Confidence Intervals

Observation: It does not matter how careful we are with our model evaluation techniques, there remains a fundamental uncertainty about the ability of our training data to effectively represent our (possibly infinite) data universe.

This uncertainty reflects into our model evaluation. If our training data is a poor representation of the data universe then the models we construct using it will generalize poorly to the rest of the data universe. If our training data is a good representation of the data universe then we can expect that our model will generalize well.

Here we will deal with this uncertainty using *confidence intervals.*

Perhaps most surprising is that we will use our training data itself in order to estimate this uncertainty using the *bootstrap*.

Confidence Intervals

First, let us define *error confidence* intervals formally.

Given a model accuracy, acc_D, over some data set D, then the error confidence interval is defined as the probability p that our model accuracy acc_D lies between some lower bound lb and some upper bound ub,

$$Pr(lb \leq acc_D \leq ub) = p.$$

Paraphrasing this equation with $p = 95\%$:

We are 95% percent sure that our accuracy acc_D is not worse than lb and not better than ub.

Percentiles

Numerical data can be sorted in increasing or decreasing order. Thus the values of a numerical data set have a *rank order*.

A *percentile* is the value at a particular rank.

For example, if your score on a test is on the 95th percentile, a common interpretation is that only 5% of the scores were higher than yours. The median is the 50th percentile; it is commonly assumed that 50% the values in a data set are above the median.

Percentiles

Example: Consider the sorted list:

sizes = [6, 7, 9, 12, 17]

The 80th percentile is a value on the list, namely 12. You can see that 80% of the values are less than or equal to it, and that it is the smallest value on the list for which this is true.

The percentile function takes two arguments: a rank between 0 and 100, and a sorted list. It returns the corresponding percentile of the list.

```
>>> percentile(sizes, 80)
12
```

Percentiles

In practical terms, suppose there are n elements in a collection. To find the pth percentile:

- Sort the collection in increasing order.
- Find p% of n: $(p/100)\times n$. Call that k.
- If k is an integer, take the kth element of the sorted collection.
- If k is not an integer, round it up to the next integer, and take that element of the sorted collection.

The Bootstrap

A particular effective and computationally straightforward way to estimate the lower and upper bounds of confidence intervals is the *bootstrap*.

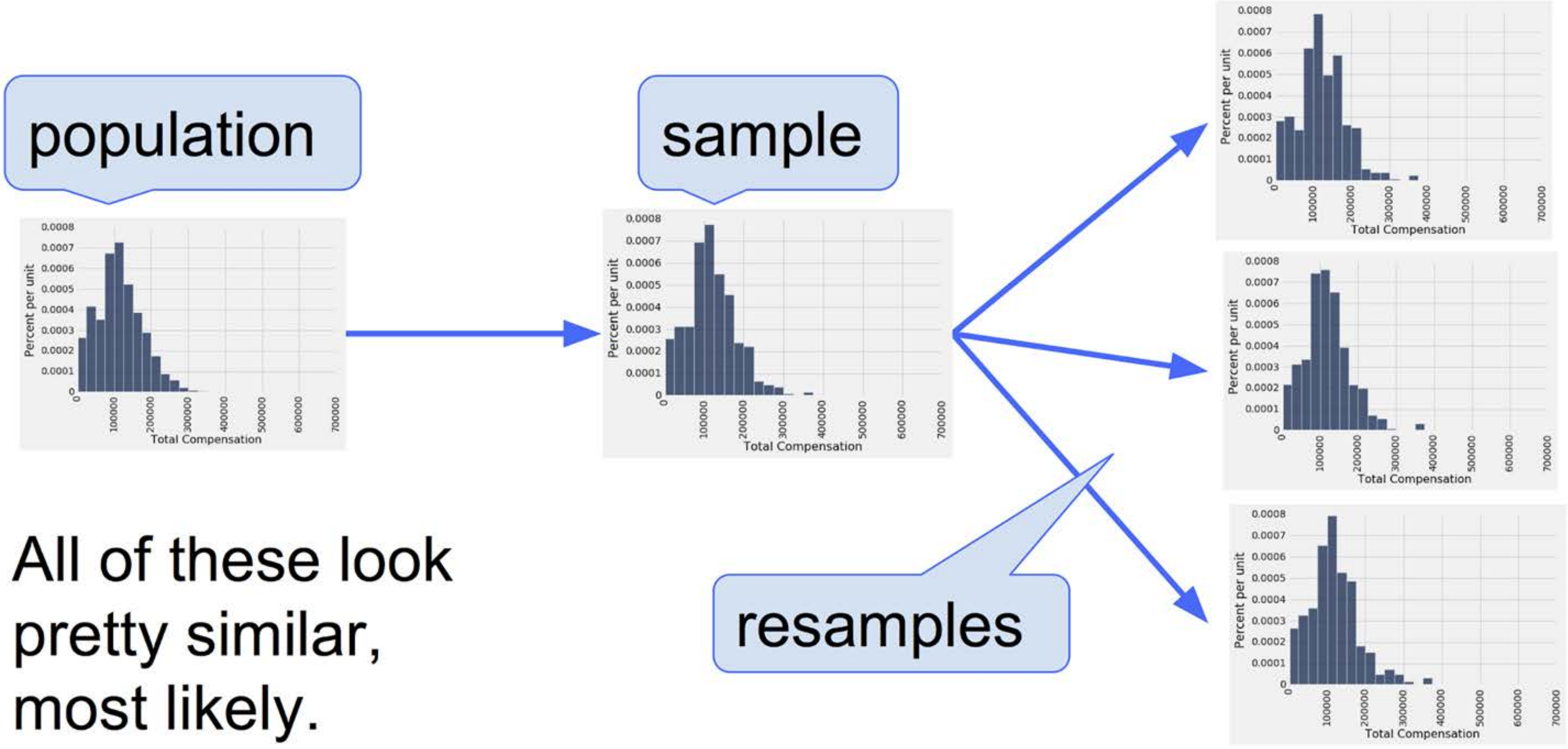
What is remarkable about the bootstrap is that we use the data set D itself to capture the uncertainty with which it represents the data universe at large.

In the bootstrap we create B *bootstrap samples* of our data set D using sampling with replacement.

We use the variation among the bootstrap samples to compute the variation in the respective model accuracies.

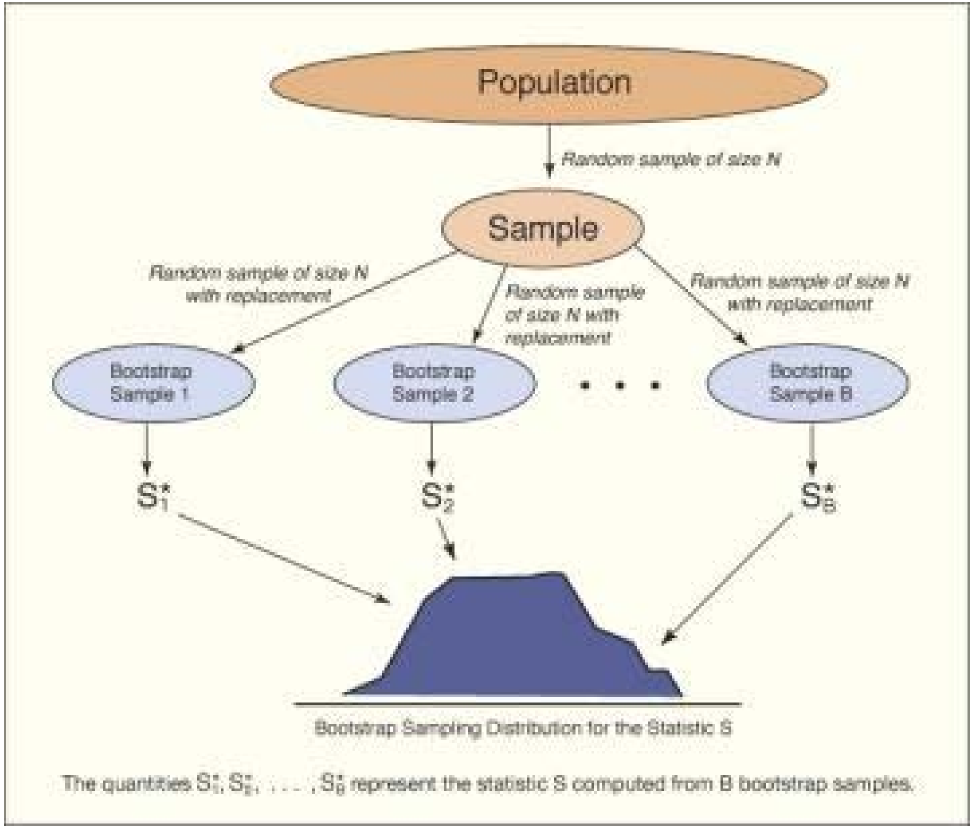
NOTE: the variation among the bootstrap sampled captures the quality of the original data set: if the variation is large then most likely your original data set does NOT represent the data universe very well, if the variation is small then most likely it does!

The Bootstrap - Resampling Technique



All of these look pretty similar, most likely.

The Bootstrap

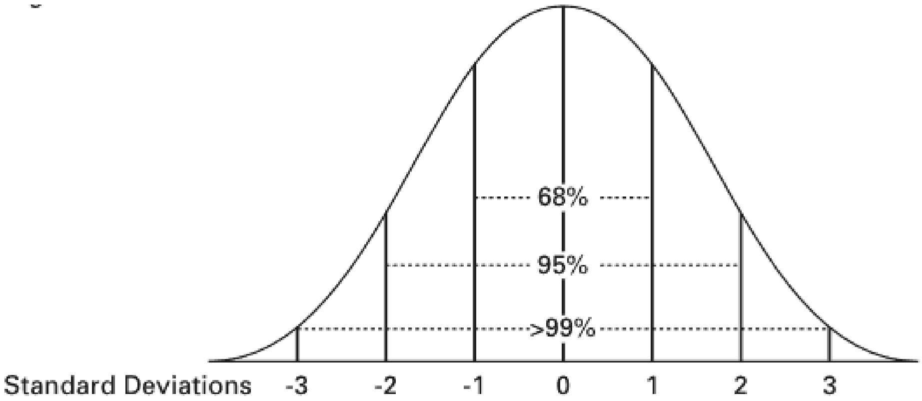


Bootstrap procedure generating B bootstrap samples

In our case the statistic is the model accuracy

We obtain a *sampling distribution* of the model accuracy!

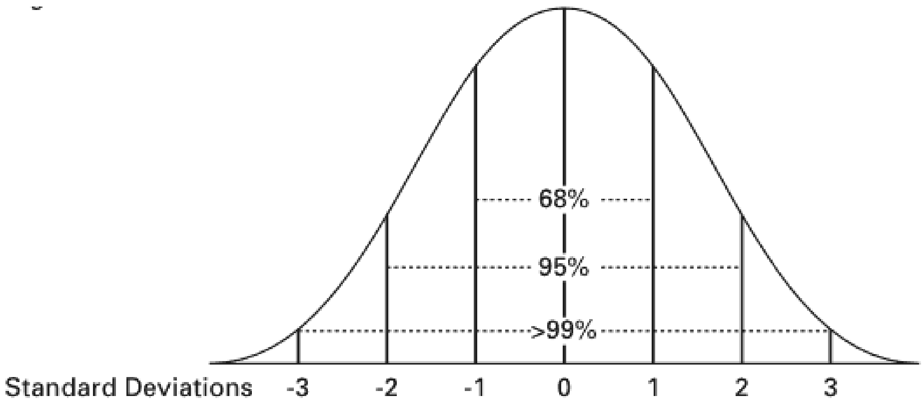
The Bootstrap

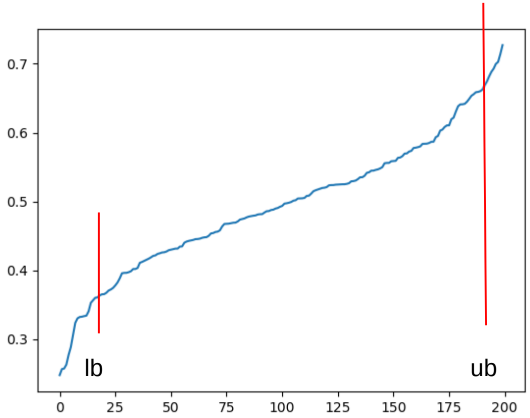


Given the distribution of our accuracies we can now estimate the probabilities and bounds of our confidence intervals.

How often does the empirical distribution of the resampled accuracies capture the actual accuracy? Let's take that to mean "the middle 95% of the resampled accuracies capture the actual accuracy" -
The 95% confidence interval!

The Bootstrap – 95% Confidence Interval





Assume we have the list, sorted_accuracies, then the values of the bounds can be estimated:

lb = percentile(sorted_accuracies, 2.5)

ub = percentile(sorted_accuracies, 97.5)

Bootstrap Procedure

```
given data set D
given model M
for i = 1 to 200 do
    B[i] ← sample D with replacement, note |B[i]| = |D|.
    acc[i] ← compute model M accuracy for B[i].
end for
sort acc in ascending fashion
ub ← percentile(acc, 97.5)
lb ← percentile(acc, 2.5)
return (lb, ub)
```

Bootstrap Procedure

```
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

def bootstrap(model,D,target_name):
    rows,__ = D.shape
    acc_list = []
    for i in range(200):
        B = D.sample(n=rows,replace=True)
        X = B.drop(target_name,1)
        y = B[target_name]
        train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.8)
        model.fit(train_X, train_y)
        predict_y = model.predict(test_X)
        acc_list.append(accuracy_score(test_y, predict_y))
    acc_list.sort()
    ub = percentile(acc_list,97.5)
    lb = percentile(acc_list,2.5)
    return (lb, ub)
```

Confidence Intervals

By now it should be clear that a single performance number computed on D is perhaps a poor indicator for models.

As an example, consider the model M* with an accuracy,

$$Acc = 0.9,$$

and a 95% confidence interval (0.88, 0.92). Consider another model M' with

$$Acc = 0.95,$$

and a 95% confidence interval (0.91, 0.99).

By just looking at the accuracy we are tempted to say that the second model is superior to the first model.

However, the *confidence intervals overlap*, meaning that the performance difference between the two models is *statistically not significant*.

Confidence Intervals

```
from sklearn import tree
from bootstrap import bootstrap

t1 = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3)
t2 = tree.DecisionTreeClassifier(criterion='entropy', max_depth=None)

print("******** iris ***********")
df = pd.read_csv("iris.csv")
print("Confidence interval max_depth=3: {}".format(bootstrap(t1,df,'Species')))
print("Confidence interval max_depth=None: {}".format(bootstrap(t2,df,'Species')))

print("******** abalone ***********")
df = pd.read_csv("abalone.csv")
print("Confidence interval max_depth=3: {}".format(bootstrap(t1,df,'sex')))
print("Confidence interval max_depth=None: {}".format(bootstrap(t2,df,'sex')))
```

Confidence Intervals

Decision trees:

******** iris ***********
Confidence interval max_depth=3: (0.93, 1.0)
Confidence interval max_depth=None: (0.97, 1.0)

This means that the difference between the performances we had seen earlier for the decision trees are not statistically significant for the iris data set.

******** abalone ***********
Confidence interval max_depth=3: (0.51, 0.59)
Confidence interval max_depth=None: (0.74, 0.80)

For the abalone data set the difference is statistically significant!

Team Exercise

- Find a classification dataset (see course website for sources)
- Find the best decision tree for the dataset using **5-fold cross-validation**
  - Recall that the decision tree has two free parameters: criterion and tree depth
  - You can use the **grid search module** to accomplish this
  - If your dataset has less than 50 rows then use 3-fold cross-validation
- Compute the **confidence interval** for the optimal parameters using the bootstrap procedure.

Lab Groups

Teams:

Group 1: Chenot, Samuel	Capaldi, Ronni Sue	Thomas, Ryan
Group 2: Lehane, Robert	Ferrell, Baheem	St-Martin, Ben
Group 3: Phillips, Connor	Turner, Jacob	Mason, Conor
Group 4: Mendonca,	Cameron O'Neill, Sydney	Leonardo Polanco, Luis
Group 5: Dunn, Kevin	Moreno, Zentonio	Cabral, Adam
Group 6: Samuel, Etebom	Wade, Brandon	Campbell, Ryan
Group 7: Bakos, Sofia	Behrens, Jack	Stone, Chad
Group 8: Gregory, Steven	O'Connell, Mark	Johnson, Ben
Group 9: Wildenhain, Evan	Babineau, Allie	Gallagher, Beibhinn
Group 10: Antranik, Antranik	Caterino, Matthew	Gaines, Leah
Group 11: Mullin, Jason	Cole, Tyler	Agronick, Austin
Group 12: Bruno, Heather	Janiszewski, Joseph	Owen, Abraham