# CS-433 Machine Learning Project 1 Report

Weiran Wang, Yuan Cheng, Yang Gao
*School of Computer and Communication Sciences, EPFL, Switzerland*

## I. Introduction

In this project, we mainly do exploratory data analysis on a specific dataset by implementing 6 different machine learning methods. Our dataset comes from a popular machine learning challenge named "finding the Higgs boson" provided by CERN mentioned above. The 6 machine learning methods are *least_squares_GD* [1], *least_squares_SGD* [2], *least_squares* [3], *ridge_regression* [4], *logistic_regression* [5] and *reg_logistic_regression* [6] respectively. Specifically speaking, we train our 6 models based on the training set and get the optimal $w$ generated by the training set. Later on, we test our model, i.e. the $w$ matrix, on the testing set and output the prediction value in 6 separate csv files. To get more persuasive results, we test our prediction on AICrowd [7] to get an overall accuracy. Furthermore, we also test our model locally by using the K-fold cross-validation function [8]. Finally, we also visualize the classification results by using TSNE [9].

## II. Models and Methods

### A. least_squares_GD

A gradient (at a point) is the slope of the tangent to the function (at that point). It points to the direction of the largest increase of the function. To achieve gradient descent for linear regression, we first define the error vector as

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w} \tag{1}$$

where $y$ is the real-valued and $Xw$ is our model. Then we calculate the mean squared error (MSE)

$$L(\mathbf{w}) := \frac{1}{2N}\sum_{n=1}^{N}\left[y_n - \mathbf{x}_n^\top \mathbf{w}\right]^2 = \frac{1}{2N}\mathbf{e}^\top \mathbf{e} \tag{2}$$

as the cost function to quantify how well our model does. With the MSE, we can get the gradient to minimize the cost function

$$\bigtriangledown L(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top \mathbf{e}. \tag{3}$$

In the method *least_squares_GD*, we set the initial weight vector $\mathbf{w}^0$ **= 0**, the maximum number of iterations *max_iters*=**5000**, and the step-size (or learning rate) $\gamma$=**0.000001**. To minimize the loss function $L(w)$, we iteratively take a step in the (opposite) direction of the gradient

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \gamma \bigtriangledown L(\mathbf{w}^{(t)}) \tag{4}$$

and finally get the optimal wight and the corresponding value of loss when $w$ converges.

### B. least_squares_SGD

It is similar to *least_squares_GD* but has a much cheaper computational cost $O(D)$ compared to $O(ND)$ in *least_squares_GD*. Even though, it can achieve unbiased estimate of the gradient. The core idea of SGD method is sampling one datapoint $n \in \{1, 2, ..., N\}$ uniformly at random and doing the iteration with stochastic gradient for the only datapoint each time. The iteration is as follow

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \gamma \bigtriangledown L_n(\mathbf{w}^{(t)}) \tag{5}$$

where $\bigtriangledown L_n(\mathbf{w}^{(t)})$ is the stochastic gradient of datapoint $n$.

### C. least_squares

This method starts from defining the Gram matrix $\mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{D\times D}$. When it is invertible, we can calculate the closed expression for the minimum by multiplying the normal equation with the inverse Gram matrix.

$$\mathbf{w}^\star = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y} \tag{6}$$

With this model, we can predict for the unseen data $x_m$ with a new $y_m$. Equation (7) shows the concrete prediction.

$$\hat{y}_m := \mathbf{x}_m^\top \mathbf{w}^\star = \mathbf{x}_m^\top \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y} \tag{7}$$

### D. ridge_regression

This method performs L2 regression. If a linear regression model has highly correlated independent variables, the precision will be highly affected. Ridge regression is a possible way to solve the multicollinear problem occurred in *least_square* which gives more accurate ridge parameter by introducing a ridge regression estimator

$$\min_{\mathbf{w}} \frac{1}{2N}\sum_{n=1}^{N}\left[y_n - \mathbf{x}_n^\top \mathbf{w}\right]^2 + \lambda\|\mathbf{w}\|_2^2. \tag{8}$$

The explicit solution is below where it's worth mentioning that $\lambda' = 2N\lambda$ and $I$ stands for the identity matrix. In the code, we set $\lambda$ as **np.logspace(-5, 0, 30)**.

$$\mathbf{w}^\star_{\text{ridge}} = \left(\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y} \tag{9}$$

### E. logistic_regression

Different from the gradient descent used in linear regression, a new implementation of gradient descent is realized in logistic regression. First we apply sigmoid

$$\sigma(x) := (1+e^{-x})^{-1}. \tag{10}$$

as the base for logistic function. Second, the negative log-likelihood loss is used for the binary classification problem (a.k.a. cross entropy loss) to compute the loss:

$$L = -\sum_{i=1}^{n} y_i log\sigma(x_i^T w) + (1-y_i) log(1-\sigma x_i^T w). \tag{11}$$

Third, we compute the gradient to minimise the loss $L$:

$$\bigtriangledown L(w) = \sum_{i=1}^{n} (\sigma(x_i^T w) - y_i) x_i. \tag{12}$$

In reality, the code is written under the matrix form, i.e.,

$$\bigtriangledown L(w) = X^T(\sigma(Xw) - y). \tag{13}$$

In a nutshell, it is the same gradient as in linear regression but with sigmoid function $\sigma$ and the cost function $\mathscr{L}$ is convex. The rest calculation step is like what we do in *least_squares_GD* equation (4) within a given max iteration.

### F. reg_logistic_regression

Similar in *ridge_regression*, a regularization term $\lambda\|w\|^2$ is added to trade off weight $w$. Specifically speaking, $\lambda\|w\|^2$ is added to loss $L$ function where the value is the same as *ridge_regression* when realized in the code.

$$L(w) = \sum_{i=1}^{n} (\sigma(x_i^T w) - y_i) x_i + \lambda\|w\|^2 \tag{14}$$

and $2\lambda w$ is added to gradient function:

$$\bigtriangledown L(w) = X^T(\sigma(Xw) - y) + 2\lambda w. \tag{15}$$

The rest steps are the same with the *logistic_regression* by converge to the optimal $w$.

## III. Discussion

As an extension, we visualize the training set and the six testing sets results by using TSNE [9] recommended by one of the TAs. In general, TSNE is a statistical method for visualizing high-dimensional data. We set 'n_components=2' to reduce dimensionality from 30 to 2 as $w$ is a 30 dimensional vector where the horizontal and vertical coordinates represent the probability distribution over pairs of dimensional map. Compared to the training set, we get a relatively good results where 'signal' and 'background' are clustered together and resemble the training set. It is worth mentioning that considering the computational load, we only choose the first 10,000 data to visualize.

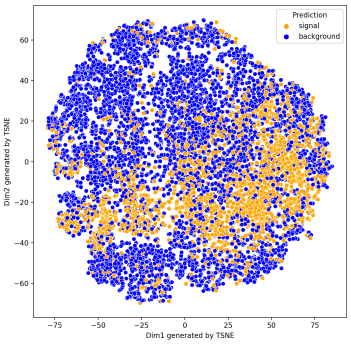


Fig. 1. training set visualization

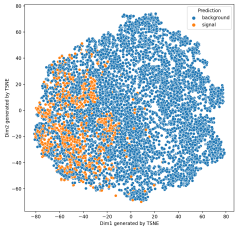

Fig. 2. least-square-GD

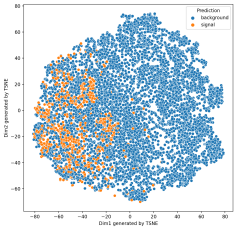

Fig. 3. least-square-SGD

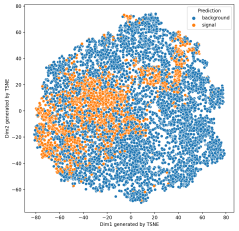

Fig. 4. least square

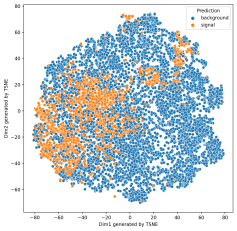

Fig. 5. ridge regression

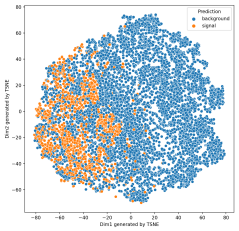

Fig. 6. logistic regression

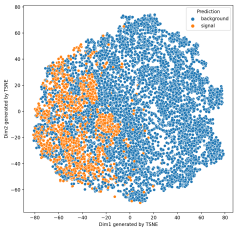

Fig. 7. reg logistic regression

## IV. Conclusion

In this project, we apply the six methods above to predict the labels for the testing set and get the accuracy of those methods after uploading the .csv files containing the obtained labels to the website of AIcrowd. Among all those methods, the method *least_squares* and *ridge_regression* achieve an accuracy higher than 0.735. Besides, we manage to visualize our result by making use of TSNE so as to better see the difference in the predictions of these methods.

## REFERENCES


[1] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[2] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[3] C. L. Lawson and R. J. Hanson, *Solving least squares problems*. SIAM, 1995.

[4] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[5] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[6] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, "Distributed newton methods for regularized logistic regression," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2015, pp. 690–703.

[7] M. Kobayashi, K. Wakabayashi, and A. Morishima, "Quality-aware dynamic task assignment in human+ ai crowd," in *Companion Proceedings of the Web Conference 2020*, 2020, pp. 118–119.

[8] Y. Bengio and Y. Grandvalet, "No unbiased estimator of the variance of k-fold cross-validation," *Journal of machine learning research*, vol. 5, no. Sep, pp. 1089–1105, 2004.

[9] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.