STAT562 Lecture 8 The Bootstrap

Beidi Qiang

SIUE

Introduction

The Bootstrap is a widely applicable resampling method to quantify the uncertainty associated with a given estimator.

- Can be used to estimate the standard errors of model coefficients (for example in regression models)
- Can be easily applied to a wide range of statistical learning methods
- Especially useful when a measure of variability is otherwise difficult to obtain

Idea of Boostrap

- Recall the concept of population and sample, sample statistics, sampling distribution.
- The observed sample should be similar to the true population.
- So we repeatedly resample from the observed sample (pseudo-population) to mimic new samples from the population.
- We calculate the desired quantity based on each of the bootstrap datasets.
- We can then get the standard error and construct confidence interval based on the bootstrap estimates.

The Bootstrap schematic

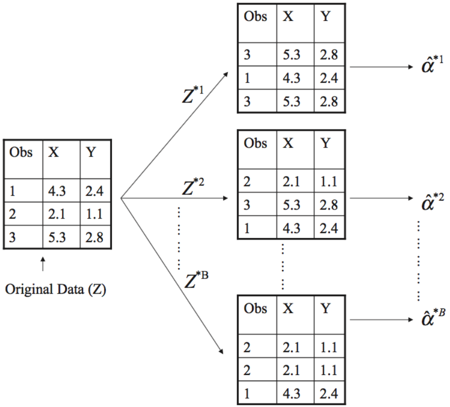

FIGURE 5.11. *A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α.*

Algorithm

In each iteration $b = 1, 2, \cdots, B$,

- generate bootstrap sample $(x_1^{(b)}, \cdots, x_n^{(b)})$ by resampling with replacement from the original data $(x_1, \cdots x_n)$.
- Compute the desired quantity $\hat{\theta}^{(b)}$ from the bootstrap sample.

$\hat{\theta}^{(1)} \cdots \hat{\theta}^{(B)}$ is approximately a sample from the sampling distribution of $\hat{\theta}$. And can be used to estimate $SE(\hat{\theta})$.

Comments

- We don't make any assumption of the population distribution. So bootstrap is completely non-parametric.
- The number of iteration B is usually a large number, so bootstrap is computationally demanding.
- If the original sample is not a good representative of the population, the bootstrap sample will not be either. Increasing B will make to bootstrap samples closer to the **empirical distribution** given by the observed sample , but not **the population distribution**.

Example: Bootstrap from a Poisson

```
set.seed(17)
x=rpois(10, lambda=2)
table(x)/10
x.uniq = unique(x)
prob0 = as.data.frame(table(x))[,2]/length(x)
m=1000
x.star= sample(x.uniq, size = m, replace = TRUE, prob = prob0)
table(x.star)/m
```

Bootstrap estimation of std. error

- Recall we generate bootstrap sample $(x_1^{(b)}, \cdots, x_n^{(b)})$ and compute the desired quantity $\hat{\theta}^{(b)}$ from each of the bootstrap sample.
- sample standard deviation of $\hat{\theta}^{(1)} \cdots \hat{\theta}^{(B)}$ is an estimate of the $SE(\hat{\theta})$. i.e.

$$\hat{SE}(\hat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}^{(b)} - \bar{\theta}^*)^2}, \text{ where } \bar{\theta}^* = mean(\hat{\theta}^{(1)} \cdots \hat{\theta}^{(B)})$$

- a choice of $B = 200$ is usually enough for estimating std. err, but it will take a larger B if seeking good estimation of confidence interval (Efron, 1993).

Example: Std .Err of Regression Coefficients

```
library(boot)

bs =function(formula, data, indices) {
d =data[indices,]
fit = lm(formula, data=d)
return(coef(fit)) }

results=boot(data = mtcars, statistic = bs, R =500, formula = mpg
wt + hp)
```

Bootstrap Confidence Intervals

- Based on the bootstrap estimation of std.error and CLT assumption, we have Standard Normal Bootstrap CI:

$$\hat{\theta} \pm Z_{\alpha/2}\hat{SE}(\hat{\theta})$$

- Based on the percentiles of the generated bootstrap estimates $\hat{\theta}^{(1)} \cdots \hat{\theta}^{(B)}$, we can get the percentile Bootstrap CI:

$$(\hat{\theta}_{\alpha/2}, \hat{\theta}_{1-\alpha/2})$$

- Another one based on the percentiles. basic Bootstrap CI:

$$(\hat{\theta} - (\hat{\theta}_{1-\alpha/2} - \hat{\theta}), \hat{\theta} + (\hat{\theta} - \hat{\theta}_{\alpha/2})) = (2\hat{\theta} - \hat{\theta}_{1-\alpha/2}, 2\hat{\theta} - \hat{\theta}_{\alpha/2})$$

Example: Logistic Regression

```
library(ISLR)
library(boot)

lr=function(data,indices){
d =data[indices,]
fit=glm(default balance+income,data=d,family="binomial")
return(coef(fit)) }

b=boot(Default,lr,100)
print(b)
summary(glm(default balance+income, data=Default,
family="binomial"))
plot(b,index=1)
plot(b,index=2)
boot.ci(b,index=1,type="perc")
boot.ci(b,index=2,type="norm")
```