ST509 Computational Statistics

Lecture 13: Bootstrap & Parallel Computing

Seung Jun Shin

Department of Statistics
Korea University

E-mail: sjshin@korea.ac.kr

Jackknife I

- Suppose $\hat{\theta}$ is an estimator based on an iid sample $Y_1, \cdots, Y_n$.
- Let $\hat{\theta}_{[i]}$ denote the "leave-1-out" estimator obtaining by computing $\hat{\theta}$ with Y_i deleted from the sample.
- Jackknife pseudo-values are defined by

$$\hat{\theta}_{ps,i} = n\hat{\theta} - (n-1)\hat{\theta}_{[i]}$$

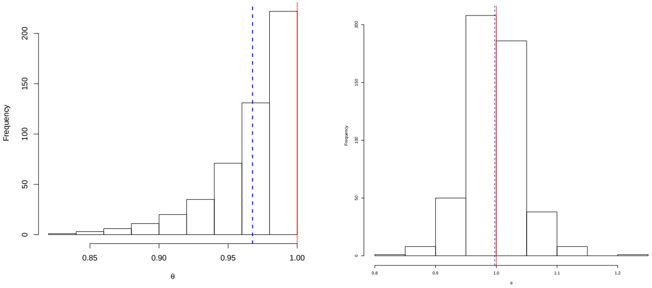
- Bias-adjusted jackknife estimator is

$$\hat{\theta}_J = \frac{1}{n}\sum_{i=1}^{n} \theta_{ps,i} = \hat{\theta} - (n-1)(\bar{\theta}_1 - \hat{\theta})$$

where $\bar{\theta}_1 = \frac{1}{n}\sum_{i=1}^{n} \hat{\theta}_{[i]}$

Jackknife II

- Illustration of the bias corrected version of the sample maximum $\hat{\theta}$ for $U_i \overset{iid}{\sim} (0,1)$. (i.e. $\theta = 1$)



(a) $\hat{\theta} = U_{(n)}$ (b) Bias-Corrected $\hat{\theta}$, $\hat{\theta}_J$

Figure: Histogram of variance estimator for 500 independent repetitions.

Jackknife III

- The jackknife variance estimator for $\hat{\theta}$ is given by

$$\hat{V}_J = \frac{(n-1)^2}{n} \frac{1}{n-1} \sum_{i=1}^{n} \left(\hat{\theta}_{[i]} - \bar{\theta}_1 \right)^2$$
$$= \frac{1}{n(n-1)} \sum_{i=1}^{n} \left(\hat{\theta}_{ps,i} - \hat{\theta}_J \right)^2$$

ex. For $\hat{\theta} = \bar{Y}_n$, jackknife variance estimator $\hat{V}_J$ of $\hat{\theta}$ is s^2_{n-1}/n, which is identical to usual variance estimator of $\bar{X}_n$.

Bootstrap I

- Bootstrap is a general technique for estimating unknown quantities associated with sampling distribution of estimators such as
  - Standard errors
  - Confidence intervals
  - p-values

Bootstrap II

- Suppose F is the true population distribution.
- We estimate the functional of F based on the the sample $X_1, \cdots, X_n$.

Ex Population expectation μ

$$\mu = E(X) = \int x f(x) dx \left(= \int x dF(x) \right)$$

can be estimated by the sample average $\bar{X}_n$:

$$\hat{\mu} = \bar{X}_n = \sum_{i=1}^{n} X_i \cdot \frac{1}{n} \left(= \int x dF_n(x) \right)$$

where $F_n(x)$ denotes the empirical distribution of $(X_1, \cdots, X_n)$,

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{x \leq X_i\}$$

Bootstrap III

- Underlying fundamentals of this idea is

$$F_n(x) \quad \rightarrow \quad F(x)$$

- Uncertainty/Randomness comes from

$$F_n(x) - F(x)$$

- Uncertainty quantification is not trivial since we only have single $F_n(x)$ for unknown $F(x)$.

Bootstrap IV

- Given a set of sample $X_1, \cdots, X_n$, a bootstrap sample denoted by $X_1^*, \cdots, X_n^*$ is a random drawing samples **with replacement** from $X_1, \cdots, X_n$.
- The idea of bootstrap is

$$F_n^*(x) \;\; \text{vs} \;\; F_n(x) \qquad \approx \qquad F_n(x) \;\; \text{vs} \;\; F(x)$$

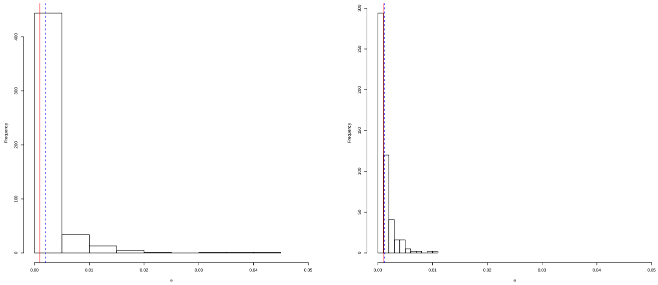
  where $F_n^*(x)$ denotes the empirical distribution of the bootstrap samples.
- We think the RHS as "real world" and the LHS as "bootstrap world"
- In the bootstrap world, we know both $F_n^*(x)$ and $F_n(x)$, and can obtain as many bootstrap samples as we want.

Illustration of Bootstrap method I

- Suppose we like to estimate θ from an iid sample $X_1, \cdots, X_n$ by $\hat{\theta}$.
- Bootstrap estimator of the variance of $\hat{\theta}$ can be obtained as follows.
  1. Generate a bootstrap sample $X_1^{*,b}, \cdots, X_n^{*,b}$ (i.e., random drawing from $X_1, \cdots, X_n$ with replacement).
  2. Compute estimator of θ based on the bootstrap sample $X_1^*, \cdots, X_n^*$, denoted by $\hat{\theta}^*$.
  3. Repeat Step 1-2 many, say B times. Then the bootstrap variance estimator is
  $$\widehat{\mathrm{Var}}_b(\hat{\theta}) = \frac{1}{B-1}(\hat{\theta}_b^* - \hat{\theta})^2.$$
- The idea is very general and can be applied to any estimator $\hat{\theta}$.

Illustration of Bootstrap method II

- Comparison of variance estimator for sample maximum $\hat{\theta}$ for $U_i \overset{iid}{\sim} (0,1)$. (i.e. $\theta = 1$)



(a) Jacknife

(b) Bootstrap

Figure: Histogram of variance estimator for 500 independent repetitions: Monte Carlo MSE is .00903 for the jackknife estimator and .00108 for the bootstrap estimator.

Illustration of Bootstrap method III

- The simplest way to construct the confidence interval of $\hat{\theta}$ is to just take the empirical $\alpha/2$ and $(1-\alpha/2)$ percentile (or quantile) from the bootstrap values $\theta_1^*, \cdots, \theta_B^*$.
- Percentile CI can be obtained as follows:
  1. Generate a bootstrap sample $X_1^{*,b}, \cdots, X_n^{*,b}$ (i.e., random drawing from $X_1, \cdots, X_n$ with replacement).
  2. Compute estimator of θ based on the bootstrap sample $X_1^*, \cdots, X_n^*$, denoted by $\hat{\theta}^*$.
  3. Repeat Step 1-2 many, say B times. Then the bootstrap percentile CI is

$$[q_b^*(\alpha/2), q_b^*(1-\alpha/2)]$$

     where $q_b^*(\alpha)$ denotes the αth sample quantile of $\hat{\theta}_1, \cdots, \hat{\theta}_B$.

Illustration of Bootstrap method IV

- Another early proposal is the reflected percentile interval obtained as follows:

$$\left\{\theta : q_b^*(\alpha/2) - \hat{\theta} \le \hat{\theta} - \theta \le q_b^*(1-\alpha/2) - \hat{\theta}\right\}$$

- The bootstrap-t interval (or percentile-t interval) is based on the empirical distribution of

$$t^* = \frac{\hat{\theta}^* - \hat{\theta}}{\hat{\sigma}^*}.$$

That is,

$$\left\{\theta : r_b^*(\alpha/2) \le \frac{\hat{\theta} - \theta}{\hat{\sigma}} \le r_b^*(1-\alpha/2)\right\}$$

where $r_b^*(\alpha)$ denotes the αth sample quantile of $t_1^*, \cdots, t_B^*$, and $\hat{\sigma}$ denotes some estimate of the standard deviation of $\hat{\theta}$.

Illustration of Bootstrap method V

- Fancier alternatives include
  - Bias-Corrected percentile interval (BC)
  - Bias-Corrected, accelerated interval (BC_a)
  - Double Bootstrap (Calibrated Percentile) Interval.

Illustration of Bootstrap method VI

- Using bootstrap, one can conduct hypothesis test.
- Suppose T_0 is the value of a test statistic T computed from an iid sample, $X_1, \cdots, X_n$.
- The bootstrap p-value is defined as

$$p_b = \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}\{T_b^* \geq T_0\}$$

where T_b^* is the statistic obtained from the bth bootstrap sample, $X_{1,b}^*, \cdots, X_{n,b}^*, b = 1, \cdots, B$. under an induced null hypothesis.

Illustration of Bootstrap method VII

```
> # two-sample mean comparison
> n <- 50;  B <- 500
> mu1 <- 0; mu2 <- 1
>
> x <- rnorm(n, mu1, 2)
> y <- rnorm(n, mu2, 4)
>
> pv <- 1 - pnorm((mean(y) - mean(x)) / sqrt(4/n + 16/n)) # true.p.value
>
> # bootstrap p.value
> pool <- c(x, y)
> count <- 0
> t <- mean(y) - mean(x)
> t.bt <- NULL
> for (b in 1:B) {
+   id.bt <- sample(2 * n, replace = T)
+   x.bt <- pool[id.bt[1:n]]
+   y.bt <- pool[-id.bt[1:n]]
+   t.bt[b] <- mean(y.bt) - mean(x.bt)
+ }
> pv.bt <- mean(t.bt >= t)
>
> print(pv)
[1] 0.02245315
> print(pv.bt)
[1] 0.03
```

Illustration of Bootstrap method VIII

```
> # one-sample mean comparison
> B <- 500
> n <- 20
>
> mu <- 1
> mu0 <- 0 # null value
> sigma <- 4
> x <- rnorm(n, mu, sigma)
> pv <- 1 - pnorm(sqrt(n) * (mean(x) - mu0)/sigma) # true p.value
>
> # bootstrap p.value
> x.tilde <- x - mean(x) + mu0        # observed values under H0
> t <- sqrt(n) * (mean(x) - mu0)/sd(x)
> t.bt <- NULL
> for (b in 1:B) {
+   id.bt <- sample(n, replace = T)
+   x.bt <- x.tilde[id.bt]
+   t.bt[b] <- sqrt(n) * (mean(x.bt) - mu0)/sd(x.bt)
+ }
>
> pv.bt <- mean(t.bt >= t)
>
> pv
[1] 0.1279085
> pv.bt
[1] 0.1
```

Parallel Computing in R I

- MC / Bootstrap/ CV are often computationally intensive due to repeated computations.
- However, repetitions are independent and hence can be readily parallelized. (i.e., distribute different jobs to multi-cores)
- There are two popular packages in `R` for parallel computing.
  - `parallel`
  - `foreach`

Parallel Computing in R II

- Simulated example: Simple logistic regression with $n = 1000$.

```
> n <- 1000
> B <- 200
>
> ....
>
> # function for bootsrap samples
> fx <- function(b) {
+   set.seed(b)
+   id.bt <- sample(n, replace = T)
+   y.bt <- y[id.bt]
+   x.bt <- x[id.bt]
+   coef(glm(y.bt ~ x.bt, family = "binomial"))
+ }
>
> # bootstrap samples
> # for (not parallelized)
> tic1 <- Sys.time()
> beta.bt1 <- matrix(0, B, 2)
> for (b in 1:B) {
+   beta.bt1[b,] <- fx(b)
+ }
> toc1 <- Sys.time()
> print(toc1 - tic1)
Time difference of 0.5886581 secs
```

Parallel Computing in R III

- `mclapply()` function in `parallel` package

```
> # parallel package
> library(parallel)
> n.cores <- detectCores()
> n.cores
[1] 12
>
> tic2 <- Sys.time()
> beta.bt2 <- mclapply(1:B, fx, mc.cores = n.cores)
> toc2 <- Sys.time()
> print(toc2 - tic2)
Time difference of 0.186506 secs
```

Parallel Computing in R IV

- `foreach()` function `foreach` package

```
> # foreach package
> library(foreach)
> library(doParallel)
> registerDoParallel(n.cores) # set cores
> tic3 <- Sys.time()
> beta.bt3 <- foreach (b=1:B) %dopar% {
+             fx(b)
+ }
> toc3 <- Sys.time()
> print(toc3 - tic3)
Time difference of 0.2491791 secs
```