Project 1 Report - Team 61

Maria CERVERA DE LA ROSA (256385) - Hugo MOREAU (238523) - Josselin VALLEE (214573)
Pattern Classification and Machine Learning, EPFL

***Abstract*—For this project, we were given a collection of samples from the CERN particle accelerator to be classified in two categories of events: Background or Signal i.e decay signature of the famous Higgs' Boson. To classify the particle detections, we implemented several machine learning methods such as least squares, grid descent, ridge regression and logistic regression. A dataset for which we had no class labels was finally tested against our classifier in order to assess the effectiveness of our implementation. The final accuracy, as given by an online software that could access the actual labels, was 80%. This work allowed us to explore the advantages and disadvantages of the different methods and become aware of the complications that arise from dealing with large amounts of data.**

I. Basic Methods Implementation

A. Least Squares GD

Gradient descent method is one of the most basic optimization methods. Using the gradient of the cost function (here MSE), it iteratively computes the weight by "taking a step" in the opposite direction of the gradient. Because of the iteration, it can be computationally costly.

B. Least Squares SGD

To reduce the complexity and cost of computing the gradient of the cost function over all the samples, a batch of points can be chosen at random to compute the gradient. This method is thus referred to as *stochastic* gradient descent. The gradient computed in this way is a cheap but unbiased estimate of the gradient.

C. Least Squares

As opposed to the above methods, least squares does not use an iterative method to compute the weights. It uses the solutions to the normal equations to compute the optimum of the MSE. This method doesn't require hyperparameter tuning but can suffer from over-fitting.

D. Ridge Regression

Lest squares can be improved by introducing a regularization term to avoid over-fitting. This method yielded the best classification results, as it is computationally cheap enough to run high range parameter selection.

E. Logistic Regression

Logistic regression is a regression model for categorical classification and it is particularly useful for binary outputs. Therefore, we expected it to give the best results. However in practice it was too costly to run proper cross validation.

We had several options on how to optimize the log-likelihood. In a first place we tried to implement Newton's method, which uses first (gradient) and second (hessian) order information about the cost function and thus converges faster but is way more computationally expensive. In fact, our computers couldn't support the computations required due to memory limitations. Therefore we decided to use a more tractable method such as gradient descent.

F. Regularized Logistic Regression

The above regression can yield some problems in the case of *linearly separable* data, because no finite-weight vector gives a minimum for the cost function; it will run indefinitely and yield infinite weights. To avoid this problem, we introduced a regularization term in the same fashion as for ridge regression.

II. Model and Parameters Selection

A. Cross Validation

Besides the optimization method, we also had the possibility to use a polynomial basis for our parameters and test whether a non linear model could better fit our data. This required the extra step of choosing the right polynomial degree.

Hence, for each model we needed to choose the appropriate degree and tune the hyper-parameters to obtain the best fit. Due to the large number of combinations that this raises, a systematic comparison of the different models was necessary: cross-validation. This allowed us to compute an *unbiased estimate* of the generalization error and its variance.

To find the optimal hyper-parameters, we combined cross-validation with a hand-search iteration over different hyper-parameter scales. We started from very large ranges of search for our parameters to determine the order of magnitude and successively refined our range for an increased precision in hyperparameter selection. We could have used an iterative and systematic way to do this as well but missed some more time to do so. Moreover, the possibility to manually and gradually select lambdas and degree allowed us to explore different possibilities and complexities in order to get a

better insight of the intricate data by playing around with the parameters.

B. Time/Accuracy Tradeoff

As we had limited access to computation power and time, we had to make choices according to the complexity of the models we wanted to implement and test. Since ridge regression seemed to have a high accuracy to computational cost ratio in our implementation, we decided to focus our efforts on it. Other more complex methods such as Logistic Regression with Newton's method that should have been the most accurate were on the other hand too greedy in terms of computational power and memory, especially in a situation where we have to exhaustively try a large number of various experiments conducted in a trial by error schema. To get a feeling on how to proceed we used a notebook and tested successively each method.

III. Preprocessing and Feature Engineering

A. Data-exploration

First of all, we are confronted with complex physical sensor data. In that case, domain knowledge is a tremendously valuable asset as it allows to determine, prior to any machine learning computation or feature engineering, the global structure of the model. Getting this pre-computational edge over the data set allows to start with a strong basis relative to choosing an approach and a strategy in order to better our predictions. Unfortunately, our multi-disciplinary group doesn't include a physician, nevertheless, we decided to spend some time on the definition of the different features, hoping to extract patterns. Importantly, we noticed that the value of PRI_jet_num strongly influences the distribution of the features. More precisely, for many of the features the value of PRI_jet_num determines if it is going to contain only undefined values. Therefore choosing to split the data according to such categorical data seems to be a good approach.

B. Preprocessing

As it is mentioned previously we wanted to split the data set according to categorical features such as PRI_jet_num. This is a pretty straightforward operation in itself, the main difficulty is to be consistent when recombining the data and when transposing this pre-processing step to the test set. After performing such step, we looked separately to each newly obtained subset. Before taking any further step, we wanted to sanitise our input, namely, we removed useless features (only containing undefined values) for each subset. We then proceeded to replace sparsely distributed NaNs by the median of its corresponding feature. The latter operation is robust to outliers and yields more homogeneous data. As last step of our data sanitation, we standardized each separate subset. As a side note, we also tried to interpolate with the mean of features, but trials were less successful. We also implemented a feature ranking analysis that gives an idea of which feature have the most singular, different and meaningful distribution compared to others. This could be used when we want to reduce the number of features while minimizing the impact on the quality of results. We could also remove data with low variance as it's not impacting our results and wasting computational power.

C. Feature Engineering

As part of feature engineering there is a wide range of things we could do to increase the relevance of features. We thought of several different approaches such as adding features as cross products of column or as logs or any aggregation of several features we see fit the classification. Although several combinations where tried, none seemed to be particularly useful. Nevertheless, as discussed in the cross validation part we built a polynomial basis for each subset.

IV. Results

The pipeline of our best submission is as follows : We first split the data, sanitised it and standardised as described in the pre-processing section. We then run a consequent number of cross validation iterations for each subset trying to get the best degree/lambda pair in terms of loss. To do so we started with a very wide logspace and degree range and progressively restricted the intervals until getting a satisfactory enough loss. As we split the model, we allowed ourselves to use relatively high degrees to fit these more specialised models. Our final results were obtained with :

PRI_jet_num	Lambda	Degree	loss_tr	loss_te
0	2569.38	5	0.7108	0.71260
1	0.4641	5	0.7978	0.8019
2	10.4622	4	0.7711	0.7773
3	0.2009	5	0.7740	0.7825

In the current state of our implementation, our data set is split in four different parts and does fairly well except for the split along PRI_jet_num = 1 where the loss is significantly worse than the others. This means that our approach might not be complex enough to grasp the structure of the underlying model. However, the aforementioned steps yielded very satisfactory improvements of the results without making use of exceedingly complex and costly models. Further classification improvements could be obtained with more domain knowledge, better feature engineering skills and further splits where our current version doesn't make the best of predictions.

Overall, given the advancement of our learning in the field, we were quite satisfied with the results that we could obtain without going into too complicated methods and felt like this project was a good opportunity to get a "hands on" approach with the methods.