Inclusive flavour tagging

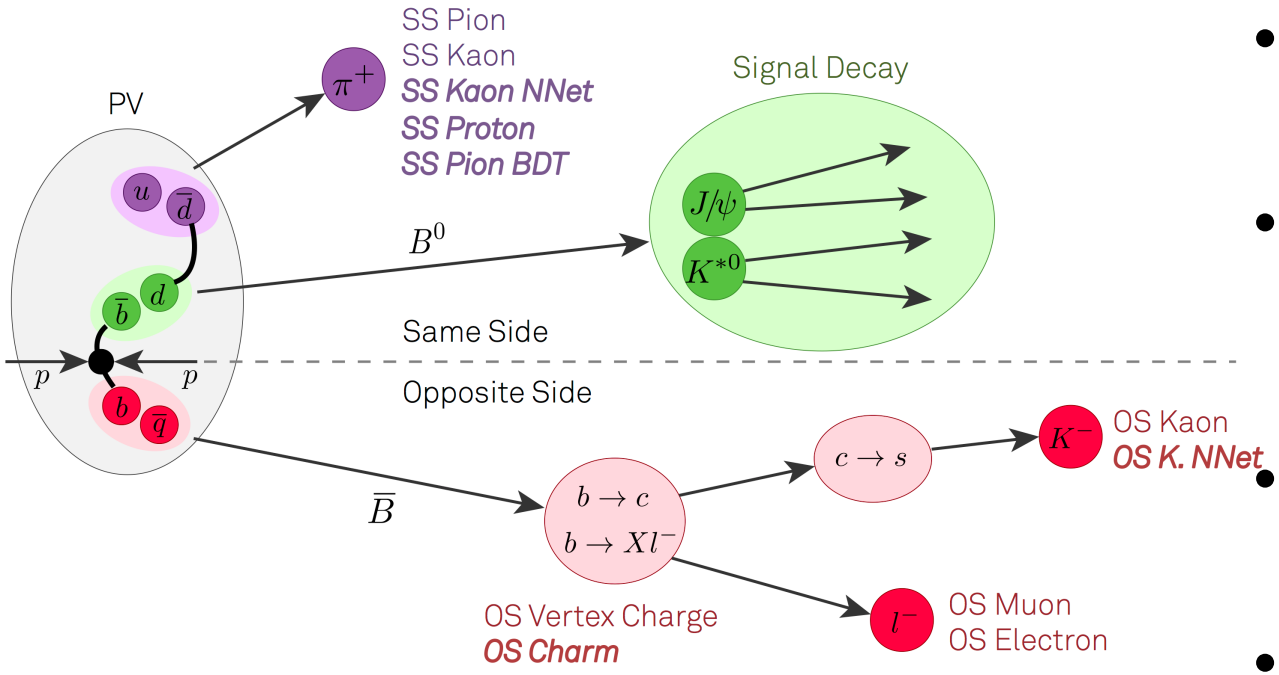
Denis Derkach, Tatiana Likhomanenko & Alex Rogozhnikov
Yandex School of Data Analysis

IML workshop, March 2017

Yandex

SCHOOL OF DATA ANALYSIS

B-tagging (picture from LHCb)



- flavour tagging is important in estimating CKM-matrix
- to predict which meson was produced (B^0 or $\bar{B}^0$), non-signal part of event is analyzed (tagging)
- signal part (green) provides information about decayed state
- there are dozens of tracks that aren't part of this scheme

Evaluation in experiments

We investigate new approaches for tagging of B mesons

- for testing purposes charged mesons ($B^{\pm}$) are used
  - allows cross-testing of models and approaches as both MC (Monte Carlo simulation) and RD (real data) are available
- for tagging both tracks and vertices of the event are used
  - in the presentation only tracks were used for tagging

Conventional tagging

Step 1. Determine candidates for tagging particles

- quite strong preselection using cuts on topology and PID
- if more than one candidate left, select one according to some criterion

Step 2. Evaluate correctness of the guess at step 1 with ML

- separately for SS / OS taggers

Inclusive tagging

Idea: leave both steps of tagging to machine learning.

- use all (non-signal) tracks from event
- let machine guess which tracks are tagging, how their charge is connected to meson flavour, and estimate probability

Inclusive tagging

Idea: leave both steps of tagging to machine learning.

- use all (non-signal) tracks from event
- let machine guess which tracks are tagging, how their charge is connected to meson flavour, and estimate probability

We use a simplistic assumption, that information from tracks is independent. Example of combining information for two tracks:

$$\frac{p(B^+|tracks)}{p(B^-|tracks)} = \frac{p(track_1|B^+)}{p(track_1|B^-)} \times \frac{p(track_2|B^+)}{p(track_2|B^-)}$$

This is called a naive Bayes assumption.

Inclusive tagging – 2

Differences compared to naive Bayes classifier:
1) number of terms in product variates as we want to encounter all tracks

$$\frac{p(B^+|tracks)}{p(B^-|tracks)} = \prod_{track} \frac{p(track|B^+)}{p(track|B^-)}$$

2) ratios in the right part are estimated with machine learning:

$$\frac{p(track|B^+)}{p(track|B^-)} = \left(\frac{p(\text{track and B have same charge})}{p(\text{track and B have diff charge})} \right)^{\text{track charge}}$$

Above trickery

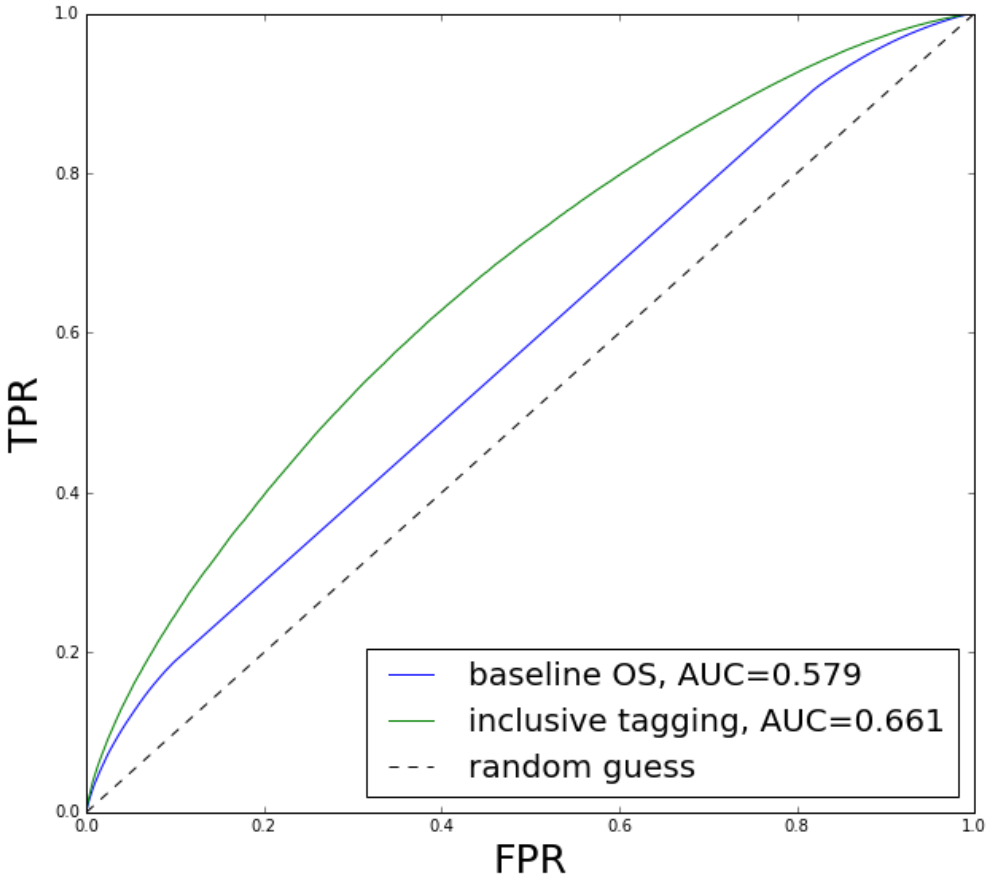
- guarantees that tagging algorithm is C-invariant (P-invariance follows from features used)
- simplifies the training of machine learning

Inclusive tagging – 3

So the pipeline simplified to:

1. train a classifier to predict for all tracks whether they have same charge with meson
    - for neutral mesons matter / antimatter sign is used
2. combine predictions with naive Bayes

Result with simple formula



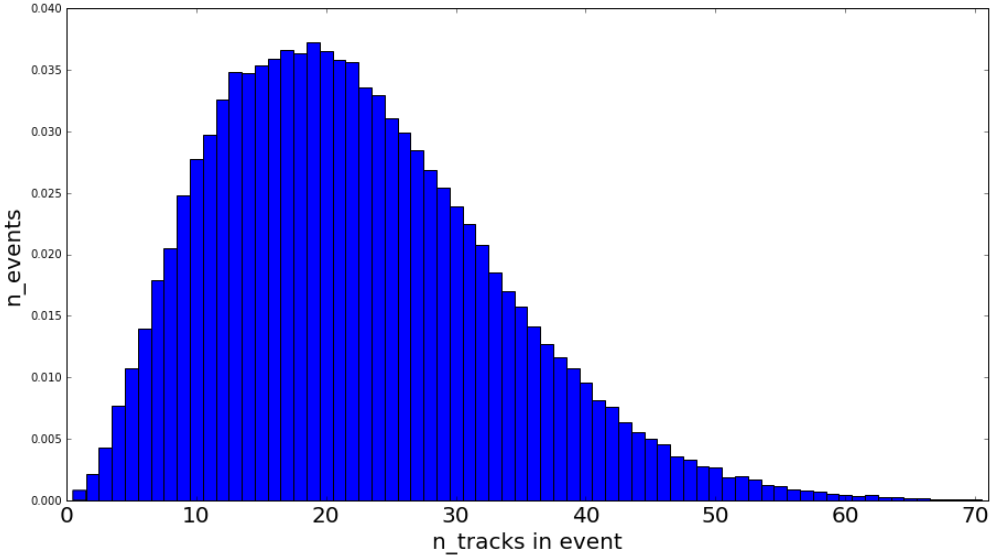
It works. And provides impressive result.

Untagged events' predictions were set to $p(B^+) = p(B^-) = 0.5$

Training of model

Specifics of ML problem:

- binary classification
  - same charge / different charge
- noise is dominating (about 20 tracks per event)
  - tagging tracks are not guaranteed to be detected
- millions of tracks are available for training
- discrimination on a per-track basis is very poor
  - per-track ROC AUC is 0.51 – 0.52



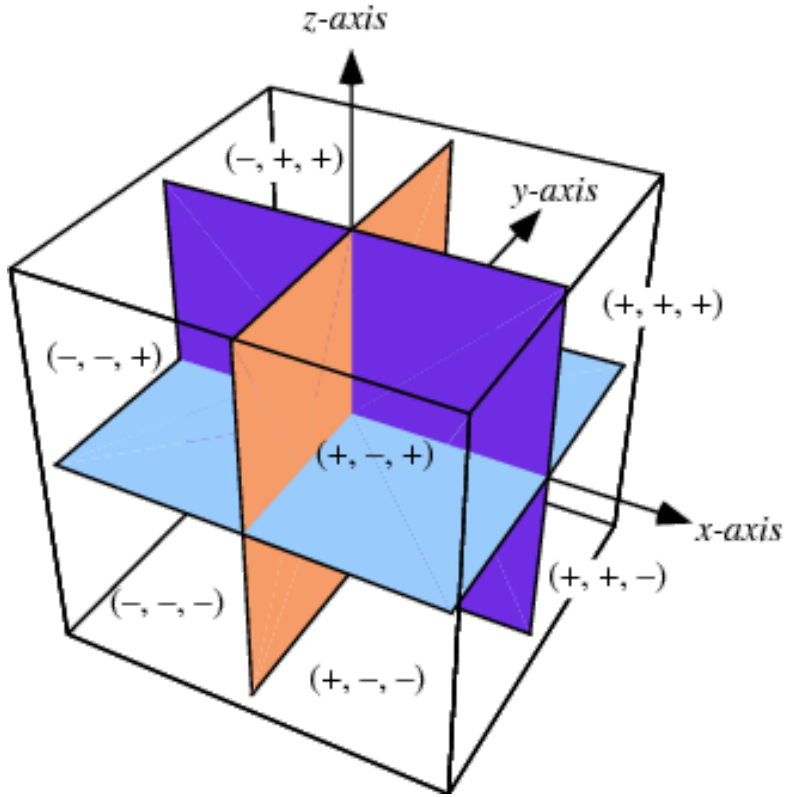
- per-event ROC AUC is much higher, which is due to combining

Boosting models

We use boosting models: DecisionTrain and XGBoost.

DecisionTrain uses oblivious trees (on the picture),
which have an additional restriction that all splits on the same level of depth are identical

- oblivious decision trees are known for their stability
- each split is made based on the information from all dataset
- DecisionTrain operates by making thousands of small steps, which drives to very stable training
- obliviousness allows very fast training/predictions, but this requires bit-hacks



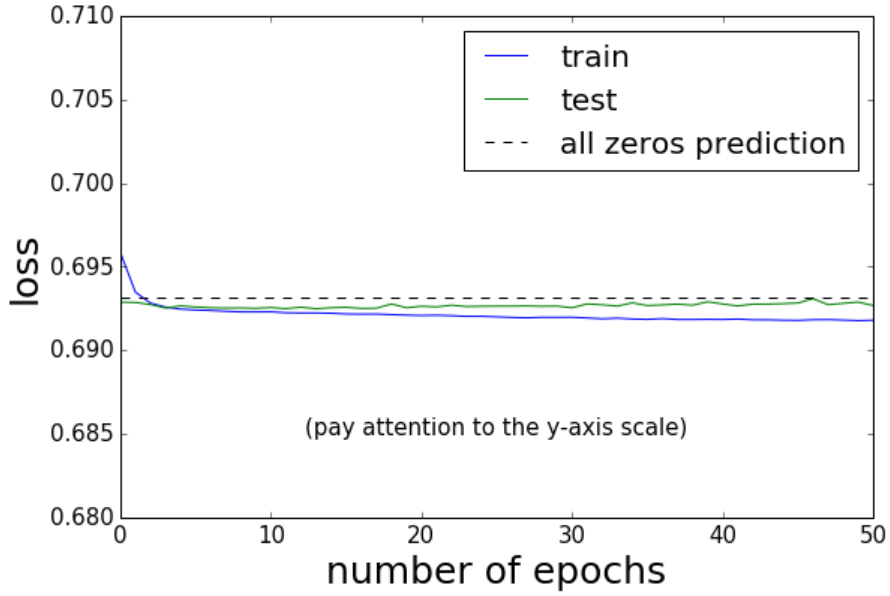
Neural networks

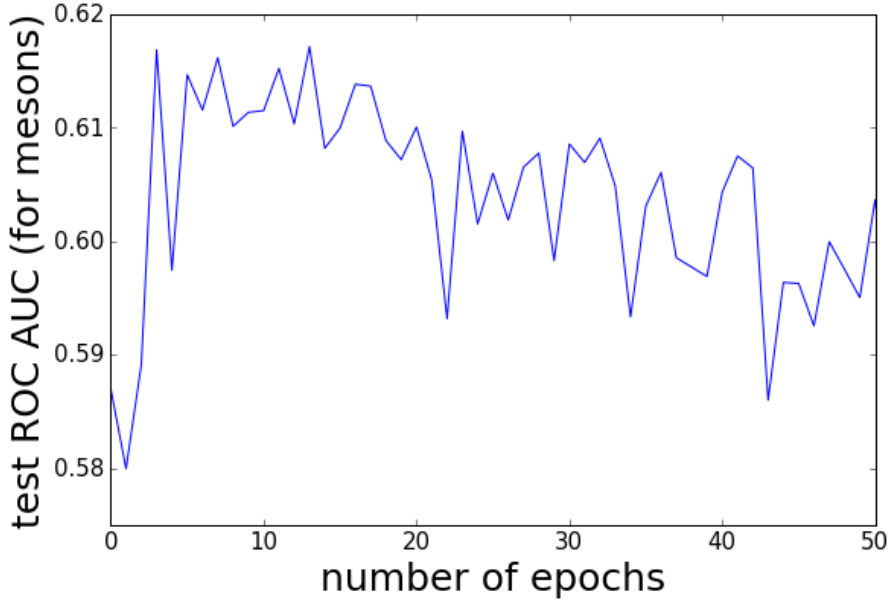
- typically, NNs are trained with adaptive versions of stochastic gradient descent.
  Each 'step' uses small subsample of data (which has few tagging tracks), thus steps are quite 'noisy'.

- speed of training should be significantly decreased to achieve stable training,

- example on plots is Adam with learning rate = 0.00005, negligible variations of loss drive to huge difference in final metric
  (see backup to compare with boosting behavior)

- to the contrast, batch optimization methods allow to reliably achieve same or better result quite effortlessly





Comparison

Results of a fairly simple comparison (1'000'000 tracks in training, 10'000'000 testing)

model	ROC AUC
Baseline OS*	0.579
XGBoost	0.646
DecisionTrain	0.654
NN + Adam	0.617
NN + iRprop–	0.620

- * Baseline OS was trained on a larger dataset (around 15 mln tracks before preselections)
  - SS tagging improves quality a little bit
- IRprop– (Impoved Rprop) is a batch optimization algorithm, which ignores scales of the gradient components.

Visual attention

A woman is throwing a frisbee in a park.

A little girl sitting on a bed with a teddy bear.

The following idea got its inspiration from visual attention — a technique used to recognize multiple objects.

Attention has applications in machine translation and image generation.

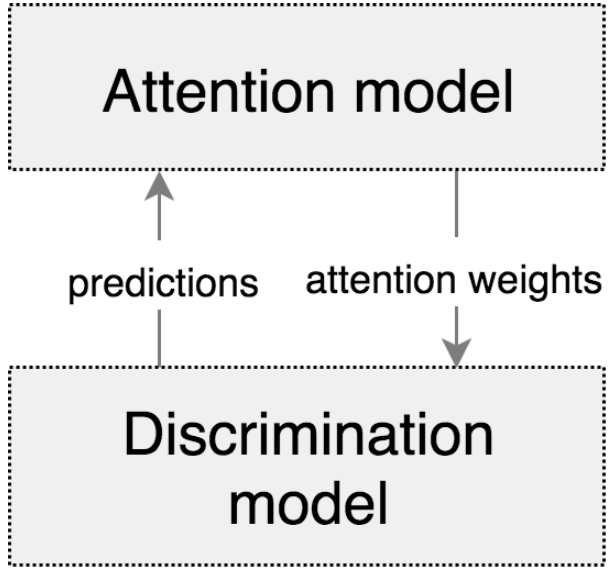
Idea: get better tagging by paying more attention to relevant tracks

Pictures taken from arxiv:1502.03044

Tagging with attention

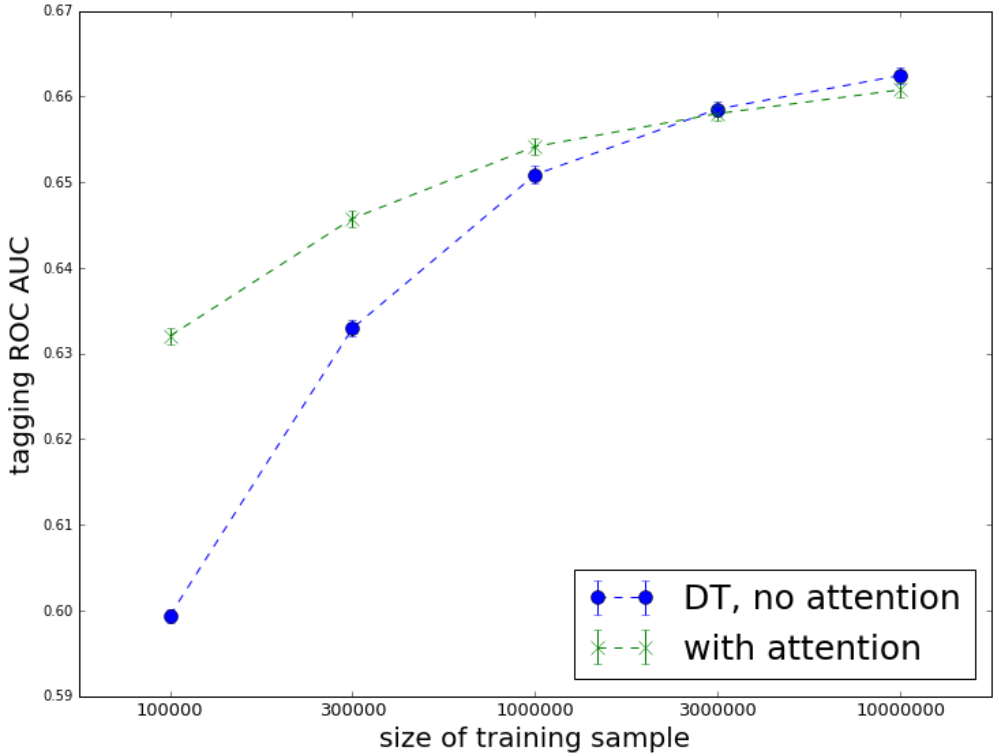
Tagging with attention incorporates two models: soft attention and discriminator, learnt together

- reduce the noise by independently guessing which tracks are irrelevant
- this helps ML to focus on tagging tracks, while allows making errors on irrelevant
- attention is trained to predict which tracks give high correct contribution
- soft attention works by assigning weights in a softmax-like manner
  - weights are used in combining and in training of discriminator



$$\frac{p(B^+|tracks)}{p(B^-|tracks)} = \prod_{track} \left[\frac{p_{\text{discriminator}}(track|B^+)}{p_{\text{discriminator}}(track|B^-)} \right]^{\text{attention(track)}}$$

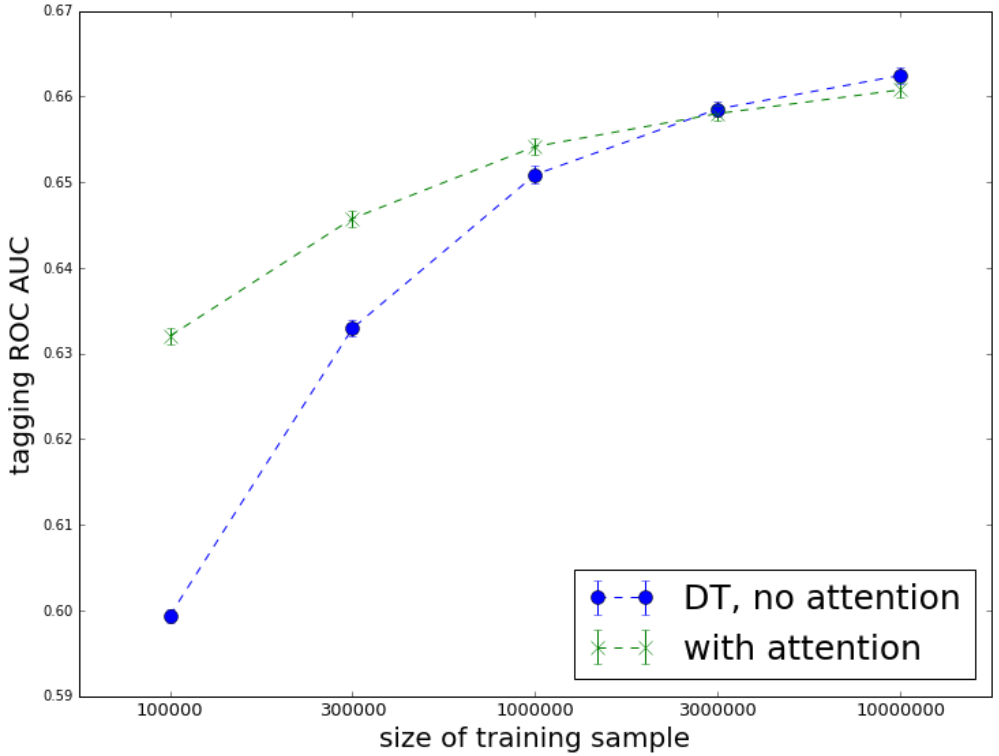
Tagging with attention: results



- works very nicely on small training datasets
- 100'000 tracks $\rightarrow$ already impressive result

But loses on large datasets.

Tagging with attention: results



- works very nicely on small training datasets
- 100'000 tracks $\rightarrow$ already impressive result

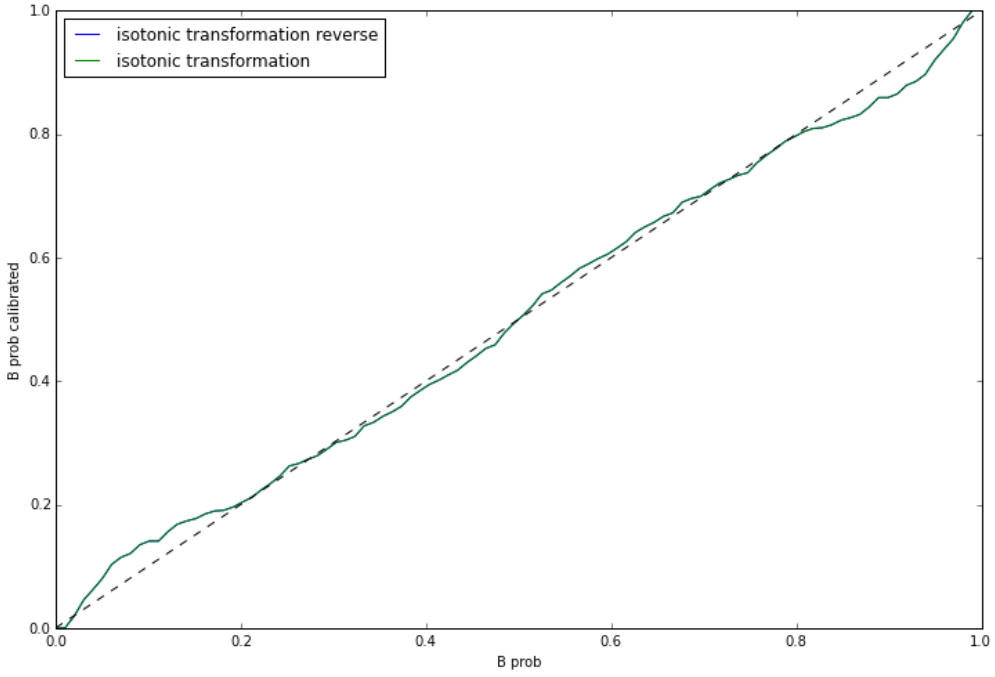
But loses on large datasets.

Reason: $\frac{s}{\sqrt{b}}$

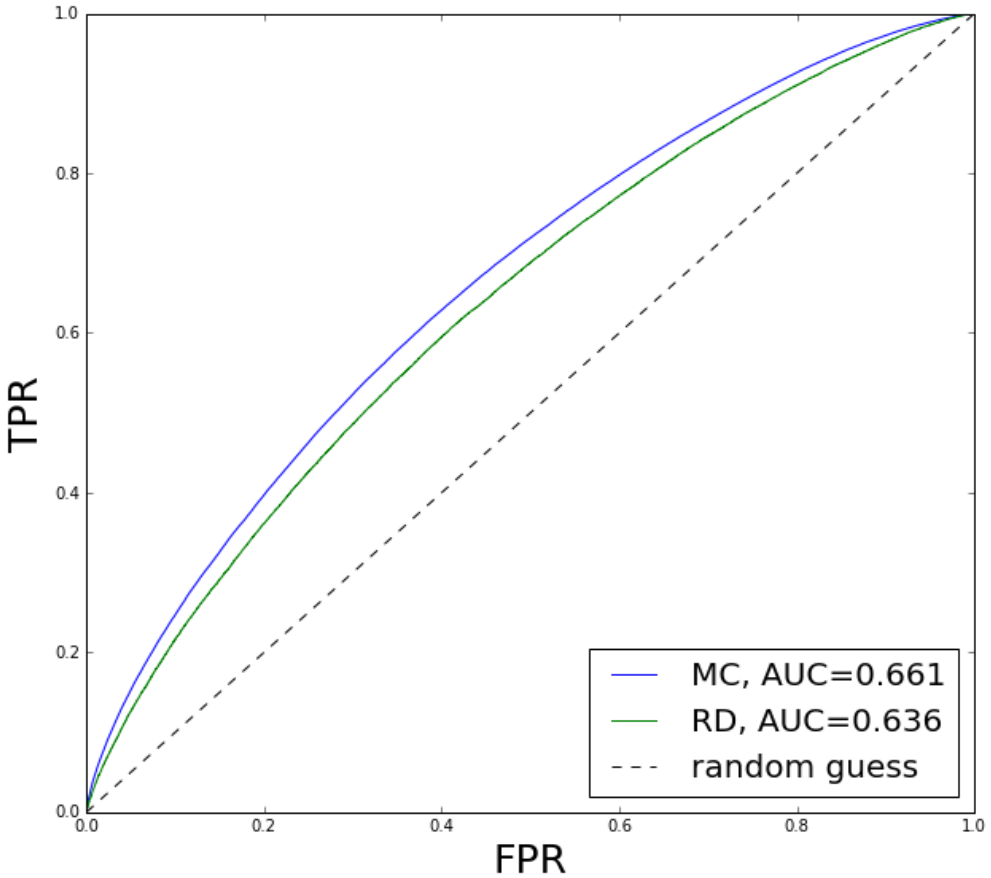
As the dataset grows, significance of correct decision rules (e.g. splits) grows as well, and (probably) there is no need in assistance with attention.

Important details

- tagging application requires well-calibrated probabilities, while ROC curve ignores this
  - calibrating is required after combining probabilities
  - optionally, per-track predictions can be calibrated before combining
  - logistic regression and isotonic regression on different steps
  - isotonic regression reconstructs very complex calibrating rules, but requires much data
- more about calibration in slides



Did we forget something?



- when the model trained on MC is applied to real data (RD), quality is still high, but quite different
- MC / RD difference is a very important factor and should be encountered

Instead of conclusion

- simplistic assumptions + probability theory can give huge boost
    - let machine learning solve the whole problem is a nice idea
- most appropriate models for particular task aren't necessarily fashionable
- large amount of data helps to fight with dominating noise,
  but it is not the only way

Thanks to LHCb tagging WG and Thomas Bayes :)

The end!

thanks for you visual attention

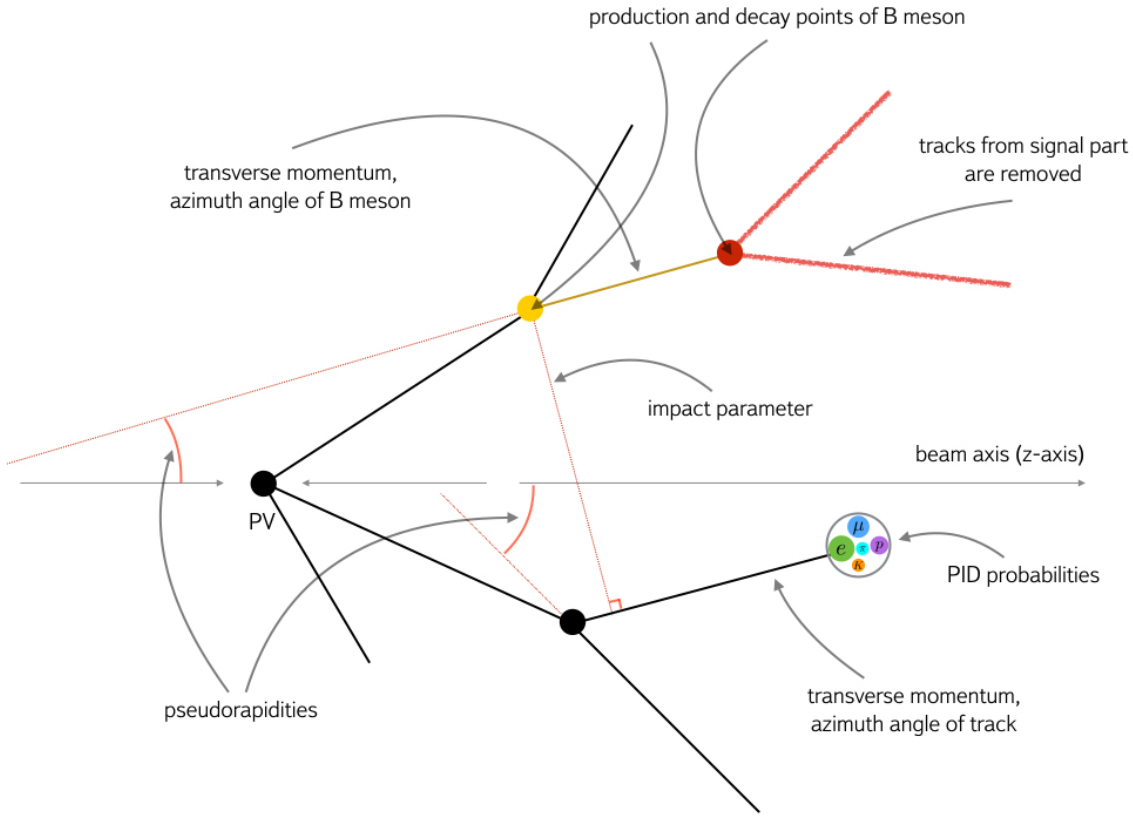
Links

- Inclusive Flavour Tagging, paper and poster,
  where probabilistic model was introduced

- T.Hastie, R.Tibshirani, J.Friedman, *Elements of statistical learning*,
  chapter "Naive Bayes Classifier", which was a start point for the probabilistic model

- C. Olah, S.Carter, Attention and Augmented Recurrent Neural Networks

Backup: features used in model



Additional simple feature engineering was done (like IP significance, R-separation between track and B-meson)

Backup

learning curve for DecisionTrain
(for comparison with Adam-trained NNs)

Quality computed after each 50 trees.