

AISC 1004- Deterministic Models and Optimization

Case study 1

**Submitted by “Group C”**

By submitting this assignment, you confirm that you alone have contributed to this submission. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “0" in the course, or possibly more severe penalties as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:<https://www.loyalistcollege.com/about-loyalist/policies/aop-216-academic-honesty/>

7.  Calculate the cost function: y’=3x+2, (x,y)={(1,6),(2,10),(3,10),(5,15),(7,20)}

A cost function is a measure of how much error the model is in terms of its ability to estimate the relationship between X and Y.

For the given problem we have calculated two error function out of many:

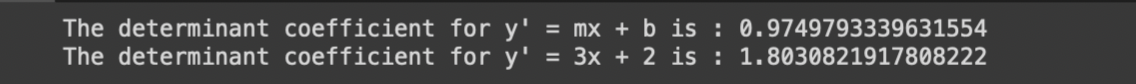
* 1. Root Mean Square Error (RMSE)
  2. Mean Square Error (MSE)

For the solution we have come up with a coding solution where we have got following values for the RMSE and MSE.

A black screen with white text

Description automatically generated with low confidence

And the determinant coefficient solution:



We also observed following regression line for the solution and the optimal solution for given X and Y.

Chart

Description automatically generated

1. Define & explain a residual, in regression fitting

The distance between the actual/observed data points and the regression line of predicted data points in a statistical or machine learning models are residuals. They are used when we want to increase the quality of a model and minimize the errors.

Let’s take the dataset of question no: 7 and analyze the graph we get:

In the figure 1 and figure 2 we can see the red line (regression line) passing between the data point is the predicted data points. The vertical distance (black dotted lines on figure 2) from the observed data points (blue dots) perpendicular to x on the regression red line or predicted line are the residuals.

Chart, scatter chart

Description automatically generated

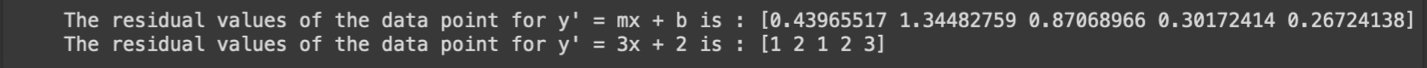
Figure 1

Chart, scatter chart

Description automatically generated

Figure 2

Finally, we calculated each residuals values for the solution.



The above residuals are absolute.

1. Why do we square residuals?

Residuals are basically the distance between the original data points from the predicted regression line [Figure 2, Q1 ]. But, data points may be above or below the regression line, that means mathematically residuals are positive and negative, while summing up the average residuals because of the negative residuals will either cancel the other residuals with same residuals or create dispersion therefor there will be even higher variance in the model. Thus, we cannot find a single straight line that minimizes all residuals simultaneously.

1. Why do we not simply connect the dots as that would minimize the residuals?

We cannot assume that the data points are linearly connected. If we connect the dots, we are just connecting the existing data points resulting very noisy data patterns and overfitting, due to which extrapolation will appear non-realistic.

1. How is a cost function optimized in linear regression using gradient descent?

The gradient is simply the slope of the surface. We try to find the steepest descent path for the next input value for the regression model which should be smaller after each iteration where the optimal descent value that cost function will generate is ~0.

Firstly, we try to find the slope and intercept for each new predicted data point regression line with a parameter called learning rate (alpha) and descent further smaller value of the slope and intercept of the predicted data point i.e., continuously find new line until the model finds the optimal value for the predicted data point or regression line.

Mathematically we compute the m and b with partial derivative resulting as in following snippet.

Table

Description automatically generated with medium confidence

For each new value of m and b we have used the variables as derivative\_m and derivative\_b and new m and b for new prediction data points.

In the below code snippet, we have created a function that replicates the gradient descent.

Text

Description automatically generated

1. Explain what the learning rate is in gradient descent, and why it is important.

The learning rate alpha(α) is the rate of approach for descent to the smallest optimal value in the gradient descent. It is a tuning parameter in an optimization of linear regression model that determines the step size at each iteration while moving toward a minimum of a loss function.

It is important because of following reasons.

If the alpha is too big, then the descent will be aggressive resulting divergence missing out the optimal value. Likewise, if the alpha is very too small then the gradient descent will be small and slow.

We can see the cost function optimization in linear regression using gradient descent in following snippets:

* When alpha is very close to zero, we can see the determinant coefficient is 0.99. We can say the two variables are positively co-related and that 99% of the variability in one is explained by the other.

Graphical user interface

Description automatically generated

* Similarly, if the learning rate is too big then there is divergence in gradient descent resulting infinite value of determinant coefficient, error in slope and intercept. Graphical user interface

  Description automatically generated

Finally, we can see the importance of learning rate in a regression line.

Diagram

Description automatically generated

1. When minimizing a surface, what problem (apart from learning rates) can occur when finding a global extremum?

Learning rate is just a hyperparameter that control the how much change to bring in the model in response to the estimated error each time the model weights are updated, learning rate is itself a manual labour. It varies according to the dataset for every gradient descent, and we cannot conclude a value for learning rate. If the learning rate is too big, the model will have divergence missing out the optimal value and if the learning rate is too small it will take too much time to reach the optimal value. We can see the result of learning rate in Q5 graph [Effect of various learning rates on convergence]. Therefore, when minimizing a surface, model may end up at local minima while finding the global extremum.

References

* <https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10>
* <https://en.wikipedia.org/wiki/Learning_rate>
* <https://www.sfu.ca/math-coursenotes/Math%20157%20Course%20Notes/sec_Optimization.html>
* <https://stats.stackexchange.com/questions/97014/what-are-alternatives-of-gradient-descent>
* <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>

**WBS:**

|  |  |  |
| --- | --- | --- |
| Student Name | Student Number | Total Contribution |
| Aashish Tangnami | 500208080 | 20% |
| Leela Prasad Kavuri | 500209500 | 20% |
| Jay Sanjay Patil | 500208915 | 20% |
| Akshay Rajeev Nambiar | 500209537 | 20% |
| Nijaba Thasneem Naseer | 500209919 | 20% |

Source code used in this assignment we all done by the team.

‘’’

Aashish Tangnami

500208080

’’’

import numpy as np

import matplotlib.pyplot as plt

import math

l1 = [1,2,3,5,7]

l2 = [6,10,10,15,20]

x = np.array(l1)

y = np.array(l2)

x\_mean = x.mean()

y\_mean = y.mean()

x\_sum = sum(x)

y\_sum = sum(y)

xy\_sum = sum(x\*y)

x2\_sum = sum(x \*\* 2)

y2\_sum = sum(y \*\* 2)

m = (xy\_sum - (y\_mean \* x\_sum)) / \

(x2\_sum - (x\_mean \* x\_sum)) #slope for the regression line

b = y\_mean - m \* x\_mean #y-intercept of the data points

y\_hat1 = m \* x + b #predicted data point.

y\_hat2 = 3 \* x + 2 #predicted data point for the given problem.

print(f' value of m : {m} and value of b : {b}')

def rsqr(y, y\_hat):

"""

The determination coefficient (square of the correlation)

shows how much variance is explained by the other variable

in between "x" and "y"

"""

r = math.sqrt(sum((y\_hat- y.mean())\*\*2)\

/sum((y-y.mean())\*\*2))

R\_sqr = r \*\* 2

return R\_sqr

# cost functons

def rmse(y,y\_hat):

"""

Finding the smallet set of distances.

Square rooting the sum of grossly inflated the values.

Root mean square error

"""

var = ((y\_hat - y)\*\*2).sum()

RMSE = math.sqrt(var/len(y))

return RMSE

def mse(y, y\_hat):

"""

Mean Square Error

"""

var = sum((y- y\_hat)\*\*2)

MSE = var/len(y)

return MSE

cost\_function1 = rmse(y, y\_hat1)

cost\_function1\_1 = rmse(y, y\_hat2)

cost\_function2 = mse(y, y\_hat1)

cost\_function2\_1 = mse(y, y\_hat2)

print(f'Root mean square error for y\' = 3x + 2 is : {cost\_function1\_1}')

print(f'Root mean square error for y\' = mx + b is : {cost\_function1}')

print(f'Mean square error for y\' = 3x + 2 is : {cost\_function2\_1}')

print(f'Mean square error for y\' = mx + b is : {cost\_function2}')

title\_label1 = 'y\' = mx + b'

title\_label2 = 'y\' = 3x + 2'

corelation1 = rsqr(y, y\_hat1)

corelation2 = rsqr(y, y\_hat2)

print(f'The determinant coefficient for y\' = mx + b is : {corelation1}')

print(f'The determinant coefficient for y\' = 3x + 2 is : {corelation2}')

def plot\_graph(y, y\_hat, title):

residuals = y\_hat - y

fig = plt.figure()

# plt.figure(figsize=(14,14))

plt.scatter(x, y, marker='o', label='Y')

plt.plot(x, y\_hat, label='y\'', color='red')

plt.bar(x, residuals, width=0.9, label='residuals', color='green')

plt.legend()

plt.title(f'Linear Regression line for {title}')

plt.xlabel('x')

plt.ylabel('y')

plot\_graph(y, y\_hat1, title\_label1)

plot\_graph(y, y\_hat2, title\_label2)

residuals1 = y\_hat1 - y

residuals2 = y\_hat2 - y

print(f'The residual values of the data point for y\' = mx + b is : {abs(residuals1), residuals1}')

print(f'The residual values of the data point for y\' = 3x + 2 is : {abs(residuals2), residuals2}')

def gradient\_descent(m,b,alpha):

iterations = 100

n = len(x)

for \_ in range(iterations):

y\_pred = m \* x + b

derivative\_m = -(2/n) \* sum(y \* (y - y\_pred))

derivative\_b = -(2/n) \* sum(y - y\_pred)

m = m - alpha \* derivative\_m

b = b - alpha \* derivative\_b

print(f'Values - m : {m}, b : {b}, over {iterations} with learning rate(alpha) : {alpha}')

y\_pred = m \* x + b

r1 = rsqr(y, y\_pred)

print(f'The value of determinant coefficient r^2 is {r1}')

r2 = rmse(y, y\_pred)

print(f'The value of root mean square error is {r2}')

fig = plt.figure()

ax1 = fig.add\_subplot(111)

ax1.scatter(x,y,color ='blue', label='y', marker='o')

ax1.plot([min(x), max(x)], [min(y\_pred), max(y\_pred)], label='y\'',color='red')

plt.legend()

plt.title('Linear Regression with Gradient Descent')

plt.xlabel('x-values')

plt.ylabel('y-values')

ax2 = ax1.twinx()

gradient\_descent(m,b,0.00001) #assuming slope and intercept with learning rate(alpha)

gradient\_descent(3,2,0.00001) #taking slope and intercept as 3, 2 respectively with learning rate(alpha)