

BAGGING

-APPLIED MULTIVARIATE ANALYSIS & STATISTICAL LEARNING-

ISL 5.3.4, 8.2

Lecturer: Darren Homrighausen, PhD

NOTATION

REMINDER: For either **classification** or **regression**, we produce **predictions** for a given feature vector X

That is, we form

$$\hat{Y} = \hat{f}(X) \in \mathbb{R} \quad \text{Regression}$$

or

$$\hat{Y} = \hat{g}(X) \in \mathcal{G} \quad \text{Classification}$$

where

- \hat{f} or \hat{g} is some procedure formed with the **training data**
(**EXAMPLE:** $\hat{\beta}$ formed by least squares)
- The prediction \hat{Y} formed at a desired feature vector X
(**EXAMPLE:** $\hat{Y} = X^\top \hat{\beta}$ formed by least squares)

BAGGING

Many methods (trees included) tend to be designed to have lower bias but high variance

HEURISTICALLY: If we split the training data into two parts at random and fit a decision tree to each part, the results could be quite **different**

A low variance estimator would yield **similar** results if applied repeatedly to distinct data sets

(consider $\hat{f}(X) = 0$ for all X)

Bagging, also known as **Bootstrap AGgregation**, is a general purpose procedure for reducing variance.

We'll use it specifically in the context of trees, but it can be applied more broadly.

Bagging for regression

BAGGING: THE MAIN IDEA

Suppose we have n uncorrelated observations Z_1, \dots, Z_n , each with variance σ^2 .

What is the variance of

$$\bar{Z} = \frac{1}{n} \sum_{i=1}^n Z_i?$$

ANSWER:

More generally, if we have B separate (uncorrelated) training sets, we could form B separate model fits,

$$\hat{f}^1(X), \dots, \hat{f}^B(X)$$

Then average them:

$$\hat{f}_B(X) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(X)$$

BAGGING: THE BOOTSTRAP PART

Of course, this isn't practical as having access to many training sets is unlikely.

We therefore turn to the **bootstrap** to simulate having many training sets.

The bootstrap is a widely applicable statistical tool that can be used to quantify uncertainty without Gaussian approximations.

BAGGING: THE BOOTSTRAP PART

Now, instead of having B separate training sets, we train on B **bootstrap** draws:

$$\hat{f}_1^*(X), \dots, \hat{f}_B^*(X)$$

and then average (i.e. **aggregate**) them:

$$\hat{f}_{\text{bag}}(X) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b^*(X)$$

This process is known as **Bagging**

Bagging trees



BAGGING TREES

The procedure for trees is the following

1. Choose a large number B .
2. For each $b = 1, \dots, B$, grow an unpruned tree on the b^{th} bootstrap draw from the data.
3. Average the predictions of each of these trees

Each tree, since it is unpruned, will have (low/high) variance and (low/high) bias

Bagging for classification

BAGGING TREES IN CLASSIFICATION

For classification, there are a few sensible methods for aggregation

For each test observation X ,

- record the length B vector $[\hat{g}_1^*(X), \dots, \hat{g}_B^*(X)]^\top$ and classify X via majority vote
- Average the length G probability vectors from each tree and choose the argmax

WARNING: One thing you definitely do not want to do is estimate probabilities via taking proportions of times X was classified to each class across the B trees

Additional tree bagging topics

BAGGING TREES

Now that we are growing a large number (B) of random trees, we can't directly look at the **dendrogram**

(we have sacrificed some interpretability for better performance)

However, we do get some helpful information instead

- Mean decrease variable importance
- Out-of-Bag error estimation (OOB)
- Permutation variable importance
- Proximity

(these ideas apply to bagging any low bias procedure, not just unpruned trees)

MEAN DECREASE VARIABLE IMPORTANCE

Observation: Every time a split of a node is made on a **feature**, the loss function is not increased

Hence, adding up the **loss decreases** for each feature over all trees gives an indication of feature importance

Intuitively an important feature is one that if split upon, it leads to a large reduction in the loss

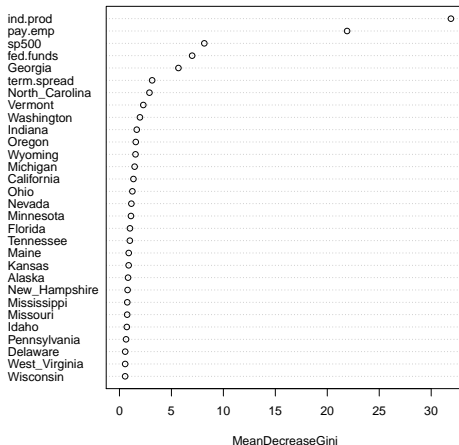
MEAN DECREASE VARIABLE IMPORTANCE

To recover some information, we can do the following:

1. For each of the B trees and each of the p features, we record the amount that the Gini index (or cross-entropy) is reduced by splitting on that feature
2. Report the average reduction over all B trees

This gives us an indication of the **importance** of a feature

MEAN DECREASE VARIABLE IMPORTANCE



OUT-OF-BAG SAMPLES (OOB)

One can show that, on average, drawing n samples from n observations with replacement results in about $2/3$ of the observations being selected.

The remaining one-third of the observations not used are referred to as **out-of-bag (OOB)**

OUT-OF-BAG SAMPLES (OOB)

We can think of it as a for-free **cross-validation**

The observations that aren't included serve as **test** data

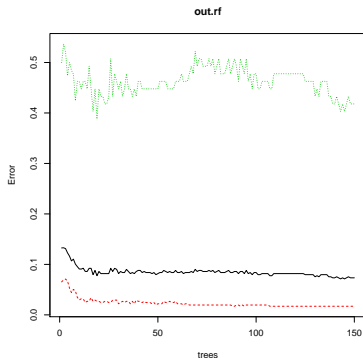
This provides a free estimate of prediction risk for each tree

We can therefore get an overall estimate of prediction risk by averaging these estimates over **all** bootstrapped trees

(Same idea applies here as for getting a prediction at a test X . For each X_i , we can take a majority vote of all the classifications when (X_i, Y_i) is OOB)

OUT-OF-BAG SAMPLES (OOB)

We can use the OOB samples to choose the number of trees B to consider



As we are taking an average, we can iteratively compute small batches, stopping when OOB error rate stabilizes

PERMUTATION VARIABLE IMPORTANCE

Consider the b^{th} bootstrap sample

1. The OOB prediction accuracy is recorded
2. Then, the j^{th} feature is randomly permuted in the OOB samples
3. The prediction error is recomputed and the change in prediction error is recorded

INTUITION: If a feature is highly important, then the OOB prediction error should increase substantially after permuting the OOB values for that feature

PROXIMITY

Choose any two observations on the training data: i, i'

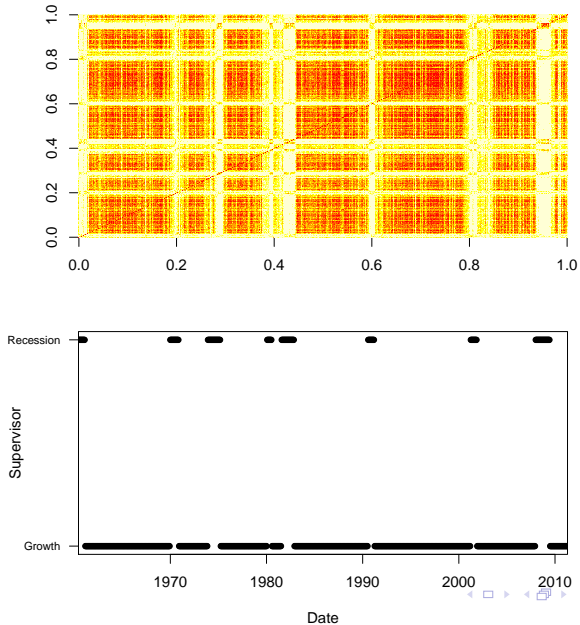
We can record

$$\text{proximity}(i, i') = \frac{\# \text{ times } i, i' \text{ are in the same leaf}}{\# \text{ times } i, i' \text{ occur in same tree}}$$

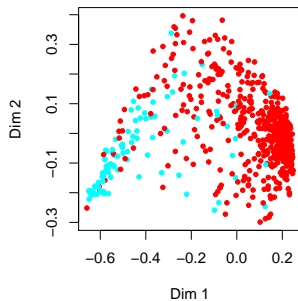
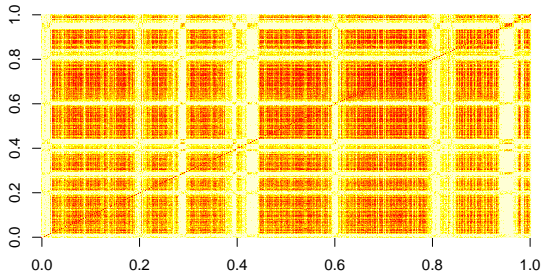
Values near 1 indicate "close" observations and values near 0 indicate "far" observations

What can we do with these proximities?

PROXIMITY



PROXIMITY



PROXIMITY

```
#First image
out.rf = randomForest(X,Y,proximity=TRUE)
par(mfrow=c(2,1),mar=c(4,4,1.2,4))
image(1 - out.rf$proximity)
plot(dates, Y,
      xlab='Date', ylab='Supervisor')
dev.off()
```

```
#Second image
out.rf = randomForest(X,Y,proximity=TRUE)
par(mfrow=c(2,1),mar=c(4,4,1.2,4))
image(1 - out.rf$proximity)
MDSplot(out.rf,Y,palette=rainbow(2))
```


Random Forest

RANDOM FOREST

Random Forest is a small extension of Bagging, in which we attempt to **decorrelate** the bootstrap trees

IDEA: Draw a bootstrap sample and start to build a tree

- At each split, we randomly select m of the possible p features as candidates for the split.
- A new sample of size m of the features is taken at each split.

Usually $m = \sqrt{p}$ for classification and $p/3$ for regression
(this would be 7 out of 56 features for GDP data)

In other words, at each split, we aren't even allowed to consider the majority of possible features!

RANDOM FOREST

What is going on here?

Suppose there is 1 really strong feature and many mediocre ones.

- Then each tree will have this one feature in it,
- Therefore, each tree will look very **similar** (i.e. highly correlated).
- Averaging highly correlated things leads to much less variance reduction than if they were uncorrelated.

If we don't allow some trees/splits to use this important feature, each of the trees will be much less similar and hence much less correlated.

Bagging is Random Forest when $m = p$

(That is, when we can consider all the features at each split)

RANDOM FOREST

An average of B uncorrelated random variables has variance

$$\frac{\sigma^2}{B}$$

An average of B random variables has variance

$$\rho\sigma^2 + \frac{(1 - \rho)\sigma^2}{B}$$

for correlation ρ

As $B \rightarrow \infty$, the second term goes to zero, but the first term remains

Hence, correlation of the trees limits the benefit of averaging

RANDOM FOREST

Another way to decorrelate the trees is by introducing **noise features**

Generate a few new features (say $0.01p$) that are not related to the supervisor

In some bootstrap samples, this feature will be included in the tree, adding a decorrelating effect

RANDOM FOREST: BIAS AND VARIANCE

With either approach, we are trading **bias** and **variance** again

Bagging has the same bias as the underlying procedure, but may not get much variance reduction

Random Forest is biased due to subsampling/noise features, but gets more variance reduction by decreasing ρ

(recall that the variance is $\rho\sigma^2 + \frac{(1-\rho)\sigma^2}{B}$)

Example: Recession data

TREE RESULTS: CONFUSION MATRICES

			Truth		Mis-Class
			Growth	Recession	
Our Preds	NULL	Growth	111	26	18.9%
		Recession	0	0	
	TREE	Growth	99	3	10.9%
		Recession	12	23	
	RANDOM FOREST	Growth	102	5	10.2%
		Recession	9	21	
	BAGGING	Growth	104	3	7.3%
		Recession	7	23	

TREE RESULTS: SENSITIVITY & SPECIFICITY

	Sensitivity	Specificity
NULL	0.000	1.000
TREE	0.884	0.891
RANDOM FOREST	0.807	0.918
BAGGING	0.884	0.936

OUT-OF-BAG ERROR ESTIMATION FOR BAGGING

		Truth		Miss-Class
		Growth	Recession	
OOB BAGGING	Growth	401	9	6.71%
	Recession	23	44	
TEST BAGGING	Growth	104	3	7.3%
	Recession	7	23	

RANDOM FOREST IN R

```
require(randomForest)
out.rf = randomForest(X,Y,importance=TRUE,mtry=ncol(X))
class.rf = predict(out.rf,X_0)
```

NOTES:

- The **importance** statement tells it to produce the variable importance measures
- the **mtry = ncol(X)** tells **randomForest** to consider all the features at each split

(This particular choice corresponds to bagging. Leaving this out uses the default \sqrt{p})

- **randomForest** also supports formulae

```
out.rf = randomForest(Y~.,data=X)
```

However, it can take much longer to run

RANDOM FOREST IN R

Call:

```
randomForest(x = X, y = Y, mtry = ncol(X), importance=T)
```

```
      Type of random forest: classification
```

```
      Number of trees: 500
```

```
No. of variables tried at each split: 56
```

```
      OOB estimate of  error rate: 7.17%
```

Confusion matrix:

```
      0  1 class.error
```

```
0 401  9  0.02195122
```

```
1  25 42  0.37313433
```

RANDOM FOREST IN R

```
> head(importance(out.rf,type=1))#Permutation
      MeanDecreaseAccuracy
Alabama          3.7277511
Alaska           1.7941463
Arizona          2.9659623
Arkansas         -0.8341577
California       7.2973572
> head(importance(out.rf,type=2))#Mean decrease
      MeanDecreaseGini
Alabama          0.4551073
Alaska           1.6440170
Arizona          0.7025527
Arkansas         0.3503138
California       1.4616203
#variable importance plot:
varImpPlot(out.rf,type=2)
```

Missing data

MISSING DATA/IMPUTATION

RANDOM FOREST provides a method for (supervised) imputation

It follows two steps:

1. **na.roughfix**: uses either the median or mode to impute missing values
$$X \leftarrow \text{na.roughfix}(X)$$
2. **rf.impute**: Gets the proximity matrix, and re-computes the imputation
 - ▶ For numeric features, it uses weighted (with respect to proximity) average
 - ▶ For categorical features, it uses the category that maximizes the average proximity

(Both of these only use the original, non-missing observations)

MISSING DATA/IMPUTATION

```
require(randomForest)
x = rnorm(12)
xNA = x
xNA[sample(12,2,replace=F)] = NA
X = matrix(xNA,nrow=6)
Y = matrix(x,nrow=6) %*% c(1,.5) + rnorm(6)
XnewImp = rfImpute(X,Y)
#First column is the supervisor vector
Xnew      = XnewImp[,-1]
```


MISSING DATA/IMPUTATION

```
> X
```

```
      NA -1.2589350282
0.6202015 0.1780122216
-0.9340213 -0.6483047015
0.1142546 0.0489260332
-1.1039581      NA
0.2064204 0.0007548357
> matrix(x,nrow=6)
0.1485857 -1.2589350282
0.6202015 0.1780122216
-0.9340213 -0.6483047015
0.1142546 0.0489260332
-1.1039581 -0.2468085986
0.2064204 0.0007548357
```

```
> Xnew
```

```
0.1287655 -1.2589350282
0.6202015 0.1780122216
-0.9340213 -0.6483047015
0.1142546 0.0489260332
-1.1039581 -0.2015763074
0.2064204 0.0007548357
```