The Higgs boson machine learning challenge

Bruno Magalhaes[1], Marie Drieghe[2], Nicolas Casademont [3]

Abstract
Decay signatures of proton-proton collisions led to the discovery of the Higgs Boson. This new particle was discovered by analysing areas with a significant amount of unknown events (signals) against others that contain known ones (background). To tackle the challenge, we present a Machine Learning (ML) algorithm for the classification of these areas. Our algorithm reaches more than 80% accuracy by combing feature engineering and logistic regression. Due to the high-dimensionality of the solution, we perform iterative stepping via the stochastic gradient descent method. We detail a set of pre-processing methods that allow for shorter data set, smaller dimensionality, faster execution time, filtering of outliers, and imputation of missing values.

Email (SCIPER): [1] bruno.magalhaes@epfl.ch (212079) [2] marie.drieghe@epfl.ch (273688) [3] nicolas.casademont@epfl.ch (223485)

We present the sequence of pre-processing filtering steps tested in Section 1, where we explain their rationale and detail the results. The data after filtering is then used in the Machine Learning methods described in Section 4 to create a classification model. Results and final remarks are detailed in Section 5.

1. Data Augmentation

1. Rotation of input images with non-orthogonal roads We increased the number of input images from 100 to 148 by rotating 90deg, 180deg and 270deg the (16) training images that are not composed only of non-horizontal and non-vertical roads. This feature is activated by the flag `DATA_AUGMENTATION`.

2. Patches from concatenation points We increase by almost two-fold the number of patches per training image by performing extraction of all patches of size `IMG_PATCH_SIZE` that are placed at every `IMG_PATCH_SIZE/2` interval. This feature is activated by the flag `ADD_INTERCALATED_PATCHES`. In practice, it extracts patches with the corners at the mid-point positions of previously extracted patches, as displayed below:

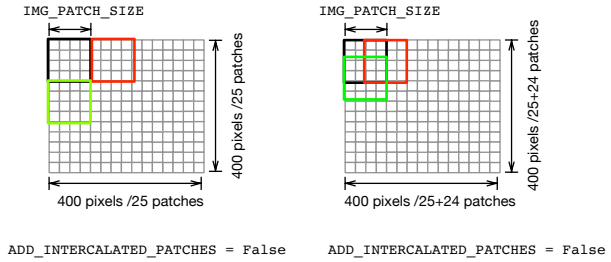

Figure 1. The collection of patches by stepping `IMG_PATCH_SIZE` (default, left) compared with `IMG_PATCH_SIZE/2` (`ADD_INTERCALATED_PATCHES` activated, right). When traversing image, next collected patch on the horizontal (vertical) direction is displayed as red (green).

2. Features

Two-step classification To reduce noise and allows classification at a finer scale, we added a second neural network to the processing. The initial one will be trained with pairs of [images,groundtruth] and outputs the segmented classification. The second network is trained with pairs [classification,groundtruth] and received the first network's classification as input, outputting the final classification. The second step of this operation is presented as `Phase 2` in the source code. A sample example is displayed below.

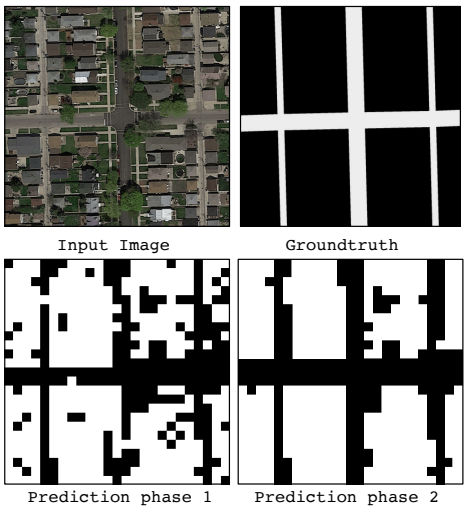

Figure 2. An example of the application of the two-phase propagation process: for a given input (top-left), the predicted segmentation at output of the neural network (bottom-left) is then re-input to a second neural network trained from groundtruth images (top-right), outputting the final classification (bottom-right).

A sample of a possible network architecture (for default parameters) is displayed in figure 3.

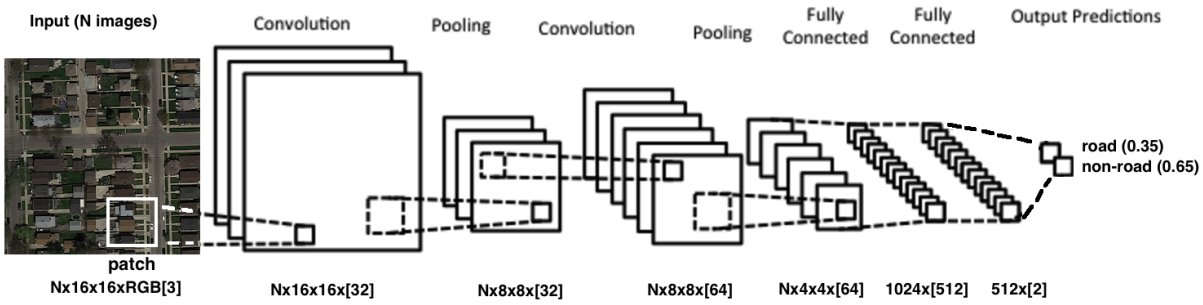

Figure 3. A general overview of the deep network architecture, with default parameters. Dimensionalities, padding types, depths and number of convolution+pooling layers are user-customisable.

Dim.	Eigen value	Explained var. ratio	Dim.	Eigen value	Explained var. ratio
0	1983617.8	7.43 e-1	**15**	3.98	1.49 e-6
1	470787.3	1.76 e-1	**16**	3.49	1.31 e-6
2	152814.2	5.72 e-2	**17**	2.74	1.02 e-6
3	35127.6	1.31 e-2	**18**	2.45	9.18 e-7
4	18184.4	6.81 e-3	**19**	1.65	6.19 e-7
5	2235.3	8.37 e-4	**20**	1.40	5.27 e-7
6	1833.1	6.86 e-4	**21**	0.99	3.73 e-7
7	1266.6	4.74 e-4	**22**	0.74	2.79 e-7
8	944.0	3.53 e-4	**23**	0.67	2.54 e-7
9	519.3	1.94 e-4	**24**	0.61	2.30 e-7
11	384.3	1.44 e-4	**25**	0.18	6.99 e-8
10	374.8	1.40 e-4	**26**	0.11	4.30 e-8
12	178.7	6.69 e-5	**27**	0.05	2.17 e-8
13	138.8	5.20 e-5	**28**	0.024	9.21 e-9
14	45.78	1.71 e-5	**29**	7.48e-8	2.80 e-14

Table 1. Details on the eigen-vectors after PCA filtering. Features are sorted by relevance (explained variance ratio).

3. Results

6. Data Standardization The final process performs data standardization by subtracting the mean and dividing the values by the standard deviation. This algorithm is particularly helpful to avoid computational overflows. It is only required when PCA is not executed beforehand, as the PCA implicitly executes the standardization.

4. Machine Learning Methods

We implemented the following three least squares methods: `least_squares`, `least_squares_GD` and `least_squares_SGD` for least squares with normal equations, regular and stochastic gradient descent. We also implemented `ridge_regression` for ridge regression using normal equations, `reg_logistic_regression` for regularized logistic regression using GD and SGD and `logistic_regression` for logistic regression with GD and SGD.

Our ideal solution relies on a combination of techniques. To obtain the best results we focus on **logistic regression**, the only binary classifier, as it provided better results than the available alternative methods. The final pre-filtering consisted of **mean amputation** over all features and standardization. After that we added a **polynomial interpolation of the fourth degree** that provided better result compared to the linear counterpart (see accuracy results for different degrees in Table 2).

Degree	Accuracy (%)
3	60
4	87
5	84
8	79

Table 2. Accuracy of predictions on the test data for different degrees of polynomials

Figure 4. Cross-validation results for test vs training data.

Due to the high amount of computation required, we use logistic regression with **stochastic gradient descent**.The step size gamma used for gradient descent was found with a binary search across a scope of parameters that yield low time to solution and low loss and set to $1.1e-03$. Finally, we tried to find a better solution using regularized logistic regression and 4-fold cross-validation for obtaining the best lambda. But since the difference between the error in the training data and the error in the test data was negligibly small (see Figure 4) we did not pursue the regularized logistic regression further.

5. Discussion and Summary

We applied machine learning techniques to data collected from the CERN particle accelerator aiming at recreating the process of discovering the Higgs particle. We detailed the challenges with the collected data — many missing values and high dimensionality — and presented an exploration of pre-filtering and transformation methods for reducing complexity (least significant dimensions) and sample size (outliers), and auto-completion of missing information (mean imputation). We presented the rationale behind an accurate classification based on a combination of algorithms. Our final algorithm can be executed by running `run.py`. It uses logistic regression with stochastic gradient descent. We cleaned and standardized the data and enriched it by creating a polynomial function of degree 4. This algorithm gave us the following score on Kaggle: 0.81192 .