CSE 474/574: Introduction to Machine Learning

Summer 2024

**Instructor: Nitin Kulkarni**

# Assignment 2
# Regression Methods

**Checkpoint: June 13, Thu 11:59pm**

**Due Date: June 20, Thu, 11:59pm**

## Description

Our second assignment is focused on learning how to build regression models from scratch. In the first part of the assignment, you are expected to preprocess, analyze and visualize datasets, this is one of the first and core steps in machine learning pipeline. In the second part, you will implement linear regression using ordinary least squares and in the third part, you will extend it to a ridge regression model.

The goal of this assignment is to practice performing data analysis, implementing, and understanding the practical implementation of different regression models.

For this assignment, sklearn or any other libraries with in-built functions that help to implement ML methods cannot be used. Submissions with used ML libraries (e.g. sklearn) will not be evaluated.

## Part I: Data Analysis [20 points]

In this part, we explore how to load, process, visualize and analyze different types of datasets. Data preprocessing and visualizing are some of the core parts of the machine learning pipeline.

Select any TWO datasets from the list below and provide a data analysis. All datasets are provided in the folder.

- Amazon Top Selling Book dataset
- Netflix dataset
- Diamond dataset

- Titanic dataset
- Wine Quality dataset
- Insurance dataset

**STEPS**

1. Pick a dataset from the list above
2. In your Jupyter Notebook or .py script, define a section as 'Part I: Data Analysis – YOUR DATASET NAME'
3. Load a dataset (e.g. `pd.read_csv()`)
4. Using inbuilt functions in 'pandas' library, extract the main statistics about the dataset (more details). Use at least 5 functions (e.g. `describe(), head()`).
5. Perform data preprocessing:
    a. Calculate the sum of missing entries in the dataset (e.g. `isnull().sum(axis=0)`)
    b. If applicable, for missing values:
        i. Fill the missing values with the most frequent value or
        ii. Drop the rows with missing values (e.g. `dropna()`)
6. If applicable: convert the columns with datatype string to data type categorical
7. Using any data visualization library (e.g. matplotlib, seaborn, plotly), provide at least 5 visualization graphs related to your dataset. You can utilize any columns or a combination of columns in your dataset to generate graphs. E.g., correlation matrix, features vs the target, counts of categorical features vs the target.
8. Go to Step 1. Altogether, you should provide data analysis for TWO datasets.

**In your report for Part I**

For each dataset:

1. Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise?
2. Provide the main statistics about the entries of the dataset (mean, std, number of missing values, etc.)
3. Provide at least 5 visualization graphs with short description for each graph, e.g., discuss if there any interesting patterns or correlations.

**Part I submission includes:**

- Report (as a pdf file)
- Jupyter Notebook (.ipynb) or python script .py – one file for the TWO datasets with all saved outputs

## Part II: Linear Regression [50 points]

Implement linear regression using the ordinary least squares (OLS) method to perform direct minimization of the squared loss function.

$$J(\boldsymbol{w}) = \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

In matrix-vector notation, the loss function can be written as:

$$J(\boldsymbol{w}) = (y - Xw)^T (y - Xw)$$

Here $\boldsymbol{X}$ is the input data matrix, $\boldsymbol{y}$ is the target vector, and $\boldsymbol{w}$ is the weight vector.

**Implement linear regression for one dataset using the following steps:**

**Steps:**

1. Import required libraries (not allowed: sklearn or any other libraries with in-built functions that help to implement ML)
2. Read, preprocess and print main statistic about the dataset (your code from Part I can be reused).
3. Convert features with string datatype to categorical and normalize non-categorical features if needed.
4. Choose your target $y$
5. Create the data matrices for X (input) and $y$ (target) in a shape $\mathrm{X} = N \times (d+1)$ and $\mathrm{y} = N \times 1$, where $N$ is the number of data samples and $d$ is the number of features.
6. Divide the dataset into training and test, as 80% - training, 20% - testing dataset.
7. Print the shape of your `X_train, y_train, X_test, y_test`
8. Calculate the weights with the OLS equation:

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

9. Get the predictions and calculate the Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \boldsymbol{w}^T x_i)^2}$$

10. Plot the predictions vs the actual data.

**In your report for Part II:**

1. Provide your loss value and the weight vector.
2. Show the plot comparing the predictions vs the actual test data.
3. Discuss the benefits/drawbacks of using OLS estimate for computing weights.

**Part II Submission:**

- Report (as a pdf file) – combined with your report from Part I.
- Jupyter Notebook (.ipynb) or python script .py with all saved outputs.

## Part III: Ridge Regression [30 points]

**Use your implementation from Part II and extend it to Ridge Regression.**

Implement parameter estimation for ridge regression by minimizing the regularized squared loss as follows:

$$J(\boldsymbol{w}) = \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

In matrix-vector notation, the squared loss can be written as:

$$J(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

OLS equation for Ridge regression can be estimated as

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X} + \boldsymbol{\lambda} \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

**Implement linear regression for one dataset using the following steps:**

**Steps:**

Reuse steps from Part II while using OLS equation for Ridge regression to learn the parameters.

**In your report for Part III:**

1. Provide your loss value and the weight vector
2. Show the plot comparing the predictions vs the actual test data
3. Discuss the difference between Linear and Ridge regressions. What is the main motivation of using l2 regularization?

**Part III Submission:**

- Report (as a pdf file) – combined with your report from Part I, Part II.
- Jupyter Notebook (.ipynb) or python script .py combined with Part II with all saved outputs.

## Bonus points [10 points]

### Gradient Descent from Scratch

Based on your implementation of Ridge Regression in Part III, implement a gradient descent method from scratch.

Provide the RMSE error and comparison with the results from Part III.

## Deliverables

There should two parts in your submission for both the checkpoint and final submissions

### Report

The report should be delivered as a separate pdf file. You can follow the NIPS template as a report structure or our provided word doc, both of which can help you to get started. You may include comments in the Jupyter Notebook, however you will need to duplicate the results in the separate pdf file.

All the references can be listed at the end of the report. There is no minimum requirement for the report size, just make sure it includes all the required information.

For the final submission, combine the reports for all parts into one file UBIT TEAMMATE1_TEAMMATE2_assignment2_report.pdf

(e.g. nitinvis_nandinic_assignment2_report.pdf)

### Code

Python is the only code accepted for this assignment. You can submit the code in the Jupyter Notebook or as Python script. Ensure that your code follows a clear structure and contains comments for the main functions and some specific attributes related to your solution. You can submit multiple files, but they all need to be labeled with a clear name.

After executing command python main.py in the first level directory or Jupyter Notebook, it should generate all of the results and plots you used in your report and print them out in a clear manner.

For the final submission we should expect to have 2 Jupyter Notebooks:

- Part I: TEAMMATE1_TEAMMATE2 _assignment2_part1.ipynb
  (e.g. nitinvis_ shekhar7_assignment2_ part1.ipynb)
- Part II & III: TEAMMATE1_TEAMMATE2 _assignment2_part2_3.ipynb
  (e.g nitinvis_shekhar7_assignment2_ part2_3.ipynb)

# ASSIGNMENT STEPS

## 1. Register your team (June 10)

You may work individually or in a team of up to 2 people. The evaluation will be the same for a team of any size.

Register your team at UBLearns (UBlearns > Tools > Groups). In case you join the wrong group, make a private post on Piazza.

## 2. Submit checkpoint (June 13)

- Complete Part I and Part II
- Submit the code in a .ipynb and your draft report in pdf at UBLearns > Assignments.
- The code of your implementations should be written in Python. You can submit multiple files, but they all need to be labeled clearly.
- All project files should be packed in a ZIP file named: UBIT TEAMMATE1_TEAMMATE2 _assignment2_checkpoint.zip
- (e.g., nitinvis_nandinic_assignment2_checkpoint.zip).

## 3. Submit final results (June 20)

- Fully complete all parts of the assignment
- Submit to UBLearns > Assignments
- The code of your implementations should be written in Python. You can submit multiple files, but they all need to be labeled clearly.
- All assignment files should be packed in a ZIP file named: UBIT TEAMMATE1_TEAMMATE2 _assignment2_final.zip
- (e.g. nitinvis_nandinic_assignment2_final.zip).
- Your Jupyter notebook should be saved with the results. If you are submitting python scripts, after extracting the ZIP file and executing command python

main.py in the first level directory, all the generated results and plots you used in your report should appear printed out in a clear manner.

- Include all the references that have been used to complete the assignment.
- If you are working in a team of two people, we expect equal contribution for the assignment. Provide a contribution summary by each team member in a form of a table:

| Team Member | Assignment Part | Contribution (%) |
|---|---|---|
| | | |

We expect equal contributions to the assignment from both team members for the code and report parts. To ensure fairness, a team member, whose final contribution is below 40% obtains a reduced grade for the assignment.

## Academic Integrity

This project can be done in a team of up to two people.

The standing policy of the Department is that all students involved in any academic integrity violation (e.g., plagiarism in any way, shape, or form) will receive an F grade for the course. The catalog describes plagiarism as “Copying or receiving material from any source and submitting that material as one’s own, without acknowledging and citing the particular debts to the source, or in any other manner representing the work of another as one’s own.”. Refer to the Office of Academic Integrity for more details.

## Late Days Policy

You can use up to three late days throughout the course that can be applied to any assignment related due dates. You do not have to inform the instructor, as the late submission will be tracked in UBLearns.

If you work in teams, the late days used will be subtracted from both partners. In other words, you have three late days, and your partner has two late days left. If you submit one day after the due date, you will have two days and your partner will have one late day left.

## Important Dates

**June 10, Monday** - Register your team (UBLearns > Tools > Groups)

**June 13, Thursday** - Checkpoint is Due

**June 20, Thursday** - Final Submission is Due