



Machine Learning

LABORATORY: REGRESSION

NAME:

STUDENT ID#:

Objectives:

- This assignment aims to develop a deeper understanding of linear regression by exploring their mathematical foundations, implementation, and evaluation using gradient descent.
- Understand the concepts and mathematics behind regression models.
- Implement and Evaluate regression models from scratch.
- Understand the key concepts of regression, including least squares, likelihood estimation, and bias–variance trade-off.

Part 1. Background

Here we will know the basics concepts that we will use for the implementation of this algorithm.

What is Regression? Regression is a statistical approach to model the relationship between a dependent variable (output) and one or more independent variables (inputs). Linear Regression is used to solve problems where the relationship between variables can be reasonably approximated by a straight line.

Tasks: In this assignment, you are required to implement linear and logistic regression models using only NumPy. You will get no points by simply calling `sklearn.linear_model.LinearRegression`. Your task is to train these models with gradient Descent on a provided dataset, evaluate their performance, and test them on unseen data. The dataset and sample code can be found here: <https://github.com/Satriosnjya/ML-Labs.git>

Part 2. Arithmetic Instructions.

Step

Procedure

- Define the Model:** Linear regression predicts the output y using a **linear function**: $y(x, w) = w_0 + w_1x_1 + \dots + w_Dx_D$. In simple linear regression with only one feature x : $y = mx + b$
- Mean Square Error (MSE) Loss Calculation**
From the textbook, *Equation (4.11)* defines the sum-of-squares error function:

$$E_D(w) = \frac{1}{2} \sum_{n=1}^N (t_n - w^T \phi(x_n))^2$$
 which directly follows the equation, computing the squared difference between predicted and actual values.
- Gradient Descent for Weight Updates**
From the textbook, *Equation (4.12)* defines the gradient of the log-likelihood:

$$\nabla_w \ln p(t | X, w, \sigma^2) = \frac{1}{\sigma^2} \sum_{n=1}^N (t_n - w^T \phi(x_n)) \phi(x_n)^T$$
 This follows the principle of **gradient descent**, adjusting weights by subtracting a scaled version of the gradient.
- Model Training Using Gradient Descent**
Equation Reference: From the textbook, *Equation (4.22)* for sequential learning (LMS Algorithm):



$$w^{(\tau+1)} = w^{(\tau)} + \eta(t_n - w^T \phi(x_n)) \phi_n$$

Follows this equation by iteratively updating w based on the gradient.

5 Mean Square Error (MSE) as Evaluation Metric

From the textbook, *Equation (4.20)* defines the MLE estimate for variance:

$$\sigma_{ML}^2 = \frac{1}{N} \sum_{n=1}^N (t_n - w_{ML}^T \phi(x_n))^2$$

Part 3. Data Transfer Instructions.

Step	Procedure
1	Download dataset and example code from https://github.com/Satriosnjva/ML-Labs.git
2	Open colab.research.google.com or you can run the python code in your own computer
3	Colab : Make a new Notebook, connect the runtime, then upload regression_data.npy
4	Load Libraries Load Libraries and Generate Data <pre>import numpy as np import matplotlib.pyplot as plt</pre>
5	Generate Data # Load dataset <pre>x_train, x_test, y_train, y_test = np.load('regression_data.npy', allow_pickle=True) # Reshape targets y_train = y_train.reshape(-1,) y_test = y_test.reshape(-1,) # Add bias term (column of ones) train_data = np.hstack((x_train, np.ones((x_train.shape[0], 1)))) test_data = np.hstack((x_test, np.ones((x_test.shape[0], 1))))</pre>

Grading & Submission Instructions

Hands-on Tasks:

- (10%) Implement Standard Linear Regression using Gradient Descent
 - Compute gradients for weight (m) and bias (b).
 - Update weights.
 - Output : Model parameters (weight and bias)
- (10%) Evaluate the model using MSE for standard regression
 - Complete compute_mse() function
 - Output : MSE for standard regression
- (10%) Implement Ridge Regression with L2 Regularization
 - Modify loss function to include L2
 - Compute updated gradients for weight and bias.
 - Output : Model parameters and MSE for Ridge Regression
- (10%) Plot the training loss curve.
 - Store the loss at each iteration and plot it using matplotlib.
 - Try different values for: Learning Rate (lr) and Number of Iterations ($iterations$)
 - Output : Loss curve comparison of Standard regression and ridge regression

Assignment:

- (20%) Implement Closed-form Ridge Regression (Refer to Equation 4.27)

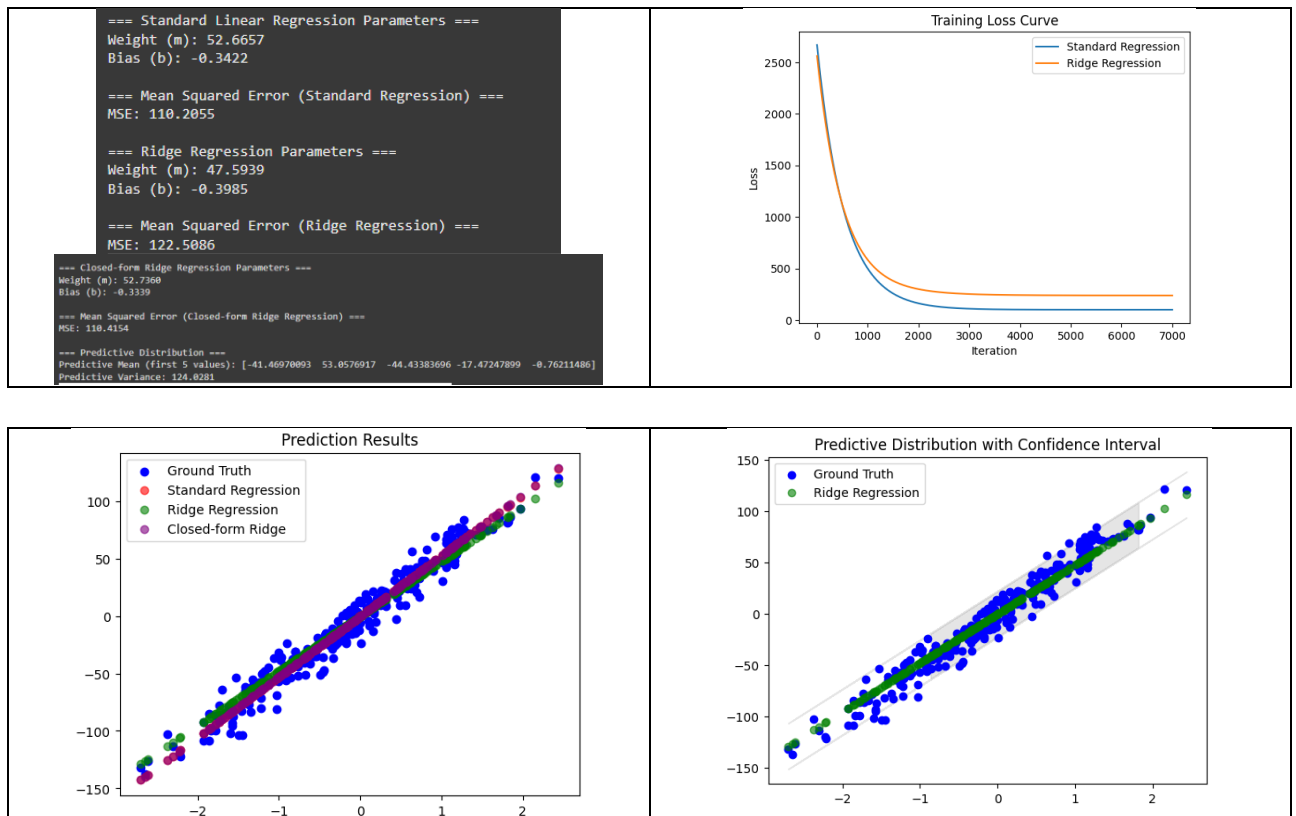


- a. Fill in `closed_form_ridge()` function
 - b. Compute `w_closed_form`
 - c. Output : Model parameters and MSE for closed-form Ridge Regression
6. (20%) Implement predictive distribution (Refer to Equation 4.33)
 - a. Fill in `predictive_mean` and `predictive_variance`
 - b. Print 5 values of `predictive_mean`
 - c. Output : Predictive variance & example predictions
7. (20%) Visualize predictions and confidence intervals
 - a. Fill in missing values in scatter plot (`plt.scatter`)
 - b. Implement confidence interval shading using `plt.fill_between()`.
 - c. Output :
 - i. Prediction plot of standard regression, ridge regression and closed-form regression.
 - ii. Predictive distribution with confidence interval.

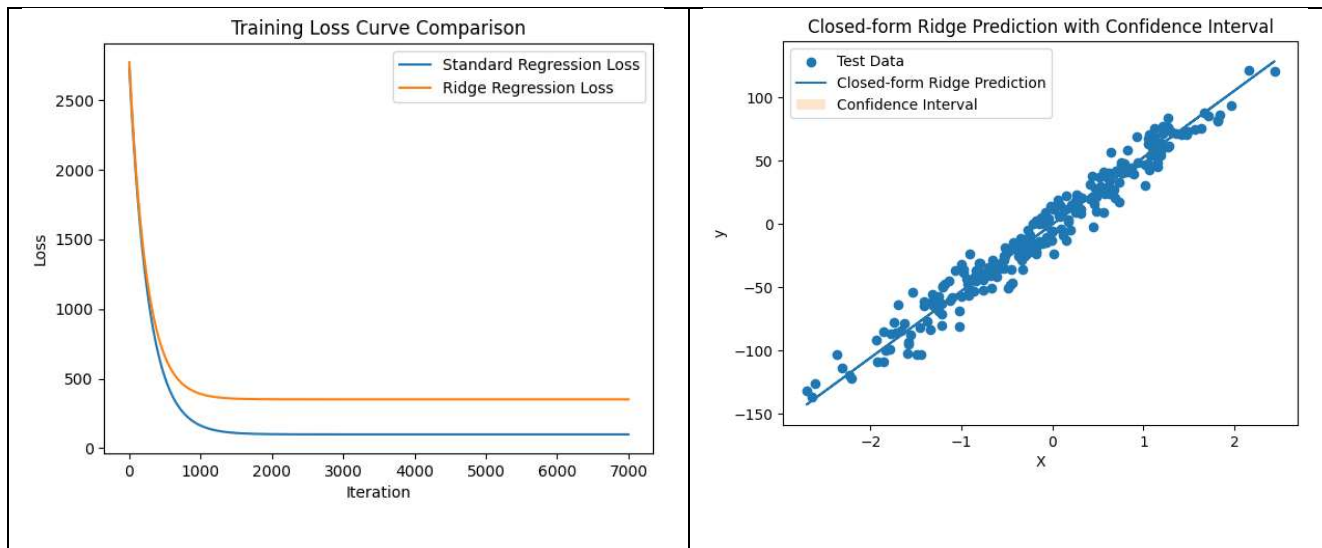
Submission:

1. Report: Include screenshots of your results for each task (model training, evaluation, and plots) in the last pages of this PDF file.
2. Code: Submit your complete Python script (.py or .ipynb notebook).
3. Upload both your report and code to the E3 system. Name your files correctly:
 - a. Report: StudentID_Lab1.pdf
 - b. Code: StudentID_Lab1.py or StudentID_Lab1.ipynb
4. 1 day late: 10% deduction from total score. (Due Date : Sunday 9:00 PM)
5. Plagiarism is **strictly prohibited**. Submitting copied work from other students will result in penalties.

Sample Output: Please note that it is for reference only



Put Your Code Results Here:



=== Standard Linear Regression Parameters ===

Weight (m): 52.74342374590863

Bias (b): -0.33377782560722613

=== Mean Squared Error (Standard Regression) ===

MSE: 110.4378316313167

=== Ridge Regression Parameters ===

Weight (m): 47.62796320640477

Bias (b): -0.39491374426823583

=== Mean Squared Error (Ridge Regression) ===

MSE: 122.25192735576522

=== Closed-form Ridge Regression Parameters ===

Weight (m): 52.73598792691733

Bias (b): -0.3338629653682158

=== Mean Squared Error (Closed-form Ridge Regression) ===

MSE: 110.41538156212263

=== Predictive Distribution ===

Predictive Mean (first 5 values): [-45.84237647 58.89774512 -49.12675743 -19.25249735 -0.73674735]

Predictive Variance (first 5 values): [0.23360961 0.31825603 0.2491052 0.14879557 0.1324595]

