

0% COMPLETE

python/lectures/1137863)

Basics

■ Boosting - wisdom of the crowd (12:13)

(/courses/practical-xgboost-in-python/lectures/1137866)

Parameters cheat-sheet
(/courses/practical-xgboost-inpython/lectures/1137867)

Using standard interface (4:07) (/courses/practical-xgboost-in-python/lectures/1137869)

Using Scikit-learn interface (2:15) (/courses/practical-xgboost-in-python/lectures/1137870)

Going deeper

■ Spotting most important features (4:00)

(/courses/practical-xgboost-inpython/lectures/1148761)

■ Bias/variance tradeoff (6:05) (/courses/practical-xgboost-inpython/lectures/1137871)

Hyperparameter tuning (4:21) (/courses/practical-xgboost-in-python/lectures/1137872)

Evaluate results (4:54) (/courses/practical-xgboost-in-python/lectures/1148760)

Deal with missing values (2:33) (/courses/practical-xgboost-in-python/lectures/1148762)

■ Handle imbalanced dataset (5:09) (/courses/practical-xgboost-in-python/lectures/1173895)

End notes

Thank you
(/courses/practical-xgboost-inpython/lectures/1437360)

♠ Previous Lecture (/courses/78878/lectures/1148762) Complete and continue → (/courses/78878/lectures/1437360)

Parameters cheat-sheet

The following list presents the full set of parameters used to tune XGBoost algorithm. Use it as a cheat-sheet while experimenting.

General

- booster=[gbtree|gblinear] which booster to use.
- silent=[0|1] 0 prints running messages, 1 means silent mode
- nthread number of threads to use. Maximum by default

Tree Booster

- eta=[0 .. 0.3 .. 1] step shrinkage used in update to prevents overfitting. After each boosting step eta shrinks instance weights. Lower value makes learning process more conservative (slower learning),
- gamma=[**0** .. ∞] minimum loss reduction required to make a further partition on a leaf node of the tree. Higher value makes the algorithm more conservative,
- max_depth=[1 .. 6 .. ∞] maximum depth of each tree,
- min_child_weight=[0 .. 1 .. ∞] minimum sum of instance weight needed in a tree node. Further
 partitioning from that node is abandoned when a sum is not obtained. Higher value makes the
 algorithm more conservative.
- max_delta_step=[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.
- subsample=[0,1] subsample ratio of randomly selected training instances used to grow trees,
- $\bullet \ \ \ colsample_by tree \hbox{\tt =[0,1]} subsample\ ratio\ of\ columns\ when\ construction\ each\ tree,$
- lambda=[1] L2 regularization term on weights
- alpha=[0] L1 regularization term on weights

Linear Booster

- alpha=[0] L1 regularization term on weights
- lambda=[0] L2 regularization term on weights
- lambda_bias=[0] L2 regularization term on bias

Learning task parameters

- objective=[
 - reg:linear # linear regression,
 - reg:logistic # logistic regression,
 - binary:logistic # logistic regression for binary classification, outputs probability,
 - binary:logitraw #logistic regression for binary classification, outputs score before logistic transformation.
 - count:poisson # poisson regression for count data, outputs mean of poisson distribution,
 - multi:softmax # do multiclass classification using softmax objective,
 - multi:softprob # same as above but outputs probability for each class,

0% COMPLETE

python/lectures/1137863)

Basics

■ Boosting - wisdom of the crowd (12:13)

(/courses/practical-xgboost-inpython/lectures/1137866)

Parameters cheat-sheet (/courses/practical-xgboost-in-python/lectures/1137867)

Using standard interface (4:07) (/courses/practical-xgboost-in-python/lectures/1137869)

Using Scikit-learn interface (2:15) (/courses/practical-xgboost-in-python/lectures/1137870)

Going deeper

Spotting most important features(4:00)

(/courses/practical-xgboost-inpython/lectures/1148761)

■ Bias/variance tradeoff (6:05) (/courses/practical-xgboost-inpython/lectures/1137871)

Hyperparameter tuning (4:21) (/courses/practical-xgboost-in-python/lectures/1137872)

Evaluate results (4:54) (/courses/practical-xgboost-in-python/lectures/1148760)

Deal with missing values (2:33) (/courses/practical-xgboost-in-python/lectures/1148762)

■ Handle imbalanced dataset (5:09) (/courses/practical-xgboost-inpython/lectures/1173895)

End notes

Thank you (/courses/practical-xgboost-in-python/lectures/1437360)

Parameters cheat-sheet | Parrot Prediction's Courses

- rank pairwise # execute ranking task by minimizing the pairwise loss Complete and continue • (/courses/78878/lectures/1148762) (/courses/78878/lectures/1437360)

- base_score=[0,5] global bias. Initial prediction score for all instances
- eval_metric=[rmse|logloss|error|merror|mlogloss|auc|...] evaluation metric. Default value will be assigned based on the objective. There is possibility of having custom metric.
- seed=[0] seed used for reproducibility