DAML Project-4: Exotic searches with ATLAS and ML Classification

Deadline: Friday 5 April at 4 PM

Christos Leonidopoulos*

University of Edinburgh

March 23, 2024

1 Description

In the fourth and last DAML project we will explore whether a ML classifier can improve a traditional search for exotic particles with the ATLAS data. The project builds upon the Week-8, checkpoint #12. You are allowed (in fact, encouraged) to recycle your old code.

We will evaluate the statistical significance of a signal-like deviation in the simulated data. We follow two approaches:

- The traditional approach where we apply some (so-called "square") selection cuts on the kinematic variables, and then attempt two fits (for the H_0 and H_1 hypotheses) on the mass spectrum. We use the fit results to quantify the size of the deviation.
- The incorporation of a ML classifier in order to improve the S/B ratio in the region of interest, to be followed by the same two-fit procedure, and the quantitative evaluation of the deviation.

The goal is to determine if the ML classifier can improve the sensitivity of the search, and quantify any gains, if applicable.

2 Input files

CSV files with the kinematic information for about 6.8M background and 50k signal simulated collisions can be found at `https://cernbox.cern.ch/s/gVPRzIdgRzDtqq5`. The

*`Christos.Leonidopoulos@ed.ac.uk`

background events are split into three files corresponding to different physics processes. The signal sample has been generated for a hypothetical heavy Higgs boson with $m_H = 1$ TeV/c^2. Every entry in these tables corresponds to a separate (signal or background) collision. We will focus on a subset of variables, summarised in Table 1.

Variable	Description
`lep1_pt`	transverse momentum of first lepton (in MeV/c)
`lep2_pt`	transverse momentum of second lepton (in MeV/c)
`fatjet_pt`	transverse momentum of fat-jet (in MeV/c)
`fatjet_eta`	η of fat-jet
`fatjet_D2`	D_2 of fat-jet
`Zll_mass`	invariant mass of dilepton system (in MeV/c^2)
`Zll_pt`	transverse momentum of dilepton system (in MeV/c)
`MET`	transverse missing energy in event (in MeV)
`reco_zv_mass`	invariant mass of dilepton-plus-fatjet ($\ell\ell\, J$) system (in MeV/c^2)
`isSignal`	boolean flag: 0 for background, 1 for signal
`FullEventWeight`	event-weight to normalise the various processes

Table 1: List of subset of kinematic variables contained in CSV files to be used in this project, and brief description.

3 Setup [20%]

cuts, remove the data entirely?

- Plot the kinematic distributions for each process and overlay them [1]. Decide "by eye" on the kind of "square" cuts you want to apply (*e.g.* `lep1_pt` $>$ `100000`, or `fatjet_pt` $>$ `150000`, etc.) in order to make the signal peak more apparent on the `reco_zv_mass` spectrum. It is not necessary to do a thorough selection optimisation at this point, we just want to end up with a set of reasonable kinematic cuts that render the signal more visible [2]. Do *not* apply any kinematic cuts on variable `reco_zv_mass`.

- We will use variable `FullEventWeight` to normalise (*i.e.* inter-calibrate) the relative contributions of the various (signal and background) processes. This simply means that the contributions from the entries in the CSV files need to be scaled up or down, according to a pre-calculated weight (which is related to the cross-section of the corresponding physics process). This is done by passing an array to the "`weights=`" option in method `matplotlib.pyplot.hist`. Make sure you understand the distinction between an *entry* (one separate line of data in a dataframe) and an *event* (the expected weighted contribution of an entry in the final dataset before or after the kinematic selections). Create `reco_zv_mass` distributions for

[1] We have done this already in Week-8, CP12.

[2] Word of caution: The harder you cut on these selection variables, the more difficult it will be to model the background distribution in the next Section.

the combined background and signal samples, and integrate them to find the total number of signal and background *events* before and after kinematic cuts.

4 Fitting and hypothesis-testing [30%]

- Model the `reco_zv_mass` mass spectrum for the signal and background processes separately after the chosen selection cuts with appropriate fitting functions. You may want to try using *e.g.* a single Gaussian for the signal and a polynomial for the background. These suggestions may not work if you have significantly altered ("sculpted") the mass spectrum in your effort to make the signal more visible. You probably want to perform the fit on a subrange of the mass spectrum (see Fig. 1).

- Model the joint signal-plus-background mass spectrum by using a weighted sum of the fit functions implemented in the previous step. Keep the parameters of the signal model (*e.g.* mean and sigma for a Gaussian) fixed to the values determined by the fit on the signal-only spectrum, but with the overall normalisation left as a free parameter. Assume that the background shape is unknown (*i.e.* you can use your function of choice, but all parameters should be free to float in the fit).

- Carry out two fits, assuming H_1 and H_0 hypotheses. Use Wilk's theorem to calculate the statistical significance of the signal-like deviation in the joint mass spectrum.

Important: Because we are dealing with *weighted* events in this project, we need a modified version of the traditional (Pearson's) χ^2 to be used for minimisation. For example, for a binned χ^2 implementation one would need to adjust the observed number of events in bin-i, $N^{(i)}_{\text{observed}}$, and standard deviation, σ_i, to take into account the weights w_j of several entries $j = 1, \ldots, n$ that contribute to the event count for that bin [1], [2]:

$$N^{(i)}_{\text{observed}} \longrightarrow N^{(i)}_{\text{observed}} = \sum_{j=1}^{n} w_j^{(i)}$$

$$\sigma_i^2 = N^{(i)}_{\text{observed}} \longrightarrow \sigma_i^2 = \sum_{j=1}^{n} (w_j^{(i)})^2$$

$$\chi^2 = \sum_i \frac{\left[N^{(i)}_{\text{observed}} - N^{(i)}_{\text{predicted}}\right]^2}{\sigma_i^2} \longrightarrow \chi^2 = \sum_i \frac{\left[\sum_{j=1}^{n} w_j^{(i)} - N^{(i)}_{\text{predicted}}\right]^2}{\sum_{j=1}^{n} (w_j^{(i)})^2}$$

5 Employing a NN classifier [40%]

- Create a NN classifier using the input features listed in Table 1. Do not use `reco_zv_mass` (or `FullEventWeight`), though. Probably the most important decision you will have to make is which dataset to use for the training of the NN

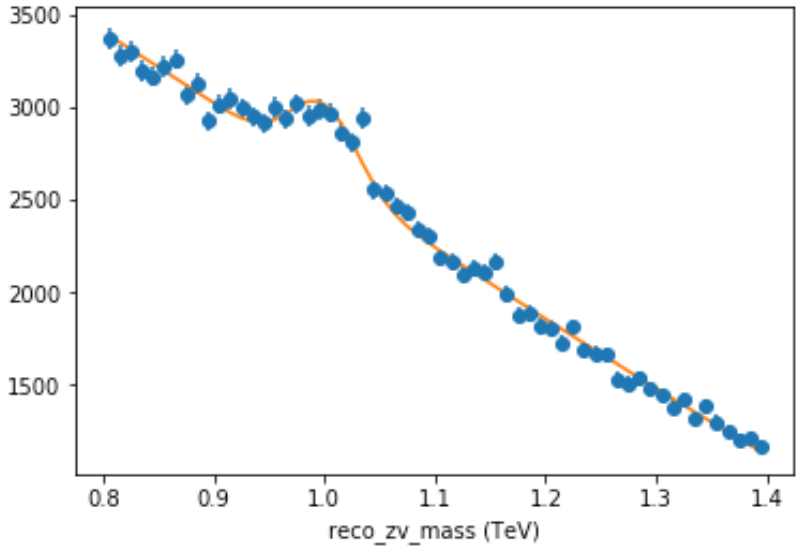

Figure 1: Example fit of the mass spectrum after some kinematic cut selection.

classifier. Should you use the original datasets before any selection cuts? Or the filtered datasets after having applying the selection cuts of the previous Section? Or maybe some intermediate choice with some looser selection cuts? Strictly speaking, there is no "correct" answer here, but some approaches work better than others.

- Make sure the target feature (`isSignal`) is not one of the inputs for the classifier. Use a 50% signal-50% background admixture for the NN training. You are welcome to try any ML algorithm you want, including algorithms that we have not explicitly covered in checkpoints before. Evalute the classifier performance using some of the standard metrics.

- Run the prediction of the model on the full (signal plus background) dataset of your choice[3]. Reject events that have a predicted probability for the `isSignal` target feature less than 50%. Plot the "cleaned" `reco_zv_mass` mass spectrum. Is the signal visible? If not, you may want to ensure that your NN filtering is not looser than the set of selection cuts you applied in the previous Section (*i.e.* use the NN output *on top* of the previously applied selection cuts).

normally predict with different stuff than what you trained with

- Repeat the two-fit procedure, assuming H_1 and H_0 hypotheses, and calculate the statistical significance of the signal-like deviation. Is the result better or worse than the one obtained in the previous Section?

[3]NB: This is not the best practice, as we are not supposed to recycle entries that were used for the NN training. In a real-life situation, we would need larger-statistics datasets.

6 Impact of training sample on classifier [10%]

- Design a new NN classifier, after incorporating `reco_zv_mass` somehow in the training: either by using a training dataset defined after some selection on the variable `reco_zv_mass` has been applied, or by explicitly making it one of the input features of the NN classifier.
- Do not attempt any fits, but do repeat the procedure of the previous Section, and overlay the mass distributions for signal and background. Is the signal more or less visible now compared to the previous Section?
- Discuss the reasons for the behaviour of the ML classifier after incorporating variable `reco_zv_mass` in the NN training, and its impact on the search sensitivity.

References

[1] "Statistical tests with weighted Monte Carlo events", G. Cowan, `http://www.pp.rhul.ac.uk/~cowan/stat/notes/weights.pdf`

[2] "Error analysis with weighted events", G. Cowan, `https://www.pp.rhul.ac.uk/~cowan/stat/notes/errors_with_weights.pdf`

Appendix: Report format

The project report has to be in the jupyter notebook format. It is expected that you are comfortable with writing, running and annotating code within a jupyter notebook.

Important: Annotation and Commentary

It is important that **all** code is annotated and that you provide brief commentary **at each step** to explain your approach. We expect well-documented jupyter notebooks, not an unordered collection of code snippets. You can also include any failed approaches if you provide reasonable explanation.

Unlike weekly checkpoints where you were being guided towards the "correct" answer, this project is by design more open ended. It is, therefore, necessary to give some justification for choosing one method over another.

This is not in the form of a written report so do not provide pages of background material. Only provide a brief explanation for each step. Aim to clearly present your work so that the markers can easily follow your reasoning and can reproduce each of your steps through your analysis.

To add commentary above (or below) a code snippet create a new cell and add your text in markdown format. **Do not** add commentary as a code comment in the same cell as the code.

20% of the mark for each exercise is allocated to coding style and clarity of comments and approach.

Submission Steps

It is important your code is fully functional before it is submitted or this will affect your final mark.

When you are ready to submit your report perform the following steps:

- In Jupyter run `Kernel >Restart & Run All` to ensure that all your analysis is reproducible and all output can be regenerated
- Save the notebook, and close Jupyter
- Tar and zip your project folder if you have multiple files in a working directory. You are free to include any supporting code. Make sure this belongs in the project folder and is referenced correctly in your notebook. Do *not* include any of the input data.
- Submit this file or zipped folder through Learn/Turnitin. In case of problems or if your compressed project folder exceeds 20 MB (first make sure you are not including any CSV files, then) email your submission to Kieran (the course administrator) and me.

Good luck!