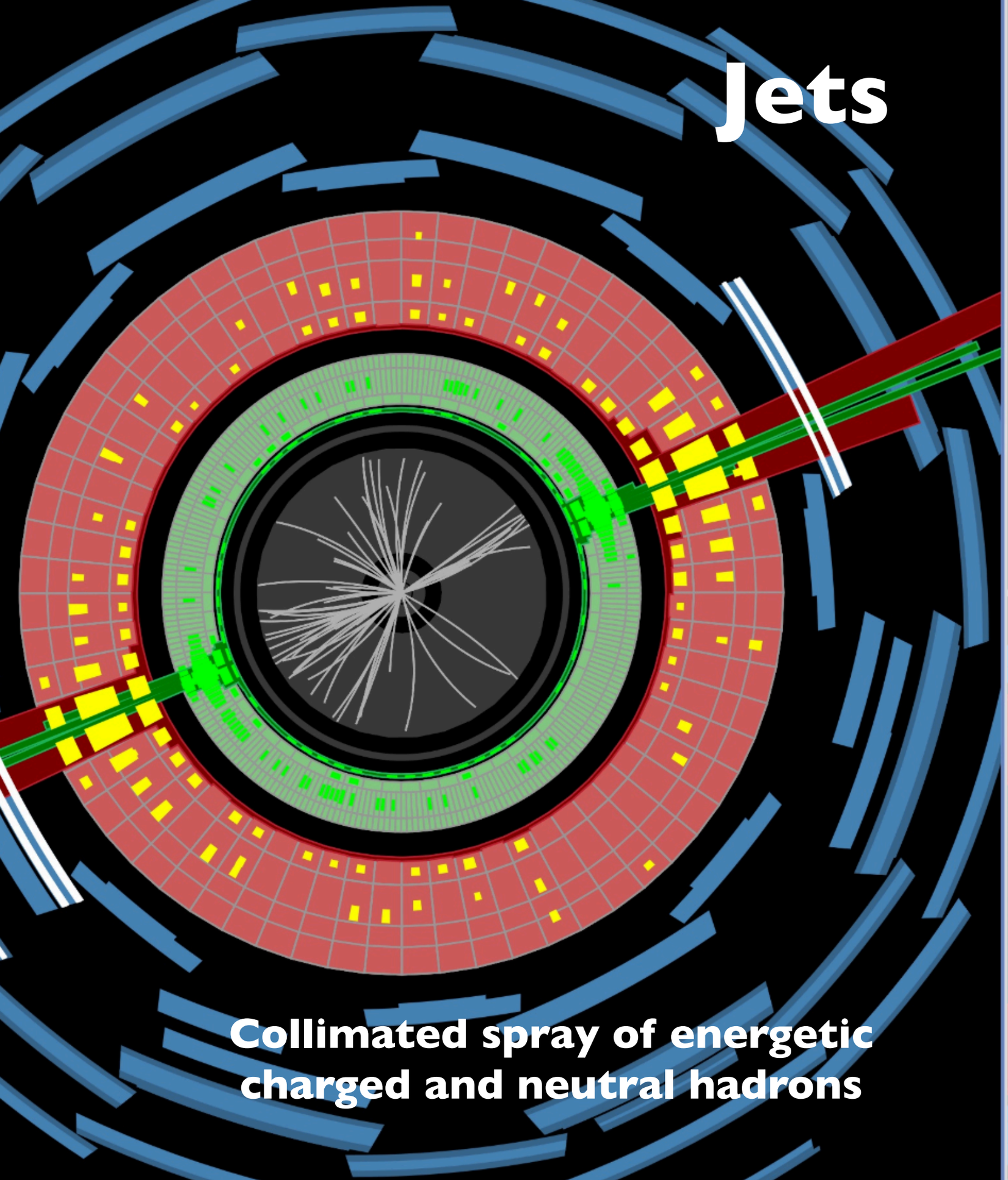
Looking into Jets with Machine Learning

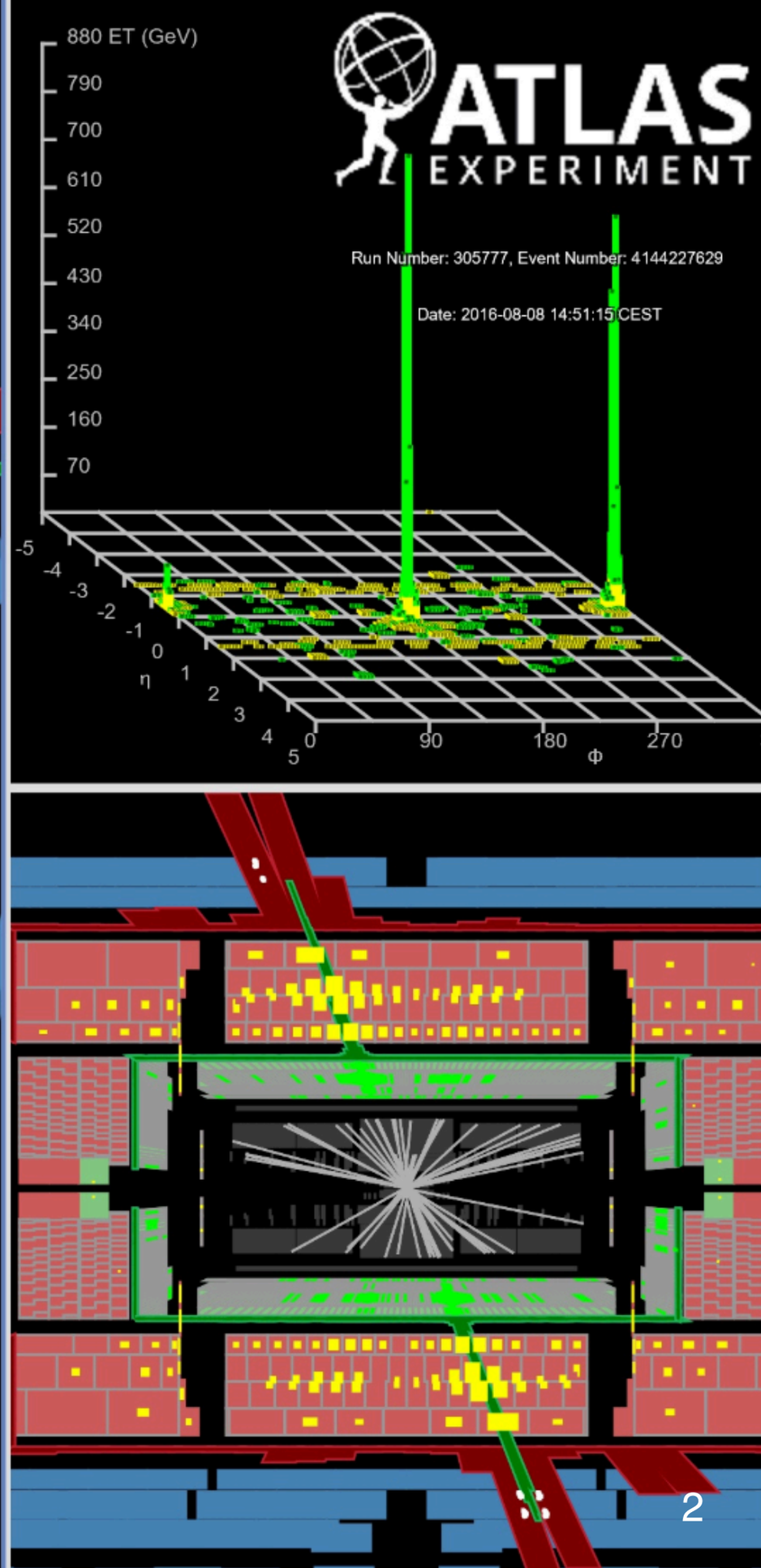
Sebastian Macaluso
New York University

Pheno & Vino
Princeton University
September 24, 2019

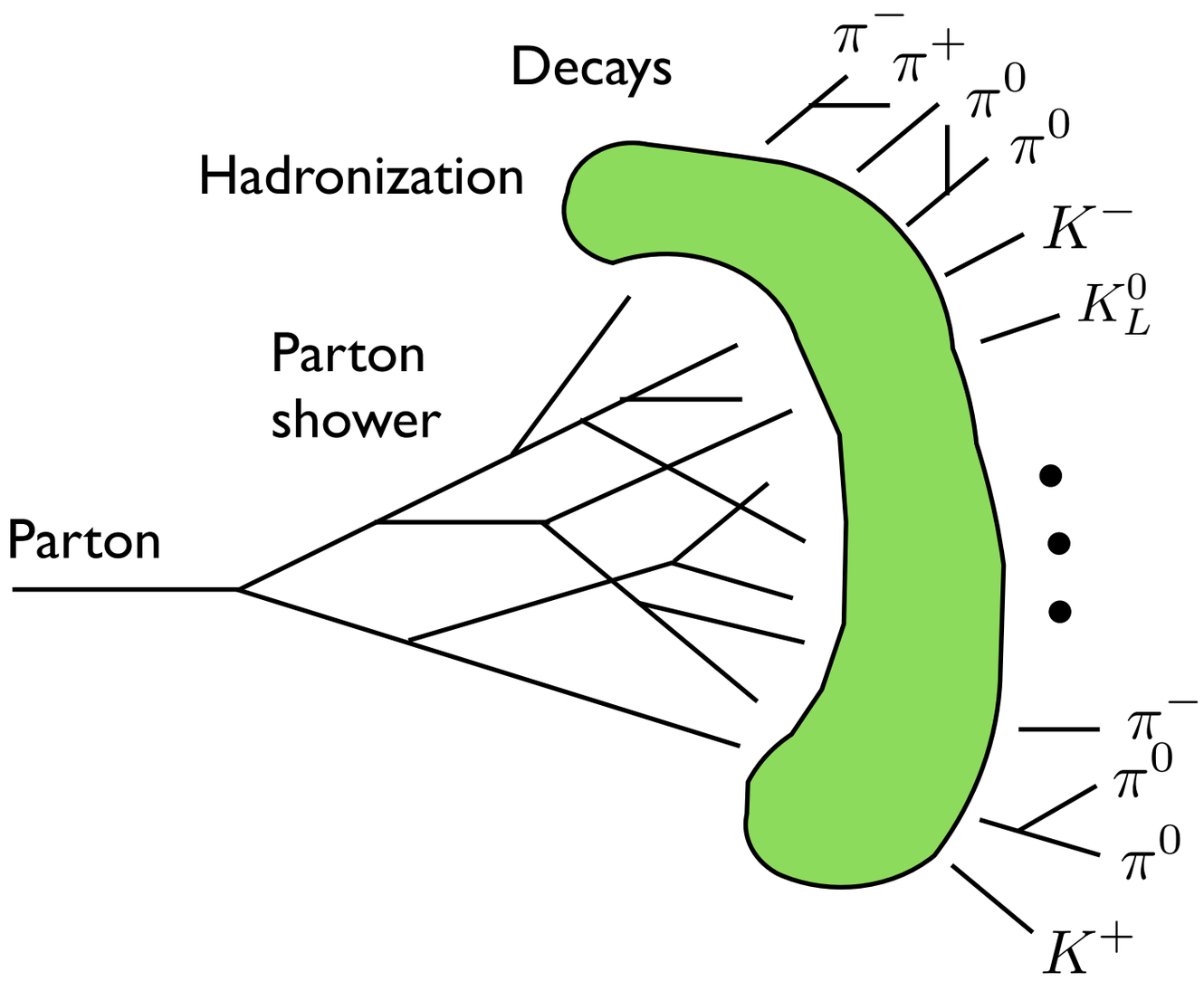
Jets



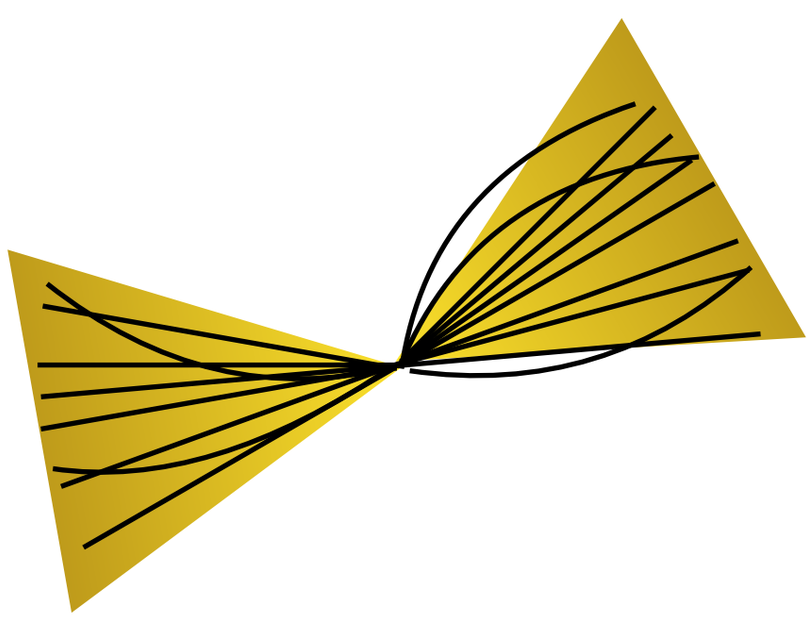
Collimated spray of energetic charged and neutral hadrons



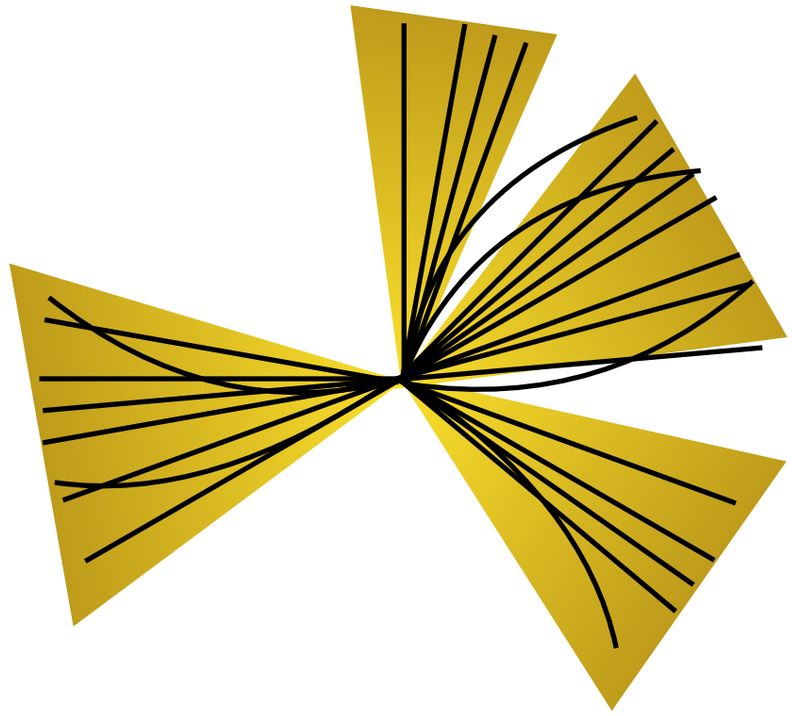
Jets


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

Jet reconstruction

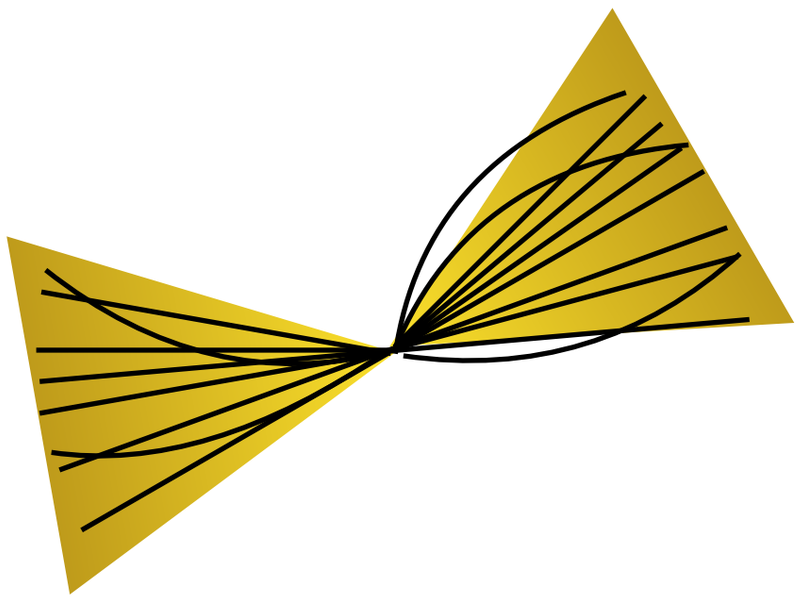


2 jets

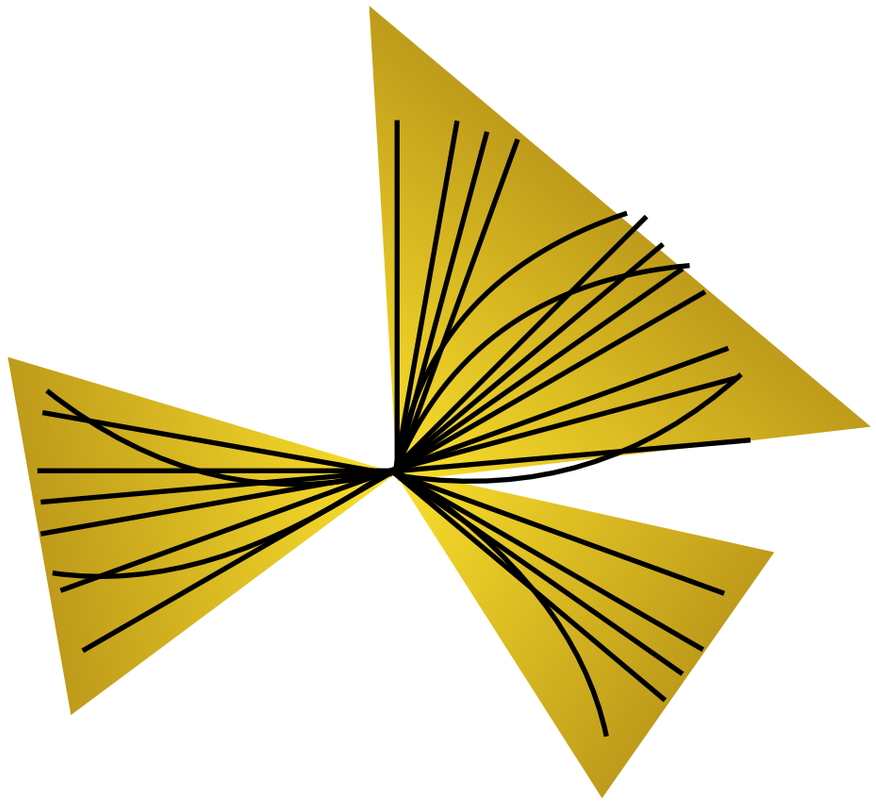


4 jets?

Jet reconstruction

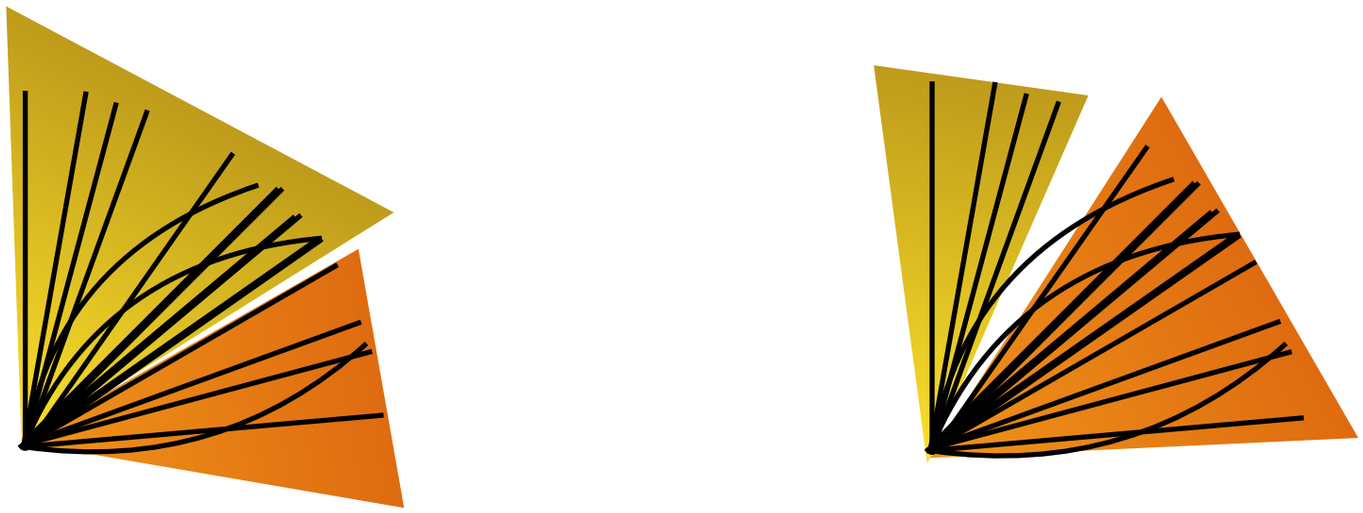


2 jets



3 jets?

Subjets reconstruction



How to group the jet constituents to reconstruct each subject?

Generalized kt clustering algorithms

- 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.

ATLAS and CMS papers since 2016

Welcome to INSPIRE, the High Energy Physics information system. Please dir

:: EXPERIMENTS ::

(collaboration:ATLAS or collaboration:CMS) and (year:2016 or year:2017 or year:2018 or yea

find j "Phys.Rev.Lett.,105*" :: more

Sort by: earliest dat

All papers

Display results: 25 results | single list

HEP **4,251** records found 1 - 25 ▶ ▶▶ jump to record: 1

1. **Search for long-lived particles using delayed photons in pro**
   CMS Collaboration (Albert M Sirunyan (Yerevan Phys. Inst.) *et al.*). Sep 13, 2
   CMS-EXO-19-005, CERN-EP-2019-185
   e-Print: arXiv:1909.06166 [hep-ex] | PDF
   References | BibTeX | LaTeX(US) | LaTeX(EU) | Harvmac | EndNote
   ADS Abstract Service
   Detailed record

HEP :: HEPNAMES :: INSTITUTI

title:jet and (collaboration:ATLAS or collaboration:CMS) and (year:2016 or year:2017 or ye

find j "Phys.Rev.Lett.,105*" :: more

Sort by: earliest date | desc. | -

Papers with "jet" as part of the title

HEP **588** records found 1 - 25 ▶ ▶▶ jump to record: 1

1. **Measurement of the $t\bar{t}b\bar{b}$ production cross section in the all-jet final state in pp collisions at $\sqrt{s} = 13$ TeV**
   CMS Collaboration (Albert M Sirunyan (Yerevan Phys. Inst.) *et al.*). Sep 11, 2019.
   CMS-TOP-18-011, CERN-EP-2019-183
   e-Print: arXiv:1909.05306 [hep-ex] | PDF

anti-kt algorithm

Welcome to IN

find eprint 0802.1189

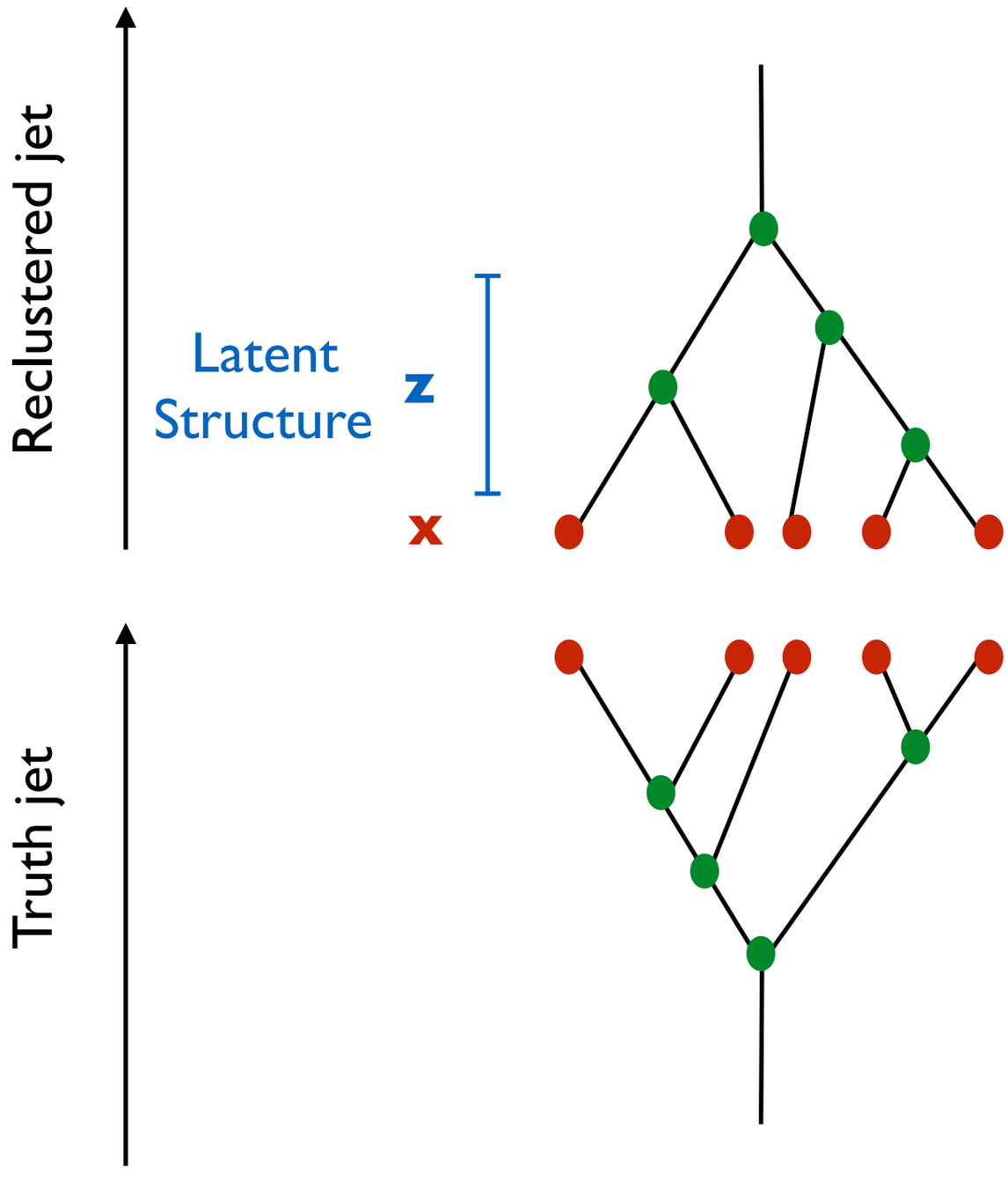
find j "Phys.Rev.Lett.,105*" :: more

Sort by: latest first | desc. | - or rank by - | Display results: 25 results | single list

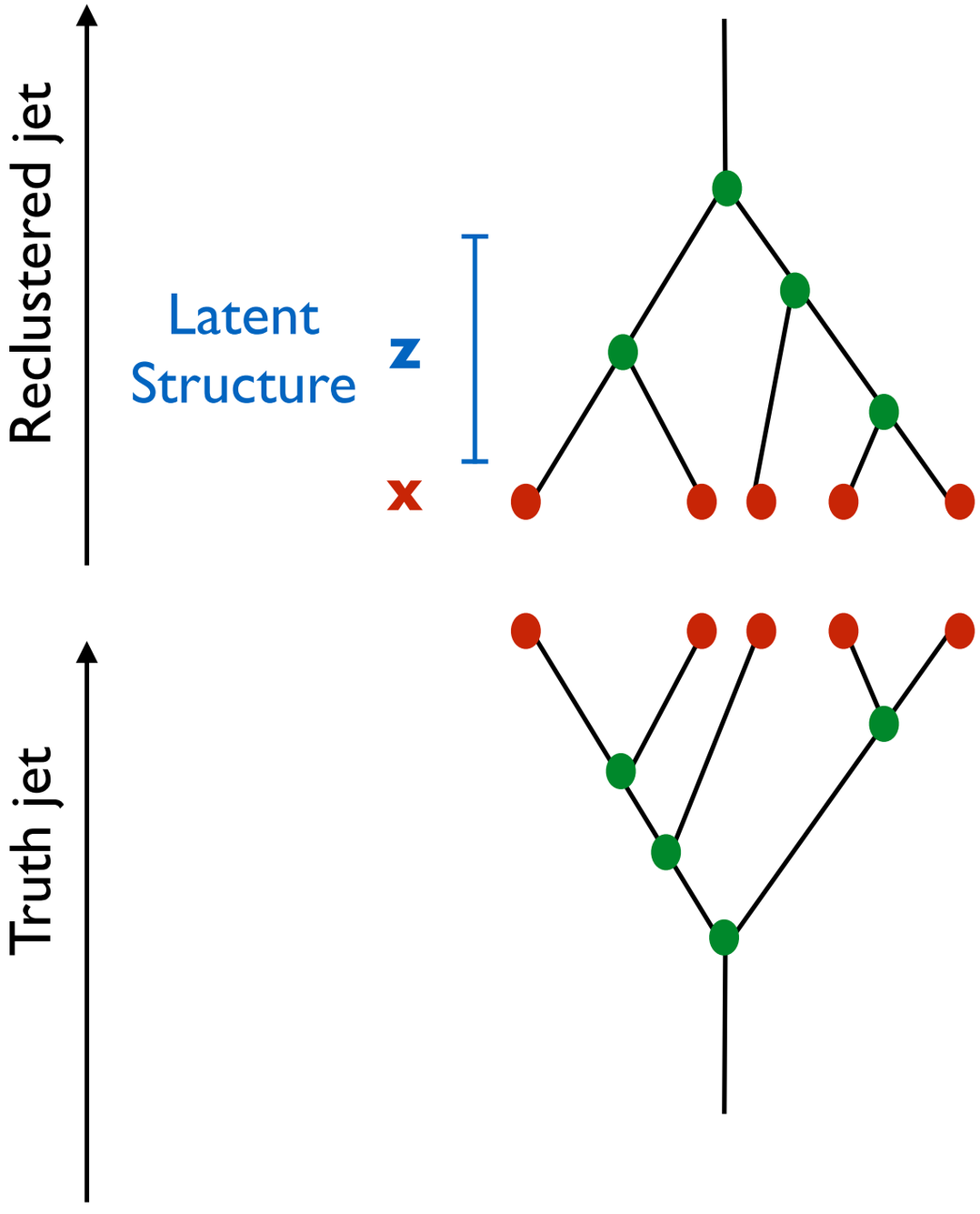
HEP **1** records found

1. **The anti-k_t jet clustering algorithm**
   Matteo Cacciari, Gavin P. Salam (Paris, LPTHE), Gregory Soyez (Brookhaven
   Published in **JHEP 0804 (2008) 063**
   LPTHE-07-03
   DOI: 10.1088/1126-6708/2008/04/063
   e-Print: arXiv:0802.1189 [hep-ph] | PDF
   References | BibTeX | LaTeX(US) | LaTeX(EU) | Harvmac | EndNote
   ADS Abstract Service
   Detailed record - Cited by 6197 records 1000+

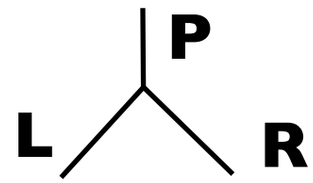
Jet clustering



Jet clustering



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



Joint likelihood:

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

Splitting likelihood:

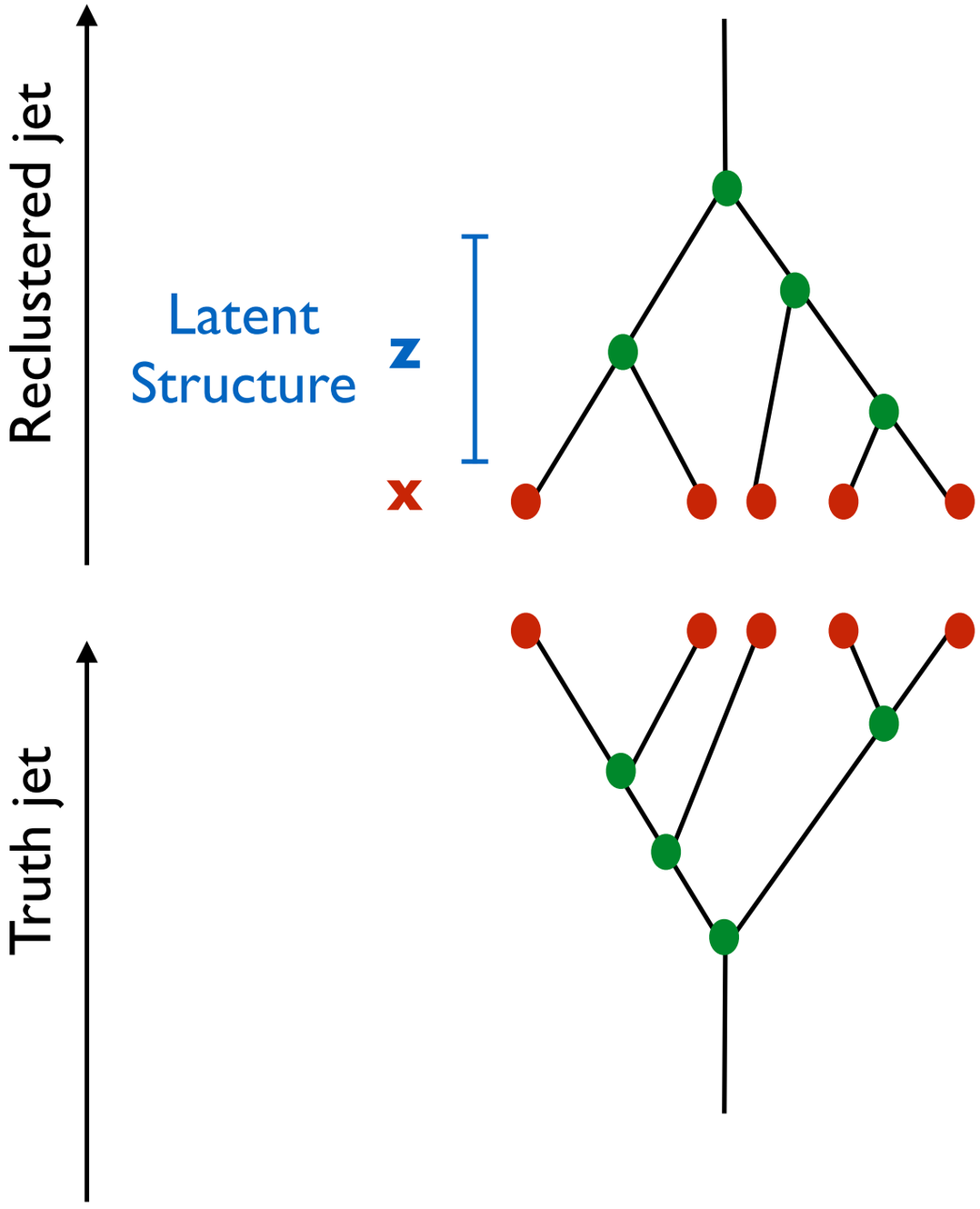
$$p(\mathrm{L}_i, \mathrm{R}_i|\mathrm{P}_i) \equiv p(\vec{p}_\mathrm{L}, \vec{p}_\mathrm{R}|\vec{p}_\mathrm{P})$$

---

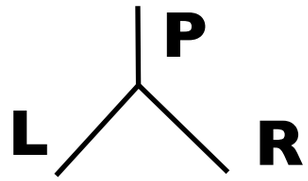
k_t distance measure

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

Jet clustering



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



Joint likelihood:

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

- Hard to do with full physics simulators, e.g. PYTHIA
- To prototype we built our own toy model!

Likelihood-based jet clustering

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

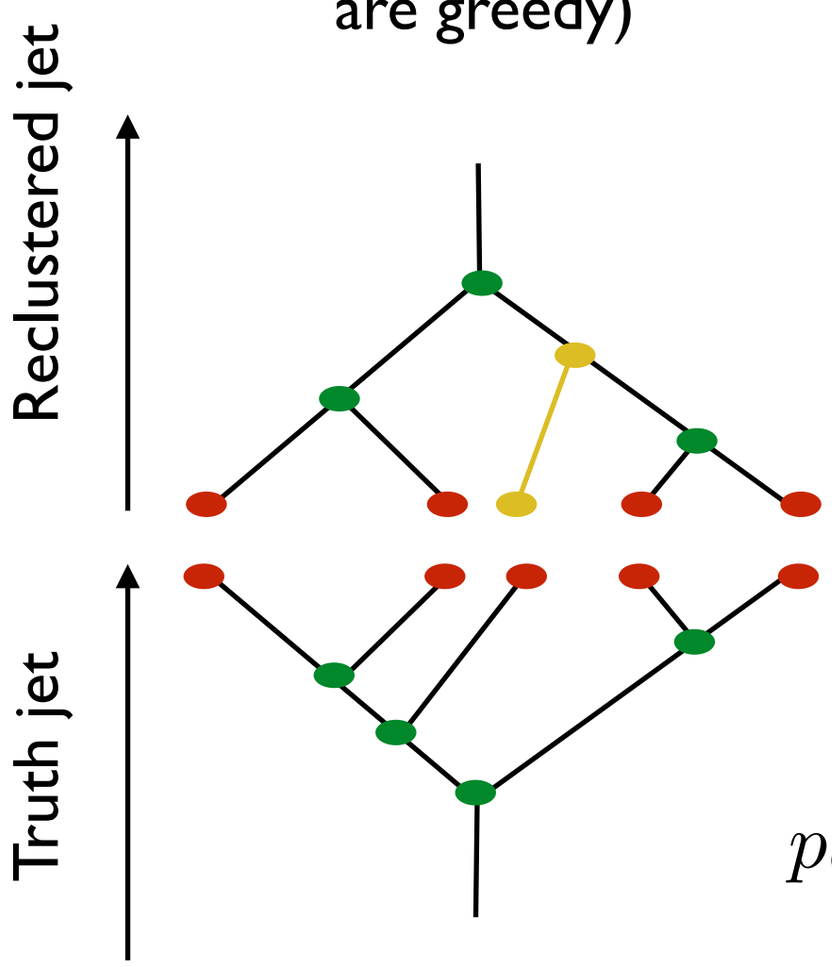
Greedy

Joint likelihood from locally maximizing the likelihood at each step.
(Generalized kt algorithms are greedy)

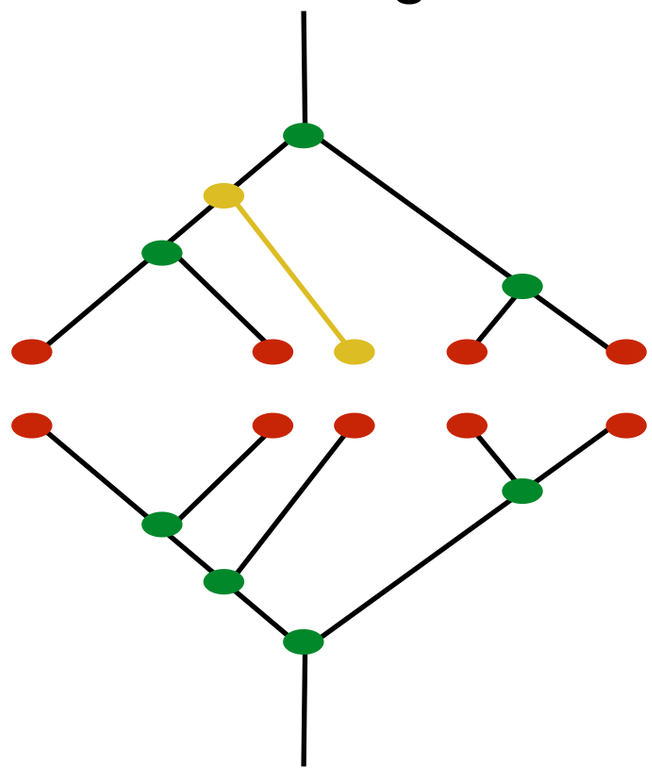
Beam Search

Idea: maximize the likelihood of multiple steps before choosing the latent path.
Maybe we could fix mistakes?

Beneficial for grooming techniques, classification/regression



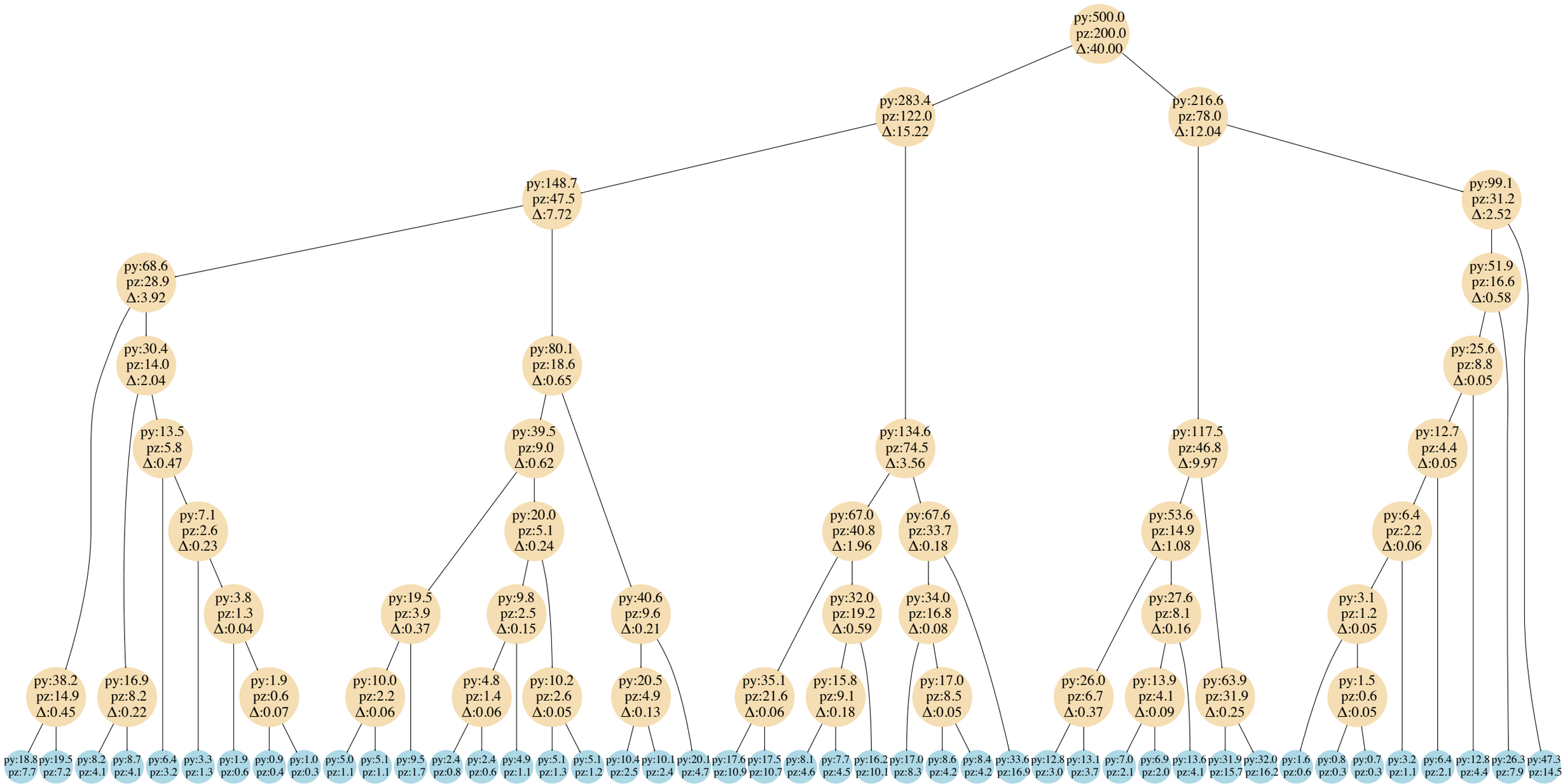
$p_{\mathrm{Greedy}} \lesssim p_{\mathrm{BS}}$



Toy Generative Model for Jets

Kyle Cranmer, SM & Duccio Pappadopulo

https://github.com/SebastianMacaluso/ToyJetsShower

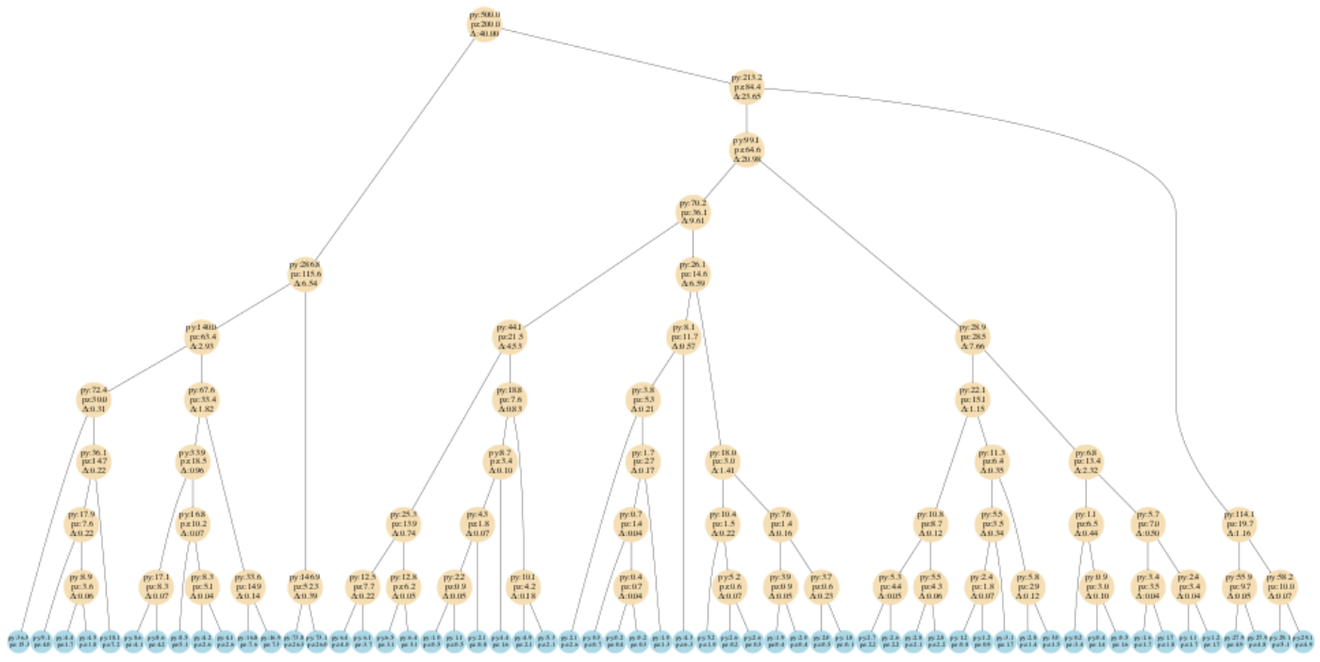


Binary Tree Visualizations

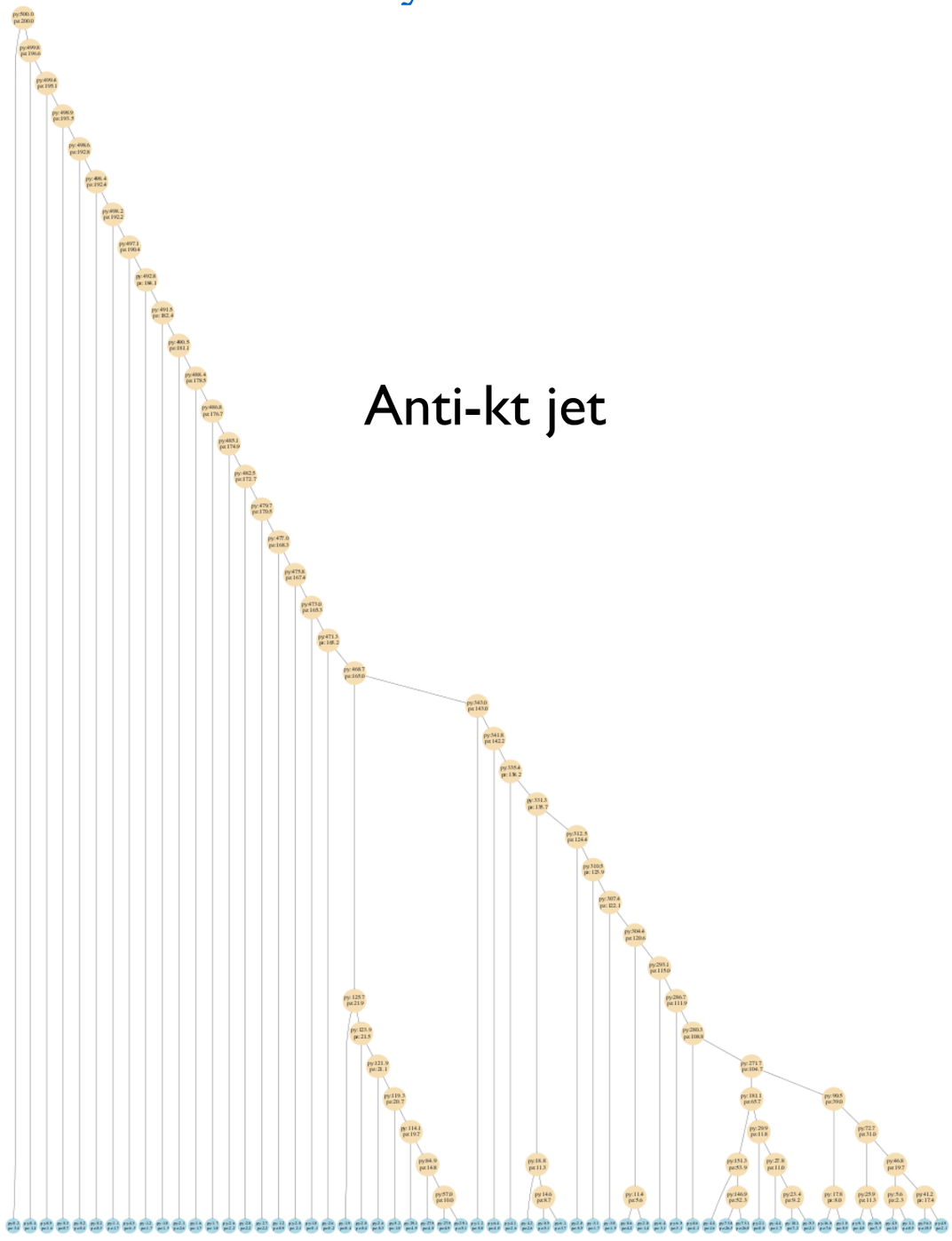
Kyle Cranmer, SM & Duccio Pappadopulo

https://github.com/SebastianMacaluso/VisualizeBinaryTrees

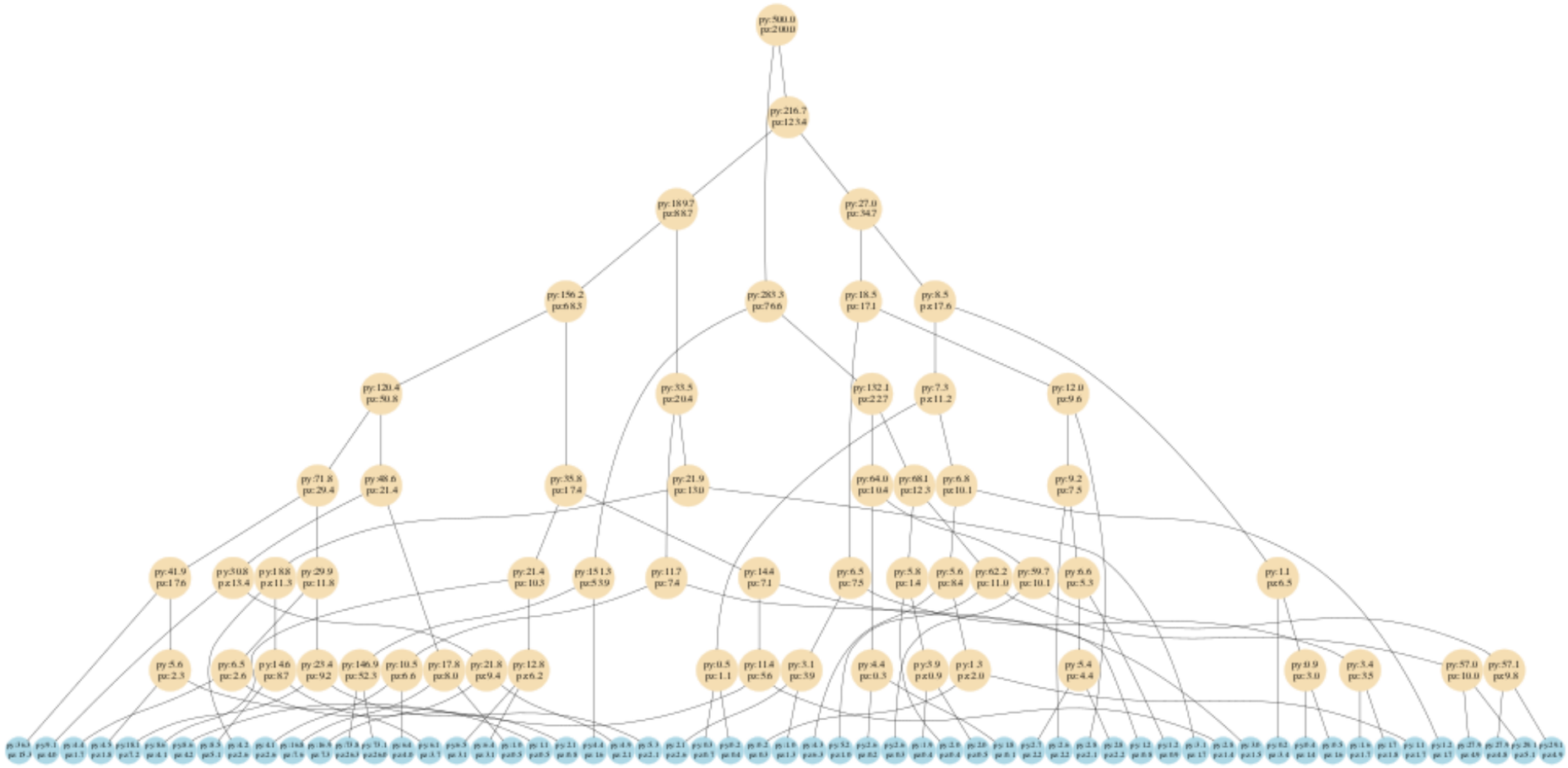
Truth jet



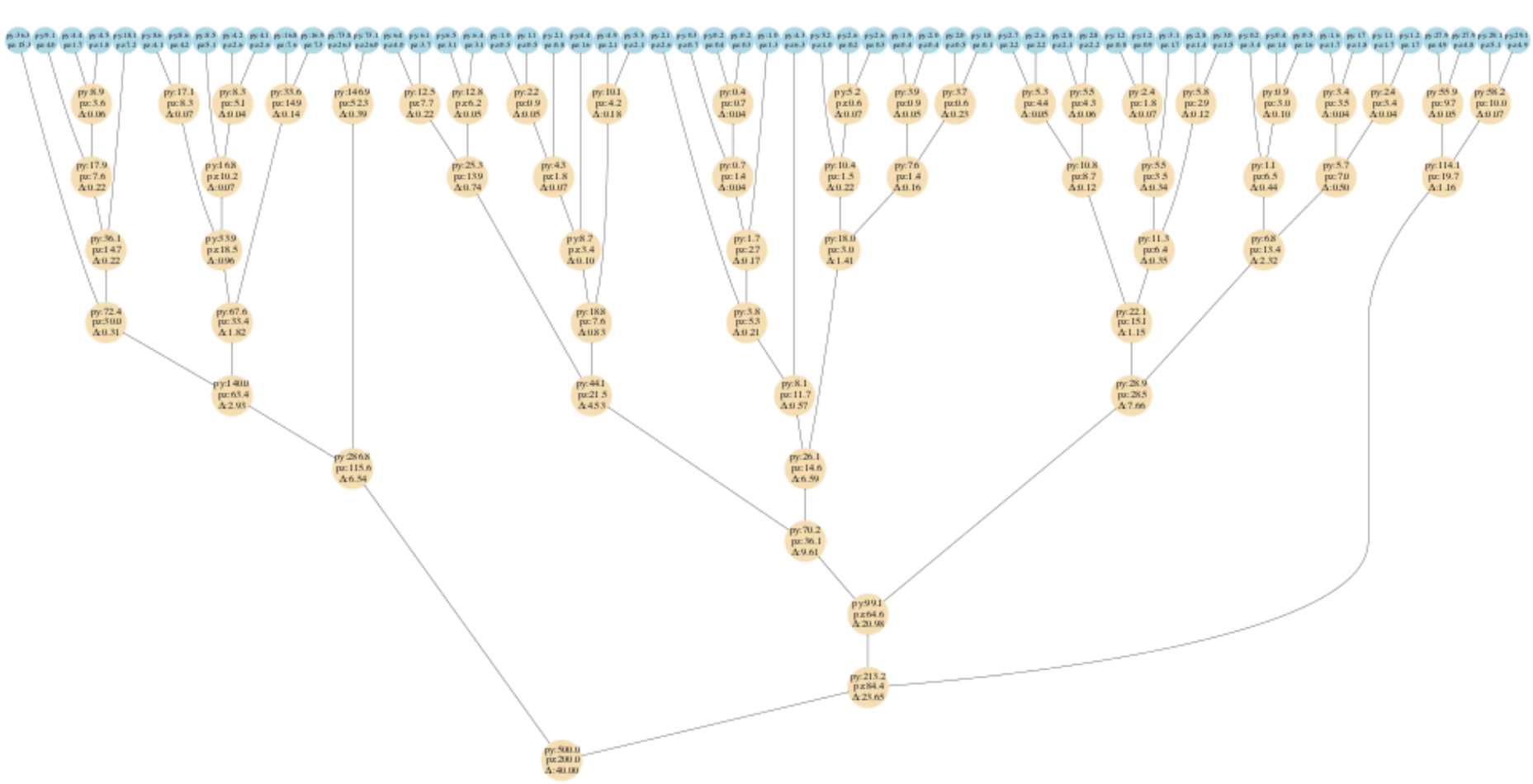
Anti-kt jet



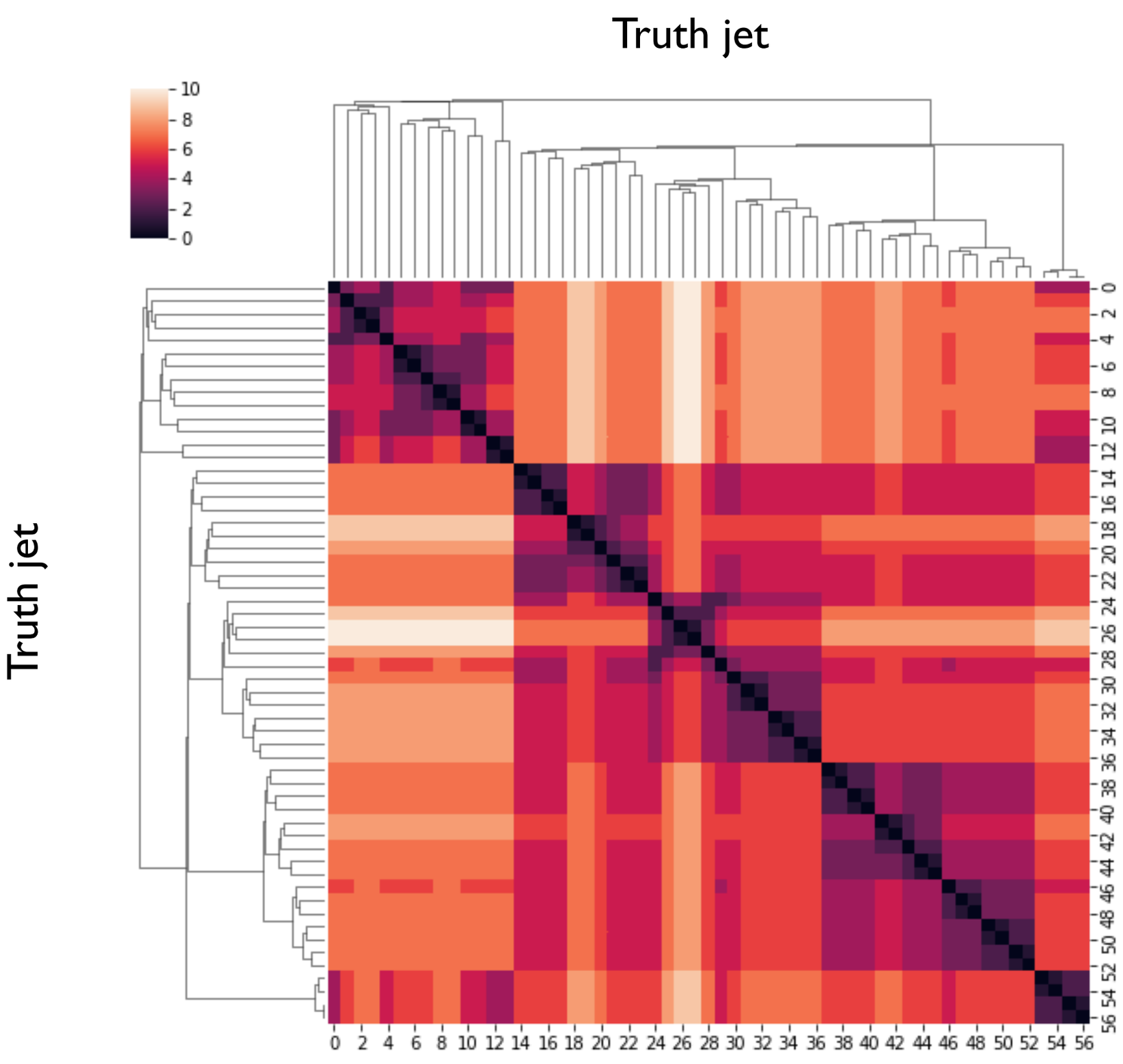
kt jet



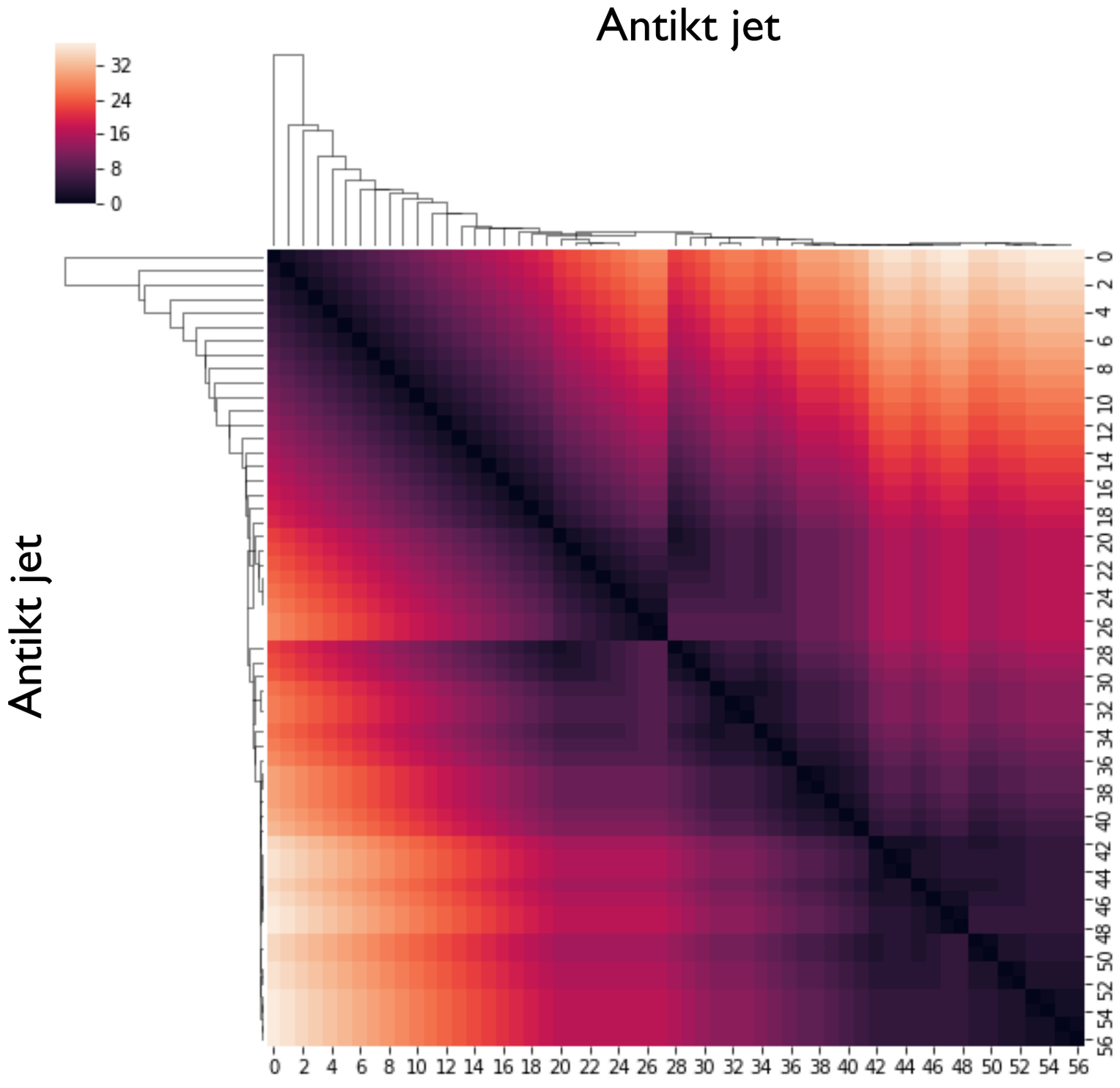
Truth jet



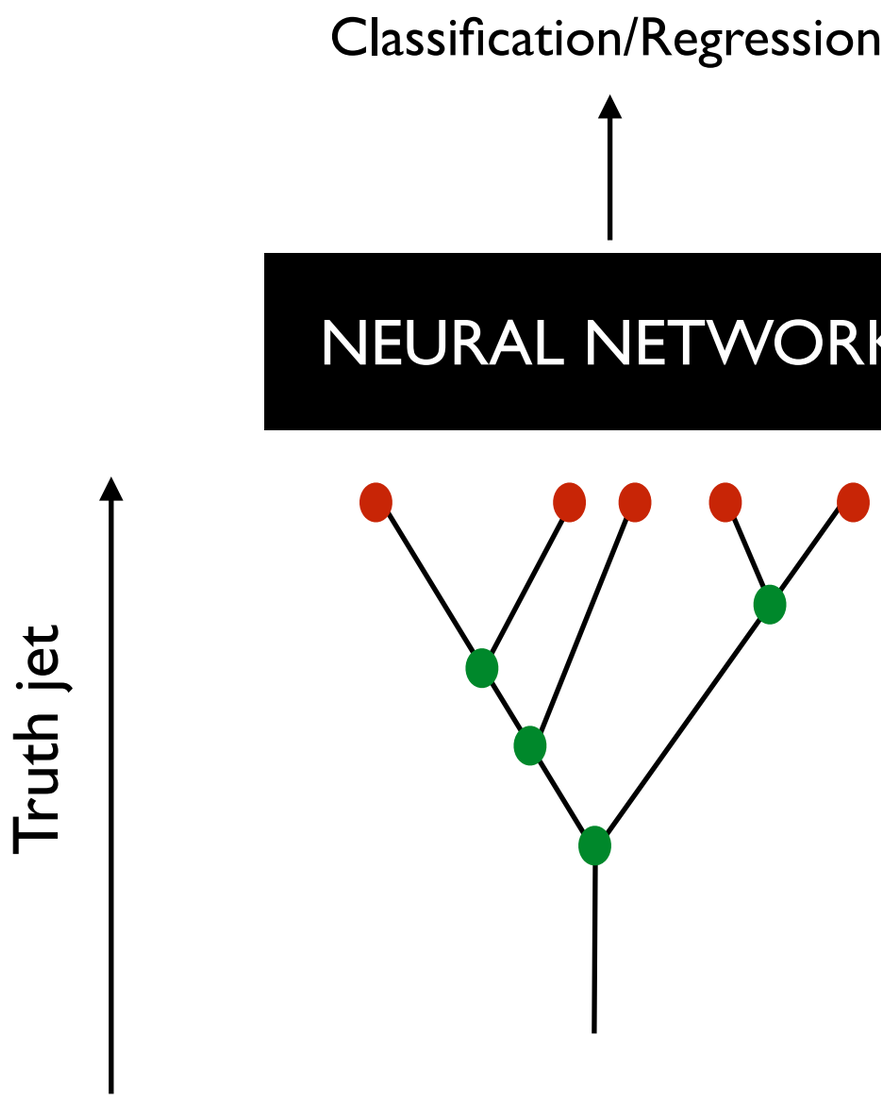
Heat dendrogram cluster maps



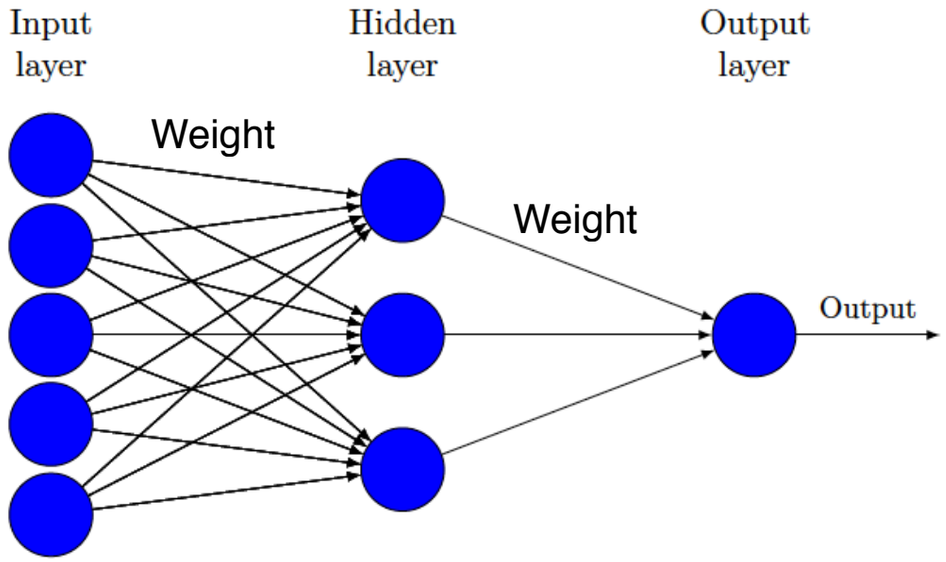
Heat dendrogram cluster maps



Classification and regression



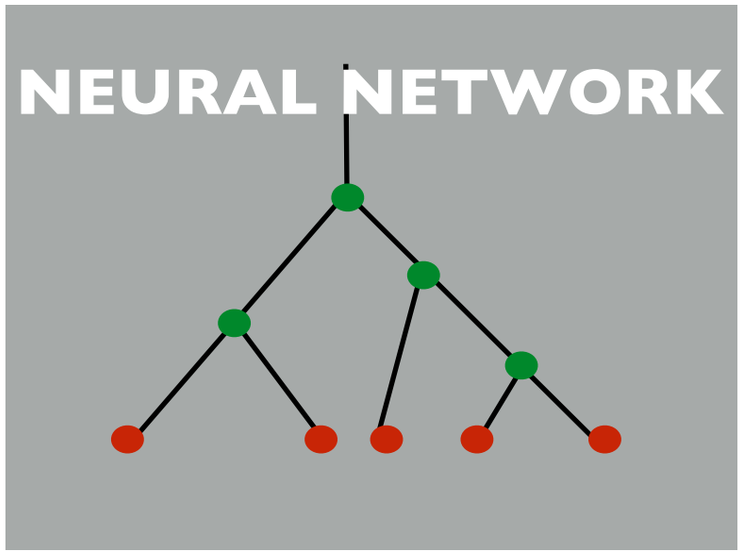
Neural Networks (NNs)



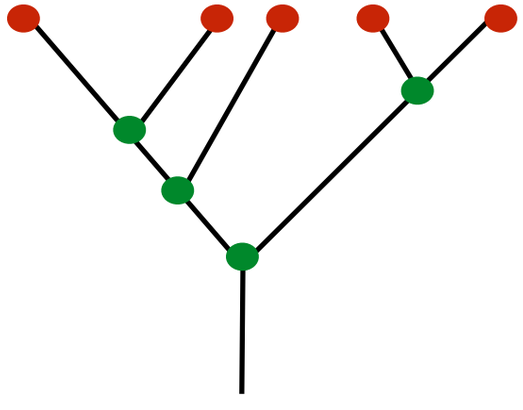
- Organize features in a hierarchical way.
- Build connections among them.
- Go from simple features to more complex and abstract ones. Features are arranged in *layers* and the connections are called *weights*.

Can we have a NN that encodes the tree structure?

Classification/Regression

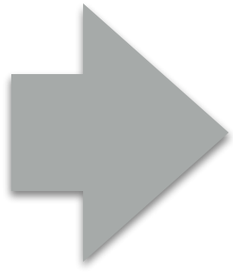


Truth jet

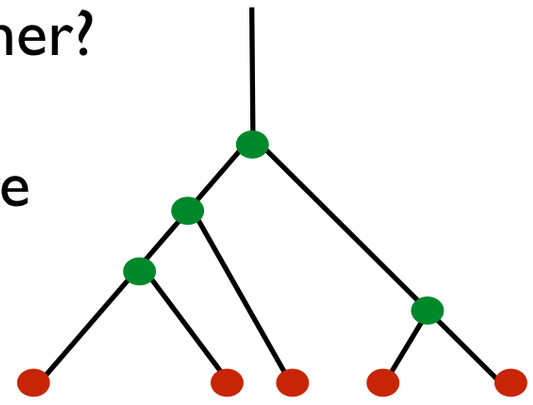


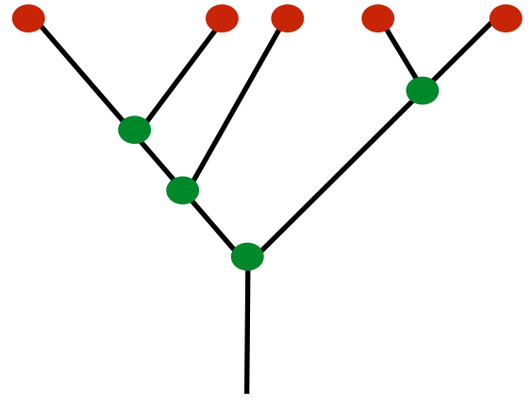
Can we solve both tasks together?

Find the right latent structure
+
classification/regression

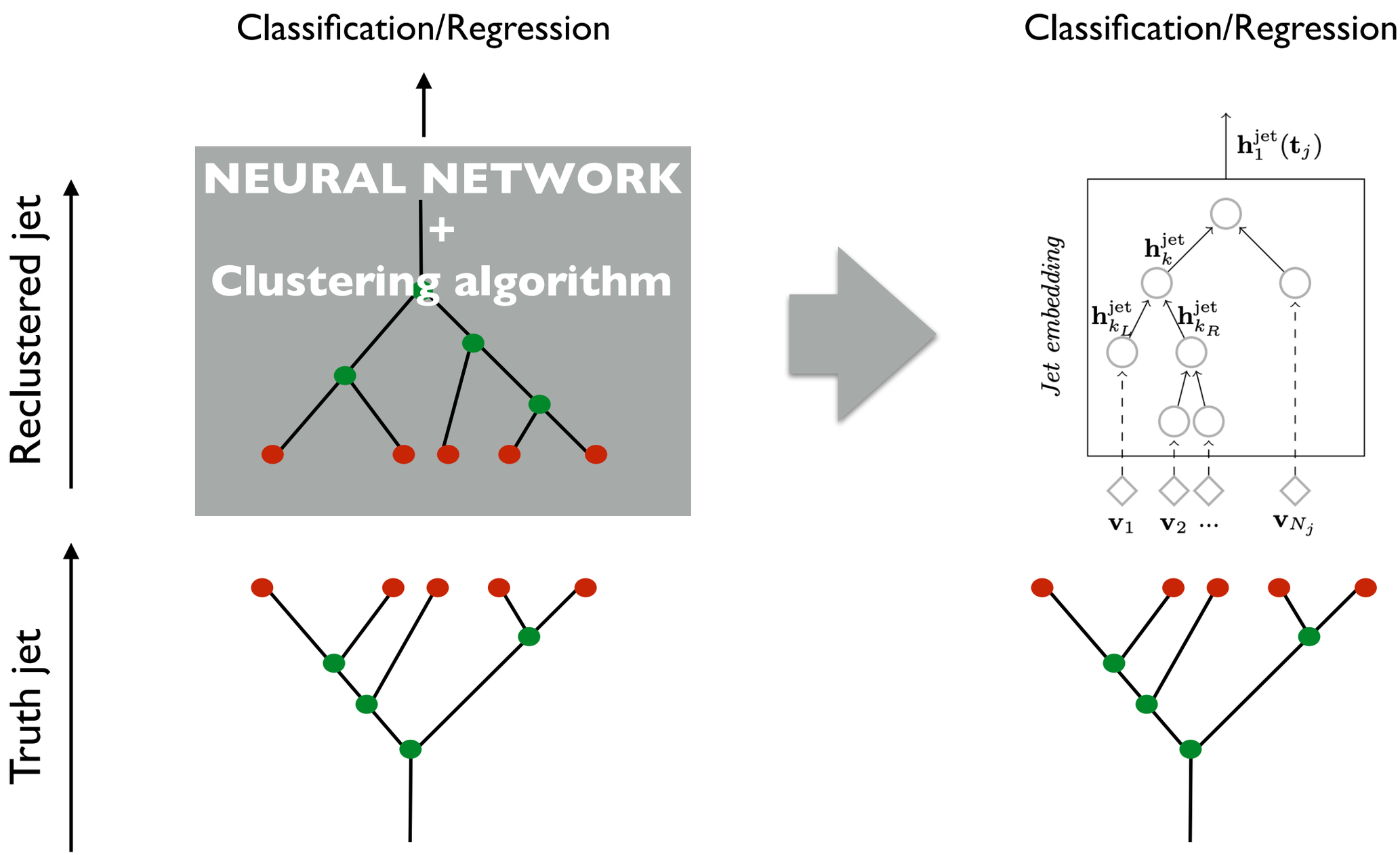


Classification/Regression

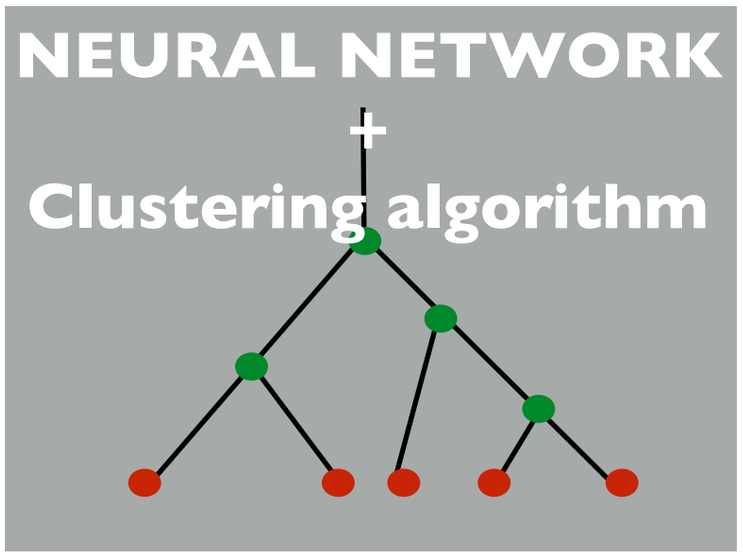




Can we have a NN that encodes the tree structure?



Can we have a NN that encodes the tree structure?



JUNIPR

Recursive NN

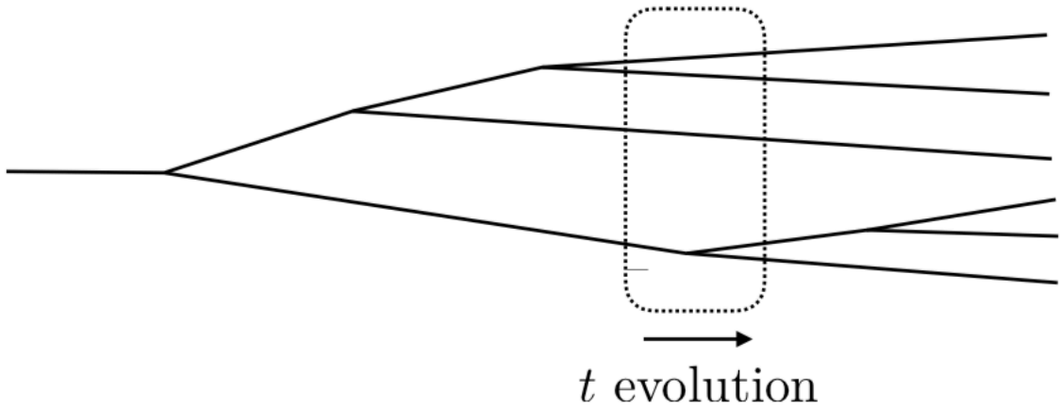
JUNIPR: Jets from UNsupervised Interpretable PRobabilistic models

[A. Andreassen, I. Feige, C. Frye, M. Schwartz, '18]
[A. Andreassen, I. Feige, C. Frye, M. Schwartz, '19]

- Discrimination: based on likelihood ratio
- Generation: yields a probabilistic model that leads to data-driven simulations

Recurrent NN implementation

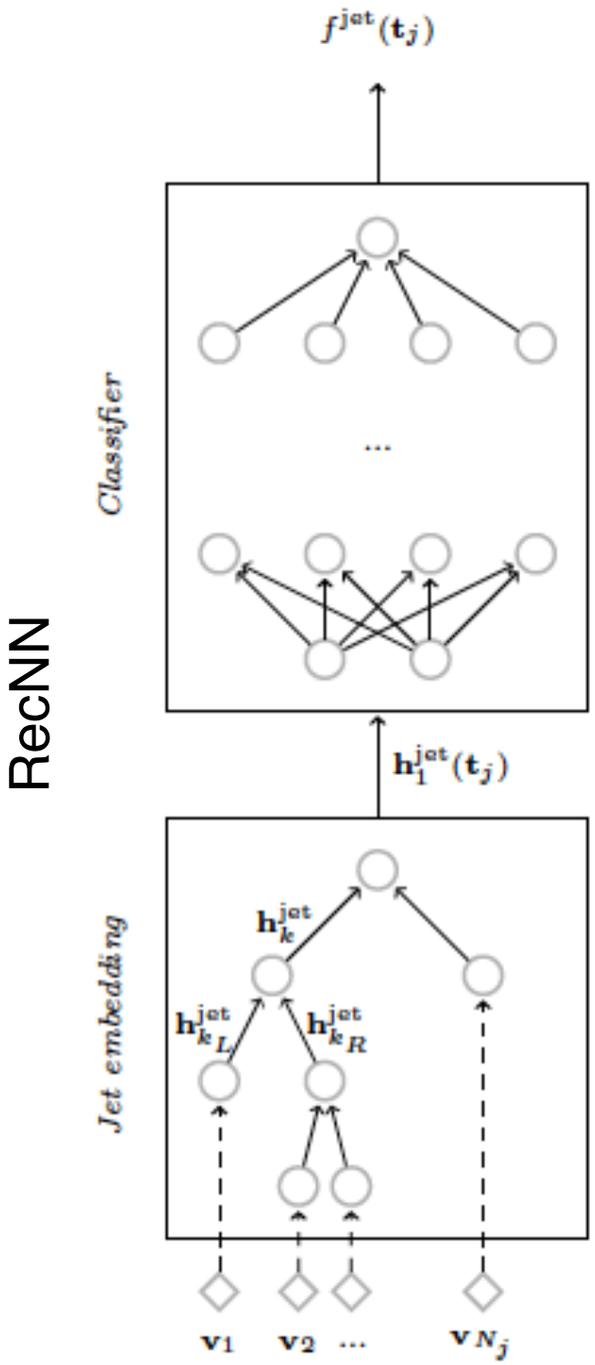
Probabilities attached to individual nodes.



We could add physics structure to the generative process!

We could have a unified picture => same physics model for generation an inference

Recursive Neural Networks



[Louppe, Cho, Becot, Cranmer '17]

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

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

- PyTorch batch training implementation (initial code in Numpy).
- RecNN much fewer trainable parameters: ~10000 vs ~1 M
- Input features:

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

Network in Network (NiN) Mechanism

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

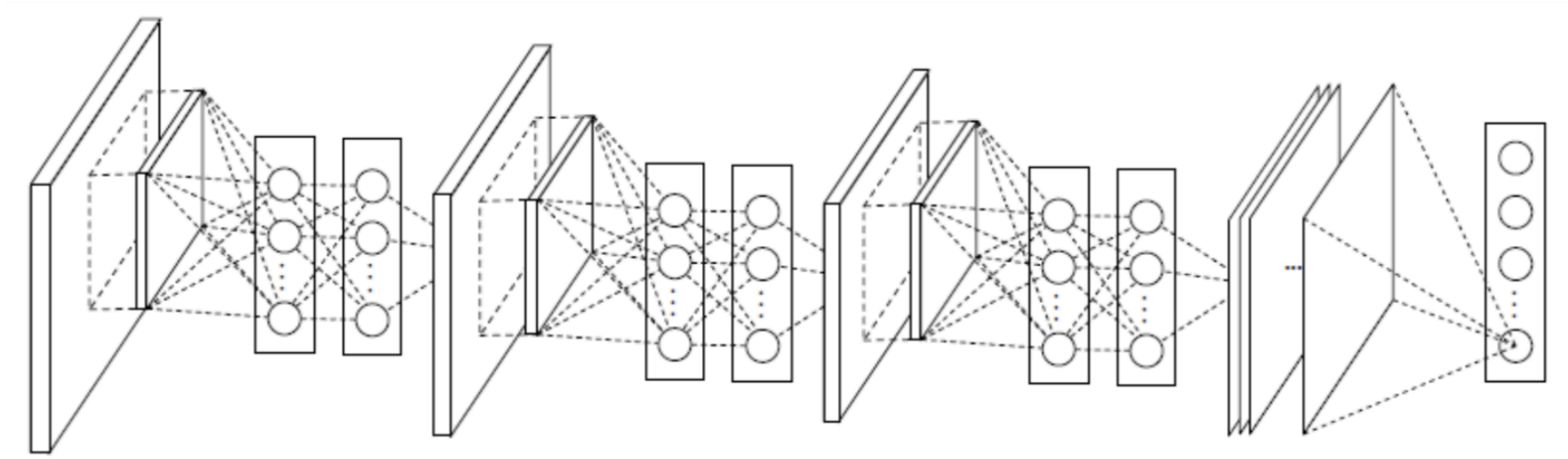


Fig. from arXiv:1312.4400

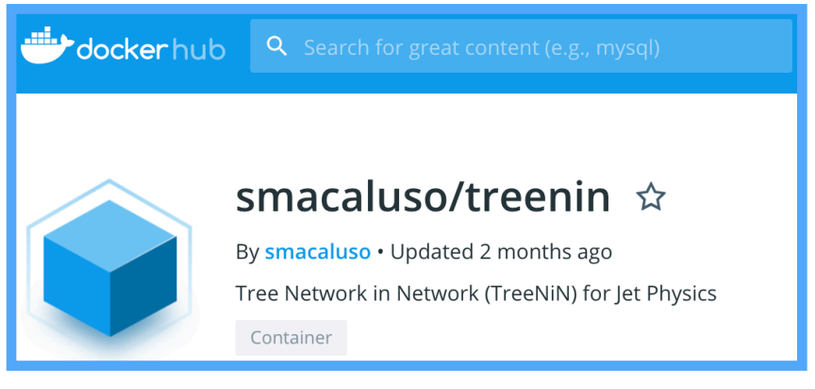
Tree Network in Network

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

Tree Network in Network (TreeNiN) for Jet Physics

Sebastian Macaluso and Kyle Cranmer

Note that this is an early development version.



DOI 10.5281/zenodo.2600148 License MIT docker pull

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

Introduction

In this method, a tree neural network (TreeNN) is trained on jet trees. The TreeNN provides a *jet embedding*, which maps a set of 4-momenta into a vector of fixed size and can be trained together with a successive network used for classification or regression (see Louppe et al. 2017, "QCD-Aware Recursive Neural Networks for Jet Physics" for more details). Jet constituents are reclustered to form binary trees, and the topology is determined by the clustering algorithm (e.g. kt, anti-kt or Cambridge/Aachen). We chose the kt clustering algorithm, and 7 features for the nodes: |p|, eta, phi, E, E/Ejet, pT and theta. We scaled each feature with the scikit-learn preprocessing method RobustScaler (this scaling is robust to outliers).

Implementing the TreeNiN on the *Top Tagging Reference Dataset*

This repository includes all the code needed to implement the TreeNiN on the *Top Tagging Reference Dataset*. A description and link to the Top Tagging Reference Dataset (provided by Gregor Kasieczka, Michael Russel and Tilman Plehn) can be found here with the link to download it here. This dataset contains about 1.2M training events, 400k validation events, 400k test events with equal numbers of top quark and qcd jets. Only 4 momentum vectors of the jet constituents.

The Machine Learning Landscape of Top Taggers

G. Kasieczka (ed)[1], T. Plehn (ed)[2], A. Butter[2], K. Cranmer[3], D. Debnath[4], B. M. Dillon[5], M. Fairbairn[6], W. Fedorko[7], D. A. Faroughy[5], C. Gay[7], L. Gouskos[8], J. F. Kamenik[5,9], P. T. Komiske[10], S. Leiss[1], A. Lister[7], S. Macaluso[3,4], E. M. Metodiev[10], L. Moore[11], B. Nachman,[12,13], K. Nordström[14,15], J. Pearkes[7], H. Qu[8], Y. Rath[16], M. Rieger[16], D. Shih[4], J. M. Thompson[2], and S. Varma[6]

[arXiv:1902.09914]

Heavy resonance tagging

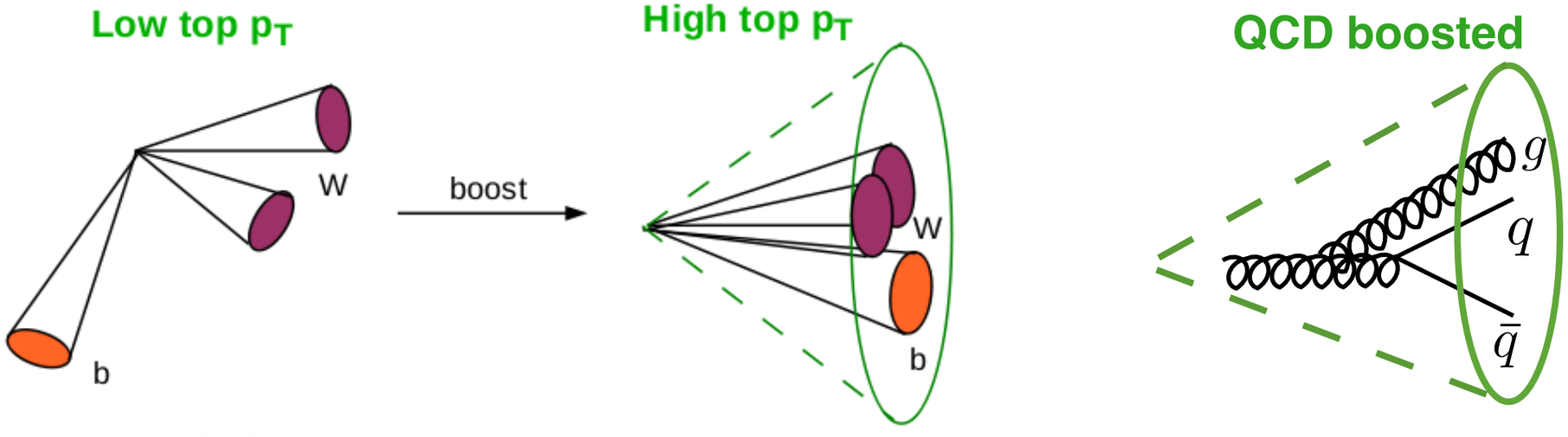


Fig. from www.quantumdiaries.org

- Typical classifiers search for:

  ✦Kinematic features induced by the top and W boson mass.
  ✦Number of prongs: N-subjettiness.
  ✦Flavor: b-tagging
  ✦Inclusive approach: e.g. HEPTopTaggerV2

- Traditional technique: apply these variables on a cut and count analysis or as high-level inputs of multivariate machine learning algorithms (BDTs).

- Goal of this study:

Compare ML based setups to classify top vs QCD jets.

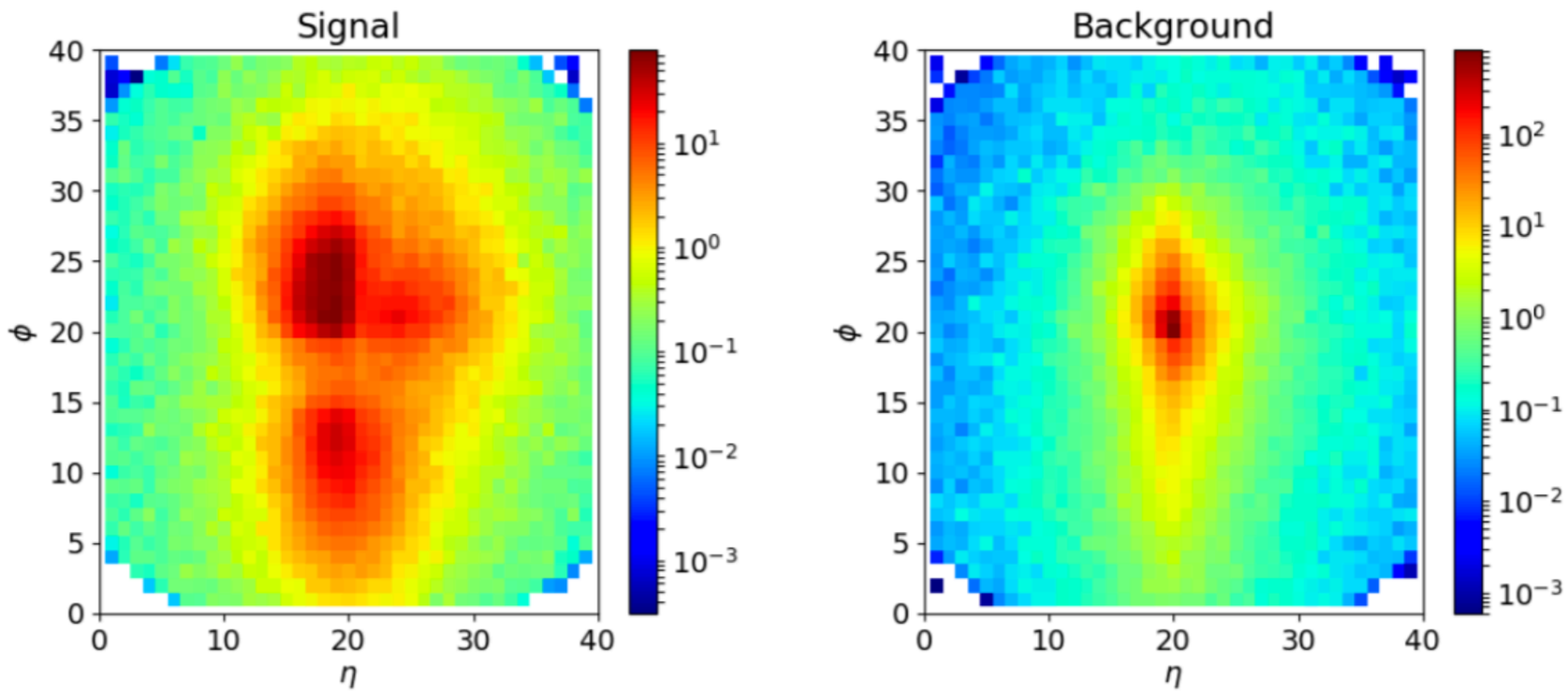
Top Tagging Reference Dataset

- **Top vs QCD jets** produced with Pythia8 @ 14TeV.
- Anti-kt R=0.8 Delphes jets (energy-flow algorithm), pT=[550,650] GeV, match and merge requirements.
- No uncertainties, pile-up.
- Only 4-vectors up to 200 constituents. (No charge or displaced vertex information)
- (1.2M, 400k, 400k) jets as (train, val, test) sets.

Contact

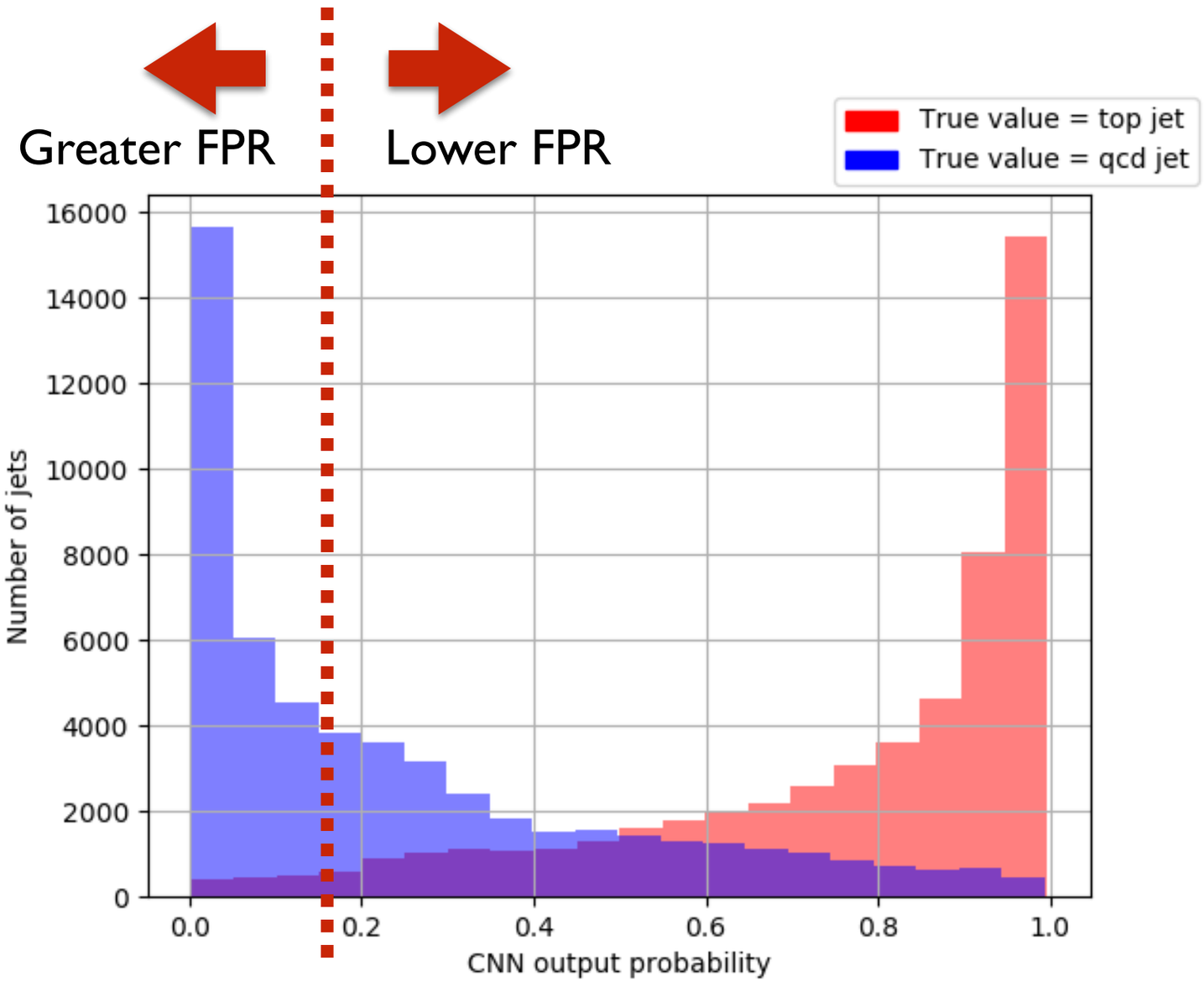
Gregor Kasieczka (gregor.kasieczka@cern.ch)
Michael Russel (russell@thphys.uni-heidelberg.de)
Tilman Plehn (plehn@uni-heidelberg.de)

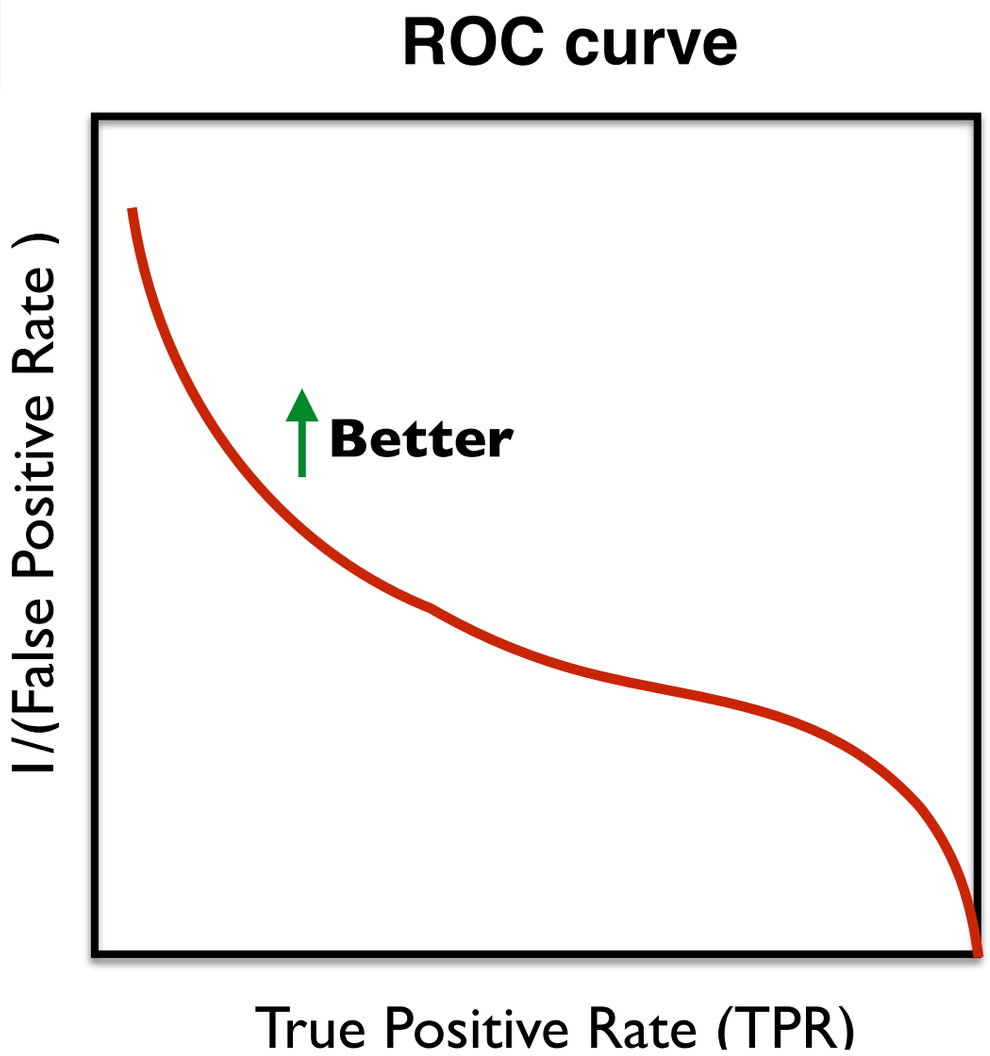
Get dataset here



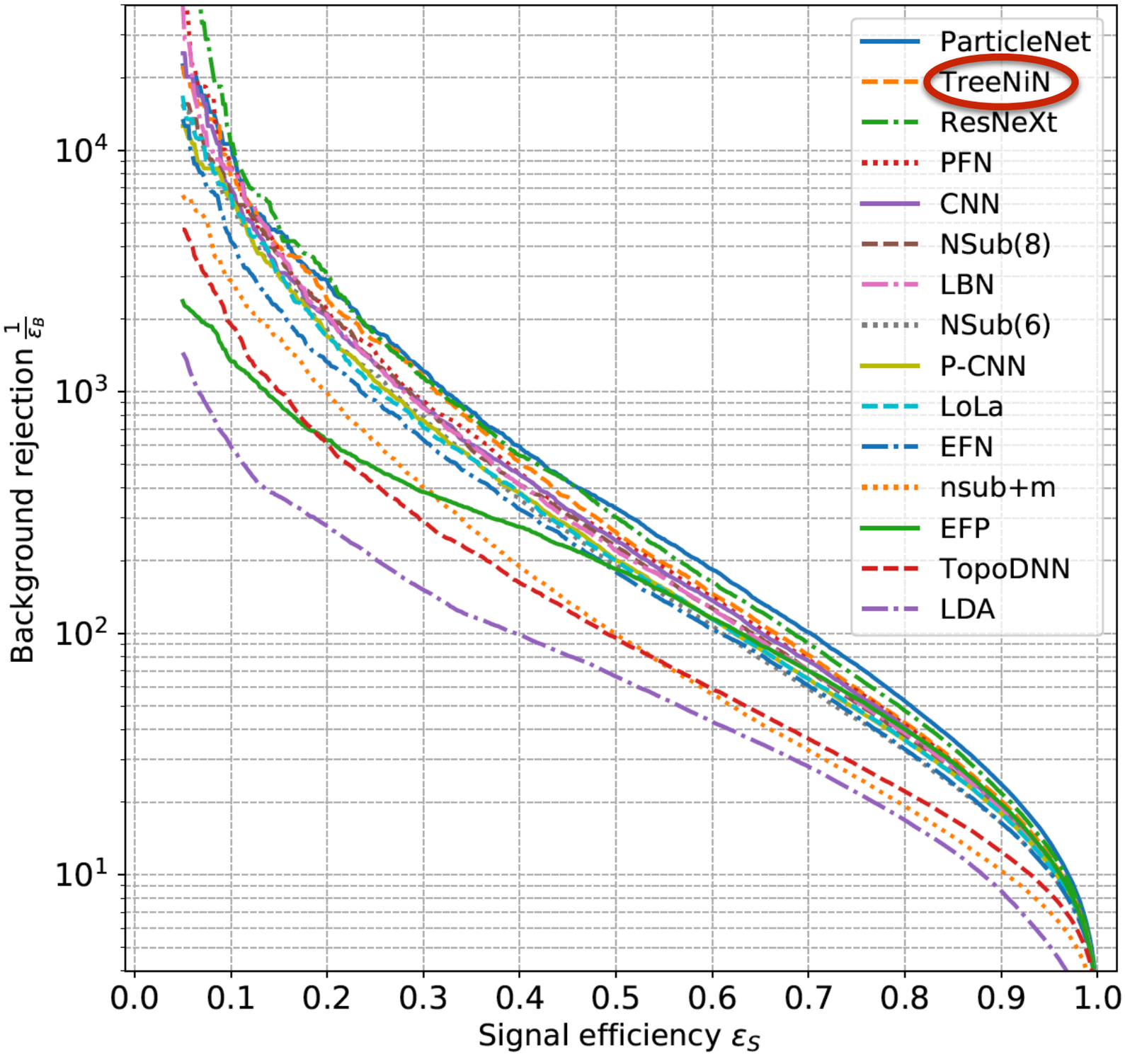
Average of 10k jet images

ROC curves


Greater FPR
Lower FPR
True value = top jet
True value = qcd jet
Number of jets
16000
14000
12000
10000
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0
CNN output probability


ROC curve
1/(False Positive Rate)
Better
True Positive Rate (TPR)

Comparison



- 9 models trained for each classifier.
- ROC curve for the model that gives the median AUC.

CNN
[SM & D. Shih, 1803.00107]

CMS ImageTop initially based on this work

Performance metrics

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$) single	mean	median	#Param
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [31]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
LDA [54]	0.955	0.892	151±0.4	151.5±0.5	151.7±0.4	184k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k

New results from ParticleNet group at Boost '19 => ParticleNet: $1/\epsilon_B(\epsilon_S = 0.3) = 1615 \pm 93$

ParticleNet-Lite $1/\epsilon_B(\epsilon_S = 0.3) = 1262 \pm 49$

After this top tagging comparison study,

let's return to discuss about jet generation and

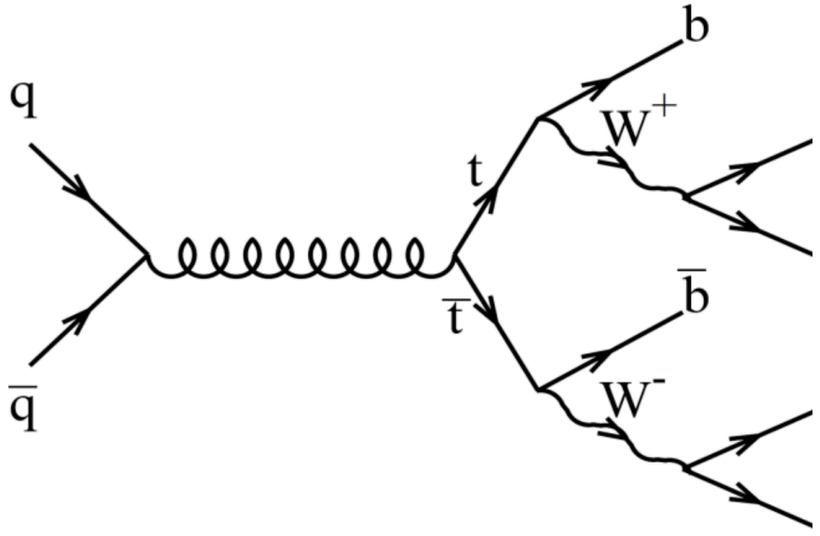
clustering.

Event generation for hadron colliders

Theory parameters

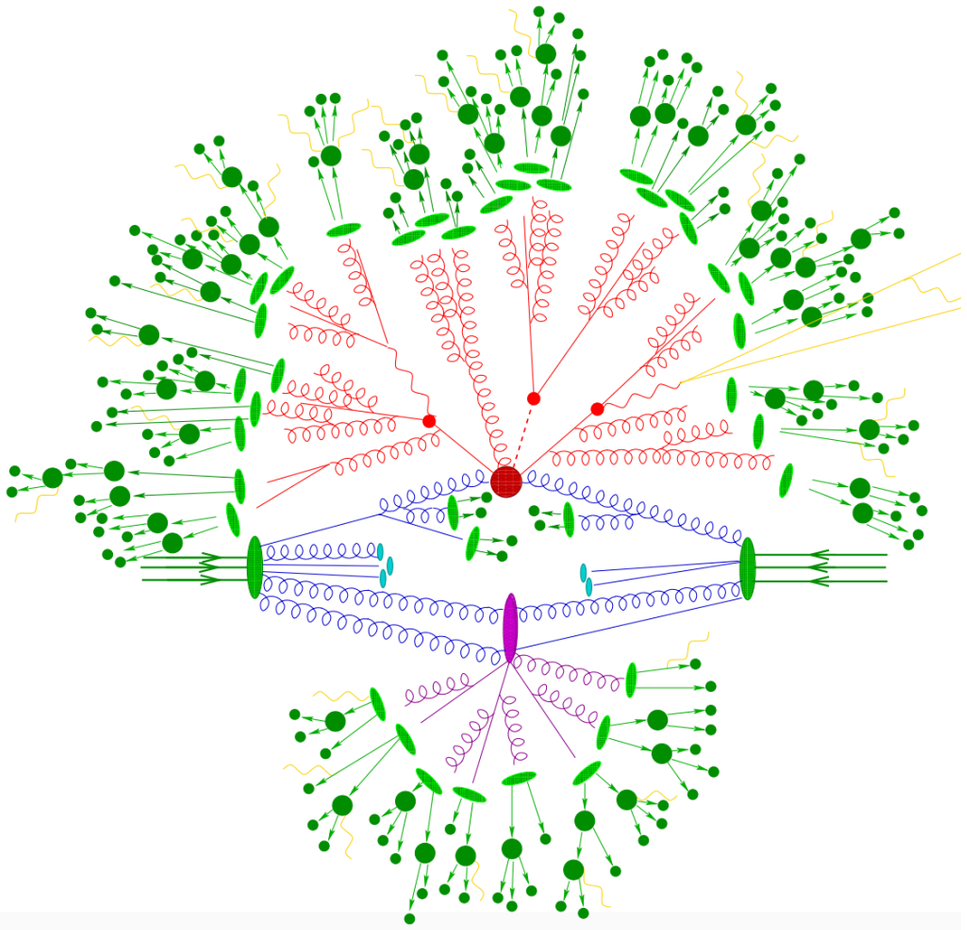
Parton-level momenta

$$\theta \longrightarrow z_p$$



Hadron - hadron collision

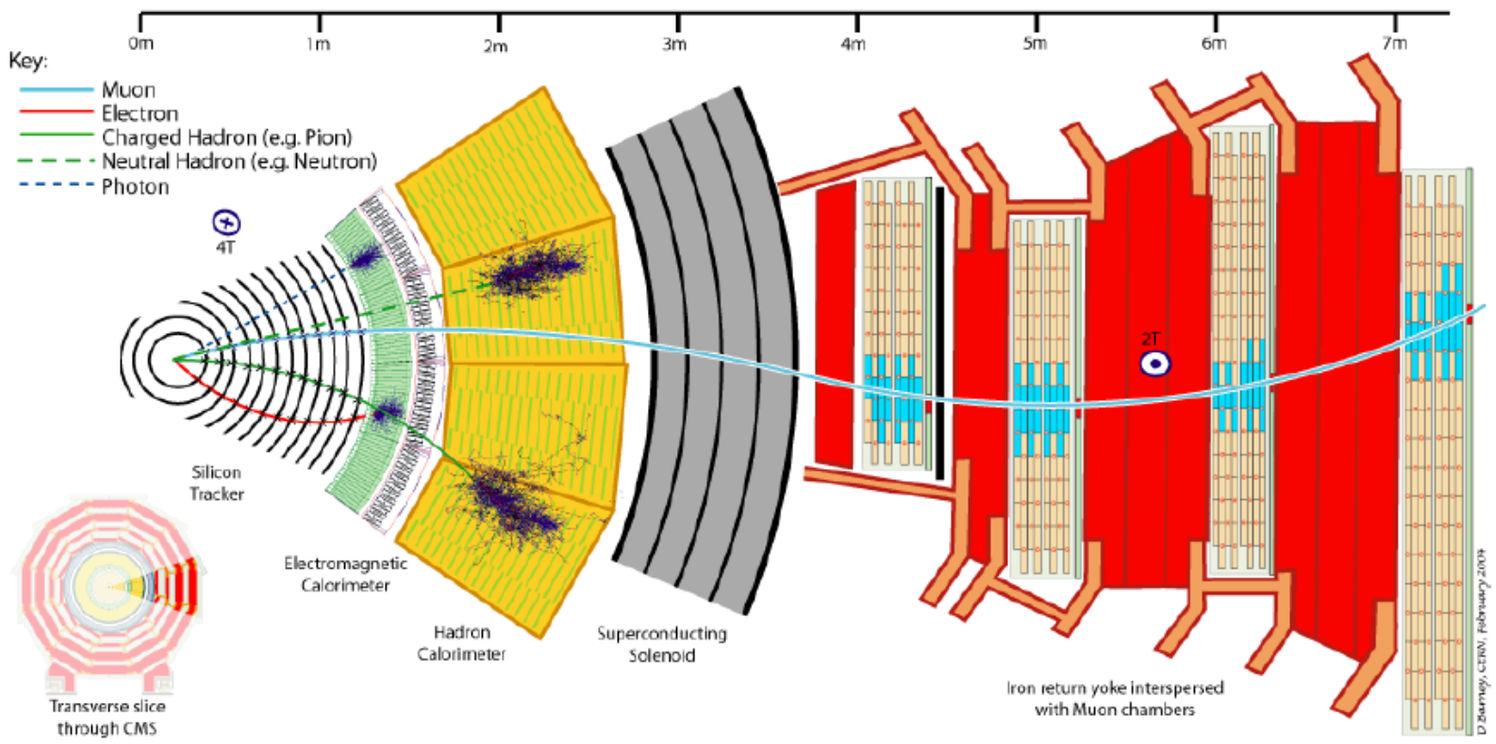
Theory parameters θ → Parton-level momenta z_p → Shower and hadronization z_s



[Stefan Hoche 1411.4085]

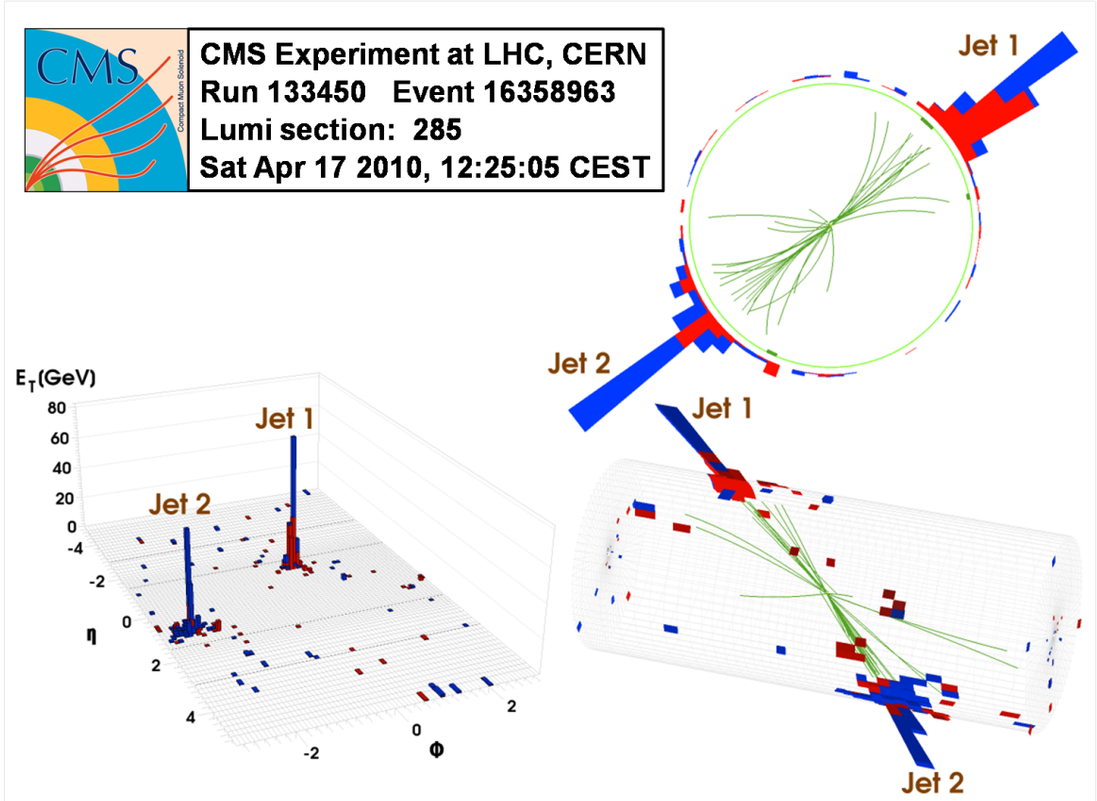
Hadron - hadron collision

Theory parameters θ → Parton-level momenta z_p → Shower and hadronization z_s → Detector interactions z_d

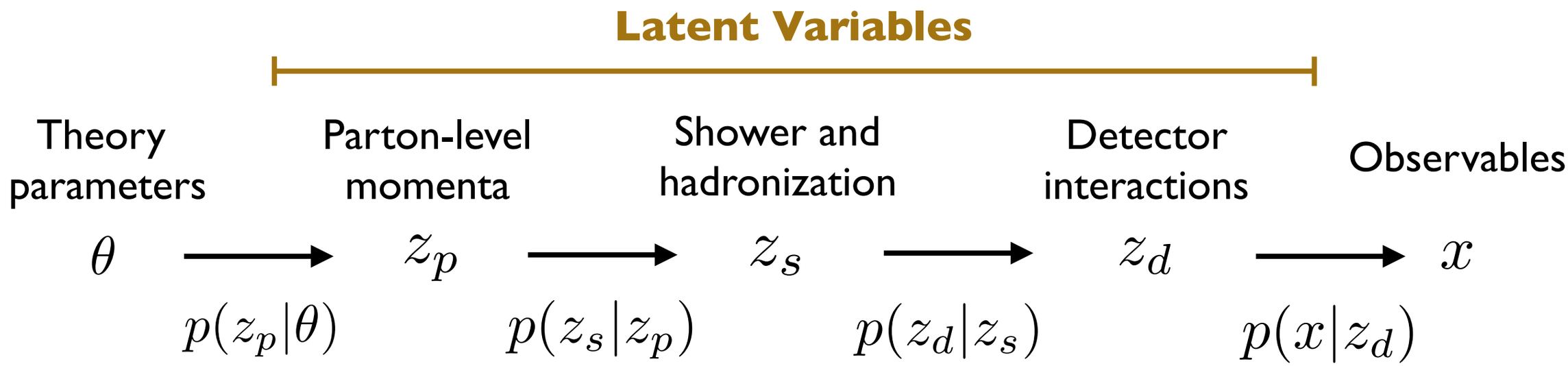


Hadron - hadron collision

Theory parameters θ → Parton-level momenta z_p → Shower and hadronization z_s → Detector interactions z_d → Observables x



Hadron - hadron collision



Sherpa

PYTHIA

GEANT4
A SIMULATION TOOLKIT

Herwig 7

Hadron - hadron collision

Latent Variables

Theory parameters		Parton-level momenta		Shower and hadronization		Detector interactions		Observables
θ	$\longrightarrow$	z_p	$\longrightarrow$	z_s	$\longrightarrow$	z_d	$\longrightarrow$	x

$$\boxed{\int dz_d \int dz_s \int dz_p} \quad \boxed{p(z_p|\theta)} \qquad p(z_s|z_p) \qquad p(z_d|z_s) \quad p(x|z_d) = p(x|\theta)$$

Infeasible to calculate the integral over this space

Parton-level likelihood Can be evaluated

Given an observable, we can calculate the joint likelihood ratio conditional on a specific latent path

$$r(x, z|\theta_0\theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)} = \frac{p(x|z_d)}{p(x|z_d)} \frac{p(z_d|z_s)}{p(z_d|z_s)} \frac{p(z_s|z_p)}{p(z_s|z_p)} \boxed{\frac{p(z_p|\theta_0)}{p(z_p|\theta_1)}}$$

[Brehmer, Cranmer, Louppe, Pavez 1805.00013, 1805.00020, 1805.12244]

Hadron - hadron collision

Latent Variables

Theory parameters θ → Parton-level momenta z_p → Shower and hadronization z_s → Detector interactions z_d → Observables x

$$\int dz_d \int dz_s \int dz_p \boxed{p(z_p|\theta)} \quad \boxed{p(z_s|z_p)} \quad p(z_d|z_s) \quad p(x|z_d) = p(x|\theta)$$

Parton-level likelihood Can be evaluated

Shower-level likelihood Can be evaluated?

$p(z_s|z_p,\theta)$ If θ describes the shower

Prediction (Generative Model)

Inference

Unify both processes → **Generative Model with a tractable likelihood**

Toy Generative Model for Jets

Kyle Cranmer, SM & Duccio Pappadopulo

https://github.com/SebastianMacaluso/ToyJetsShower

Generation

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

Inference

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

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

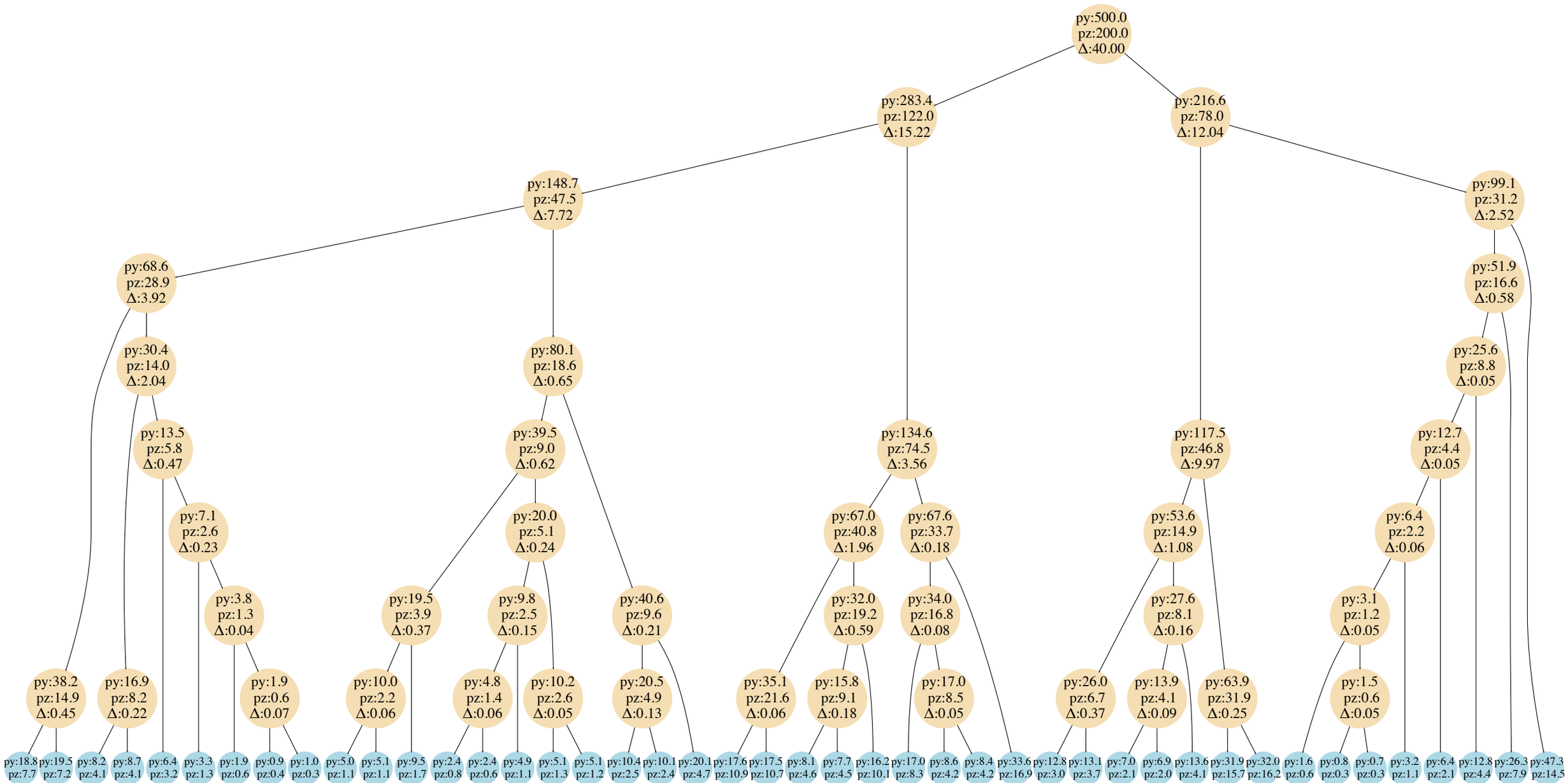
Jets properties

- Permutation invariance
- Distance measure

=> An analogue of the generalized k_t clustering algorithms can be defined.

Sample jet

Jet constituents + showering path => jet binary tree



Generative process

We define a model with a given decaying rate λ, initial splitting scale Δ_0 and cut-off scale $\Delta_{\min}$.

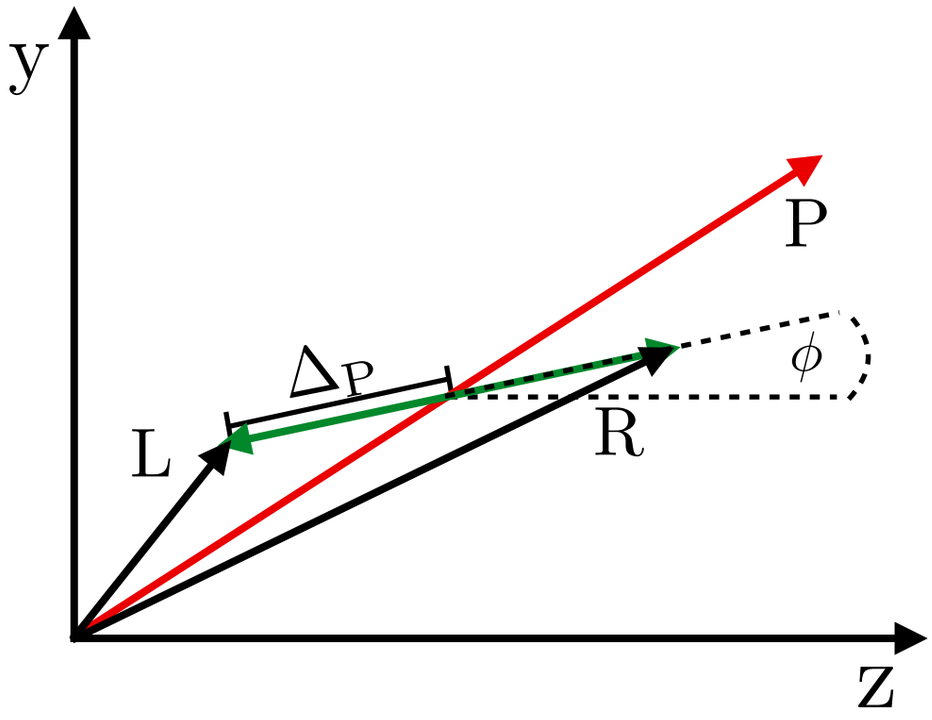
Splitting of a parent node

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

$$\Delta_{\mathrm{p}} = |\vec{\Delta}_{\mathrm{p}}|$$

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

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



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

$$\phi \sim \mathrm{Uniform}(0, 2\pi)$$

Heavy resonance X jet:

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

Algorithm

Algorithm 1: Toy Parton Shower Generator

1 function Exp2DShower $(\vec{p}_{\mathrm{p}}, \Delta_{\mathrm{p}}, \Delta_{\mathrm{cut}}, \lambda, \mathrm{tree})$

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

2 Add parent node to tree.

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

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

5 $\vec{\Delta}_{\mathrm{p}} = \Delta_{\mathrm{p}}\ (\sin\phi_{\mathrm{p}}, \cos\phi_{\mathrm{p}})$

6 $\vec{p}_{\mathrm{L}} = \frac{1}{2}\vec{p}_{\mathrm{p}} - \vec{\Delta}_{\mathrm{p}}$

7 $\vec{p}_{\mathrm{R}} = \frac{1}{2}\vec{p}_{\mathrm{p}} + \vec{\Delta}_{\mathrm{p}}$

8 draw Δ_{L}, Δ_{R} from the exponential distribution in Eq. 3.

9 Exp2DShower $(\vec{p}_{\mathrm{L}}, \Delta_{\mathrm{L}}, \Delta_{\mathrm{cut}}, \lambda, \mathrm{tree})$

10 Exp2DShower $(\vec{p}_{\mathrm{R}}, \Delta_{\mathrm{R}}, \Delta_{\mathrm{cut}}, \lambda, \mathrm{tree})$

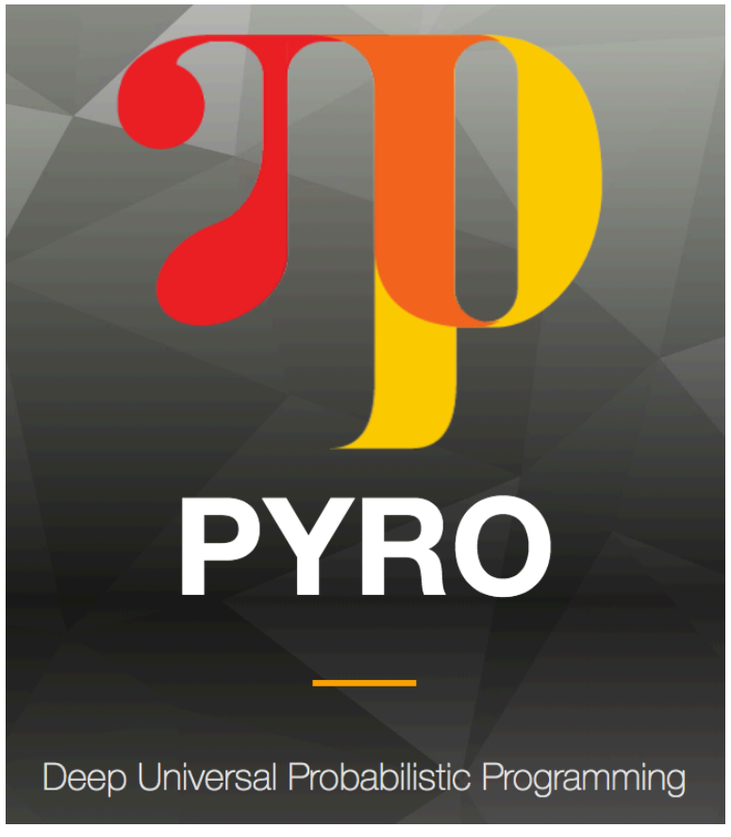
Augmented data

Additional information that characterizes the latent process can be extracted:

- Joint likelihood
- Joint likelihood ratio
- Joint score

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

=> The model allows the implementation of new techniques for likelihood-free inference.
[J. Brehmer, G. Louppe, J. Pavez & K. Cranmer 2018]



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

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

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

Greedy algorithms

Choose the optimal decision locally at each step.

Generalized kt clustering algorithms

Current implementations of the generalized kt clustering algorithms locally minimize the distance measure d_{ij}

Greedy likelihood-based algorithm

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

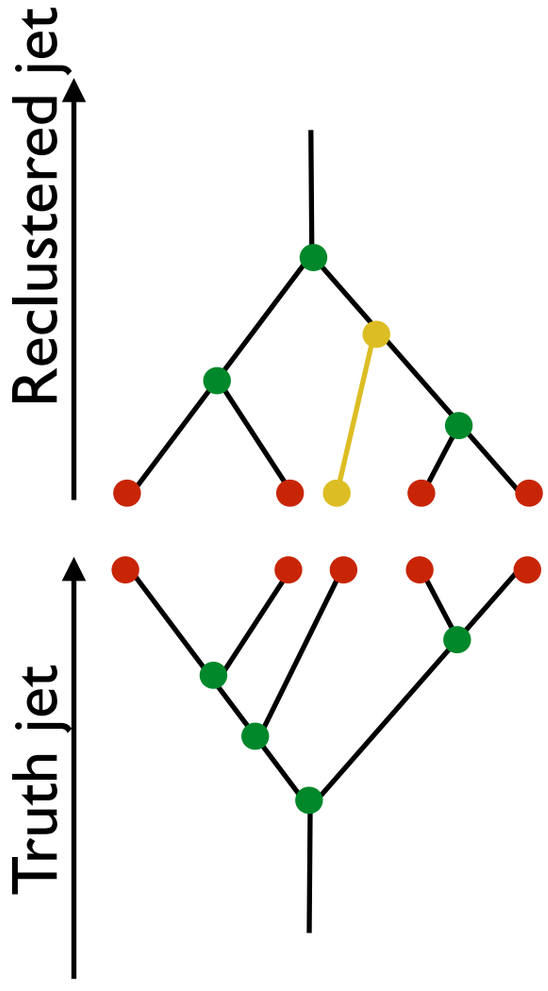
$$p(\Delta_{\mathrm{L}}, \Delta_{\mathrm{R}} | \Delta_{\mathrm{p}})$$

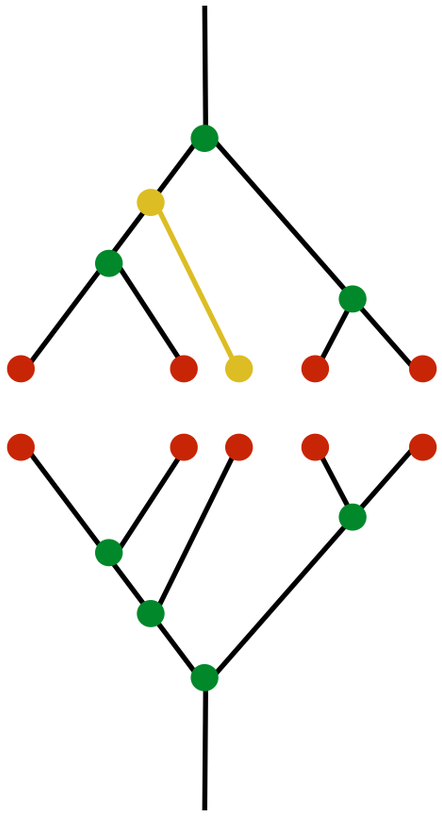
Our implementation improves the N^3 brute force time complexity to N^2 following the approach based on nearest neighbors introduced in FastJet [hep-ph/0512210, G. Salam and M. Cacciari LPTHE-06-02].

Likelihood-based 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.





Joint likelihood:

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

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

Given an algorithm, z is fixed.

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

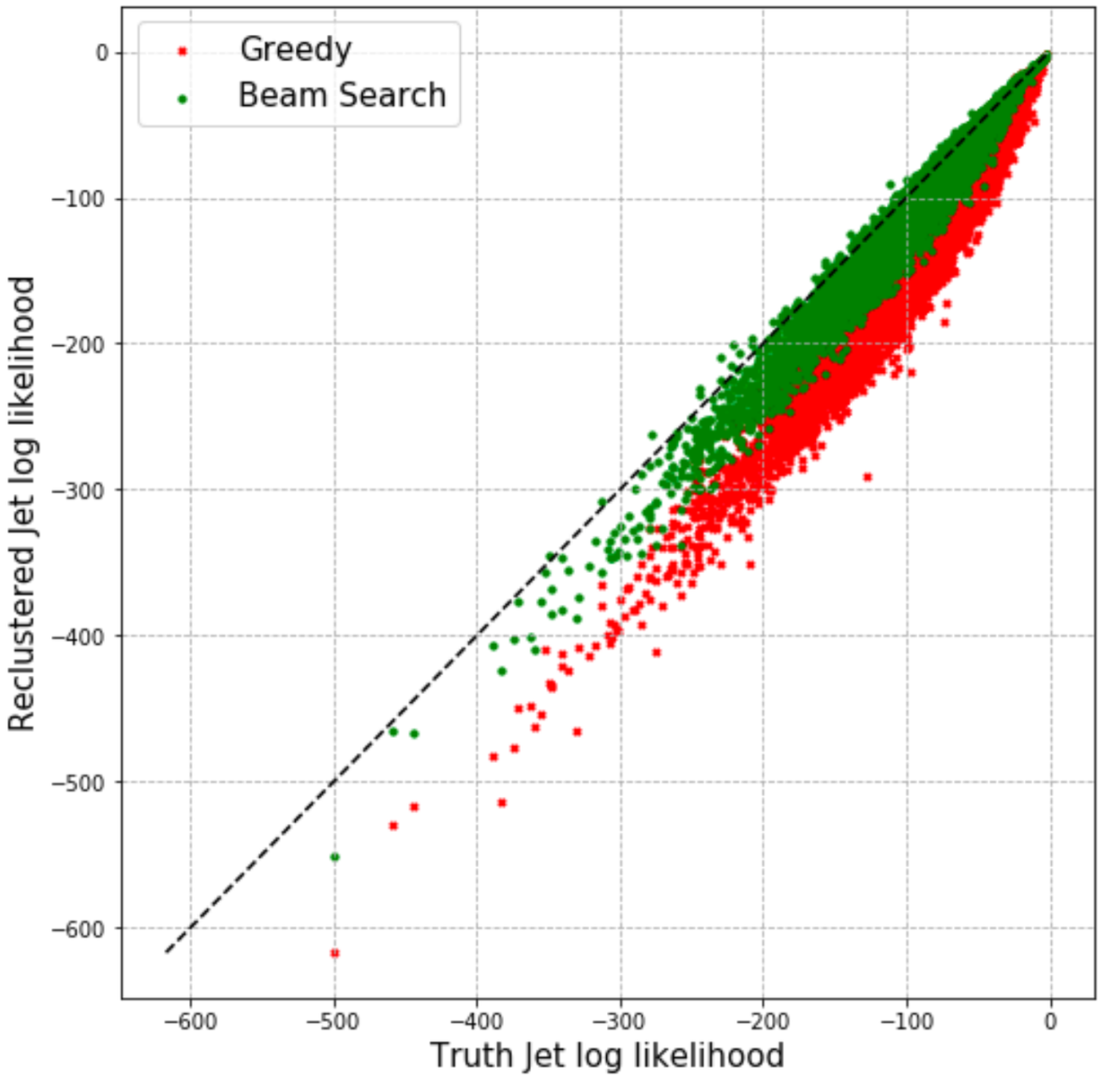
$\hat{z}_{\mathrm{BS}}$

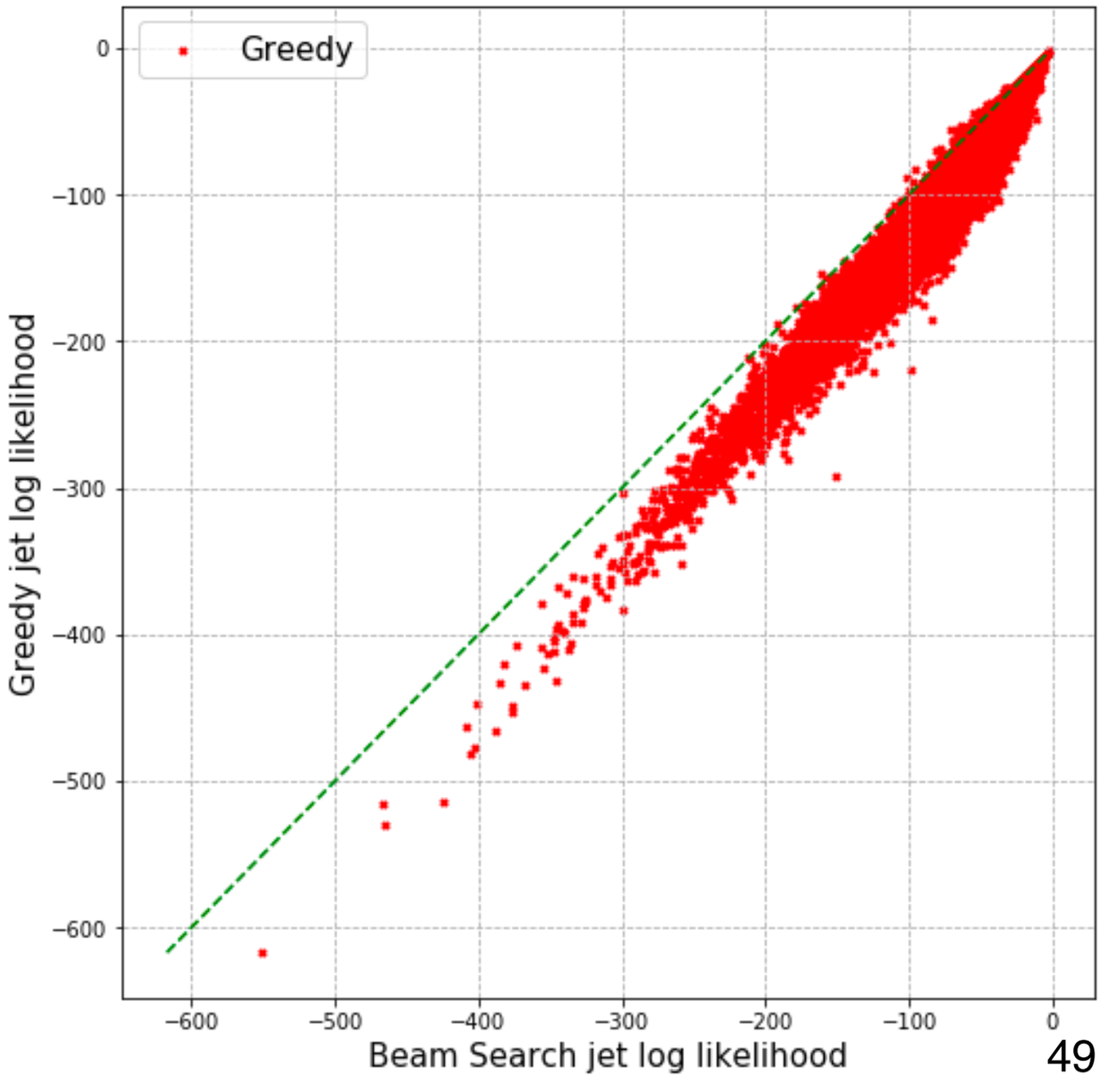
$\hat{z}_{\mathrm{MLE}} = \mathrm{argmax}_z \; p(x|z)$
Viterbi algorithm
Computationally "infeasible" for jets with ~ 100 constituents

Jets log likelihood

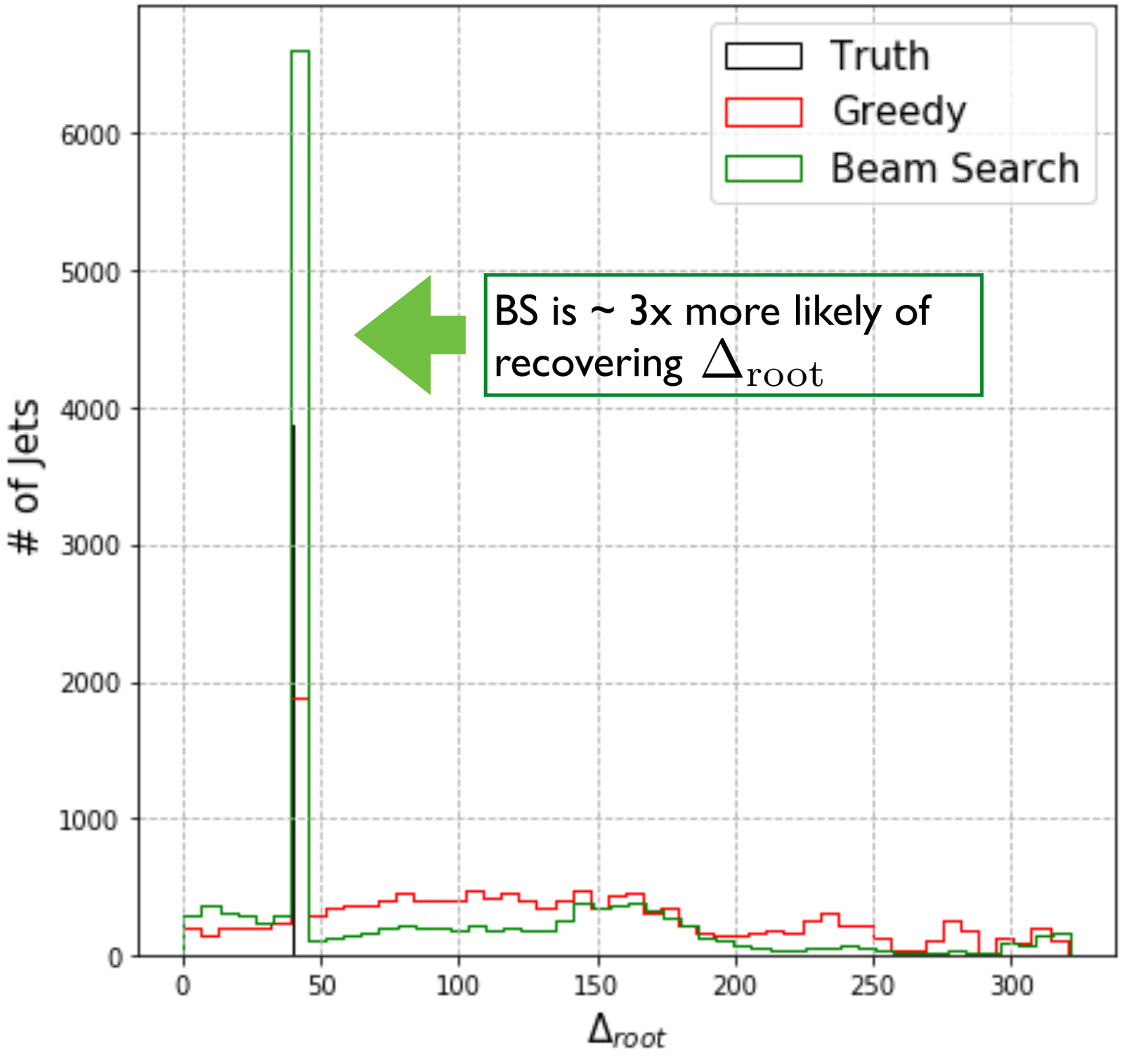
Mean values:

$$\log p_{\text{Greedy}} = -111.2 \pm 3.1$$
$$\log p_{\text{BS}} = -85.7 \pm 2.6$$
$$\log p_{\text{Truth}} = -76.8 \pm 2.5$$





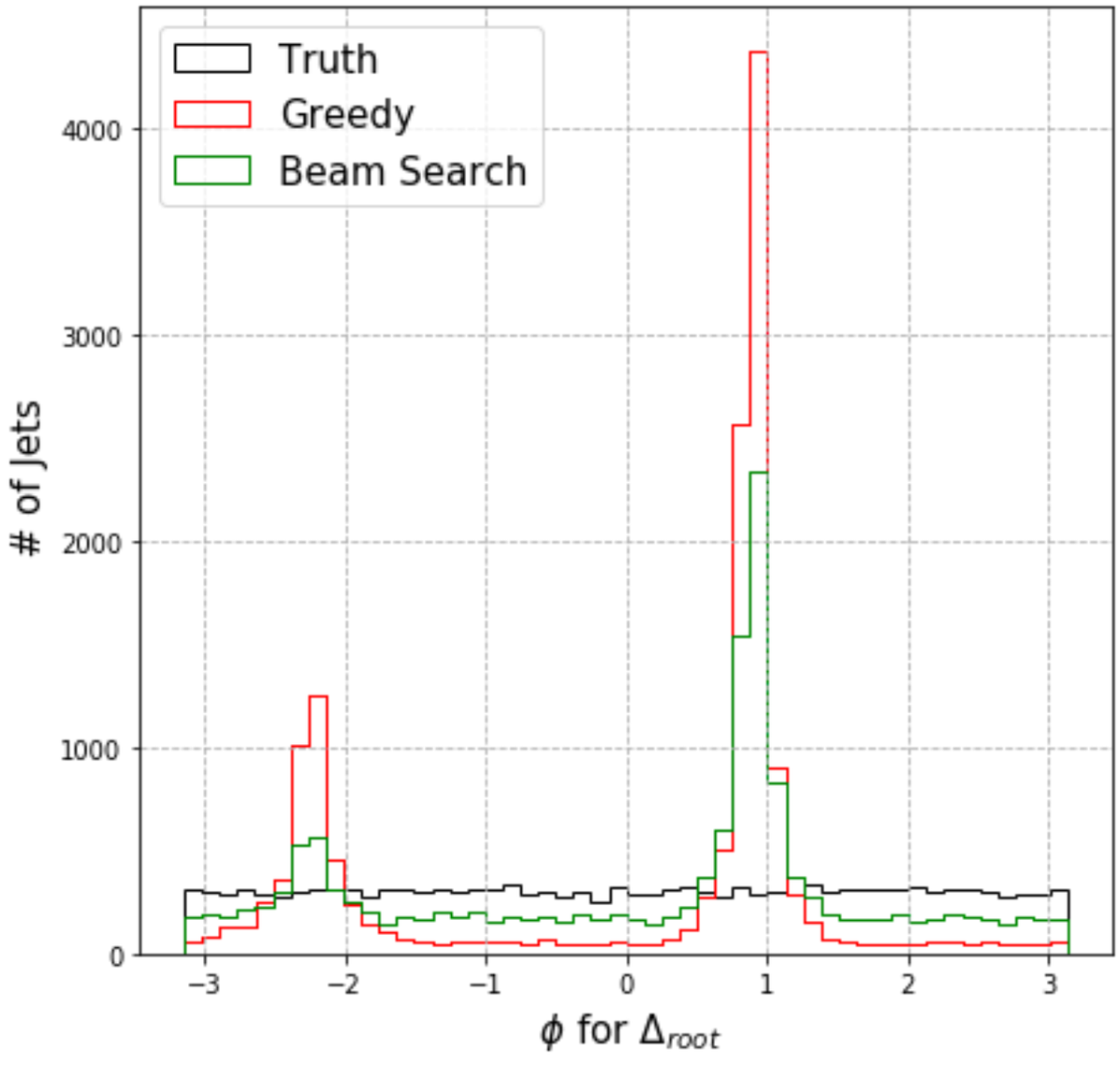
Reconstruction of Δ_{root}

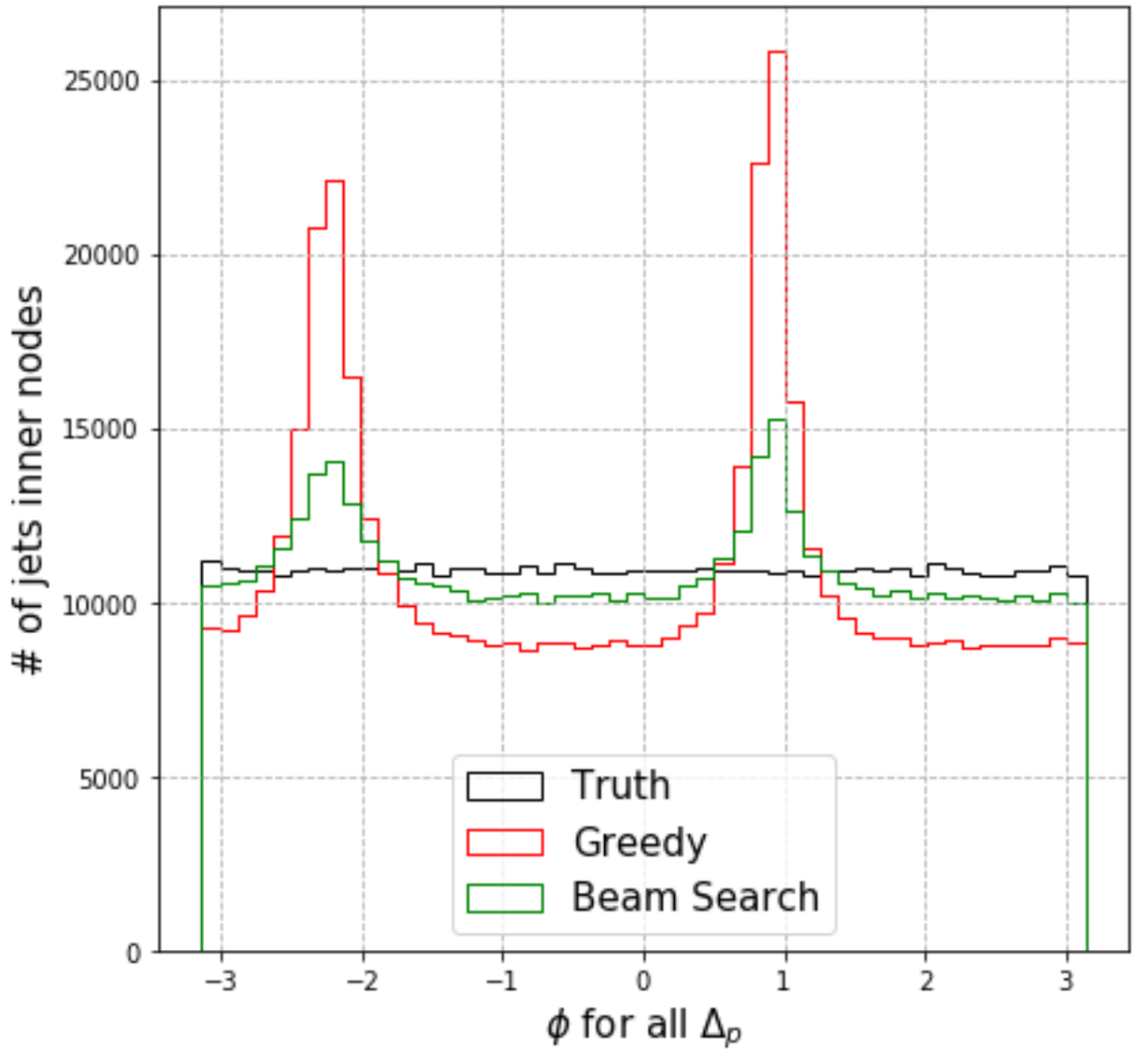


$$\vec{p}_{\mathrm{L}} = \frac{1}{2}\vec{p}_{\mathrm{p}} - \vec{\Delta}_{\mathrm{p}}$$

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

ϕ angle


Truth
Greedy
Beam Search
# of Jets
ϕ for Δ_{root}


Truth
Greedy
Beam Search
# of jets inner nodes
ϕ for all Δ_p

Final remarks

- Goal: Unification of generation and inference.
  - Study the implementation of this technique on full physics simulators.

- Could we learn the node splitting from fitting to data and systematically improve the generative model?

- Study metrics to compare jet trees latent structures.

- Beam search is strictly better than the greedy algorithm (e.g. kt) in terms of estimating the most likely tree.

- Explore other possibilities for jet clustering algorithms and compare their performance.

Thanks for your attention!