

Review: Statistical Learning in Matlab. Bootstrap.

Mathematical Tools for ITS (11MAI)

Mathematical tools, 2022

Jan Přikryl

(based on the book “Introduction to Statistical Learning”, <https://www.statlearning.com/>)

11MAI, lecture 7

Monday, November 14, 2022

version: 2022-11-14 12:09

Department of Applied Mathematics, CTU FTS

Review — Model Flexibility, Classification

Review of Model Felxibility

Matlab session 7.1 — Classification: Stock market data

Bootstrap

We collect a set of data ($n = 100$ observations) containing a single predictor x and a quantitative response y . We then fit a linear regression model to the data, as well as a separate cubic regression, i.e., $y = \beta_0 + \beta_1x + \beta_2x^2 + \beta_3x^3 + \epsilon$.

- a) Suppose that the true relationship between x and y is linear, i.e., $y = \beta_0 + \beta_1x + \epsilon$. Consider the training residual sum of squares (RSS) for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.
- b) Answer a) using RSS on a testing set rather than training RSS.

- c) Suppose now that the true relationship between x and y is non-linear, but we don't know how much non-linear it is. Consider the training RSS for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.
- d) Answer c) using RSS on a testing set rather than training RSS.

Matlab Session 7.1

Is there a way to seriously predict the development of a stock market index?

Let's start with selected historical daily data of S&P between 2001–2005.

Read and display the general characteristics

```
smarket = readtable('islr_smarket.csv');  
summary(smarket)
```

The **Direction** is a categorical variable (either **Up** or **Down**) and we have to tell that to Matlab:

```
smarket.Direction = categorical(smarket.Direction)
```

We might be interested in correlations between different numerical variables in the data. Matlab has the `corrcoef()` function exactly for this purpose ...

```
corrcoef(smarket)
```

...but the input has to be a matrix:

```
smarket_matrix = table2array(smarket(:,2:end-1))  
smarket_cc = corrcoef(smarket_matrix);
```

Explain, why we index `smarket(:,2:end-1)`.

Can you find some values in the correlation matrix indicating a correlation between some columns? If yes, to which variables do these columns correspond?

We might be interested in correlations between different numerical variables in the data. Matlab has the `corrcoef()` function exactly for this purpose ...

```
corrcoef(smarket)
```

...but the input has to be a matrix:

```
smarket_matrix = table2array(smarket(:,2:end-1))  
smarket_cc = corrcoef(smarket_matrix);
```

Explain, why we index `smarket(:,2:end-1)`.

Let's plot it:

```
plot(smarket.Volume)
```

Based on the graph data explain why are **Year** and **Volume** positively correlated.

Let's now train a generalised linear model of **Direction** as a function of **Lag1** through **Lag5**. The logistic model can be specified using '**Distribution**', '**binomial**':

```
mdl = fitglm(smarket,  
'Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume',  
'Distribution', 'binomial')
```

Which regression coefficient has the smallest p -value? Does this value suggest a strong influence on the model output?

Have a look at the model prediction on the original data:

```
probs = predict mdl
```

The quality of the model can be assessed by comparing the model prediction with the training values stored in `Direction`. But first we have to convert `probs` to categorical values `Up` and `Down`:

```
predictions = repmat(categorical({'Down'}), mdl.NumObservations, 1);  
predictions(probs>0.5) = 'Up'; %
```

We continue with the confusion matrix:

```
confusionmat(predictions, smarket.Direction)  
(507+145)/1250  
mean(predictions == smarket.Direction)
```

Is our model better than a random choice? What is its training error?

A significantly better estimate of a real model error can be obtained by splitting the data into training and testing datasets. Let's try to fit the model to 2001–2004 data and verify it on data from 2005.

```
train = (smarket.Year<2005); % Boolean column vector of true/false  
smarket_train = smarket(train,:);  
smarket_test = smarket(~train,:);
```

What does `smarket(train,:)`, and `smarket(~train,:)` mean?

How large are the training and testing datasets?

```
size(smarket_train)  
size(smarket_test)
```

Fit the model to the training data and verify it on data from 2005:

```
mdl = fitglm(smarket_train,  
    'Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume',  
    'Distribution', 'binomial')  
probs = predict(mdl, smarket_test); % Test set data  
% Conversion to Up/Down  
predictions = repmat(categorical({'Down'}),mdl.NumObservations,1);  
predictions(probs>0.5) = 'Up';  
% Confusion matrix and the percentage of correct predictions  
confusionmat(predictions, smarket_test.Direction)  
mean(predictions == smarket_test.Direction)
```

How large is the test set error?

Let us fit the model using only **Lag1** and **Lag2**, which had the strongest predictive power in the original logistic regression:

```
mdl1 = fitglm(smarket_train,  
    'Direction_~_Lag1+Lag2',  
    'Distribution', 'binomial')  
probs = predict(mdl1, smarket_test);  
predictions = repmat(categorical({'Down'}),252,1);  
predictions(probs>0.5) = 'Up';  
confusionmat(predictions, smarket_test.Direction)  
mean(predictions == smarket_test.Direction)
```

Did the testing error estimate improve now? What is the probability of market increase? What about market decrease? And what about some simple strategies?

Finally, we will compute predictions for new values **Lag1** and **Lag2** given by the following table:

Lag1	Lag2
1,2	1,1
1,5	-0,8

```
% Create a new data table for the two data records  
pt = table([1.2;1.5], [1.1;-0.8], 'VariableNames', {'Lag1', 'Lag2'});  
% Evaluate the model 'mdl2' on data from 'pt'  
predict(mdl2, pt)'
```

Instead of using a Matlab table we could use the **pt** directly in a matrix form. How can we do it?

Review — Model Flexibility, Classification

Bootstrap

Matlab session 7.2

Definition (Bootstrap)

The bootstrap is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.

For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

- This is not really necessary for linear regression ([why?](#))
- But it can be easily applied to a wide range of other statistical learning methods where a measure of variability is difficult to obtain

The use of the term bootstrap derives from the phrase *to pull oneself up by one's bootstraps*, widely thought to be based on one of the eighteenth century "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe:

The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.

- It is not the same as the term "bootstrap" used in computer science meaning to "boot" a computer from a set of core instructions, though the derivation is similar.
- It is unrelated to Twitter Bootstrap, a framework for web development.

Let's start with a simple example:

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y , respectively, where X and Y are random quantities.
- We will invest a fraction α of our money in X , and will invest the remaining $1 - \alpha$ in Y .
- We wish to choose α to minimize the total risk (expressed as **variance**), of our investment. In other words, we want to minimize $\text{var}(\alpha X + (1 - \alpha)Y)$.
- One can show that the value that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

where $\sigma_X^2 = \text{var}(X)$, $\sigma_Y^2 = \text{var}(Y)$, and $\sigma_{XY} = \text{cov}(X, Y)$.

Let's start with a simple example:

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y , respectively, where X and Y are random quantities.
- We will invest a fraction α of our money in X , and will invest the remaining $1 - \alpha$ in Y .
- We wish to choose α to minimize the total risk (expressed as **variance**), of our investment. In other words, we want to minimize $\text{var}(\alpha X + (1 - \alpha)Y)$.
- One can show that the value that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

where $\sigma_X^2 = \text{var}(X)$, $\sigma_Y^2 = \text{var}(Y)$, and $\sigma_{XY} = \text{cov}(X, Y)$.

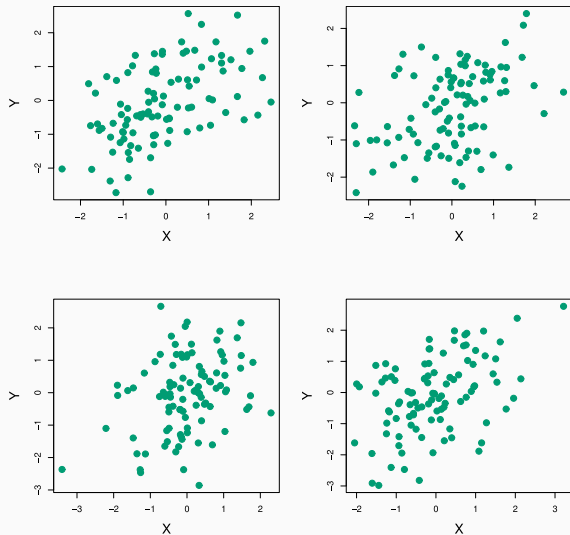
- We have from the previous slide

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

but the values of σ_X^2 , σ_Y^2 , and σ_{XY} are unknown.

- We can compute estimates for these quantities, $\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$ and $\hat{\sigma}_{XY}$, using a data set that contains measurements for X and Y .
- We can then estimate the value of α that minimizes the variance of our investment using

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}.$$



Each panel displays 100 simulated returns for investments X and Y . From left to right and top to bottom, the resulting estimates for α are equal to 0.576, 0.532, 0.657, and 0.651.

To estimate the standard deviation of $\hat{\alpha}$, we repeat the process of simulating 100 paired observations of X and Y , and estimating α 1000 times. We obtain 1000 estimates for α : $\hat{\alpha}_1, \hat{\alpha}_2, \dots, \hat{\alpha}_{1000}$.

- The mean over all 1000 estimates for α is

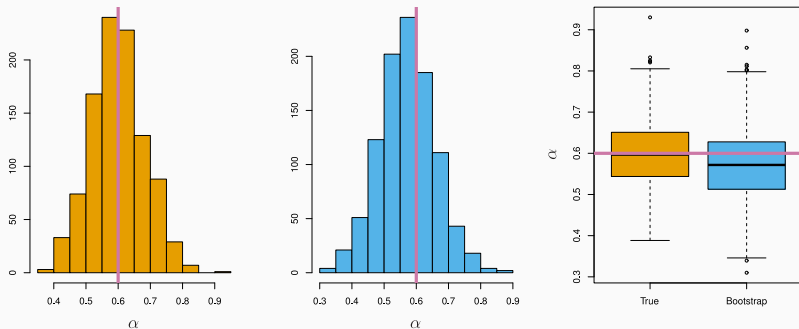
$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996,$$

and the standard deviation of the estimates is

$$\sqrt{\frac{1}{1000 - 1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083.$$

- This gives us a very good idea of the accuracy of $\hat{\alpha}$: $SE(\hat{\alpha}) \approx 0.083$.

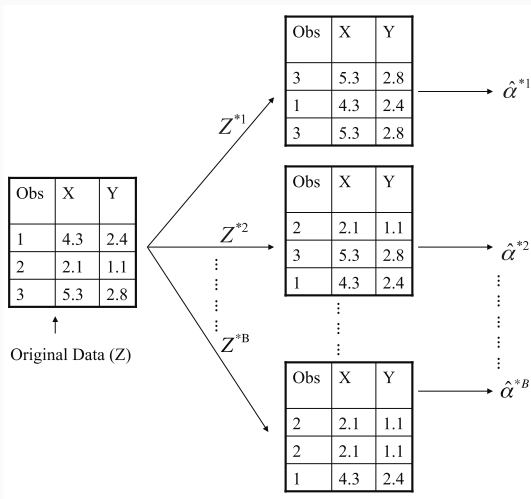
- The value of $SE(\hat{\alpha}) \approx 0.083$ suggests that, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from the **unknown** α by approximately 0.08, on average.
- The left-hand panel of the Figure on slide 22 displays a histogram of the resulting estimates.
- For these simulations the parameters were set to $\sigma_X^2 = 1$, $\sigma_Y^2 = 1.25$ a $\sigma_{XY} = 0.5$, and so we know that the true value of α is 0.6 (indicated by the red line).
- Remember that our estimated value of α from 1000 samples is $\bar{\alpha} = 0.5996$, which is pretty close.



Left: A histogram of the estimates of α obtained by generating 1000 simulated data sets from the true population. **Center:** A histogram of the estimates of α obtained from 1000 bootstrap samples from a single data set. **Right:** The estimates of α displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of α .

- The procedure outlined above cannot be applied, because for real data we cannot generate new samples from the original population.
- However, the bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set *with replacement*.
- Each of these “bootstrap data sets” is created by sampling *with replacement*, and is the *same size* as our original dataset. As a result some observations may appear more than once in a given bootstrap data set and some not at all.

Example with just 3 observations



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.

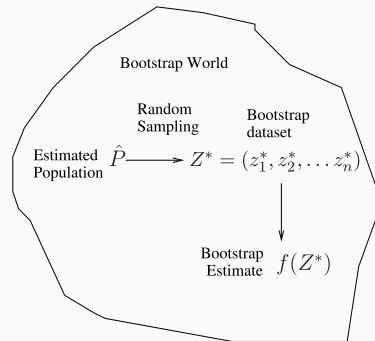
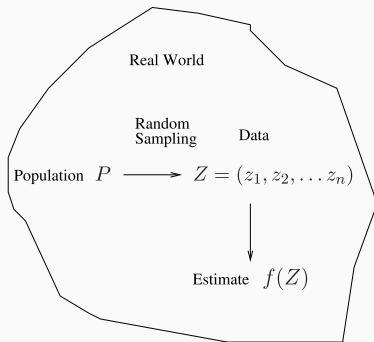
Each bootstrap data set is used to obtain an estimate of α .

- Denoting the first bootstrap data set by Z^{*1} , we use Z^{*1} to produce a new bootstrap estimate for α , which we call $\hat{\alpha}^{*1}$.
- This procedure is repeated B times for some large value of B (say 100 or 1000), in order to produce B different bootstrap data sets, $Z^{*1}, Z^{*2}, \dots, Z^{*B}$, and B corresponding α estimates: $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \dots, \hat{\alpha}^{*B}$.
- We estimate the standard error of these bootstrap estimates using the formula

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1} \sum_{r=1}^B (\hat{\alpha}^{*r} - \bar{\hat{\alpha}}^*)^2}.$$

- This serves as an estimate of the standard error of $\hat{\alpha}$, estimated from the original data set. See centre and right panels of Figure on slide 22. Bootstrap results are in blue. For this example $SE_B(\hat{\alpha}) = 0.087$.

A general picture for the bootstrap



- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can't simply sample the observations with replacement (*why not?*).
- We can instead create blocks of consecutive observations, and sample those with replacements. Then we paste together sampled blocks to obtain a bootstrap dataset.

- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can't simply sample the observations with replacement (*why not?*).
- We can instead create blocks of consecutive observations, and sample those with replacements. Then we paste together sampled blocks to obtain a bootstrap dataset.

- Primarily used to obtain **standard errors of an estimate**.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 22, the 5 % and 95 % quantiles of the 1000 values are equal to 0.43 and 0.72, respectively.
- This represents an approximate 90 % confidence interval for the true α . *How do we interpret this confidence interval?*
- The above interval is called a *Bootstrap Percentile* confidence interval. It is the simplest method (among many approaches) for obtaining a confidence interval from the bootstrap.

- Primarily used to obtain **standard errors of an estimate**.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 22, the 5 % and 95 % quantiles of the 1000 values are equal to 0.43 and 0.72, respectively.
- This represents an approximate 90 % confidence interval for the true α . *How do we interpret this confidence interval?*
- The above interval is called a *Bootstrap Percentile* confidence interval. It is the simplest method (among many approaches) for obtaining a confidence interval from the bootstrap.

- Primarily used to obtain **standard errors of an estimate**.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 22, the 5 % and 95 % quantiles of the 1000 values are equal to 0.43 and 0.72, respectively.
- This represents an approximate 90 % confidence interval for the true α . *How do we interpret this confidence interval?*
- The above interval is called a *Bootstrap Percentile* confidence interval. It is the simplest method (among many approaches) for obtaining a confidence interval from the bootstrap.

- In cross-validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around — with original sample = training sample, bootstrap dataset = validation sample — is worse!

- In cross-validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around — with original sample = training sample, bootstrap dataset = validation sample — is worse!

- In cross-validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around — with original sample = training sample, bootstrap dataset = validation sample — is worse!

- In cross-validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around — with original sample = training sample, bootstrap dataset = validation sample — is worse!

- In cross-validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around — with original sample = training sample, bootstrap dataset = validation sample — is worse!

- In cross-validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around — with original sample = training sample, bootstrap dataset = validation sample — is worse!

- In cross-validation, each of the K validation folds is distinct from the other $K - 1$ folds used for training: *there is no overlap*. This is crucial for its success. *Why?*
- To estimate prediction error using the bootstrap, we could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample.
- But each bootstrap sample has significant overlap with the original data. About two-thirds of the original data points appear in each bootstrap sample. *Can you prove this?*
- This will cause the bootstrap to seriously underestimate the true prediction error. *Why?*
- The other way around — with original sample = training sample, bootstrap dataset = validation sample — is worse!

- Can partly fix this problem by only using predictions for those observations that did not (by chance) occur in the current bootstrap sample.
- But the method gets complicated, and in the end, cross-validation provides a simpler and more attractive approach for estimating prediction error.

Matlab Session 7.2

We will first replicate the process of estimating the α statistics on `islr_portfolio` data.

In the second part we will estimate the *Bootstrap Percentile* confidence interval of the estimated α^* .

Read in the portfolio dataset. The first column contains row names and we have to read them as such, otherwise an extra variable will appear, automatically named **Var1**.

```
tbl = readtable('islr_portfolio.csv', 'ReadRowNames', true);
```

Like many function from the Statistical Toolbox, Matlab's `bootstrap()` requires input data to be a scalar, column vector or matrix. Table is not accepted.

We have to convert the data table to a matrix:

```
portfolio = table2array(tbl);
```

Run the bootstrap sampling for 1000 different datasets resampled from the original Portfolio dataset and estimate the alpha coefficient

```
N = 1000;  
alpha_boot = bootstrp(N, @mai_71_alpha, portfolio);
```

The expected value would be the mean of all bootstrapped alphas

```
alpha_est = mean(alpha_boot);
```

And we can estimate the standard error

```
alpha_se = std(alpha_boot);
```

We can also get the bounds, i.e. minimum and maximum, although they are probably of minimal importance here

```
[alpha_min, alpha_max] = bounds(alpha_boot);
```