COL341 Spring 2023
Assignment 1: Linear Regression
(To be done Individually)

Due Date: 12th February 2023, Sunday, 11:55 PM (No extensions)

1 Introduction

1.1 Linear Regression

Regression is a functional relationship between two or more correlated variables. Linear regression is an algorithm used to find such a function when it is linear. For example:

$$y = c + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + ... \tag{1}$$

It is a supervised learning algorithm because the function parameters are learned by using available data. In this assignment, you will implement the linear regression algorithm on a given data and perform analysis.

1.2 Problem Background and Motivation

The goal is to perform an evaluation of Microsuturing which is a surgical technique of stitching under a microscope. Currently, senior doctors evaluate Microsuturing performed by trainees by images and give them a Likert scale score between 1-9. Figure 1 shows examples of such images with the one on the right being evaluated as higher scoring by a senior doctor. To aid in the manual process, requiring precious time of a senior neurosurgeon who would otherwise be performing surgeries and saving many lives, we would like to predict the score from the image using machine learning techniques.

1.3 Features

Since the course has not exposed you to computer vision techniques, we will do that part for you. You will be given extracted features from the image in the form of a 2048 dimensional vector. The features have been extracted using a separate algorithm, the details of which are not relevant to perform the task.

2 Dataset

The dataset can be downloaded from this link. The dataset contains the following three files - `train, validation` and `test` in `csv` format. In `train` and `validation` files, each row contains one sample. The first column contains the sample name; the second column contains the output score between 1-9; columns 3 to 2050 contain the scalars corresponding to 2048 dimensional feature vector. The `test` file does not contain the output score. You need to train your model using `train`

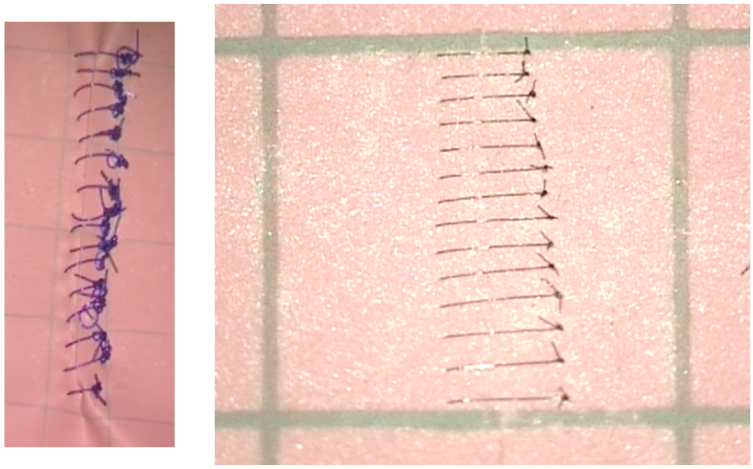

Figure 1: Two examples of images of Microsuturing

data and assess the fitness of your model using the `validation` data. You are expected to perform the following tasks.

3 Tasks

3.1 Basic Implementation

You need to implement the gradient descent algorithm on mean squared error (MSE) loss to find linear regression parameters. Experiment with learning rates, [0.1, 0.01, 0.001]. You require to specify an appropriate halting mechanism (stopping criteria) to stop the iterations of gradient descent. We would like you to experiment with the following stopping criteria:

1. `maxit`: stop when the iterations reach a predefined maximum number of iterations
2. `reltol`: stop when the relative decrease in the cost function (loss) on `validation` data drops below a threshold. You can experimentally determine a threshold.

You need to show and analyze the results w.r.t. all three learning rates and both stopping criteria on the on `train` and `validation` data. You must report your model's MSE and MAE scores on training and validation data. You would also need to store the value of the training and validation MSE loss obtained in the training iterations for the visualization part.

For the `test` data, you are free to choose other learning rates and a stopping criterion. A typical strategy is to check which parameters give the best results on the validation data and fix that set of parameters for the testing cohort.

3.2 Ridge Regression

As the dataset has a very huge number of parameters, many of the parameters may be correlated with each other. A method of regression that handles this is Ridge regression. This method is similar to linear regression, except the equation for loss to be minimized is as follows:

$$\sum_{i=1}^{m}(y_i - \sum_{j=1}^{n}\theta_j x_{ij})^2 + \lambda \sum_{j=1}^{n}\theta_j^2 \tag{2}$$

Based on this, find the new equation for the cost function and parameter update function and mention them in the report. Read more about the bias-variance trade-off and report MSE loss, MAE loss, and your observations by implementing ridge regression for the values of λ as `5` and `25`. Also, compare the results with linear regression and report your observations.

3.3 Using Scikit-Learn Library

In this part, you need to use the scikit-learn library's classifier, train it using `train` data and report the MSE and MAE on `validation` data. Compare and analyze the performance of your models in sections `3.1` and `3.2` with the scikit-learn model. Reference can be found here.

3.4 Feature Selection

As you can observe, there are 2048 dimensional feature vectors in this dataset. It is difficult to keep track of such large number of dimensions and analyze how each input dimension/feature affect the output score. We would like to find out whether it is possible to predict the value of the score accurately by using a lesser number of features/dimensions. There are various methods of feature selection that can be used for this. Find about their implementation available using scikit-learn here. Use the methods `SelectKBest` to find the 10 best features. Train your linear regression model using these 10 features and compare the MSE and MAE w.r.t. the model trained with all features. Repeat the same with `SelectFromModel` method of feature extraction. Use the `ridge` estimator. You may fix the learning rate to be 0.1 for this experiment, and use the `reltol` stopping with appropriate threshold determined in 3.1.

3.5 Classification

On observing the dataset, it can be inferred that the values of scores aren't continuous, but they are discrete and there are a finite number (9) of values that score can take. Even if a model predicts a value that is pretty close to the correct score say - `1.49` for a sample with a score of `2`, rounding it off will lead to an incorrect value. Logistic regression can be used as a binary classifier. The cost function for logistic regression is as follows:

$$\mathrm{J}(\theta) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x)) \tag{3}$$

Having multiple classes here, we will try to find 8 classifiers, such that $h_\theta(x)$ gives the probability of the sample being in one of the classes each for scores 1 to 8. And 1 - (sum of these probabilities) is the probability of the sample being in class 9. For that, we define the probability of the score being equal to one of the classes for each r in 1 to 8 as:

$$h_{\theta_r}(x) = e^{(\theta_r)^T X} / (1 + \sum_{p=1}^{8} e^{(\theta_p)^T X}) \tag{4}$$

Find the equation for the update function and mention it in the report. And perform logistic regression to find 8 such sets of θ parameters and give a predicted score as the score of the class having the highest probability.

3.6 Visualization

You need to plot the training and validation MSE loss (in the same graph) over the iterations (x-axis: iterations, y-axis: loss). Find an estimate of the number of iterations required by your algorithm to converge; divide that by 20 and then store the values of loss after each such interval

to get roughly 20 points, and mention your observations in the report. You need to perform this for sections `3.1`, `3.2`, `3.4` and `3.5`. Compare them and report your observations.
Normalize the data by the following equation:

$$x_i = \frac{x_i - \mu_i}{\sigma_i} \tag{5}$$

where μ_i and σ_i are the mean and variance values of that feature respectively. Perform section `3.1` using normalized data and plot the training and validation MSE loss over the iterations.
Repeat section `3.1` using one-fourth, half, and three-fourths of the training data. Plot four graphs of training and validation MSE loss over the iterations for four sizes of data - $[\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1]$ and report your observations. You can use scikit-learn library for this.
Divide the training data into two equal parts. Train the model on each half of the data and get predictions. Find the mean absolute difference between the predictions for both sets of training data. Perform this by training using the method of section `3.1` and then again using the method of section `3.2`. Report your observations.
For section `3.1`, plot a graph of total MSE loss against the correct score values - i.e. calculate the sum of losses of all the samples having score 1 and plot it against x-coordinate 1 and respectively for other classes. Similarly, in the same graph, also plot the average of MSE loss against score values and report your observations.
Using the feature selection method that gave better accuracy, repeat feature selection for 100 and 1000 best features. Train your model again for them and plot a graph of MSE loss over the number of features used in the range - [10, 100, 1000, 2048]

3.7 Generalization Analysis

Generalization is the machine learning model's capacity to perform well on unseen data after being trained on test data. For empirical analysis, we would take MSE on `train` data as E_{in} - the error on seen data and we would take MSE on `test` data to be E_{out} - the error on unseen data. Theoretically, the generalization bound on Linear Regression is found to be as follows:

$$E_{out} = E_{in} + \mathcal{O}(\sqrt{\frac{d}{N}}) \tag{6}$$

where d is the number of dimensions of the data and N is the number of samples of the data and E_{out} is the expectation of error on unseen data. Here you will find `test` and `train` files for four datasets of different dimensions. In each file, the last column contains the correct labels and the rest contain input features of the data. For each dimension, train your model using the `train` file and calculate MSE to find E_{in} and E_{out} using predictions on `train` and `test` files respectively. And plot the value of $E_{out} - E_{out}$ against the dimensions - [2, 5, 10, 100]. Report your observations.

3.8 Bonus

This is **not** a mandatory part of the assignment. Another approach that can use multiple binary classification boundaries to separate the data amongst all the possible score values is One-vs-Rest (also known as One-vs-All) method. In this method, nine boundaries are to be found using logistic regression. For each score value, a boundary is found by labeling samples of that class as `1` and the rest as `0`. After finding nine such lines, each sample is given a score of the line at the least distance. You can read more about this concept here.

4 Evaluation

The efficacy of your solution will be judged by running your implemented algorithm. You need to create `main.py` file to train your model and generate the labels. You may create additional source files that will be used by the `main.py` to compute the output scores. The usage of the `main.py` should be the following to generate the output scores on `test` data:

```
python main.py  --train_path=<path to train file> \
                --val_path=<path to validation file> \
                --test_path=<path to test file> \
                --out_path=<path to generated output scores> \
                --section=<1 or 2 or 5>
```

The `section` argument is used for determining which method is used for regression 1 - for linear regression, 2 - for ridge regression and 5 - for classification. The output format is the following for test data:

```
<sample name 1>, <output score 1>
.
.
.
<sample name N>, <output score N>
```

Create a csv file to store the output labels according to the given format. We will evaluate the performance of your test data by running an automated script. So please make sure you follow the name of the files, arguments, etc., as per the given format.

5 Submission

You are required to submit your complete source code (including the completed `main.py`) and a report containing the following information:

1. Detailed analysis regarding the performance of your model on `train` and `validation` data for the different experimental setups. Include all numbers and plots.
2. Your observation on the Visualization part.

Submit a `README.txt` for your code, if necessary with all instructions for generating the required output. You must zip the code file, and any other files that may have been created to be used by main.py and report in a single zip file, rename the zip file as `<Your-Entry-Number>.zip` (e.g., `2019CSZ8406.zip`), and submit in Moodle.

6 Rubric

1. Full marks - 15 (can be scaled later)
   (a) Basic implementation – 2 mark
   (b) Ridge regression – 1.5 mark
   (c) Comparison with Scikit model – 1 mark
   (d) Feature selection – 1 mark
   (e) Classification – 3 mark

(f) Generalization analysis and visualization – 2.5 marks

(g) Report containing other analysis and observations from experiments – 1 marks

(h) Performance of 3.1, 3.2, and 3.5 on held-out test data – 3 Marks

2. Bonus: – 1 Mark

7 Constraints

1. You need to do this assignment individually.
2. You need to use `Python 3.6+` for implementation. You are allowed to use basic standard libraries like `math`, `numpy`, `pandas`, etc. You are allowed to use `sklearn` only for sections - `3.3, 3.4` and `3.6`. No other available linear regression models are allowed for the implementation in the first two sections - they should be your own algorithms. No other third-party library is allowed. Please clarify any further doubts using piazza.
3. Any plagiarism will attract penalties as described in the course policy.
4. There are no late submissions or extensions allowed. Medical/ emergency cases will be dealt with on a case-to-case basis.