TUTORIAL : Introduction to Regression

These are the computational exercises in learning how to program using **Python**. Don't be afraid of making mistakes as programmers learned from mistakes they made in programming.

Numerical advice:

It is good practice to plot the function that you are going to solve for example the integrand function to check whether it is "well behaved" before you attempt to integrate. Singularities, discontinuities or other irregular behaviour are difficult to handle numerically and may need special methods.

1. You are given a data on the conductivity of mercury as a function of temperature:

Temperature (K)	Conductivity (W/(cmK))
300	0.084
400	0.098
500	0.109
600	0.120
700	0.127

and a code, **leastsquare.py** that implement the least squares fitting and quadratic fitting to the data. The code reads a CSV file with **TWO** text headers and having the **comma delimiter**, **data2.csv** in the Github. A prediction for the conductivity is calculated at a selected temperature and a png graph is also produced. Run the code to understand the python code.

Supervised Learning-Regression

In this note, **supervised learning** is described for **linear and polynomial regressions**. First, we are doing to learn to implement the linear regression algorithm in machine learning, and its step-by-step implementation in Python for **supervised learning-regression analysis.** We are going to write a python program that uses **linear regression** in one dimension as the machine learning algorithm for **supervised** learning-regression analysis.

In **regression** problems, we have a **labelled training dataset** of input variables, x and a **numerical** output variable, y. The idea is when we apply **simple linear regression** to find the best fit **linear** relationship between x and y denoted it as the linear function $f(x)$. In any new observations or events, the input variables, χ and the linear function f are used to predict its output value:

$$y = f(\chi).$$

For the case with more than one input variable, it is **multiple linear regression**.

A polynomial of the first order (linear), $n = 1$:

$$y = a_0 x + a_1.$$

Now it is clear that the linear regression is an algorithm that finds the best values of a_0 and a_1 to fit the training dataset. Namely, if x and y represent the observations from the training dataset, then we want to find the line that gives the best fitted linear relationship based on this training dataset.

Remember that the format of the polynomial follows the arrangement of coefficients in python.

*As a final note, since linear regression is well studied topic in statistic, input variables, x and output variable, y are also called with various terms. For example input variables, x are also called **features**, **explanatory variables** and **independent variables** while the output variable, y are also be called as **target**, **response** and **dependent variable**.*

Below is the step-by-step implementation in a python code of linear regression algorithm in machine learning. The steps are also applicable to other machine learning algorithms.

Linear Regression Algorithm

Step 1: Import Python packages

In addition to the standard python packages; Numpy, Pandas and Matplotlib, we will use the **Scikit Learn** (**sklearn**) package for the tools necessary for predictive data analysis, including linear regression. **Scikit Learn** is a popular package for machine learning.

Step 2: Generate random training dataset

We will generate a random sample dataset as the training set for the linear regression algorithm. First generate a random dataset of certain size x as the single input variable. Then we set y as the output as a linear relationship with x. The random variable ***noise*** is added to create noise to the dataset. Now we have a simple linear regression model.

Step 3: Fit the linear regression model

We use the linear regression algorithm within the **Scikit Learn** package to create a model. After fitting the model, we will obtain all the coefficients.

Step 4: Predictions with the linear regression

We can make predict using our new model. This is the most exciting and powerful aspects of this model. We have done this also in Question 5.

The **polynomial regression** is a special case of **multiple linear regression** where the model is described by

$$y = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-2}x^2 + a_{n-1}x + a_n$$

In the case of a polynomial of the second order (quadratic), n=2:

$$y = a_0x^2 + a_1x + a_2.$$

and a polynomial of the first order (linear), n=1:

$$y = a_0x + a_1.$$

The steps are the same as in the linear case.

2. (a) You are given a code, **linear_regression.py** for the linear regression machine learning algorithm. It generates a random dataset of size 100 with x as the single input variable to the following linear equation with added noise:

$$y = 2x + 5 + noise$$

where the random variable ***noise*** is added to create noise to the dataset by using the random number generator, **random.uniform** from numpy.

(b) The code uses the linear regression algorithm, **LinearRegression** and **fit** within the scikit learn package to create and fit the dataset to a linear regression model.

(c) After fitting the model in Step (b), the code prints out the coefficients of the linear equation. Then **coef_** and **intercept_** are used to obtain the gradient and intercept. The random dataset in (a) i.e. (x,y) is plotted using scatterplot and includes the linear equation as well. The plot is saved as **linear_regression.png**.

(d) Prediction to some values of x can be made using the code.

3. Question 3 repeats Question 2 but now the code, **poly_regression.py** is for the polynomial regression with equation given by:

$$y = x^2 + 3x + 7 + noise$$

Notice that the linear regression gives poor fit to the dataset.