Bagging and Boosting

Machine Learning Practice

Dr. Ashish Tendulkar

IIT Madras

Contents

Voting estimators

Class: `sklearn.ensemble.VotingClassifier`

Class: `sklearn.ensemble.VotingRegressor`

Both these estimators take the following **common parameters**:

`base_estimator` `weights`

Both these estimators implement the following **functions**:

`fit` `predict` `fit_transform` `score`

VotingClassifier takes an **additional argument**:

`voting` `hard` `soft`

Bagging estimators

Class:	sklearn.ensemble.BaggingClassifier
Class:	sklearn.ensemble.BaggingRegressor

Common parameters

base_estimator	default=None	base estimator to fit on random subsets of dataset
n_estimators	default=10	number of base estimators in the ensemble
max_samples	default=1.0	number of samples to draw from X to train each base estimator (**with** replacement by default)
max_features	default=1.0	number of samples to draw from X to train each base estimator (**without** replacement by default)
bootstrap	default=True	Whether samples are drawn with replacement

Common parameters

bootstrap_features	default=False	Whether features are drawn with replacement
oob_score	default=False	Whether to use out-of-bag samples to estimate generalization error

Random forest estimators

Class: `sklearn.ensemble.RandomForestClassifier`

Class: `sklearn.ensemble.RandomForestRegressor`

The parameters can be classified as

- Bagging parameters
- Decision tree parameters

Bagging parameters

- The number of trees are specified by `n_estimators`.
  - Default #trees for classification = 10
  - Default #trees for regression = 100
- `bootstrap` specifies whether to use bootstrap samples for training.
  - `True`: bootstrapped samples are used.
  - `False`: whole dataset is used.
- `oob_score` specifies whether to use out-of-bag samples for estimating generalization error. It is only available when `bootstrap` = `True`.

Bagging parameters

- `max_samples` specifies the number of samples to be drawn while bootstrapping.
  - `None` : Use all samples in the training data.
  - `int` : Use `max_samples` samples from the training data.
  - `float` : Use `max_samples*total number of samples from training data`

    The value should be between 0 and 1.

- `random_state` controls randomness of features and samples selected during bootstrap.

- The number of features to be considered while splitting is specified by `max_features` .
  - `auto` , `sqrt` , `log2` , `int` , `float`

Value	**max_features**
int	value specified
float	value * # features
auto	sqrt(#features)
sqrt	sqrt(#features)
log2	log2(#features)
None	#features

Decision tree parameters

- The criteria for splitting the node is specified through `criterion`.
  - Default for classification: `gini`
  - Default for regression: `squared_error`

- The depth of the tree is controlled by `max_depth`. The default value is `None`, which means the tree will be grown until all leaf nodes are pure or until leaves contain less than `min_samples_splits` samples.

- We will continue to split the internal node until they contain `min_samples_splits` samples.
  - Whenever it is specified as an integer, then it is considered as a number.
  - Whenever it is specified as a float, and the `min_samples_splits` is calculated as $\texttt{min\_samples\_splits} \times n$.

- The tree growth can also be controlled by `min_impurity_decrease` parameter.
  - A node will be split if it reduces impurity at least by the value specified in this parameter.

- The complexity of tree can also be controlled by `ccp_alpha` parameter through minimal cost complexity pruning procedure.

Trained random forest estimators

- `estimators_` member variable contains a collection of fitted estimators.

- `feature_importances_` member variable contains a list of important features.

Training and inference for random forest

- `fit` builds forest of trees from the training dataset with the specified parameters.
- `decision_path` returns decision path in the forest.
- `predict` returns class label in classification and output value in regression.
- `predict_proba` and `predict_log_proba` returns probabilities and their logs for classification set up.