Ensembles: Bagging and Random Forests

Clayton W. Schupp, Galvanize

Spring 2015

What Constitutes an Ensemble Method

Ensemble: a technique for combining many weak learners in an attempt to produce a strong learner

Basic Idea: run multiple models on the data and aggregate the predictive results to produce an overall prediction that is better than any of the individual models could do on their own.

Ensembles Continued

Intuition

Classifier Example with Majority Vote:

- Suppose we have 5 completely independent classifiers with an accuracy of 70% for each
  - $\binom{5}{3}(0.7)^3(0.3)^2 + \binom{5}{4}(0.7)^4(0.3)^1 + \binom{5}{5}(0.7)^5(0.3)^0$
  - yields 83.7% majority vote accuracy
- If we had 101 such classifiers
  - 99.9% majority vote accuracy

It's clear that utilizing multiple weak classifiers can yield a very strong classifier

Bootstrap Sampling Revisited

Advantages:

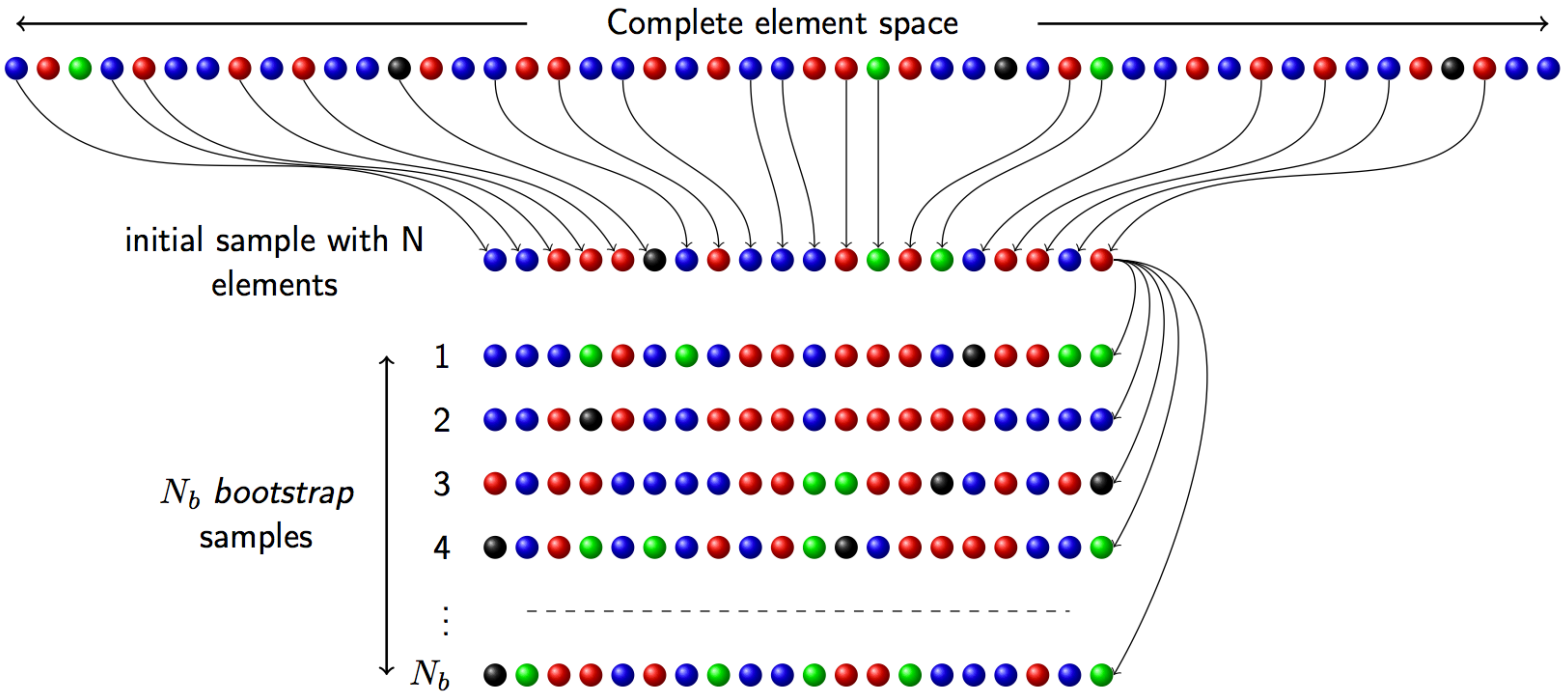
- Completely automatic
- Requires no theoretical calculations
- Not based on asymptotic results
- Available regardless of how complicated the estimator might be

Bootstrap Sampling Revisited

Method:

- Start with your dataset of size n
- Sample from your dataset with replacement to create 1 bootstrap sample of size n which means many of the observations will be repeated
- Repeat B times
- Each bootstrap sample can then be used as a separate dataset for estimation or model fitting

Bootstrap Sampling Revisited



Bootstrap Aggregation (Bagging)

Use bootstrap to improve predictions by using different samples of observations and/or features to generate diverse classifiers

- Improves stability and accuracy of ML algorithms
- Also reduces variance and helps to avoid overfitting
- Although usually applied to decision tree methods, it can be used with any type of method

Bootstrap Aggregation (Bagging)

Method:

- Create bootstrap samples, fit a model from each bootstrap sample; can use a subset of possible features to build models
- Each model then gets a vote in the final results
- Final selection based on aggregate predictions
  - majority vote if classification (i.e. predicting a category)
  - average if regression (i.e. predicting a continuous value)

Out-of-Bag Error

The key to bagging is that models are repeatedly fit to bootstrapped subsets of observations. Therefore:

- Each bagged model makes use of about 2/3 of the observations
- Remaining 1/3 of the observations are referred to as the *out-of-bag* (OOB) observations
- We can predict the i^{th} response using the bagged model in which the observation was OOB
- This will yield approximately B/3 predictions for the i^{th} observation and aggregate the predictions as above
- We now have predictions for all observations and can then calculate the OOB Error

Out-of-Bag Error

- OOB error is a valid estimate of the test error since the observation is predicted using models that did not use that observation
- When B is sufficiently large, OOB error is virtually equivalent to leave-one-out cross-validation error
- OOB approach is convenient when bagging on large data sets for which cross-validation would be computationally intensive

Feature Importance

However, the collection of bagged models is much more difficult to interpret than a single model

- Bagging improves prediction accuracy at the expense of interpretability
- We can obtain an overall summary of the importance of each predictor
  - Classification Trees: use Gini index
  - Regression Trees: use RSS

Random Forests vs Bagging Alone

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees

- Both methods utilize bootstrap samples
- RF also randomly selects features

Intuition behind this:

- In bagged trees, a very strong predictor will show up in the top split so all trees will look quite similar and highly correlated
- In RF, the very strong predictor will be included only a portion of the time, allowing moderately strong predictors to have a chance

Random Forest Algorithm

For $b = 1, \ldots, B$:

- Draw a bootstrap sample of size N from the training data
- Grow a random-forest tree to the bootstrapped data by recursively repeating the following for each terminal node of the tree, until minimum node size is reached
  - Picking a subset of features (m) from the p features
  - Pick the best variable/split-point among the m variables
  - Split the node into two child nodes
- Output the ensemble of trees

Forest Error

Overall forest error rate depends on two things:

- The *correlation* between any two trees in the forest
- The *strength* of each individual tree in the forest

Reducing *m* reduces both the correlation and strength, while increasing it increases both. Use OOB error to find the optimal value of m.

Some Advantages of Random Forests

- All trees are trained independently
  - possibly in parallel reducing computation time
- Can handle thousands of input variables without variable deletion
- Gives estimates of which features are important in the classification

If the number of variables is very large, forests can be run will all variables, then run again using only the most important from the first run

Pruning

- A single decision tree which can suffer from overfitting and needs to be pruned
- In Random Forests, each true is grown to the largest extent possible
- Pruning is not needed because overfitting is not an issue and due to the uncorrelated nature of the individual trees

Bias-Variance Tradeoff

- In Random Forests the bias of the full model is equivalent to the bias of a single decision tree (which itself has high variance)
- By creating a forest and averaging them the variance of the final model can be greatly reduced over that of a single tree

Only limitation of the size of the forest is computing time as an infinite number of trees could be trained without ever increasing bias and with a continually decrease in the variance (although asymptotically limited)

Regression Forests

I want to reiterate:

- The Random Forest method can be applied to regression trees to create regression forests
- Use RSS instead of Gini Index for OOB error and feature importance

General Form of all scikit.learn Models

```
from sklearn.ensemble import Model
from sklearn.metrics import confusion_matrix,
        precision_score, recall_score

model = Model()
model.fit(X_train, y_train)
model.score(X_test, y_test)

y_predict = model.predict(X_test)
confusion_matrix(y_test, y_predict)
precision_score(y_test, y_predict)
recall_score(y_test, y_predict)
```