Apprentice for Event Generator Tuning

Mohan Krishnamoorthy[1]
Holger Schulz[2], Xiangyang Ju[3], Wenjing Wang[3],Sven Leyffer[1], Zachary Marshall[3], Stephen Mrenna[4], Juliane Muller[3] , James B. Kowalkowski[4]

[1]Argonne National Laboratory
[2]Department of Computer Science, Durham University
[3]Lawrence Berkeley National Laboratory
[4]Fermi National Accelerator Laboratory

May 18, 2021

25th International Conference on Computing in High-Energy and Nuclear Physics (2021)

Outlines

Apprentice: Project Details

Problem of Tuning Monte Carlo Event Generators

Solutions from Apprentice

Summary

Apprentice: Project Details

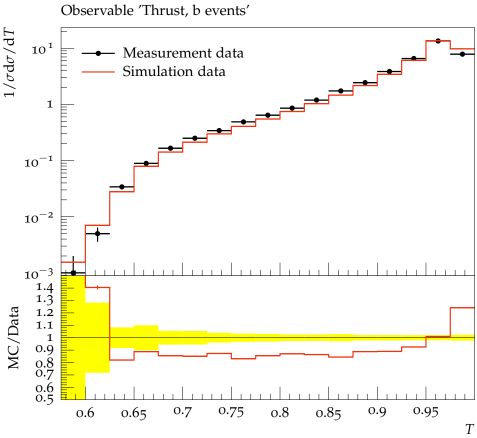
- Download from `https://github.com/HEPonHPC/apprentice`
- Install Apprentice locally by executing

```
$git clone --recurse-submodules
                git@github.com:HEPonHPC/apprentice.git
$cd apprentice
$pip install .
$cd pyoo
$pip install .
```

- Can be used for
  1. building polynomial and rational approximations
  2. solving χ^2 minimization problem
  3. event generator tuning by automatic selection of observable weights
- Successor to PROFESSOR

Tuning Monte Carlo Event Generators

- Events are observed in nature
- Events have properties for which data is collected
- Events are organized as bins in a histogram



Tuning Monte Carlo Event Generators

$$\min_{\boldsymbol{p}\in\Omega} \sum_b \frac{(MC_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta MC_b(\boldsymbol{p})^2 + \Delta\mathcal{R}_b^2}.$$

Tuning Monte Carlo Event Generators

$$\min_{\boldsymbol{p}\in\Omega} \sum_{b} \frac{(MC_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta MC_b(\boldsymbol{p})^2 + \Delta \mathcal{R}_b^2}.$$

- However, performing MC simulations can be very expensive

$$MC_b(\boldsymbol{p}) \approx r_b(\boldsymbol{p}) \text{ and } \Delta MC_b(\boldsymbol{p}) \approx \Delta r_b(\boldsymbol{p}), \quad \forall b. \tag{1}$$

Tuning Monte Carlo Event Generators

$$\min_{\boldsymbol{p}\in\Omega} \sum_b \frac{(MC_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta MC_b(\boldsymbol{p})^2 + \Delta\mathcal{R}_b^2}.$$

- However, performing MC simulations can be very expensive

$$MC_b(\boldsymbol{p}) \approx r_b(\boldsymbol{p}) \text{ and } \Delta MC_b(\boldsymbol{p}) \approx \Delta r_b(\boldsymbol{p}), \quad \forall b. \tag{1}$$

$$\min_{\boldsymbol{p}\in\Omega} \sum_b \frac{(r_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta r_b(\boldsymbol{p})^2 + \Delta\mathcal{R}_b^2}. \tag{2}$$

Tuning Monte Carlo Event Generators

$$\min_{\boldsymbol{p}\in\Omega} \sum_b \frac{(MC_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta MC_b(\boldsymbol{p})^2 + \Delta\mathcal{R}_b^2}.$$

- However, performing MC simulations can be very expensive

$$MC_b(\boldsymbol{p}) \approx r_b(\boldsymbol{p}) \text{ and } \Delta MC_b(\boldsymbol{p}) \approx \Delta r_b(\boldsymbol{p}), \quad \forall b. \tag{1}$$

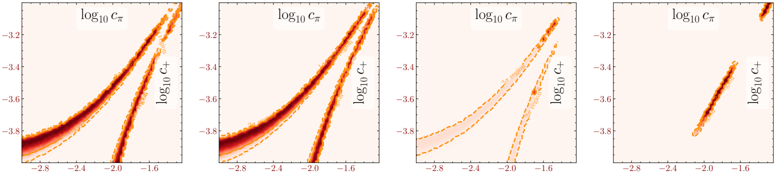
$$\min_{\boldsymbol{p}\in\Omega} \sum_b \frac{(r_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta r_b(\boldsymbol{p})^2 + \Delta\mathcal{R}_b^2}. \tag{2}$$

- Goal is to tune the parameter $\boldsymbol{p}$ for a large number of events

$$\min_{\boldsymbol{p}\in\Omega} \sum_{\mathcal{O}\in\theta} w_{\mathcal{O}} \sum_{b\in\mathcal{O}} \frac{(r_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta r_b(\boldsymbol{p})^2 + \Delta\mathcal{R}_b^2}. \tag{3}$$

Polynomial and Rational Approximations

- Build polynomial or rational approximation for a given order of numerator and/or denominator polynomials
- Input for approximation: parameters $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_{N_{\mathrm{ens}}}]$, simulation output $\mathbf{y} = [y_1, \ldots, y_{N_{\mathrm{ens}}}]$ order of numerator (m) and denominator (n)
- Input formats: HDF5 file, directory of Yoda files, CSV of $\mathbf{P}$ and $\mathbf{y}$, or $\mathbf{P}$ as array of array and $\mathbf{y}$ as array
- Output: JSON file
- A starter script is available in **bin/app-build** in apprentice. See `https://doi.org/10.1016/j.cpc.2020.107663` for more details



(a) Full simulation (b) Pole-free rational approx (c) Non-pole-free rational approx (d) Polynomial approx

Solving χ^2 Minimization

$$\boldsymbol{p}^* = \underset{\boldsymbol{p}\in\Omega}{\operatorname{argmin}} \quad \chi^2(\boldsymbol{p},\boldsymbol{w}) = \sum_{\mathcal{O}\in\mathcal{S}_\mathcal{O}} w_\mathcal{O} \sum_{b\in\mathcal{O}} \frac{(r_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta r_b(\boldsymbol{p})^2 + \Delta\mathcal{R}_b^2}.$$

- Input: Experimental data JSON with $\mathcal{R}_b$ and $\Delta\mathcal{R}_b, \forall b$; Approximation and error approximation JSON with coeffecients of $r_b(\boldsymbol{p})$ and $\Delta r_b(\boldsymbol{p}), \forall b$; and weights file with $w_\mathcal{O}, \forall\mathcal{O}$
- Output: $\boldsymbol{p}^*$
- Multiple SciPy solvers supported
- Optimization can be performed for a smaller bin slice
- Optimization can be started from multiple starting points
- Optimization from multiple starting points can be performed in parallel
- A starter script is available in **bin/app-tune2** in apprentice.

Event Generator Tuning

$$\boldsymbol{p}^* = \underset{\boldsymbol{p}\in\Omega}{\operatorname{argmin}} \quad \chi^2(\boldsymbol{p},\boldsymbol{w}) = \sum_{\mathcal{O}\in\mathcal{S}_{\mathcal{O}}} w_{\mathcal{O}} \sum_{b\in\mathcal{O}} \frac{(r_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\Delta r_b(\boldsymbol{p})^2 + \Delta\mathcal{R}_b^2}.$$

Automatic Selection of Weights: Bilevel Optimization

$$\underset{\boldsymbol{w} \in [0,1]}{\text{minimize}}\ h(\hat{\boldsymbol{p}}_{\boldsymbol{w}}, \boldsymbol{w})$$

subject to

$$\hat{\boldsymbol{p}}_{\boldsymbol{w}} = \underset{\boldsymbol{p} \in \Omega}{\text{argmin}} \sum_{\mathcal{O} \in \theta} w_{\mathcal{O}} \sum_{b \in \mathcal{O}} \frac{(r_b(\boldsymbol{p}) - \mathcal{R}_b)^2}{\sigma_b^2}.$$

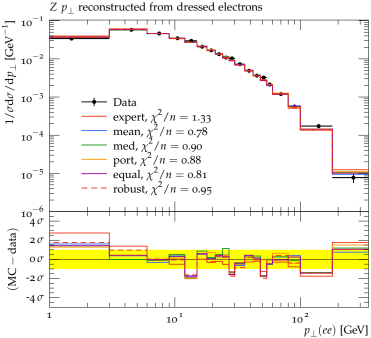
- $h(\cdot, \cdot)$: Merit function
  - portfolio function
  - mean of the scoring function (meanscore)
  - median of the scoring function (medianscore)
- Interface scripts are available at **apprentice/pyoo/bin/pyoo-run-portfolio**, **apprentice/pyoo/bin/pyoo-run-medianscore**, and **apprentice/pyoo/bin/pyoo-run-medianscore**.
- See `https://arxiv.org/abs/2103.05751` for more details

Automatic Selection of Weights: Robust Optimization

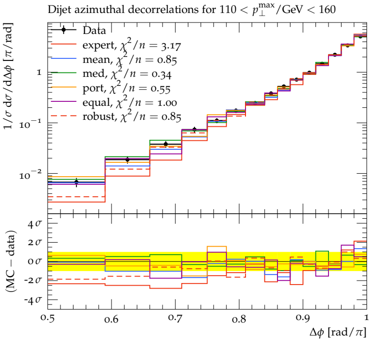
$$\underset{\boldsymbol{w}\in[0,1],\boldsymbol{p}\in\Omega}{\text{minimize}} \sum_{\mathcal{O}\in\theta} w_{\mathcal{O}} \sum_{b\in\mathcal{O}} \underset{d_b\in\mathcal{U}_b}{\text{maximize}} \left(r_b(\boldsymbol{p}) - d_b\right)^2 .$$

- Because of the square function, the problem can be converted to a non-linear optimization problem
- Optimization problem solved using IPOPT with AMPL solver interface (ASL) using PYOMO or cyipopt
- Hyperparameter used in constraint to the optimization problem to avoid tivial solutions of all weights being zero.
- Interface scripts is available at **apprentice/pyoo/bin/pyoo-robopt**
- See `https://arxiv.org/abs/2103.05751` for more details

Result of Event Generator Tuning



(a) p_T^Z



(b) Dijet decorr

Summary

- Apprentice can be used for solving the problems of
  - building polynomial and rational approximations
  - solving χ^2 minimization problem
  - event generator tuning by automatic selection of observable weights using bilevel and single level robust optimization
- Future work includes:
  - Implementing an iterative workflow to solve the tuning problem using rational approximations within a trust region algorithm
  - Filtering to remove or fix the parameter dimensions that are invariant to the data
  - Global optimization of the non-linear χ^2 minimization problem

Thank you

mkrishnamoorthy@anl.gov