# Implementing Robust Linear Regression

Aishee Bhattacharya Bipayan Banerjee
Rupayan Chowdhury Rahul Konar

December 2023

## 1 Introduction

This report aims to implement the robust linear regression algorithm proposed in the paper titled "Robust Linear Regression: Gradient-descent, Early-stopping, and Beyond". This report describes the step-by-step implementation process of the algorithm in python programming language.

## 2 Background

Linear Regression is a very common algorithm used in Machine Learning. It involves modelling the linear relationship between two variables in a given dataset. Traditional linear regression techniques are sensitive to outliers, leading to biased parameter estimates and reduced predictive accuracy. This paper proposes an innovative way to consider linear regression under adversarial test-time attacks w.r.t arbitrary norms, although we have implemented it for $l_2$ and $l_\infty$ norms.

## 3 Program Implementation

### Inputs

The user is expected to provide the following inputs:

- The file path of the Excel file(.xlsx). Note we expect the output values to be stored under a column named 'y'.
- The attack budget of the adversarial norm attack

### Tools we have created:

Our program consists of a main file along with three other files where important functions pertaining to our algorithm are defined. We have defined functions which perform the following tasks:

- **read_excel_dataset**: To read the Excel file.
- **proj_l2_ball**: To return a unit vector along an input vector $v$, if $||v|| > 1$, or the vector itself otherwise.
- **prox_dual_norm**: Returns the maximum out of 0 and $v - v\frac{r}{||v||}$, where $r$ is attack budget. We have only implimented it for $l_2$ norm attack, in case of any general norm attack, this should be replaced by the corresponding proximal operator of the norm.
- **chambolle_pock_algorithm**:This function implements the Chambolle-Pock algorithm, a method used for solving convex optimization problems.This algorithm iteratively updates $z$, $w$, and $u$ using specific operations (projection and proximal operator) to converge towards a solution that minimizes the objective function. The specific operations **proj_l2_ball** and **prox_dual_norm** are crucial parts of the algorithm, handling the projection onto the $l_2$ ball and the proximal operator for a given norm, respectively. These operations help in enforcing constraints and handling the optimization problem being solved.
- **almost_diagonal**: Checks whether a matrix is almost diagonal (i.e, the absolute value of all entries that are not diagonal entries are less than a certain cut-off) or not.
- **max_product_of_elements**: Basically computes the maximum of the product of two entries of two given lists, that are at the same position.
- **soft_threshold**: This is the soft-threshold operator, which can be used at any given threshold level. In essence, this function applies a shrinkage operation to each element of the input array x, reducing its magnitude towards zero based on the specified threshold.
- **l_infty_adverserial_risk**: This computes the corresponding adverserial risk for $l_\infty$ norm attacks.
- **min_adverserial_risk**: This function minimizes the adversarial risk. It performs a grid search over a range of values to find the best $w_t$ that minimizes the adversarial risk. The code uses a grid search approach to explore various values of t, calculates corresponding $w_t$, computes the adversarial risk associated with each $w_t$, and keeps track of the best $w_t$ that minimizes this risk across the grid values.

## Main Implementation:

The step by step implementation of the algorithm is presented here:

1. Read the data-set and seperate the input matrix and output values.

2. Compute the sample covariance, store it as sigma_hat. Also compute w0_hat using the method of Ordinary Least squares (we have used Linear regression model to compute it). Note, these two operators are consistent in our given-setup.

Now for $l_2$ norm attacks:

1. Calculate the operator norm of sigma_hat and choose eta_1, eta_2 such that eta_1 $*$ eta_2 $*$ sigma_hat $< 1$. (Here we have chosen eta_1 $= 0.3$ and the product also $= 0.3$, hence $< 1$.)

2. Apply the **chambolle_pock_algorithm** function on the previously calculated w0_hat, square root of sigma_hat (we have used the cholesky square-root), eta_1, eta_2 (these are the respective step-sizes), and r which is the attack budget taken from user,

For $l_\infty$ attacks, we need to:

1. Check whether the estimated co-variance matrix, sigma_hat is almost diagonal.

2. If so, compute the product of the absolute value the jth component of w0_hat vector and the (j,j) entry of sigma_hat matrix for all j starting from 1 till the length of w0_hat vector. Store the maximum of the product as c.

3. Calculate the w0_hat using the **min_adverserial_risk** function which runs in the range of 0 to c, taking inputs w0_hat, the diagonal entries of the estimated co-variance matrix and the attack budget, r.

### Outputs

We have applied our algorithm on a data consisting of 500 sample points, with 2-dimensional input vectors. The code used for generating the xlsx file is also submitted. The output vectors are a linear combination of the input vector with an added random error. Our algorithm gives the desired output upto a level of $10^{-3}$ accuracy for $l_2$ attack norm. The accuracy can be increased by increasing the number of iterations for $l_2$ attack norms.

## 4 Conclusion

Our implementation has showcased the effectiveness of this algorithm for attacks on $l_2$, $l_\infty$ norms. Further improvements could be made on generalizing the proximal operator for any given norm.

# 5 Acknowledgements

We would like to express our sincere gratitude to Prof.Malay Bhattacharyya and Prof.Arnab Chakraborty for their invaluable guidance and support throughout the development of this project. Their expertise and insightful feedback played a crucial role in implementing the Robust Linear Regression algorithm. Additionally, we extend our thanks to our team members for their collaborative efforts, dedication, and perseverance. This project would not have been possible without the collective contributions of everyone involved. We are also grateful to our parents , our friends and to the entire academic community of ISI Kolkata for completion of the project,without them it wouldn't have been possible.

# 6 Bibliography

We have used the following libraries:

1. **pandas**: To read the Excel file
2. **numpy**: To compute norm of a vector, perform operations on matrices, etc.
3. **scikit-learn**: To find regression coefficient