Finding a shower in the OPERA

Machine learning-powered searches of EM-showers in the data of the OPERA experiment

Alex Rogozhnikov

Yandex School of Data Analysis
INFN Napoli Visiting Researcher

Naples, International Workshop on Nuclear Emulsions, 2016

Outline

- EM-showers in the OPERA
- Machine learning
- Machine learning for EM-shower searches in the OPERA

Yandex School of Data Analysis

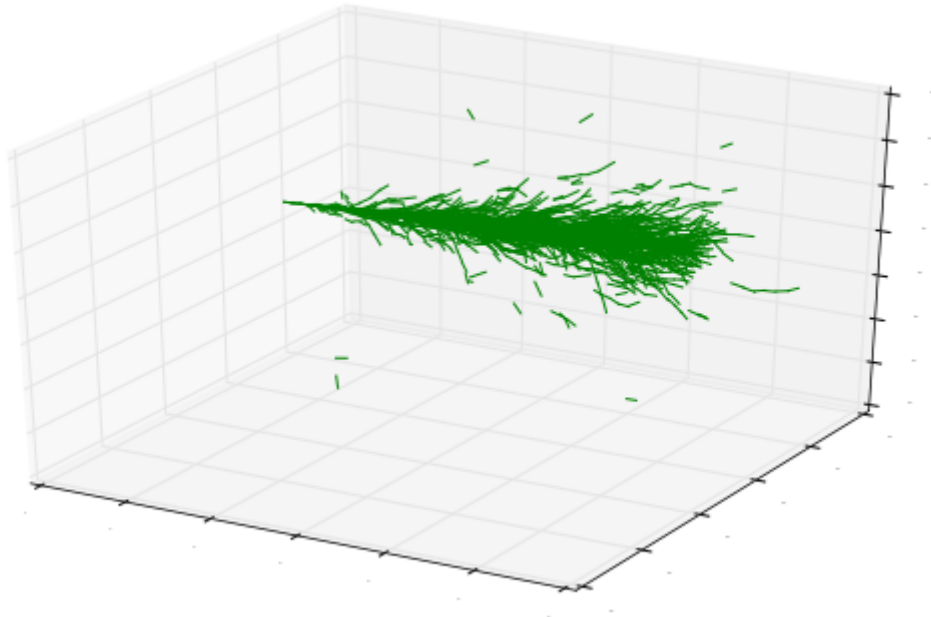
- Education:
  - Strong Curricula in Data & Computer Science
  - Free tuition
  - No employment obligations on the students
    - yet many join Yandex
  - 500+ students graduated since 2007

- Research
  - Interest in interdisciplinary research (eScience)
  - Full member of LHCb and SHiP
  - Organizes Machine Learning Conference

Motivation for shower searches

By studying EM-showers we can infer information about energy of produced particles.

E.g. this is important for detecting $\tau^- \rightarrow e^- \nu_\tau \bar{\nu}_e$ in ν_τ searches using emulsion.



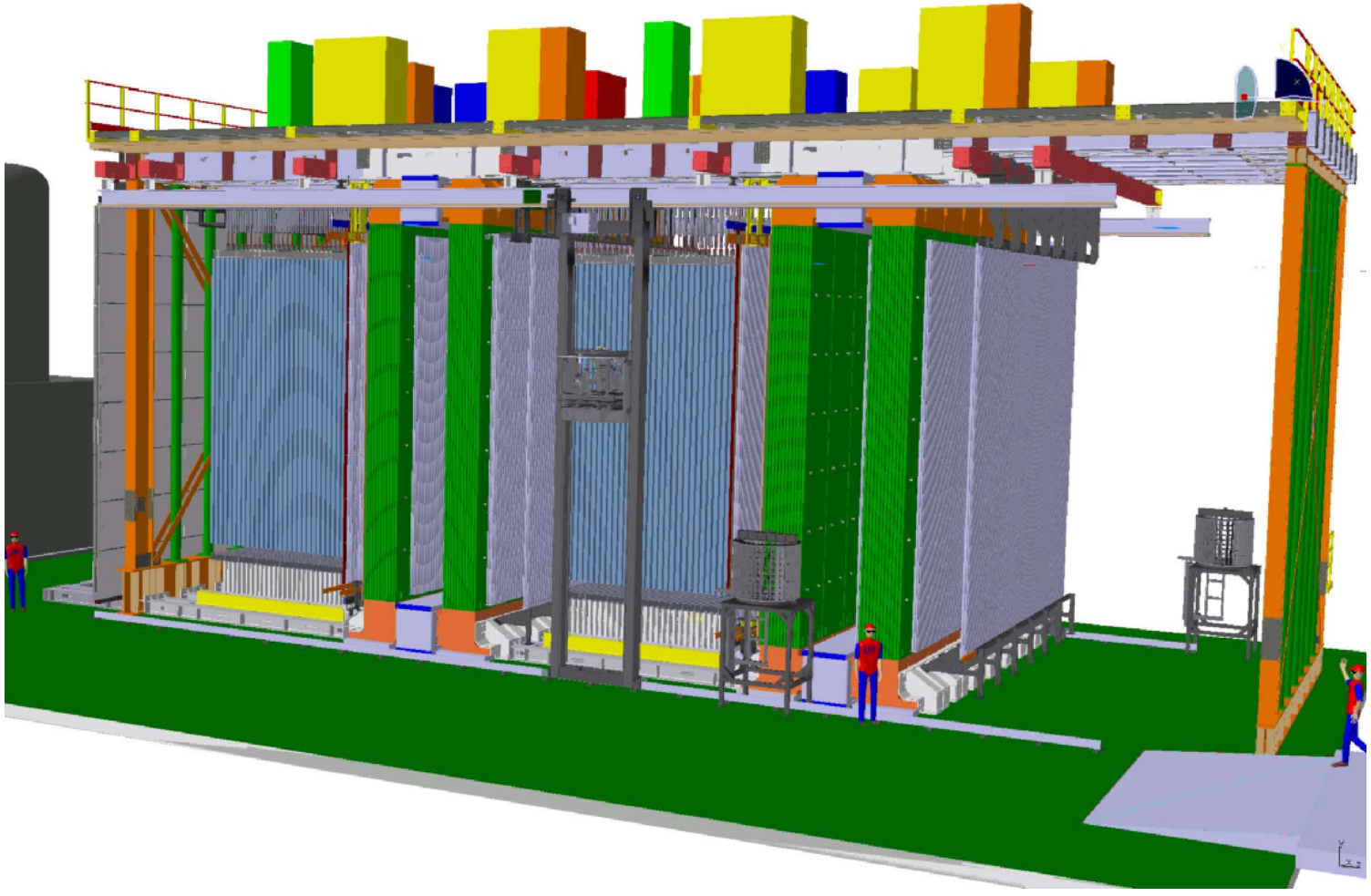
Neutrino Oscillations

- OPERA = Oscillation Project with Emulsion-tRacking Apparatus (2008-2014)
- mostly known due to "faster-than-light neutrinos"
- experiment: start from Super Proton Synchrotron (SPS), beam of pions and kaons
  - which decay into muons and muonic neutrinos
  - beam is sent through the mountains to the Gran Sasso laboratory
  - two detectors: OPERA and ICARUS are detecting neutrinos
  - 730 km, ~ 0.9% of ν_e
  - looking for tau neutrinos ν_τ

Neutrino Oscillations

- 2015 Nobel prize:
  - Super-Kamiokande demonstrated disappearance of atmospheric neutrinos
  - Sudbury Neutrino observatory published clear evidence conversion of solar e-neutrinos to ν_τ, ν_μ
  - details by committee
- future competitor: DUNE (collaboration formed in Jan 2015)
  - LBNF: Fermilab to Sanford Undeground research facility (1300 km)
  - DUNE doesn't own a detector, but has a nice animation

Opera detector



Opera ECC Brick

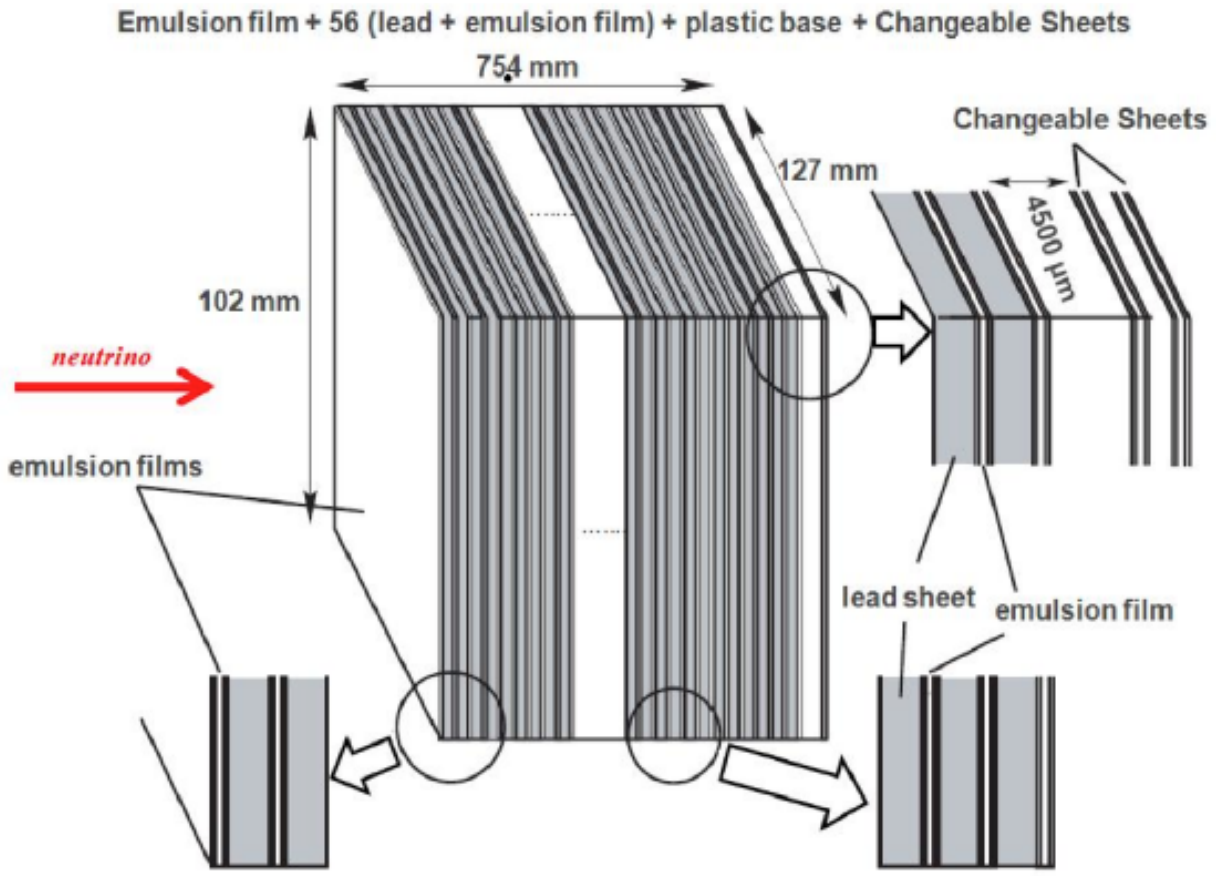
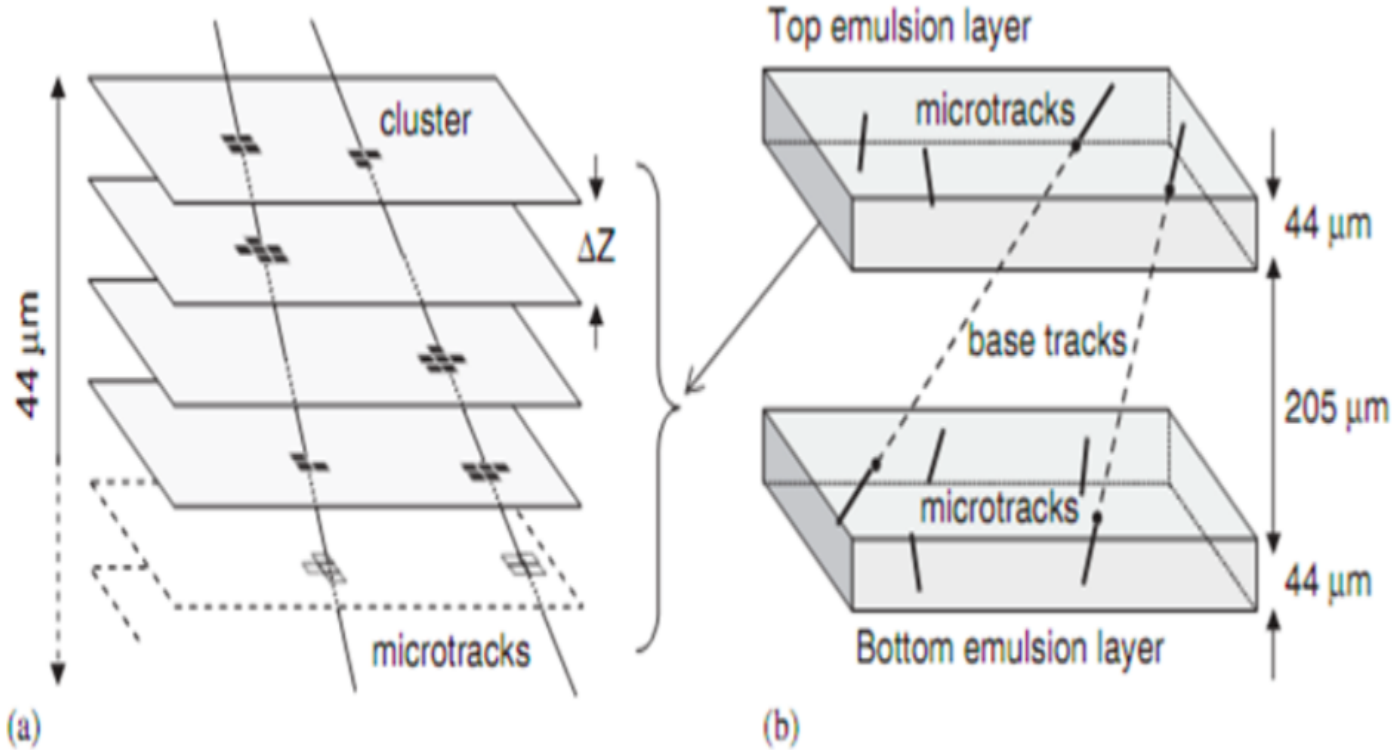


Figure 2.4 – Schematic structure of an ECC brick.

Track reconstruction

After the brick is taken out, it is scanned (which is quite a long process). After scanning a set of **base tracks** is collected

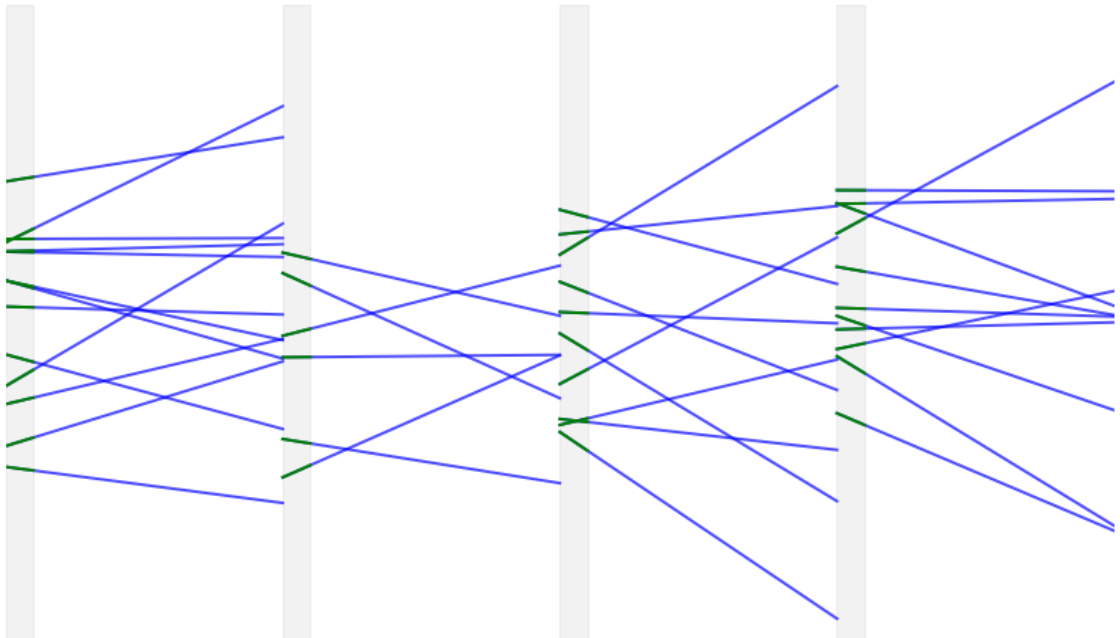


Data structure

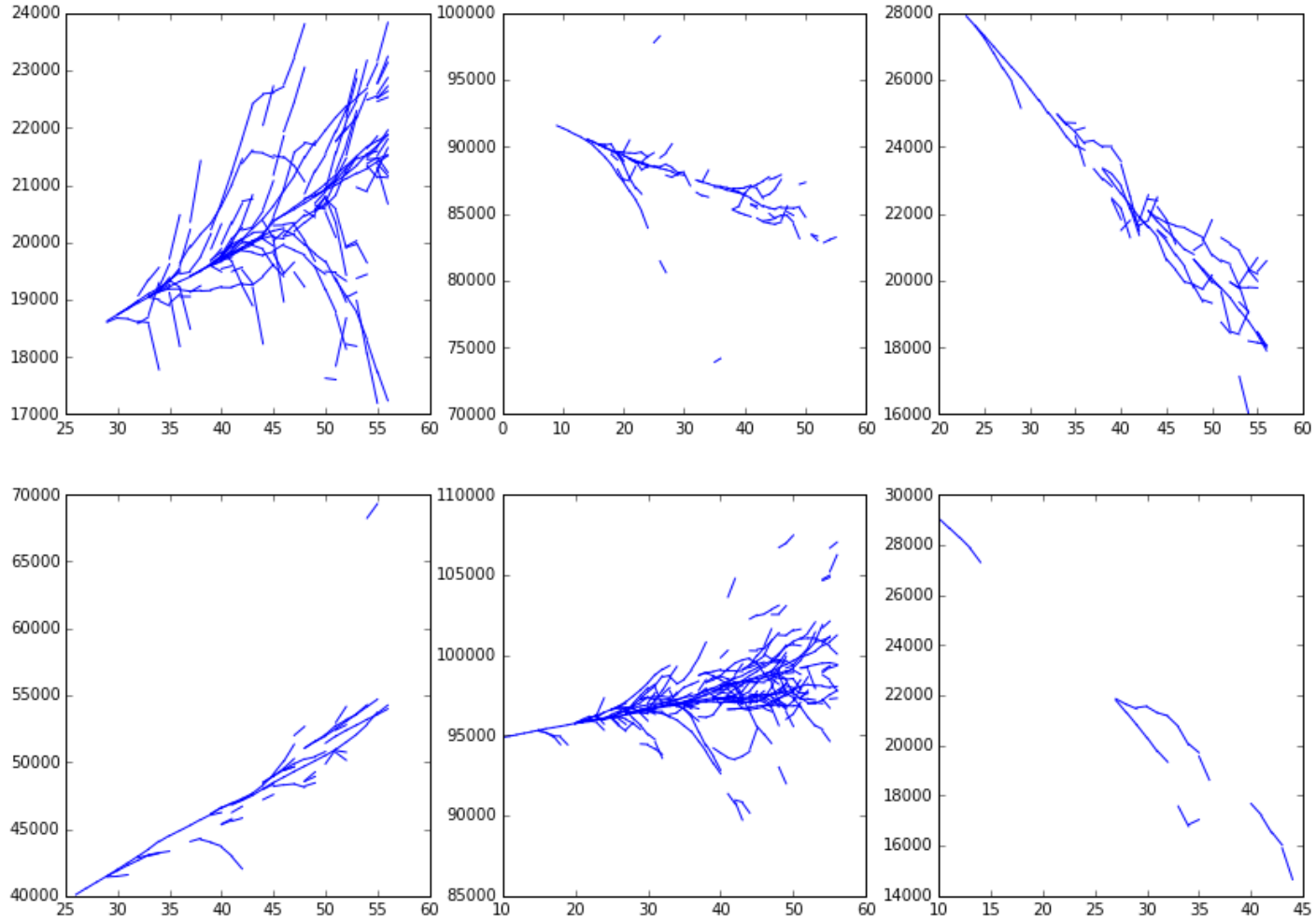
We're given a set of base tracks, each is described by

- plate id
- X, Y in the plate
- slope SX, SY defining a direction of a track
- track quality χ^2

In my plots I continue base track to the next plate.



EM-showers approximately look like (2d projections):

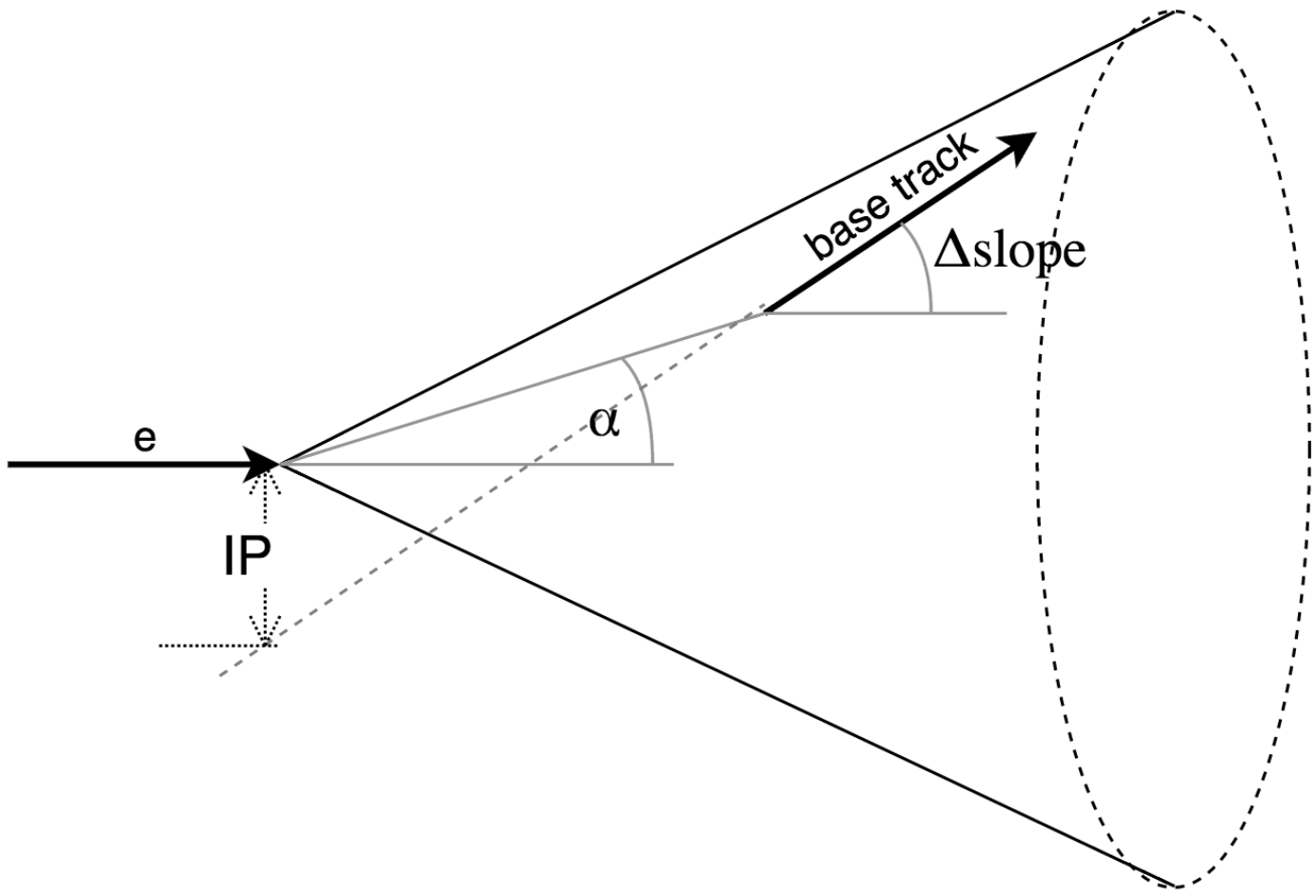


Basic approach from [1] to analyze showers

1. We start from *knowing* position and direction of a shower
2. Classify tracks into background and signal individually
   - the only information used is position / direction of base track (BT), shower position / direction and track χ^2
   - we can use machine learning here
3. Estimate number of tracks in the shower knowing signal and background efficiencies.

[1] PhD thesis of B. Hosseini.

Features visualization:



Data provided for experiment with algorithm

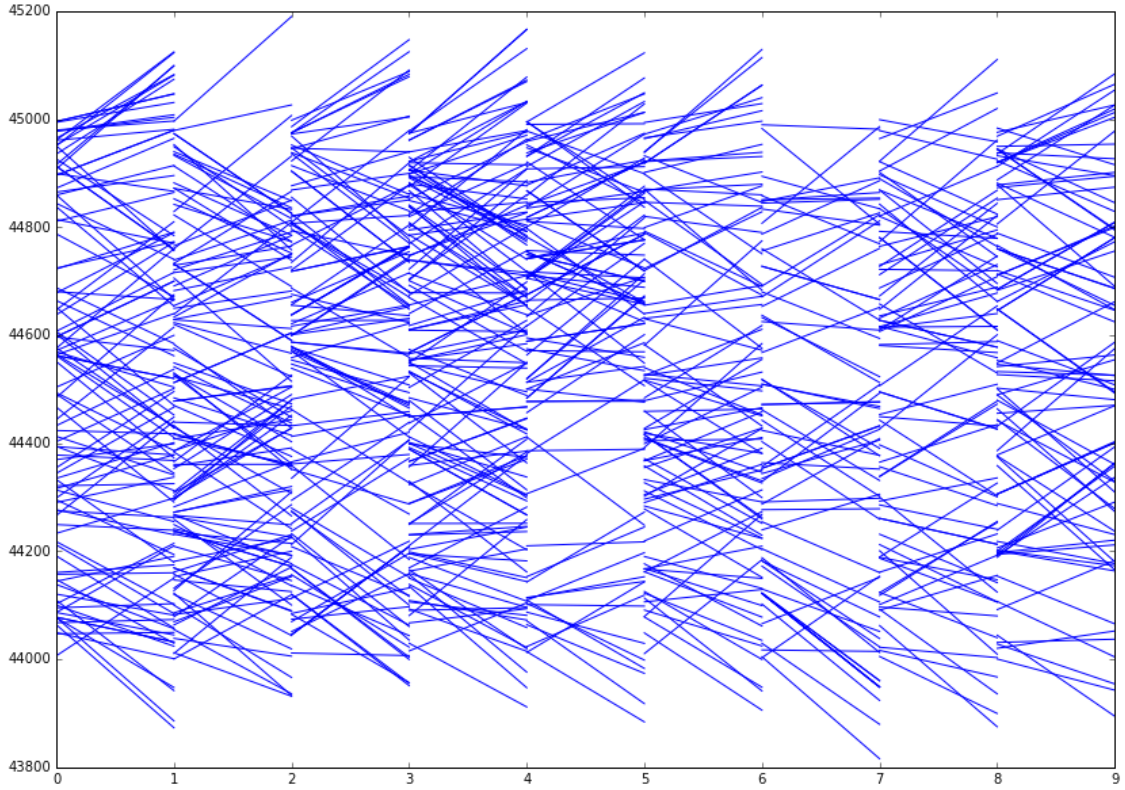
- 8 plates from one brick (48-56) as pure background
  - ~ 4 000 000 base tracks
  - background is generated by subsampling of (right/left) part of the brick
  - and using looping over Z axis (=plate id)
- MC simulation of EM showers (~12000 samples after preselections)
  - ~ 3 000 000 of base tracks in total

Background sample

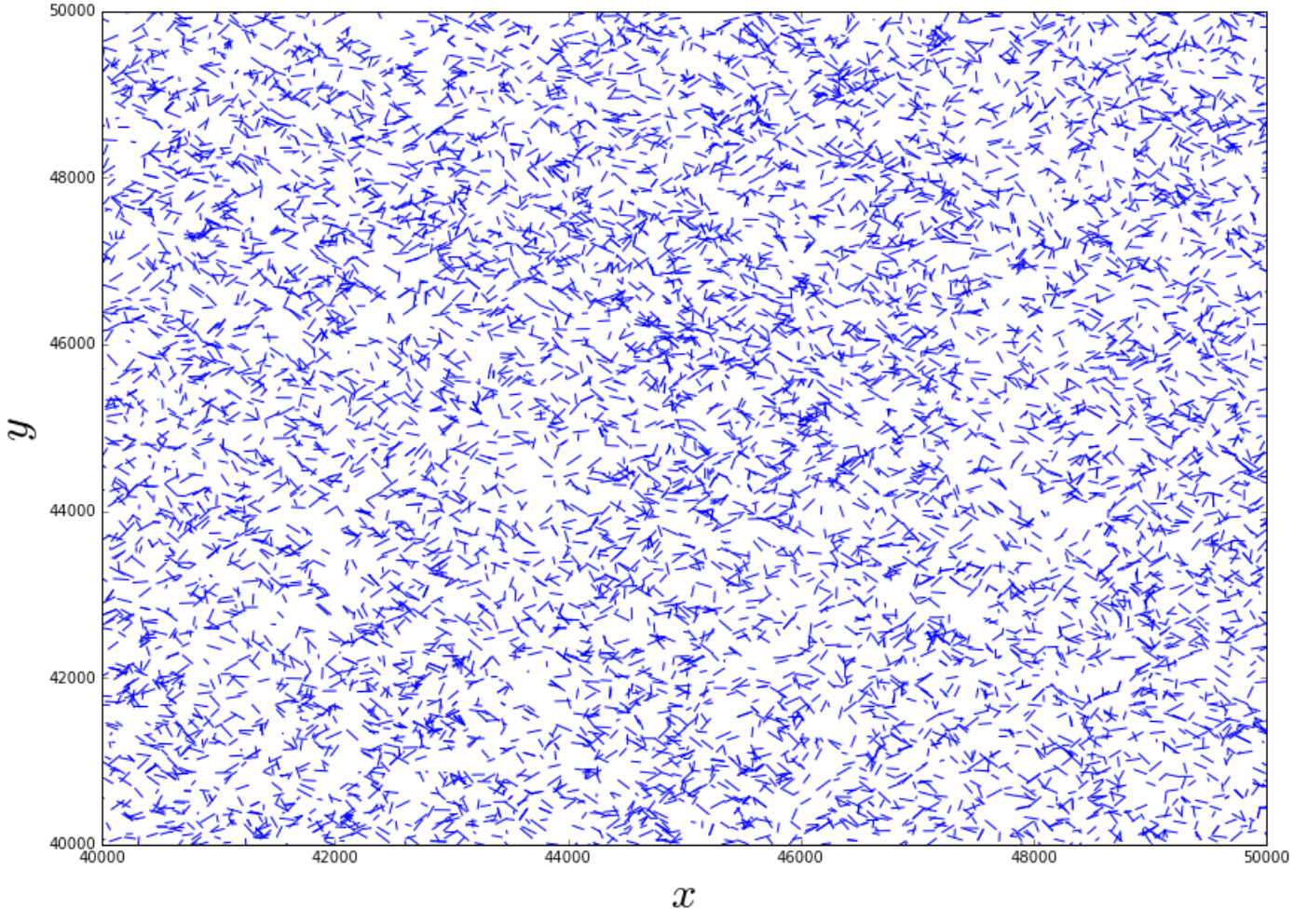
(very small piece with preselections on the track direction shown)

I haven't found signs of any significant inter-plate correlation.

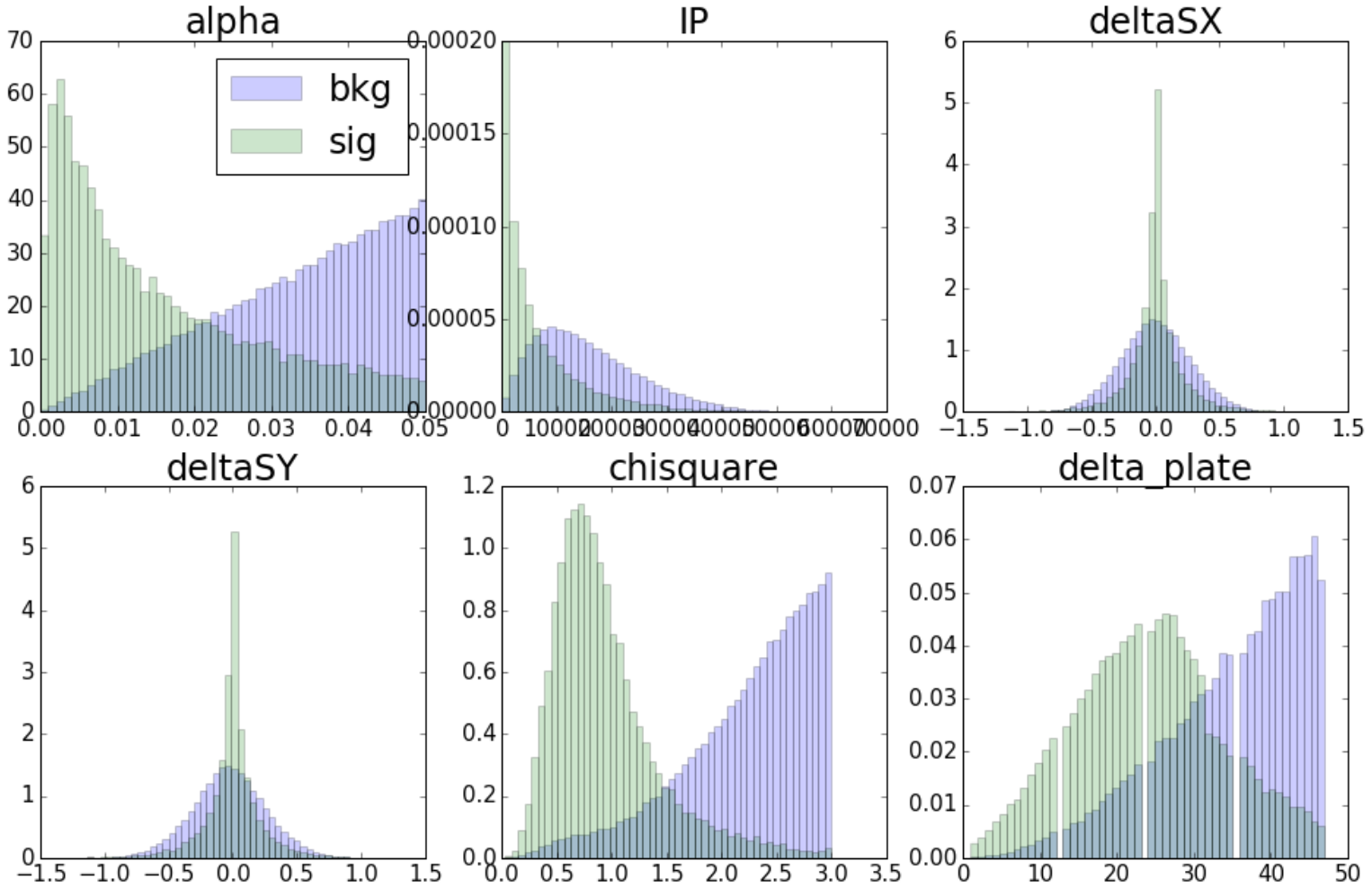
Good: it's mostly instrumental background!



Background sample (xy — projection)



Distribution of features



Some additional geometric features

- distance between track and shower line
- distance between track line and shower line
- angle between direction from electron and track direction (DIRA)
- difference in Z projections between track and shower start

The process of inventing variables (*features*) with high discrimination is called *feature engineering*.

There's not much to add to this set of features.

How can we effectively *combine* information from introduced features to make decision?

Machine Learning

- reconstruction of statistical dependencies without having *explicit form of dependency*
- active area of research
- many algorithms are developed
- for HEP purposes NN (neural networks) and GBDT (gradient boosting over decision trees) are most appropriate choices
- in this analysis I'm using XGBoost (open-source GBDT implementation)

Machine Learning

- reconstruction of statistical dependencies without having *explicit form of dependency*
- active area of research
- many algorithms are developed
- for HEP purposes NN (neural networks) and GBDT (gradient boosting over decision trees) are most appropriate choices
- in this analysis I'm using XGBoost (open-source GBDT implementation)

I'll outline the idea of GBDT in this presentation

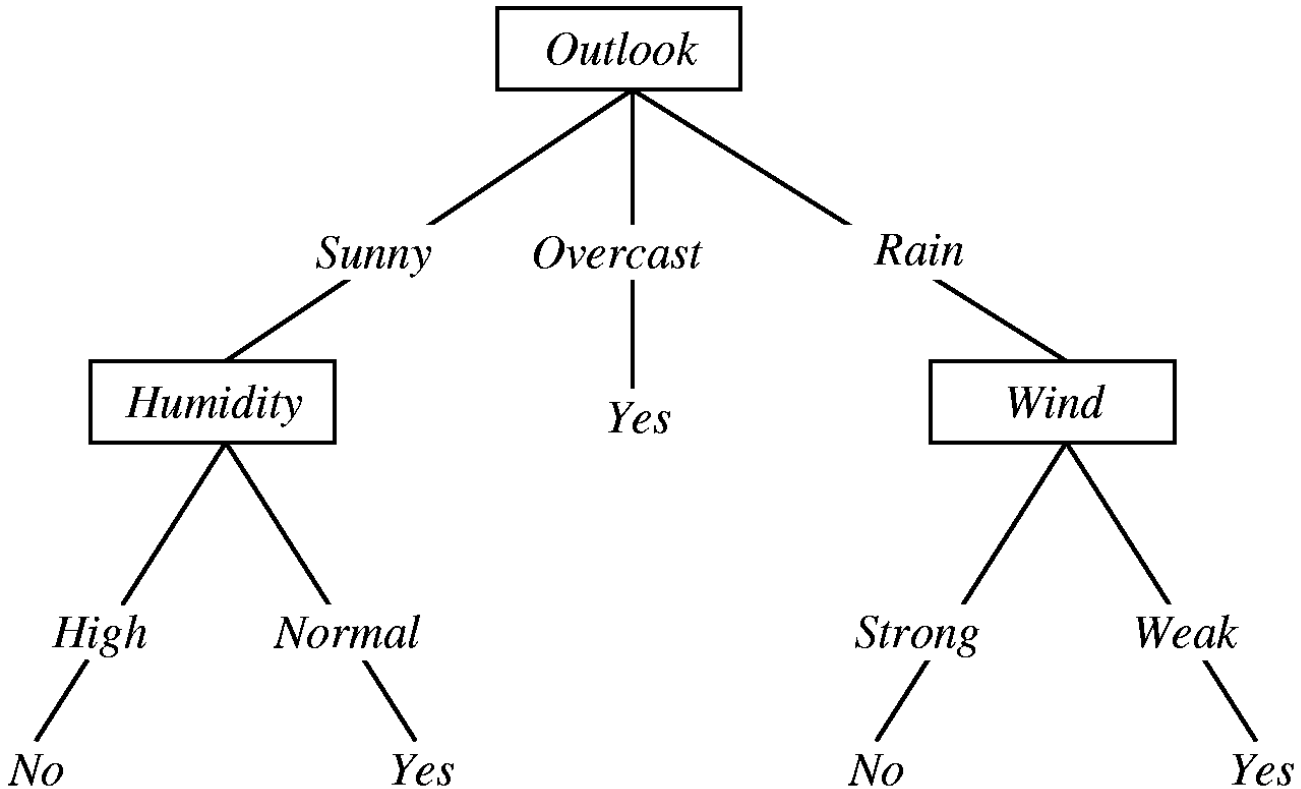
Machine Learning is used at Yandex

- search engines
- spam detection
- traffic jam prediction
- computer vision and speech recognition
- market basket analysis, customer relationship management (CRM), churn prediction
- advertisement systems
- recommendation systems
- news clustering
- translation systems / natural language processing

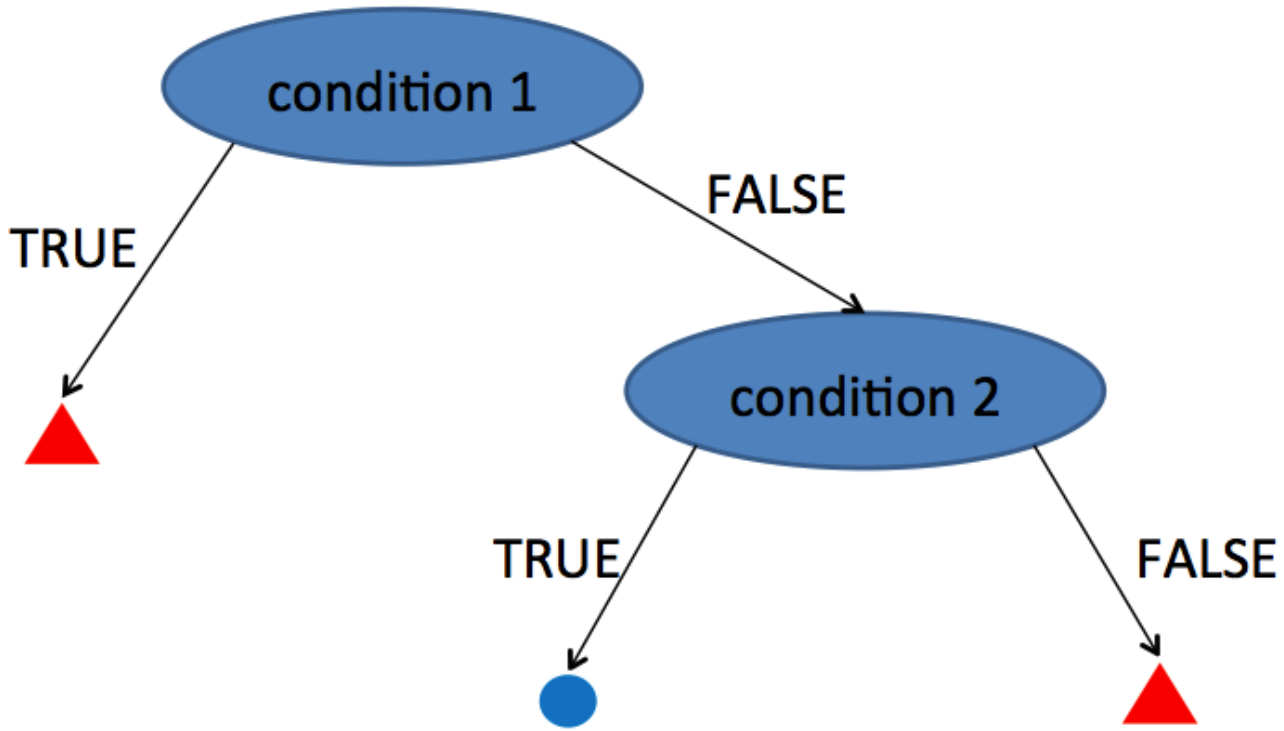
and has lots of applications outside ...

Decision tree

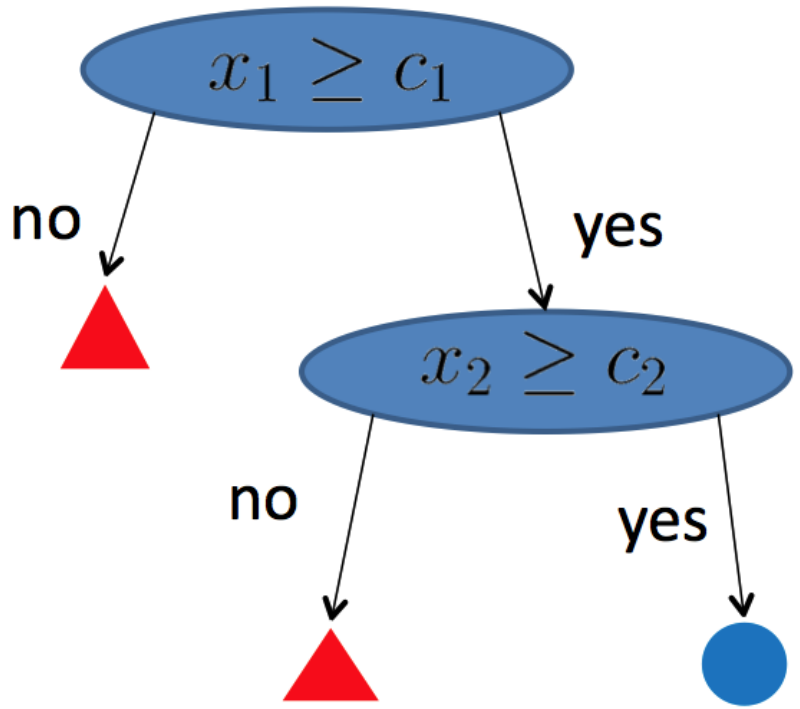
Example: predict outside play based on weather conditions.

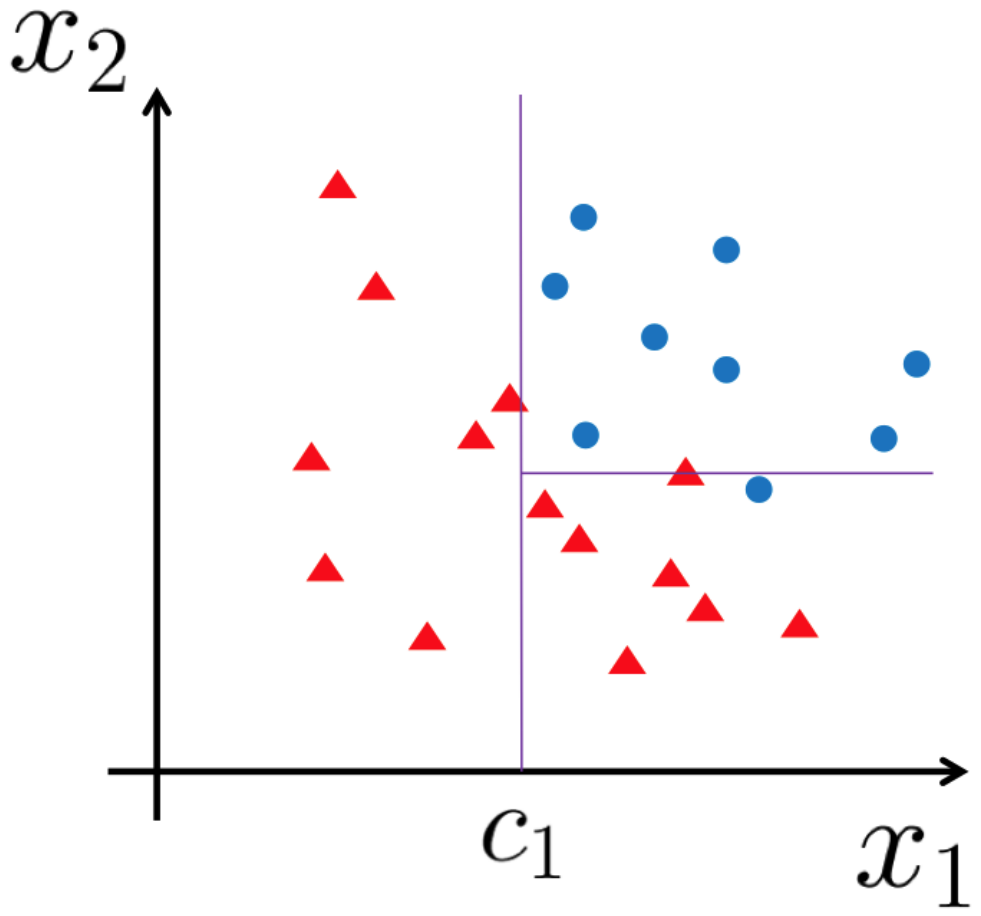


Decision tree: binary tree



Decision tree: splitting space





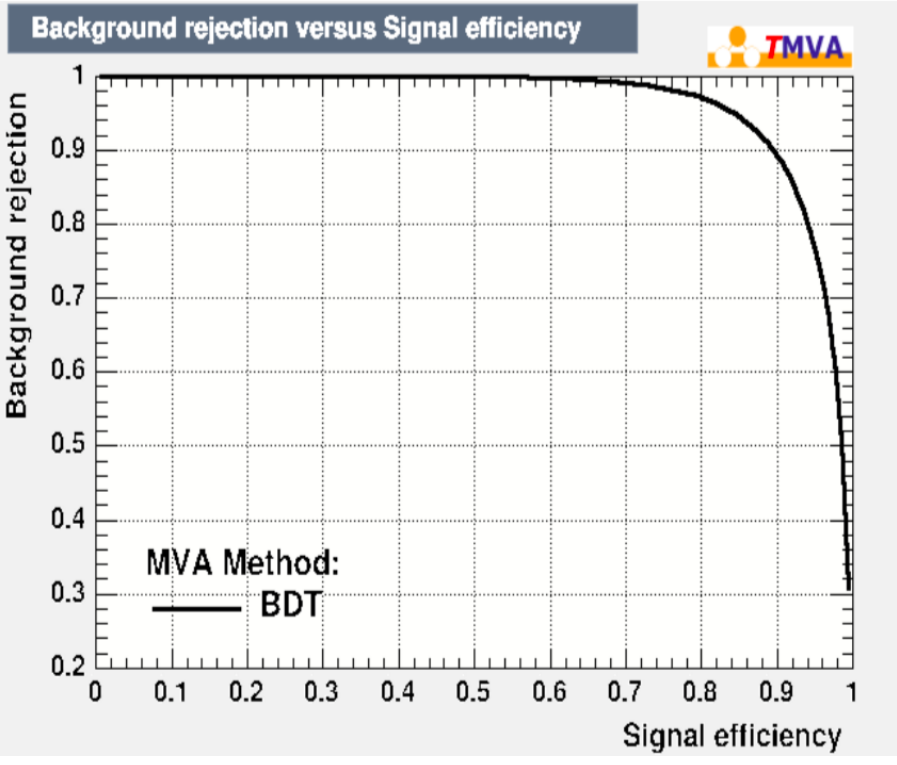
GBDT explanation using 2d regression problem

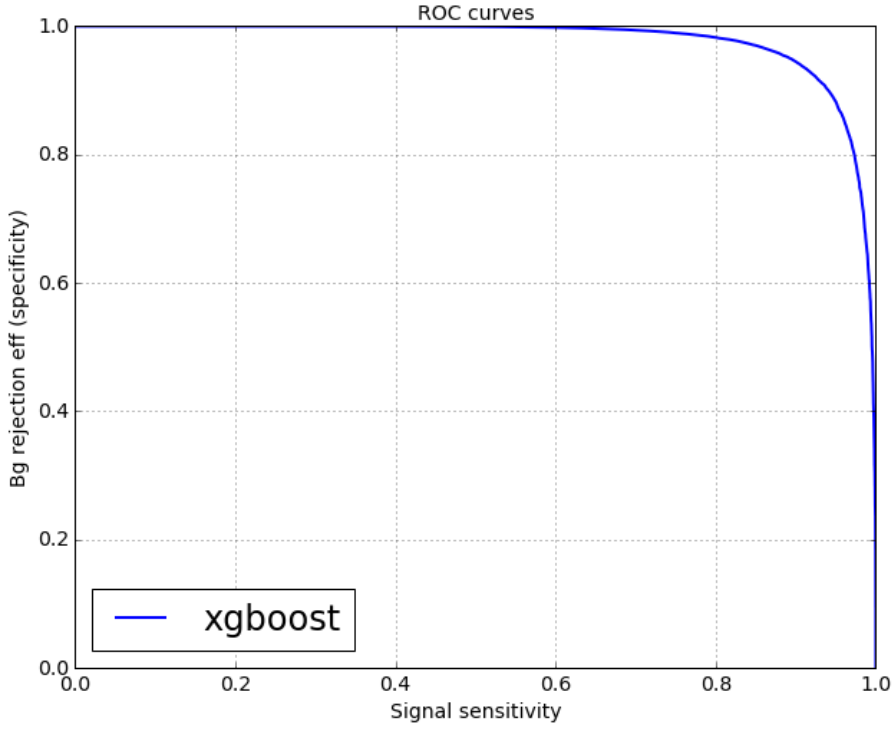
(shown interactively)

GBDT for classification

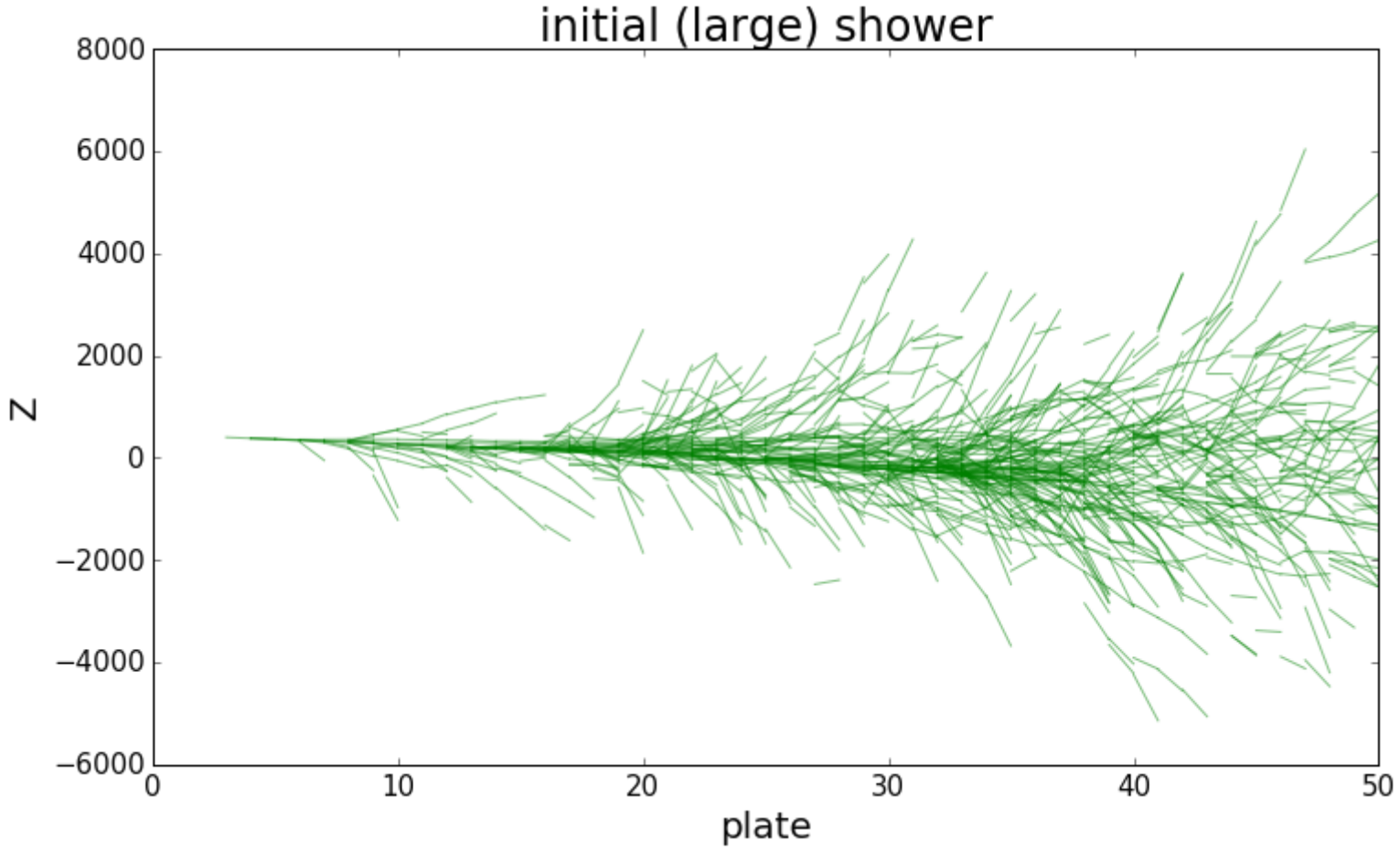
(shown interactively)

ROC curve vs original ROC from [1]



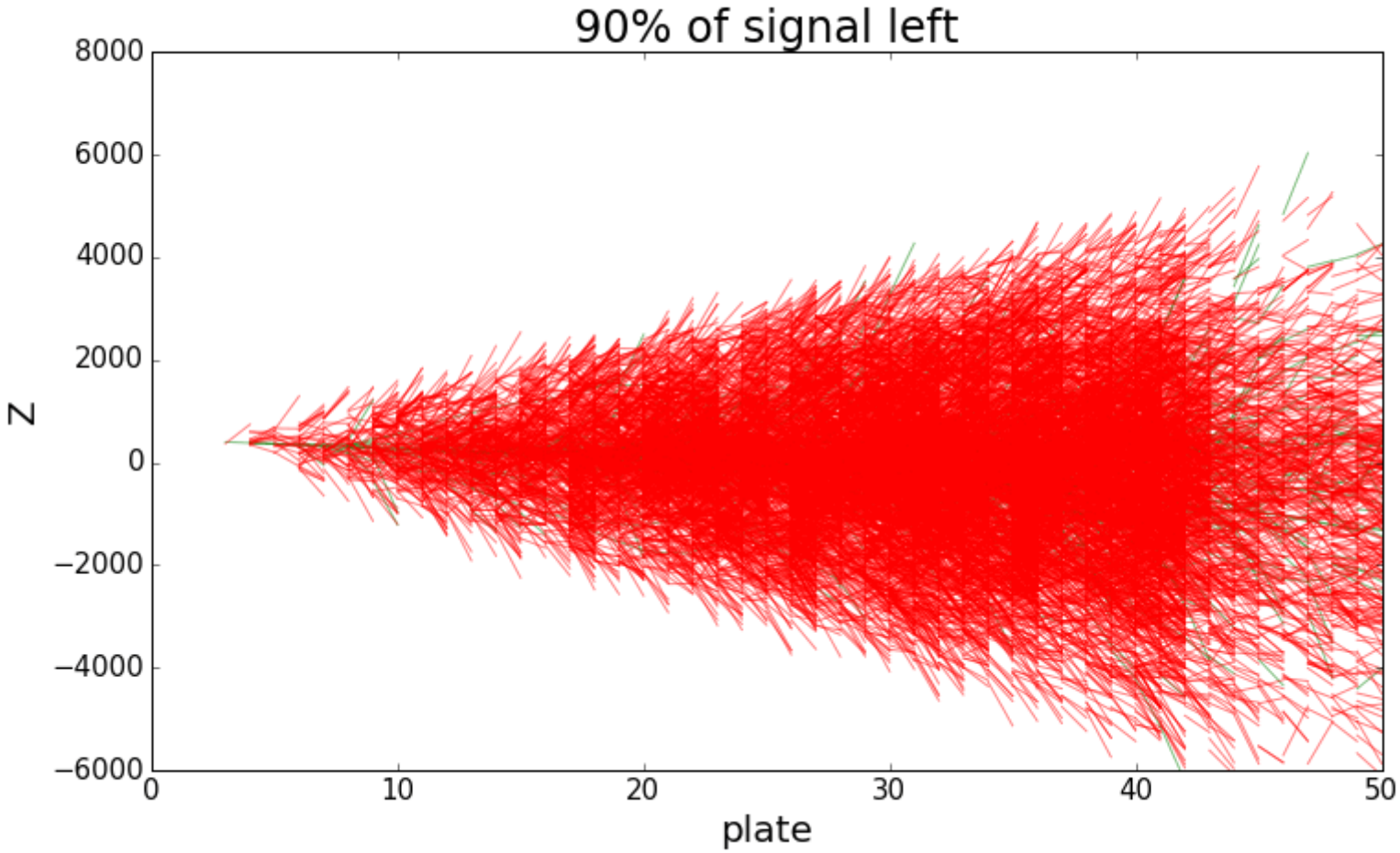


Looking at the decisions

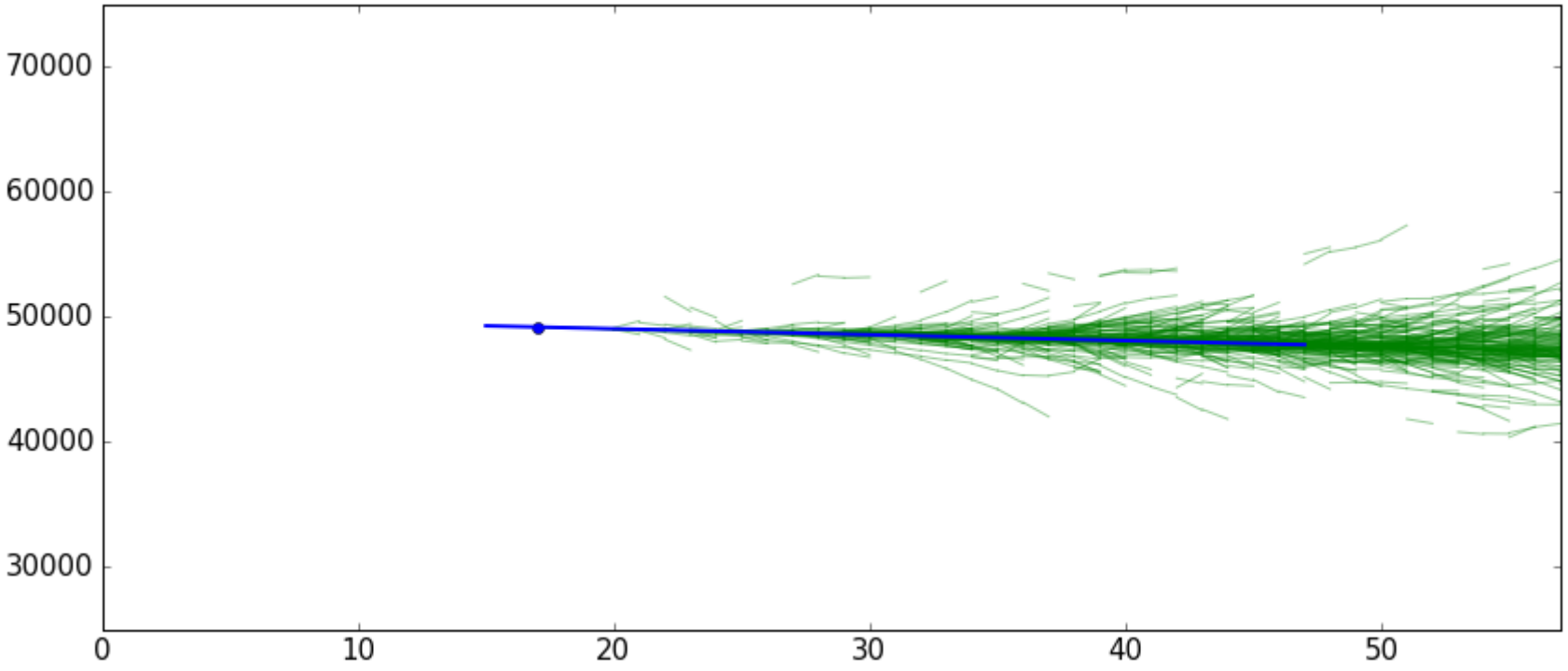


Took a large shower to see how it's geometry affected by preselections.

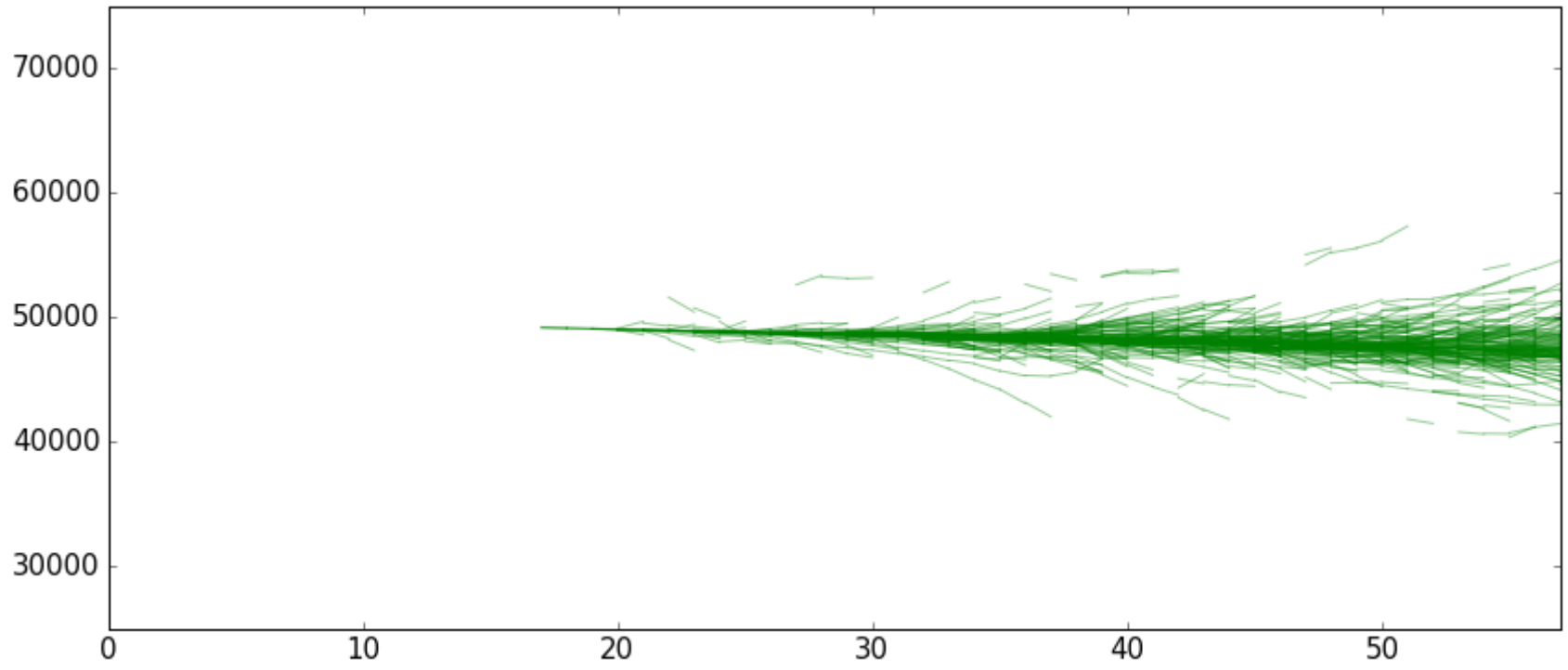
Looking at the decisions (animated)



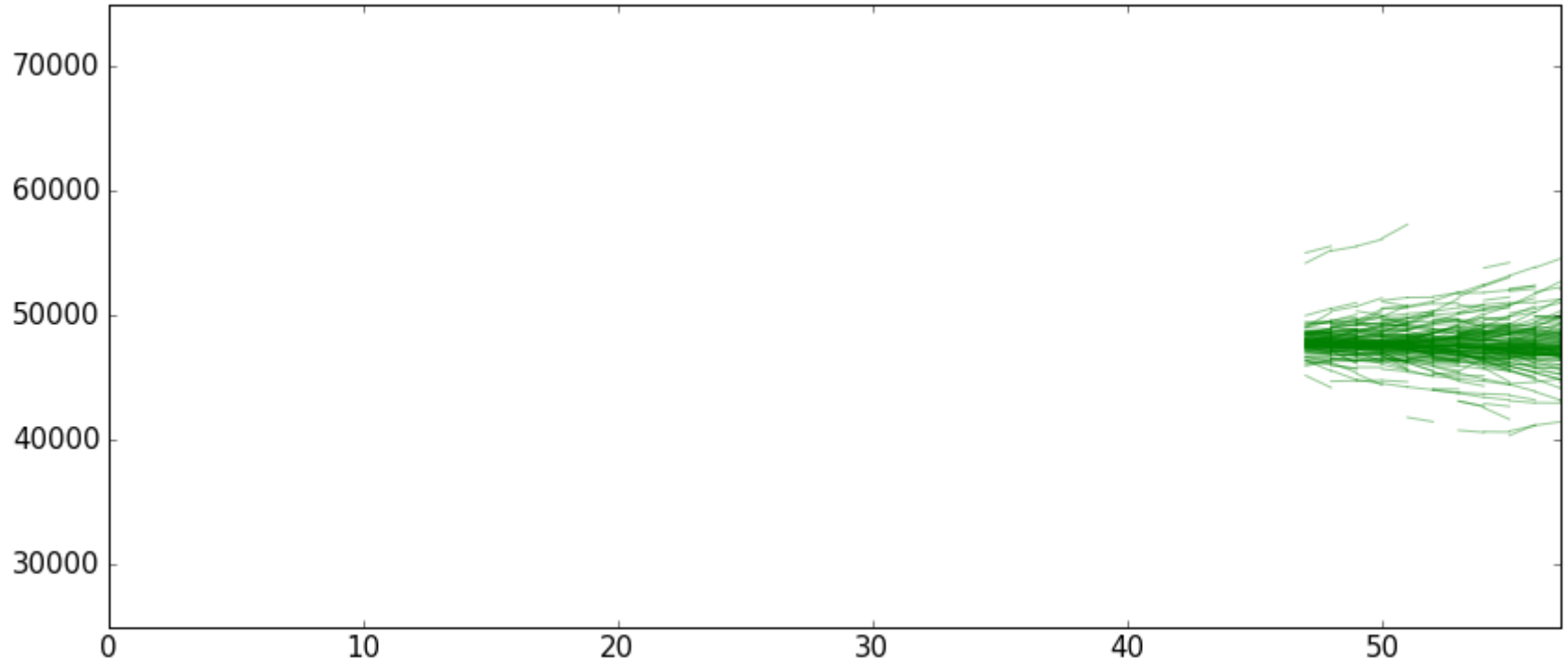
Approach "without shower position"



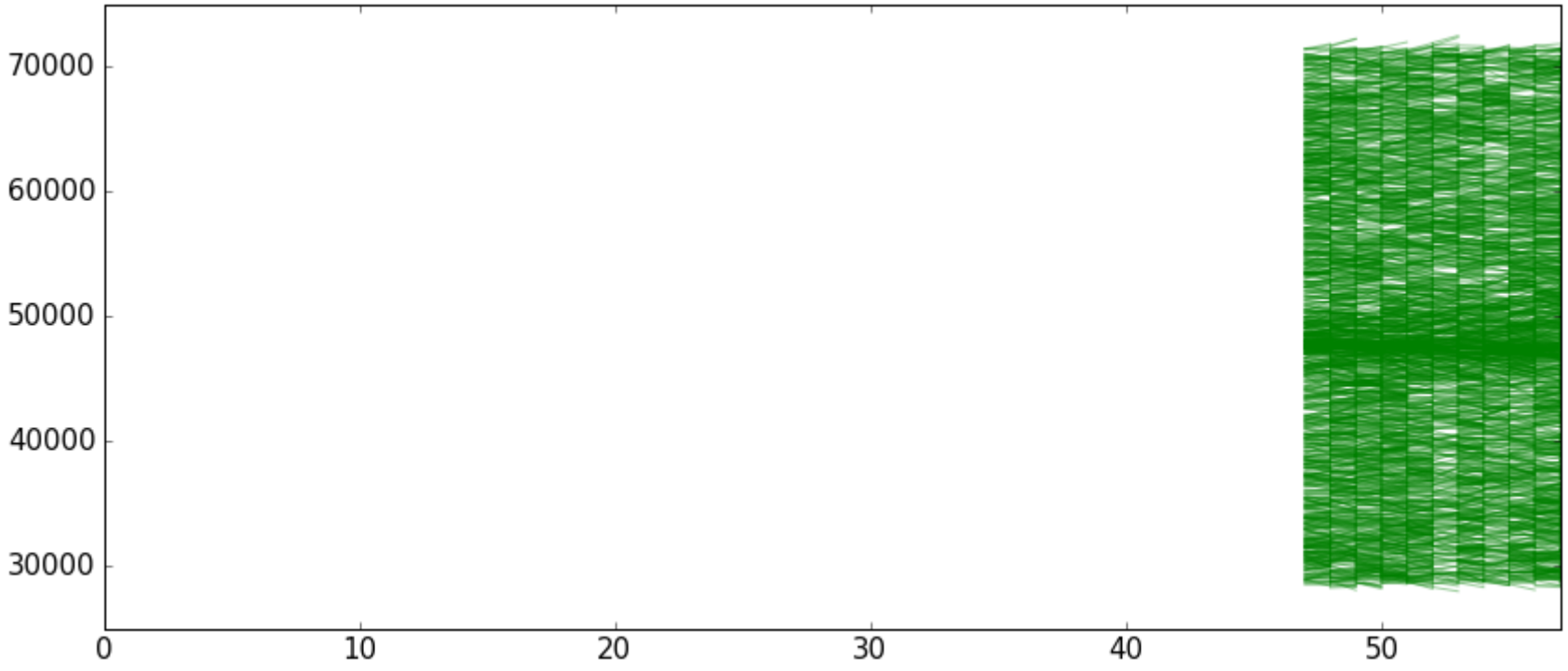
We're looking for EM-showers. Basic approach works when position and direction of the shower are provided.



But actually we may not have any information about it's source or position



... and do not have all the picture, but only part of the shower detected.



which is hidden among background ($\frac{1}{1000}$ of background events are shown on the plot)

Approach "without shower position"

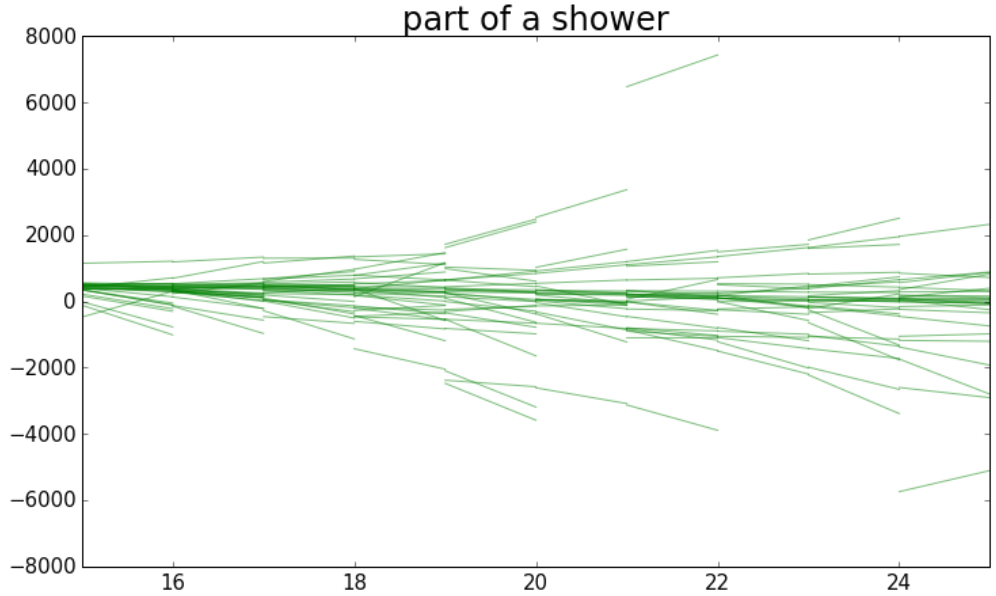
We need to filter tracks mostly based on the local information.

Solution: filter tracks based on information from *neighbouring tracks*:

- one base track on the next plate
- one base track on the previous plate

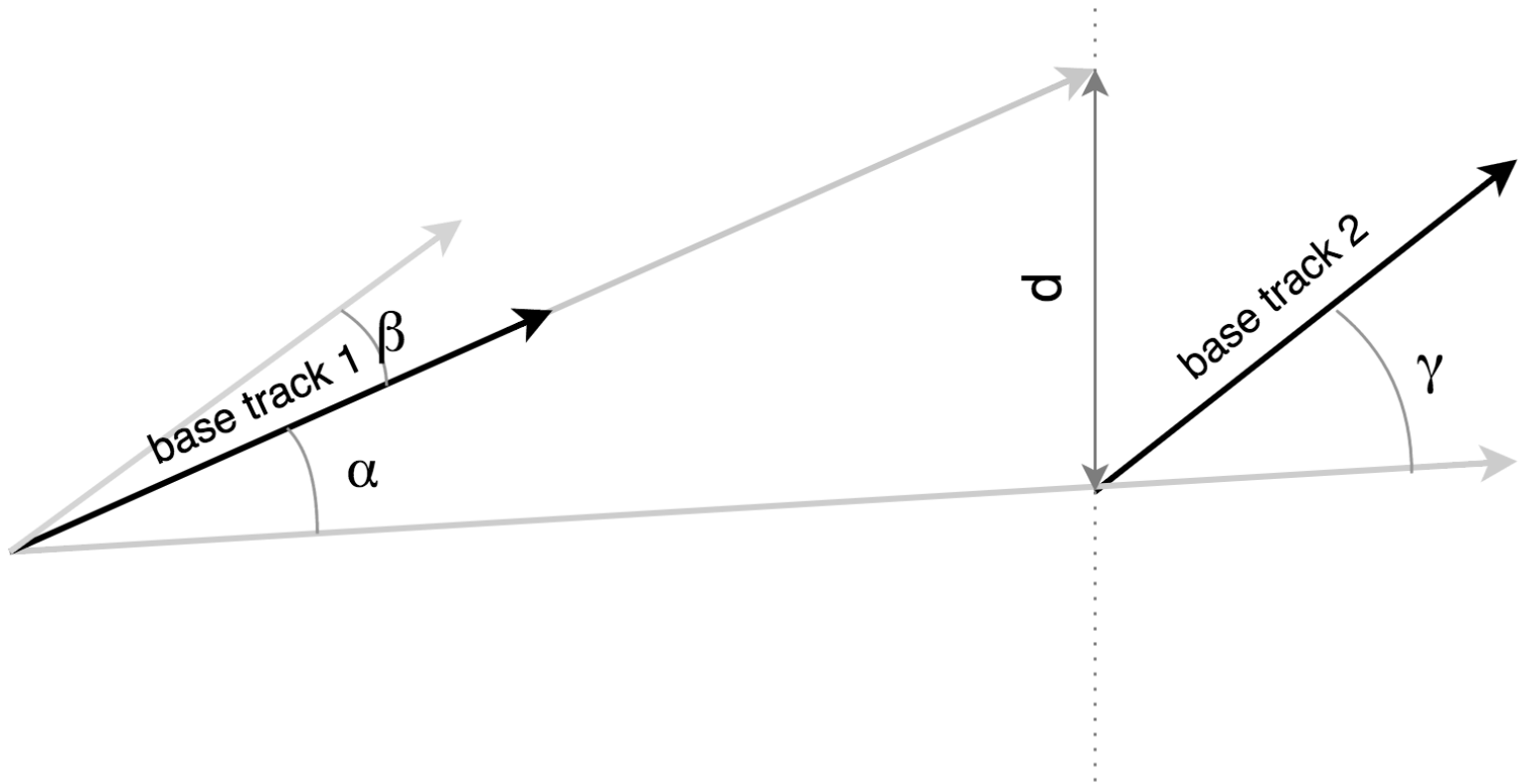
Compute:

- distance between tracks
- angle between directions
- impact parameters
- etc.



Features for new approach

remark: we don't have any information about position of mother electron
(only part of features shown)

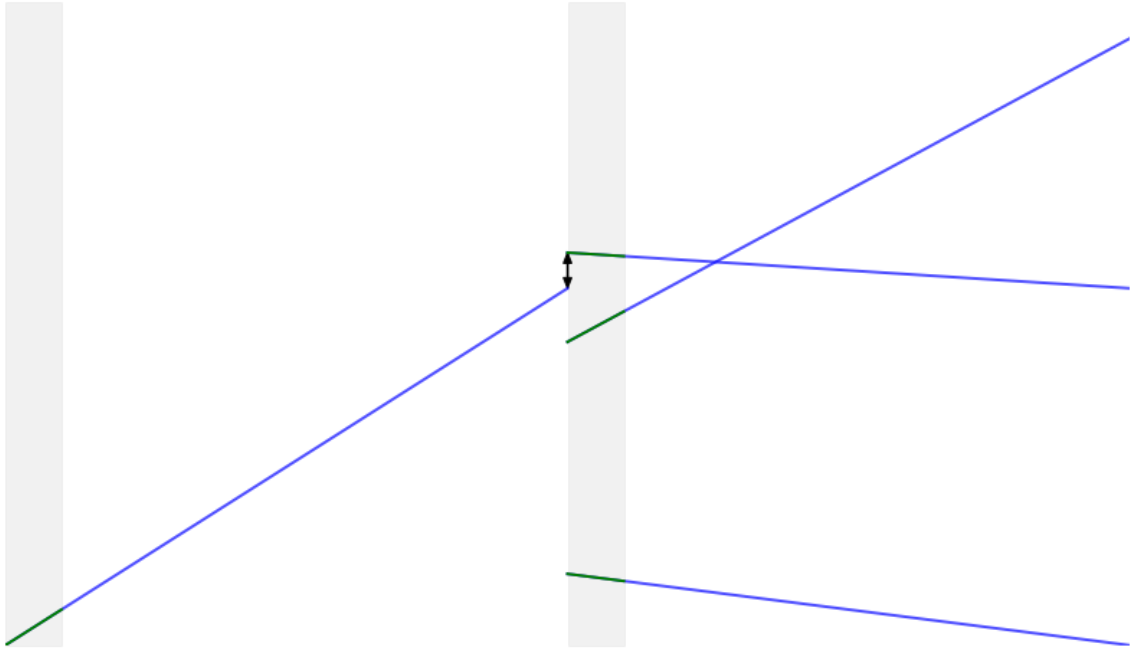


Finding most appropriate neighbouring tracks

Finding 'next track' for the base track from left plate.

Simply taking nearest neighbor is quite faulty.

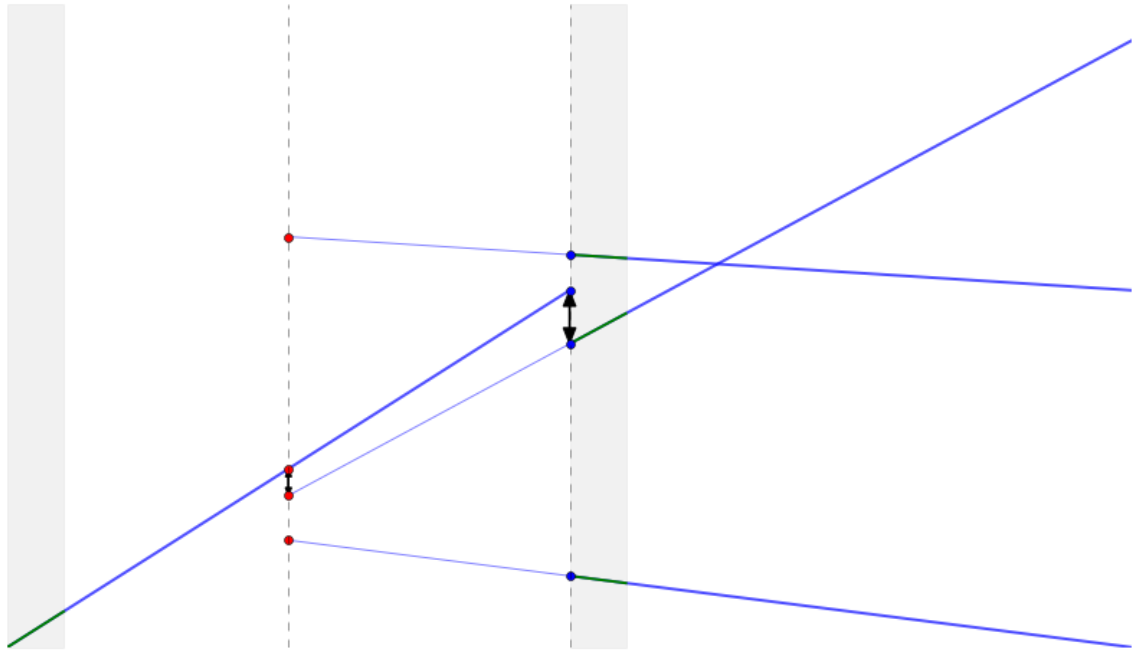
We need also to control direction of the track.



Finding most appropriate neighbour tracks

After selecting two hyperplanes we produce a pair of points, which are expected to be close if tracks are on the same line. This is considered as mapping to 6-dimensional space.

Positions of hyperplanes are subject to optimization.



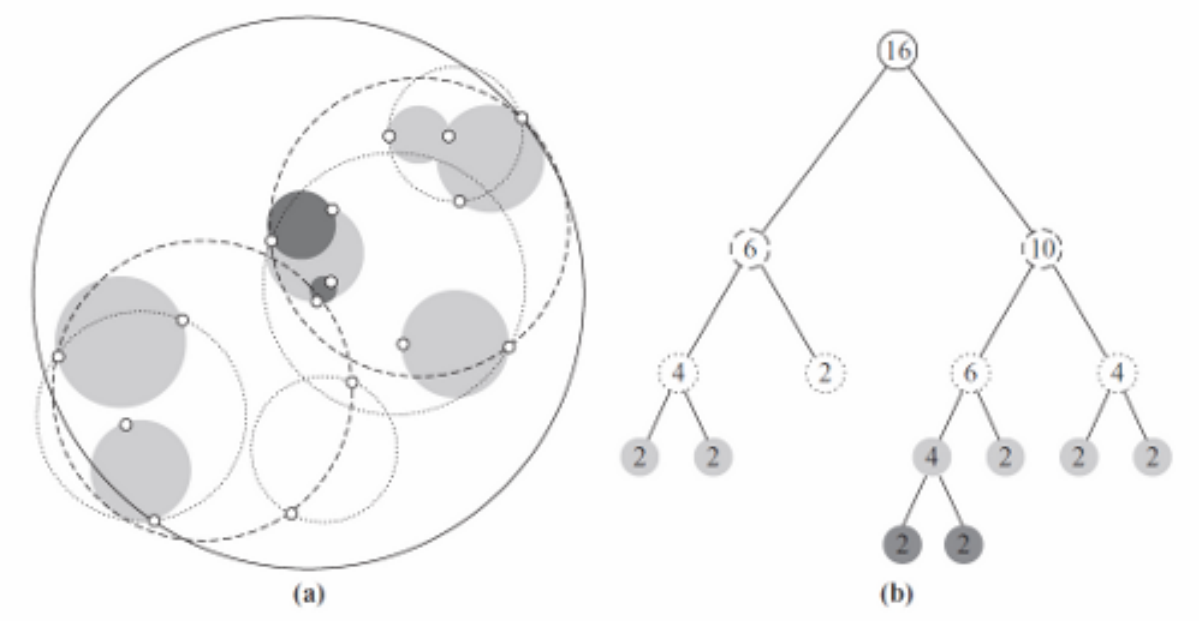
Fast search for nearest neighbors

Brute force solution takes
$N_{samples} \times N_{samples} \times N_{dimensions}$.

(impossible for $N_{samples} \sim 10^7$)

Better options exist:

- Ball tree
- KD-tree



(BallTree scheme)

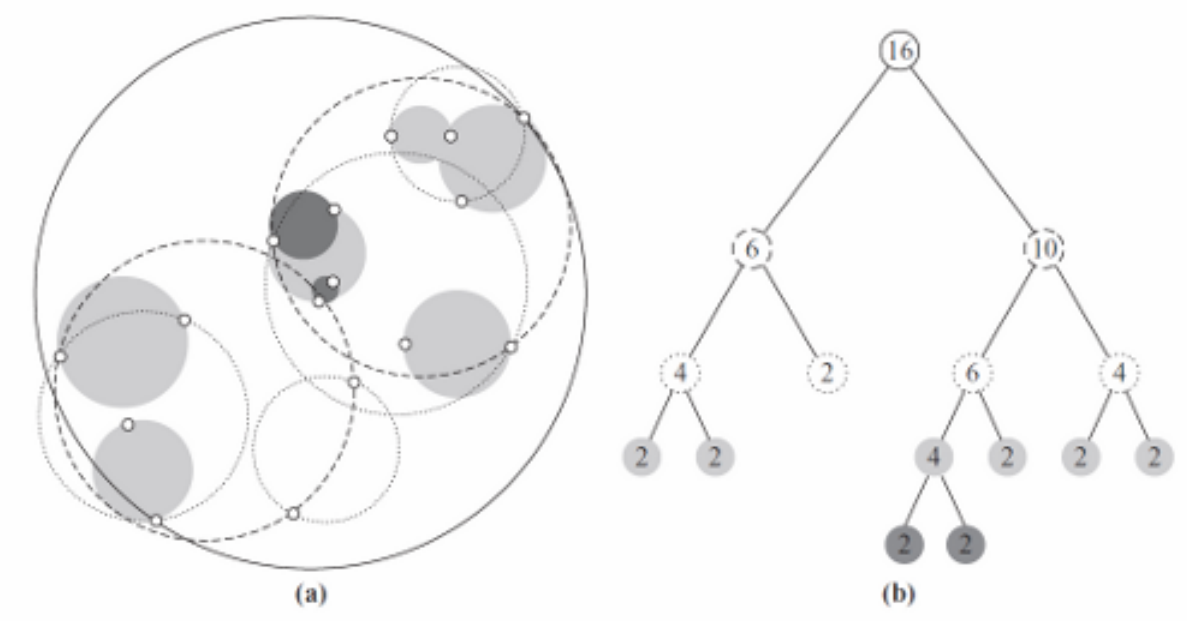
Fast search for nearest neighbors

Brute force solution takes
$N_{samples} \times N_{samples} \times N_{dimensions}$.

(impossible for $N_{samples} \sim 10^7$)

Better options exist:

- Ball tree
- KD-tree

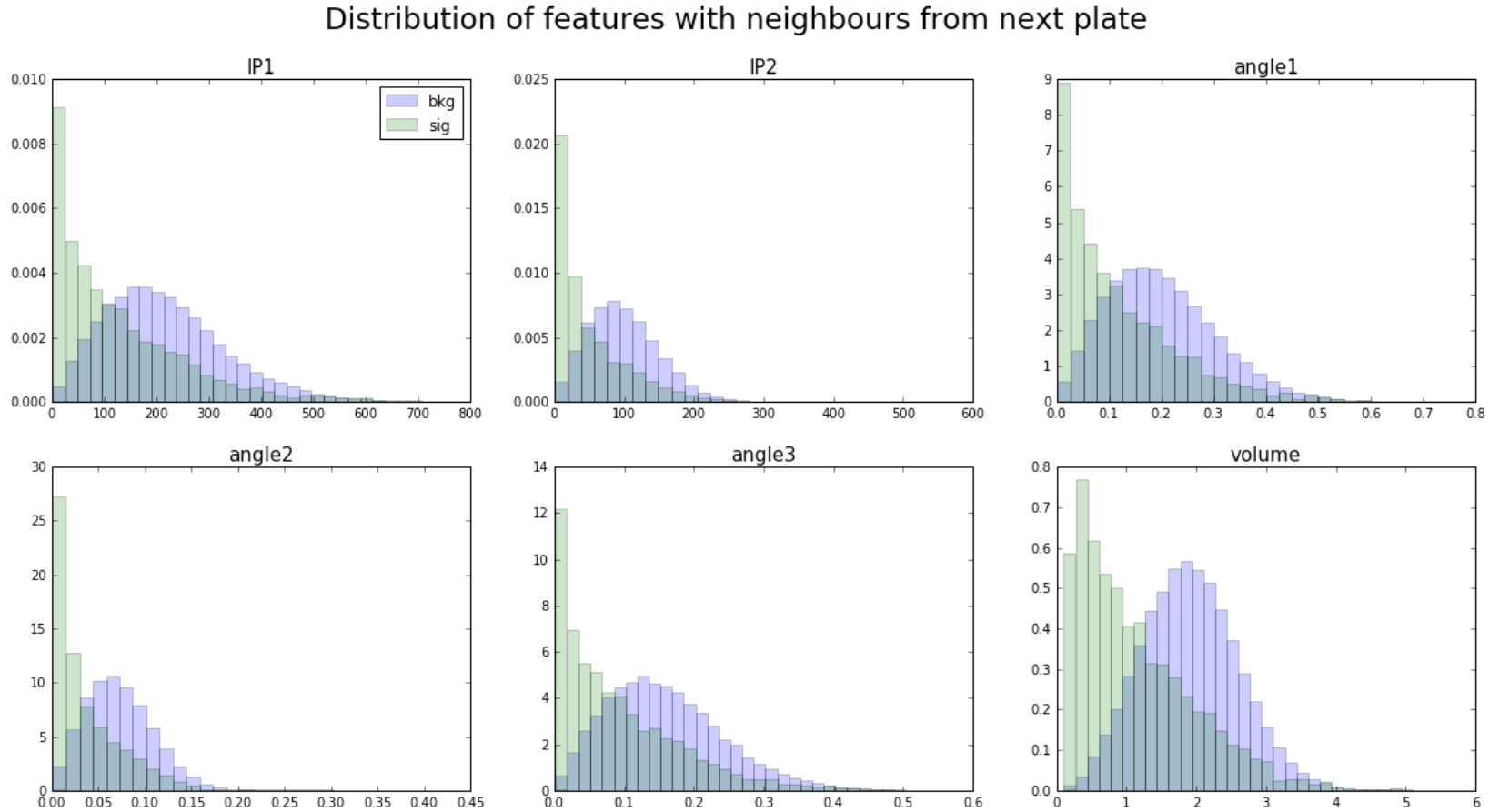


(BallTree scheme)

Even KD-tree wasn't fast enough, I had to write custom method based on KD-tree.

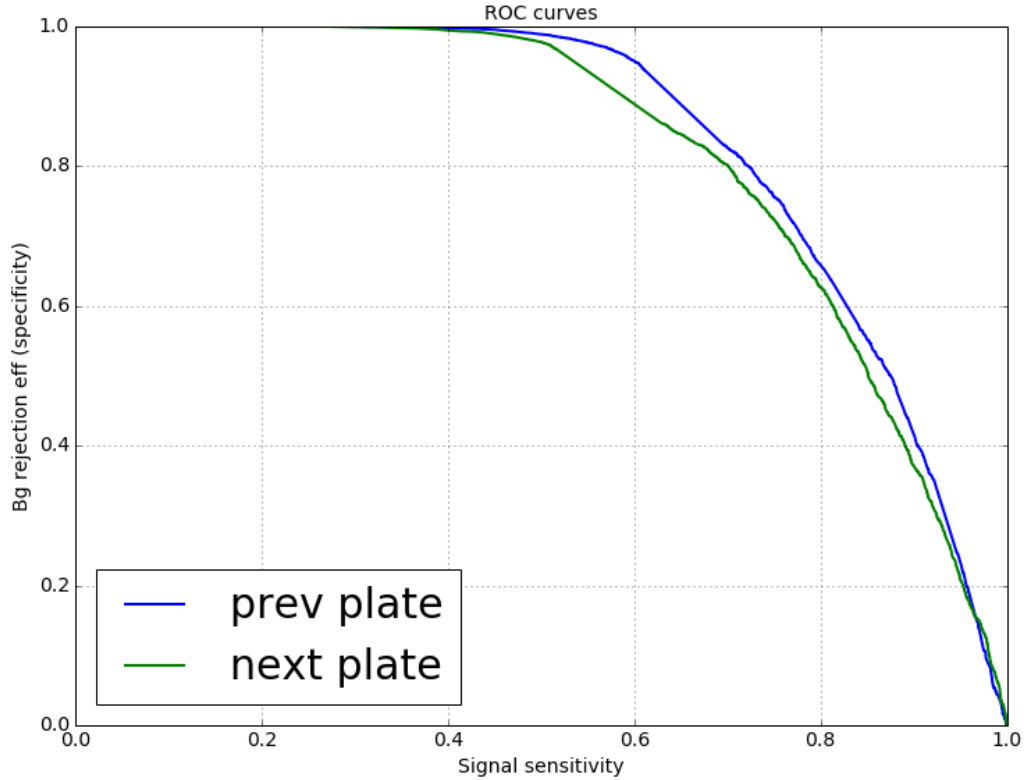
Distribution of some of the features:

(assuming that neighboring track exists)



ROC curve

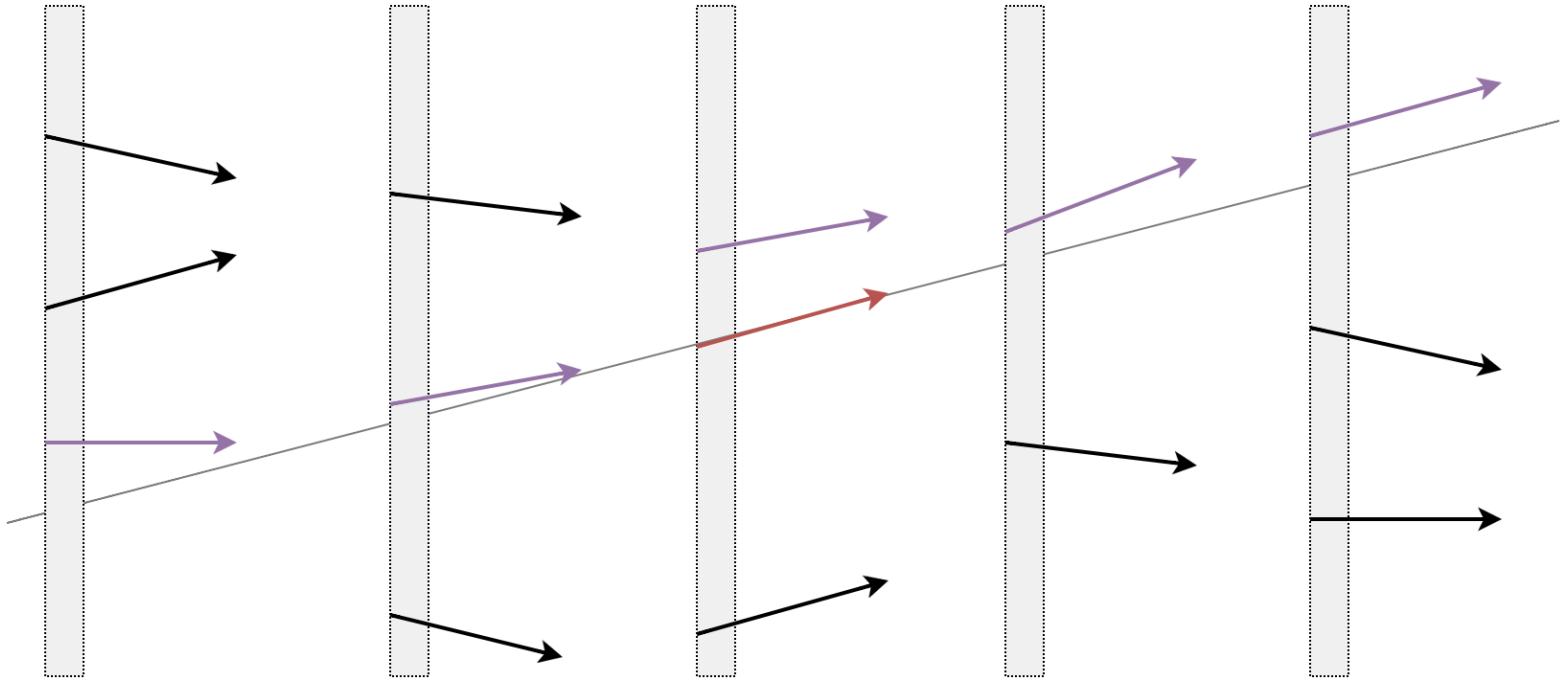
based on the information from the closest neighbouring base tracks



Improving guess-without-shower-position

This quality is still not enough, but we can analyze wider region

- taking base track from plates shifted on [-2, -1, 0, +1, +2]
- taking prediction of previous classifier as features (input) at this step



Scheme of classification: overview

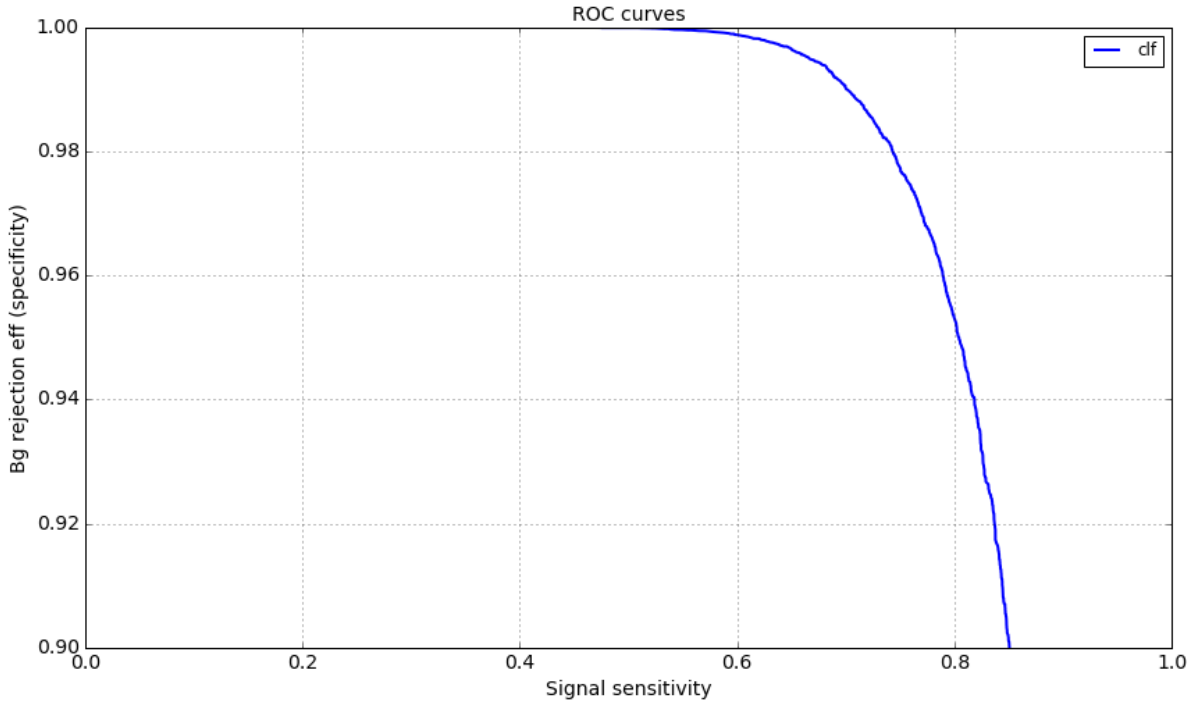
1. for each base track find neighboring tracks from [-1, +1] plates
   - using described approach to select neighbours
   - feature engineering using geometry
   - predict whether each base track looks like signal with ML
2. for each base track collect 5 neighbors from plates with shifts [-2, -1, 0, +1, +2]
   - using the same approach to select neighbours
   - use information about base tracks collected on previous step
     - next track of next tracks $\neq$ neighbour track from after-next layer
     - allows some bending of trajectory
   - add some geometry-based feature engineering
   - use another ML model on the top of this

After this magic ...

we got not-that-bad classification.

(showing part of ROC curve over y here)

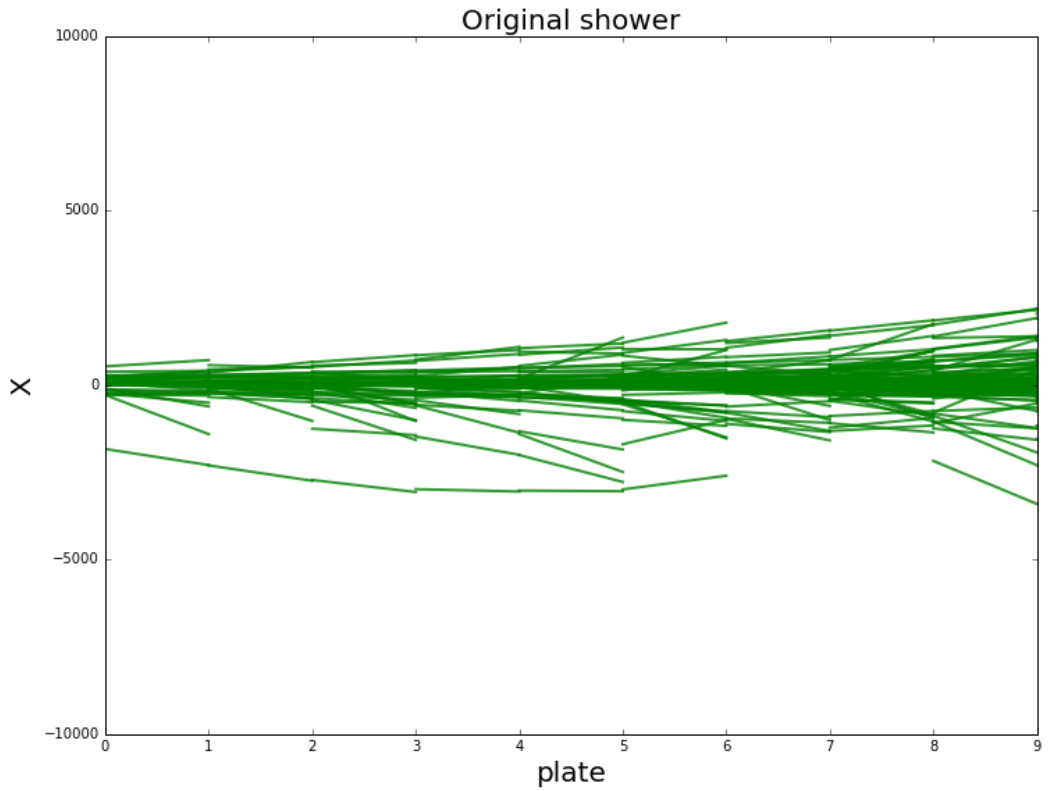
Still we used no guess about shower position ...



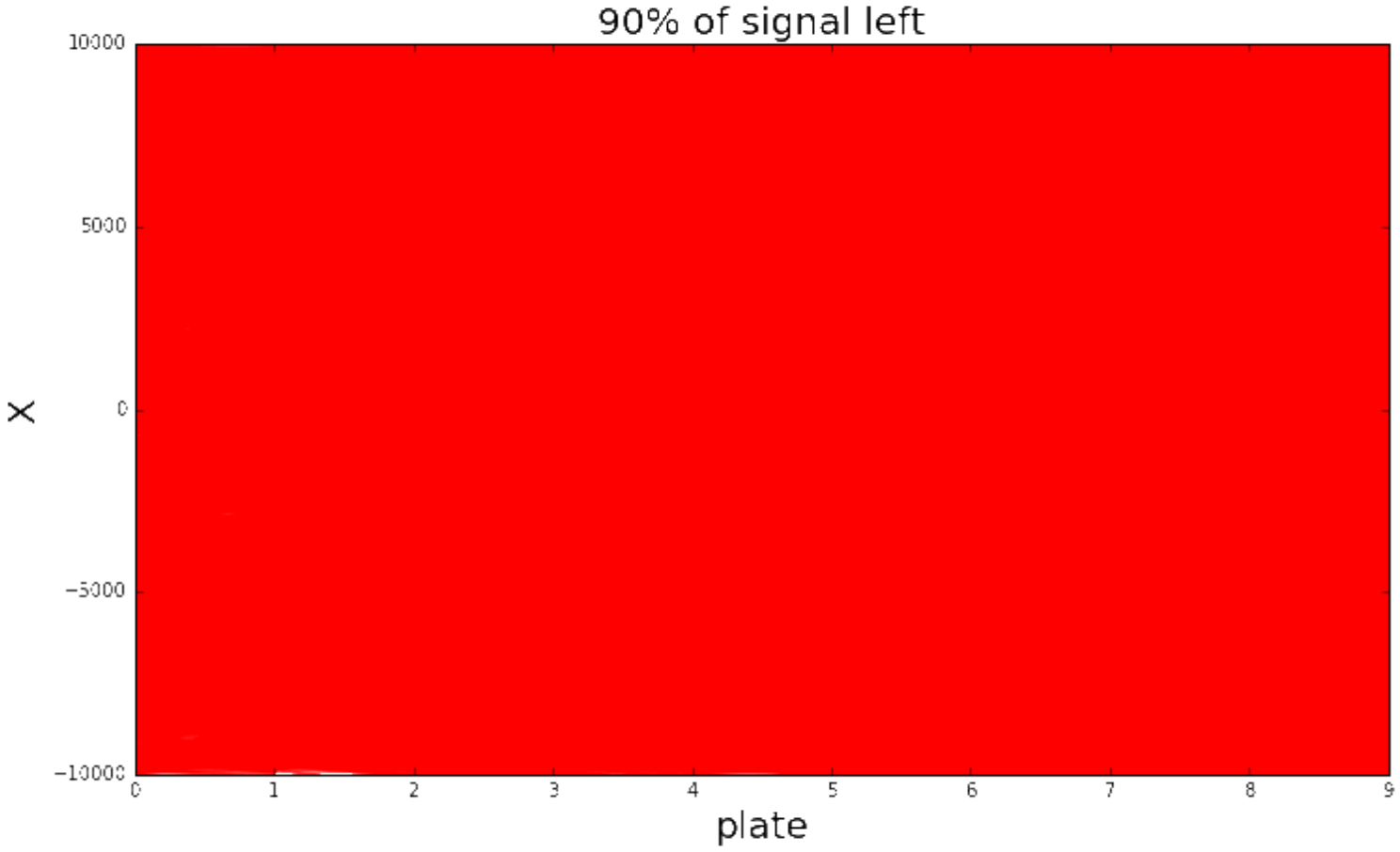
Looking at decisions of new approach

Again, take some large shower to see how classification rule works

This time we have only part of a brick (10 layers)



Looking at decisions of new approach (animated)



Finding and optimizing track position

Assuming we want to test if a line l looks like approximate direction of a shower, we can introduce some simple figure of merit:

$$\text{line quality} = \sum_{\text{tracks}} \exp\left[-\frac{\rho(l, \text{track})}{\sigma_\rho^2} - \frac{1 - cos(l, \text{track})}{\sigma_\alpha^2}\right]$$

- approximately corresponds to number of tracks close to line and having close direction
- smooth function, thus gradients can be used in optimization

Detecting a shower

After tracks are evaluated with 2-stage pipeline, those are filtered using some threshold.

Using selected tracks I find a line with maximal line quality

Possible approaches:

1. take only track lines as candidates for lines
2. convert pair of tracks to line passing through their Positions
3. global optimization
4. guesses from 1. or 2. , but refined with continuous optimization

Detecting a shower

After tracks are evaluated with 2-stage pipeline, those are filtered using some threshold.

Using selected tracks I find a line with maximal line quality

Possible approaches:

1. take only track lines as candidates for lines
2. convert pair of tracks to line passing through their Positions
3. global optimization
4. guesses from 1. or 2. , but refined with continuous optimization

First approach gives best results, because others frequently align random background tracks.

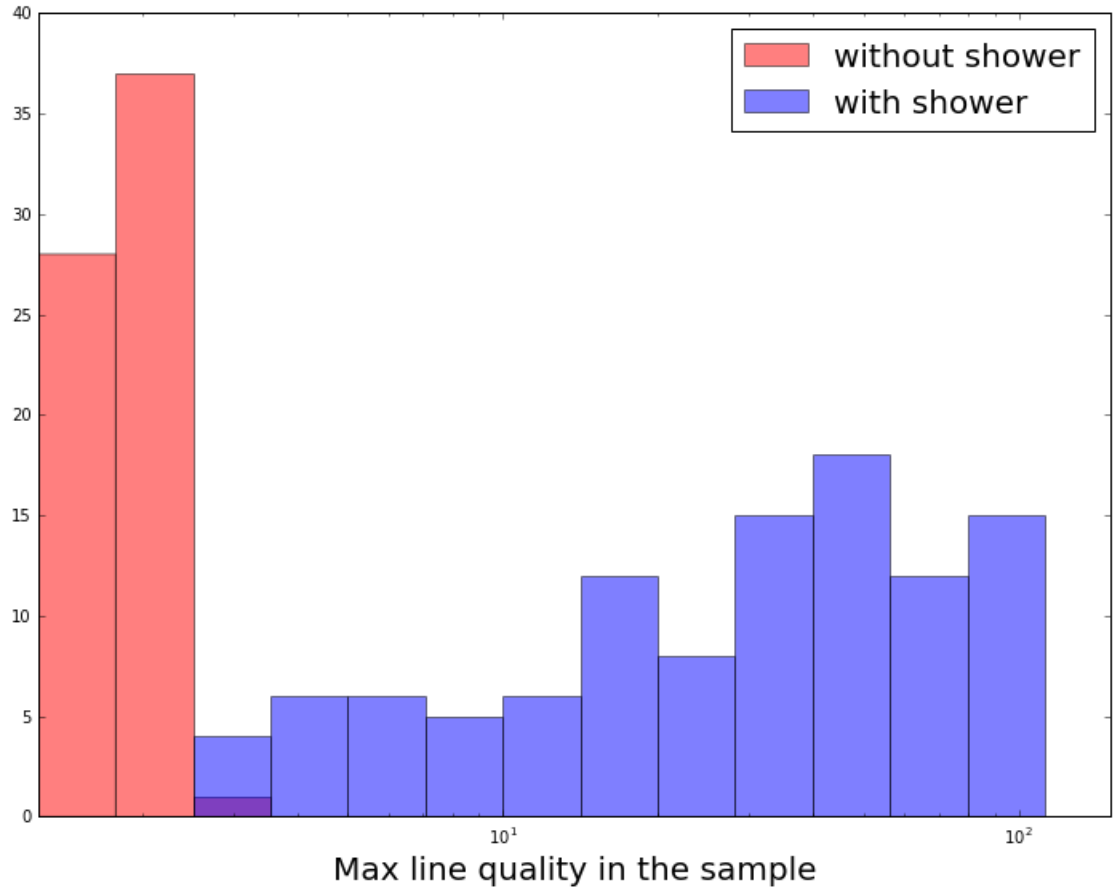
Distribution of line quality

For a sample we can analyze the largest of line qualities

In simple tests I've generated

- 10 plates
- shower begins 10-20 plates earlier available region
- shower left > 40 tracks in this region

I can separate 80-90% of showers from instrumental background.



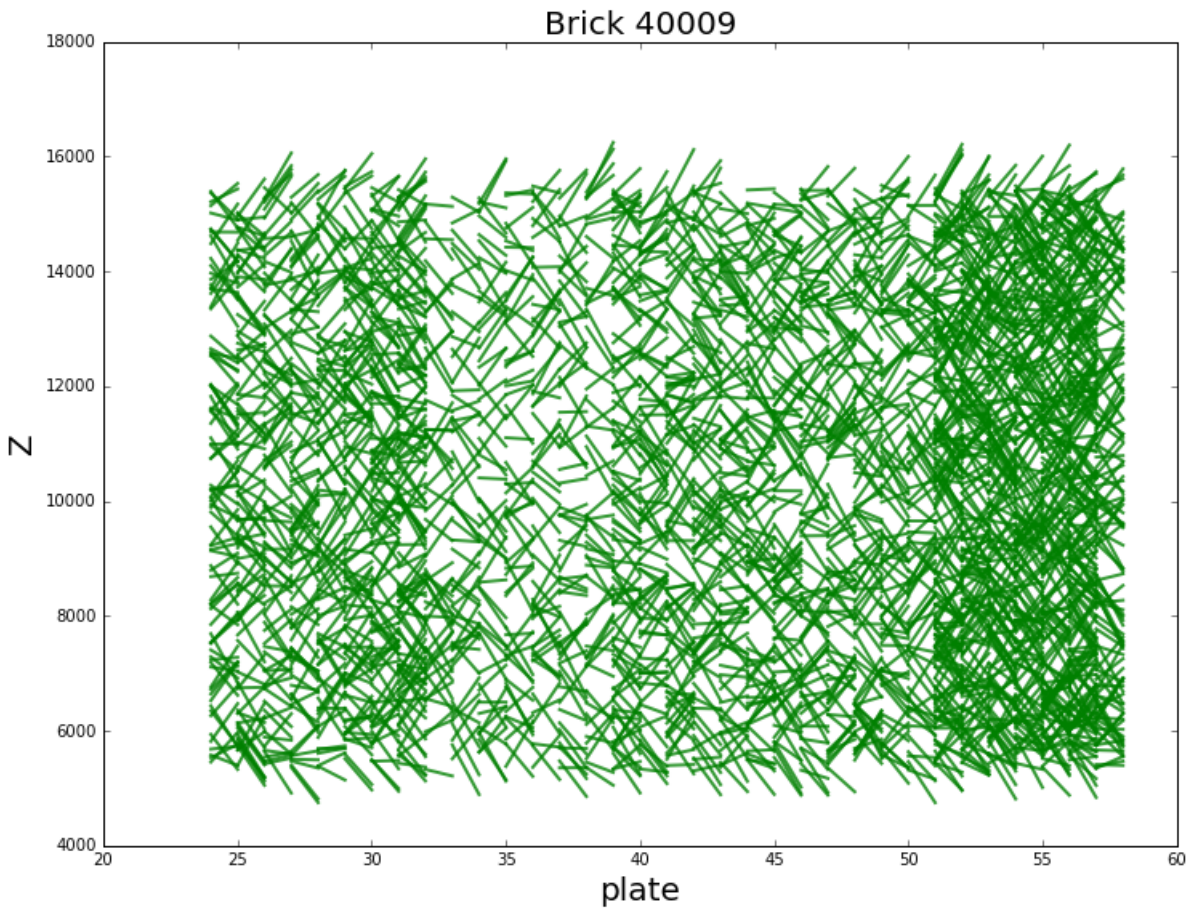
Brick 40009

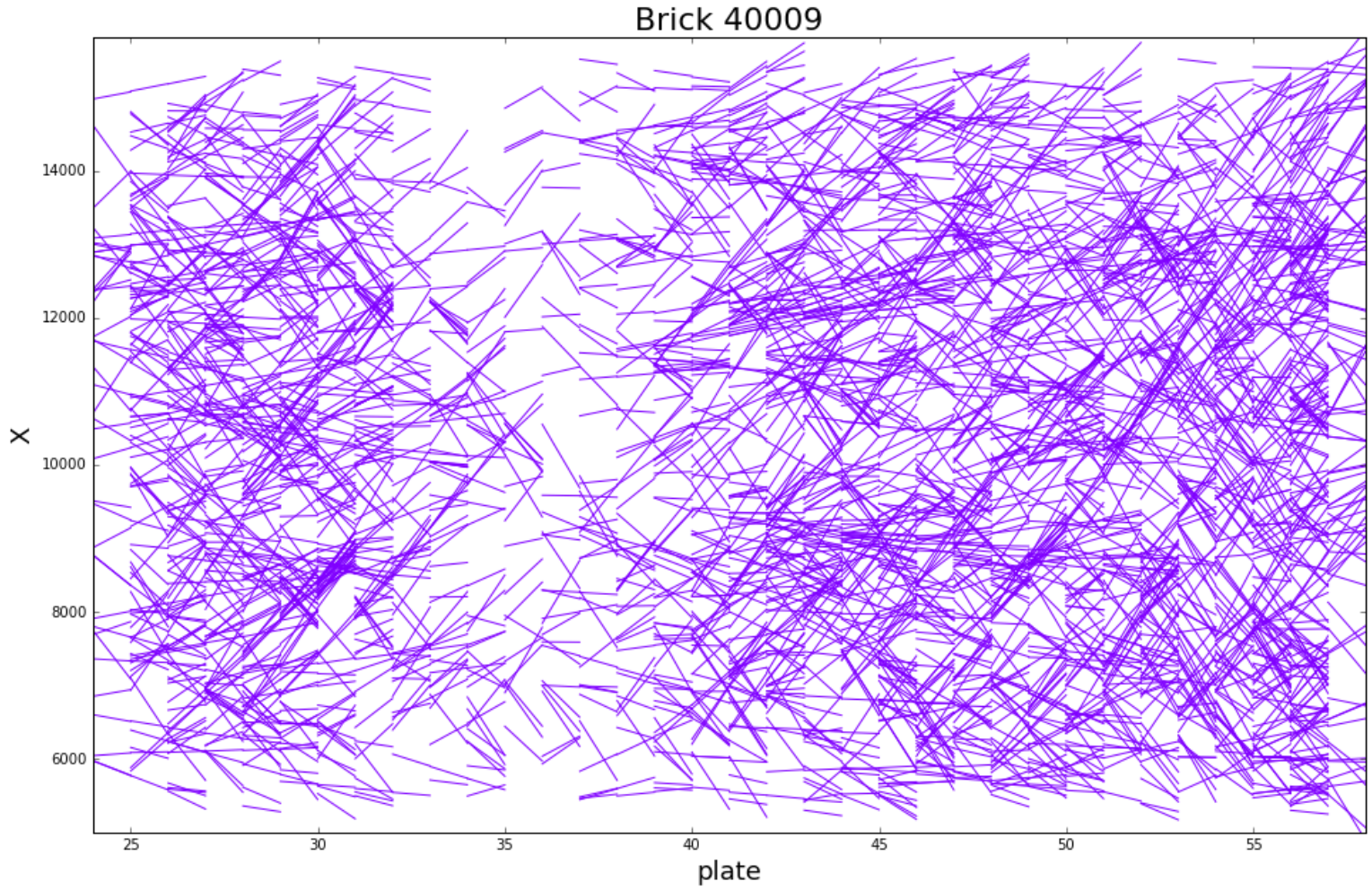
the rest will be shown using brick 40009 as a demonstration

Distribution of tracks

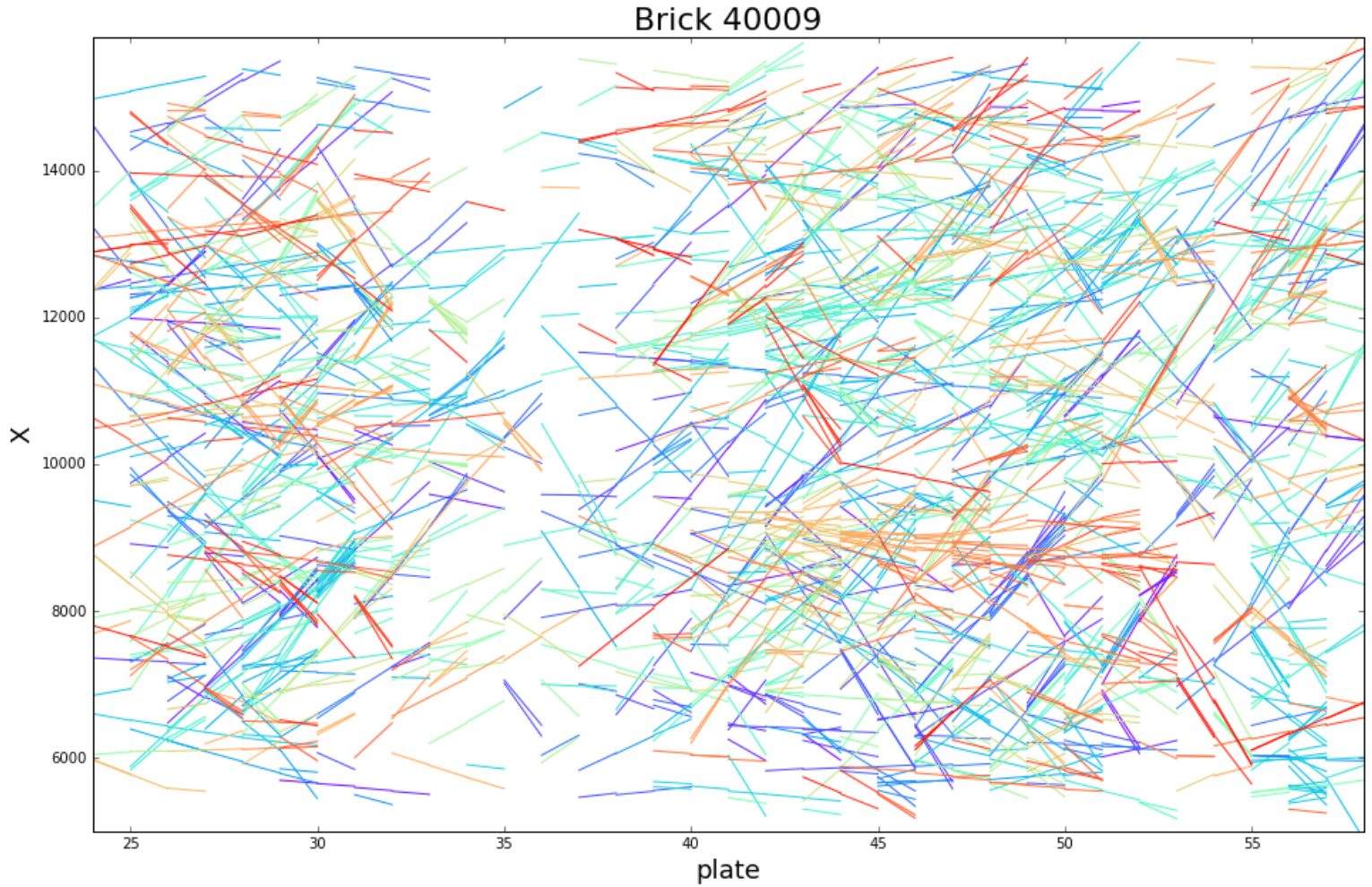
(1 / 1000 of all tracks shown on the plot)

- contains many layers, which is nice
- obvious non-uniformity in plates ...
  - ??
- very strange peaks in slope SX, SY
  - ???
- ok, let's try anyway

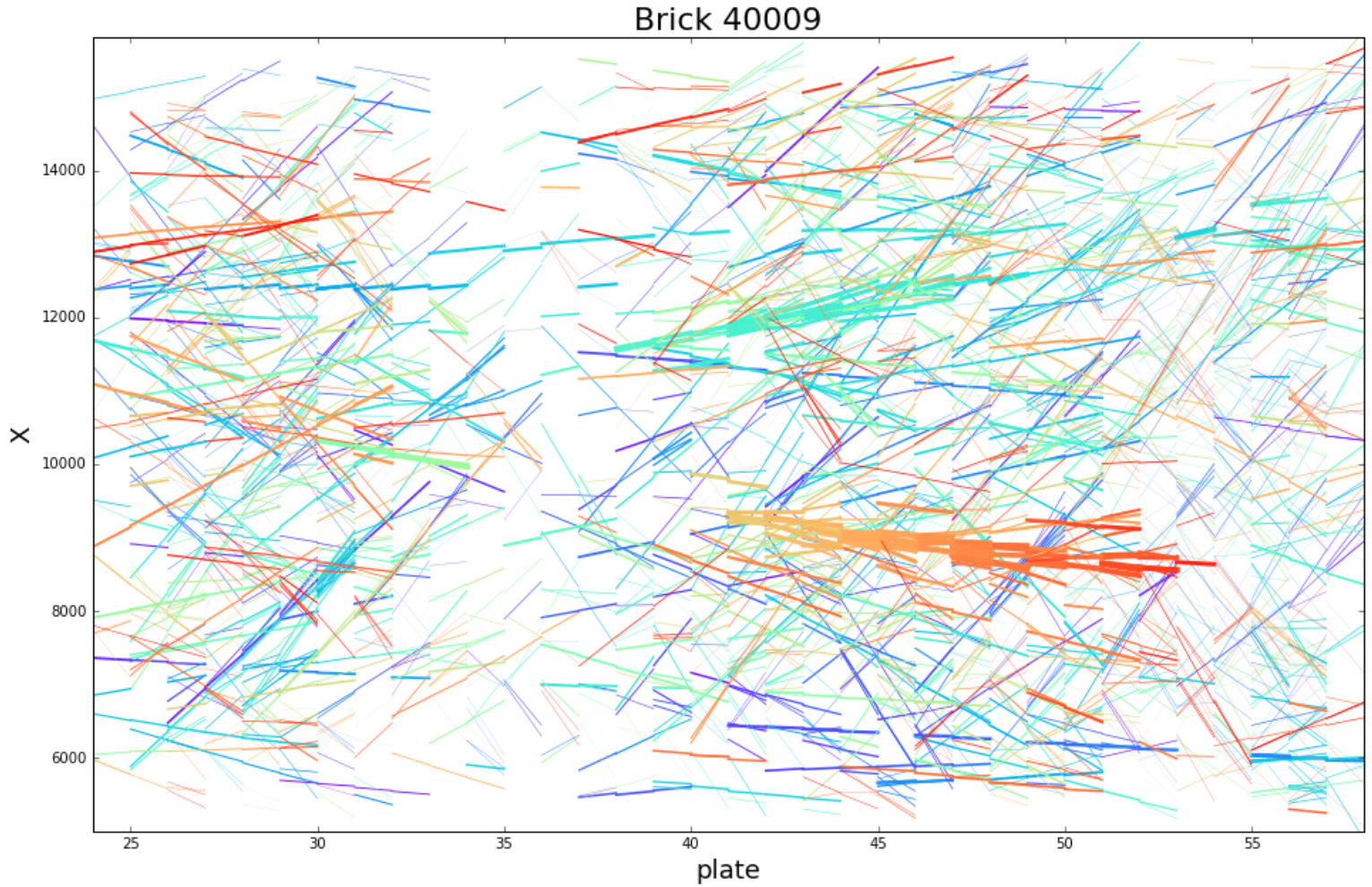




Algorithm selected ~4 000 tracks from 3 500 000
(it was trained only against instrumental background).



For each track we can investigate its line and compute for each line it's line quality
I have encoded information about Y in color and used preselection on line quality.

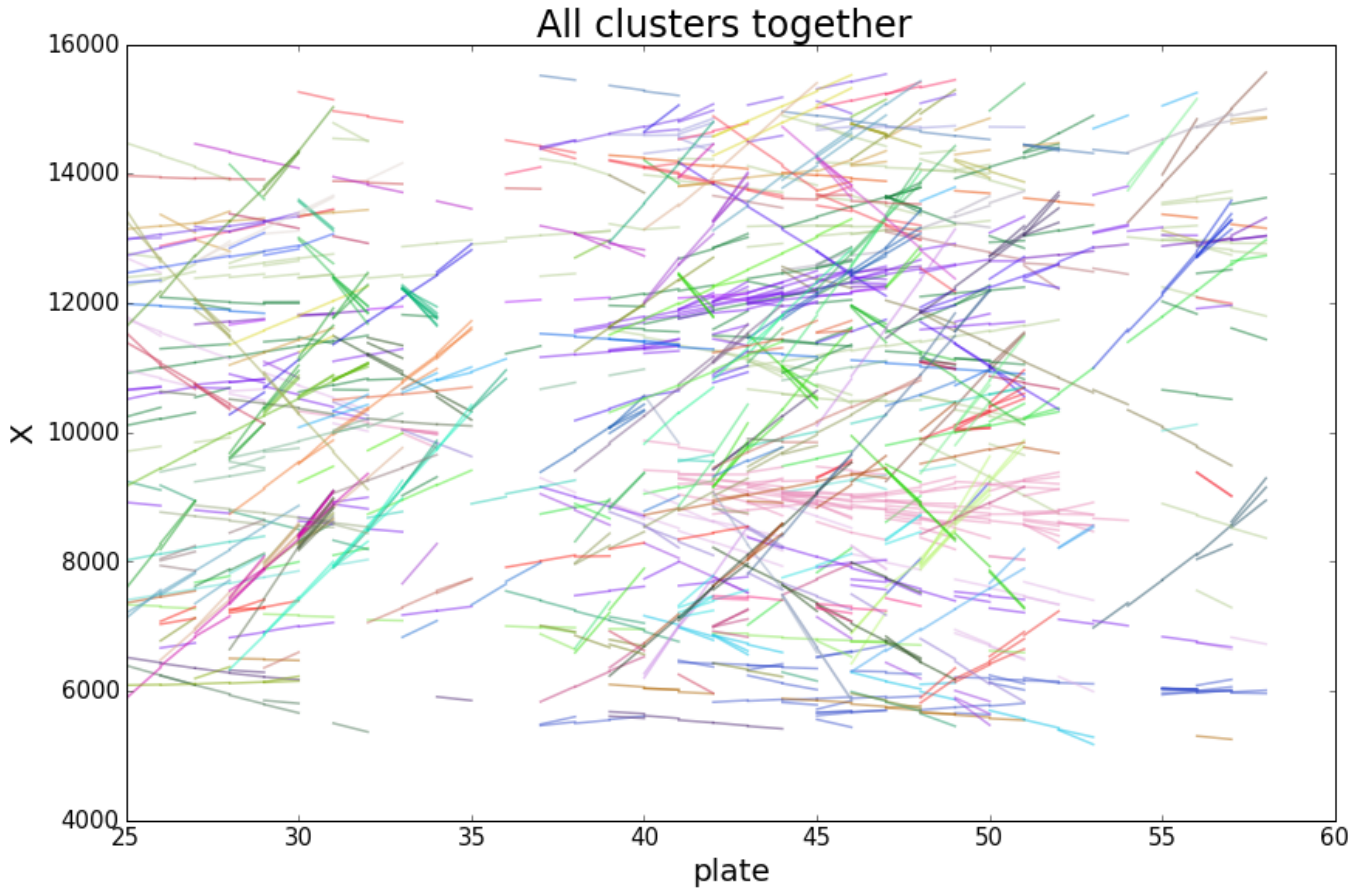


Visualization can be improved if we encode the line quality using stroke width.
Algorithm clearly found 2 showers in the brick.

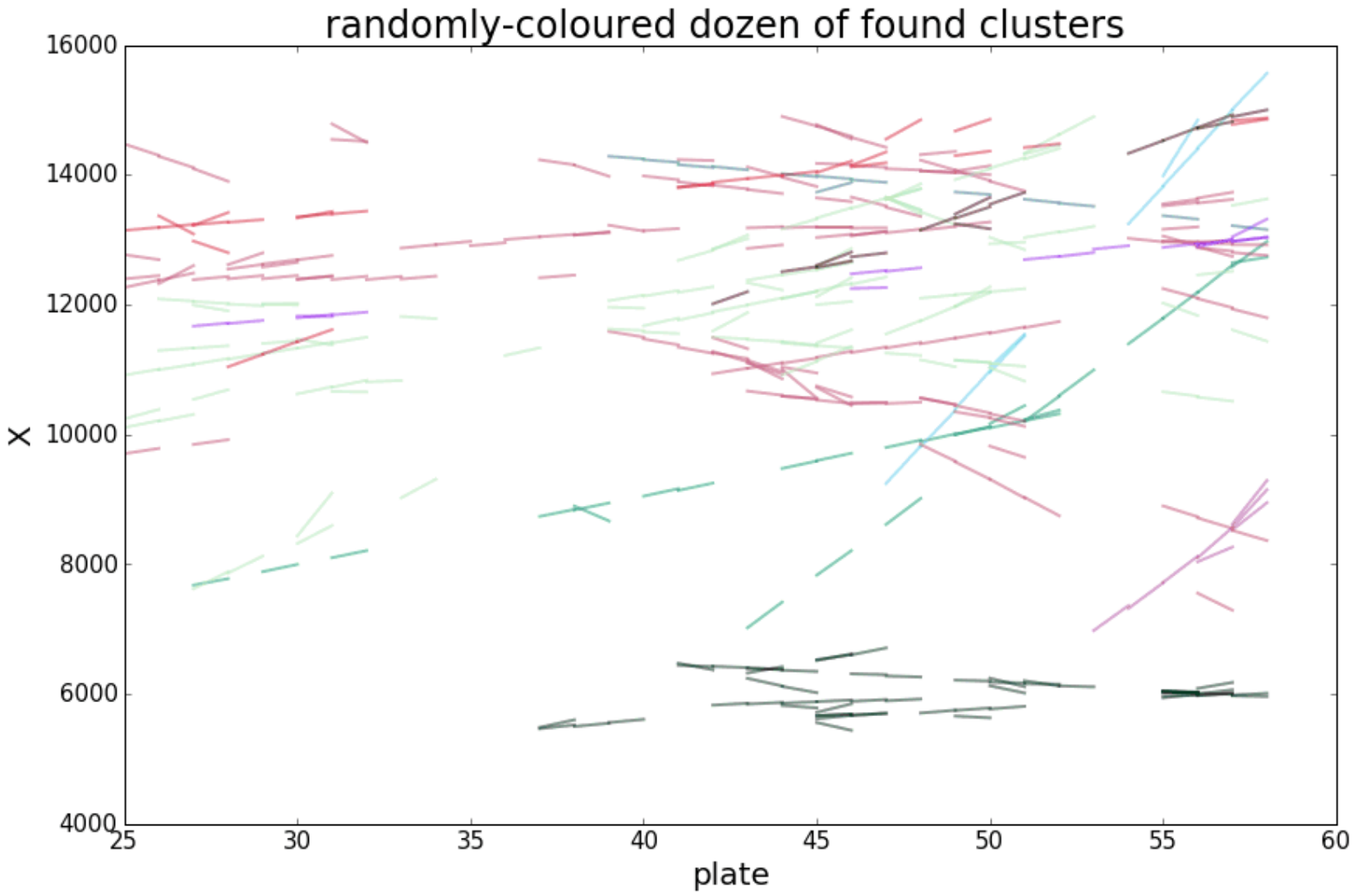
DBSCAN fun

DBSCAN (Density-based spatial clustering of applications with noise) is a scalable clustering algorithm widely used.

I am using scikit-learn implementation.



Split selected tracks into clusters with DBSCAN (requires some nice cooking...) and plot those. Each cluster is drawn with random colour.


randomly-coloured dozen of found clusters
16000
14000
12000
10000
8000
6000
4000
x
25
30
35
40
45
50
55
60
plate

([animated])


randomly-coloured dozen of found clusters
16000
14000
12000
10000
8000
6000
4000
X
25
30
35
40
45
50
55
60
plate

Other bricks

I had also 16 other bricks (downstream plates), but algorithm finds only some parts of the track.

- in two bricks top line quality ~8, but those could be simply tracks
- from detected pattern inter-plate misalignment is clearly seen
  - this drastically affects algorithm power
- missing plates is another tragedy

Conclusion

- we can search for showers in nuclear emulsion
  - even without any preliminary guess
  - shower patterns are still visible
- required more systematical data collection / processing
  - e.g. no gaps in the plates
  - inter-plate alignment is also important
- machine learning can do the rest of the job

The end !

Links

1. PhD thesis of B. Hosseini
2. materials of machine learning in High Energy Physics 2016 school
3. Machine learning at Imperial College London course materials
4. XGBoost paper
5. DBSCAN paper