HOML Chapter 7 - Ensemble Learning and Random Forests

Voting Classifiers

Bagging and Pasting

Random Patches and Random Subspaces

Boosting

Stacking

Introduction

Ensemble - a group of predictors.

Ensemble Method - an ensemble learning algorithm.

An aggregation of the predictions of a group of predictors can often lead to better predictions than any given individual predictor within the ensemble.

Random Forest - ensemble of decision trees based on predicting the class that gets the most votes.

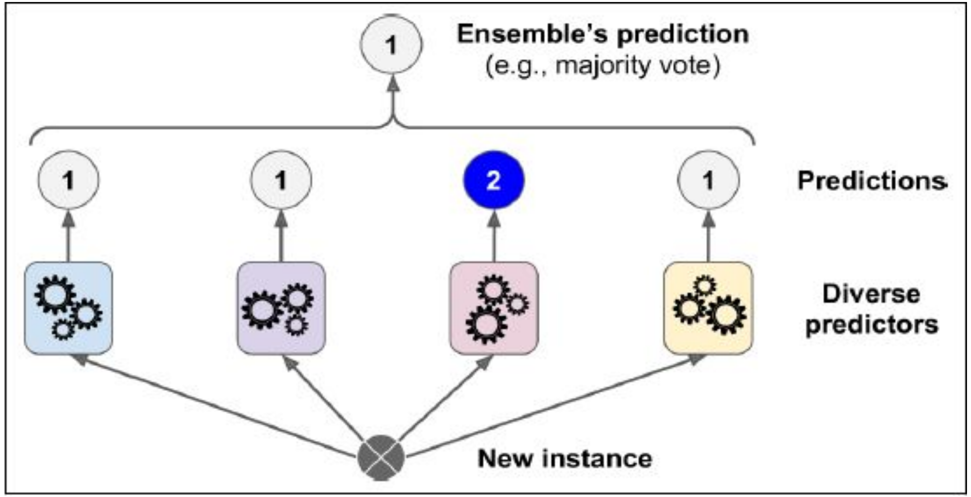
Voting Classifiers

Hard voting classifier - majority vote classification.

Even if every classifier within a hard voting classifier is individually weak, the ensemble can still be a strong learner given diverse and plentiful learners.

Predictors should be as independent as possible. To increase diversity in classifiers, train them using different algorithms.

Soft voting classifier - predicting the highest class probability, averaged over all individual classifiers.



Bagging and Pasting

Bagging (bootstrap aggregating) - sampling performed with replacement.

Pasting - sampling performed without replacement.

Statistical mode - aggregation function (most frequent prediction) for classification or average for regression.

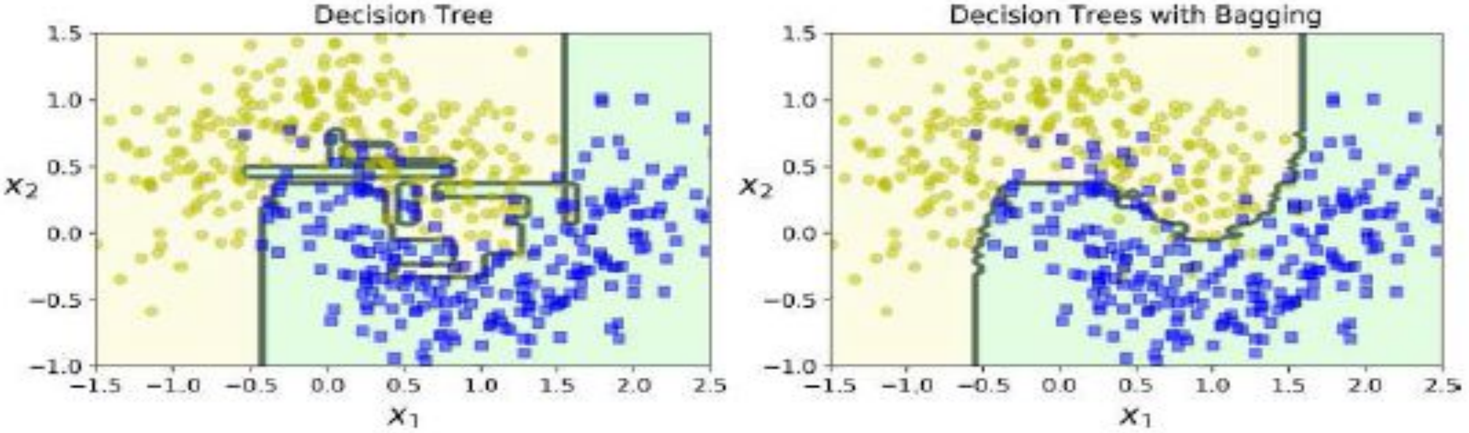
Aggregation reduces both bias and variance. Ensemble method generally has similar bias but lower variance than a single predictor.

Predictors can be trained in parallel (using different cores or servers) making bagging and pasting very scaleable.

Bagging and Pasting cont. - Scikit-Learn

Using Scikit-Learn’s BaggingClassifier/Regressor with DecisionTreeClassifier/Regressor results in a better generalization than that of a single Decision Tree.

Bagging generally produces better results than pasting because it allows for more diversity and lower ensemble variance.



Bagging and Pasting cont. - Out-of-Bag Evaluation

Only about 63% of training instances are sampled on average for each predictor using BaggingClassifier. The remaining 37% are out-of-bag-instances (oob) (but aren't the same 37% for each predictor).

Predictors never see the oob instances during training, so there's no need for a separate validation sets.

A predictor can be evaluated on oob by setting oob_score=True under BaggingClassifier and avoids the need to create a separate validation set.

Random Patches and Random Subspaces

BaggingClassifier allows for sampling of features.

Hyperparameters max_features and bootstrap_features are analogous to max_samples and bootstrap but for features rather than samples.

Useful for high-dimensional inputs (images) and results in higher predictor diversity, as well slightly higher bias for a lower variance.

Random Patches method - sampling both training instances and features.

Random Subspaces method - keeping all training instances and sampling features.

Random Forests

RandomForestClassifier/Regressor is an ensemble method optimized for Decision Trees, typically trained using bagging.

Contains most hyperparameters of DecisionTreeClassifer/Regressor and all hyperparameters of BaggingClassifier/Regressor.

Unlike Decision Trees, Random Forest searches the best features among a random subset of features rather than when splitting a node. It results in greater tree diversity but higher bias and lower variance.

Random Forests cont. - Extra-Trees

Extremely Randomized Trees ensemble - trees with increased randomness by using random thresholds for each feature rather than searching for the best thresholds (as Decision Trees do).

Allows for faster training than regular Random Forest because finding the best threshold is often the most time consuming aspect of growing trees.

Can be created using Scikit-Learn's ExtraTreeClassifier/Regressor (API identical to RandomForestClassifier/Regressor).

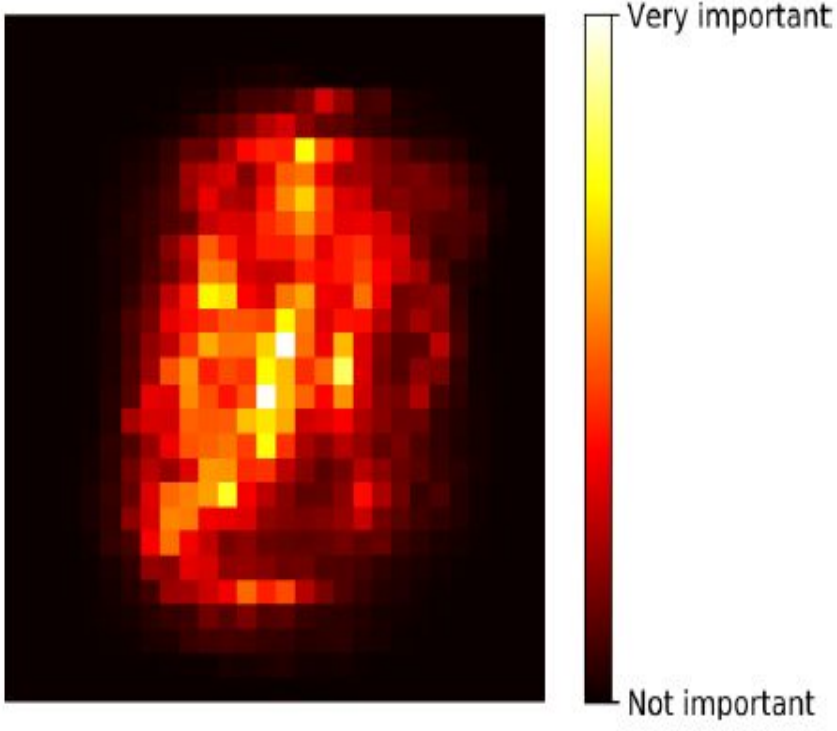
Hard to tell in advance whether ExtraTrees will perform better than RandomForest, so it's best to try them both and compare using cross validation.

Random Forests cont. - Feature Importance

A feature’s importance is measured by how much the tree nodes use the feature’s impurity
 on average across all trees.

Accessed using the feature_importances_ variable.

MNIST feature importance according to random classifier.



Boosting

Booting is an ensemble method that combines weak learners into a strong learner.

Train predictors sequentially, with each trying to correct its predecessor.

AdaBoost (Adaptive Boosting) and Gradient Boosting are the most popular.

Boosting cont. - AdaBoost

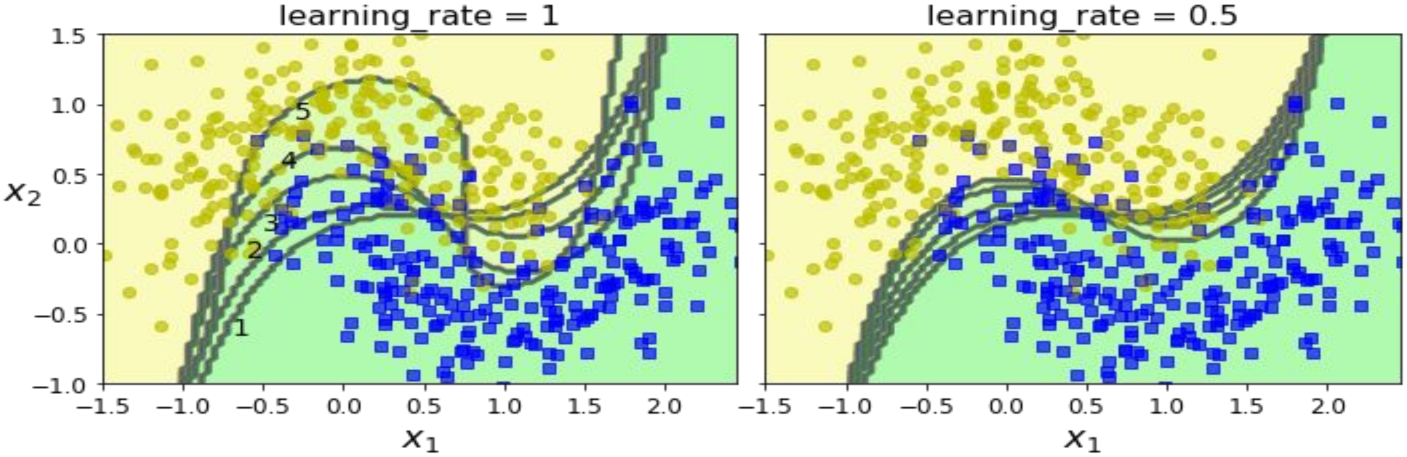
AdaBoost works by focusing on training instances that the predecessor underfitted, thus focusing increasingly on hard cases.

Starting with training a base classifier, the relative weights of misclassified training instances are increased. A second classifier is trained on the updated weights. After its predictions, the weights are once again updated and the third classifier trains on the newly updated weights, and so on.

After the predictors are trained, the predictors have different weights depending on their overall accuracy on the weighted training set.

Boosting cont. - AdaBoost

Has some similarities to Gradient Descent, except it adds predictors to an ensemble rather than adjusting a single predictors parameters to minimize a cost function.



SAMME (Stagewise Additive Modeling using a Multiclass Exponential loss function) - multiclass version of AdaBoost.

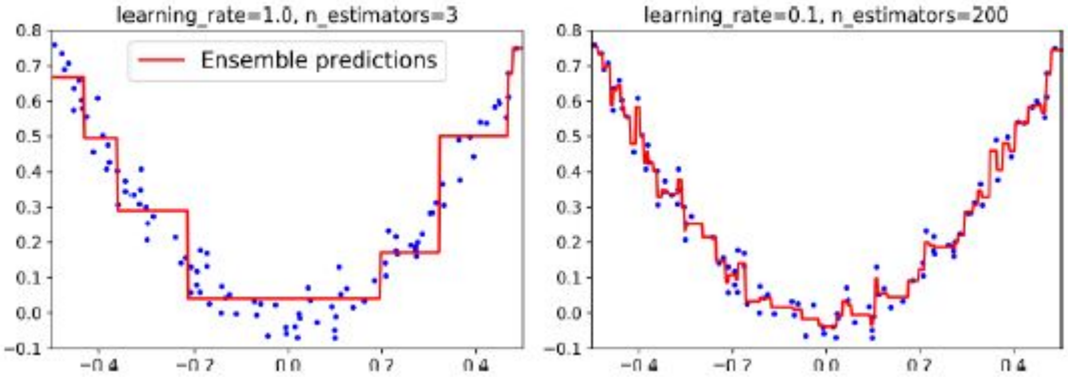
SAMME.R (Real) relies on class probabilities rather than predictions and often performs better than SAMME.

Boosting cont. - Gradient Boosting

Also sequentially adds predictors to an ensemble, with each correcting its predecessor, unlike AdaBoost, it fits a new predictor to residual errors made by the previous predictor.

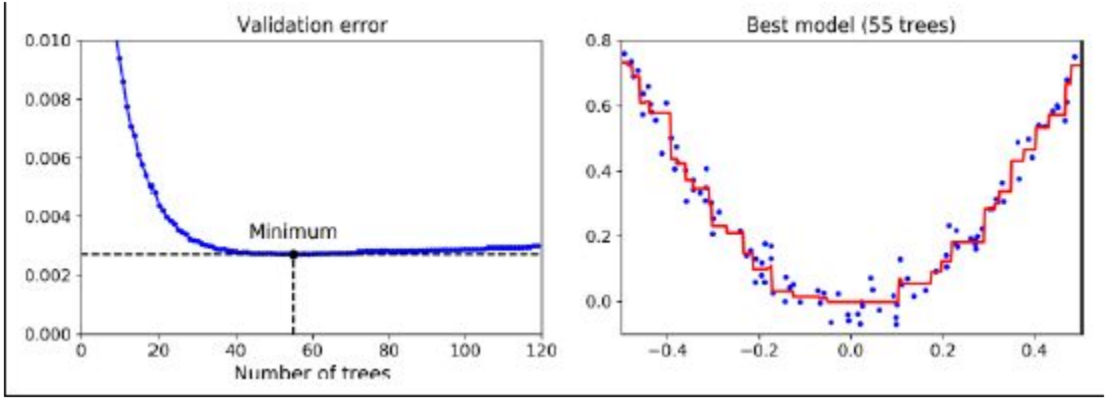
An easier way to train Gradient Boosting Regression Trees (GBRT) is to use GradientBoostingRegressor class. Its hyperparameters can control tree growth and ensemble training (control the number of trees with n_estimator).

Shrinkage - setting learning_rate hyperparameter to low value to add more trees to fit a training set but allows for better generalization.



Boosting cont. - Gradient Boosting

Employ early stopping to find the optimal number of trees using staged_predict() method, which returns an iterator over the predictors at each stage of training.



The subsample hyperparameter specifies a fraction of training instances (selected randomly) to be used for each training tree. Also referred to Stochastic Gradient Boosting. Again, higher bias for lower variance.

XGBoost (Extreme Gradient Boosting) library has optimized implementation of Gradient Boosting. It's meant to be fast and scalable and is often used in ML competitions.

Stacking

Stacking (stacked generalization) - an ensemble algorithm where a new model is trained to combine predictions from two or more models previously trained on a dataset.

Stacking is also referred to as blending since predictions form submodels are ‘blended’ together using a new model.

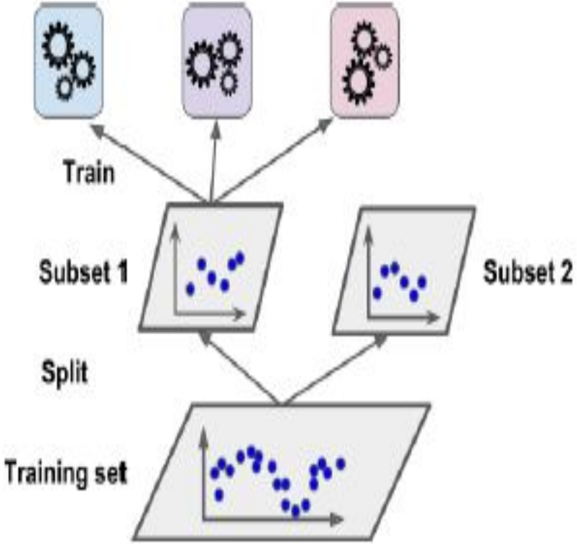
Submodels don’t necessarily have to be the best possible models, so they don’t often require a good amount of tuning, but the new model should show an improvement over a baseline model.

It’s important that the submodels make uncorrelated predictions. One way to achieve this is to use different types of algorithms (KNN, logistic regression, etc).

Stacking cont.

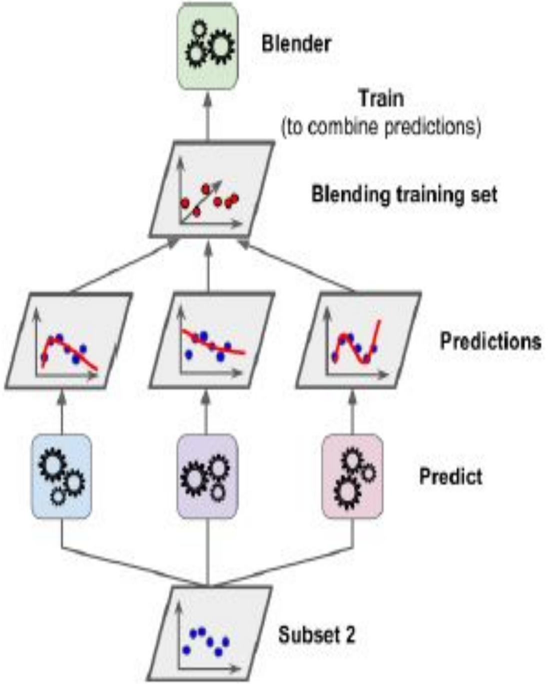
To train a blender, basic approach involves using a hold-out set. Split the training set into two subsets and use the first subset to train the predictors in the first layer.

The first layer predictors are then used to make predictions of the second held-out set.



Stacking cont.

A new training set can be created using these predicted values as input features. Blender is then trained on the new training set, learning to predict the target value given the first layer's predictions.



Key to training several different blenders using this method is to have three training subsets - The first to train the first layer, the second to create the training set to train the second layer (using predictions made on the first layer), and the third to create the training set to train the third layer (using predictions made on the second layer).