

Bootcamp Machine Learning



Day01 - Linear Regression

During this day you will learn the first concepts constituting the field of machine learning.

Notions of the day

Matrix operations, gradient descent, cost function, normal equation, MSE, RMSE R-score and learning rate.

General rules

- The version of Python to use is 3.7, you can check the version of Python with the following command: `python -V`
- The norm: during this bootcamp you will follow the Pep8 standards <https://www.python.org/dev/peps/pep-0008/>
- The function eval is never allowed.
- The exercises are ordered from the easiest to the hardest.
- Your exercises are going to be evaluated by someone else, so make sure that your variable names and function names are appropriate and civil.
- Your manual is the internet.
- You can also ask question in the dedicated channel in Slack: 42-ai.slack.com.
- If you find any issue or mistakes in the subject please create an issue on our dedicated repository on Github: https://github.com/42-AI/bootcamp_machine-learning/issues.

Helper

Ensure that you have the right Python version.

```
> which python
/goinfre/miniconda/bin/python
> python -V
Python 3.7.*
> which pip
/goinfre/miniconda/bin/pip
```

Exercise 00 - Hypothesis Concept in Machine Learning

Exercise 01 - Cost Function Concept

Exercise 02 - Descent Gradient

Exercise 03 - The Linear Regression with Class

Exercise 04 - Linear Regression

Exercise 05 - Multiples features and Linear Gradient Descent

Exercise 06 - Multiples features and Normal Equation

Exercise 07 (Bonus) - Learning Rate and Quadratic Hypothesis

Exercise 01 - Hypothesis Concept in Machine Learning

Turnin directory :	ex00
Files to turn in :	pred.py
Authorize module :	numpy,
Forbidden function :	None
Links :	https://www.coursera.org/learn/machine-learning/home/week/1
	https://www.coursera.org/learn/machine-learning/home/week/2
Hint :	Focus on the "Model and Cost Function" section of week 1
	and "Multivariate Linear Regression" section of week 2"

Objectives :

- Reinforce the mathematical skills tackled in **Mathematical Delights**, especially the **matrix-matrix operations**.
- Be able to explain what is an hypothesis in the machine learning context.
- Be able to implement a basic method based on the hypothesis function to obtain predicted output.

Instructions :

First of all, you have to get familiar with the concept of features, training set, training examples, output and hypothesis.

As hypothesis function h , you will choose:

$$\hat{y}^{(i)} = h(X^{(i)}) = \sum_{j=0}^N \theta_j \cdot x_j^{(i)}$$

Where $X^{(i)}$ is the i th training examples (lign vector of N components), θ_j is the j th component of the θ vector.

$\hat{y}^{(i)}$ is what we called the i th predicted output.

I would like to highlight: X is a matrix of dimension (M, N) and θ is a matrix of dimension $(N+1, 1)$. Thus, I let you search and do what it is necessary to allow the matrix product..

Then you will code a function named `.predict_` as per the instructions below:

```
def predict_(theta, X):
    """
    Description:
        Prediction of output using the hypothesis function (linear model).
    Args:
        theta: has to be a numpy.ndarray, a vector of dimension (number of
        features + 1, 1).
        X: has to be a numpy.ndarray, a matrix of dimension (number of
        training examples, number of features).
    Returns:
        pred: numpy.ndarray, a vector of dimension (number of the training
        examples, 1).
        None if X does not match the dimension of theta.
    Raises:
        This function should not raise any Exception.
    """
    ... your code here ...
```

Examples :

```
>>>import numpy as np
>>>X1 = np.array([[0.], [1.], [2.], [3.], [4.]])
>>>theta1 = np.array([[2.], [4.]])
>>>predict(theta1, X1)
array([[2], [6], [10], [14.], [18.]])
>>>X2 = np.array([[1], [2], [3], [5], [8]])
>>>theta2 = np.array([[2.]])
>>>predict(theta2, X2)
Incompatible dimension match between X and theta.
None
>>>X3 = np.array([[0.2, 2., 20.], [0.4, 4., 40.], [0.6, 6., 60.], [0.8, 8.,
80.]])
>>>theta3 = np.array([[0.05], [1.], [1.], [1.]])
>>>predict(theta3, X3)
array([[22.25], [44.45], [66.65], [88.85]])
```

Questions :

Be sure to understand the underlying concept and be able to answer to those questions during your evaluation:

- What is the hypothesis and what is its goal ?
- Considering a training set with 4242 examples and 3 features, how many components there is in the vector θ ?

- Considering the vector θ has the shape $(6,1)$ and the output has the shape $(7,1)$, what is the shape of the training set X ?

Exercise 01 - Cost function Concept

Turnin directory :	ex01
Files to turn in :	cost_function.py
Authorize module :	numpy,
Forbidden function :	None
Links :	https://www.coursera.org/learn/machine-learning/home/week/1
	https://www.coursera.org/learn/machine-learning/home/week/2
Hint :	Focus on the "Model and Cost Function" section of week 1
	and "Multivariate Linear Regression" section of week 2"

Objectives :

- Reinforce the mathematical skills tackled in **Mathematical Delights**, especially the **matrix-matrix operations**.
- Be able to explain what is the cost function in the machine learning context.
- Be able to implement a basic method based on the hypothesis function and output to obtain the value of the cost function.

Instructions :

In this exercise you will be interested with the concept of cost function usually noted J.

The cost function is defined by the following formula:

$$J(\theta) = \frac{1}{2M} \sum_{i=1}^M \left(\hat{y}^{(i)} - y^{(i)} \right)^2 = \frac{1}{2M} \sum_{i=1}^M \left(h_{\theta}(X^{(i)}) - y^{(i)} \right)^2$$

Where :

- θ is the coefficient vector,
- M is the number of training examples,
- $\hat{y}^{(i)}$ is the ith predicted output.
- $y^{(i)}$ is the ith real output.
- $X^{(i)}$ is the ith training examples (lign vector of N components),

Then you will code the functions named **cost_elem_** and **cost_** as per the instructions bellow:

```
def cost_elem_(theta, X, Y):
    """
    Description:
        Calculate all the elements  $0.5 * M * (y_{pred} - y)^2$  of the cost
    function.
    Args:
        theta: has to be a numpy.ndarray, a vector of dimension (number of
        features + 1, 1).
        X: has to be a numpy.ndarray, a matrix of dimension (number of
        training examples, number of features).
    Returns:
        J_elem: numpy.ndarray, a vector of dimension (number of the training
        examples, 1).
        None if X does not match the dimension of theta.
    Raises:
        This function should not raise any Exception.
    """
    ... your code here ...

def cost_(theta, X, Y):
    """
    Description:
        Calculate the value of cost function.
    Args:
        theta: has to be a numpy.ndarray, a vector of dimension (number of
        features + 1, 1).
        X: has to be a numpy.ndarray, a vector of dimension (number of
        training examples, number of features).
    Returns:
        J_value : has to be a float.
        None if X does not match the dimension of theta.
    Raises:
        This function should not raise any Exception.
    """
    ... your code here ...
```

Examples :


```

>>>import numpy as np
>>>X1 = np.array([[0.], [1.], [2.], [3.], [4.]])
>>>theta1 = np.array([[2.], [4.]])
>>>Y1 = np.array([[2.], [7.], [12.], [17.], [22.]])
>>>cost_elem(theta1, X1, Y1)
array([[0.], [0.1], [0.4], [0.9], [1.6]])
>>>cost_(theta1, X1, Y1)
3.0
>>>
>>>X2 = np.array([[0.2, 2., 20.], [0.4, 4., 40.], [0.6, 6., 60.], [0.8, 8.,
80.]])
>>>theta2 = np.array([[0.05], [1.], [1.], [1.]])
>>>Y2 = np.array([[19.], [42.], [67.], [93.]])
>>>cost_elem(theta2, X2, Y2)
array([[1.3203125], [0.7503125], [0.0153125], [2.1528125]])
>>>cost_(theta2, X2, Y2)
4.2387500000000004

```

Remarks :

You may notice that the cost function is very similar to the mse (see day00).

Questions :

Be sure to understand the underlying concept and be able to answer to those questions during your evaluation:

- What is the cost function and what is its goal ?
- What will be the interest of the cost function derivative (you may look a few more videos of the week 2 on coursera) ?
- **Bonus (hard) question** : Are there other forms of the cost function ? Cite at least 2 other definitions (with formula) of the cost function and give a very short description.

Exercise 02 - Descent Gradient

Turnin directory :	ex02
Files to turn in :	fit.py
Authorize module :	numpy
Forbidden function :	all functions that perform derivative at your place
Links :	https://www.coursera.org/learn/machine-learning/home/week/1
	https://www.coursera.org/learn/machine-learning/home/week/2
Hint :	Focus on the "Parameter Learning" section of week 1
	and "Multivariate Linear Regression" section of week 2"

Objectives :

- Reinforce the mathematical skills tackled in **Mathematical Delights**, especially the **matrix-matrix operations**.
- Be able to explain what is the fit in the machine learning context.
- Be able to implement a function which will perform a linear gradient descent (LGD).

Instructions :

In this exercise you will be interested with the concept of linear gradient descent to perform fit of dataset.

The linear gradient descent allows to correct the coefficients θ thanks to the gradient of the cost function.

You are expected to code a function named **fit_** as per the instructions bellow:

```
def fit_(theta, X, Y):
    """
    Description:
        Perform a fit of Y with respect to X.
    Args:
        theta: has to be a numpy.ndarray, a vector of dimension (number of
        features + 1, 1).
        X: has to be a numpy.ndarray, a matrix of dimension (number of
        training examples, number of features).
        Y: has to be a numpy.ndarray, a vector of dimension (number of
        training examples, 1).
    Returns:
        new_theta: numpy.ndarray, a vector of dimension (number of the
        training examples,1).
        None if there is a matching dimension problem.
    Raises:
        This function should not raise any Exception.
    """
    ... your code here ...
```

Hopefully, you have already code a function to calculate the linear gradient (I say that because you might have forgot).

Examples :

```
>>>import numpy as np
>>>X1 = np.array([[0.], [1.], [2.], [3.], [4.]])
>>>Y1 = np.array([[2.], [6.], [10.], [14.], [18.]])
>>>theta1 = np.array([[1.], [1.]])
>>>theta1 = fit_(theta1, X1, Y1, alpha = 0.01, n_cycle=2000)
>>>theta1
array([2.0023..., [6.002..., [10.0007..., [13.99988..., [17.9990...])
>>>
>>>X2 = np.array([[0.2, 2., 20.], [0.4, 4., 40.], [0.6, 6., 60.], [0.8, 8.,
80.]])
>>>Y2 = np.array([[19.6.], [-2.8], [-25.2], [-47.6]])
>>>theta2 = np.array([[4.], [3.], [2.], [1.]])
>>>theta2 = fit_(theta2, X2, Y2, alpha = 0.005, n_cycle=2000)
>>>theta2
array([[19.5938..., [-2.8030..., [-25.1999..., [-477.5969...]])
```

Remarks :

You can generate other examples by choosing arbitrary X array and declare an Y an linear expression of the X column. Notice also that you can find [nan] the components of theta. In that case you probably used a too big learning rate.

Questions :

Be sure to understand the underlying concept and be able to answer to those questions during your evaluation:

- Can you explain the different step in the fit method (hint: you have to talk about J , its gradient and the θ)?
- What happens if you choose a too big learning rate ?
- What can you say if you choose a very small learning rate and a reasonable number of cycle ?

Exercise 03 - The Linear Regression with Class

Turnin directory :	ex03
Files to turn in :	mylinearregression.py
Authorize module :	numpy
Forbidden module :	sklearn

Objectives :

- Code your class containing all the methods to perform linear regression.

Instructions :

In this exercise you will not learn something new but do not worried it is ffor your own well-being (at least I think it is...).

You are expected to code your own class **MyLinearRegression** which looks similar to the class **sklearn.linear_model.LinearRegression**:

```
class MyLinearRegression():
    """
    Description:
        My personnal linear regression class to fit like a boss.
    """
    def __init__(self, theta):
        """
        Description:
            generator of the class, initialize self.
        Args:
            theta: has to be a list or a numpy array, it is a vector of
            dimension (number of features + 1, 1).
        Raises:
            This method should noot raise any Exception.
        """
        ... your code here ...
        ... other methods ...
```

You will add the methods **predict_(self, X)**, **cost_elem_(self, X, Y)**, **cost_(self, X, Y)** and **fit_(self, X, Y)**.

You have already coded the functions, it will just need little modifications to put them in **MyLinearRegression**.

Examples :

```
>>>import numpy as np
>>>from mylinearregression import MyLinearRegression as MyLR
>>>X = np.array([[1., 1., 2., 3.], [5., 8., 13., 21.], [34., 55., 89.,
144.]])
>>>Y = np.array([[23.], [48.], [218.]])
>>>mylr = MyLR([[1.], [1.], [1.], [1.]])
>>>mylr.predict_(X)
array([[8.], [48.], [323.]])
>>>mylr.cost_elem_(X,Y)
array([[37.5], [0.], [1837.5]])
>>>mylr.cost_(X,Y)
1875.0
>>> mylr.fit_(X, Y, alpha = 1.6e-4, n_cycle=200000)
>>>mylr.theta
array([[18.023..], [3.323..], [-0.711..], [1.605..], [-0.1113..]])
>>>mylr.predict_(X)
array([[23.499..], [47.385..], [218.079...]])
>>>mylr.cost_elem_(X,Y)
array([[0.041..], [0.062..], [0.001..]])
>>>mylr.cost_(X,Y)
0.1056..
```

Exercise 04 - Linear Regression

Turnin directory :	ex04
Files to turn in :	linear_model.py
Authorize module :	numpy, matplotlib
Forbidden module :	sklearn
Remarks :	Read the doc

Objectives:

- Reinforce the mathematical skills tackled in **Mathematical Delights**, especially the **matrix-matrix operations**.
- Be able to perform a fit with a very small data set, knowing the hypothesis function h .
- Manipulate the cost function J , plot it and briefly analyze the plot.

Instructions:

You can find in the ressources a tiny data set called "**are_blue_pills_magics.csv**" which give you mathematical performance of patients in function of the quantity of the "blue pills" they took before the test. You have a description of the data in the file named "**are_blue_pills_magics.txt**".

As hypothesis function h , you will choose:

$$h(x) = \theta_0 + \theta_1 x$$

Where x is the variable, θ_0 and θ_1 are the coefficients of the hypothesis.

The hypothesis is a function of x . It gives you the predicted values usually note $y^{(i)}$ or

y_{pred} .

You will model the data and plot 2 differents graphs:

- The graph with the data and the best hypothesis you find for the spacecraft piloting score versus the quantity of "blue pills" (see example figure 1),

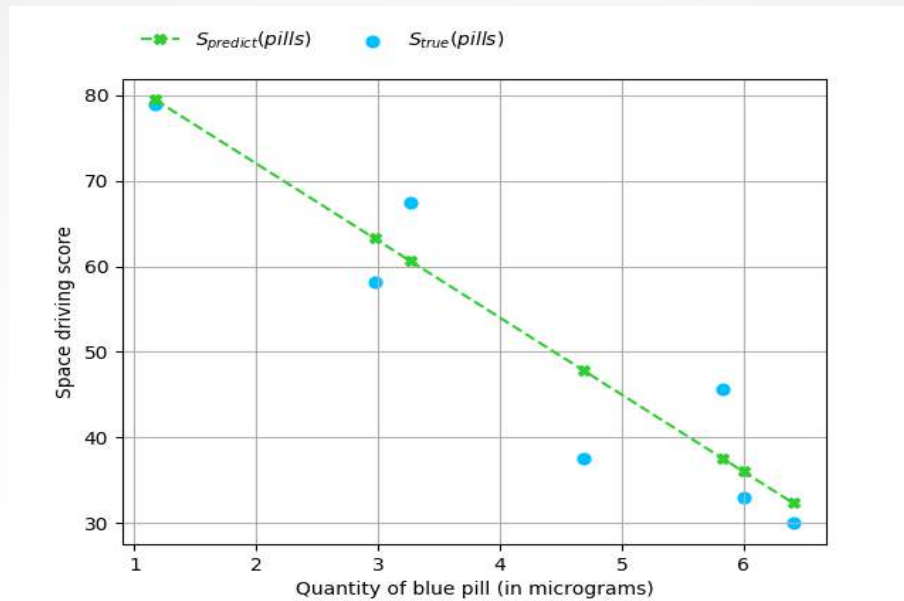


Figure 1: Evolution of the space driving score in function of the quantity of blue pill (in micrograms).

- The cost function $J(\theta)$ in function of the θ values (see example in figure 2),

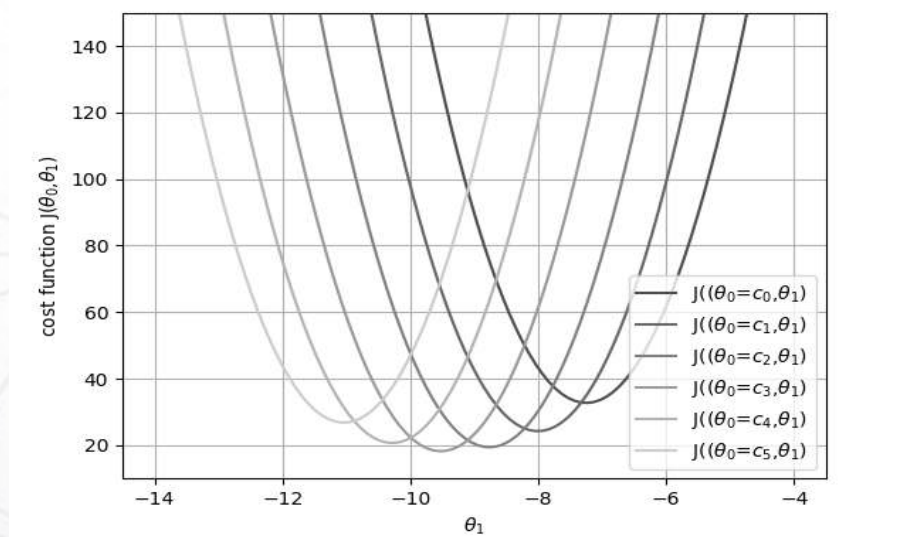


Figure 2: Evolution of the cost function J in function of θ_1 for different values of θ_0 .

- You will calculate the MSE of the hypothesis you chose (you know how to do it already) add it to your class **MyLinearRegression**.

Examples

```

>>>import pandas as pd
>>>import numpy as np
>>>from sklearn.metrics import mean_squared_error
>>>from mylinearregression import MyLinearRegression as MyLR
>>>
>>>data = pd.read_csv("are_blue_pills_magics.csv")
>>>Xpill = np.array(data[Micrograms]).reshape(-1,1)
>>>Yscore = np.array(data[Score]).reshape(-1,1)
>>>
>>>linear_model1 = MyLR([[89.0], [-8]])
>>>linear_model2 = MyLR([[89.0], [-6]])
>>>Y_model1 = linear_model1.predict_(Xpill)
>>>Y_model2 = linear_model2.predict_(Xpill)
>>>
>>>print(linear_model1.mse_(Xpill, Yscore))
# 57.60304285714282
>>>print(mean_squared_error(Yscore, Y_model1))
# 57.603042857142825
>>>print(linear_model2.mse_(Xpill, Yscore))
# 232.16344285714285
>>>print(mean_squared_error(Yscore, Y_model1))
# 232.16344285714285

```

Clarifications and hints

There is no method named **.mse_** in the class LinearRegression of the module

sklearn.linear_model but there is a method named **.score**. The **.score** method correspond to the R^2 score.

The metric MSE is available in the module sklearn.metrics.

Be sure to understand the underlying concept and be able to answer to those questions evaluators may ask:

- What is the hypothesis and what is its goal ? (It is a second chance to let you say something intelligible, no need to thank me)
- What is the cost function and what it is representing ?
- What is the linear gradient descent and what is it doing ?
- Can you explain the MSE and what it is spotting ?

Exercise 05 - Multiples features and Linear Gradient Descent

Turnin directory :	ex05
Files to turn in :	multi_linear_model.py
Authorize module :	numpy, matplotlib
Forbidden module :	sklearn
Forbidden function :	LinearRegression
Links :	https://www.coursera.org/learn/machine-learning/home/week/2
Hint :	Really, spend time on "Multivariate Linear Regression" section until it is clear

Objectives:

- Reinforce the mathematical skills tackled in **Mathematical Delights**, especially the vectorized form of **linear cost function** and **gradient descent**.
- Be able to manipulate the **linear cost function** and the **linear gradient descent** for a multiple features problem (and knowing what you are doing of course).
- Be able to visualize the different objects via graphics and extract basics informations based on its.

Instructions:

As you are able to perform a simple linear regression with one feature (well done !) it is time to dream bigger.

Lucky you are, we give you a new dataset with multiple features that you will find in the ressources attached.

The dataset is called "**spacecraft_data.csv**" which contains the prices of spacecrafts in function of multiples features (multiple features means you will need a multi-linear model but hold on for the moment). A description of the dataset is provided in the file "**spacecraft_data_description.txt**".

Part One: single linear regression

As a starter, you will try to fit the data with a single linear regression and see what we get. Thus, your hypothesis $h(X)$ would be given by:

$$h(X) = X \cdot \theta = \begin{bmatrix} 1 & x_1^{(1)} \\ \vdots & \vdots \\ 1 & x_1^{(M)} \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \theta_0 + \theta_1 x_1^{(1)} \\ \vdots \\ \theta_0 + \theta_1 x_1^{(M)} \end{bmatrix}$$

assuming X is an array of dimension $M \times (N + 1)$ (M corresponding to the number of training examples and N to the number of features). Here in this part 1, N is equal to 1.

As you already noticed, an extra column of 1 is add at the beginning of the X matrix such as

the matrix product gives a vector of dimension $M \times 1$ where the ith component is

$\theta_0 + \theta_1 x_1^{(i)}$. This is just a trick to perform the calculation of $h(X)$ in the vectorized way.

You are expected to :

- The graph with the data, the hypothesis $h_{\theta_0, \theta_{age}}^{LGD}(age)$ obtained via linear gradient descent versus age (see example figure 1),

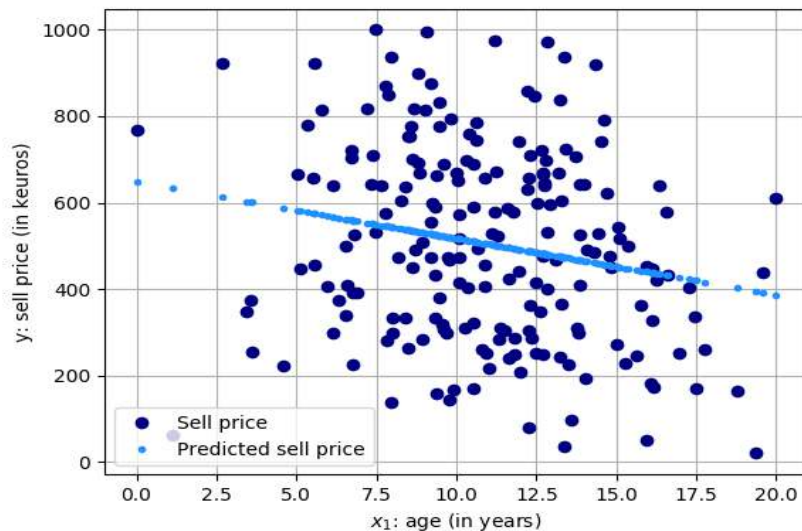


Figure 1: Evolution of the sell price of spacecrafts with respect to the age of the spacecraft and representation of the predicted values of our first model.

- The graph with the data, the hypothesis $h_{\theta_0, \theta_{thrust}}^{LGD}(thrust)$ obtained via linear gradient descent versus thrust (see example figure 2),



Figure 2: Evolution of the sell price of spacecrafts with respect to the thrust power of the spacecraft engines and representation of the predicted values of our second model.

- The graph with the data, the hypothesis $h_{\theta_0, \theta_{Tmeters}}^{LGD}(Tmeters)$ obtained via linear gradient descent versus Tmeters (see example figure 3),

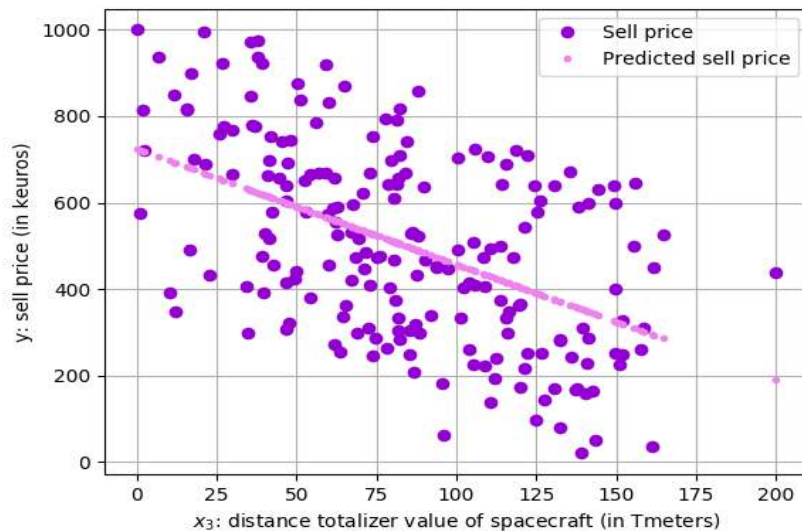


Figure 3: Evolution of the sell price of spacecrafts with respect to the terameters driven and the predicted values of our third model.

Reminder:

- You may obtain after the fit $\theta=[\text{nan}, \text{nan}]$. It may come from a too large learning rate.
- Also, be aware that you set the number of cycle of the fitting process, but it does not guaranty you the fit is the best or good. For this purpose, a tolerance parameter might be used (not see here, so do not worried).

Here, you can increase the number of cycle and play around with the learning rate to try to get a good fit.

Hint:

- Plot the data in function of the feature you are working with, allows you to guess initial values of theta that are not too bad, which will help to converge to a good fit.

Examples:

```
>>>import pandas as pd
>>>import numpy as np
>>>from mylinearregression import MyLinearRegression as MyLR
>>>
>>>data = pd.read_csv("spacecraft_data.csv")
>>>[...]
>>>myRL_age = MyLR([[1000.0], [-1.0]])
>>>myLRage.fit(X[:,0].reshape(-1,1), Y, alpha = 2.5e-5, n_cycle = 100000)
>>>
>>>RMSE_age = myLRage.mse_(X[:,0].reshape(-1,1),Y)
>>> print (RMSE_age)
57636.77729...
```

Are the fits with a single variable precised ? Why ? (What did I say at the beginning ?)

Part Two: Multilinear Regression (A New Hope)

Now, it is time for your first multilinear regression !

As you might expected, the formula of the hypothesis change a little and is given by:⁽¹⁾

$$h(X) = X\theta = \begin{bmatrix} x_0^{(1)} & \cdots & x_N^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(M)} & \cdots & x_N^{(M)} \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_N \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N \theta_i x_i^{(1)} \\ \vdots \\ \sum_{i=1}^N \theta_i x_i^{(M)} \end{bmatrix}$$

where X is the training dataset matrix, θ the coefficients vector, M is the number of training samples and N the number of features.

But It should not change the methods you coded, hopefully !

Examples

```
>>>import pandas as pd
>>>import numpy as np
>>>from mylinearregression import MyLinearRegression as MyLR
>>>
>>>data = pd.read_csv("spacecraft_data.csv")
>>>X = np.array(data[['Age', 'Thrust_power', 'Terameters']])
>>>Y = np.array(data[['Sell_price']])
>>>my_lreg = MyLR([1.0, 1.0, 1.0, 1.0])
>>>my_lreg.mse_(X,Y)
144044.877...
>>>my_lreg.fit_(X,Y, alpha = 1e-4, n_cycle = 600000)
>>>my_lreg.theta
array([[334.994...], [-22.535...], [5.857...], [-2.586...]])
>>>my_lreg.mse_(X,Y)
586.896999...
```

Remarks:

You can obtain a better fit, if you increase the number of cycles.

You are expected to:

- Plot the output and predicted output on the same graph in function of the age(see figure below),

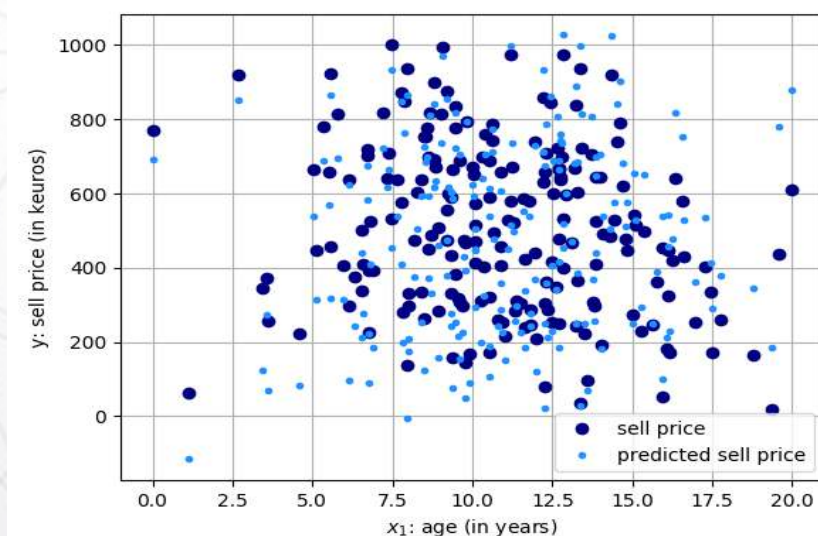


Figure 1: Evolution of the sell price of spacecrafts and evolution of predicted sell price of spacecrafts with the multi-variables hypothesis, with respect to the age.

- Plot the output and predicted output on the same graph in function of the thrust power (see figure below),

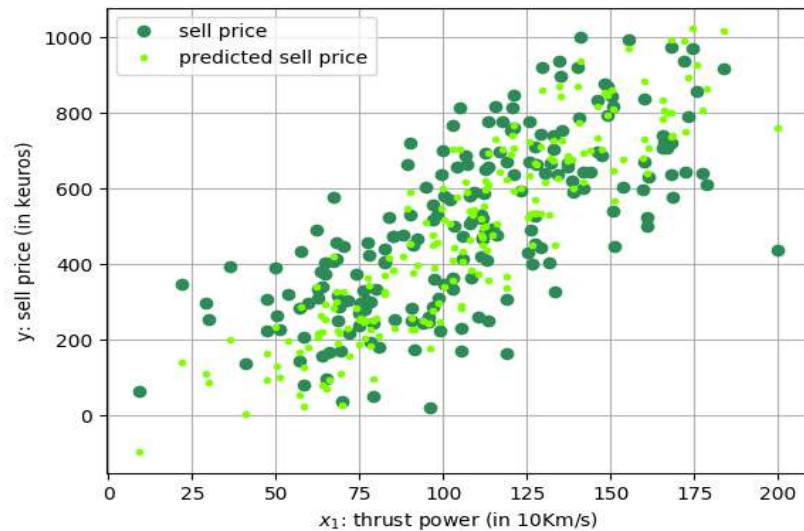


Figure 2: Evolution of the sell price of spacecrafts and the evolution of predicted sell price of spacecrafts with the multi-variables hypothesis, with respect to the thrust power of the engines.

- Plot the output and predicted output on the same graph in function of the distance (see figure below),

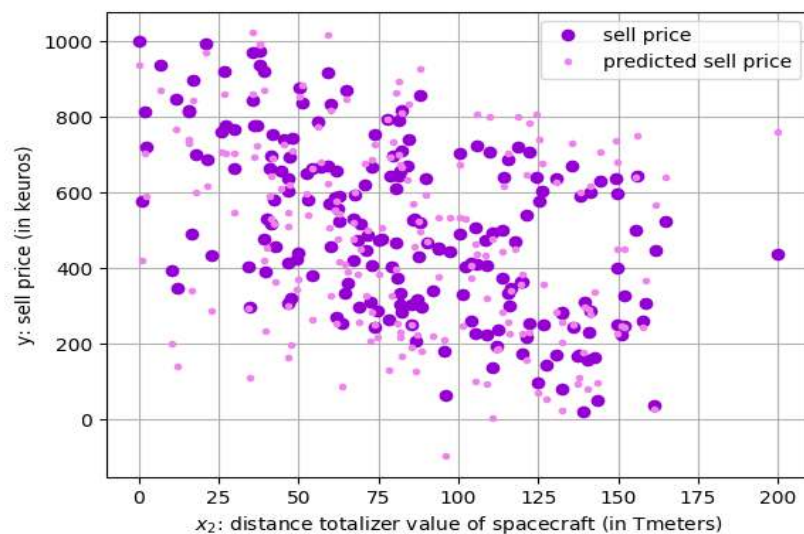


Figure 3: Evolution of the sell price of spacecrafts and evolution of the predicted sell price of spacecrafts with multi-variables hypothesis, with respect to the terameters driven.

What conclusion can you do in regards of the results of both parts ?

Questions:

Be sure to understand the underlying concept and be able to answer to those questions evaluators may ask:

- What can you say on the influence of the choice of the hypothesis ?

Exercise 06 - Multiples features and Normal Equation

Turnin directory :	ex06
Files to turn in :	normal_equation_model.py
Authorize module :	numpy, matplotlib
Forbidden module :	sklearn
Forbidden function :	LinearRegression
Links	https://www.coursera.org/learn/machine-learning/home/week/2
Remarks :	Take a look to the "Computing Parameters Analytically" section

Objectives:

- Reinforce the mathematical skills tackled in **Mathematical Delights**, especially the vectorized form of **matrix-matrix operations** (intermediate manipulation).
- Be able to perform **linear cost function** and what one names **normal equation** for a multiple features problem (and still knowing what you are doing of course obviously).
- Be able to implement a vectorized method **normalequation** in order to perform a full linear regression with the normal equation method.

Instructions:

We continue to play with the same dataset (**spacraft_data.csv**) given in the previous exercise.

You will code a new method named **normalequation** which will allows you to perform a fit of the dataset based on the method called **normalequation**.

The normal equation is:

$$\theta_{NE} = (X^T X)^{-1} \cdot X^T Y$$

where X is the training dataset matrix, θ_{NE} the parameter's hypothesis vector obtained with the normal equation, Y are the output vector.

For folks who do not understand what the meaning of the superscripts "T" and "-1" over a matrix, they respectively tell you we take the transpose and the inverse of the concerned matrix. Do not worry, for those who do not know what it means or do not remember how to do, well you can still do the exercise because numpy will run the maths for you, but you will not understand what happens behind the scene for the moment. Fortunately, the mathematic workshops will be coming soon, what a wonderful opportunity to learn to do transposition and inversion of matrices. Lucky you !

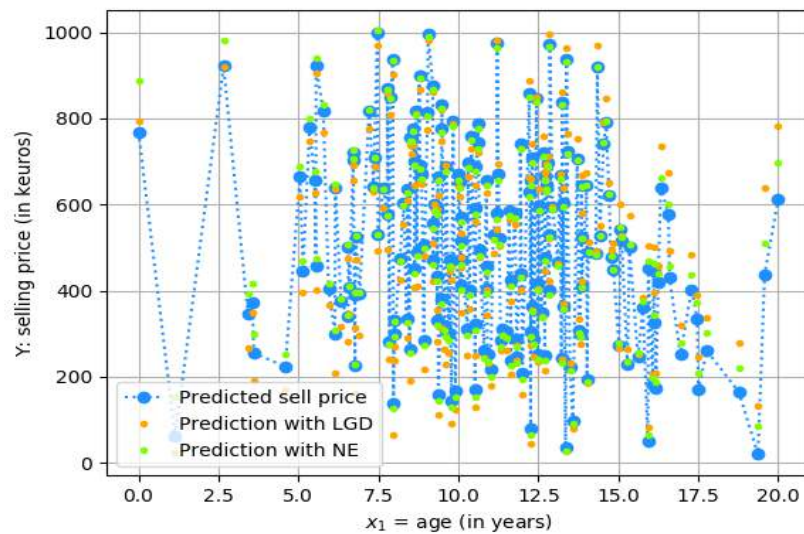
You are expected to:

- Add the method **normalequation** in the class **MyLinearRegression** as per the instructions given below:

```
def normalequation_(self, X, Y)
    """
    Description:
        Perform the normal equation to get the theta parameters of the
        hypothesis h and stock them in self.theta.
    Args:
        X: has to be a numpy.ndarray, a matrix of dimension (number of
        training examples, number of features)
        Y: has to be a numpy.ndarray, a vector of dimension (number of
        training examples,1)
    Returns:
        No return expected.
    Raises:
        This method should not raise any Exceptions.
    """
    ... your code ...
```

You will model the data and plot different graphs (yes again):

- The graph with the data, the hypothesis $h_{\theta}^{LGD}(X)$ obtained via linear gradient descent and $h_{\theta}^{NE}(X)$ (see example figure below) with respect to the feature of your choice,



- Calculate the MSE of the data and the predicted data generate with the model train with the linear gradient descent, then with normal equation technique.

Example

```
>>>import pandas as pd
>>>import numpy as np
>>>from mylinearregression import MyLinearRegression as MyLR
>>>data = pd.read_csv("spacecraft_data.csv")
>>>[...]
>>>myLR_ne = MyLR([1., 1., 1., 1.])
>>>myLR_lgd = MyLR([1., 1., 1., 1.])
>>>myLR_lgd.fit_(X,Y, alpha = 5e-5, n_cycle = 10000)
>>>myLR_ne.normalequation_(X,Y)
>>>myLR_lgd.mse_(X,Y)
2265.1048...
>>>myLR_ne.mse_(X,Y)
413.05299...
```

Be sure to understand the underlying concept and be able to answer to those questions evaluators may ask:

- Between the LGD and the NE fits which one is the best model ? (quantitative arguments are welcome)
- What are the advantages and drawbacks of the linear gradient descent and normal equation ?
- In which case, the method normal equation cannot be used ?

Bonus Exercise - Learning Rate and Quadratic Hypothesis

Turnin directory :	ex07
Files to turn in :	so_much_hyp.py
	alpha.py
Authorize module :	numpy, matplotlib
Forbidden module :	sklearn
Forbidden function :	LinearRegression
Links :	Did I already give you a link to coursera Machine Learning MOOC ?
Remarks :	You know what to look at, I think...

Objectives:

- Grasp other metrics such as RMSE and R2score, and understand their limits.
- Discover the non-linear dependency to elaborate more complex hypothesis and reinforce mathematical skills at the same time.
- Gain a deeper understanding of the learning rate.

Instructions:

___ With great power comes great responsibility ___ (Uncle Ben).

Uncle Ben was right, a large power thrust comes a risk of spacecraft crash.

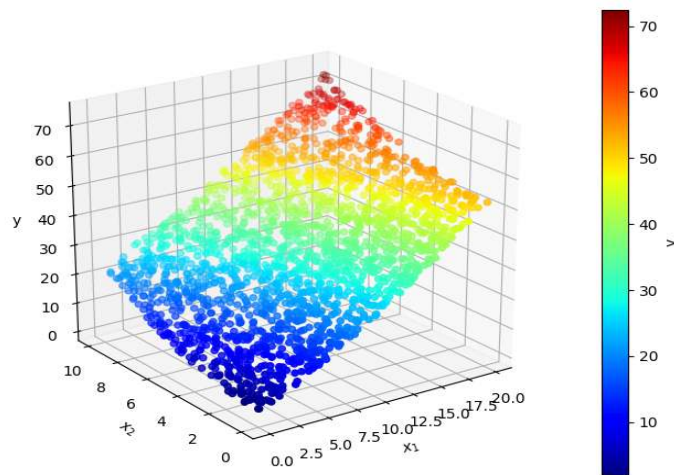
With the apparition of individual spacecraft, the number of wild rides explodes, especially in the asteroid Saturn Belt.

Experts are concerned with the impact of these rides and worried about the consequence on the number of asteroids orbiting and the number of asteroids leaving the orbit of Saturn !

For this last exercise you have to find a model to describe the number of asteroids leaving the orbit of Saturn due to wild rides.

The dataset related to this important problem is named "**saturn_asteroids.csv**". The data description is in the file "**saturn_asteroids_description.txt**".

Also, you have a really nice graphical representations just below.



You will use the methods you code during the day to fit the data.
The hypothesis function you will need is a more complex one. This time you have to find out the best model to fit the data.
Do not worry, it is not too complicated.

You have to justify your hypothesis (know what and why you are doing).

Part Zero - Nothing Else (but) Metrics

This part will focus on the metrics.

You already know about one of them : MSE.

There are several more which are quite usual : RMSE, MAE and R2score.

In this part you will code 2 more methods you will add to **MyLinearRegression** class : RMSE and R2score, as per the instructions below:

```

def rmse_(self, X, Y):
    """
    Description:
        Calculate the RMSE between the predicted output and the real output.
    Args:
        X: has to be a numpy.ndarray, a matrix of dimension (number of
training examples, number of features).
        Y: has to be a numpy.ndarray, a vector of dimension (number of
training examples, 1).
    Returns:
        rmse: has to be a float.
        None if there is a matching dimension problem.
    Raises:
        This function should not raise any Exception.
    """
    ... your code here ...

def r2score_(self, X, Y):
    """
    Description:
        Calculate the R2score between the predicted output and the output.
    Args:
        X: has to be a numpy.ndarray, a matrix of dimension (number of
training examples, number of features).
        Y: has to be a numpy.ndarray, a vector of dimension (number of
training examples, 1).
    Returns:
        r2score: has to be a float.
        None if there is a matching dimension problem.
    Raises:
        This function should not raise any Exception.
    """
    ... your code here ...

```

Remarks:

You might consider to code 2 more methods **rmse_elem_()** and **r2score_()** similar to what you did for the cost function in the exercise 01.

Examples:

to complete

Part One - Find the best hypothesis

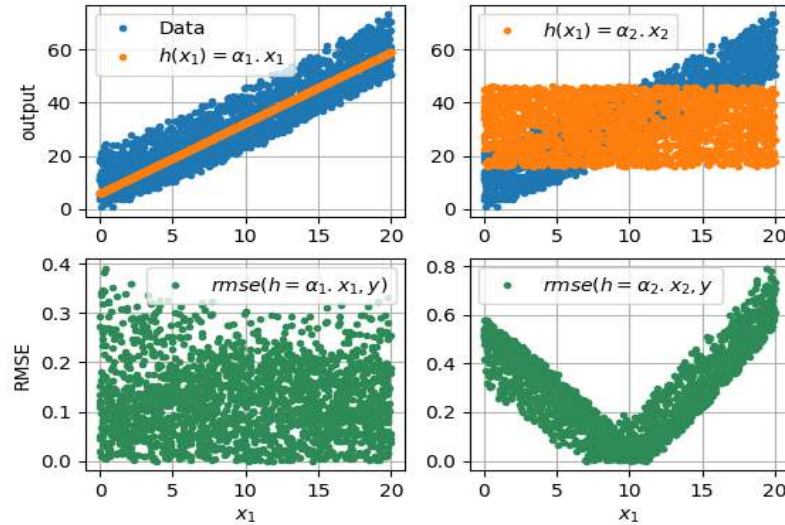
For this part, you will write your code in the file **"so_much_hyp.py"**.

You can code new methods to help you to justify the choice of the hypothesis but it is not compulsory.

The major work for this exercise concerns the data.

You are expected to:

- Find a good hypothesis and justify that your model is a good one. So you have to imagine different hypothesis and try them (see figure below).



Hints:

Take a look to the metrics (the RMSE for example) with respect to the features.

Notice that the graphs $\sqrt{1/M(\widehat{y^{(i)}} - y^{(i)})^2}$ is not the rmse that we plot precisely, but

the term in the sum (**rmse_elem_ ...**).

Here, what you can get:

```

>>>import pandas as pd
>>>import numpy as np
from mylinearregression import MyLinearRegression as MyLR
>>>data = pd.read_csv("saturn_asteroids.csv")
>>>X = np.array(data[['Mean_speed', 'N_spacecraft']])
>>>Y = np.array(data[['ALSB']])
>>>hypo1 = MyLR([1., 1.])
>>>hypo2 = MyLR([1., 1.])
>>>hypo1.fit_(X[:,0], Y, alpha = 1e-4, n_cycle = 1e5)
>>>hypo2.fit_(X[:,1], Y, alpha = 1e-4, n_cycle = 1e5)
>>>hypo1.rmse_(X[:,0],Y)
6.439...
>>>hypo2.rmse_(X[:,0],Y)
15.089...

```

Best fit to bet: RMSE = 0.05977051791842336, good luck. Of course, if you are using normal equation, You can not consider you compete fairly (boo !!).

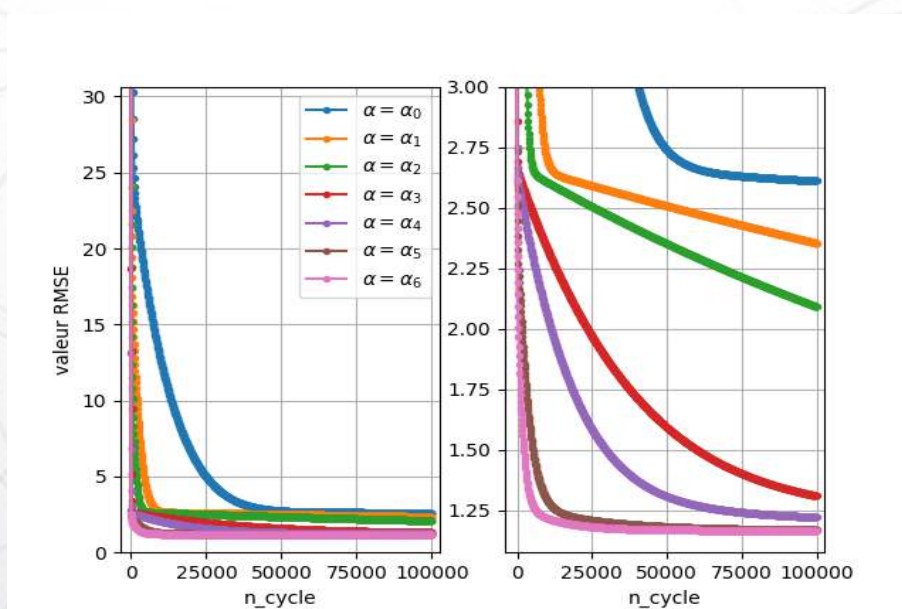
Part Two - The learning rate α

For this part you will write your code in the file **alpha.py**.

We will focus on the concept of learning rate, in consequence, you will have to fit with the method **MyLinearRegression.fit** you coded previously.

You are expected to:

- Find a good learning rate α to fit the data with the hypothesis you have chosen in the previous part, your choice will be motivated by the convergence reached in the minimum cycle (see figure below).



Be sure to understand the underlying concept and be able to answer to those questions evaluators may ask:

- What is the difference between MSE and R2score ?
- What happens if you choose a learning rate too big ?
- How the metric and the θ 's evolve with a slightly too big learning rate ?
- How the convergence evolves in function of the cycle ? Do you have any idea to speed up the convergence as the number of cycle increase ? (It is an open question, be creative and smart).