Bootstraps, permutation tests, and cross-validation

Joe Felsenstein
The bootstrap

estimate of $\theta$  (unknown) true value of $\theta$

empirical distribution of sample (unknown) true distribution
The bootstrap

estimate of $\theta$ \rightarrow (unknown) true value of $\theta$

empirical distribution of sample

Bootstrapped replicate

(unknown) true distribution
The bootstrap

estimate of $\theta$ (unknown) true value of $\theta$

empirical distribution of sample (unknown) true distribution

Bootstrap replicates
The bootstrap

- Estimate of $\theta$ to (unknown) true value of $\theta$
- Empirical distribution of sample to (unknown) true distribution
- Bootstrap replicates
- Distribution of estimates of parameters

Bootstrap, permutation tests, and cross-validation – p.5/14
Bootstrap sampling

To infer the error in a quantity, \( \theta \), estimated from a sample of points \( x_1, x_2, \ldots, x_n \) we can

- Do the following \( R \) times (\( R = 1000 \) or so)
Bootstrap sampling

To infer the error in a quantity, $\theta$, estimated from a sample of points $x_1, x_2, \ldots, x_n$ we can

- Do the following $R$ times ($R = 1000$ or so)
- Draw a “bootstrap sample” by sampling $n$ times with replacement from the sample. Call these $x_1^*, x_2^*, \ldots, x_n^*$. Note that some of the original points are represented more than once in the bootstrap sample, some once, some not at all.
Bootstrap sampling

To infer the error in a quantity, $\theta$, estimated from a sample of points $x_1, x_2, \ldots, x_n$ we can

- Do the following $R$ times ($R = 1000$ or so)
- Draw a "bootstrap sample" by sampling $n$ times with replacement from the sample. Call these $x^*_1, x^*_2, \ldots, x^*_n$. Note that some of the original points are represented more than once in the bootstrap sample, some once, some not at all.
- Estimate $\theta$ from the bootstrap sample, call this $\hat{\theta}^*_k$ ($k = 1, 2, \ldots, R$)
Bootstrap sampling

To infer the error in a quantity, $\theta$, estimated from a sample of points $x_1, x_2, \ldots, x_n$ we can

- Do the following $R$ times ($R = 1000$ or so)

- Draw a “bootstrap sample” by sampling $n$ times with replacement from the sample. Call these $x_1^*, x_2^*, \ldots, x_n^*$. Note that some of the original points are represented more than once in the bootstrap sample, some once, some not at all.

- Estimate $\theta$ from the bootstrap sample, call this $\hat{\theta}_k^*$ ($k = 1, 2, \ldots, R$)

- When all $R$ bootstrap samples have been done, the distribution of $\hat{\theta}_i^*$ estimates the distribution one would get if one were able to draw repeated samples of $n$ points from the unknown true distribution.
Doing the bootstrap in R

To do bootstrapping in R we can use the `sample` function with `replace=TRUE`:

For example, to get a good idea of the distribution of means for the distribution from which vector `a` was drawn:

```r
m <- length(a)  # a is the vector, m how long it is
b <- rep(a, times=100)  # repeat a end to end 100 times
d <- array(b, c(m,100))  # make it into a 100-column array
    # (this puts a down each column)
samp <- function(x){return(sample(x, m, replace=TRUE))}  # samp is a wrapper function around sample
e <- apply(d, 2, samp)  # apply samp to each column
f <- apply(e, 2, mean)  # apply mean to each column of that
    # (f is an array of 100 means)
```
An example: means of 100 uniform variates

![Histogram of means](image)
Bootstrap intervals and tests

You can make confidence intervals on quantities such as the mean by taking the upper and lower 2.5% points of the bootstrap distribution of the quantity.

You can do tests by seeing whether null hypothesis values of the parameter fall in that confidence limit.
Permutation tests

Given a set of data points in two samples, under the null hypothesis any of the $m + n$ points could have been in any of the samples.

So all permutations of the points, shuffling them among samples, are equally likely (under the null hypothesis).

Does our sample show more difference than expected, among all these shuffles?

Here’s how we test:

- Compute the difference of means (or some other reasonable statistic) between the two groups.
- Make a large number of random shufflings of the points.
- For each, compute this statistic.
- See whether, out of (say) 9,999 shuffles, when the true value is added in, it is in the top 5% (the top 500) of these 10,000.

Note that this test does not assume normality, just that the points are drawn from the same (unknown) distribution, independently.
Permutation tests

There are many variations on permutation tests:

- If the test is a paired test, to see whether the mean difference is zero, shuffle within each pair (i.e. flip each pair the other way with probability 50%)

- If it is a regression, and if the $Y$ points are randomly associated with the $X$ points under the null hypothesis, so that the true slope is zero, we can shuffle $Y$s, associating them with the $X$s at random. Each time, we compute the slope.
How to do a permutation test in R

Given two samples called \( a \) and \( b \):

\[
\text{mean}(a) - \text{mean}(b) \quad \# \text{ the original difference of means}
\]
\[
m \leftarrow \text{length}(a) \quad \# \text{ get the sample size as } m, n
\]
\[
n \leftarrow \text{length}(b)
\]
\[
d \leftarrow c(a, b) \quad \# \text{ make a long vector with both}
\]
\[
e \leftarrow \text{sample}(d, m+n, \text{replace}=\text{FALSE}) \quad \# \text{ shuffle them}
\]
\[
g \leftarrow e[1:m] \quad \# \text{ take the first } m
\]
\[
h \leftarrow e[m+1;m+n] \quad \# \ldots \text{ and the next } n
\]
\[
\text{mean}(g) - \text{mean}(h) \quad \# \text{ and compute the mean difference}
\]

Actually, you want to make a whole array whose columns (or rows) are each a shuffle of your sample, a compute the mean differences for each column (row).
**Cross-validation**

If we fit a polynomial of degree \( n - 1 \) to \( n \) points, we can always have the curve pass through all the points:

But how well do they really fit? They might predict a new point rather badly.
Cross-validation

Cross-validation removes one point at a time, then fits to the remaining points, then sees how well the removed point is fit:

This gives a good idea of how large the error of prediction actually is.