PCML Project 1

Max Premi, Martin Barry, Loris Aiulfi

***Abstract*—By reading this report, you must be familiar with some basic notions of Machine Learning. As this essay is done by student to show their understanding of a course, it is no scientific discovery about Higgs Boson. The problematic is to differentiate a boson from a particle, and we will se that ridge regression is the best method found with different set of training.**

I. Introduction

In March 2013, the CERN discovered the Higgs boson particle with the help of the LHC. Scientists smash two protons with high energy in order to reach elastic energy equal or higher than the intrisic mass of the Higgs boson and thus being able to product boson. The problem is that it cannot be observed, but it can be measured thanks to its decay signature. We will in this project analyze all the features of collisions, and classify the reaction into Boson or other particles thanks to Machine Learning (ML) Methods.

II. Models and Methods

A. Data Analysis

By looking at the given data, some particular properties can be observed.
First, there are values at -999 , meaning the measure could have not been recorded (these data are outliers but useful).
Then, there is the jet number column that only contains integers in $\{0, 1, 2, 3\}$.
For the given classification, boson is translated as -1 in import and other particules as 1, so the vector of given result y is a vector of 1 and -1.
The prediction will be done on 30 features, gathering mass detection, location of the detection... ending with b if boson , s otherwise. We will see in the next part how we delt with the particularities stated above.

B. Data Preprocessing

For -999, as this data seems useful, but too big to be taken into account, what can be done is replace them by the median value of the column (calculated without them).
-999 should not disappear since the fact of not having value is an information that is important, adding a column of feature which is the number of -999 in each line seems to be a good idea.

C. Gradient Descent with Least Squares

From here, the vectors w will be the weight predicted, x the features and y the predictions given.
As usual in ML, we want to minimize the cost function, so that the model fits the most with the Data.
To minimize the loss function we take a step in the opposite direction of the gradient, i.e. gradient descent method. The cost function is MSE:
$\frac{1}{N}\sum_{n=1}^{N}(y_n - x_n w)^2$ where N is the number of data samples.
The initial w is initialized to a vector of 0, and the gradient is $\nabla = \frac{-1}{N}x^T(y - xw)$
To get an even better result, polynomial basis is used, as the classification is surely not simply linear. After few tests, the basis 5 was one of the best for Gradient Descent.

To choose a good step, we use the backtracking line search implemented to follow the ArmijoGoldstein condition. It helps finding the maximum amount to move along a direction, so here the maximum amount to move along the direction of the gradient. It slows down the computation, but it'll bring the best convergence possible in a faster way as the two graphics show.

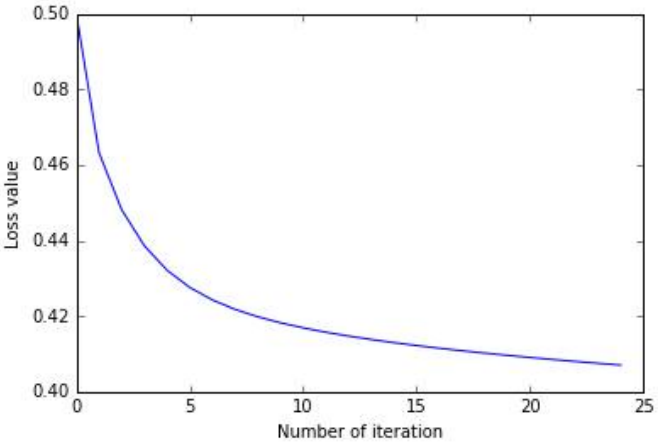

Fig. 1. Loss function in function of iterations with gamma = 0.1.

D. Stochastic Gradient Descent with Least Squares

Here the idea is the same as classic Gradient Descent except we compute the gradient of small batch of data, instead of computing over all examples. This is much more faster than classical gradient descent so we can allow ourselves to put a small step size and a lot of iterations. Even though this method is useful to obtain quick results, it is more appropriate for online learning than for this report where we try to obtain the higher precision.

E. Least Squares

Least squares use normal equation to solve the w vectors, it resolves sets of equations in which there are more equations than unknowns.
i.e. we had to minimize the equation $||Aw-b||_2$ with respect to

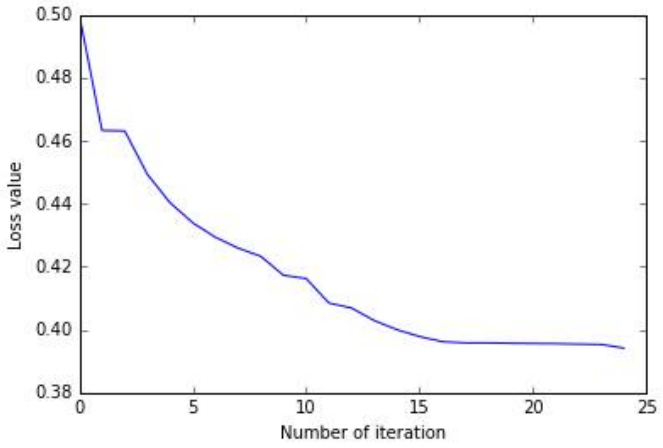

Fig. 2. Loss function in function of iterations with backtracking.

w. To do so and using convex optimization it suffices to set the gradient to 0. Which leads to solve linear system of equations $Aw = b$, note that the matrix A might be ill-conditioned and non-square, thus we used algorithms in the numpy library to solve it.

F. Ridge Regression

Ridge regression is the solution to Least Squares if the number of equations is not superior to the number of unknowns, or as it is said, when the problem is said to be not well posed. We introduce a regularizer that improves the conditioning of the problem, thus enabling a direct numerical solution. To find the best regularizer λ, we test the function with different value with grid search and take the values that minimize the cost.
What is a little different here is that we separate data into 4 sets , depending on the value of the jet column.
Then with grid search we find the best lambda and degree for each set, which are considered independent from each others and we find:
$\min_w \frac{1}{2N} \sum_{n=1}^{N} [y_n - \tilde{\phi}(x_n)^T w]^2 + \lambda \mid\mid w \mid\mid_2^2$ with $\tilde{\phi}$ the polynomial basis.

The lambda avoid the overfitting of the predicted weight, else the prediction will fit perfectly our data with training but not the future data to test. It will be commented in the code as it takes a long time so the values will be directly assigned.

G. Logistic Regression

Logistic regression is used for binary classification. Here the y are either -1 or 1, we just transform the -1 in 0 (for the log function) and apply the logistic function which is $\sigma(x) = \frac{1}{1+e^{-x}}$
Thanks to the maximum likelihood criteria, we can find an optimal solution easily using similar gradient descent as for the first method.
The cost function is :
$\sum_{n=1}^{N} ln[1 + exp(x_n^T w)] - y_n x_n^T w$
Then gradient descent is used as the first function, and we still use the backtracking algorithm.
This gives a really slow algorithm, but which is supposed to perform better as it's a binary classification.

H. Regularized Logistic Regression

The goal is the same as the Ridge Regression, we add a λ that helps reduce overfitting.
The formula is as follow:
$\min_w - \sum_{n=1}^{N} lnp(y_n \mid x_n^T, w) + \frac{\lambda}{2} \mid\mid w \mid\mid^2$
Where $\min_w$ is the previously found weight. We do the same research over the λ with a classical grid search over the lambdas.

I. Best Method

After computing all the methods that were asked, the goal of this project was to reach the best results we could. First, one realized that standardize the features was not improving the results and thus we did not use this. Then one split the data into 4 different sets each defined by there jet number $\{0, 1, 2, 3\}$, in fact this number will define in which part of the detector the particle will be detected and thus it makes much sense to train theses data separately. to complete this research of accuracy one also added to the polynomial features the cross terms $x_i x_j$ $i \neq j$, the absolute values of the square root of x and of the cross terms. After computing the polynomial dependencies we used as explained previously a grid search on lambda and the degree to obtain the lowest losses for each of the data and then used their respective weights on the given prediction data. So the best result are given with ridge_regression.

III. Result And Conclusion

With several submit to Kaggle and comparison between losses, we will discuss here the results.
First let's talk about Gradient Descent (GD) and Stochastic Gradient Descent (SGD), it gives quite good result, but not if the minimum of a function is close to lot of local minimum, then it'll not efficiently converge. It seems this is the case, as the prediction done on Kaggle are not quite good, we get a loss of 0.449 for GD in 10 iterations and 0.443 for SGD with 999. There is the log regression, which are supposed to be good with binary classification, but here it's not quite the case, maybe our implementation is not the best either.
Then Least Square, was the first good result on Kaggle, as the dependent features corresponding to such future application would be subject to the same types of error as those in the training data, still get 0.824 in rmse.
Then Ridge Regression, the least-square with parameter to avoid overfitting and in addition, we treated each set with different valeus of jet independently which increase our prediction. The RMSE are 0.665 , 0.739 , 0.678 and 0.740 on the 4 set separated in function of the jet number.
And it's the best result we get at the end