Tree Clustering Algorithms for Jets Physics

Kyle Cranmer[1], Sebastian Macaluso[1] and Duccio Pappadopulo[2]

1 Center for Cosmology and Particle Physics & Center for Data Science, New York University, USA

2 Bloomberg LP, New York, NY 10022, USA.

1 Motivation and problem formulation

At physics colliders, e.g. the Large Hadron Collider at CERN, two beams of particles (accelerated to very high energies) directed against each other are collided head-on. As a result, new unstable particles get created and a *showering process* happens, where successive binary splittings of these initial unstable particles are produced until all the particles are stable. This process gives rise to jets, which are a collimated spray of energetic charged and neutral particles. We refer to these final particles as the jet constituents.

As a result of this showering process, there could be many latent paths that may lead to a specific jet (i.e. the set of constituents). Thus, it is natural to represent a jet and the particular showering path that gave rise to it as a binary tree, where the inner nodes represent each of the unstable particles and the leaves represent the jet constituents.

In this context, it becomes relevant and interesting to study algorithms to reconstruct the jet constituents (leaves) into a binary tree (clustering algorithms) and how close these algorithms can reconstruct the truth latent path. Being able to perform a precise reconstruction of the truth tree would assist in physics searches at the Large Hadron Collider. In particular, determining the nature (type) of the initial unstable particle (and its children and grandchildren) that gave rise to a specific jet is essential in searches of new physics as well as precision measurements of the current model, i.e. the Standard Model of Particle Physics.

There are software tools called **parton showers**, e.g. PYTHIA, Herwig, Sherpa, that encode a physics model for the simulation of jets that are produced at colliders. Currently, parton shower generators in full physics simulations are implicit models, i.e.

they are a probabilistic model defined only in terms of the samples they generate. Thus, the likelihood of a particular showering process is intractable. 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 was provided in [1]. This model has a tractable likelihood, and is as simple and easy to describe as possible but at the same time captures the essential ingredients of parton shower generators in full physics simulations (see [2] for more details).

In particle physics collisions, the likelihood $p(x|\theta)$ for parameters θ defined in the physics model given data x, can be factorized into a parton level-process, a parton shower and detector interactions, as introduced in [3].

$$p(x|\theta) = \int dz_{\text{detector}} \int dz_{\text{shower}} \int dz\, p(x|z_{\text{detector}}) p(z_{\text{detector}}|z_{\text{shower}}) p(z_{\text{shower}}|z, \theta_s) p(z|\theta_t) \tag{1}$$

where each term in the integral describes the likelihood of a latent path z given the model parameters $\{\theta_t, \theta_s\}$. In particular, $p(z|\theta_p)$ is the likelihood of getting matrix element level momenta z conditional on the theory parameters θ_t, and $p(z_{\text{shower}}|z, \theta_s)$ the likelihood of a particular showering history conditional on z and shower model parameters θ_s. Also, $p(z_{\text{detector}}|z_{\text{shower}})$ and $p(x|z_{\text{detector}})$ describe how the process evolves to reconstructed observables x through detector effects and the reconstruction method, respectively.

In this work we are interested in studying how well we can reconstruct the joint likelihood $p(z_{\text{shower}}|z, \theta_s)$ for a given shower history z_{shower} (tree latent path) of a jet, as we show schematically in fig. 1. Thus, we will focus on the showering likelihood and for our purposes we will consider the final states of the shower as the observables x.[1]

Traditionally, jets are generated with a parton shower and then one of the generalized k_t clustering algorithms is used to reconstruct them. These algorithms sequentially cluster jet constituents by merging the closest pair based on a distance measure:

$$d_{ij} = \min(p_{ti}^{2\alpha}, p_{tj}^{2\alpha}) \frac{\Delta R_{ij}^2}{R^2} \tag{2}$$

where ΔR_{ij} is the angular distance for the pair $\{i, j\}$, R is a fixed value for the jet radius and $\alpha = \{-1, 0, 1\}$ specifies the anti-k_t, Cambridge/Aachen and k_t algorithms respectively.

[1] We leave the study of detector and reconstruction effects for future work. Also, for simplicity we do not include hadronization effects.

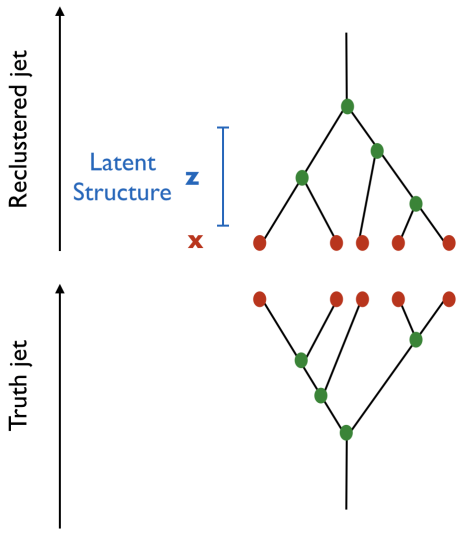

Figure 1: Schematic representation of the tree structure of a sample jet generated with our Toy Model for Jets and the reclustered tree for some clustering algorithm. For a given algorithm, z labels the different variables that determine the latent structure of the tree. The tree leaves x are labeled in red and the inner nodes in green.

The Toy Generative Model for Jets enables us to explore and compare different jet clustering algorithms based on the jet likelihood, given that we can reconstruct $p(z_{\text{shower}}|z, \theta_s)$. As a result, this model allows to have a unified picture for generation and inference, and could lead to a systematic fit of shower parameters θ_s to data, following the technique introduced in [3].

2 Likelihood-based clustering algorithms

There can be many techniques to cluster the set of jet constituents (leaves) into a binary tree, from machine learning based ones to more traditional algorithms.

2.1 Maximum likelihood estimate

We start by studying algorithms that aim to maximize the joint likelihood of a jet. Thus, we want to find the maximum likelihood estimate (MLE) for the latent structure of a jet, given a set of leaves. In this approach, the tree latent structure z_{shower} is fixed by the algorithm we choose. Also, as already mentioned, for our purposes we will consider the final states of the shower as the observables x. A complete solution

can be found using the Viterbi algorithm, obtaining $\hat{z}_{\text{Viterbi}} = \text{argmax}_{z_{\text{shower}}} p(x|z_{\text{shower}})$. However, this approach becomes computationally infeasible for a large number of leaves, e.g. we typically consider jets with 50 to 100 constituents. There are other algorithms that provide an approximate solution, i.e. greedy and beam search algorithms.

- Greedy Likelihood: tree joint likelihood $\hat{z}_{\text{Greedy}}$ from locally maximizing the likelihood of a node pairing at each step.
- Beam Search Likelihood: tree joint likelihood $\hat{z}_{\text{BS}}$ from maximizing the likelihood of multiple steps before choosing the latent path.

Next, in [4] we introduce implementations of the greedy and beam search algorithms to jets physics and study how well they reconstruct the truth joint likelihood and latent structure of jets generated with the Toy Model. Visualizations to compare the latent structure are obtained using the python package in [5].

Below we compare visualizations of the same sample jet (with 50 constituents) for the truth-level, greedy and beam search trees. First, in fig. 2 we show three 1D tree-only visualizations where each node is connected to its parent. In each of the three cases, we compare the truth-level tree (bottom) versus the one we get with a clustering algorithm. Typically we expect that the fewer the lines crossings, the closer the latent structure to the truth tree. From this point of view, we could order the algorithms in increasing performance as k_t, greedy and beam search. Next, in fig. 3 we show 2D heat clustermaps where the color scale specifies the total number of steps needed to connect two leaves through their closest common ancestor using the truth-level jet tree. We show the leaves (and tree latent structure) ordered following the truth-level tree for columns and a clustering algorithm for rows (except for the bottom right heat map, where both columns and row represent the truth-level tree). The better the truth tree latent structure is reconstructed, the more the heat map structure looks block diagonal, as in the bottom right subfigure.

2.2 Machine learning based algorithms

- Reinforcement Learning
- Monte Carlo tree search (MCTS)
- Unsupervised Latent Tree Induction with Deep Inside-Outside Recursive Autoencoders (DIORA) ?

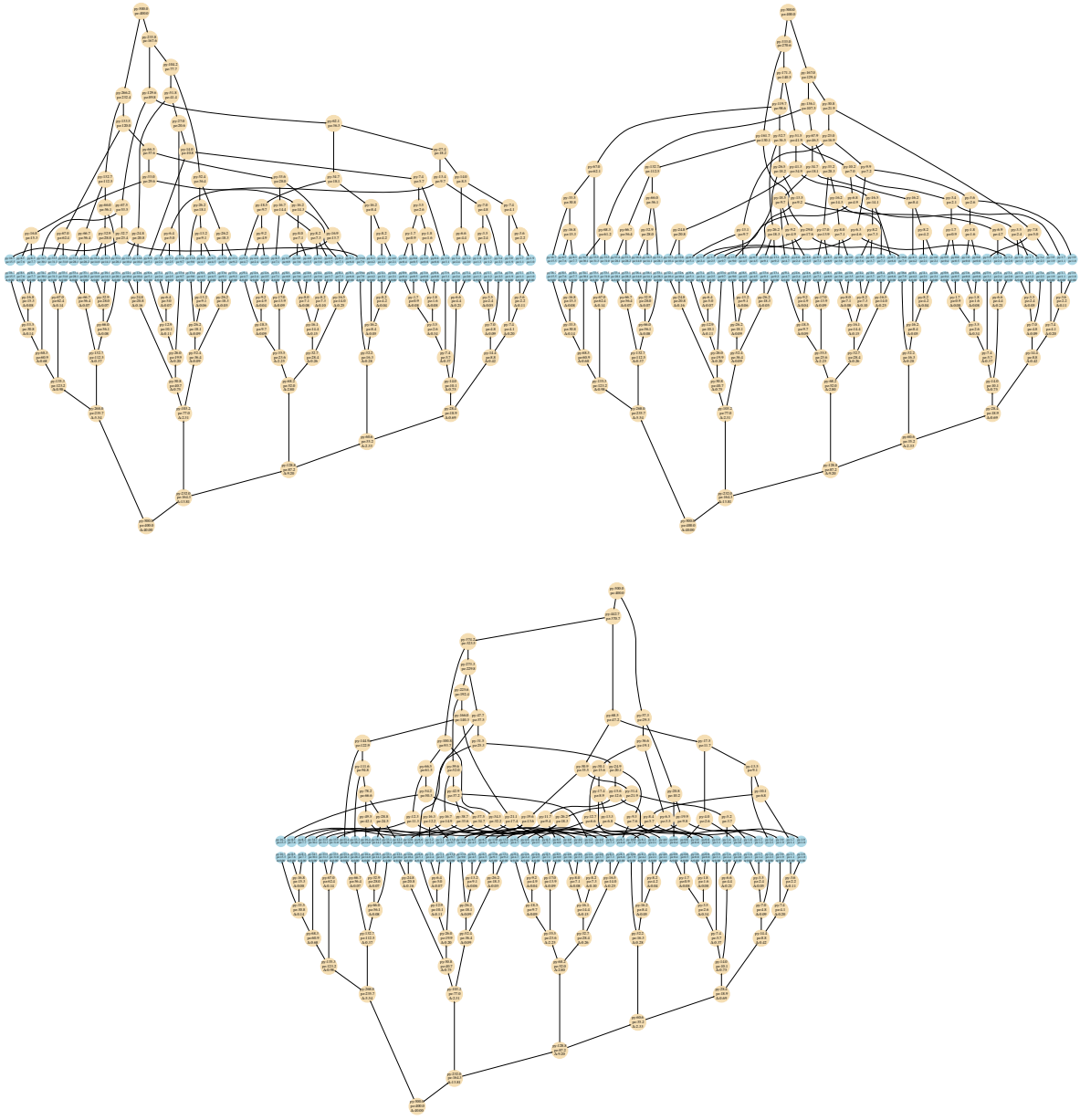

Figure 2: 1D Tree-only visualization of the same jet constituents reconstructed with three different algorithms, where we show the reclustered jet on top and the truth one at the bottom. Figures correspond to beam search (top left), greedy (top right) and k_t (bottom) algorithms. The fewer the line crossings, the closer the latent structure is to the truth tree. The horizontal ordering of the leaves corresponds to the order in which the leaves are accessed when traversing the truth tree.

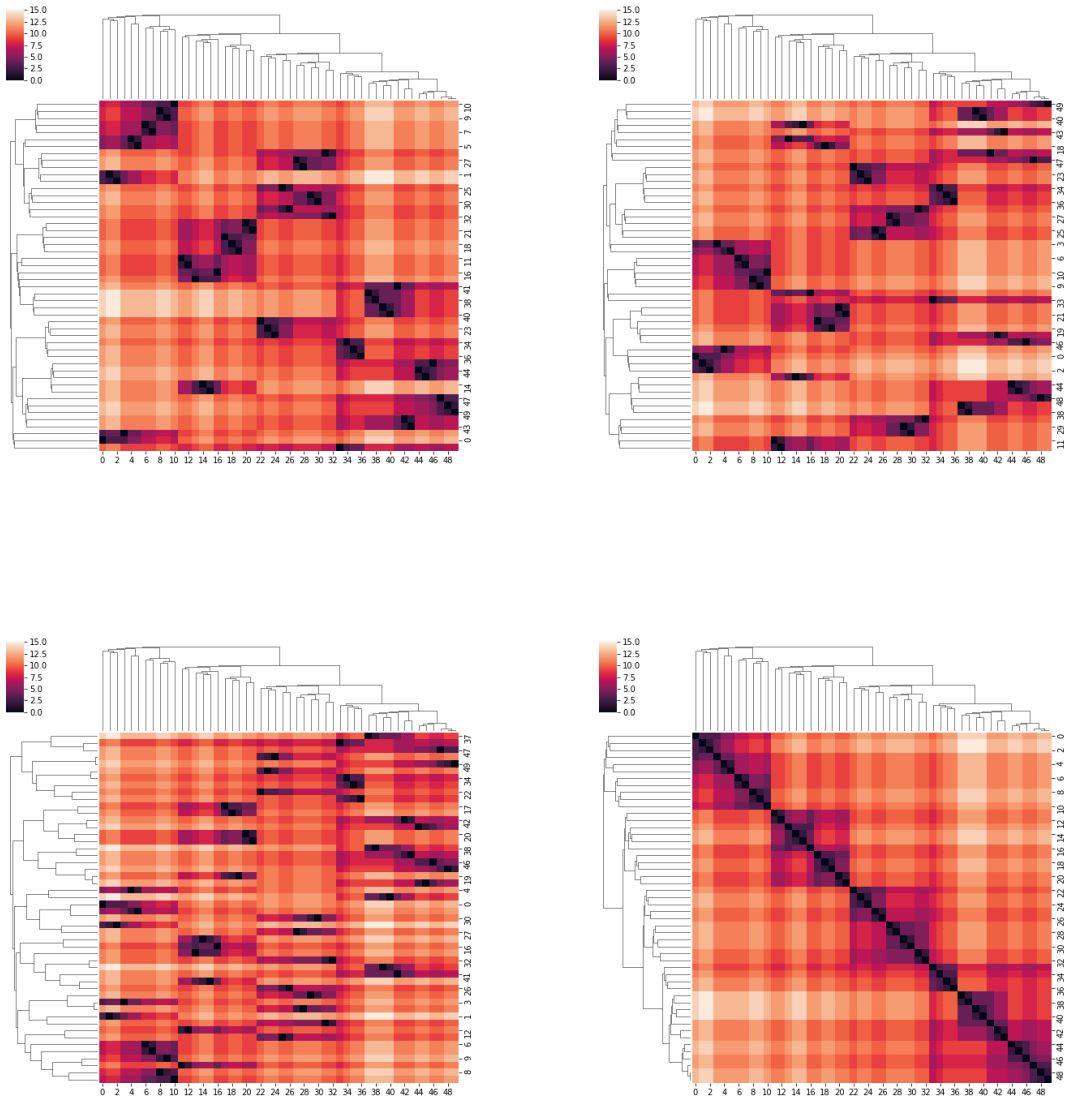

Figure 3: 2D heat clustermap visualizations of the same truth jet. The leaves ordering corresponds to the order that they are accessed when traversing the truth tree (columns) and the clustering algorithm (rows), except for the bottom right figure that shows the truth tree both for rows and columns. We show the cases of beam search (top left), greedy (top right) and k_t (bottom left).

arXiv:1904.02142

Code available here: github.com/iesl/diora

- Compound Probabilistic Context-Free Grammars for Grammar Induction

  arXiv:1906.10225

3 Latent path likelihood

We could invert the problem and study the likelihood $p(z_{\text{shower}}|x)$ of each possible latent path conditioned on a set of constituents. Related work in this direction includes [6], where different trees are weighted by a metric that depends on the distance d_{ij} in (2).

References

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

[2] K. Cranmer, S. Macaluso, D. Pappadopulo, "Toy Generative Model for Jets." https://github.com/SebastianMacaluso/ToyJetsShower/blob/master/notes/toyshower_v4.pdf.

[3] 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].

[4] K. Cranmer, S. Macaluso, D. Pappadopulo, "Clustering algorithms for jet physics." https://github.com/SebastianMacaluso/ReclusterTreeAlgorithms.

[5] K. Cranmer, S. Macaluso, D. Pappadopulo, "Visualize Binary Trees Package." https://github.com/SebastianMacaluso/VisualizeBinaryTrees.

[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].