Looking into Jets with Machine Learning

Sebastian Macaluso
New York University

Particle Theory Seminar
Harvard University
November 19, 2019

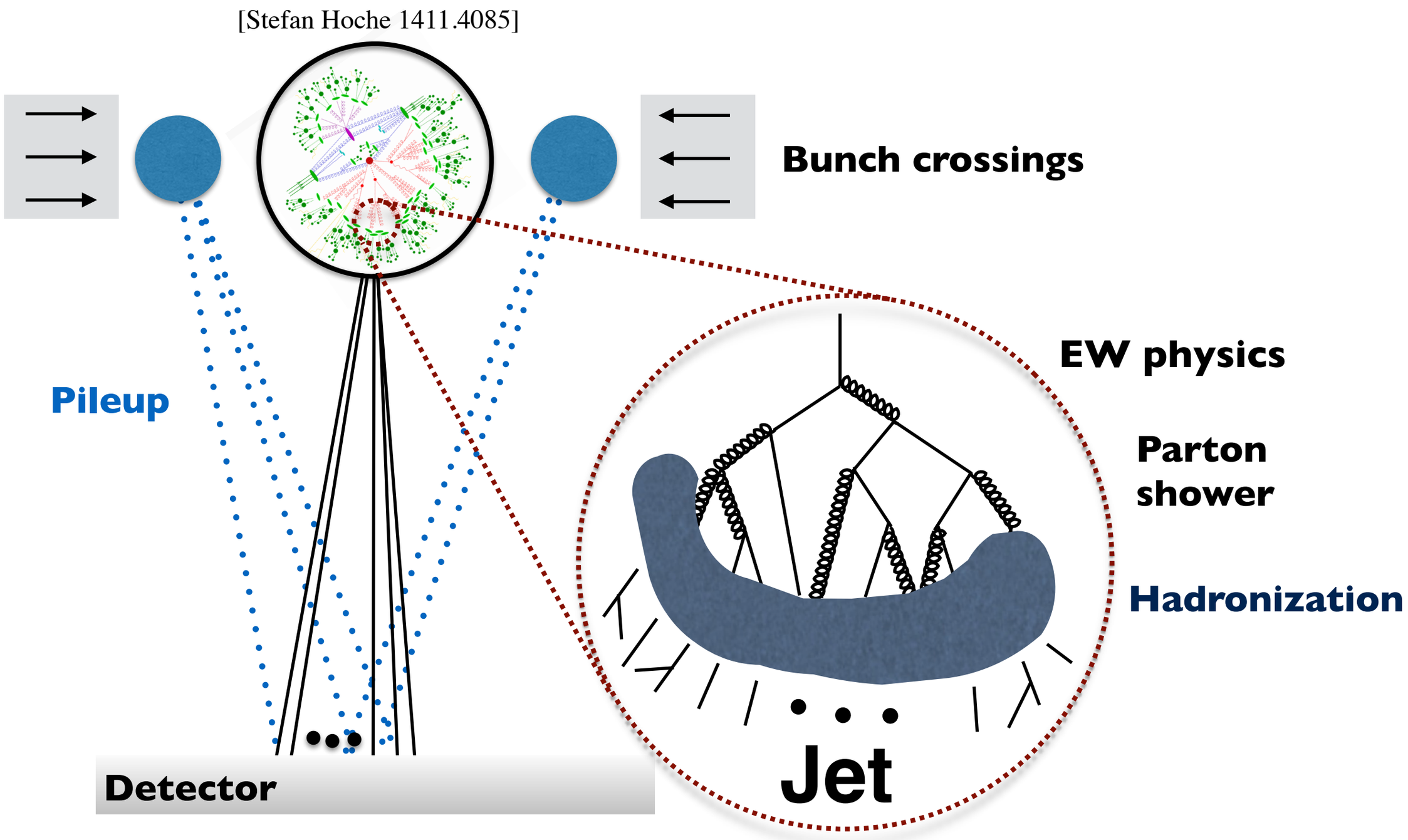
JETS

Collimated spray of energetic charged and neutral hadrons

ATLAS EXPERIMENT

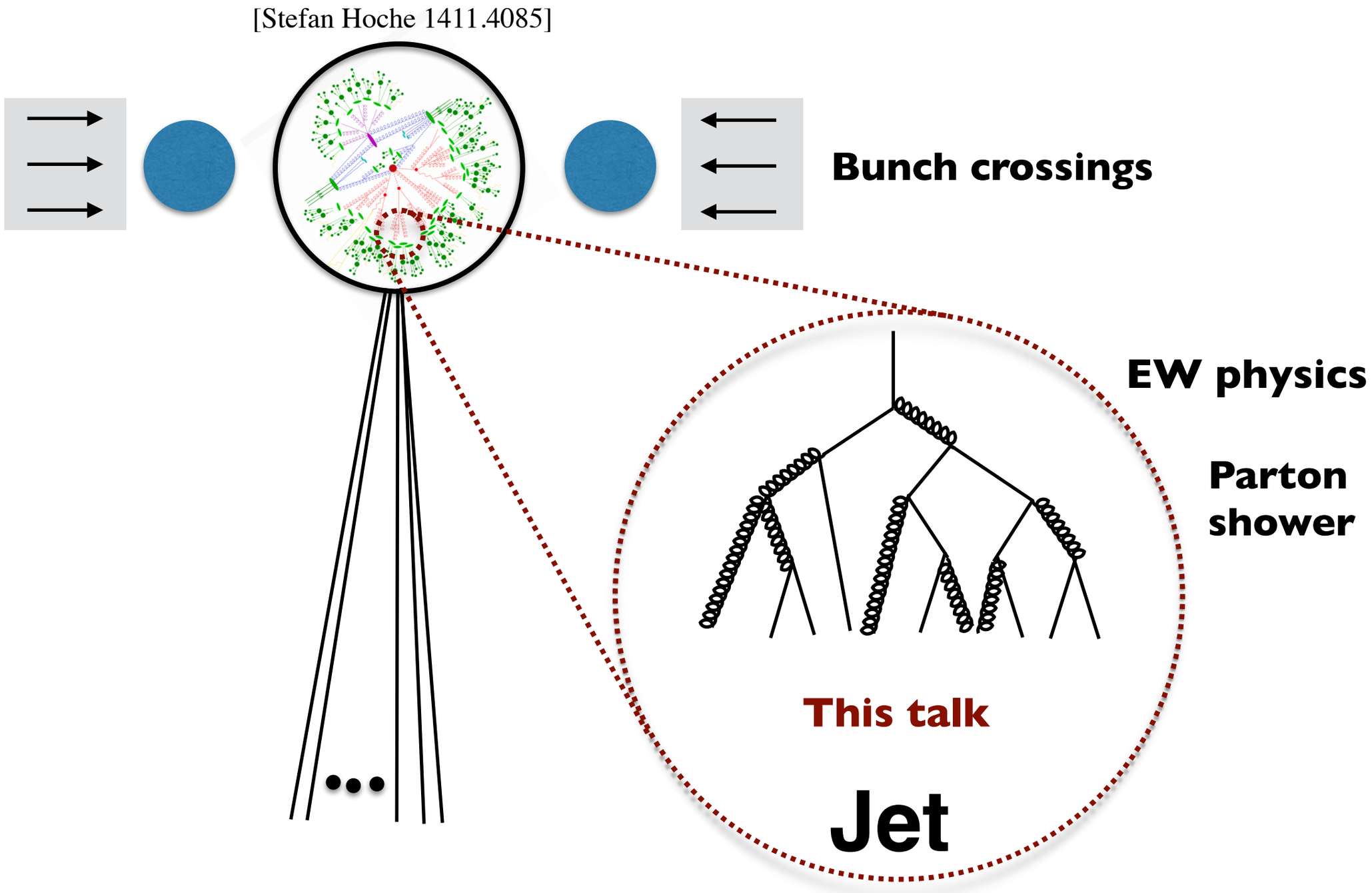
Run: 282712
Event: 474587238
2015-10-21 06:26:57 CEST

Hadron-Hadron Collisions

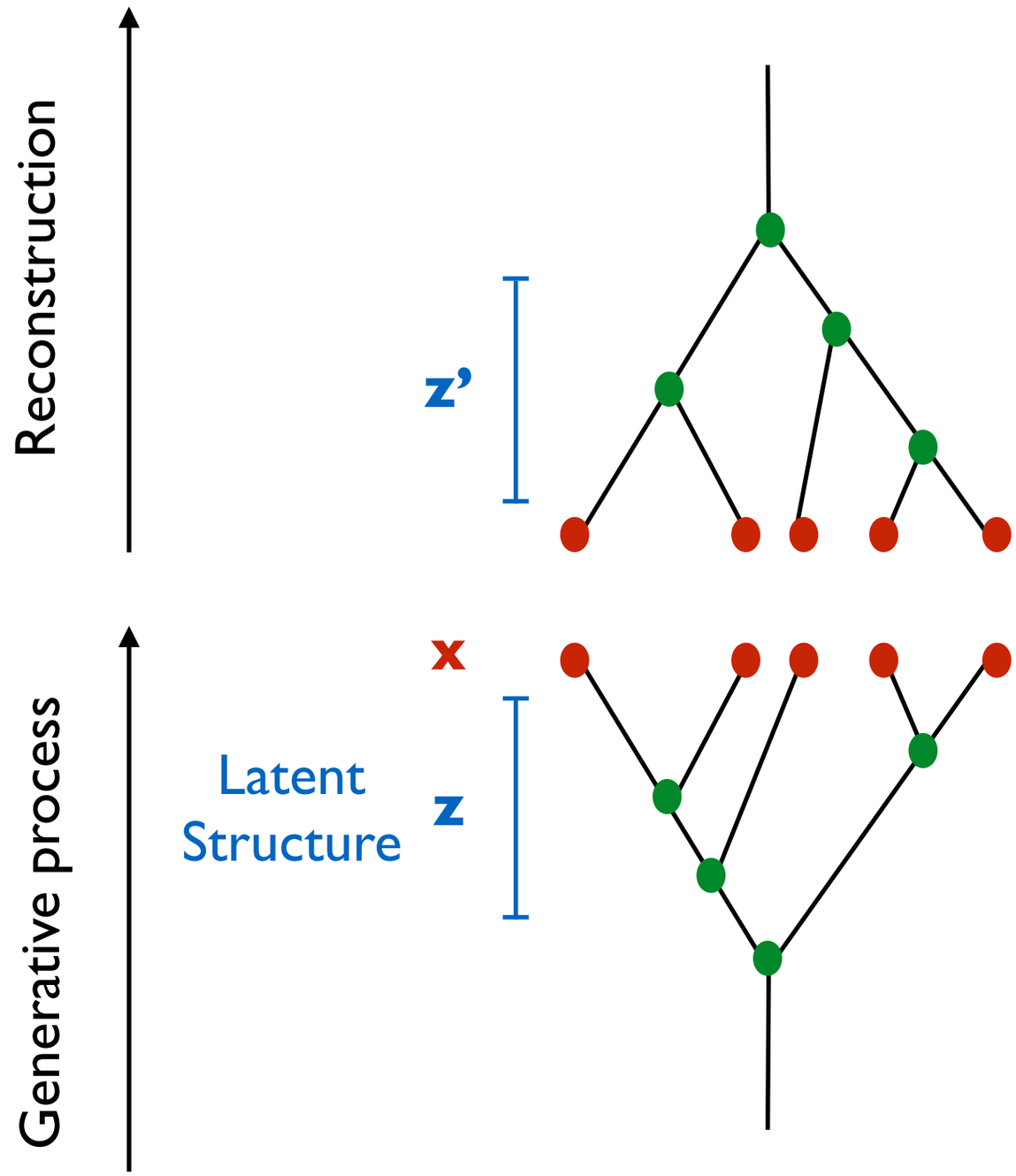


Hadron-Hadron Collisions

[Stefan Hoche 1411.4085]



Jet clustering



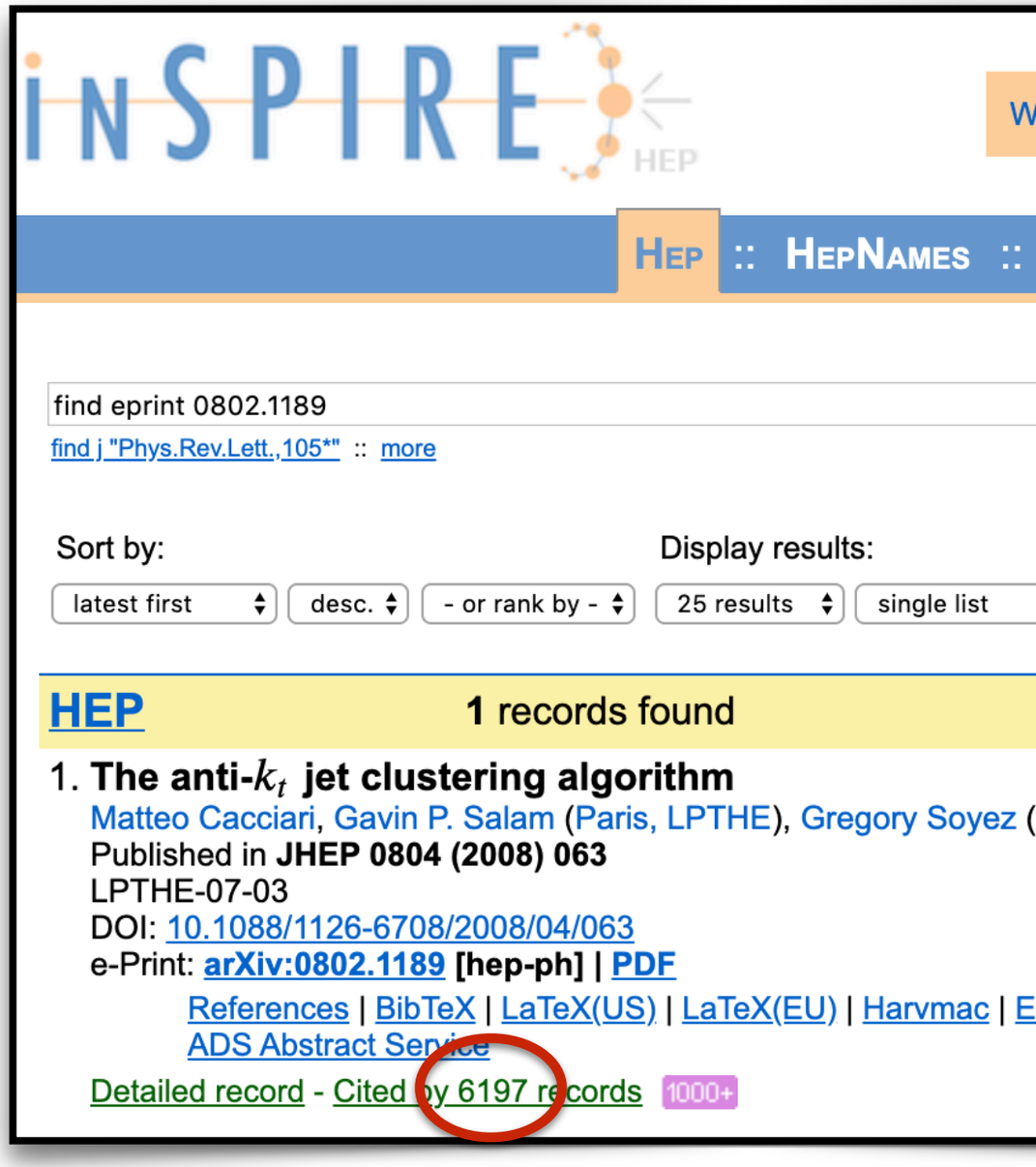
Generalized kt clustering algorithms

- "Heuristic" domain-specific clustering algorithms that do not use the generative model.
- Idea: sequentially cluster jet constituents.
- Permutation invariance: add 4-momenta of each pair that is clustered.
- Merge closest pair based on the distance measure:

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

$$\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$$

$\alpha = \{1, 0, -1\}$ specifies the the kt, Cambridge/ Aachen and anti-kt algorithms respectively.

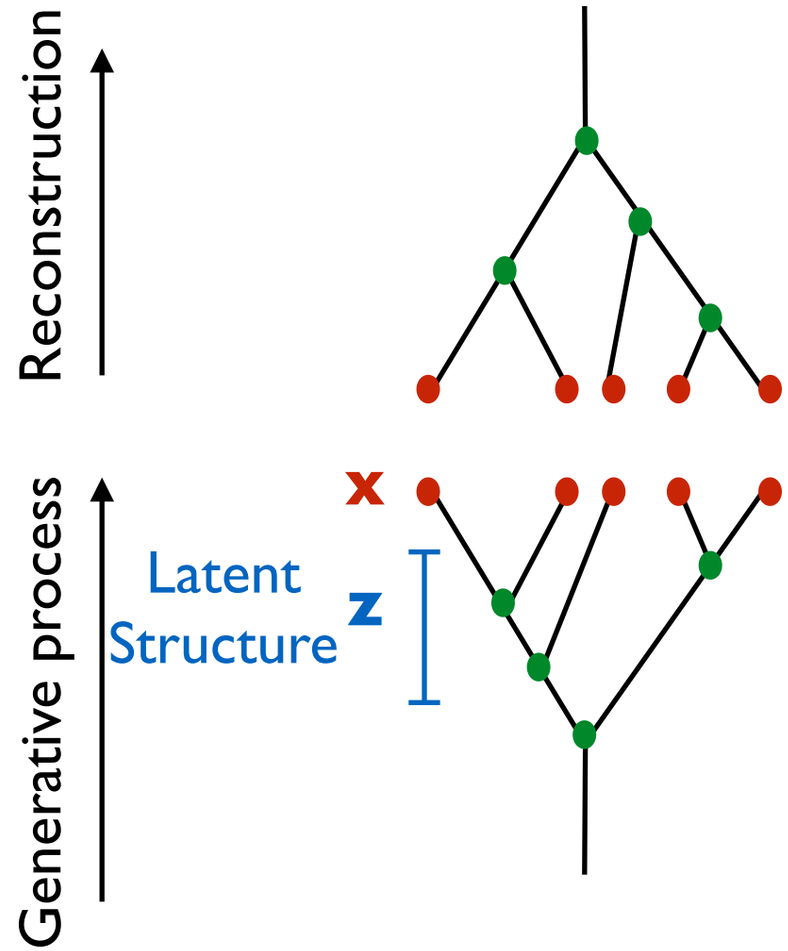


Problem settings

Classification

Maximum likelihood estimate

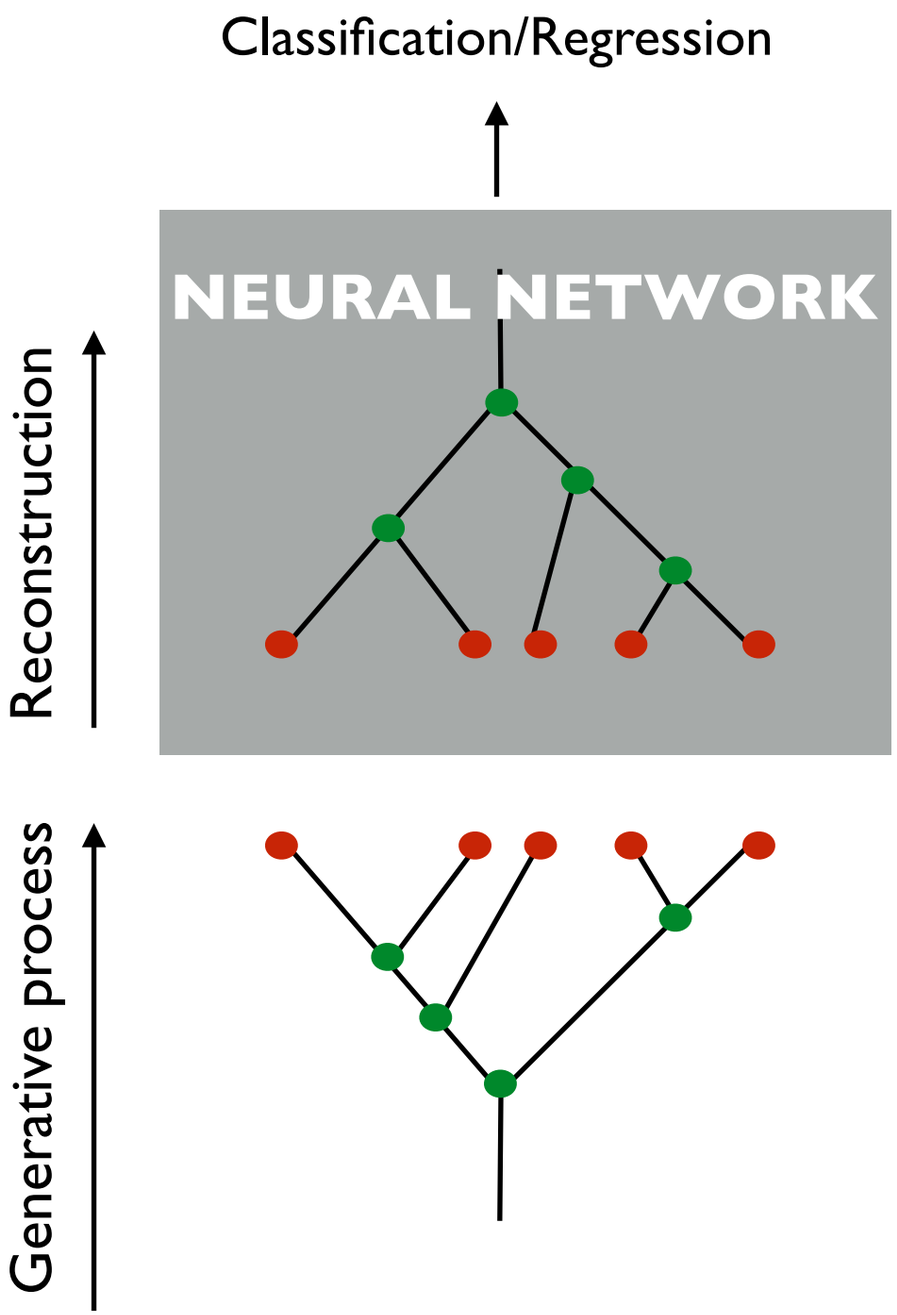
Posterior distribution on histories



Outline

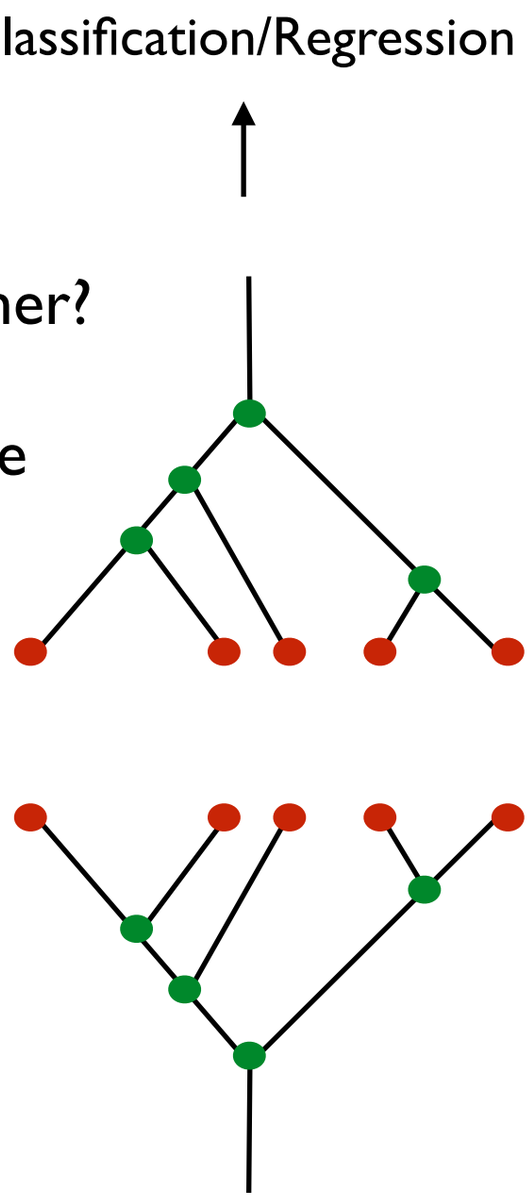
- **Neural networks and binary trees**
- **Unification of generation and inference**
- **Ginkgo: Toy Generative Model for Jets**
- **Maximum likelihood estimate**
- **Visualizations**
- **Preliminary results**
- **Final remarks and future directions**

Neural Networks that conserve the tree structure

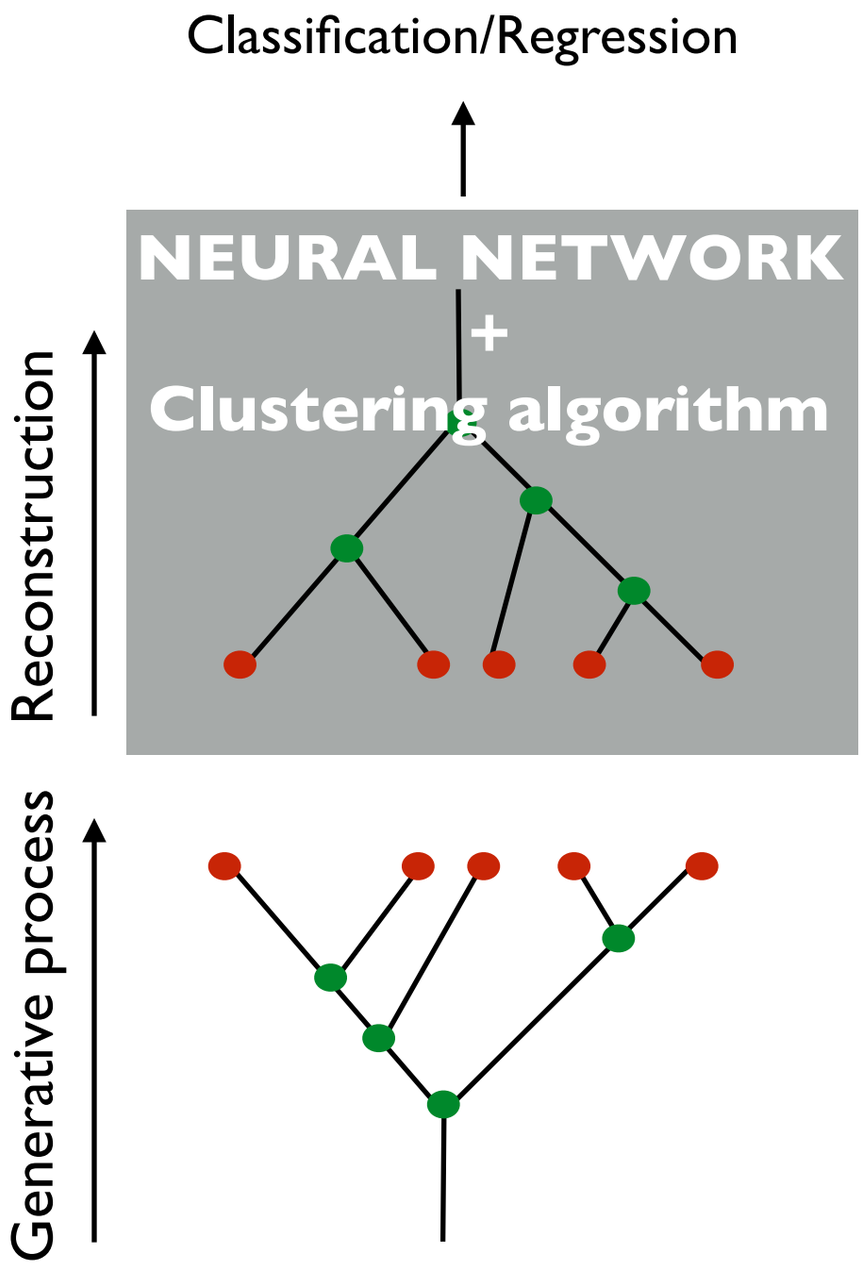


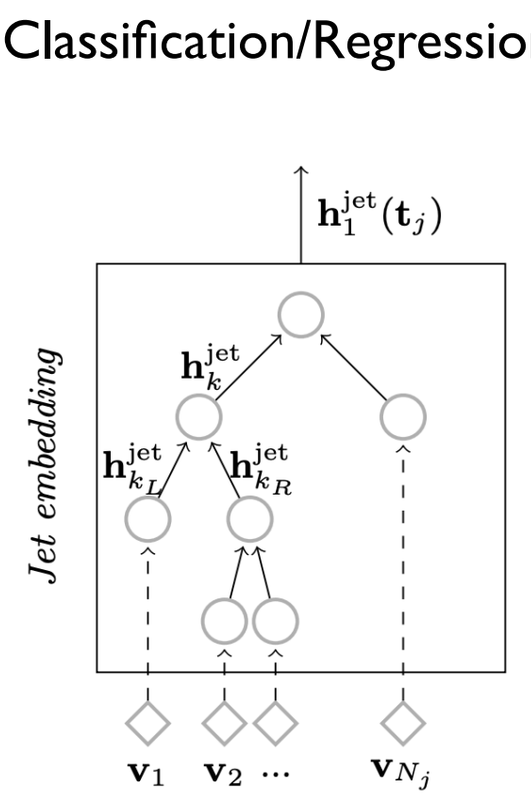
Can we solve both tasks together?

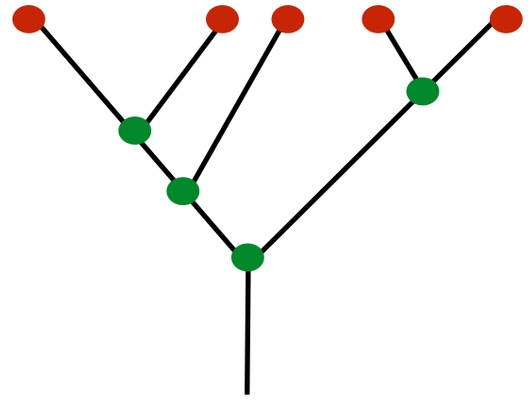
Find the right latent structure
+
classification/regression



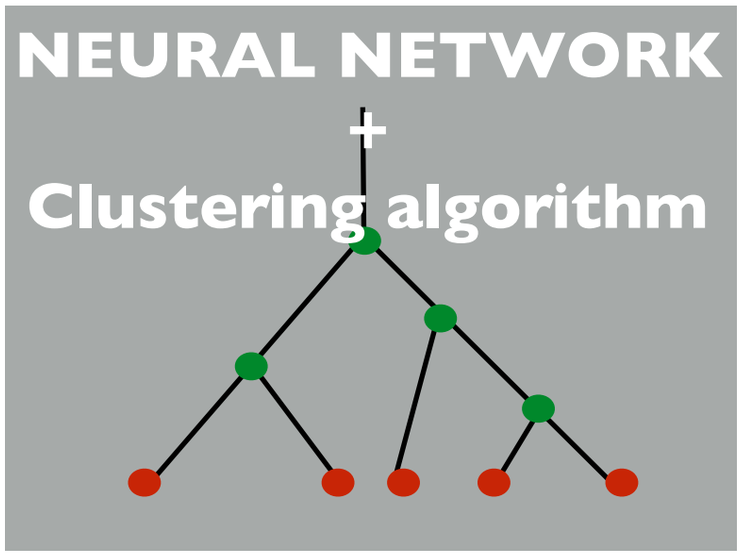
Neural Networks that conserve the tree structure







Neural Networks that conserve the tree structure



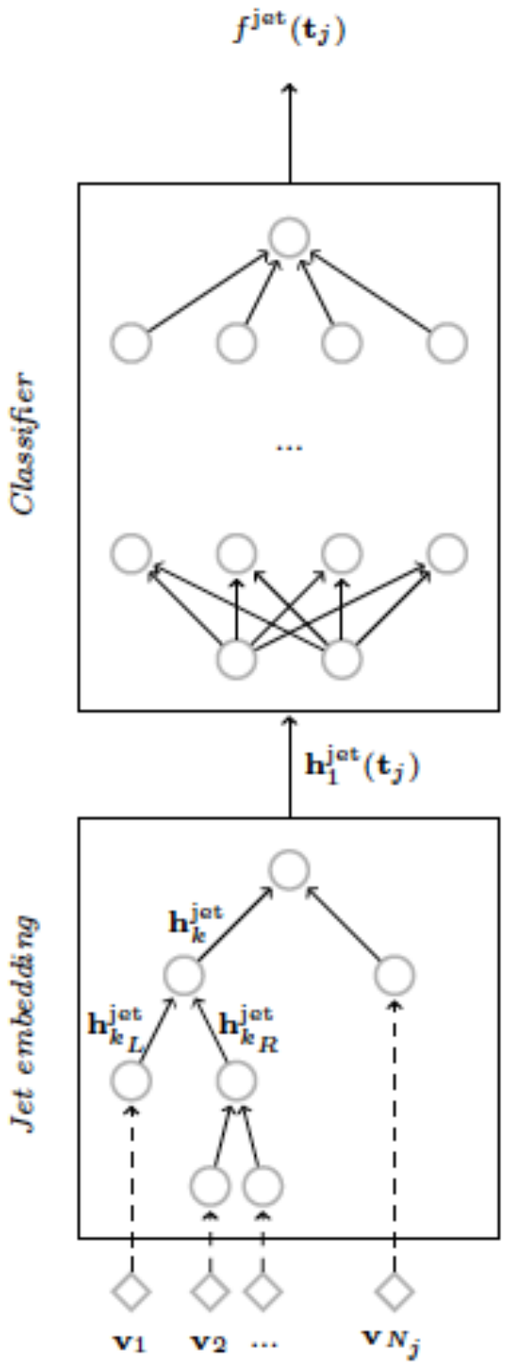
JUNIPR

JUNIPR
[Andreassen, Feige, Frye & Schwartz ’18]
Binary JUNIPR
[Andreassen, Feige, Frye & Schwartz ’19]

Recursive NN

RecNN
[Louppe, Cho, Becot & Cranmer ’17]
TreeNiN
[Macaluso & Cranmer ’19]

Recursive Neural Networks for Jet Physics



$$\mathbf{h}_k^{\text{jet}} = \begin{cases} \mathbf{u}_k & \text{if } k \text{ is a leaf} \\ \sigma\left(W_h \begin{bmatrix} \mathbf{h}_{k_L}^{\text{jet}} \\ \mathbf{h}_{k_R}^{\text{jet}} \\ \mathbf{u}_k \end{bmatrix} + b_h\right) & \text{otherwise} \end{cases}$$

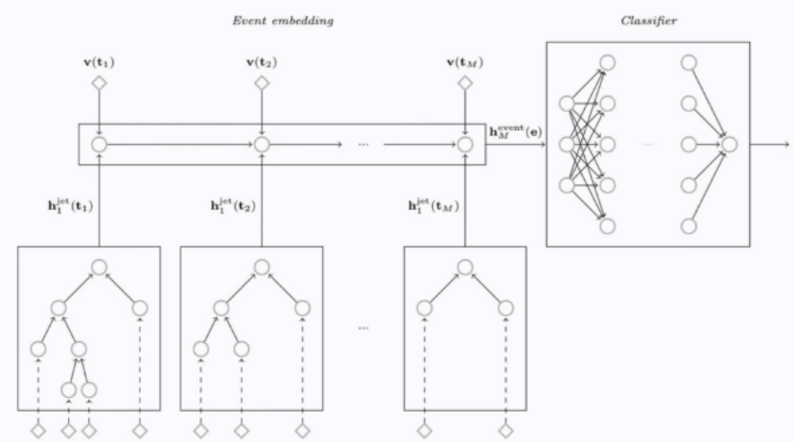
$$\mathbf{u}_k = \sigma\left(W_u g(\mathbf{o}_k) + b_u\right)$$

- RecNN much fewer trainable parameters: ~10000 vs ~1 M
- Low level input features:

$$|\vec{p}\,|, \eta, \phi, E, \frac{E}{E_{\text{jet}}}, p_{\text{T}}, \theta$$

[Louppe, Cho, Becot, Cranmer '17]

5. QCD-Aware Recursive Neural Networks for Jet Physics
Gilles Louppe, Kyunghun Cho, Cyril Becot, Kyle Cranmer (2017)



Stanford CS224N: NLP with Deep Learning Winter 2019 | Lecture 18 – Constituency Parsing, TreeRNNs

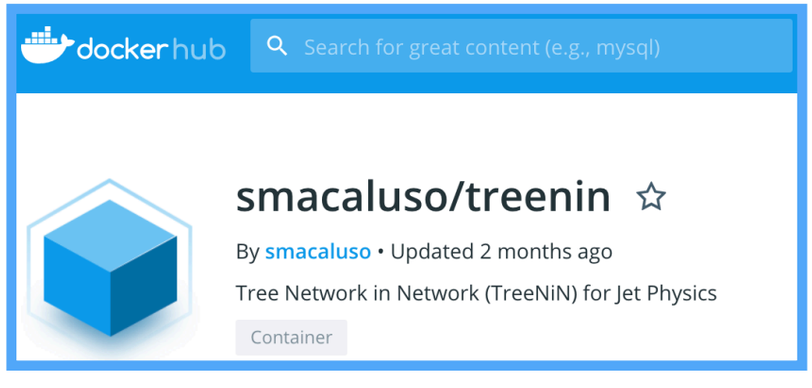
APRIL 5, 2019

Professor Christopher Manning, Stanford University http://onlinehub.stanford.edu/ Professor Christopher Manning Thomas M. Siebel Professor in Machine Learning, Professor of Linguistics and

Tree Network in Network (TreeNiN) for Jet Physics

[Macaluso & Cranmer ‘19]

- Split weights between leaves, inners nodes and main subtrees.
- GPU implementation of a (RecNN + 2 NiN layers).



https://github.com/diana-hep/TreeNiN

Network in Network (NiN) Mechanism

- Extra layers within each node
  + non-linear activation function

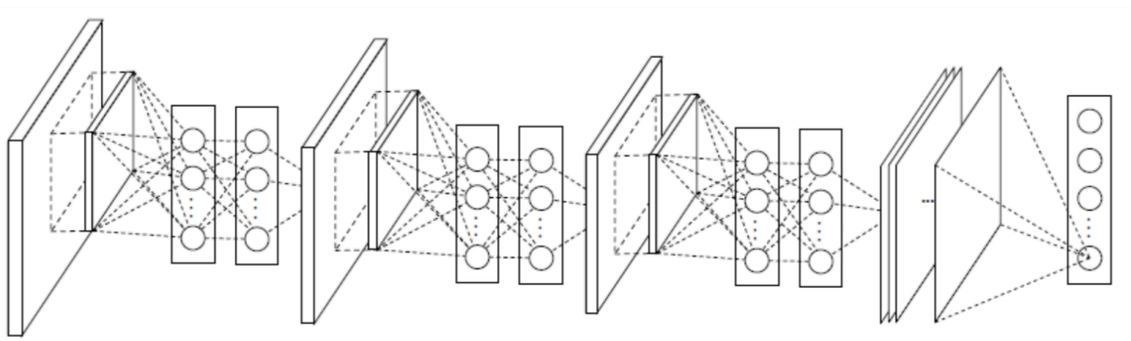
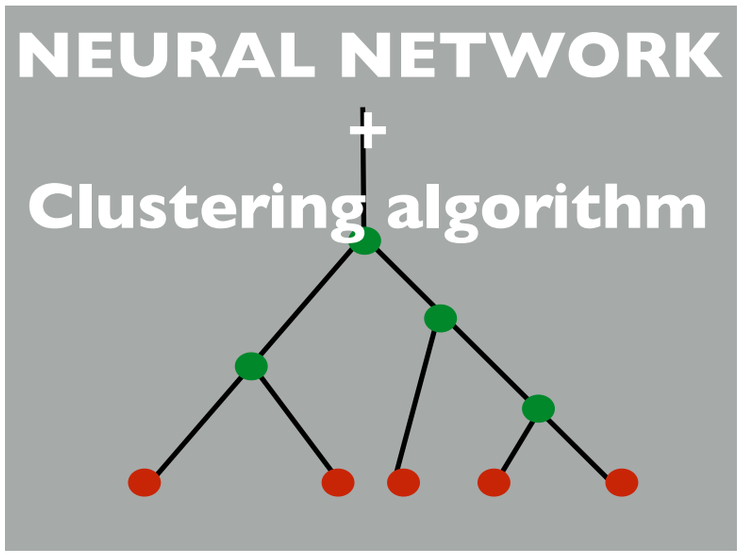


Fig. from arXiv:1312.4400

One of the best performing algorithms in “The Machine Learning Landscape of Top Taggers” comparison [arXiv:1902.09914]

Neural Networks that conserve the tree structure



JUNIPR

JUNIPR
[Andreassen, Feige, Frye & Schwartz ’18]
Binary JUNIPR
[Andreassen, Feige, Frye & Schwartz ’19]

Recursive NN

RecNN
[Louppe, Cho, Becot & Cranmer ’17]
TreeNiN
[Macaluso & Cranmer ’19]

JUNIPR: Jets from UNsupervised Interpretable PRobabilistic models

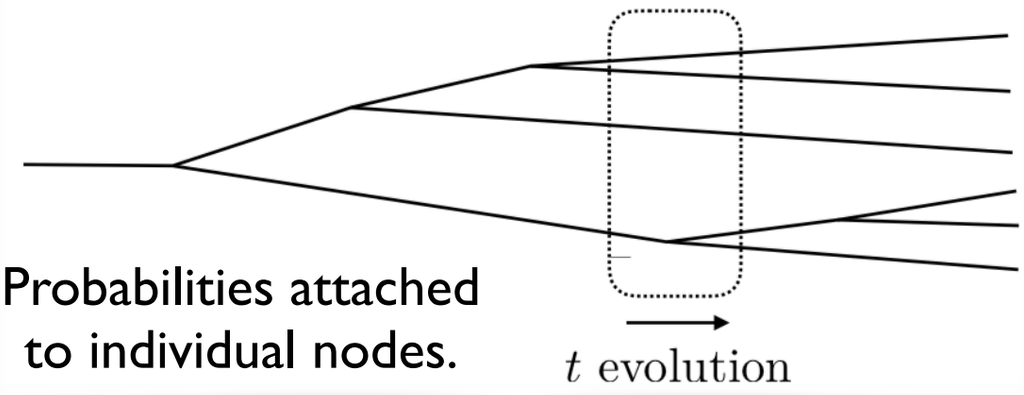
Yields an interpretable **probabilistic** model:

Allows to access the probability of each step in the clustering history of a jet (conditioned on the training set).

- **Discrimination**: based on likelihood ratio.
- **Sampling**: leads to data-driven simulations.
- Event **reweighting**.

- Tree structure is fixed by a clustering algorithm.
- Model is formally independent of chosen algorithm. (C/A better performance)

RNN implementation



JUNIPR [Andreassen, Feige, Frye & Schwartz '18]
Binary JUNIPR Andreassen, Feige, Frye & Schwartz '19]

- Could we learn physics structure from fitting to data?
- Could we have a unified picture? Same physics model for generation an inference.

Generation and inference unification

Aim to invert the generative model

- Inference on model parameters

  [Brehmer, Louppe, Pavez & Cranmer '18]
  [Brehmer, Cranmer, Louppe & Pavez '18]
  [Andreassen & Nachman '19]

$$p(\theta|x) = \frac{p(x|\theta)\ p(\theta)}{\int d\theta' p(x|\theta')p(\theta')}$$

- Marginal likelihood:

$$p(x|\theta) = \int dz\ p(x|z,\theta)\ p(z,\theta)$$

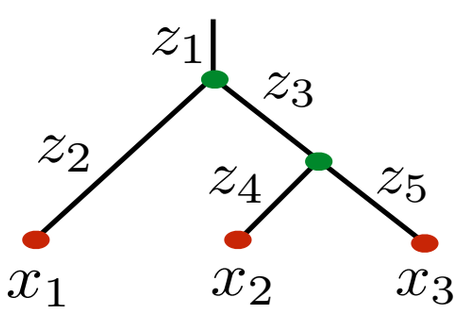
- Attempt to find the most likely splitting history
- Reconstructing the splitting history is like inverting the generative model.

- **Hard**! There are a combinatorial number of histories to consider.
- Usually jet reconstruction algorithms don't use the probability model directly.
- We use the likelihood for the history as the optimization **objective.**

Latent Structure

z

x



Tree joint likelihood:

$$p(x, z|\theta) = \prod_j p(x_j|z_{\text{parent}(x_j)}, \theta) \prod_i p(z_i|z_{\text{parent}(z_i)}, \theta)$$

Generation and inference unification

Aim to invert the generative model

- Estimate the posterior distribution on histories conditioned on the observed particles.

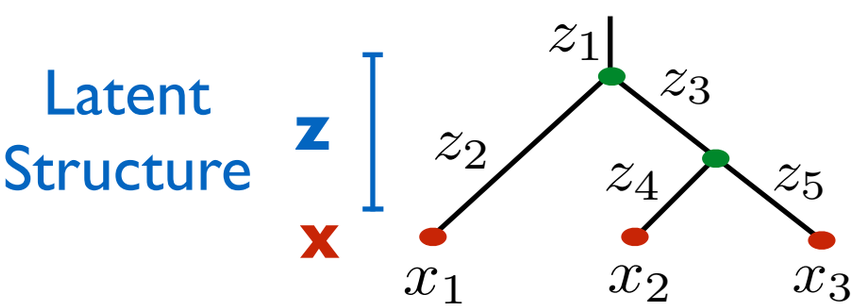
(Distribution over generated jet)

$$p(z|x,\theta) = \frac{p(x|z,\theta)\ p(z|\theta)}{p(x|\theta)}$$

- Marginal likelihood:

$$p(x|\theta) = \int dz\ p(x|z,\theta)\ p(z,\theta)$$

Sampling with MCMC or probabilistic programing techniques ?



- Related work: Q-jets
[Ellis, Hornig, Roy, Krohn & Schwartz '12]

(Distribution over reconstructed jet)

Assemble trees via a series of 2 to 1 mergings:

- Compute a weight for every pair at each stage:

$$\omega_{ij}^{(\alpha)} \equiv \exp\left(-\alpha \frac{(d_{ij} - d^{\min})}{d^{\min}}\right)$$

- Choose a pair with probability $\Omega_{ij} = \omega_{ij}/N$

- Related work: Shower deconstruction
[Soper & Spannowsky '11] [Soper & Spannowsky '12]

- Simplified shower model.
- Add likelihood of each clustering history over *microjets* conditioned on S or B.
- Define the likelihood ratio: $\chi(p_N) = \frac{P(p_N|S)}{P(p_N|B)}$

Parton Showers

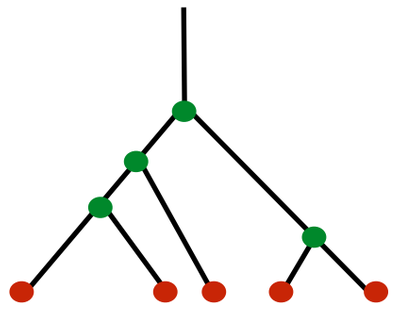
- It is hard to access the joint likelihood of a showering process.

- Typically, four-momentum conservation generates cross correlations between different splittings.

➡ Cannot write the joint likelihood as the product of single branch probabilities:

$$\mathcal{P}_{\mathrm{PS}}(\{t_i, z_i\}) = \mathcal{P}(t_1, z_1) \times \mathcal{P}(t_2, z_2) \times \cdots$$

(Analytic control could be restored [Bauer & Tackmann '08])

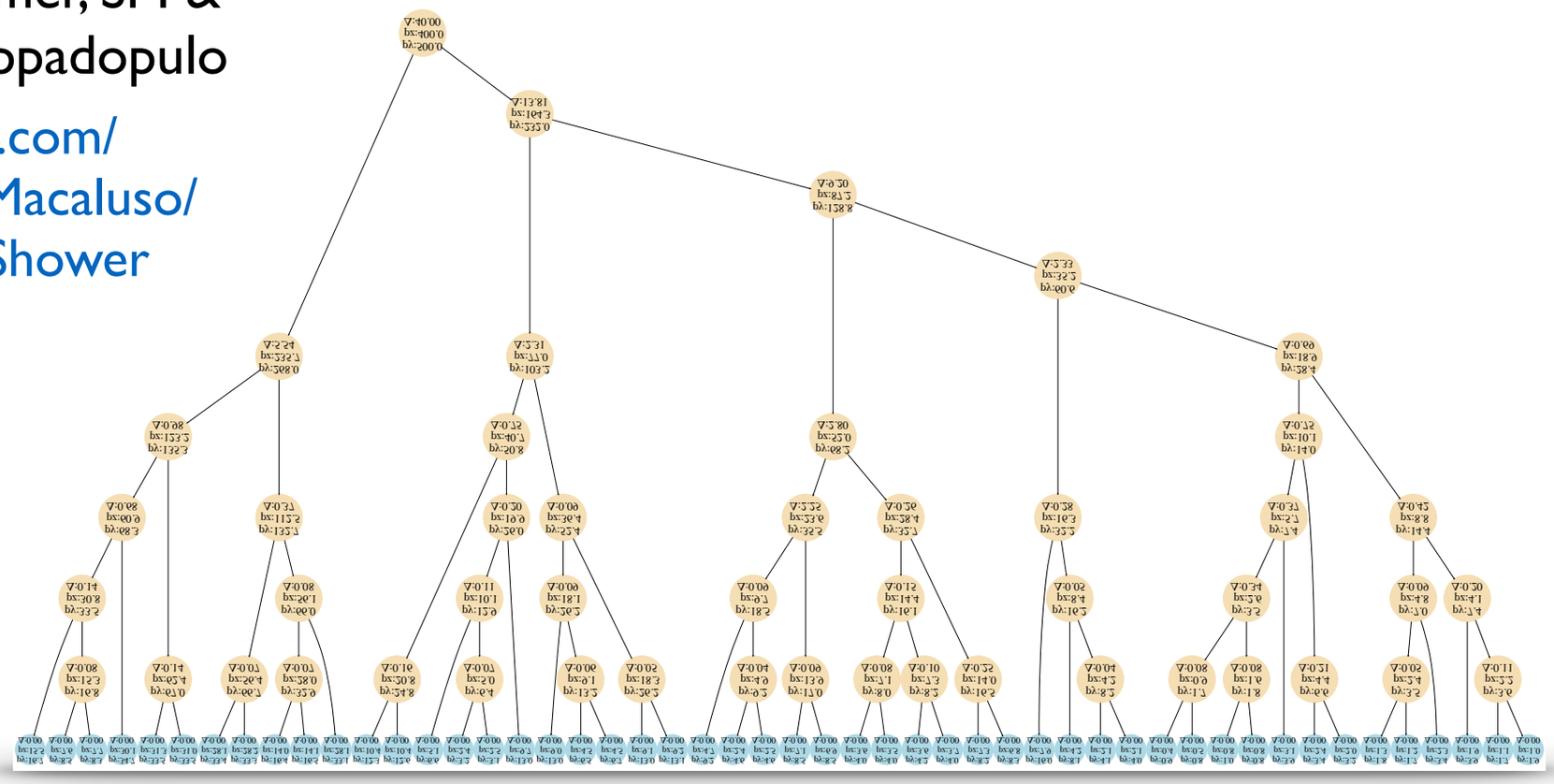


To prototype we built our own toy model!

Ginkgo: Toy Generative Model for Jets

Kyle Cranmer, SM &
Duccio Pappadopulo

github.com/
SebastianMacaluso/
ToyJetsShower



Motivation

- Build a model to aid in ML research for jet physics.

Generation

- Tractable joint likelihood.
- Captures essential ingredients of parton shower generators in full physics simulations.
- Implements an analogue to a parton shower (no hadronization effects).
- Python implementation with few software dependencies.

Inference

- E.g. tuning of simulation parameters (PYTHIA TUNES) to optimize a fit to the data.

Model Specification

- Stand-alone mathematical **specification** of model.
- Document describing model to non-physicists.
- Algorithm described in computer science style.
- Python implementation with few software dependencies.

Toy Generative Model for Jets

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 Introduction

In this notes, we provide a standalone description of a generative model to aid in machine learning (ML) research for jet physics. The motivation is to build a model that 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. The aim is for the model to have a python implementation with few software dependencies.

Parton shower generators are software tools that encode a physics model for the simulation of jets that are produced at colliders, e.g. the Large Hadron Collider at CERN. Jets are a collimated spray of energetic charged and neutral particles. Parton showers generate the particle content of a jet, going through a cascade process, starting from an initial unstable particle. In this description, there is a recursive algorithm that produces binary splittings of an unstable parent particle into two children particles, and a stopping rule. Thus, starting from the initial unstable particle, successive splittings are implemented until all the particles are stable (i.e. the stopping rule is satisfied for each of the final particles). We refer to this 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 and straightforward 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.

1

Model description

Recursive algorithm to generate a binary tree with jet constituents as the leaves.
Showering process: binary splittings + stopping rule.

Features

- Momentum conservation
- Running of the splitting scale

Algorithm 1: Toy Parton Shower Generator

function NodeProcessing $(\vec{p}_{\rm p}, \Delta_{\rm p}, \Delta_{\rm cut}, \lambda, \text{tree})$

Input : parent momentum $\vec{p}_{\rm p}$, parent splitting scale $\Delta_{\rm p}$, cut-off scale $\Delta_{\rm cut}$, rate for the exponential distribution λ, binary tree *tree*

Add parent node to tree.

if $\Delta_p > \Delta_{cut}$ **then**

draw ϕ from uniform distribution in $\{0, 2\pi\}$

$\vec{\Delta}_{\rm p} = \Delta_{\rm p}\ (\sin\phi_{\rm p}, \cos\phi_{\rm p})$

$\vec{p}_{\rm L} = \frac{1}{2}\vec{p}_{\rm p} - \vec{\Delta}_{\rm p}$

$\vec{p}_{\rm R} = \frac{1}{2}\vec{p}_{\rm p} + \vec{\Delta}_{\rm p}$

draw $\Delta_{\rm L}$, $\Delta_{\rm R}$ from the exponential distribution in Eq. 3.

NodeProcessing $(\vec{p}_{\rm L}, \Delta_{\rm L}, \Delta_{\rm cut}, \lambda, \text{tree})$

NodeProcessing $(\vec{p}_{\rm R}, \Delta_{\rm R}, \Delta_{\rm cut}, \lambda, \text{tree})$

Generative process

Start with $\Delta_{\mathrm{P}} = \Delta_{\mathrm{root}}$

Sample $\phi \sim \mathrm{Uniform}(0, 2\pi)$ to pick a direction for Δ_{P}

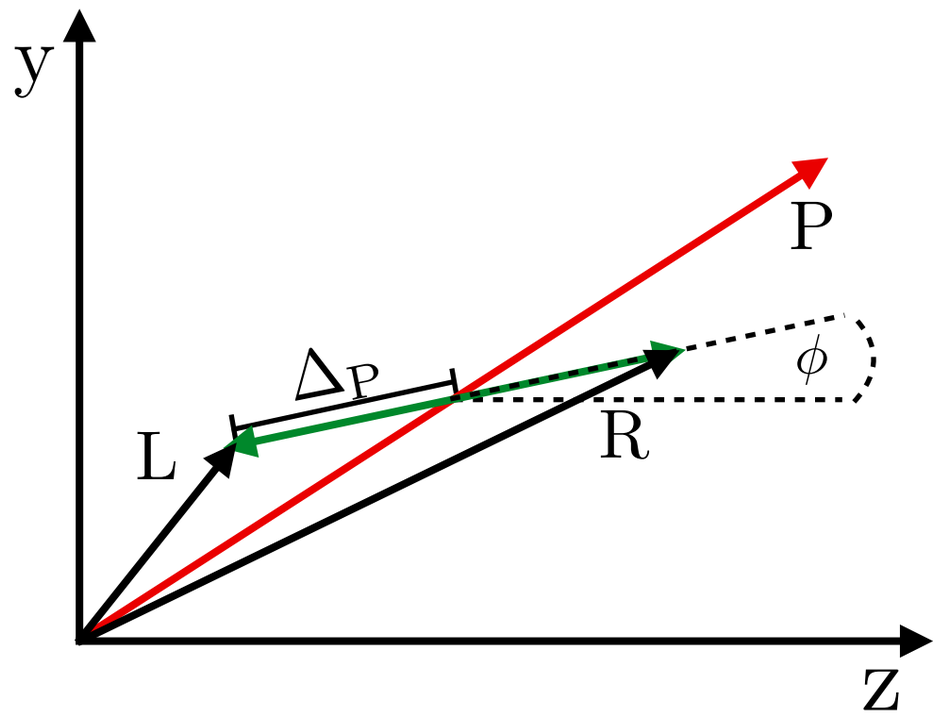
Heavy resonance X jet

$$\Delta_{\mathrm{root}} = \frac{m_X}{2}$$

Split parent node

$$\vec{p}_{\mathrm{L}} = \frac{1}{2}\vec{p}_{\mathrm{p}} - \vec{\Delta}_{\mathrm{p}}$$

$$\vec{p}_{\mathrm{R}} = \frac{1}{2}\vec{p}_{\mathrm{p}} + \vec{\Delta}_{\mathrm{p}}$$



Sample independently Δ_{L} and Δ_{R}

$$\Delta \sim f(\Delta|\lambda, \Delta_{\mathrm{p}}) = \frac{\lambda}{\Delta_{\mathrm{p}}} e^{-\frac{\lambda}{\Delta_{\mathrm{p}}}\Delta}$$

Recurse over (Δ_{L},L) and (Δ_{R},R).

Three latent variables per particle

$\vec{p} = (p_y, p_z)$ and Δ

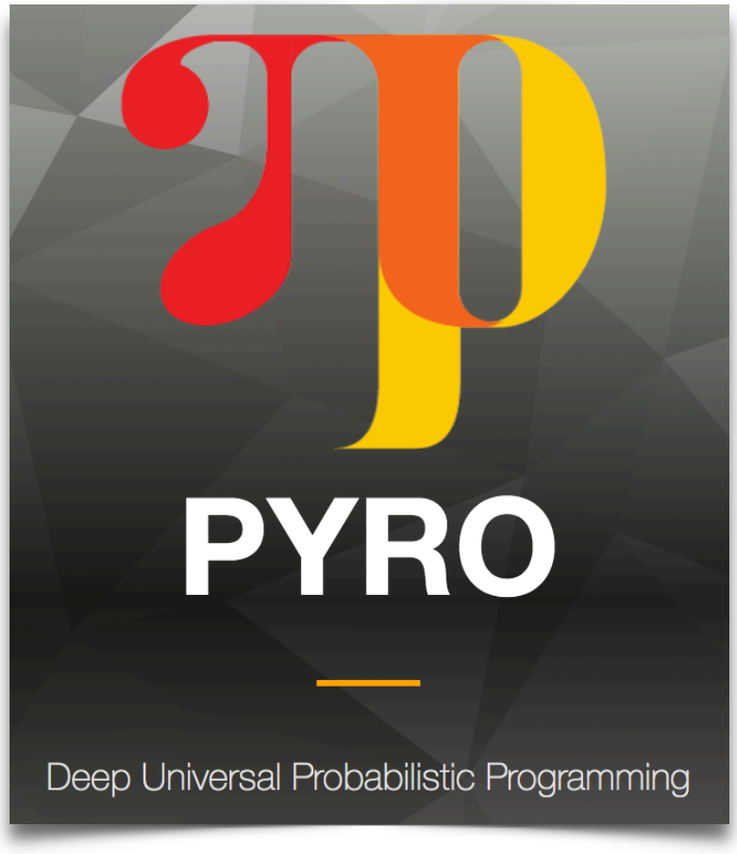
Augmented data

Additional information that characterizes the latent process can be extracted in Ginkgo:

- Joint likelihood
- Joint likelihood ratio
- Joint score

The model keeps track of the augmented data based on a **PYRO** implementation.

New techniques for likelihood-free inference can be applied.
[J. Brehmer, G. Louppe, J. Pavez & K. Cranmer '18]

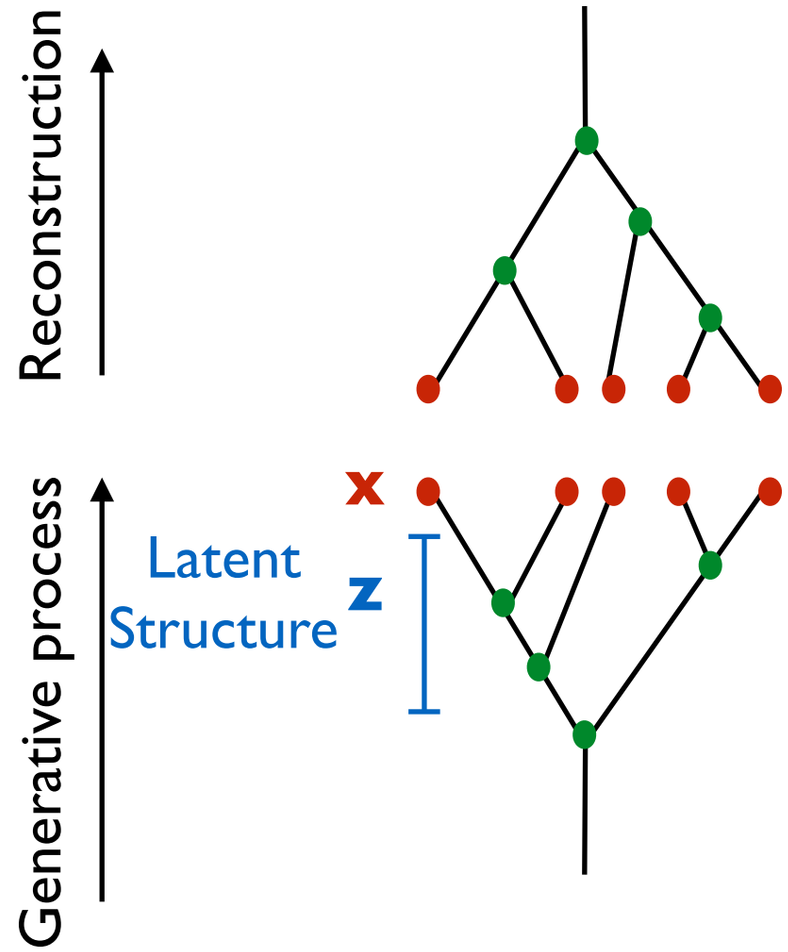


Problem settings

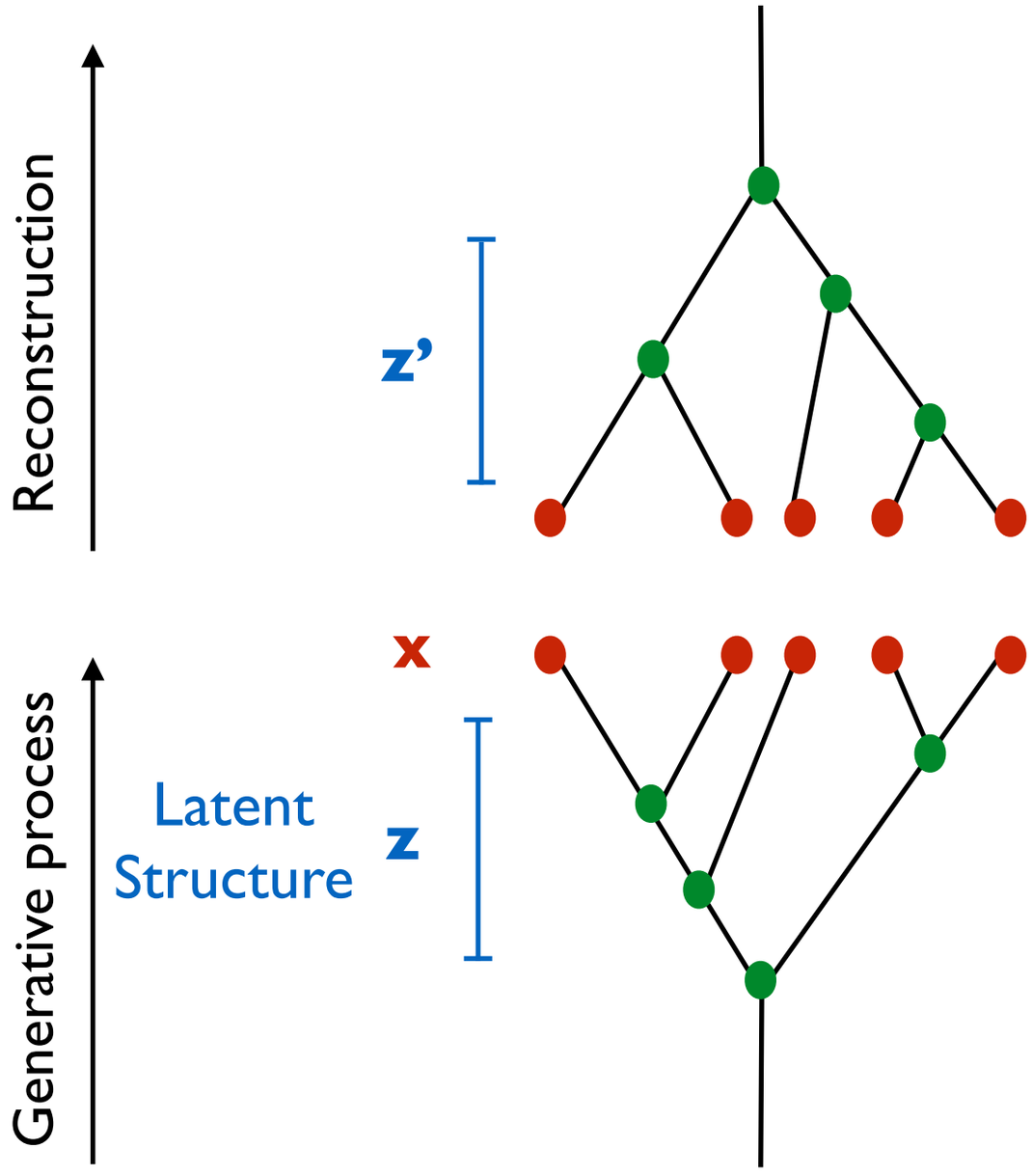
Classification

Maximum likelihood estimate

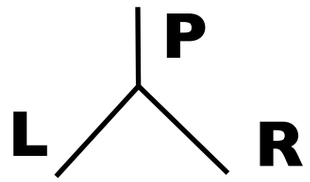
Posterior distribution on histories



Jet clustering



Can we cluster jets based on the likelihood of each splitting?



Joint likelihood:

$$p(x, z|\theta) = \prod_i p(\mathrm{L}_i, \mathrm{R}_i|\mathrm{P}_i, \theta)$$

Split likelihood:

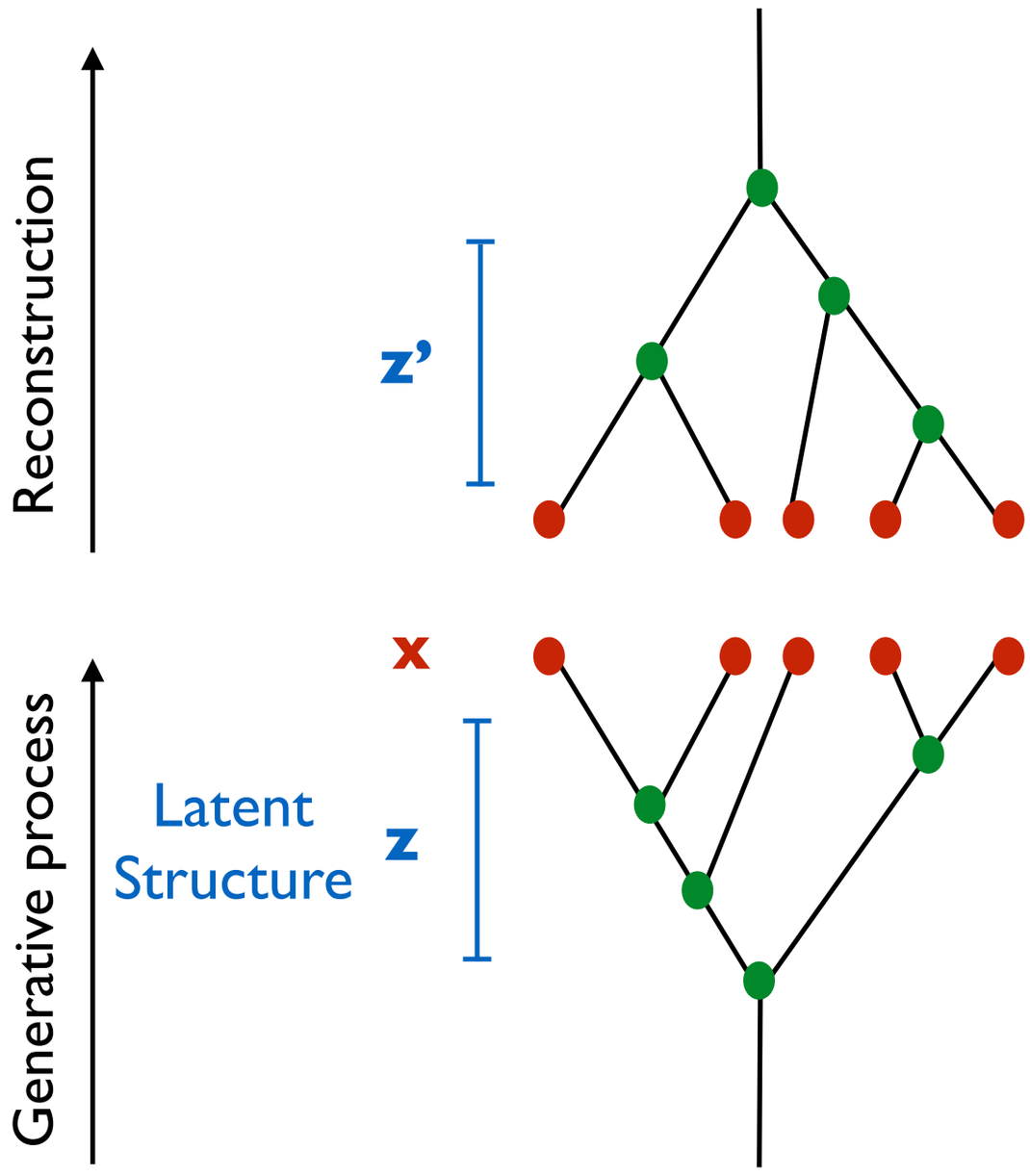
$$p(\mathrm{L}_i, \mathrm{R}_i|\mathrm{P}_i) \equiv p(\vec{p}_\mathrm{L}, \vec{p}_\mathrm{R}|\vec{p}_\mathrm{P})$$

---

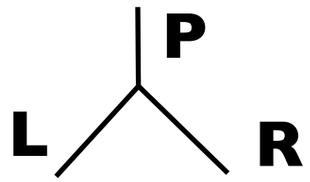
k_t distance measure

$$d(\mathrm{L}_i, \mathrm{R}_i) \equiv d(\vec{p}_\mathrm{L}, \vec{p}_\mathrm{R})$$

Jet clustering



Can we cluster jets based on the likelihood of each splitting?



Joint likelihood:

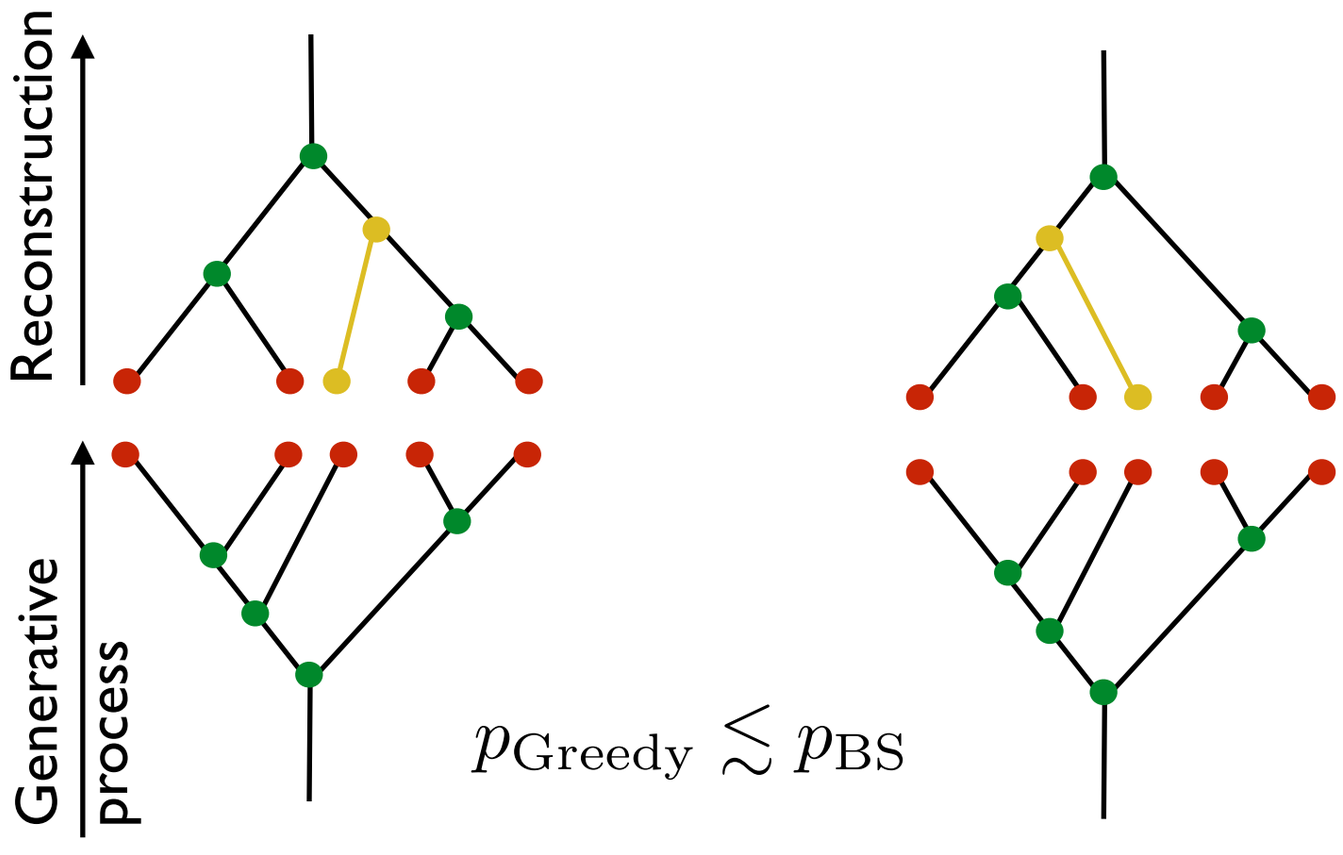
$$p(x, z|\theta) = \prod_i p(\mathrm{L}_i, \mathrm{R}_i|\mathrm{P}_i, \theta)$$

- Hard to do with parton shower generators in full physics simulations, e.g. PYTHIA
- To prototype we built our own toy model!

Likelihood-based jet clustering

Greedy
Joint likelihood from locally maximizing the likelihood at each step.

Beam Search
Maximize the likelihood of multiple steps before choosing the latent path. Maybe we could fix mistakes?



Joint likelihood:

$$p(x, z|\theta) = \prod_i p(\mathrm{L}_i, \mathrm{R}_i|\mathrm{P}_i, \theta)$$

Goal: find the latent structure that maximizes the jet likelihood (MLE).

Given an algorithm, z is fixed.

$\hat{z}_{\text{Greedy}}$

$\hat{z}_{\text{BS}}$

$\hat{z}_{\text{MLE}} = \text{argmax}_z\ p(x|z)$

Viterbi algorithm
Computationally "infeasible" for jets with ~ 100 constituents

Greedy algorithms

Choose the optimal decision locally at each step.

Jets properties

- Permutation invariance
- Distance measure

Generalized kt clustering algorithms

- Locally minimize the distance measure d_{ij}
- An analogue of the generalized k_t clustering algorithms can be defined for Ginkgo jets.

Greedy likelihood-based algorithm

- Choose the nodes pairing that locally maximizes the likelihood at each step given by

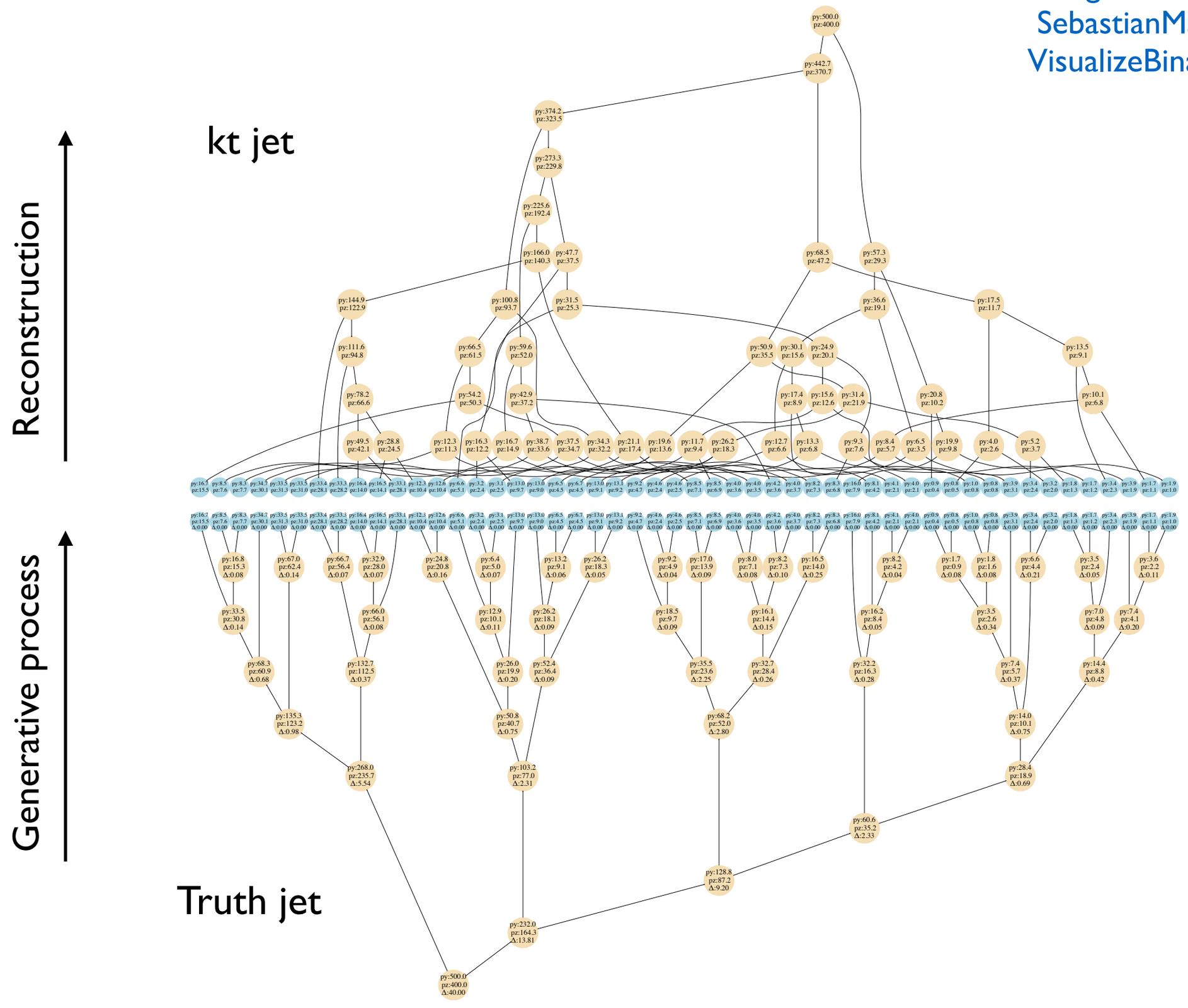
$$p(\Delta_{\mathrm{L}}, \Delta_{\mathrm{R}} | \Delta_{\mathrm{p}})$$

Our implementation improves the N^3 brute force time complexity to N^2.
Approach based on nearest neighbors introduced in FastJet
[hep-ph/0512210, G. Salam and M. Cacciari LPTHE-06-02].

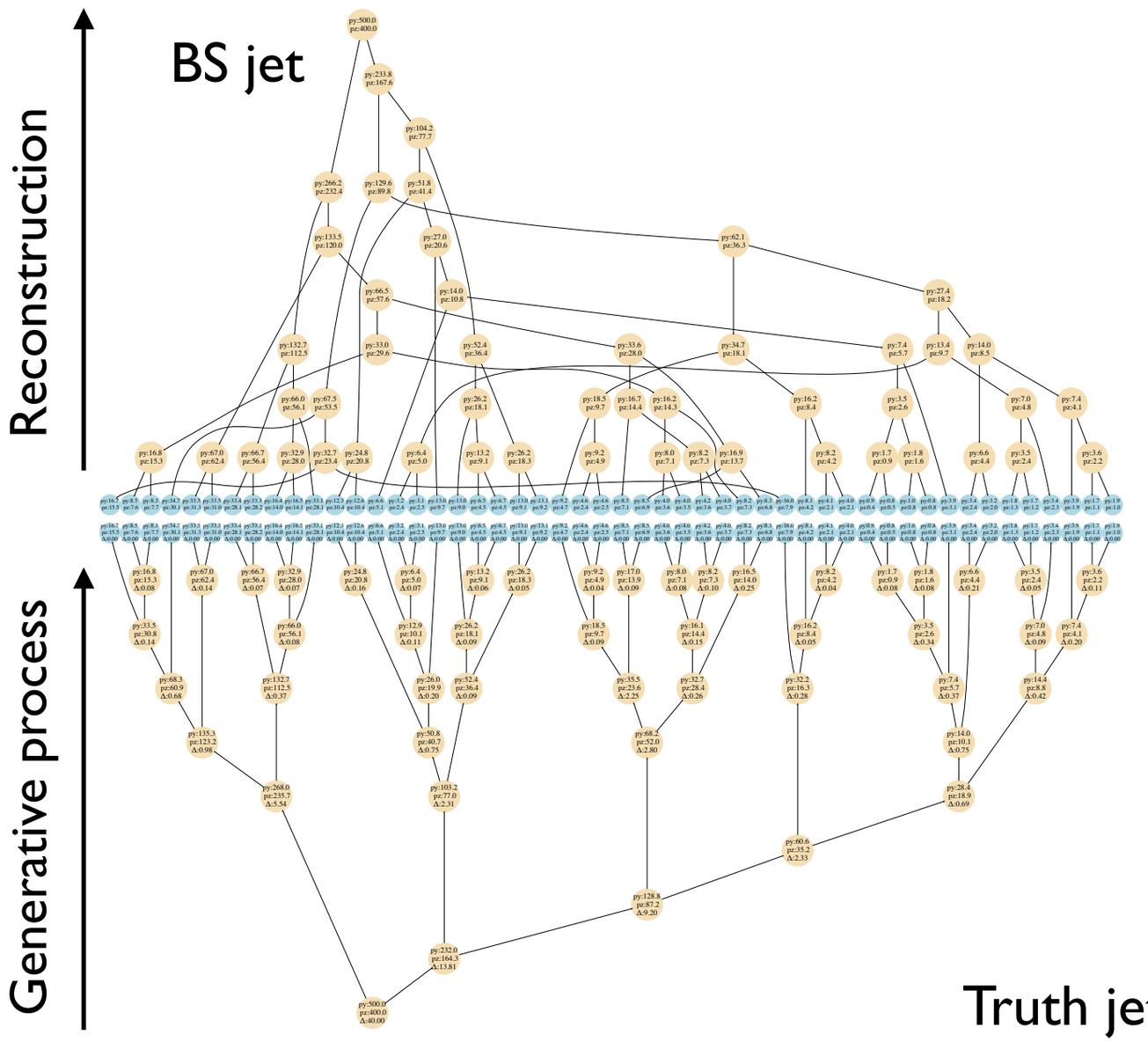
1-D binary tree visualizations

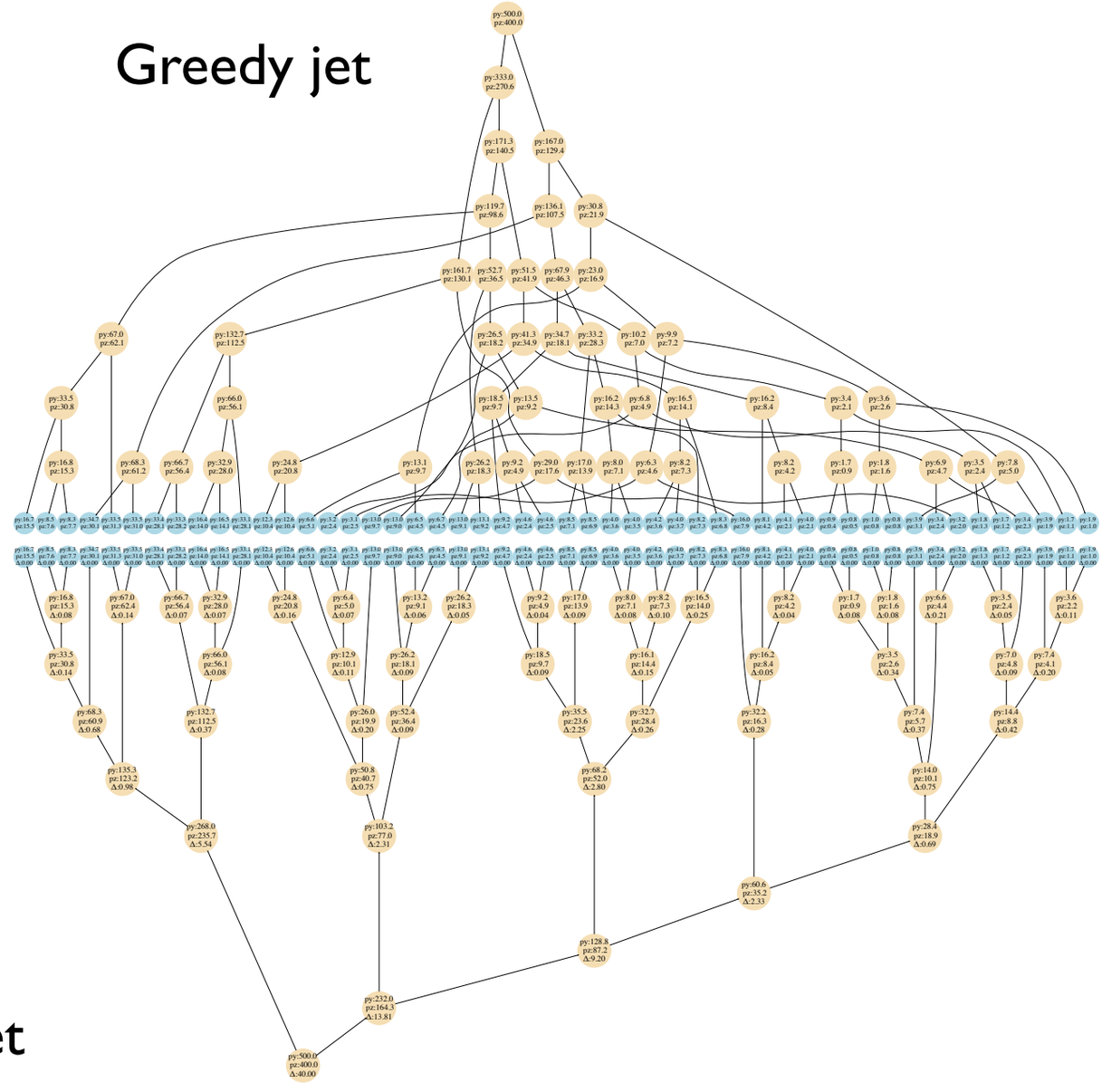
Kyle Cranmer, SM &
Duccio Pappadopulo

github.com/
SebastianMacaluso/
VisualizeBinaryTrees



1-D binary tree visualizations



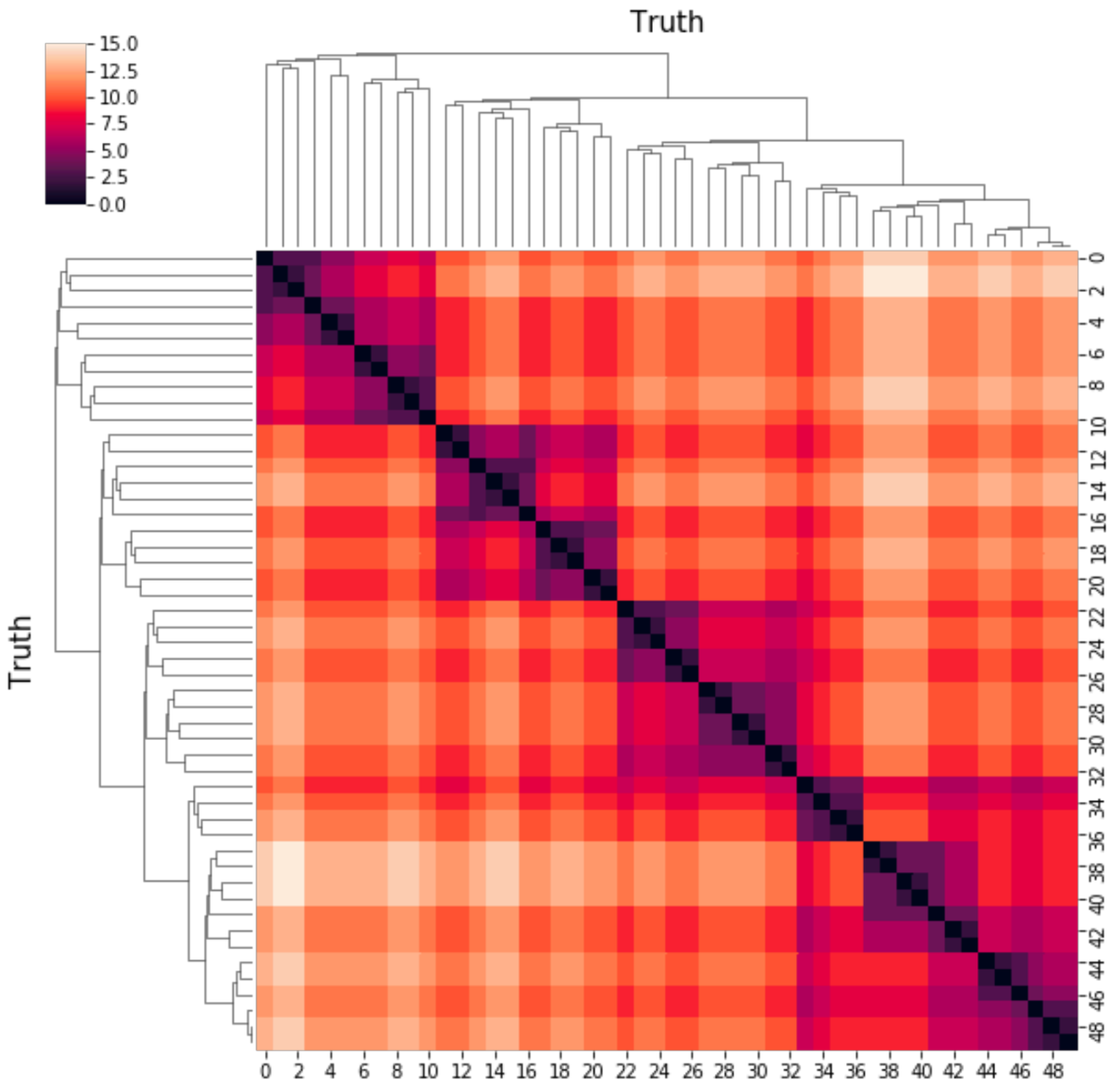


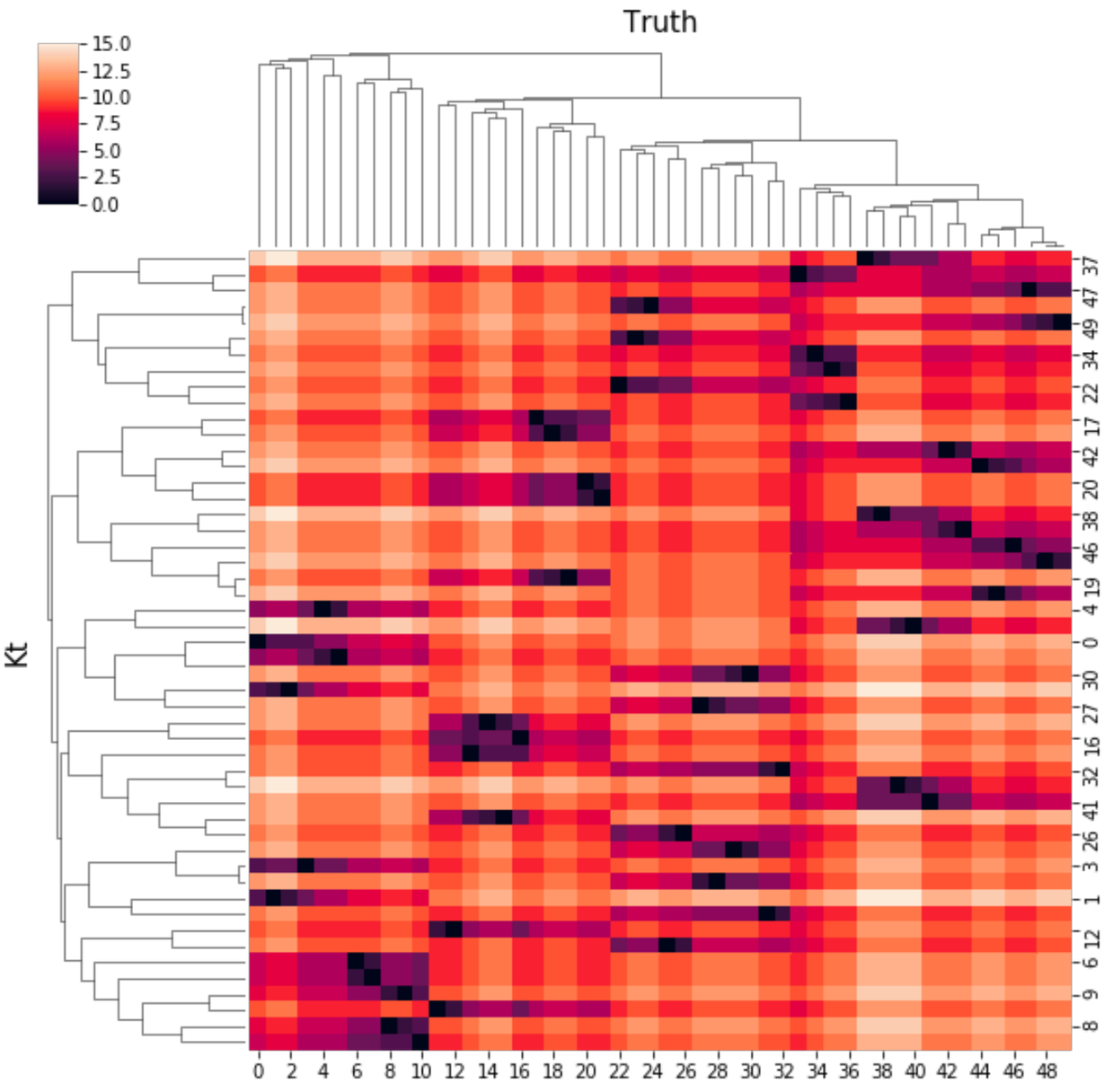
Truth jet

Heat cluster maps visualizations

Kyle Cranmer, SM &
Duccio Pappadopulo

github.com/SebastianMacaluso/VisualizeBinaryTrees

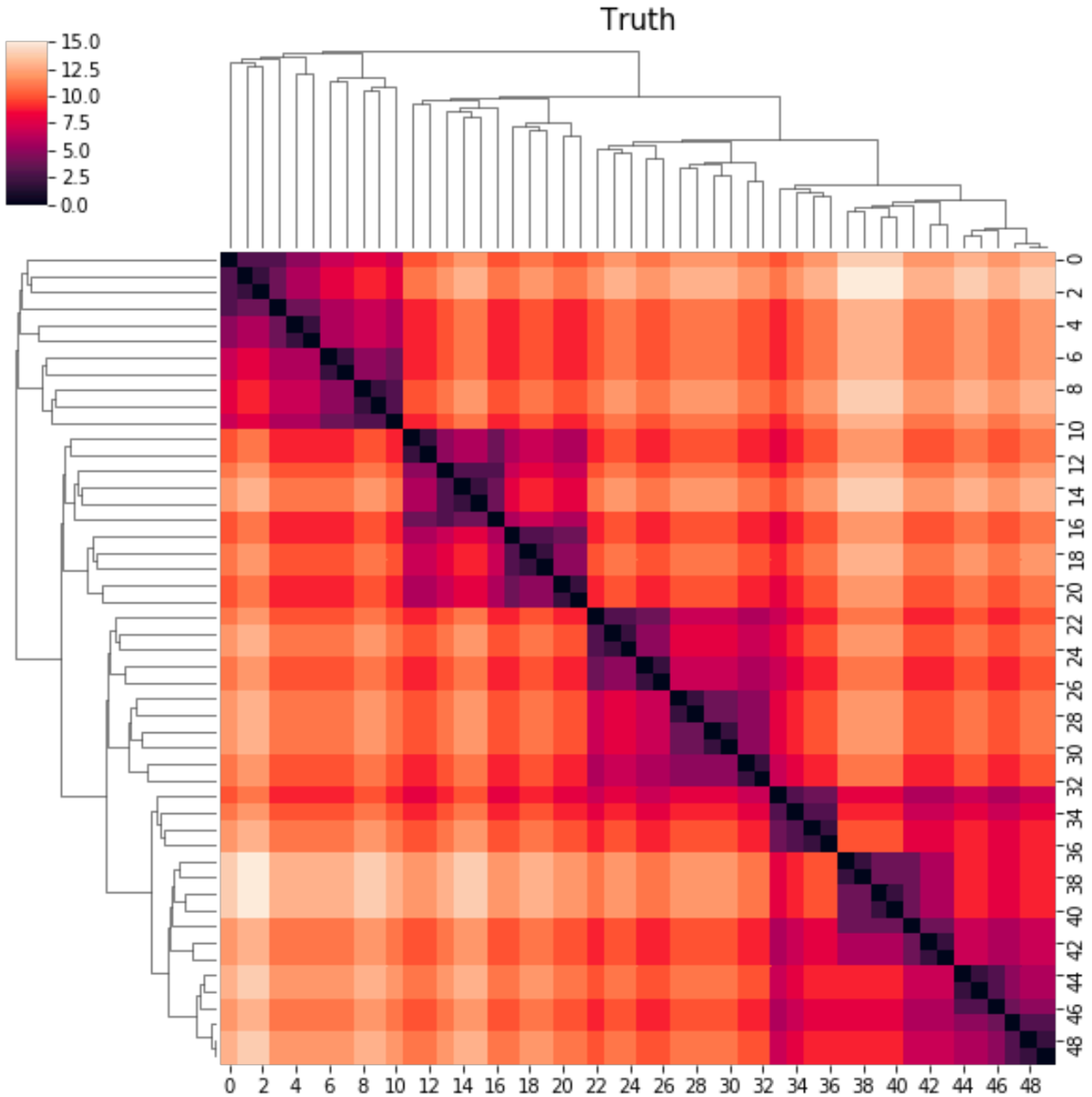


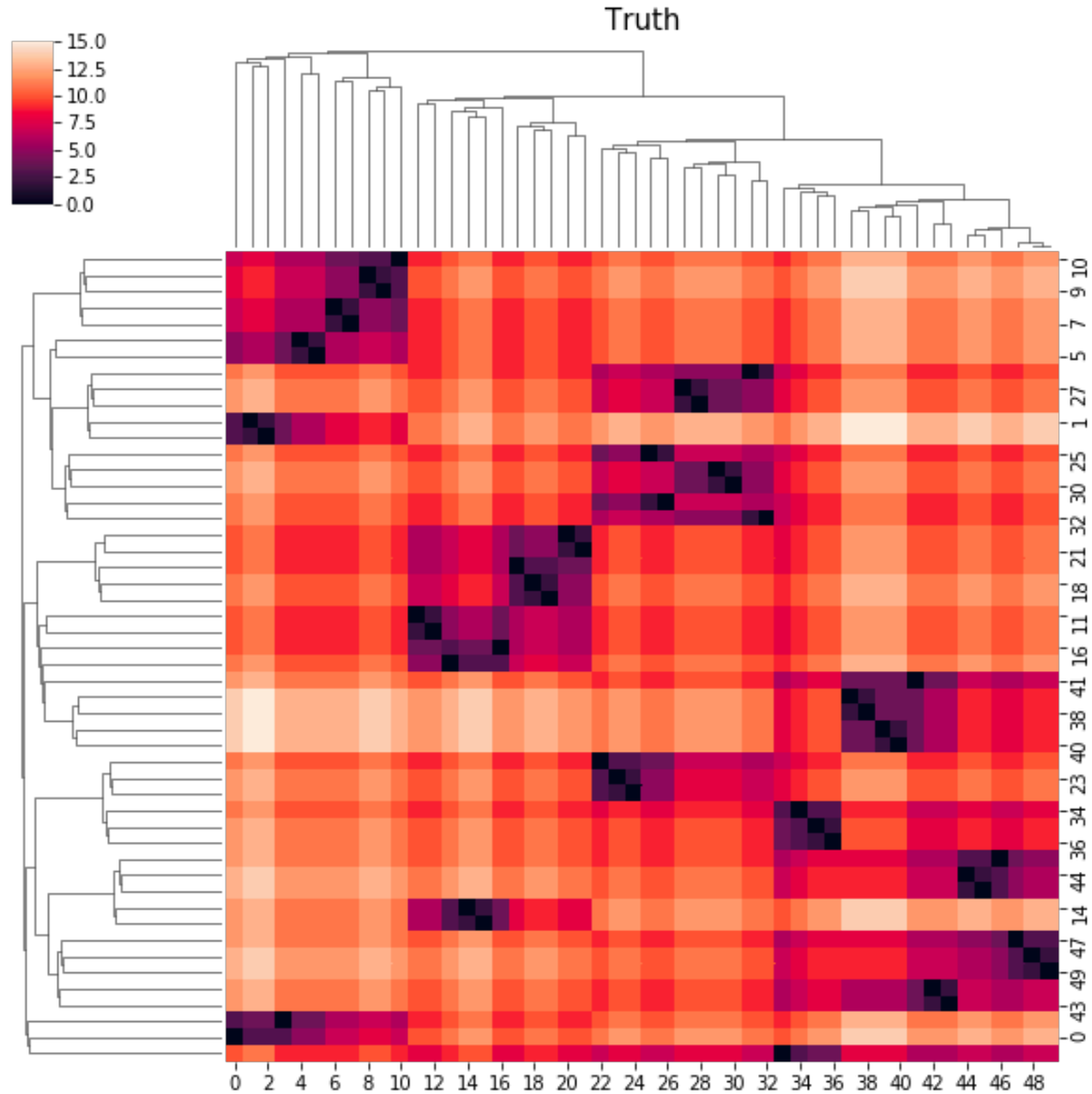


Heat cluster maps visualizations

Kyle Cranmer, SM &
Duccio Pappadopulo

github.com/SebastianMacaluso/VisualizeBinaryTrees





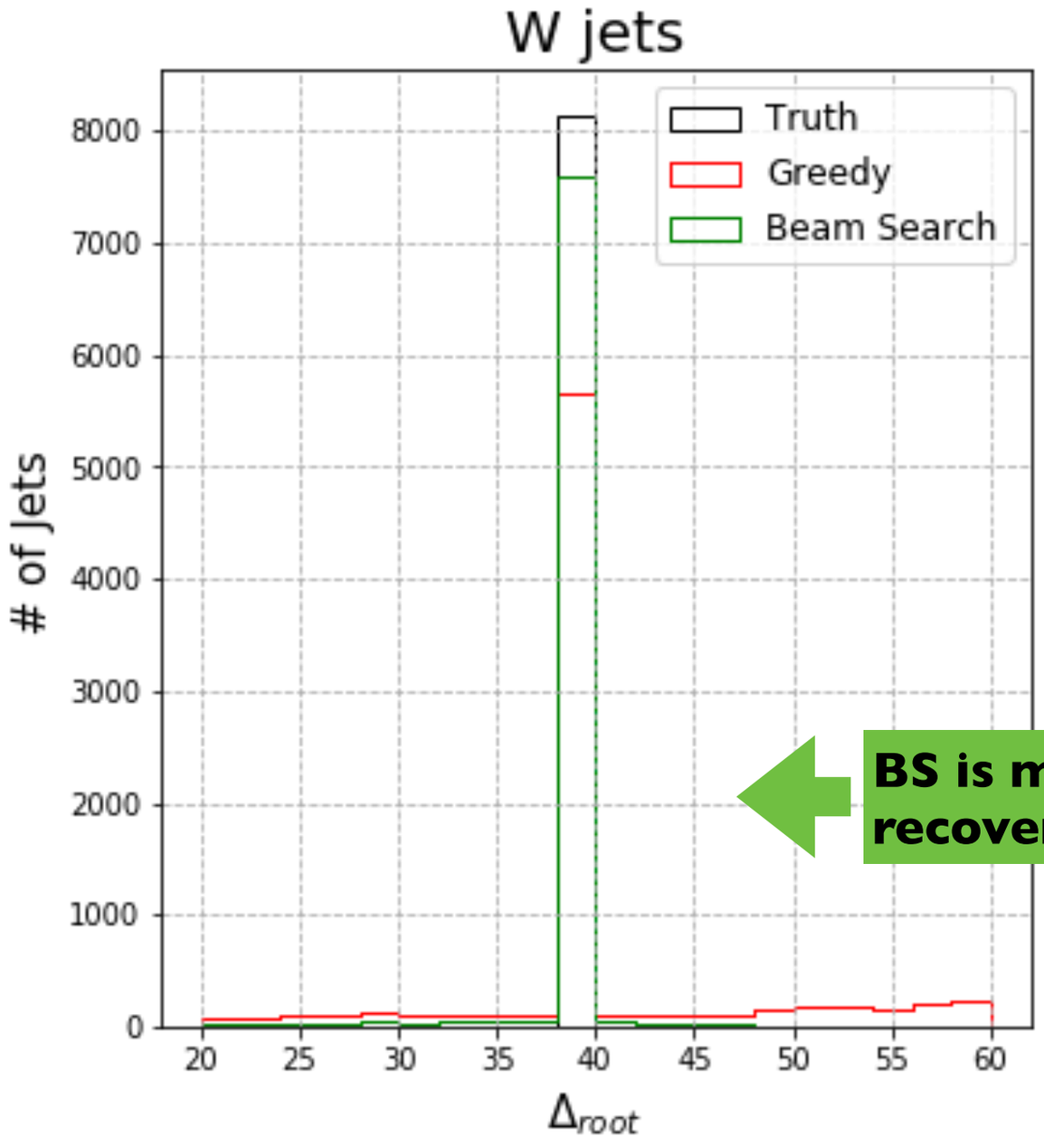
Reconstruction of Δ_{root}

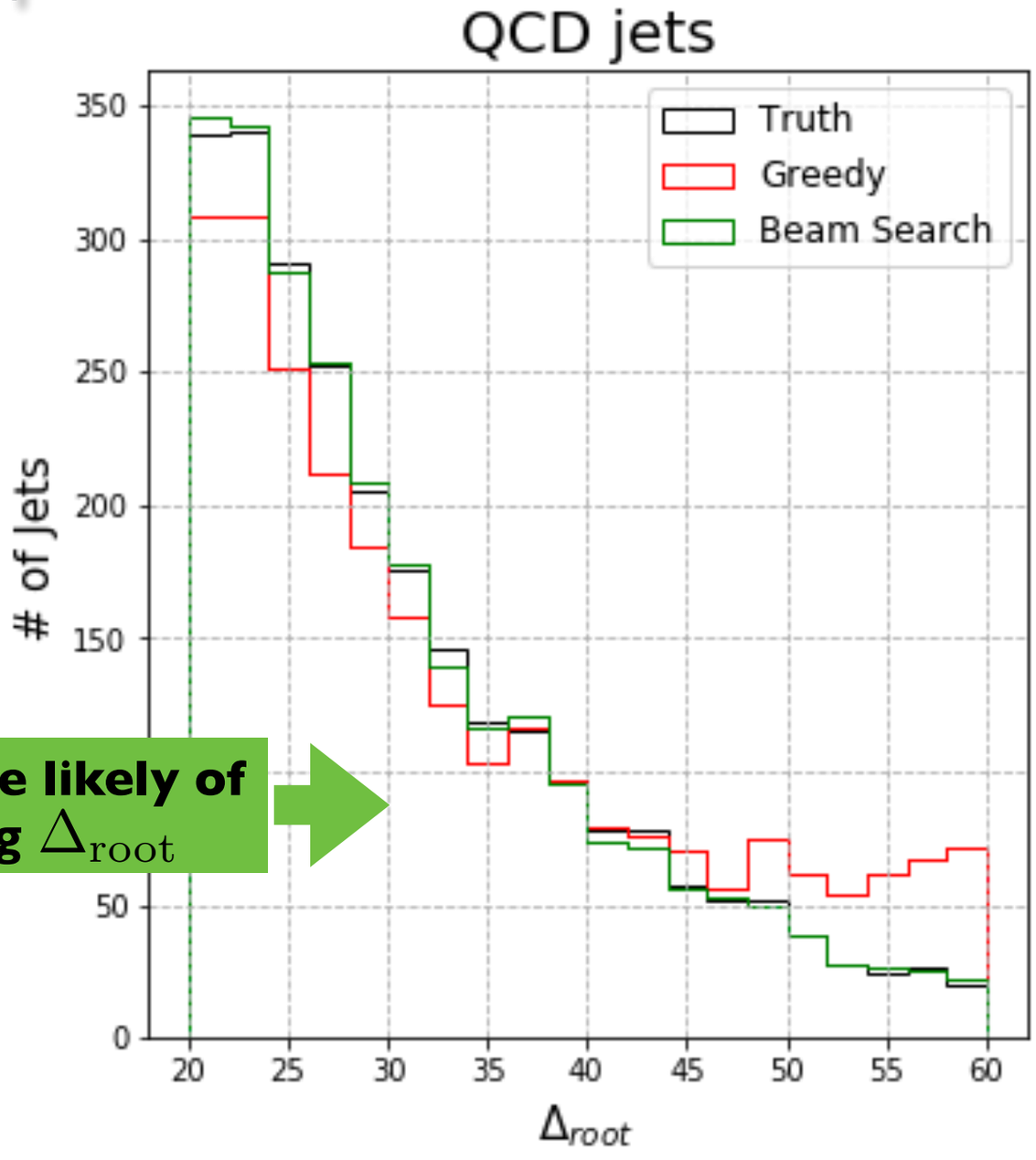
Ginkgo W jets

$$\Delta_{\text{root}} = \frac{m_W}{2}$$

Ginkgo QCD jets

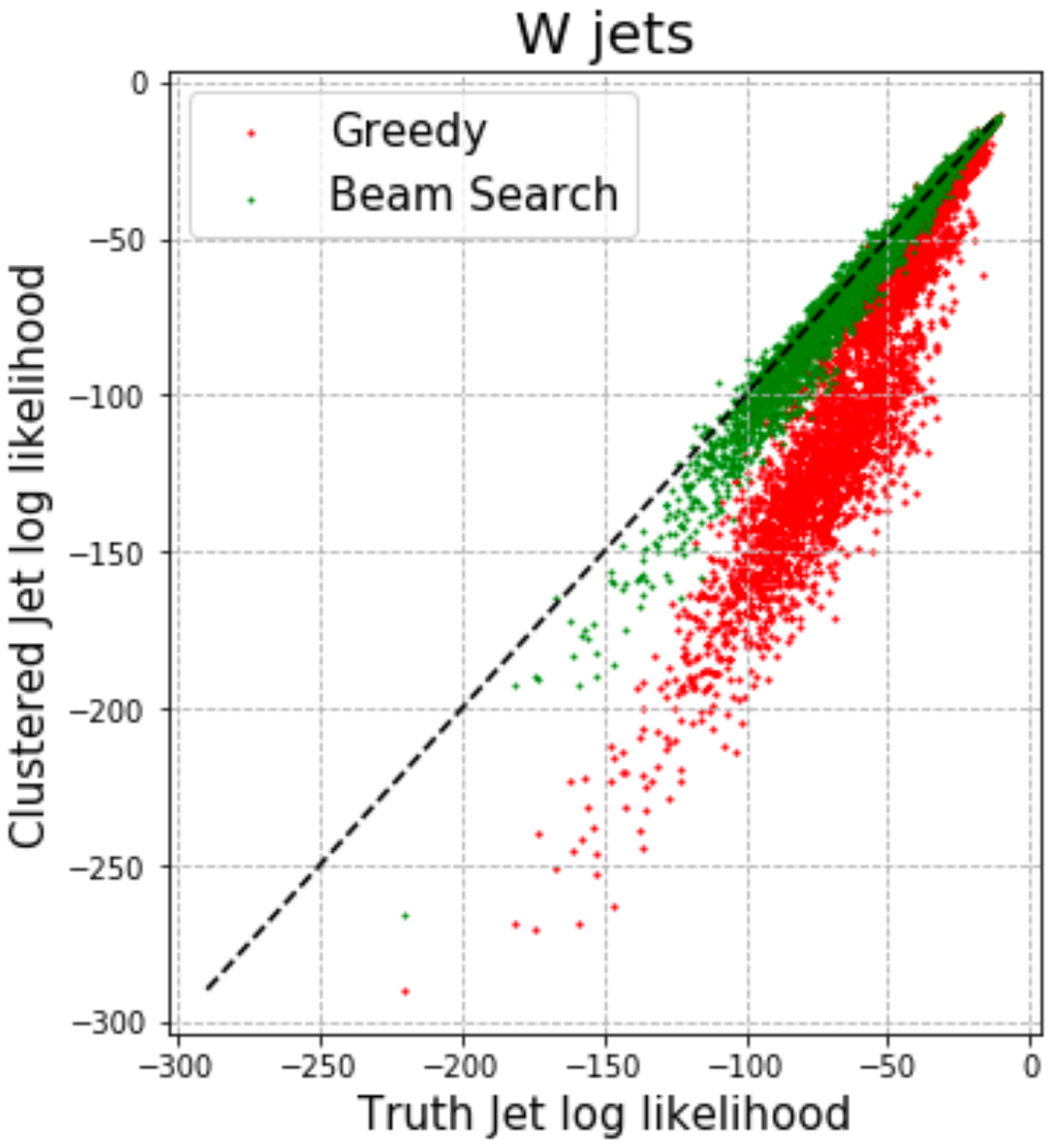
$$\Delta_{\text{root}} \sim f(\Delta|\lambda, \Delta_0) = \frac{\lambda}{\Delta_0} e^{-\frac{\lambda}{\Delta_0}\Delta}$$

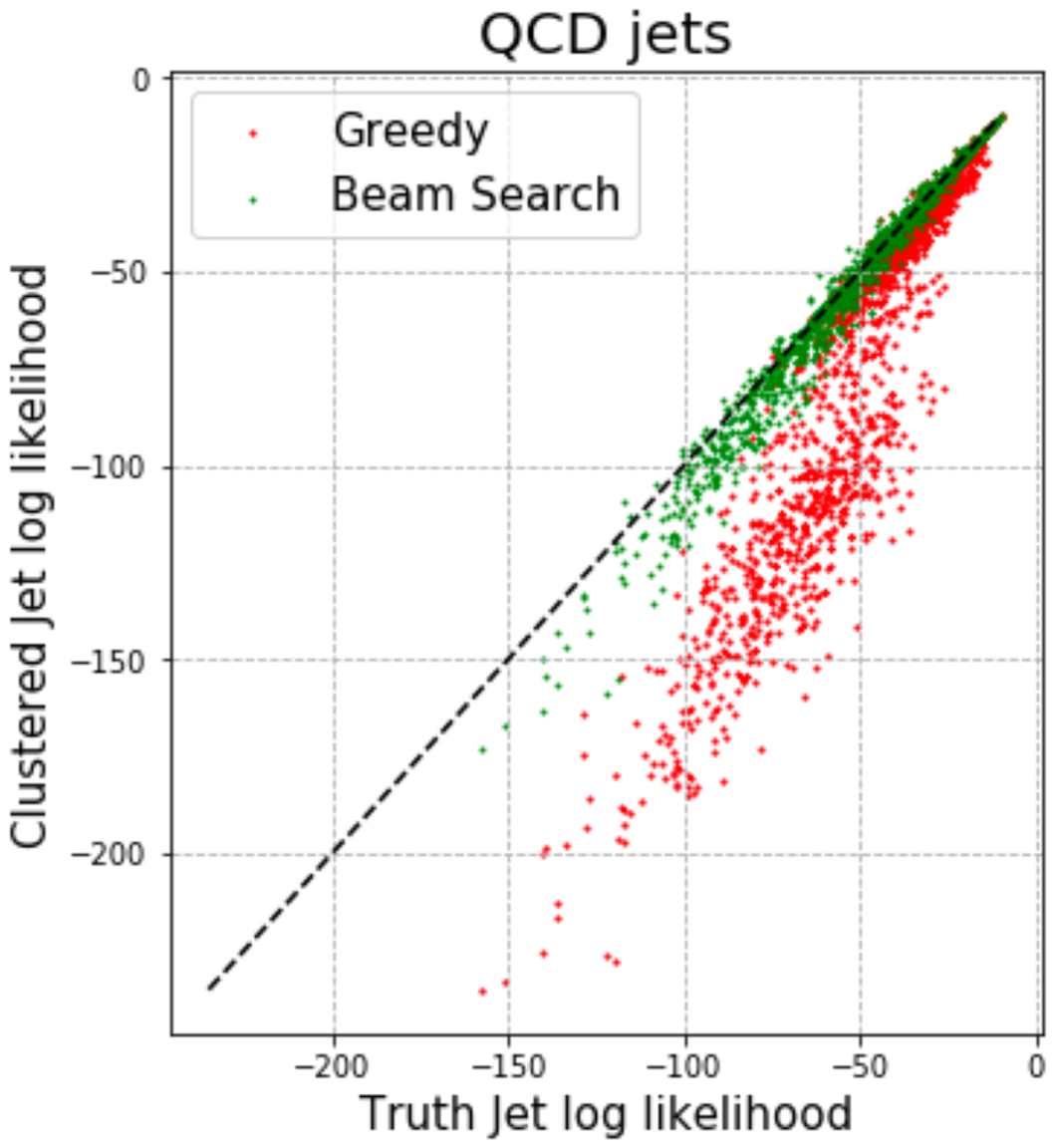




BS is more likely of recovering Δ_{root}

Jets log likelihood



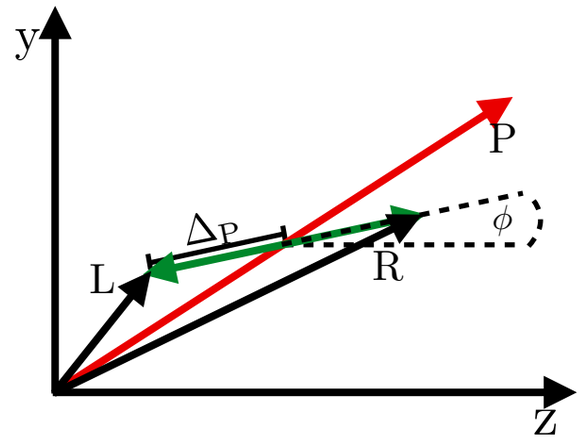


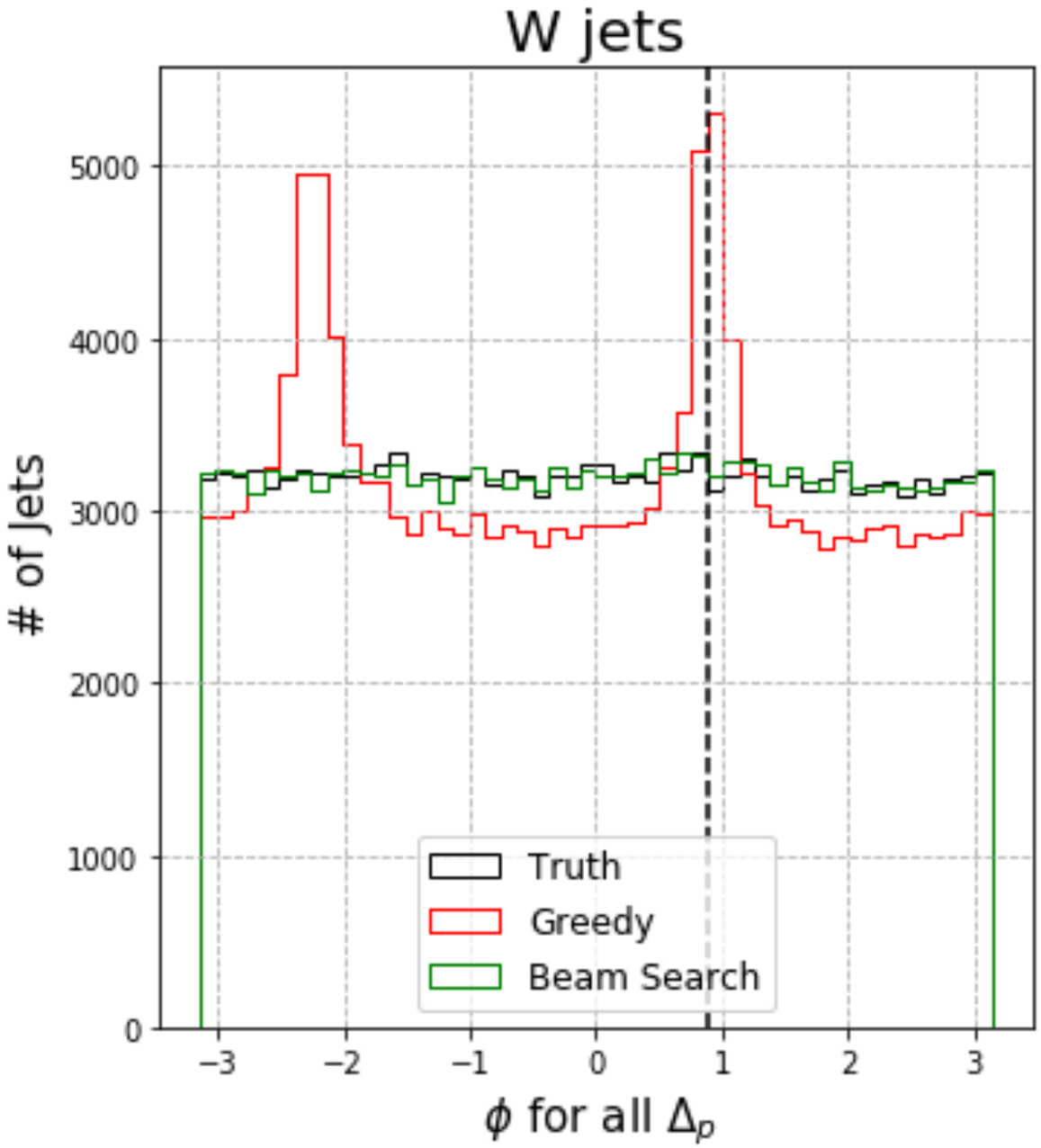
Mean values:

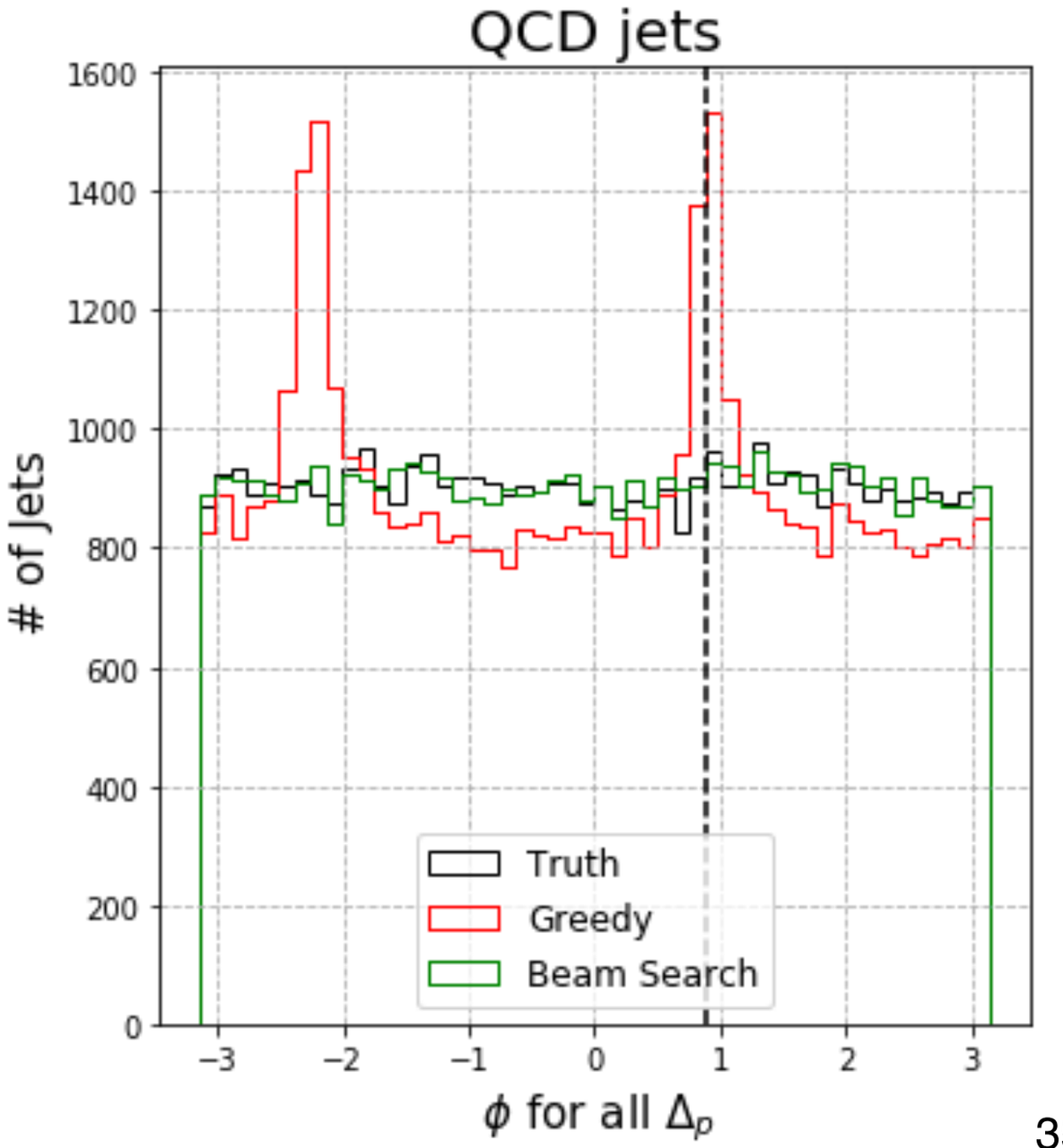
$$\log p_{\text{truth}} = -45.5 \pm 3.1$$
$$\log p_{\text{BS}} = -46.7 \pm 3.2$$
$$\log p_{\text{Greedy}} = -60.9 \pm 5.2$$

$$\log p_{\text{truth}} = -40.8 \pm 2.9$$
$$\log p_{\text{BS}} = -41.4 \pm 3.3$$
$$\log p_{\text{Greedy}} = -53.5 \pm 5.6$$

ϕ angle

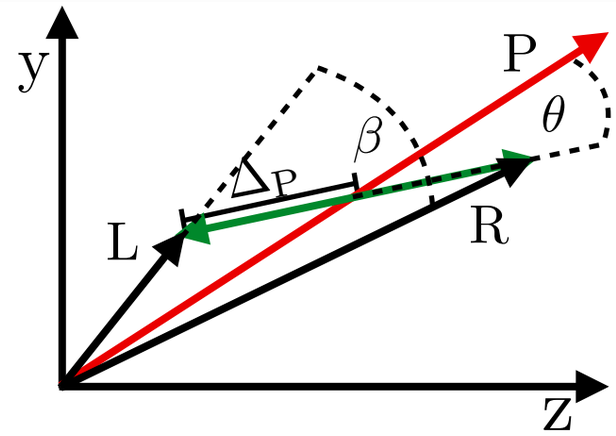

y
P
Δ_P
ϕ
L
R
z

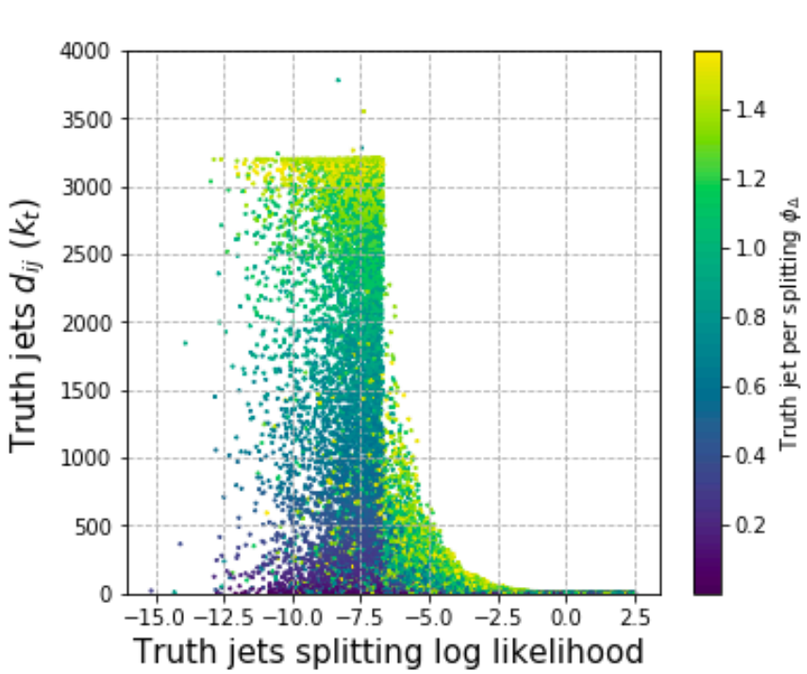

W jets
# of Jets
ϕ for all Δ_p
Truth
Greedy
Beam Search

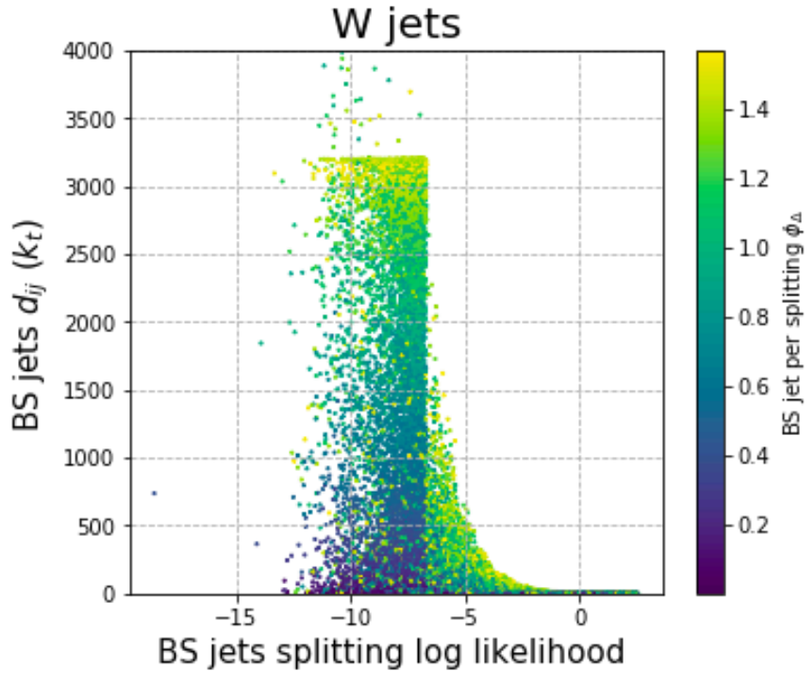

QCD jets
# of Jets
ϕ for all Δ_p
Truth
Greedy
Beam Search

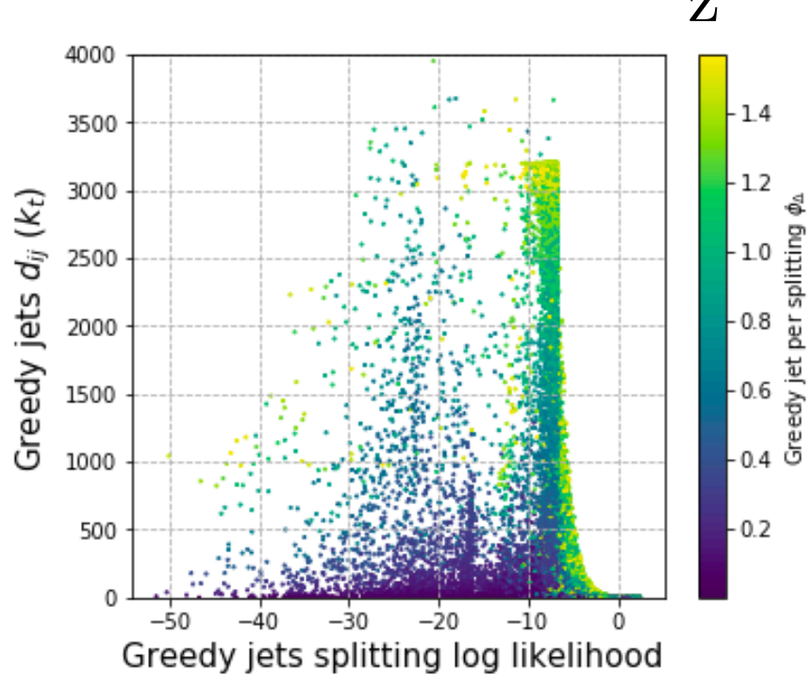
Splitting log LH vs kt distance metric dij

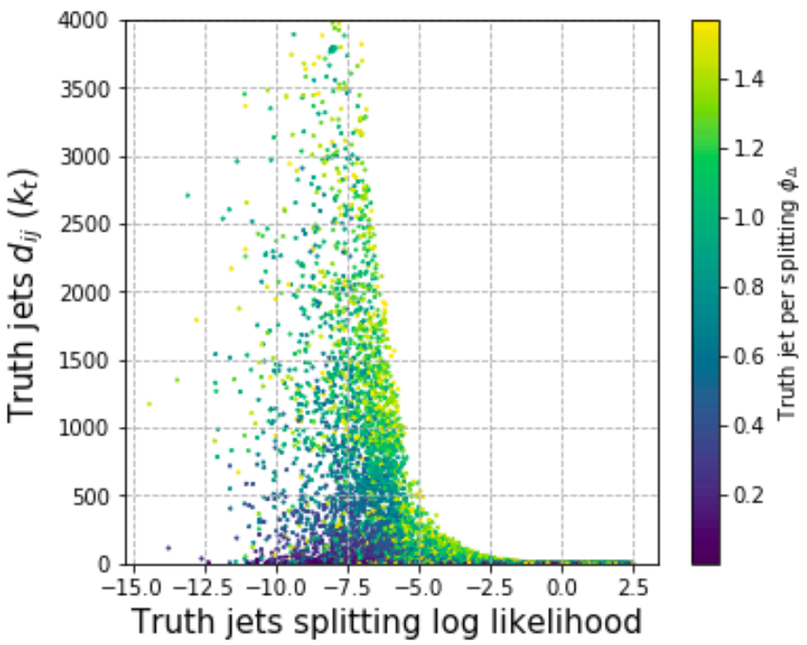
$$d_{ij} \sim \beta^2$$

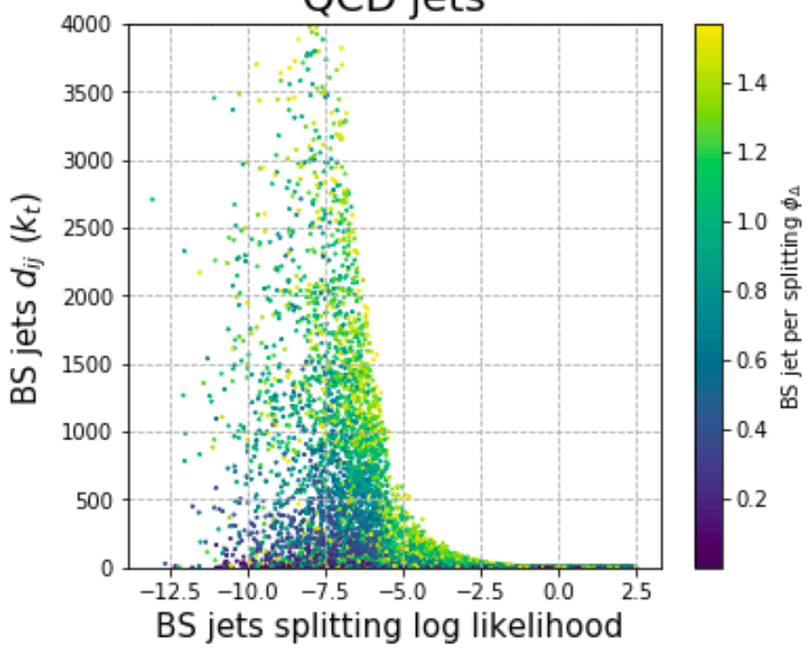


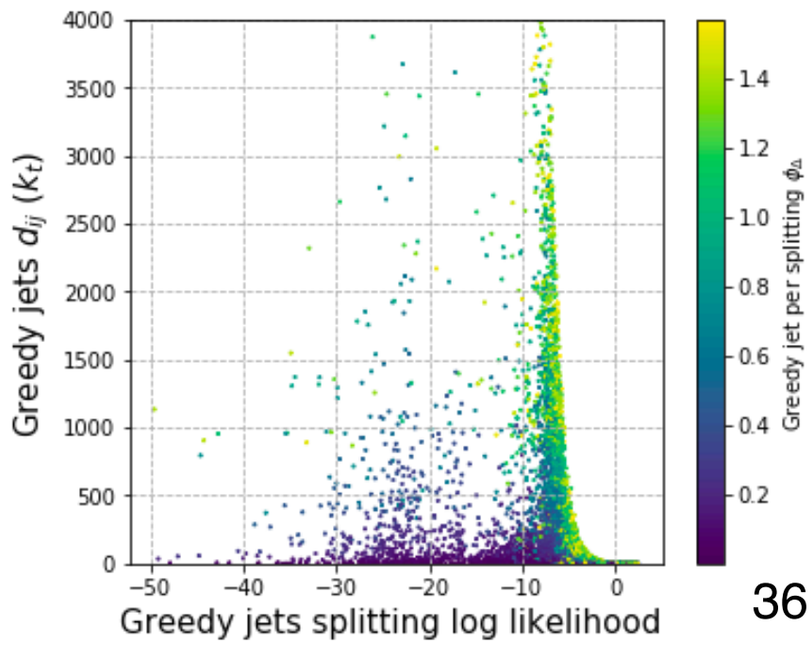






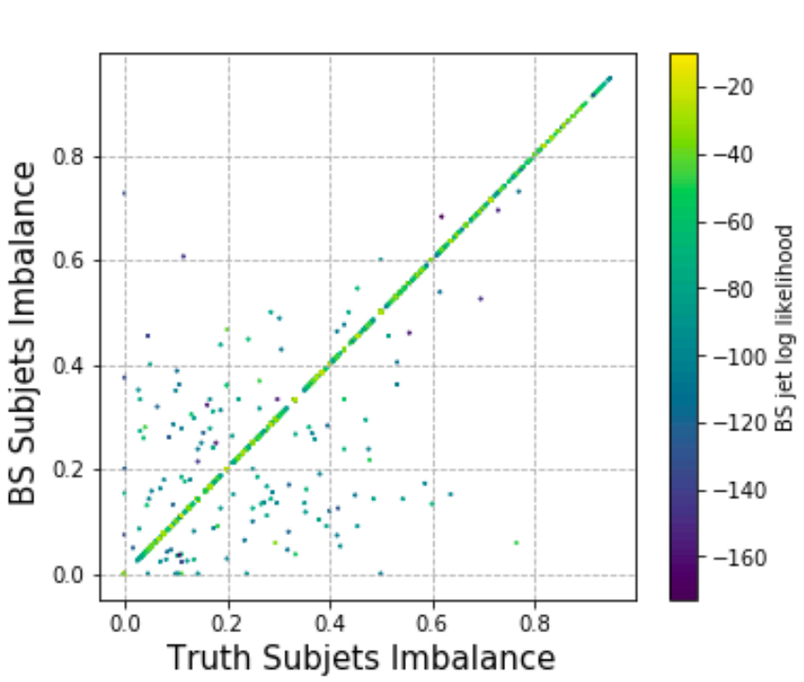


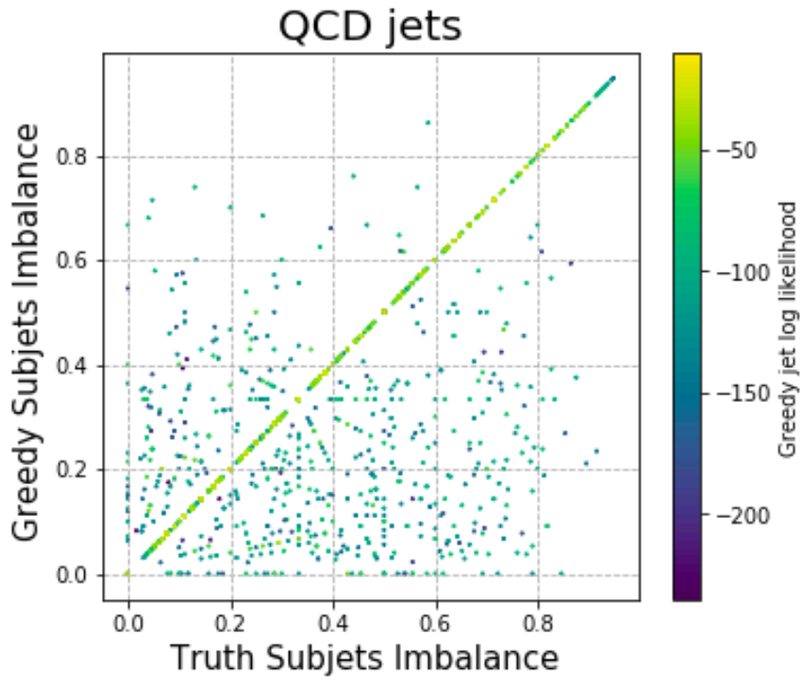


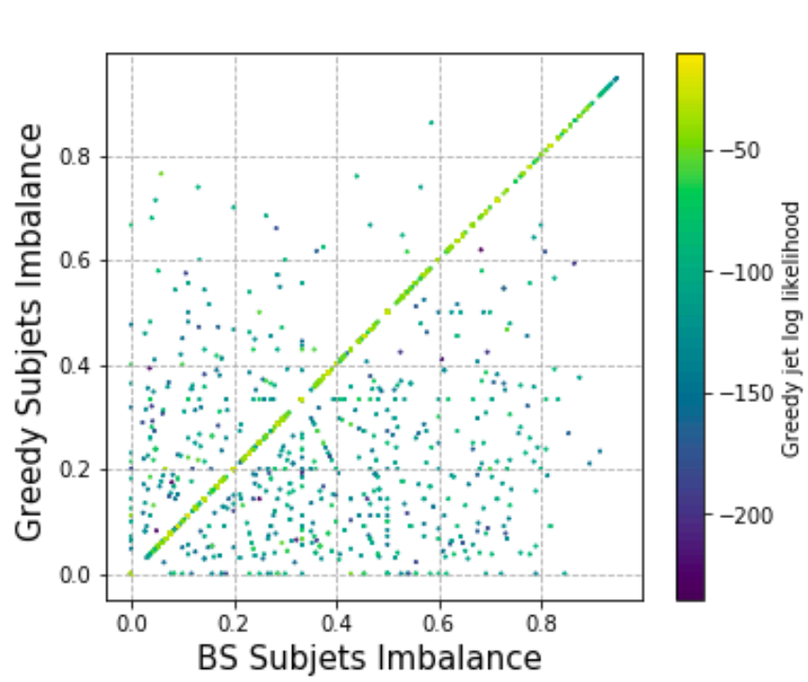


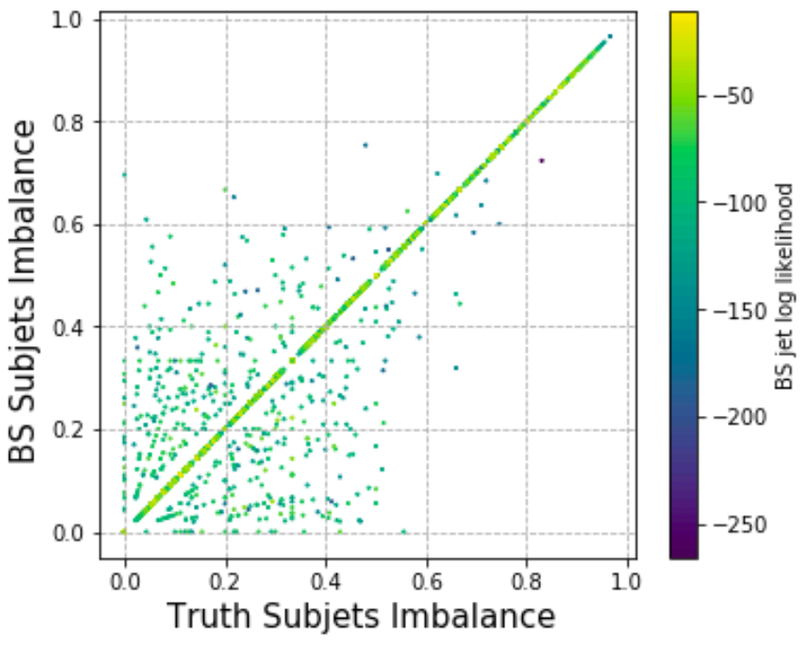
Subjets imbalance

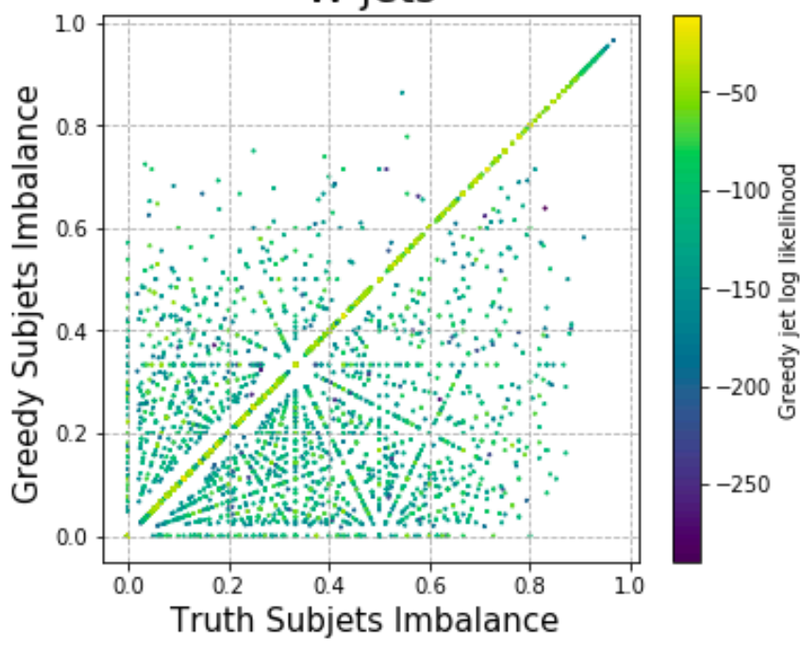
$$\mathcal{I} = \mathrm{abs}\left(\frac{N_L - N_R}{N_L + N_R}\right)$$

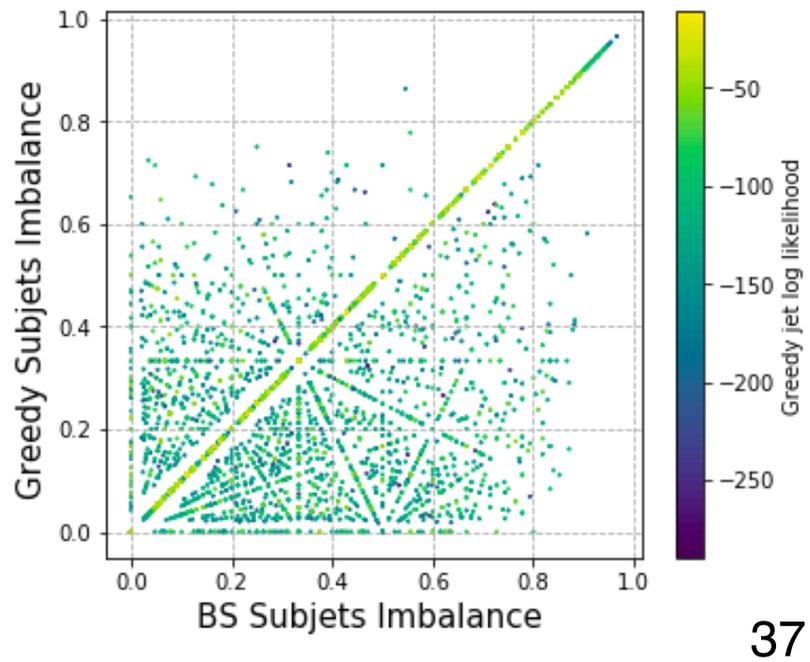






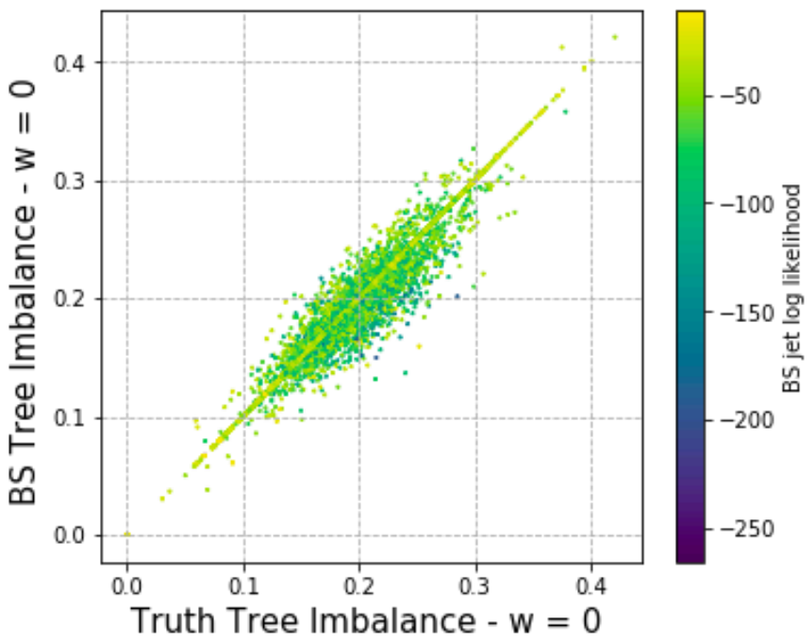


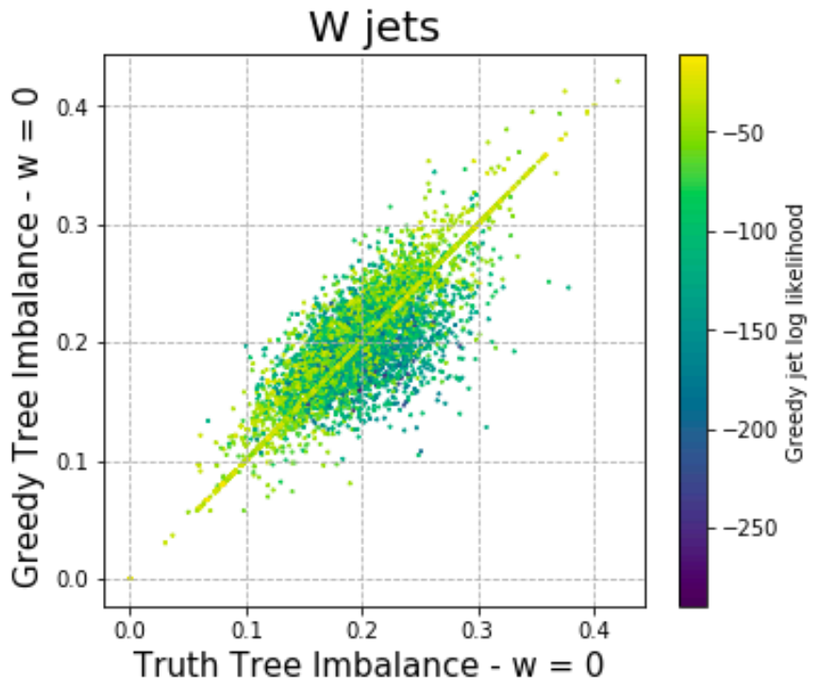


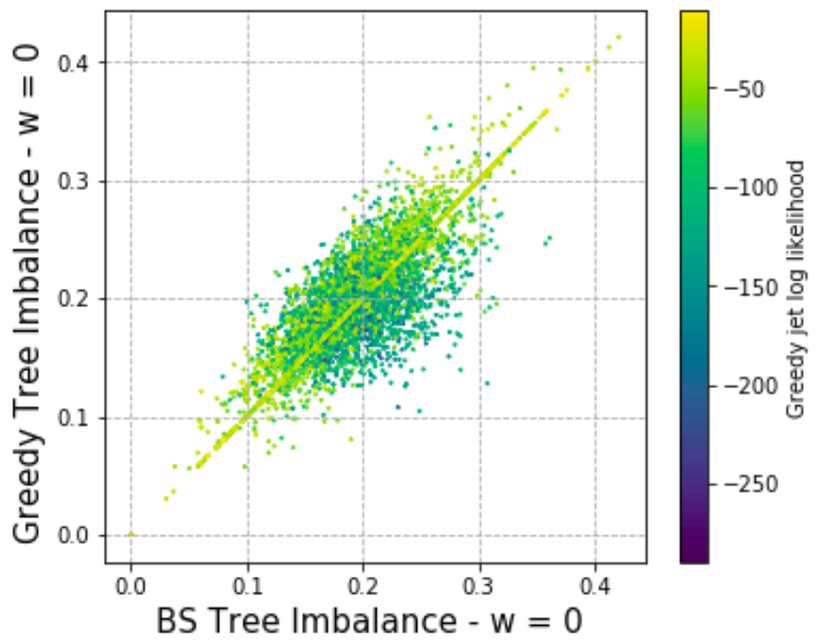


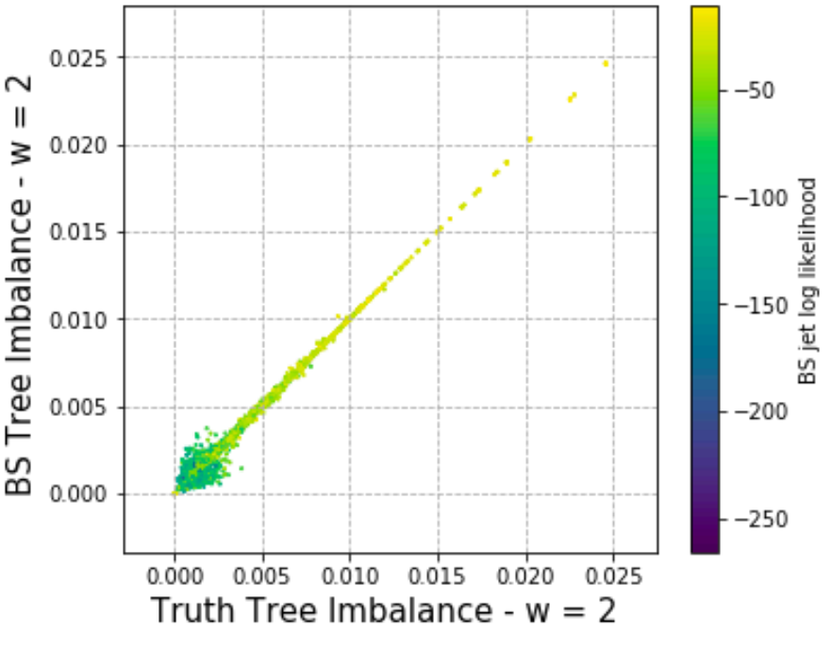
Tree imbalance

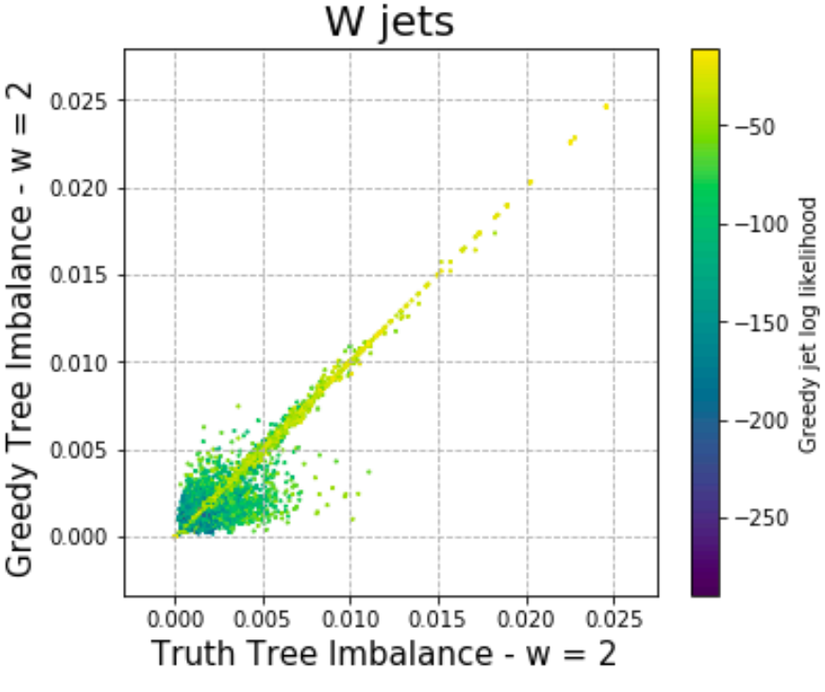
$$\mathcal{T} = \frac{1}{N_{\text{inner}}} \sum_i \exp(-wN_\ell)\, \mathcal{I}_i$$

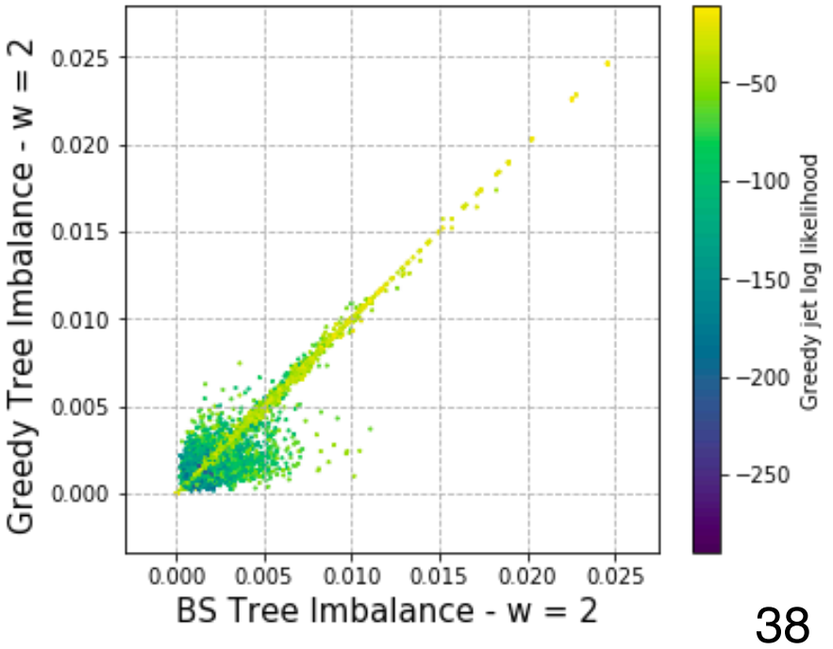












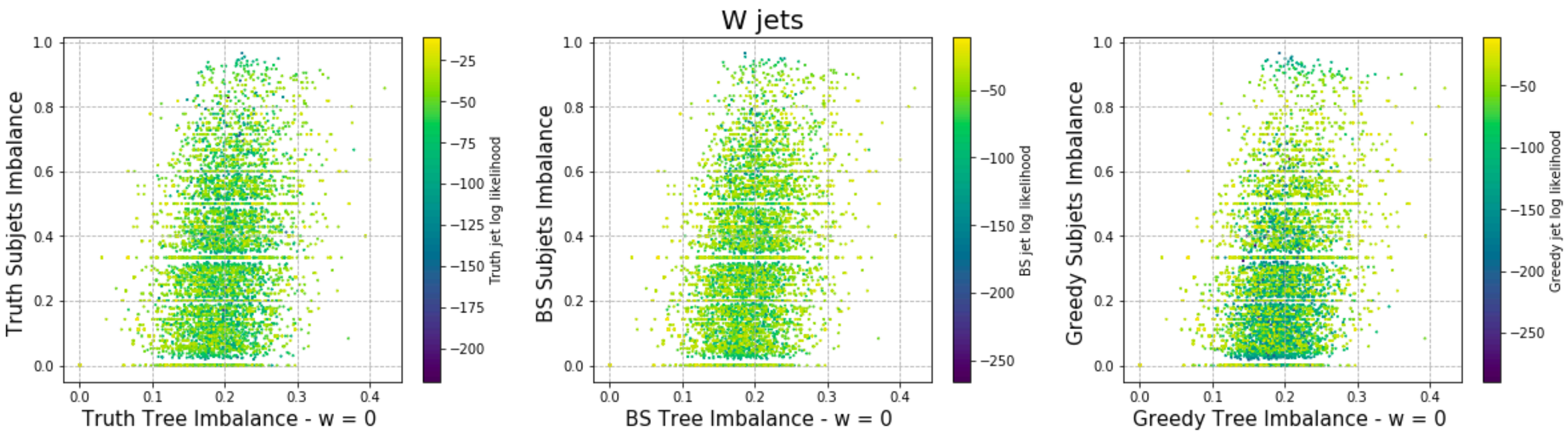
Final remarks and future directions

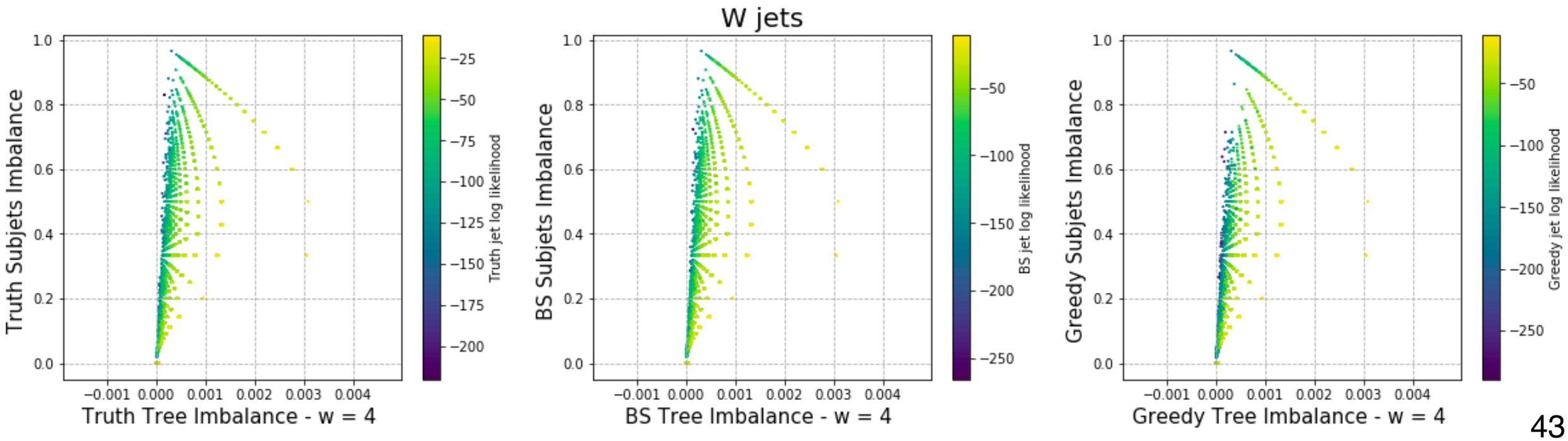
- Goal: Unification of generation and inference.
- Beam search is strictly better than the greedy algorithm (e.g. kt) in terms of estimating the most likely tree.
- Greedy and beam search algorithms are just starting points to find the most likely splitting history. New ideas to study this problem?
- Study algorithms for subjects reconstruction and pileup removal, e.g reinforcement learning based algorithms (related work [Carrazza & Dreyer '19]).
- Study the posterior distribution $p(z|x,\theta)$ on clustering histories, conditioned on the observed particles.
  Obtain an estimate from sampling with MCMC or probabilistic programing techniques?
- Could we learn the node splitting from fitting to data and systematically improve the generative model?
- Study metrics to compare binary trees latent structures.

Thanks for your attention!

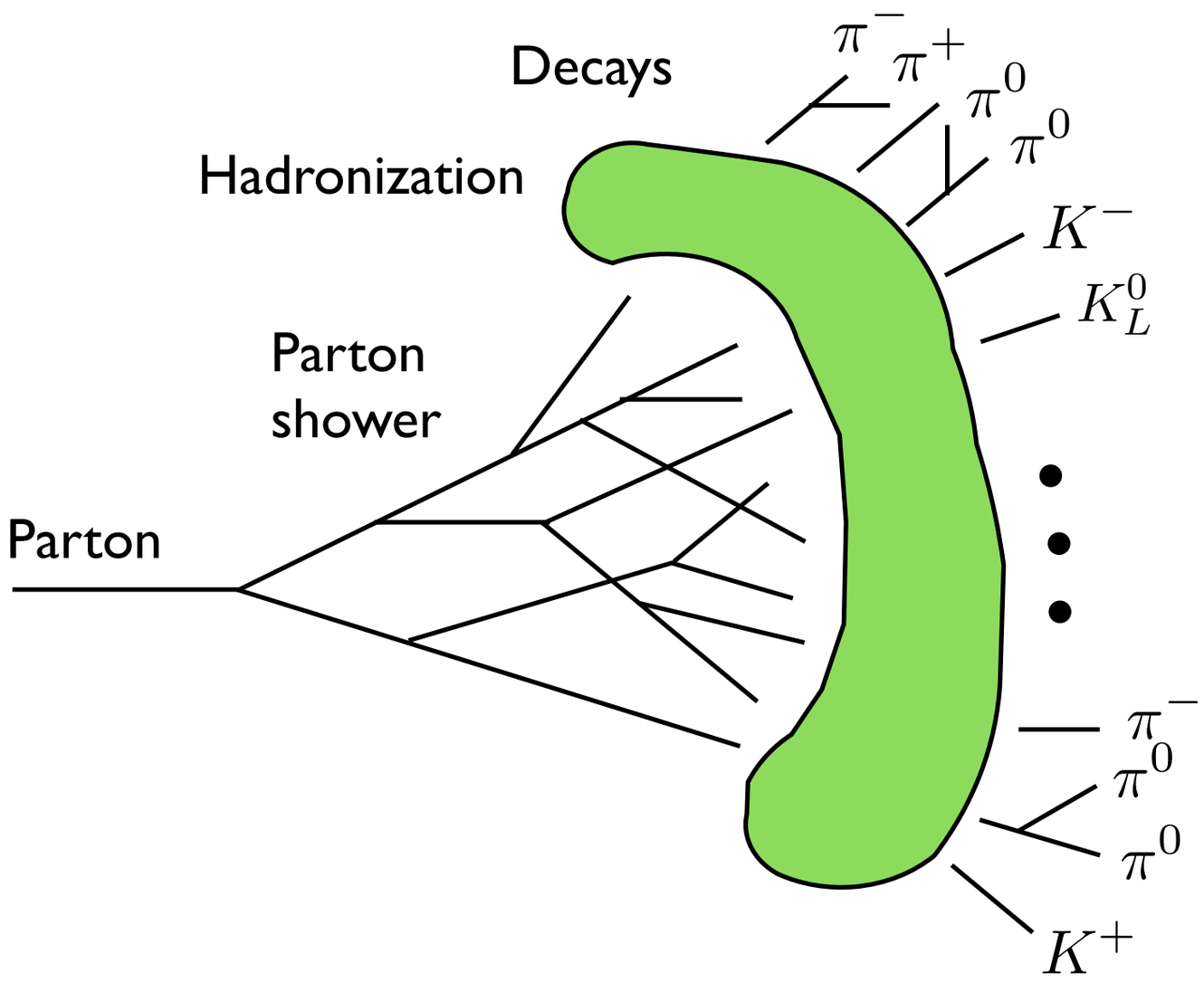
Tree imbalance

$$\mathcal{T} = \frac{1}{N_{\text{inner}}} \sum_i \exp(-wN_\ell)\, \mathcal{I}_i$$





Jets


Decays
Hadronization
Parton shower
Parton
π^-
π^+
π^0
π^0
K^-
K_L^0
π^-
π^0
π^0
K^+

Likelihood-based clustering

Bottom-up approach: nodes pairing likelihood in the Toy Generative model for jets

For our model, given λ and $\Delta_{\min}$, and a pair of nodes to be clustered, we calculate

$$\vec{\Delta}_{\mathrm{p}} = \frac{1}{2}(\vec{p}_{\mathrm{R}} - \vec{p}_{\mathrm{L}}) \qquad \Delta_{\mathrm{p}} = |\vec{\Delta}_{\mathrm{p}}|$$

$$p(\Delta_{\mathrm{L}}, \Delta_{\mathrm{R}}|\Delta_{\mathrm{p}}) = p(\Delta_{\mathrm{L}}|\Delta_{\mathrm{p}})\ p(\Delta_{\mathrm{R}}|\Delta_{\mathrm{p}})\ p(\phi)$$

where

$$p(\phi) = \frac{1}{2\pi}$$

$$p(\Delta|\Delta_{\mathrm{p}}) = \begin{cases} \dfrac{\lambda}{\Delta_{\mathrm{p}}} e^{-\frac{\lambda}{\Delta_{\mathrm{p}}}\Delta} & \text{if } \Delta > \Delta_{\min} \\ 1 - e^{-\frac{\lambda}{\Delta_{\mathrm{p}}}\Delta_{\min}} & \text{if } \Delta \leqslant \Delta_{\min} \end{cases}$$

Tree imbalance

$$\mathcal{T} = \frac{1}{N_{\text{inner}}} \sum_i \exp(-wN_\ell)\, \mathcal{I}_i$$