Machine Learning Efforts in SHERPA

1. Neural Importance Sampling [2001.05478] $\rightarrow$ increase unweighting efficiency
2. Neural Rejection Sampling [2109.11964] $\rightarrow$ reduce #calls to matrix elements

Timo Janßen

Institut für Theoretische Physik, Georg-August-Universität Göttingen

ACAT 2021

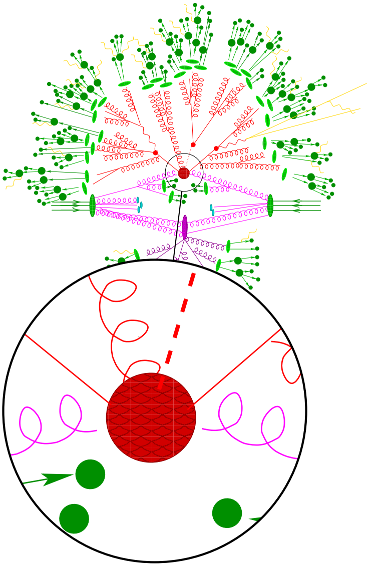
GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Introduction

see also talks from Anja Butter (Monday) and Joshua Isaacson (Thursday)

Monte Carlo Event Generators like HERWIG, PYTHIA and SHERPA produce pseudo-data that is post-processed by the experiments

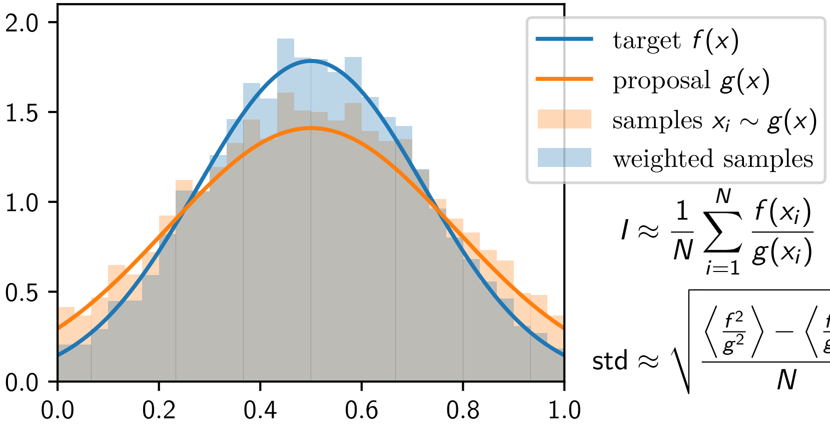
- phase-space points are sampled from a proposal distribution constructed from physics knowledge
- rejection sampling for unweighting
- HL-LHC forces us to improve efficiency
- machine learning might be helpful



This talk: How to make event generation more efficient

How to generate weighted events

importance sampling:



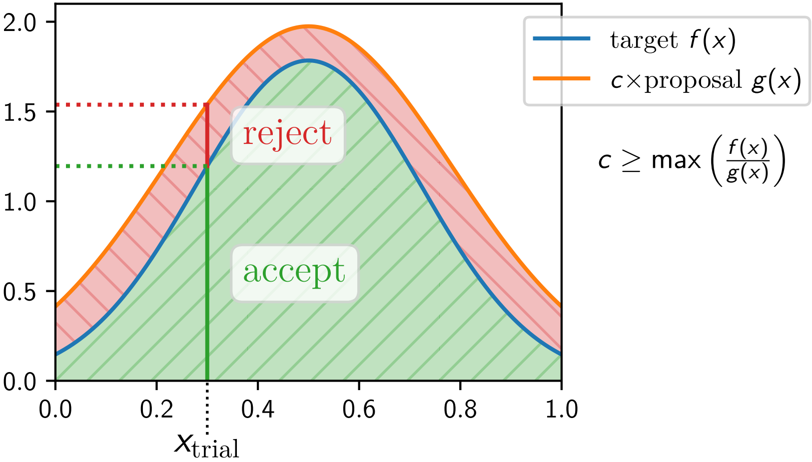
$$I \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)}$$

$$\text{std} \approx \sqrt{\frac{\left\langle \frac{f^2}{g^2} \right\rangle - \left\langle \frac{f}{g} \right\rangle^2}{N}}$$

HEP example: Breit-Wigner distribution for resonances

How to generate unweighted events

rejection sampling (hit-or-miss):

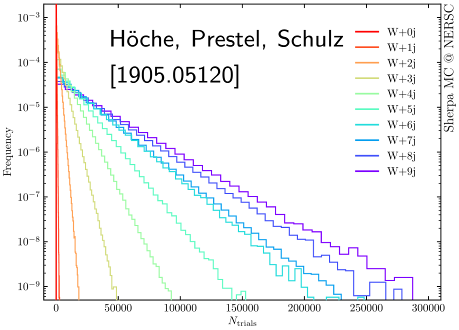


$c \geq \max\left(\frac{f(x)}{g(x)}\right)$

Unweighting efficiency: $\epsilon = \frac{N_{\text{accepted}}}{N_{\text{trials}}} \approx \frac{\left\langle \frac{f}{g} \right\rangle}{c}$

Motivation for increasing the unweighting efficiency

- observation: high-multiplicity processes suffer from small unweighting efficiencies
- at the same time, the matrix elements of these processes can be expensive to evaluate
- these are highly relevant processes for the HL-LHC



---

idea: use neural networks to improve proposals → increase unw. efficiency

Our requirements:

- full phase space coverage
- convergence to the target distribution
- general method, lending itself to automation
- uncorrelated events

Increasing the unweighting efficiency using Deep Learning

Bothmann, TJ, Knobbe, Schmale, Schumann [2001.05478]

Our tool of choice: Normalizing Flows

(see also talk by Humberto Reyes-González right after this)

- provide bijective mapping with analytic inverse
- diffeomorphisms parameterized by Neural Networks
- training is comparatively stable
- *Neural Importance Sampling* (Müller et al. [1808.03856])
- *Neural Spline Flows* (Durkan et al. [1906.04032])

$$x \rightarrow \text{Flow } f(x) \rightarrow z \rightarrow \text{Inverse } f^{-1}(z) \rightarrow x$$

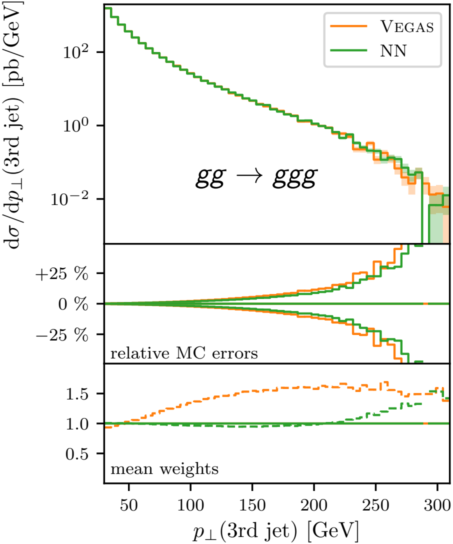
- idea: NF to remap input variables of existing phase-space generator
  $\rightarrow$ reduce mismatch between proposal and target
- direct replacement of VEGAS algorithm (LePage, 1978)
  $\rightarrow$ better handling of correlations

Exploring phase space with Neural Importance Sampling

Bothmann, TJ, Knobbe, Schmale, Schumann [2001.05478]

	unweighting efficiency	
	$gg \to ggg$	$gg \to gggg$
Uniform	3 %	3 %
VEGAS	28 %	32 %
NN	64 %	49 %

- method is able to improve proposal, increase unw. eff.
- less gain for higher multiplicities
- training is expensive
- similar results in other study looking at $\{W, Z\} + n$ jets @{LO, NLO} (Gao et al. [2001.10028])



Motivation for reducing the number of calls to the ME

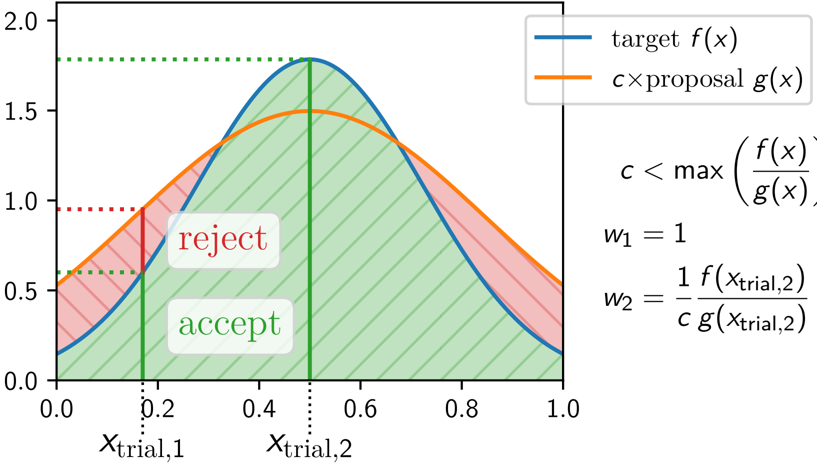
- #Feynman diagrams grows factorially with #particles
  $\rightarrow$ high-multiplicity matrix elements are really expensive
- we have to evaluate the matrix element for each trial event in unweighting
- if unw. eff. is small, ME evaluation time is a bottleneck

Idea:

- reduce event generation time by reducing the number of calls to the ME
  $\rightarrow$ use a fast surrogate (our choice: neural networks)
  $\rightarrow$ see also talk by Henry Truong (Monday)
- correct all errors from the approximation in a 2nd unweighting step
  $\rightarrow$ method is unbiased by design

Partial unweighting

rejection sampling with overweights:



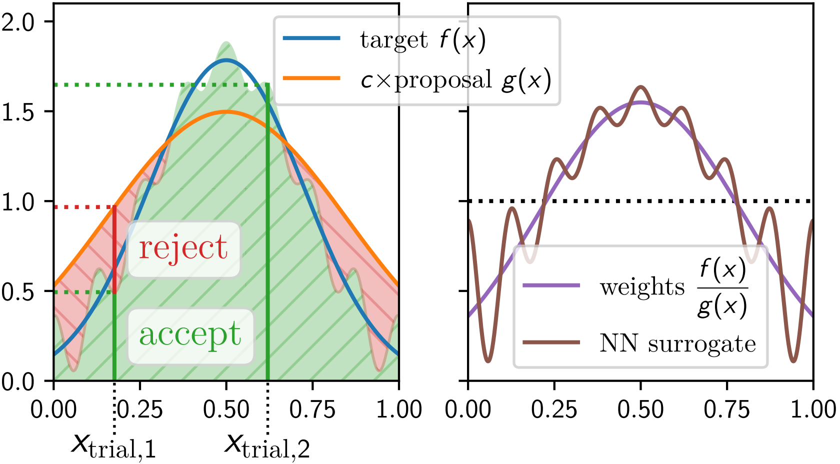
$$c < \max\left(\frac{f(x)}{g(x)}\right)$$

$$w_1 = 1$$

$$w_2 = \frac{1}{c}\frac{f(x_{\text{trial},2})}{g(x_{\text{trial},2})}$$

Overweights are allowed by default in SHERPA (and other generators)

Surrogate unweighting

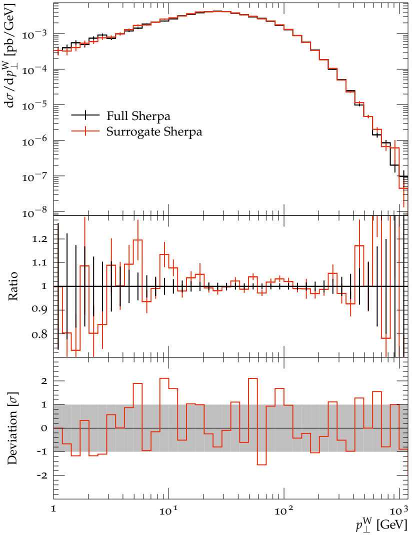


- surrogate should be fast and approximate the weights well
- have to correct for the wrong accept/reject probabilities
  $\rightarrow$ 2nd unweighting against true target for all accepted points

Results $Z+4$jets, $W+4$jets, $t\bar{t}+3$jets @ LHC

Danziger, TJ, Schumann, Siegert [2109.11964]

- no systematic deviations (190 observables)
- surrogate up to 40000 times faster than full weight
- overall speedup up to 10 times
- overall slower for some subprocesses (with simpler ME's)
- if maximum reduced too much there are many (and large) overweights
- NN training is almost for free
- generalisation to NLO is straightforward



$W+4$jets example: $dd \to e^- \bar{\nu}_e ggdu$

Summary & Outlook

Two compatible unbiased methods to increase event generation efficiency:

Neural Importance Sampling

- increase unw. eff. by remapping proposal
- proof of principle for LO gluon scattering
- performance worsens with increasing multiplicity
- training needs to be more efficient

Neural Rejection Sampling

- make unweighting faster by using NN surrogates
- 2nd unweighting step guarantees unbiasedness
- up to 10 times faster for LHC processes
- can serve as an interface for ME surrogates

Other efforts include:

- gpu matrix elements (Bothmann et al. [2106.06507], talk by Joshua Isaacson on Thu)
- reduce fraction of negative weights @NLO (Danziger, Höche, Siegert [2110.15211], talk by Andreas Maier on Thu)