Bootstrap,
Random Forest,
and all sorts of magic

DataHack 2018

Dana Kaner, PerimeterX

October 2018

The (Amazing) Bootstrap

Ng's Problem

Andrew Ng teaches a class of Data Scientists, and would like to determine its quality as follows:

Ng's Problem

Andrew Ng teaches a class of Data Scientists, and would like to determine its quality as follows:

Ask each student how to choose the number of trees in Random Forest.

Ng's Problem

Andrew Ng teaches a class of Data Scientists, and would like to determine its quality as follows:

Ask each student how to choose the number of trees in Random Forest.

"using cross validation" $\Rightarrow$ mark $\mathbf{cv}_i\mathbf{=1}$
"just by guessing" $\Rightarrow$ mark $\mathbf{cv}_i\mathbf{=0}$

Ng's Problem

Andrew Ng teaches a class of Data Scientists, and would like to determine its quality as follows:

Ask each student how to choose the number of trees in Random Forest.

"using cross validation" $\Rightarrow$ mark $\mathbf{cv}_i\mathbf{=1}$
"just by guessing" $\Rightarrow$ mark $\mathbf{cv}_i\mathbf{=0}$

The average of $\{cv_i\}_{i=1}^{n}$ is our final estimator

Ng's Problem

Andrew Ng teaches a class of Data Scientists, and would like to determine its quality as follows:

Ask each student how to choose the number of trees in Random Forest.

"using cross validation" $\Rightarrow$ mark $\mathbf{cv}_i\mathbf{=1}$
"just by guessing" $\Rightarrow$ mark $\mathbf{cv}_i\mathbf{=0}$

The average of $\{cv_i\}_{i=1}^{n}$ is our final estimator

But Ng still wonders what's the quality of this estimator - will it work on other classes? Is it stable enough?
His problem: he only has this one class of Data Scientists

Ng's Problem

Andrew Ng teaches a class of Data Scientists, and would like to determine its quality as follows:

Ask each student how to choose the number of trees in Random Forest.

"using cross validation" $\Rightarrow$ mark $\mathbf{cv}_i\mathbf{=1}$
"just by guessing" $\Rightarrow$ mark $\mathbf{cv}_i\mathbf{=0}$

The average of $\{cv_i\}_{i=1}^{n}$ is our final estimator

But Ng still wonders what's the quality of this estimator - will it work on other classes? Is it stable enough?
His problem: he only has this one class of Data Scientists

And now he's asking for your help

A Possible Solution: Bootstrap

An unknown distribution of "the world", F

$$cv \sim F$$

A Possible Solution: Bootstrap

An unknown distribution of "the world", F

$$cv \sim F$$

We observe some data and estimate based on it a property of the world

$$L = \{cv_i\}_{i=1}^{n} \;\rightarrow\; \hat{p} = \frac{1}{n}\sum_{i=1}^{n} cv_i$$

A Possible Solution: Bootstrap

An unknown distribution of "the world", F

$$cv \sim F$$

We observe some data and estimate based on it a property of the world

$$L = \{cv_i\}_{i=1}^{n} \;\rightarrow\; \hat{p} = \frac{1}{n}\sum_{i=1}^{n} cv_i$$

Now, we would like to perform inference on this estimation

$$SE_F\left(\hat{p}\right)$$

A Possible Solution: Bootstrap

An unknown distribution of "the world", F

$$cv \sim F$$

We observe some data and estimate based on it a property of the world

$$L = \{cv_i\}_{i=1}^{n} \;\rightarrow\; \hat{p} = \frac{1}{n}\sum_{i=1}^{n} cv_i$$

Now, we would like to perform inference on this estimation

$$SE_F\left(\hat{p}\right)$$

- Option 1 - Make additional assumptions, complicated formulas

A Possible Solution: Bootstrap

An unknown distribution of "the world", F

$$cv \sim F$$

We observe some data and estimate based on it a property of the world

$$L = \{cv_i\}_{i=1}^{n} \ \rightarrow \ \hat{p} = \frac{1}{n}\sum_{i=1}^{n} cv_i$$

Now, we would like to perform inference on this estimation

$$SE_F\left(\hat{p}\right)$$

- Option 1 - Make additional assumptions, complicated formulas
- Option 2 - **Bootstrap**: Create an "alternative world" $\hat{F}$ based on L

What is Bootstrap?

Main concerns

1. How do we build $\hat{F}$ that is similar to F?
    - **Non-parametric Bootstrap:** use the empirical distribution of the data which puts probability $\frac{1}{n}$ on each observation of L

$$L = \{1, 1, 1, 0, 0\} \rightarrow p_{\hat{F}}\left(x = t\right) = \begin{cases} \frac{2}{5} & t = 0 \\ \frac{3}{5} & t = 1 \end{cases}$$

    - **Parametric Bootstrap:** assume a distribution with unknown parameters, estimate them from L and sample from this distribution

$$L \rightarrow \hat{F}\left(\hat{p}_L\right)$$

2. How to perform the estimation in the bootstrap world?

Example: Standard Error of the Mean

The plug-in principle

Real World: F

$$\begin{tikzcd} F \arrow[r] \arrow[d] & L = \{cv_i\}_{i=1}^{n} \arrow[d] \\ SE_F\left(\hat{p}\right) & \hat{p} = \frac{1}{n}\sum_{i=1}^{n} cv_i \end{tikzcd}$$

Example: Standard Error of the Mean

The plug-in principle

Real World: F

$$
\begin{array}{ccc}
F & \rightarrow & L = \{cv_i\}_{i=1}^{n} \\
\downarrow & & \downarrow \\
SE_F(\hat{p}) & & \hat{p} = \frac{1}{n}\sum_{i=1}^{n} cv_i
\end{array}
$$

Bootstrap World: $\hat{F}$

$$
\begin{array}{ccl}
\hat{F} & \rightarrow & \multirow{2}{*}{$\left\{ \begin{array}{lcl} L_1 = \left\{cv_i^1\right\}_{i=1}^{n} & \rightarrow & \hat{p}_1^* = \sum_{i=1}^{n} cv_i^1 \\ \vdots & & \vdots \\ \vdots & & \vdots \\ L_B = \left\{cv_i^B\right\}_{i=1}^{n} & \rightarrow & \hat{p}_B^* = \sum_{i=1}^{n} cv_i^B \end{array} \right.$} \\
SE_{\hat{F}}(\hat{p}^*) & \leftarrow &
\end{array}
$$

Example: Standard Error of the Mean

Real World: F

$$\begin{array}{ccc} F & \rightarrow & L = \{cv_i\}_{i=1}^{n} \\ \downarrow & & \downarrow \\ SE_F(\hat{p}) & & \hat{p} = \frac{1}{n}\sum_{i=1}^{n} cv_i \end{array}$$

Bootstrap World: $\hat{F}$

$$\begin{array}{ccc} \hat{F} & \rightarrow & \left\{\begin{array}{lcl} L_1 & \rightarrow & \hat{p}_1^* \\ \vdots & & \vdots \\ \vdots & & \vdots \\ L_B & \rightarrow & \hat{p}_B^* \end{array}\right. \\ SE_{\hat{F}}(\hat{p}^*) & \leftarrow & \end{array}$$

1. Draw B Bootstrap samples from $\hat{F}$
2. Estimate Bootstrap replications $\hat{p}_b^*$, $b = 1, \ldots, B$
3. Compute based on the empirical distribution of the replications:

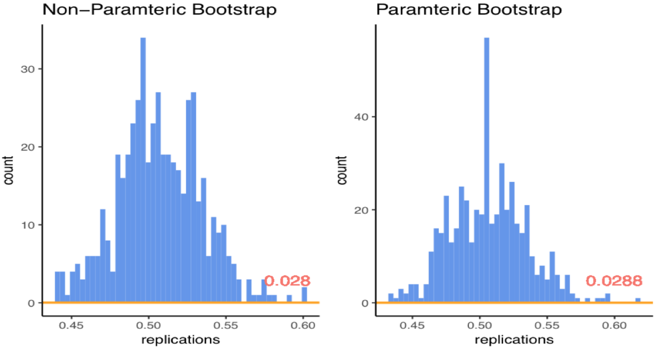
$$SE_{\hat{F}}(\hat{p}^*) := \sqrt{\sum_{b=1}^{B} \frac{\left(\hat{p}_b^* - \frac{1}{B}\sum_{b=1}^{B} \hat{p}_b^*\right)^2}{B-1}}$$

Bootstrap in Practice (R): Parametric vs. Non-Parametric

$$cv \sim Ber\left(p = 0.5\right),\ L = \{cv_i\}_{i=1}^{300} \Rightarrow SE_F\left(\hat{p}\right) \approx 0.0288$$

Bootstrap estimation of $SE_{\hat{F}}\left(\hat{p}^*\right)$:

- Non-parametric Bootstrap using the empirical distribution as $\hat{F}$
- Parametric Bootstrap using $Ber\left(\hat{p} \approx 0.506\right)$ as $\hat{F}$



Bootstrap in Practice (R): Parametric vs. Non-Parametric

Performing a manual calculation

Non-parametric Bootstrap

```
set.seed(5)
non_param_mu_replications_manual <- c()

for (b in 1:B) {

  ## draw a BS sample from the
  ## empirical distribution of the sample L
  L_b <- sample(L, n, replace = T)

  ## compute a BS replication on the sample
  p_b <- mean(L_b)

  non_param_mu_replications_manual <-
    c(non_param_mu_replications_manual, p_b)
}

## compute the empirical SE of the replications
se_nparam_bs_manual <-
  round(sd(non_param_mu_replications_manual),4)
```

Parametric Bootstrap $Ber\left(\hat{p}\right)$

```
set.seed(6)
param_mu_replications_manual <- c()

for (b in 1:B) {

  ## draw a BS sample from Bernoulli(p_hat)
  ## with p_hat estimated on L
  L_b <- rbern(n, p_hat)

  ## compute a BS replication on the sample
  p_b <- mean(L_b)

  param_mu_replications_manual <-
    c(param_mu_replications_manual, p_b)
}

## compute the empirical SE of the replications
se_param_bs_manual <-
  round(sd(param_mu_replications_manual),4)
```

Bootstrap in Practice (R): Parametric vs. Non-Parametric

Using "Boot" library

Non-parametric Bootstrap

```
func <- function(d, i)
{
  d2 <- d
  return(mean(d2[i]))
}
```

```
set.seed(2)
npBootstrap <- boot(L, func, R = 500)

se_nparam_bs <- round(sd(npBootstrap$t),4)
```

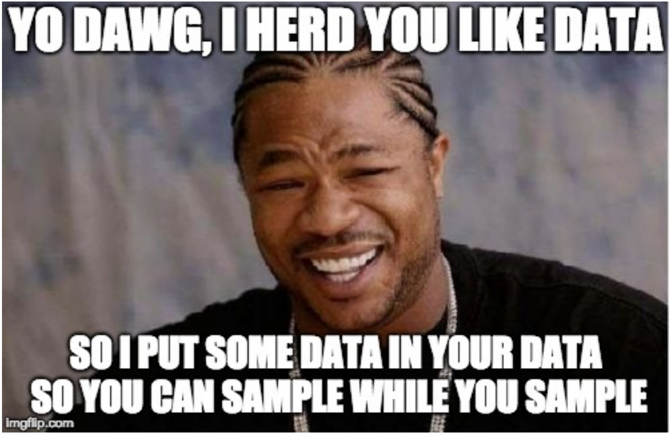
Parametric Bootstrap $Ber(\hat{p})$

```
func2 <- function(d, i)
{
  d2 <- d
  return(mean(d2[i]))
}
```

```
funcran <- function(data, mle) {
  d <- rbern(n, p_hat)
  return (d)
}
```

```
set.seed(3)
pBootstrap <- boot(L, func2, R = 500,
                     sim = "parametric",
                     ran.gen = funcran,
                     mle = p_hat)

se_param_bs <- round(sd(pBootstrap$t),4)
```


YO DAWG, I HERD YOU LIKE DATA
SO I PUT SOME DATA IN YOUR DATA
SO YOU CAN SAMPLE WHILE YOU SAMPLE
imgflip.com

Real World: F

$$\begin{tikzcd} F \arrow[r] \arrow[d] & L \arrow[d] \\ \theta_F = t(F) & s(L) \end{tikzcd}$$

Bootstrap World: $\hat{F}$

$$\begin{tikzcd} \hat{F} \arrow[r] & L_b \arrow[d] \\ \hat{\theta} = t\left(\hat{F}\right) & s^*(L_b) \arrow[l] \end{tikzcd}$$

Generally, Bootstrap is a powerful tool that can be used for many kinds of statistical tasks: hypothesis testing, confidence intervals, bias estimation, evaluation of models accuracy and more

Fun Fact

Each bootstrap sample will contain approximately 0.632 of the sample

Consider $x \sim F$ and a sample $L = \{x_i\}_{i=1}^{n}$

Draw a Bootstrap sample L_b from the empirical distribution $\hat{F}$

The probability of never choosing a certain observation: $\left(1 - \frac{1}{n}\right)^n$

$$\lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.368$$

Therefore, the probability of an observation being chosen is ≈ 0.632

Bagging and Random Forest

N PREDICTORS $\succ$ 1 PREDICTOR

Bagging

Real World: F

$$\begin{array}{ccc} F & \rightarrow & L = \{(x_i, y_i)\}_{i=1}^{n} \\ & & \downarrow \\ & & \varphi(x, L) \end{array}$$

$$\downarrow$$

Bootstrap World: $\hat{F}$

$$\begin{array}{ccc} \hat{F} & \rightarrow & \\ & & \begin{cases} L_1 \rightarrow & \varphi_1^*(x, L_1) \\ \vdots & \vdots \\ \vdots & \vdots \\ L_B \rightarrow & \varphi_B^*(x, L_B) \end{cases} \\ \hat{f}_B(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{\varphi}_b^*(x, L_b) & \leftarrow & \end{array}$$

- Justification [1]: a number of predictors is better than one
- The larger the variance of $\varphi(x, L)$ is, the more improvement

[1]See proof in this **link**

Random Forest Motivation

Assume each predictor φ_b^* is a decision tree [2]

- The bias of a single tree = the bias of the ensemble

$$bias^2\left(\varphi_b^*(x)\right) = bias^2\left(\hat{f}_B(x)\right)$$

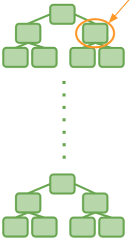
- Assume for each tree $var\left(\varphi_b^*(x)\right) = \sigma^2$ and that the pairwise correlation between trees is ρ. Then

$$var\left(\hat{f}_B(x)\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \overset{B\to\infty}{\to} \rho\sigma^2$$

Reduce the var through lowering ρ, avoid effecting the bias and σ^2

[2] For more information about decision trees go to this **slide**

Random Forest



$$\hat{F} \;\rightarrow\; \begin{cases} L_1 \rightarrow \quad \rightarrow \hat{T}_1\left(x, L_1\right) \\ \vdots \\ \vdots \\ L_B \rightarrow \quad \rightarrow \hat{T}_B\left(x, L_B\right) \end{cases}$$

$$\downarrow$$

$$\hat{f}_B\left(x\right) = \frac{1}{B}\sum_{b=1}^{B} \hat{T}_b\left(x, L_b\right)$$

Random Forest in Practice (Python): Bot Detection

The task: detect malicious IPs trying to attack our website

- B-I-G data

- Features x_i: number of http requests, frequency of requests, history on the website, user interaction, technical features (device, browser etc.) and many more
- Labels y_i: legitimate users (0) and bots (1)

Random Forest in Practice (Python): Bot Detection

Using sklearn.ensemble.RandomForestClassifier

```
# hyper parameters options
param_grid = {
    'max_depth': [80,120],
    'max_features': [4,6],
    'min_samples_split': [5,10],
    'n_estimators': [100,300]
}
# Create a base model
rf = RandomForestClassifier(bootstrap = True)

# Instantiate the grid search cv model
grid_search = GridSearchCV(estimator = rf,
                           param_grid = param_grid,
                           cv = 10, # number of folds
                           n_jobs = -1, # number of processors
                           verbose = 1)

grid_search.fit(X = train_features,
                y = train_labels,
                sample_weight = train_weights)

best_grid = grid_search.best_estimator_
```

Thanks

perimeterx

dana@perimeterx.com

Decision Trees

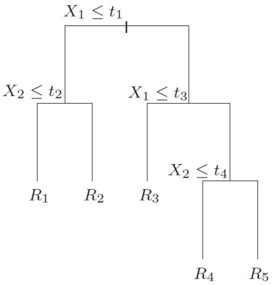
We will define a decision tree by $\Theta = \{(R_k, c_k)\}_{k=1}^{K}$

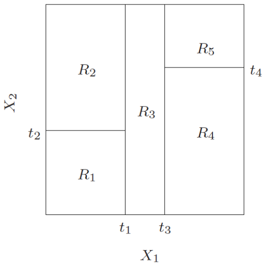
Tree prediction:

$$\hat{f}(x) = \sum_{k=1}^{K} c_K \mathbb{1}_{x_k}$$

Choosing c_k: average or majority of votes

Choosing R_k: at each stage, minimize the error by split s_k of feature x_j





[2] Back to origin **slide**