

Institut für Theoretische Physik

Exploring phase space with Neural Importance Sampling



8th annual conference on Large Hadron Collider Physics Tools parallel session 25th May 2020



Enrico Bothmann*, Timo Janßen, Max Knobbe, Tobias Schmale, Steffen Schumann [2001.05478]

Background

Neural Importance Sampling

Why should we improve MC sampling efficiency?

- Reliable automated methods do exist ...
 - → but we want to study high multiplicities
 - → issue: many events with tiny weights
 - → driver for MC event generation cost
 - → room for improvement
- discrepancy between resource needs and budget at HL-LHC expected
 - → 2026 only 2 PhDs from now
 - → sampling can be part of the solution
 - → (other ideas: more GPU/TPU/... usage [European Strategy input 2020, Matteo's talk tomorrow], positive resampling [2005.09375], improve scaling for (Exascale) HPC [1905.05120], manual optimisation of single processes ...)

number of trial events needed for a single unweighted event in W+n jets production:



 \rightarrow Can we come up with a smarter sampling?

Background

Neural Importance Sampling

MC refresher

MC integral estimate

$$I = \int_{\Omega} f(x) dx \approx \frac{V}{N} \sum_{i=1}^{N} f(x_i) = V \langle f \rangle = \hat{I}$$

$$\delta I \approx \sqrt{\operatorname{Var}(\hat{I})} = V \frac{\sigma_N}{\sqrt{N}}$$
$$\sigma_N^2 = \frac{1}{N-1} \sum_{i=1}^N (f(x) - \langle f \rangle)^2$$

Reduce variance by importance sampling

$$I = \int_{\Omega} \frac{f(x)}{g(x)} g(x) dx \approx \frac{V}{N} \sum_{i=1}^{N} \frac{f(x_i)}{g(x_i)} = V \langle f/g \rangle = \hat{I}$$

with $x_i \sim g(x)$; variance reduced if g similar to f

construct g iteratively from building blocks, one for each singularity/resonance \rightarrow multi-channel importance sampling:

$$g(x) = \sum_{j=1}^{N_c} \alpha_j g_j(x)$$
 with $\sum_{j=1}^{N_c} \alpha_j =$

- even with a good multi-channel ...
 - channels correspond ~ denominator of \rightarrow each squared diagram, but misses numerators and interference terms
 - arbitrary phase-space cuts might apply \rightarrow
- combine channels that reflect the singularities with an optimiser for the unknown non-singular structure (usually VEGAS [Lepage CLNS-80/447) [Ohl hepph/9806432]
- VEGAS adapts well when dimensions factorise:





How that's used in practice

goals:

- → event sample with narrow weight distribution (i.e. $f \sim g$)
- > without large-weight outlier for efficient unweighting
- 2-step approach
 - → integration/optimisation phase
 - → set-up multi-channel for process
 - \rightarrow optimise channel weights and VEGAS grids
 - → monitor maximum event weight w_{max}
 - → event generation phase
 - → generate weighted events from the (now frozen) multi-channel
 - → if required, unweight events (accept with probability w/w_{max})
 - → write (un)weighted event sample to disk



Background

Neural Importance Sampling

NN for sampling in HEP MC: overview

- train BDT/GAN to sample phase-space [Bendavid 1707.00028]
 - → application to non-separable high-dimensional toy functions
- train DNN to sample phase-space [Klimek Perelstein 1810.11509]
 - → application to 3-body decay with resonances & ee → qqg
 - → improve efficiency over standard MC ~ O(10) over MG5
- g given by Jacobian of var. transf. given by NN, train to be close to f
- even if g is not perfect, we can rest assured physics is still the same (only efficiency is lower)
- caveats
 - → gradient/determinant expensive, in general $O(d^3)$
 - → no combination with multi-channel

Requirements for any new sampler

- correctness: samples must converge to true target distribution for $N \to \infty$
- events must be uncorrelated
- automation: no manual interaction
- w/r/t coverage, a sampler can have ...
 - → no guarantee :(
 - → "weak guarantee":

full coverage for $N_{\rm train} \to \infty$

- → even if NN output function is surjective, input values or hidden-layer functions might be bounded; such bounds must be trained to be as close to the target space edge as possible, otherwise e.g. high p_T tail might be cut off
- → strong guarantee: full coverage always, even for small training sets



Flow-based deep generative model

[Weng lilianweng.github.io/lil-log] construct an estimate of a complex PDF by [Dinh et al 1605.08803] a sequence of simple invertible mappings: $f_1(\mathbf{z}_0)$ $f_{i+1}(\mathbf{z}_i)$ $f_i(\mathbf{z}_{i-1})$ $= \mathbf{x}$ \mathbf{z}_1 \mathbf{z}_i \mathbf{Z}_K \mathbf{z}_0 $\mathbf{z}_i \sim p_i(\mathbf{z}_i)$ $\mathbf{z}_K \sim p_K(\mathbf{z}_K)$ $\mathbf{z}_0 \sim p_0(\mathbf{z}_0)$ data space latent space

 \rightarrow what about cost of determinant

Coupling layers

• split input into partition $x = (x^A, x^B)$ and map

$$y^{A} = x^{A}$$
$$y^{B} = C(x^{B}; m(x^{A}))$$

where C invertible & separable

• Jacobian matrix & determinant simple

$$\frac{\partial y(x)}{\partial x^T} = \begin{pmatrix} I_n & 0\\ \frac{\partial C}{\partial (x^A)^T} & \frac{\partial C}{\partial (x^B)^T} \end{pmatrix}$$

$$\det\left(\frac{\partial y(x)}{\partial x^{T}}\right) = \prod_{i=1}^{|B|} \frac{\partial C_{i}\left(x^{B}; m\left(x^{A}\right)\right)}{\partial\left(x^{B}\right)^{T}}$$

 $\sim \mathcal{O}(d)$, *m* does not appear and can be arbitrarily complex (i.e. use a DNN)

need several partitions to transform everything at least once, stack layers:



[adapted from 1808.03856]

for C, use piecewise quadratic coupling layers as proposed in [Müller et al 1808.03856] "Neural importance sampling" fun fact: Disney research \sim connection to sampling light rays in rendering images

Training and event generation

 \rightarrow use Neural Importance Sampling as a drop-in replacement for the VEGAS optimisation within the multi-channel Monte-Carlo

- training's conventional enough
 - → minimise Pearson χ^2 divergence in mini-batches

$$D_{\chi^2} = \frac{1}{n} \sum_{i=1}^{n} \frac{(f(p_i) - g(p_i))^2}{g(p_i)}$$

→ use gradient descent to
optimise NN weights (Adam)

- generate one event
 - → draw random $x \in [0,1]^d$ uniformly
 - \rightarrow choose channel *c* randomly
 - → map $x \mapsto y \in [0,1]^d$ using the coupling layers
 - → use channel mapping $g_c(y)$ to get the set of momenta
 - → calculate weight

$$w = \left| \det \left(\frac{\partial y(x)}{\partial x^T} \right) \right| \left| \det \left(\frac{\partial p(y)}{\partial y^T} \right) \right| f(p)$$

Background

Neural Importance Sampling

Top-quark decay

• single channel with Breit-Wigner mapping

$$g(s) = \frac{1}{\left(s - M_{\rm W}^2\right)^2 + M_{\rm W}^2 \Gamma_{\rm W}^2}$$

• *d* = 2

- Monte-Carlo estimate (with MC error) E_N
- unweighting efficiency $\epsilon_{\rm uw} = \langle w \rangle / w_{\rm max}$

sample	ϵ_{uw}	E_N [GeV]
Uniform	59 %	0.1679(2)
VEGAS	50 %	0.16782(4)
NN	84 %	0.167865(5)





 \rightarrow significant improvement in all measures

Top-quark pair production

• still a single (shared) channel

• *d* = 5

 $1/\sigma\,\mathrm{d}\sigma/\mathrm{d}m_\mathrm{ee}~[1/\mathrm{GeV}]$

• $\sqrt{s} = 500 \, \mathrm{GeV}$







 \rightarrow similar to top decay, even harder for Uniform sampler

Gluon scattering gg \rightarrow 3g

- 2 independent channels, based on HAAG phase-space mappings [van Hameren, Papadopoulos hep-ph/0204055]
- d = 5, $\sqrt{s} = 1$ TeV
- $p_T > 30 \text{ GeV}$ and $m_{ij} > 30 \text{ GeV}$

sample	$\epsilon_{\rm uw}$	E_N [pb]	P _{acc}
Uniform	3 %	24806(55)	89 %
VEGAS	27.7 %	24813(23)	32 %
NN	64.3 %	24847(21)	34 %

variance now driven by zero-weight peak!



Gluon scattering gg \rightarrow 4g

changes with respect to $gg \rightarrow 3g$:

- $2 \rightarrow 3$ independent channels
- $d = 5 \rightarrow d = 8$



sample	$\epsilon_{\rm uw}$	E_N [pb]	P _{acc}
Uniform	2.7 %	9869(20)	57 %
VEGAS	31.8 %	9868(10)	17 %
NN	33.6 %	9859(10)	16 %

variance now driven by zero-weight peak!



Complementary study results for V+jets

independent implementation by FNAL group [Gao et al 2001.10028]

- implemented in SHERPA multi-channel MC as VEGAS replacement
- different paradigm: single network adapting all channels (and channel weights)

Unweighting efficiency		LO QCD			NLO QCD (RS)			
$\langle w \rangle / w_{\rm max}$		n = 0	n = 1	n = 2	<i>n</i> = 3	<i>n</i> = 4	n = 0	n = 1
$W^+ + n$ jets	SHERPA	2.8×10^{-1}	3.8×10^{-2}	7.5×10^{-3}	1.5×10^{-3}	8.3×10^{-4}	9.5×10^{-2}	4.5×10^{-3}
	NN + NF	6.1×10^{-1}	1.2×10^{-1}	1.0×10^{-2}	1.8×10^{-3}	8.9×10^{-4}	1.6×10^{-1}	4.1×10^{-3}
	Gain	2.2	3.3	1.4	1.2	1.1	1.6	0.91
$W^- + n$ jets	SHERPA	2.9×10^{-1}	4.0×10^{-2}	7.7×10^{-3}	2.0×10^{-3}	9.7×10^{-4}	1.0×10^{-1}	4.5×10^{-3}
	NN + NF	7.0×10^{-1}	1.5×10^{-1}	1.1×10^{-2}	2.2×10^{-3}	7.9×10^{-4}	1.5×10^{-1}	4.2×10^{-3}
	Gain	2.4	3.3	1.4	1.1	0.82	1.5	0.91
Z + n jets	SHERPA	3.1×10^{-1}	3.6×10^{-2}	1.5×10^{-2}	4.7×10^{-3}		1.2×10^{-1}	5.3×10^{-3}
	NN + NF	3.8×10^{-1}	1.0×10^{-1}	1.4×10^{-2}	2.4×10^{-3}		1.8×10^{-3}	5.7×10^{-3}
	Gain	1.2	2.9	0.91	0.51		1.5	1.1

TABLE II. Unweighting efficiencies at the LHC at $\sqrt{s} = 14$ TeV using the NNPDF 3.0 NNLO PDF set and a correspondingly defined strong coupling. Jets are identified using the k_T clustering algorithm with R = 0.4, $p_{T,j} > 20$ GeV and $|\eta_j| < 6$. In the case of Z/γ^* production, we also apply the invariant mass cut $66 < m_{ll} < 116$ GeV.

ightarrow as for our 4g case, scaling with multiplicity is an issue

Introduction Background Neural Importance Sampling Results Conclusions

Conclusions

- successful proof-of-principle for Neural Importance Sampling in a HEP context
- pros: strong phase-space coverage guarantee, cheap weight determination
- tested as drop-in replacement for VEGAS within a multi-channel Monte Carlo, finding ...
 - → significant improvement for simpler examples with $d \le 5$
 - → similar performance as VEGAS for 4-jet production (d = 8, and more channels)
- similar findings in independent study

- resource requirements for higher multi: could change drastically when ME generator available on accelerator such as GPU or TPU
- compare different ways of combining NN and multi-channel, study if scaling behaviour is related to these algorithmic choices
- try alternatives to piecewise quadratic coupling layers (there are plenty)
- try alternative training objectives, e.g. for reduction of zero-weight events, or for an explicit reduction of $w_{max}/\langle w \rangle$