BAGGING

Bagging, which is known as bootstrap aggregation, is a technique that repeatedly samples (with replacement) from a data set according to a uniform probability distribution. Each bootstrap samples are of same size as the original data. Because the sampling is done with replacement, some instances may appear several times in some training set, while others may be omitted from the training set.

Result: On average, a bootstrap sample D_i contains approximately 63% of original training data.

proof:

We have an original sample of n observations in it:

$$x_1, x_2, \dots, x_n$$

each of which have a probability $1/n$ of getting selected on the first draw.

(Recall that sample is drawn with replacement according to a uniform probability distribution. This type of sampling is called simple random sampling with replacement)

That is, in the first draw,

$$P(\text{choosing any one item, say } x_i) = \frac{1}{n}$$

$$\therefore P(\text{Not choosing that item, say } x_i) = 1 - \frac{1}{n}$$

∴ In n draws, the probablity of not choosing this obs. in any of those n draws is

$$\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n}\right)\cdots\left(1-\frac{1}{n}\right) \text{ n times} = \left(1-\frac{1}{n}\right)^n$$

B

, Since the drawings are independent of one another

Now, as n becomes larger and larger, i.e. as $n \to \infty$

$$\lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.368$$

$\therefore$ The probability of an item being chosen (as $n \to \infty$) is

$$1 - 0.368 = 0.632$$

i.e. 63.2%. (approx.)

BAGGING ALGORITHM

1: Let k be the number of bootstrap samples.

2: for $i = 1$ to k do

3: Create a bootstrap sample D_i of size N

4: Train a base classifier C_i on the bootstrap sample D_i.

5: end for

6: $C^*(x) = \underset{y}{\arg\max} \sum_i I[C_i(x) = y]$ {where $I(\cdot) = \begin{cases} 1 & \text{if its argument is true} \\ 0 & \text{, o.w.} \end{cases}$ }

The value of y that maximizes

the sum

2

Why bagged model?

Models like decision tree, ANN, etc. suffer from high variance. This means that 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. In contrast, a procedure with low variance will yield similar results if applied repeatedly to distinct data sets; for example linear regression tends to have low variance, if the ratio of n to p is moderately large.

Bagging is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision tree.

Recall that given a set of n independent observations $X_1, X_2, \ldots, X_n$, each with variance σ^2, following a Normal distribution, i.e. if

Intuition:

$$X_i \sim N(\mu, \sigma^2) \text{ , independently}$$

Then,

$$\bar{X} \sim N(\mu, \sigma^2/n)$$

This indicates that averaging a set of observations reduces variance. Hence a natural way to reduce the variance and hence increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set and take a majority vote.

3

Of course, this is not practical because we generally do not have access to multiple training sets. Instead, we bootstrap, by taking repeated samples from the (single) training data set.

While bagging can improve predictions for many models, it is particularly useful for decision trees. To apply bagging to decision trees we simply construct B decision trees using B bootstrapped training sets. These trees are grown deep and are not pruned. Hence each individual tree has high variance but low bias. Averaging these B trees reduces the variance. Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure.

A BOOTSTRAP AGGREGATION EXAMPLE

Table 5.4. Example of data set used to construct an ensemble of bagging classifiers.

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	−1	−1	−1	−1	1	1	1

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

Figure 5.35. Example of bagging.

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Figure 5.36. Example of combining classifiers constructed using the bagging approach.

CHOOSING THE NO. OF BOOTSTRAP SAMPLES

If you visualize the test error rate as a function of B (the no. of bootstrap samples), you are most likely to see that the bagging test error rate is lower than the test error rate obtained from a single tree. The no. of trees B is not a critical parameter with bagging; using a very high value of B will not lead to overfitting. In practice we use a value of B sufficiently large that the error has settled.

BAGGING FOR REGRESSION

For problems involving regression, we build separate prediction model using each bootstrapped training data sets. We then train our method on the bth ~~booths~~ bootstrapped sample in order to get $\hat{f}^{*b}(x)$ and finally average all the predictions, to obtain

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

where B is the no. of bootstrap training data.

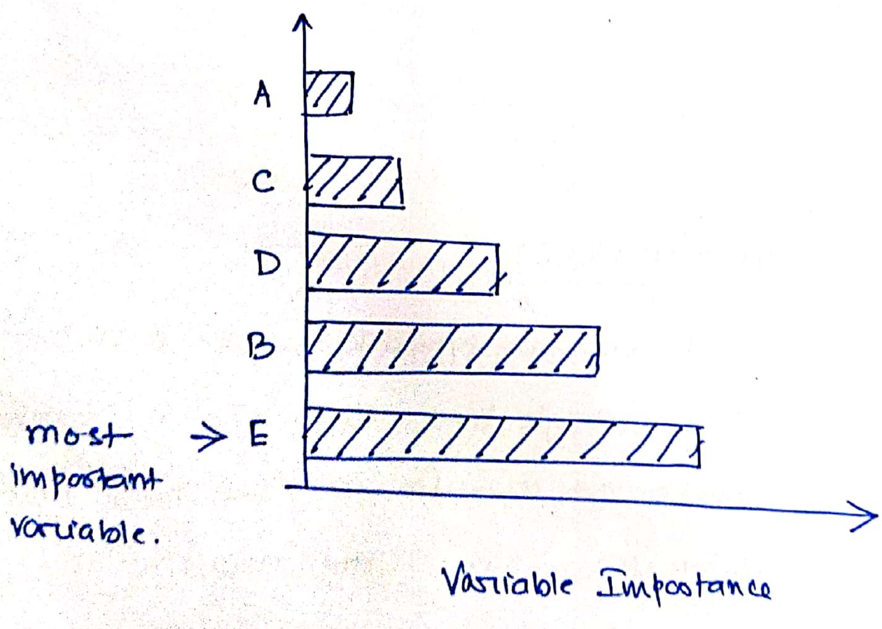
OUT-OF-BAG ERROR ESTIMATION

It turns out that there is a very straightforward way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach. The key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. We have already seen that on average, each bagged tree makes use of around 63% 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 the observation was OOB. This will yield around (B x 0.37) predictions for the ith observation. In order to obtain a single prediction for the ith observation, we can average these predicted response (if regression is the goal) or take a majority vot (if classification is the goal). This leads to a single OOB prediction for the ith observation. An OOB prediction can be obtained in this way for each of n observations from which the overall OOB MSE (for regression problem) or classification error (for a classification problem) can be computed. The resulting OOB error is a valid estimate of test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation.

VARIABLE IMPORTANCE MEASURES

As we have discussed, bagging typically results in improved accuracy over prediction using a single tree. Unfortunately, however, it can be difficult to interpret the resulting model. Recall that one of the advantages of decision tree is the attractive and easily interpreted diagram that results. However, when we bag a large no. of trees, it is no longer possible to represent the resulting statistical learning procedure using a single tree, and it is no longer clear which variables are most important to the procedure. Thus, bagging improves prediction accuracy at the expense of interpretability

Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor using the RSS (for bagged regression trees) or the Gini index (for the bagging classification trees). In case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B. A large value indicates an important predictor. Similarly, in the context of bagging classification trees, we can add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.



WHEN TO APPLY & NOT APPLY BAGGING?

Bagging improves generalization error by reducing the variance of the base classifiers. The performance of bagging depends on the stability of the base classifiers. If a base classifier is unstable, bagging helps to reduce the errors associated with random fluctuations in the training data. If a base classifier is stable, i.e., robust to minor purtubations in the training data, then the error of the ensemble is primarily caused by bias in the base classifier. In this situation, bagging may not be able to improve the performance of the base classifier significantly. It may even degrade the classifier's performance because the effective size of each training set is about 37% smaller than the original data.

Table: Stable or Unstable Classification

Classification Algo.	Stable/Unstable
CART	Unstable
C4.5	Unstable
Neural Networks	Unstable
k-NN	Stable
Discriminant Analysis	Stable
Naive Bayes	Stable.