Project 1 - finding the Higgs Boson

Léa Bommottet, Nicolas Brunner, Karine Perrard
CS-433 Machine Learning, EPFL, Switzerland

***Abstract*—Regression is one of the foundation element of Machine Learning, that can yields to way more complex models such as Neural networks. Applying regression to a known dataset and output gives a parameter that can be later applied on dataset to predict an output. There are several models that can be used to achieve regression, and some of them fit better for some data.**

I. Introduction

In this project we have to solve a binary classification problem. In function of a vector of features regarding a proton collision event, we have to determine if this collision produced an Higgs boson or another process, there are at most 30 parameters provided for each classification. We have to implement and use machine learning methods, then feature processing and engineering to solve this problem.

II. Models and methods

A. Choice of Models

First of all, we have to select the core of our model. In this project, there is 6 possible models on which we can base.

1) Linear regression using gradient descent: This model try to minimize the error of the function, by moving the vector w in the reverse trajectory of the gradient.

$$w^{(t+1)} = w^{(t)} - \gamma \nabla L(w^{(t)})$$

$$e = y - Xw$$

$$\nabla L(w) = -\frac{1}{N} X^t e$$

2) Linear regression using stochastic gradient descent: The error function is defined as a sum over the training errors:

$$L(w) = \sum_{n=1}^{N} L_n(w)$$

As for gradient descent, we want to follow the gradient, but this time the updating step is done on the cost contributed by one of the training examples. We can also compute the gradient on a subset of the examples, this is mini-batch stochastic gradient descent.

$$w^{(t+1)} = w^{(t)} - \gamma \nabla L_n(w^{(t)})$$

3) Least squares regression using normal equations: We try to solve: $\nabla L(w^*) = 0$

Then we derive

$$w^* = (X^\top X)^{-1} X^\top y$$

and a unknown data-point would have

$$\hat{y}_m := x_m^\top w^*$$

4) Ridge regression using normal equations: We want to punish complex models and conversely choose simpler ones. In the case of ridge regression, we add a regularizer $\Omega(w) = \lambda ||w||_2^2$ in the quest to minimize w:

$$\min_w L(w) + \Omega(w)$$

the explicit solution of w become:

$$w^*_{ridge} = (X^\top X + 2N\lambda I)^{-1} X^\top y$$

5) Logistic regression using gradient descent or SGD: The problem is about separating the outputs into different labels. To do so we use the logistic function, that gives values in the range [0, 1]:

$$\sigma(z) := \frac{e^z}{1 + e^z}$$

Again we want to minimize the cost function defined as

$$L(w) = \sum_{n=1}^{N} ln[1 + e^{x_n^T w}] - y_n x_n^T w$$

To compute the model, we use gradient descent. The formula is

$$w^{(t+1)} = w^{(t)} - \gamma \nabla L(w^{(t)})$$

where

$$\nabla L(w) = X^T[\sigma(Xw) - y]$$

6) Regularized logistic regression using gradient descent or SGD: We can have troubles if the data is linearly separable. In that case, any vector **w** having the right direction is solution gives the minimum for **L(w)**. The goal of regularization is to add a penalty term, which is the length of the vector.

$$w* = argmin_w - \sum_{n=1}^{N} lnp(y_n|x_n^T, w) + \frac{\lambda}{2}||w||^2$$

B. Choice of parameters

We use cross-validation by K-fold to compute the most efficient lambda and gamma; the plot are below in Fig. 3. The parameters that reduce most the train error in our function are lambda = 10^{-5} and gamma = 10^{-5}.

C. Choice of features

We can improve our model by adding new features, which are function of other parameters. We want more features that have a Normal distribution. Furthermore, in function of the feature named "fit", some parameters are unavailable. Thanks to this important information, we don't train only one model, but 4 separate, since "fit" can be 0,1,2,3.

Natural logarithm
: Applying the logarithm on a parameter can highlight its Normal behavior. One example is this function on the feature 21 (Fig. 1). In this case, we notice a much nicer Gaussian distribution on the values, than the initial data. By doing an exhaustive research, we find that the useful parameters with this function are {0, 1, 2, 3, 4, 5, 8, 9, 10, 13, 16, 19, 21, 23, 26, 29}, when discarding the NaN value.

Square root
: Same as the natural logarithm, the square root can bring out a normal distribution. The feature 21 is also transformed with this function in our model (Fig. 1). Again if we don't consider the NaN value, the parameters {0, 13, 16, 21, 23, 26, 29} have a Normal behavior under the square root

Threshold
: Some features are mostly distributed along two peaks. In that case we move values to their closest peak to get two well defined peaks. This is the case for parameters {11, 12}.

Nothing max
: Some features have no defined distribution, but their values explode to big numbers. In that case we divide by maximum to get new value in range [0, 1]. Such distributed parameters are parameters {6, 14, 17, 24, 27}

Nothing norm
: Some features have no distribution and their values are already between 0 and 1. For parameter 7 we do nothing.

Distance
: When we plot 2 parameters in a graph, we can find some correlation between (Fig. 2). To reduces this property to a normal distribution, we compute the Manhattan distance, to flatten all the perpendicular result on the plot $y = -x$. The Manhattan distance is computed as follow:

$$dist(x_1, x_2) = |x_1 - x_2|$$

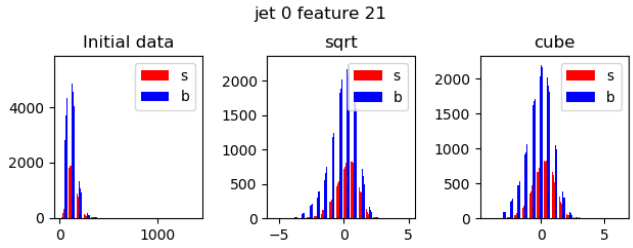

Figure 1. Function applied on the data of the feature 21

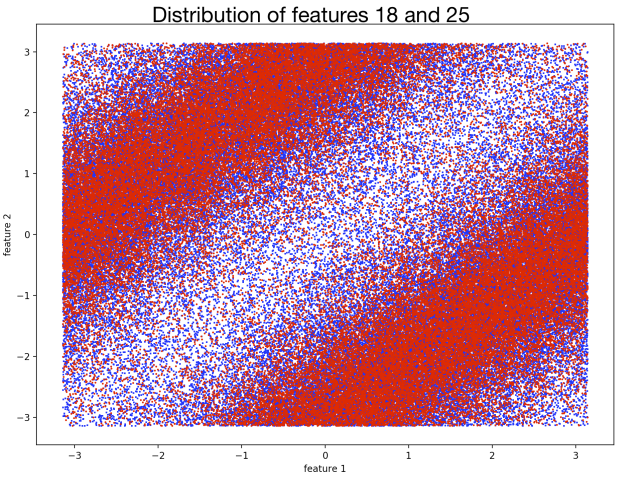

Figure 2. Repartition of features 18 and 25

N^{th} power
: Some feature sometimes have a better behavior when they are raised to a power greater than 2, as it is the case for feature 19 when raise to the cube (Fig. 4)

In addition with those feature processing, our model we select is logistic regression separated by the 4 jets and . The reason is that we have the least global error in comparison with the other models (Table I).

III. Results

Separating our model into 4 different ones was a great idea, reducing considerably our global error (Fig. I) as well as including more preprocessed features. Those added criterion improve the quality of our model, adjusting from

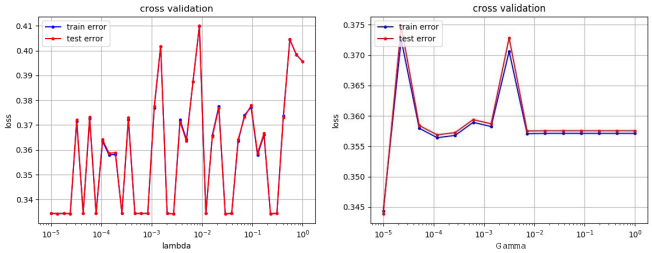

Figure 3. Cross validation for lambda and gamma

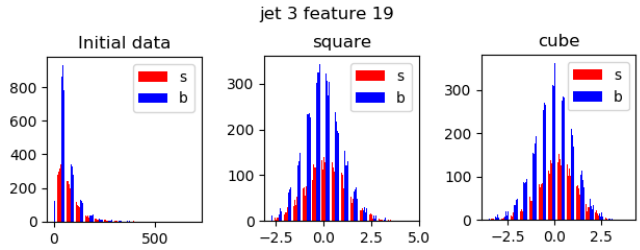

Figure 4. Function applied to feature 19

Table I
GLOBAL ERROR PERCENTAGE FOR EACH MODEL AND METHOD

Model	Linear regression using gradient descent	Linear regression using stochastic gradient descent	Least squares regression using normal equations	Ridge regression using normal equations	Logistic regression using gradient descent or SGD	Regularized logistic regression using gradient descent or SGD
1 jet (3000 iterations)	1.0	0.36664	0.26512	0.265312	0.3196	0.319616
1 jet with feature processing (3000 iteration)	0.325024	0.338288	0.240192	0.256256	0.22632	0.226352
4 jets (3000 iterations)	1.0	0.898050	0.248029	0.250557	0.383947	0.431784
4 jets with feature processing (3000 iterations)	0.302155	0.338330	0.224749	0.239293	0.195069	0.195053
4 jets with feature processing (30000 iterations)	0.255292	0.282860	0.224748	0.239292	0.189837	0.189933

our best global error to 0.18%. If we want to improve it more, we would need to find more useful parameters to boost our data.

IV. SUMMARY

The best prediction we could get to was obtained by using logistic regression with data cleaning, feature processing and engineering.

REFERENCES

[1] Machine Learning course and labs CS-433 *Martin Jaggi, Ruediger Urbanke* https://mlo.epfl.ch/page-146520.html