Instructions:

1. `Published Date :  14-Oct-2021`
2. `Submission Date :  24-Oct-2021`

DA 202: Introduction to Data Science 3:1

Instructions

1. Codes should be original and should not be copied from others, including fellow participants.
2. Plagiarised reports/codes results in zero mark

Prob: 1 Regression - [20 Points]

A CSV file (transistor.csv) with the details of Transistor count and the Transistor size across different years is provided. Do the following process to fit a linear regression model to the given data

- Use the Gradient Descent method (self-coded) to compute the parameters of the linear regression line.
- The given data $(y1, y2)$ follows exponential distribution (Moore's Law). Therefore, identify a way such that this problem could be fitted using a first order linear regression model.
- Obtain the parameters for "transistor_count($y1$)" and "transistor_size($y2$)" independent to each other (i.e, use x, $y1$ to obtain the parameters $(b0, b1)$ for "transistor_count($y1$)" and use x, $y2$ to obtain the parameters for "transistor_size($y2$) separately".

Points to be Noted:

- Please do not use any libraries such as Scikit-learn for this problem. Use NumPy arrays and implement your algorithm to code gradient descent and other methods.
- Your code should contain a method "predict", that accepts the year as a parameter and returns both "transistor_size" and "transistor_count" as outputs.
- Optimise your code locally to tune all the hyper parameters related to this problem and use only the final code (with optimized values that you have identified) in the final submission of the code on moodle.
- The code will be evaluated based on how close your prediction is to the test data set (which will be shared after evaluation).

Submissions

Part A - Code submission on Moodle

1. Submit your code which fits the given data to linear regression and a method which predicts the actual values $(y1, y2)$ for the given year.

Part B - Report Submission

1. Plot the actual data for both cases, ($xvs.y1$, $xvs.y2$)
2. Plot the log value of actual data for both cases ($xvs.y1$, $xvs.y2$)
3. Provide a brief description of the choice of parameters used for the computation (Learning Rates, selection of validation dataset).
4. Provides a brief description of data over-fitting and under-fitting of data. Which of the above categories your current model fits?
5. Document your training error and validation error for both $y1$ and $y2$ separately.
6. Plot your actual data along with the linear regression line for both $y1$, $y2$ separately.

Prob: 2 Finite Difference Methods - [20 Points]

Consider a scenario in your favourite game Angry bird, where the bird hits the target at an altitude of $50m$ above the ground at a time of '8' seconds from the ground after it was launched. In order to break the obstacle located at that point, the bird should hit the target location with a minimum $y - velocity$ magnitude of ($\frac{dy}{dt}$) of 35 m/s .

Consider only the motion along $y - axis$ (altitude) for this problem. The motion of the bird under the influence of gravity (ignoring air drag) is given by

$$\frac{d^2y}{dt^2} = -g$$

Here, g is acceleration due to gravity, which is given by -9.81 ms^{-2}

- Write a python code which solves for y (vertical displacement) by solving the system of algebraic equations that arises by discretising the above equation using 3-point central difference formula for second derivative.
- once the data is obtained for y (vertical displacement), use first order backward difference formula to compute the velocity at all points (except the first point).
- Does the bird has sufficient velocity in the $y-$direction to break the obstacle?
- Consider one other scenario, where the bird is flung in such a way that it hits the target at $t = 9\ s$. In this scenario, does the bird has sufficient velocity in the $y-$ direction to break the obstacle ?

Submissions

Part A - Code submission on Moodle

- The code should contain part which computes the y (vertical displacement) for 100 time steps between $t = 0$ and $t = 8$ or $t = 9$.
- The code should contain logic which computes the velocity ($\frac{dy}{dt}$) from the obtained from the displacement value (y)

Part B - Report Submission

- Provide the plots for $y-$displacement vs time for both scenarios in a single graph.
- Provide the plots for velocity vs time for both scenarios in a single plot.
- Derive the matrix coefficients for the second derivative formulation for the above problem.

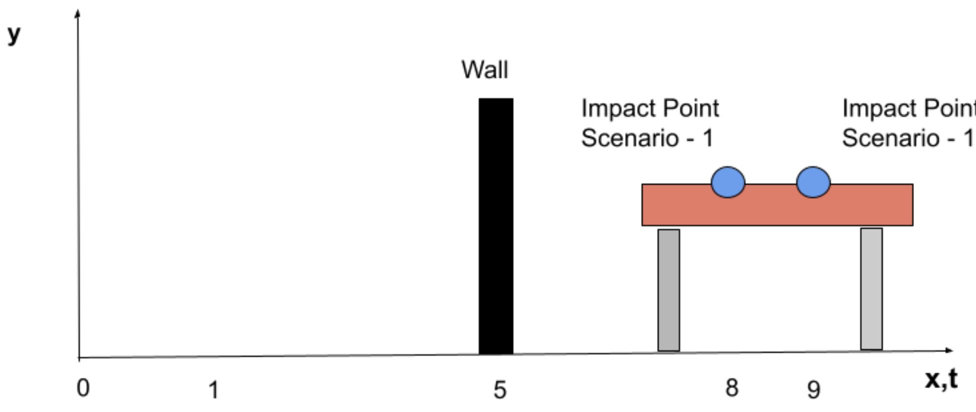

Figure 1: Problem - 2

Prob: 3 Finite Difference Method - Formula - [10 Points]

Derive the five point finite difference formula to find the second derivative at a point (in 1D). Comment on truncation error and accuracy (in terms of the step size h).

Appendix

Gradient Descent

The objective of least squares is to minimise

$$Error(g) = \sum_{i=0}^{n} |y_{actual} - y_{predicted}|^2 \tag{1}$$

This will be referred as the objective function. For a first order least square, the $y_{predicted}$ is given by $b_1 x + b0$ where b_1 and b_0 are the (unknowns) co-coefficients.

The gradient decent algorithm for minimising an objective function can be written as,

```
while ( residual < tolerance ):
        gb0 = compute sum of gradient of objective function with respect to b0
        gb1 = compute sum of gradient of objective function with respect to b1

        ## Update the guess of b0 and b1 based on the gradients obtained
        b0 = b0 - learning_rate*gb0
        b1 = b1 - learning_rate*gb1

        ## compute the error ( objective function ) with the newly computed b0 and b1
        Error =  sum of (y_actual-y_predicted) for all points
        residual = abs(Error - oldError);
        old_error = Error;
```

To obtain the expression for gb0 and gb1 in the above formula, substitute $y_{predicted} = b_1x+b0$ in equation (1), and then use chain rule to obtain the values of $\frac{dg}{db_0}$(gb0) and $\frac{dg}{db_1}$ (gb1).

Note:

- The tolerance value can be any minimal value, that is, to check whether the residual has reached a minimal value (which does not change on subsequent iterations) use any value between $1e-10$ and $1e-5$ in your computations.
- please make a restriction on the maximum number of iterations so that the computation does not run an infinite loop.
- The learning rate is a parameter that determines how much you have to move along the direction of the gradient identified, usually a value below "1".
- The learning rate is one of the parameter that you have to fix based on the trail and error to obtain proper convergence. Fix these parameters locally in your systems and use the final value in your submission.