4.1 LightGBM

CSDN学院

2017年11月

大纲

- LightGBM优化实现
- 独立调用LightGBM
- Scikit learn框架下调用LightGBM

LightGBM
(Light Gradient Boosting Machine)

- LightGBM 是Microsoft开发的一个GBDT算法框架，支持高效率的并行训练，并且具有以下优点：
  - 更快的训练速度
  - 更低的内存消耗
  - 更好的准确率
  - 分布式支持，可以快速处理海量数据

https://github.com/Microsoft/LightGBM

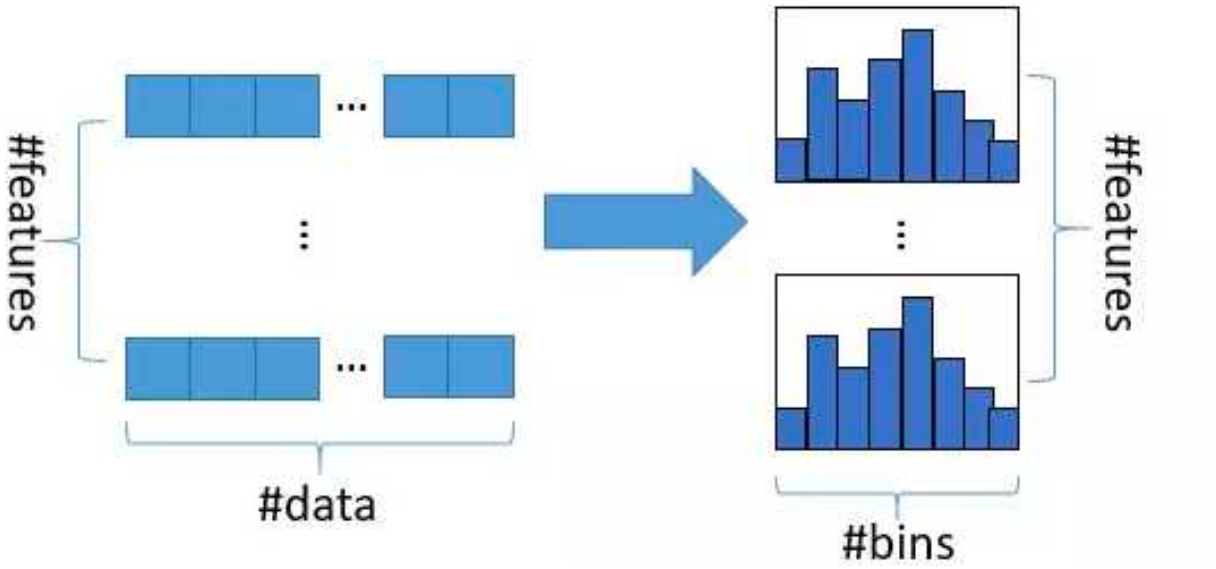
```
conda install -c conda-forge lightgbm
```

LightGBM的优化

- 基于Histogram的决策树算法
- 带深度限制的Leaf-wise的叶子生长策略
- 直方图做差加速
- 直接支持类别特征(Categorical Feature)
- Cache命中率优化
- 基于直方图的稀疏特征优化
- 多线程优化

基于Histogram的决策树算法

- 直方图算法的基本思想：
  - 计算直方图
  - 遍历数据：根据特征所在的bin值作为索引，遍历寻找最优的分割点



相当于数值特征离散化 / 分组：一个bin为一组

histogram 算法的计存储优势

N：样本数目
D：特征维数

- 相比于pre-sorted（如 xgboost 中的 exact 算法），histogram 在内存消耗和计算代价上优势明显：
  - Pre-sorted 算法：需要的内存约是训练数据的两倍(2 *N** *D** 4Bytes)，其中*N*为样本数目，*D*为特征维数。它需要用32位浮点来保存 特征值，并且对每一列特征，需要一个额外的排好序的索引（需要32位的存储空间）。
  - histogram 算法：只需要*N** *D** 1Bytes) 的内存，仅为 pre-sorted算法的1/8。因为 histogram 算法仅需要存储 特征离散化后的数值，不需要原始的特征值，也不用排序，而 bin索引用 8个bit(256个bin)一般也足够了。

histogram 算法的计算优势

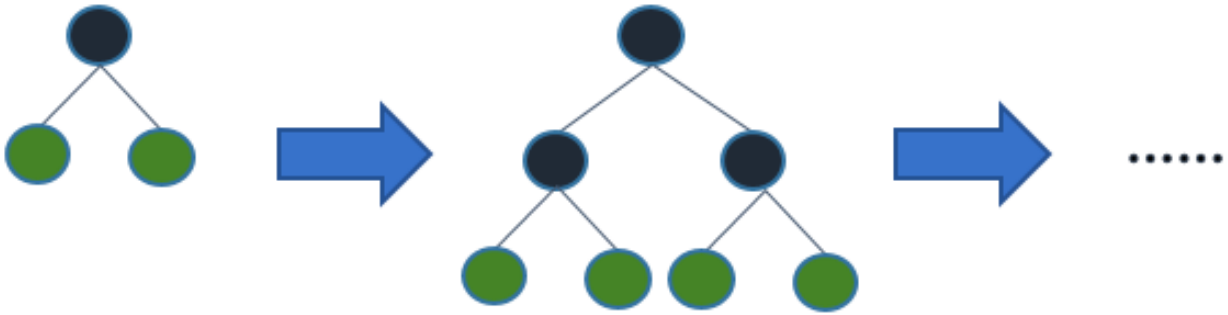
- histogram 算法可大幅减少计算分割点增益的次数
  - 对于一个特征，pre-sorted 需要对每一个不同特征值都计算一次分割增益，而 histogram 只需要计算 #bin 次。

histogram 算法的缺点

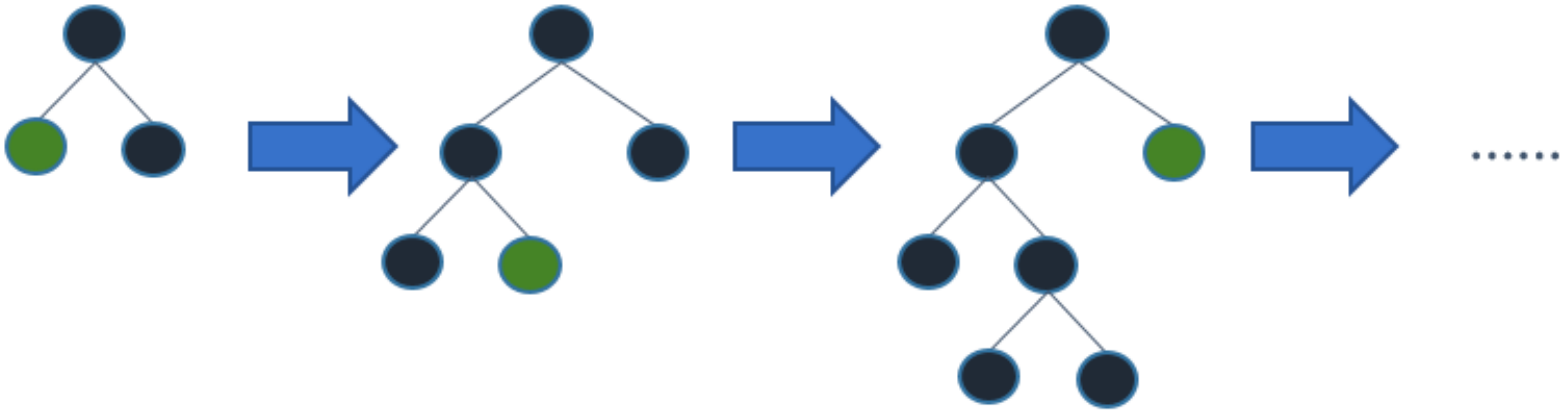
- histogram 算法不能找到精确的分割点，训练误差没有 pre-sorted 好。
- 但从实验结果来看， histogram 算法在测试集的误差和 pre-sorted 算法差异并不是很大，甚至有时候效果更好。
- 实际上可能决策树对于分割点的精确程度并不太敏感，而且较“粗”的分割点也自带正则化的效果。

带有深度限制的按叶子生长 (leaf-wise) 算法

- level-wise过一次数据可以同时分裂同一层的叶子，容易进行多线程优化，不容易过拟合。
- 但level-wise不加区分的对待同一层的叶子，带来了很多没必要的开销，比较低效（很多叶子的分裂增益较低，没必要进行搜索和分裂）
- leaf-wise每次从当前所有叶子中，找到分裂增益最大(一般也是数据量最大)的一个叶子进行分裂。同 level-wise 相比，在分裂次数相同的情况下，leaf-wise 可以降低更多的误差，得到更好的精度。
- leaf-wise 的缺点是可能会长出比较深的决策树，产生过拟合。因此 LightGBM 在leaf-wise 之上增加了一个最大深度的限制，在保证高效率的同时防止过拟合。



Level-wise tree growth



Leaf-wise tree growth

histogram 做差加速

- histogram 做差加速：一个叶子的直方图可以由它的父亲节点的直方图与它兄弟的直方图做差得到。
- 通常构造直方图，需要遍历该叶子上的所有数据
- 直方图做差仅需遍历直方图的K个桶。利用这个方法，LightGBM 可以在构造一个叶子的直方图后，可以用非常微小的代价得到它兄弟叶子的直方图，在速度上可以提升一倍。

直接支持类别特征

- 实际上大多数机器学习工具都无法直接支持类别特征，一般需要把类别特征，转化到多维的0/1特征，降低了空间和时间的效率。
- LightGBM优化了对类别特征的支持，可以直接输入类别特征，不需要额外的0/1展开。并在决策树算法上增加了类别特征的决策规则。

并行

- LightGBM原生支持并行学习：支持特征并行和数据并行
- 特征并行：不同机器在不同的特征集合上分别寻找最优的分割点，然后在机器间同步最优的分割点。在本地保存全部数据可避免对数据切分结果的通信
- 数据并行：不同的机器先在本地构造直方图，然后进行全局的合并，最后在合并的直方图上面寻找最优分割点。在数据并行中使用分散规约 (Reduce scatter) 把直方图合并的任务分摊到不同的机器，降低通信和计算，并利用直方图做差，进一步减少了一半的通信量。
- 基于投票的数据并行则进一步优化数据并行中的通信代价，使通信代价变成常数级别。

LightGBM使用

- 同xgboost类似，LightGBM既支持独立使用，也支持在scikit learn框架下使用

- 独立调用LightGBM
  - import lightgbm as lgb

- 与scikit-learn一起使用
  - lgb.sklearn.LGBMClassifier

单独调用LightGBM

- 读取数据，构造Dataset
- 设置参数，train/cv
- predict

8.1 Data Structure API

class lightgbm.**Dataset**(*data*, *label=None*, *max_bin=255*, *reference=None*, *weight=None*, *group=None*, *silent=False*, *feature_name='auto'*, *categorical_feature='auto'*, *params=None*, *free_raw_data=True*)

Bases: object

Dataset in LightGBM.

Constract Dataset.

Parameters

- **data** (*string, numpy array or scipy.sparse*) – Data source of Dataset. If string, it represents the path to txt file.
- **label** (*list or numpy 1-D array, optional (default=None)*) – Label of the data.
- **max_bin** (*int, optional (default=255)*) – Max number of discrete bins for features.
- **reference** (Dataset *or None, optional (default=None)*) – If this is Dataset for validation, training data should be used as reference.
- **weight** (*list, numpy 1-D array or None, optional (default=None)*) – Weight for each instance.
- **group** (*list, numpy 1-D array or None, optional (default=None)*) – Group/query size for Dataset.
- **silent** (*bool, optional (default=False)*) – Whether to print messages during construction.
- **feature_name** (*list of strings or 'auto', optional (default="auto")*) – Feature names. If 'auto' and data is pandas DataFrame, data columns names are used.

用于排序任务

8.1 Data Structure API

(cont.)

class `lightgbm.`**`Dataset`**(*data*, *label=None*, *max_bin=255*, *reference=None*, *weight=None*, *group=None*, *silent=False*, *feature_name='auto'*, *categorical_feature='auto'*, *params=None*, *free_raw_data=True*)

Bases: `object`

Dataset in LightGBM.

Constract Dataset.

- **`categorical_feature`** (*`list of strings or int, or 'auto', optional (default="auto")`*) – Categorical features. If list of int, interpreted as indices. If list of strings, interpreted as feature names (need to specify `feature_name` as well). If 'auto' and data is pandas DataFrame, pandas categorical columns are used.
- **`params`** (*`dict or None, optional (default=None)`*) – Other parameters.
- **`free_raw_data`** (*`bool, optional (default=True)`*) – If True, raw data is freed after constructing inner Dataset.

8.2 Training API

`lightgbm.`**`train`**(*params*, *train_set*, *num_boost_round=100*, *valid_sets=None*, *valid_names=None*, *fobj=None*, *feval=None*, *init_model=None*, *feature_name='auto'*, *categorical_feature='auto'*, *early_stopping_rounds=None*, *evals_result=None*, *verbose_eval=True*, *learning_rates=None*, *keep_training_booster=False*, *callbacks=None*)

Perform the training with given parameters.

Parameters

- **params** (*dict*) – Parameters for training.
- **train_set** (Dataset) – Data to be trained.
- **num_boost_round** (*int, optional (default=100)*) – Number of boosting iterations.
- **valid_sets** (*list of Datasets or None, optional (default=None)*) – List of data to be evaluated during training.
- **valid_names** (*list of string or None, optional (default=None)*) – Names of valid_sets.
- **fobj** (*callable or None, optional (default=None)*) – Customized objective function.
- **feval** (*callable or None, optional (default=None)*) – Customized evaluation function. Note: should return (eval_name, eval_result, is_higher_better) or list of such tuples.
- **init_model** (*string or None, optional (default=None)*) – Filename of LightGBM model or Booster instance used for continue training.

参数说明详见用户手册（lightgbm.pdf）Chapter 6 Parameters

8.2 Training API

(cont.)

`lightgbm.`**`train`**(*params*, *train_set*, *num_boost_round=100*, *valid_sets=None*, *valid_names=None*, *fobj=None*, *feval=None*, *init_model=None*, *feature_name='auto'*, *categorical_feature='auto'*, *early_stopping_rounds=None*, *evals_result=None*, *verbose_eval=True*, *learning_rates=None*, *keep_training_booster=False*, *callbacks=None*)

- **feature_name** (*list of strings or 'auto', optional (default="auto")*) – Feature names. If 'auto' and data is pandas DataFrame, data columns names are used.
- **categorical_feature** (*list of strings or int, or 'auto', optional (default="auto")*) – Categorical features. If list of int, interpreted as indices. If list of strings, interpreted as feature names (need to specify `feature_name` as well). If 'auto' and data is pandas DataFrame, pandas categorical columns are used.
- **early_stopping_rounds** (*int or None, optional (default=None)*) – Activates early stopping. The model will train until the validation score stops improving. Requires at least one validation data and one metric. If there's more than one, will check all of them. If early stopping occurs, the model will add `best_iteration` field.
- **evals_result** (*dict or None, optional (default=None)*) – This dictionary used to store all evaluation results of all the items in `valid_sets`.
- **verbose_eval** (*bool or int, optional (default=True)*) – Requires at least one validation data. If True, the eval metric on the valid set is printed at each boosting stage. If int, the eval metric on the valid set is printed at every `verbose_eval` boosting stage. The last boosting stage or the boosting stage found by using `early_stopping_rounds` is also printed.

8.2 Training API

(cont.)

lightgbm.**train**(*params*, *train_set*, *num_boost_round=100*, *valid_sets=None*, *valid_names=None*, *fobj=None*, *feval=None*, *init_model=None*, *feature_name='auto'*, *categorical_feature='auto'*, *early_stopping_rounds=None*, *evals_result=None*, *verbose_eval=True*, *learning_rates=None*, *keep_training_booster=False*, *callbacks=None*)

- **learning_rates** (*list, callable or None, optional (default=None)*) – List of learning rates for each boosting round or a customized function that calculates `learning_rate` in terms of current number of round (e.g. yields learning rate decay).
- **keep_training_booster** (*bool, optional (default=False)*) – Whether the returned Booster will be used to keep training. If False, the returned value will be converted into _InnerPredictor before returning. You can still use _InnerPredictor as `init_model` for future continue training.
- **callbacks** (*list of callables or None, optional (default=None)*) – List of callback functions that are applied at each iteration. See Callbacks in Python API for more information.

Returns **booster** – The trained Booster model.

Return type *Booster*

1.lightgbm.**cv**(*params*, *train_set*, *num_boost_round=10*, *folds=None*, *nfold=5*, *stratified=True*, *shuffle=True*, *metrics=None*, *fobj=None*, *feval=None*, *init_model=None*, *feature_name='auto'*, *categorical_feature='auto'*, *early_stopping_rounds=None*, *fpreproc=None*, *ver- bose_eval=None*, *show_stdv=True*, *seed=0*, *callbacks=None*)

•**metrics** (*string, list of strings or None, optional (default=None)*) – Evaluation metrics to be monitored while CV. If not None, the metric in params will be overridden.

1.返回结果：eval_hist – Evaluation history. The dictionary has the following format: {'metric1-mean': [values], 'metric1-stdv': [values], 'metric2-mean': [values], 'metric1-stdv': [values], ...}.

例：

```
cv_result = lgbm.cv(…)
print cv_result['multi_logloss-mean']
```

objective

- `application`, default=`regression`, type=enum, options=`regression`, `regression_l2`, `regression_l1`, `huber`, `fair`, `poisson`, `binary`, `lambdarank`, `multiclass`, alias=`objective`, `app`
  - `regression`, regression application
    - `regression_l2`, L2 loss, alias=`mean_squared_error`, `mse`
    - `regression_l1`, L1 loss, alias=`mean_absolute_error`, `mae`
    - `huber`, Huber loss
    - `fair`, Fair loss
    - `poisson`, Poisson regression
  - `binary`, binary classification application
  - `lambdarank`, lambdarank application
    - the label should be `int` type in lambdarank tasks, and larger number represent the higher relevance (e.g. 0:bad, 1:fair, 2:good, 3:perfect)
    - `label_gain` can be used to set the gain(weight) of `int` label
  - `multiclass`, multi-class classification application, `num_class` should be set as well

metrics

Metric Parameters

- `metric`, default={ `l2` for regression}, { `binary_logloss` for binary classification}, { `ndcg` for lambdarank}, type=multi-enum, options= `l1`, `l2`, `ndcg`, `auc`, `binary_logloss`, `binary_error` ...
  - `l1`, absolute loss, alias= `mean_absolute_error`, `mae`
  - `l2`, square loss, alias= `mean_squared_error`, `mse`
  - `l2_root`, root square loss, alias= `root_mean_squared_error`, `rmse`
  - `huber`, Huber loss
  - `fair`, Fair loss
  - `poisson`, Poisson regression
  - `ndcg`, NDCG
  - `map`, MAP
  - `auc`, AUC
  - `binary_logloss`, log loss
  - `binary_error`. For one sample: `0` for correct classification, `1` for error classification
  - `multi_logloss`, log loss for mulit-class classification
  - `multi_error`, error rate for mulit-class classification
  - support multi metrics, separated by `,`
- `metric_freq`, default= `1`, type=int
  - frequency for metric output
- `train_metric`, default= `false`, type=bool, alias= `training_metric`, `is_training_metric`
  - set this to `true` if you need to output metric result of training
- `ndcg_at`, default= `1,2,3,4,5`, type=multi-int, alias= `ndcg_eval_at`, `eval_at`
  - NDCG evaluation positions, separated by `,`

► 例：

- import lightgbm as lgb
- train = pd.read_csv('../input/train_2016.csv')
- y_train = train ['logerror']
- x_train = train.drop([‘logerror’], axis=1)
- train_data=lgb.Dataset(x_train, label=y_train)

- *#setting parameters for lightgbm*
- param = {'num_leaves':150, 'objective':'binary','max_depth':7,'learning_rate':.05,'max_bin':200} param['metric'] = ['auc', 'binary_logloss']

- *#training our model using lightgbm*
- num_round=50
- lgbm=lgb.train(param, train_data, num_round)

- *#predicting on test set*
- ypred2=lgbm.predict(x_test)

与scikit-learn结合使用

- LightGBM也提供wrapper类，允许模型可以和scikit-learn框架中其他分类器或回归器一样对待
- 分类模型：LGBMClassifier
- 回归模型：LGBMRegressor
- 排序模型：LGBMRanker

class lightgbm.**LGBMClassifier**(*boosting_type='gbdt', num_leaves=31, max_depth=-1, learning_rate=0.1, n_estimators=10, max_bin=255, subsample_for_bin=200000, objective=None, min_split_gain=0.0, min_child_weight=0.001, min_child_samples=20, subsample=1.0, subsample_freq=1, colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0, **kwargs*)

与scikit-learn结合使用

- scikit-learn框架中其他分类器或回归器一样对待：三部曲
  - 构造实例，传递参数
  - fit

fit(*X*, *y*, *sample_weight=None*, *init_score=None*, *eval_set=None*, *eval_names=None*, *eval_sample_weight=None*, *eval_init_score=None*, *eval_metric='logloss'*, *early_stopping_rounds=None*, *verbose=True*, *feature_name='auto'*, *categorical_feature='auto'*, *callbacks=None*)

  - predict

predict_proba(*X*, *raw_score=False*, *num_iteration=0*)

特征重要性

- LightGBM也提供feature_importances_
- 可以使用LightGBM内嵌的函数，按特征重要性排序
  - plot_importance(model_XGB)
  - pyplot.show()

lightgbm.**plot_importance**(*booster*, *ax=None*, *height=0.2*, *xlim=None*, *ylim=None*, *title='Feature importance'*, *xlabel='Feature importance'*, *ylabel='Features'*, *importance_type='split'*, *max_num_features=None*, *ignore_zero=True*, *figsize=None*, *grid=True*, ***kwargs*)

影响模型预测性能的参数

- **num_leaves** : 模型中叶子结点数目。叶子结点越多，模型越复杂
  - 理论上num_leaves= 2^(max_depth)，但由于LightGBM采用的是叶子有限的分裂方式，这样样可能会过拟，所以通常设置为比2^(max_depth) 小的数值（如max_depth=6， num_leaves =70～80）
- **min_data_in_leaf** : 叶子结点最小的样本数目，设置得太小会过拟合。对大数据集，通常设置为几百～几千
- **max_depth**: 树的最大深度。深度越大，模型越复杂

影响模型速度的参数

- bagging_fraction : 每次选择部分样本更快
- feature_fraction : 每次迭代选择更少的特征速度更快
- max_bin : 较小的max_bin 速度更快

LightGBM参数调优

- 步骤参照xgboost

实验结果比较

- 实验数据

Data	Task	Link	#Train_Set	#Fea- ture	Comments
Higgs	Binary classification	link	10,500,000	28	use last 500,000 samples as test set
Yahoo LTR	Learning to rank	link	473,134	700	set1.train as train, set1.test as test
MS LTR	Learning to rank	link	2,270,296	137	{S1,S2,S3} as train set, {S5} as test set
Expo	Binary classification	link	11,000,000	700	use last 1,000,000 as test set
Allstate	Binary classification	link	13,184,290	4228	use last 1,000,000 as test set

实验结果比较

- 实验参数

xgboost

eta = 0.1
max_depth = 8
num_round = 500
nthread = 16
tree_method = exact
min_child_weight = 10

xgboost_hist

eta = 0.1
num_round = 500
nthread = 16
tree_method = approx
min_child_weight = 100
tree_method = hist
grow_policy = lossguide
max_depth = 0
max_leaves = 255

LightGBM

learning_rate = 0.1
num_leaves = 255
num_trees = 500
num_threads = 16
min_data_in_leaf = 0
min_sum_hessian_in_leaf = 100

实验环境

OS	CPU	Memory
Ubuntu 14.04 LTS	2 * E5-2670 v3	DDR4 2133Mhz, 256GB

实验结果比较

- 速度比较

Data	xgboost	xgboost_hist	LightGBM
Higgs	3794.34 s	551.898 s	238.505513 s
Yahoo LTR	674.322 s	265.302 s	150.18644 s
MS LTR	1251.27 s	385.201 s	215.320316 s
Expo	1607.35 s	588.253 s	138.504179 s
Allstate	2867.22 s	1355.71 s	348.084475 s

实验结果比较

- 性能比较（ **Accuracy** ）

Data	**Metric**	**xgboost**	**xgboost_hist**	**LightGBM**
Higgs	AUC	0.839593	0.845605	0.845154
Yahoo LTR	$NDCG_1$	0.719748	0.720223	0.732466
	$NDCG_3$	0.717813	0.721519	0.738048
	$NDCG_5$	0.737849	0.739904	0.756548
	$NDCG_{10}$	0.78089	0.783013	0.796818
MS LTR	$NDCG_1$	0.483956	0.488649	0.524255
	$NDCG_3$	0.467951	0.473184	0.505327
	$NDCG_5$	0.472476	0.477438	0.510007
	$NDCG_{10}$	0.492429	0.496967	0.527371
Expo	AUC	0.756713	0.777777	0.777543
Allstate	AUC	0.607201	0.609042	0.609167

实验结果比较

- 内存消耗比较

Data	xgboost	xgboost_hist	LightGBM
Higgs	4.853GB	3.784GB	0.868GB
Yahoo LTR	1.907GB	1.468GB	0.831GB
MS LTR	5.469GB	3.654GB	0.886GB
Expo	1.553GB	1.393GB	0.543GB
Allstate	6.237GB	4.990GB	1.027GB

实验结果比较

- 并行计算性能比较

Data	Task	Link	#Data	#Feature
Criteo	Binary classification	link	1,700,000,000	67

OS	CPU	Memory	Network Adapter
Windows Server 2012	2 * E5-2670 v2	DDR3 1600Mhz, 256GB	Mellanox ConnectX-3, 54Gbps, RDMA support

#Machine	Time per Tree	Memory Usage(per Machine)
1	627.8 s	176GB
2	311 s	87GB
4	156 s	43GB
8	80 s	22GB
16	42 s	11GB

近似线性的速度提升

References

- LightGBM文档：
  - http://lightgbm.readthedocs.io/en/latest/index.html
- LightGBM参数说明：
  - http://lightgbm.readthedocs.io/en/latest/Parameters.html

CSDN
不 止 于 代 码

THANK YOU

AI100