Exhaustive Neural Importance Sampling applied to Monte Carlo event generation

NuFact 2021

Sebastian Pina-Otey*, Federico Sánchez, Thorsten Lux and Vicens Gaitan

*sebastian.pinaotey@unige.ch

September 8, 2021
Contents

1 Motivation

2 Normalizing Flows

3 Exhaustive neural importance sampling

4 Cross section example
Contents

1. Motivation
2. Normalizing Flows
3. Exhaustive neural importance sampling
4. Cross section example
Rejection sampling

- A numerical method for sampling from an analytical PDF.
- Samples generated via a similar proposal function $q(x)$, a PDF which can be both evaluated and sampled from.
- Proposal function multiplied by a constant $k \geq 1$ such that $p(x) \leq k \cdot q(x), \forall x$.
- $x \sim q(x)$ is accepted with probability $p(x)/(k \cdot q(x))$. 
Goal

Find suitable proposal $q(x)$ for rejection sampling for better efficiency.

Main issues

1. Designing a suitable proposal function can be very costly in human time.
2. Generic proposal functions, e.g. a uniform distribution, makes the algorithm usually very inefficient.
3. The inefficiency grows rapidly with the number of dimensions.

Normalizing flows proposal

1. Adapts to a given target density automatically.
   - Barely human time cost.
   - Good acceptance efficiency.
2. Grows properly with the number of dimensions.
3. Produces exact samples through rejection sampling.
Contents

1 Motivation

2 Normalizing Flows

3 Exhaustive neural importance sampling

4 Cross section example
Normalizing flows

- Define a transformation $T_\phi$ from a complex target density $q_\phi(x)$ to a simple base density $f(u)$:
  
  \[ u = T_\phi(x). \]

- $T_\phi$ is invertible and differentiable, and satisfies:

  \[ q_\phi(x) = f(T_\phi(x)) | \det J_T(x) | \]

- $f(u)$ can be evaluated and sampled from.

- $T_\phi$ allows to sample and evaluate from $q_\phi(x)$ using $f(u)$ via $T_\phi^{-1}$.

- Example of transforming from $f(u)$ Gaussian to $q_\phi(x)$ in star shape.

Source: arXiv:1912.02762
Standard problem and objective function

- **Standard problem:**
  Given data $x \sim p(x)$, find $q_\phi(x) \approx p(x)$ with only samples.

- **How?** Minimizing the Kullback-Leibler divergence:

  $$D_{\text{KL}}(p(x) \| q_\phi(x)) = \int p(x) \log \left( \frac{p(x)}{q_\phi(x)} \right) \, dx.$$ 

  $$\arg \min_\phi D_{\text{KL}}(p(x) \| q_\phi(x)) = \arg \min_\phi - \int p(x) \log q_\phi(x) \, dx$$

  $$\approx \arg \max_\phi \sum \log q_\phi(x) \text{ with } x \sim p(x).$$
Contents

1 Motivation

2 Normalizing Flows

3 Exhaustive neural importance sampling

4 Cross section example
Modifying Neural Importance Sampling

To minimize $D_{KL}(p(x) \parallel q_\phi(x))$, Müller et al. (1808.03856) propose to use the gradient

$$\frac{1}{N} \sum_{i=1}^{N} w(x_i) \nabla_\phi \log q_\phi(x_i), \quad x_i \sim q_\phi(x_i) \text{ and } w(x_i) = \frac{p(x_i)}{q_\phi(x_i)}.$$  \hspace{1cm} (1)

We propose to additionally redefine the target density with a background (e.g., uniform):

$$p_{\text{target}}(x) = (1 - \alpha) \cdot p(x) + \alpha \cdot p_{\text{bg}}(x).$$

Aim:

- **Improve initial training**, ensuring the full support of $p(x)$ (better than randomly initialized NF).
- **Ensure exhaustive coverage** of the phase space.
ENIS general scheme

Warm up phase

- Generate samples \( x_{np} - p_{bg}(x) \) with weights \( w_{q0}(x_{np}) \)
- Density \( p_{bg}(x) \)
- Generate samples \( x_{np} - p_{bg}(x) \) with weights \( w_{q0}(x_{np}) = p(x_{np})/p_{bg}(x_{np}) \)
- Get sets of \( x = (x_{np}, x_{bg}) \) and \( w = (w_{q0}(x_{np}), w_{q0}(x_{bg})) \)
- Optimize \( \phi \) with Eq. (1) \( \langle L_w(x) \rangle \nabla \log q_\phi(x)/\nabla \phi \nabla \log q_\phi(x)/\nabla \phi \)
- Finished warm up phase?
- Yes: Update \( q_\phi(x) \)
- No: Finished warm up phase?
- Yes: After finishing warm up phase

Iterative phase

- Generate samples \( x_{np} - p_{bg}(x) \) with weights \( w_{q0}(x_{np}) \)
- Get sets of \( x = (x_{np}, x_{bg}) \) and \( w = (w_{q0}(x_{np}), w_{q0}(x_{bg})) \)
- Generate samples \( x_{np} - q_\phi(x) \) with weights \( w_{q0}(x_{np}) = q(x_{np})/q_\phi(x_{np}) \)
- Optimize \( \phi \) with Eq. (1) \( \langle L_w(x) \rangle \nabla \log q_\phi(x)/\nabla \phi \nabla \log q_\phi(x)/\nabla \phi \)
- Finished iterative phase?
- Yes: Update \( q_\phi(x) \)
- No: Finished iterative phase?
- Yes: Obtain final proposal model \( q_\phi(x) \)

NF initialized randomly

Uses only \( p_{bg}(x) \) to sample

Uses both \( p_{bg}(x) \) and mainly \( q_\phi(x) \) to sample

NF is trained
Contents

1 Motivation

2 Normalizing Flows

3 Exhaustive neural importance sampling

4 Cross section example
CCQE cross section

- Charged-Current Quasi-Elastic (CCQE) interaction:
  \[ \nu_l + n \rightarrow l^- + p \]
  \[ \bar{\nu}_l + p \rightarrow l^+ + n \]

- Cross section is the probability of a specific process taking place:
  - Cross section of a CCQE interaction.

Feynman diagrams:

- $\nu_l$ CCQE scattering.
- $\bar{\nu}_l$ CCQE scattering.

Variables: $E_\nu, E_l, \theta_l, p_{\text{nucleon}}$
True vs proposal 1D

- Original target in blue.
- NF proposal in orange.
True vs proposal 2D

Marginalized $p(x)$ density

Marginalized $q_\phi(x)$ density
Weights \( w_q(x) = p(x)/q_\phi(x) \) distributions for rejection sampling
To perform rejection sampling, we need:

\[ k \cdot q(x) \geq p(x) \quad \forall x : p(x) > 0. \]

Relax \( k \) with \( Q \)-quantile of weights \( p(x)/q(x) w_Q \), denoted by \( k_Q = (Q\text{-quantile}(w))^{-1} = w_Q^{-1} \), to improve \( p_{\text{accept}} \).

Define coverage with the new \( k_Q \):

\[
\text{Coverage} = \frac{\sum_{i=1}^{N} w'(x_i)}{\sum_{i=1}^{N} w(x_i)}.
\]
Marginalized coverage

Marginalized coverage of $q_{\phi}(x)$

Marginalized coverage of $p_{\text{Unif}}(x)$

Coverage

NuFact 2021: Exhaustive Neural Importance Sampling applied to Monte Carlo event generation

Sebastian Pina-Otey
Conclusions

- Utilize normalizing flows to find suitable proposal functions to perform rejection sampling.
  - Finds automatically a good proposal function.
  - Exact sampling (corrects inefficiencies of the flow).
- Propose redefining target with background:
  - Improve initial training.
  - Ensure exhaustive coverage.
- Study the possibility of relaxing constrain on rejection sampling through the concept of coverage (see backup).
- Compare it to generic proposal, the uniform distribution, on a simple 4D cross section.
Exhaustive Neural Importance Sampling applied to Monte Carlo event generation,
S. Pina-Otey, F. Sanchez, T. Lux and V. Gaitan,
Phys. Rev. D 102, 013003 (2020).

THANK YOU!
Backup slides
Normalizing Flows: Transformation

- Transformation $T$ is partially defined through a Neural Network.
- $T$ is usually broken down into simpler transformations:

$$T = T_K \circ \cdots \circ T_1.$$ 

- Taking $z_0 = x$ and $z_K = u$:

$$z_k = T_k(z_{k-1}), \quad k = 1 : K,$$

$$\left| \det J_T(x) \right| = \left| \det \prod_{k=1}^{K} J_{T_k}(z_{k-1}) \right|.$$

- We will consider a single transformation $T(x) = u$. 

NuFact 2021: Exhaustive Neural Importance Sampling applied to Monte Carlo event generation

Sebastian Pina-Otey 22
Normalizing Flows: Autoregressive transformation

- \( | \det J_T(x) | \) has to be **easy to compute.**
  - Idea: Autoregressive transformations:

  \[
  u_i = \tau(x_i; h_i) \text{ with } h_i = c_i(x_{<i}; \phi), \quad x_{<i} = x_1:i-1.
  \]

- **Transformer** \( \tau : \mathbb{R} \to \mathbb{R} \) is bijective and differentiable, usually a **monotone** function.

- \( h_i \) are the parameters of these transformers for each component \( i \).

- \( c_i(x_{<i}; \phi) \) is the **conditioner** for the \( i \)-th component, usually a NN of parameters \( \phi \).

- All conditioners can be computed at ones efficiently using a **Masked Autoregressive Neural Network**.

- \( J_T(x) \) is now a triangular matrix, hence \( | \det J_T(x) | \) is the product of the diagonal.
Normalizing Flows: Masked Autoregressive Flow

- Simplest transformer, a linear one:

$$\tau(x_i; \alpha_i, \beta_i) = x_i \exp \alpha_i + \beta_i.$$  

- Conditioner introduces non-linearities of the density $q(x)$:

$$f_{\alpha_i}(x_{<i}; \phi_\alpha) = \alpha_i; \quad f_{\beta_i}(x_{<i}; \phi_\beta) = \beta_i.$$  

- Jacobian is trivial to compute:

$$| \det J_T(x) | = \exp \left( \sum_i \alpha_i \right).$$

G. Papamakarios et al., NeurIPS 2017
Normalizing Flows: Neural Spline Flow

- Transformers are rational quadratic monotonic splines.
  - Very flexible, infinite Taylor series.
  - Easily differentiable.
  - Analytically invertible.

- Parameters of transformer:
  - Position of knots.
  - Derivative of knots.

C. Durkan et al., NeurIPS 2019
Normalizing Flows: Neural Spline Flow

Spline:

\[ g_\theta(x) = \text{RQ Spline} \quad \text{Inverse} \quad \text{Knots} \]

Derivative:

\[ g'_\theta(x) \]

C. Durkan et al., NeurIPS 2019
Normalizing Flows: Neural Spline Flow

Example (I) of samples of $p(x)$ vs samples of $q_\phi(x)$:

Data:

NSF:

C. Durkan et al., NeurIPS 2019
Normalizing Flows: Neural Spline Flow

Example (II) of samples of $p(x)$ vs samples of $q_\phi(x)$:

Data:

NSF:

C. Durkan et al., NeurIPS 2019
ENIS algorithm

1. Warm-up phase:
   (i) Sample $x_p \sim p_{bg}(x)$ and compute their weights $w_p(x_p) = p(x_p)/p_{bg}(x_p)$.
   (ii) Sample background $x_{bg} \sim p_{bg}(x)$ with associated weights
        
        $w_{bg}(x_{bg}) = C_{w_{bg}} \cdot p_{bg}(x_{bg})$, where $C_{w_{bg}} = \frac{\alpha}{1-\alpha} \frac{\langle w_p(x_p) \rangle}{\langle p_{bg}(x_{bg}) \rangle}$.
   (iii) Optimize the parameters of $q_\phi(x)$ via Eq. (1) using $x = \{x_p, x_{bg}\}$ with weights
        
        $w(x) = \{w_p(x_p), w_{bg}(x_{bg})\}$.

2. Iterative phase:
   (i) Sample $x_q \sim q_\phi(x)$ and compute their weights $w_q(x_q) = p(x_q)/q_\phi(x_q)$.
   (ii) Sample background $x_{bg} \sim p_{bg}(x)$ with associated weights
        
        $w_{bg}(x_{bg}) = C'_{w_{bg}} p_{bg}(x_{bg})$, where $C'_{w_{bg}} = \frac{\alpha}{1-\alpha} \frac{\langle w_q(x_q) \rangle}{\langle p_{bg}(x_{bg}) \rangle}$.
   (iii) Optimize the parameters of $q_\phi(x)$ via Eq. (1) using $x = \{x_q, x_{bg}\}$ with weights
        
        $w(x) = \{w_q(x_q), w_{bg}(x_{bg})\}$.

S. Pina-Otey et al., Phys. Rev. D 102, 013003 (2020)
Visualization of modification

- Visualization of full support modification of target density.
  - No overlap in the original target $\rightarrow$ no gradient.
  - Redefined target does overlap $\rightarrow$ gradients.

Initial normalizing flow

Initial normalizing flow + original target

Initial normalizing flow + redefined target
Proposal training validation loss

- 400k training steps.
- 5 flow steps, depth 2 of transforming blocks.
- 32 hidden units per layer and 8 bins for the splines.
- 37,220 learnable parameters.
- 0.0005 learning rate and batch size of 5k.
- 200k samples for validation every 1k steps.

S. Pina-Otey et al., Phys. Rev. D 102, 013003 (2020)