INTRODUCTION TO MACHINE LEARNING

Alex Rogozhnikov, 2015

WHAT IS ML ABOUT?

Inference of statistical dependencies which give us ability to predict

Data is cheap, knowledge is precious

WHERE ML IS CURRENTLY USED?

- Search engines, spam detection
- Security: virus detection, DDOS defense
- Speech recognition
- Video recognition: faces detection / identification, pedestrian detection
- Credit scoring, fraud detection
- Market basket analysis, Customer relationship management (CRM)
- Brain-computer interface
- Churn prediction
- ... and hundreds more

PROBLEMS ADDRESSED BY ML

1. Classification (binary classification, multiclassification)
2. Regression

AND ALSO

1. Outlier detection
2. Density estimation
3. Representation learning
4. Clustering
5. Dimensionality reduction
6. etc.

SUPERVISED LEARNING: NOTION

training data is represented as set of pairs

$$x_i, y_i$$

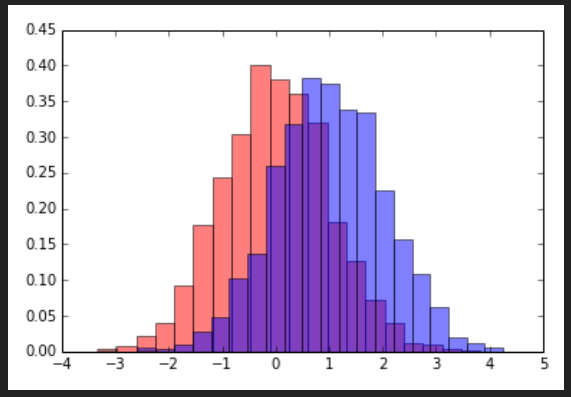
- i is index of event
- x_i is vector of features available for event
- y_i is target — the value we need to predict

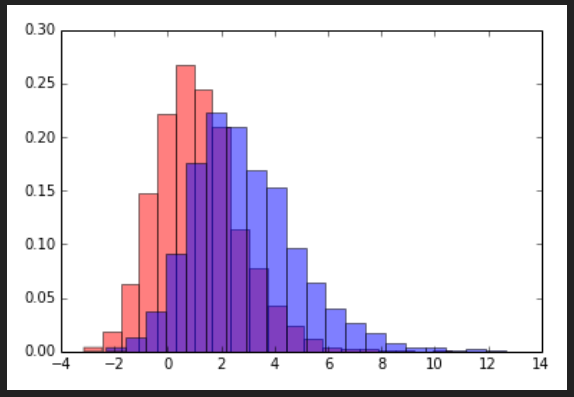
Examples:

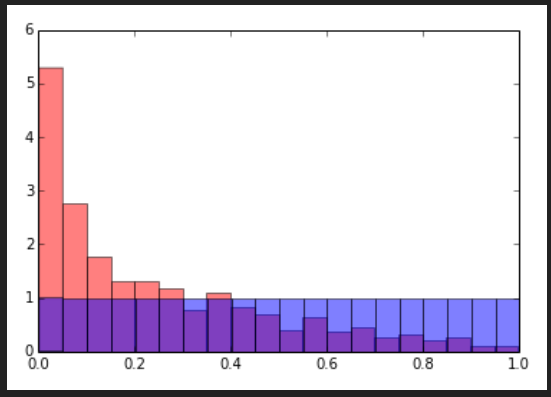
- defining type of particle (or decay channel)
- $y_i \in \{0, 1\}$ — binary classification, 1 is signal, 0 is bck

MEASURING QUALITY

Output of classification algorithm is probability

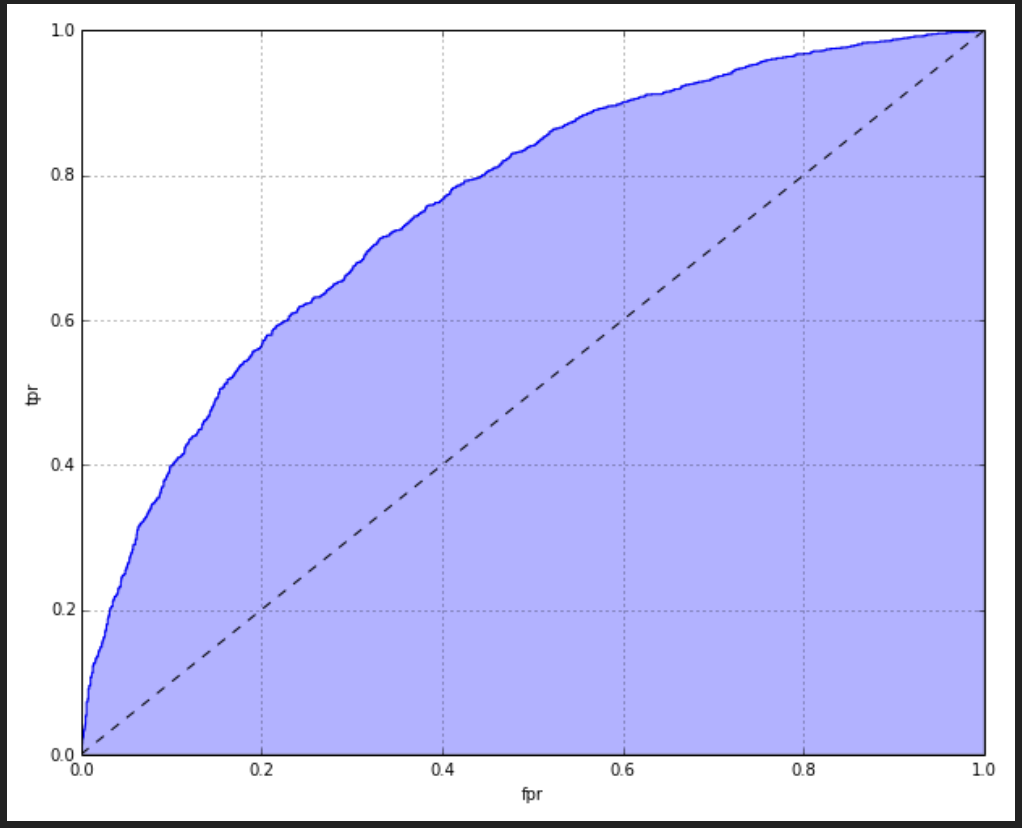






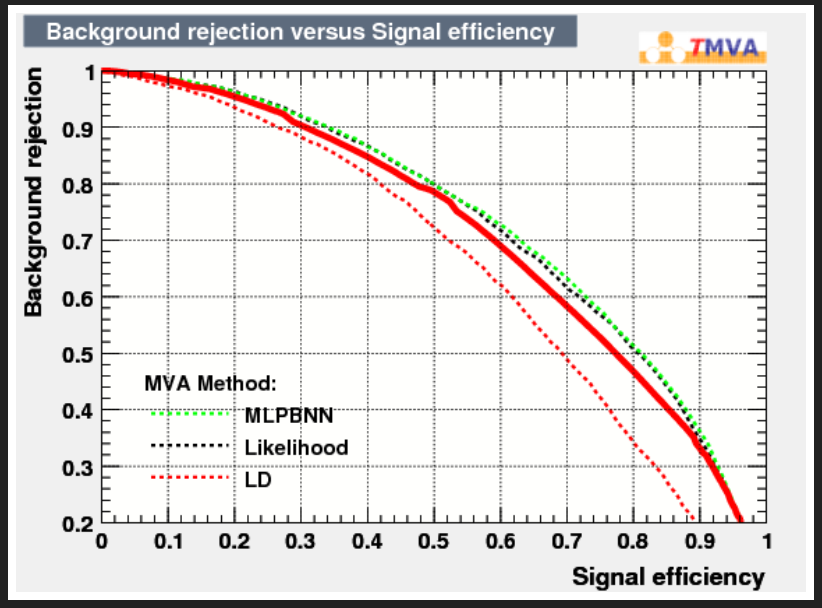
ROC curve demonstration

ROC AUC
(AREA UNDER THE ROC CURVE)



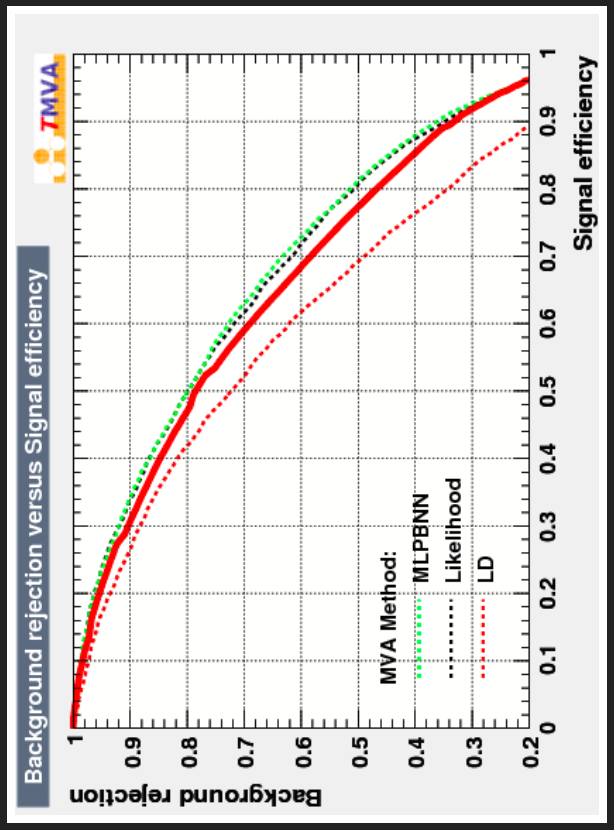
$ROC\ AUC = P(x < y)$ where x, y are predictions of random background and signal events.

ROC: NOTION DIFFERENCE



HEP notion

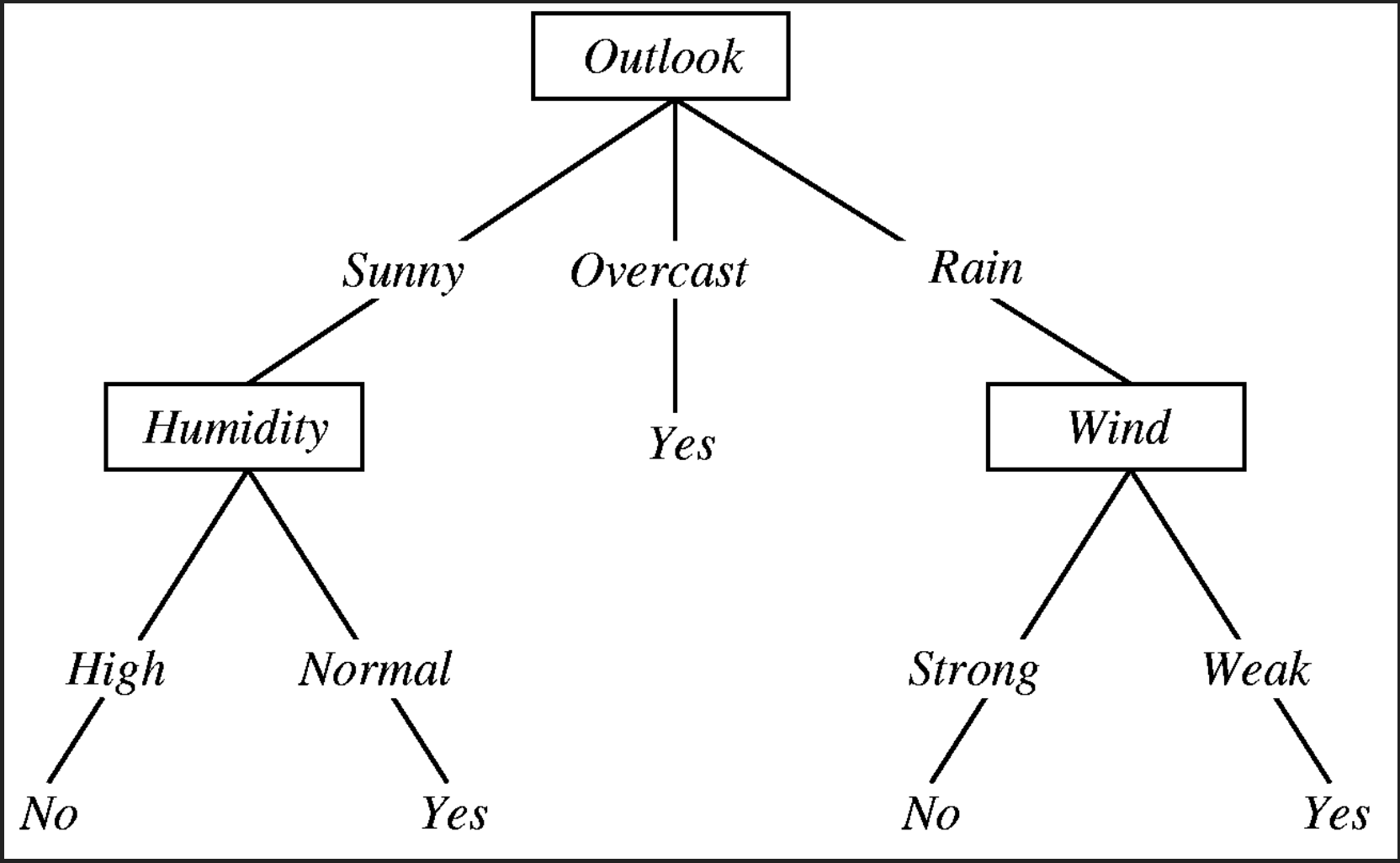
→



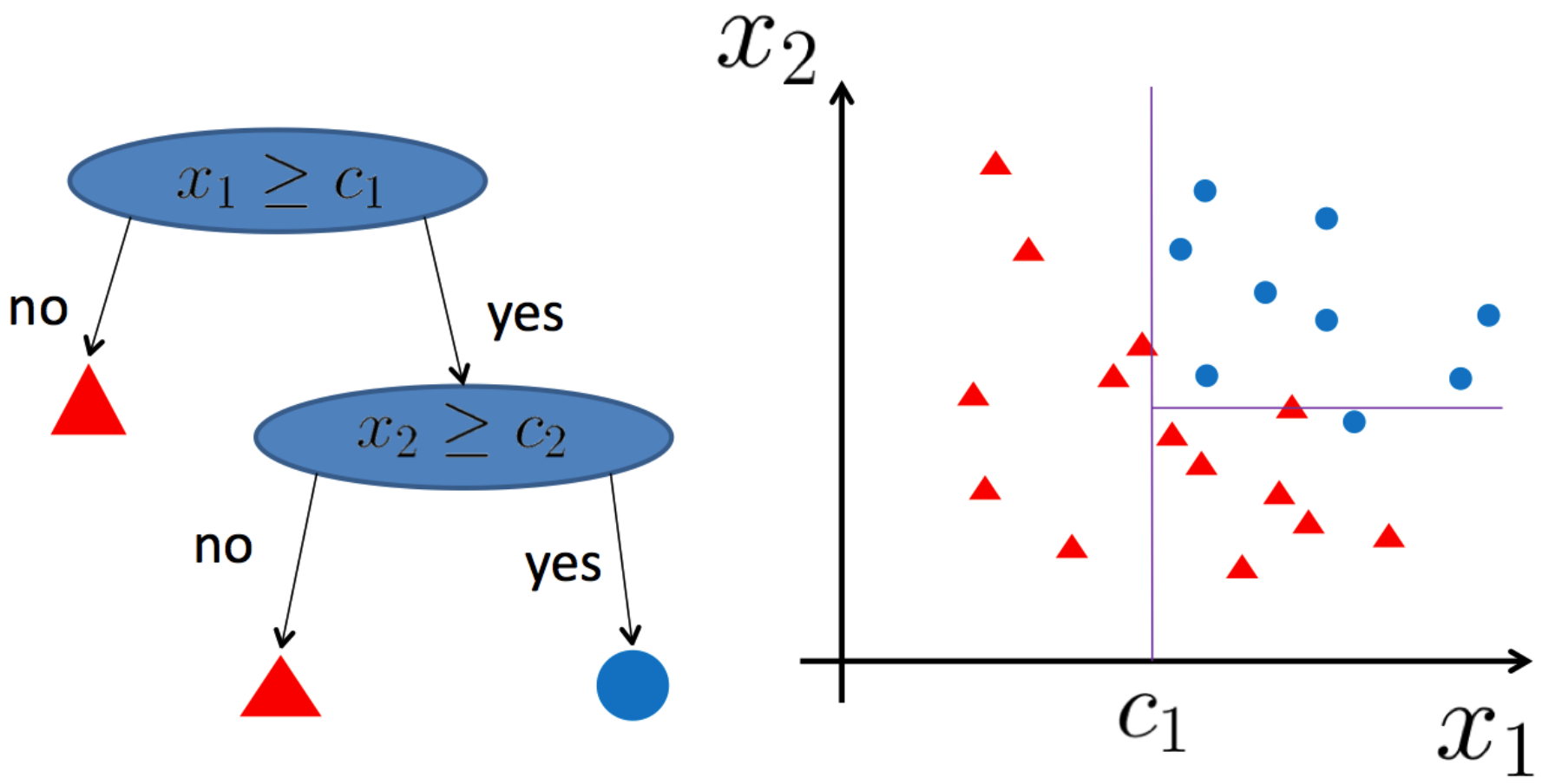
ML notion

DECISION TREES

Example: predict outside play based on weather conditions.



DECISION TREE



Decision trees in ML check only simple conditions: $x^j > c$

DECISION TREE

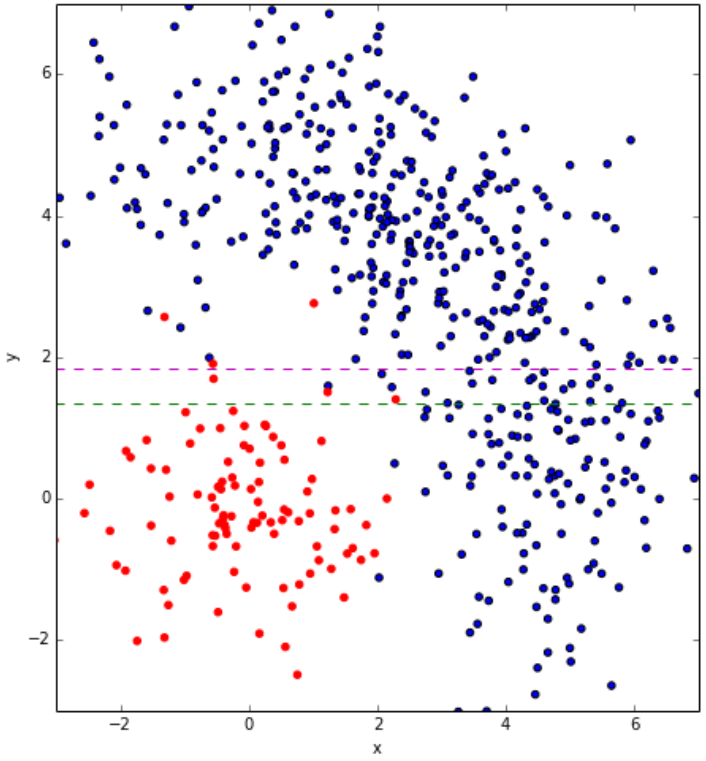
- fast & intuitive prediction
- building optimal decision tree is NP complete
- building tree from root using greedy optimization

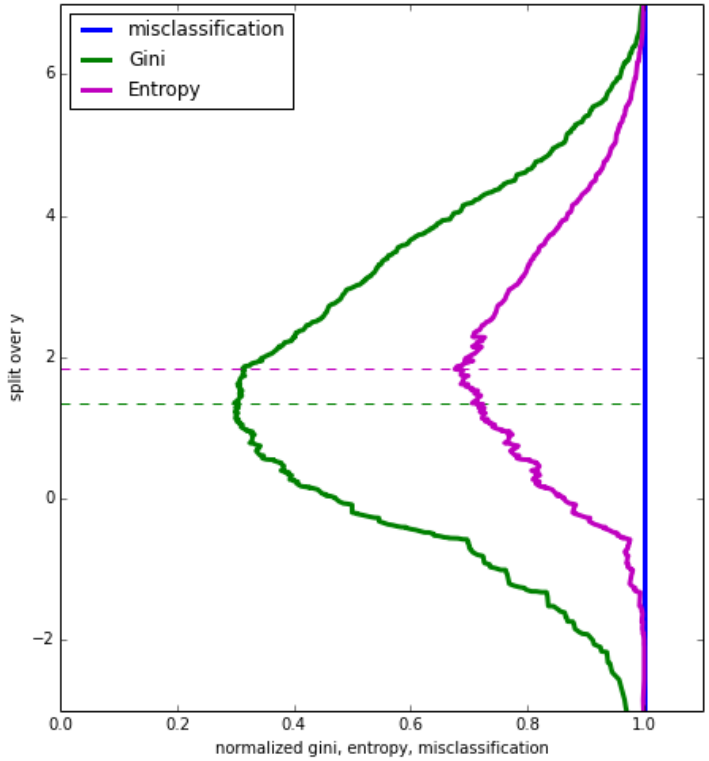
  each time we split one leaf, finding optimal feature and threshold
- need criterion to select best splitting (feature, threshold)

NAIVE APPROACH TO BUILD TREE

Take split, which provides minimal misclassification

Why this is bad approach:



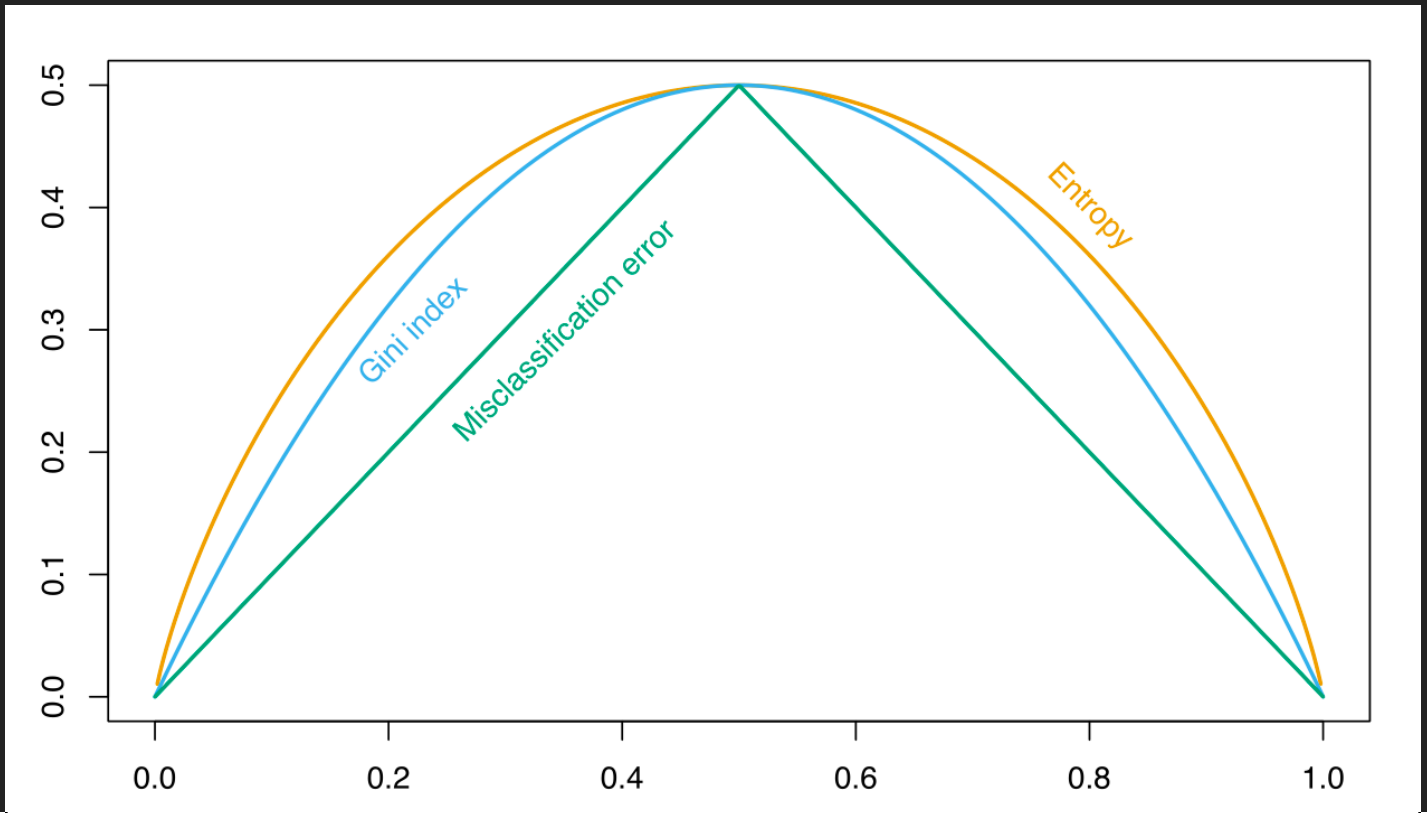


DECISION TREE: SPLITTING CRITERION

$$\text{TotalImpurity} = \sum_{\text{leaf}} \text{impurity}(leaf) \times \#\text{ samples in leaf}$$

Impurity functions (p - portion of signal events in leaf):

$$\begin{aligned} \text{Misclass.} &= \min(p, 1-p) \\ \text{Gini} &= p(1-p) \\ \text{Entropy} &= -p\log p - (1-p)\log(1-p) \end{aligned}$$



DEMONSTRATION HOW TREE GROWS

OVERFITTING

There are two definitions of overfitting, which often coincide:

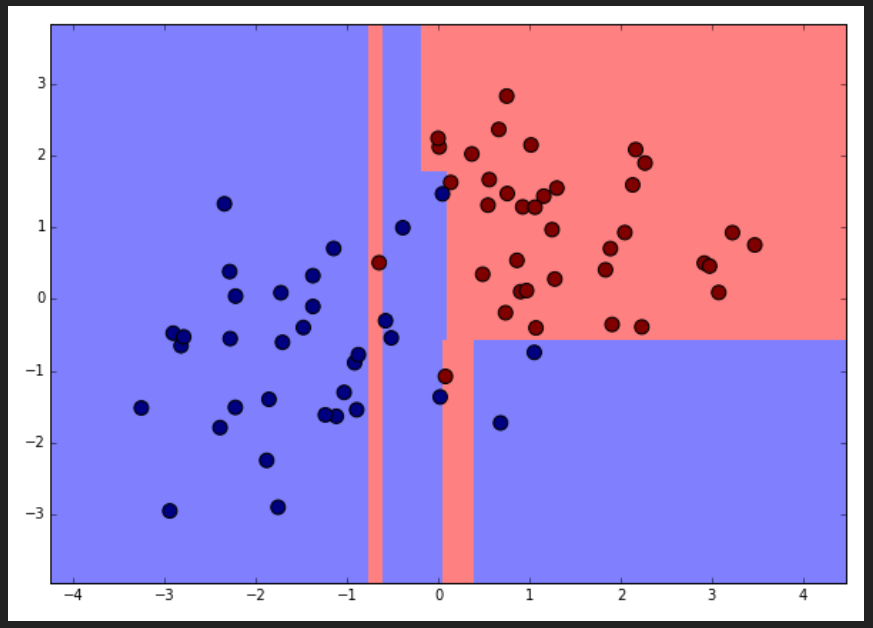
Difference-overfitting

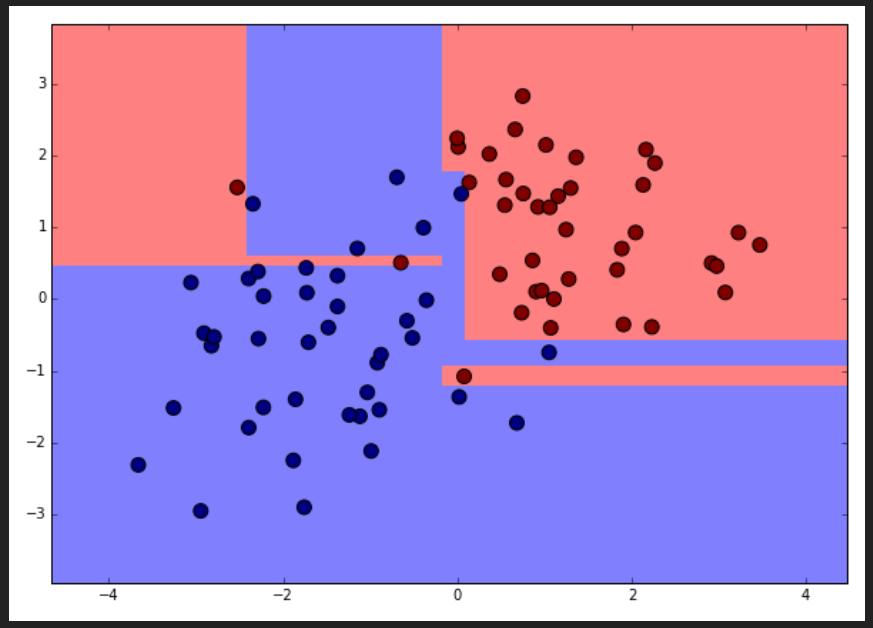
There is significant difference in quality of predictions between train and test.

Complexity-overfitting

Formula has too high complexity (e.g. too many parameters), increasing the number of parameters drives to lower quality.

INSTABILITY OF DECISION TREE





minor modifications in training dataset drive to completely different trees

DECISION TREE SUMMARY

- fast, intuitive and numerically stable
- works with features of different nature
- instable to modifications in train sample
- ovefits (can be prevented by pre-stopping or post-pruning)
- never used in applications

but ensembling of trees is popular approach

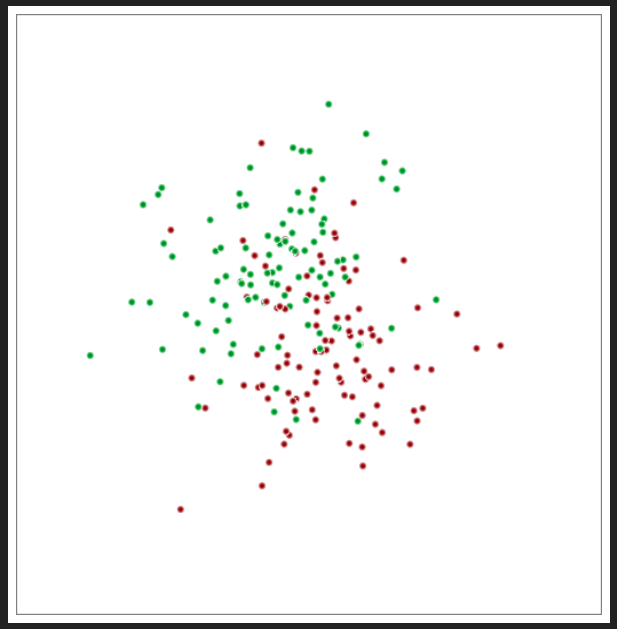
RANDOM FOREST

Simple ensembling algorithm over trees

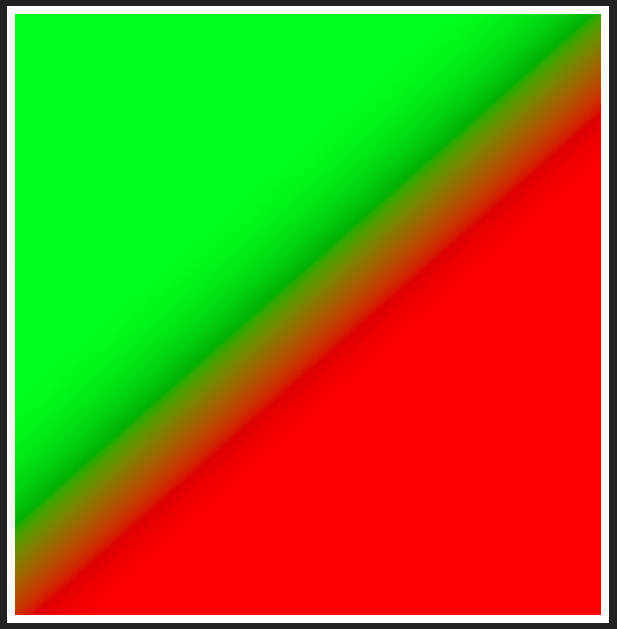
Train independently many trees:

- using random part of data
- using only random subset of features

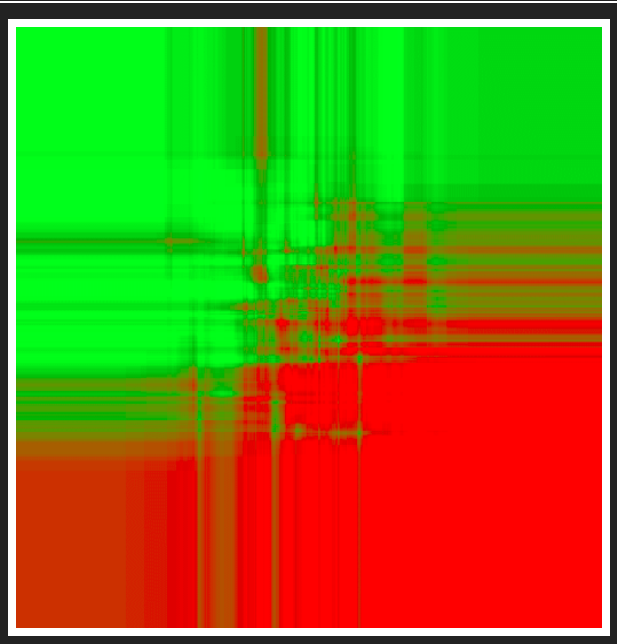
And simply averaging predictions of trees.



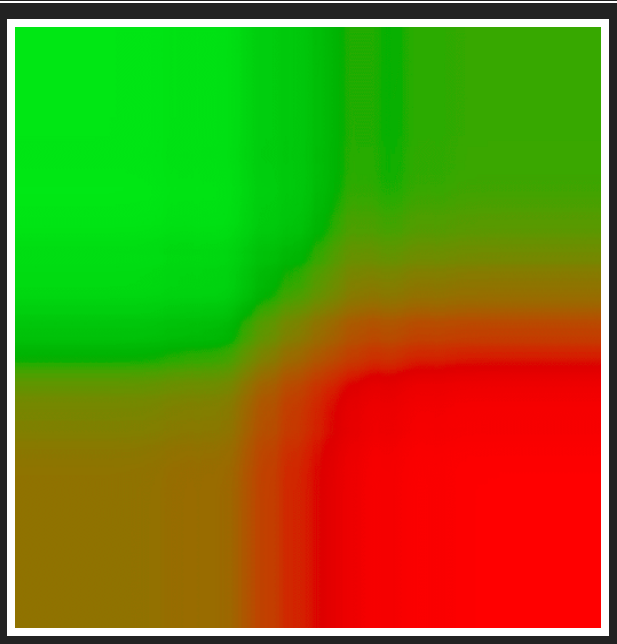
data



optimal boundary



50 trees

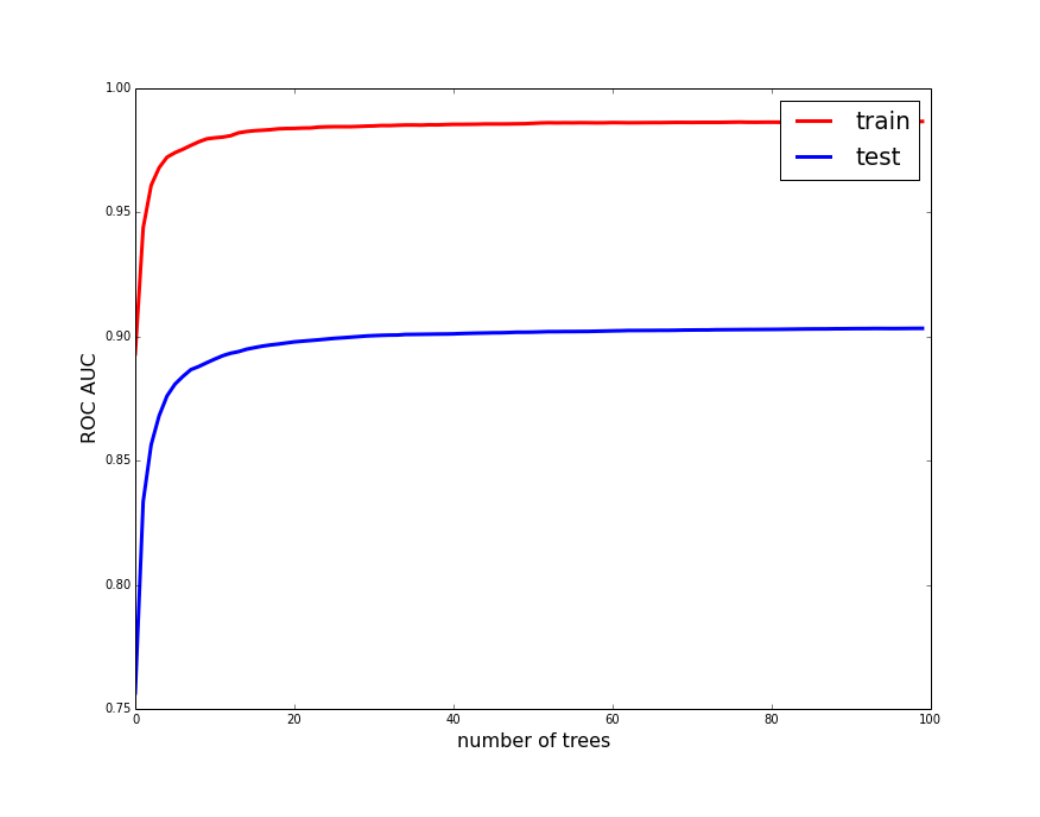


2000 trees

50 trees

2000 trees

OVERFITTING

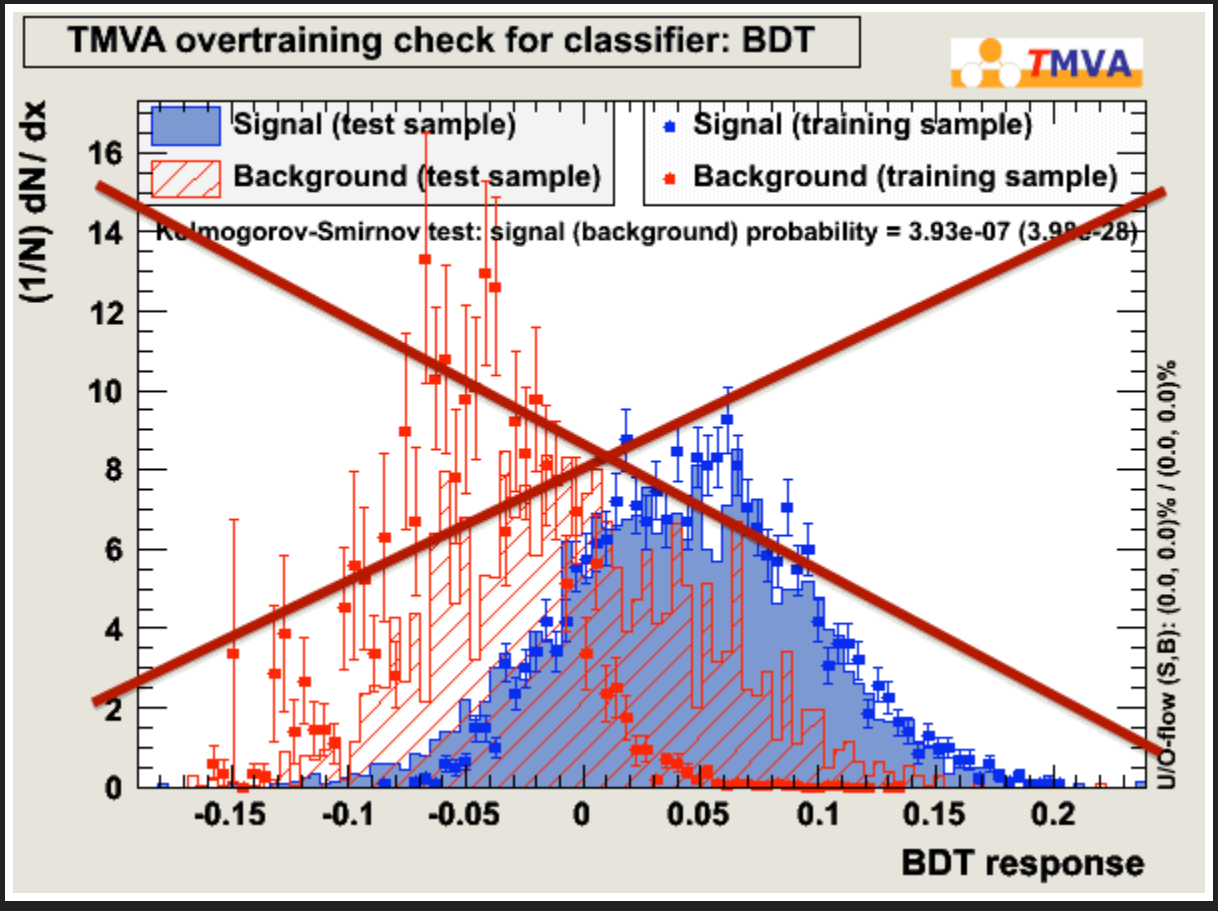


- difference-overfitted (predictions for train and test are different), but it doesn't matter
- doesn't overfit: increasing complexity (adding more trees) doesn't spoil classifier

Difference-overfitting is inessential, provided that we measure quality on holdout (though easy to check).

Complexity-overfitting is problem — we need to test different parameters for optimality.

Don't use distribution comparison to detect overfitting



RANDOM FOREST SUMMARY

- Works with features of different nature
- Stable to noise/modifications in data
- Doesn't need much tuning
- Doesn't correct mistakes done by previous trees
- May generate huge formulas
  From 'Testing 179 Classifiers on 121 Datasets'

> *The classifiers most likely to be the bests are the random forest (RF) [...] achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets.*

OTHER POPULAR ALGORITHMS:

- Logistic regression
- SVM (support vector machines)
- ANN (artificial neural networks)
- GBDT (gradient boosting over decision trees)

Libraries: start with scikit-learn.

BREAK

APPLYING ML TO TRACKING IN COMET

(with Ewen Gillies)

COMET EXPERIMENT

COMET = COherent Muon to Electron Transition

Searches for CLFV process:

$$\mu^- + Al \rightarrow e^- + Al$$

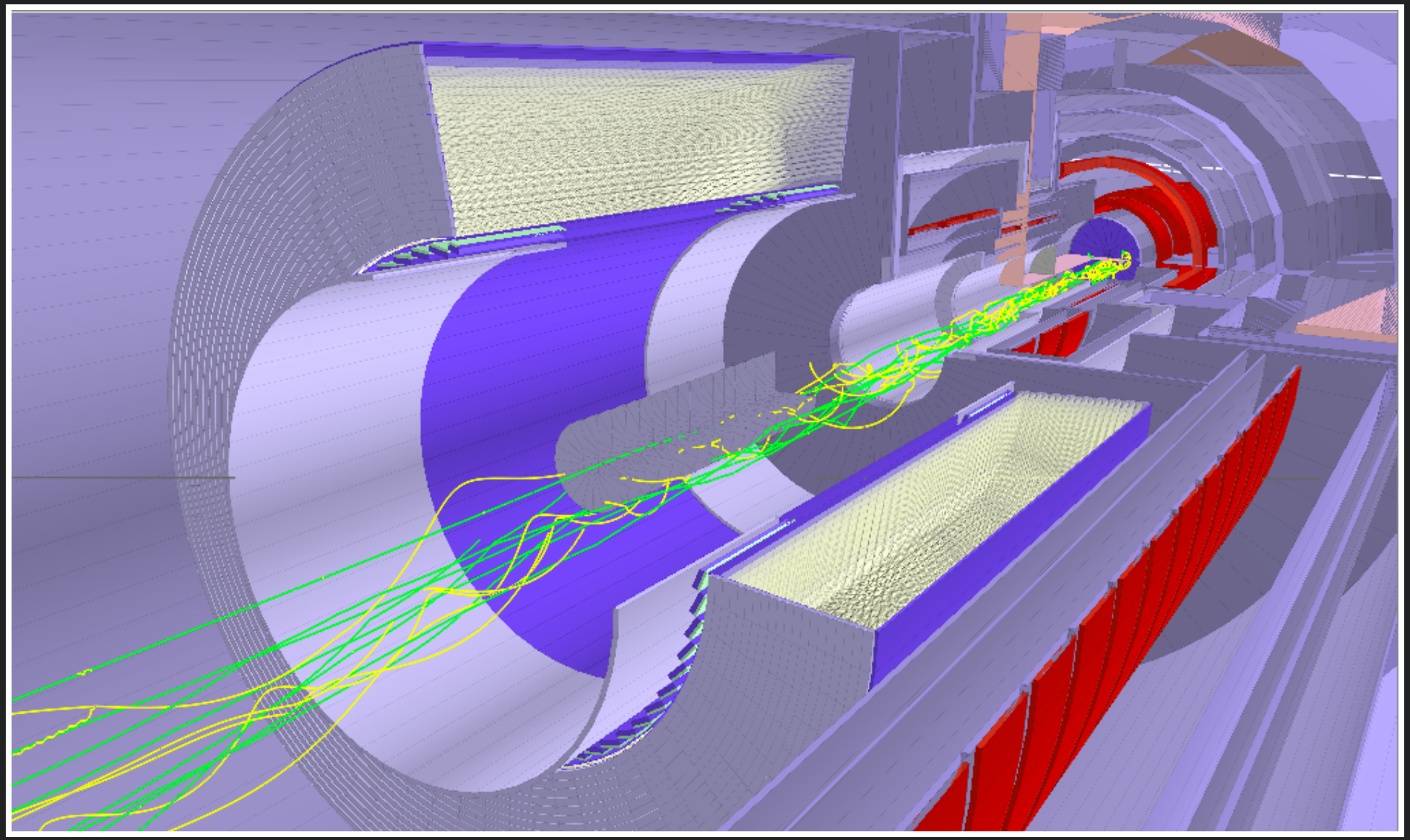
Expected sensitivity of COMET run I:

$$B(\mu^- + Al \rightarrow e^- + Al) < 7.2 \times 10^{-15}$$

Previous result by SINDRUM II:

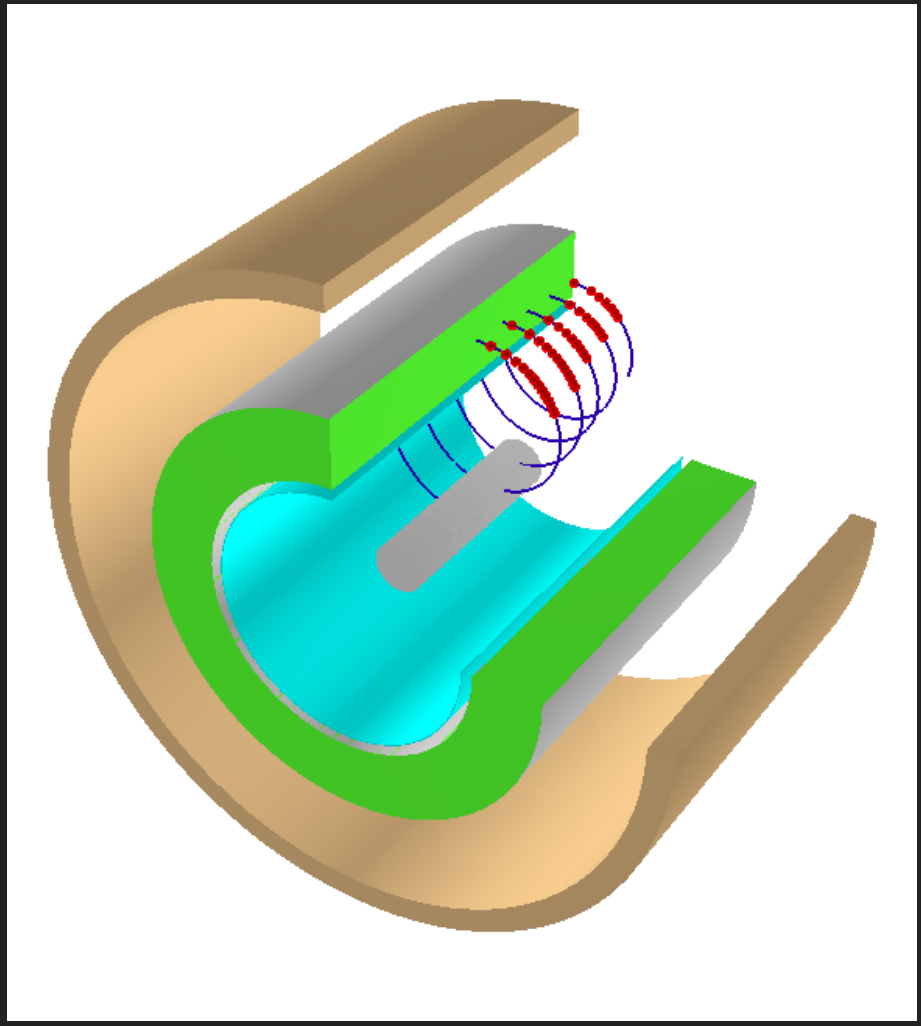
$$B(\mu^- + Al \rightarrow e^- + Al) < 7 \times 10^{-13}$$

COMET (PHASE I)

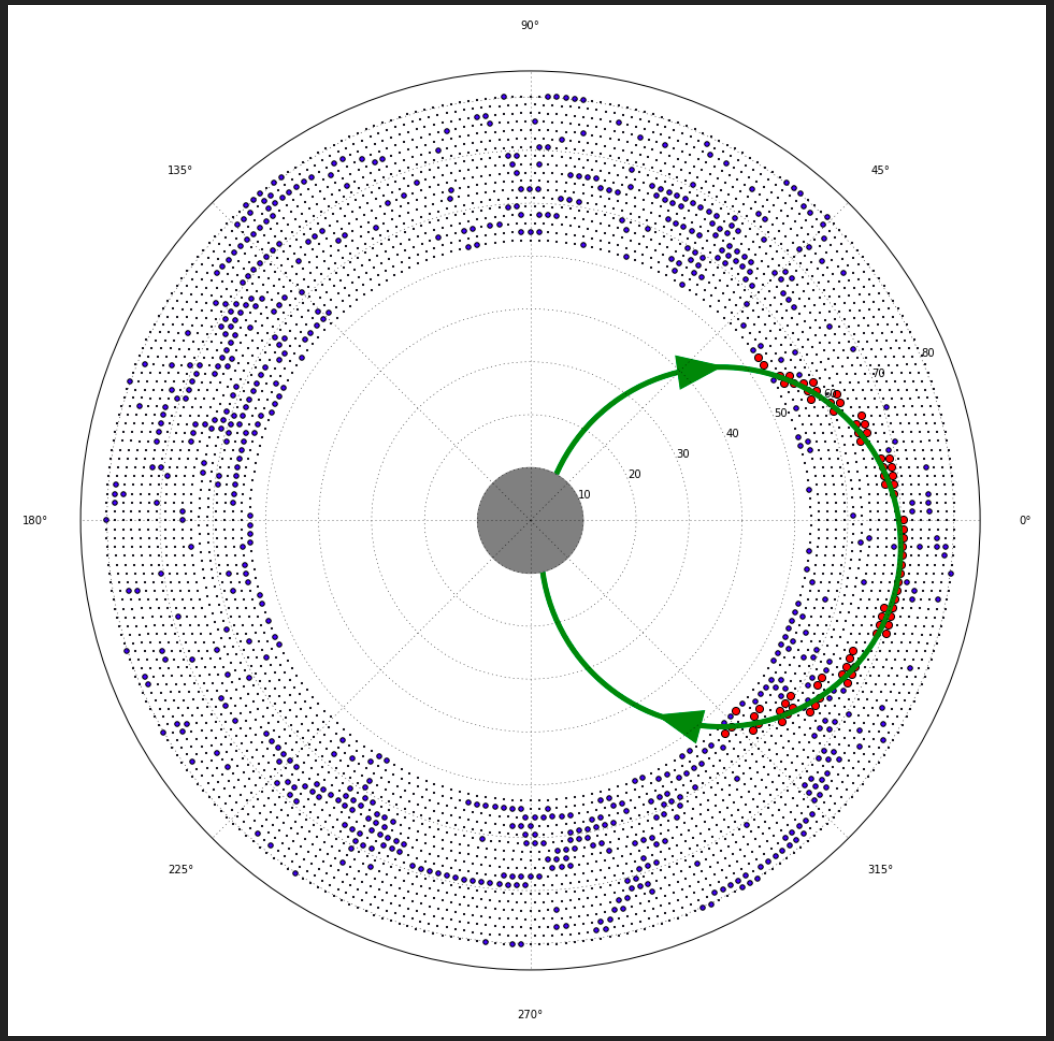


10^9 muons are hitting aluminium target every second

TRAJECTORY OF EMITTED ELECTRON



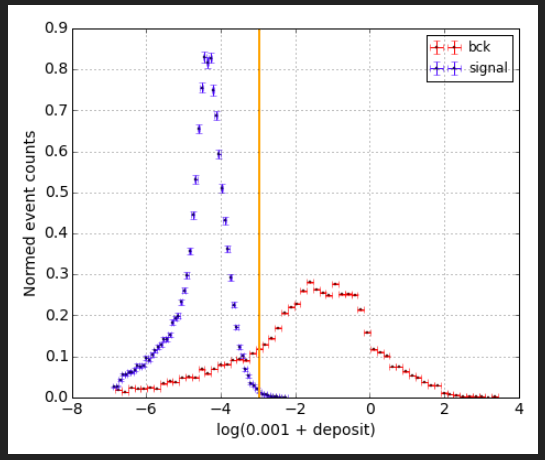
COMET: STRUCTURE OF TRACK



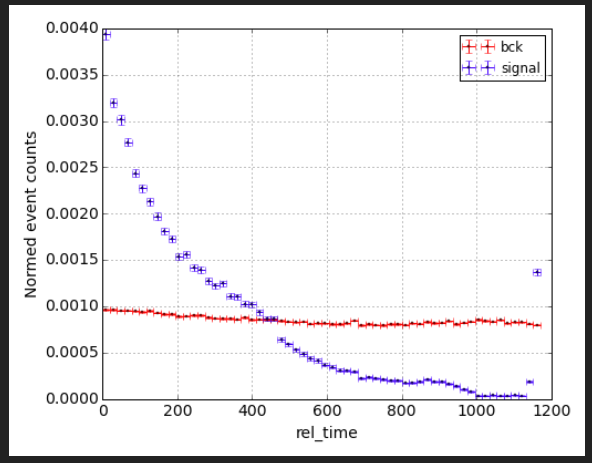
- signal hits are red
- bkg hits are blue
- target: cleaning background (for easier fitting of signal track)
- data structure: event consists of many hits, for each hit we need to predict its type

For each hit we have

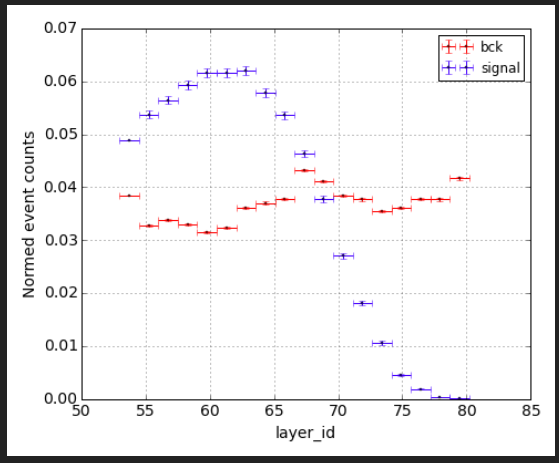
- energy deposited at wire
- readout time



energy deposit



relative timing



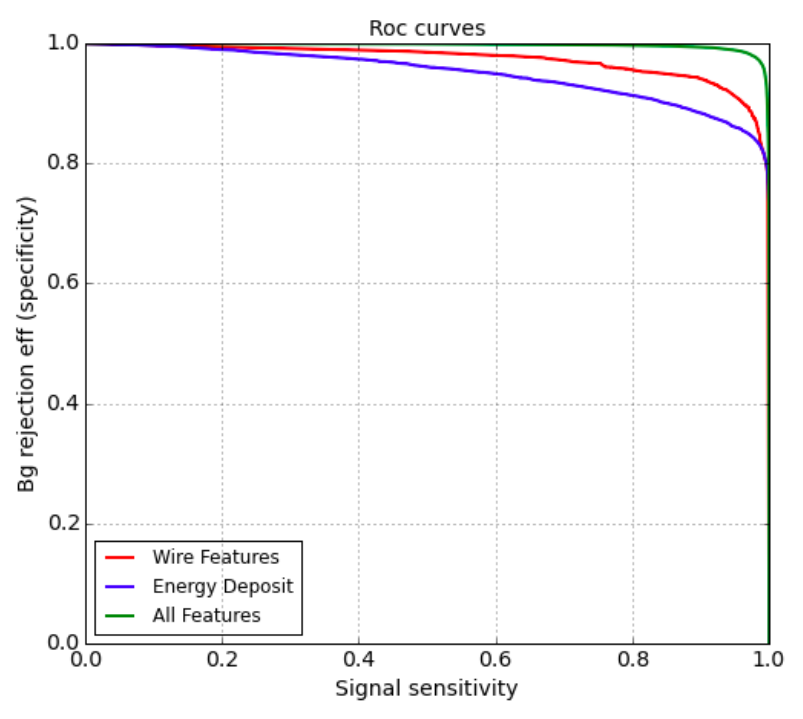
distance

Energy deposit, time and distance can be efficiently combined by ML algorithms, this already clears large amount of background hits

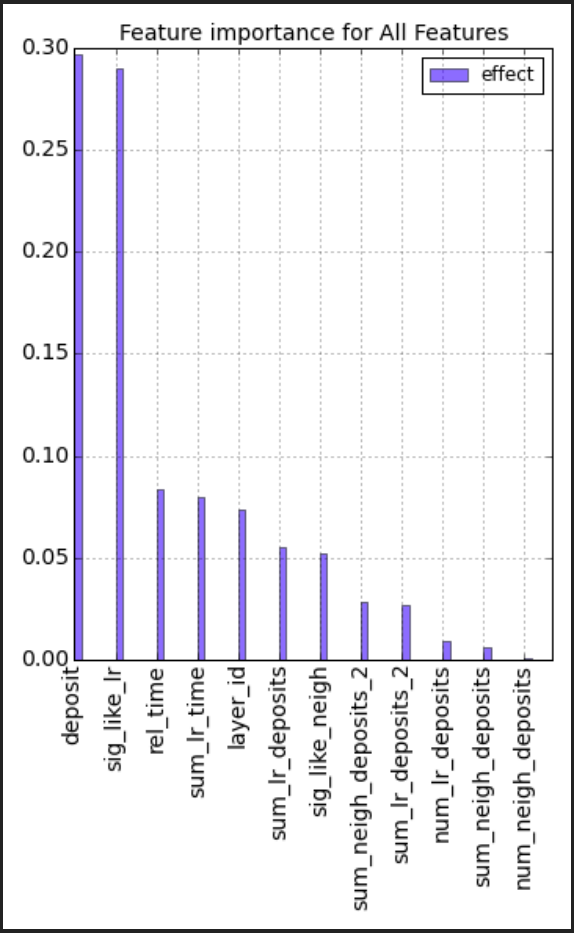
Better approach: use information from nearest wires:

- number of hits on neighbouring wires
- total energy deposited on neighboring wires
- average relative time
- etc

RESULTS OF GBDT



- blue - energy deposition
- red - three local features
- green - local features + collected from neighbors



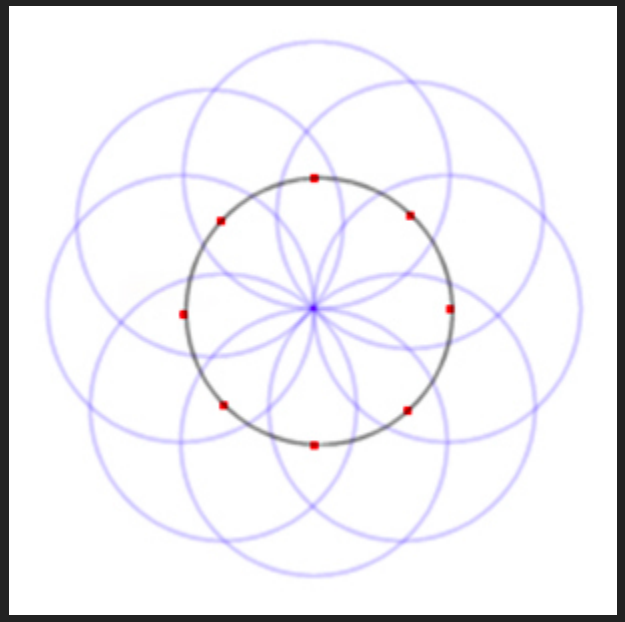
Feature importances sorted by importance

sig_like is output of classifier based only on local wire features.

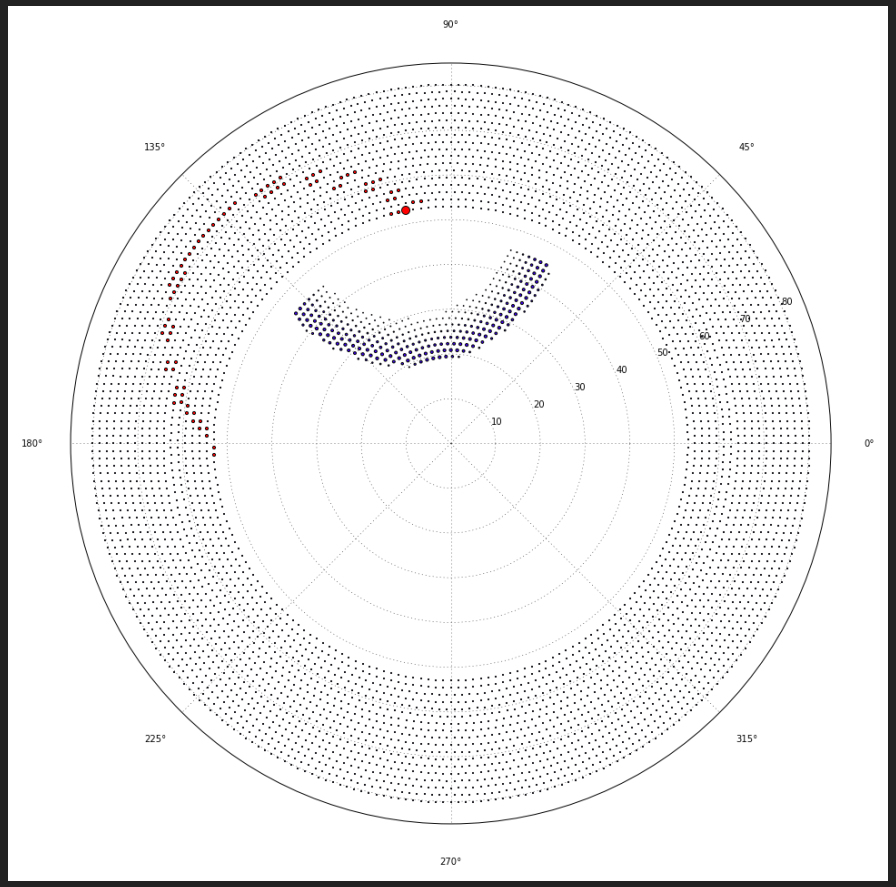
This output is used as input to second stage model.

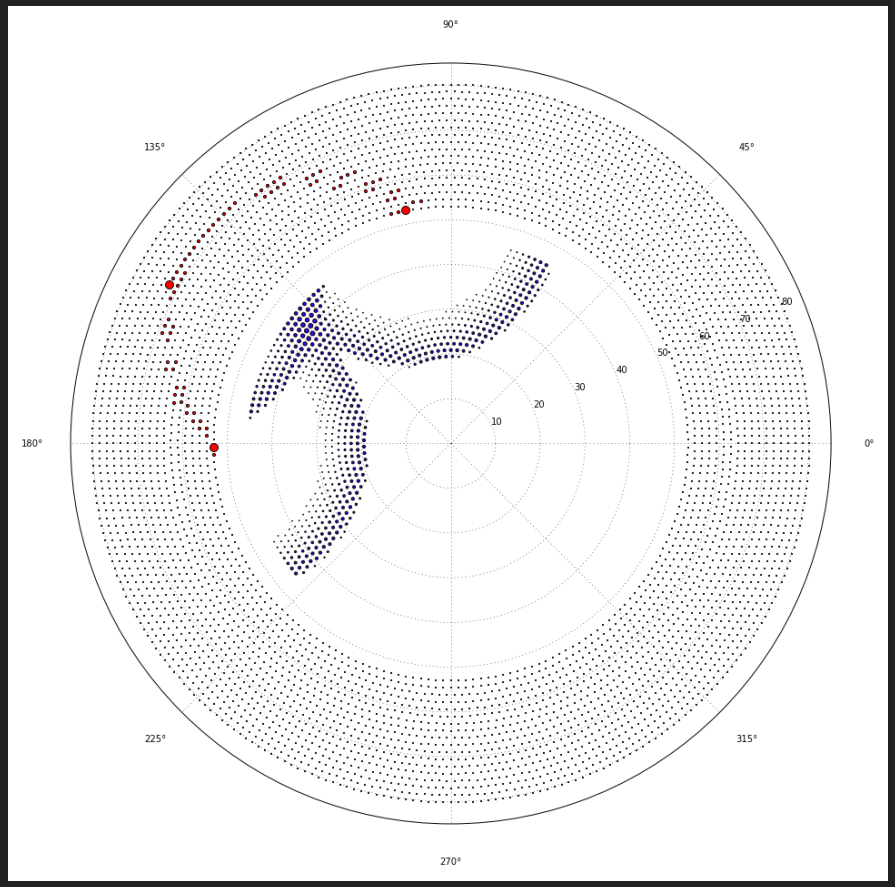
HOUGH TRASFORM

How to find center of a track on a plane if you know the track radius:



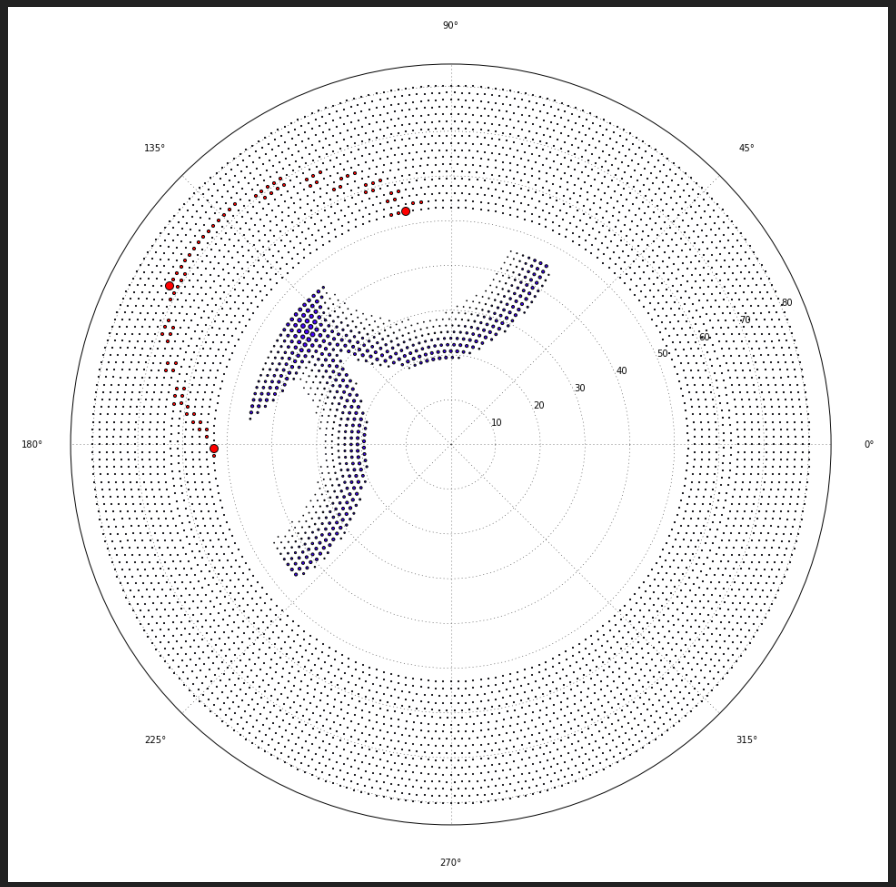
SOFT HOUGH TRANSFORM

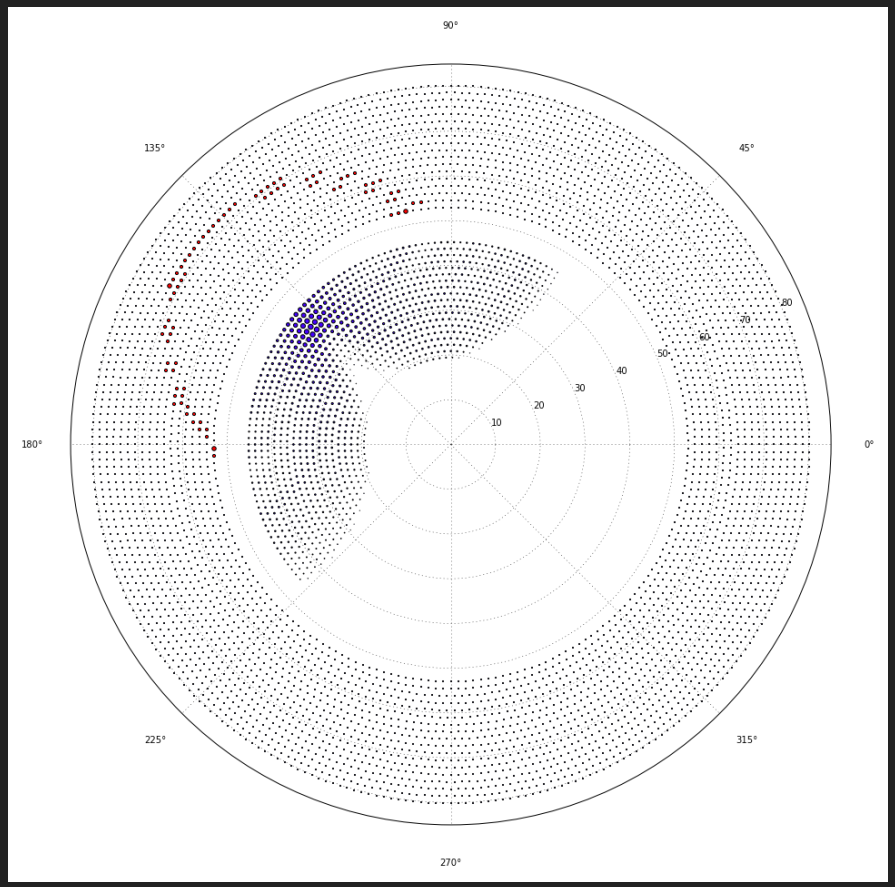




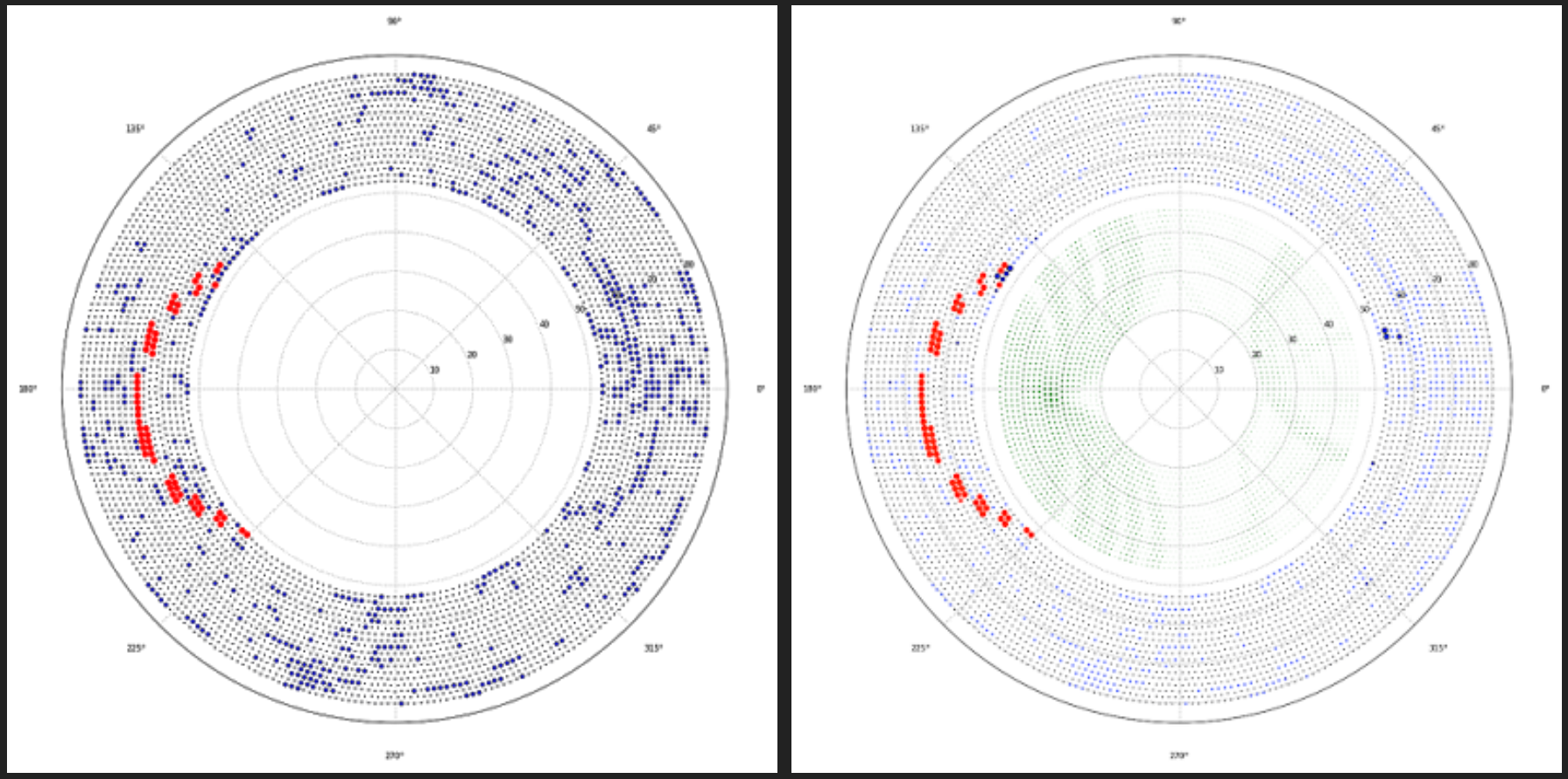
we have good estimation of radius, so using Hough transform to find track centers

SOFT HOUGH TRANSFORM

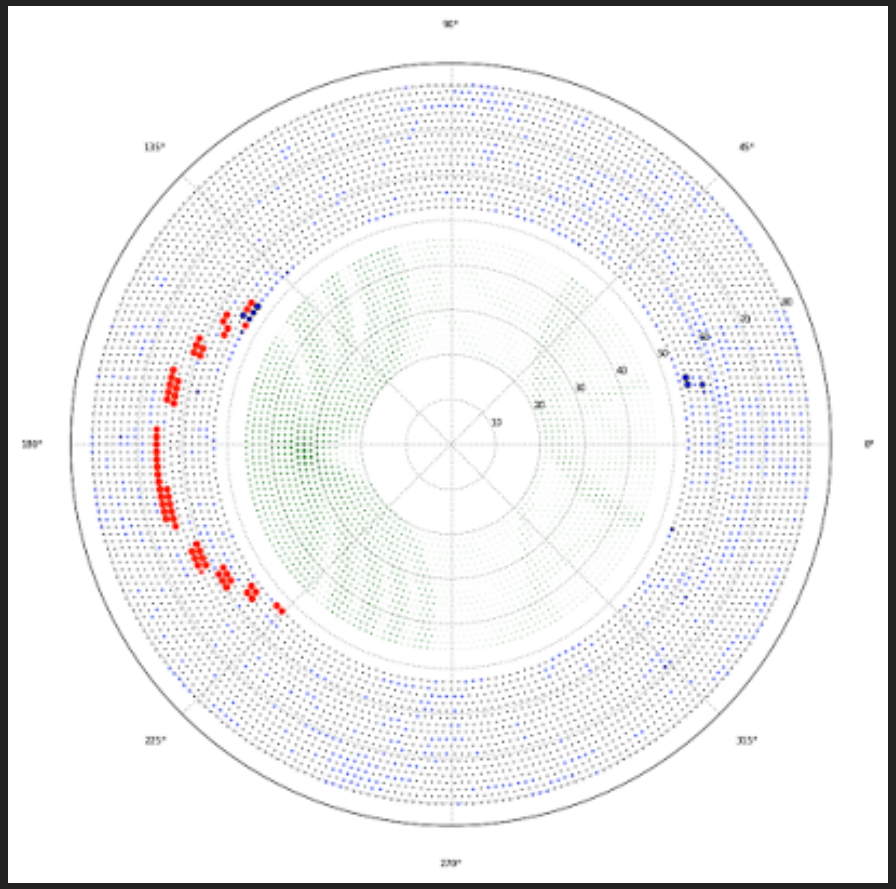


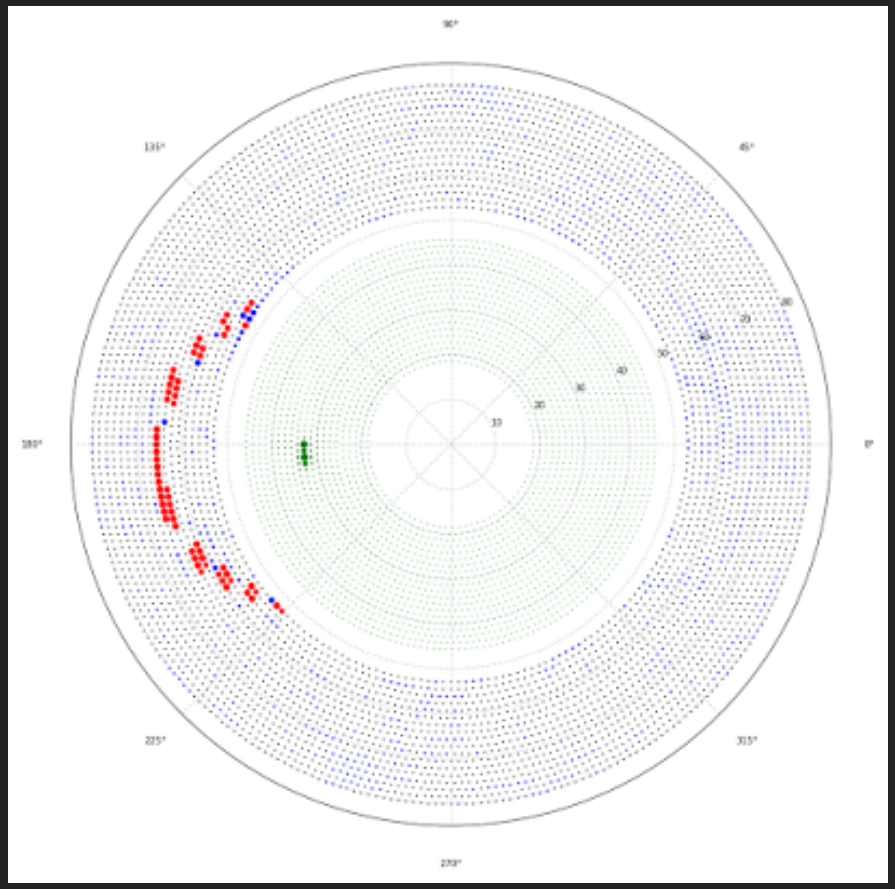


an area with most probable track centers can be clearly seen



Cleaning with GBDT + soft Hough transform





exponentiating (sharpening the peak) + soft inverse Hough transform

HOUGH TRANSFORM

Direct: $\sum_j T_{ij} W_j = C_i$

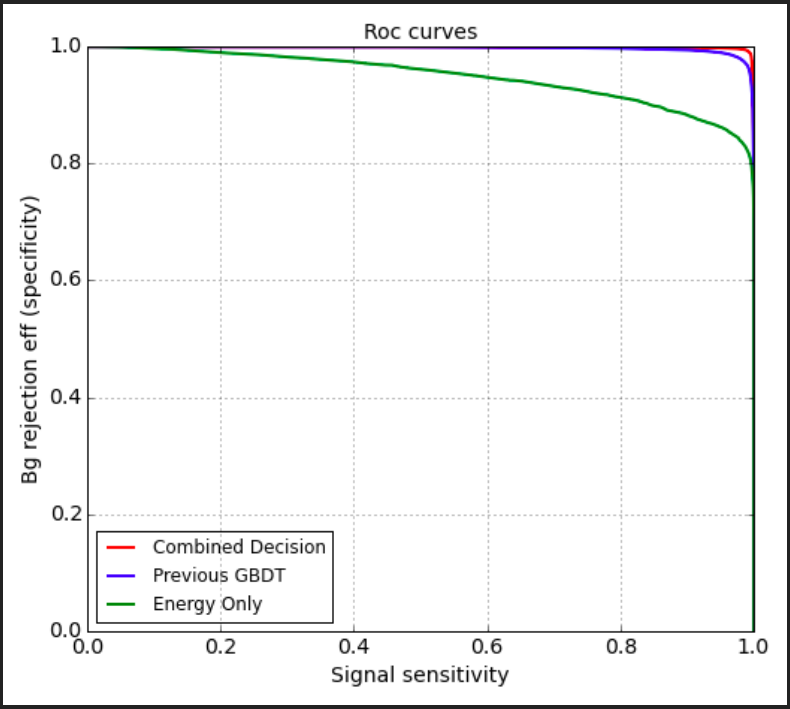
Inverse: $\sum_j S_{ji} C_i = W_j$

- T, S are Hough transform (and inv. transform) matrices
- W_j are outputs of local GBDT
- C_i correspond to track centers, value is higher for more probable centers

ALGORITHM OVERVIEW

1. Cleaning with GBDT on neighbour features
2. Generating one more feature:
    1. Soft Hough transform
    2. Exponentiation
    3. Inverse Soft Hough
3. GBDT on neighbour features + inverse Hough

RESULTS



- adding inverse hough as a feature to GBDT
- it's the most important feature
- ROC AUC:
  0.95 (energy) → 0.9993 (combined with inv. Hough)
  (only preliminary MC)
- Still room for improvement