XGBoost-Guide to parameter tuning for XGBoost

XGBoost is an advanced implementation of gradient boosting algorithm.

Advantages:

1. Regularisation: Standard GBM has no regularisation like XGBoost, therefore it also helps to reduce overfitting. In fact, XGBoost is also known as ‘regularised boosting’ technique.
2. Parallel Processing: XGBoost implements parallel processing and is blazing faster as compared to GBM. XGBoost also support implementation on Hadoop.
3. High Flexibility: XGBoost allows users to define custom optimization objectives and evaluation criteria. This adds a whole new dimension to the model.
4. Handling missing values: XGBoost has an in-built routine to handle missing values. The user is required to supply a different value than other observations and pass that as a parameter. XGBoost tries different things as it encounter a missing value on each node and learns which path to take for missing values in future.
5. Tree Pruning: A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a ‘greedy’ algorithm. XGBoost on the other hand make splits up to the max\_depth specified and then starts pruning the tree backwards and removes splits beyond which there is no positive gain. Another advantage is that sometimes a split of negative loss say -2 may be followed by a split of positive loss +10. GBM would stop as it encounters -2. But XGboost will go deeper and it will see a combined effect of +8 of the split and keep both.
6. Built-in Cross-Validation: XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run.
7. Continue on Existing Model: User can start training an XGBoost model from its last iteration of previous run. This can be of significant advantage in certain specific applications. GBM implementation of sklearn also has this feature so they are even on this point.

XGBoost Parameters

The overall parameters have been divided into 3 categories by XGBoost authors:

-General Parameters: Guide the overall functioning  
-Booster Parameters: Guide the individual booster (tree/regression) each step  
-Learning Task Parameters: Guide the optimization performed

General Parameters

These define the overall functionality of XGBoost

1. Booster [default=gbtree]  
   -Select the type of model to run at each iteration. It has two options:  
    -gbtree: tree-based models  
    -gblinear: linear models
2. Silent[default=0]:  
   -silent mode is activated is set to 1, i.e. no running messages will be printed.   
   - It’s generally good to keep it 0 as the messages might help in understanding the model.
3. Nthread [default to maximum number of threads available if not set]  
   -This is used for parallel processing and number of cores in the system should be entered  
   -If you wish to run on all cores, value should not be entered and algorithm will detect automatically

Note: There are two more parameters which are set automatically by XGBoost.

Booster Parameters

There are two types of boosters however, the tree booster is seen to outperform the linear booster almost always.

-eta [default=0.3]: Analogous to learning rate in GBM. Makes the model more robust by shrinking the weights on each step. Typical final values to be used: 0.01-0.2.

-min\_child\_weight [default=1]: Defines the minimum sum of weights of all observations required in a child. Similar to min\_leaf\_child in GBM but not exactly. Refers to min ‘sum of weights’ of observations while GBM has min ‘number of observations’. Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree. Too high values can lead to under-fitting hence, it should be tuned using CV.

-max\_depth [default=6]: The maximum depth of a tree, same as GBM. Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample. Should be tuned using CV. Typical values are 3-10.

-max\_leaf\_nodes: The maximum number of terminal nodes or leaves in a tree. Can be defined in place of max\_depth. Since binary tress are created, a depth of ‘n’ would produce a maximum of 2^n leaves. If this is defined, GBM will ignore max\_depth.

-gamma[default=0]: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned.

-max\_delta\_step [default=0]: