

Emerging Computational Techniques for Jet Physics

Sebastian Macaluso, Kyle Cranmer, Matthew Drnevich, Johann Brehmer (*New York University*);
Duccio Pappadopulo (*N.A.*); Atılım Güneş Baydin (*Oxford*); Matthew Schwartz (*Harvard*)

1 Introduction

Jets are the most copiously produced objects at the LHC and the subject of intense experimental and theoretical study. Improvements to our understanding and treatment of jets can have significant impact on the physics program of the LHC; however, various computational bottlenecks appear in this quest. Below we will discuss a few areas where such computational bottlenecks appear and identify emerging computational techniques that may be able to address them. We hope that this may challenge some assumptions about the computational demands of simulation, reconstruction, and analysis of LHC data when jets are involved.

Probabilistic programming systems use the simulator directly during inference. They are integrated with the control flow of the event generator though they require a tractable joint likelihood. Thus, they offer both a natural way to improve the simulation efficiency, as well as the possibility of doing systematic fits to high-dimensional data by a more direct implementation of backpropagation.

1.1 Notation: a probabilistic framing of jet physics

Monte Carlo event generators (e.g. simulators like PYTHIA, Herwig, and Sherpa) encode a physics model for the fragmentation and hadronization of quarks and gluons produced at colliders. In statistical and machine learning language, they are generative models for jets. Following the notation of Ref. [1, 2], we denote the parameters of the (Monte Carlo) simulation θ , the observable output of the simulator x , and latent variables (aka Monte Carlo truth record or showering history) z . The simulators typically evolve the latent state sequentially as a Markov process and model the physics of each splitting, clustering, etc. The detector simulation fits in this framing as well, but we do not discuss it explicitly in this short letter.

We find it elucidating to reframe the following concepts in jet physics in probabilistic terms:

- Joint likelihood for latent shower and observed constituents $p(x, z|\theta)$
- Marginal likelihood for observed constituents $p(x|\theta) = \int dz p(x, z|\theta)$
- Maximum likelihood showering history $\hat{z} = \operatorname{argmax}_z p(x, z|\theta)$
- Maximum likelihood parameters for showering model $\hat{\theta} = \operatorname{argmax}_\theta p(x|\theta) = \operatorname{argmax}_\theta \int dz p(x, z|\theta)$
- Posterior distribution on showering histories $p(z|x, \theta)$

Most of these quantities are intractable to compute or inconvenient to access. For example, the joint likelihood $p(x, z|\theta)$ is exactly what is coded in PYTHIA, Herwig, and Sherpa, but often in terms of accept-reject sampling and procedural code that does not explicitly expose the probabilities themselves. In addition, the joint likelihood $p(x, z|\theta)$ is not immediately of interest to experimentalists since the (latent) showering history z is not observed. Quantities such as the marginal likelihood $p(x|\theta)$ and the maximum likelihood parameter $\hat{\theta}$ involve integration (sums) over all possible showering histories, which grows like $(2N - 3)!!$ where N is the number of jet constituents (see Fig. 1). This super-exponential growth in the number of showering histories is at the heart of many computational bottlenecks in jet physics.

2 Bottlenecks and emerging techniques

Tuning the parameters of the shower model: Ideally, if we wanted to fit (tune) the parameters θ of PYTHIA, Herwig, and Sherpa (and we had infinite computing power), then we would compute the maximum likelihood $\hat{\theta}$. Since we do not, we resort to tools like **Professor** [3] which compare projections of complicated events to individual variables (marginal distributions), which is blind to various forms of mismodelling in the high-dimensional structure of the jets. The fact that tuning the generators is itself a bottleneck suppresses the motivation to add even more flexibility and parameters to the shower models, even if they might lead to more accurate description of the jets.

Event Generation for events with large jet multiplicity Similar bottlenecks appear with the simulation of multijets events [4, 5] and shower deconstruction [6, 7, 8], where there is a need to consider

every possible clustering history at parton level. When implementing the CKKW-L matching algorithm [4, 5], parton final states need to be reweighted with the corresponding Sudakov form factors of each history, $p(x, z|\theta)$. The standard algorithm typically becomes infeasible for parton level configurations that exceed the complexity of $W/Z + 6$ jet final state [9] due to the super-exponential growth in the number of clustering histories.

Simulating jet backgrounds in signal-rich regions of phase space: Simulating sufficient numbers of multijet background events is a computational challenge due to the enormous rate of multijet events and their steeply falling spectra. The experiments have traditionally sliced the phase space into exclusive regions (eg. based on the p_T of the leading jet at parton level). This is an effective strategy for populating the tails of that distribution, but it is not effective for populating the tails of complicated phase space regions (eg. events that satisfy the cut on a boosted top-tagger). If we denote passing the cut with the indicator function $\mathbb{1}(x)$, then we are interested in efficiently sampling $z \sim p(z|\mathbb{1}(x), \theta)$.

2.1 Emerging computational techniques

Likelihood-free inference: The first emerging technique in this direction is **likelihood-free inference** or **simulation-based inference** [1, 10, 11]. Recent progress in this direction includes likelihood-free inference methods [1, 2, 10, 11]. These methods approximate the intractable $p(x|\theta)$ using machine learning and bypass an explicit marginalization over the latent state z . The techniques can exploit the joint likelihood $p(x, z|\theta)$ if it is available. An implementation of these techniques for events simulated with Pythia was introduced in [12].

Hierarchical Cluster Trellis: In [13], we introduced a novel data structure, called hierarchical cluster trellis, that can be used to efficiently represent the distribution over trees. The trellis can be used to compute the marginal likelihood $p(x|\theta)$ or the exact maximum likelihood showering history $\hat{z}$ in time and memory proportional to the significantly smaller 2^N . In particular, we showed that the trellis allows us to perform these operations for larger values of N where the naive iteration over the $(2N - 3)!!$ trees is impractical. The maximum likelihood shower history $\hat{z}$ provides a principled alternative to the generalized k_t algorithms, which are based on a greedy sequential clustering algorithm.

Thus far the implementation is based on binary trees, $1 \rightarrow 2$ splittings; however, it is possible to extend the cluster trellis to consider $2 \rightarrow 3$ splittings required in the CKKW-L algorithm.

As a result, to aid in machine learning (ML) research for jet physics, a python package for a toy generative model of a parton shower, called Ginkgo, was introduced in [14]. This model is as simple and easy to describe as possible but at the same time captures essential ingredients of parton shower generators in full physics simulations (see [15] for more details). Though this is a toy model, it ensures 4-momenta conservation as well as analytic control of the probability distribution of a clustering history.

Probabilistic Programming: The Monte Carlo simulators implicitly describe the complicated distribution $p(x, z|\theta)$ and implement sampling through random number generators. Probabilistic programming extends this functionality with the ability to condition values of some of the random variables x or z [16], and it achieves this by hijacking the random number generators. This controlling inference algorithm uses those hooks to bias the simulator towards the desired output (eg. importance sampling [17]) or through Monte Carlo sampling. In particular, it provides the ability to sample $z \sim p(z|x, \theta)$ and was applied to the **Sherpa** event generator in Refs. [18, 19]. This technique can also be used to efficiently sample the tails of backgrounds in signal-rich regions of phase space $z \sim p(z|\mathbb{1}(x), \theta)$.

3 Conclusion

These techniques have the potential to remove current bottlenecks associated to jets at the LHC, which may influence our thinking about event generation on HPCs, the need for fast simulation (eg. with deep learning), and the boundary between traditional simulators like Pythia and data-driven, neural generative models like Junipr [20]. We have been studying these approaches with a simplified shower algorithm Ginkgo that provides a tractable joint likelihood, automatic differentiation, gold mining, and probabilistic programming capabilities [14]. We intend to continue prototyping these and other computational techniques with Ginkgo and Junipr and prepare a whitepaper with our findings.

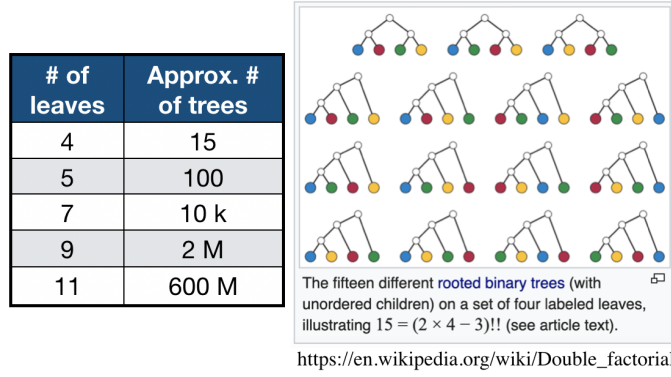


Figure 1: Left: Number of clustering histories vs number of leaves (jet constituents). Right: representation of the 15 possible clustering histories given four jet constituents.

References

- [1] J. Brehmer, K. Cranmer, G. Louppe, and J. Pavez, “A Guide to Constraining Effective Field Theories with Machine Learning,” *Phys. Rev. D* **98** no. 5, (2018) 052004, [arXiv:1805.00020 \[hep-ph\]](#).
- [2] K. Cranmer, J. Brehmer, and G. Louppe, “The frontier of simulation-based inference,” *Proceedings of the National Academy of Sciences* (2020) , <https://www.pnas.org/content/early/2020/05/28/1912789117.full.pdf>. <https://www.pnas.org/content/early/2020/05/28/1912789117>.
- [3] A. Buckley, H. Hoeth, H. Lacker, H. Schulz, and J. E. von Seggern, “Systematic event generator tuning for the LHC,” *Eur. Phys. J. C* **65** (2010) 331–357, [arXiv:0907.2973 \[hep-ph\]](#).
- [4] S. Catani, F. Krauss, R. Kuhn, and B. Webber, “QCD matrix elements + parton showers,” *JHEP* **11** (2001) 063, [arXiv:hep-ph/0109231](#).
- [5] L. Lönnblad, “Correcting the color dipole cascade model with fixed order matrix elements,” *JHEP* **05** (2002) 046, [arXiv:hep-ph/0112284](#).
- [6] D. E. Soper and M. Spannowsky, “Finding physics signals with shower deconstruction,” *Phys. Rev. D* **84** (2011) 074002, [arXiv:1102.3480 \[hep-ph\]](#).
- [7] D. E. Soper and M. Spannowsky, “Finding top quarks with shower deconstruction,” *Phys. Rev. D* **87** (2013) 054012, [arXiv:1211.3140 \[hep-ph\]](#).
- [8] D. Ferreira de Lima, P. Petrov, D. Soper, and M. Spannowsky, “Quark-Gluon tagging with Shower Deconstruction: Unearthing dark matter and Higgs couplings,” *Phys. Rev. D* **95** no. 3, (2017) 034001, [arXiv:1607.06031 \[hep-ph\]](#).
- [9] S. Höche, S. Prestel, and H. Schulz, “Simulation of Vector Boson Plus Many Jet Final States at the High Luminosity LHC,” *Phys. Rev. D* **100** no. 1, (2019) 014024, [arXiv:1905.05120 \[hep-ph\]](#).
- [10] J. Brehmer, K. Cranmer, G. Louppe, and J. Pavez, “Constraining Effective Field Theories with Machine Learning,” *Phys. Rev. Lett.* **121** no. 11, (2018) 111801, [arXiv:1805.00013 \[hep-ph\]](#).
- [11] J. Brehmer, G. Louppe, J. Pavez, and K. Cranmer, “Mining gold from implicit models to improve likelihood-free inference,” [arXiv:1805.12244 \[stat.ML\]](#).
- [12] A. Andreassen and B. Nachman, “Neural Networks for Full Phase-space Reweighting and Parameter Tuning,” *Phys. Rev. D* **101** no. 9, (2020) 091901, [arXiv:1907.08209 \[hep-ph\]](#).

- [13] C. S. Greenberg, S. Macaluso, N. Monath, J.-A. Lee, P. Flaherty, K. Cranmer, A. McGregor, and A. McCallum, “Compact Representation of Uncertainty in Hierarchical Clustering,” [arXiv:2002.11661 \[cs.DS\]](#).
- [14] K. Cranmer, S. Macaluso, D. Pappadopulo, “Toy Generative Model for Jets Package.” <https://github.com/SebastianMacaluso/ToyJetsShower>.
- [15] K. Cranmer, S. Macaluso, D. Pappadopulo, “Toy Generative Model for Jets.” https://github.com/SebastianMacaluso/Paper_Ginkgo/blob/master/main.pdf.
- [16] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani, “Probabilistic programming,” in *Proceedings of the on Future of Software Engineering*, pp. 167–181. 2014.
- [17] T. A. Le, A. G. Baydin, and F. Wood, “Inference compilation and universal probabilistic programming,” in *Artificial Intelligence and Statistics*, pp. 1338–1348. 2017.
- [18] A. G. Baydin, L. Shao, W. Bhimji, L. Heinrich, S. Naderiparizi, A. Munk, J. Liu, B. Gram-Hansen, G. Louppe, L. Meadows, *et al.*, “Efficient probabilistic inference in the quest for physics beyond the standard model,” in *Advances in neural information processing systems*, pp. 5459–5472. 2019.
- [19] A. G. Baydin, L. Shao, W. Bhimji, L. Heinrich, L. Meadows, J. Liu, A. Munk, S. Naderiparizi, B. Gram-Hansen, G. Louppe, *et al.*, “Etalumis: Bringing probabilistic programming to scientific simulators at scale,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–24. 2019.
- [20] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, “JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics,” *Eur. Phys. J. C* **79** no. 2, (2019) 102, [arXiv:1804.09720 \[hep-ph\]](#).