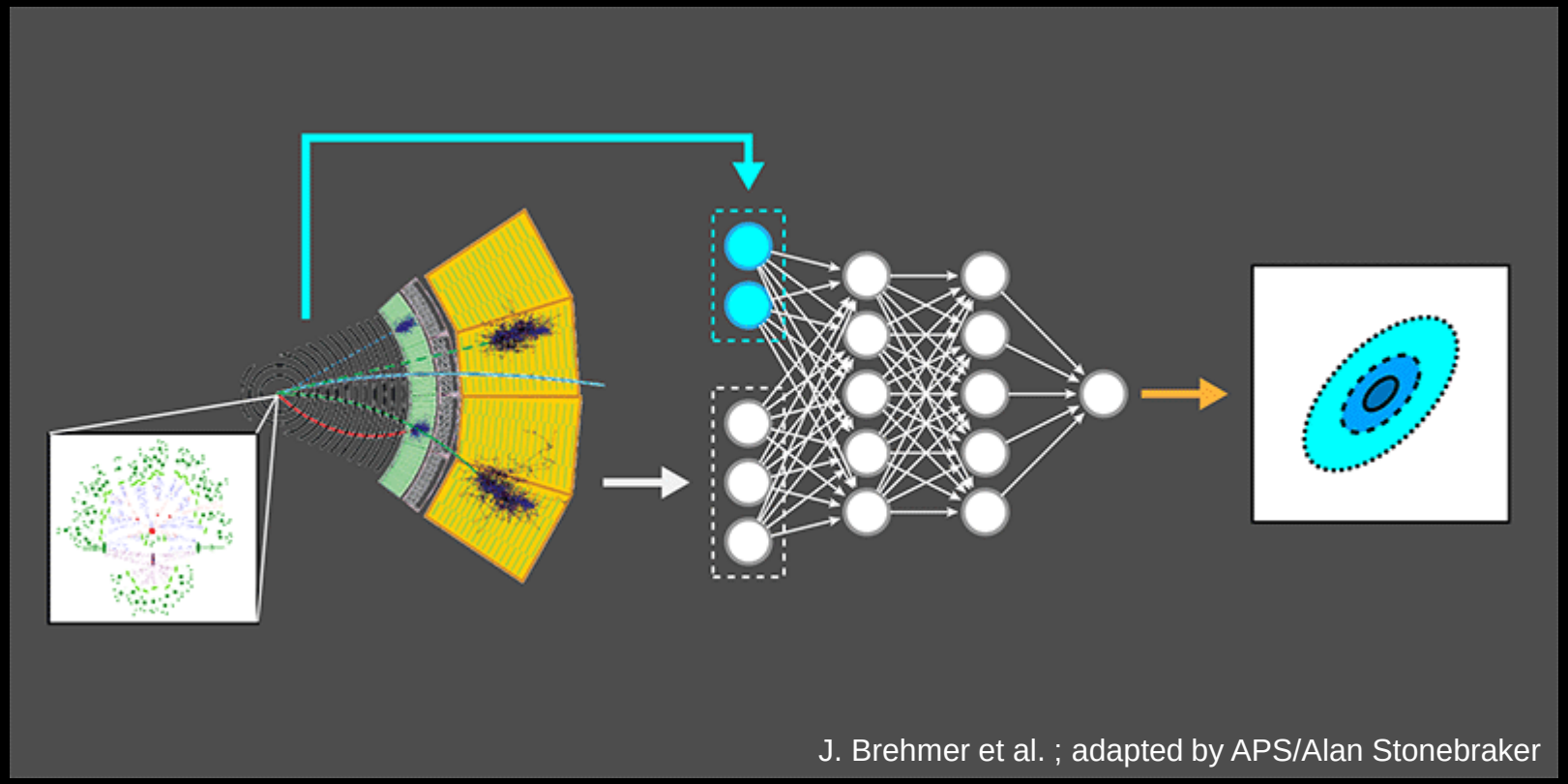


J. Brehmer et al. ; adapted by APS/Alan Stonebraker

Machine Learning for Likelihood-Free Inference

Luc Le Pottier
University of Michigan, CERN ATLAS Group

Outline

- Project Intro
- Project Work
- Future Work
- Project Lessons
- Cultural Experiences

Project Intro

- Working under Dr. Tancredi Carli on ATLAS group data analysis
- Software developer
  - .. Writing tools to simplify the generation and implementation of certain analyses
- Looking for ways to measure **Effective Field Theory** (EFT) parameters
  - EFTs describes physics at an energy scale expanded about $1/\Lambda$, meaning physics at an energy scale $E \ll \Lambda$
- Assuming symmetry of SM we can describe any new LHC physics signatures…
  - Use the 59 dim-six operators $\mathcal{O}_o$, with Wilson coefficients f_o
  - Easy parameterization of many observables
  - **Excellent theory-experiment interface**(1)

(1, 2, 3)

$$\mathcal{L}_{\mathrm{EFT}} = \mathcal{L}_{\mathrm{SM}} + \sum_o \frac{f_o}{\Lambda^2}\mathcal{O}_o$$

Project Intro

- **Likelihood function** allows for EFT parameter calculation
  - Describes the compatibility of data with model parameters
  - must be approximated: difficult to do precisely in high-dim parameter/observable spaces
- Current methods:
  - Matrix Element Method, Optimal observables, naïve Parameter scans, Neural Network classification
  - All trained on event/parameters sample pairs – no extra use of particle physics structures

$$r(x, z|\theta_0, \theta_1) = \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} \qquad t(x, z|\theta_0) = \nabla_\theta \log p(x, z|\theta)\Big|_{\theta_0}$$

- New method: training deep neural networks to approximate the **likelihood function**, using the **joint likelihood ratio** and **joint score** in loss functions.

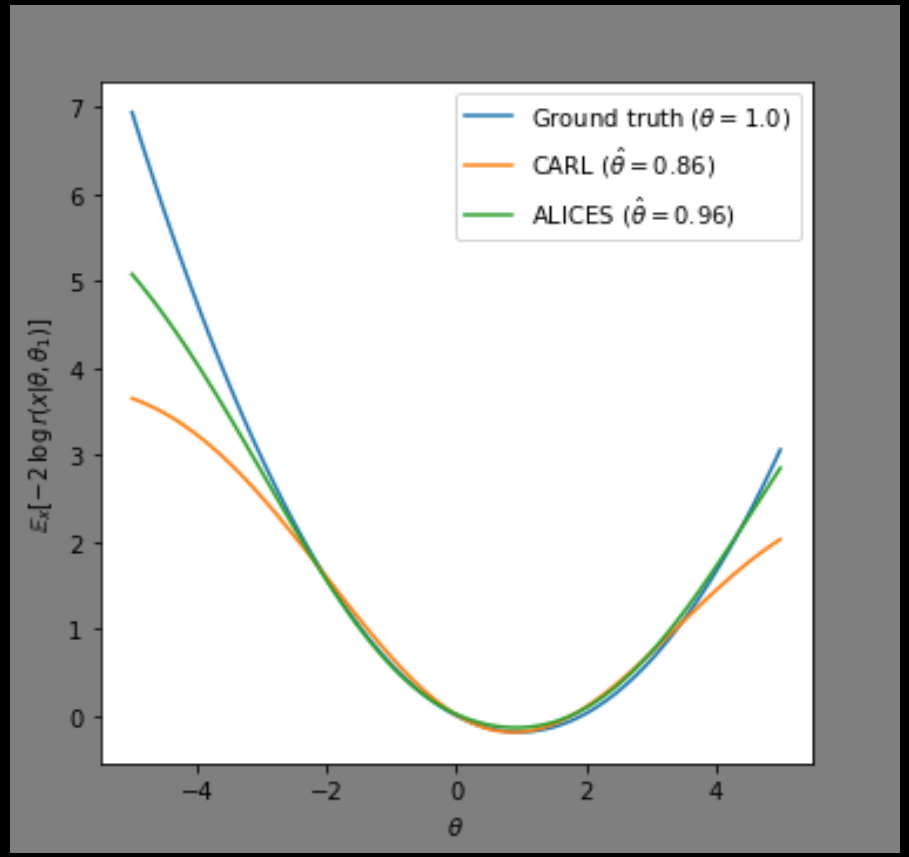
$$L_{\text{ALICE}}\left[\hat{r}(x)\right] = \frac{1}{N}\sum \Big[\; s(x_i, z_i|\theta_0, \theta_1) \log\left(\hat{s}(x_i)\right) + (1 - s(x_i, z_i|\theta_0, \theta_1)) \log(1 - \hat{s}(x_i))\Big]$$

ALICE Method Cross-Entropy Estimator-based loss function

Finished Work

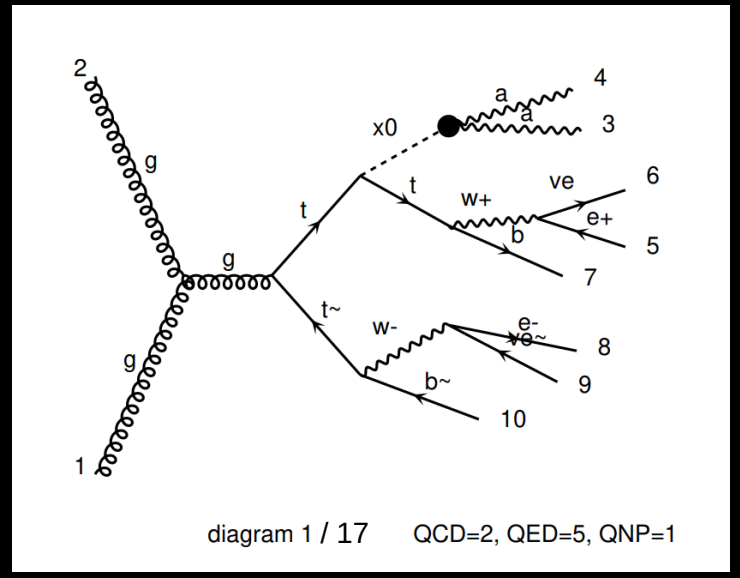
- Using the MadMiner machine learning toolkit [1, 2, 3] , with an implementation of the ALICE algorithm, claimed to be one of the more ‘sample-efficient’ algorithms (good for testing and debugging).[2]

- Problem Introduction (January)
  - Wrote a python module for testing ML algorithms on arbitrary-dimensional ‘toy’ processes
  - Validated precision results of various algorithms using the tractable likelihoods of these toy processes
  - Made clear that ALICES is a good algorithm to prototype new processes with



Finished Work

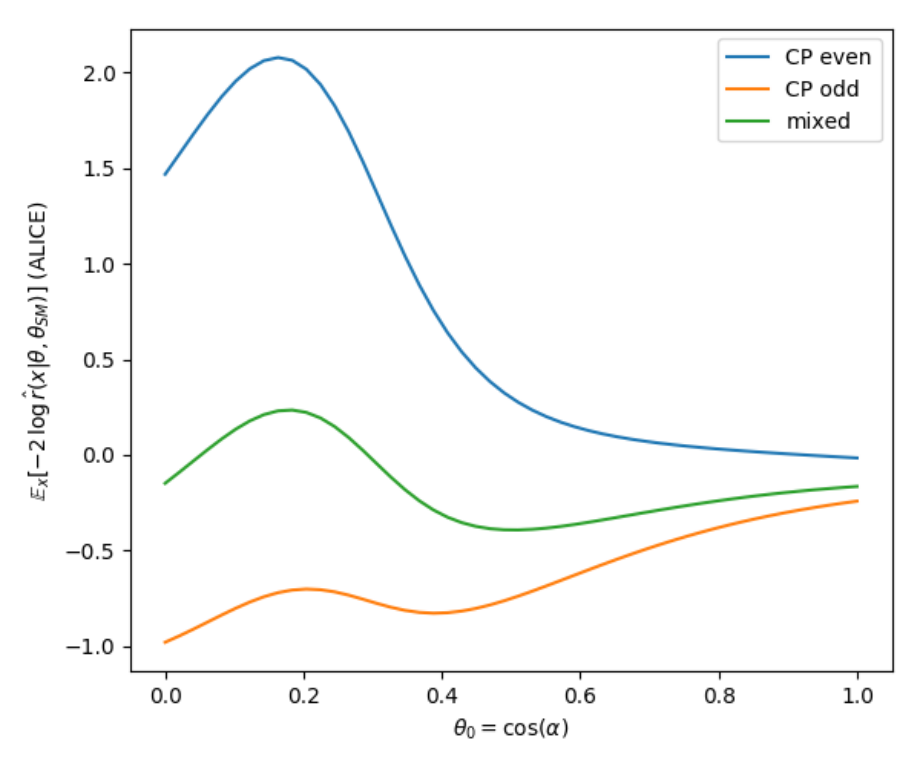
- ttH Process joint-likelihood ratio estimation (February – March)
  - Measuring whether process has Charge conjugation-Parity (CP) symmetry (antiparticles + flipped coordinates)
  - 2 observables, one parameter in this case
    - x_0, pseudo-scalar observable, Higgs boson in SM
    - Delta_t_tbar observable, the angle between the t and tbar jets
    - Cos(alpha), parameter, parameterization of CP sym.
  - Promising interaction:
    - mass of constituent particles
    - ttH coupling strength



(one) Leading order Feynman

Slight Problems

- ttH CP parity parameterization is limited to the range [0, 1], with simulation benchmarks at the boundary points
- Results in strange predicted likelihood functions



- Assuming a gaussian likelihood function, extrapolated variances of log-likelihood are terrible
- Determined after hundreds of different deep neural net configurations and sample augmentations that a new statistical approach is needed
- Will likely receive attention from stats PHD in the near future

Finished Work

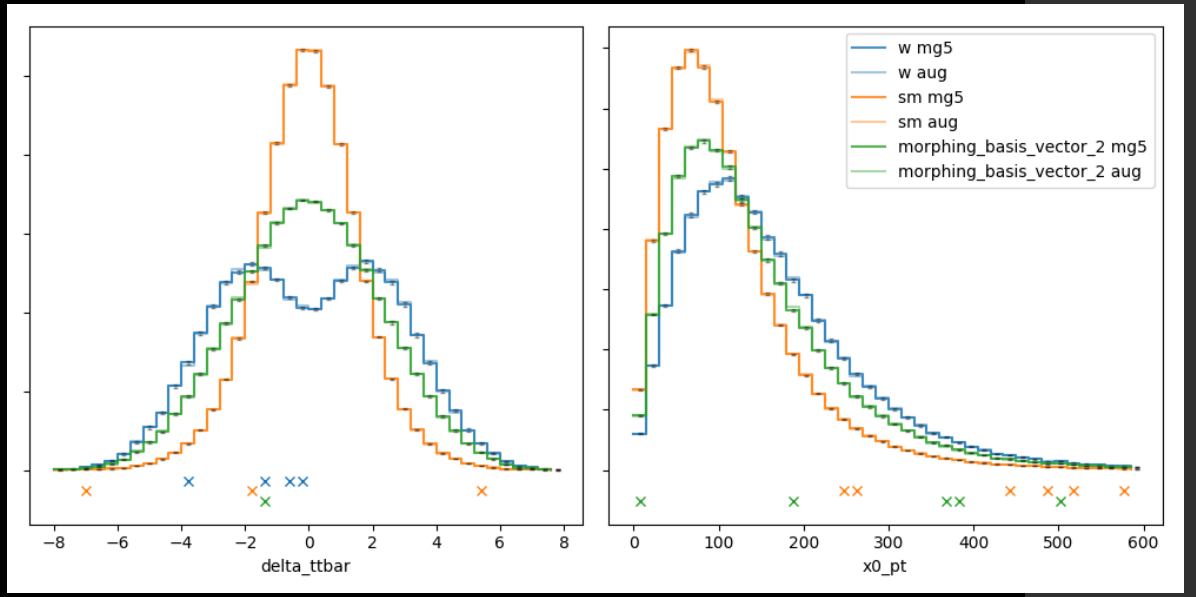
- Authored a python module and linux CLI utility
- Provides a wrapper for
  - Complex environment setup for madminer/madgraph
  - Process backend setup
  - Generation and storage of a wide array of samples, models, and evaluations
- Allows for near-instant prototyping of the `madminer` framework with arbitrarily complex processes
  - Helps a lot with finding a suitable set of ML parameters for a given process

```
llepotti@atlas-t3-ubc : /data/llepotti
$ mmsc -h
usage: run.py [-h] [-CD CARD_DIRECTORY] [-LL LOG_LEVEL]
              [-MLL MADMINER_LOG_LEVEL]
              NAME BACKEND {ls,simulate,plot,augment,train,evaluate} ...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                       %%
%%  MM_SCRIPTING_UTIL COMMAND LINE PROGRAM                               %%
%%                                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                       %%
%%  The cmd line program for the mm_scripting_util class.                %%
%%  Gives (almost) all of the functionality of the base class,           %%
%%  in a single (great) command line program.                            %%
%%  Tips:                                                                %%
%%   - always specify NAME, BACKEND, and then the command you'd          %%
%%     like to run, otherwise you'll have a bad time                     %%
%%   - For more in-depth documentation, see the python code itself       %%
%%     (which you definitely have documented if you're running this)     %%
%%                                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                       %%
%%  Fully avaliable BACKEND specifications in your current directory:   %%
%%   - tth                                                               %%
%%   - VBF_higgs                                                         %%
%%                                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

positional arguments:
  NAME                  name for overall sample
  BACKEND               backend; default choices
  {ls,simulate,plot,augment,train,evaluate}

optional arguments:
  -h, --help            show this help message and exit
  -CD CARD_DIRECTORY, --CARD-DIRECTORY CARD_DIRECTORY
  -LL LOG_LEVEL, --LOG-LEVEL LOG_LEVEL
  -MLL MADMINER_LOG_LEVEL, --MADMINER-LOG-LEVEL MADMINER_LOG_LEVEL
```

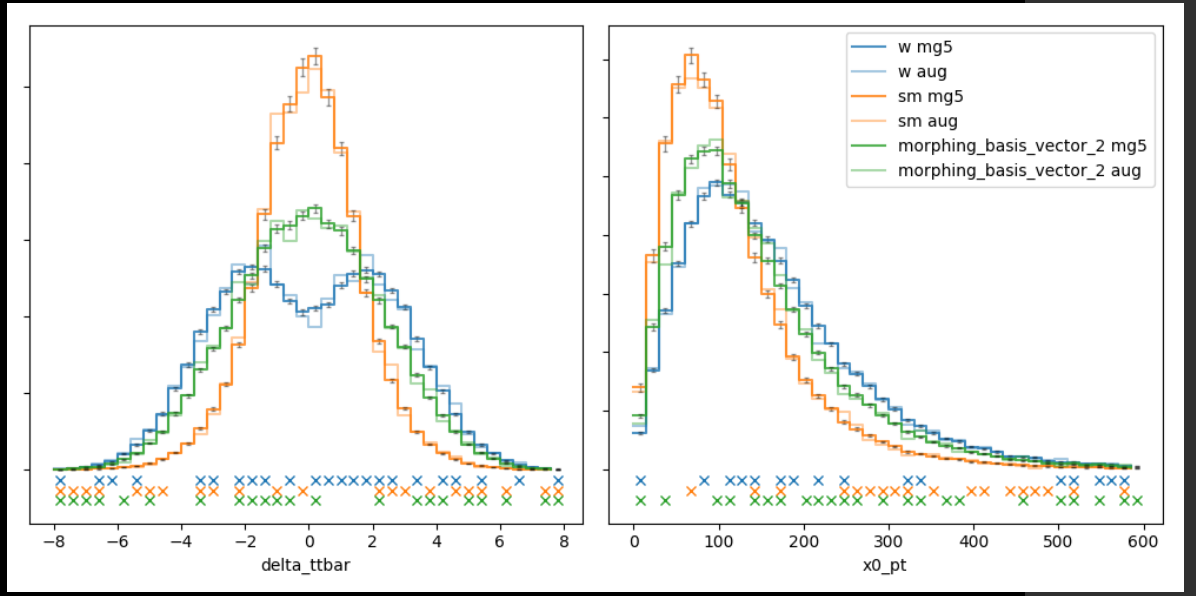
```
In [1]: import mm_scripting_util as mm

In [2]: temp_miner = mm.core.miner(
   ...:     'miner_name',
   ...:     'miner_backend'
   ...: )
```

Finished Work

- quick scripting and safe setup of large batch jobs (full control of logging/exception handling => quicker debug)
- many batch jobs to determine ideal sample augmentation ratios (generated : augmented events)
- Many integrated visualization/plotting tools
- See: good vs. poor sample augmentation

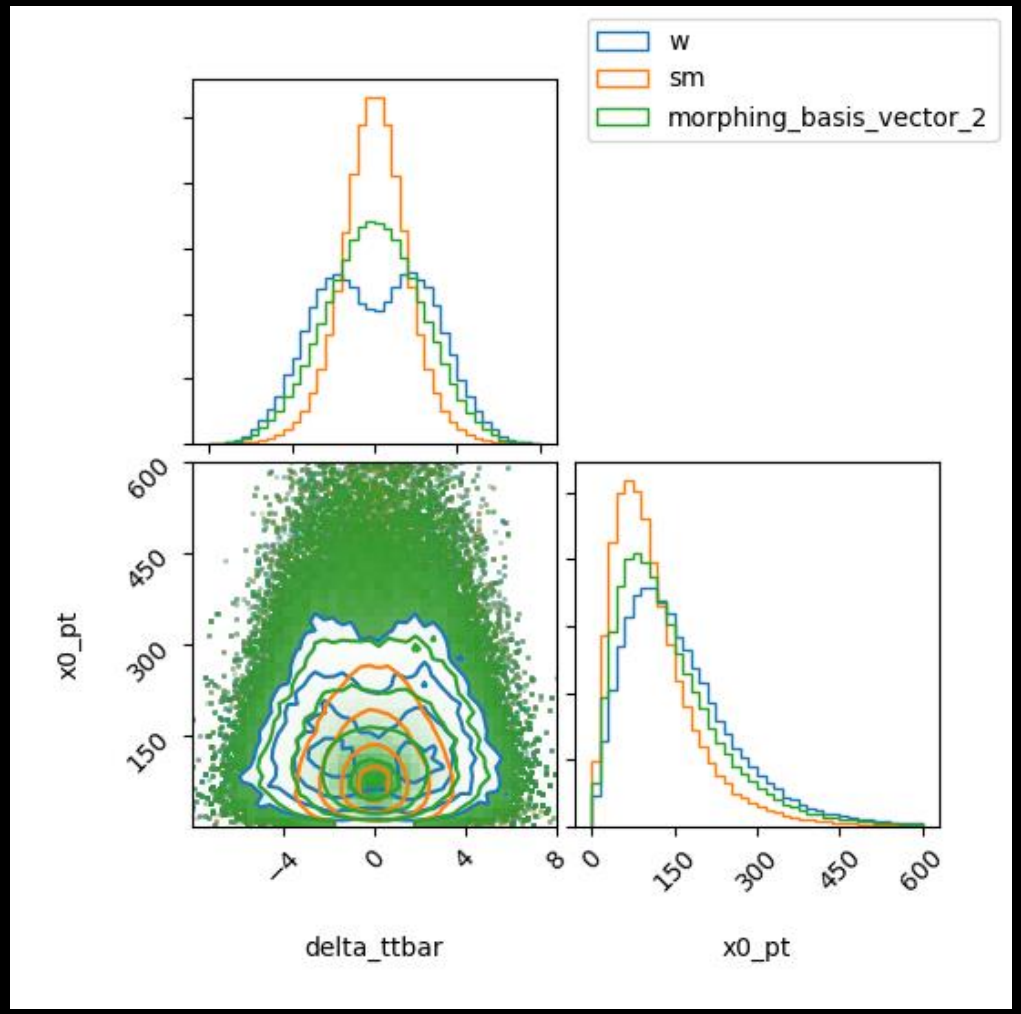


Good sample augmentation

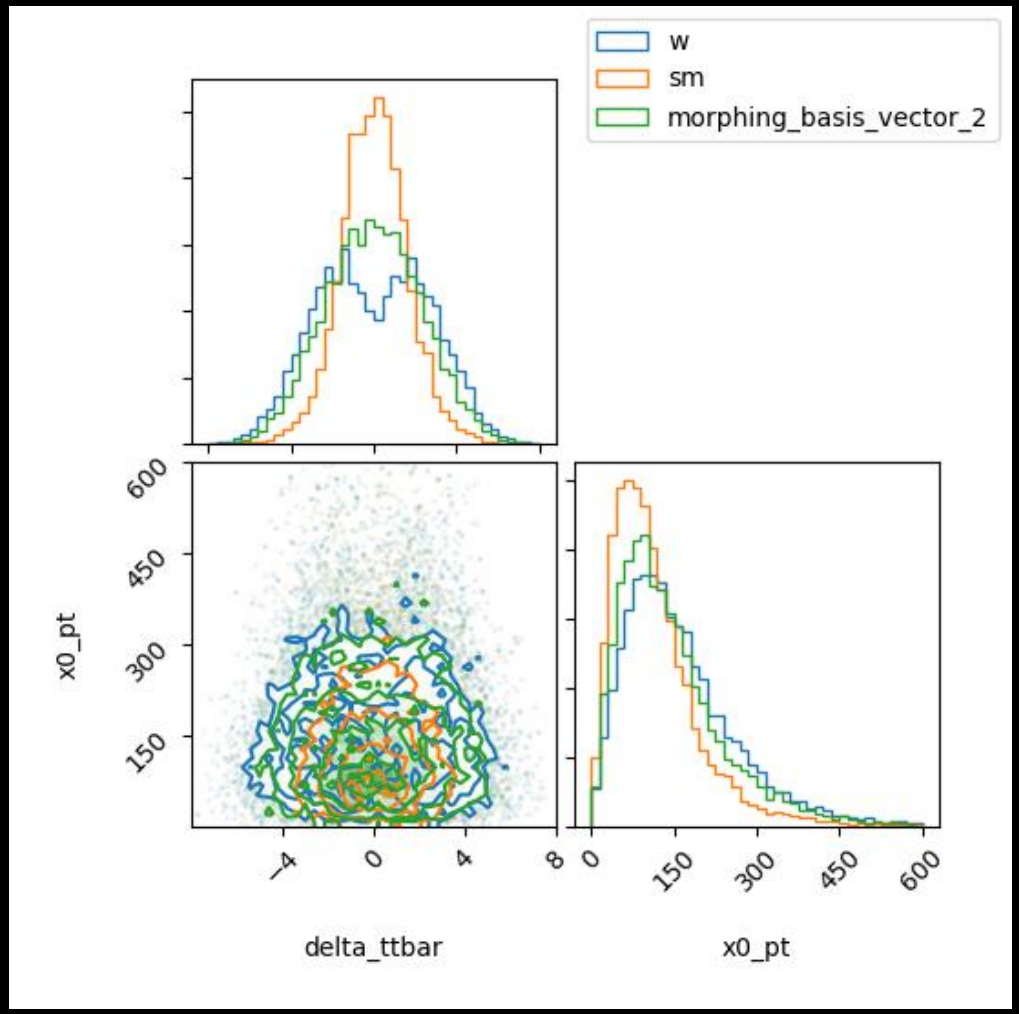


Poor sample augmentation

Progress thus far



Good sample augmentation



Poor sample augmentation

Future Work

- Update module to work with recently-released madminer version
- Finish (majority) of documentation
- Add 4-5 more ‘sample processes’ to demonstrate applicability of the program to arbitrary processes with minimal effort
- Perform and write up a short study on how to quickly find optimal ML parameters for a new problem (lots of research wrt this, but not much written down)

Luckily, I have until May 10th to do these things!

Project Lessons

- Software development
  - Architecture, OO design of large modules
  - Debugging in Linux environment
  - Server resource management
  - Environmental management
  - General programming skill in python/bash (~ 50k lines of code committed over 3 repos)
  - Writing, reading, and gen. management of physics and ML utilities (madgraph, pytorch, LHE/LHA files, etc.)
- BSM HEP models
  - Useful characterizations of HEP models for BSM physics
  - How to find useful parameters for these models
- Research ideas
  - Isolating important parameters in a high-dimensional system

Culture!

Zz in cully

Vallee blanche in Chamonix

Hiking in the jura

Questions?

References

1. Brehmer, Johann, et al. “A Guide to Constraining Effective Field Theories with Machine Learning.” Physical Review D, vol. 98, no. 5, 2018, doi:10.1103/physrevd.98.052004.
2. Brehmer, Johann, et al. “Likelihood-free inference with an improved cross-entropy estimator,” arXiv:1808.00973v1 [stat.ML], 2 Aug 2018.
3. Brehmer, J., Louppe, G., Pavez, J., & Cranmer, K. (2018). Mining gold from implicit models to improve likelihood-free inference. CoRR, abs/1805.12244.