Linear Regression using Machine Learning

Yi Qiang Ji Zhang
Biel Galiot Pérez
Gerard Villalta Quintana
Oriol Miras Robles

Professor: Alex Ferrer Ferre

Aerospace Engineering
Numerical Tools in Machine Learning for Aeronautical Engineering
Polytechnic University of Catalonia — BarcelonaTech

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Date: 11 May 2021

I. Introduction

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable (data), and the other is considered to be a dependent variable (outcome). It is used to predict values within a continuous range rather than trying to classify them into categories [1] [2].

There are two main types:

- **Simple regression**: Simple linear regression uses traditional slope-intercept form, where m and n are the variables the algorithm will try to "learn" to produce the most accurate predictions. x represents the input data and y represents the algorithm's prediction.
$$y = mx + n \tag{1}$$
- **Multi-variable regression**: This is a more complex linear regression. Where w_i represents the coefficients, or weights, that the algorithm model will try to "learn".
$$f(x_i) = \sum_{i}^{N} w_i x_j \quad \forall\, i, j \in \mathbb{N} \tag{2}$$
and x_j represents the different variables.

II. Linear Regression

In order to implement a linear regression predictor, as a first approach, a simple linear regression without an independent parameter n will be implemented and so the model will simply output

$$y = mx \tag{3}$$

Where m is a vector of parameters (weights) and x is the vector of input data. In order to do so, a set of data is randomly generated using the function `generateData`. This function is set to generate a quadratic set of data with some error that is artificially added.

Secondly, 25% of the data points are reserved in order to be used to perform the test of our model, while the rest is set to feed the training of the linear regression model. This is done with `splitDataTrainAndTest`.

Once the data has been separated, the model itself is finally trained, by minimizing the MSE of the data destined to train. Attending to [3], this is done following the following procedure

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\text{train}} = 0 \tag{4}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} \left\| \hat{\boldsymbol{y}}^{(\text{train})} - \boldsymbol{y}^{(\text{train})} \right\|_2^2 = 0 \tag{5}$$

$$\Rightarrow \frac{1}{m} \nabla_{\boldsymbol{w}} \left\| \boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right\|_2^2 = 0 \tag{6}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right)^{\top} \left(\boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right) = 0 \tag{7}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{w}^{\top} \boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \boldsymbol{w} - 2 \boldsymbol{w}^{\top} \boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} + \boldsymbol{y}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} \right) = 0 \tag{8}$$

$$\Rightarrow 2 \boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \boldsymbol{w} - 2 \boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} = 0 \tag{9}$$

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \right)^{-1} \boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} \tag{10}$$

Which finally leads to the expression used in the `computeW` function

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \right)^{-1} \boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} \tag{11}$$

With the vector of $\boldsymbol{w}$, the predictor function can be obtained simply by means of equation (3).

Finally the MSE of the predicted vs actual datapoints for the $\boldsymbol{y}^{(\text{train})}$ is computed with the function `computeMSE`, which in turn uses MATLAB's `immse` function.

III. K-Fold Cross-Validation

Cross-validation techniques are used to train algorithms when the information given by the datasets is not large enough. Thus, splitting this datasets in different permutations of training and test sets allows to reuse the original data and improve the performance of the algorithm.

One of these techniques is K-Fold Cross-Validation. This method splits the dataset in k number of folds, which will create k mutually exclusive subsets, whose union results on the original dataset.

Algorithm 1 KFold Algorithm

Define KFoldXV$(\mathbb{D}, A, L, k)$
Require: $\mathbb{D}$, the given dataset, with elements $\boldsymbol{z}^{(i)}$
Require: A, the learning algorithm, seen as a function that takes a dataset as input and outputs a learned function
Require: L, the loss function, seen as a function from a learned function f and an example $\boldsymbol{z}^{(i)} \in \mathbb{D}$ to a scalar $\in \mathbb{R}$
Require: k, the number of folds
for i from 1 to k **do**
  $f_i = A\left(\mathbb{D} \backslash \mathbb{D}_i\right)$
  for $\boldsymbol{z}^{(j)}$ in $\mathbb{D}_i$ **do**
    $e_j = L\left(f_i, \boldsymbol{z}^{(j)}\right)$
  end for
end for

Each of this combinations of test subsets, with their own training subsets (the remaining values for each one) will return an error. The total error of the algorithm is calculated with the mean value of the k number of errors.

$$E = \frac{1}{K} \sum_{i=1}^{K} E_i \tag{12}$$

IV. Results

To begin with, the following report tries random points of a parabolic curve

$$y = x^2 \tag{13}$$

for values of $x \in (-1, 1)$

A. Figure 1

MSE plot (mean and standard deviation) based on training set / test set ratio.

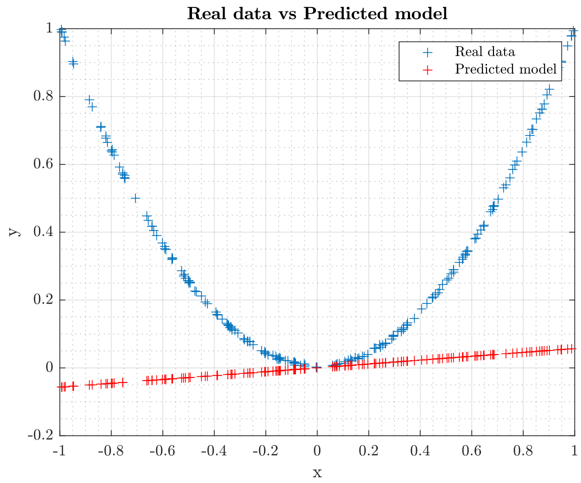

Fig. 1. Real data vs Predicted model

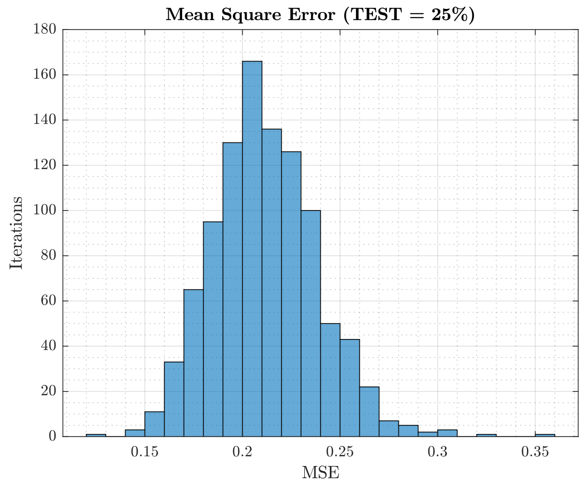

Fig. 2. Histogram for a Test ratio of (25%)

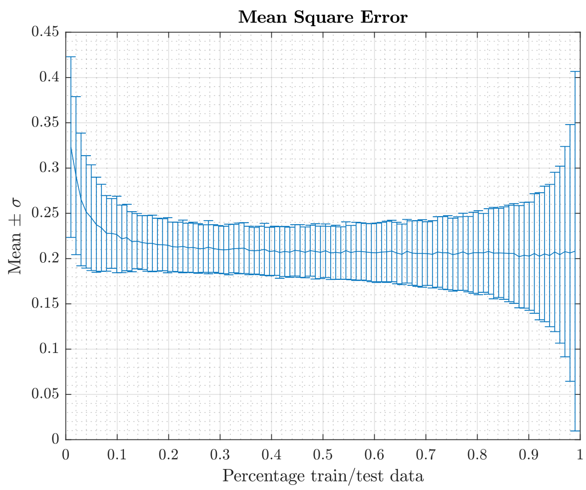

Fig. 3. MSE as a function of the Training/test ratio

B. Figure 2

MSE plot (mean and standard deviation) based on nData for a specific training set / test set ratio value.

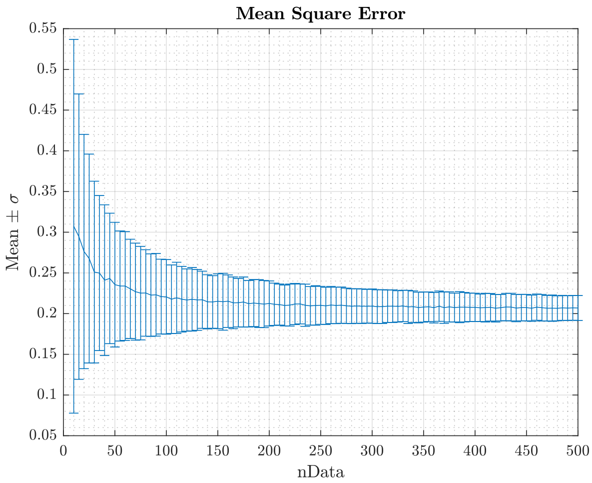

Fig. 4. MSE for a Test ratio of (25%) for different values of **nData**.

C. *Figure 3*

MSE plot (mean and standard deviation) based on the number of folds using the **k-fold** algorithm.

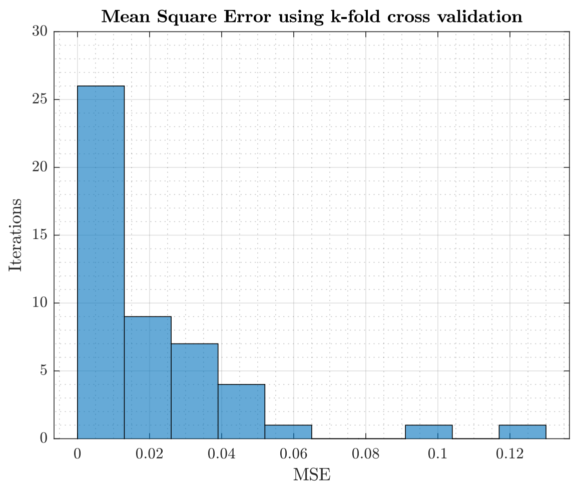

Fig. 5. Histogram as a function of folds using **k-fold** algorithm.

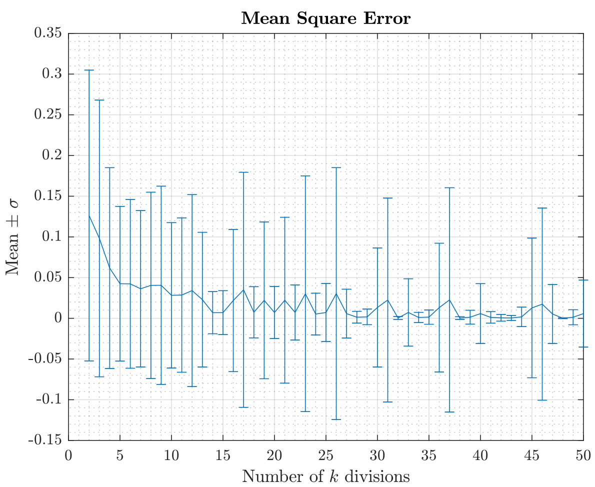

Fig. 6. MSE as a function of folds using **k-fold** algorithm.

D. *Figure 4*

MSE plot (mean and standard deviation) based on **nData** for a specific value of folds.

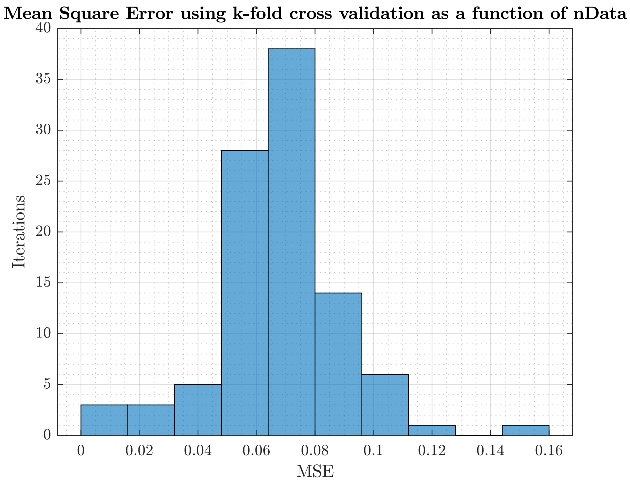

Fig. 7. Histogram as a function of **nData** for a specific value of folds.

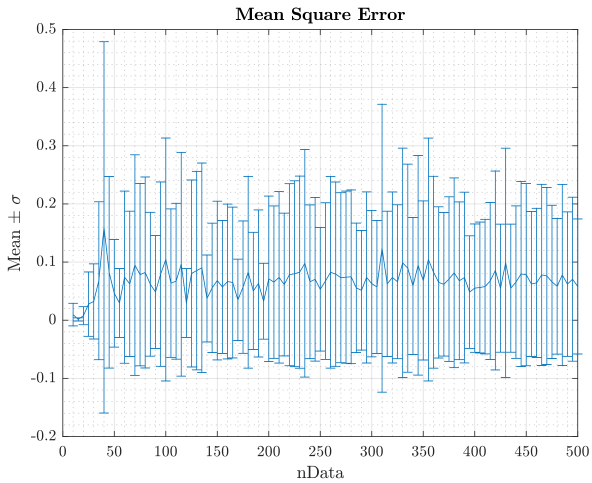

Fig. 8. MSE as a function of **nData** for a specific value of folds.

E. *Extra*

Same problem with the term constant (i.e. add an unknown bias term b within the parameters to be found). This model is known as affine.

Comparison of the two models (linear and affine)

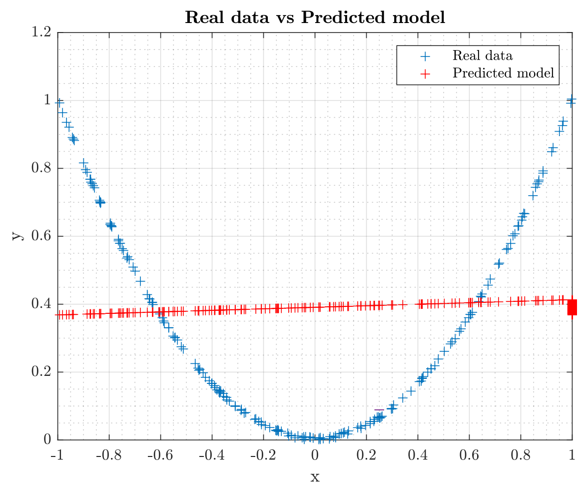

Fig. 9. Real data vs Predicted affine model.

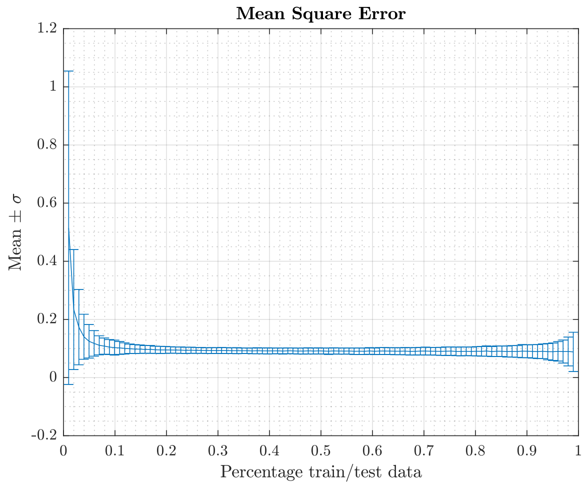

Fig. 10. MSE for affine model as a function of the Training/test ratio.

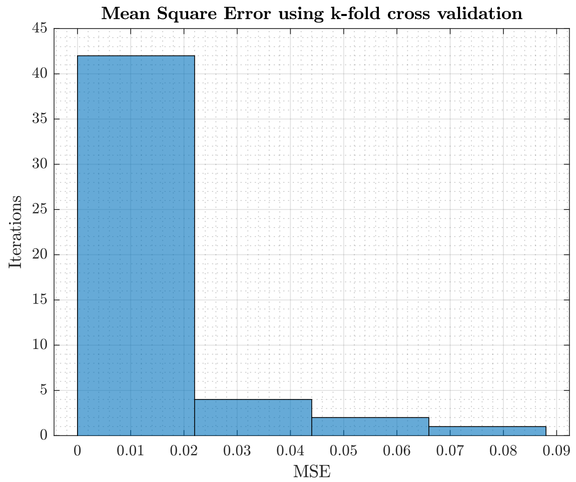

Fig. 11. Histogram for affine model using `k-fold` algorithm.

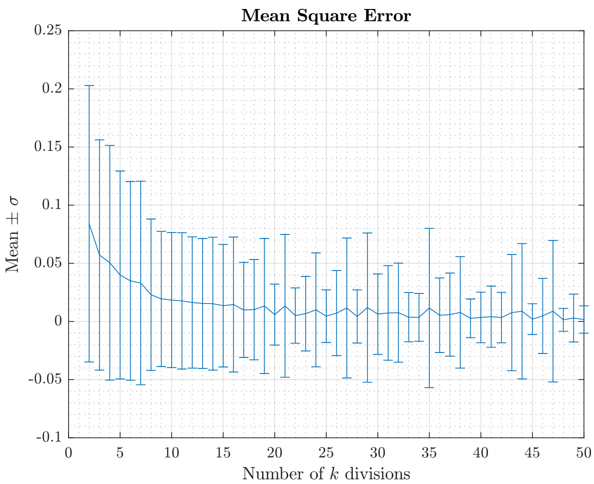

Fig. 12. MSE as a function of folds using `k-fold` algorithm.

V. Conclusions

This report shows different methods for predicting data using the linear regression method, along with the errors associated to it.

Comparing affine model and the linear one, it is clear how the error diminishes more rapidly in the latter. This is evidenced by comparing Figure 3 and Figure 10. The MSE when using the affine model is not only appreciably lower, but also much more defined, in the sense that the σ is very close to 0 between ratios of train/test data 0.3 to 0.6.

In both the affine and linear models, comparing Figures 5 and 6 fom the linear model and their analogous Figures for the affine model 11 and 12, it can be seen that the K-Fold algorithm is somewhat effective in reducing the MSE, as it is diminished with an increasing number of folds. This, however, does not apply to its standard deviation, which does not have an apparent correlation with the number of folds in either of the models.

In summary, with the different conclusions extracted from the aforementioned figures, along with the general result presented in Figures 1 and 9, it can be stated that the affine model better predicts the data generated by a quadratic curve with some randomized error. Futhermore, both models benefited from the use of the K-Fold Cros-Validation in its training.

References

[1] Yale University. *Linear Regression*. 2021. URL: http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm.

[2] Machine Learning Glossary. *Linear Regression*. 2021. URL: https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html#:~:text=Linear%5C%20Regression%5C%20is%5C%20a%5C%20supervised,Simple%5C%20regression.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. The MIT Press, 2017.

VI. Code

A. Figure 1

```
%-------------------------------------------------------------------------%
% Assignment 2: Linear Regression using Machine Learning
%-------------------------------------------------------------------------%

% Date: 11/05/2021
% Author/s:
% Yi Qiang Ji
% Biel Galiot
% Gerar Villalta
% Oriol Miras

% Subject: Robotic Exploration of the Solar System
% Professor: Manel Soria & Arnau Miro

clc;
close all;
clear all;

% Set interpreter to latex
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(groot,'defaulttextinterpreter','latex');
set(groot,'defaultLegendInterpreter','latex');


% Loop for every iteration
iter = 1000;
MSE = zeros(1,iter);

for i=1:1:iter
   % Number of points
   nData = 200;

   % Generate data
   [xdata,ydata] = generateData(nData);

   % Split train and test data
   [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData);

   % Compute coefficients
   w = computeW(Xtrain,Ytrain);

   % Predicted Y
   YtestPredicted = predictor(w,Xtest);

   % Minimum Squared Error
   MSE(i) = computeMSE(YtestPredicted,Ytest);

% % Plots
% figure(1)
% plot(xdata,ydata,'+');
% hold on
% plot(Xtest,YtestPredicted,'r+')

end

% plot_pdf = figure(1);
% grid on;
% grid minor;
% box on;
% xlabel('x');
% ylabel('y');
% title('\textbf{Real data vs Predicted model}');
% legend('Real data', 'Predicted model');

% % Save pdf
% set(plot_pdf, 'Units', 'Centimeters');
% pos = get(plot_pdf, 'Position');
% set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf, 'data_vs_predicted.pdf', '-dpdf', '-r0')

% Save png
% print(gcf,'data_vs_predicted.png','-dpng','-r600');

plot_pdf2 = figure(2);
histogram(MSE);
grid on;
grid minor;
```

```
box on;
xlabel('MSE');
ylabel('Iterations');
title('\textbf{Mean Square Error (TEST = 25\%)}');

% % Save pdf
% set(plot_pdf2, 'Units', 'Centimeters');
% pos = get(plot_pdf2, 'Position');
% set(plot_pdf2, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf2, 'MSE_25_test.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_25_test.png','-dpng','-r600');

% Functions
function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData)

   % Split data
   % Holdout validation percentage
   h_validation = 0.25; % Percentage

   train_data = round(h_validation*nData);
   % test_data = nData - train_data;

   Xtrain = xdata(1:train_data);
   Ytrain = ydata(1:train_data);
   Xtest = xdata((train_data+1):end);
   Ytest = ydata((train_data+1):end);
end

function [xdata,ydata] = generateData(nData)

   % Uniform distributed values between 'init' and 'final'
   % r = a + (b-a).*rand(N,1)

   % Initial and final range
   init = -1;
   final = 1;

   % Generate data
   xdata = -init + (init-(final))*rand(nData,1);
   ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
end

function MSE = computeMSE(YtestPredicted,Ytest)
   MSE = immse(Ytest,YtestPredicted);
end

function w = computeW(Xtrain,Ytrain)
   w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
end

function y = predictor(w,x)
   y = w*x;
end

function y = f(x)
y = -x.^2;
end
```

B. Figure 1.1

```
%-------------------------------------------------------------------------%
% Assignment 2: Linear Regression using Machine Learning
%-------------------------------------------------------------------------%

% Date: 11/05/2021
% Author/s:
% Yi Qiang Ji
% Biel Galiot
% Gerar Villalta
% Oriol Miras

% Subject: Robotic Exploration of the Solar System
% Professor: Manel Soria & Arnau Miro

clc;
close all;
clear all;
```

```
% Set interpreter to latex
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(groot,'defaulttextinterpreter','latex');
set(groot,'defaultLegendInterpreter','latex');


% Loop for 1000 times
iter = 1000;

% Loop for each percentage
p_number = 100; % ∆of increment
percentage = linspace(0.01,0.99,p_number); % From [0,1]
MSE = zeros(p_number,iter);

% Loop for every percentage
for j = 1:1:length(percentage)

   for i=1:1:iter

      % Number of points
      nData = 200;

      % Generate data
      [xdata,ydata] = generateData(nData);

      % Split train and test data
      [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData,percentage(j));

      % Compute coefficients
      w = computeW(Xtrain,Ytrain);

      % Predicted Y
      YtestPredicted = predictor(w,Xtest);

      % Minimum Squared Error
      MSE(j,i) = computeMSE(YtestPredicted,Ytest);

      % Plots
% plot(xdata,ydata,'+');
% hold on
% plot(Xtest,YtestPredicted,'r+')
   end
   mean_MSE(1,j) = mean(MSE(j,:));
   std_MSE(1,j) = std(MSE(j,:));
end

plot_pdf = figure(1);
errorbar(percentage,mean_MSE(1,:),std_MSE(1,:));
grid on;
grid minor;
box on;
xlabel('Percentage train/test data');
ylabel('Mean $\pm$ $\sigma$');
title('\textbf{Mean Square Error}');

% % Save pdf
% set(plot_pdf, 'Units', 'Centimeters');
% pos = get(plot_pdf, 'Position');
% set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf, 'MSE_error.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_error.png','-dpng','-r600');

% Functions
function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData,percentage)

   % Split data
   % Holdout validation percentage
   h_validation = percentage; % Percentage

   train_data = round(h_validation*nData);
   % test_data = nData - train_data;

   Xtrain = xdata(1:train_data);
   Ytrain = ydata(1:train_data);
   Xtest = xdata((train_data+1):end);
   Ytest = ydata((train_data+1):end);
end

function [xdata,ydata] = generateData(nData)

```

```
    % Uniform distributed values between 'init' and 'final'
    % r = a + (b-a).*rand(N,1)

    % Initial and final range
    init = -1;
    final = 1;

    % Generate data
    xdata = -init + (init-(final))*rand(nData,1);
    ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
end

function MSE = computeMSE(YtestPredicted,Ytest)
    MSE = immse(Ytest,YtestPredicted);
end

function w = computeW(Xtrain,Ytrain)
    w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
end

function y = predictor(w,x)
    y = w*x;
end


function y = f(x)
y = -x.^2;
end
```

C. Figure 2

```

clc;
close all;
clear all;

% Loop for 1000 times
iter = 1000;

% Loop for each nData
data = 10:5:500; % From [10,500]
MSE = zeros(length(data),iter);

% Loop for every nData
for j = 1:1:length(data)

    for i=1:1:iter

        % Number of points
        nData = data(j);

        % Generate data
        [xdata,ydata] = generateData(nData);

        % Split train and test data
        [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData);

        % Compute coefficients
        w = computeW(Xtrain,Ytrain);

        % Predicted Y
        YtestPredicted = predictor(w,Xtest);

        % Minimum Squared Error
        MSE(j,i) = computeMSE(YtestPredicted,Ytest);

    end
    mean_MSE(1,j) = mean(MSE(j,:));
    std_MSE(1,j) = std(MSE(j,:));
end

plot_pdf = figure(1);
errorbar(data,mean_MSE(1,:),std_MSE(1,:));
grid on;
grid minor;
box on;
xlabel('nData');
ylabel('Mean $\pm$ $\sigma$');
title('\textbf{Mean Square Error}');

```

```
% % Save pdf
% set(plot_pdf, 'Units', 'Centimeters');
% pos = get(plot_pdf, 'Position');
% set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf, 'MSE_error_ndata_25.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_error_ndata_25.png','-dpng','-r600');

% Functions
function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData)

   % Split data
   % Holdout validation percentage
   h_validation = 0.25; % Percentage

   train_data = round(h_validation*nData);
   % test_data = nData - train_data;

   Xtrain = xdata(1:train_data);
   Ytrain = ydata(1:train_data);
   Xtest = xdata((train_data+1):end);
   Ytest = ydata((train_data+1):end);
end

function [xdata,ydata] = generateData(nData)

   % Initial and final range
   init = -1;
   final = 1;

   % Generate data
   xdata = -init + (init-(final))*rand(nData,1);
   ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
end

function MSE = computeMSE(YtestPredicted,Ytest)
   MSE = immse(Ytest,YtestPredicted);
end

function w = computeW(Xtrain,Ytrain)
   w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
end

function y = predictor(w,x)
   y = w*x;
end


function y = f(x)
y = -x.^2;
end
```

D. Figure 3

```
%-----------------------------------------------------------------------%
% Assignment 2: Linear Regression using Machine Learning
%-----------------------------------------------------------------------%

% Date: 11/05/2021
% Author/s:
% Yi Qiang Ji
% Biel Galiot
% Gerar Villalta
% Oriol Miras

% Subject: Robotic Exploration of the Solar System
% Professor: Manel Soria & Arnau Miro

clc;
close all;
clear all;

% Set interpreter to latex
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(groot,'defaulttextinterpreter','latex');
set(groot,'defaultLegendInterpreter','latex');


```

```
% Divisions
k_div = 2:1:50;

% MSE vector
MSE = zeros(1,length(k_div));

% Number of points
nData = 200;

% Generate data
[xdata,ydata] = generateData(nData);


for j=1:1:length(k_div)
   % Loop for every division
   for i=1:1:k_div(j)

      MSE_div = zeros(k_div(j),1);
      % Split train and test data
      [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, i, k_div(j));

      % Compute coefficients
      w = computeW(Xtrain,Ytrain);

      % Predicted Y
      YtestPredicted = predictor(w,Xtest);

      % Minimum Squared Error
      MSE_div(i,:) = computeMSE(YtestPredicted,Ytest);
   % % Plots
   % figure(1)
   % plot(xdata,ydata,'+');
   % hold on
   % plot(Xtest,YtestPredicted,'r+')
   end
      MSE(1,j) = mean(MSE_div);
      mean_MSE = MSE;
      std_MSE(1,j) = std(MSE_div);
      hold on;
end

plot_pdf = figure(1);
histogram(MSE,10);
grid on;
grid minor;
box on;
xlabel('MSE');
ylabel('Iterations');
title('\textbf{Mean Square Error using k-fold cross validation}');

% % Save pdf
% set(plot_pdf, 'Units', 'Centimeters');
% pos = get(plot_pdf, 'Position');
% set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf, 'MSE_kfold.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_kfold.png','-dpng','-r600');

plot_pdf2 = figure(2);
errorbar(k_div,mean_MSE(1,:),std_MSE(1,:));
grid on;
grid minor;
box on;
xlabel('Number of $k$ divisions');
ylabel('Mean $\pm$ $\sigma$');
title('\textbf{Mean Square Error}');

% % Save pdf
% set(plot_pdf2, 'Units', 'Centimeters');
% pos = get(plot_pdf2, 'Position');
% set(plot_pdf2, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf2, 'MSE_kfold_error.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_kfold_error.png','-dpng','-r600');

% Functions
function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, index, k_div)

   % Split data
   % Holdout validation percentage
```

```
    % h_validation = 0.25; % Percentage

    % train_data = round(h_validation*nData);
    % test_data = nData - train_data;

    div = floor(nData/k_div);
    Xtest = xdata(((index-1)*div + 1):index*div);
    Ytest = ydata(((index-1)*div + 1):index*div);
    Xtrain = setdiff(xdata,Xtest);
    Ytrain = setdiff(ydata,Ytest);
end

function [xdata,ydata] = generateData(nData)

    % Uniform distributed values between 'init' and 'final'
    % r = a + (b-a).*rand(N,1)

    % Initial and final range
    init = -1;
    final = 1;

    % Generate data
    xdata = -init + (init-(final))*rand(nData,1);
    ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
end

function MSE = computeMSE(YtestPredicted,Ytest)
    MSE = immse(Ytest,YtestPredicted);
end

function w = computeW(Xtrain,Ytrain)
    w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
end

function y = predictor(w,x)
    y = w*x;
end


function y = f(x)
y = -x.^2;
end
```

E. *Figure 4*

```
%---------------------------------------------------------------------%
% Assignment 2: Linear Regression using Machine Learning
%---------------------------------------------------------------------%

% Date: 11/05/2021
% Author/s:
% Yi Qiang Ji
% Biel Galiot
% Gerar Villalta
% Oriol Miras

% Subject: Robotic Exploration of the Solar System
% Professor: Manel Soria & Arnau Miro

clc;
close all;
clear all;

% Set interpreter to latex
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(groot,'defaulttextinterpreter','latex');
set(groot,'defaultLegendInterpreter','latex');


% Divisions
k_div = 4;

% MSE_vector
MSE = zeros(1,length(k_div));

% Number of points
nData = 10:5:500; % From [10,500]


for j=1:1:length(nData)
```

```

    % Generate data
    [xdata,ydata] = generateData(nData(j));

    % Loop for every division
    for i=1:1:k_div

       MSE_div = zeros(k_div,1);

       % Split train and test data
       [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData(j), i, k_div);

       % Compute coefficients
       w = computeW(Xtrain,Ytrain);

       % Predicted Y
       YtestPredicted = predictor(w,Xtest);

       % Minimum Squared Error
       MSE_div(i,:) = computeMSE(YtestPredicted,Ytest);
    % % Plots
    % figure(1)
    % plot(xdata,ydata,'+');
    % hold on
    % plot(Xtest,YtestPredicted,'r+')
    end
       MSE(1,j) = mean(MSE_div);
       mean_MSE = MSE;
       std_MSE(1,j) = std(MSE_div);
       hold on;
end

plot_pdf = figure(1);
histogram(MSE,10);
grid on;
grid minor;
box on;
xlabel('MSE');
ylabel('Iterations');
title('\textbf{Mean Square Error using k-fold cross validation as a function of nData}');

% % Save pdf
% set(plot_pdf, 'Units', 'Centimeters');
% pos = get(plot_pdf, 'Position');
% set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos( 4)]);
% print(plot_pdf, 'MSE_kfold_nData.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_kfold_nData.png','-dpng','-r600');

plot_pdf2 = figure(2);
errorbar(nData,mean_MSE(1,:),std_MSE(1,:));
grid on;
grid minor;
box on;
xlabel('nData');
ylabel('Mean $\pm$ $\sigma$');
title('\textbf{Mean Square Error}');

% % Save pdf
% set(plot_pdf2, 'Units', 'Centimeters');
% pos = get(plot_pdf2, 'Position');
% set(plot_pdf2, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf2, 'MSE_kfold_nData_error.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_kfold_nData_error.png','-dpng','-r600');

% Functions
function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, index, k_div)

    % Split data
    % Holdout validation percentage
    % h_validation = 0.25; % Percentage

    % train_data = round(h_validation*nData);
    % test_data = nData - train_data;

    div = floor(nData/k_div);
    Xtest = xdata(((index-1)*div + 1):index*div);
    Ytest = ydata(((index-1)*div + 1):index*div);
    Xtrain = setdiff(xdata,Xtest);
```

```
    Ytrain = setdiff(ydata,Ytest);
end

function [xdata,ydata] = generateData(nData)

    % Uniform distributed values between 'init' and 'final'
    % r = a + (b-a).*rand(N,1)

    % Initial and final range
    init = -1;
    final = 1;

    % Generate data
    xdata = -init + (init-(final))*rand(nData,1);
    ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
end

function MSE = computeMSE(YtestPredicted,Ytest)
    MSE = immse(Ytest,YtestPredicted);
end

function w = computeW(Xtrain,Ytrain)
    w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
end

function y = predictor(w,x)
    y = w*x;
end


function y = f(x)
y = -x.^2;
end
```

F. *Extra 1*

```
%-----------------------------------------------------------------------%
% Assignment 2: Linear Regression using Machine Learning
%-----------------------------------------------------------------------%

% Date: 11/05/2021
% Author/s:
% Yi Qiang Ji
% Biel Galiot
% Gerar Villalta
% Oriol Miras

% Subject: Robotic Exploration of the Solar System
% Professor: Manel Soria & Arnau Miro

clc;
close all;
clear all;

% Set interpreter to latex
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(groot,'defaulttextinterpreter','latex');
set(groot,'defaultLegendInterpreter','latex');


% Loop for 1000 times
iter = 1000;

% Loop for each percentage
p_number = 100; % Δof increment
percentage = linspace(0.01,0.99,p_number); % From [0,1]
MSE = zeros(p_number,iter);

% Loop for every percentage
for j = 1:1:length(percentage)

    for i=1:1:iter

        % Number of points
        nData = 200;

        % Generate data
        [xdata,ydata] = generateData(nData);

        % Split train and test data
```

```
        [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData,percentage(j));

        % Compute coefficients
        w = computeW(Xtrain,Ytrain);

        % Predicted Y
        YtestPredicted = predictor(w,Xtest);

        % Minimum Squared Error
        MSE(j,i) = computeMSE(YtestPredicted,Ytest);

        % Plots
% plot(xdata,ydata,'+');
% hold on
% plot(Xtest,YtestPredicted,'r+')
    end
    mean_MSE(1,j) = mean(MSE(j,:));
    std_MSE(1,j) = std(MSE(j,:));
end

plot_pdf = figure(1);
errorbar(percentage,mean_MSE(1,:),std_MSE(1,:));
grid on;
grid minor;
box on;
xlabel('Percentage train/test data');
ylabel('Mean $\pm$ $\sigma$');
title('\textbf{Mean Square Error}');

% % Save pdf
% set(plot_pdf, 'Units', 'Centimeters');
% pos = get(plot_pdf, 'Position');
% set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf, 'MSE_afi.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_afi.png','-dpng','-r600');

% Functions
function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData,percentage)

    % Split data
    % Holdout validation percentage
    h_validation = percentage; % Percentage

    train_data = round(h_validation*nData);
    % test_data = nData - train_data;

    Xtrain = xdata(1:train_data);
    Xtrain = [xdata(1:train_data) ones(size(Xtrain,1),1)];
    Ytrain = ydata(1:train_data);

    Xtest = xdata((train_data+1):end);
    Xtest = [xdata((train_data+1):end) ones(size(Xtest,1),1)];
    Ytest = ydata((train_data+1):end);
end

function [xdata,ydata] = generateData(nData)

    % Uniform distributed values between 'init' and 'final'
    % r = a + (b-a).*rand(N,1)

    % Initial and final range
    init = -1;
    final = 1;

    % Generate data
    xdata = -init + (init-(final))*rand(nData,1);
    ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
end

function MSE = computeMSE(YtestPredicted,Ytest)
    MSE = immse(Ytest,YtestPredicted);
end

function w = computeW(Xtrain,Ytrain)
    w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
end

function y = predictor(w,x)
    y = x*w;
end
```

```

function y = f(x)
y = -x.^2;
end
```

G. Extra 2

```
%-------------------------------------------------------------------------%
% Assignment 2: Linear Regression using Machine Learning
%-------------------------------------------------------------------------%

% Date: 11/05/2021
% Author/s:
% Yi Qiang Ji
% Biel Galiot
% Gerar Villalta
% Oriol Miras

% Subject: Robotic Exploration of the Solar System
% Professor: Manel Soria & Arnau Miro

clc;
close all;
clear all;

% Set interpreter to latex
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(groot,'defaulttextinterpreter','latex');
set(groot,'defaultLegendInterpreter','latex');


% Divisions
k_div = 2:1:50;

% MSE vector
MSE = zeros(1,length(k_div));

% Number of points
nData = 200;

% Generate data
[xdata,ydata] = generateData(nData);


for j=1:1:length(k_div)
    % Loop for every division
    for i=1:1:k_div(j)

        MSE_div = zeros(k_div(j),1);
        % Split train and test data
        [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, i, k_div(j));

        % Compute coefficients
        w = computeW(Xtrain,Ytrain);

        % Predicted Y
        YtestPredicted = predictor(w,Xtest);

        % Minimum Squared Error
        MSE_div(i,:) = computeMSE(YtestPredicted,Ytest);
    % % Plots
    % figure(1)
    % plot(xdata,ydata,'+');
    % hold on
    % plot(Xtest,YtestPredicted,'r+')
    end
        MSE(1,j) = mean(MSE_div);
        mean_MSE = MSE;
        std_MSE(1,j) = std(MSE_div);
        hold on;
end

plot_pdf = figure(1);
histogram(MSE);
grid on;
grid minor;
box on;
xlabel('MSE');
ylabel('Iterations');
title('\textbf{Mean Square Error using k-fold cross validation}');
```

```
% % Save pdf
% set(plot_pdf, 'Units', 'Centimeters');
% pos = get(plot_pdf, 'Position');
% set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf, 'MSE_afi_kfold.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_afi_kfold.png','-dpng','-r600');

plot_pdf2 = figure(2);
errorbar(k_div,mean_MSE(1,:),std_MSE(1,:));
grid on;
grid minor;
box on;
xlabel('Number of $k$ divisions');
ylabel('Mean $\pm$ $\sigma$');
title('\textbf{Mean Square Error}');

% % Save pdf
% set(plot_pdf2, 'Units', 'Centimeters');
% pos = get(plot_pdf2, 'Position');
% set(plot_pdf2, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
% 'PaperSize',[pos(3), pos(4)]);
% print(plot_pdf2, 'MSE_afi_kfold_error.pdf', '-dpdf', '-r0')
%
% % Save png
% print(gcf,'MSE_afi_kfold_error.png','-dpng','-r600');

% Functions
function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, index, k_div)

   % Split data
   % Holdout validation percentage
   % h_validation = 0.25; % Percentage

   % train_data = round(h_validation*nData);
   % test_data = nData - train_data;

   div = floor(nData/k_div);
   Xtest = xdata(((index-1)*div + 1):index*div);
   Xtest = [xdata(((index-1)*div + 1):index*div) ones(size(Xtest,1),1)];
   Ytest = ydata(((index-1)*div + 1):index*div);

   Xtrain = setdiff(xdata,Xtest);
   Xtrain = [setdiff(xdata,Xtest) ones(size(Xtrain,1),1)];
   Ytrain = setdiff(ydata,Ytest);
end

function [xdata,ydata] = generateData(nData)

   % Uniform distributed values between 'init' and 'final'
   % r = a + (b-a).*rand(N,1)

   % Initial and final range
   init = -1;
   final = 1;

   % Generate data
   xdata = -init + (init-(final))*rand(nData,1);
   ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
end

function MSE = computeMSE(YtestPredicted,Ytest)
   MSE = immse(Ytest,YtestPredicted);
end

function w = computeW(Xtrain,Ytrain)
   w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
end

function y = predictor(w,x)
   y = x*w;
end


function y = f(x)
y = -x.^2;
end
```