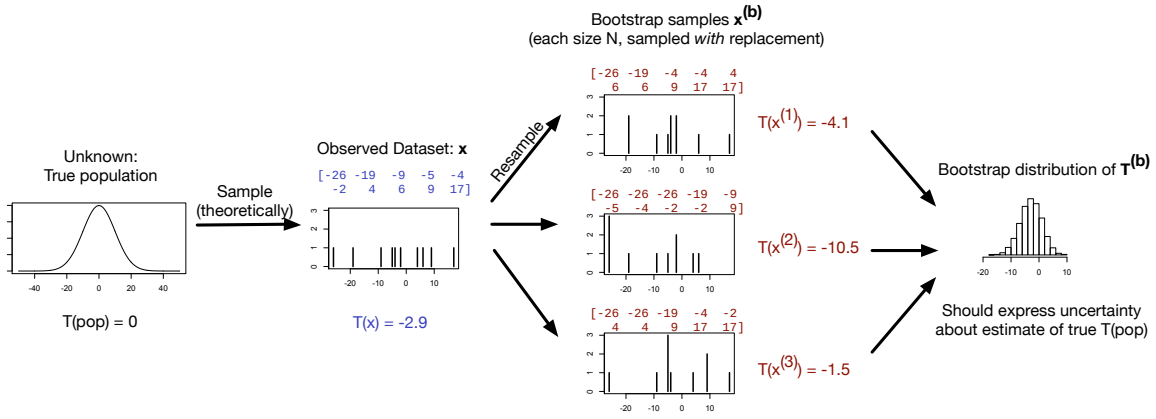
Bootstrapping

Brendan O'Connor

November 15, 2018



1 Bootstrap resampling

Assume our observed data, $x_1..x_n$, was drawn from some underlying true population that we don't have access to. We'd like to estimate some quantity about the true population. We could calculate it from our dataset. But do we have enough data to be *confident* that our dataset accurately represents the true population?

Bootstrapping, a.k.a. bootstrap methods, are a family of sampling-based procedures that can *estimate uncertainty* for a wide range of estimators and models. They're a way to tell us how much we *don't* know.

Say we'd like to estimate a function $T(\text{population})$, where, for example, T could be a simple summary statistic, like mean or median, or something complicated, like a decision tree learning algorithm. Consider the case of just the mean. The strategy for bootstrapping is to *resample* the data, which simulates the variability we might get if we were to draw a new sample from the underlying true population.

In the above sample, we have a dataset x of 10 numbers. To draw a bootstrap sample, we take n random choices of items from x, sampled *with replacement*, into a new bootstrap sample $x^*_1, ..x^*_n$. Then we calculate our function on this data, $T(x^*)$, and store the result. We repeat this procedure a large number, say $B = 10,000$, times. Call each bootstrap data sample $x^{(b)}$. We then have a distribution of B different $T(x^{(b)})$ values, which itself has some variance. There are a variety of ways to calculate confidence intervals or hypothesis tests from it.

The bootstrap sampling process is:

- Input: dataset $x_1..x_n$
- For each $b = 1..B$,

- Draw $x_1^{(b)}..x_n^{(b)}$ with replacement from $x_1..x_n$. This is called a "bootstrap sample." Specifically:
  - For each $i = 1..n$, let $x_i^{(b)} = x_j$ where $j \sim \text{Unif}([1, 2, ..., n])$
- Calculate (and save) $T^{(b)} = T(x^{(b)})$.

The diagram above shows an example of bootstrapping the sample mean function, $T(x) = \frac{1}{n}\sum_i x_i$. For simulation purposes, we have a true mean of the population was 0, and the dataset, consisting of 10 data points, was -2.9. However, after bootstrapping 10,000 times, it turns out the confidence of that estimate is quite low: resampled datasets could range from much smaller to much larger values.

It's fine to have a particular item from the dataset not appear in a particular bootstrap sample. In fact, this happens quite a bit of the time—roughly a third of the dataset won't appear in a particular bootstrap sample, and others may appear multiple times. This simulates variability in what could be drawn from the population—you saw certain items, but if you were to redraw a whole new dataset, perhaps you wouldn't see the all the same or similar items again.

The key idea is: let F is the underlying population distribution, and assume the observed dataset is a sample from it: $x_1..x_n \sim F$. Let $\hat{F}$ be the *empirical* distribution defined by the observed dataset; it has a mean, a variance, a cumulative distribution function, etc. We then draw bootstrap samples $x_1^*..x_n^* \sim \hat{F}$. The key theoretical justification for the bootstrap is that sampling from $\hat{F}$ gives a similar distribution as sampling from F.

From this bootstrap distribution of T, we'd like to calculate a *confidence interval*: a range of possible values for T where we expect the *true*, unknown, value $T(\text{population})$ to be within. A confidence interval requires a confidence level, C; for example, we will use $C = 0.95$ to get a 95% confidence interval. The desired property of a $C\%$ confidence interval is that, if we were to apply the whole entire method again on datasets we encounter in the future, that $C\%$ of the time, the true value will be within the confidence interval we infer.[1]

2 Bootstrap percentile interval

A very intuitive method to calculate a CI is to use the 2.5th and 97.5th percentiles of the bootstrap distribution to define an interval—so 95% of the bootstraps yielded a value of T within this range. This method, called the *bootstrap percentile interval*, thus calculates, for desired confidence level C,

$$CI(C) = \left[Q\left(\frac{C}{2},\ T^{(1)}..T^{(B)}\right),\ Q\left(1-\frac{C}{2},\ T^{(1)}..T^{(B)}\right)\right]$$

where $Q(p, z_1..z_n)$ is the quantile or percentile function (a.k.a. inverse CDF), returning the z^* where $p\%$ of the of the items have a smaller value than it; specifically, the z^* where $p = \frac{1}{n}\sum_i 1\{z_i \leq z^*\}$.[2]

The bootstrap distribution looks a little bit like Bayesian methods—if we were doing this with a Bayesian posterior, we'd call it a *credible interval*. However, the bootstrap does not require a Bayesian prior, or even a probabilistic model at all. It only requires an algorithm to implement T; you can use weird things like the median, or the output of a graph search algorithm, or whatever.

The bootstrap percentile method is less theoretically secure than some other methods of calculating uncertainty via the bootstrap, but it's the most intuitive, and is commonly used.

[1] Some people point out the term *uncertainty interval* is a much better term for a CI: the more uncertainty you have, the bigger this interval is.

[2] Detail: there are different ways to define this if p does not neatly divide into $1/n$.

3 Bootstrap normal interval

There are a variety of other methods to infer confidence intervals from bootstrap samples; another simple one is to calculate the *bootstrap standard error*:

$$\sigma_{boot} = \sqrt{\frac{1}{B}\sum_{b}^{B}\left(T^{(b)} - \frac{1}{B}\sum_{b'} T^{(b')}\right)}$$

which is simply the standard deviation of $T^{(b)}$, that is, the square root of the average squared distance from the mean. This can be used to construct confidence intervals as $T(x) \pm z_{(1-C)/2}\sigma_{boot}$ where z is the standard normal (complementary) quantile function; for example, for $C = .95$, $z = 1.96$. This works well if $T^{(b)}$ is close to a normal distribution. The style of these intervals is heavily related to the central limit theorem.[3]

There are many other methods for bootstrap CIs; the *R* package *boot* implements a number of them. It is not clear to me which one is best.

4 How many samples?

As you draw more and more bootstrap samples, your estimate of the CI will converge to some particular value (due to the law of large numbers). If you don't draw enough samples, you'll notice that the bootstrap's results are unstable. A simple check is to run it a few times and see if results differ by much.

For example, I tried drawing 100 bootstrap samples from the above dataset. I got a 95% interval of $[-10.4, 4.0]$. But if I do it again, I get $[-9.5, 4.1]$. That's a bit different. Therefore I need to increase the number of bootstrap samples—say, increase it an order of magnitude. It's usually good to use only 100 samples when implementing and debugging a bootstrap algorithm, but when getting an analysis you want to trust, you should use at least 10,000. If I'm reporting results for a published research paper, I prefer to use 100,000 if possible.

5 Comparison to analytic frequentist methods

A wide variety of confidence interval methods are based on analytic or closed-form calculations. For example, a model's inverse Fisher information can be applied to the central limit theorem to derive a confidence interval; these or other theoretical principles have been applied to a wide variety of situations, resulting in methods like the t-test, z-test, ANOVA, chi-square, etc. These methods are great, much faster than the bootstrap, and run nicely out of the box without having to fiddle with the number of bootstrap samples. However, they are specific to particular settings, and much less flexible; you can apply the bootstrap to a wide variety of different settings, like many different functions T, and it is (often? usually?) still valid.

6 Paired tests with the bootstrap

Assume you have paired samples (y_i, z_i) with, for example, the output predictions from two different ML models on the ith example in the test set. One has 86.1% accuracy, and other has 87.4%

[3] The above equation is from Wasserman's *All of Statistics*. Some sources instead calculate the squared distances from $T(x)$; I don't understand which is better, though they may yield similar under certain conditions.

accuracy. You'd like to know which is better. Your test set is only size 1000; since it's somewhat small, it might be case one model just got lucky. How much uncertainty is there in our estimate of model performance?

Berg-Kirkpatrick et al. (2012) describes a method to analyze this via paired bootstrap testing. The idea is that we can bootstrap *pairs* of predictions. This may give us a more fine-grained (higher statistical power) way of assessing the difference between the models, since their predictions might be correlated. For each bootstrap sample, T is the difference in performance between the models; then the distribution of $T^{(b)}$ describes the uncertainty in the estimate of the difference in model performance. If the 95% interval crosses zero, that indicates we are not certain which method is better.

7 Bagging: Bootstrapping the training data for ensembles

The above example deals with *evaluation* uncertainty by resampling a test set. But we might also have *training* uncertainty, about how well the training set represents properties of the underlying population. High-variance learning algorithms, like decision trees, can be quite unstable in this regard, since they rely on fine-grained correlations between features. Thus, if we train a decision tree for each bootstrap sample, this can yield very different decision trees each time, which might usefully range over the space of parameters. Here, the goal is not a summary statistic of the training data, but instead to average out the variance of the intermediate models.

Let $T(train, x^{(new)})$ be the prediction, for a new data point $x^{(new)}$, based on a decision tree learned from the training set *train*. If the model isn't regularized well, that prediction might be unstable from the selection of training data. Thus, bagging computes the prediction as the expectation of the bootstrap distribution of predictions, $\frac{1}{B}\sum_b T(train^{(b)}, x^{(new)})$.