STAT562 Lecture 11: Bagging and Random Forests

Beidi Qiang

SIUE

Introduction

- A single decision trees suffer from high variance: if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different.
- Bagging and random forests are designed for reducing the variance of a statistical learning method. These methods use trees as building blocks to construct more powerful prediction models.
- Idea: to reduce variance, we can average a set of observations. We can take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.

Bagging in general

We want to take many training sets from the population. However, we generally do not have access to multiple training sets. So, instead, we can bootstrap.

- We take repeated samples from the training data set and generates B different bootstrapped training data sets.
- We then train our method on each bootstrapped training set in order to get $f^{*b}(x)$, $b = 1, \cdots, B$.
- Finally we average all the predictions, to obtain

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} f^{*b}(x)$$

.

Bagging with trees

Bagging can improve predictions for many regression methods, it is particularly useful for decision trees.

- In regression problems (quantitative outcome), we simply construct B regression trees using B bootstrapped training sets, and average the resulting predictions.
- In classification problems, we simply construct B classification trees using B bootstrapped training sets, record the class predicted from each of the B trees, and take a majority vote.

Estimate the test error: Out-of-Bag Error

Recall that the key to bagging is that trees are repeatedly fit to a subsets of the observations. On average, each bagged tree only makes use of around two-thirds of the observations.

- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB (not used), which gives about B/3 predictions.
- We can get a single OOB prediction by average these predicted responses or can take a majority vote.
- The resulting OOB error is a valid estimate of the test error (no need to perform cross-validation!).

Feature Importance

Bagging typically improves accuracy over prediction using a single tree. However, it can be difficult to interpret the resulting model. i.e. it improves prediction accuracy at the expense of interpretability. We seek a way to obtain an overall summary of the importance of each predictor.

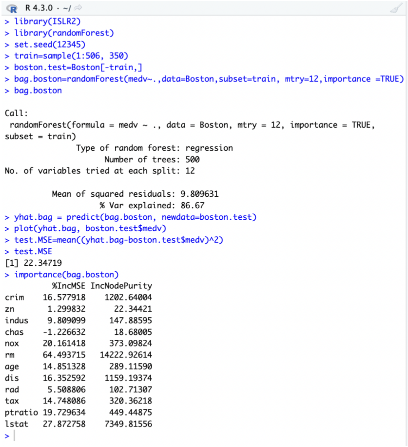
- We can record the total amount that the SSE (or Gini index in classification case) is decreased due to splits over a given predictor, averaged over all B trees.
- A large value indicates an important predictor.

Random Forests

Random forests (RF) provide an improvement over bagged trees by a small tweak.

- At each split in the tree, the RF algorithm only consider a small subset of m predictors (features) out of total p available predictors ($m \approx \sqrt{p}$).
- Random forests eliminate the chance that all of the bagged trees look quite similar to each other (highly correlated).
- We can think of this process as "de-correlating" the trees.
- Random forest will typically be helpful when we have a large number of correlated predictors.

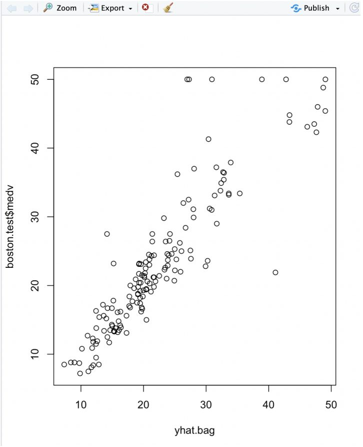
Example: Boston Data



```
> library(ISLR2)
> library(randomForest)
> set.seed(12345)
> train=sample(1:506, 350)
> boston.test=Boston[-train,]
> bag.boston=randomForest(medv~.,data=Boston,subset=train, mtry=12,importance =TRUE)
> bag.boston

Call:
 randomForest(formula = medv ~ ., data = Boston, mtry = 12, importance = TRUE,
subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 12

          Mean of squared residuals: 9.809631
                    % Var explained: 86.67
> yhat.bag = predict(bag.boston, newdata=boston.test)
> plot(yhat.bag, boston.test$medv)
> test.MSE=mean((yhat.bag-boston.test$medv)^2)
> test.MSE
[1] 22.34719
> importance(bag.boston)
          %IncMSE IncNodePurity
crim    16.577918    1202.64004
zn       1.299832      22.34421
indus    9.809099     147.88595
chas    -1.226632      18.68005
nox     20.161418     373.09824
rm      64.493715   14222.92614
age     14.851328     289.11590
dis     16.352592    1159.19374
rad      5.508806     102.71307
tax     14.748086     320.36218
ptratio 19.729634     449.44875
lstat   27.872758    7349.81556
>
```



Example: Iris Data

```
> lda.out=lda(default~income+student+balance)
> lda.out
Call:
lda(default ~ income + student + balance)

Prior probabilities of groups:
    No    Yes
0.9667 0.0333

Group means:
      income studentYes   balance
No  33566.17  0.2914037  803.9438
Yes 32089.15  0.3813814 1747.8217

Coefficients of linear discriminants:
                     LD1
income      3.367310e-06
studentYes -1.746631e-01
balance     2.243541e-03
> lda.class=predict(lda.out,Default)$class
> table(lda.class,default)
         default
lda.class   No  Yes
      No  9645  254
      Yes   22   79
```