Tracing the signatures of the Higgs Boson

Ioannis Bantzis
EPFL
ioannis.bantzis@epfl.ch

Manos Chatzakis
EPFL
emmanouil.chatzakis@epfl.ch

Maxence Hofer
EPFL
maxence.hofer@epfl.ch

***Abstract*—In this work we present a binary classifier for the CERN signature dataset, which predicts if a decay signature of a particle belongs to a Higgs Boson, by separately training parts of the data. Our experimental evaluation shows that our model achieves 81.65% accuracy with Cross Validation and 81.60% accuracy in the Aicrowd competition.**

I. Introduction

Motivation. Tracing the presence of Higgs bosons after proton collisions is one of the most prominent research directions of CERN[1] and has provided a rich database of particle footprints, by gathering data of their decaying signatures[2]. By examining those signatures, one can determine if the collision generated a Higgs boson. In this work, we develop a binary classification prototype, able to predict if a particle signature corresponds to such bosons. Our classifier uses regression models (Ridge Regression) and it is trained separately for different partitions of the dataset, in order to achieve high accuracy scores.

Challenge. Working with decaying signatures is a challenging task. These data have invalid values in multiple columns, while certain values follow specific distributions or have discrete values within a range. Based on these values, one may decide to split the dataset into parts corresponding to such values and train the data separately. Furthermore, in order to achieve high accuracy results for classification, preprocessing is crucial. It is useful to add the artificial constant feature to the data and perform polynomial feature expansion, being aware of the danger of overfitting. Finally, after the data preprocessing, we need to evaluate different regression models in order to find the most appropriate one.

Contributions. Our contributions can be summarized as follows:

- We preprocess the signature dataset and evaluate multiple regression models
- We present a model that trains different parts of the dataset separately in order to have optimal results
- We conduct an experimental evaluation over our model, which shows that our partial training technique can achieve 81.65% accuracy for Cross Validation[4] and 81.60% accuracy for the Aicrowd competition[3].

The rest of this work is organized as follows: Section I is the introduction, Section 2 presents the regression models, Section 3 shows how we preprocessed the data, Section 4 is the experimental evaluation and Section 5 is the conclusion.

II. Models

In this section we briefly describe our regression models. We denote w as the weight vector, $\mathcal{L}$ the loss function, γ the step size, t the current iteration, λ the regularization parameter, I the identity matrix, X the feature matrix and $y \in \{0,1\}^N$ the label vector.

Mean Squared Error with Gradient Descent (MSE_GD). MSE_GD finds the optimal weights by: $w^{(t+1)} = w^{(t)} - \gamma \nabla \mathcal{L}(w^{(t)})$ (1), where $\mathcal{L}$ is the Mean Squared Error.

Mean Squared Error with Stochastic Gradient Descent (MSE_SGD). Just like MSE_GD, MSE_SGD find the optimal weights by using one point of the dataset in each iteration (batch size 1), using $w^{(t+1)} = w^{(t)} - \gamma \nabla \mathcal{L}_n(w^{(t)})$ where $\mathcal{L}_n$ is the cost contributed by the n-th datapoint.

Least squares (LS). LS calculates the optimal weights using $w^* = (X^\top X)^{-1} X^\top y$.

Ridge regression (RR). RR calculates the optimal weights using $w^* = (X^\top X + 2N\lambda I)^{-1} X^\top y$.

Logistic regression (LR). LR calculates the optimal weights by (1) where $\mathcal{L}(w) = \frac{1}{N} \sum_{n=1}^{N} ln[1 + exp(x_n^T w)] - y_n x_n^T w$

Regularized logistic regression (RLR). RLR calculates the optimal weights by (1) where $\mathcal{L}(w) = \frac{1}{N} \sum_{n=1}^{N} ln[1 + exp(x_n^T w)] - y_n x_n^T w + \lambda \|w\|_2^2$.

III. Data Preprocessing

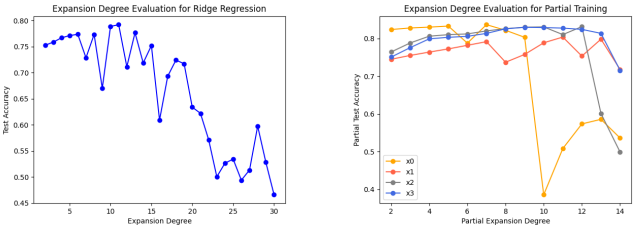
In this section we present the preprocessing methods we applied. We describe only the methods that led to better results, as other methods, e.g. standardization, feature crosses, log transformation led to worse accuracy.

Data Partitioning. First, we observed that the feature PRI_jet_num was a discrete value in range of 0 to 3. Thus, we separated the data in four different sub-datasets, x_0, x_1, x_2, x_3, according to their PRI_jet_num value, in order to be able to apply different preprocessing methods to each partition, based on the best accuracy results for each partition. Note that using this method we can evaluate each partition separately, in order to calculate the per-partition-accuracy. To obtain the final prediction label y we just need to concatenate the predictions of each partition.

Missing Values Removal. Some unobtained values in the experiments were replaced by -999.0. After partitioning the data, we observed that for each partition x_i these values were either in a fully invalid column or in the first column, 'DER_mass_MMC'. Naturally, we dropped the columns that contained only invalid values. For the first column, which contained mixed data, we replaced the missing values with

Model	Parameter	Acc_{test}
MSE_GD	$\gamma = 1e{-}08$, it=100	0.6573
MSE_SGD	$\gamma = 1e{-}11$, it=100	0.6573
LS		**0.7335**
RR	$\lambda = 0.00056$	**0.7334**
LR	$\gamma = 0.0001$, it=100	0.6816
RLR	$\gamma = 1e{-}06$, $\lambda = 1e{-}06$, it=100	0.6573

TABLE I: Regression Model Comparison.



(a) (b)

Fig. 1: (a) Feature expansion for RR_{naive} (b) Feature expansion for $RR_{partial}$, for each different partition x_i

the average of this column. Note that we also tried dropping the whole column, but it led to worse results. It is important to mention that different partitions x_i had different proportions of invalid values in the first column.

Polynomial Feature Expansion. We expanded the features of each partition x_i up to a certain degree d. A constant column containing the term 1 was also added as the first column. The final matrix has the column form $(1, x_i, x_i^2, ..., x_i^d)$ where the power refers to the element-wise power. This allows us to obtain non-linear relations between y and x and greatly improves the accuracy of the models.

IV. Results

In this section, we present our experimental evaluation. We compare the regression models, we conduct experiments about feature expansion and we evaluate the partial training method with data partitioning. Our results are determined by the test accuracy (Acc_{test}) on the signature dataset using Cross Validation (CV) and the Aicrowd competition.

Model Comparison. We used 5-fold Cross Validation (CV) to find the best-fitting parameters for our models, and then we compared their accuracy. For this set of experiments, we do not perform partitioning and expansion over the input. Table I contains the accuracy (Acc_{test}) of each model and their parameters. The initial weights for all models were set to 0, as experiments with random initialization led to worse results. We conclude that Ridge Regression (RR) along with Least Squares (LS) have the best accuracy, and thus, for the rest of our experiments we proceed with RR. We do not use LS, because in feature expansion certain columns are linearly dependent thus the final matrix is non-invertible.

Feature Expansion and Partial Training. Using Ridge Regression (RR) we study the effect of Feature Expansion (FE) and the Data Partitioning technique, which allows us to partially train the data. We denote d as the degree of expansion, RR_{naive} the instance of Ridge Regression that operates on the non-partitioned data and $RR_{partial}$ the instance of Ridge Regression that operates in the partitioned data. Also, for simplicity, we denote x_0, x_1, x_2, x_3 as the preprocessed (i.e. expanded) partitions.

x_i	λ	d
x_0	0.0200	7
x_1	0.1228	11
x_2	0.0328	12
x_3	0.0328	9

TABLE II: Partial Training Parameters.

Model	CERN Local Dataset	Aicrowd
RR_{naive}	0.7923	0.7900
$RR_{partial}$	**0.8165**	**0.8160**

TABLE III: Overall Results for Acc_{test}.

Figure 1a shows the trend of training accuracy for RR_{naive} for multiple degree d values. For $d = 11$ we obtain Acc_{test}=0.7923 for the signature dataset and Acc_{test}=0.7900 for the Aicrowd, thus it is not overfitting. This shows that the preprocessing technique of FE drastically improves the accuracy.

Although RR_{naive} along with Feature Expansion (FE) seem to be very powerful, we are still limited by the fact that this expansion (and the λ) are applied all over the dataset. By using our Data Partitioning method, we are able to partially train each x_i with different λ and d. Figure 1b shows the Acc_{test} obtained from Cross Validation for each one of the four partitions, plotted for different expansion degrees. We see that the best partial accuracy for each x_i might be in a different d, and partial training enables us to train each x_i separately, using the best corresponding d. The configurations (we choose the λ using heuristics and tuning) for the partial training are summarized in table II. By training and evaluating those values for Ridge Regression $RR_{partial}$, we managed to get Acc_{test}=0.8165 for the signature dataset and Acc_{test}=0.8160 for Aicrowd.

The final results for Aicrowd are depicted in table III. We conclude that our partial training method, which is based on Data Partitioning and Ridge Regression has great accuracy, both for Cross Validation and Aicrowd. We should mention that this method is orthogonal to the regression model; We tried different models (e.g. Logistic Regression) but it did not lead to better results.

V. Conclusion

We presented a machine learning prototype able to perform binary classification over the CERN signature dataset in order to predict if a particle is a Higgs Boson. Our experimental evaluation shows that our prototype, which works through partial data training, is able to achieve 81.60% accuracy on the Aicrowd competition and 81.60% with Cross Validation over the local data.

References

[1] Georges Aad et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 1–29.

[2] Claire Adam-Bourdarios et al. “Learning to discover: the higgs boson machine learning challenge”. In: *URL http://higgsml. lal. in2p3. fr/documentation* 9 (2014).

[3] *Aicrowd: Crowdsourcing AI to Solve Real-World Problems*. URL: https://www.aicrowd.com/.

[4] Payam Refaeilzadeh, Lei Tang, and Huan Liu. “Cross-validation.” In: *Encyclopedia of database systems* 5 (2009), pp. 532–538.