Writing Scientific Papers and Software

Cheng Soon Ong
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—The Higgs Boson challnege invovles using a large dataset to classify whether or not a particle is a Higgs Boson particle. We assessed various regression and classificaiton algorithms that could be used to predict the outcome and implemented some on the dataset. To further improve the results we preconditioned the data through various methods. Finally, a testing system was written to test all the combinations of preconditioning algorithms with the machine learning algorithms and output the one with least errors.**

I. Introduction

The Higgs Boson is a recently discovered very unpredictable particle. Through the particle collider at CERN, much data has been collected regarding the features of a decay processes when a Higgs Boson particle was present and was not present. We have analyze and automate several machine learning algorithms to predict whether or not the Higgs Boson particle is present given these features.

When creating an algorithm there many choices for how to precondition the dataset and predict the output. We attempted to discover the optimal model by running various algorithms with various preconditioned datasets, and choosin the best one. We describe each of the techniques we use in the in the Precondition section and Classification section.

II. Preconditioning

A. Filling in Missing Data

After observing the dataset we deduced that the values -999 and 0 were fillers for when the actual value is unknown. They occurred too frequently and were irregular compared to the other values of the same feature. We tried several approaches to filling in these gaps.

1) Mean/Median: We tried taking the mean of the rest of the data, ignoring the fillers, for every feature are replacing their mean or median. Either cases resulted in improvement in the accuracy.
2) Regression Filling We used the made the features datapoints and each datapoint a feature and used linear regression to fill the blanks. This method was more complex and time consuming to train but resulted in slightly better improvements in accuracy.
3) Delete features Eliminating the features with a 0 or -999 reduced the number of features to 19 which resulted in a poor accuracy.
4) Deleting samples 181886 of the data points have a 0 or -999 so deleting all such points causes our algorithm to drastically underfit the model since the remaining sample size is too small, resulting in a poor accuracy.

B. Feature Extraction and Augmentation

We tried a few computational methods of feature extraction with few results. These include removing features with a large variance or correlation. Intuitively, if a feature has a large variance it may seem that it is just background noise and can be ignored; however many natural measuerments have noise and outliers so it is not enough of a reason to discard the value. Furthermore, we tried visualizing the data and removing features that are highly correlated with each other. If one feature can predict another it seems unlikely that we need both features since they would be near linear combinations of each other and that is already accounted for with the weights. Finally, we wrote python scripts try removing one feature at a time and running the regression. This method seemed to have the most success and improved the accuracy the most.

C. Normalization

The data as given has a wide domain and is hard to statistically analyze, so we used several normalization techniques on the data.

1) Guassian Normalizer
2) Decimal Scaling
3) Minmax Normalizer
4) Guassian Outlier removal

III. Calssification Models

We used linear models and ran various classification algorithms

A. Stochastic Gradient Descent

detail parameters and stuff

B. Least Squares

C. Ridge Regression

This was the given figure example
Referring to Figure 1 and Figure 2.

D. Logistic Regression

this is a citation [1], [2].

E. Testing and Results

Describe how testing algorithms works Use figures and maybe tables in results: See this example: Table I.

Basis	Support	Suitable signals	Unsuitable signals
Fourier	global	sine like	localized
wavelet	local	localized	sine like

Table I
CHARACTERISTICS OF FOURIER AND WAVELET BASIS.

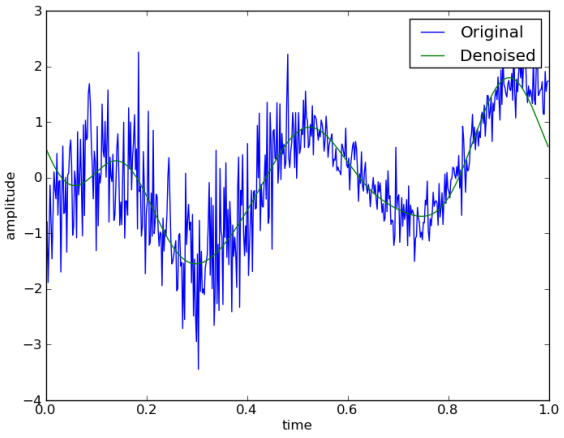

Figure 1. Signal compression and denoising using the Fourier basis.

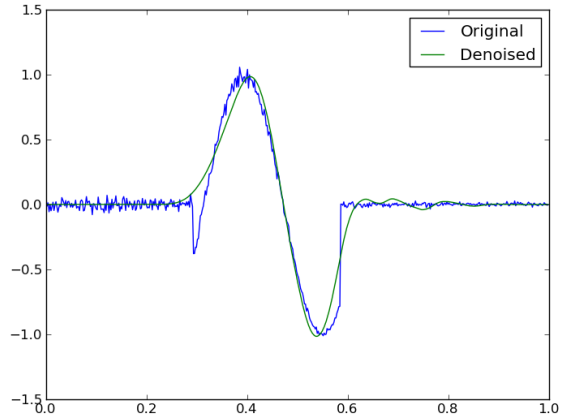

Figure 2. Signal compression and denoising using the Daubechies wavelet basis.

IV. TIPS FOR GOOD SOFTWARE

There is a lot of literature (for example [3] and [4]) on how to write software. It is not the intention of this section to replace software engineering courses. However, in the interests of reproducible research [5], there are a few guidelines to make your reader happy:

- Have a `README` file that (at least) describes what your software does, and which commands to run to obtain results. Also mention anything special that needs to be set up, such as toolboxes[1].
- A list of authors and contributors can be included in a file called `AUTHORS`, acknowledging any help that you may have obtained. For small projects, this information is often also included in the `README`.
- Use meaningful filenames, and not `temp1.py`, `temp2.py`.
- Document your code. Each file should at least have a short description about its reason for existence. Non obvious steps in the code should be commented. Functions arguments and return values should be described.
- Describe how the results presented in your paper can be reproduced.

A. LaTeX Primer

LaTeX is one of the most commonly used document preparation systems for scientific journals and conferences. It is based on the idea that authors be able to focus on the content of what they are writing without being distracted by its visual presentation. The source of this file can be used as a starting point for how to use the different commands in LaTeX. We are using an IEEE style for this course.

1) Installation: There are various different packages available for processing LaTeX documents. On OSX use MacTeX (http://www.tug.org/mactex/). On Windows, use for example MikTeX (http://miktex.org/).

2) Compiling LaTeX: Your directory should contain at least 4 files, in addition to image files. Images should be in `.png`, `.jpg` or `.pdf` format.

- IEEEtran.cls
- IEEEtran.bst
- groupXX-submission.tex
- groupXX-literature.bib

Note that you should replace groupXX with your chosen group name. Then, from the command line, type:

```
$ pdflatex groupXX-submission
$ bibtex groupXX-literature
$ pdflatex groupXX-submission
$ pdflatex groupXX-submission
```

This should give you a PDF document `groupXX-submission.pdf`.

[1]For those who are particularly interested, other common structures can be found at http://en.wikipedia.org/wiki/README and http://www.gnu.org/software/womb/gnits/.

3) Equations: There are three types of equations available: inline equations, for example $y = mx+c$, which appear in the text, unnumbered equations

$$y = mx + c,$$

which are presented on a line on its own, and numbered equations

$$y = mx + c \tag{1}$$

which you can refer to at a later point (Equation (1)).

4) Tables and Figures: Tables and figures are "floating" objects, which means that the text can flow around it. Note that `figure*` and `table*` cause the corresponding figure or table to span both columns.

V. Summary

Do this too. The aim of a scientific paper is to convey the idea or discovery of the researcher to the minds of the readers. The associated software package provides the relevant details, which are often only briefly explained in the paper, such that the research can be reproduced. To write good papers, identify your key idea, make your contributions explicit, and use examples and illustrations to describe the problems and solutions.

Acknowledgements

probs acknowledge class and stuff The author thanks Christian Sigg for his careful reading and helpful suggestions.

oh and see the code for how to do the bibliography

References

[1] G. Anderson, "How to write a paper in scientific journal style and format," 2004, http://abacus.bates.edu/ ganderso/biology/resources/writing/HTWtoc.html.

[2] J. B. Buckheit and D. L. Donoho, "Wavelab and reproducible research," Stanford University, Tech. Rep., 2009.

[3] A. Hunt and D. Thomas, *The Pragmatic Programmer*. Addison Wesley, 1999.

[4] J. Spolsky, *Joel on Software: And on Diverse & Occasionally Related Matters That Will Prove of Interest etc..: And on Diverse and Occasionally Related Matters ... or Ill-Luck, Work with Them in Some Capacity*. APRESS, 2004.

[5] M. Schwab, M. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science and Engg.*, vol. 2, no. 6, pp. 61–67, 2000.