

# MATLAB Companion Script for *Machine Learning* ex1 (Optional)

## Introduction

Coursera's *Machine Learning* was designed to provide you with a greater understanding of machine learning algorithms- what they are, how they work, and where to apply them. You are also shown techniques to improve their performance and to address common issues. As is mentioned in the course, there are many tools available that allow you to use machine learning algorithms *without* having to implement them yourself. This Live Script was created by MathWorks to help *Machine Learning* students explore the data analysis and machine learning tools available in MATLAB.

## FAQ

### Who is this intended for?

- This script is intended for students using MATLAB Online who have completed ex1 and want to learn more about the corresponding machine learning tools in MATLAB.

### How do I use this script?

- In the sections that follow, read the information provided about the data analysis and machine learning tools in MATLAB, then run the code in each section and examine the results. You may also be presented with instructions for using a MATLAB machine learning app. This script should be located in the ex1 folder which should be set as your Current Folder in MATLAB Online.

### Can I use the tools in this companion script to complete the programming exercises?

- No. Most algorithm steps implemented in the programming exercises are handled automatically by MATLAB machine learning functions. Additionally, the results will be similar, but not identical, to those in the programming exercises due to differences in implementation, parameter settings, and randomization.

### Where can I obtain help with this script or report issues?

- As this script is not part of the original course materials, please direct any questions, comments, or issues to the *MATLAB Help* discussion forum.

## Linear Regression

In this script, linear regression models for both single and multiple variables are implemented using the `fitlm` function from the [Statistics and Machine Learning Toolbox](#). A quick tutorial is also included on using the *Regression Learner App*, which provides a graphical interface for creating regression models.

## Files needed for this script

- ex1data1.txt - Dataset for linear regression with one variable
- ex1data2.txt - Dataset for linear regression with multiple variables

## Table of Contents

MATLAB Companion Script for Machine Learning ex1 (Optional).....	1
Introduction.....	1
FAQ.....	1
Linear Regression.....	1
Files needed for this script.....	1
Linear Regression with One Variable.....	2
Load the data into a table.....	2
Fit a linear model using the fitlm function.....	4
Making predictions using the LinearModel variable.....	5
Linear Regression with Multiple Variables.....	6
Load and preview the data.....	6
Fit a linear model using fitlm and estimate housing prices.....	7
Using the Regression Learner App.....	7
Open the app and select the variables.....	7
Select and train the model.....	8
Evaluate the model.....	8
Export the model.....	8

## Linear Regression with One Variable

This section covers the MATLAB implementation of single variable linear regression corresponding to the first part of ex1. Recall that the file ex1data1.txt contains two columns of data: the first column corresponds to the populations of cities and the second column contains the profit of food trucks in those cities.

### Load the data into a table

The `table` datatype is now the preferred datatype for most data analysis and machine learning tasks in MATLAB. In this script we will use tables instead of vectors and matrices. A `table` consists of rows and columns where each column corresponds to a variable, and each row corresponds to an observation.

Run the code below to:

- Read the profit data into a `table` using the `readtable` function
- Add descriptive names to the `table` variables
- Compute summary statistics on each variable
- Display the methods available for working with `table` variables.

```
clear;
data = readtable('ex1data1.txt'); % read tabular data into a table
data.Properties.VariableNames = {'Population', 'Profit'} % name the variables
```

```
data = 97x2 table
```

	Population	Profit
1	6.1101	17.5920
2	5.5277	9.1302
3	8.5186	13.6620

	Population	Profit
4	7.0032	11.8540
5	5.8598	6.8233
6	8.3829	11.8860
7	7.4764	4.3483
8	8.5781	12
9	6.4862	6.5987
10	5.0546	3.8166
11	5.7107	3.2522
12	14.1640	15.5050
13	5.7340	3.1551
14	8.4084	7.2258

⋮

summary(data)

Variables:

**Population:** 97×1 double

Values:

Min	5.0269
Median	6.5894
Max	22.203

**Profit:** 97×1 double

Values:

Min	-2.6807
Median	4.5623
Max	24.147

methods(data)

类 table 的方法:

addprop	convertvars	inner2outer	isempty	issortedrows	movevars	outerjoin	rmprop
addvars	head	innerjoin	ismember	join	ndims	removevars	rowfun
cat	horzcat	intersect	isprop	mergevars	numel	renamevars	rows2var

Use the help control below to view descriptions of the functions for working with table variables displayed above.

help table.size

**size** Size of a table.  
D = **size**(T) returns the two-element row vector D = [NROWS,NVARS] containing the number of rows and variables in the table T.

[NROWS,NVARS] = **size**(T) returns the number of rows and variables in the table T as separate output variables.

[M1,M2,M3,...,MN] = **size**(T), for N>1, returns the sizes of the first N dimensions of the table T. If the number of output arguments N does not equal NDIMS(T), then for:

N > NDIMS(T), **size** returns ones in the "extra" variables, i.e., outputs NDIMS(T)+1 through N.

N < NDIMS(T), MN contains the product of the sizes of dimensions N through NDIMS(T).

M = **size**(T,DIM) returns the lengths of the specified dimensions in a row vector. DIM can be a scalar or vector of dimensions. For example, **size**(T,1) returns the number of rows of T, and **size**(T,[1 2]) returns a row vector containing the number of rows and variables.

M = **size**(X,DIM1,DIM2,...,DIMN) returns the lengths of the dimensions DIM1,...,DIMN as a row vector.

[M1,M2,...,MN] = **size**(X,DIM) OR [M1,M2,...,MN] = **size**(X,DIM1,...,DIMN) returns the lengths of the specified dimensions as separate outputs. The number of outputs must equal the number of dimensions provided.

See also height, width, numel, ndims.

**table/size** 的帮助是从超类 `tabular` 继承的

[table/size 的文档](#)

## Fit a linear model using the `fitlm` function

There are many functions available in the Statistics and Machine Learning Toolbox for fitting a model to data. To fit a linear regression model to the data in our `table`, we'll use the `fitlm` function. As we will see, there is no need to add a column of ones to the data for a bias term, to create a separate cost function, or to implement gradient descent to converge on the model parameters as in ex1- these tasks are automatically taken care of by `fitlm`. The output of `fitlm` is a `LinearModel` variable which contains all of the information about the model.

Run the code in this section to fit the linear model. The model coefficients ( $\theta$ ) are extracted from the model variable and printed out for you below- compare to your results from ex1. After running, double-click on the variable `linMdl` in the MATLAB workspace to examine its properties further. Variable properties can also be extracted and displayed using the `.'` operator, as in the code below used to extract the `Coefficients` property. The result is a table of the model coefficients and additional statistical information.

```
linMdl = fitlm(data);  
linMdl.Coefficients
```

ans = 2x4 table

	Estimate	SE	tStat	pValue
1 (Intercept)	-3.8958	0.7195	-5.4147	4.6079e-07
2 Population	1.1930	0.0797	14.9608	1.0232e-26

```
theta = linMdl.Coefficients.Estimate;  
fprintf('Theta computed by fitlm: [%f; %f]',theta(1),theta(2))
```

Theta computed by fitlm: [-3.895781; 1.193034]

## Making predictions using the LinearModel variable

Below we use the `predict` function to predict profit for different populations using the model trained in the previous section. Note that the first input to `predict` is a trained model variable, while the second input is a feature value or list of values in the form of a vector, matrix or table. Run the code in this section and compare with your results from ex1.

```
% Predict values for population sizes of 35,000 and 70,000
predict1 = predict(linMdl,3.5);
fprintf('For population = 35,000, we predict a profit of %f', predict1*10000);
```

For population = 35,000, we predict a profit of 2798.368764

```
predict2 = predict(linMdl,7);
fprintf('For population = 70,000, we predict a profit of %f', predict2*10000);
```

For population = 70,000, we predict a profit of 44554.546310

```
% Plot the linear fit
plot(data.Population,data.Profit,'rx'); hold on
profit = predict(linMdl,data);
plot(data.Population, profit, '-')
legend('Training data', 'Linear regression'); hold off
```

# Linear Regression with Multiple Variables

## Load and preview the data

Recall that the file `ex1data2.txt` contains a training set of housing prices in Portland, Oregon. Each row corresponds to a house where the element in the first column is the size of the house (in square feet), the second column is the number of bedrooms, and the third column is the price of the house. Run this section to load the data into a `table`. After a `table` is displayed as output you can perform basic sort and filter operations on a given columns as follows:

1. Hover the mouse pointer over the desired variable name (column heading)
2. Click on the down arrow when it appears
3. Choose the desired sort option to sort all rows in the table based on the value in that variable OR
4. Enter a specific range in the boxes provided to select only rows whose value for the selected variable falls inside that range
5. The MATLAB code required to perform 3 and/or 4 is automatically displayed below the `table`. If desired, the code can be copied to the clipboard or added to the script using the 'Copy' and 'Update' code buttons.

After running this section, experiment with sorting and filtering the data `table` below. (Since we want to use all the observations in data, there is no need to copy the code or add it to the script.)

```
clear;  
% Load Data  
data = readtable('ex1data2.txt');  
data.Properties.VariableNames = {'Size', 'Bedrooms', 'Price'}
```

data = 47×3 table

	Size	Bedrooms	Price
1	2104	3	399900
2	1600	3	329900
3	2400	3	369000
4	1416	2	232000
5	3000	4	539900
6	1985	4	299900
7	1534	3	314900
8	1427	3	198999
9	1380	3	212000
10	1494	3	242500
11	1940	4	239999
12	2000	3	347000
13	1890	3	329999

	Size	Bedrooms	Price
14	4478	5	699900
⋮			

## Fit a linear model using `fitlm` and estimate housing prices

The `fitlm` function can be used to fit a regression model with multiple variables as well. Note that the last column (variable) in the input table is considered the response variable by `fitlm` by default while all other variables are considered predictors. A different response variable can be specified, if desired- see the `fitlm` documentation for more information. Run this section to fit a linear model, display the model the coefficients, and predict the price of a 1650 sqft, 3-bedroom house. Compare with your results from ex1.

```
linMdl = fitlm(data);
theta = linMdl.Coefficients.Estimate;
fprintf('Theta computed by fitlm: [%f; %f]', theta(1), theta(2))
```

Theta computed by fitlm: [89597.909543; 139.210674]

```
price = predict(linMdl, [1650 3]); % Enter your price formula here
fprintf('Predicted price of a 1650 sq-ft, 3 br house: $%f', price);
```

Predicted price of a 1650 sq-ft, 3 br house: \$293081.464335

## Using the Regression Learner App

In this section, we reproduce the results of the previous section while providing an introduction to the [Regression Learner App](#). This app offers a graphical interface for building, training, and evaluating linear regression models. Run the code below to clear the workspace and reload the housing data, then follow the steps in the next few sections.

```
clear;
% Load Data
data = readtable('ex1data2.txt');
data.Properties.VariableNames = {'Size', 'Bedrooms', 'Price'};
```

**Note:** If you have difficulty reading the instructions below while the app is open in MATLAB Online, export this script to a pdf file which you can then use to display the instructions in a separate browser tab or window. To export this script, click on the 'Save' button in the 'Live Editor' tab above, then select 'Export to PDF'.

### Open the app and select the variables

1. In the MATLAB Apps tab, select the **Regression Learner** app from the Machine Learning section (you may need to expand the menu of available apps).
2. Select **'New Session -> From Workspace'** to start a new interactive session.
3. Under **'Data Set Variable'**, select **'data'** (if not already selected).
4. Under **'Response'** select **'From data set variable'** and **'Price'** (if not already selected).
5. Under **'Predictors'** select **'Size'** and **'Bedrooms'** (if not already selected).

6. Under '**Validation**' select '**Resubstitution Validation**'
7. Click the '**Start Session**' button.

## Select and train the model

There are many available models to choose from (the default model is 'Fine Tree'). To train a regression model corresponding to the one in the previous section obtained using `fitlm`:

1. Expand the model list and select '**Linear**' from the '**Linear Regression Models**' list.
2. Select '**Train**' to fit the model.

## Evaluate the model

After training, there are several options available for evaluating model performance.

- The training results, including error values, prediction speed, and training time for the selected model is shown in the '**Current Model**' pane.
- The model predictions for the training data are visualized in the '**Response Plot**', which is shown by default. To plot the response by variable, select the desired feature variable from the choices in '**X-axis**'.
- The '**Predicted vs. Actual**' and '**Residuals**' plots offer additional means of visualizing and evaluating model performance.

## Export the model

To export a linear regression model variable from the app into the workspace:

1. Select '**Export Model -> Export Model**'.
2. Select the default output variable name ('trainedModel').
3. Extract the linear model to the variable `LinMdl` by running the command below

```
linMdl = trainedModel.LinearModel
```

`linMdl` variable now contains the linear regression model. It can be used to predict housing prices using the `predict` function using the methods provided earlier for working with `LinearModel` variables returned from `fitlm`.