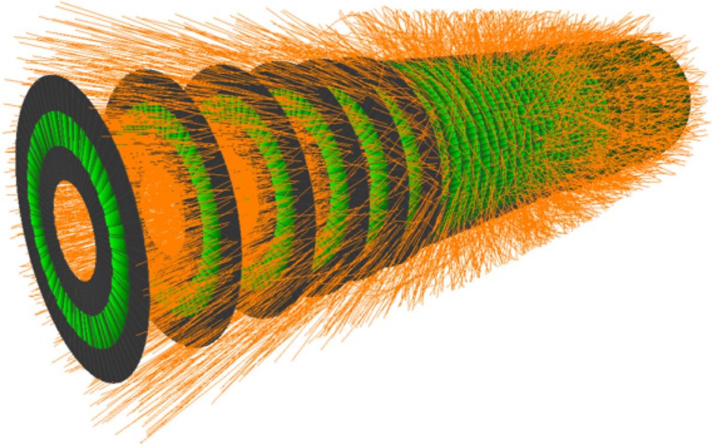
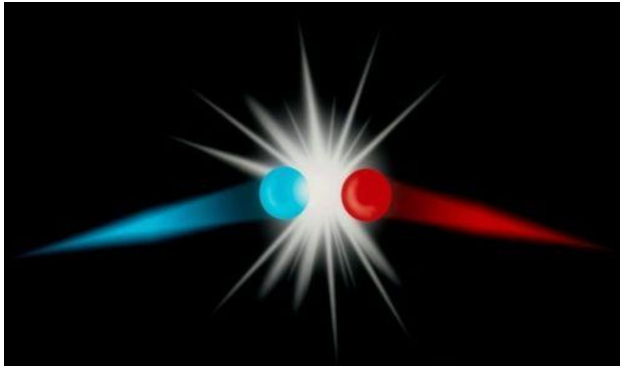
TrackML Particle Tracking Challenge

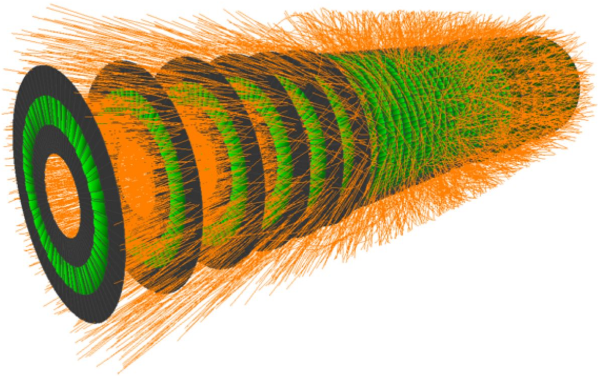


Konstantin Gavrilchik
Artur Fattakhov

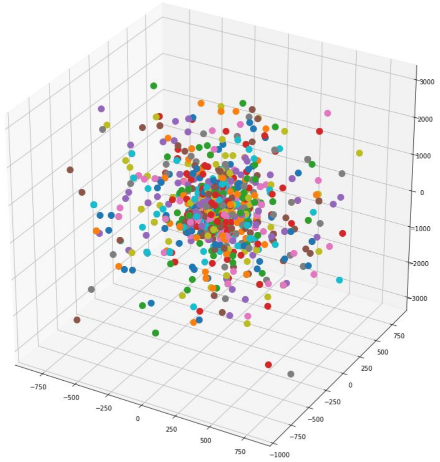
Problem Statement



Some physical event happened

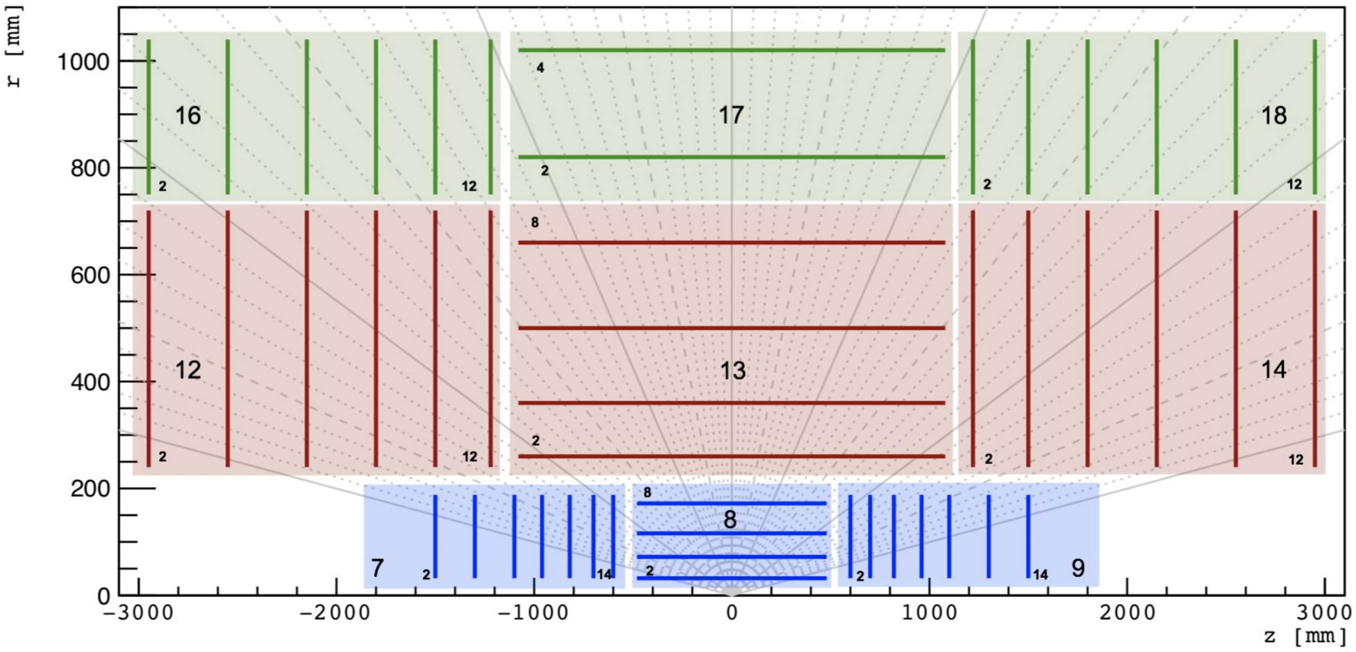


Detector fix particles produced in the event



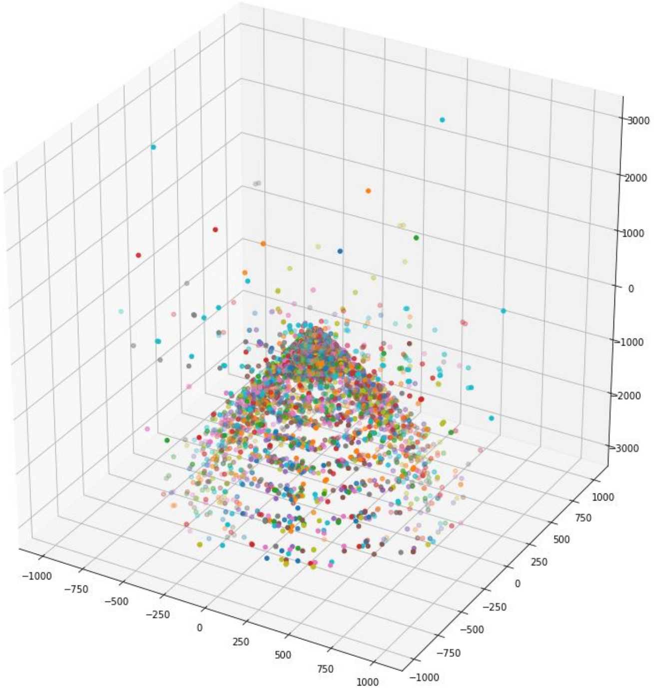
Result - number of points in 3D space

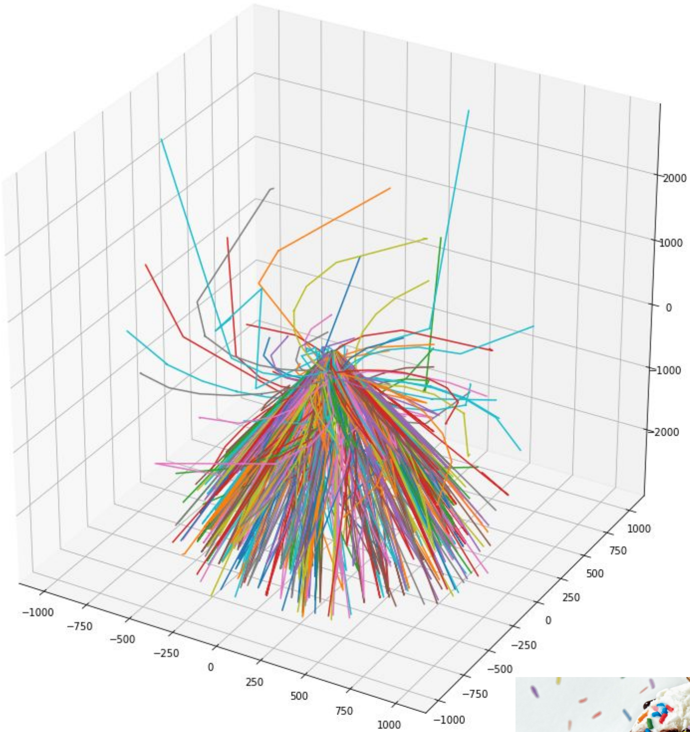
Detector



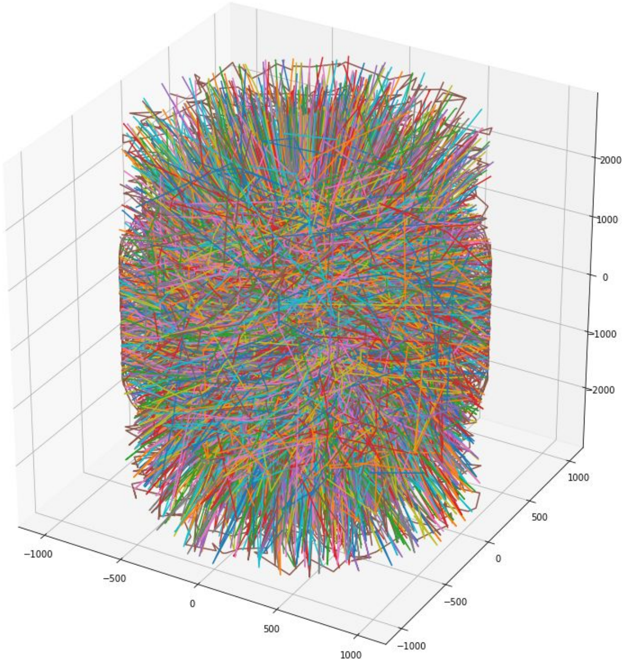
r [mm]
1000
800
600
400
200
0
−3000
−2000
−1000
0
1000
2000
3000
z [mm]
16
17
18
12
13
14
7
8
9
2
12
4
8
14

Main idea

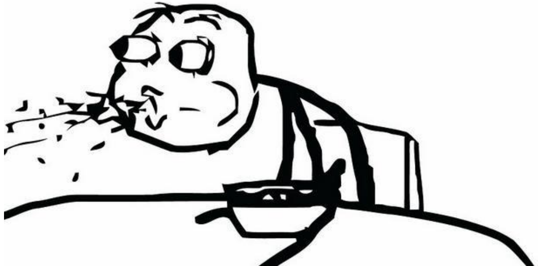

3000
2000
1000
0
−1000
−2000
−3000
−1000 −750 −500 −250 0 250 500 750 1000
−1000 −750 −500 −250 0 250 500 750 1000


2000
1000
0
−1000
−2000
−1000 −750 −500 −250 0 250 500 750 1000
−1000 −750 −500 −250 0 250 500 750 1000

Reality



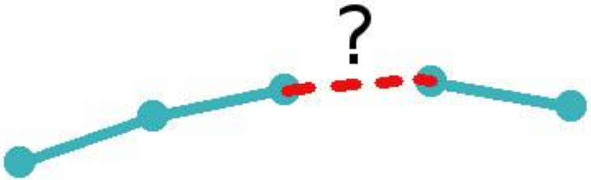
- 100k points per event
- 10% of trash points
- many similar points
- metric?



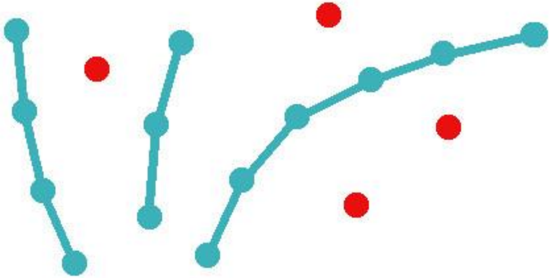
Train	Test
5000x events (**300Gb**) 3D coordinates - some additional characteristic (momentum, charge, etc) - known which hit belongs to which particle	125 events (**10Gb**) - only 3D coordinates

Supervised or Unsupervised?

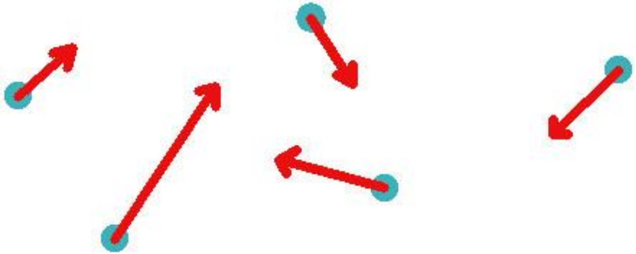
Targets for supervised learning



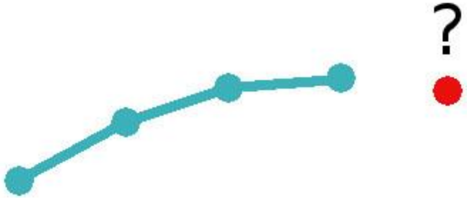
Is pair belongs to the same track?



Is trash hit?



Momentum



Coordinates of the next hit in particle

Supervised learning

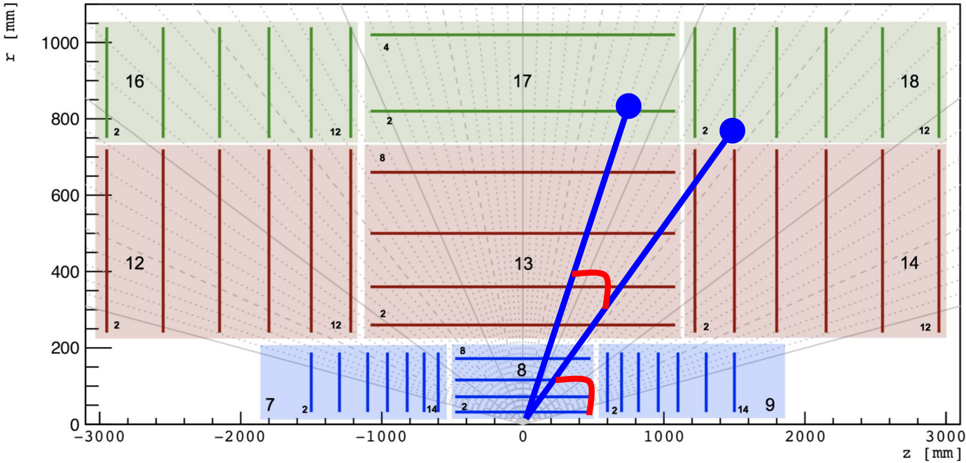
Main troubles:

- 5000x events (~100k points each)
- Pair classification: over 100k*100k points (impossible to run model even on the one event)

Let’s find 30 candidates (closest) for each point and fit model on the dataset with size 30*100k

Features

- Coordinates
- Angle with Z-axis
- Radius and angle in polar/cylindrical coordinate system
- Distance between points
- Angle between points
- Many aggregation features on cells, modules such that:
  - frequenties
  - numbers



Score: ~0.23 (at the end of competition extended to ~0.6)

Right solution

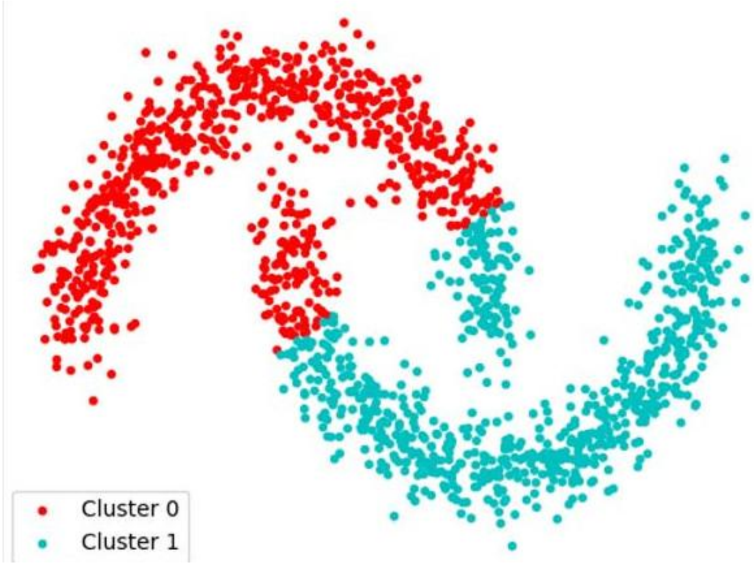
- Use another strategy of pair selection rather than topk-closest.
  - With right heuristics we can cover for about 99% of scores rather than 80% in our simple approach
- Extending the line passing through a pair, and looking where it hits the next adjacent detector layers using 3d geometry then select top-10 closest to this line.
- Fit the helixes
- Fit the random forest

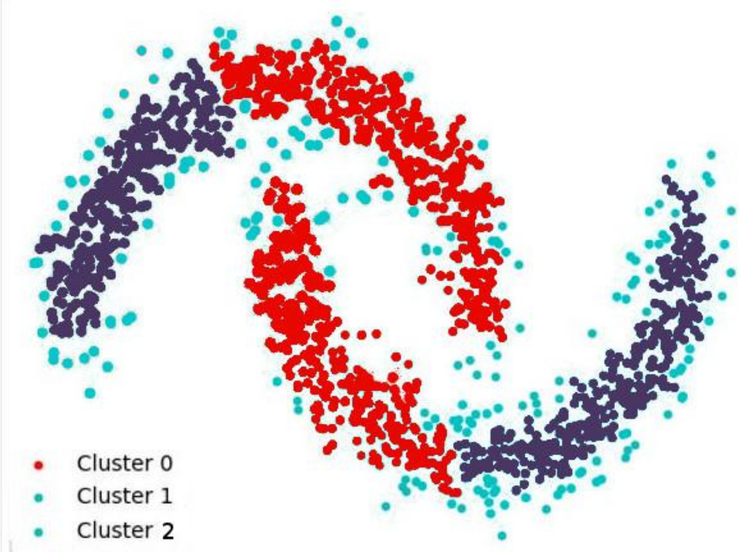
Right solution

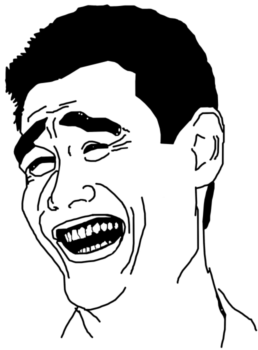
- Use another strategy of pair selection rather than topk-closest.
  - With right heuristics we can cover for about 99% of scores rather than 80% in our simple approach
- Extending the line passing through a pair, and looking where it hits the next adjacent detector layers using 3d geometry then select top-10 closest to this line.
- Fit the helixes
- Fit the random forest

Score ~0.92 - 1st place solution

Unsupervised learning: clustering

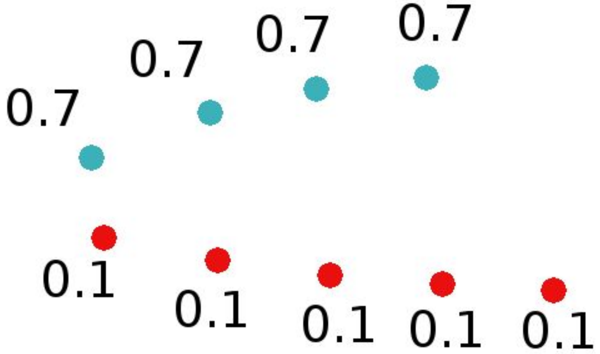




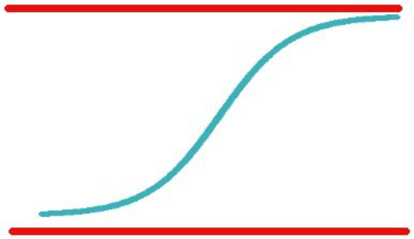


Good features for clustering

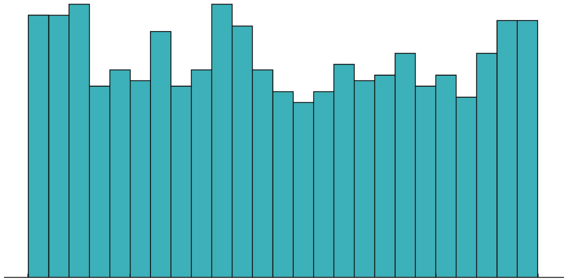
constant in an ideal track



different values for different tracks

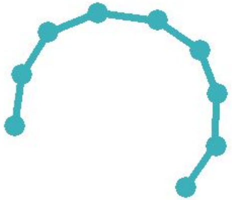


bounded

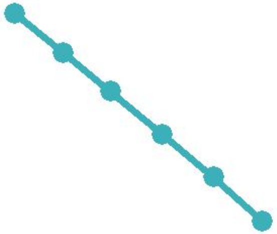


distributed as uniformly as possible

Dbscan



Rounded



Straight tracks

Features example

- x / y
- x / r
- y / r
- r / rt
- cos(phi)
- sin(phi)

Unrolling heilxes

Idea for 0.3x solution: DBSCAN on unrolled helixes

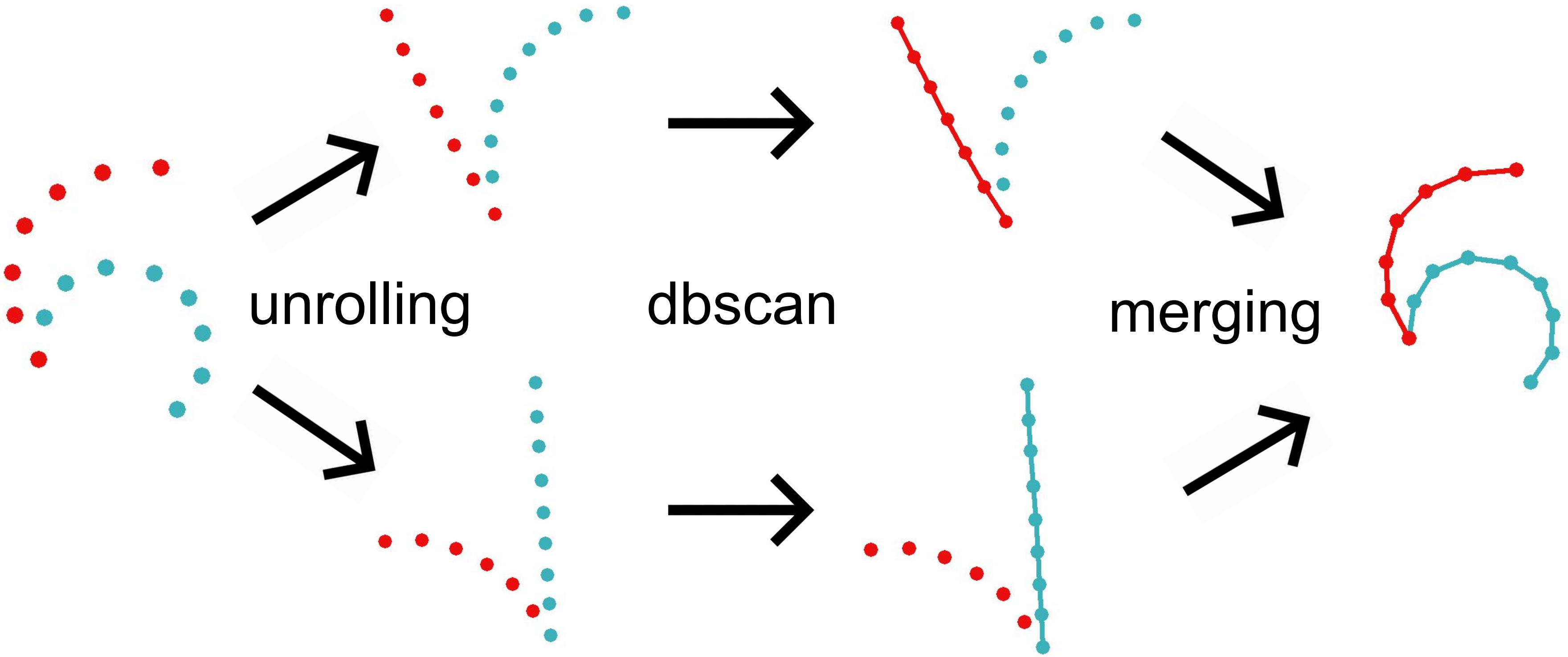
posted in TrackML Particle Tracking Challenge 3 months ago

43

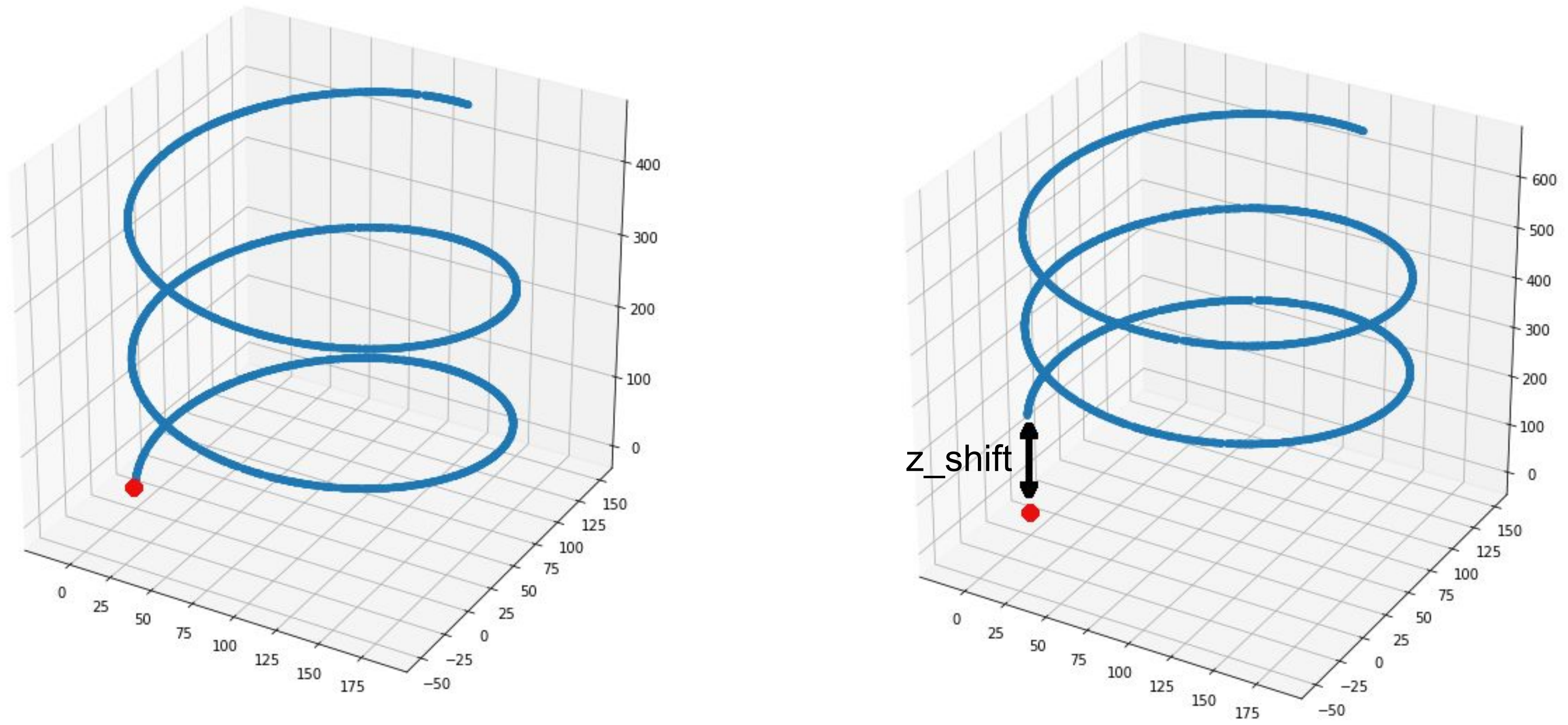
Konstantin Lopuhin

427th place

I'd like to share an approach I used in the last few submissions, getting 0.28 -- 0.38 depending on some details. This is quite far even from the current leaders, and I'm not sure how viable is this approach in the long run, but maybe it will be helpful. At least it's fast to run and does not require any training data.



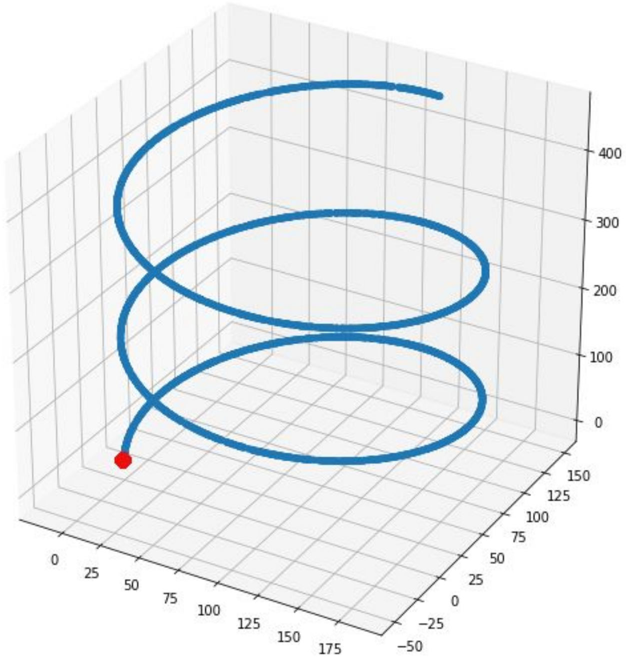
Z-shifting



my_super_feature = f(z)

my_super_feature = f(z-z_shift)

Helix parametrization



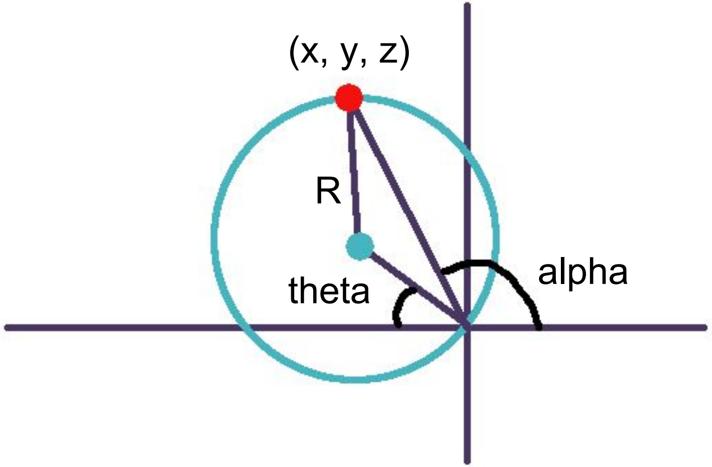
X=-R*sin(w*t+theta)+(R-D)*sin(theta)

Y=-R*cos(w*t+theta)+(R-D)*cos(theta)

Z=Z0+v*t

R, theta

Generate angle of unrolling is equal to generate R



Fix parameter R

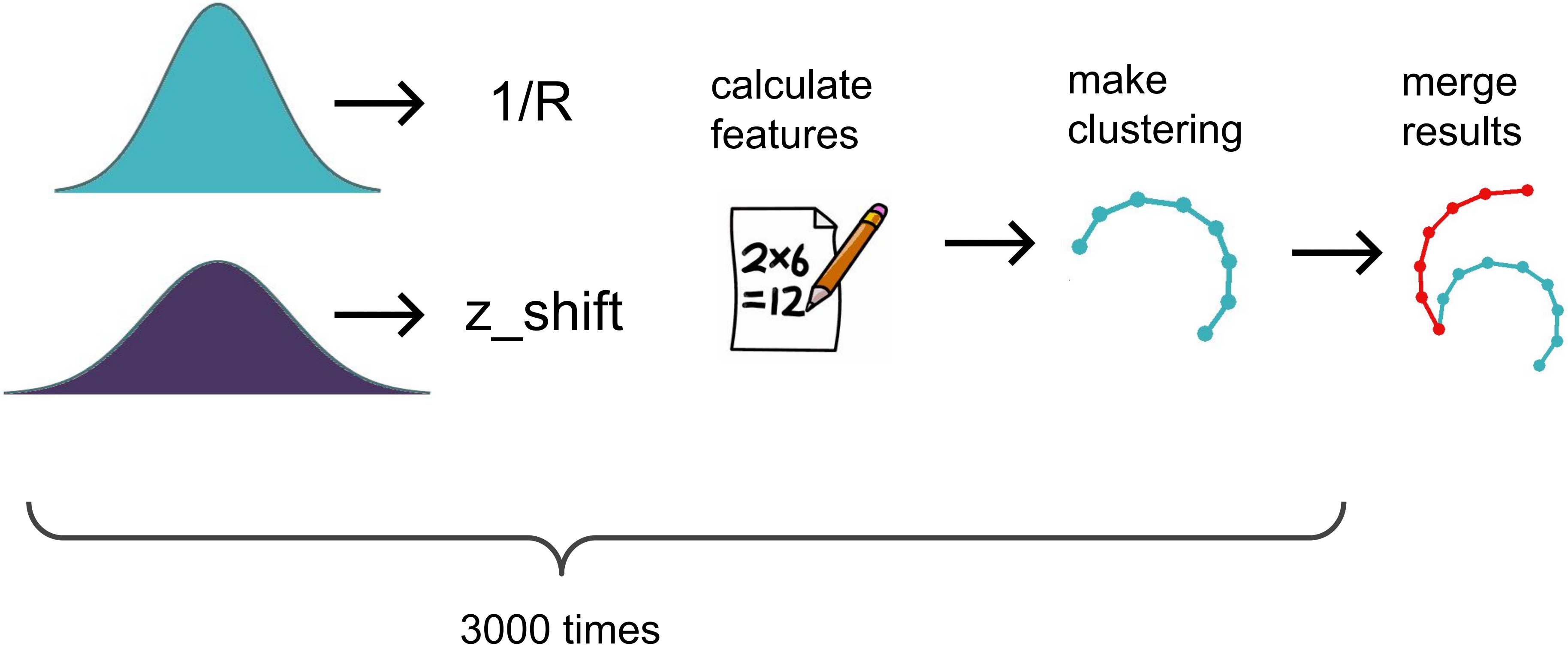
calculate:

R0 = sqrt(x**2+y**2+z**2)

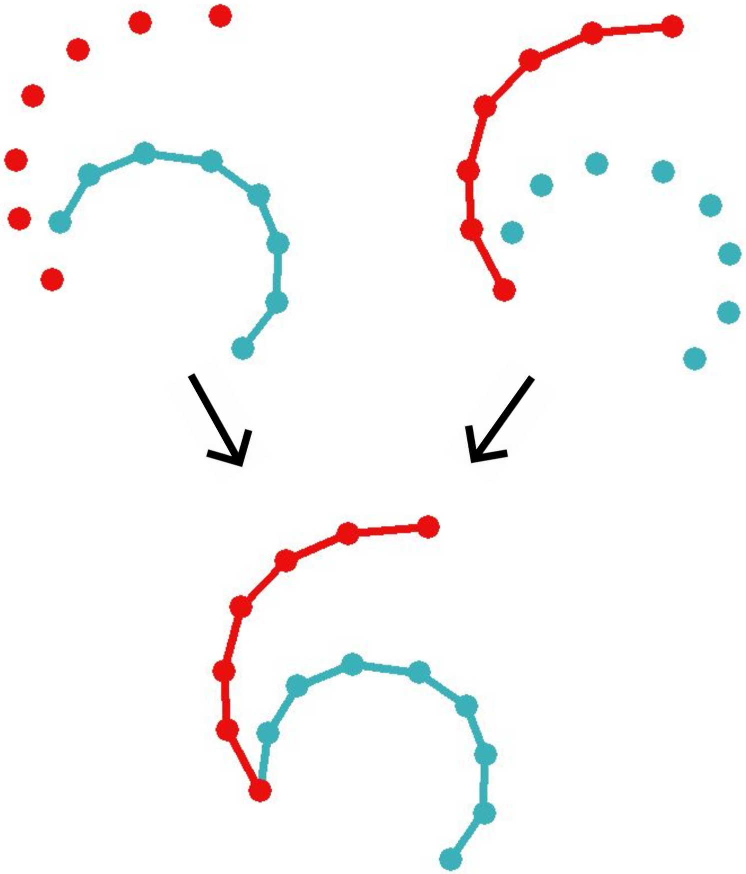
theta = 180 - alpha + arccos(R0/2R)

Use **sin(theta)** and **cos(theta)** as features

Pipeline


1/R
z_shift
calculate
features
2×6
=12
make
clustering
merge
results
3000 times

Simple merging

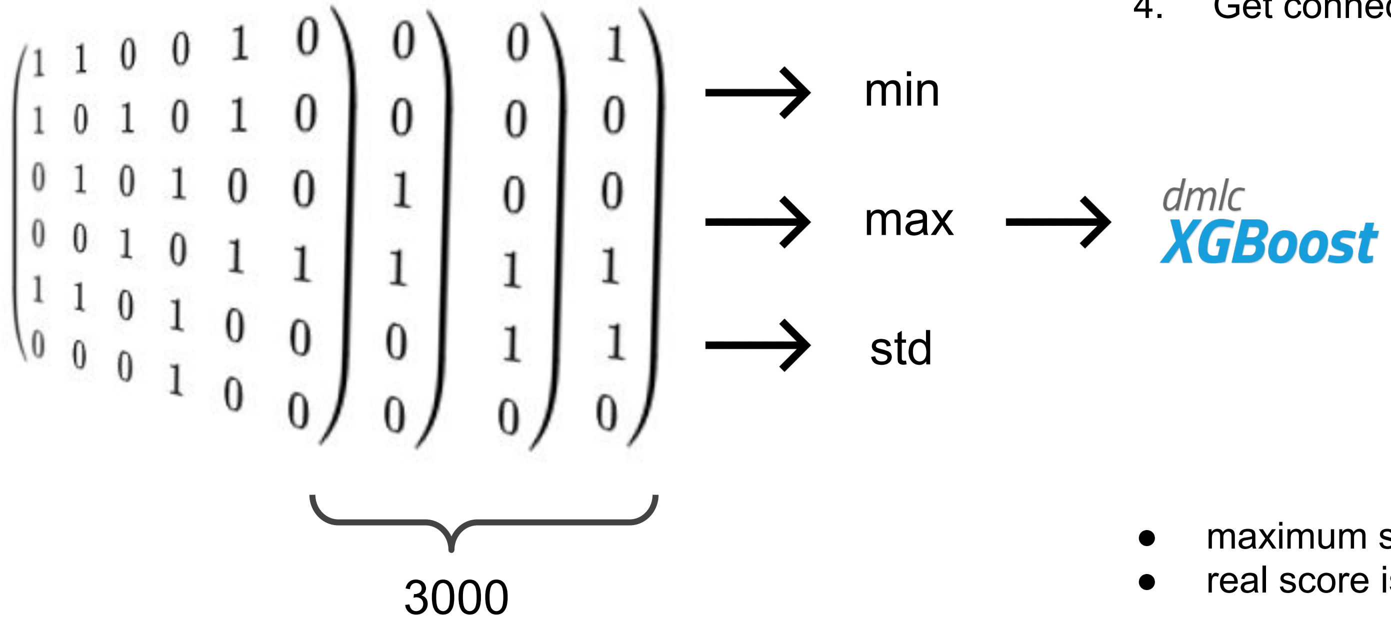


- Tracks can overlap
- Length can be too big
-

Get the longest track for hit

Supervised merging

1. Get adjacency matrices
2. Calculate statistics
3. Fit supervised model
4. Get connectivity components



- maximum score is 0.8
- real score is 0.6 (like simple merging)

Deeper

- Random shuffle of 3000 runnings - better then fixed order. (score ~ 0.6)

Solution: shuffle 10 times than merge 10 submissions (score ~ 0.63)