

MACHINE LEARNING IN HIGH ENERGY PHYSICS

LECTURE #3



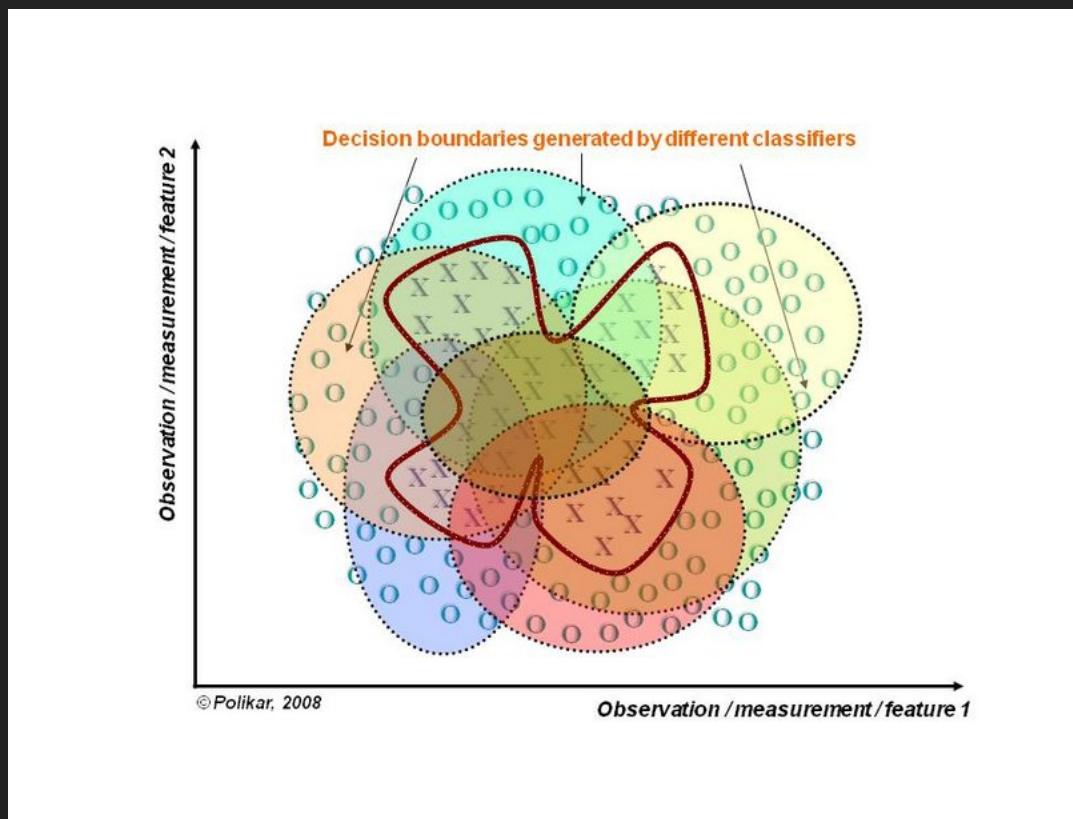
Alex Rogozhnikov, 2015

RECAPITULATION

- logistic regression
- overfitting (2 definitions)
- regularizations
- neural networks
- stochastic optimization
- decision tree for classification and regression

COMPOSITIONS

Basic motivation: improve quality of classification by reusing strong sides of classifiers.



SIMPLE VOTING

- Averaging predictions of

$$\hat{y} = [0, 1, 1, 1, 0] \Rightarrow P_1 = 0.6, P_0 = 0.4$$

- Averaging predicted probabilities

$$P_1 = \frac{1}{J} \sum_{j=1}^J p_{1,j}(x)$$

- Averaging decision function

$$D(x) = \frac{1}{J} \sum_{j=1}^J d_j(x)$$

WEIGHTED VOTING

The way to introduce importance of classifiers

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

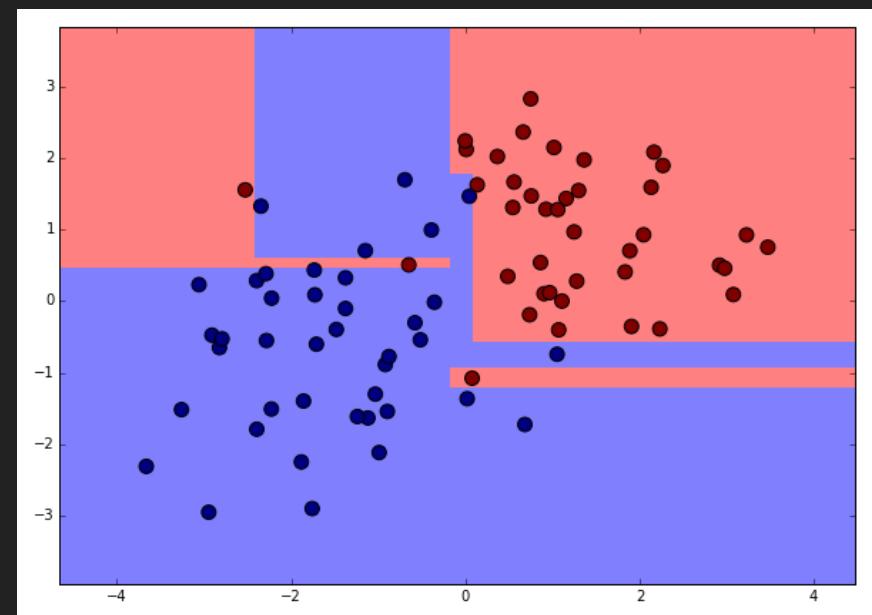
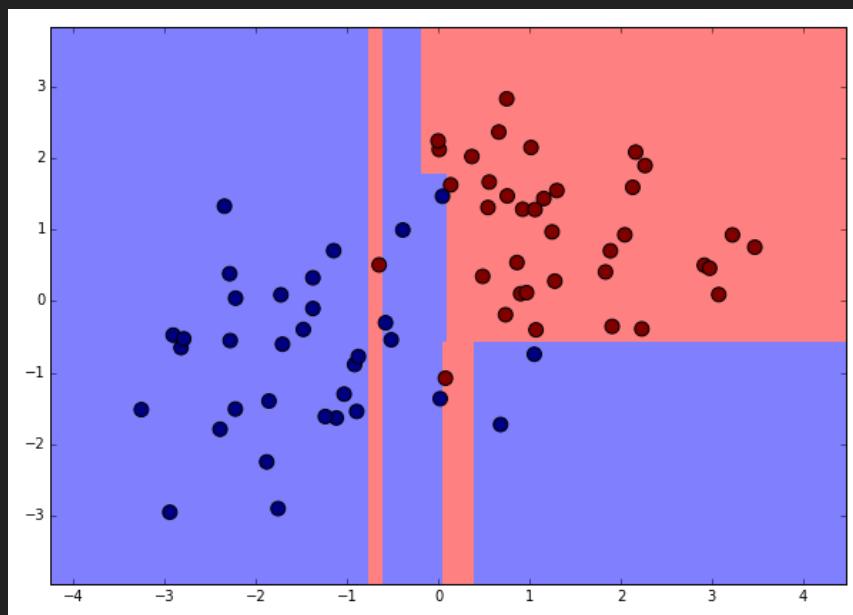
GENERAL CASE OF ENSEMBLING:

$$D(x) = f(d_1(x), d_2(x), \dots, d_J(x))$$

PROBLEMS

- very close base classifiers
- need to keep variation
- and still have good quality of basic classifiers

DECISION TREE



GENERATING TRAINING SUBSET

- **subsampling**
taking fixed part of samples (sampling without replacement)
- **bagging**(Bootstrap AGGregating) sampling with replacement,

If #generated samples = length of dataset, the fraction of unique samples in new dataset is $1 - \frac{1}{e} \sim 63.2\%$

BAGGING (BOOTSTRAP AGGREGATING)

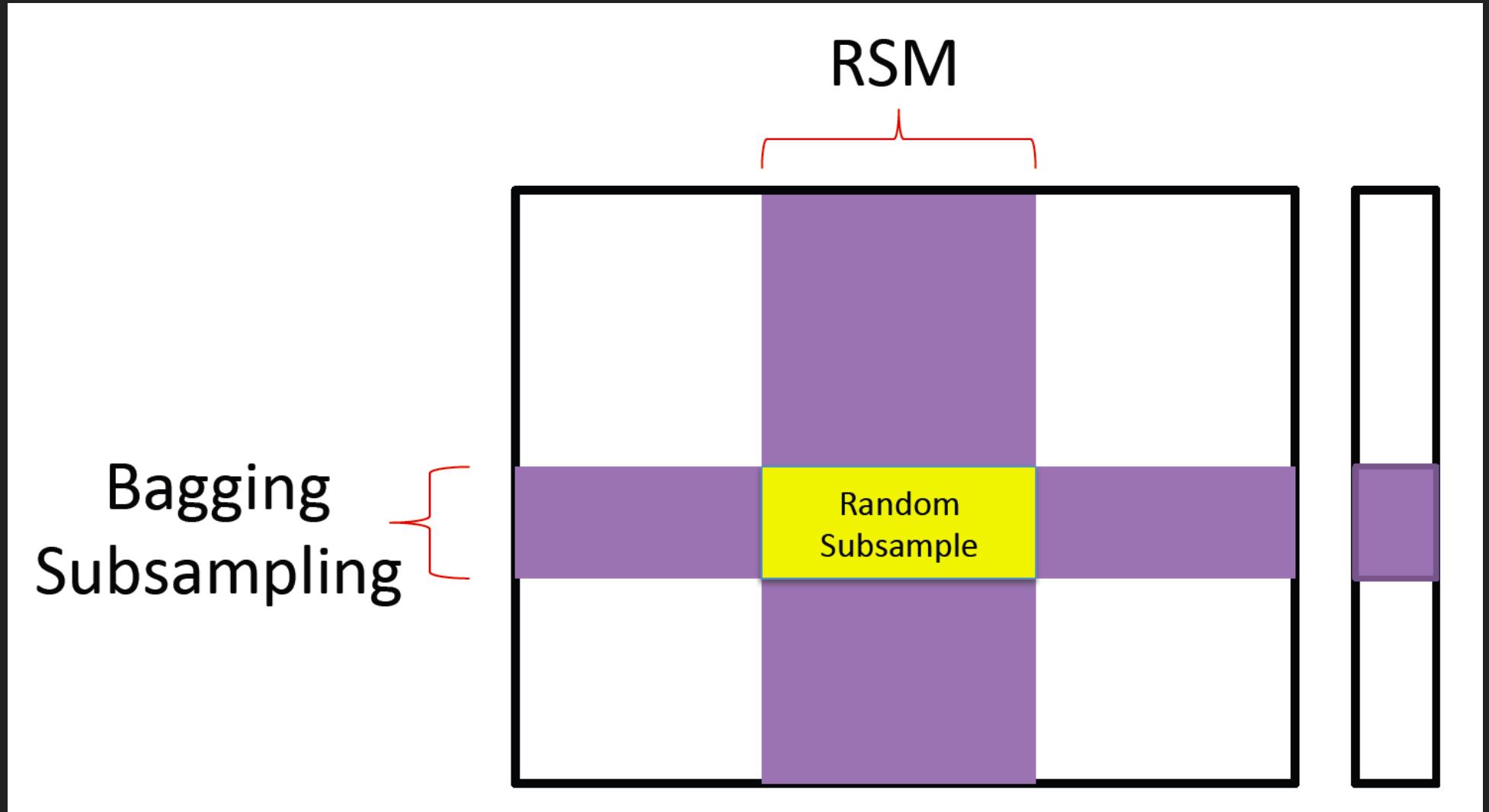
From training data with N samples independently take N samples (sampling with replacement).

The fraction of unique samples in new dataset is

$$1 - \frac{1}{e} \sim 63.2$$

RANDOM SUBSPACE MODEL (RSM)

Generating subspace of features by taking random subset of features



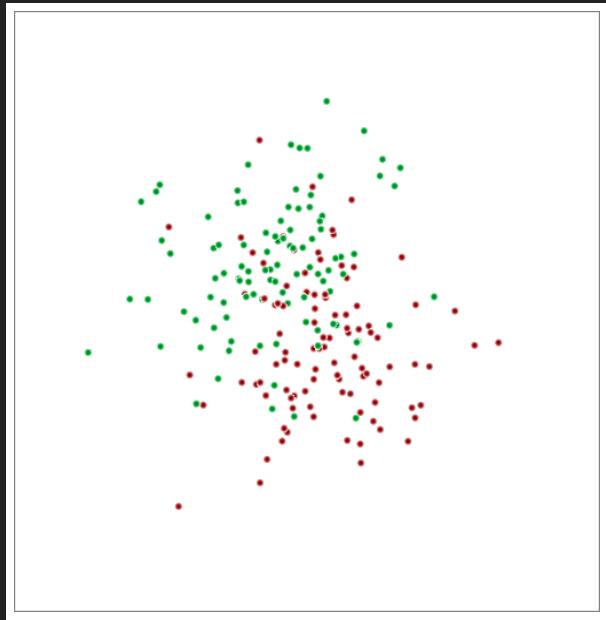
RANDOM FOREST [LEO BREIMAN, 2001]

Random forest is composition of decision trees.

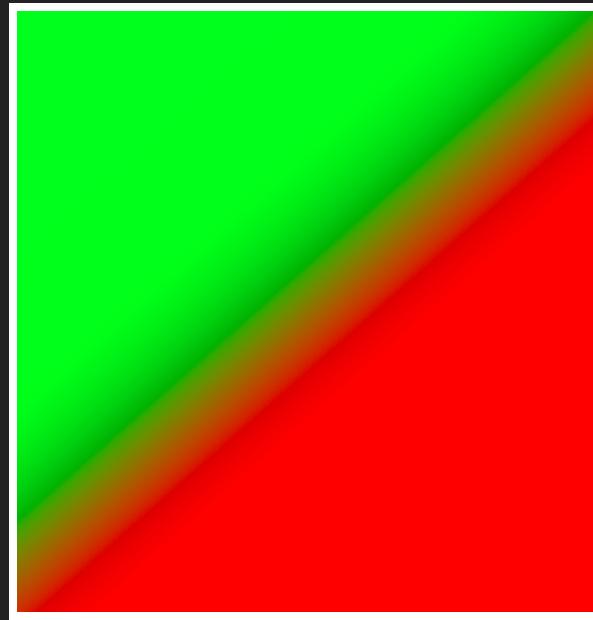
For each tree is trained by

- bagging samples
- taking m random features

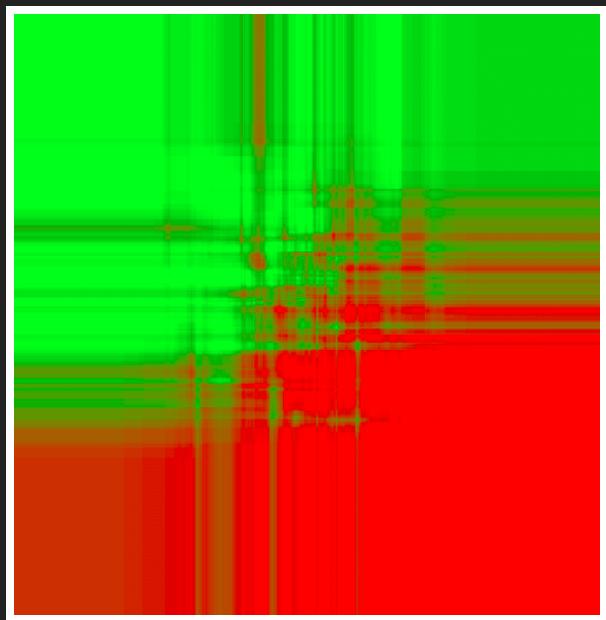
Predictions are obtained via simple voting.



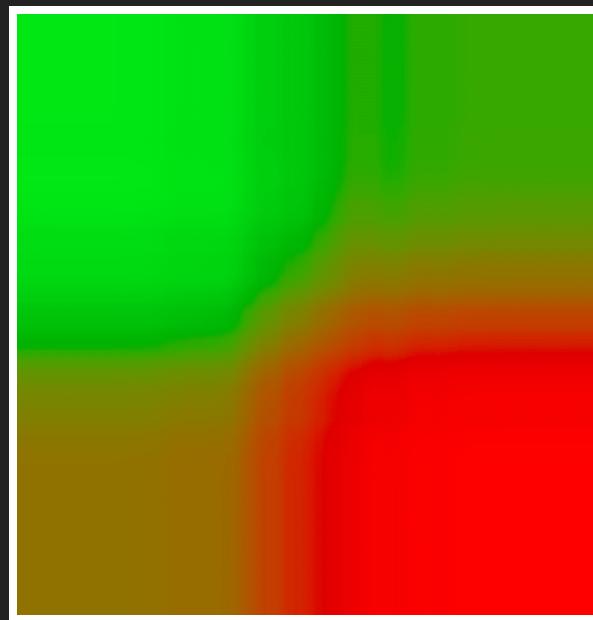
data



optimal boundary



feature scaling

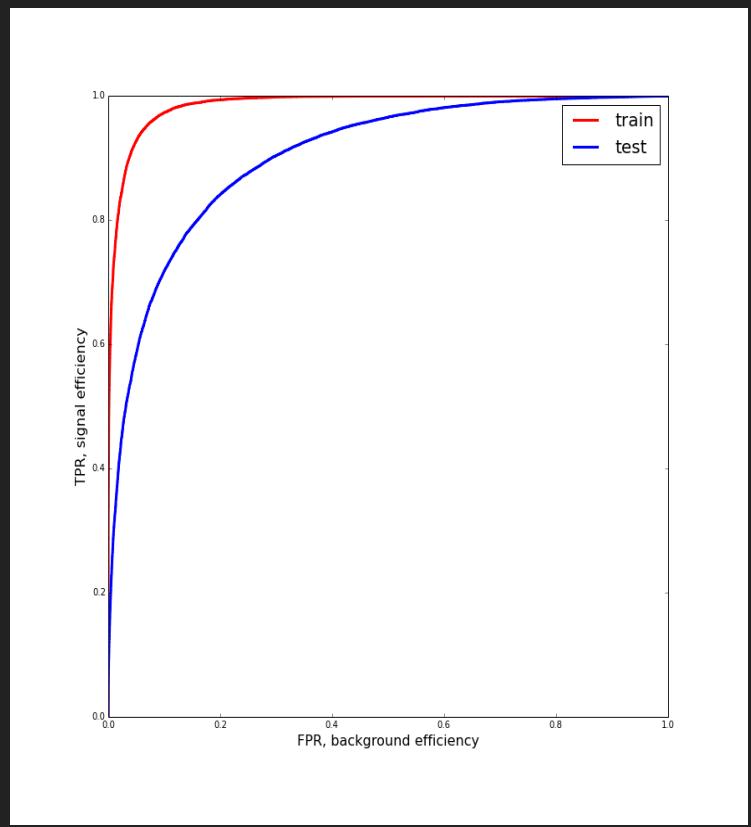
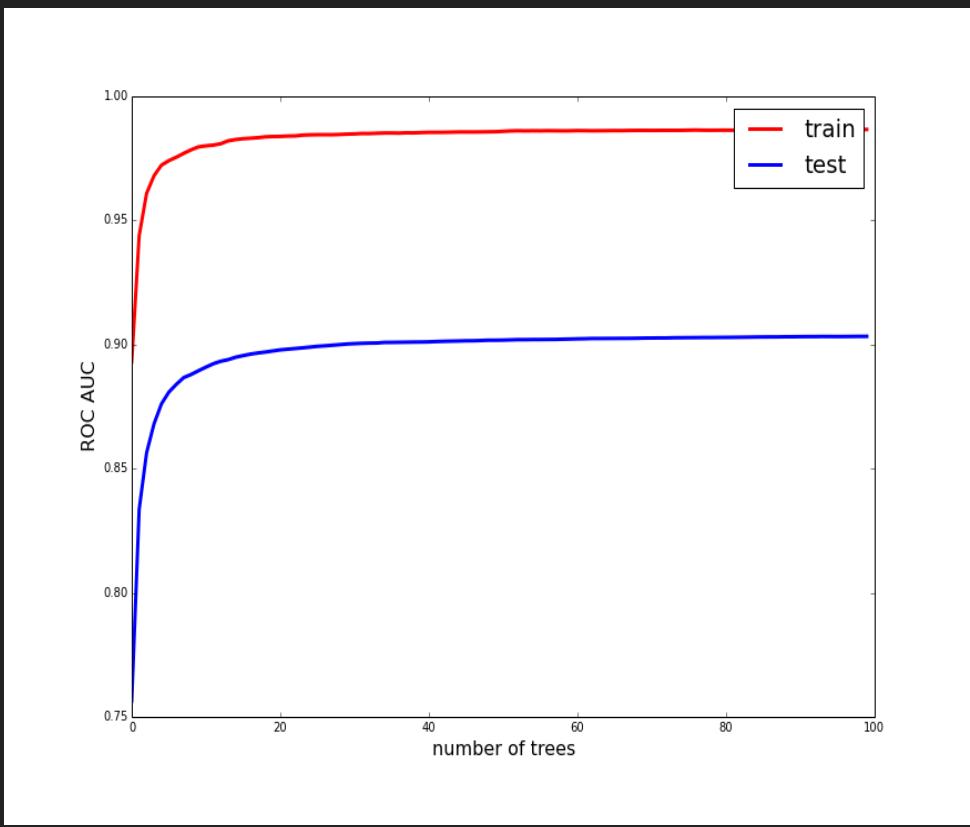


no feature scaling

50 trees

2000 trees

OVERFITTING



- overfitted (in the sense that predictions for train and test are different)
- **doesn't overfit:** increasing complexity (adding more trees) doesn't spoil classifier

- Works with features of different nature
- Stable to noise in data

From '[Testing 179 Classifiers on 121 Datasets](#)'

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

RANDOM FOREST SUMMARY

- Impressively simple
- Trees can be trained in parallel
- Doesn't overfit
- Doesn't require much tuning

Effectively only one parameter:

number of features used in each tree

Recommendation: $N_{\text{used}} = \sqrt{N_{\text{features}}}$

- Hardly interpretable

ADABOOST [FREUND, SHAPIRE, 1995]

Bagging: information from previous trees **not taken into account**.

Adaptive Boosting is weighted composition of weak learners:

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

We assume $d_j(x) = \pm 1$, labels $y_i = \pm 1$,
 j th weak learner misclassified i th event iff $y_i d_j(x_i) = -1$

ADABOOST

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

Weak learners are built in sequence

- each next classifier is trained using different weights
- initially $w_i = 1$ for each training sample
- After building j th tree:

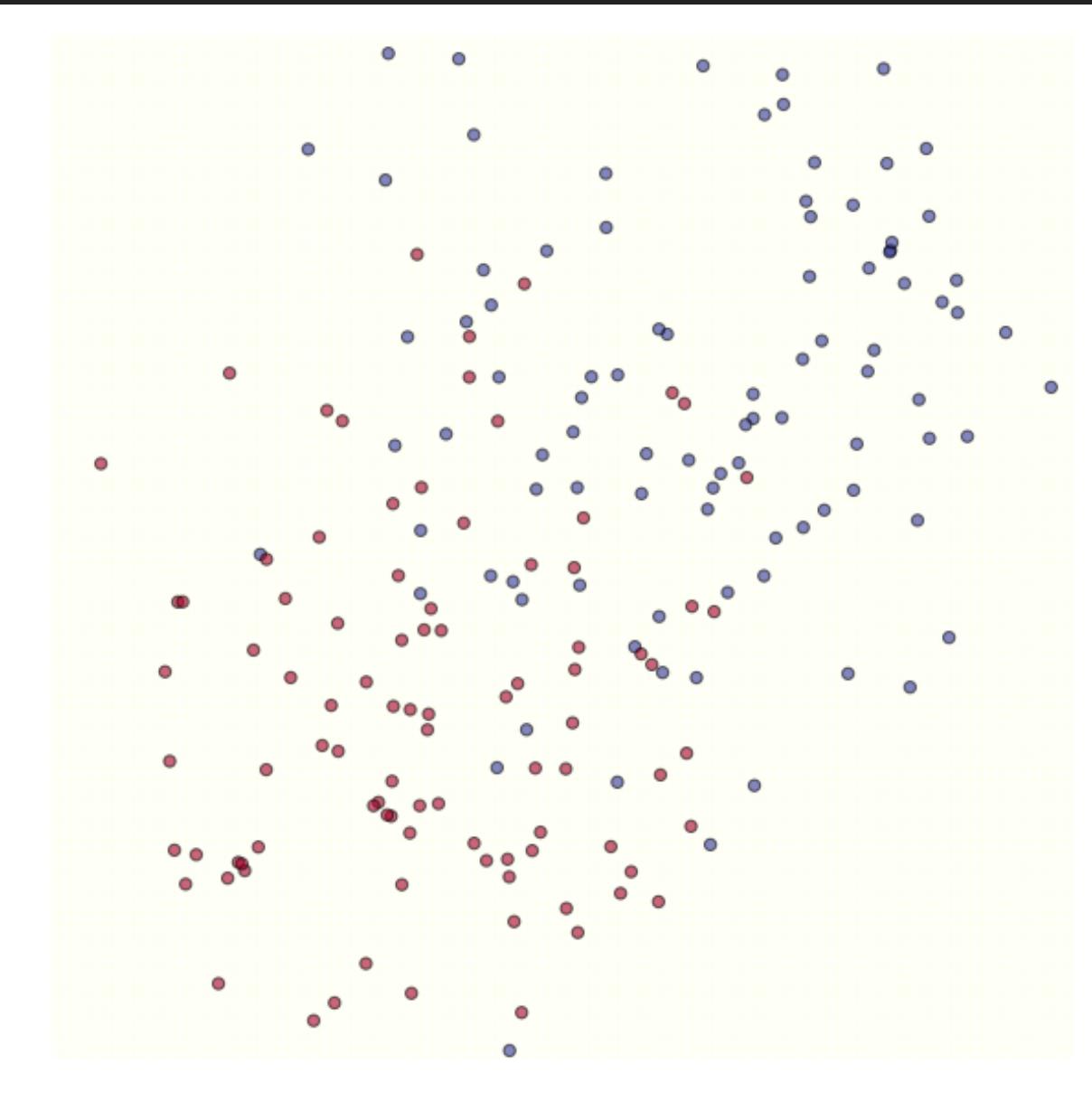
$$1. \alpha_j = \frac{1}{2} \ln \left(\frac{w_{\text{correct}}}{w_{\text{wrong}}} \right)$$

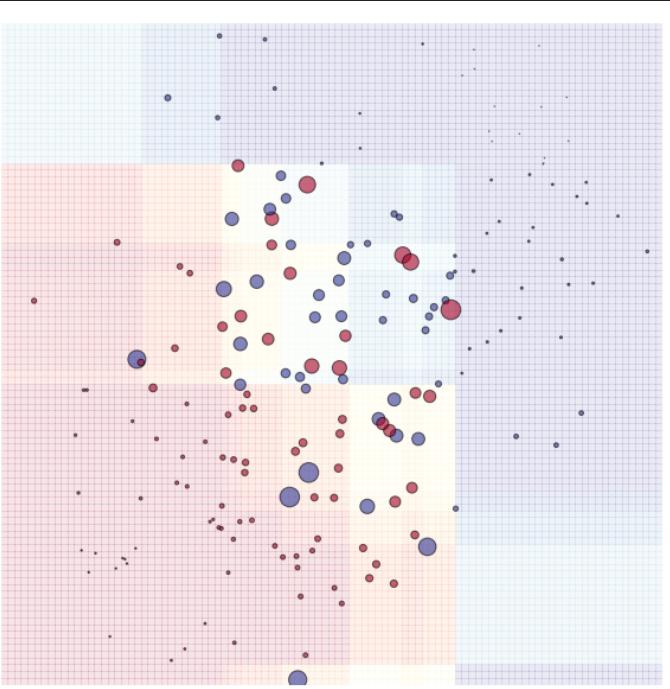
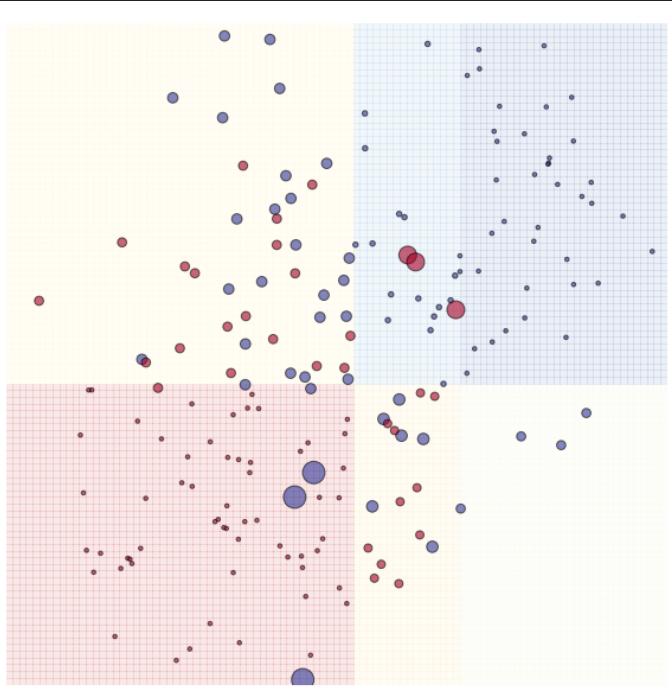
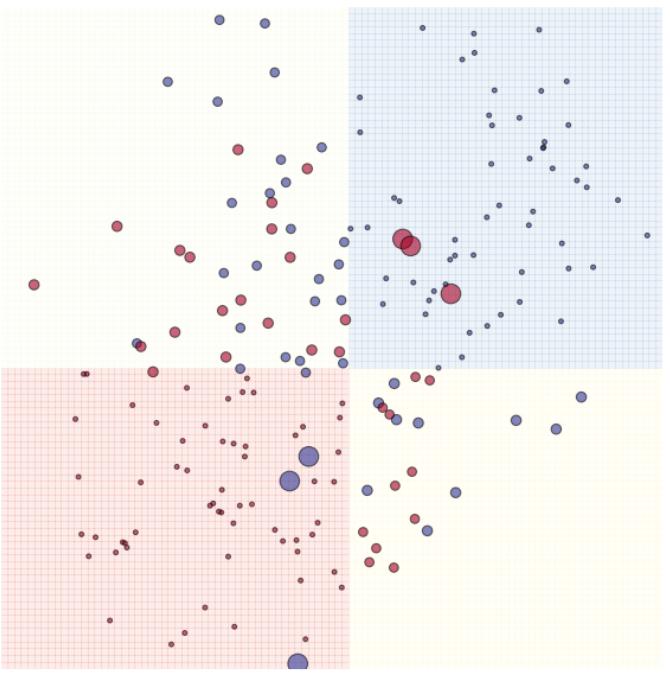
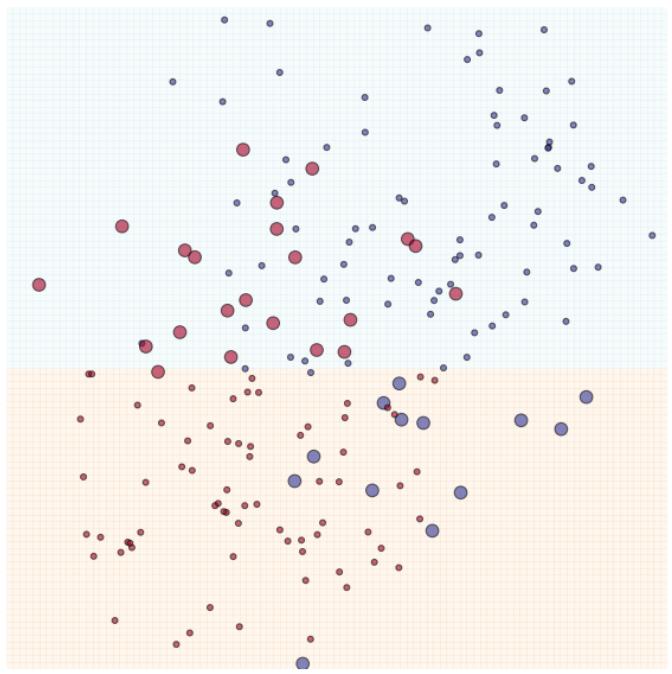
2. increase weight of misclassified

$$w_i \leftarrow w_i \times e^{-\alpha_j y_i d_j(x_i)}$$

ADABOOST EXAMPLE

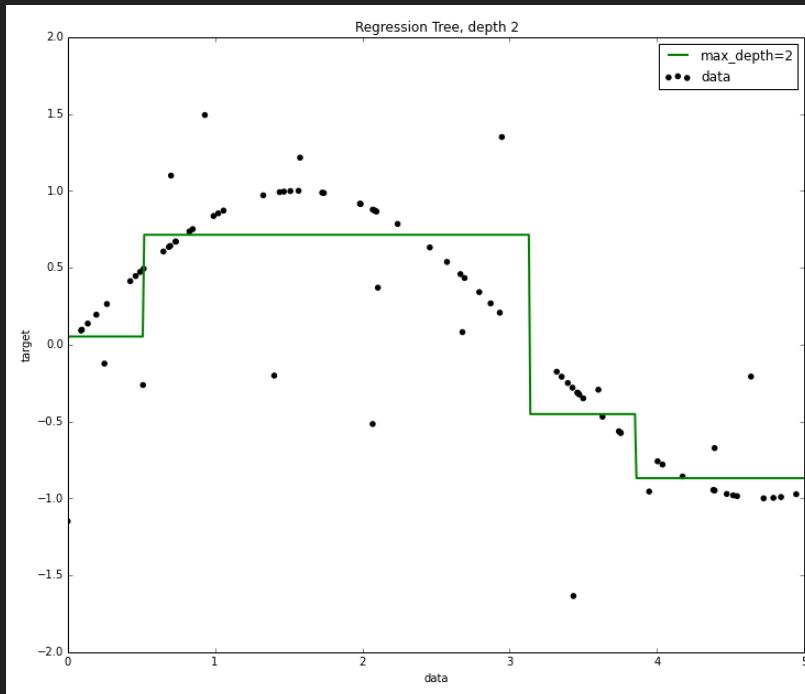
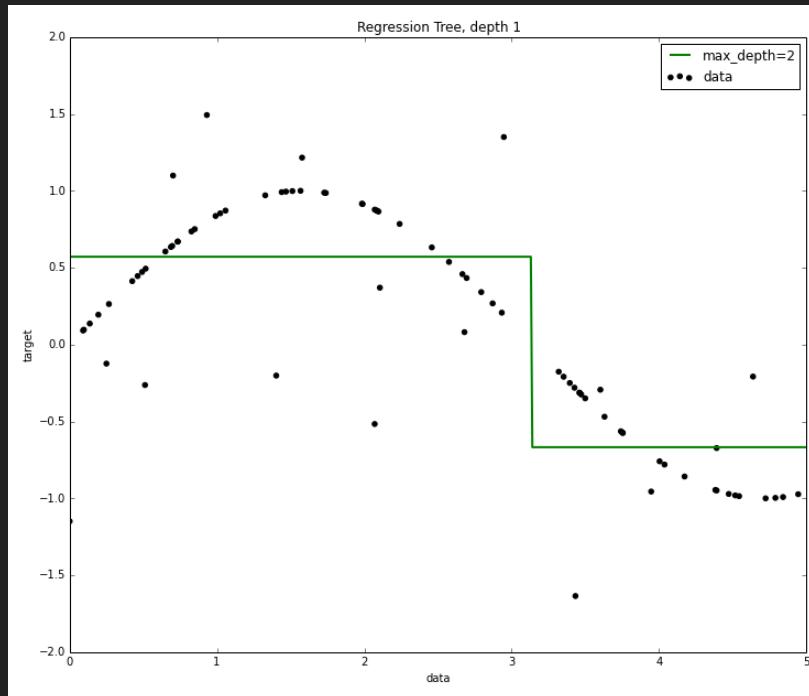
Decision trees of depth 1 will be used.





X MINUTES BREAK

DECISION TREES FOR REGRESSION



GRADIENT BOOSTING [FRIEDMAN, 1999]

composition of weak learners,

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

$$p_1(x) = \sigma(d(x)) = 1 - p_{-1}(x)$$

Optimization of log-likelihood:

$$\mathcal{L} = \sum_i L(x_i, y_i) = \sum_i \ln \left(1 + e^{-y_i d(x_i)} \right) \rightarrow \min$$

GRADIENT BOOSTING

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

$$\mathcal{L} = \sum_i \ln \left(1 + e^{-y_i D(x_i)} \right) \rightarrow \min$$

- Optimization problem: find all α_j , weak learners d_j
- Mission **impossible**
- Main point: greedy optimization of loss function by training one more weak learner d_j
- Each new estimator follows the gradient of loss function

GRADIENT BOOSTING

Gradient boosting ~ steepest gradient descent.

$$D_j(x) = \sum_{j'=1}^j \alpha_{j'} d_{j'}(x)$$

$$D_j(x) = D_{j-1}(x) + \alpha_j d_j(x)$$

At j th iteration:

- pseudo-residual $z_i = -\frac{\partial}{\partial D(x_i)} \mathcal{L} \Big|_{D(x)=D_{j-1}(x)}$
- train regressor d_j to minimize MSE:
 $\sum_i (d_j(x_i) - z_i)^2 \rightarrow \min$
- find optimal α_j

ADDITIONAL GB TRICKS

to make training more stable, add learning rate

$$D_j(x) = \sum_j \eta \alpha_j d_j(x)$$

randomization to fight noise and build different trees:
subsampling of features and training samples

AdaBoost is particular case of gradient boosting with different target loss function*:

$$\mathcal{L} = \sum_i e^{-y_i D(x_i)} \rightarrow \min$$

This loss function is called ExpLoss or AdaLoss.

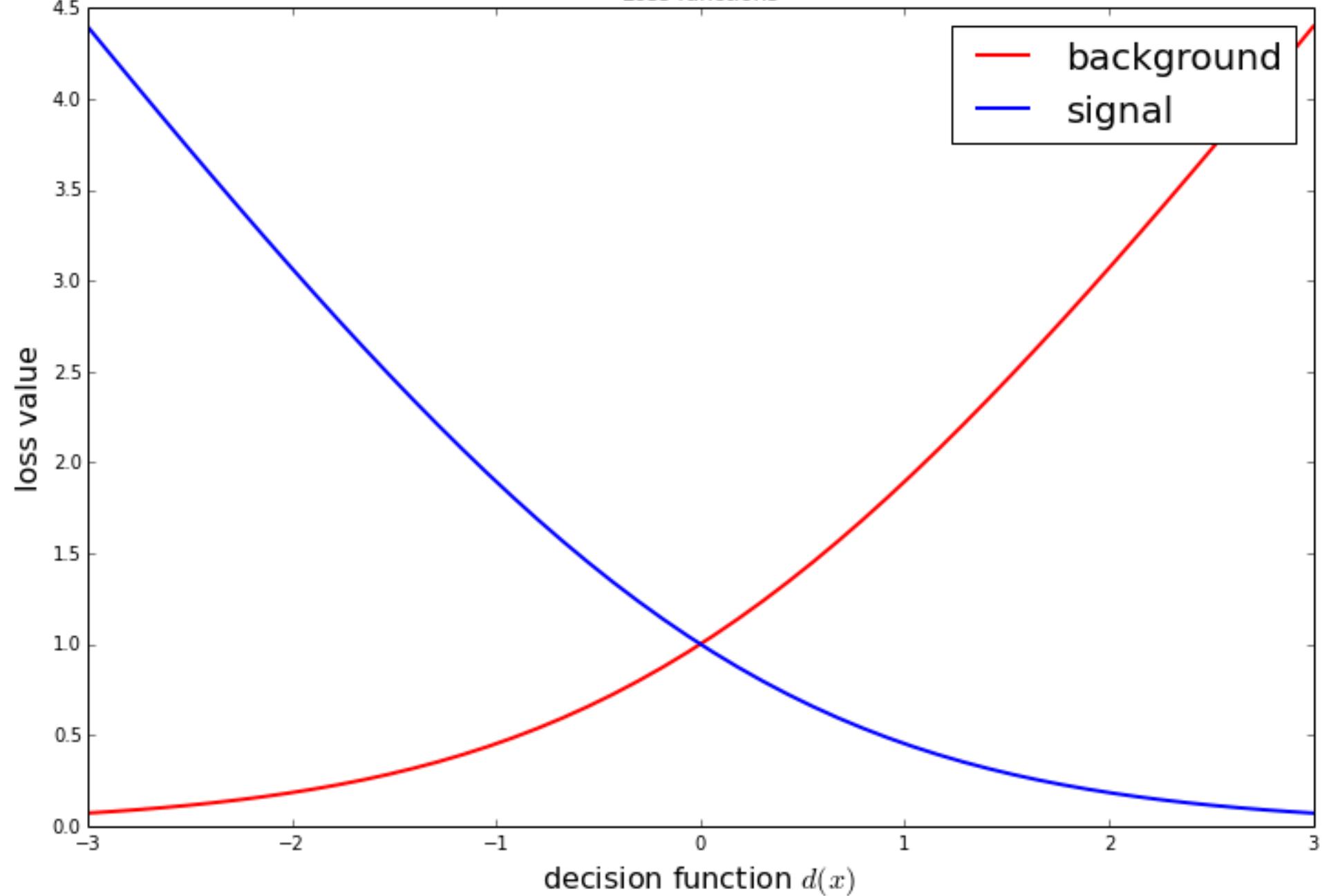
*(also AdaBoost expects that $d_j(x_i) = \pm 1$)

LOSS FUNCTIONS

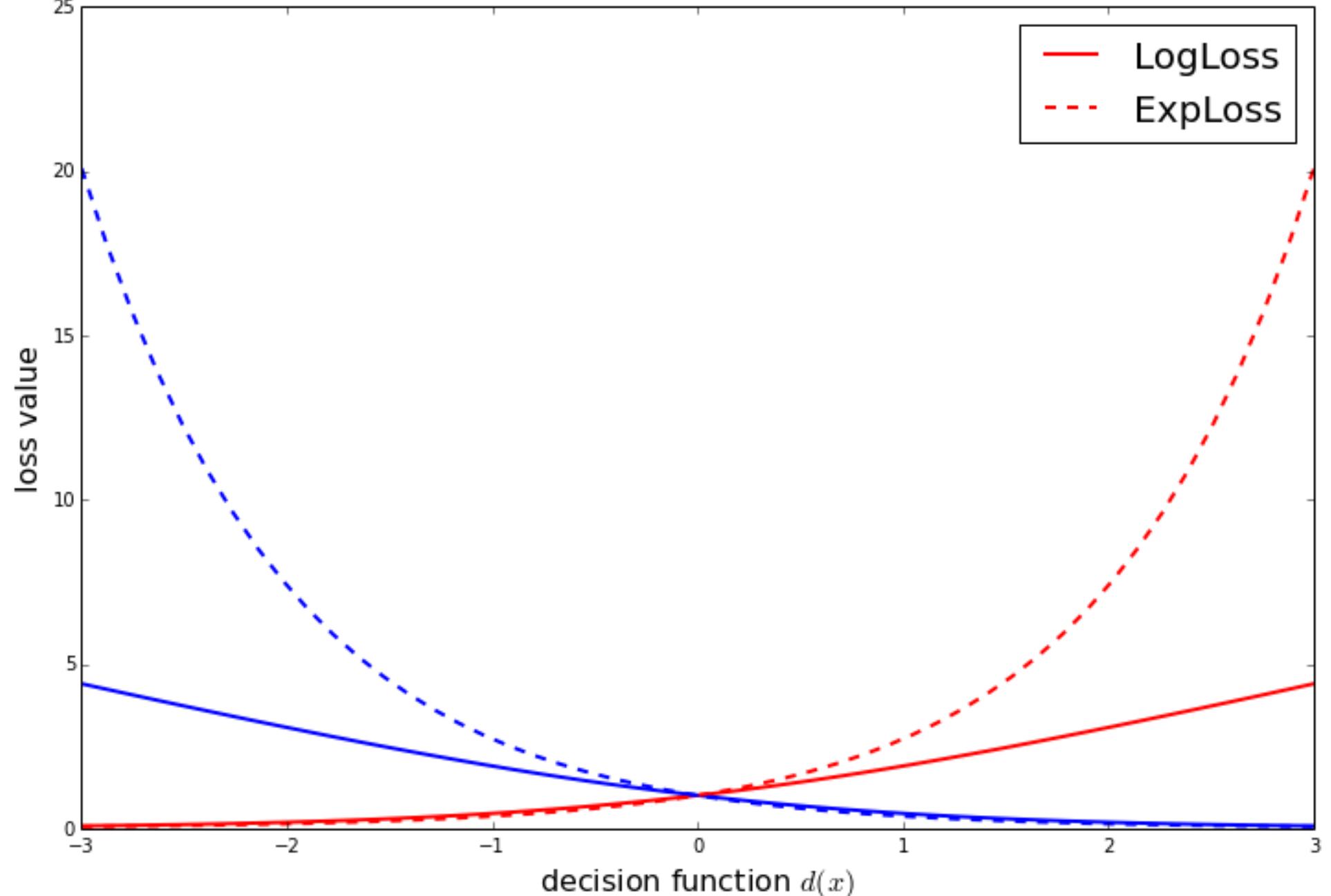
Gradient boosting can optimize different smooth loss function.

- **regression, $y \in \mathbb{R}$**
 - Mean Squared Error $\sum_i (d(x_i) - y_i)^2$
 - Mean Absolute Error $\sum_i |d(x_i) - y_i|$
- **binary classification, $y_i = \pm 1$**
 - ExpLoss (ada AdaLoss) $\sum_i e^{-y_i d(x_i)}$
 - LogLoss $\sum_i \log(1 + e^{-y_i d(x_i)})$

Loss functions



Loss functions



LOSS FUNCTION: RANKING EXAMPLE

In ranking we need to order items by y_i :

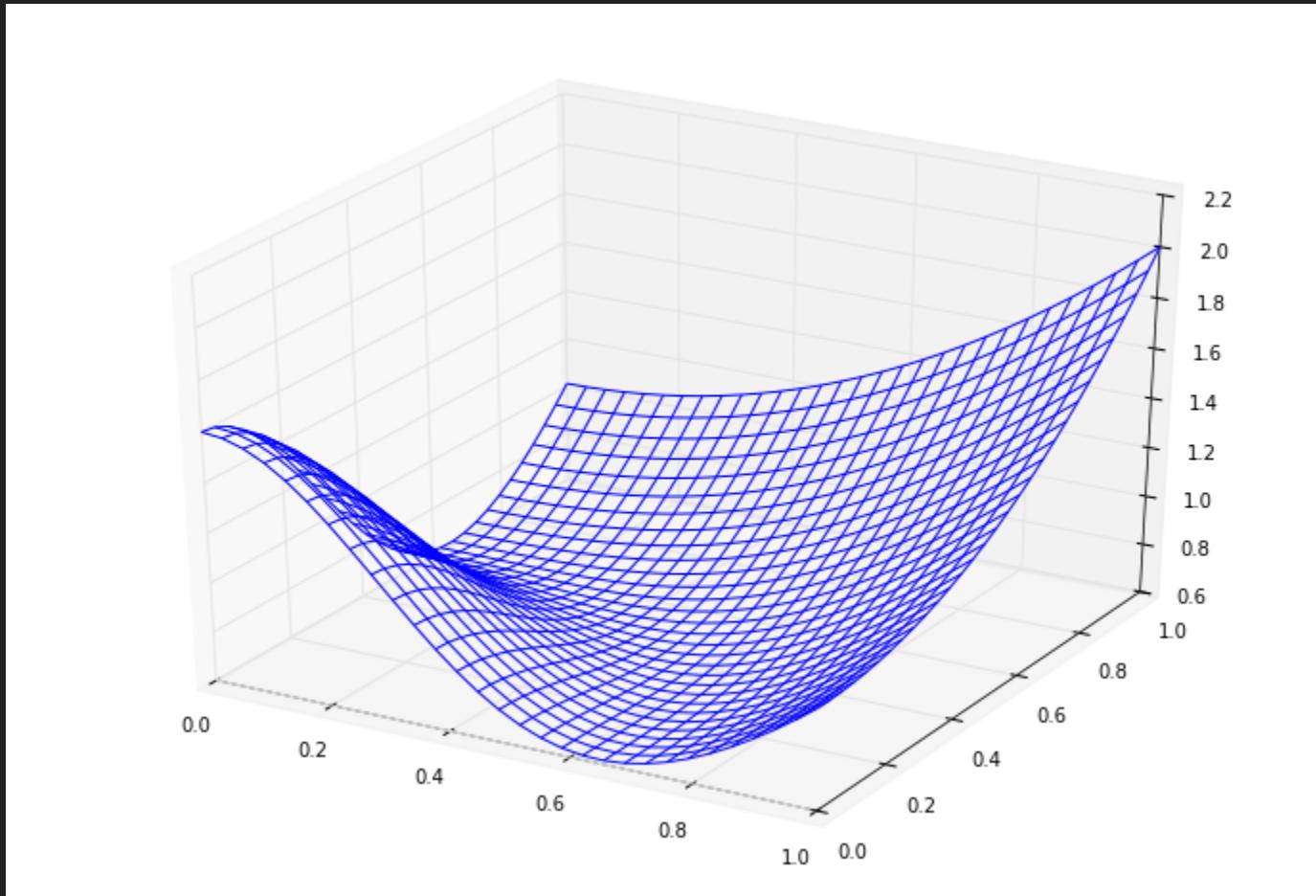
$$y_i < y_j \Rightarrow d(x_i) < d(x_j)$$

We can add penalization term for misordering:

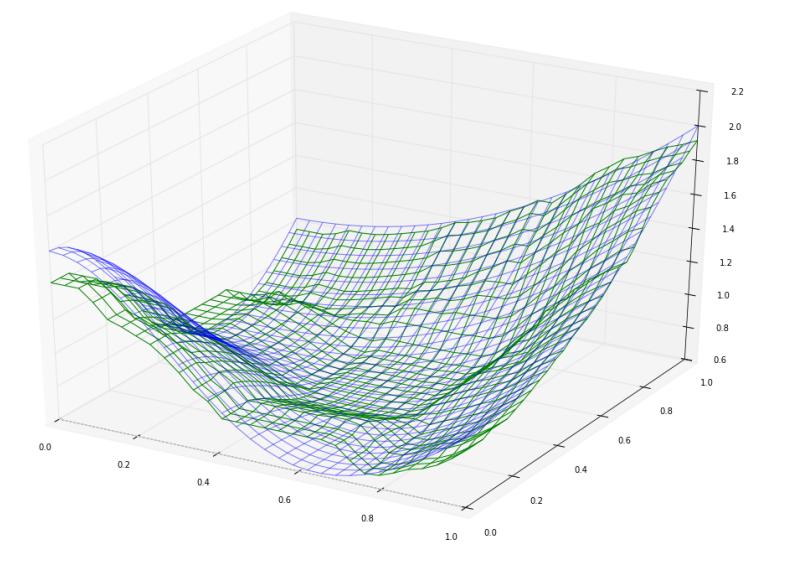
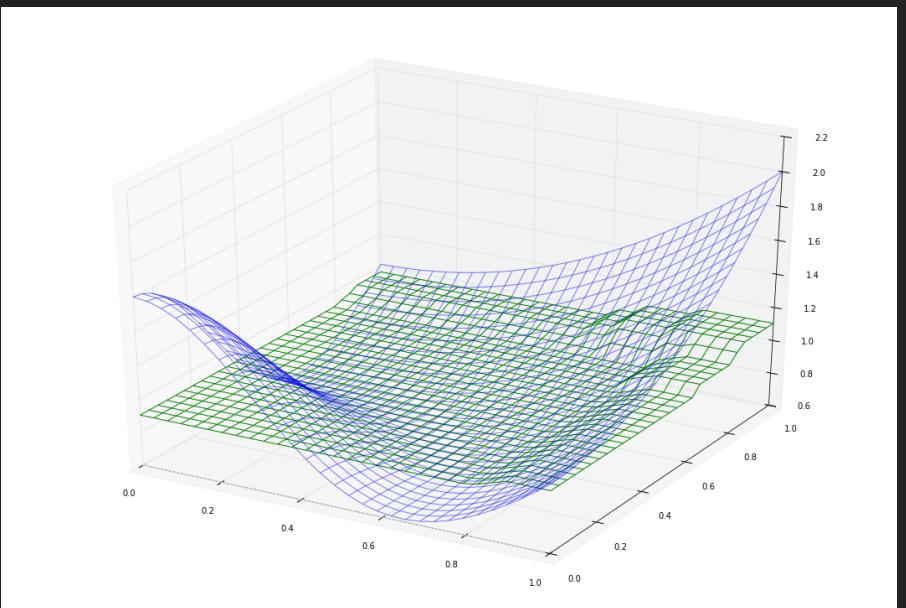
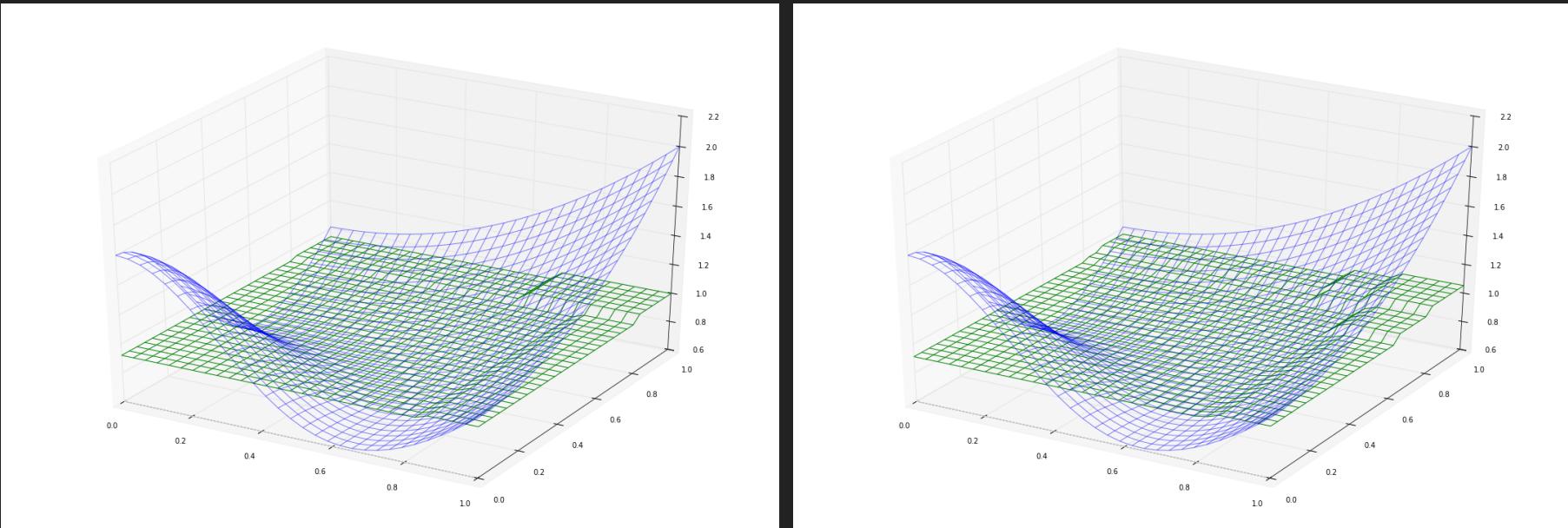
$$\mathcal{L} = \sum_{ij} L(x_i, x_j, y_i, y_j)$$

$$L(x_i, x_j, y_i, y_j) = \begin{cases} \sigma(d(x_j) - d(x_i)), & y_i < y_j \\ 0, & \text{otherwise} \end{cases}$$

EXAMPLE: REGRESSION WITH GB



using regression trees of depth=2



number of trees = 1, 2, 3, 100

ADAPTING BOOSTING

By modifying boosting or changing loss function we can solve different problems

- classification
- regression
- ranking

Also we can add restrictions, i.e. fight correlation with mass

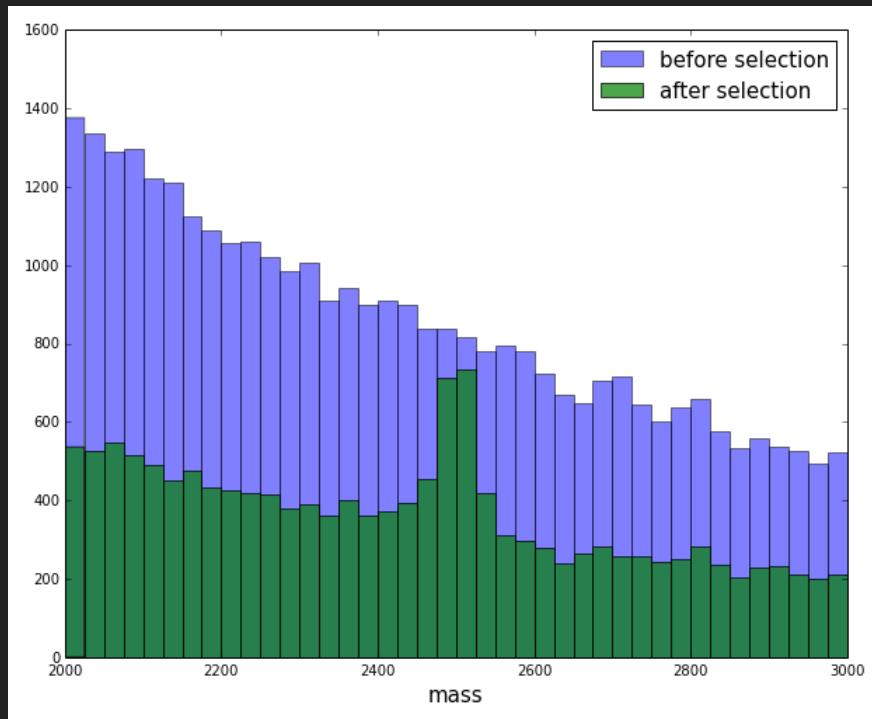
BOOSTING TO UNIFORMITY

Point of uniform boosting - have constant efficiency against some variable.

Examples:

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

EXAMPLE: NON-FLAT BACKGROUND EFFICIENCY ALONG MASS



High correlation with mass will create from pure background **false peaking signal**

Aim: $\text{FPR} = \text{const}$ for different regions in mass.

uBOOST

Variation of AdaBoost approach,

aim $\text{TPR}_{\text{region}} = \text{const.}$

fix target efficiency (say $\text{TPR}_{\text{target}} = 70\%$), find corresponding threshold

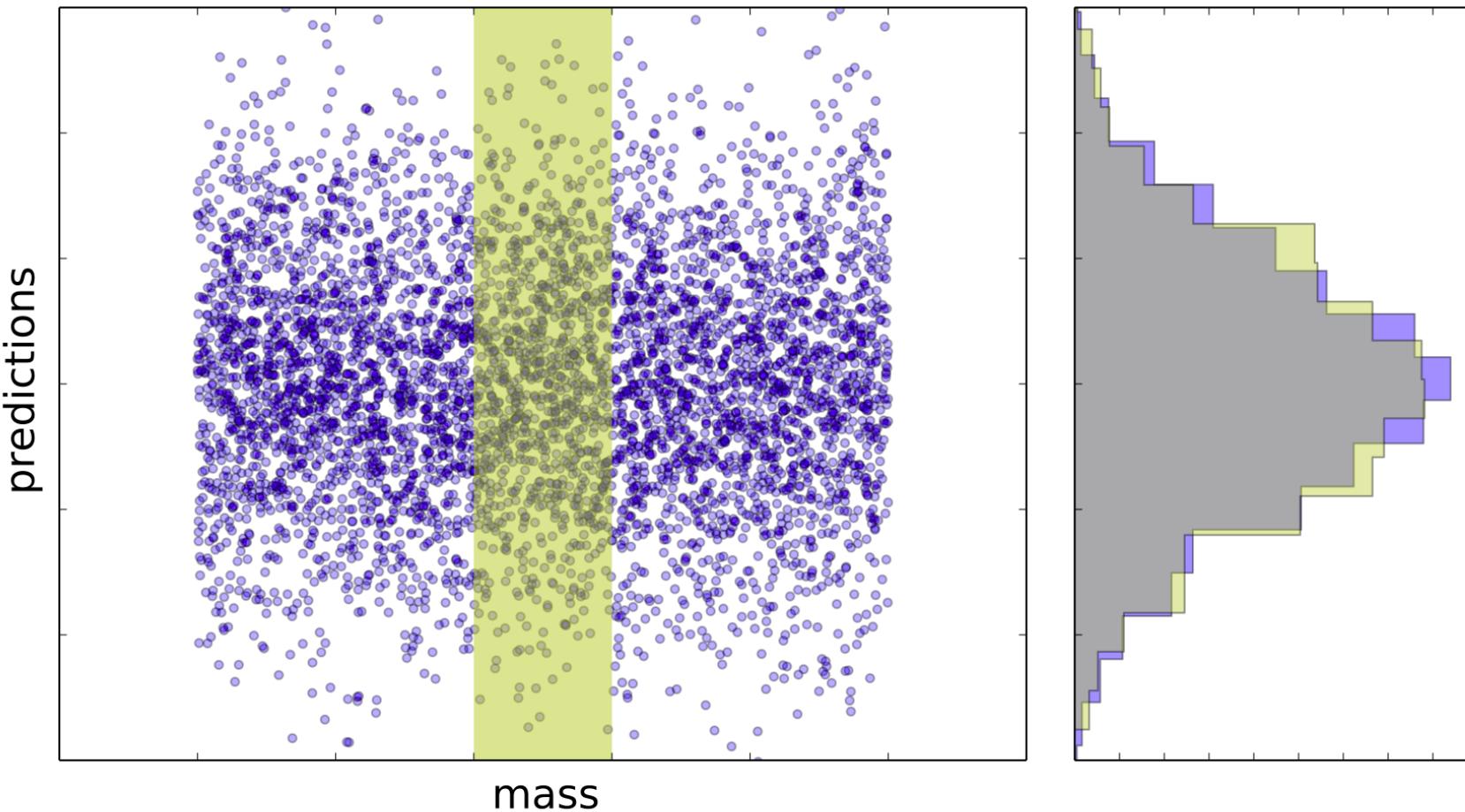
- Train a tree, add it's predictions $d_j(x)$
- increase weight for misclassification:

$$w_i \leftarrow w_i \exp(\alpha y_i d(x))$$

- increase weight of signal events in regions with low TPR

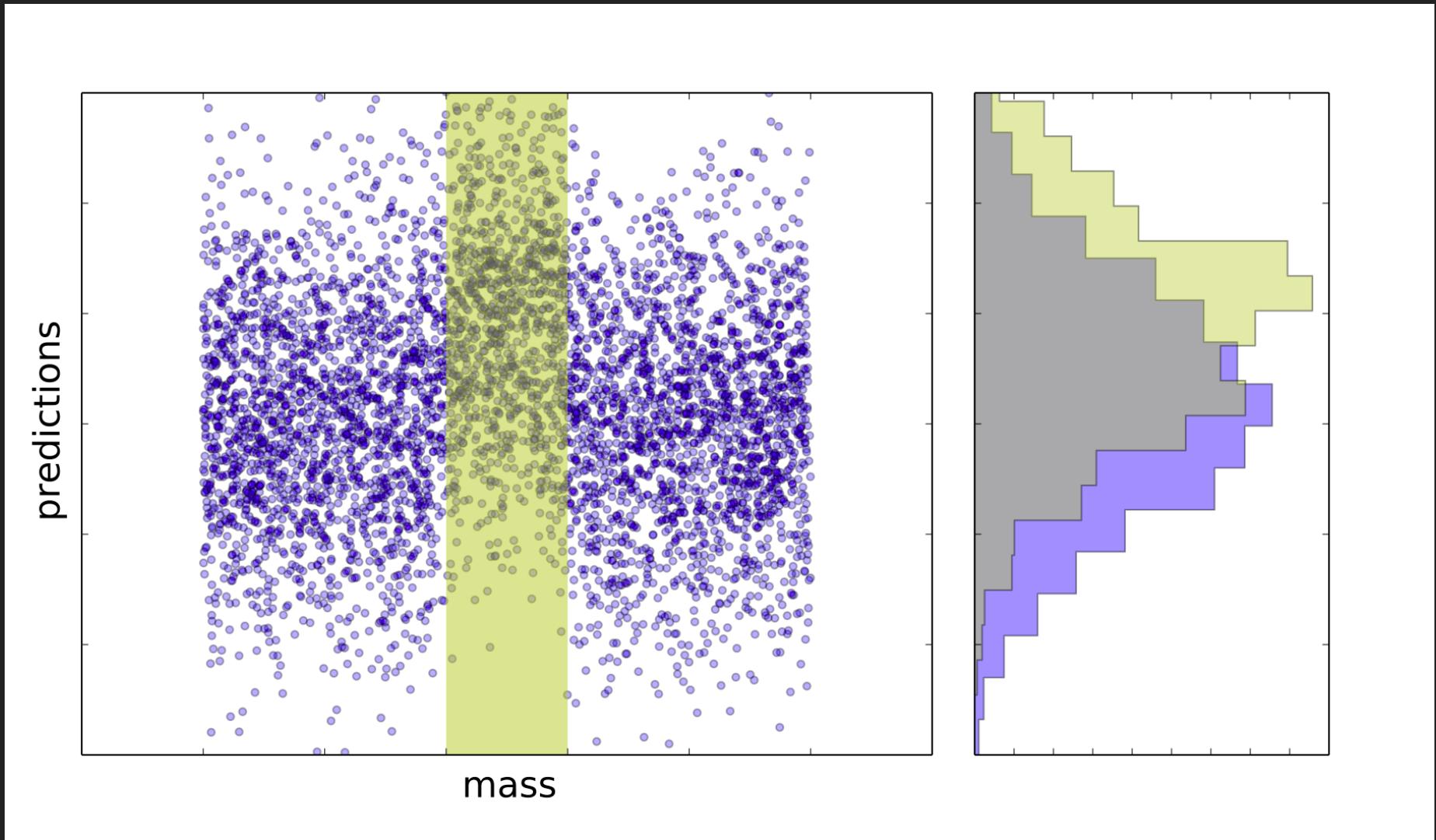
$$w_i \leftarrow w_i \exp(\beta(\text{TPR}_{\text{target}} - \text{TPR}_{\text{region}}))$$

MEASURING NON-UNIFORMITY



MEASURING NON-UNIFORMITY

$$CvM = \sum_{region} \int |F_{\text{region}}(x) - F_{\text{global}}(x)|^2 dF_{\text{global}}(x)$$



FLATNESS LOSS

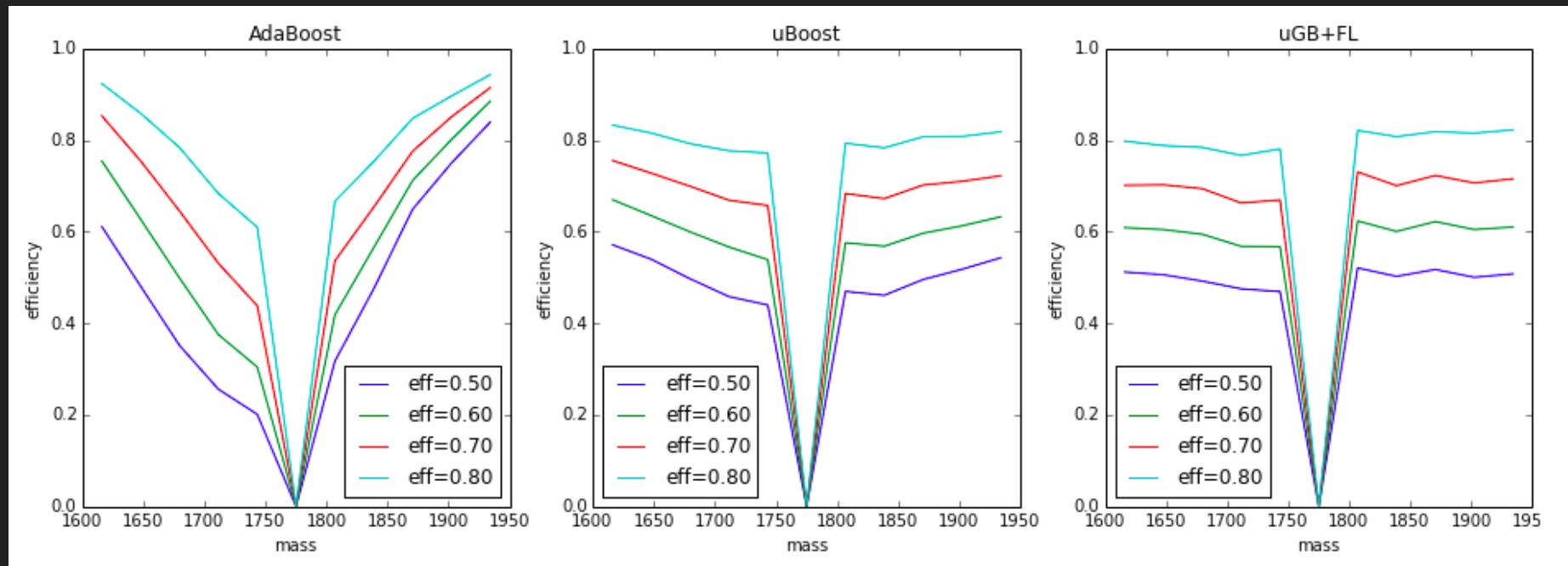
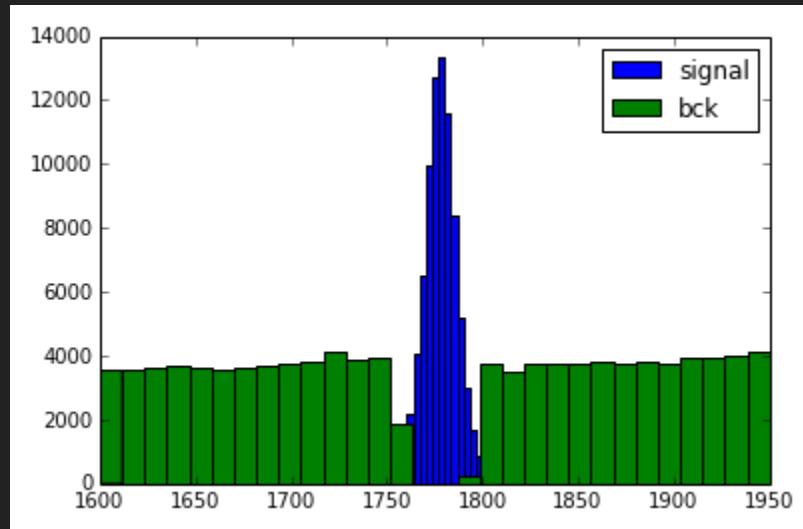
Put an additional term in loss function which will penalize for non-uniformity

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

Flatness loss approximates (non-differentiable) CvM metrics:

$$\mathcal{L}_{\text{FL}} = \sum_{region} \int |F_{\text{region}}(x) - F_{\text{global}}(x)|^2 dx$$

EXAMPLE (EFFICIENCY OVER BACKGROUND)



GRADIENT BOOSTING

- general-purpose flexible algorithm
- usually over trees
- State-of-art results in many areas
- Can overfit
- Usually needs tuning

THE END