XGBoost Parameters

Before running XGboost, we must set three types of parameters: general parameters, booster parameters and task parameters.

- General parameters relates to which booster we are using to do boosting, commonly tree or linear model
- Booster parameters depends on which booster you have chosen
- Learning Task parameters that decides on the learning scenario, for example, regression tasks may use different parameters with ranking tasks.
- Command line parameters that relates to behavior of CLI version of xgboost.

Parameters in R Package

In R-package, you can use .(dot) to replace under score in the parameters, for example, you can use max.depth as max_depth. The underscore parameters are also valid in R.

General Parameters

- booster [default=gbtree]
  - which booster to use, can be gbtree or gblinear. gbtree uses tree based model while gblinear uses linear function.
- silent [default=0]
  - 0 means printing running messages, 1 means silent mode.
- nthread [default to maximum number of threads available if not set]
  - number of parallel threads used to run xgboost
- num_pbuffer [set automatically by xgboost, no need to be set by user]
  - size of prediction buffer, normally set to number of training instances. The buffers are used to save the prediction results of last boosting step.
- num_feature [set automatically by xgboost, no need to be set by user]
  - feature dimension used in boosting, set to maximum dimension of the feature

Parameters for Tree Booster

- eta [default=0.3]
  - step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features. and eta actually shrinks the

feature weights to make the boosting process more conservative.
    - range: [0,1]
- gamma [default=0]
    - minimum loss reduction required to make a further partition on a leaf node of the tree. the larger, the more conservative the algorithm will be.
    - range: [0,∞]
- max_depth [default=6]
    - maximum depth of a tree
    - range: [1,∞]
- min_child_weight [default=1]
    - minimum sum of instance weight(hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning. In linear regression mode, this simply corresponds to minimum number of instances needed to be in each node. The larger, the more conservative the algorithm will be.
    - range: [0,∞]
- max_delta_step [default=0]
    - Maximum delta step we allow each tree's weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative. Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced. Set it to value of 1-10 might help control the update
    - range: [0,∞]
- subsample [default=1]
    - subsample ratio of the training instance. Setting it to 0.5 means that XGBoost randomly collected half of the data instances to grow trees and this will prevent overfitting.
    - range: (0,1]
- colsample_bytree [default=1]
    - subsample ratio of columns when constructing each tree.
    - range: (0,1]
- lambda [default=1]
    - L2 regularization term on weights
- alpha [default=0]
    - L1 regularization term on weights

Parameters for Linear Booster

- lambda [default=0]
    - L2 regularization term on weights
- alpha [default=0]

    - L1 regularization term on weights
- lambda_bias
    - L2 regularization term on bias, default 0(no L1 reg on bias because it is not important)

Learning Task Parameters

Specify the learning task and the corresponding learning objective. The objective options are below:

- objective [default=reg:linear]

- "reg:linear" –linear regression
- "reg:logistic" –logistic regression
- "binary:logistic" –logistic regression for binary classification, output probability
- "binary:logitraw" –logistic regression for binary classification, output score before logistic transformation
- "count:poisson" –poisson regression for count data, output mean of poisson distribution
    - max_delta_step is set to 0.7 by default in poisson regression (used to safeguard optimization)
- "multi:softmax" –set XGBoost to do multiclass classification using the softmax objective, you also need to set num_class(number of classes)
- "multi:softprob" –same as softmax, but output a vector of ndata * nclass, which can be further reshaped to ndata, nclass matrix. The result contains predicted probability of each data point belonging to each class.
- "rank:pairwise" –set XGBoost to do ranking task by minimizing the pairwise loss

- base_score [default=0.5]
    - the initial prediction score of all instances, global bias
- eval_metric [default according to objective]
    - evaluation metrics for validation data, a default metric will be assigned according to objective(rmse for regression, and error for classification, mean average precision for ranking)
    - User can add multiple evaluation metrics, for python user, remember to pass the metrics in as list of parameters pairs instead of map, so that latter 'eval_metric' won't override previous one
    - The choices are listed below:
    - "rmse": root mean square error
    - "logloss": negative log-likelihood
    - "error": Binary classification error rate. It is calculated as #(wrong cases)/#(all cases). For the predictions, the evaluation will regard the instances with

prediction value larger than 0.5 as positive instances, and the others as negative instances.
    - “merror”: Multiclass classification error rate. It is calculated as #(wrong cases)/#(all cases).
    - “mlogloss”: Multiclass logloss
    - “auc”: Area under the curve for ranking evaluation.
    - “ndcg”:Normalized Discounted Cumulative Gain
    - “map”:Mean average precision
    - “ndcg@n”,”map@n”: n can be assigned as an integer to cut off the top positions in the lists for evaluation.
    - “ndcg-”,”map-”,”ndcg@n-”,”map@n-”: In XGBoost, NDCG and MAP will evaluate the score of a list without any positive samples as 1. By adding “-” in the evaluation metric XGBoost will evaluate these score as 0 to be consistent under some conditions. training repeatively
- seed [default=0]

- random number seed.

Command Line Parameters

The following parameters are only used in the console version of xgboost

- use_buffer [default=1]

- Whether to create a binary buffer from text input. Doing so normally will speed up loading times

- num_round

- The number of rounds for boosting

- data
    - The path of training data
- test:data
    - The path of test data to do prediction
- save_period [default=0]

- the period to save the model, setting save_period=10 means that for every 10 rounds XGBoost will save the model, setting it to 0 means not save any model during training.

- task [default=train] options: train, pred, eval, dump

    - train: training using data
    - pred: making prediction for test:data
    - eval: for evaluating statistics specified by eval[name]=filename
    - dump: for dump the learned model into text format(preliminary)
- model_in [default=NULL]
    - path to input model, needed for test, eval, dump, if it is specified in training, xgboost will continue training from the input model
- model_out [default=NULL]
    - path to output model after training finishes, if not specified, will output like 0003.model where 0003 is number of rounds to do boosting.
- model_dir [default=models]
    - The output directory of the saved models during training
- fmap
    - feature map, used for dump model
- name_dump [default=dump.txt]
    - name of model dump file
- name_pred [default=pred.txt]
    - name of prediction file, used in pred mode
- pred_margin [default=0]
    - predict margin instead of transformed probability