Randomization methods: Bagging

-Statistical Machine Learning-

Lecturer: Darren Homrighausen, PhD

Notation

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

That is, we form

$$\hat{Y} = \hat{f}(X)$$

where

- $\hat{f}$ 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: The main idea

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

What is the variance of

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

Bagging: The main idea

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

What is the variance of

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

Answer: σ^2/n.

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

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

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.

Let's look at an example.

Bootstrap detour

Bootstrap detour

Suppose we are looking to invest in two financial instruments, X and Y. The return on these investments is random, but we still want to allocate our money in a risk minimizing way.

That is, for some $\alpha \in (0, 1)$, we want to minimize

$$\mathrm{Var}(\alpha X + (1 - \alpha)Y)$$

The minimizing α is:

$$\alpha_* = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

(Here, σ_{XY} is the covariance between X and Y)

Bootstrap detour

We can estimate α_* via a plug-in estimator

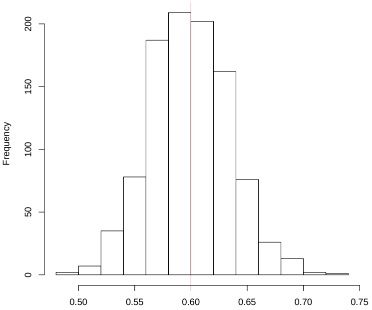
$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}^2}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}^2}$$

Now that we have an estimator of α_*, it would be nice to have an estimator of its variability.

In this case, computing a standard error is difficult.

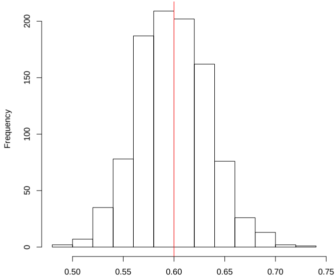
Bootstrap detour

Suppose for a moment that we can simulate a large number of draws (say 1000) of the data, which has actual value $\alpha = 0.6$. Then we could get estimates $\hat{\alpha}_1, \ldots, \hat{\alpha}_{1000}$:



This is the sampling distribution of $\hat{\alpha}$

Bootstrap detour



The mean of all of these is:

$$\overline{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.599,$$

which is very close to 0.6 (red line), and the standard error is

$$\sqrt{\frac{1}{1000 - 1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \overline{\alpha})^2} = 0.079$$

Bootstrap detour

The standard error of 0.035 gives a very good idea of the accuracy of $\hat{\alpha}$ for a single sample.

Roughly speaking, for a new random sample, we expect

$$\hat{\alpha} \in (\alpha - 2 * 0.079, \alpha + 2 * 0.079) = (0.442, 0.758)$$

In practice, of course, we cannot use this procedure as it relies on being able to draw a large number of (independent) samples from the same distribution as our data.

This is where the bootstrap comes in.

We instead draw a large number of samples directly from our observed data. This sampling is done with replacement, which means that the same data point can be drawn multiple times.

Bootstrap detour: Small example

Suppose we have data $\mathcal{D} = (4.3, 3, 7.2, 6.9, 5.5)$.

Then we can draw bootstrap samples, which might look like:

$$\begin{aligned}
\mathcal{D}_1^* &= (7.2, 4.3, 7.2, 5.5, 6.9) \\
\mathcal{D}_2^* &= (6.9, 4.3, 3.0, 4.3, 6.9) \\
&\vdots \\
\mathcal{D}_B^* &= (4.3, 3.0, 3.0, 5.5, 6.9)
\end{aligned}$$

It turns out each of these $\mathcal{D}_b^*$ have very similar properties as $\mathcal{D}$

Bootstrap detour: Small example

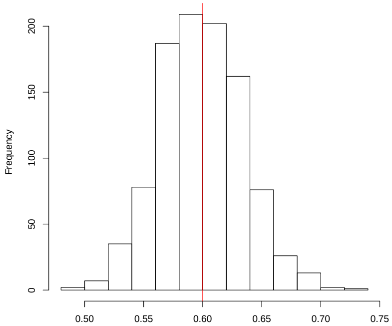
Now, we form the bootstrap mean:

$$\text{mean}_B = \frac{1}{B}\sum_{b=1}^{B} \hat{\alpha}_b^*$$

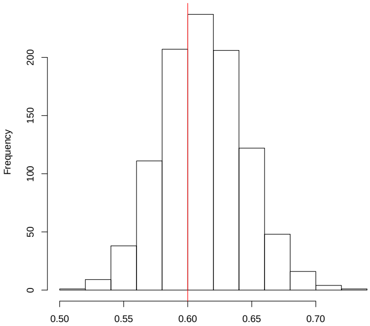
The bootstrap estimator of the standard error is:

$$\text{SE}_B = \sqrt{\frac{1}{B}\sum_{b=1}^{B} \left(\hat{\alpha}_b^* - \text{mean}_B\right)^2}$$

Bootstrap detour



Sampling distribution of $\hat{\alpha}$ (impossible to form)



Bootstrap distribution of $\hat{\alpha}$ (possible to form)

Bootstrap: End detour

Summary:

Suppose we want to get an idea of the sampling distribution of some statistic $\hat{f}$ trained on $\mathcal{D}$.

Then we do the following: Fix a large number B

(B could be, say, 1000)

Then for each $b = 1, \ldots, B$

1. Form a new bootstrap draw from $\mathcal{D}$, call it $\mathcal{D}^*$
2. Compute $\hat{f}_b^*$ from $\mathcal{D}^*$

Now, we can estimate the distribution of $\hat{f}$ trained on $\mathcal{D}$ by looking at the distribution of the B draws, $\hat{f}_b^*$

End detour

Bagging: The bootstrap part

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

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

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

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

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 $b = 1, \ldots, B$, grow an unpruned tree on the b^{th} bootstrap draw from the data.
3. Average all these trees together.

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

Bagging trees

The procedure for trees is the following

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

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

Therefore averaging many trees results in an estimator that has lower variance and still low bias.

Bagging trees in classification

For classification, there are a few sensible methods for aggregation

For each test observation X,

- record the length B vector $[\hat{f}_1^*(X), \ldots, \hat{f}_B^*(X)]^\top$ 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

This means 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

(Note that these ideas apply to bagging any low bias procedure, not just to 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

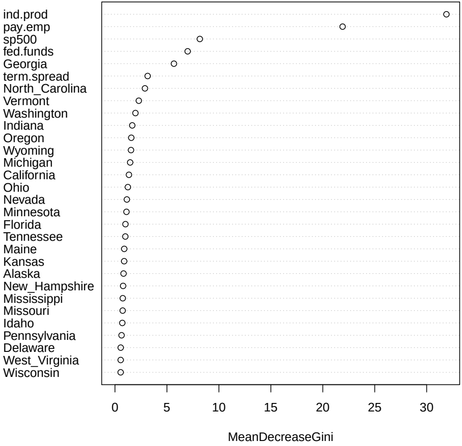
Mean decrease variable importance

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

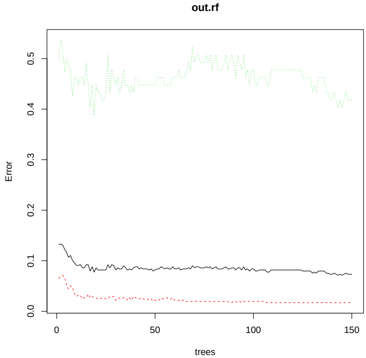
This provides a free estimate of prediction risk for each tree

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

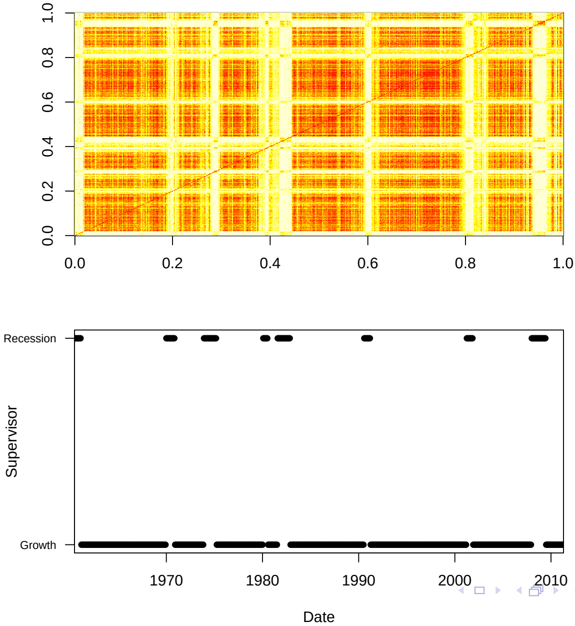
Choose any two observations on the training data: i, i'

We can record

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

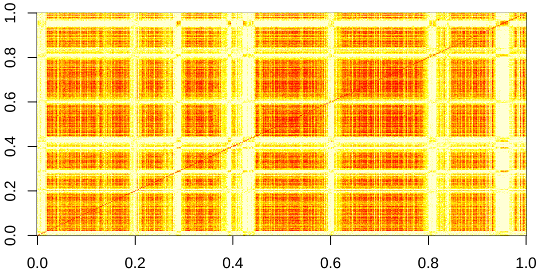
Values near 1 indicate "close" observations and values near 0 indicate "far" observations

PROXIMITY

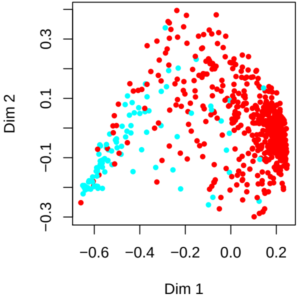


0.0
0.2
0.4
0.6
0.8
1.0
Recession
Supervisor
Growth
1970
1980
1990
2000
2010
Date

Proximity



0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0



Dim 2
0.3
0.1
−0.1
−0.3
−0.6
−0.4
−0.2
0.0
0.2
Dim 1

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

Evaluating Classifications

Sensitivity and specificity

Sensitivity: The proportion of times we label recession, given that recession is the correct answer.
Specificity: The proportion of times we label no recession, given that no recession is the correct answer.

We can think of this in terms of hypothesis testing. If

$$H_0 : \text{ no recession},$$

then

Sensitivity: $P(\text{reject } H_0 | H_0 \text{ is false})$, [1 - P(Type II error)]
Specificity: $P(\text{accept } H_0 | H_0 \text{ is true})$, [1 - P(Type I error)]

Confusion matrix

We can report our results in a matrix:

		Truth	
		Recession	No Recession
Our	Recession	(A)	(B)
Predictions	No Recession	(C)	(D)

The total number of each combination is recorded in the table.

The overall miss-classification rate is

$$\frac{(B) + (C)}{(A) + (B) + (C) + (D)} = \frac{(B) + (C)}{\text{total observations}}$$

What is the sensitivity/specificity?

Confusion matrix

We can report our results in a matrix:

		Truth	
		Recession	No Recession
Our	Recession	(A)	(B)
Predictions	No Recession	(C)	(D)

The total number of each combination is recorded in the table.

The overall miss-classification rate is

$$\frac{(B) + (C)}{(A) + (B) + (C) + (D)} = \frac{(B) + (C)}{\text{total observations}}$$

What is the sensitivity/specificity?

(Sensitivity is (A)/[(A) + (C)], Specificity is (D)/[(B) + (D)])

Tree results: Confusion matrices

			Truth		
			Growth	Recession	Mis-Class
Our Preds	Null	Growth	111	26	
		Recession	0	0	18.9%
	Tree	Growth	99	3	
		Recession	12	23	10.9%
	Random Forest	Growth	102	5	
		Recession	9	21	10.2%
	Bagging	Growth	104	3	
		Recession	7	23	7.3%

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		
		Growth	Recession	Miss-Class
OOB Bagging	Growth	401	9	
	Recession	23	44	6.71%
Test Bagging	Growth	104	3	
	Recession	7	23	7.3%

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/imputation

In practice, there will often be missing data

Estimating this missing data is known as imputation

Random Forest provides a method for 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)
Xnew = rfImpute(X,Y)
```

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
```

Additional random forest topics

Bagging theory

Suppose we are using random forest for regression

The bootstrap estimator is:

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

For bootstrap trees:

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

To see what is really going on, we need some bootstrap theory

Bootstrap theory

Suppose we have i.i.d. observations Z_i from $\mathbb{P}$

(Call the data $\mathcal{D}$)

Suppose we want to know the distribution of some procedure that is a function of $\mathcal{D}$, $\hat{\theta}(\mathcal{D})$

$$\mathbb{P}(\hat{\theta}(\mathcal{D}) \in A)$$

This is unknown and can be estimated with a plug-in estimator of the measure $\mathbb{P}$

- Parametric:

$$\mathbb{P}_{\hat{\theta}}(\hat{\theta}(\mathcal{D}) \in A)$$

- Nonparametric:

$$\hat{\mathbb{P}}(\hat{\theta}(\mathcal{D}) \in A)$$

Bootstrap theory

In either case, computing with respect to $\mathbb{P}_{\hat{\theta}}$ or $\hat{\mathbb{P}}$ is commonly infeasible but sampling from them is easy

Example: Estimate the statistical functional $\theta(\mathbb{P}) = \int z d\mathbb{P}(z) = \mathbb{E}Z$ with the plug-in estimator $\theta(\hat{\mathbb{P}}) = \overline{Z}$.

Compute the mean of $\theta(\hat{\mathbb{P}})$ w.r.t $\hat{\mathbb{P}}$:

$$\hat{\mathbb{P}}\theta(\hat{\mathbb{P}})$$

Let $\mathcal{D} = \{4, 10\}$. Then there are 4 possibilities (in general n^n)

$$\hat{\mathbb{P}}\theta(\hat{\mathbb{P}}) = \frac{1}{4}\left(\frac{1}{2}(4+4) + \frac{1}{2}(10+4) + \frac{1}{2}(4+10) + \frac{1}{2}(10+10)\right) = 7$$

Bootstrap theory

Due to the extremely fast growth of n^n or intractability of $\mathbb{P}_{\hat{\theta}}$, we need to approximate the distribution

$\rightarrow$ replace them with a Monte-Carlo approximation

Draw $\mathcal{D}^* = Z_1^*, Z_2^*, \ldots, Z_B^*$ from $\hat{\mathbb{P}}$ or $\mathbb{P}_{\hat{\theta}}$:

$$\hat{\mathbb{P}}(\hat{\theta}(\mathcal{D}) \in A) \text{ replace with } \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}(\hat{\theta}(\mathcal{D}^*) \in A)$$

Important question: How good is this approximation?

Bootstrap theory

Write $\mathbb{P}^*$ as the bootstrap distribution.

Suppose we want to show some result about

$$\sqrt{n}(\theta(\mathbb{P}^*) - \theta(\mathbb{P})) = \sqrt{n}(\theta(\mathbb{P}^*) - \theta(\hat{\mathbb{P}})) + \sqrt{n}(\theta(\hat{\mathbb{P}}) - \theta(\mathbb{P}))$$

Example: If $\mathbb{P}Z^2 < \infty$, then

$$\sqrt{n}(\theta(\hat{\mathbb{P}}) - \theta(\mathbb{P})) = \sqrt{n}(\overline{Z} - \mu) \xrightarrow{D} N(0, \sigma^2)$$

It turns out that

$$\sqrt{n}(\overline{Z}^* - \overline{Z}) \xrightarrow{D} N(0, \sigma^2)$$

as well

Additional random forest topics

Claim: Random forest cannot overfit.

This is and isn't true. Write

$$\hat{f}_{rf}^{B}(X) = \frac{1}{B}\sum_{b=1}^{B} T(X;\Theta_b)$$

(Θ_b are all the feature split points and terminal node values of b^{th} tree)

Increasing B does not cause Random forest to overfit, rather removes the Monte-Carlo-like approximation error

$$\hat{f}_{rf}(X) = \mathbb{E}_{\Theta} T(X,\Theta) = \lim_{B\to\infty} \hat{f}_{rf}^{B}(X)$$

However, this limit can overfit the data, the average of fully grown trees can result in too complex of a model

(Note that Segal (2004) shows that a small benefit can be derived by stopping each tree short, but thus induces another tuning parameter)

ADDITIONAL RANDOM FOREST TOPICS

- Leo Breiman advocated for a "linear combination" random forest. At each node, random linear combinations of subselected features with weights in $[-1, 1]$ are split upon. He found this approach improved upon the standard random forest.
- Introduction of noise features improves performance
- Using subsampling instead of bootstrap to generate trees
- Adaptive nearest neighbors via proximity scores