# README : Searching for the Boson

Ducommun Colin, Elalamy Rayan, Van de Velde Anne-Sophie

## 1 Prerequisites

The code is written in Python3, and makes extensive use of Numpy, the fundamental package for scientific computing in Python. Another external module used is csv - it intervenes in importing ghe flow of data to and from the Python environment, in order to read the data from a .csv file and to write the predictions in a .csv file respectively. Therefore the user must have both packages to run this code. For installation information, please refer to the Python documentation.

## 2 Executing the code

Extract the .zip and open the terminal at the location of the extracted files. Then enter: python3 run.py . The file run.py contains an executable Python script that takes train data to compute a model. To see more details about the objective of this scrpit, one should read our report. The desired train and tests sets are to be put in a train.csv and test.csv , which are provided as placeholders.

## 3 Description of the code

The script first loads the data and initializes the hyperparameters.

First missing values are treated by deleting the columns where they occur. Then, a polynomial extension up to the value of ”degree” follows, this method also add a column of 1's (instead of performing a standardization of the data). Finally, pairwise products of some of the features are performed. After this treatment, data is put in an appropriate format for the learning process.

The initial weights $w$ are set to zero arrays. The training is done using a ridge regression. The method outputs the optimal weights $w$ and the loss that corresponds to it. The loss is here the ratio of good predictions.

Then, the test data goes through the treatment described above. The predictions are performed then converted to the appropriate format $\{-1, 1\}$ according to the best threshold (found via Kaggle). The predictions are then put out using the csv module. They can be found in the same folder as the run.py script.

## 4 implementations.py

This file contains the methods that were to be implemented :

i) least_squares_GD(y, tx, initial_w , max_iters, gamma) : performs a linear regression using a gradient descent algorithm.

ii) least_squares_SGD(y, tx, initial_w , batch_size , max_iters, gamma) : performs a linear regression using a stochastic gradient descent algorithm.

iii) least_squares (y, tx) : performs a least squares regression using normal equations.

iv) ridge_regression (y, tx, lambda_) : performs a ridge regression using normal equations.

v) logistic_regression (y, tx, initial_w , max_iters, gamma) : performs a logistic regression using gradient descent.

vi) reg_logistic_regression (y, tx, initial_w , max_iters, gamma, lambda_) : performs a regularized logistic regression using a gradient descent.

The inputs, when present, are :

- y which is a vector of flags (here it is an array of $-1$'s and 1's);
- tx which is the matrix of features (in an order corresponding to y );
- initial_w which is the starting weight for the iterative methods;
- max_iters which sets the maximum number of iterations for the iterative methods, set to 6000 (but it should converge after 5000 iterations);
- gamma which is the step size for the iterative methods, set to $10^{-6}$ according to the length of the data set;
- lambda_ which is the factor of penalization in the ridge regression;

All of these methods return the predicted weights $w$ and the loss, which is the ration of predictions (see report). They are to be used cautiously and basic knowledge in machine learning is assumed.

## 5 utilities.py

This file contains auxiliary methods that are used in the methods from 4. One can find here the methods used to standardize, to calculate the gradient, to build polynomials from a matrix $z$. It contains also the methods used to manipulate the data, i.e. the one performing cross-validation, the one adding new features by multiplying the old ones and the one separating the categorical features (one-hot). For more details about why we implemented them, one should read our report.

One can also find a method definitive_res_logistic (x, threshold) that returns the decision $-1$ or 1 from our matrix $z$ if the regression ligne is under or above a certain threshold.

## 6 proj1_helpers.py

This file contains auxiliary routines that would otherwise make the script less readable. For most, they are of no interest by themselves and perform only data treatment like reading/writing from/in a csv file.