Hierarchical clustering in particle physics through reinforcement learning

Johann Brehmer
New York University
johann.brehmer@nyu.edu

Sebastian Macaluso
New York University
sm4511@nyu.edu

Duccio Pappadopulo
Bloomberg L. P.
dpappadopulo@bloomberg.net

Kyle Cranmer
New York University
kyle.cranmer@nyu.edu

Abstract

Particle physics experiments often require the reconstruction of decay patterns through a hierarchical clustering of the observed final-state particles. We show that this task can be phrased as a Markov Decision Process and adapt reinforcement learning algorithms to solve it. In particular, we show that Monte-Carlo Tree Search guided by a neural policy can construct high-quality hierarchical clusterings and outperform established greedy and beam search baselines.

1 Introduction

Particle interactions in collider experiments often produce highly energetic quarks and gluons (or *partons*). These elementary particles then radiate more and more quarks and gluons in a series of successive binary splittings, ultimately leading to a *jet*: a spray of stable particles that can be measured in a detector. This *parton shower* process follows the laws of quantum chromodynamics, our understanding of it is encoded in sophisticated simulators.

Reconstructing the properties of the original elementary particles from the observed final-state particles is an important step in the data analysis in particle physics experiments, including those at the Large Hadron Collider at CERN. It is often crucial in the search for new particles or the measurements of the properties of the Higgs boson. This *jet clustering* problem can be phrased as an inference task that aims to invert the generative radiation process. Given a set of observed final-state particles or leaf nodes, the goal is to construct the most plausible binary tree of particle splittings.

Unfortunately, for more than a few final-state particles, the search space of this combinatorial optimization problem is too large to find the true maximum-likelihood tree. The industry standard is to solve this problem with a greedy algorithm based on one of several heuristics (for instance the popular k_T, Cambridge-Aachen, and anti-k_T algorithms [1–4]). Improvements may come from questioning two aspects of this industry standard. First, we may be able to improve clustering by switching from optimizing heuristics to maximizing the likelihood [5]. Second, there are more powerful algorithms than the greedy optimization, for instance beam search, probabilistic methods [6], and simulation-based inference methods [7].

In this paper we propose to phrase this combinatorial optimization problem as a Markov Decision Process (MDP), which allows us to use reinforcement learning (RL) methods to solve it. In particular, we adapt Monte-Carlo Tree Search (MCTS) guided by a neural network policy to the problem of jet clustering. This approach closely follows the AlphaZero algorithm [8–10], which achieved superhuman performance in a range of board games, demonstrating its ability to efficiently search large combinatorial spaces. We also test imitation learning, specifically Behavioral Cloning, and

Third Workshop on Machine Learning and the Physical Sciences (NeurIPS 2020), Vancouver, Canada.

train a policy to imitate the actions that reproduce the true tree from the generative model. While (model-free) RL methods have been used in the context of jet grooming, i. e. pruning an existing tree to remove certain backgrounds [11], they have not yet been used for clustering, that is, the construction of the binary tree itself. In first experiments we demonstrate that the MCTS clustering agent outperforms not only the greedy industry standard, but also beam search. Finally, we discuss which steps need to be taken to scale this approach to real-life applications. The code used for this study is available at `https://github.com/johannbrehmer/ginkgo-rl`.

2 Jet clustering as a Markov Decision Process

Problem statement. We describe the parton shower as a stochastic generative process in which a set of particles described by their four-momenta at time step t, $z_t = \{p_{t,1}, p_{t,1}, \dots, p_{t,n_t}\}$, undergoes successive binary splittings $z_t \to z_{t+1} \sim p_s(z_{t+1}|z_t)$. The successor state z_{t+1} contains $n_t - 1$ unchanged particles from z_t as well as the children of one radiating particle. The whole decay process thus forms a binary tree. The individual splittings are Markov, and their probability densities $p_s(z_{t+1}|z_t)$ are known and tractable. In particular, each splitting has to satisfy the energy-momentum conservation law: a splitting $p_{t,i} \to p_{t+1,i}, p_{t+1,j}$ satisfies $p_{t,i} = p_{t+1,i} + p_{t+1,j}$. The initial state is a single elementary particle, $z_1 = \{p_{1,1}\}$. The process terminates after N splittings, after which the final-state momenta $x = z_N$ are observed. For a detailed description of the parton shower, see e. g. Refs. [12, 13].

Given an observed final state $x = z_N = \{p_{N,1}, p_{N,2}, \dots, p_{N,n_N}\}$, we study the problem of inferring the maximum-likelihood latent binary tree $z^* = \arg\max_{\{z_1,\dots,z_N\}} p(x|\{z_1, \dots, z_N\})$ with

$$p(x|\{z_1, \dots, z_N\}) = p_s(x|z_{N-1}) \prod_{t=1}^{N-2} p_s(z_{t+1}|z_t)\,. \tag{1}$$

Markov Decision Process (MDP). We treat the problem of clustering as an MDP $(\mathcal{S}, \mathcal{A}, P, R)$:

- The state space $\mathcal{S}$ is given by all possible particle sets at any given point during the clustering process, $s = z_t$.
- The actions $\mathcal{A}$ are the choice of two particles $a = (i, j)$ with $1 \le i < j \le n_t$ to be merged.
- The state transitions P are deterministic and update z_t to z_{t-1} by replacing the particles $p_{t,i}$ and $p_{t,j}$ with a parent $p_{t-1,i} = p_{t,i} + p_{t,j}$. All other particles are left unchanged, each state transition thus reduces the number of particles by one.
- The rewards R are the splitting probabilities, $R(s = z_t, a = (i, j)) = \log p_s(z_t|z_{t-1}(i, j))$.
- The MDP is episodic and terminates when only a single particle is left.

An agent solves the jet clustering problem by first considering the state of all observed, final-state particles and choosing which two to merge into a parent. It receives the log likelihood of this splitting as reward. Next, it considers the reduced set of particles where the two chosen particles have been replaced by their proposed parent, chooses the next pair of particles to merge, and so on. Rolling out an episode leads to a proposed clustering tree $z = \{z_1, \dots, z_N\}$, with the total received reward being equal to the log likelihood of this tree following Eq. (1). We illustrate this setup in Fig. 1.

Limitations. Our description of the parton shower as a binary process and in particular the assumption that the final state is perfectly observable are approximations to the true physical process. In reality, the final states of the shower first have to form stable particles (hadrons) and then interact with a complicated detector to be measured. The clustering problem is thus actually a partially observable MDP. These effects, which are also not modeled in state-of-the-art clustering algorithms, go beyond the scope of this proof-of-concept paper.

3 Algorithms

Monte-Carlo Tree Search (MCTS). The formulation of jet clustering as an MDP allows us to use any (model-free) reinforcement learning (RL) algorithm to tackle it. Since the state transition model is known (and deterministic), we instead use a model-based planning approach to leverage this knowledge. We choose MCTS [8], which builds a search tree over possible clusterings z by rolling out a number of clusterings. During these roll-outs, at each state s we choose the action a that maximizes the upper confidence bound on the action values (PUCT)

$$U_{s,a} = Q_{s,a} + c\,\pi(s,a)\,\frac{\sqrt{N_s}}{1+N_{s,a}}\,. \qquad (2)$$

Here $\pi(s,a)$ is a learnably policy implemented as a neural network, $Q_{s,a}$ is the mean normalized reward received after chosing action a in state s, N_s ($N_{s,a}$) is the number of times a state s has been visited (and action a has been chosen), and c is a hyperparameter that balances exploration and exploitation. Since the episode length is given by the number of leaf particles, we roll out all MCTS trajectories until termination.

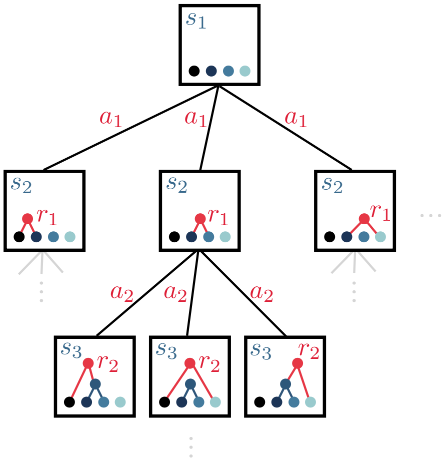

Figure 1: Jet clustering as a Markov Decision Process. States s (squares) represent (partial) clusterings of the original particles (small circles), the agent begins in the unclustered state (top square). Each action a chooses a pair of particles in the current state (which may be either part of the original particle set or the result of a partial clustering) to be merged next. The reward r is the log likelihood of the corresponding $1 \to 2$ splitting.

After a fixed number of MCTS roll-outs, the agent ultimately picks the action a^* that lead to the largest individual roll-out reward. The policy $\pi(s,a)$ is trained to agree with the MCTS decisions by maximizing the log likelihood $\log \pi(s,a^*)$.

While the MCTS algorithm should be able to learn good policies from raw data, we find that with limited training time it benefits from feature engineering and a suitable initialization. As inputs into the neural network $\pi(s,a)$ we use not only the four-momenta of all current particles, but also the splitting probabilities $p_s(z_{t+1}|z_t(i,j))$ for all possible actions $a=(i,j)$. In addition, we initialize the MCTS search tree at each step by running beam search with a small beam size b.

Behavioral Cloning (BC). Next, we consider a clustering algorithm based on imitation learning, specifically Behavioral Cloning: a policy π is trained to imitate the actions that reconstruct the true trees, which we can extract from the generative model, by maximizing $\log \pi(s, a_{\mathrm{truth}})$.

MLE-BC. For samples with a small number of final-state particles n_N we can also construct the exact maximum-likelihood tree using the algorithm from Ref. [14], and train a policy to imitate it. The MLE tree becomes impractically slow for $n_N > 10$; for these samples we continue to use the simulator truth trees as demonstrator actions.

BC-MCTS. Finally, we consider an MCTS planner where the policy π is pretrained by BC, again using the true decay trees as demonstrator actions.

4 Experiments

The GINKGO simulator. We demonstrate the MCTS clustering algorithm in a simplified setup where the likelihood of each individual tree is tractable. To this end we use GINKGO [15], a toy generative model for jet physics that captures essential aspects of realistic simulators of the parton shower. We implement the MDP described above based on this simulator and provide a standard OPENAI GYM [16] interface.

Setup. We repeatedly simulate the parton shower for a single initial elementary particle. The final states of these decays are then clustered by our algorithms. Neural policies are trained for 60 000 steps (MCTS) or 10^6 steps (BC). The clustered trees are evaluated on 500 samples, using the mean log likelihood as given in Eq. (1) as metric.

Baselines. We compare MCTS (with $c = 1$) and BC agents to a greedy algorithm that at each state picks the action with the maximum splitting likelihood p_s, a beam search algorithm that maintains the b most likely clusterings while descending down the search tree, and a random policy. For jets with a small number of final-state particles n_N we also compute the exact MLE tree following Ref. [14], though the $O(3^{n_N})$ complexity of that algorithm makes this intractable for large n_N.

Results. We show the results in Tbl. 1 and in Fig. 2, where we plot the clustering log likelihood against the computational cost of the clustering algorithms (left panel) and against the number of final-state particles (middle and right panel). While the greedy and beam search baselines lead to a robust performance at low computational cost, MCTS planning can generate hierarchical clusterings of a markedly higher likelihood. This advantage is more pronounced at larger number of final-state particles, showing that MCTS can explore large combinatorial spaces better than the baselines. We also observe that for small-to-medium trees (where maximum likelihood trees are tractable) the MCTS clusterings are close to optimal.

Algorithm	Log likelihood
Random	-198.3 ± 23.8
Greedy	-96.0 ± 0.0
Beam search ($b = 5$)	-94.1 ± 0.0
($b = 20$)	-93.5 ± 0.0
($b = 100$)	-93.3 ± 0.0
($b = 1000$)	-93.3 ± 0.0
MCTS ($b = 3$, $n_{\mathrm{MCTS}} = 10$)	-93.4 ± 0.1
($b = 5$, $n_{\mathrm{MCTS}} = 20$)	-93.3 ± 0.1
($b = 20$, $n_{\mathrm{MCTS}} = 50$)	-93.0 ± 0.1
($b = 100$, $n_{\mathrm{MCTS}} = 200$)	$\mathbf{-92.8} \pm 0.1$
BC	-108.5 ± 1.6
MLE-BC	-108.3 ± 1.7
BC-MCTS ($b = 3$, $n_{\mathrm{MCTS}} = 10$)	-93.5 ± 0.1
($b = 5$, $n_{\mathrm{MCTS}} = 20$)	-93.3 ± 0.1
($b = 20$, $n_{\mathrm{MCTS}} = 50$)	-93.0 ± 0.1
($b = 100$, $n_{\mathrm{MCTS}} = 200$)	$\mathbf{-92.8} \pm 0.1$
MCTS ($b = 5$, $n_{\mathrm{MCTS}} = 20$) variations:	
exploit more ($c = 0.1$)	-93.4 ± 0.1
explore more ($c = 10$)	-93.3 ± 0.1
no p_s as input to policy	-93.8 ± 0.0
final decision based on PUCT	-94.6 ± 0.3
no beam search roll-outs	-97.3 ± 1.3
no PUCT roll-outs	-93.9 ± 0.0
instead of NN, use policy $\propto p_s$	-93.5 ± 0.0
instead of NN, use random policy	-93.9 ± 0.0

Table 1: Mean log likelihood of clustered trees (larger is better, best results are bold). We show the mean and its standard error between five models trained with different random seeds. MCTS (middle) yields higher-quality hierarchical clusterings than the baselines (top).

Imitation learning does not perform very well. While the BC and MLE-BC policies generate clusterings that are substantially better than random, they are still worse than the greedy baseline. Pretraining the policy that guides the MCTS planner on BC does not make a significant difference compared to the simple MCTS setup. Further ablation studies of our MCTS setup show that the neural network policy performs better than simply using a random policy or a policy that is proportional to the splitting likelihood p_s to explore the search tree. Feeding the splitting likelihoods p_s as additional features into the policy network also helps the models, and the initialization of the search tree by running beam search is particularly important for a good performance.

5 Discussion

We have presented new algorithms for a hierarchical clustering problem in particle physics based on Monte-Carlo Tree Search guided by a neural policy as well as on Behavioral Cloning. In a simplified scenario using the GINKGO simulator, the MCTS algorithm generated higher-quality hierarchical clusterings than greedy and beam search baselines.

Scaling this approach to real-life particle analyses requires a few more steps. First, we need to switch to a realistic model of the parton shower. This is straightforward for the RL algorithm itself. To compute the reward function, one could add the capability to evaluate the splitting likelihoods to existing simulators or switch to heuristics. Unlike greedy and beam search algorithms, the MCTS clustering algorithm can optimize hierarchical clusterings based on *any* reward function, which may be delayed up to the termination of the episode. This allows us to optimize it directly based on some downstream task, letting us tailor clustering algorithms to a given problem.

A second necessary step is the careful analysis of infrared and collinear safety, invariance properties of the hierarchical clustering under certain additional splittings in the generative process. In addition, we will need to make sure that the algorithm performs reliably when presented with high-energy events, which are scarce in the training data, but of particular interest in many physics analyses. More

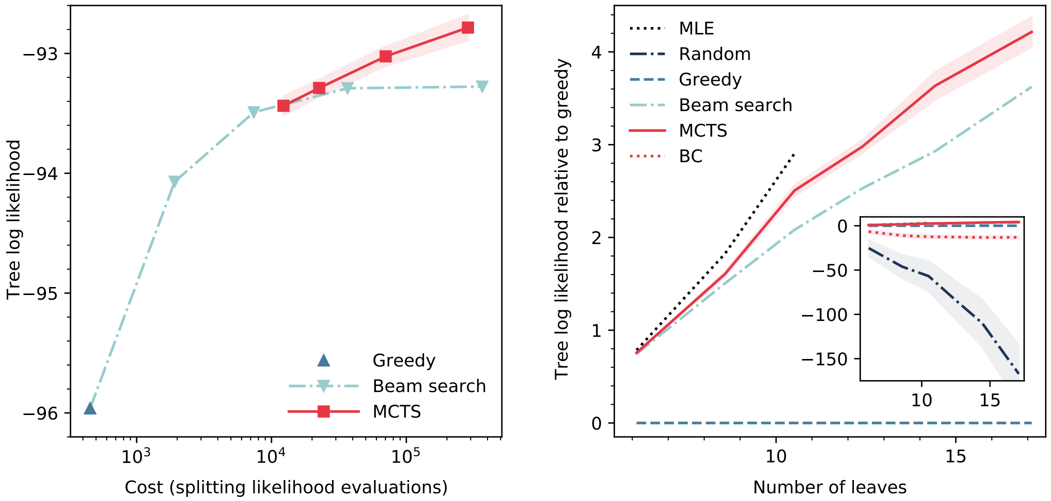

Figure 2: Mean log likelihood of clustered trees (larger is better). We show the mean and its standard error between five models trained with different random seeds. **Left**: against the computational cost, measured as the number of evaluations of the splitting likelihood p_s required by the different algorithms. For beam search and MCTS we show four different hyperparameter settings. **Right**: as a function of the number of final-state particles (leaves of the tree), using the best-performing (and most computationally expensive) hyperparameter setup for each algorithm. MCTS (solid, red) gives the highest-quality tree clusterings.

broadly, we need to analyze how much higher-likelihood reconstructed trees can improve the results of downstream physics analyses.

Finally, the RL algorithms need to become faster to be useful under real-life conditions. Adding a value network instead of rolling out all trajectories until termination as well as using neural architectures and search trees that directly reflect the hierarchical clustering structure of the problem may help improve the performance.

Broader impact

The application of reinforcement learning methods to hierarchical clusterings in particle physics has the potential to improve data analysis methods in collider experiments. In a simplified setup we demonstrated that it can outperform a greedy algorithm, which is the current industry standard in particle physics analyses. If this encouraging result persists under more realistic conditions and the algorithm can be made faster, our approach may make precision measurements of the Standard Model of Particle Physics, studies of properties of the Higgs boson, and searches for new particles more efficient, and ultimately help us understand the fundamental properties of the universe a little better. The RL approach may also improve the quality of hierarchical clusterings in other contexts, for instance in genomics [14].

Acknowledgements

We would like to thank Craig S. Greenberg, Nicholas Monath, Ji-Ah Lee, Patrick Flaherty, Andrew McGregor, and Andrew McCallum for their collaboration on efficient maximum likelihood estimation on Ginkgo with a trellis. We are grateful to Matt Drnevic for his work on Ginkgo and want to thank Marcel Gutsche for great discussions. Our thanks also goes to the authors and maintainers of gym [16], Jupyter [17], Matplotlib [18], NumPy [19], PyTorch [20], sacred [21], and stable-baselines [22]. We are grateful for the support of the National Science Foundation under the awards ACI-1450310, OAC-1836650, and OAC-1841471, the Moore-Sloan data science environment at NYU, as well as the NYU IT High Performance Computing resources, services, and staff expertise.

References

[1] S. Catani, Y. L. Dokshitzer, M. H. Seymour, and B. R. Webber, *Longitudinally invariant k_t clustering algorithms for hadron hadron collisions*, Nucl. Phys. B **406** (1993) 187.

[2] S. D. Ellis and D. E. Soper, *Successive combination jet algorithm for hadron collisions*, Phys. Rev. D **48** (1993) 3160, arXiv:9305266 [hep-ph].

[3] Y. L. Dokshitzer, G. D. Leder, S. Moretti, and B. R. Webber, *Better jet clustering algorithms*, JHEP **08** (1997) 1, arXiv:9707323 [hep-ph].

[4] M. Cacciari, G. P. Salam, and G. Soyez, *The anti-k_t jet clustering algorithm*, JHEP **04** (2008) 63, arXiv:0802.1189 [hep-ph].

[5] Cranmer, Kyle and Macaluso, Sebastian and Pappadopulo, Duccio, "Clustering algorithms for jet physics." `https://github.com/SebastianMacaluso/ReclusterTreeAlgorithms`, 2019.

[6] S. D. Ellis, A. Hornig, T. S. Roy, D. Krohn, and M. D. Schwartz, *Qjets: A Non-Deterministic Approach to Tree-Based Jet Substructure*, Phys. Rev. Lett. **108** (2012) 182003, arXiv:1201.1914 [hep-ph].

[7] K. Cranmer, J. Brehmer, and G. Louppe, *The frontier of simulation-based inference*, Proceedings of the National Academy of Sciences (2020) , arXiv:1911.01429 [stat.ML].

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and Others, *Mastering the game of go with deep neural networks and tree search*, nature **529** (2016) 7587, 484.

[9] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, and Others, *Mastering the game of go without human knowledge*, nature **550** (2017) 7676, 354.

[10] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, and Others, *Mastering chess and shogi by self-play with a general reinforcement learning algorithm*, arXiv:1712.01815 (2017) .

[11] S. Carrazza and F. A. Dreyer, *Jet grooming through reinforcement learning*, Phys. Rev. **D100** (2019) 1, 14014, arXiv:1903.09644 [hep-ph].

[12] R. Ellis, W. Stirling, and B. R. Webber, *QCD and collider physics*, vol. 8. Cambridge University Press, 2011.

[13] A. Buckley and Others, *General-purpose event generators for LHC physics*, Phys. Rept. **504** (2011) 145, arXiv:1101.2599 [hep-ph].

[14] 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 (2020) .

[15] K. Cranmer, S. Macaluso, and D. Pappadopulo, "Toy Generative Model for Jets Package." `https://github.com/SebastianMacaluso/ToyJetsShower`, 2019.

[16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *OpenAI Gym*, arXiv:1606.01540 [cs.LG].

[17] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and E. al., *Jupyter Notebooks - a publishing format for reproducible computational workflows*, in *ELPUB*. 2016.

[18] J. D. Hunter, *Matplotlib: A 2D graphics environment*, Computing In Science & Engineering **9** (2007) 3, 90.

[19] S. van der Walt, S. C. Colbert, and G. Varoquaux, *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science and Engineering **13** (2011) 2, 22, arXiv:1102.1523 [cs.MS].

[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in PyTorch*, in *Advances in Neural Information Processing Systems*. 2017.

[21] K. Greff, A. Klein, M. Chovanec, F. Hutter, and J. Schmidhuber, *The Sacred Infrastructure for Computational Research*, in *Proceedings of the 16th Python in Science Conference*, K. Huff, D. Lippa, D. Niederhut, and M. Pacer, eds. 2017.

[22] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, “Stable Baselines.” `https://github.com/hill-a/stable-baselines`, 2018.