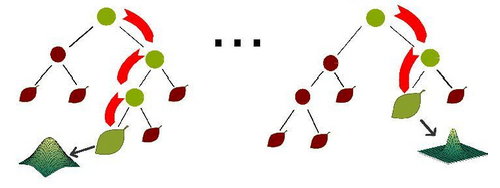
Spark mllib 分析报告-Random Forest

**一：算法介绍**

多个决策树构成的森林，算法分类结果由这些决策树投票得到，决策树在生成的过程当中分别在行方向和列方向上添加随机过程，行方向上构建决策树时采用放回抽样（bootstraping）得到训练数据，列方向上采用无放回随机抽样得到特征子集，并据此得到其最优切分点，这便是随机森林算法的基本原理。图 3 给出了随机森林算法分类原理，从图中可以看到，随机森林是一个组合模型，内部仍然是基于决策树，同单一的决策树分类不同的是，随机森林通过多个决策树投票结果进行分类，算法不容易出现过度拟合问题。

随机森林示意图



1. **单机模式算法过程：**

**对于每棵树，它们使用的训练集是从总的训练集中有放回采样出来的，这意味着，总的训练集中的有些样本可能多次出现在一棵树的训练集中，也可能从未出现在一棵树的训练集中。在训练每棵树的节点时，使用的特征是从所有特征中按照一定比例随机地无放回的抽取的，根据Leo Breiman的建议，假设总的特征数量为M，这个比例可以是sqrt(M),1/2sqrt(M),2sqrt(M)。**

* **随机森林的训练过程可以总结如下：**

(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i棵树，i ϵ {1-n}：

(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。有关分类效果的评判标准在后面会讲。

(4)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。

(5)重复(2),(3),(4)直到所有CART都被训练过。

* **利用随机森林的预测过程如下：**

对于第i棵树，i ϵ {1-n}：

(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。

* **分类效果的评判标准：**

因为使用的是CART，因此使用的也是CART的评判标准，和C3.0,C4.5都不相同。

**对于分类问题（将某个样本划分到某一类）**

也就是离散变量问题，CART使用Gini index值作为评判标准:

定义为Gini=1-∑(P(i)\*P(i)),P(i)为当前节点上数据集中第i类样本的比例。

例如：分为2类，当前节点上有100个样本，属于第一类的样本有70个，属于第二类的样本有30个，则Gini=1-0.7×07-0.3×03=0.42，可以看出，类别分布越平均，Gini值越大，类分布越不均匀，Gini值越小。

在寻找最佳的分类特征和阈值时，评判标准为：argmax（Gini-GiniLeft-GiniRight），即寻找最佳的特征f和阈值th，使得当前节点的Gini值减去左子节点的Gini和右子节点的Gini值最大。

**对于回归问题**

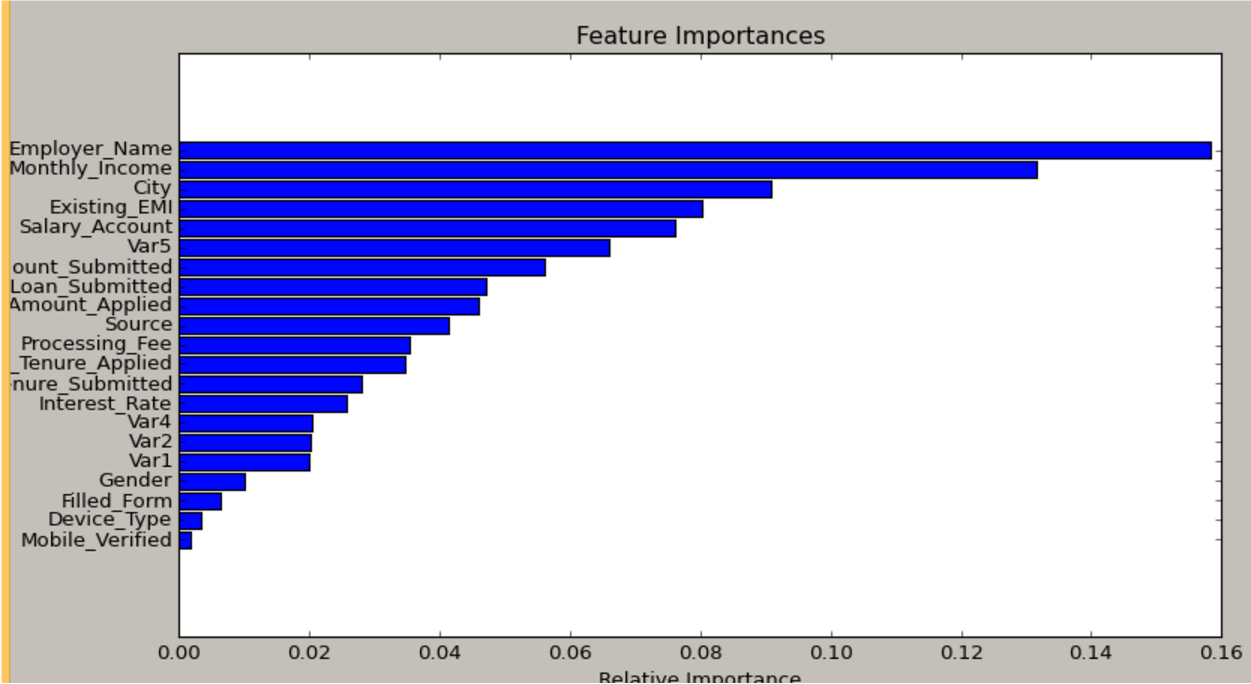
相对更加简单，直接使用argmax(Var-VarLeft-VarRight)作为评判标准，即当前节点训练集的方差Var减去减去左子节点的方差VarLeft和右子节点的方差VarRight值最大。

* + **算法优缺点：**

**优点：**

**1.正如上文所述，随机森林算法能解决分类与回归两种类型的问题，并在这两个方面都有相当好的估计表现；**

**2.随机森林对于高维数据集的处理能力令人兴奋，它可以处理成千上万的输入变量，并确定最重要的变量，因此被认为是一个不错的降维方法。此外，该模型能够输出变量的重要性程度，这是一个非常便利的功能。下图展示了随机森林对于变量重要性程度的输出形式：**



**3.在对缺失数据进行估计时，随机森林是一个十分有效的方法。就算存在大量的数据缺失，随机森林也能较好地保持精确性；**

**4.当存在分类不平衡的情况时，随机森林能够提供平衡数据集误差的有效方法；**

**5.模型的上述性能可以被扩展运用到未标记的数据集中，用于引导无监督聚类、数据透视和异常检测；**

**6.随机森林算法中包含了对输入数据的重复自抽样过程，即所谓的bootstrap抽样。这样一来，数据集中大约三分之一将没有用于模型的训练而是用于测试，这样的数据被称为out of bag samples，通过这些样本估计的误差被称为out of bag error。研究表明，这种out of bag方法的与测试集规模同训练集一致的估计方法有着相同的精确程度，因此在随机森林中我们无需再对测试集进行另外的设置。**

**缺点：**

**1.随机森林在解决回归问题时并没有像它在分类中表现的那么好，这是因为它并不能给出一个连续型的输出。当进行回归时，随机森林不能够作出超越训练集数据范围的预测，这可能导致在对某些还有特定噪声的数据进行建模时出现过度拟合。**

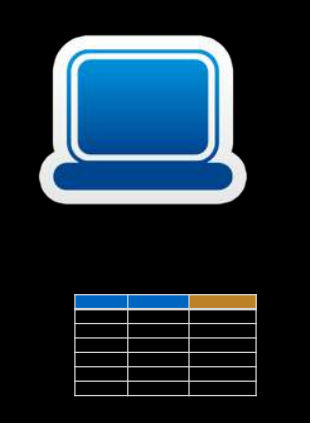
**2.对于许多统计建模者来说，随机森林给人的感觉像是一个黑盒子——你几乎无法控制模型内部的运行，只能在不同的参数和随机种子之间进行尝试。**

**（二）分布式环境存在的问题：**

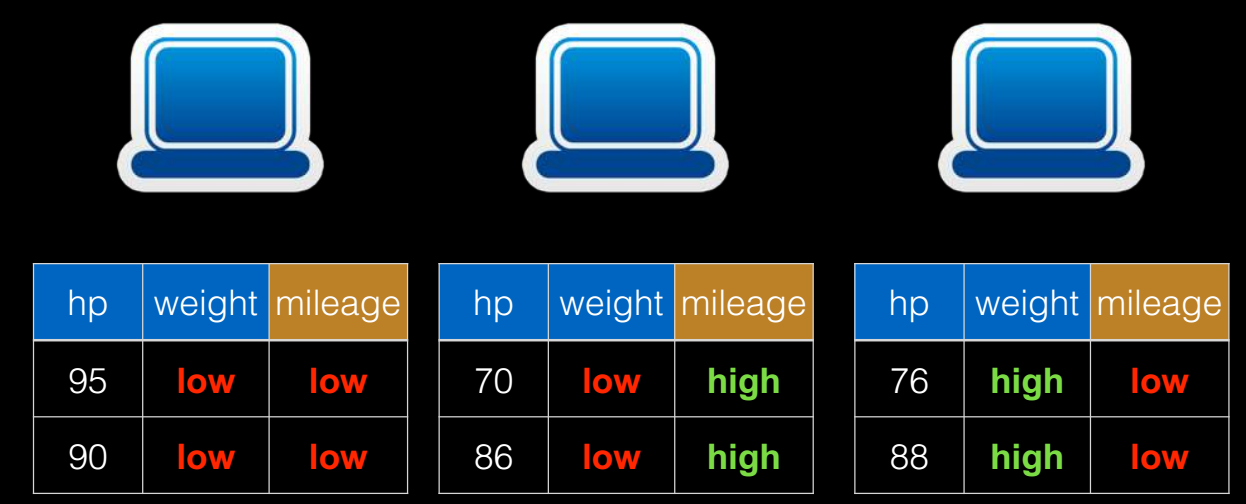
随机森林在分布式环境下的优化策略

随机森林算法在单机环境下很容易实现，但在分布式环境下特别是在 Spark 平台上，传统单机形式的迭代方式必须要进行相应改进才能适用于分布式环境，这是因为在分布式环境下，数据也是分布式的（如图 5 所示），算法设计不得当会生成大量的 IO 操作，例如频繁的网络数据传输，从而影响算法效率。

单机环境下数据存储



分布式环境下数据存储



因此，在 Spark 上进行随机森林算法的实现，需要进行一定的优化，Spark 中的随机森林算法主要实现了三个优化策略：

1. **切分点抽样统计**

如图 6 所示。在单机环境下的决策树对连续变量进行切分点选择时，一般是通过对特征点进行排序，然后取相邻两个数之间的点作为切分点，这在单机环境下是可行的，但如果在分布式环境下如此操作的话，会带来大量的网络传输操作，特别是当数据量达到 PB 级时，算法效率将极为低下。为避免该问题，Spark 中的随机森林在构建决策树时，会对各分区采用一定的子特征策略进行抽样，然后生成各个分区的统计数据，并最终得到切分点。

1. **特征装箱（Binning）**

如图 7 所示。决策树的构建过程就是对特征的取值不断进行划分的过程，对于离散的特征，如果有 M 个值，最多IMG_259个划分，如果值是有序的，那么就最多 M-1 个划分。比如年龄特征，有老，中，少 3 个值，如果无序有IMG_260个，即 3 种划分：老|中，少；老，中|少；老，少|中；如果是有序的，即按老，中，少的序，那么只有 m-1 个，即 2 种划分，老|中，少；老，中|少。对于连续的特征，其实就是进行范围划分，而划分的点就是 split（切分点），划分出的区间就是 bin。对于连续特征，理论上 split 是无数的，在分布环境下不可能取出所有的值，因此它采用的是（1）中的切点抽样统计方法。

1. **逐层训练（level-wise training）**

如图 8 所示。单机版本的决策树生成过程是通过递归调用（本质上是深度优先）的方式构造树，在构造树的同时，需要移动数据，将同一个子节点的数据移动到一起。此方法在分布式数据结构上无法有效的执行，而且也无法执行，因为数据太大，无法放在一起，所以在分布式环境下采用的策略是逐层构建树节点（本质上是广度优先），这样遍历所有数据的次数等于所有树中的最大层数。每次遍历时，只需要计算每个节点所有切分点统计参数，遍历完后，根据节点的特征划分，决定是否切分，以及如何切分。

切分点抽样统计

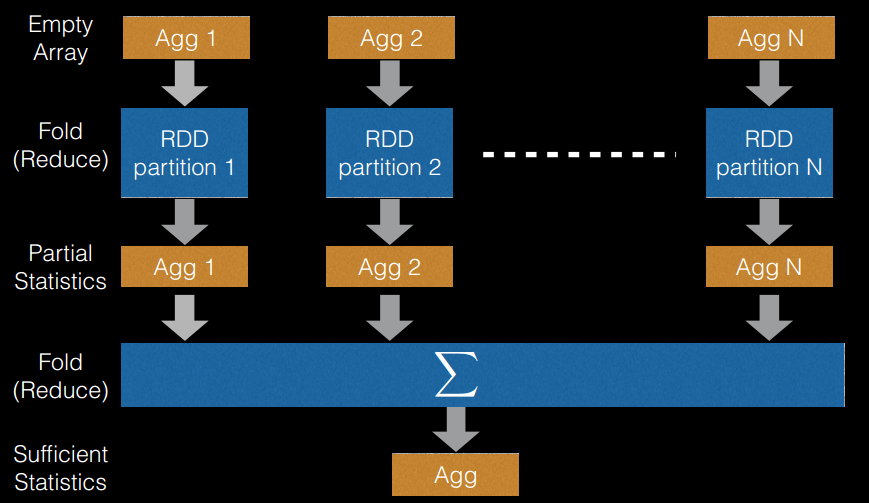


图 7. 特征装箱

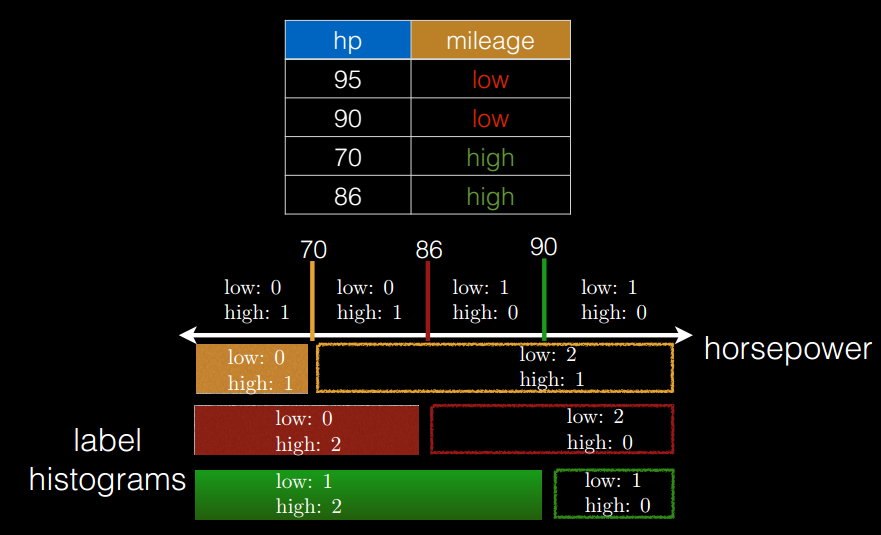
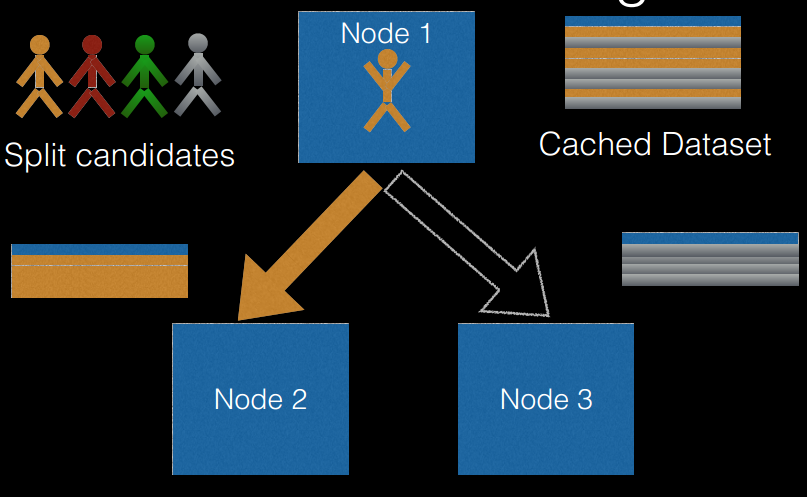


图 8. 逐层训练



**二：算法过程**

**三：数据流图：**

RDD



说明:

1:RDD[LabeledPoint]

2:TreeRDD[TreePoint]

*\*Binned feature values.*

*\*Same length as LabeldPoint.features, but values are bin indices.*

3: BaggedRDD

\**This holds one instance, as well as an array of weights which represent the (weighted)  
\* number of times which this instance appears in each subsamplingRate.  
\* E.g., (datum, [1, 0, 4]) indicates that there are 3 subsamples of the dataset and that  
\* this datum has 1 copy, 0 copies, and 4 copies in the 3 subsamples, respectively.*

Cache

If(strategy.useNodeIdCache == True):

Create an RDD of of node Id cache.

**Code:**

