JMLR: Workshop and Conference Proceedings 42:109-127, 2015 HEPML 2014

Optimization of AMS using Weighted AUC optimized models

Roberto Díaz-Morales RDIAZM@TSC.UC3M.ES

Ángel Navia-Vázquez NAVIA@TSC.UC3M.ES

DTSC, Univ. Carlos III de Madrid

Editor: Glen Cowan, Cécile Germain, Isabelle Guyon, Balàzs Kégl, David Rousseau

Abstract

In this paper, we present an approach to deal with the maximization of the approximate median discovery significance (AMS) in high energy physics. This paper proposes the maximization of the Weighted AUC as a criterion to train different models and the subsequent creation of an ensemble that maximizes the AMS.

The algorithm described in this paper was our solution for the Higgs Boson Machine Learning Challenge and we complement this paper describing the preprocessing of the dataset, the training procedure and the experimental results that our model obtained in the challenge. This approach has proven its good performance finishing in ninth place among the solutions of 1785 teams.

Keywords: Machine Learning, High Energy Particles, Weighted AUC, Median Discovery Significance

1. Introduction

High energy physics aims to discover the structure of matter by searching and studying its particles. Currently the main focus is on subatomic particles that are produced in collisions at accelerators. Processing fast and correctly the data generated is critical because these experiments are very expensive and produce a huge quantity of data but most of the events are irrelevant. For these reasons, the field of event selection using Machine Learning techniques is a very important research line.

Machine learning has proven to be a very helpful tool in the field of high energy particles and it has been profusely used in the last decade. If we focus on the area of event selection, many techniques have been tested. Some works like (Cranmer and Bowman (2005); Link et al. (2005); Berlich and Kunze (2004)) use genetic programming that is a kind of optimization algorithm inspired by biological evolution. In (Teodorescu and Sherwood (2008)) they use gene expression programming, a tree structure that can be adapted by changing their sizes and composition. Non evolutionary algorithms like decision trees have been used in (Roe et al. (2005)) to search neutrino oscillations and Neural Networks were used in (Whiteson and Whiteson (2009)) for top quark selection using the data of the Fermilab Tevatron accelerator. Recently, the number of publications in this field has greatly increased due to collaborations with the Large Hadron Collider (LHC) and the release of new datasets. In this new period, many papers have been published like (O'Neil et al. (2012)) that uses multivariate techniques or (Sadowski et al. (2014)) and (Baldi et al. (2015)) that use deep neural networks.

Inside this scope, ensembles (Zhang and Ma (2012)) are very flexible techniques that have shown to increase performance by means of combining several models (Kuncheva (2004)). The premise of using this kind of models is very simple and it is coherent with our real life behaviour,

we consult with others before making a decision. These techniques are used in many different tasks very successfully. For example, in computer vision (Ciresan et al. (2012)) uses an ensemble of Deep Neural Nets to classify images reaching the "state-of-art" and, in the area of weather prediction, ensembles of different models are the most common and successful techniques (Raftery et al. (2005)).

In this paper we present a procedure to maximize the Approximate Median Significance (AMS), which was the goal of the Higgs Boson Machine Learning Challenge. It is based on an ensemble that aims to maximize the median discovery significance and is composed of models that have been trained to maximize the Weighted AUC.

This paper is organized as follows. In Section 2, we talk about the Higgs Boson Machine Learning Challenge and we briefly describe the problem. Section 3 presents the dataset and the AMS as an evaluation metric. The preprocessing of the data, dimensionality reduction and missing values imputation are explained in section 4. In Section 5 we describe our model and the training procedure. The experimental results are presented in Section 6. Finally, we provide some conclusions and further work based on our results in section 7.

2. The Higgs Boson Machine Learning Challenge

2.1. The ATLAS and CMS experiments

High Energy Physics aims at discovering and studying new particles and their properties in order to develop a model of the real world at a subatomic level.

At the European Organization for Nuclear Research, CERN, inside the LHC, proton beams are accelerated in both directions using a circular trajectory and some protons collide, when this happens a variety of different particles with a broad range of energies are produced.

ATLAS (A Toroidal LHC Apparatus) and CMS (Compact Muon Solenoid) are two of the general purpose particle physics detectors constructed at the LHC. One of their most important goals was to investigate a piece of the Standard Model, the Higgs Boson.

The ATLAS and CMS experiments (Aad et al. (2012); Chatrchyan et al. (2012)) claimed the discovery of the Higgs Boson through decay mechanisms opening many new research lines to measure its characteristics.

2.2. The Challenge

The Higgs Boson Machine Learning Challenge (HiggsML (2014)) was organized to encourage the collaboration between high energy physicist and data scientist, its goal was to explore the potential of machine learning methods to improve the analysis of data produced by the ATLAS experiment.

This challenge, hosted by Kaggle (kaggle), took place from May 12th to September 15th 2014 and it was very successful given that at the end of the competition it had brought together 1785 teams.

2.3. The Goal

Particle decay is the spontaneous process by which elementary particles can be transformed into other elementary particles. If the particles created are not stable the decay process can continue.

A Higgs boson has many different processes through which it can decay producing other particles. The ATLAS experiment has recently observed signals of the Higgs boson decaying into two tau particles, but this decay is a small signal buried in background noise.

The events for the challenge have been produced using a simulator and they have been labeled in two categories:

- Background: They are known processes and are produced by the decay of particles discovered in previous generations of experiments.
- Signal: Events that known background processes cannot explain.

Having these two categories of events, the goal of the challenge was, given the feature space for the events, to find the region where the events labeled as signal are significantly higher than the events labeled as background. A statistical test is then applied to this region to determine the significance of this excess of signal to background.

3. The dataset and the evaluation metric

3.1. The dataset

The training set $\mathcal{D}$ was provided by the official ATLAS detector simulator and it contains a set of events $\{\mathbf{x}_i\}\, i = 1, .., n$. These events are labeled as signal ($y_i = s$) if the decay of Higgs bosons were produced and are labeled as background ($y_i = b$) if they were produced by three different processes, the decay of the Z boson in two taus, a pair of top quarks which can have a lepton and a hadronic tau among their decay and the decay of an W boson. The number of data patterns n was 250000.

Due to the complexity of the simulation process, each simulated event has a positive weight w_i so that the sum of weights of events falling in the region is an unbiased estimate of the expected number of events falling in the same region of the feature space during a given fixed time interval. The weights of the events are provided only for the training set so they are not part of the input to the classifier.

$$\begin{aligned} &\mathcal{D} = \{(\mathbf{x}_1, y_1, w_1), ..., (\mathbf{x}_n, y_n, w_n)\}, \\ &\mathbf{x}_i \in \mathbb{R}^d, \\ &y_i \in \{b, s\}, \\ &w_i \in \mathbb{R}^+. \end{aligned} \tag{1}$$

The number of features d is 30 (17 primitive features measured by the detector and 13 features computed from the primitive features). The different features of the dataset are briefly described in the Apendix A and more extensively in the official documentation of the challenge (Adam-Bourdarios et al., 2014).

3.2. The Evaluation metric

The objective of the challenge is to get the classifier with the highest approximate median significance AMS (See Formula 2).

$$AMS = \sqrt{2\left((s_T + b_T + b_r)log\left(1 + \frac{s_T}{b_T + b_r}\right) - s_T\right)},$$
$$s_T = \sum_{i=1}^{n} w_i 1\{y_i = s\}1\{\hat{y}_i = s\},$$
$$b_T = \sum_{i=1}^{n} w_i 1\{y_i = b\}1\{\hat{y}_i = s\}. \quad (2)$$

Where s_T and b_T are the sum of the weights of the true positive and false positive events respectively, $\hat{y}_i$ is the prediction obtained by the classifier and b_r is a regularization term that bias the optimal selection region of the feature space towards larger regions and decreases the variance of the AMS and whose value is 10 in this challenge.

4. Preprocessing

4.1. Dimensionality reduction

When we look at the reference frame (figure 1), we can see that the distributions of the events are invariant for a rotation around the **z** axis.

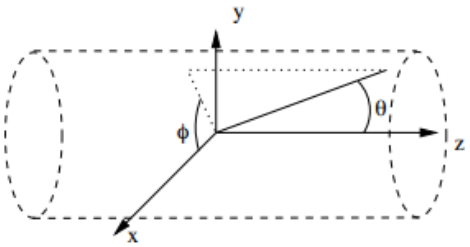


Figure 1: Reference Frame

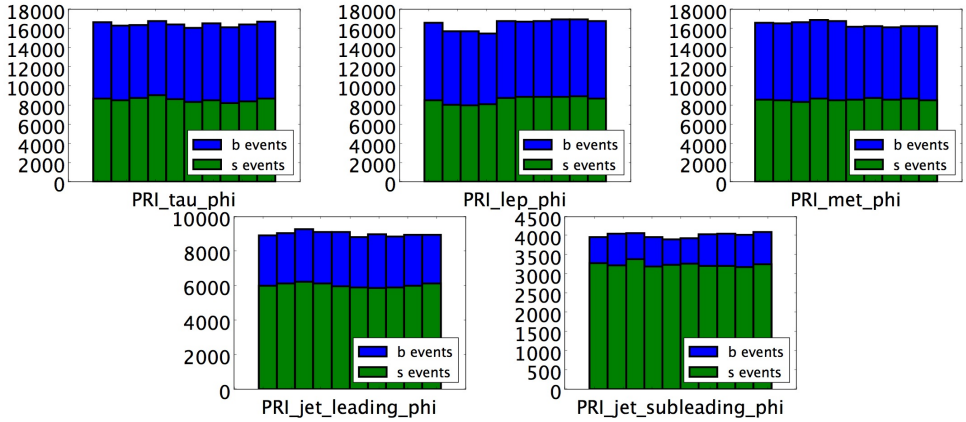
This invariability allows two types of pre-processing techniques:

- Data Augmentation: It is possible to create new training patterns by sampling events from the original training set and rotating them randomly around the z axis.
- Feature transformation: Applying this prior knowledge to transform the feature space. This allows us to crate models that present robustness to rotations around the z axis and are capable of learning the nature of the problem using a fewer number of samples because the classifiers do not waste effort learning the relationship that causes this invariability.

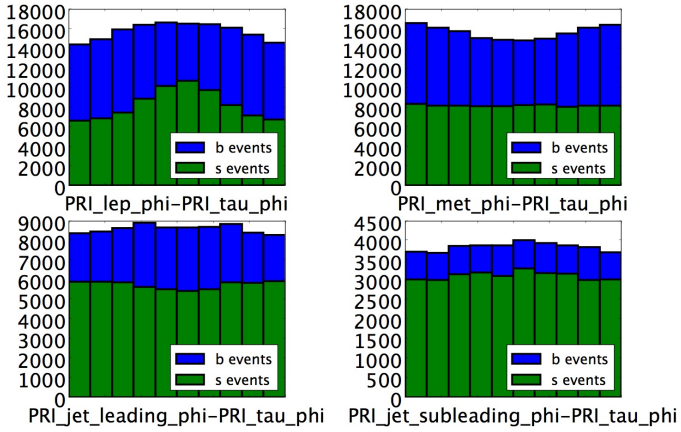
The number of samples in this training set is high, so we have chosen a feature transformation technique to reduce the computational cost. The feature space contains five features that are ϕ angles (PRI_tau_phi, PRI_lep_phi, PRI_met_phi, PRI_jet_leading_phi and PRI_jet_subleadint_phi, see Apendix A to see description of every feature). We have rotated every event to have PRI_tau_phi=0

by subtracting the value of PRI_tau_phi from the other angles and we have removed this feature from the feature space.

Figures 2(*a*) and 2(*b*) show the histograms of the original and rotated features respectively. We can see that features that didn't look informative because they had a uniform distribution for signal and background events are now apparently important with this transformation (see figure, showing a distribution with different parameters in the case of the signal and background events.



(*a*) The histogram of the features related with the ϕ angle in the reference frame (see Apendix A), abscissa axis is in the $[-\pi, \pi)$ range



(*b*) Histogram of the new features after a rotation of every event to make PRI_tau_phi equal to zero (see section 4.1), abscissa axis is in the $[0, 2\pi)$ range

Figure 2: Change of variables to create a feature space robust to rotations around the z axis

Furthermore, we have changed the sign of the variables PRI_tau_eta, PRI_lep_eta, PRI_jet_leading_eta, PRI_jet_subleading_eta when PRI_tau_eta is lower than zero. With this procedure the range of PRI_tau_eta covers 180° instead of 360° without loss of useful information for the classification.

For more information about the change of variables see the official forum of the challenge (HiggsML Forum (2014)).

4.2. Missing Values

Some features may be undefined in this dataset. There are two possible reasons a missing value:

- The feature DER_mass_MMC (see Appendix A) is an estimate mass of the Higgs boson candidate, if the topology of the event was too far from the expected topology this estimate appears as a missing value.
- Jets are particles than can appear 0, 1, 2 or 3 times in an event. Other features can be undefined depending on the number of jets in an event. For example, PRI_jet_leading_phi is the azimuth angle of the leading jet, which is obviously undefined when the number of jets is zero or PRI_jet_subleading_eta that is the pseudorapidity separation between jets and is undefined when the number of jets is 0 or 1. In this case the value is undefined because there is not a physical meaning for this feature.

Due to the fact that there is not a physical meaning for these features, we have avoided the difficulties of trying to estimate a value. We have used a standard procedure that consists on replacing every missing data by the average value of the non-missing data.

To avoid the loss of information in this imputation procedure, new features were also created to indicate what features with missing values there were in the event. If we separate the events attending the number of jets (0, 1, 2 or 3) and the availability of the estimated mass (missing,non-missing) there are eight different classes of events. We have added eight binary features to the data set to indicate the kind of pattern of every event.

There were two underlying motivations to apply this preprocessing procedure. In the case of using a linear classifier, thanks to the new features, a different bias term can be used for every case of missing data and in if we are using a tree-based algorithm the new features can help to create leaves that can separate perfectly the eight classes of missing data if necessary.

5. The model

Maximizing the AMS is not an easy task. The function is not smooth neither continuous, so gradient descent techniques over this function were discarded. Instead, we have chosen the approach described in this section.

5.1. Weighted area under ROC curve

The Receiver Operating Characteristics (ROC) curve is a very common tool for model evaluation and parameter selection that was used originally in signal detection theory and currently in machine learning research.

The ROC curve is a representation of the true positive rate (TPR) as a function of the false positive rate (FPR). The curve is obtained varying the value of the classification threshold from the highest output value to the lowest one. It illustrates the performance and quality of a binary classifier.

The area under the curve (AUC) is the area under the ROC, which is 1 for a perfect classifier and 0.5 for a fully random classification result. The AUC is closely related to the quality of the classification and as seen in (Ling et al., 2003) is statistically consistent and a more discriminating measure than the accuracy.

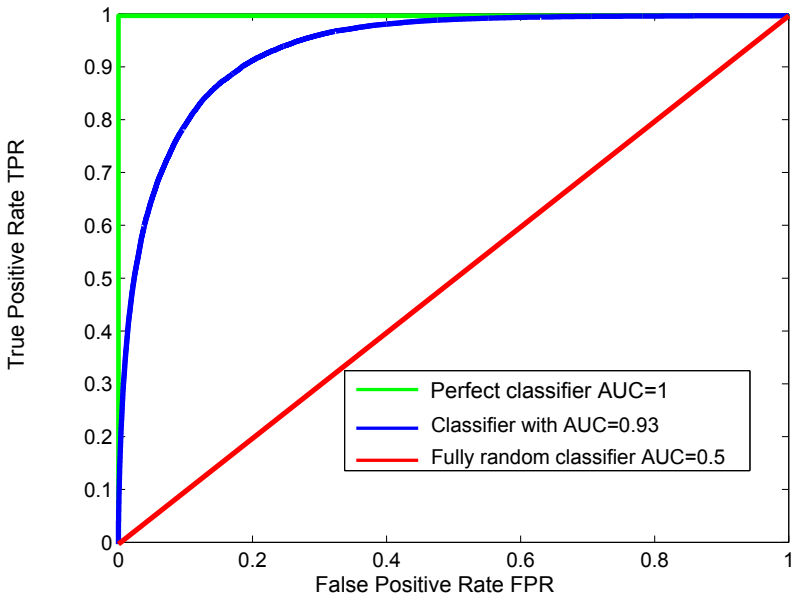


Figure 3: ROC curves

The AMS depends on the sum of the weights of the true positive events (s_T) and the sum of the weights of the false positive events (b_T). An increase of s_T or a reduction of b_T, keeping the other value constant, leads to a higher value of the AMS.

As a measure of the quality of the classifier we have selected a weighted AUC (WAUC) that takes into account the weights of the events in the evaluation of the TPR and FPR, called now WTPR and WFPR:

$$WTPR = \frac{\sum_{i=1}^{n} w_i 1\{y_i = s\} 1\{\hat{y}_i = s\}}{\sum_{i=1}^{n} w_i 1\{y_i = s\}}, \tag{3}$$

$$WFPR = \frac{\sum_{i=1}^{n} w_i 1\{y_i = b\} 1\{\hat{y}_i = s\}}{\sum_{i=1}^{n} w_i 1\{y_i = b\}}. \tag{4}$$

WTPR and WFPR are indeed the normalized b_T and s_T, so the WAUC seems to be a reasonable measure of quality for this problem:

$$WTPR = \frac{s_T}{max(s_T)}, \tag{5}$$

$$WFPR = \frac{b_T}{max(b_T)}. \tag{6}$$

A tradicional way of performing hyperparameter optimization is a grid search strategy, which is an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. Given the excessive computational cost of using this technique in this particular problem we have used a different strategy that, while is true that has a certain probability of finding hyperparameter values that are in a local maximum/minimum, has an acceptable run time:

1. Initialization: A set of possible values of every hyperparameter is defined manually (section 5.5 describes the possible values). The initial value of every hyperparameter has been selected randomly among the candidates.

2. Optimization: A hyperparameter is selected iteratively following a sequential order. For this hyperparameter, we explore the two previous and following values of the range evaluating the WAUC using a 10 fold cross validation technique.

3. Stopping criteria: If the WAUC has not been improved after a number of iterations equal to the number of hyperparameters the algorithm stops, otherwise it returns to step 2.

5.2. Bootstrap Aggregating

When we started to work with the dataset of this challenge we observed some instability in the results. Using many different machine learning algorithms, small changes in the parameters or removing a small number of samples could lead to a very different result.

To illustrate the instability of the AMS, we have trained ten times a Gradient Boosting classifier (see section 5.5) using 90% of the data set and evaluating the AMS on the other 10% of the data set.

Figure 4(a) shows the ten values of the AMS as a function of the rate of events classified as signal and figure 4(b) shows the average and the standard deviation. Results differ significantly every time that we train the algorithm, reaching an AMS higher than 4.0 in the best case and lower than 3 in the worst case.

Looking at the average of figure 4(b), a threshold that classify as signal a rate from 0.14 to 0.16 of the events seems to be a reasonable value.

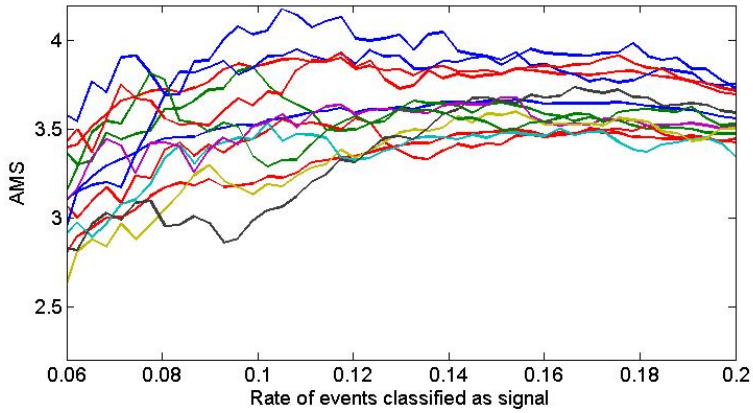
Bootstrap aggregating (Breiman, 1996), also called bagging, is a technique that allows us to improve the stability of machine learning algorithms. It reduces the variance of the results as well as help avoiding overfitting, resulting in an improvement of the accuracy.

Having a training set $\mathbf{X}$ with n samples it is possible to generate different training sets sampling from $\mathbf{X}$ uniformly with replacement, then a model for every dataset can be built and all of them can be combined by averaging the output. Experimental and theoretical results tell that this technique can push unstable procedures towards optimality.

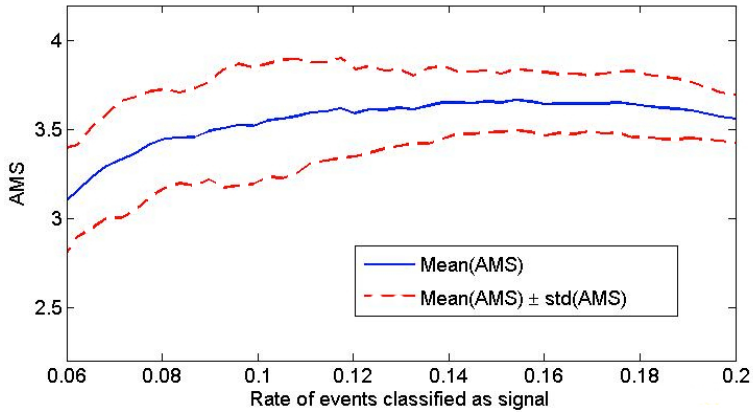
5.3. Stacking

In machine learning, ensemble methods use the output of multiple learning classifiers to obtain a better predictive performance than using a single classifier (Wolpert, 1992).

Empirically, these techniques are able to achieve better results if there is diversity among the classifiers. The statistical explanation is that without enough data in a region of the feature space a learning algorithm can find different hypothesis with the same accuracy on the training data, if different models are trained using different criteria and data we reduce the risk of choosing the wrong hypothesis.



(*a*) Ten values of the AMS as a function of the ratio of events selected as signal using 90% of the training set to train a Gradient Boosting classifier and evaluating the AMS over the other 10%,



(*b*) Average AMS and Average $\pm$ Standard Deviation AMS as a function of the ratio of events selected as signal using 90% of the training set to train a Gradient Boosting classifier and evaluating the AMS over the other 10%

Figure 4: AMS as a function of the rate of events classified as signal

The second reason to use an ensemble is the computational cost, simple models with a reasonable computational cost may be combined to obtain an accuracy only achievable by algorithms with a higher computational cost.

The third reason is that when a machine learning algorithm is looking for the optimum of a cost function, it may converge to a local optimum. Using the combination of different algorithms that use different criteria to find the objective we reduce this possibility.

The main disadvantage of the ensembles is that complex models are more willing to overfit than a single model due to the increase of flexibility.

To avoid overfitting problems we have chosen a stacking model because of its simplicity. The final output is a linear combination of l models.

$$f(\mathbf{x_i}) = \sum_{a=1}^{l} \alpha_a f_a(\mathbf{x_i}) \underset{0}{\overset{1}{\gtrless}} \eta. \tag{7}$$

The parameters of the ensemble, the weights $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_l\}$ and the threshold η have been chosen to maximize the AMS according to the following optimization procedure:

1. Initialization: Every weight is initialized using a uniform distribution defined over the interval $[0, 1]$. After that, we obtain the threshold values that classify 14% and 16% of the events as signal and we initialize the threshold η using a uniform distribution defined over the interval of both thresholds.

2. Optimization: A parameter is randomly selected. In the case of a weight, we evaluate every value from -1 to 1 using a step size of 0.01. Otherwise, if the parameter is η we evaluate the values that classify as signal from 14% of the events to 16% using steps of 0.05% (the threshold values evaluated in different iterations may change).

3. Stopping criteria: If the AMS has not been improved over the last ten iterations the algorithm stops, otherwise it returns to step 2.

This optimization procedure has been executed several times to reduce the probability of falling in a local maximum.

5.4. The algorithm

The whole procedure can be seen in table 1 and consists on a combination of the techniques described in this section.

As a first step, we have selected a group of algorithms that can handle a dataset of this size within reasonable runtime and we have created different training sets using transformations of the input space.

The second step is the parameter selection of every combination of algorithm and dataset using the procedure described in section 5.1 to reach the highest WAUC.

Once the parameters are chosen, the models are trained using bootstrap aggregating (Section 5.2) and the final step consists of an ensemble of the models whose weights and threshold are selected to maximize the AMS (Section 5.3).

5.5. Algorithms in the Ensemble

We have selected common machine learning algorithms that can handle a high number of patterns within a reasonable run time.

The algorithms selected and the sets of possible values of every hyperparameter were:

- Gradient Boosting Machines: Boosting techniques build models using the information of weak predictors, typically decision trees. The software that we used was XGBoost (Chen, 2014). This algorithm partitions the data into clusters and uses the cluster index as new features to further minimize the objective function. To create these clusters the algorithm recursively partitions data instances by growing a binary tree.

  Hyperparameters:

Table 1: Ensemble of maximized WAUC models

Given a data set with n samples $\mathcal{D} = \{(\mathbf{x}_1, y_1, w_1), ..., (\mathbf{x}_n, y_n, w_n)\}$

Step 0: Initialization

- A set of m training sets generated using different transformations of the feature space. $\mathbf{D} = \{\mathcal{D}_1, ..., \mathcal{D}_m\}$, where $\mathcal{D}_i = \{(\mathbf{x}_{i1}, y_1, w_1), ..., (\mathbf{x}_{in}, y_n, w_n)\}$ and $\mathbf{x}_{ia} = g_i(\mathbf{x}_a)$
- A set of r algorithms that can handle n samples within a reasonable run time is selected.
- The number of possible models is then $k = mr$.

Step 1: Parameter selection

The parameters of the k models have been chosen to maximize the WAUC using the optimization procedure described in section 5.1.

Step 2: Bagging

The l models with highest WAUC are selected and trained using bagging.

Step 3: Stacking

The final classifier is built as a linear combination of the models:

$f(\mathbf{x_i}) = \sum_{a=1}^{l} \alpha_a f_a(\mathbf{x_i}) \underset{0}{\overset{1}{\gtrless}} \eta$

The weights $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_l\}$ and the threshold η are obtained to maximize the AMS following the technique described in 5.3.

  - Rounds for boosting: 70, 90, 110, 130, 150, 170, 190, 210, 230, 250
  - Maximum depth: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  - Subsampling ratio: 0.8, 0.85, 0.9, 0.95, 1.0
  - Minimum loss reduction for leave partition: 0.5, 1, 1.5, 2, 2.5, 3
  - Step size: 0.1, 0.3, 0.5, 0.7, 0.9

- Random Forests: They are a technique that operates by constructing a multitude of decision trees. The output of the algorithm is based on statistical parameters like the mode, median or mean of the individual trees. The implementation used was the class RandomForestClassifier from the python library sklearn (Pedregosa et al., 2011).
  Hyperparameters:

  - Number of trees: 50, 100, 150, 200, 250, 300
  - Maximum depth: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  - Minimum number of samples to split a node: 1, 3, 5, 7, 9
  - Minimum number of samples a new leave: 1, 2, 3, 4, 5

- Linear Classifiers: Some linear models were trained using different cost functions and different regularization terms (L1 and L2 norm). We used the class linear_model.SGDClassifier from the python library sklearn (Pedregosa et al., 2011) that uses stochastic gradient descent over a cost function. We used different loss functions:
  - Logistic Regression: A probabilistic statistical classification algorithm that uses log loss in the cost function to model the probability of an event occurring.
  - Support Vector Machines (Vapnik, 1995): It is a non-probabilistic binary classifier that uses a hinge loss in the cost function. Nonlinear classification using Kernel functions was discarded because of their run time.

  Hyperparameters:
  - Regularization term: 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000
  - Initial learning rate: 0.2, 0.4, 0.6, 0.8, 1
  - Epochs: , 500, 1000, 1500, 2000, 2500, 3000
- Multilayer Perceptron (MLP): A feedforward artificial neural network trained using backpropagation with cross entropy error function (Haykin, 2004). We have used one hidden layer in the schema due to our run time limitations.

  Hyperparameters:
  - Neurons in the hidden layer: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
  - Step size: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000
  - Epochs: 50, 100, 200, 300

5.6. Expanding the feature space

We have built a group of training sets using different polynomial feature expansions. There were two reasons to do this, in the first place, to promote diversity in our classifiers increasing the probability of achieving a higher AMS with the ensemble, in the second place, because non linear classification problems are assumed to be equivalent to a linear classification problem in a higher dimensional feature space.

- $\mathcal{D}_0$ contains the training set after the preprocessing steps.
- $\mathcal{D}_1$ contains the features of $\mathcal{D}_0$ and new features composed by the product of every pair of features.
- $\mathcal{D}_2$ contains the features of $\mathcal{D}_0$ and new features composed by the ratios of every pair of features.
- $\mathcal{D}_3$ contains the features of $\mathcal{D}_0$ and new features composed by the subtraction of every pair of features.
- $\mathcal{D}_4$ contains the features of $\mathcal{D}_0$ and the products and ratios added in $\mathcal{D}_1$ and $\mathcal{D}_2$.
- $\mathcal{D}_5$ contains the features of $\mathcal{D}_0$ and the products and subtractions added in $\mathcal{D}_1$ and $\mathcal{D}_3$.
- $\mathcal{D}_6$ contains the features of $\mathcal{D}_0$ and the products, ratios, subtractions added in $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$.

6. Results

The score of this competition was evaluated using a test set of 550000 events whose labels and weights were unknown.

During the challenge, the leaderboard was calculated on 18% of the test data (called public leaderboard). After the competition ended, the final result was known and was based on the other 82% of the test data (called private leaderboard).

After obtaining the parameters of every model (section 5.1), we trained the models using 10 baggers and 90% of the training data in every bagger (section 5.2).

Using just one algorithm and the basic training set $\mathcal{D}_0$ the algorithm that performed best was Gradient Boosting that reached an AMS of 3.5 when it was trained without using bootstrap aggregation and an AMS of 3.65 using it.

We built several classifiers varying the models in the ensemble (section 5.3). For the challenge we selected the solution that obtained the best AMS in the public leaderboard. It was an ensemble composed by a linear combination of models obtained using Gradient Boosting Machines and the data sets $\mathcal{D}_1$, $\mathcal{D}_4$ and $\mathcal{D}_5$. This model reached an AMS of 3.76 in the private leaderboard finishing ninth of 1785 teams.

To see the effect of every dataset on the final score table 2 shows the results of Gradient Boosting Machines trained on every dataset and the result of our final ensemble.

Table 2: Leaderboard Result of using Gradient Boosting trained with the procedure described in section 5.4 for every dataset and the best ensemble obtained.

Leaderboard	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$	Best Ensemble
Public	3.65	3.69	3.66	3.66	3.68	3.70	3.72	3.77
Private	3.71	3.74	3.73	3.74	3.74	3.75	3.75	3.76

Figure 5 shows the AMS obtained by our final model in both public and private datasets as a function of the rejected events. This figure was provided by the organizers after the end of the challenge. We can observe the same instability that was shown in Figure 4(*a*) when the percentage of events classified as signal is under 0.14%. The shape of the AMS obtained in the public leaderboard presents a lack of a plateau and looks less stable. A possible explanation is that, due to the lower number of events in the public leaderboard, outliers can affect the AMS to a greater extent.

7. Conclusions and Further Work

In this paper we have presented the design, preprocessing, implementation and evaluation of a simple but efficient way to achieve a good approximate median significance. It is based on the combination of models trained to get the highest weighted area under the curve.

The main advantage of this model is its scalability. Every algorithm and bootstrap aggregating step can be executed in a distributed way and the software that we have used can run in a multithread environment, what allows us to reduce the runtime increasing the number of cores in the system.

This procedure has been tested in the Higgs Boson Machine Learning Challenge finishing in ninth place among the solutions of 1785 teams and reaching an AMS of 3.76.

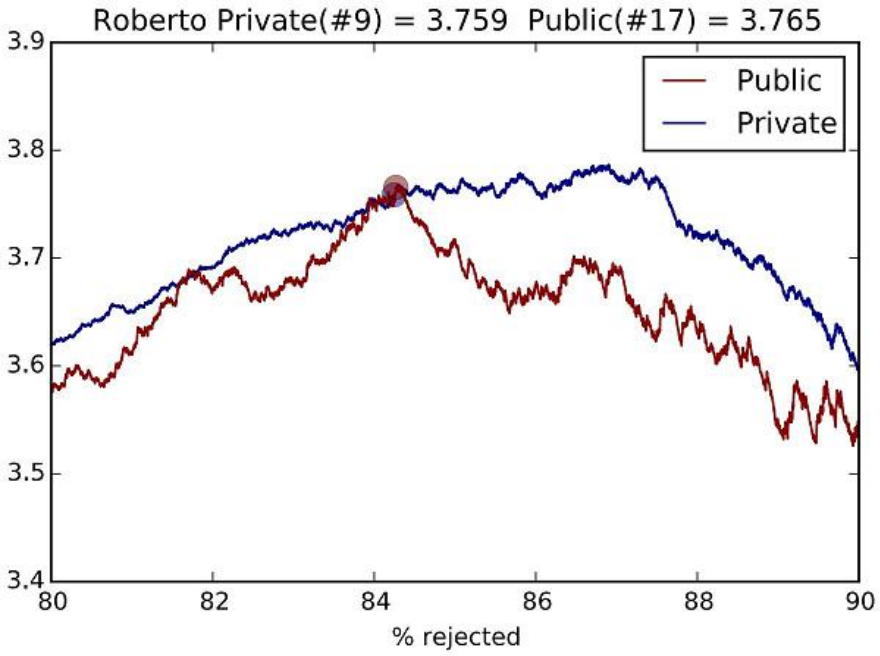


Figure 5: AMS as a function of rejected the events. This figure is reproduced with permission of the author (Kégl et al.)

After the very promising initial results that we have obtained and presented, we can suggest some future research lines:

- In this paper we have worked in an event selection problem using the Approximated Median Significance described in Section 3.2 as an evaluation metric. As a first research line we propose to deal with different particles and evaluation metrics. For example, a version of the AMS that can handle systematic uncertainties (Kégl et al.).
- We have used a linear combination of models as an ensemble method, but it would be interesting to use more complex aggregation techniques. One way is using post-aggregation (Omari and Figueiras-Vidal (2015)) that is a local form of ensemble that improve the performance of a previously aggregated classifier.
- In the ensemble we have included fast machine learning algorithms that can handle a high number of patterns. Another option is including sparse versions of more complex algorithms. For example, we discarded SVMs with Gaussian kernel due to its complexity but if we use a sparse and parallel solution (Díaz-Morales et al. (2011)) we can keep the complexity under control.

Acknowledgments

We would like to thank, in the first place, the Higgs Machine Learning Challenge organizers for their hard work and this exceptional competition, and in the second place, to everyone who partici-

pated in the challenge making it more interesting and competitive day after day.

References

Georges Aad, T Abajyan, B Abbott, J Abdallah, S Abdel Khalek, AA Abdelalim, O Abdinov, R Aben, B Abi, M Abolins, et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012.

Claire Adam-Bourdarios, Glen Cowan, Cecile Germain, Isabelle Guyond, Balázs Kégl, and David Rousseau. Learning to discover : the Higgs boson machine learning challenge v1.8, 2014. URL http://higgsml.lal.in2p3.fr/documentation/.

P Baldi, P Sadowski, and D Whiteson. Enhanced higgs boson to τ+ τ- search with deep learning. *Physical review letters*, 114(11):111801, 2015.

Rüdiger Berlich and Marcel Kunze. Parametric optimization with evolutionary strategies in particle physics. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 534(1):147–151, 2004.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Serguei Chatrchyan, Vardan Khachatryan, Albert M Sirunyan, A Tumasyan, W Adam, E Aguilo, T Bergauer, M Dragicevic, J Erö, C Fabjan, et al. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61, 2012.

Tianqi Chen. A general purpose parallel gradient boosting (tree) library, 2014. URL https://github.com/tqchen/xgboost.

Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

Kyle Cranmer and R Sean Bowman. Physicsgp: A genetic programming approach to event selection. *Computer Physics Communications*, 167(3):165–176, 2005.

R Díaz-Morales, Harold Y Molina-Bulla, and Angel Navia-Vázquez. Parallel semiparametric support vector machines. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 475–481. IEEE, 2011.

Simon Haykin. A comprehensive foundation. *Neural Networks*, 2(2004), 2004.

HiggsML. Higgs Boson Machine Learning Challenge, 2014. URL http://www.kaggle.com/c/higgs-boson.

HiggsML Forum. Higgs Boson Machine Learning Challenge Forum, 2014. URL http://www.kaggle.com/c/higgs-boson/forums/t/9576/reducing-the-feature-space.

B Kégl, D Rousseau, C Germain, I Guyon, and G Cowan. Introduction to the hepml workshop and the higgsml challenge. In *HEPML workshop at NIPS14-Neural Information Processing Systems Conference*.

Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

Charles X Ling, Jin Huang, and Harry Zhang. Auc: a statistically consistent and more discriminating measure than accuracy. In *IJCAI*, volume 3, pages 519–524, 2003.

JM Link, PM Yager, JC Anjos, I Bediaga, C Castromonte, C Göbel, AA Machado, J Magnin, A Massafferri, JM de Miranda, et al. Application of genetic programming to high energy physics event selection. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 551(2):504–527, 2005.

Adil Omari and Anbal R. Figueiras-Vidal. Post-aggregation of classifier ensembles. *Information Fusion*, (0):–, 2015. ISSN 1566-2535. doi: http://dx.doi.org/10.1016/j.inffus.2015.01.003.

Dugan C O'Neil, ATLAS Collaboration, et al. Tau identification using multivariate techniques in atlas. In *Journal of Physics: Conference Series*, volume 368, page 012029. IOP Publishing, 2012.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Adrian E Raftery, Tilmann Gneiting, Fadoua Balabdaoui, and Michael Polakowski. Using bayesian model averaging to calibrate forecast ensembles. *Monthly Weather Review*, 133(5):1155–1174, 2005.

Byron P Roe, Hai-Jun Yang, Ji Zhu, Yong Liu, Ion Stancu, and Gordon McGregor. Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 543(2):577–584, 2005.

Peter J Sadowski, Daniel Whiteson, and Pierre Baldi. Searching for higgs boson decay modes with deep learning. In *Advances in Neural Information Processing Systems*, pages 2393–2401, 2014.

Liliana Teodorescu and Daniel Sherwood. High energy physics event selection with gene expression programming. *Computer Physics Communications*, 178(6):409–419, 2008.

V Vapnik. The nature of statical learning theory, 1995.

Shimon Whiteson and Daniel Whiteson. Machine learning for event selection in high energy physics. *Engineering Applications of Artificial Intelligence*, 22(8):1203–1217, 2009.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

Cha Zhang and Yunqian Ma. *Ensemble machine learning*. Springer, 2012.

Appendix A. Brief description of the features

This is a list and a brief description of the features that appear in the dataset.

For a more detailed description go to the official documentation of the challenge (Adam-Bourdarios et al., 2014).

The angles are in radian with $-\pi$ to π range. The energy, Mass and momentum are in GeV and the other variables are unit less.

- **DER_mass_MMC**: The estimated mass of the Higgs boson candidate, obtained through a probabilistic phase space integration (may be undefined if the topology of the event is too far from the expected topology)
- **DER_mass_transverse_met_lep**: The transverse mass between the missing transverse energy and the lepton.
- **DER_mass_vis**: The invariant mass of the hadronic tau and the lepton.
- **DER_pt_h**: The modulus of the vector sum of the transverse momentum of the hadronic tau, the lepton, and the missing transverse energy vector.
- **DER_deltaeta_jet_jet**: The absolute value of the pseudorapidity separation between the two jets (undefined if PRI_jet_num ≤ 1).
- **DER_mass_jet_jet**: The invariant mass of the two jets (undefined if PRI_jet_num ≤ 1).
- **DER_prodeta_jet_jet**: The product of the pseudorapidities of the two jets (undefined if PRI_jet_num ≤ 1).
- **DER_deltar_tau_lep**: The R separation between the hadronic tau and the lepton.
- **DER_pt_tot**: The modulus of the vector sum of the missing transverse momenta and the transverse momenta of the hadronic tau, the lepton, the leading jet (if PRI_jet_num ≥ 1) and the subleading jet (if PRI_jet_num $= 2$) (but not of any additional jets).
- **DER_sum_pt**: The sum of the moduli of the transverse momenta of the hadronic tau, the lepton, the leading jet (if PRI_jet_num ≥ 1) and the subleading jet (if PRI_jet_num $= 2$) and the other jets (if PRI_jet_num $= 3$).
- **DER_pt_ratio_lep_tau**: The ratio of the transverse momenta of the lepton and the hadronic tau.
- **DER_met_phi_centrality**: The centrality of the azimuthal angle of the missing transverse energy vector w.r.t. the hadronic tau and the lepton.
- **DER_lep_eta_centrality**: The centrality of the pseudorapidity of the lepton w.r.t. the two jets (undefined if PRI jet num ≤ 1).
- **PRI_tau_pt**: The transverse momentum of the hadronic tau.
- **PRI_tau_eta**: The pseudorapidity η of the hadronic tau.

- **PRI_tau_phi**: The azimuth angle ϕ of the hadronic tau.
- **PRI_lep_pt**: The transverse momentum of the lepton (electron or muon).
- **PRI_lep_eta**: The pseudorapidity η of the lepton.
- **PRI_lep_phi**: The azimuth angle ϕ of the lepton.
- **PRI_met**: The missing transverse energy.
- **PRI_met_phi**: The azimuth angle ϕ of the missing transverse energy.
- **PRI_met_sumet**: The total transverse energy in the detector.
- **PRI_jet_num**: The number of jets (integer with value of 0, 1, 2 or 3; possible larger values have been capped at 3).
- **PRI_jet_leading_pt** The transverse momentum of the leading jet, that is the jet with largest transverse momentum (undefined if PRI jet num $= 0$).
- **PRI_jet_leading_eta**: The pseudorapidity η of the leading jet (undefined if PRI_jet_num $= 0$).
- **PRI_jet_leading_phi**: The azimuth angle ϕ of the leading jet (undefined if PRI_jet_num $= 0$).
- **PRI_jet_subleading_pt**: The transverse momentum of the leading jet, that is, the jet with second largest transverse momentum (undefined if PRI_jet_num ≤ 1).
- **PRI_jet_subleading_eta**: The pseudorapidity η of the subleading jet (undefined if PRI_jet_num ≤ 1).
- **PRI_jet_subleading_phi**: The azimuth angle ϕ of the subleading jet (undefined if PRI_jet_num ≤ 1).
- **PRI_jet_all_pt**: The scalar sum of the transverse momentum of all the jets of the events.