NON-TRIVIAL APPLICATIONS OF BOOSTING:

REWEIGHTING DISTRIBUTIONS WITH BDT, BOOSTING TO UNIFORMITY

Alex Rogozhnikov
YSDA, NRU HSE

Heavy Flavour Data Mining workshop, Zurich, 2016

Yandex

SCHOOL OF DATA ANALYSIS

BOOSTING

- a family of machine learning algorithms which convert weak learners to strong ones
- usually built over decision trees
- state-of-art results in many areas
- usually general-purpose implementations are used for classification and regression
- how can we get more by adapting boosting for our applications?

ADABOOST (ADAPTIVE BOOSTING)

- the first famous example of boosting
- building weak learners one-by-one, predictions are summed up:

$$D(x) = \sum_j \alpha_j d_j(x)$$

- each time increase weights of events incorrectly classified by tree $d(x)$

$$w_i \leftarrow w_i \exp\left(-\alpha y_i d(x_i)\right)), \;\; y_i = \pm 1$$

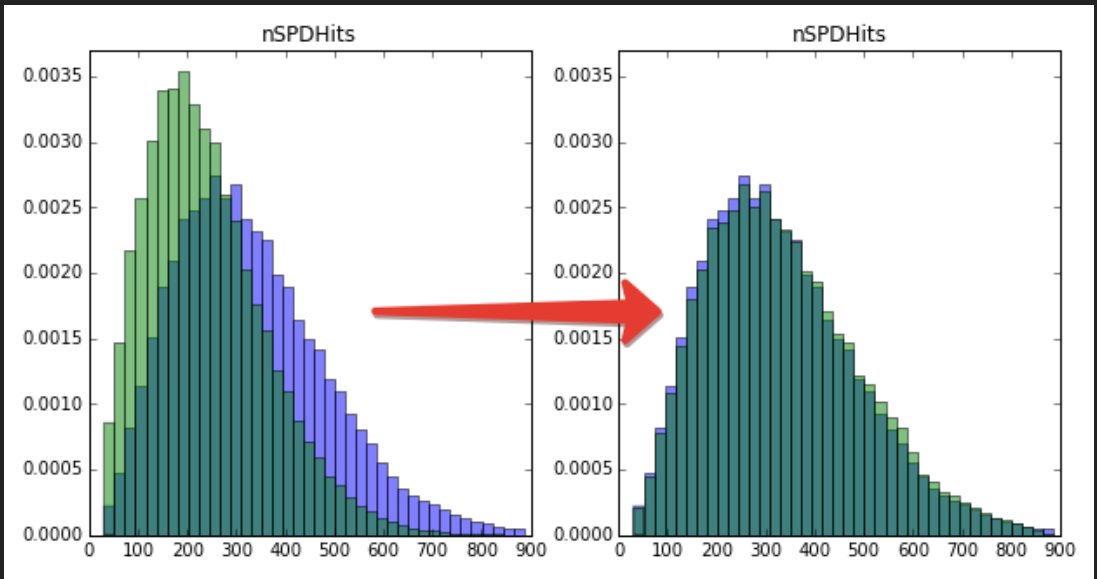
- main idea: provide base estimator (weak learner) with information about which samples have higher importance

REWEIGHTING OF DISTRIBUTIONS

Reweighting in HEP is used to minimize difference between real data (RD) and Monte-Carlo (MC) simulation.

Known process is used, for which real data can be obtained.

Goal of reweighting:
assign weights to MC s.t. MC and RD distributions coincide:



APPLICATIONS BEYOND PHYSICS

Introducing corrections to fight non-response bias: assigning higher weight to answers from groups with low response.

See e.g. R. Kizilcec, "Reducing non-response bias with survey reweighting: Applications for online learning researchers", 2014.

I'll talk in physical terms of original (MC) and target (RD) distributions, since this is applicable to any reweighting.

TYPICAL APPROACH

Usually histogram reweighting is used: the variable(s) is splitted into bins, in each bin the weight of original distribution is multiplied by:

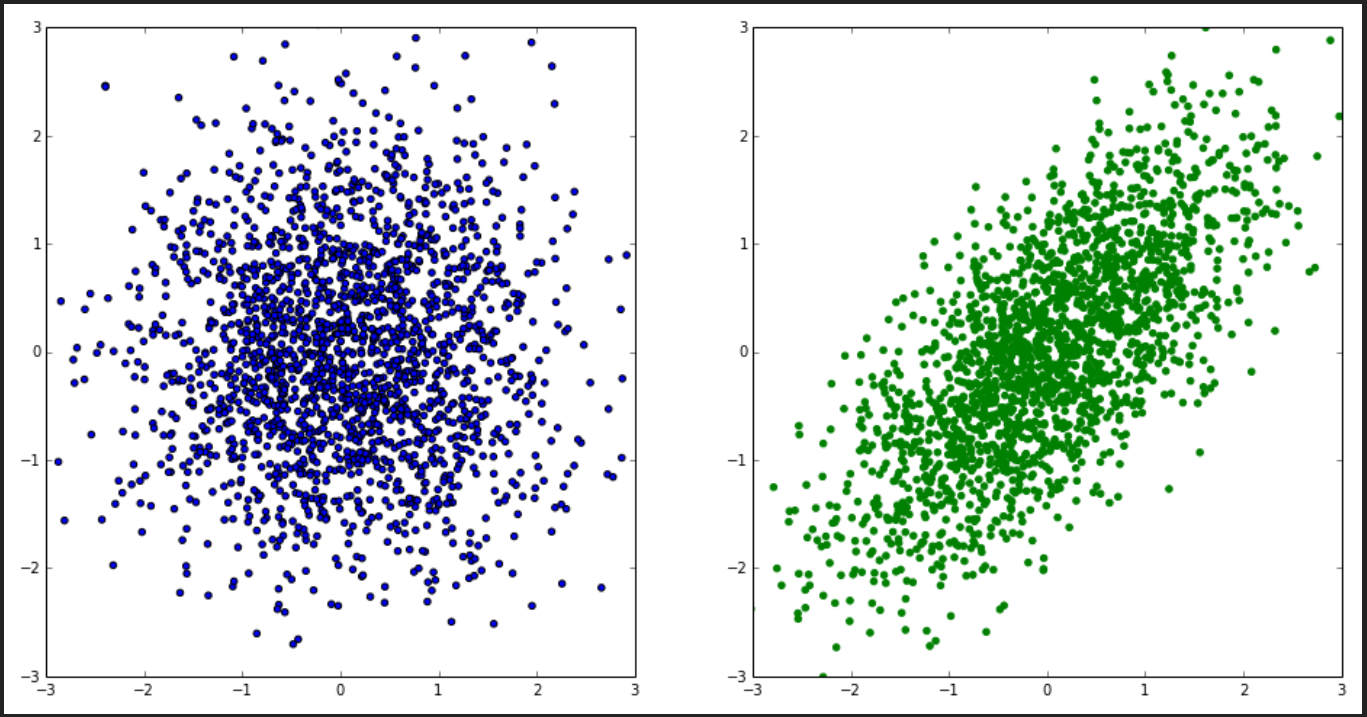
$$\text{multiplier}_{\text{bin}} = \frac{w_{\text{target, bin}}}{w_{\text{original, bin}}}$$

$w_{\text{target, bin}}$, $w_{\text{original, bin}}$ — total weight of events in bin for target and original distributions.

1. Simple and fast!
2. Very few (typically, one or two) variables
3. Reweighting one variable may bring disagreement in others
4. Which variable to use in reweighting?

COMPARING DISTRIBUTIONS

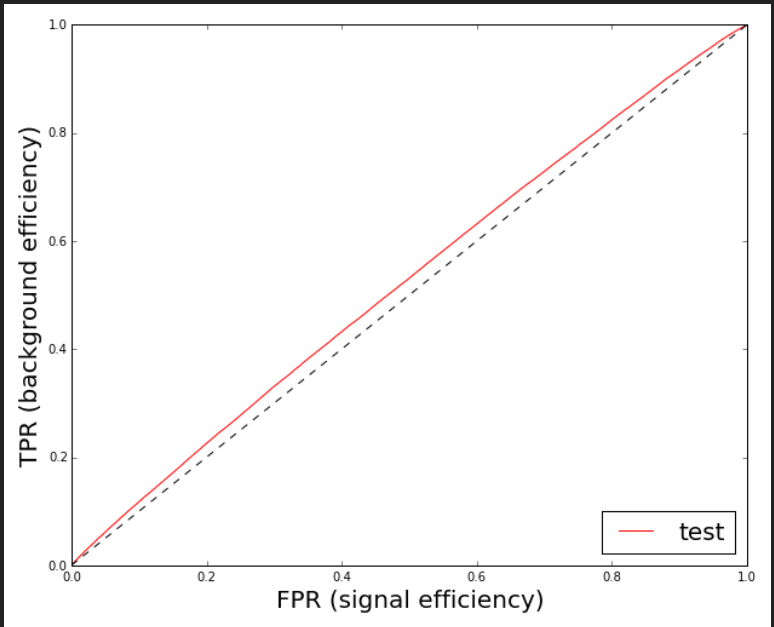
- i.e. to check the quality of reweighting
- one dimension: KS-test, CvM, Mann-Whitney (U-test)
- two or more dimensions?
- comparing 1d projections is not the way:



COMPARING DISTRIBUTIONS USING ML

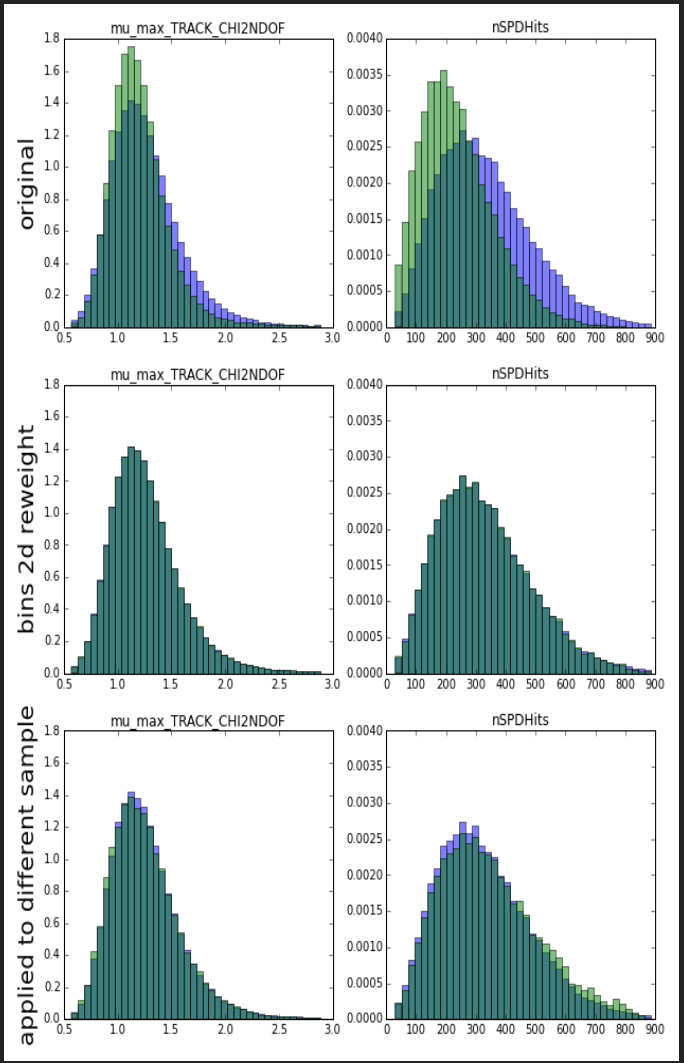
Final goal: define if the classifier discriminates RD and MC.

Comparison shall be done using ML, output of classifier is 1-dimensional. Looking at ROC curve on a holdout:



See also: J. Friedman, On Multivariate Goodness–of–Fit and Two–Sample Testing, 2003

REWEIGHTING WITH HISTOGRAMS: EXAMPLE



Problems arise when there are too few events in bin.

Those can be detected on a holdout.

Issues:

1. few bins — rule is rough
2. many bins — rule is not reliable

REUSING ML CLASSIFIERS TO REWEIGHTING

We need to estimate density ratio $\frac{f_1(x)}{f_0(x)}$

Classifier trained to discriminate MC and RD should reconstruct probabilities $p_0(x)$ and $p_1(x)$.

So, for reweghting we can use $\frac{p_1(x)}{p_0(x)} \sim \frac{f_1(x)}{f_0(x)}$

- able to reweight in many variables
- successfully tried in HEP, see D. Martschei et al, "Advanced event reweighting using multivariate analysis", 2012
- poor reconstruction when ratio is too small / high
- slow

BETTER IDEA:

Write ML algorithm to solve directly reweighting problem

1. Split space of variables in several large regions
2. Find this regions 'intellectually'!

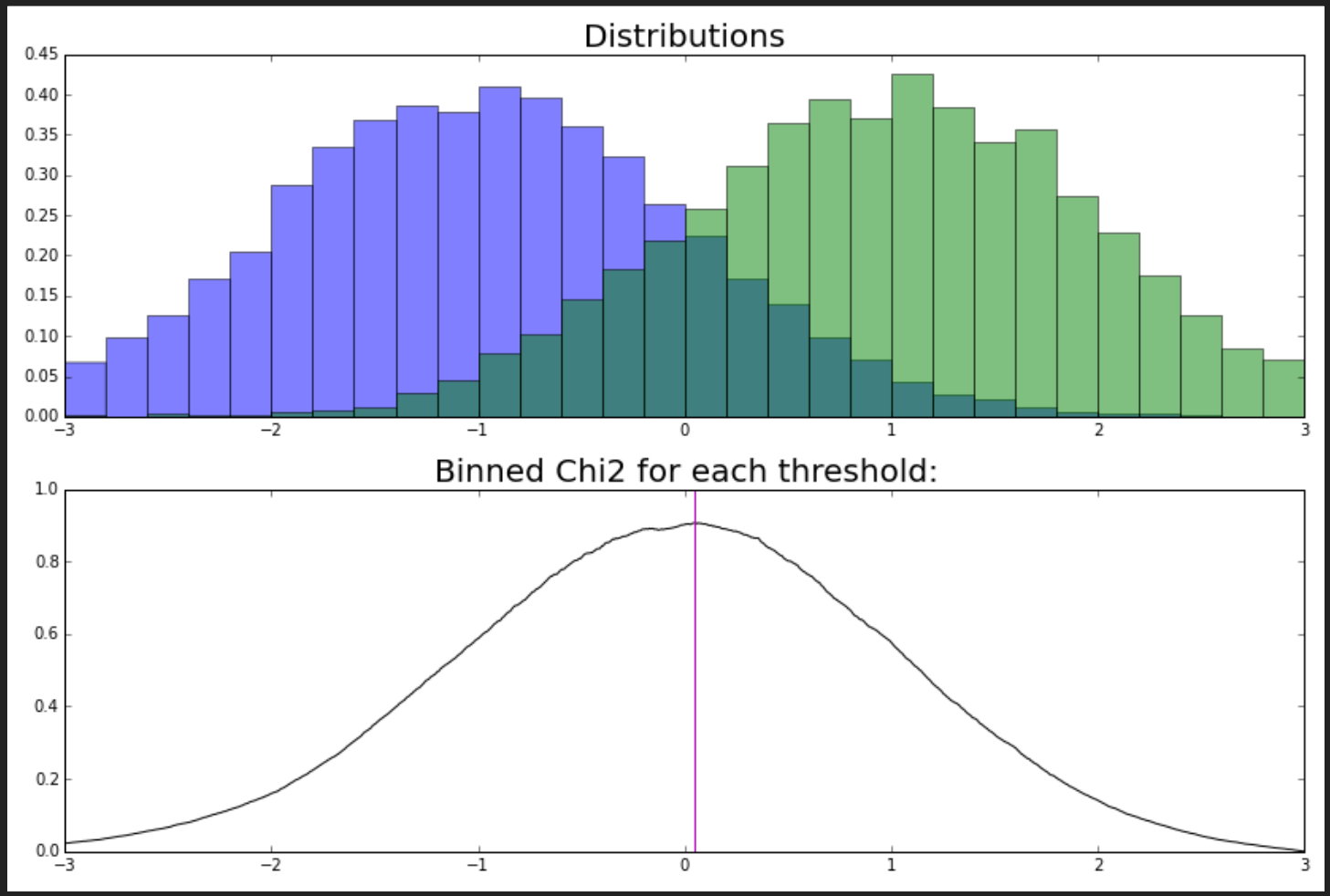
USING DECISION TREE TO FIND REGIONS

1. Tree splits the space of variables by orthogonal splits
2. Finding regions with high difference by maximizing symmetrized χ^2:

$$\chi^2 = \sum_{\text{leaf}} \frac{(w_{\text{leaf, original}} - w_{\text{leaf, target}})^2}{w_{\text{leaf, original}} + w_{\text{leaf, target}}}$$

SYMMETRIZED BINNED χ^2

Finding optimal threshold to split variable into two bins:



BDT REWEIGHTER

To train reweighter many times repeat following steps:

1. build a shallow tree to maximize symmetrized χ^2
2. compute predictions in leaves:

$$\text{leaf\_pred} = \log \frac{w_{\text{leaf, target}}}{w_{\text{leaf, original}}}$$

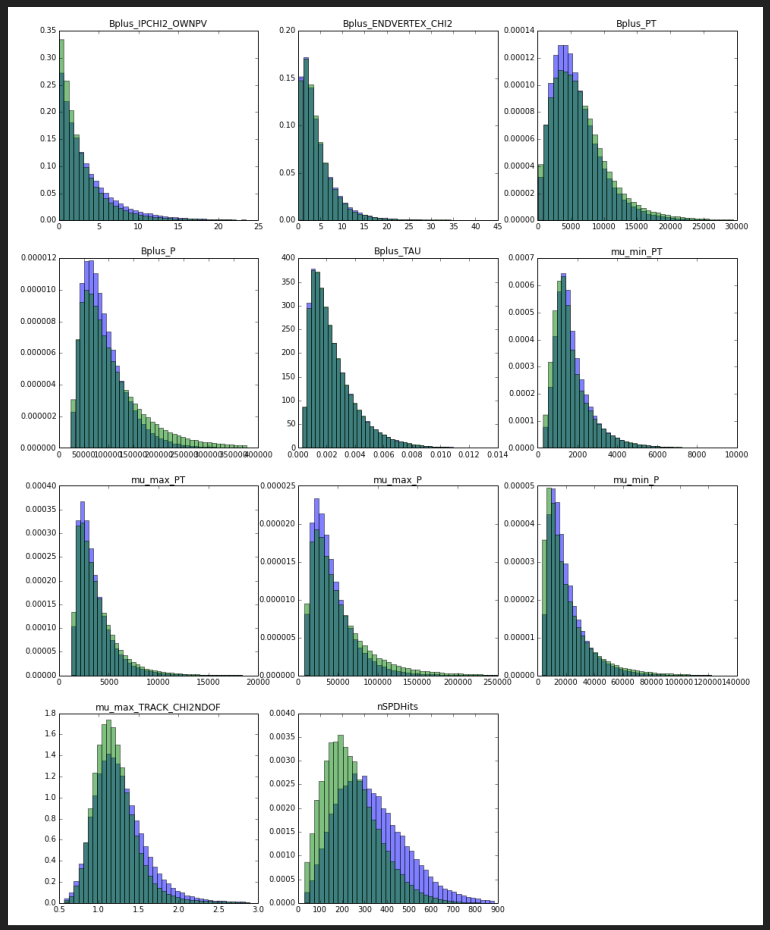
3. reweight distributions (compare with AdaBoost):

$$w = \begin{cases} w, & \text{if event from target (RD) distribution} \\ w \times e^{\text{pred}}, & \text{if event from original (MC) distribution} \end{cases}$$

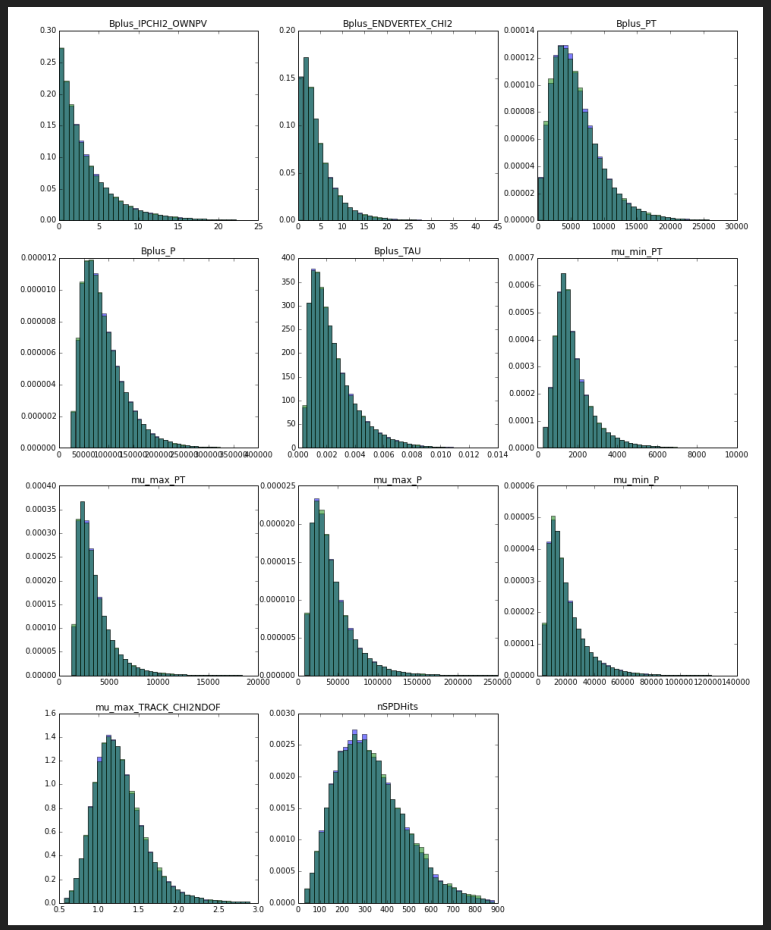
CHANGES COMPARED TO GBDT

› tree splitting criterion (MSE $\rightarrow \chi^2$)
› different boosting procedure

DEMONSTRATION



original



result of BDT reweighter

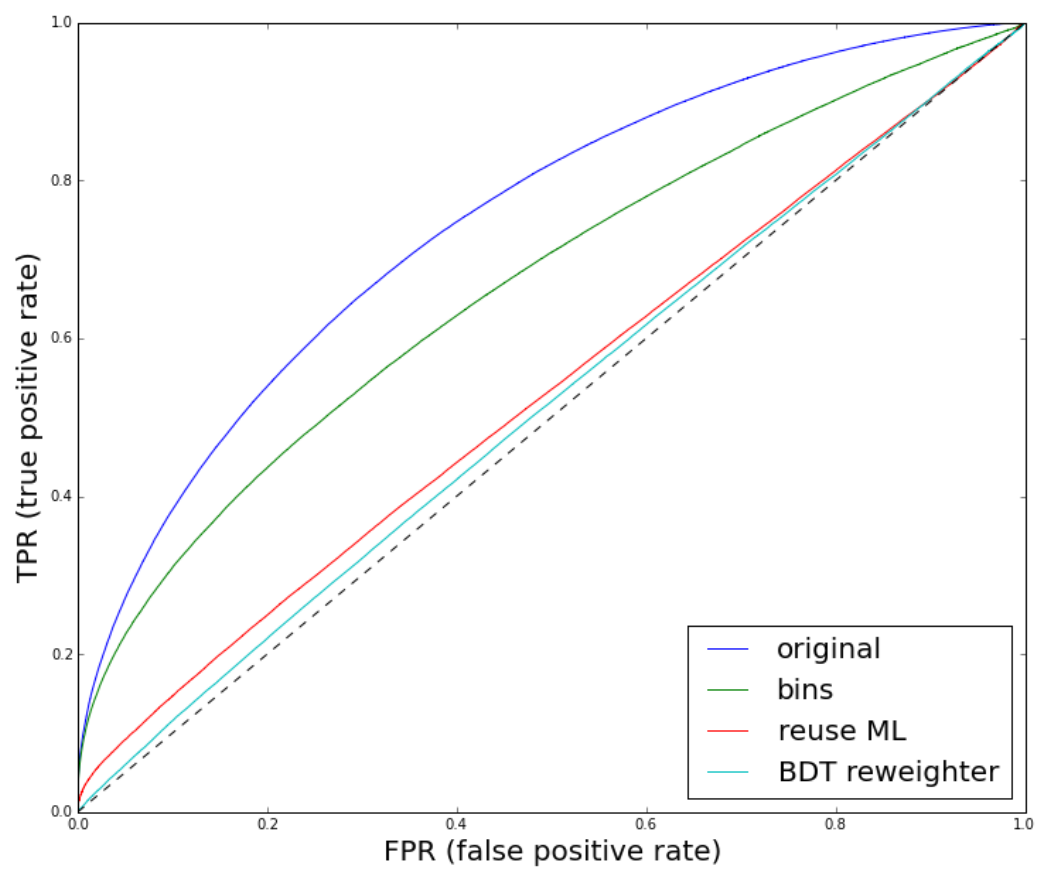
KS DISTANCES AFTER REWEIGHTING

Bins reweighter uses only 2 last variables (60×60 bins); ML and BDT use all variables

	original	bins	reuse ML	BDT reweighter
Feature				
Bplus_IPCHI2_OWNPV	0.0796	0.0642	0.0463	0.0028
Bplus_ENDVERTEX_CHI2	0.0094	0.0175	0.0490	0.0021
Bplus_PT	0.0586	0.0679	0.0126	0.0053
Bplus_P	0.1093	0.1126	0.0044	0.0047
Bplus_TAU	0.0037	0.0060	0.0324	0.0044
mu_min_PT	0.0623	0.0604	0.0017	0.0036
mu_max_PT	0.0483	0.0561	0.0053	0.0035
mu_max_P	0.0906	0.0941	0.0084	0.0036
mu_min_P	0.0845	0.0858	0.0058	0.0043
mu_max_TRACK_CHI2NDOF	0.0956	0.0042	0.0128	0.0043
nSPDHits	0.2478	0.0098	0.0180	0.0075

COMPARING RESULTS WITH ML

Using approach to distribution comparison described earlier.
Reweighting two variables wasn't enough:



FEATURE IMPORTANCES

Being a variation of GBDT, BDT reweighter is able to calculate feature importances. Two features used in reweighting with bins are indeed most important

	importance
feature	
mu_max_TRACK_CHI2NDOF	0.240272
nSPDHits	0.209090
Bplus_P	0.122314
mu_min_P	0.115245
Bplus_PT	0.080641
Bplus_IPCHI2_OWNPV	0.068209
mu_max_P	0.060518
mu_max_PT	0.037863
mu_min_PT	0.037761
Bplus_ENDVERTEX_CHI2	0.026598
Bplus_TAU	0.001489

BOOSTED REWEIGHTING

- uses each time few large bins
- is able to handle many variables
- requires less data (for same performance)
- ... but slow

SUMMARY ABOUT REWEIGHTING

1. Comparison of multidimensional distributions is ML problem
2. Reweighting of distributions is ML problem

RECALL: GRADIENT BOOSTING

Gradient boosting greedily builds an ensemble of estimators (regressors)

$$D(x) = \sum_j \alpha_j d_j(x)$$

by optimizing some loss function. Those could be:

- MSE: $\mathcal{L} = \sum_i (y_i - D(x_i))^2$
- AdaLoss: $\mathcal{L} = \sum_i e^{-y_i D(x_i)}, y_i = \pm 1$
- LogLoss: $\mathcal{L} = \sum_i \log(1 + e^{-y_i D(x_i)}), y_i = \pm 1$

Next estimator in series approximates gradient of loss in the space of functions.

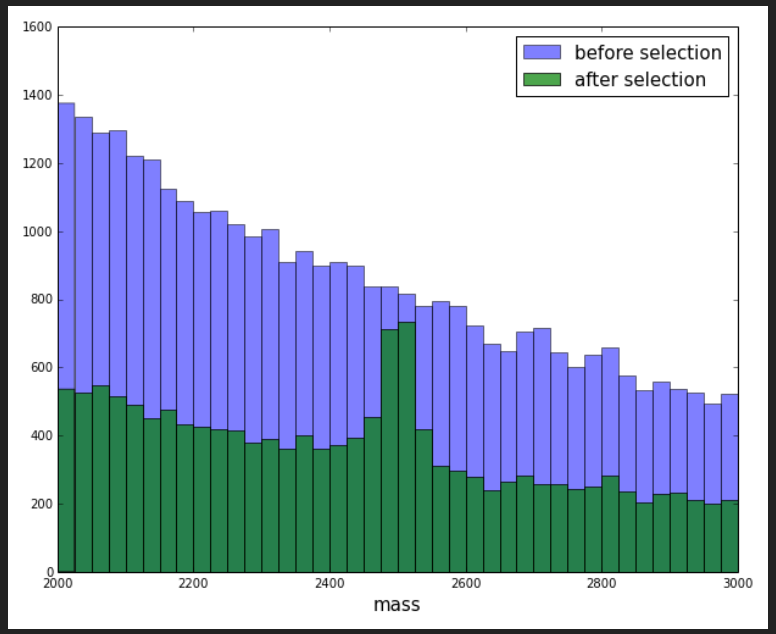
BOOSTING TO UNIFORMITY

Point of uniform boosting — have constant efficiency (FPR/TPR) against some variable.

Examples:

- flat background efficiency along mass
- flat signal efficiency for different flight time
- flat signal efficiency along Dalitz variables

EXAMPLE: NON-FLAT BACKGROUND EFFICIENCY (FPR) ALONG MASS



High correlation with mass will create from pure background false peaking signal (specially if we use mass sidebands for training)

Goal: $\mathrm{FPR} = const$ for different regions in mass.

Reminder: FPR — background efficiency

BASIC APPROACH

- reduce the number of features used in training
- leave only the set of features, which do not give enough information to reconstruct mass of particle
- simple and works
- sometimes we have to lose much information

Can we modify ML to use all features, but provide uniform background efficiency (FPR) along mass?

uBoostBDT

Aims to get $\mathrm{FPR}_{\mathrm{region}} = const.$

fix target efficiency (say $\mathrm{FPR}_{\mathrm{target}} = 30\%$), find corresponding threshold

- train a tree, its decision function $d(x)$
- increase weight for misclassification: $w_i \leftarrow w_i \exp(-\alpha y_i d(x)))$
- increase weight of signal events in regions with high FPR
  $w_i \leftarrow w_i \exp\left(\beta(\mathrm{FPR}_{\mathrm{region}} - \mathrm{FPR}_{target})\right)$

This way we achieve $\mathrm{FPR}_{\mathrm{region}} = 30\%$ in all regions only for some threshold on training dataset.

uBoost

uBoost is an ensemble of uBoostBDTs, each uBoostBDT uses own $\mathrm{FPR}_{\mathrm{target}}$ (all possible FPRs with step of 1%).

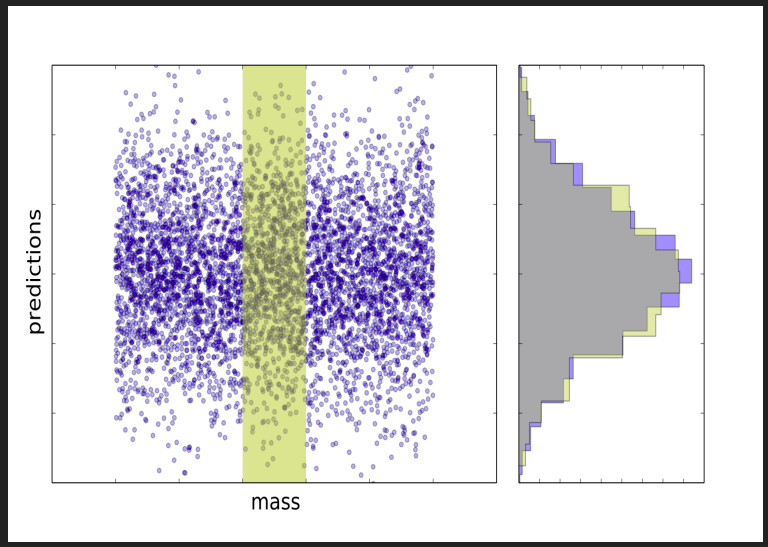
uBoostBDT returns 0 or 1 (passed or not the threshold corresponding to $\mathrm{FPR}_{\mathrm{target}}$), simple averaging is used to obtain predictions.

- drives to uniform selection
- very complex training
- many trees
- estimation of threshold in uBoostBDT may be biased

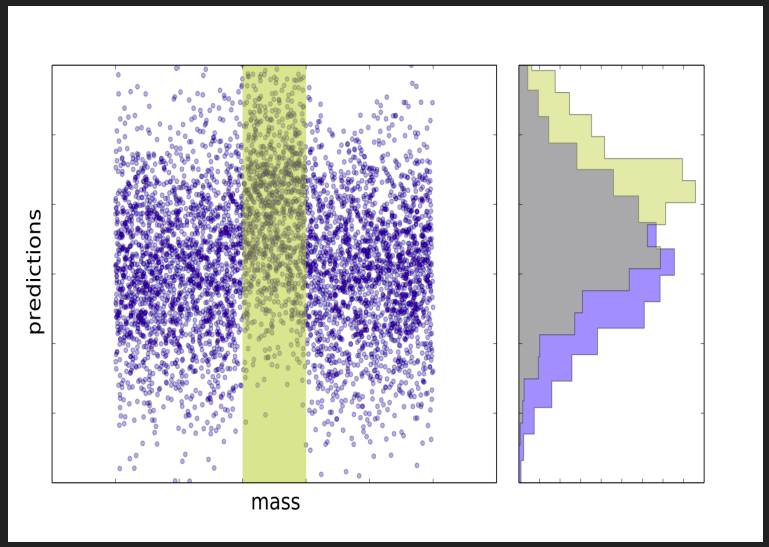
MEASURING NON-UNIFORMITY

› difference in efficiency can be detected by analyzing distributions

› uniformity ↔ no dependence between mass and predictions



Uniform predictions

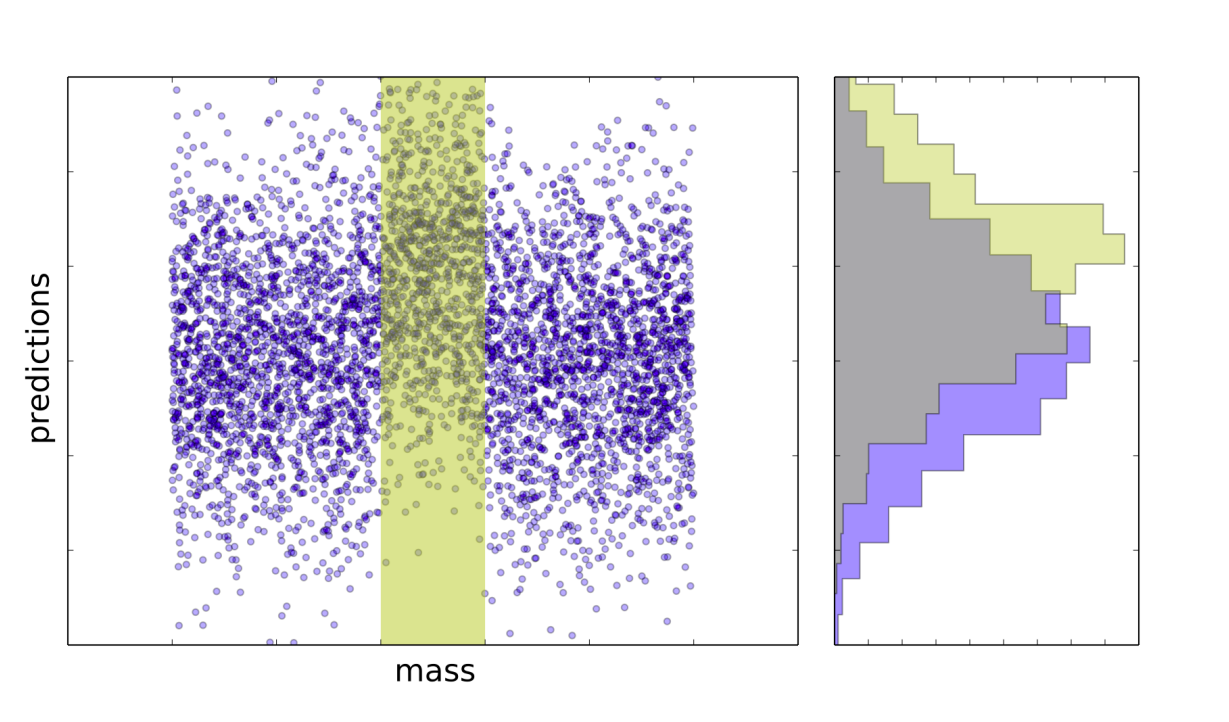


Non-uniform predictions
(peak in highlighted region)

MEASURING NON-UNIFORMITY

Sum up contributions from different regions in mass

$$\mathrm{CvM} = \sum_{region} \int \left| F_{\mathrm{region}}(s) - F_{\mathrm{global}}(s) \right|^2 dF_{\mathrm{global}}(s)$$



MINIMIZING NON-UNIFORMITY

- why not minimizing CvM inside with GB?
- ... because we can't compute gradient
- ROC AUC, classification accuracy are not differentiable too
- also, minimizing CvM doesn't encounter classification problem:

  the minimum of CvM is achieved i.e. on a classifier with random predictions

FLATNESS LOSS (FL)

Put an additional term in loss function which will penalize for non-uniformity

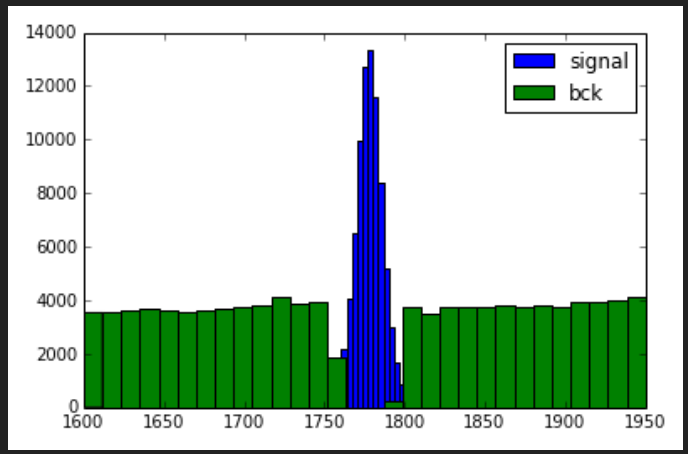
$$\mathcal{L} = \mathcal{L}_{\text{adaloss}} + c\mathcal{L}_{\text{FL}}$$

Flatness loss approximates (non-differentiable) CvM metrics:

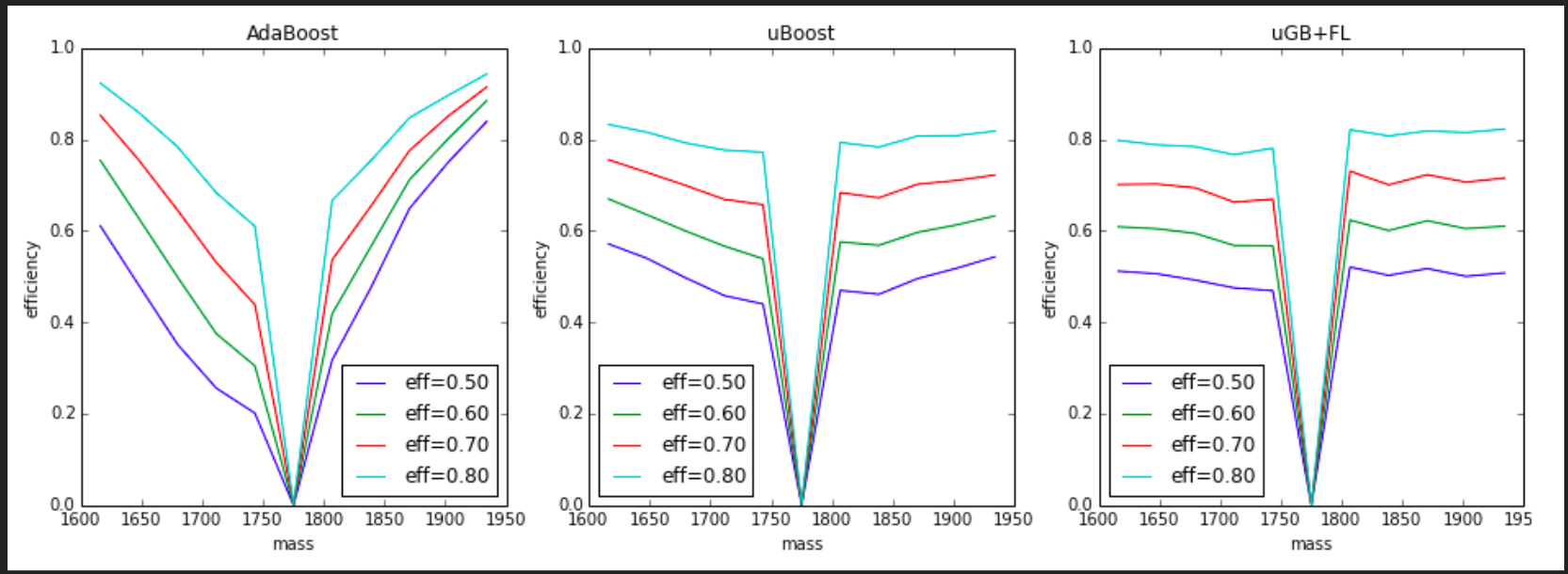
$$\mathcal{L}_{\text{FL}} = \sum_{region} \int \left| F_{\text{region}}(s) - F_{\text{global}}(s) \right|^2 ds$$

$$\frac{\partial}{\partial D(x_i)} \mathcal{L}_{\text{FL}} \cong 2(F_{\text{region}}(s) - F_{\text{global}}(s))\Big|_{s=D(x_i)}$$

EXAMPLE (EFFICIENCY OVER BACKGROUND)



- when we train on a sideband using many features, we easily can run into problems
- all models use same features for discrimination, but AdaBoost got serious correlation with mass



BOOSTING SUMMARY

- powerful general-purpose algorithm
- most known applications: classification, regression and ranking
- widely used, considered to be well-studied
- can be adapted to different specific scientific problems

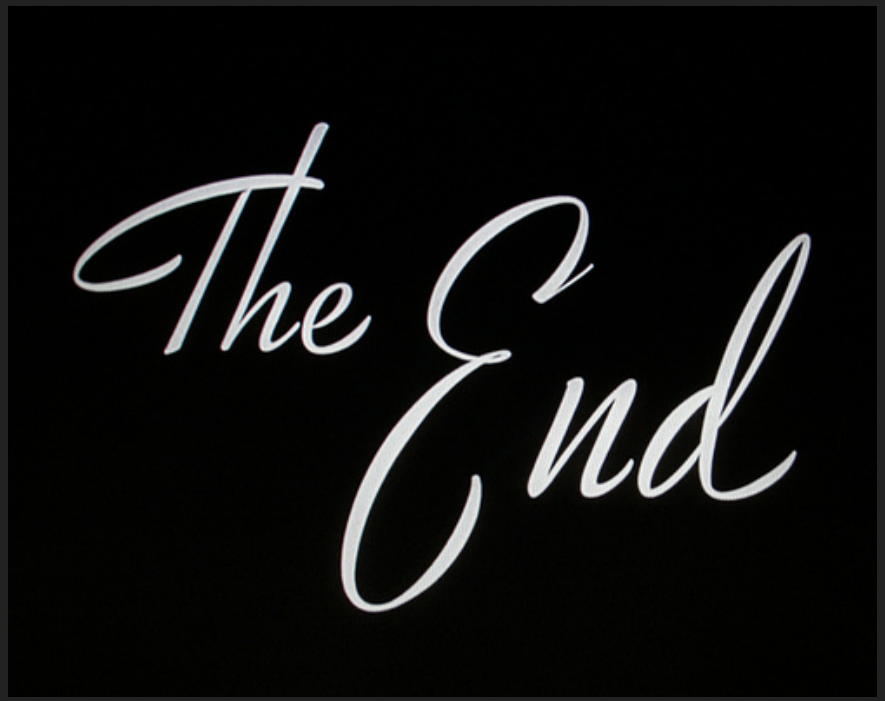
EXAMPLES DEMONSTRATED TODAY:

1. reweighting of multidimensional distributions
2. boosting to uniformity

These algorithms are available in hep_ml package (based on sklearn)

REFERENCES

- D. Martschei, M. Feindt, S. Honc, J. Wagner-Kuhr, Advanced event reweighting using multivariate analysis
- Reweighting with Boosted Decision Trees
- J. Stevens, M. Williams, uBoost: A boosting method for producing uniform selection efficiencies from multivariate classifiers
- A. Rogozhnikov, A. Bukva, V. Gligorov, A. Ustyuzhanin, M. Williams, New approaches for boosting to uniformity
- Implementations of algorithms: github repository


The End