参数

这个页面包含了 LightGBM 的所有参数.

一些有用的链接列表

- Python API
- Parameters Tuning

外部链接

- Laurae++ Interactive Documentation

更新于 08/04/2017

以下参数的default已经修改:

- `min_data_in_leaf` = 100 => 20
- `min_sum_hessian_in_leaf` = 10 => 1e-3
- `num_leaves` = 127 => 31
- `num_iterations` = 10 => 100

参数格式

参数的格式为 `key1=value1 key2=value2 ...`. 并且, 在配置文件和命令行中均可以设置参数. 使用命令行设置参数时, 在 `=` 前后都不应该有空格. 使用配置文件设置参数时, 一行只能包含一个参数. 你可以使用 `#` 进行注释.

如果一个参数在命令行和配置文件中均出现了, LightGBM 将会使用命令行中的该参数.

核心参数

- `config`, default= `""`, type=string, alias= `config_file`
  - 配置文件的路径
- `task`, default= `train`, type=enum, options= `train`, `predict`, `convert_model`
  - `train`, alias= `training`, for training
  - `predict`, alias= `prediction`, `test`, for prediction.
  - `convert_model`, 要将模型文件转换成 if-else 格式, 可以查看这个链接获取更多信息 Convert model parameters

- `application`, default=`regression`, type=enum, options=`regression`, `regression_l1`, `huber`, `fair`, `poisson`, `quantile`, `quantile_l2`, `binary`, `multiclass`, `multiclassova`, `xentropy`, `xentlambda`, `lambdarank`, alias=`objective`, `app`
  - regression application
    - `regression_l2`, L2 loss, alias=`regression`, `mean_squared_error`, `mse`
    - `regression_l1`, L1 loss, alias=`mean_absolute_error`, `mae`
    - `huber`, Huber loss
    - `fair`, Fair loss
    - `poisson`, Poisson regression
    - `quantile`, Quantile regression
    - `quantile_l2`, 类似于 `quantile`, 但是使用了 L2 loss
  - `binary`, binary log loss classification application
  - multi-class classification application
    - `multiclass`, softmax 目标函数, 应该设置好 `num_class`
    - `multiclassova`, One-vs-All 二分类目标函数, 应该设置好 `num_class`
  - cross-entropy application
    - `xentropy`, 目标函数为 cross-entropy (同时有可选择的线性权重), alias=`cross_entropy`
    - `xentlambda`, 替代参数化的 cross-entropy, alias=`cross_entropy_lambda`
    - 标签是 [0, 1] 间隔内的任意值
  - `lambdarank`, lambdarank application
    - 在 lambdarank 任务中标签应该为 `int` type, 数值越大代表相关性越高 (e.g. 0:bad, 1:fair, 2:good, 3:perfect)
    - `label_gain` 可以被用来设置 `int` 标签的增益 (权重)
- `boosting`, default=`gbdt`, type=enum, options=`gbdt`, `rf`, `dart`, `goss`, alias=`boost`, `boosting_type`
  - `gbdt`, 传统的梯度提升决策树
  - `rf`, Random Forest (随机森林)
  - `dart`, Dropouts meet Multiple Additive Regression Trees
  - `goss`, Gradient-based One-Side Sampling (基于梯度的单侧采样)
- `data`, default=`""`, type=string, alias=`train`, `train_data`
  - 训练数据, LightGBM 将会使用这个数据进行训练
- `valid`, default=`""`, type=multi-string, alias=`test`, `valid_data`, `test_data`
  - 验证/测试 数据, LightGBM 将输出这些数据的度量
  - 支持多验证数据集, 以 `,` 分割
- `num_iterations`, default=`100`, type=int, alias=`num_iteration`, `num_tree`, `num_trees`, `num_round`, `num_rounds`, `num_boost_round`
  - boosting 的迭代次数
  - **Note**: 对于 Python/R 包, **这个参数是被忽略的**, 使用 `train` and `cv` 的输入参数 `num_boost_round` (Python) or `nrounds` (R) 来代替
  - **Note**: 在内部, LightGBM 对于 `multiclass` 问题设置 `num_class * num_iterations` 棵树
- `learning_rate`, default=`0.1`, type=double, alias=`shrinkage_rate`
  - shrinkage rate (收缩率)

    - 在 `dart` 中, 它还影响了 dropped trees 的归一化权重
- `num_leaves`, default=`31`, type=int, alias=`num_leaf`
    - 一棵树上的叶子数
- `tree_learner`, default=`serial`, type=enum, options=`serial`, `feature`, `data`, `voting`, alias=`tree`
    - `serial`, 单台机器的 tree learner
    - `feature`, alias=`feature_parallel`, 特征并行的 tree learner
    - `data`, alias=`data_parallel`, 数据并行的 tree learner
    - `voting`, alias=`voting_parallel`, 投票并行的 tree learner
    - 请阅读 并行学习指南 来了解更多细节
- `num_threads`, default=`OpenMP_default`, type=int, alias=`num_thread`, `nthread`
    - LightGBM 的线程数
    - 为了更快的速度, 将此设置为真正的 CPU 内核数, 而不是线程的数量 (大多数 CPU 使用超线程来使每个 CPU 内核生成 2 个线程)
    - 当你的数据集小的时候不要将它设置的过大 (比如, 当数据集有 10,000 行时不要使用 64 线程)
    - 请注意, 任务管理器或任何类似的 CPU 监视工具可能会报告未被充分利用的内核. **这是正常的**
    - 对于并行学习, 不应该使用全部的 CPU 内核, 因为这会导致网络性能不佳
- `device`, default=`cpu`, options=`cpu`, `gpu`
    - 为树学习选择设备, 你可以使用 GPU 来获得更快的学习速度
    - **Note**: 建议使用较小的 `max_bin` (e.g. 63) 来获得更快的速度
    - **Note**: 为了加快学习速度, GPU 默认使用32位浮点数来求和. 你可以设置 `gpu_use_dp=true` 来启用64位浮点数, 但是它会使训练速度降低
    - **Note**: 请参考 安装指南 来构建 GPU 版本

学习控制参数

- `max_depth`, default=`-1`, type=int
    - 限制树模型的最大深度. 这可以在 `#data` 小的情况下防止过拟合. 树仍然可以通过 leaf-wise 生长.
    - `< 0` 意味着没有限制.
- `min_data_in_leaf`, default=`20`, type=int, alias=`min_data_per_leaf`, `min_data`, `min_child_samples`
    - 一个叶子上数据的最小数量. 可以用来处理过拟合.
- `min_sum_hessian_in_leaf`, default=`1e-3`, type=double, alias=`min_sum_hessian_per_leaf`, `min_sum_hessian`, `min_hessian`, `min_child_weight`
    - 一个叶子上的最小 hessian 和. 类似于 `min_data_in_leaf`, 可以用来处理过拟合.
- `feature_fraction`, default=`1.0`, type=double, `0.0 < feature_fraction < 1.0`, alias=`sub_feature`, `colsample_bytree`
    - 如果 `feature_fraction` 小于 `1.0`, LightGBM 将会在每次迭代中随机选择部分特征. 例如, 如果设置为 `0.8`, 将会在每棵树训练之前选择 80% 的特征
    - 可以用来加速训练

    - 可以用来处理过拟合
- `feature_fraction_seed`, default=`2`, type=int
    - `feature_fraction` 的随机数种子
- `bagging_fraction`, default=`1.0`, type=double, `0.0 < bagging_fraction < 1.0`, alias=`sub_row`, `subsample`
    - 类似于 `feature_fraction`, 但是它将在不进行重采样的情况下随机选择部分数据
    - 可以用来加速训练
    - 可以用来处理过拟合
    - **Note**: 为了启用 bagging, `bagging_freq` 应该设置为非零值
- `bagging_freq`, default=`0`, type=int, alias=`subsample_freq`
    - bagging 的频率, `0` 意味着禁用 bagging. `k` 意味着每 `k` 次迭代执行bagging
    - **Note**: 为了启用 bagging, `bagging_fraction` 设置适当
- `bagging_seed`, default=`3`, type=int, alias=`bagging_fraction_seed`
    - bagging 随机数种子
- `early_stopping_round`, default=`0`, type=int, alias=`early_stopping_rounds`, `early_stopping`
    - 如果一个验证集的度量在 `early_stopping_round` 循环中没有提升, 将停止训练
- `lambda_l1`, default=`0`, type=double, alias=`reg_alpha`
    - L1 正则
- `lambda_l2`, default=`0`, type=double, alias=`reg_lambda`
    - L2 正则
- `min_split_gain`, default=`0`, type=double, alias=`min_gain_to_split`
    - 执行切分的最小增益
- `drop_rate`, default=`0.1`, type=double
    - 仅仅在 `dart` 时使用
- `skip_drop`, default=`0.5`, type=double
    - 仅仅在 `dart` 时使用, 跳过 drop 的概率
- `max_drop`, default=`50`, type=int
    - 仅仅在 `dart` 时使用, 一次迭代中删除树的最大数量
    - `<=0` 意味着没有限制
- `uniform_drop`, default=`false`, type=bool
    - 仅仅在 `dart` 时使用, 如果想要均匀的删除, 将它设置为 `true`
- `xgboost_dart_mode`, default=`false`, type=bool
    - 仅仅在 `dart` 时使用, 如果想要使用 xgboost dart 模式, 将它设置为 `true`
- `drop_seed`, default=`4`, type=int
    - 仅仅在 `dart` 时使用, 选择 dropping models 的随机数种子
- `top_rate`, default=`0.2`, type=double
    - 仅仅在 `goss` 时使用, 大梯度数据的保留比例
- `other_rate`, default=`0.1`, type=int
    - 仅仅在 `goss` 时使用, 小梯度数据的保留比例
- `min_data_per_group`, default=`100`, type=int
    - 每个分类组的最小数据量
- `max_cat_threshold`, default=`32`, type=int

    - 用于分类特征
    - 限制分类特征的最大阈值
- `cat_smooth`, default=`10`, type=double
    - 用于分类特征
    - 这可以降低噪声在分类特征中的影响, 尤其是对数据很少的类别
- `cat_l2`, default=`10`, type=double
    - 分类切分中的 L2 正则
- `max_cat_to_onehot`, default=`4`, type=int
    - 当一个特征的类别数小于或等于 `max_cat_to_onehot` 时, one-vs-other 切分算法将会被使用
- `top_k`, default=`20`, type=int, alias=`topk`
    - 被使用在 Voting parallel 中
    - 将它设置为更大的值可以获得更精确的结果, 但会减慢训练速度

IO 参数

- `max_bin`, default=`255`, type=int
    - 工具箱的最大数特征值决定了容量 工具箱的最小数特征值可能会降低训练的准确性, 但是可能会增加一些一般的影响（处理过度学习）
    - LightGBM 将根据 `max_bin` 自动压缩内存。 例如, 如果 maxbin=255, 那么 LightGBM 将使用 uint8t 的特性值
- `max_bin`, default=`255`, type=int
- `min_data_in_bin`, default=`3`, type=int - 单个数据箱的最小数, 使用此方法避免 one-data-one-bin（可能会过度学习）
- `data_r和om_seed`, default=`1`, type=int
    - 并行学习数据分隔中的随机种子 (不包括并行功能)
- `output_model`, default=`LightGBM_model.txt`, type=string, alias=`model_output`, `model_out`
    - 培训中输出的模型文件名
- `input_model`, default=`""`, type=string, alias=`model_input`, `model_in`
    - 输入模型的文件名
    - 对于 `prediction` 任务, 该模型将用于预测数据
    - 对于 `train` 任务, 培训将从该模型继续
- `output_result`, default=`LightGBM_predict_result.txt`, type=string, alias=`predict_result`, `prediction_result`
    - `prediction` 任务的预测结果文件名
- `model_format`, default=`text`, type=multi-enum, 可选项=`text`, `proto`
    - 保存和加载模型的格式
    - `text`, 使用文本字符串
    - `proto`, 使用协议缓冲二进制格式
    - 您可以通过使用逗号来进行多种格式的保存, 例如 `text,proto`. 在这种情况下, `model_format` 将作为后缀添加 `output_model`
    - **Note**: 不支持多种格式的加载

    - **Note**: 要使用这个参数, 您需要使用 build 版本 <./Installation-Guide.rst#protobuf-support>`__
- `pre_partition`, default=`false`, type=bool, alias=`is_pre_partition`
    - 用于并行学习(不包括功能并行)
    - `true` 如果训练数据 pre-partitioned, 不同的机器使用不同的分区
- `is_sparse`, default=`true`, type=bool, alias=`is_enable_sparse`, `enable_sparse`
    - 用于 enable/disable 稀疏优化. 设置 `false` 就禁用稀疏优化
- `two_round`, default=`false`, type=bool, alias=`two_round_loading`, `use_two_round_loading`
    - 默认情况下, LightGBM 将把数据文件映射到内存, 并从内存加载特性。 这将提供更快的数据加载速度。但当数据文件很大时, 内存可能会耗尽
    - 如果数据文件太大, 不能放在内存中, 就把它设置为 `true`
- `save_binary`, default=`false`, type=bool, alias=`is_save_binary`, `is_save_binary_file`
    - 如果设置为 `true` LightGBM 则将数据集(包括验证数据)保存到二进制文件中。 可以加快数据加载速度。
- `verbosity`, default=`1`, type=int, alias=`verbose`
    - `<0` = 致命的, `=0` = 错误 (警告), `>0` = 信息
- `header`, default=`false`, type=bool, alias=`has_header`
    - 如果输入数据有标识头, 则在此处设置 `true`
- `label`, default=`""`, type=string, alias=`label_column`
    - 指定标签列
    - 用于索引的数字, e.g. `label=0` 意味着 column_0 是标签列
    - 为列名添加前缀 `name:`, e.g. `label=name:is_click`
- `weight`, default=`""`, type=string, alias=`weight_column`
    - 列的指定
    - 用于索引的数字, e.g. `weight=0` 表示 column_0 是权重点
    - 为列名添加前缀 `name:`, e.g. `weight=name:weight`
    - **Note**: 索引从 `0` 开始. 当传递 type 为索引时, 它不计算标签列, 例如当标签为 0 时, 权重为列 1, 正确的参数是权重值为 0
- `query`, default=`""`, type=string, alias=`query_column`, `group`, `group_column`
    - 指定 query/group ID 列
    - 用数字做索引, e.g. `query=0` 意味着 column_0 是这个查询的 Id
    - 为列名添加前缀 `name:`, e.g. `query=name:query_id`
    - **Note**: 数据应按照 query_id. 索引从 `0` 开始. 当传递 type 为索引时, 它不计算标签列, 例如当标签为列 0, 查询 id 为列 1 时, 正确的参数是查询 =0
- `ignore_column`, default=`""`, type=string, alias=`ignore_feature`, `blacklist`
    - 在培训中指定一些忽略的列
    - 用数字做索引, e.g. `ignore_column=0,1,2` 意味着 column_0, column_1 和 column_2 将被忽略
    - 为列名添加前缀 `name:`, e.g. `ignore_column=name:c1,c2,c3` 意味着 c1, c2 和 c3 将被忽略
    - **Note**: 只在从文件直接加载数据的情况下工作
    - **Note**: 索引从 `0` 开始. 它不包括标签栏

- `categorical_feature`, default=`""`, type=string, alias=`categorical_column`, `cat_feature`, `cat_column`
  - 指定分类特征
  - 用数字做索引, e.g. `categorical_feature=0,1,2` 意味着 column_0, column_1 和 column_2 是分类特征
  - 为列名添加前缀 `name:`, e.g. `categorical_feature=name:c1,c2,c3` 意味着 c1, c2 和 c3 是分类特征
  - **Note**: 只支持分类与 `int` type. 索引从 `0` 开始. 同时它不包括标签栏
  - **Note**: 负值的值将被视为 **missing values**
- `predict_raw_score`, default=`false`, type=bool, alias=`raw_score`, `is_predict_raw_score`
  - 只用于 `prediction` 任务
  - 设置为 `true` 只预测原始分数
  - 设置为 `false` 只预测分数
- `predict_leaf_index`, default=`false`, type=bool, alias=`leaf_index`, `is_predict_leaf_index`
  - 只用于 `prediction` 任务
  - 设置为 `true` to predict with leaf index of all trees
- `predict_contrib`, default=`false`, type=bool, alias=`contrib`, `is_predict_contrib`
  - 只用于 `prediction` 任务
  - 设置为 `true` 预估 SHAP values, 这代表了每个特征对每个预测的贡献. 生成的特征+1的值, 其中最后一个值是模型输出的预期值, 而不是训练数据
- `bin_construct_sample_cnt`, default=`200000`, type=int, alias=`subsample_for_bin`
  - 用来构建直方图的数据的数量
  - 在设置更大的数据时, 会提供更好的培训效果, 但会增加数据加载时间
  - 如果数据非常稀疏, 则将其设置为更大的值
- `num_iteration_predict`, default=`-1`, type=int
  - 只用于 `prediction` 任务
  - 用于指定在预测中使用多少经过培训的迭代
  - `<= 0` 意味着没有限制
- `pred_early_stop`, default=`false`, type=bool
  - 如果 `true` 将使用提前停止来加速预测。可能影响精度
- `pred_early_stop_freq`, default=`10`, type=int
  - 检查早期early-stopping的频率
- `pred_early_stop_margin`, default=`10.0`, type=double
  - t提前early-stopping的边际阈值
- `use_missing`, default=`true`, type=bool
  - 设置为 `false` 禁用丢失值的特殊句柄
- `zero_as_missing`, default=`false`, type=bool
  - 设置为 `true` 将所有的0都视为缺失的值 (包括 libsvm/sparse 矩阵中未显示的值)
  - 设置为 `false` 使用 `na` 代表缺失值
- `init_score_file`, default=`""`, type=string
  - 训练初始分数文件的路径, `""` 将使用 `train_data_file` + `.init` (如果存在)
- `valid_init_score_file`, default=`""`, type=multi-string

    - 验证初始分数文件的路径, `""` 将使用 `valid_data_file` + `.init` (如果存在)
    - 通过 `,` 对multi-validation进行分离

目标参数

- `sigmoid`, default= `1.0`, type=double
    - sigmoid 函数的参数. 将用于 `binary` 分类 和 `lambdarank`
- `alpha`, default= `0.9`, type=double
    - Huber loss 和 Quantile regression 的参数. 将用于 `regression` 任务
- `fair_c`, default= `1.0`, type=double
    - Fair loss 的参数. 将用于 `regression` 任务
- `gaussian_eta`, default= `1.0`, type=double
    - 控制高斯函数的宽度的参数. 将用于 `regression_l1` 和 `huber` losses
- `poisson_max_delta_step`, default= `0.7`, type=double
    - Poisson regression 的参数用于维护优化
- `scale_pos_weight`, default= `1.0`, type=double
    - 正值的权重 `binary` 分类 任务
- `boost_from_average`, default= `true`, type=bool
    - 只用于 `regression` 任务
    - 将初始分数调整为更快收敛速度的平均值
- `is_unbalance`, default= `false`, type=bool, alias= `unbalanced_sets`
    - 用于 `binary` 分类
    - 如果培训数据不平衡 设置为 `true`
- `max_position`, default= `20`, type=int
    - 用于 `lambdarank`
    - 将在这个 NDCG 位置优化
- `label_gain`, default= `0,1,3,7,15,31,63,...`, type=multi-double
    - 用于 `lambdarank`
    - 有关获得标签. 列如, 如果使用默认标签增益 这个 `2` 的标签则是 `3`
    - 使用 `,` 分隔
- `num_class`, default= `1`, type=int, alias= `num_classes`
    - 只用于 `multiclass` 分类
- `reg_sqrt`, default= `false`, type=bool
    - 只用于 `regression`
    - 适合``sqrt(label)`` 相反, 预测结果也会自动转换成 `pow2(prediction)`

度量参数

- `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`
    - `quantile`, Quantile regression
    - `huber`, Huber loss
    - `fair`, Fair loss
    - `poisson`, Poisson regression
    - `ndcg`, NDCG
    - `map`, MAP
    - `auc`, AUC
    - `binary_logloss`, log loss
    - `binary_error`, 样本: `0` 的正确分类, `1` 错误分类
    - `multi_logloss`, mulit-class 损失日志分类
    - `multi_error`, error rate for mulit-class 出错率分类
    - `xentropy`, cross-entropy (与可选的线性权重), alias=`cross_entropy`
    - `xentlambda`, "intensity-weighted" 交叉熵, alias=`cross_entropy_lambda`
    - `kldiv`, Kullback-Leibler divergence, alias=`kullback_leibler`
    - 支持多指标, 使用 `,` 分隔
- `metric_freq`, default=`1`, type=int
    - 频率指标输出
- `train_metric`, default=`false`, type=bool, alias=`training_metric`, `is_training_metric`
    - 如果你需要输出训练的度量结果则设置 `true`
- `ndcg_at`, default=`1,2,3,4,5`, type=multi-int, alias=`ndcg_eval_at`, `eval_at`
    - NDCG 职位评估, 使用 `,` 分隔

网络参数

以下参数用于并行学习, 只用于基本(socket)版本。

- `num_machines`, default=`1`, type=int, alias=`num_machine`
    - 用于并行学习的并行学习应用程序的数量
    - 需要在socket和mpi版本中设置这个
- `local_listen_port`, default=`12400`, type=int, alias=`local_port`
    - 监听本地机器的TCP端口
    - 在培训之前, 您应该再防火墙设置中放开该端口
- `time_out`, default=`120`, type=int
    - 允许socket几分钟内超时
- `machine_list_file`, default=`""`, type=string, alias=`mlist`
    - 为这个并行学习应用程序列出机器的文件
    - 每一行包含一个IP和一个端口为一台机器。格式是ip port, 由空格分隔

GPU 参数

- `gpu_platform_id`, default=`-1`, type=int
  - OpenCL 平台 ID. 通常每个GPU供应商都会公开一个OpenCL平台。
  - default为 `-1`, 意味着整个系统平台
- `gpu_device_id`, default=`-1`, type=int
  - OpenCL设备ID在指定的平台上。 在选定的平台上的每一个GPU都有一个唯一的设备ID
  - default为``-1``, 这个default意味着选定平台上的设备
- `gpu_use_dp`, default=`false`, type=bool
  - 设置为 `true` 在GPU上使用双精度GPU (默认使用单精度)

模型参数

该特性仅在命令行版本中得到支持。

- `convert_model_language`, default=`""`, type=string
  - 只支持``cpp``
  - 如果 `convert_model_language` 设置为 `task``时 该模型也将转换为 ``train`,
- `convert_model`, default=`"gbdt_prediction.cpp"`, type=string
  - 转换模型的输出文件名

其他

持续训练输入分数

LightGBM支持对初始得分进行持续的培训。它使用一个附加的文件来存储这些初始值, 如下:

```
0.5
-0.1
0.9
...
```

它意味着最初的得分第一个数据行是 `0.5`,第二个是 ``-0.1` 等等。 初始得分文件与数据文件逐行对应, 每一行有一个分数。 如果数据文件的名称是 `train.txt`, 最初的分数文件应该被命名为 ``train.txt.init` 与作为数据文件在同一文件夹。 在这种情况下, LightGBM 将自动加载初始得分文件, 如果它存在的话。

权重数据

LightGBM 加权训练。它使用一个附加文件来存储权重数据, 如下:

```
1.0
0.5
0.8
...
```

它意味的重压着第一个数据行是 `1.0`, 第二个是 `0.5`, 等等. 权重文件按行与数据文件行相对应, 每行的权重为. 如果数据文件的名称是 `train.txt`, 应该将重量文件命名为 `train.txt.weight` 与数据文件相同的文件夹. 在这种情况下, LightGBM 将自动加载权重文件, 如果它存在的话.

update: 现在可以在数据文件中指定 `weight` 列。请参阅以上参数的参数.

查询数据

对于 LambdaRank 的学习, 它需要查询信息来训练数据. LightGBM 使用一个附加文件来存储查询数据, 如下:

```
27
18
67
...
```

它意味着第一个 `27` 行样本属于一个查询和下一个 `18` 行属于另一个, 等等. **Note**: 数据应该由查询来排序.

如果数据文件的名称是``train.txt`,这个查询文件应该被命名为``train.txt.query``查询在相同的培训数据文件夹中。 在这种情况下, LightGBM将自动加载查询文件, 如果它存在的话。

update: 现在可以在数据文件中指定特定的 query/group id。请参阅上面的参数组。