预测。下图是建立在3个树之上的一个非常简单的例子。



**CSDN官方微信**  
扫描二维码,向CSDN吐槽  
微信号: CSDNnews



程序员移动端订阅下载

## 每日资讯快速浏览

### 微博关注

CSDN云计算 北京 朝阳区

加关注

【Spark 1.3更新概述：176个贡献者，1000+ patches】千呼万唤，合176个贡献者之力，Spark 1.3终于发布，其主要更新包括：2014年到2015年Spark最大的API改动DataFrame、内置支持Spark Packages、更好的Kafka支持，以及MLlib中的多个算法引入。http://t.cn/RwkDX0m

3月16日 17:15

转发(3) | 评论

### 相关热门文章

陈超：Spark这一年，从开源到火爆

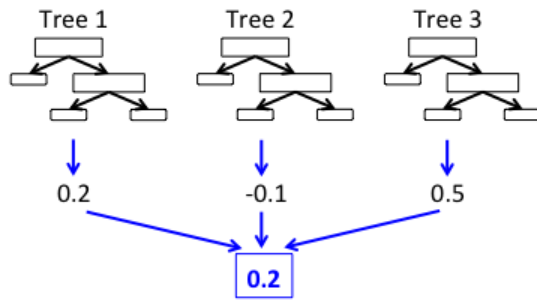
Databricks、Intel、BAT齐聚，2015 Spark峰会...

开源界迎来新杀手，用互联网思维攻克企业级市场

【机器人读报】Google云服务为Docker应用提...

“互联网+”升国家层面，互联网连接器背后的腾讯...

## Ensemble Model: example for regression



在上图的回归集成中，每棵树都会产生一个实数值，随后这3个值被整合以产生一个最终的结果。这里使用的是均值计算，当然你也可以根据预测任务来选择使用不同技术。

### Distributed Learning of Ensembles

在MLlib，不管是Random Forests还是GBTs都通过实例（行）对数据进行分割。其实现依赖于原始Decision Tree代码，对多个独立的树进行分布式训练，详情见之前发布的 [博文](#)。其中大量的优化方式基于Google的 [PLANET](#)项目——分布式环境中做基于树的集成学习的一个主要项目。

**Random Forests:** 鉴于Random Forests中每棵树都独立地进行训练，因此多个树的训练可以并行进行（同时，单个树上的训练也可以并行地执行）。MLlib就是这样做的：可变数量的子树并行地进行训练，而具体的数量则在内存限制的基础上进行迭代优化。

**GBTs:** 鉴于GBTs一次只能训练一棵树，只能实现单棵树级别的并行化。

在这里，我们看一下MLlib完成的两个关键优化：

- 内存：Random Forests中每棵树训练都使用了数据的不同子样本。取代显式复制数据，我们使用了一个TreePoint结构来节省内存，TreePoint存放了各个子样本集中每个实例的副本数量。
- 通信：Decision Trees在每个决策点上都会从所有的特征中选择进行训练，然而Random Forests通常在每个节点采用有限的随机选择的特征子集。MLlib在实现时采用了这个二次抽样的特性来减少通信：如果每个节点上训练的子特征集只占有所有特征集的三分之一，那么通信将会减少到三分之一。

更多详情可查看 [Ensembles Section in the MLlib Programming Guide](#)一文。

### 使用MLlib Ensembles

下面我们将展示如何使用MLlib做集成学习。下面这个Scala示例展示了如何读入一个数据集，将数据分割到训练和测试环境，学习一个模型，打印这个模型以及测试精确度。Java和Python实例可以参考MLlib Programming Guide (<http://spark.apache.org/docs/latest/mllib-ensembles.html>)。需要注意的是，GBTs当下还没有Python API，GBTs的Python API可能在Spark 1.3版本发布（通过 [Github PR 3951](#)）。

### Random Forest Example

```
import org.apache.spark.mllib.tree.RandomForest
import org.apache.spark.mllib.tree.configuration.Strategy
import org.apache.spark.mllib.util.MLUtils

// Load and parse the data file.
val data =
  MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
// Split data into training/test sets
val splits = data.randomSplit(Array(0.7, 0.3))
val (trainingData, testData) = (splits(0), splits(1))

// Train a RandomForest model.
```

【机器人读报】谷歌 百度 Facebook IBM，人工...

为何走上OpenStack不归路？Monty Taylor、Da...

Spark 1.3更新概述：176个贡献者，1000+ patc...

MLlib中的Random Forests和Boosting

2015 Container技术峰会！各路英豪齐聚紫禁城！

### 热门标签

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

### CSDN Share PPT下载



编写“高性能”Python  
代码



流程设计与重组提纲



Concurrency  
Control and  
Recovery



How the Compute  
rWorks

```
val treeStrategy = Strategy.defaultStrategy("Classification")
val numTrees = 3 // Use more in practice.
val featureSubsetStrategy = "auto" // Let the algorithm choose.
val model = RandomForest.trainClassifier(trainingData,
    treeStrategy, numTrees, featureSubsetStrategy, seed = 12345)

// Evaluate model on test instances and compute test error
val testErr = testData.map { point =>
    val prediction = model.predict(point.features)
    if (point.label == prediction) 1.0 else 0.0
}.mean()
println("Test Error = " + testErr)
println("Learned Random Forest:n" + model.toDebugString)
```

## Gradient-Boosted Trees Example

```
import org.apache.spark.mllib.tree.GradientBoostedTrees
import org.apache.spark.mllib.tree.configuration.BoostingStrategy
import org.apache.spark.mllib.util.MLUtils

// Load and parse the data file.
val data =
    MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
// Split data into training/test sets
val splits = data.randomSplit(Array(0.7, 0.3))
val (trainingData, testData) = (splits(0), splits(1))

// Train a GradientBoostedTrees model.
val boostingStrategy =
    BoostingStrategy.defaultParams("Classification")
boostingStrategy.numIterations = 3 // Note: Use more in practice
val model =
    GradientBoostedTrees.train(trainingData, boostingStrategy)

// Evaluate model on test instances and compute test error
val testErr = testData.map { point =>
    val prediction = model.predict(point.features)
    if (point.label == prediction) 1.0 else 0.0
}.mean()
println("Test Error = " + testErr)
println("Learned GBT model:n" + model.toDebugString)
```

## 可扩展性

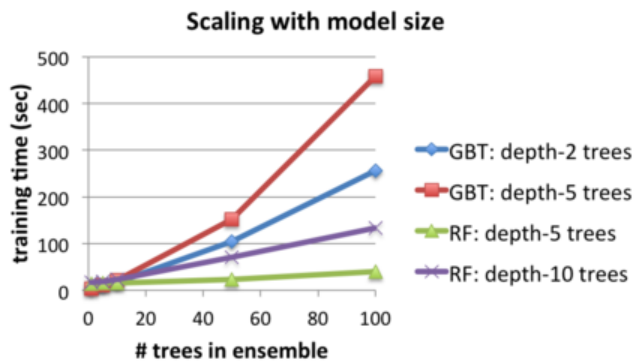
我们在一个二分类实验上展示了MLlib Ensembles的可扩展性。下面的每张图都对比了**Gradient-Boosted Trees ("GBT")** 和 **Random Forests ("RF")**，树根据最大深度区分。

测试的场景是一个根据音频特征集（UCI ML知识库中的**YearPredictionMSD**数据集）预测歌曲发布日期的回归任务，我们使用了EC2 r3.2xlarge主机。另外，除特殊说明，算法参数均选择默认。

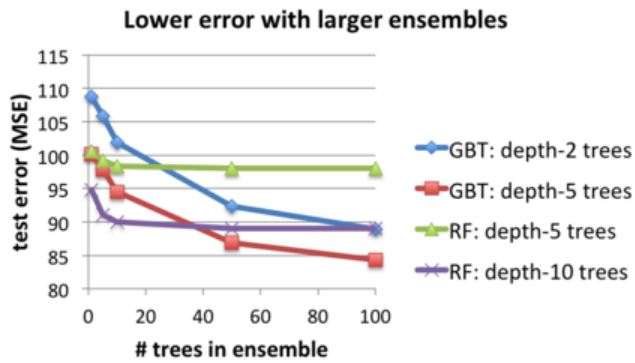
### 扩展模型体积：训练时间和测试错误

下文两张图片展示了在集成中增加树的数量时的效果。对于两个方法来说，增加树需要更多的学习时间，同样也意味着更好的结果（采用**Mean Squared Error (MSE)** 进行评估）。

对比两种方法，**Random Forests**训练的速度无疑更快，但是如果想达到同样的误差，它们往往需要深度更大的树。GBTs每次迭代都可以进一步减少误差，但是如果迭代次数太多，它很可能造成过拟合。



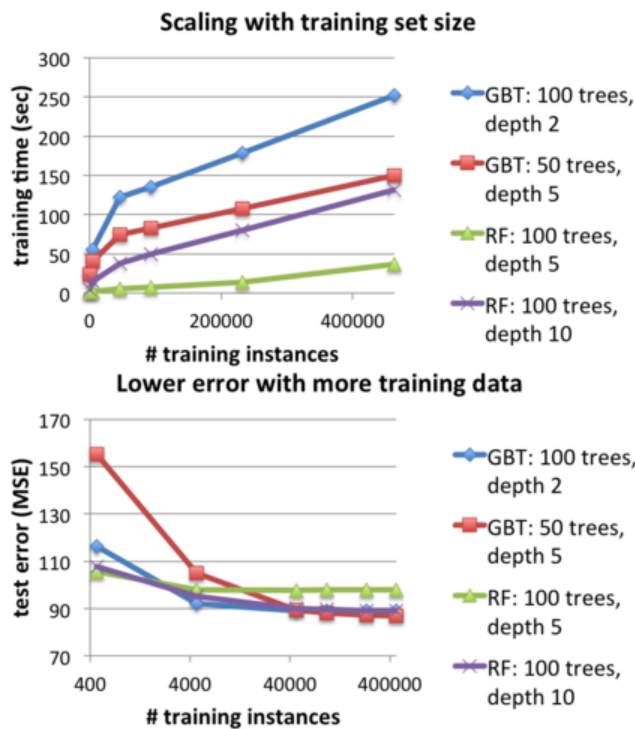
下图可以帮助对MSE产生一定程度的了解，最左边的点表示了使用单一决策树时产生的误差。



详情：463715个训练模型。16个从节点。

缩放训练数据集的体积：训练时间和测试误差。

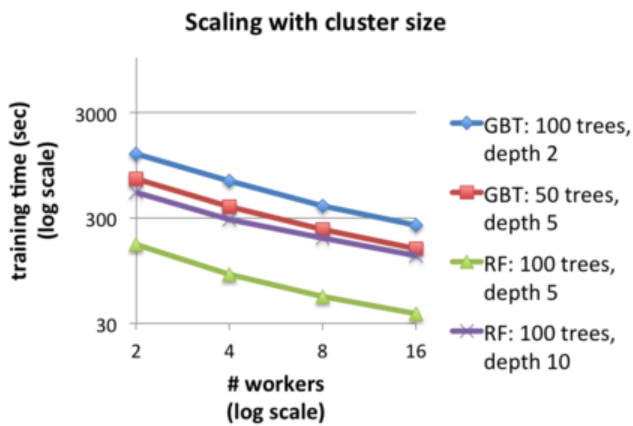
下面的两张图表示了在大训练数据集上的效果。使用更多的数据时，两个方法的训练时间都有所增长，但是显然也都得到了一个更好的结果。



详情：16个工作节点

强扩展性：使用更多的节点做更快速的训练

下面这张图体现了使用更大的计算集群来处理同样的问题。显然，在使用了更多的从节点后，两种方法的训练速度都得到了显著提升。举个例子，使用GBTs时，在树深度为2的情况下，16个工作者节点的速度将是2个工作者节点速度的4.7倍。同时，数据集越大，速度的提升越明显。



详情：463715个训练实例。

## 下一步

GBTs的Python API将在不久后实现。后续开发的重点是可插拔性：集成可以应用到几乎所有分类或者回归算法，不仅仅是决策树。对于这一点，Spark 1.2中引入的 Pipelines API 支持对集成算法进行扩展，实现真正的可插拔。

原文链接：[Random Forests and Boosting in MLlib](#)（译者/童阳 友情审校/王静，浙江大学博士）

**OpenCloud 2015**将于2015年 4月16-18日在北京召开。大会包含“2015 OpenStack技术大会”、“2015 Spark技术峰会”、“2015 Container技术峰会”三大技术峰会及多场深度行业实战培训，主题聚焦技术创新与应用实践，荟萃国内外真正的云计算技术的大牛讲师。这里都是一线接地气的干货，扎实的产品、技术、服务和平台。**OpenCloud 2015**，懂行的人都在这里！

更多讲师和日程信息请关注**OpenCloud 2015**[介绍](#)和[官网](#)。

本文为CSDN编译整理，未经允许不得转载，如需转载请联系market#csdn.net(#换成@)

顶

4

踩

0



推荐阅读相关主题：[random](#) [麻省理工学院](#) [计算机科学](#) [浙江大学](#) [开源软件](#) [新浪微博](#)

## 相关文章

## 最新报道

【快讯】阿里开放5.7亿条脱敏数据，90后“Marvel”...  
一周热点：从打造国内最大的OpenStack公有云开始  
“天网”降临 机器人或将崛起？  
收藏！斯坦福Andrew Ng教授“机器学习”26篇教程全译  
【机器人读报】Slack：日活跃用户50万人、6周增...  
Hadoop进军机器学习：Cloudera收购Myrixx共创“Bi...

已有3条评论

还可以再输入500个字



有什么感想，你也来说说吧！

luchunminglu 欢迎您！

最新评论

最热评论



baidu\_26579881 2015-03-15 00:40

ok

回复



考拉kaola 2015-03-13 12:37

说的好

回复



codemosi 2015-03-13 09:27

一线干货，赞一个，spark发展得越来越好，希望早点壮大成熟。

回复

共1页

首页

上一页

1

下一页

末页

请您注意

- 自觉遵守：爱国、守法、自律、真实、文明的原则
- 尊重网上道德，遵守《全国人大常委会关于维护互联网安全的决定》及中华人民共和国其他各项有关法律法规
- 严禁发表危害国家安全，破坏民族团结、国家宗教政策和社会稳定，含侮辱、诽谤、教唆、淫秽等内容的作品
- 承担一切因您的行为而直接或间接导致的民事或刑事责任
- 您在CSDN新闻评论发表的作品，CSDN有权在网站内保留、转载、引用或者删除
- 参与本评论即表明您已经阅读并接受上述条款

热门专区



容联云通讯开发者技术专区



腾讯云技术社区



IBM新兴技术大学



高效能团队解决方案



高通开发者专区

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服

杂志客服

微博客服

webmaster@csdn.net

400-600-2320

| 北京创新乐知信息技术有限公司 版权所有 |

江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 070598 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved