Sparse Binning Clustering Enhanced by ML Track Scoring

Trian Xylouris
Yuval Reina

Yuval Reina, Tel-Aviv Israel, Yuval.reina@gmail.com

Trian Xylouris, Frankfurt am Main Germany, t.xylouris@gmail.com

Team Name: Yuval & Trian

Private Leaderboard Score: 0.80414

Private Leaderboard Place: 7

Github : https://github.com/tx1985/kaggle-trackML

Kaggle Kernel : https://www.kaggle.com/yuval6967/7th-place-clustering-extending-ml-merging-0-75

Introduction

Main stages

- Clustering – using sparse binning
  - Also include basic track merging and outliner filtering
- ML track merging
- Track extension

Clustering

Main Principle:

- For every hit, calculate N features.
- Try to find hits who's features are close enough and give them the same TrackID

Particle's Parameters

a) (px,py,pz) ≅ Particle Initial Momentum

b) z_0 ≅ initial position in the Z axis

Helix Parametes

a) R ≅ Helix Radius

b) θ ≅ Tangential angle in XY plane

c) ϕ ≅ Slope

Features

Particle's Parameters

a) (px,py,pz) ≅ Particle Initial Momentum

b) z_0 ≅ initial position in the Z axis

Helix Parametes

a) R ≅ Helix Radius

b) θ ≅ Tangential angle in XY plane

c) ϕ ≅ Slope

Features

Particle's Parameters

a) (px,py,pz) ≅ Particle Initial Momentum

b) z_0 ≅ initial position in the Z axis

Helix Parametes

a) R ≅ Helix Radius

b) θ ≅ Tangential angle in XY plane

c) ϕ ≅ Slope

Z

ϕ

l

Features

Features

Particle's Parameters

a) (px,py,pz) ≅ Particle Initial Momentum

b) z_0 ≅ initial position in the Z axis

Helix Parametes

a) R ≅ Helix Radius

b) θ ≅ Tangential angle in XY plane

c) ϕ ≅ Slope

d) $\theta = \arctan\frac{py}{px}$

e) $\phi = \arctan\frac{pz}{\sqrt{px^2+py^2}}$

Independent variables

I. 1/R

II. z_0

Hit's features

1. $\theta\_ = \arctan\frac{y}{x}$

2. $r = \sqrt{x^2 + y^2}$

3. $\Delta\theta = \arcsin\left(\frac{r}{2R}\right)$

4. $\theta = \theta\_ + \Delta\theta$

5. $\phi\_ = \arctan\frac{(z-z_0)}{2R\cdot\Delta\theta}$

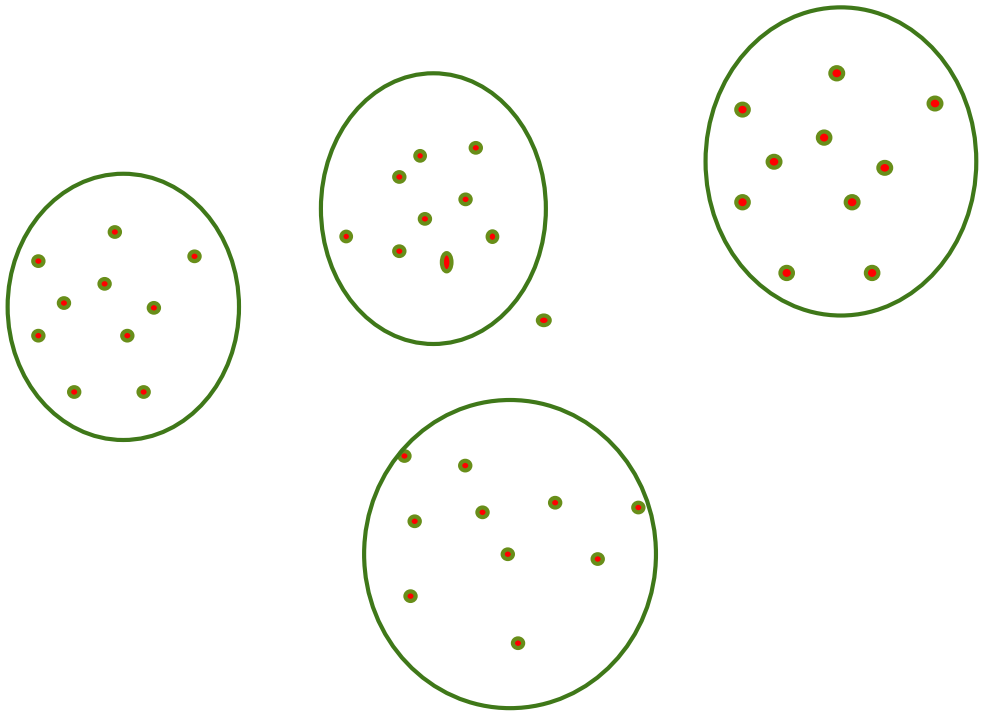
Final features

1. $\text{sint} = \sin(\theta)$

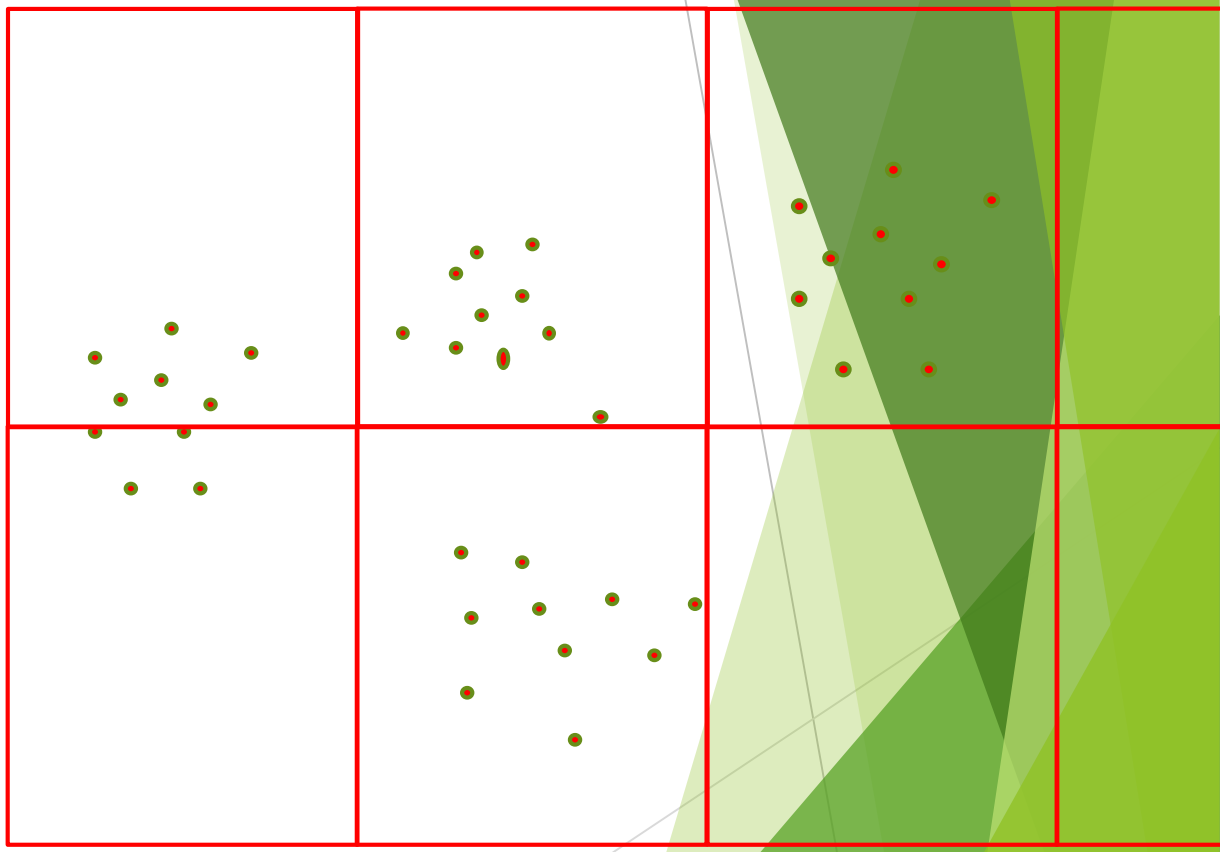
2. $\text{cost} = \cos(\theta)$

3. $\phi^* = \arctan\frac{(z-z_0)}{10.3\cdot\Delta\theta\cdot R}$

Binning



Clustering



Binning

- Bins are spread uniformly
- Most bins are empty and there is a large distance between non-empty bins

For features x,y in [-1,1]

ex , ey = rand([-0.5,0.5])

ID(x) = int(x*Num_bins/2+ex)

ID(x,y) = ID(x)*Num_bins+ID(y)

For a given set of independent variables
the track ID for every hit can be calculated independently

Sparse Uniform Binning

Track Merging/Selection

- A set of tracks is formed by randomly selecting Z_0, R, Ex, Ey
- We count the number of hits with the same track ID
- When merging 2 sets of tracks, every hit gets the track ID which has the largest number of members (and the number of members is recalculated).
- Every N sets of tracks, the surviving tracks are refined, and long enough tracks are permanently set:
  - Refined – can't have 2 hits with same detector ID, at most 2 hits can be from the same layer.
  - Permanently set – Hit's belonging to these tracks are removed from the pool

Choose Tracks according to a Good/Bad critiria using ML

- Extract features for every track:
  - variance of x,y,z (these are the most important)
  - minimum of x,y,z
  - maximum of x,y,z
  - mean of z
  - volume_id of first hit
  - number of clusters per track (i.e. are there many hits, which are close together?)
  - number of hits divided by number of clusters
- Use LightGBM to score the tracks
- Training:
  - Target = 1 - True tracks from the training set
  - Target = 0 – False tracks created by the clustering algorithem
- Results ~ 95% for precision, accuracy and recall

ML – Track Merging/Selection

Track Extension

- Sparse binning only clusters very “accurate” hits => Some of the track’s hits are left out.
- In the track extension phase we try to extend a track by adding loose hit’s which are close to the track.
- In practice we try to add hits from short tracks to long tracks

The Full Solution

- First Clustering Stage

  For j in range(N):

  For i in range(500):

  Randomly choose 2 independent variables

  Calculate TrackID for every hit (Clustering) to create a single TrackID set

  Merge New TrackID set with the current merged TrackID set

  Check all tracks and remove outliner hits

  If len(track)>MaxLen – fix the track by removing it's hits from the hit pool

- ML merging:

  Do the clustering stage M times

  Use ML to score every track

  Every hit get the TrackID with the maximum score

- Extend tracks

Basic Clustering and Merging

Number Of track sets	Score	Score after extension
1000	0.51	
1600	0.56	
5500	0.636	0.73
100000	0.73	0.78
7 * 100000 (+ML)	0.78	0.804

Machine Learning

- 7 * 100000 +0.01
- 64 * 1000 0.51 ⟶ 0.78

Results

Further Improvements

- Speed
  - Feature Calculation
    - Computing TrackID for 120,000 hits on single i5 core: 4mSec
    - TrackID for 100,000*120,000 hits should take a few seconds on Tesla V100
  - Merging
    - The slow part in the algorithm – counting the number of hits per unique TrackID – in python needs to use numpy.unique – O(Nlog(N)) [use sort]
    - Can be O(N) => huge speed up (takes more memory)
- Hybrid solution:
  - Can be used to create a large number of good short tracks fast, as the 1st stage in a more accurate algorithm
- Use a corrected magnetic field – add 0.02 to the final score

Weaknesses

- Tracks that starts far from the origin are very hard to find
- We couldn’t find an improvement which would find such tracks

Thank You