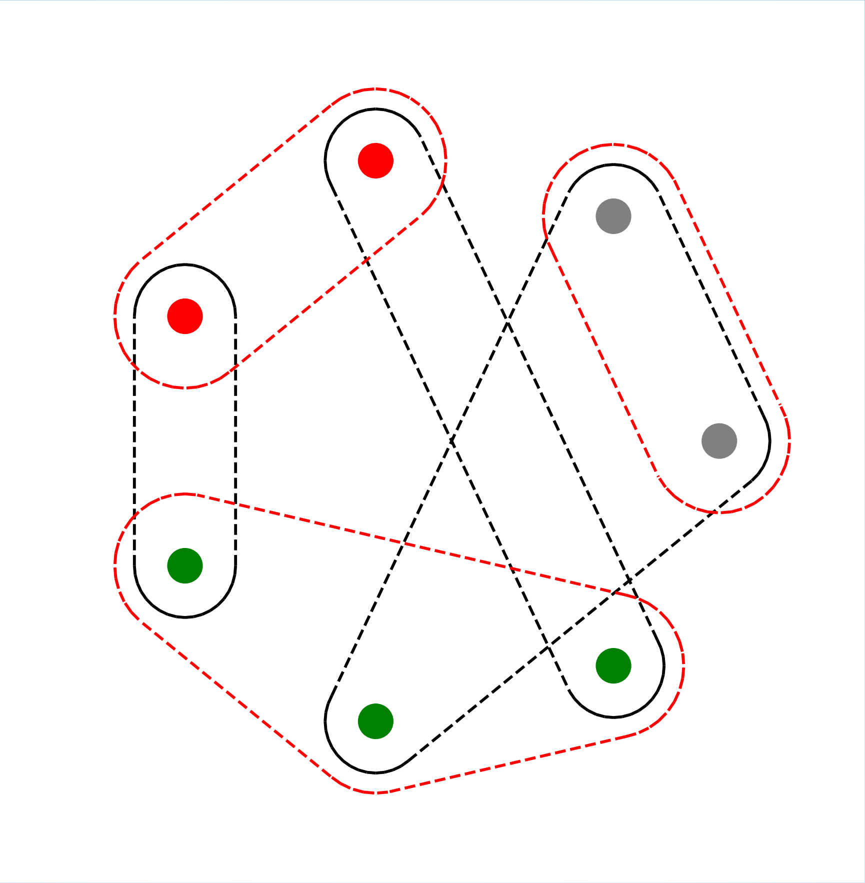
Graph networks for vertex reconstruction



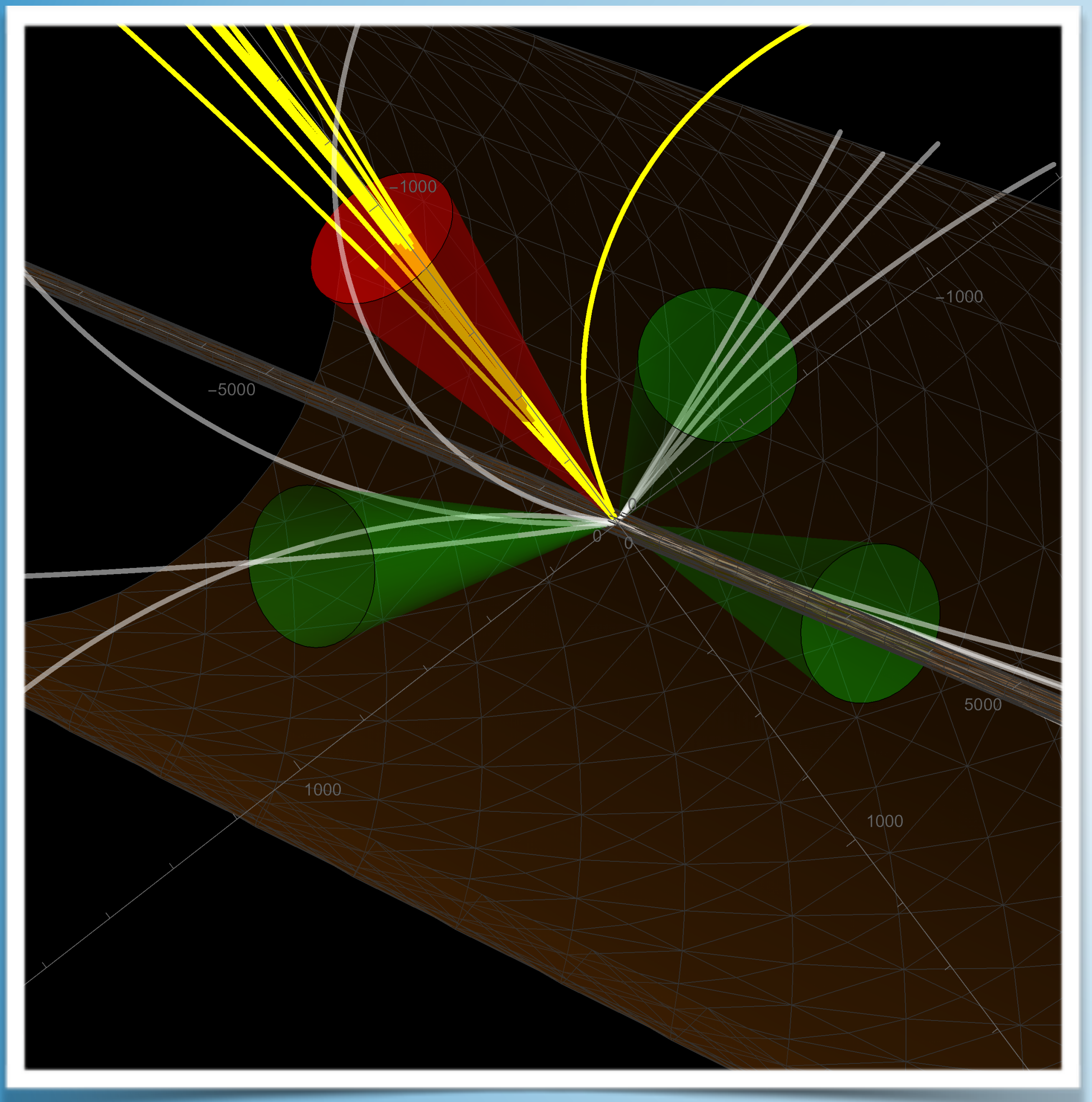
Jonathan Shlomi, Sanmay Ganguly, Eilam Gross
In collaboration with Kyle Cranmer/NYU

Learning to Discover : Advanced Pattern Recognition Institut Pascal, Orsay

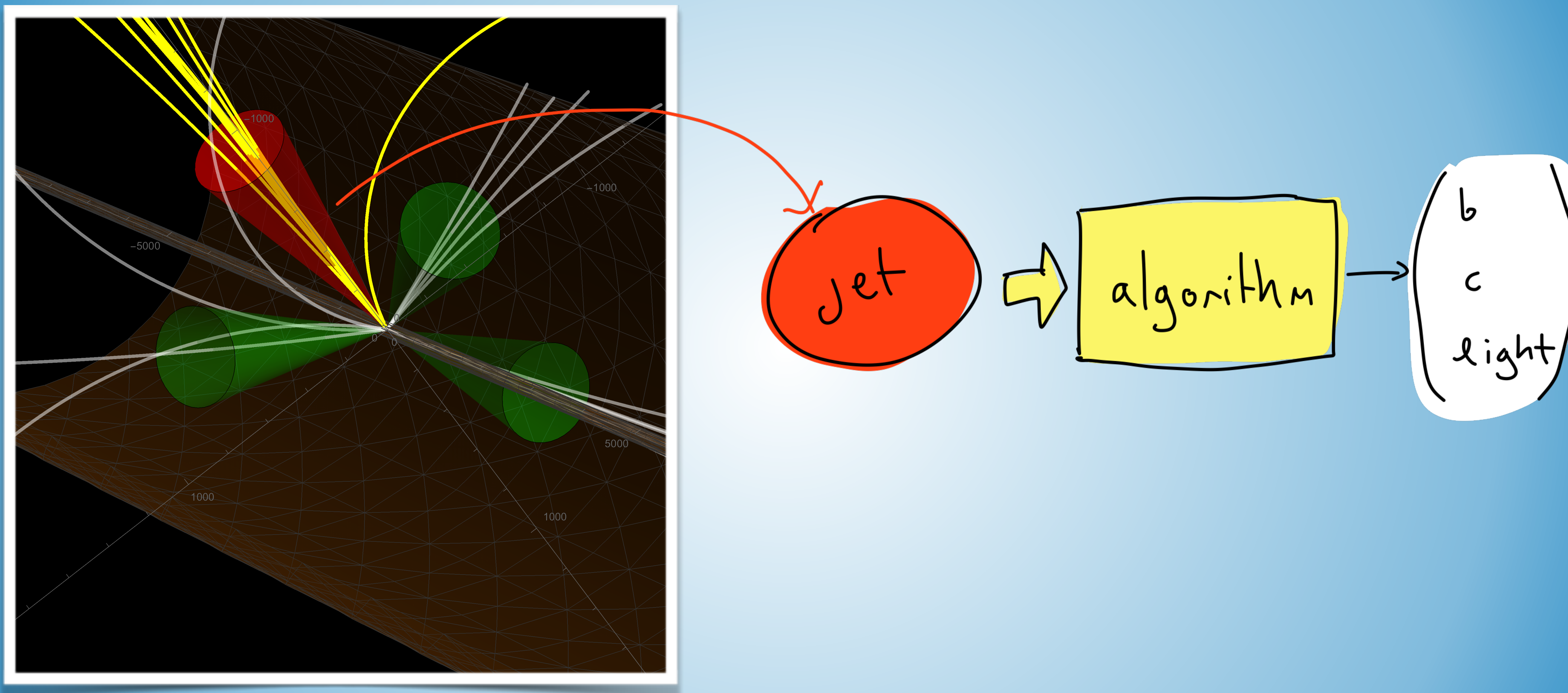
24 October 2019

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE
NYU
ATLAS EXPERIMENT

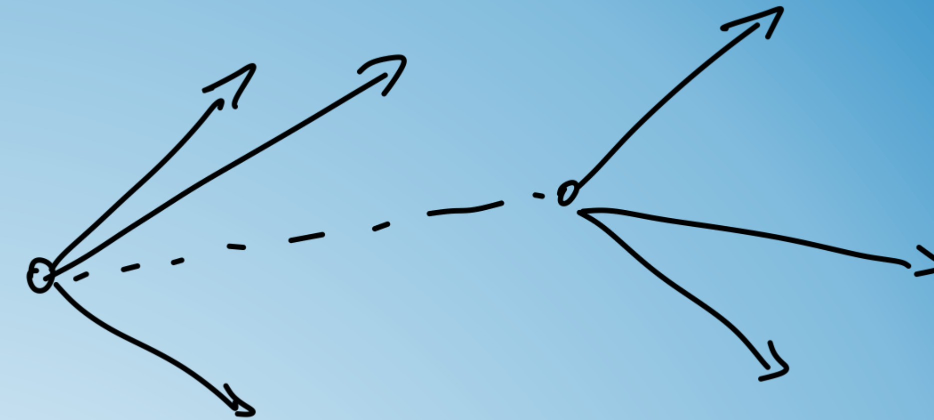
Jet Flavour tagging

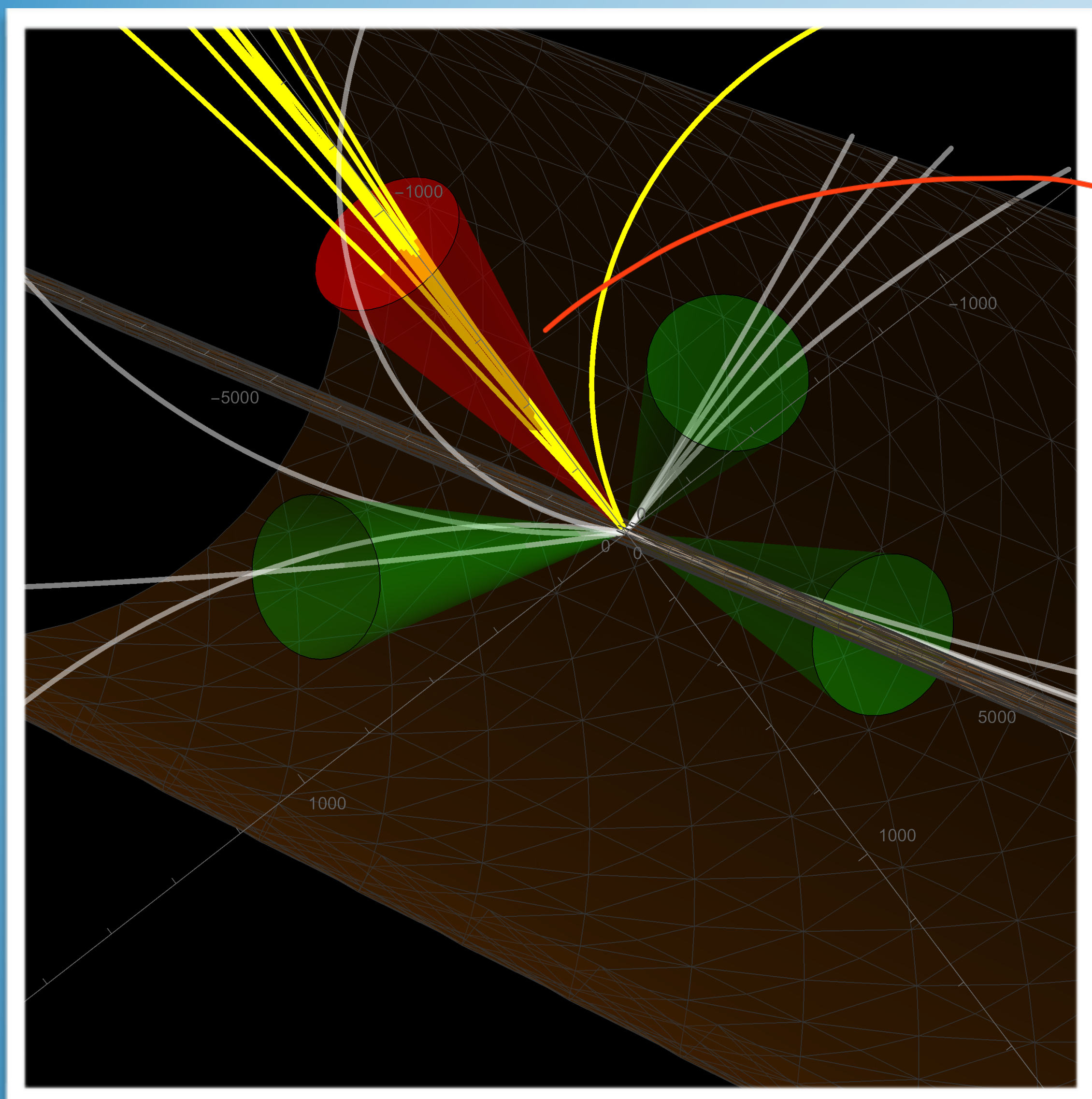


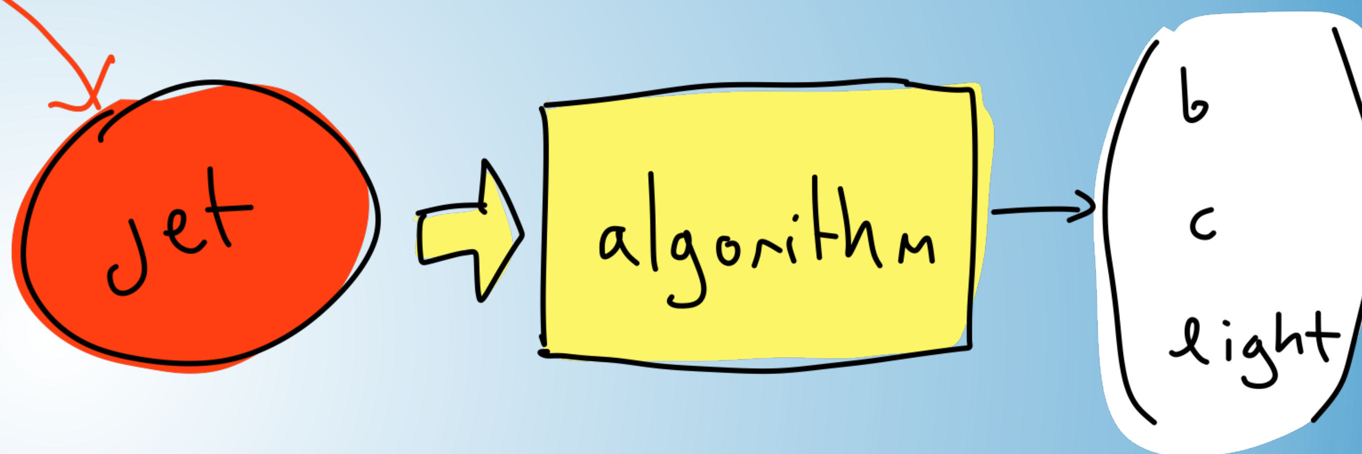
Jet Flavour tagging

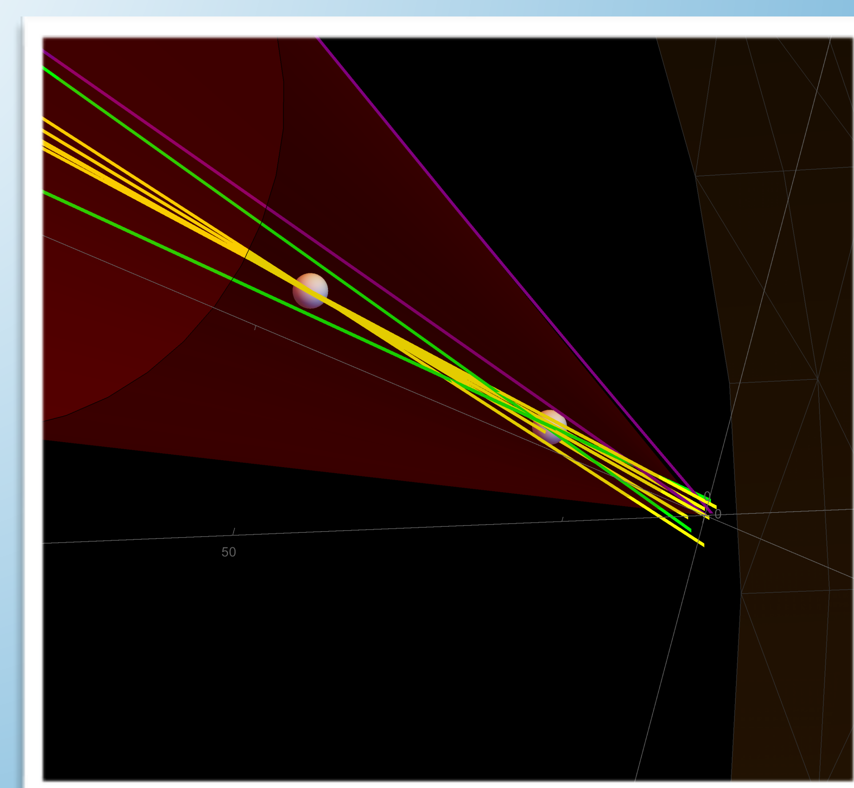


Jet Flavour tagging









Data and
Task Definition

Algorithms

Performance
Evaluation
+
Event Display

Data and
Task Definition

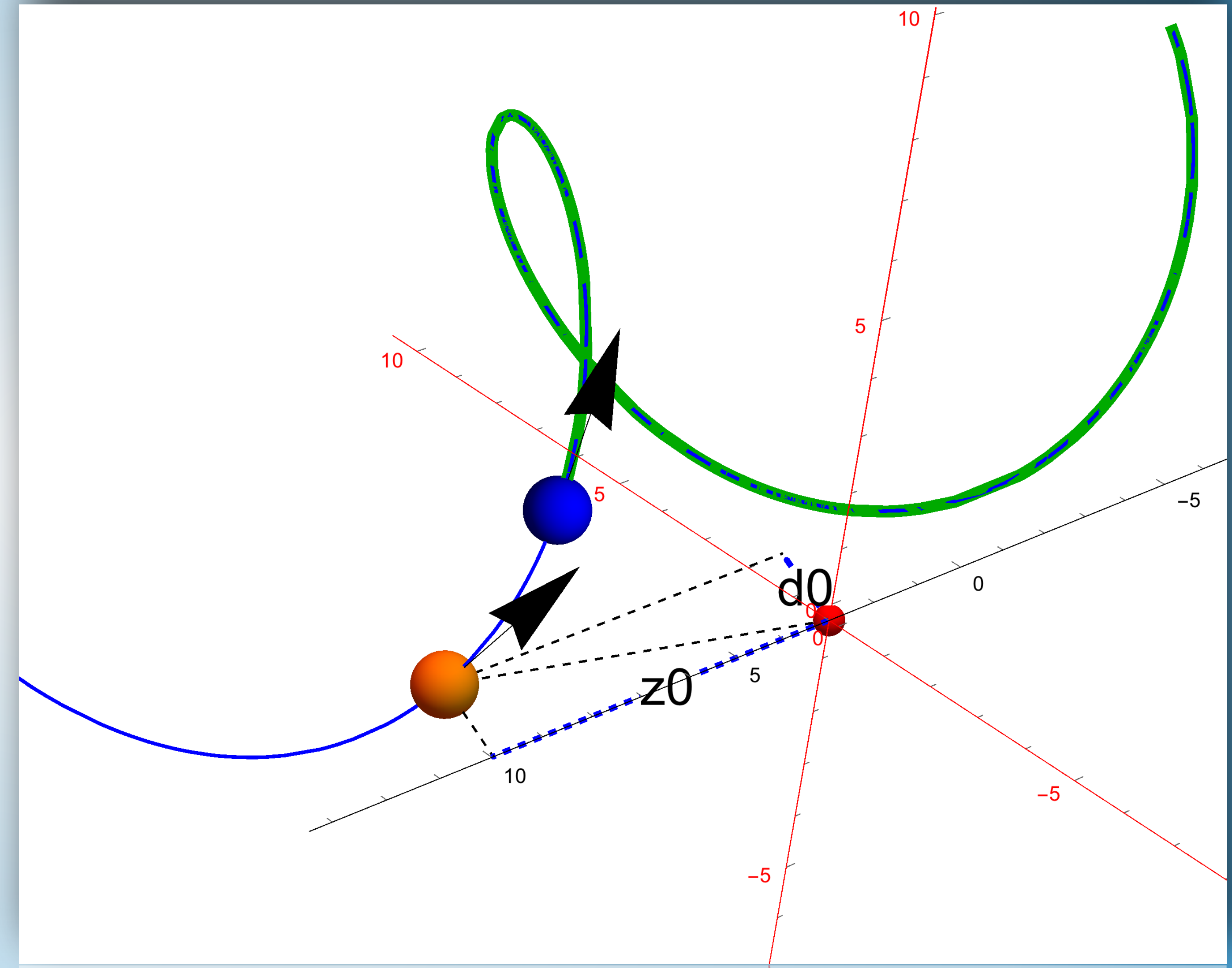
Algorithms

Performance
Evaluation
+
Event Display

- What kind of data are we talking about
- What is the task?
- Motivation
- Why is it an interesting task?

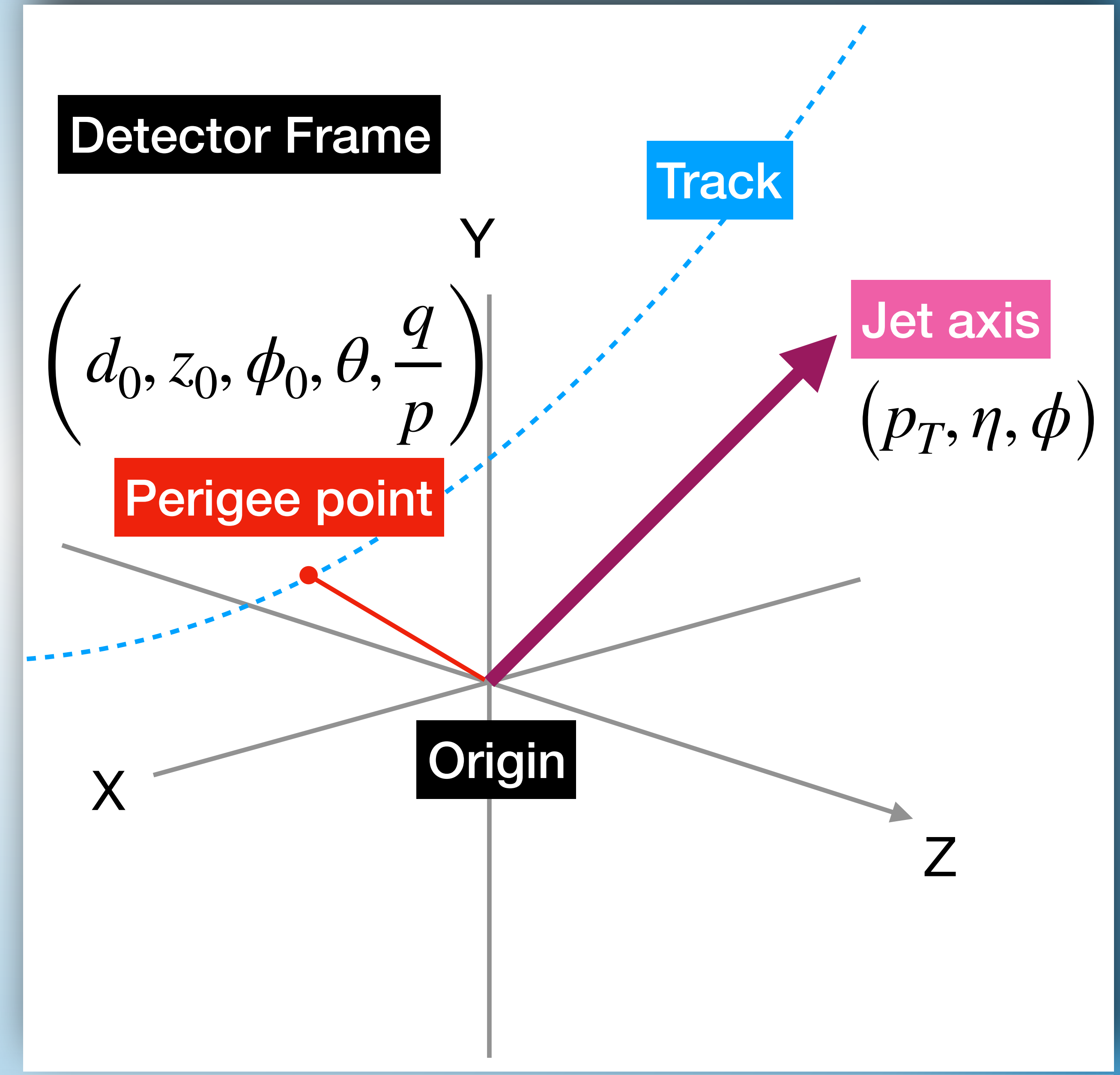
The Dataset

- A set of tracks
  (perigee parameters + cov matrix)



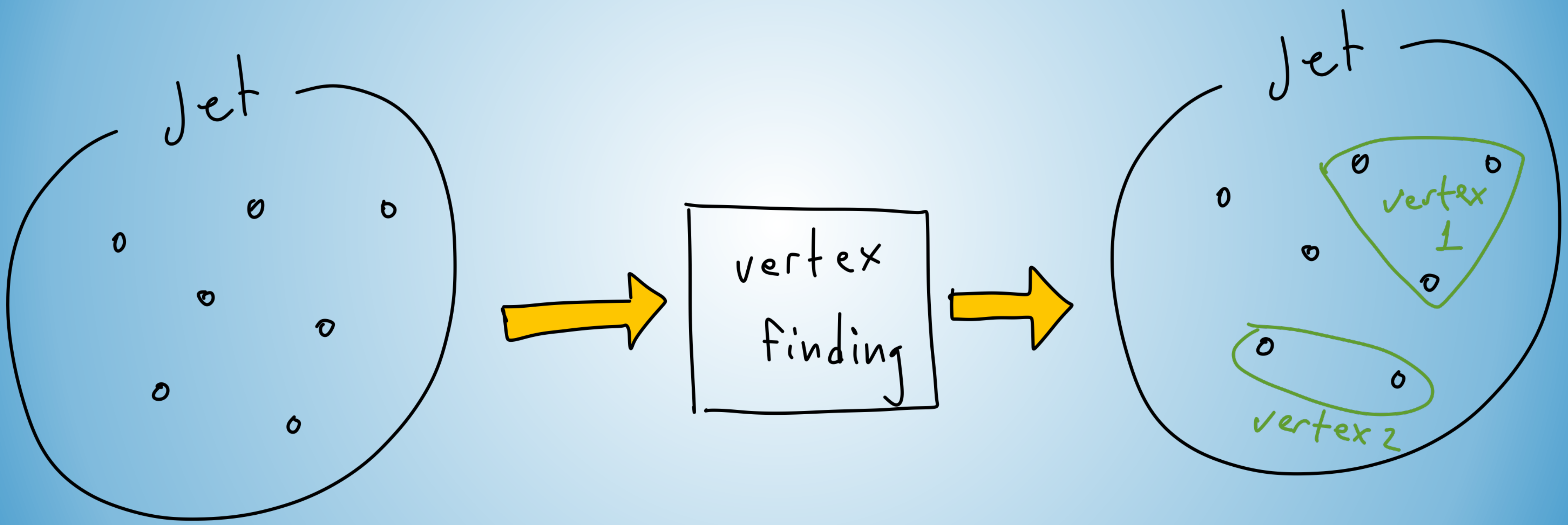
The Dataset

- A set of tracks
  (perigee parameters + cov matrix)
- Jet direction
- Jet momentum



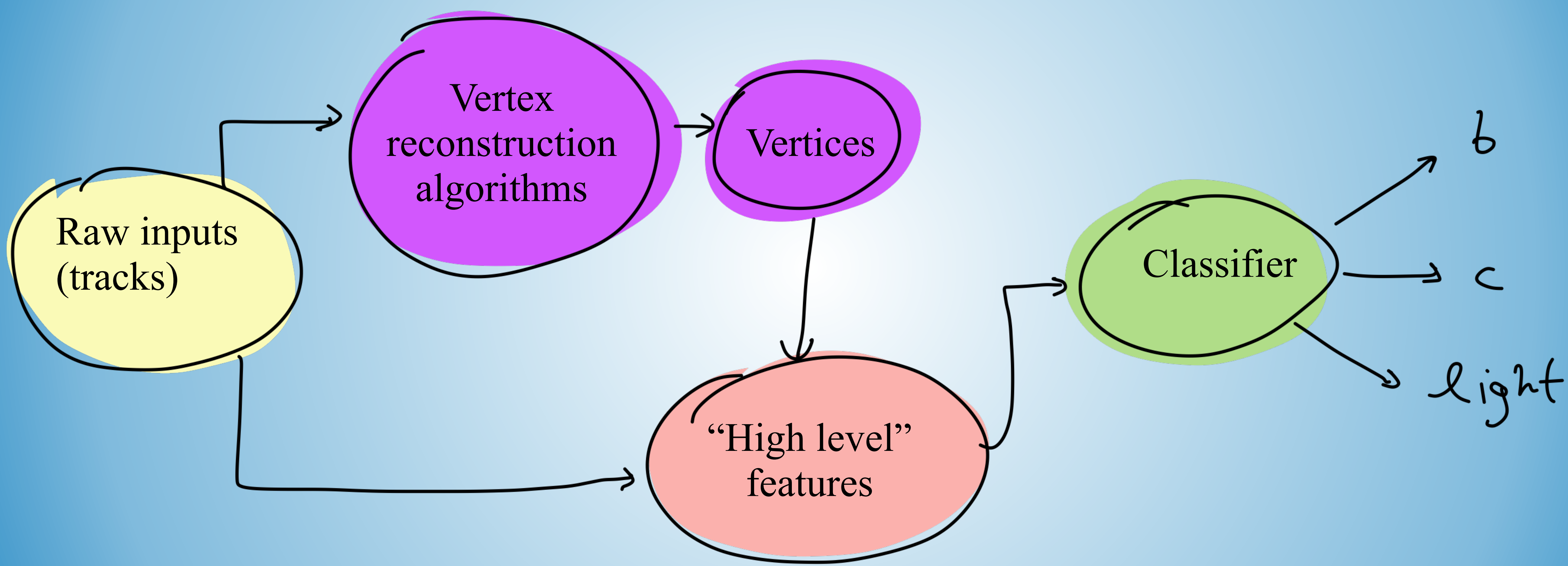
The Task

- Partition



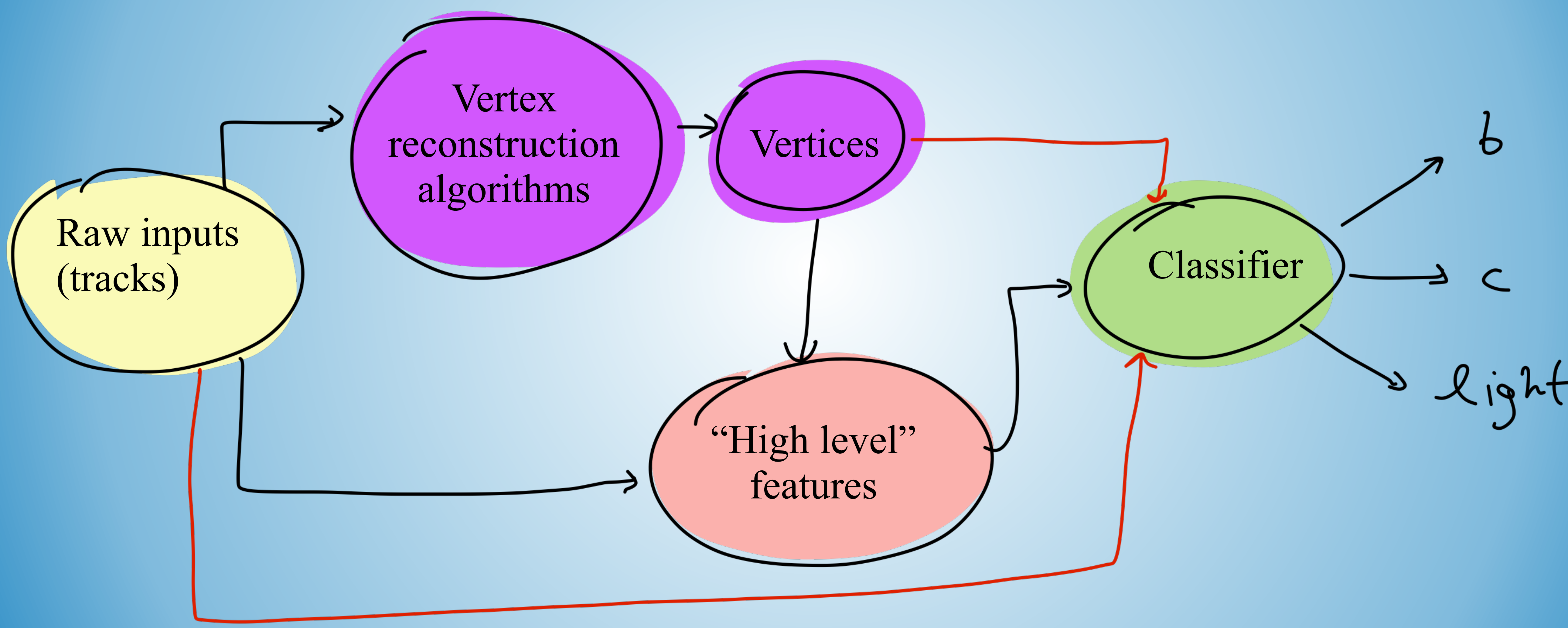
The Task

What we know so far about ML for flavour tagging



The Task

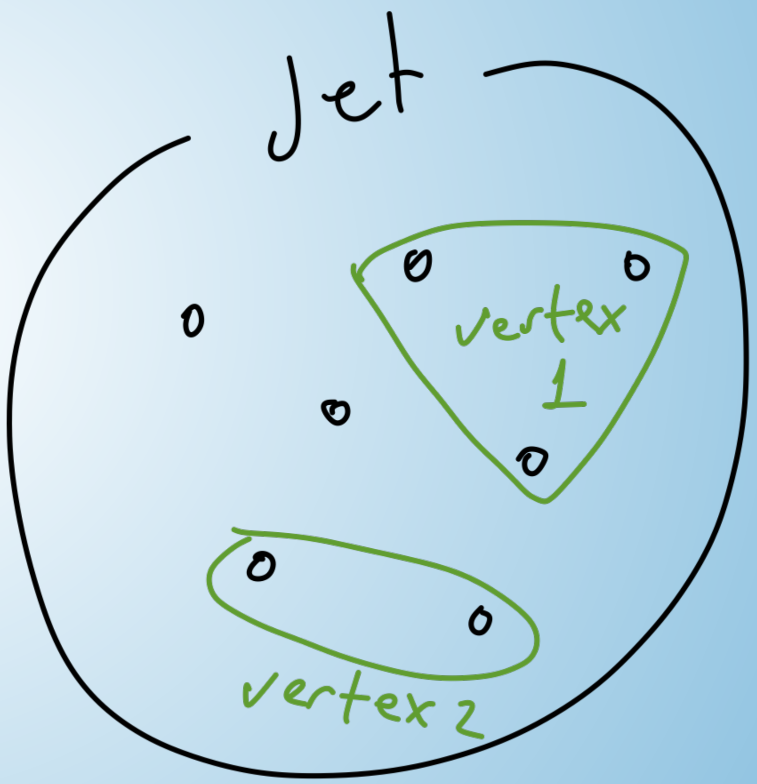
Adding the “low” and “medium” level features to the neural network classifier is helping the classification performance



Motivation - zooming in on the vertex finding

- End-to-End solution
  Existing algorithms require multiple stages
  and manual fine-tuning

- Context Sensitive Vertex finding
  can we take into consideration
  factors other than "geometry"

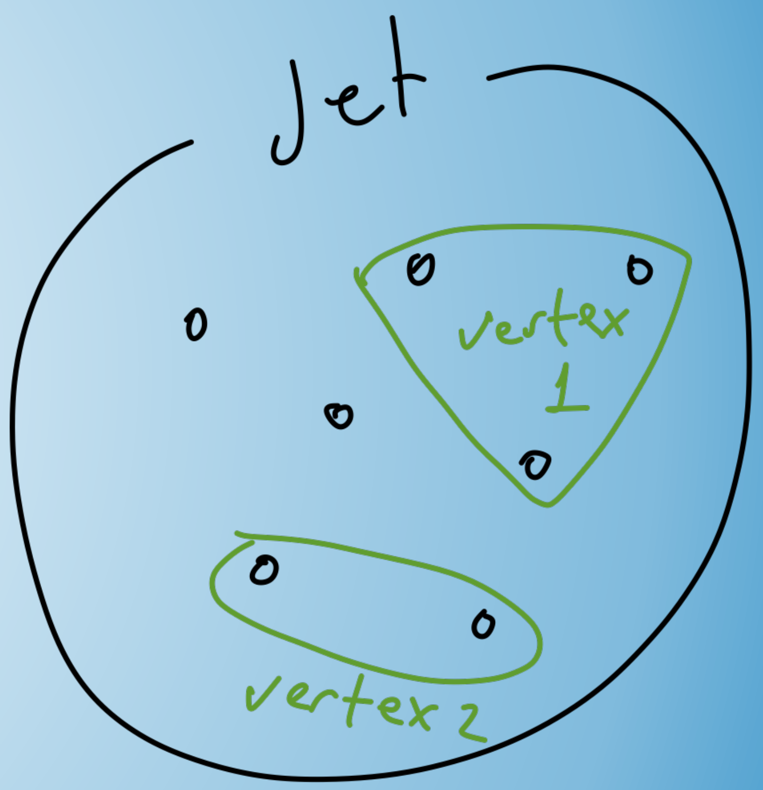


Why is it an interesting task?

- **Easy.**
  small sets, easy to explore, easy to compare solutions

- **Not straight-forward Classification**
  It is clustering, but fully supervised clustering (more exotic)

- **Room for improvement**



Motivation - better charm tagging

Out of the three classes (b, c, light) - charm is the hardest to correctly identify
If we have a better picture of the underlying decay, we can identify them more effectively

b jets

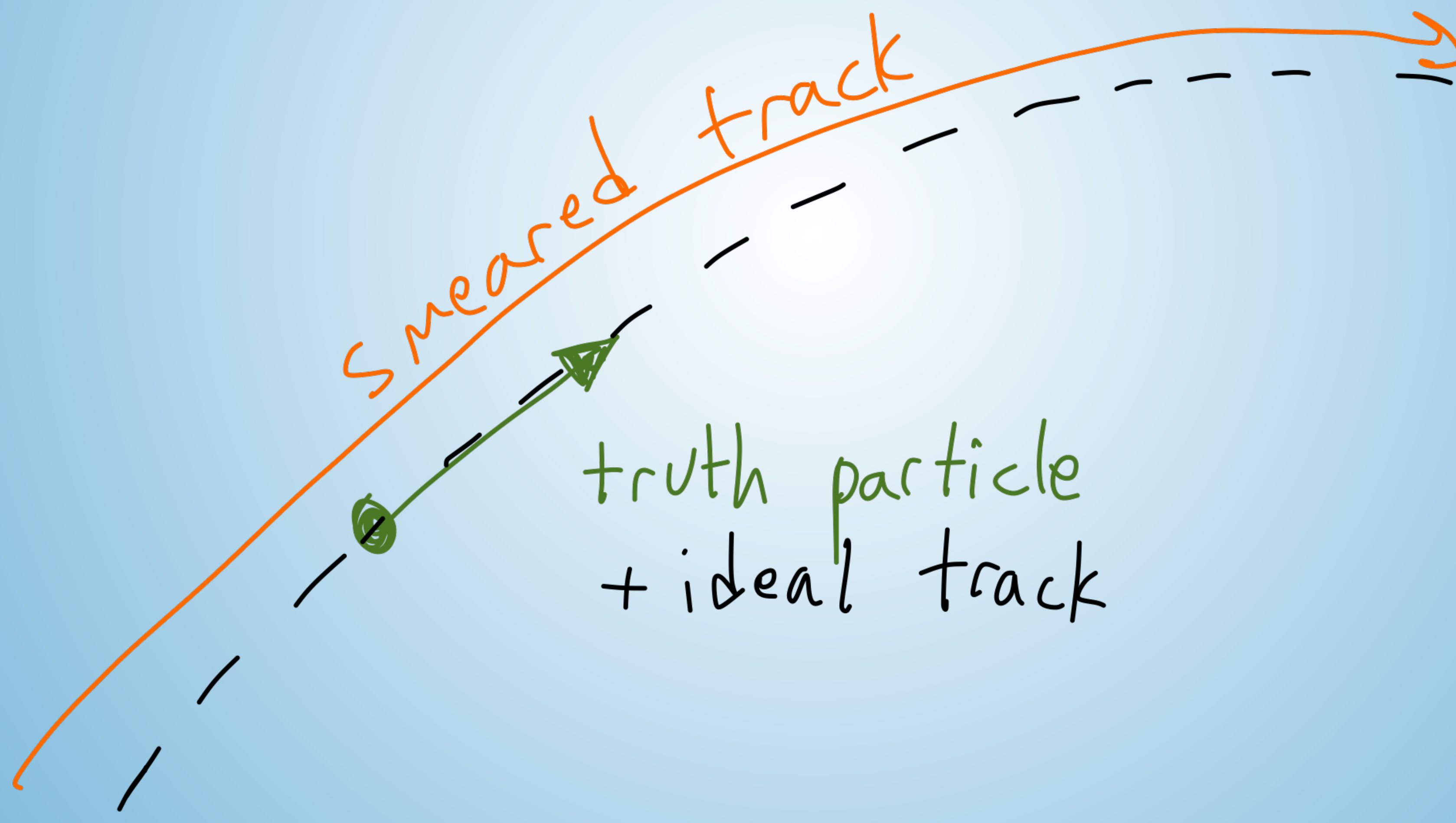
~80% efficiency

c jets

~40% efficiency

The Dataset

This dataset is generated with a "fast simulation", meaning the truth particles are "smeared"
Ideally we would need a "full simulation" but for the purposes of trying different algorithms this is sufficient.



Data and
Task Definition

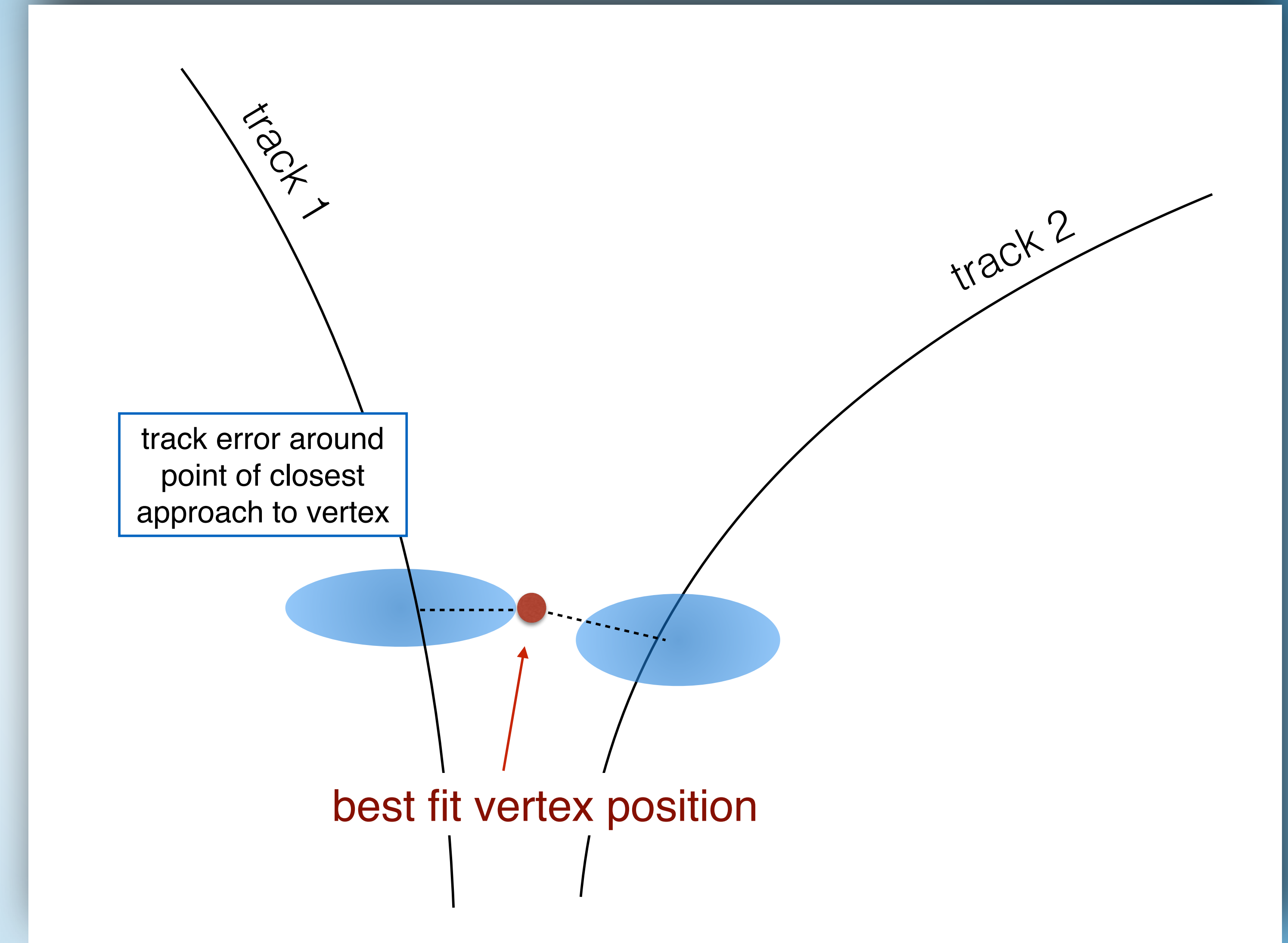
Algorithms

- the "baseline" algorithm
- How to describe task for neural network?
- What kind of architectures we can use

Performance
Evaluation
+
Event Display

the "baseline" algorithm

Adaptive vertex finding/fitting



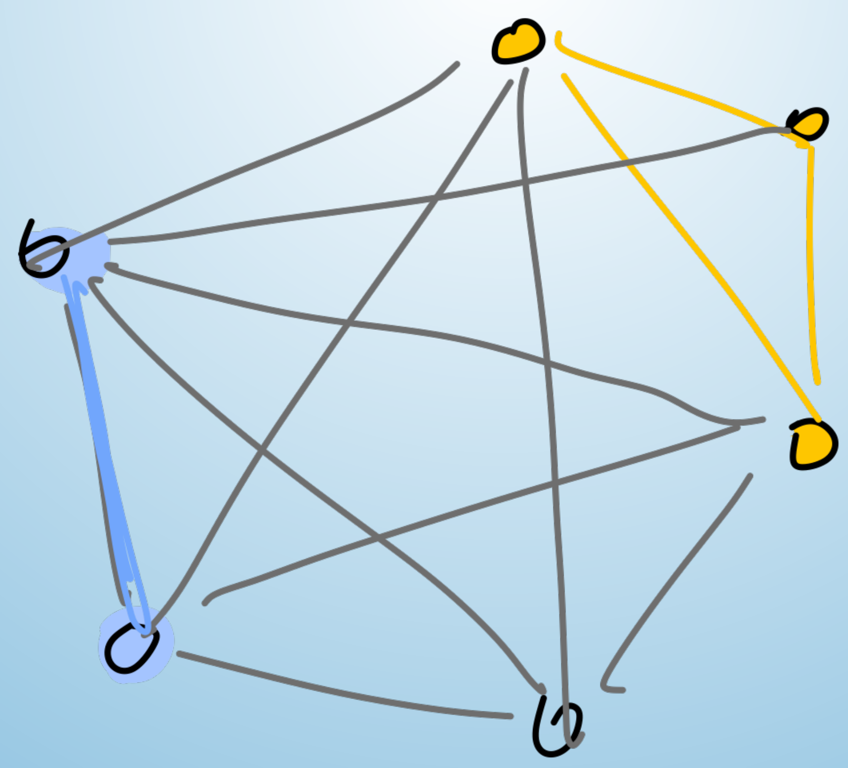
https://rave.hepforge.org/

W. Waltenberger."RAVE: A detector-independent toolkit to reconstruct vertices". In:IEEETrans. Nucl. Sci.58 (2011), pp. 434–444

How to describe the task for a neural network?

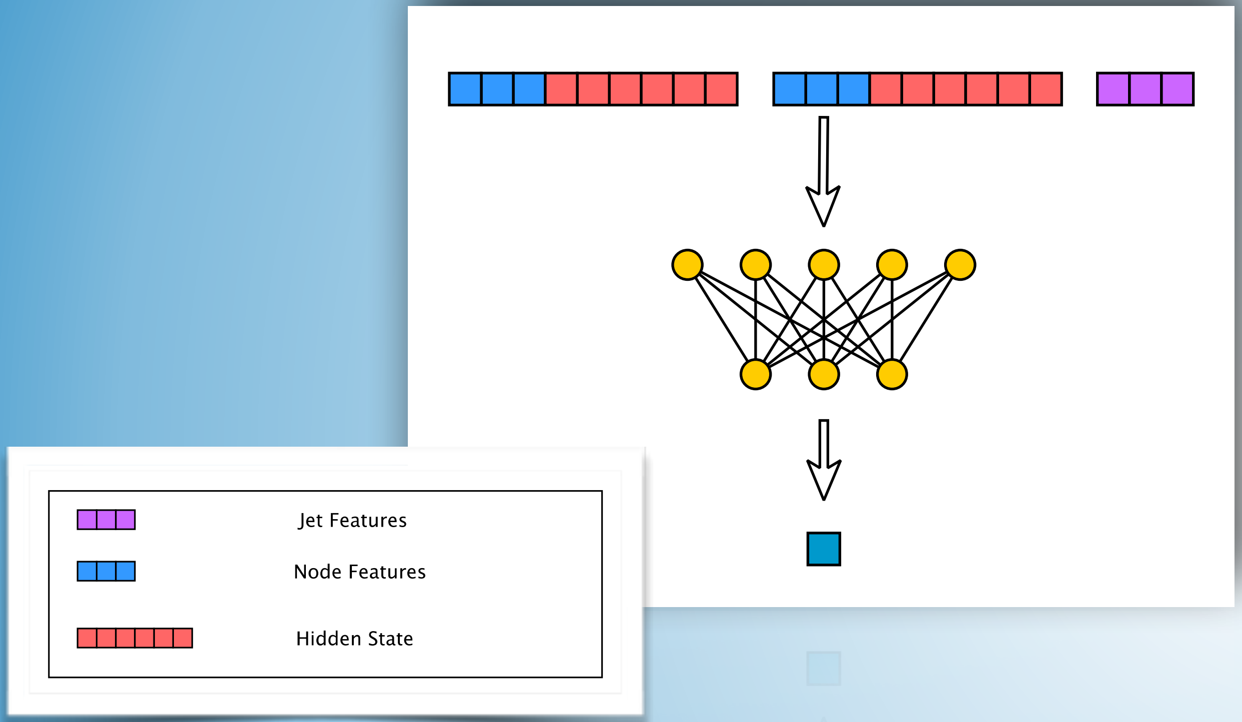
Take the set of tracks, and describe the tracks as nodes in a graph,
Fully connect the graph with edges between every node

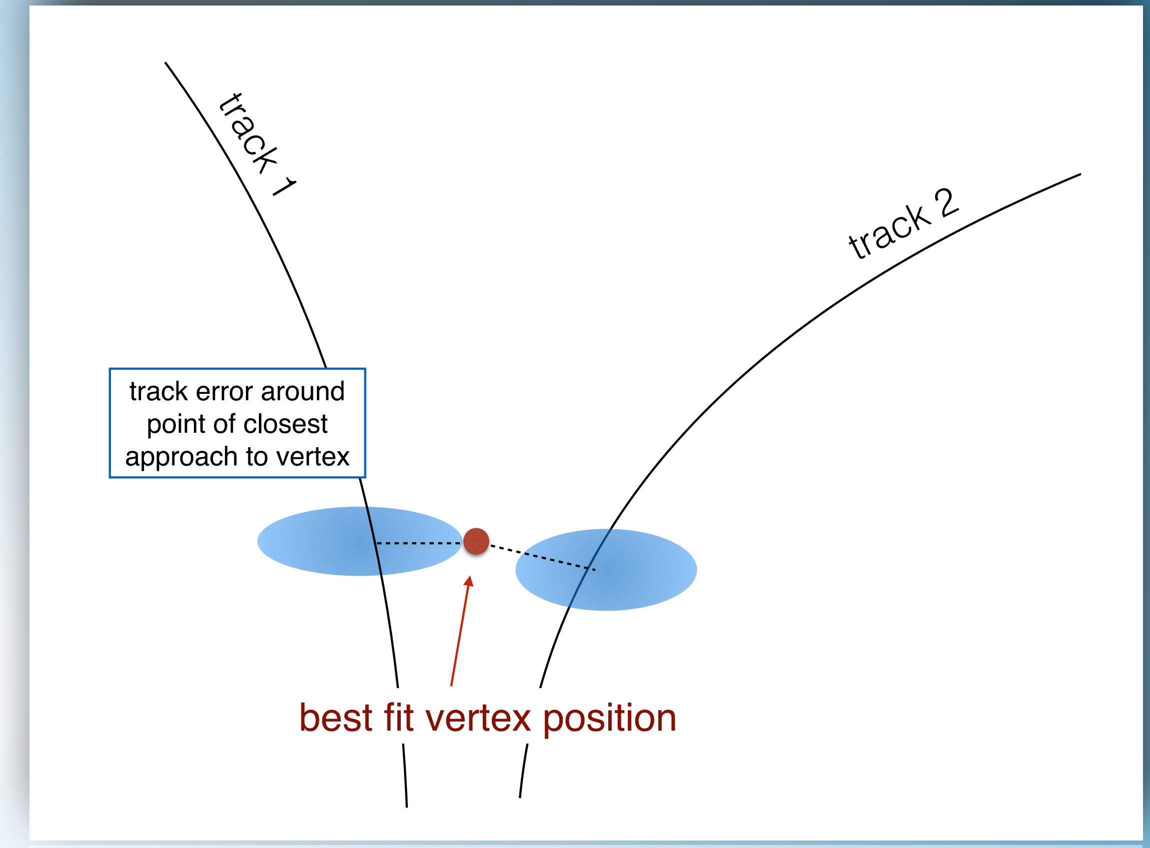
Now the task can be thought of as "edge classification"



the "baseline" neural network solution

Track pair classifier:



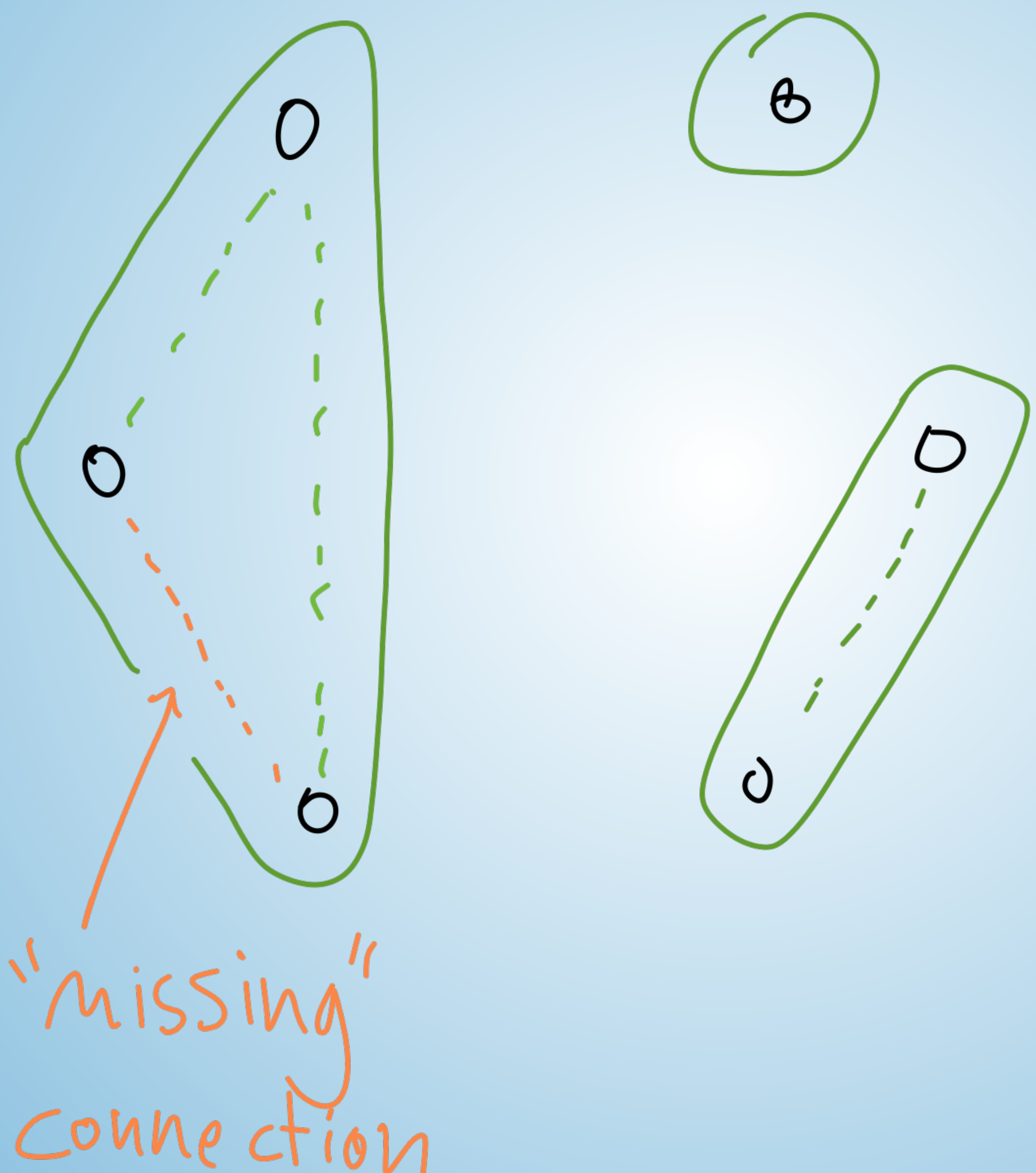


Binary classifier:

input: a pair of tracks (+jet 4 vector)

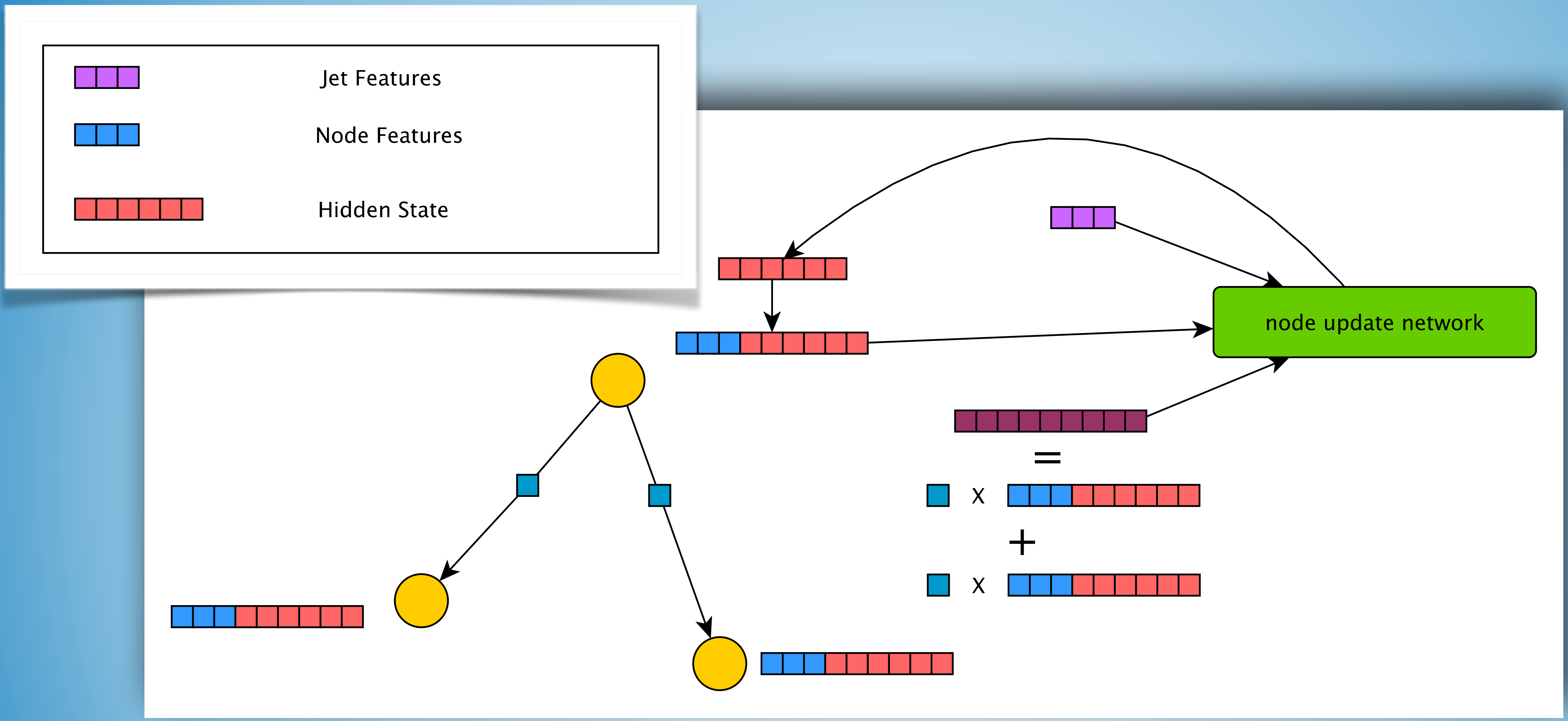
output: is this pair in the same vertex

Once the pairs are classified, join together clusters for tracks that have connecting edges



Fill in "missing connections"

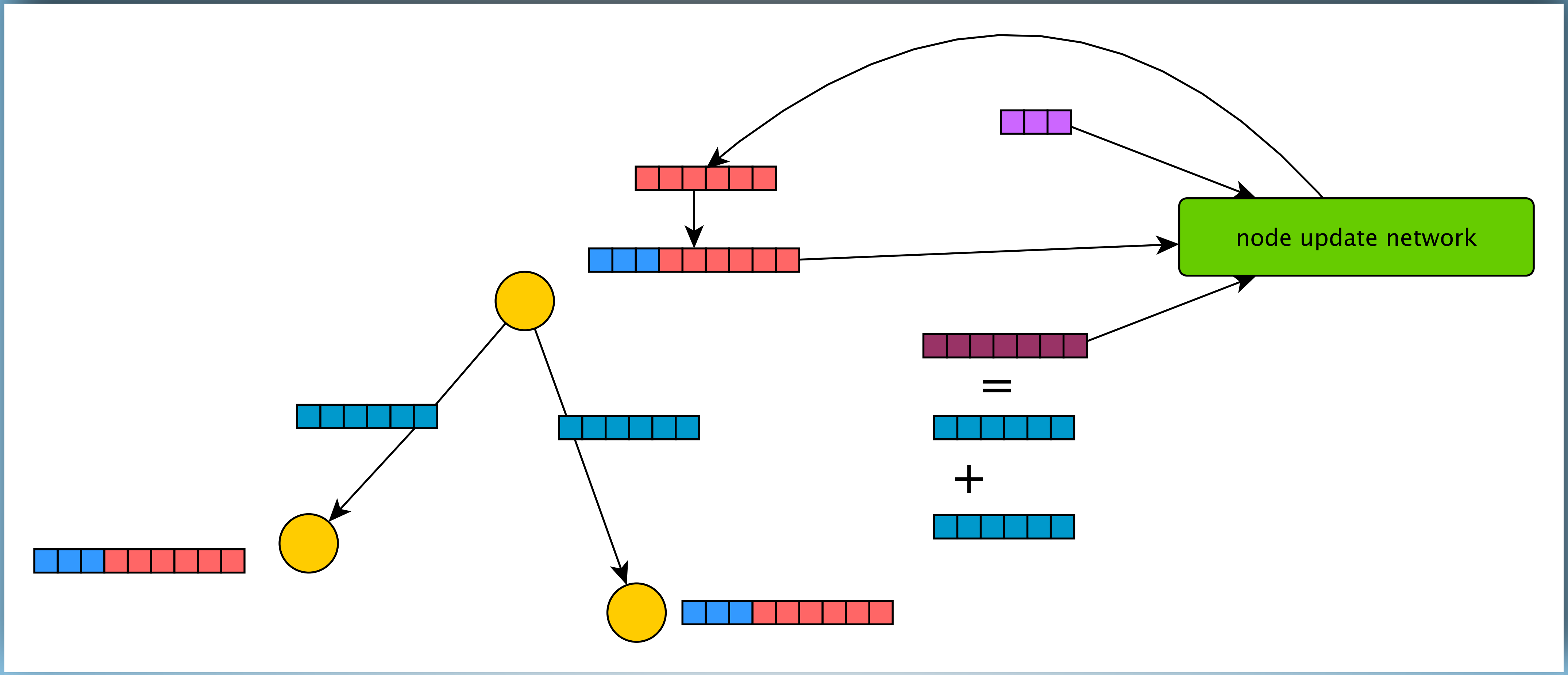
Message passing “graph neural network”



Each “node” (track) looks at a weighted sum of the rest of the nodes in the jet, and updates its latent representation.

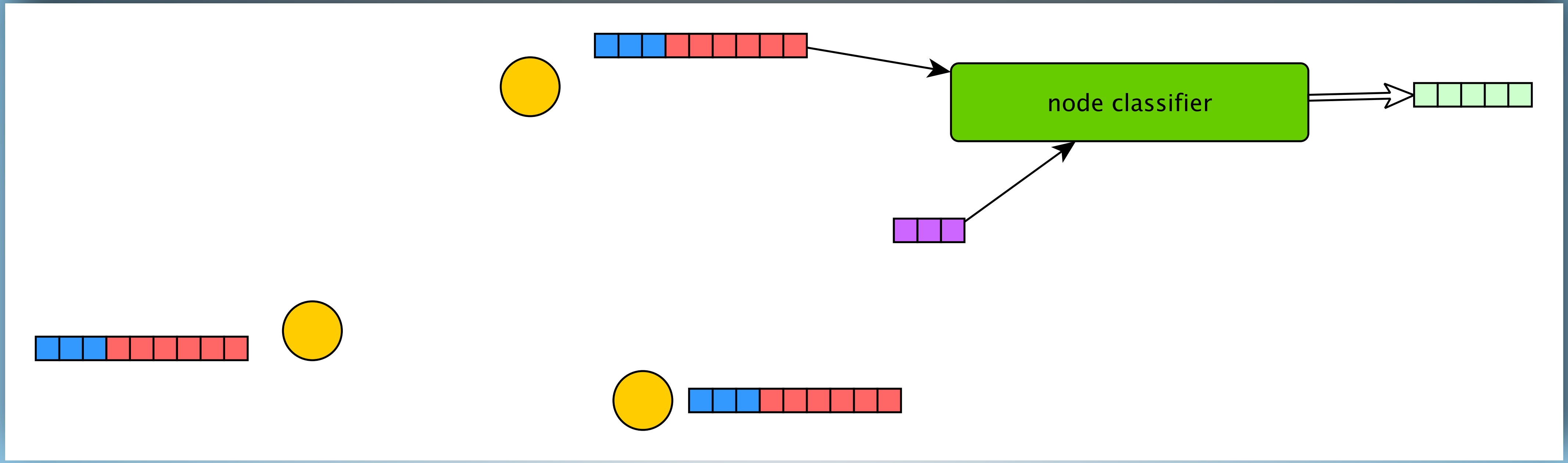
Then - apply same binary classifier as baseline solution

Message passing "graph neural network"
+ edge latent representation



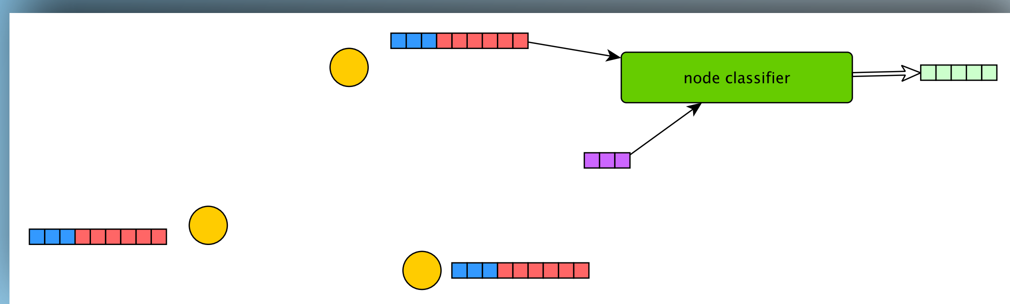
The weighted sum of node representations is replaced by a sum of edge latent representations

Message passing “graph neural network” + node classification



Instead of predicting a binary classification for the pairs of tracks,
Output a multi-class output for the individual nodes

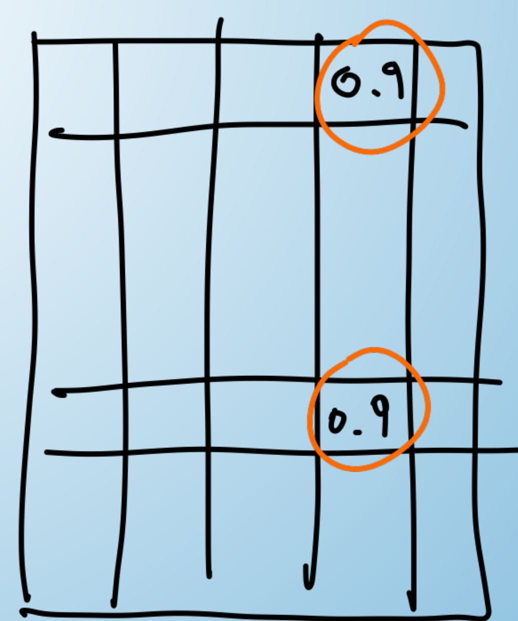
Message passing "graph neural network"
+ node classification



max number of vtx

trained for the target vtx assignment

n nodes



[0, 0, 0, 1, 1, 2, 2, 2]

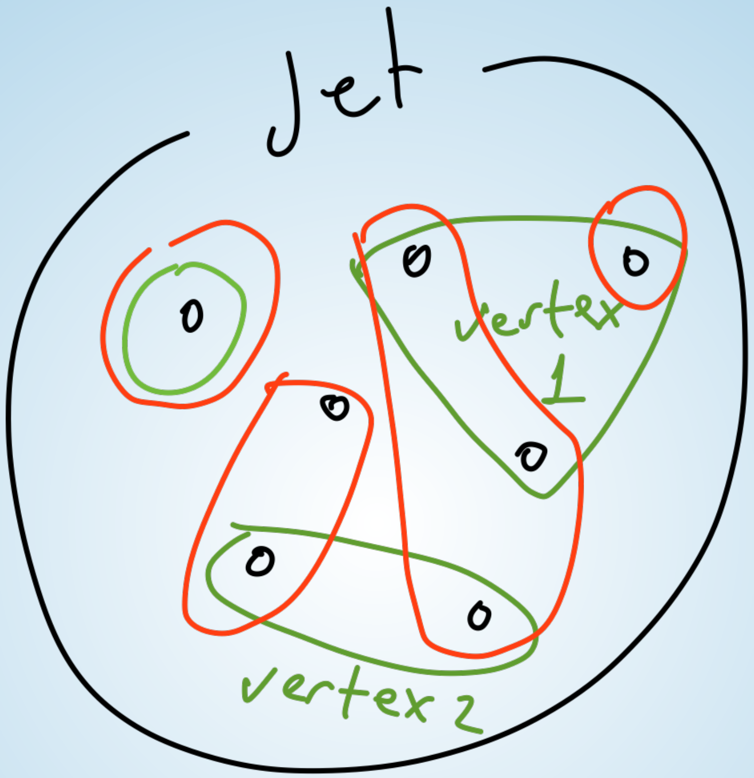
Data and
Task Definition

Algorithms

Performance
Evaluation
+
Event Display

- How to quantify vertex finding performance?
- What performance metrics do we care about?
- How to visualise the results

How to quantify vertex finding performance?

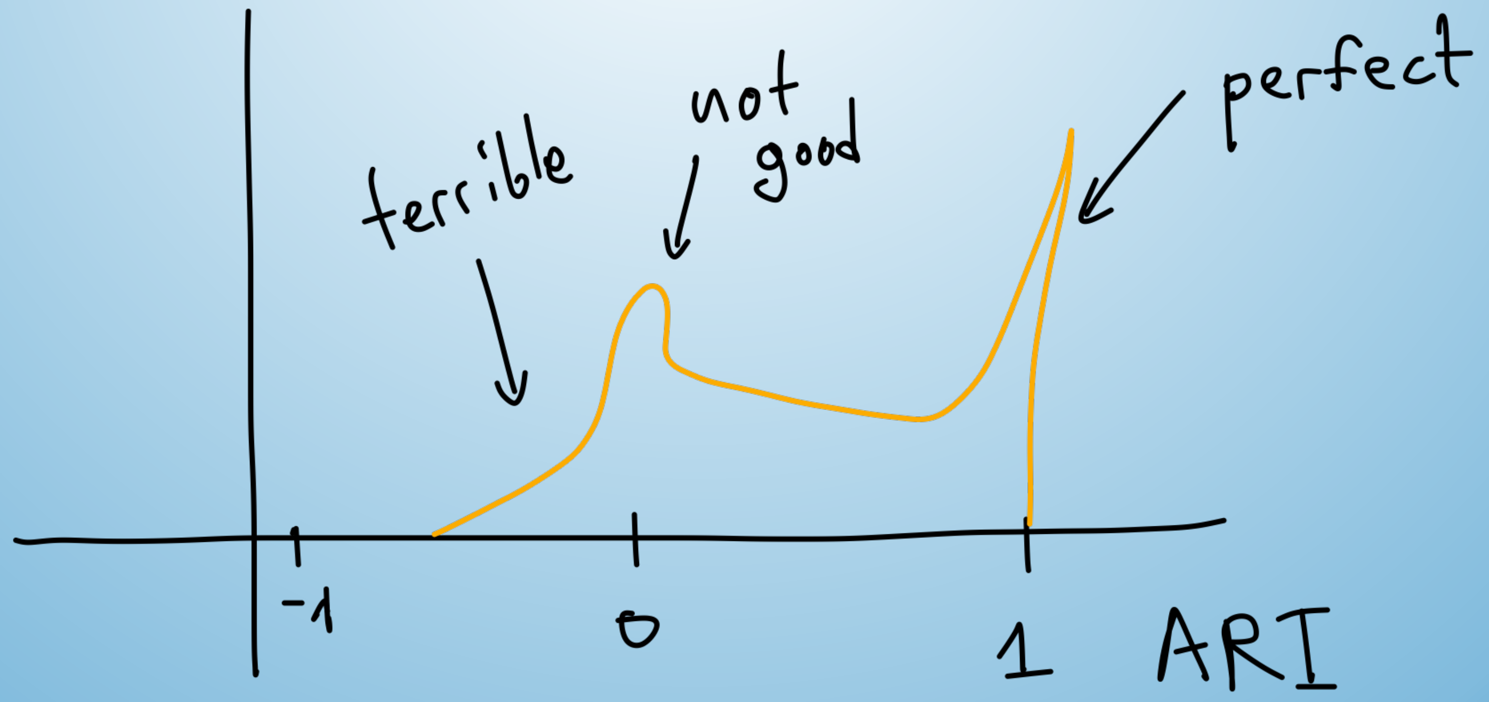


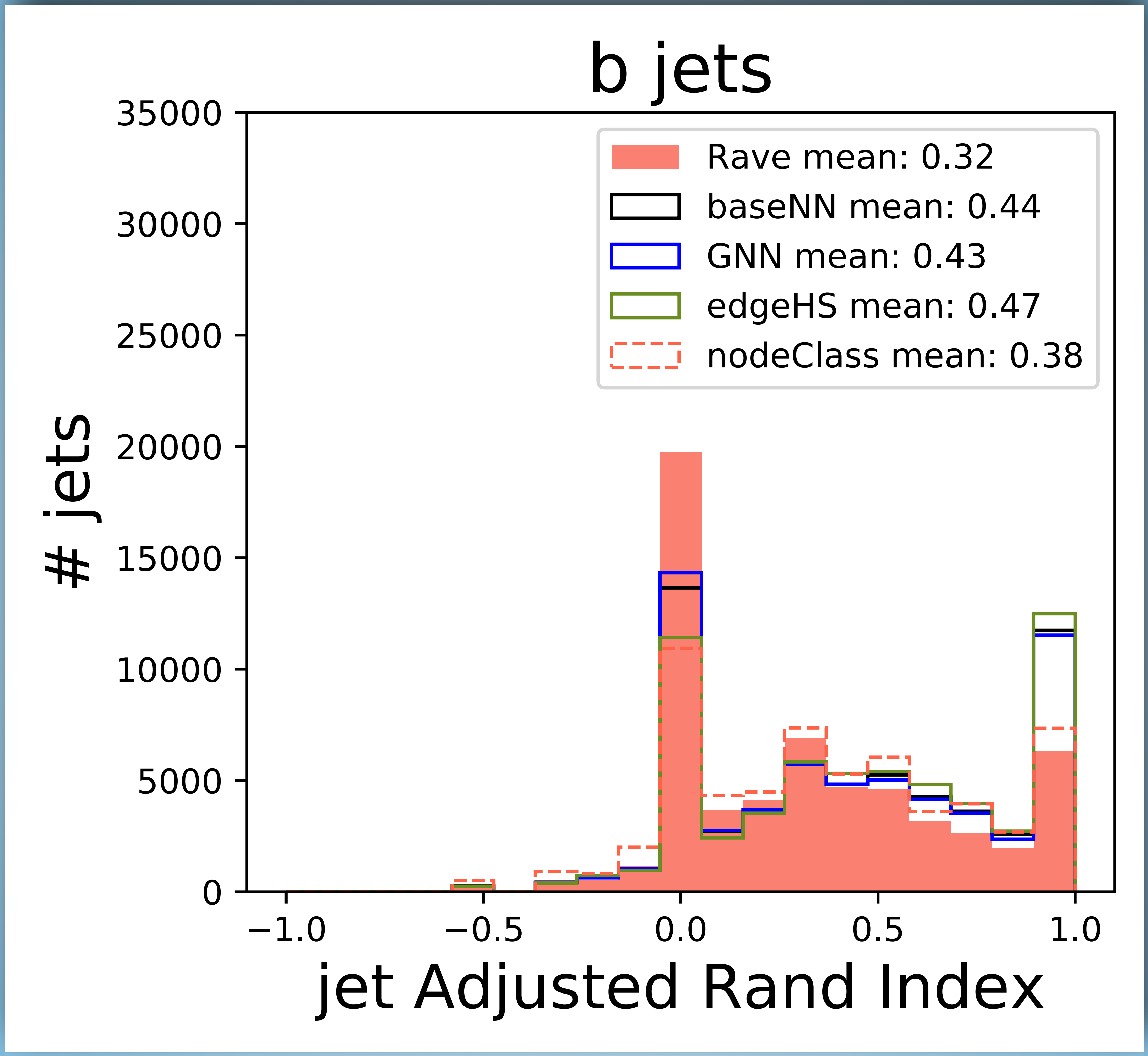
$$\text{Rand Index} = \frac{\#\ \text{correct pairs}}{\binom{n}{2}}$$

How to quantify vertex finding performance?

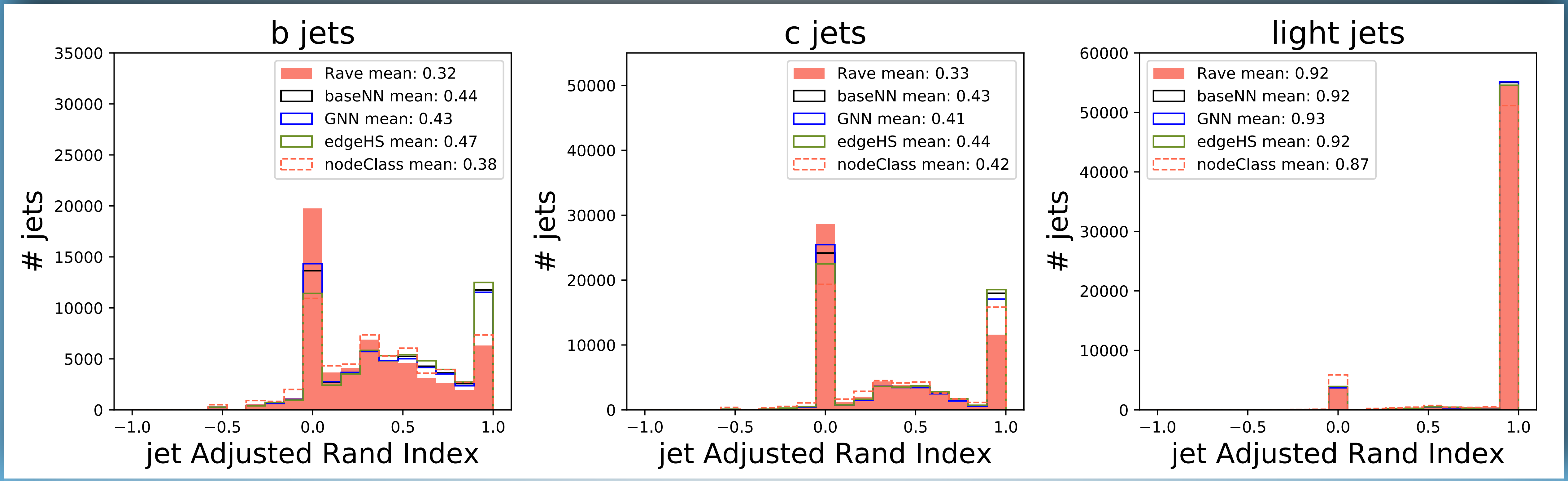
$$\text{Adjusted Rand Index} = \frac{\text{RI - E[RI]}}{\text{1 - E[RI]}}$$

$$\text{Adjusted Rand Index} = \begin{cases} 1 & \text{perfect} \\ 0 & \text{as good as random guessing} \\ < 0 & \text{worse then guessing} \end{cases}$$




b jets
Rave mean: 0.32
baseNN mean: 0.44
GNN mean: 0.43
edgeHS mean: 0.47
nodeClass mean: 0.38
jets
35000
30000
25000
20000
15000
10000
5000
0
−1.0
−0.5
0.0
0.5
1.0
jet Adjusted Rand Index

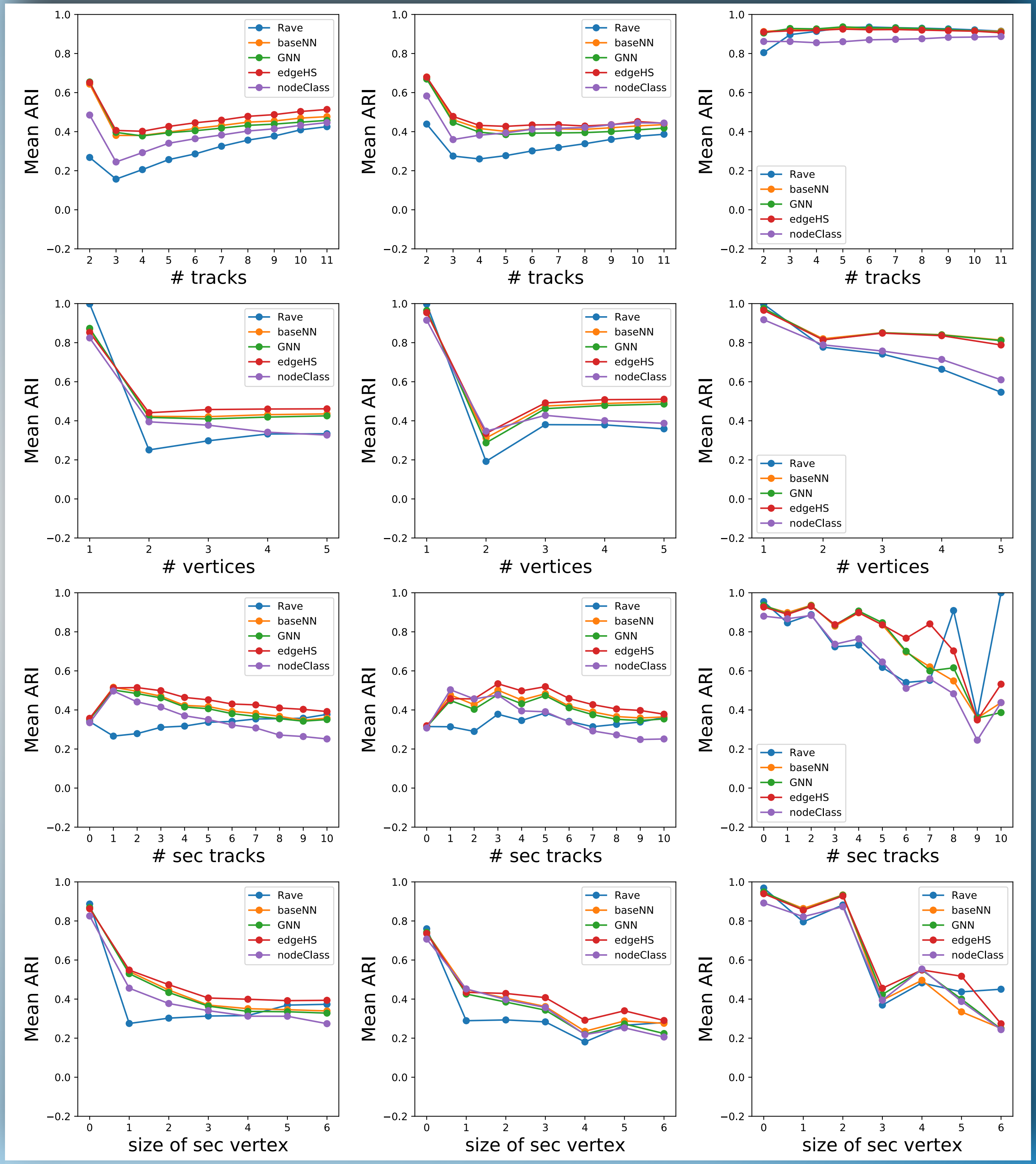
Slight (very slight) improvement for the neural networks over the geometric algorithm



Another small advantage to the network with edge latent representation

The same picture remains when we look at the mean ARI as a function of:

#tracks,
#vertices,
# of displaced tracks,
size of the secondary vertex,
…

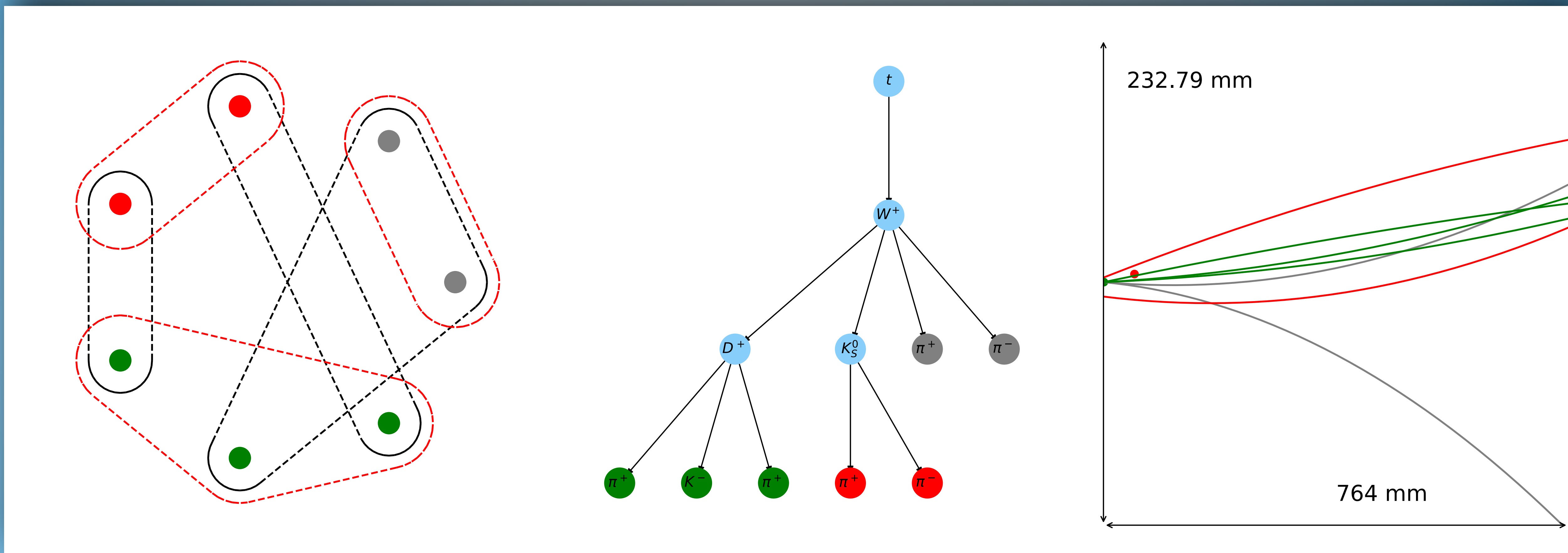


A really great way to understand the dataset is to visualise individual examples -

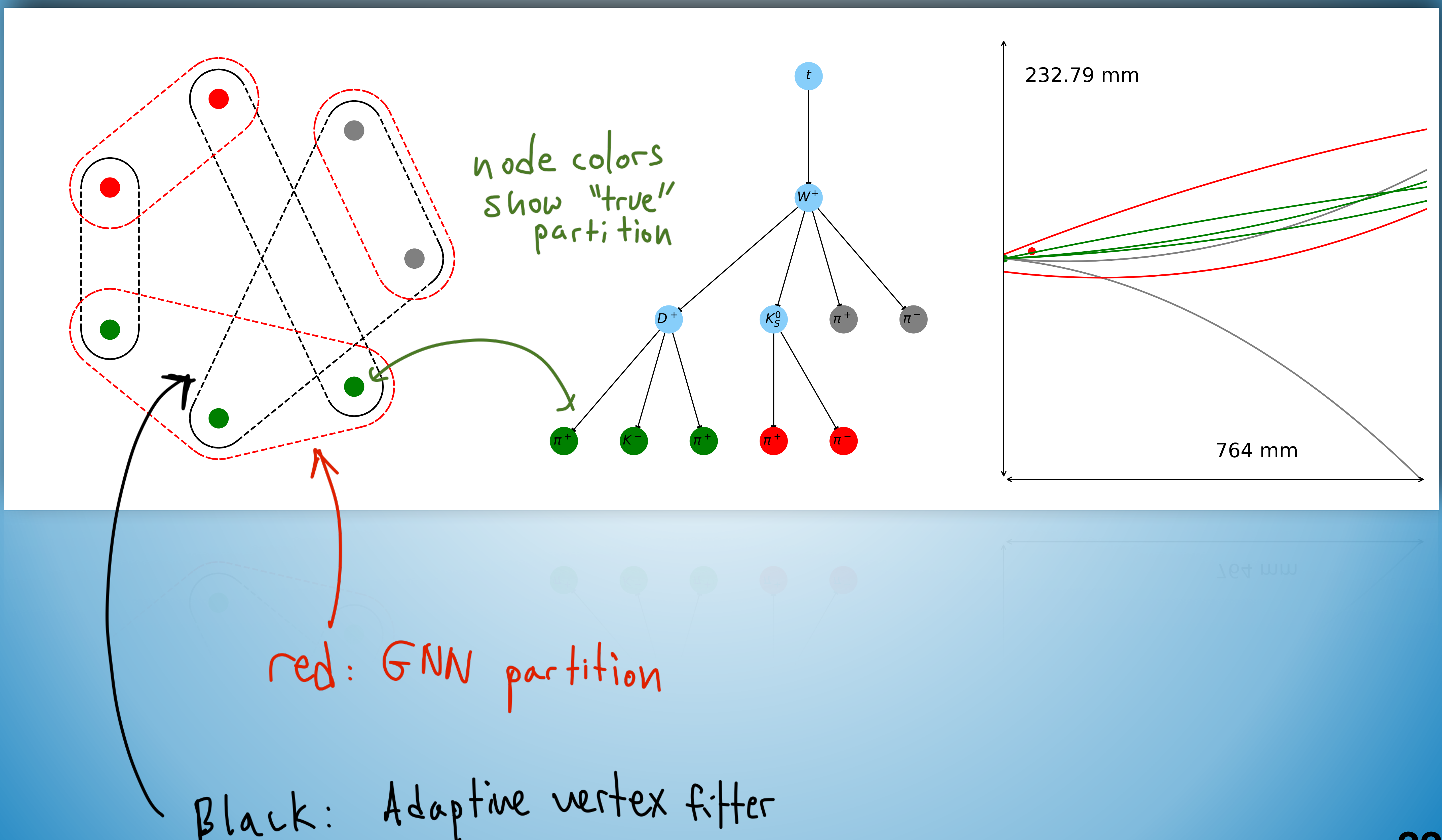
What we need:

- to see the secondary vertices and tracks clearly
- To understand what tracks the algorithm clustered together
- To know what the “truth” was

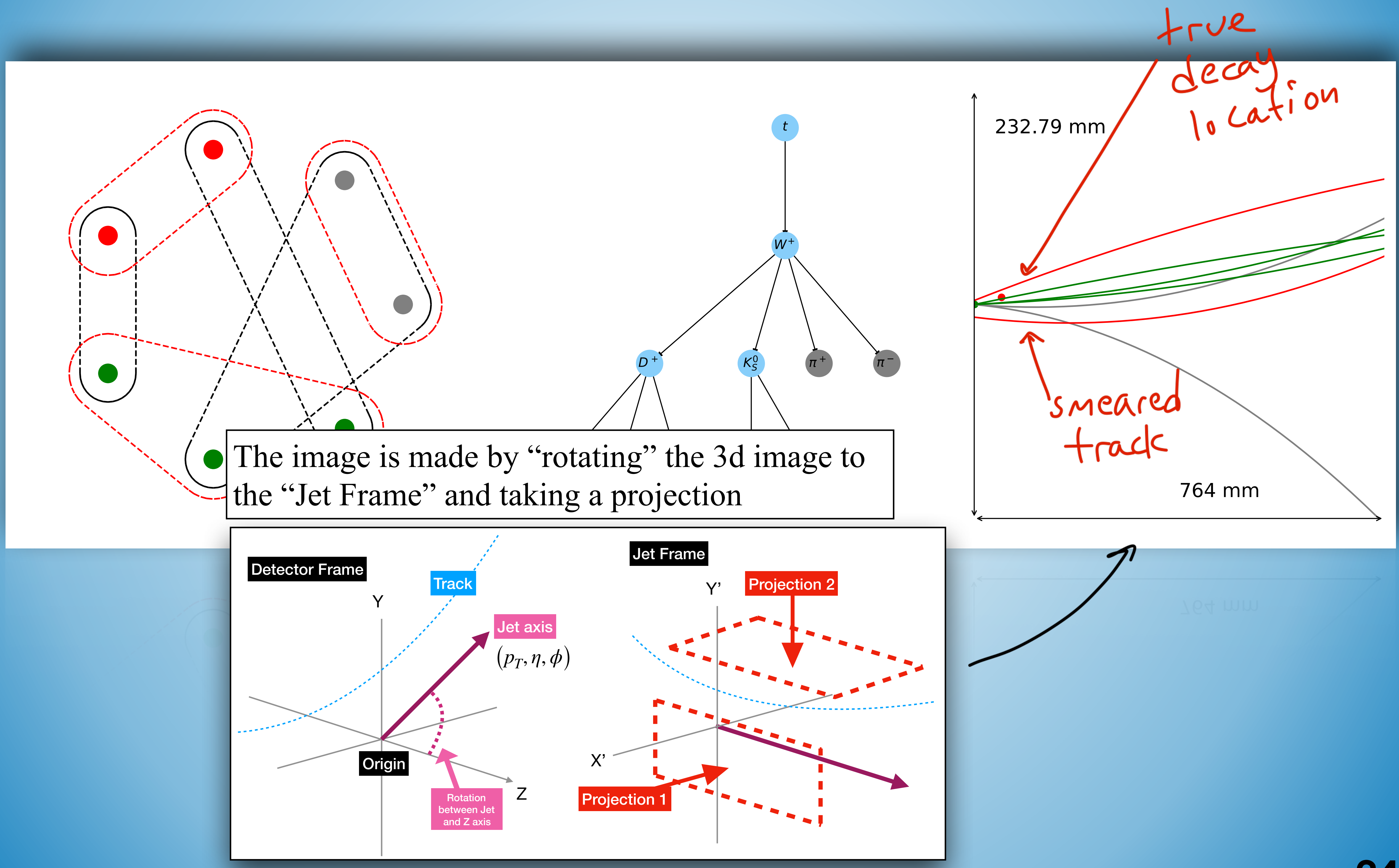
This 3-piece event display gives us all the information,



This 3-piece event display gives us all the information,



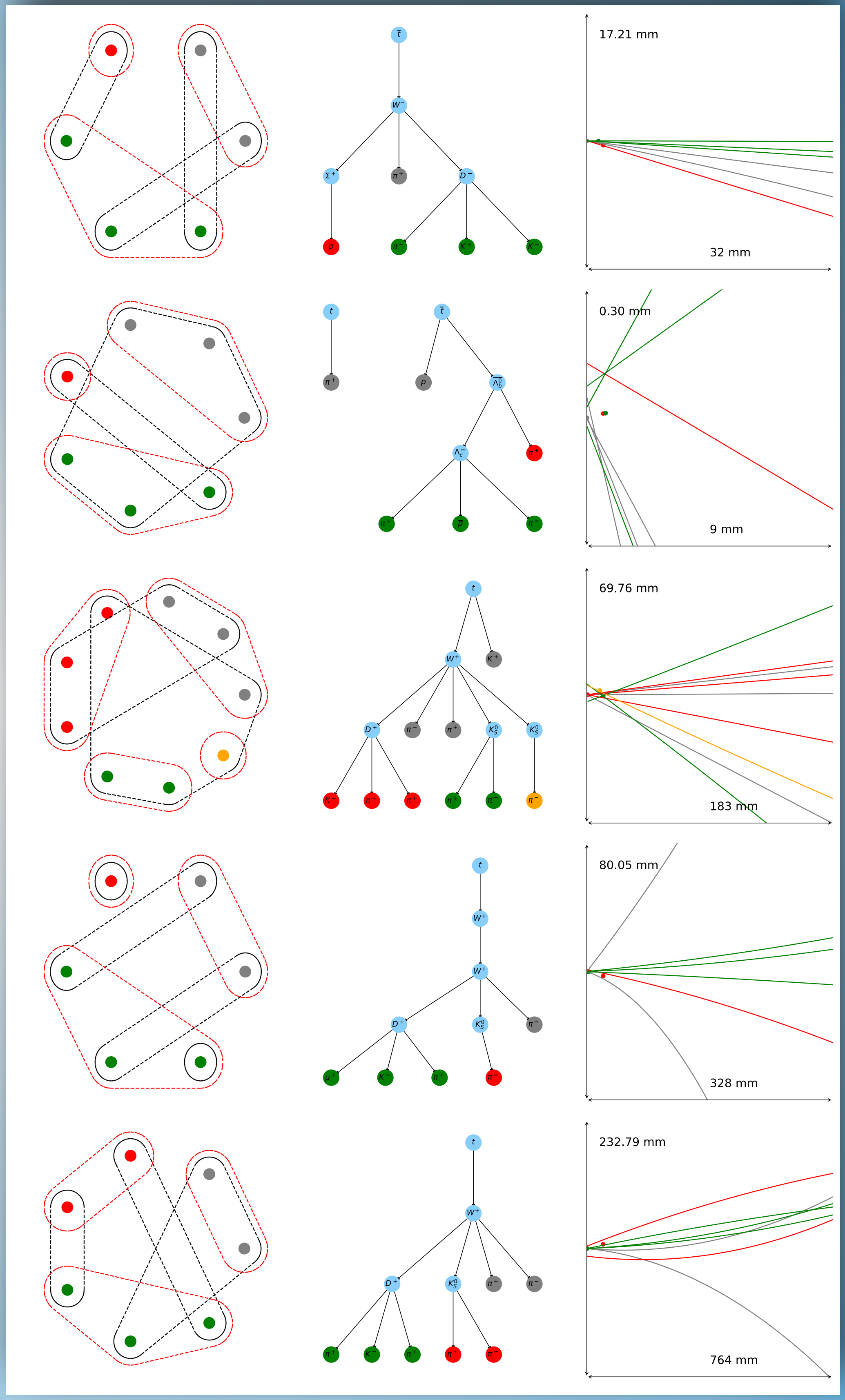
This 3-piece event display gives us all the information,



And its easily browsable in a Jupyter notebook

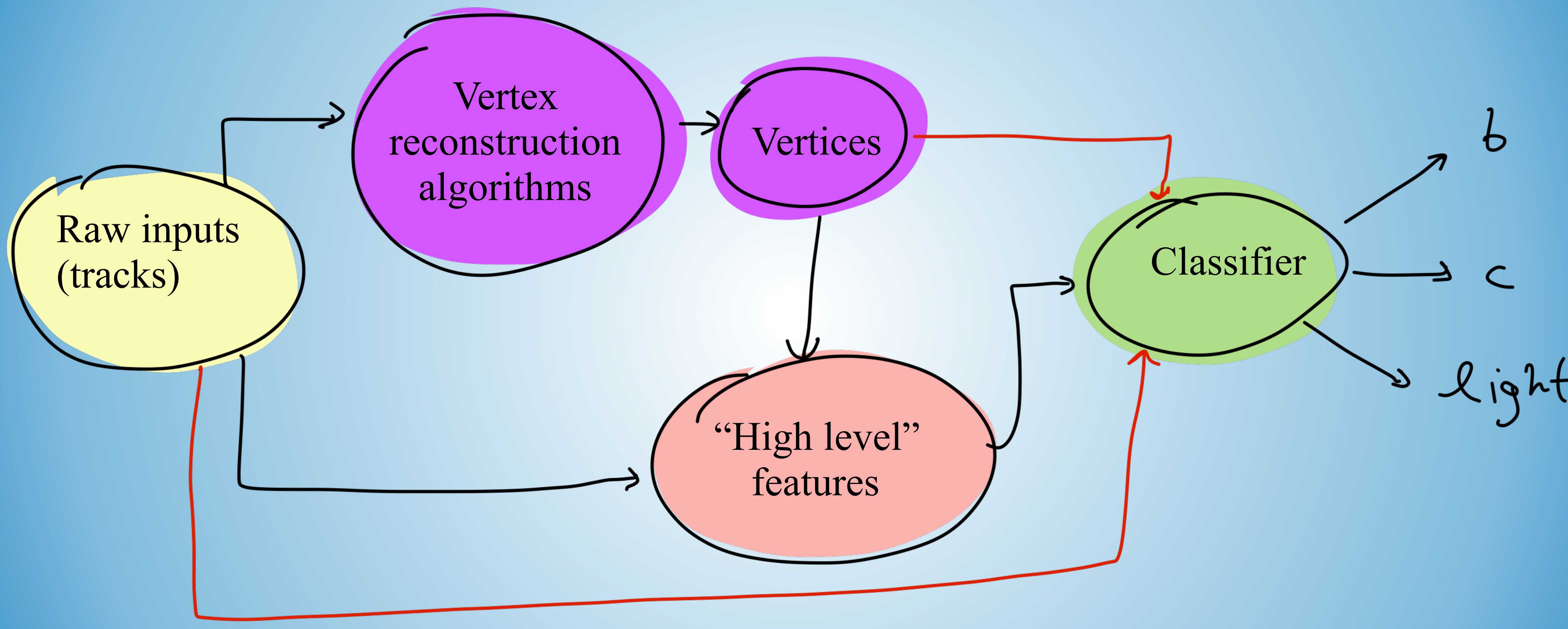
```
In [62]: np.where( ( algo_ari['edgeHS']==1 ) & (algo_ari['Rave']<0.2) & (sec_vtx_max_size_all_jets > 2)
          & (n_vertices > 2))[0]
```

Here are some examples of the neural network performing much better than the adaptive vertex fitter



Summary

- Neural networks are a viable option for this task



- Graph neural networks are a natural fit for the task
- The GNN performs better than the baseline algorithm, but there is a lot of room for improvement - looking for an architecture that takes the context into account

Next Step

- Use particle flow particles (include neutral particles in the vertex finding)
- Release the dataset to the world to try out different solutions

Open question

- Scalability to large graphs (event-level vertex finding)
- What are the causes for vertex finding "failure"