Assignment 1

Fundamentals of Data Analytics (SPM4450)
Delft University of Technology

Jan Cees van Senden - 4178017
Thijs ter Horst - 4156749

May 1, 2016

Contents

1 Introduction **2**

2 Project description **3**

2.1 Goal 3

2.2 Hypothesis 3

3 Implementation **4**

4 Results & Discussion **5**

5 Conclusion **7**

Appendices **9**

A Matlab code **10**

1 Introduction

Being able to reliably predict future data points of a random variable Y is important in data analytics. One way to tackle this problem is using regression. In that case several functions varying from linear to higher order functions are fitted to a training dataset T to optimize the weights β_i of that function. A linear function then looks like

$$f(x) = \beta_0 + \beta_1 x \tag{1.1}$$

and a second order function looks like

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 \tag{1.2}$$

The fitted function should then be able to predict values from a new sample T'. Optimization is usually done by minimizing the sum of squared errors [1].

A good performance indicator for a model function is the Mean Squared Error (MSE), which is given by

$$E_T[(f(x) - \hat{f}(x|T))^2] = (f(x) - E_T[\hat{f}(x|T)])^2 + E_T[(\hat{f}(x|T) - E_T[\hat{f}(x|T)])^2] \tag{1.3}$$

We see from equation 1.3 that we can decompose the MSE into two components. The left component is the squared bias and the right component is the variance[1]. Both these terms determine the MSE and depending on the sample size and the nature of the distribution of Y, one function will fit better than another (i.e. have a lower MSE).

2 Project description

2.1 Goal

The goal of this project is to create a dataset based on a particular function and then evaluate how well different order functions are able to fit that data. In particular we will investigate the influence of the sample size on the squared bias, variance and mean squared error.

We will generate our dataset according to equation 2.1.

$$f(x) = \ln\left(\frac{1}{x}\right) \tag{2.1}$$

This equation was chosen because it can be expanded as a Taylor series. We will create samples in the range [0.01, 0.99].

2.2 Hypothesis

Since equation 2.1 can be expanded as a Taylor series we expect that the higher the order the better the fit because the higher the order, the smaller the bias should be.

However since the complexity of the higher order models inherently comes with a larger variance than the lower order models, the sample size will have to be large enough in order to decrease the variance such that the bias term will dominate the MSE. If the sample size is smaller, the lower order models will have a smaller MSE because then the variance will dominate the MSE.

3 Implementation

Our implementation is very similar to the implementation in [1]. First we generate 1000 datasets T_i based on the distribution given in equation 3.1.

$$Y_i \sim \mathcal{N}(\mu = \ln{(\frac{1}{x_i})}, \sigma_\epsilon^2 = 1) \tag{3.1}$$

This is done for a sample size of 10, 100, 1000 and 10000. Then we fit all the β_i's and calculate the bias, variance and MSE. We fit from a first up to a sixth order function.

4 Results & Discussion

The results from the simulation are given in table 4.1 and plotted in figure 4.1. We have highlighted the best values in the table by making them bold.

	Squared bias				Variance				MSE			
	10	100	1000	10000	10	100	1000	1000	10	100	1000	10000
1st	0.4821	0.1677	0.1422	0.1399	**0.2005**	**0.0206**	**0.020**	**0.0002**	0.6826	0.1883	0.1442	0.1401
2nd	0.1679	0.0569	0.0444	0.0433	0.3079	0.0307	0.0030	0.0003	0.4758	0.0876	0.0473	0.0436
3rd	0.0546	0.0240	0.0173	0.0167	0.4027	0.0404	0.0039	0.0004	**0.4573**	0.0644	0.0212	0.0171
4th	0.0153	0.0114	0.0076	0.0073	0.4987	0.0504	0.0049	0.0005	0.5139	**0.0617**	0.0125	0.0078
5th	0.0040	0.0058	0.0036	0.0034	0.6010	0.0599	0.0059	0.0006	0.6050	0.0657	0.0096	0.0040
6th	**0.0013**	**0.0031**	**0.0018**	**0.0017**	0.7055	0.0705	0.0068	0.0007	0.7069	0.0736	**0.0086**	**0.0024**

Table 4.1: Squared bias, variance and MSE results of the simulation for a first up to a sixth order model 10, 100, 1000 and 10000.

The results in table 4.1 show that the higher the order, the smaller the squared bias. This is just as we expected in our hypothesis. The simulation also confirms the higher order models have a larger variance than the lower order models.

Based on the MSE results we see that the higher order models perform better at fitting the data, but the highest order does not necessarily fit best for all sample sizes. As expected the variance at first dominates the MSE for the higher order models. This is why for smaller sample sizes a third or fourth order model will fit the data better.

Because the bias does not scale alot with the sample size whereas the variance V falls off with the sample size as $V \sim \frac{1}{N}$ the higher order models perform better with larger sample sizes.

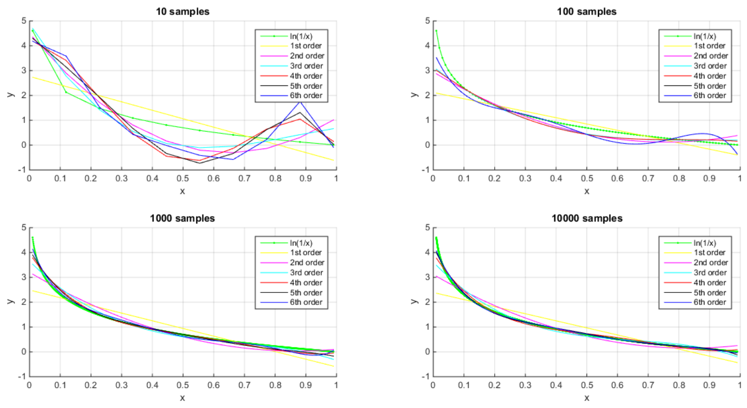

Figure 4.1: Visualization of a first up to a sixth order model fitted to samples of sample sizes 10, 100, 1000 and 10000.

The plot of 10 and 100 samples in figure 4.1 show what happens when a function that is too complex is used with small samples sizes. Overfitting occurs and hence curves arise that are not present in the original distribution. In that case a lower order model will fit the data better and give a more reliable prediction.

As the sample size increases we see that the higher order models fit the original distribution quite nicely, whereas the lower order models (especially the linear model) are unable to adjust enough. This is due to the large bias that is inherently present for these models.

5 Conclusion

In this report we described the simulation of the regression fitting of a distribution based on a function that can be Taylor expanded. The results confirm our hypothesis, showing that for such a distribution higher order models perform best, but only if the sample size is large enough. This arises from the fact that for smaller sample sizes the variance dominates the MSE and the variance for higher order models is inherently larger. Therefore if the sample size is too small it might be better to choose a lower order model.

Bibliography

[1] M. Berthold & D. Hand, *Intelligent Data Analysis*. Springer, 2nd edition, pp. 46-52, 2003.

Appendices

A Matlab code

```
close all
clear

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate k points for an actual function %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = 1000;
sigma = 1;

samples = [10 100 1000 10000];

colors = ['y' 'm' 'c' 'r' 'k' 'b'];

figure

for k = 1:length(samples)

    x = linspace(0.01, 0.99, samples(k))';

    % Choose a function ln(1/x) (log in matlab returns ln)
    % Choose because of taylor series -> higher polynomial should do better
    y_act = log(x.^-1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate n datasets y_samp, sigma = 1 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    y_samp = normrnd(repmat(y_act, 1, n), sigma);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate variance etc, fit and plot with m-polynomial %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    A{1} = [ ones(samples(k), 1), x ];
    A{2} = [ ones(samples(k), 1), x, x.^2 ];
    A{3} = [ ones(samples(k), 1), x, x.^2 x.^3 ];
    A{4} = [ ones(samples(k), 1), x, x.^2 x.^3 x.^4 ];
    A{5} = [ ones(samples(k), 1), x, x.^2 x.^3 x.^4 x.^5 ];
    A{6} = [ ones(samples(k), 1), x, x.^2 x.^3 x.^4 x.^5 x.^6];

    subplot(2,2,k);
    xlim([0 1]);
    hold on;
    grid on;

    plot(x, y_act, 'g.-');
    for i = 1:length(A)

        % matlab returns a least squares solution here,
        % since the matrix dimensions will not match
        beta = A{i}\y_samp;
        y_hat = A{i} * beta;

        % bias and variance taken from page 50 of the book. Because we have
        % n datasets instead of one, I added an extra 'mean' around the
        % definition: E_total = mean(E_T_i) forall i = 1:n
        bias_squared(i, k) = mean((y_act - mean(y_hat,2)).^2);
        variance(i, k) = mean(mean((y_hat - repmat(mean(y_hat, 2),1, n)).^2));

        % calculate the Mean Squared Error twice, once by adding bias and
        % variance, and once by its definition, as a sanity check. Obviously,
        % the two should be equal
        mse(i, k) = bias_squared(i, k) + variance(i, k);
        mse2(i,k) = mean(mean((repmat(y_act, 1, n) - y_hat).^2));

        % plot y_hat
        plot(x, y_hat(:,1), colors(i));
        title([num2str(samples(k)) ' samples']);
        xlabel('x');
        ylabel('y');
    end
    legend('ln(1/x)', '1st order', '2nd order', '3rd order', '4th order', '5th
end

bias_squared;
variance;
mse;
mse2;
```