**Assignment 2**

**Instructions:**

* Type your answers in the spaces provided in this Word document. Your submission should not exceed 11 pages, including this page.
* Submit the *Declaration of Academic Integrity* before submitting your assignment.

--------------------------------------------------------------------------------------------------------------------

**Introduction**

Given a set of data points with at least one predictor and one continuous response variable, we want to construct a linear model to predict the response. This is the aim of **Linear Regression**, which is a supervised learning technique.

In the context of this assignment, measurements of 25 fishes of the species Smelt are collected. The data can be found in the file *fish.csv*. The following table lists the variables used in the file and their descriptions:

|  |  |
| --- | --- |
| **Variable** | **Description** |
| *Weight* | Weight in 0.1gram |
| *Length* | Length of the fish in cm |
| *Width* | Width of the fish in cm |

The response variable is *Weight,* and the predictors are *Length* and *Width*.

**Simple Linear Regression (SLR)**

We will first build a SLR model using *Length* as the predictor to predict *Weight*.

In SLR notations, let:

= predictor value of the *i*-th data point

= actual response value of the *i*-th data point

= predicted response value of the *i*-th data point based on model

Thus, , where values of *a* (intercept) and *b* (slope) are to be determined.

The squared-error of the *i*th prediction is . Errors (also known as residuals) are squared to remove the signs, so that errors of opposite signs do not cancel out each other, giving the false impression of small aggregated errors.

Then, we define **Error function** as the mean of squared-error (of the whole data set):

We want to find the values of *a* and *b* such that the Error function is **minimised**.

The resultant equation will give the best-fit line that passes through the data points.

**MODEL 1: SLR with intercept *a* fixed ⇒**  (25 marks)

We will first build a SLR model to predict *Weight* (*y*) using *Length* (*x*) as the predictor.

Suppose it is believed that weight is directly proportional to length. This means that  is a constant multiple of  and . Then, in the SLR model, we will only need to determine slope *b*.

(a) Express Error function in terms of *b* only. Hence, derive .

|  |
| --- |
|  |

(b) Use univariate gradient descent algorithm to find the value of *b* for which is at its minimum. Write your Python code in a single cell and copy-paste your code below.

|  |
| --- |
|  |

(c) Describe the changes and decisions you made on the parameters for your solution to reach convergence.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Attempt** | **Parameters (starting value (b), Learning Rate, Epsilon, Max iterations)** | **Results (Local Min, No. of Iterations, E(b))** | **Changes made to parameters** | **Observations** | | 1 | Starting value: 0,  Learning Rate: 0.00001, Epsilon: 0.0001,  Max. iterations: 10000 | Local min:10.517, Iterations: 2126,  E(b): 851.496 | nil | nil | | 2 | Starting value: 0,  Learning Rate: 0.000001,  Epsilon: 0.0001,  Max. iterations: 100000 | Local min:10.177,  Iterations: 12578, E(b): 870.17 | - Increased Max iterations  - decreased learning rate | - Local min. decreased  - Error function increased | | 3 | Starting value: 0,  Learning rate: 0.001,  Epsilon: 0.001  Max iterations: 100000 | Local min: 10.552  Iterations: 27  E(b): 851.3102 | -Increased learning rate  - Increased epsilon | - No. of iterations taken reduced  - Local min increased  - Error function decreased | | 4 | Starting value: 0,  Learning rate: 0.005,  Epsilon: 0.001  Max iterations: 100000 | Local Min:10.555354  Iterations: 10  E(b): 851.309384 | -Increased learning rate | - Local min increased  - Iterations decreased  - Err function decreased | | 5 | Starting value: 5,  Learning rate: 0.005,  Epsilon: 0.001  Max iterations: 50 | Local Min:10.555706  Iterations: 9  E(b): 851.309388 | -Starting value is changed to 5 | - Local min, E(b) increased but change is negligible (as good as 0)  - Iters taken: 9 | | 6 | Starting value: 10,  Learning rate: 0.005,  Epsilon: 0.001  Max iterations: 50 | Local Min:10.5556960  Iterations: 7  E(b): 851.309387 | -starting value changed to 10 | -Local min, E(b) increased but change is negligible (as good as 0)  - Iters taken: 7 | | 7 | Starting value: 15,  Learning rate: 0.005,  Epsilon: 0.001  Max iterations: 50 | Local Min:10.5553141  Iterations: 9  E(b): 851.309386 | -starting value changed to 15 | - Local min, E(b) increased but change is negligible (as good as 0)  - Iters taken: 9 |   Explanation:  My starting parameters are: Starting value= 5, Learning Rate= 0.00001, Epsilon= 0.0001, Max. iterations= 10000. After getting the results (local min=10.517, E(b)=851.496), I decided to decrease the learning rate to 0.000001 and max iterations to 100k as I wanted to see if it would improve the accuracy. Observations from decreasing the learning rate is that the local minimum decreased to 10.177 and E(b) increased to 870.17 but this took too many iterations (12578). So I decided to decrease the iterations by increasing the learning rate to 0.001 and epsilon to 0.001. After doing so, it produced similar results but with lesser iterations which is what I wanted. I was satisfied with 25 iterations but wanted to see if increasing the learning rate would affect the results much, so I decided to change the learning rate to 0.005. The new set of parameters (attempt 5) produced similar results but used only 9 iterations. The difference in local min and E(b) in attempt 4 and 5 (refer to table) do not vary much. As the parameters no longer require so many iterations, I decided to lower the max. iterations to 50. Finally, I tuned the starting value. I started with 0 which gave the results as shown on the table (attempt 4). Then I increased the starting value to 5. As it decreased the number of iterations yet maintain the results with minimal change, I decided to increase the starting value in multiples of 5 (5,10,15..) After trying the different values, I have concluded that the most optimal starting value is 10, without compromising on the local minimum and E(b). Even though there is a slight change in the local min and E(b) from attempt 4 to attempt 7 , but it is very minimal/close to 0. So I picked the parameters from attempt 7.  Final parameters: Starting value =10, learning rate=0.005, epsilon= 0.001, max iterations= 50  Final values : Local Min:10.5553141/ 10.6 (3sf) , Iterations: 9, E(b): 851.309386/ 851.3 (3sf) |

(d) Describe your MODEL 1 by filling the information below.

|  |
| --- |
| Final MODEL 1 equation is:  Minimum value of Error function is: 851.3 (3sf)  Number of iterations ran to reach convergence: 7 |

**MODEL 2: SLR ⇒**  (25 marks)

Now we apply the SLR model where both intercept *a* and slope *b* are to be determined, when predicting *Weight* (*y*) using *Length* (*x*) as the predictor.

(a) Express Error function in terms of *a* and *b*. Hence, derive and .

|  |
| --- |
|  |

(b) Use gradient descent algorithm to find the values of *a* and *b* for which is at its minimum. Write your Python code in a single cell and copy-paste your code below.

|  |
| --- |
|  |

(c) Describe the changes and decisions you made on the parameters for your solution to reach convergence.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Attempt** | **Parameters (Next\_a, Next\_b, Alpha, Epsilon, Max iterations)** | **Results (a,b, E(a,b), Iterations taken)** | **Changes made to parameters** | **Observations** | | 1 | Next\_a: 0,  Next\_b: 10.6,  Alpha: 0.0001,  Epsilon: 0.001,  Max. iterations: 100,000 | a = -48.21  b = 14.72  E(a,b) = 660.43 | Nil | -hit max iterations but E(a,b) is still quite high  - took 100k iterations but E(a,b) only decreased by ~200 | | 2 | Next\_a: 0,  Next\_b: 10.6,  Alpha: 0.001,  Epsilon: 0.001,  Max. iterations: 50,000 | a = -244.114  b= 31.664  E(a,b) = 142.48 | -increased learning rate  - decreased iterations | - iterations taken is lesser  - E(a,b) reduced | | 3 | Next\_a: 0,  Next\_b: 10.6,  Alpha: 0.009,  Epsilon: 0.00001,  Max. iterations: 50,000 | Nil [See observations col] | -increased learning rate to 0.009  - decreased epsilon to 0.00001 | -After 1k+ iterations, E(a,b) became infinity and around the 2k+ mark, a,b and E(a,b) became NaN  - shows that alpha=0.009 is too high as the values started swinging from positive to negative midway. | | 4 | Next\_a: 0,  Next\_b: 10.6,  Alpha: 0.005,  Epsilon: 0.00001,  Max. iterations: 50,000 | a = -264.199  b = 33.401  E(a,b) = 137.53  Iterations taken: 47911 | - decreased alpha to 0.005 | - E(a,b) decreased  - a,b increased | | 5 | Next\_a: 5,  Next\_b: 10.6,  Alpha: 0.005,  Epsilon: 0.00001,  Max. iterations: 50,000 | a = -264.199  b = 33.401  E(a,b) = 137.53  Iterations taken: 48096 | -increased next\_a to 5 | - values are the same [comparing to 3rd attempt]  - Took more iterations | | 6 | Next\_a: 10,  Next\_b: 10.6,  Alpha: 0.005,  Epsilon: 0.00001,  Max. iterations: 50,000 | a = -264.199  b = 33.401  E(a,b) = 137.53  Iterations taken: 48227 | -increased next\_a to 10 | -all values remained the same  - took more iterations |   Explanation:  My starting parameters are: Next\_a=0, Next\_b=10.55535434 (from Model 1), Alpha=0.0001, Epsilon=0.001 and Max iterations = 100k. However, it hit the max iterations and E(a, b) only decreased by ~200. This could be because my alpha is too small hence the model is taking a long time to reach the minimum point. So I decided to increase the alpha to 0.001 and decrease the maximum iterations to 50k while retaining the other parameters. The results from the new params are a= -244.144 , b=31.664, E(a,b)= 142.48. E(a,b) reduced tremendously so this means that the parameters used are somewhat correct. However, I wanted to see if I can decrease the number of iterations taken and test if E(a,b) is the lowest/ best loss value that I can get. So I decided to increase the alpha to 0.009 and decrease the epsilon to 0.00001. However, it gave infinity after 1k+ iterations and NaN after 2k+ iterations. This shows that alpha is too large so the model has difficulties converging at the minimum point. So I decided to decrease the alpha to 0.005 and retain the remaining parameters. The results from this set of parameters are: a=-264.199, b=33.401 and 3(a,b)= 137.53 with 47911 iterations. Afterwards, I wanted to see if next\_a affects E(a,b) so I increased next\_a to 5 and 10 [attempts 5 and 6] and decreased alpha back to 0.05 [was 0.09 but too high]. However, both attempts took more and more iterations yet produced the same results. Hence I concluded that for this model, a higher next\_a value would require more iterations. Attempts 4,5,6 gave the same values however attempt 4 took significantly lesser iterations. Hence I will choose the parameters from attempt 4 for my model  Final parameters: Next\_a: 0, Next\_b: 10.6 (3sf), Alpha: 0.005, Epsilon: 0.00001, Max. iterations: 50,000  Final Values: a = -264.199/ -264.2 (3sf), b = 33.401/ 33.4 (3sf) , E(a,b) = 137.53/ 137.5 (3sf) , Iterations taken: 47911 |

(d) Describe your MODEL 2 by filling the information below.

|  |
| --- |
| Final MODEL 2 equation is:  Minimum value of Error function is: 137.5 (3sf)  Number of iterations ran to reach convergence: 47911 |

**Conclusion on SLR** (15 marks)

(a) Using Python (or other software), in a single figure, plot the data points (scatterplot) together with the linear lines representing the two models. Insert the figure below and describe what you observe regarding the location of the data and the linear lines.

|  |
| --- |
| Description:  In the above plot, both models 1 and 2 have a positive relationship as seen from the upward slope. The green scatter points are the observations while the remaining 2 are linear lines. Our aim is to find the best fit line, one which minimises the difference between the estimated line.  For model 1 (blue line), the location of the data points are far from the best fit line. Furthermore, the number of points above the line varies from the number of points below the line and model 1’s line does not pass through as many points as possible. This shows that there isn’t a good approximation of the data if we use model 1. For model 2 (orange line), the location of the data points are close to the linear line. In addition, the number of points above and below the line is rather equal and the model passes through quite a few points.  Taking into consideration the above mentioned points as well as the Error function value (Model 1: 851.309 and model 2: 137.53) , model 2 is of a better fit. |

(b) In a linear regression model, the constant  is commonly interpreted as the value of the response variable when the predictor variable is zero. In your Model 2, can you interpret your value of  as such? Explain.

|  |
| --- |
| In model 2, ‘a’ means that when there is no length (length=0), the weight will be -264.2. However, this does not make sense as weight can only be positive. |

**MODEL 3: MLR ⇒**  (25 marks)

We can extend the SLR model to include more predictors. A linear regression model with more than 1 predictor is called **Multiple Linear Regression** (MLR) model.

Apply the MLR model where intercept *a*, and slopes *b* and *c* are to be determined, when predicting *Weight* (*y*) using *Length* (*x*) and *Width* (*w*) as the predictors.

(a) Explain how gradient descent algorithm can be extended for MODEL 3.

|  |
| --- |
| For model 3, the equation requires 3 constants: a,b and c. Like how I did model 2, I can use a and b derived from the previous model as a guiding starting point without brute forcing. Additionally, we will also require the following: 1) E(a,b,c) 2) Ea(a,b,c) 3) Eb(a,b,c) and 4) Ec(a,b,c) [Working is shown below] |

(b) Use gradient descent algorithm to find the values of *a*, *b* and *c* for which Error function is at its minimum. Write your Python code in a single cell and copy-paste your code below.

|  |
| --- |
|  |

(c) Describe the changes and decisions you made on the parameters for your solution to reach convergence.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Attempt** | **Parameters (Next\_a, Next\_b, Alpha, Epsilon, Max iterations)** | **Results (a,b,c, E(a,b,c,), Iterations taken)** | **Changes made to parameters** | **Observations** | | 1 | Next\_a: -264.2,  Next\_b: 33.4,  Next\_c: 0  Alpha: 0.005,  Epsilon: 0.00001,  Max. iterations: 50,000 | a = -212.737  b = 21.6173  c = 59.4936  E(a,b,c ) = 46.6132  Iterations : 46235 | Nil | Nil | | 2 | Next\_a: -264.2,  Next\_b: 33.4,  Next\_c: 0  Alpha: 0.004,  Epsilon: 0.00001,  Max. iterations: 50,000 | a = -213.10901  b = 21.665  c = 59.3621  E(a,b,c ) = 46.6320  Iterations : 55695 | -decreased learning rate to 0.004 | - Slight decrease in ‘a’  - slight increase in ‘b’ and ‘c’  -took more iterations  -E(a,b,c) increased | | 3 | Next\_a: -264.2,  Next\_b: 33.4,  Next\_c: 0  Alpha: 0.006,  Epsilon: 0.00001,  Max. iterations: 50,000 | a = -212.4629  b = 21.58164  c = 59.590  E(a,b,c ) = 46.6006  Iterations : 39673 | -increased learning rate to 0.006 | - ‘a’ and ‘c’ increased  - ‘b’ decreased  - took lesser iterations  - E(a,b,c) decreased | | 4 | Next\_a: -264.2,  Next\_b: 33.4,  Next\_c: 5  Alpha: 0.006,  Epsilon: 0.00001,  Max. iterations: 50,000 | a = -212.4629  b = 21.58164  c = 59.590  E(a,b,c ) = 46.6006  Iterations : 39383 | -increased next\_c to 5 | -everything remained the same but with lesser iterations | | 5 | Next\_a: -264.2,  Next\_b: 33.4,  Next\_c: 10  Alpha: 0.006,  Epsilon: 0.00001,  Max. iterations: 50,000 | a = -212.4629  b = 21.58164  c = 59.590  E(a,b,c ) = 46.6006  Iterations : 39088 | -increased next\_c to 10 | -everything remained the same but with lesser iterations |   Explanation:  My starting parameters are Next\_a: -264.2, Next\_b: 33.4, Next\_c: 0, Alpha: 0.005, Epsilon: 0.00001 and Max. iterations: 50,000 which gave E(a,b,c) of 46.6132. I took next\_a and next\_b values from the previous model as a guiding point and left c as 0. Then, I decided to decrease the learning rate to 0.004 to see if it would decrease E(a,b,c) and the number of iterations. However, E(a,b,c) increased to 46.6320 which means that decreasing the learning rate is not helping as it has skipped the optimal value so I decided to increase the learning rate to 0.006. With 0.006 as the new learning rate, it gave E(a,b,c) as 46.6006 which is significantly lower than the previous 2 attempts so I decided to keep the learning rate as It as and tuned the next\_c value with 5 and 10 which gave the same values except a change in number of epsilons. Next\_c=5 gave 39383 iterations and next\_c=10 gave 39088 iterations. For this model, a higher next\_c value uses lesser iterations. In the end, I picked the parameters from my 5th attempt as it gave the same results but with lesser iterations.  Final parameters: Next\_a: -264.2, Next\_b: 33.4, Next\_c: 10, Alpha: 0.006, Epsilon: 0.00001, Max. iterations: 50,000  Final values: a = -212.4629 / -212.5 (3sf) , b = 21.58164/ 21.6 (3sf), c = 59.590/ 59.6 (3sf) , E(a,b,c ) = 46.6006/ 46.6 (3sf) , Iterations : 39088 |

(d) Describe your MODEL 3 by filling the information below.

|  |
| --- |
| Final MODEL 3 equation is:  Minimum value of Error function is: 46.6 (3sf)  Number of iterations ran to reach convergence: 39088 |

**Conclusion** (10 marks)

(a) David used gradient descent algorithm to find the 3 models. Next, he computed the predicted weights using the 3 models for all the data points in the dataset. He noticed that for one of the data points, the error of the predicted weight in Model 1 from the actual weight is the smallest, compared to the other 2 models. Is this possible, assuming he has done his gradient descent algorithm correctly? Explain.

|  |
| --- |
| Yes, it is still possible for one of the data points to have an error that is the smallest as compared to the 2 models. Model 1’s linear line only passes through 2 points as compared to model 2 which passed through more points. Looking at the screenshot below, for the point circled in green, if it is predicted using model 1, it would have a smaller residual error (or close to 0 since it is touching the linear line) as compared to if it is predicted using model 2’s linear line whereby the residual error will be much higher. |

(b) Compare the 3 models. Which model will you use to predict weight in this context? Explain.

|  |
| --- |
| I would use model 3 as it has the lowest residual error as compared to the other 2 models. A model with a lower residual error means that it is better fitted and has more accurate predictions as compared to a model with a higher residual error. The second reason is because model 3 passes through more points as compared to model 1 and 2. When a model passes through many points, it mimics the trendline and minimizes the distance of those points from that line, making the prediction more accurate. |