LightGBM算法研究

kimmyzhang@tencent.com

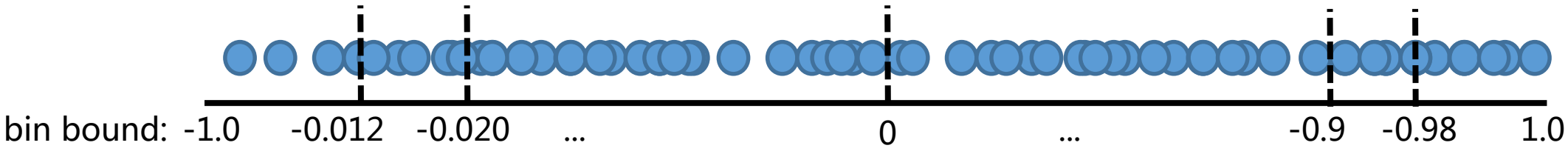
大纲

- 算法创新
  - bin & histogram
  - leaf-wise split
  - 分布式训练方法(communication efficient parallel voting)
  - DART(Dropout + GBDT)
  - GOSS(Gradient-based One-Side Sampling)
- 优点
- 缺点

Bin & Histogram

- xgboost等传统实现: pre-sort + 精确查找/分位点近似查找
  - 找分裂点时速度慢
  - 精确
- LightGBM: bin & histogram近似查找
  - 找分裂点时速度贼快
  - 不那么精确
  - 实际效果不差

Bin Mapper的构造



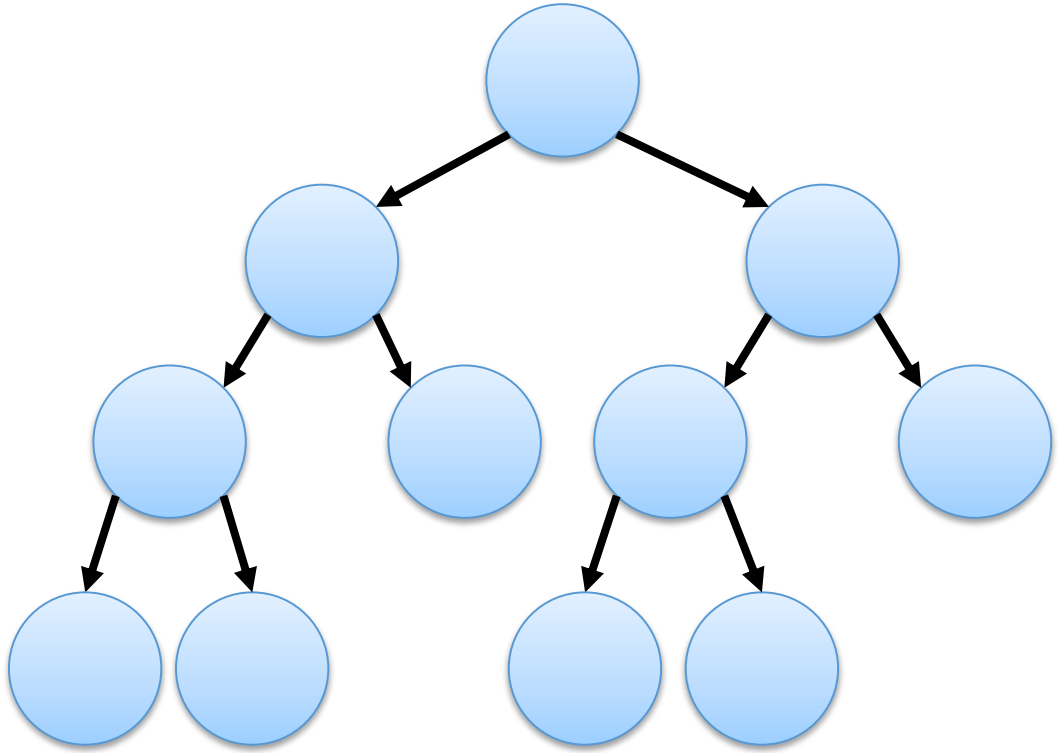
- 代码: DatasetLoader::ConstructBinMappersFromTextData

Bin Mapper的使用

- 读取instance时, feature value被映射为bin index
- 内存中保存bin mapper和bin中的所有instance index
- 所有的bin bound(除了最小和最大)将作为split candidate

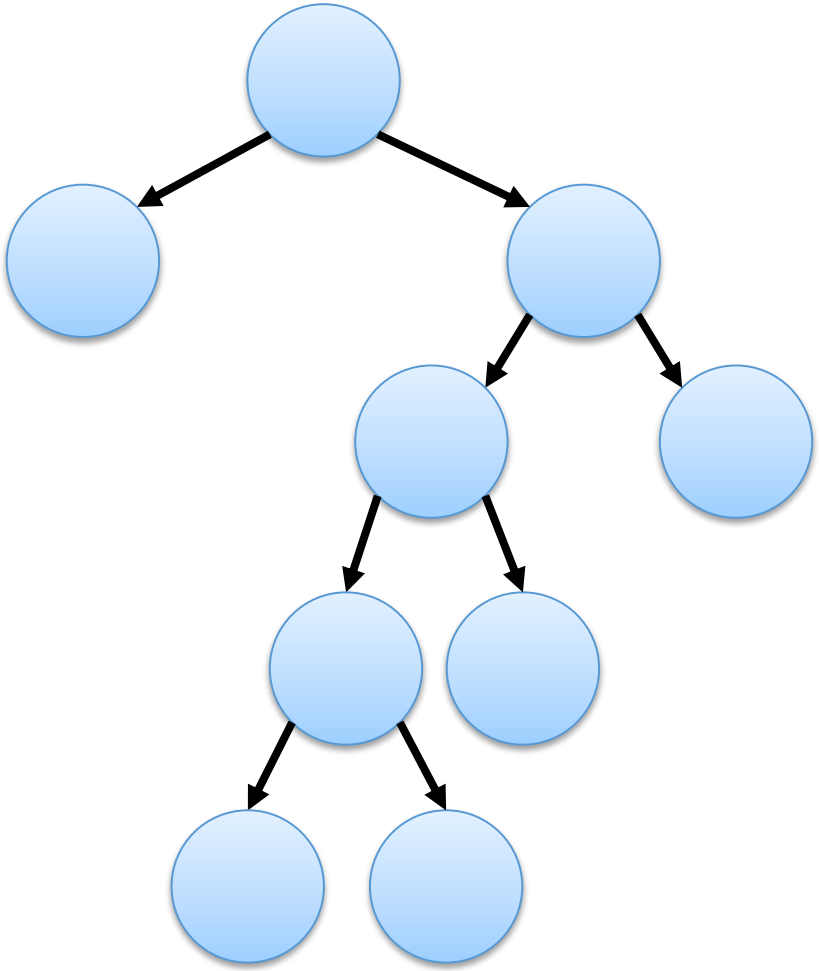
- 给听众的问题:
- 1. 如果某些特征值不在bin mappers的范围内如何处理?
- 2. 分布式情况下, 各个机器如何构造全局一致的bin mappers?

传统的Layer-wise Split



一层一层往下推进

LightGBM的Leaf-wise Split



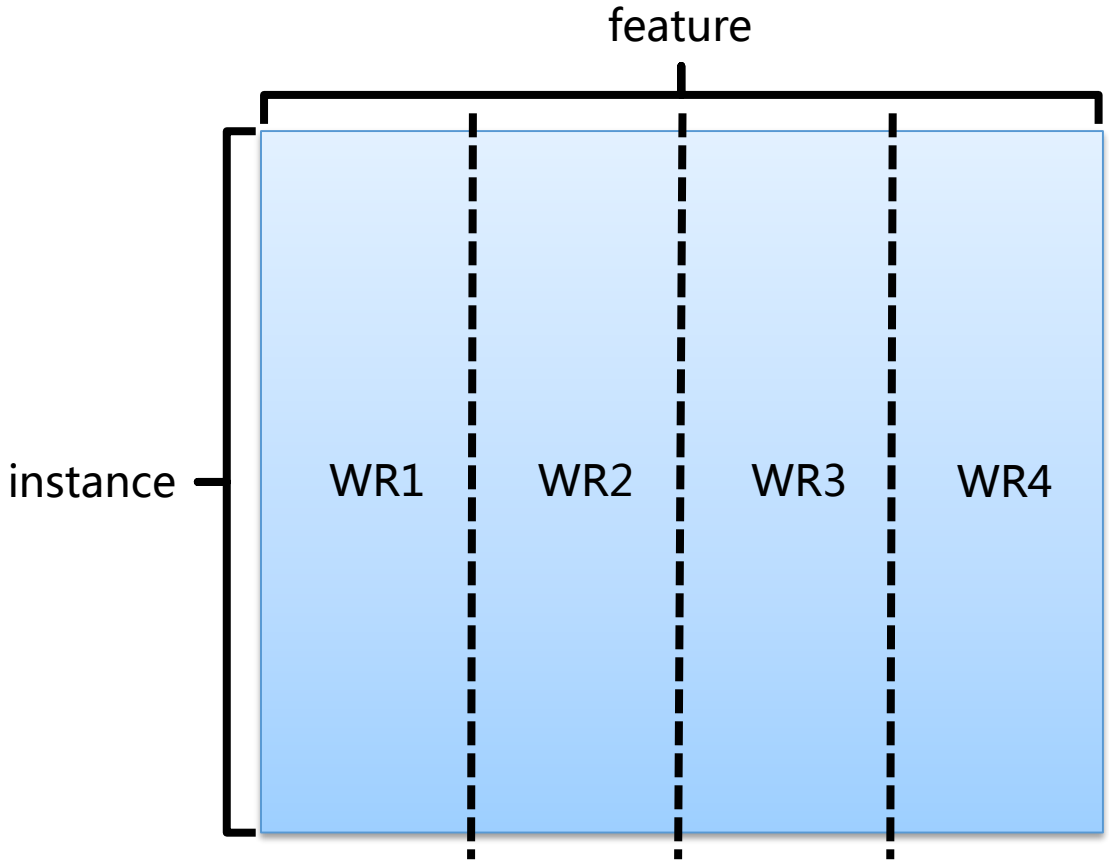
- 代码:
  - GBDT::TrainOneIter
  - SerialTreeLearner::Train
- 给听众的问题:
  - 每次分裂都要计算所有叶子节点的增益吗?

完全以增益为导向分裂
有更好的效果

分布式训练方法

- 传统的Attribute(Feature) Parallel
  - 通信量高
  - 精确
- LightGBM的Parallel Voting
  - 大部分计算在单个节点内, 通信量低
  - 不那么精确
  - 实际效果不差

传统的Feature Parallel



- 分布式分裂一个叶子节点的流程
  - WR并行找到local BSP(best split plan)
  - CD聚合并找到global BSP
  - 找到global BSP的worker进行instance切分, 将切分后的instance indices广播给所有WR

LightGBM的Parallel Voting

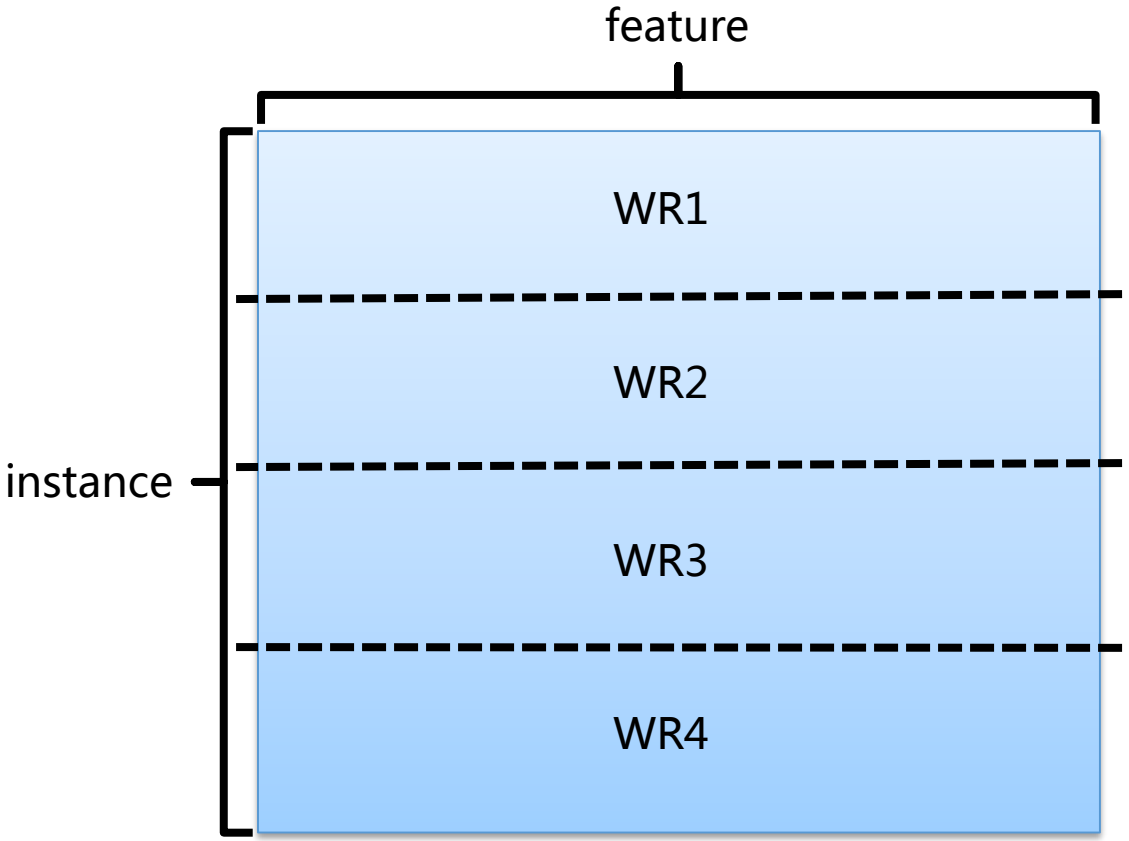
Algorithm 2 FindBestSplit

```
Input: DataSet D
for all X in D.Attribute do
    ▷ Construct Histogram
    H = new Histogram()
    for all x in X do
        H.binAt(x.bin).Put(x.label)
    end for
    ▷ Find Best Split
    leftSum = new HistogramSum()
    for all bin in H do
        leftSum = leftSum + H.binAt(bin)
        rightSum = H.AllSum - leftSum
        split.gain = CalSplitGain(leftSum, rightSum)
        bestSplit = ChoiceBetterOne(split,bestSplit)
    end for
end for
return bestSplit
```

Algorithm 3 PV-Tree_FindBestSplit

```
Input: Dataset D
localHistograms = ConstructHistograms(D)
▷ Local Voting
splits = []
for all H in localHistograms do
    splits.Push(H.FindBestSplit())
end for
localTop = splits.TopKByGain(K)
▷ Gather all candidates
allCandidates = AllGather(localTop)
▷ Global Voting
globalTop = allCandidates.TopKByMajority(2*K)
▷ Merge global histograms
globalHistograms = Gather(globalTop, localHis-
tograms)
bestSplit = globalHistograms.FindBestSplit()
return bestSplit
```

LightGBM的Parallel Voting



- 分布式分裂一个叶子节点的流程
  - WR并行找到k个local BSP
  - CD聚合保留top 2k个BSP
  - CD向所有WR收集这top 2k个BSP的histogram data
  - CD计算这top 2k个BSP的global 增益, 找到global BSP
  - CD将global BSP广播给所有WR
  - WR基于bin mapper做本地数据切分

Histogram Data与Global增益

$$Gain = 2\mathcal{L}(\mathbf{b}^*, \mathbf{R}) - 2\mathcal{L}(\mathbf{b}^{*\prime}, \mathbf{R}')$$

$$= \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

- 对于一个BSP, histogram data除了包含一些元信息外, 就是上面G_L, G_R, H_L, H_R.
- CD将收集到所有histogram data后, 可计算出global增益
- 核心代码: VotingParallelTreeLearner类

DART

随机丢掉一些树

修正丢弃掉树的预测值

Algorithm 1 The DART algorithm

Let N be the total number of trees to be added to the ensemble
$S_1 \leftarrow \{x, -L'_x(0)\}$
T_1 be a tree trained on the dataset S_1
$M \leftarrow \{T_1\}$
for $t = 2, \ldots, N$ **do**
  $D \leftarrow$ the subset of M such that $T \in M$ is in D with probability p_{drop}
  if $D = \emptyset$ **then** $D \leftarrow$ a random element from M
  end if
  $\hat{M} \leftarrow M \setminus D$
  $S_t \leftarrow \left\{x, -L'_x\left(\hat{M}(x)\right)\right\}$
  T_t be a tree trained on the dataset S_t
  $M \leftarrow M \cup \left\{\frac{T_t}{|D|+1}\right\}$
  for $T \in D$ **do**
    Multiply T in M by a factor of $\frac{|D|}{|D|+1}$
  end for
end for
Output M

DART效果

测试集square loss

Ensemble size	25	50	100	250	500	1000
MART	35.13	31.79	30.92	30.07	29.76	29.28
DART	**32.50**	**30.50**	**29.66**	**28.14**	**28.11**	**27.98**
Random Forest	32.76	33.21	32.88	32.36	32.66	32.33

测试集AUC

Ensemble size	50	100	250	500	1000
MART	**0.9687**	**0.9699**	0.9707	**0.9704**	0.9695
DART	0.9676	0.9692	**0.9714***	0.9693	**0.9699**
Random Forest	0.9627	0.9629	0.9629	0.9630	0.9628

代码: DART::TrainOneIter

GOSS

- Gradient-based One-Side Sampling
  - 一种新的Bagging(row subsample)方法
  - 前若干轮(1.0f / gbdt_config_->learning_rate)不Bagging
  - 之后Bagging时, 采样一定比例g(梯度)大的样本
- 直观解释: g越大, 降低loss的能力越大
- 代码: GOSS::Bagging

优点

- 目前最先进的GBDT工具包
- 支持分布式
- 速度快
- 效果好
- 省内存
- openmp和MPI使用的淋漓尽致

缺点

- 分布式没有fault tolerance
- typo非常非常非常多
- c++写的很junior
  - 作者很多best practice都不会
- 代码构架差
  - 部分代码结构混乱, 函数有些长, 看起来费劲

参考文献

- https://github.com/Microsoft/LightGBM
- Qi Meng, et al., A Communication-Efficient Parallel Algorithm for Decision Tree, NIPS, 2016.
- K. V. Rashmi, et al., DART: Dropouts meet Multiple Additive Regression Trees, arXiv, 2015.

谢谢聆听