Fact sheet for the Higgs ML challenge

Answering this fact sheet is mandatory to win one of the money price or the "HEP meets ML award". We encourage all other participants to answer it as well (even if they have not packaged their software). Your answers will remain private and will be used to analyze the results of the challenge. Statistics and anonymized excerpts might be made public. If your best submissions (best on the public leaderboard or chosen by you) use really different models, please fill the form twice (common information can be filled only once). If they use similar models, please fill the form only once.
Note : if on some mandatory question (red star) you want to choose the "Other" answer, you have to both tick and fill in some text.

* Required

Team name *

Christian Bracher

Team website URL

https://github.com/cbracl

Team Leader Name

Christian Bracher

Contact email *

cbracher69@gmail.com

Please ignore, do not fill.

Public score *

To allow crosschecks, please indicate the public leaderboard score (with all digits) your submission has obtained (one value if this questionnaire concerns one submission, two values separated by a space if this questionnaire concerns two similar submissions)

2.70912

Team Leader Address and phone number

Who are you ?

Describe yourself in a few words, education, occupation

Theoretical physicist, turning his attention to data science

Background *

What is your scientific community?

☐ Machine Learning

☐ High Energy Physics

☑ Other: Theoretical Physics

Other team members, names, description and background

Method summary

Title of your contribution *

Name of your submission file

Higgs Linear-Gaussian Mc

General description

Provide a general summary with relevant background information: Where does the method come from? Is it novel? Name the prior art.

I wrote a classifier based on linear regression of event weights, using a kernel-type method (Gaussian splits, overlaps with shifted Gaussians) to increase the number of features for each event. It's a fast and compact method, even though it does not come close in accuracy to the boosted tree and neural network classifiers proposed by others. It typically scored in internal validation between 2.7 and 2.9.
I wrote most of this myself, based on an analysis of the specific data, and original ideas (I employ the scikit-learn ML library for Python). My goal was to learn more about classification methods for a realistic, "difficult" data set.

References

How should people reference your work? What other references are relevant to your work?

Work is free for use by everyone, provided a notice of my original authorship is kept.

Supplementary on-line material

Provide a URL to a web page, technical memorandum or a paper

Preprocessing and feature construction

Normalization *

Check all that apply.

☐ 1. Feature standardization (for numerical variables)
☑ 2. Sample normalization (for numerical variables)
☐ 3. Replacement of the missing values
☐ 4. Grouping modalities (for categorical variables)
☐ 5. Discretization (for numerical variables)
☐ Other:

Feature extraction *

Check all that apply.

☐ 1. Application of random functions
☐ 2. Application of filter banks
☑ 3. Hand-crafted features
☐ 4. Trained feature extractors
☐ 5. Sparse coding
☐ Other:

Dimensionality reduction *

Check all that apply.

☐ 1. Linear manifold transformations (e.g. factor analysis, PCA, ICA)
☐ 2. Non-linear dimensionality reduction (e.g. KPCA, MDS, LLE, Laplacian Eigenmaps, Kohonen maps)
☐ 3. Clustering (e.g. K-means, hierarchical clustering)
☐ 4. Deep Learning (e.g. stacks of auto-encoders, stacks of RBMs)

☐ 5. Feature selection

☑ Other: None performed

Prediction

Base predictor *

Check all that apply.

☐ 1. Decision tree, stub, or Random Forest

☑ 2. Linear classifier (Fisher's discriminant, SVM, linear regression)

☐ 3. Non-linear kernel method (SVM, kernel ridge regression, kernel logistic regression)

☐ 4. Gaussian Process

☐ 5. Bayesian non parametric (other that Gaussian Process)

☐ 6. Naïve Bayes

☐ 7. Bayesian Network (other than Naïve Bayes)

☐ 8. Neural Network (or Deep Learning Method)

☐ 9. Bayesian Neural Network

☐ 10. Nearest neighbors

☐ 11. Don't know

☐ Other:

Loss function *

Check all that apply.

☐ 1. Hinge loss (like in SVM)

☑ 2. Square loss (like in ridge regression)

☐ 3. Logistic loss or cross-entropy (like in logistic regression)

☐ 4. Exponential loss (like in boosting)

☐ 5. None

☐ 6. Don't know

☐ Other:

Regularizer *

Check all that apply.

☐ 1. One-norm (sum of weight magnitudes, like in Lasso)

☐ 2. Two-norm (||w||^2, like in ridge regression and regular SVM)

☑ 3. None

☐ 4. Don't know

☐ Other:

Ensemble method *

Check all that apply.

☐ 1. Boosting

☐ 2. Bagging (check this if you use Random Forest)

☐ 3. Bayesian ensemble

☐ 4. Other ensemble method

☑ 5. None

□ 6. Don't know

□ Other:

Integration of the AMS in the method

Have you used the AMS at the training stage beyond determining an optimal threshold ?

□ 1. Yes

☑ 2. No

If yes, please explain how

also comment whether your method could be adapted to a different scoring formula

Model selection and transfer learning *

Check all that apply.

□ 1. Leaderboard performance on validation data used for model selection

□ 2. K-fold or leave-one-out cross-validation (using training data)

□ 3. Virtual leave-one-out (close form estimations of LOO with a single classifier training)

□ 4. Out-of-bag estimation (for bagging methods such as Random Forest)

□ 5. Bootstrap estimation (other than out-of-bag)

☑ 6. Other cross-validation method

□ 7. Bayesian model selection

□ 8. Penalty-based method (non-Bayesian)

□ 9. Bi-level optimization

□ 10. Unsupervised learning with unlabeled validation and/or test data or transduction

□ 11. Knowledge transfer from the development data of past phases

□ Other:

Method advantages

Algorithmic complexity *

Check all that apply without contradicting each other.

□ 1. Linear in number of features

☑ 2. Quadratic in number of features

□ 3. Super-quadratic in number of features

☑ 4. Linear in number of training examples

□ 5. Quadratic in number of training examples

□ 6. Super-quadratic in number of training examples

☑ 7. Linear in number of test examples

□ 8. Quadratic in number of test examples

□ 9. Super-quadratic in number of test examples

□ 10. Adaptive algorithmic complexity

□ 11. Don't know; too difficult to evaluate

□ Other:

Qualitative advantages *

Check all that apply.

☑ 1. Simple method

☑ 2. Easy to implement

□ 3. Easy to parallelize

☑ 4. Require little memory

□ 5. Self-contained (does not rely on third party libraries)

☑ 6. Require only freeware libraries

□ 7. Theoretically motivated

□ 8. Novel / original

□ 9. Ease of interpretability of the classifier parameters or model.

□ 10. Robustness with respect to lack of training data

□ 11. Flexibility w.r.t different target of optimization

☑ Other: Some parallelization poss

Comparison with other methods

Contrast your proposed method with others you tried or know of. What is a critical element of success of your method?

have different features.
* Splitting features using Gaussian basis functions boosts the score significantly.

Extensions; I tried to build a staged classifier that first applies a linear classifier, or PCA/LDA, to eliminate more obvious background events first, before fine-tuning the Gaussian kernel/linear regressor scheme on the remaining entries, but in the end, ran out of time to implement it. There were indications that such a two-step scheme would lead to a modest increase in score.

Physics aspects

Meaning of the variables *

Has the knowledge of the physical meaning of the variables helped you significantly?

☑ Yes

□ No

If yes please explain

Indirectly - for instance, rotational symmetry of the setup implied to consider only relative angles (phi features). I also helped me eliminate features that were linear combinations of other features in some classes, reducing the effective number of features and enabling me to go ahead with a simple linear regressor.

Source of the data *

Has the knowledge on the source of the data (what kind of simulator) helped you significantly ?

□ Yes

☑ No

If yes, please explain

Physics advantage

Please comments on the suitability of your method for a real physics analysis

Even though tree-based methods yield better classification scores, my approach might be useful as an input filter, to sort out many 'obvious' background events quickly. It could probably be implemented as a filter for streaming data.

Software implementation

Location of the code (upload to Kaggle or URL)

To package the code, please follow all instructions in : https://github.com/sbinet/hml

https://github.com/cbracl

Platform *

☑ 1. Windows

□ 2. Linux

□ 3. Mac OS

☑ Other: platform-independent

Availability *

Check all that apply.

□ 1. Proprietary in house software

□ 2. Commercially available in house software

□ 3. Freeware or shareware in house software

□ 4. Off-the-shelf third party commercial software

☑ 5. Off-the-shelf third party freeware or shareware

□ 6. Not ready yet, but may share later

□ Other:

Language *

Check all that apply.

□ 1. C/C++/C#

□ 2. Java or Weka

□ 3. Matlab or Octave

☑ 4. Python

□ 5. Go

□ 6. R or S

□ Other:

Details on software implementation. Please describe precisely any library, software suite and configuration that would be required to run your software.

The classifier is built upon Python 2.7.
Libraries needed:

pandas - manipulating data sets (series, data frames)
numpy - numerical support (arrays), vectorized math operations
scipy - efficient Gaussian functions, normal distribution
sklearn - linear regressor (linear_methods), validation (cross_validation)

Hardware implementation

Memory *

○ 1. <= 2 GB

● 2. > 2GB but <= 8 GB

○ 3. > 8 GB but <= 32 GB

○ 4. > 32 GB

Parallelism *

Check all that apply

□ 1. Multi-processor machine

□ 2. Run in parallel different algorithms on different machines

☑ 3. None

☑ Other: (could be parallelized)

Development effort

Total human effort *

How much time did you spend developing algorithms and customizing code specifically for the challenge?

○ 1. A few man hours, even less

● 2. A few man days

○ 3. 1-2 man weeks

○ 4. 3-4 man weeks

○ Other:

Total machine effort *

How much time did your machines spend crunching algorithms to improve your solution?

● 1. A few hours, even less

○ 2. A few days

○ 3. 1-2 weeks

○ 4. > 2 weeks

Challenge duration OK? *

Did you get enough development time?

○ 1. Yes

● 2. No, but I could spend more time

○ 3. No, I could have spent more time

Final training time in seconds *

How long did it take in seconds for your machine to do the training for the final submission

29

Final evaluation time in seconds *

How long did it take in seconds for your machine to do the evaluation for the final submission

16

Around the challenge

How did you learn about the challenge ?

Check all that apply

○ 1. Kaggle web site

○ 2. poster

○ 3. social media

○ 4. blogs

○ 5. physics conference

○ 6. ATLAS meeting

○ 7. mailing lists

● 8. colleagues/friends/family

○ Other:

What was your primary motivation to participate to the HiggsML challenge ?

Check all that apply

○ 1. earn money

○ 2. visit CERN

○ 3. earn Kaggle points and tier

○ 4. spirit of competition

○ 5. make your work known

○ 6. learn about physics, by using simulated data used for real by physicists working on Higgs boson physics

● 7. learn about machine learning techniques

○ 8. requested by your teacher/tutor/supervisor

○ 9. have fun

○ 10. be part of it

○ Other:

Would you be interested to participate in 2015 to a challenge on a different high energy physics topic (no commitment) ?

☑ Yes

□ No

Final comments

anything you would like to express about the challenge which did not fit elsewhere ?

Note: I removed a small bug from the code which will likely lead to a score that slightly varies from the one generated by my submitted competition code.

Submit

Never submit passwords through Google Forms.