

# MACHINE LEARNING IN HIGH ENERGY PHYSICS

## LECTURE #4



Alex Rogozhnikov, 2015

# WEIGHTED VOTING

The way to introduce importance of classifiers

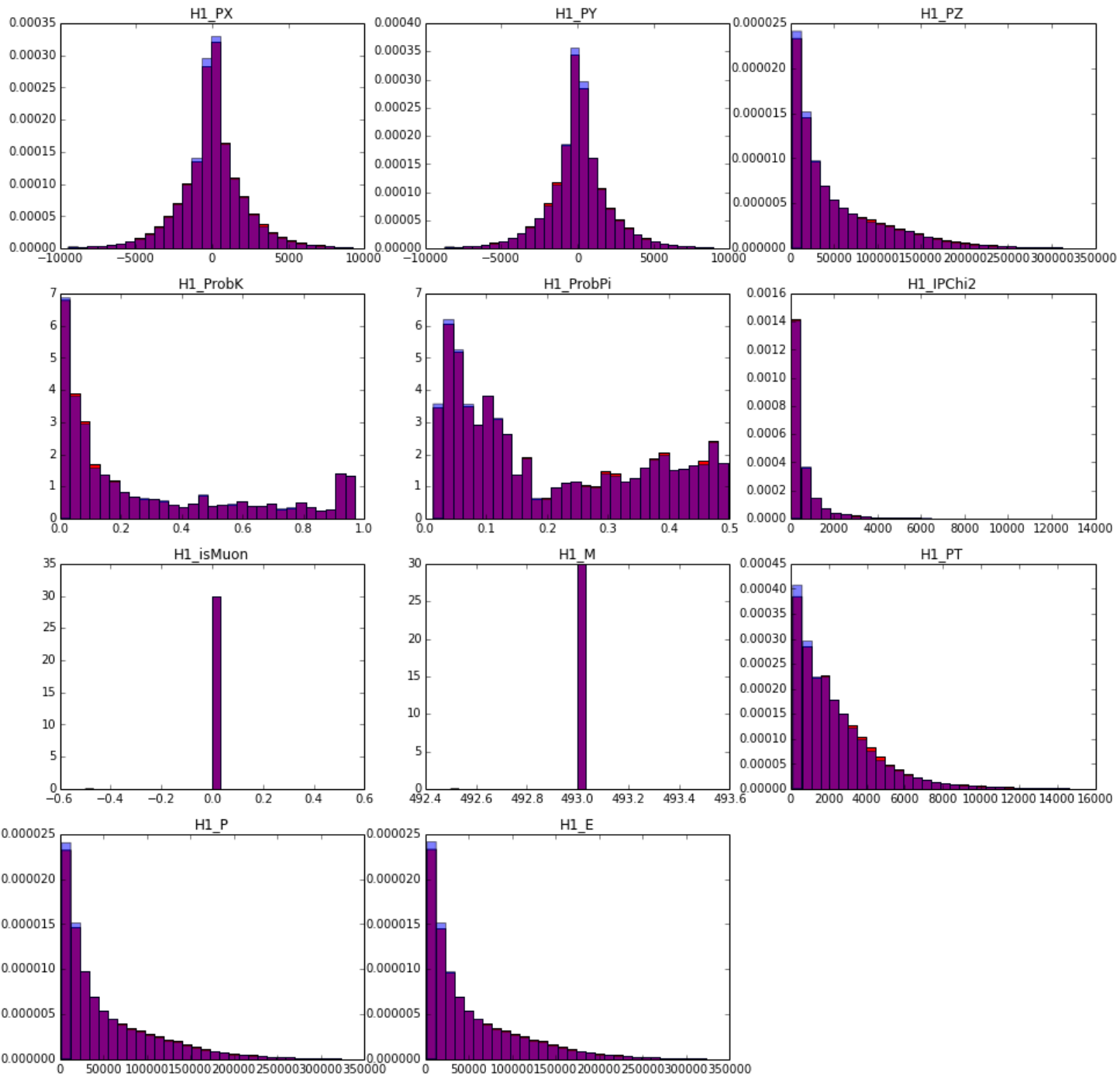
$$D(x) = \sum_j \alpha_j d_j(x)$$

## GENERAL CASE OF ENSEMBLING:

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

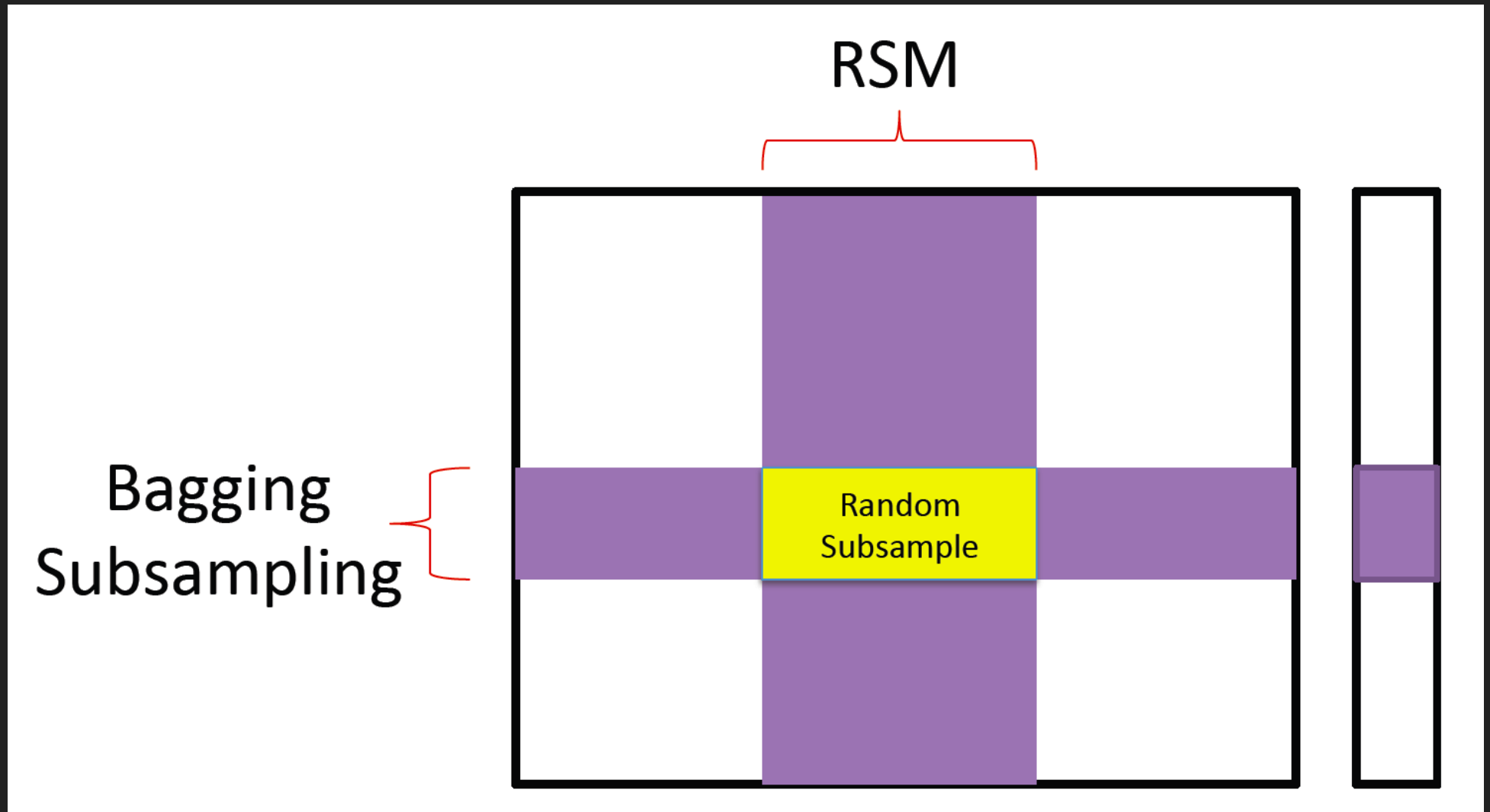
# COMPARISON OF DISTRIBUTIONS

- good options to compare 1d
- use classifier's output to build discriminating variable
- ROC AUC + Mann Whitney to get significance



# RANDOM FOREST

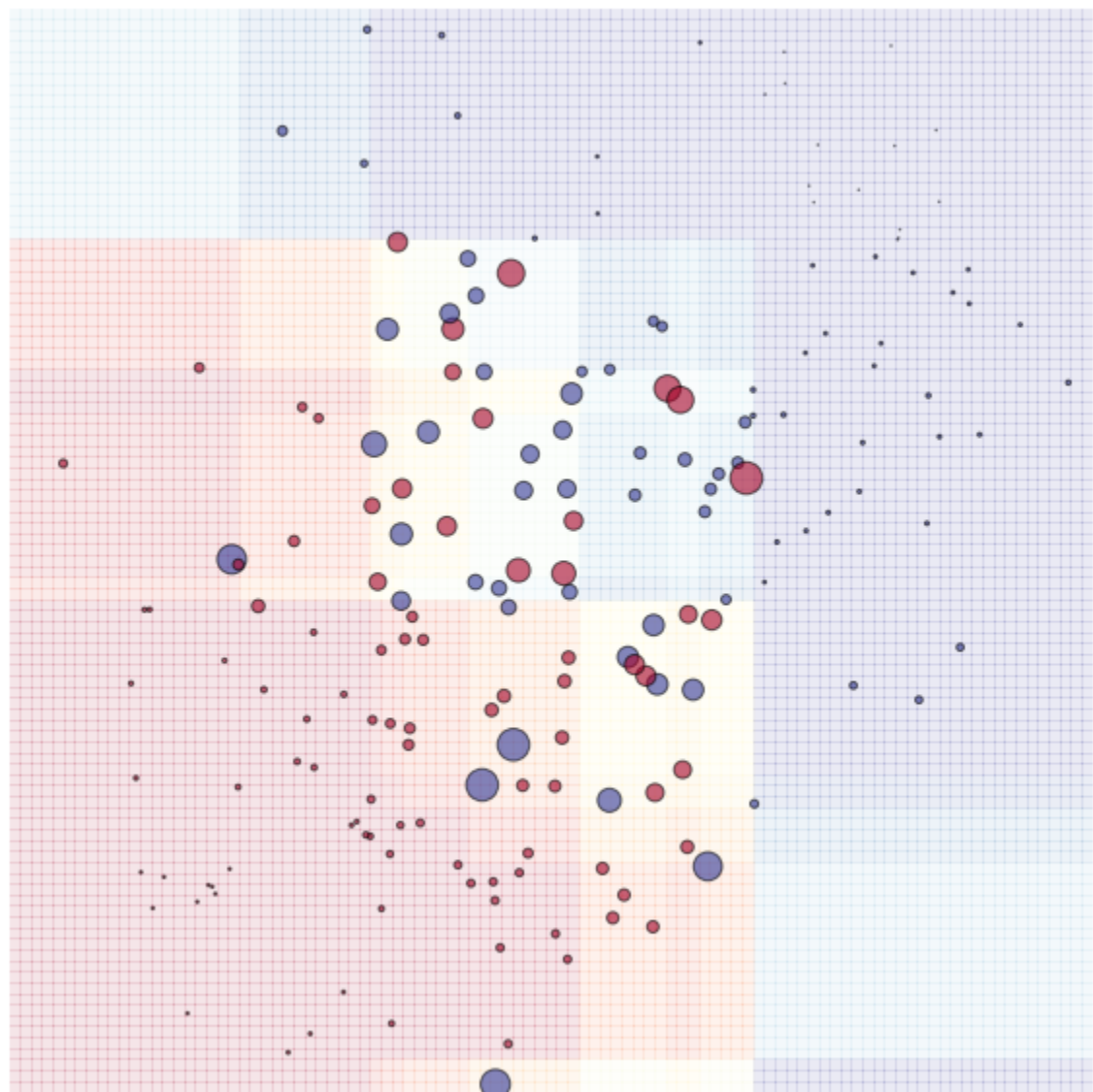
composition of independent trees



# ADABOOST

After building  $j$ th base classifier:

1.  $\alpha_j = \frac{1}{2} \ln \left( \frac{w_{\text{correct}}}{w_{\text{wrong}}} \right)$
2. increase weight of misclassified  
 $w_i \leftarrow w_i \times e^{-\alpha_j y_i d_j(x_i)}$



# GRADIENT BOOSTING

$$\mathcal{L} \rightarrow \min$$

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

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

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  $\alpha_j$



# LOSSES

- 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 (aka AdaLoss)  $\sum_i e^{-y_i d(x_i)}$
  - LogLoss  $\sum_i \log(1 + e^{-y_i d(x_i)})$
- ranking losses
- boosting to uniformity: FlatnessLoss

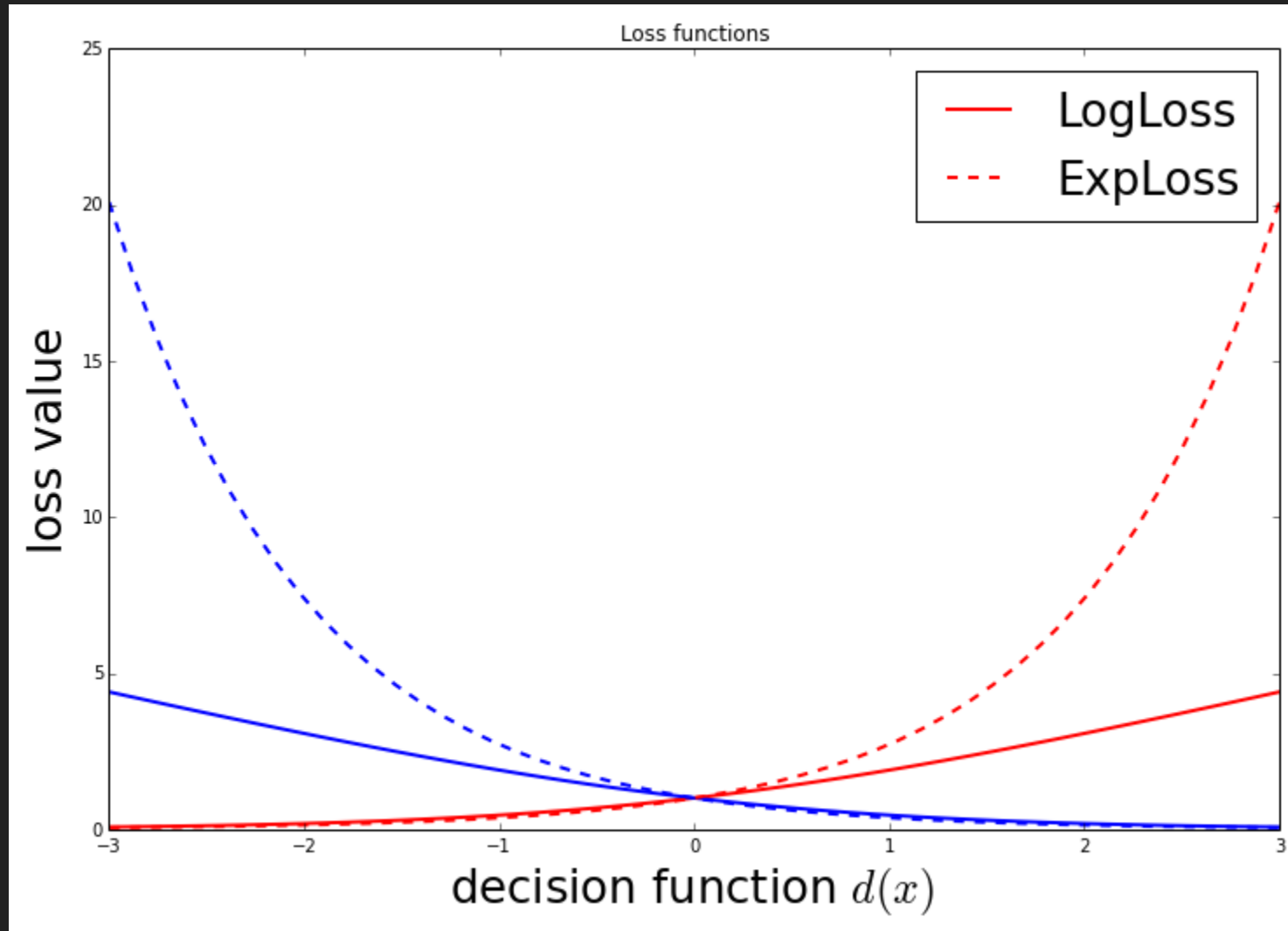
# TUNING GRADIENT BOOSTING OVER DECISION TREES

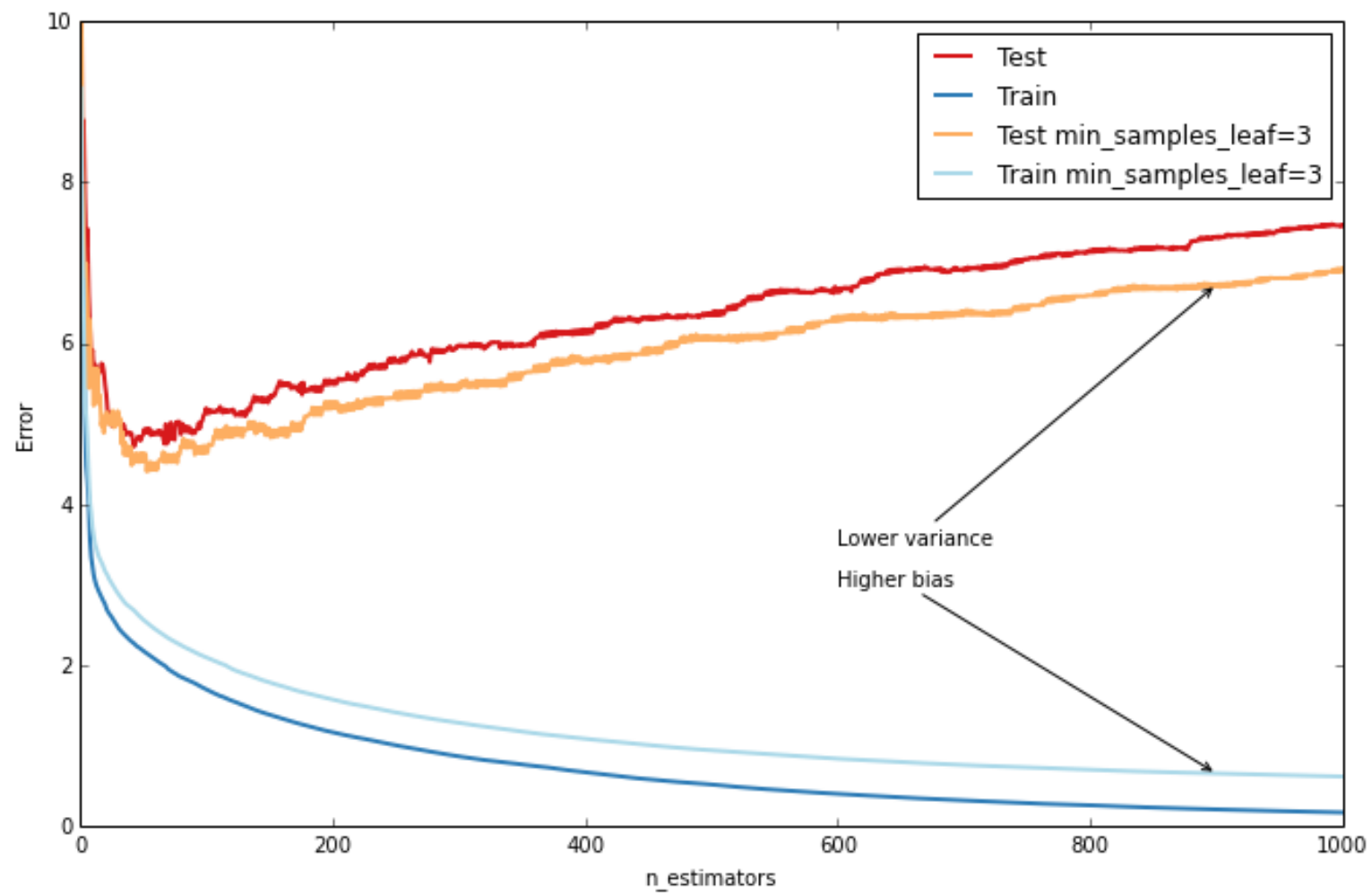
Parameters:

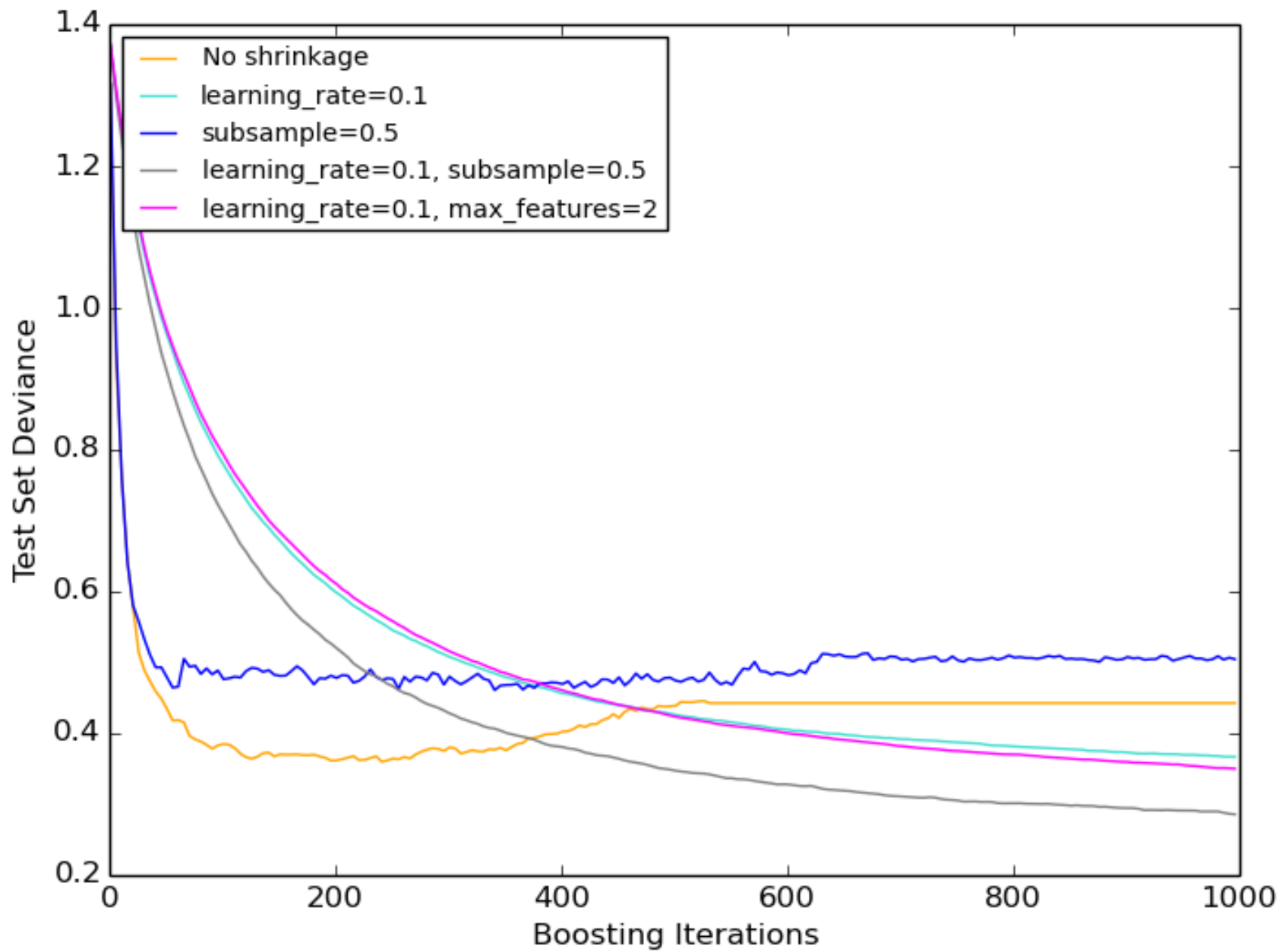
- loss function
- pre-pruning: maximal depth, minimal leaf size
- subsample,  $\text{max\_features} = N/3$
- $\eta$  (`learning_rate`)
- number of trees

# LOSS FUNCTION

different combinations also may be used







# TUNING GBDT

1. set high, but feasible number of trees
2. find optimal parameters by checking combinations
3. decrease learning rate, increase number of trees

See also: [GBDT tutorial](#)

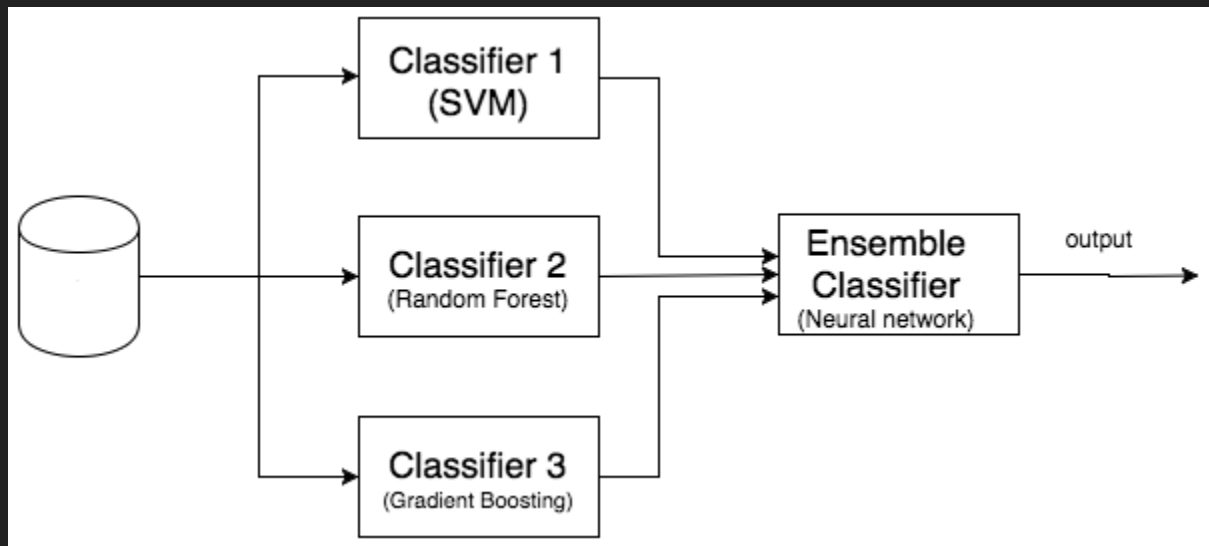
# ENSEMBLES: BAGGING OVER BOOSTING

Different variations [1], [2] are claimed to overcome single GBDT.

Very complex training, better quality if GB estimators are overfitted

```
BaggingClassifier(base_estimator=GradientBoostingClassifier(),  
                  n_estimators=100)
```

# ENSEMBLES: STACKING



Correcting output of several classifiers with new classifier.

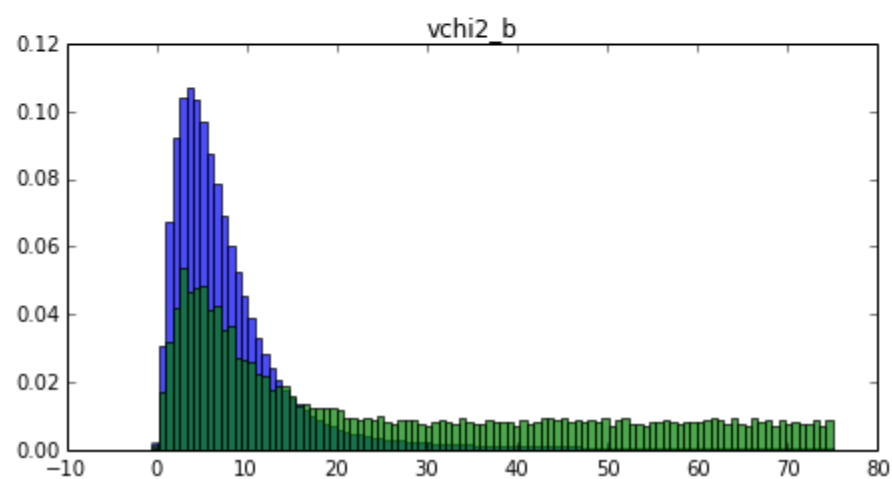
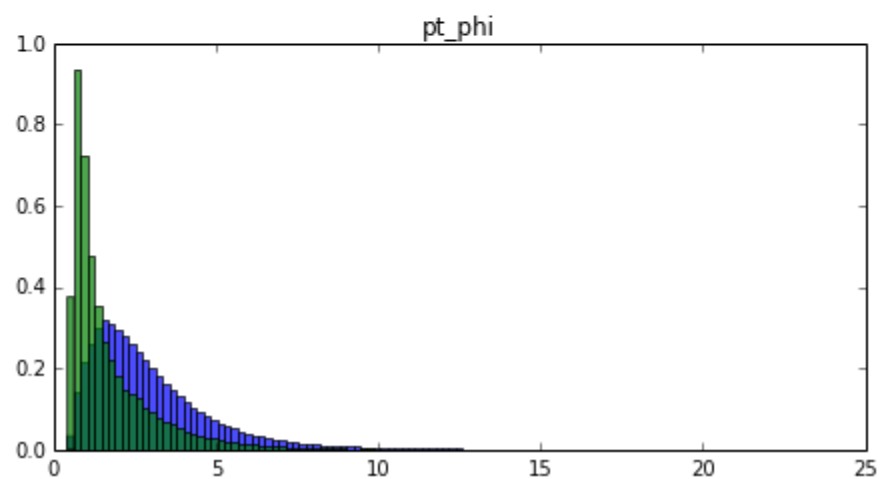
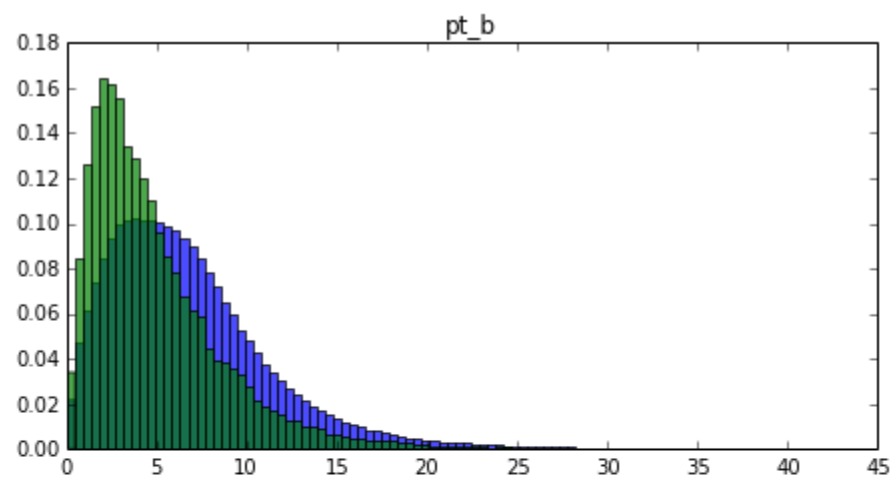
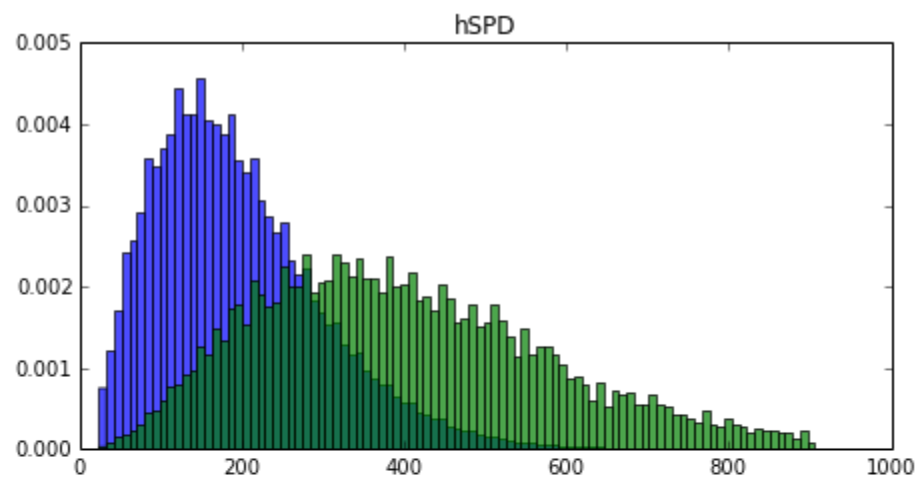
$$D(x) = f(d_1(x), d_2(x), \dots, d_J(x))$$

To use unbiased predictions, use holdout or kFolding.



# REWEIGHTING

Given two distributions: target and original, find new weights for original distributions, that distributions will coincide.



# REWEIGHTING

Solution for 1 or 2 variables: reweight using bins (so-called 'histogram division').

Typical application: Monte-Carlo reweighting.

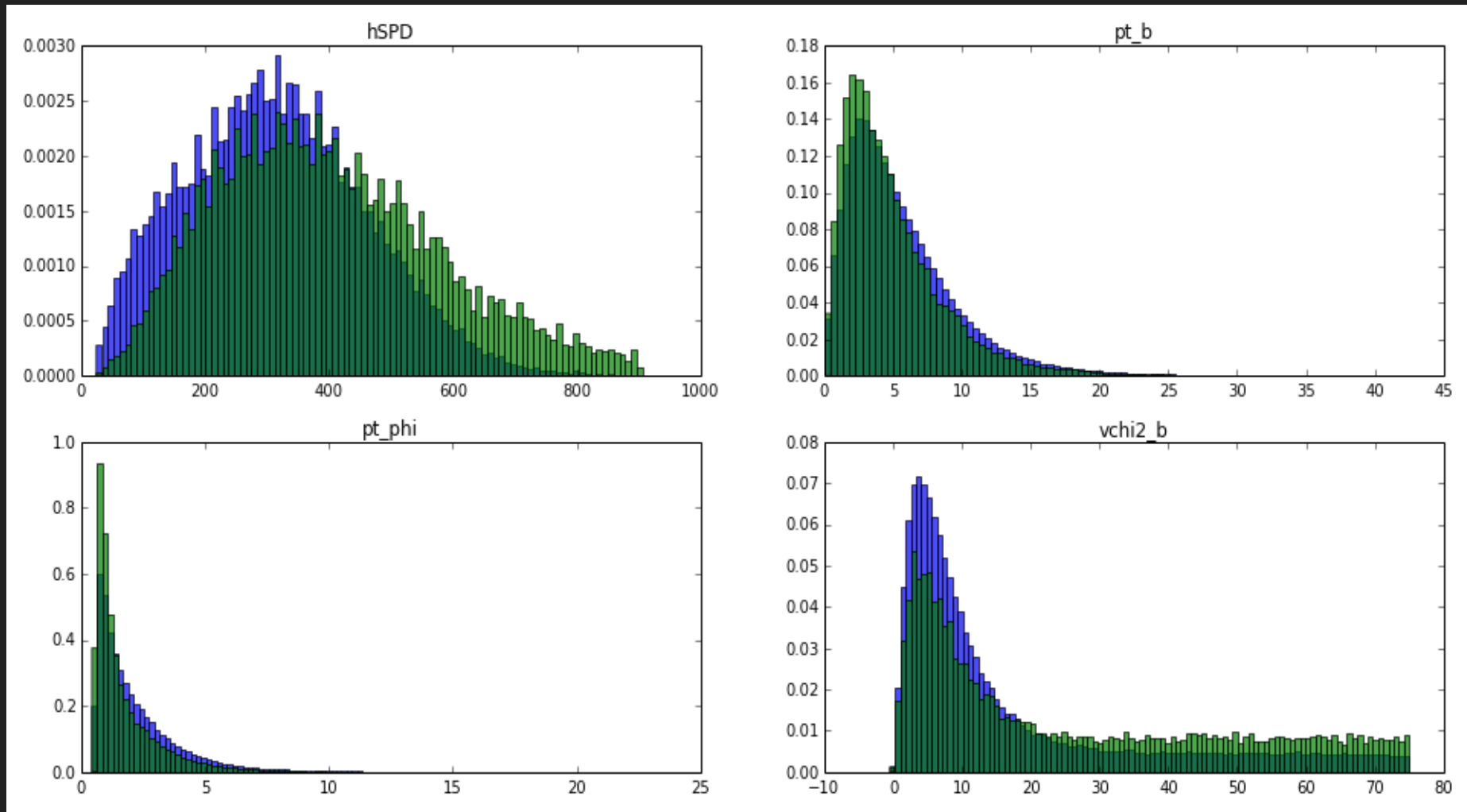
# REWEIGHTING WITH BINS

Splitting all space in bins, for each bin:

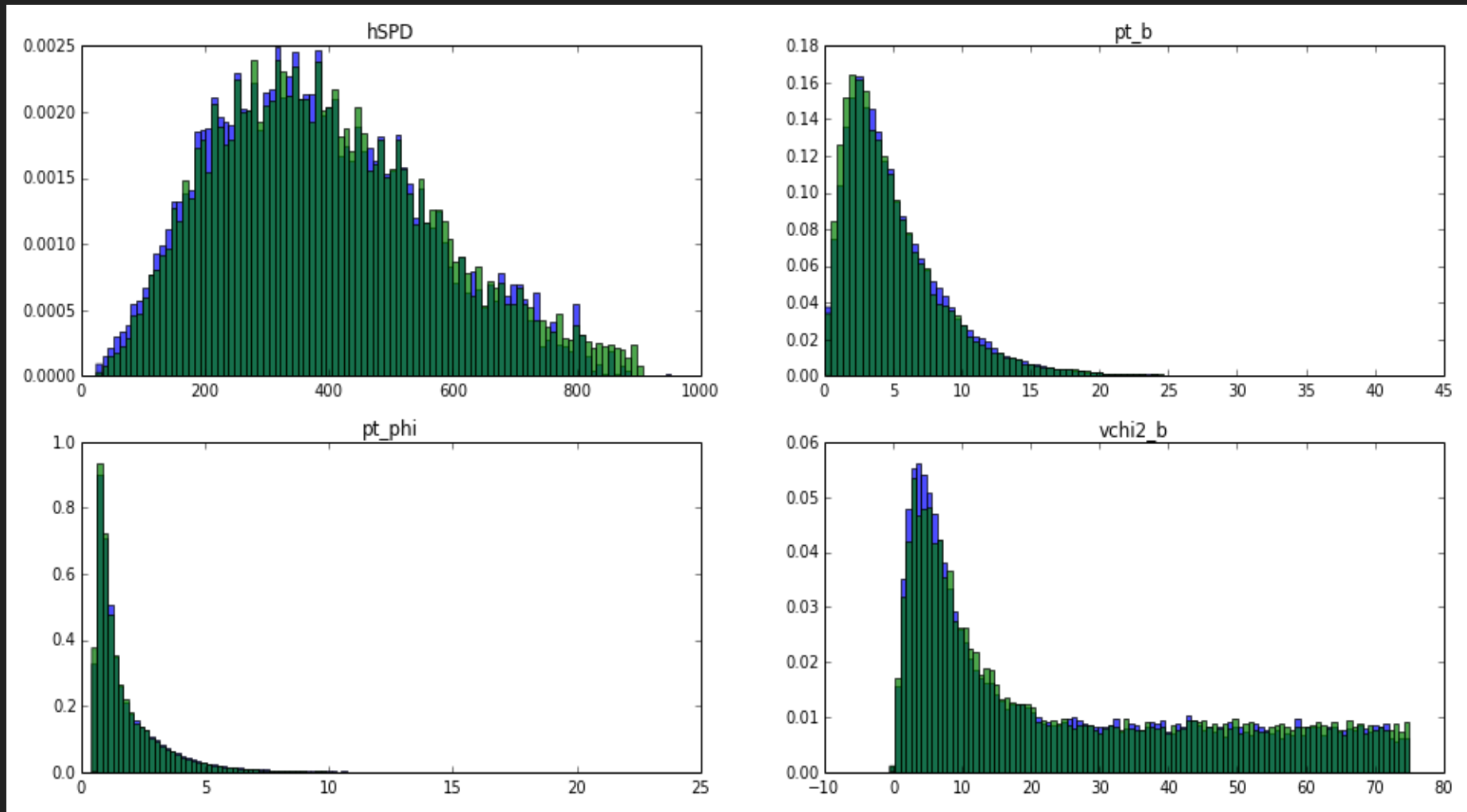
- compute  $W_{original}, W_{target}$  in each bin
- multiply weights of original distribution in bin to compensate the difference  $w_i \leftarrow w_i \frac{W_{target}}{W_{original}}$

## Problems

- good in 1d, works sometimes in 2d, nearly impossible in dimensions  $> 2$
- too few events in bin  $\Rightarrow$  reweighting rule is unstable
- we can reweight several times over 1d if variables aren't correlated



reweighting using enhanced version of bin reweighter



reweighting using gradient boosted reweighter

# GRADIENT BOOSTED REWEIGHTER

- works well in high dimensions
- works with correlated variables
- produces more stable reweighting rule

# HOW DOES GB REWEIGHTER WORKS?

iteratively:

1. Find the tree, which is able to discriminate two distributions
2. Correct weights in each leaf
3. Reweight original distribution and repeat the process.

Less bins, bins are guaranteed to have high statistic in each.



# DISCRIMINATING TREE

When looking for a tree with good discrimination, optimize

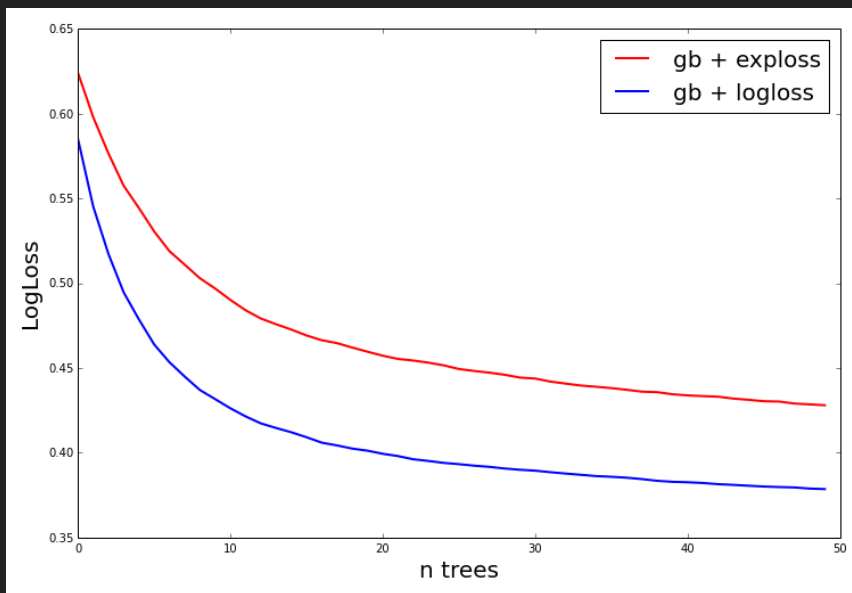
$$\chi^2 = \sum_{l \in \text{leaves}} \frac{(w_{l,\text{target}} - w_{l,\text{original}})^2}{w_{l,\text{target}} + w_{l,\text{original}}}$$

As before, using greedy minimization to build a tree, which provides maximal  $\chi^2$ .

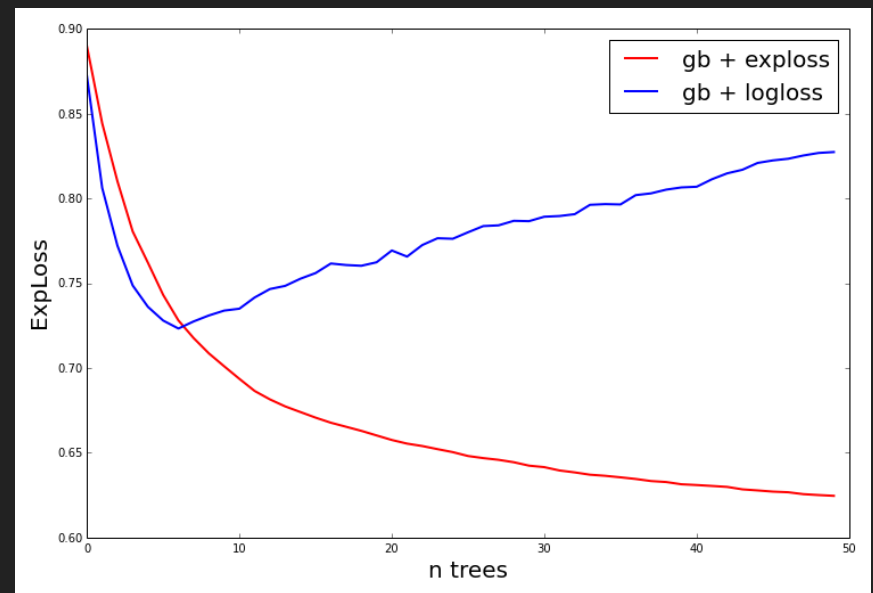
+ introducing all heuristics from standard GB.

# QUALITY ESTIMATION

Quality of model can be estimated by the value of optimized loss function. Though remember that different classifiers use different optimization target

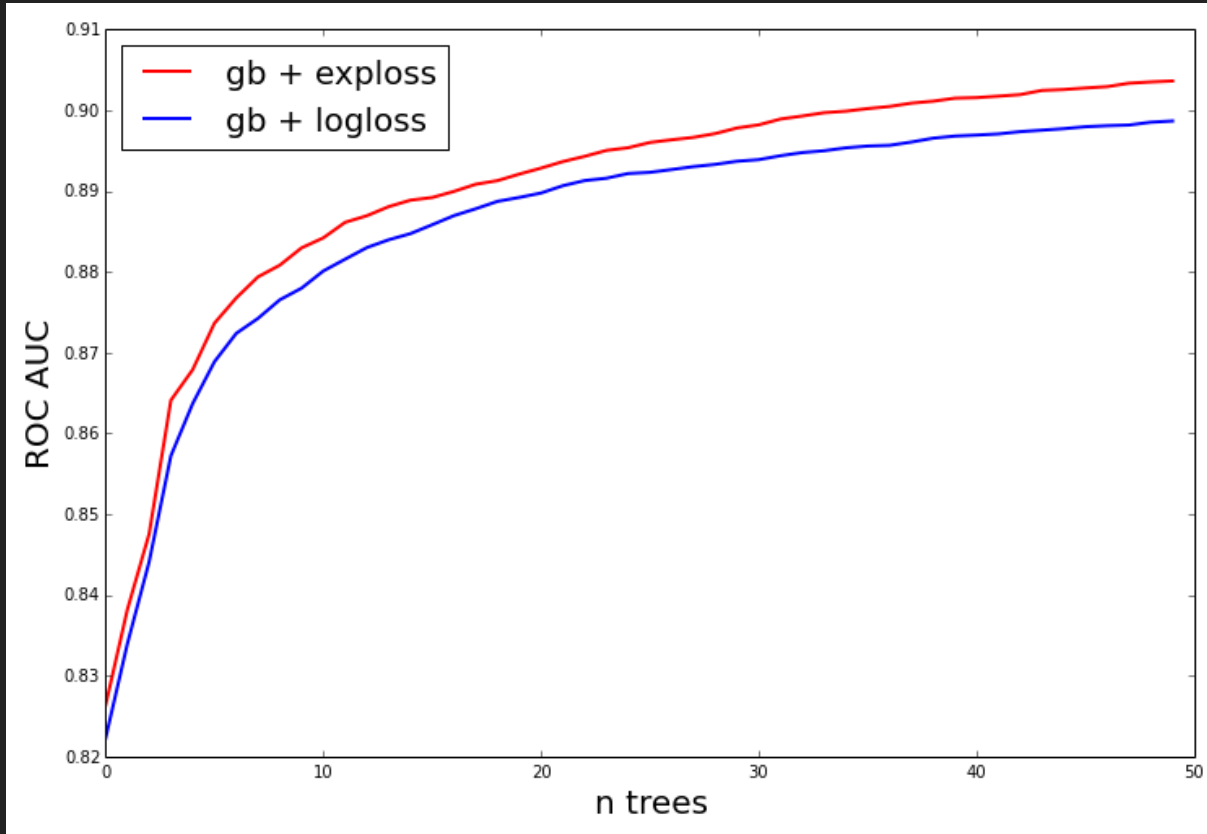


LogLoss



ExpLoss

# QUALITY ESTIMATION



different target of optimization  $\Rightarrow$  use general quality metrics

# TESTING HYPOTHESIS WITH CUT

Example: we test hypothesis that there is signal channel with fixed  $Br \neq 0$  vs. no signal channel ( $Br = 0$ ).

$H_0$ : signal channel with  $Br = \text{const}$

$H_1$ : there is no signal channel

Putting a threshold:  $d(x) > \text{threshold}$

Estimate number of signal / bkg events that will be selected:

$s = \alpha$  tpr,  $b = \beta$  fpr

$H_0: n_{\text{obs}} \sim \text{Poiss}(s + b)$

$H_1: n_{\text{obs}} \sim \text{Poiss}(b)$

# TESTING HYPOTHESIS WITH CUT

Select some appropriate metric:

$$\text{AMS}^2 = 2(s + b) \log\left(1 + \frac{s}{b}\right) - 2s$$

Maximize it by selecting best threshold

- special holdout shall be used
- or use kFolding
- very poor usage of information from classifier

# TESTING HYPOTHESIS WITH BINS

Splitting all data in  $n$  bins:

$$x \in \text{k-th bin} \Leftrightarrow \text{thr}_{k-1} < d(x) \leq \text{thr}_k.$$

Expected amounts of signal:

$$s_k = \alpha(\text{tpr}_{k-1} - \text{tpr}_k), \quad b_k = \beta(\text{fpr}_{k-1} - \text{fpr}_k)$$

$$H_0: n_k \sim \text{Poiss}(s_k + b_k)$$

$$H_1: n_k \sim \text{Poiss}(b_k)$$

Optimal statistics:

$$\sum_k c_k n_k, \quad c_k = \log\left(1 + \frac{s_k}{b_k}\right)$$

take some approximation to test power and optimize thresholds

# FINDING OPTIMAL PARAMETERS

- some algorithms have many parameters
- not all the parameters are guessed
- checking all combinations takes too long

# FINDING OPTIMAL PARAMETERS

- randomly picking parameters is a partial solution
- given a target optimal value we can optimize it
- no gradient with respect to parameters
- noisy results
- function reconstruction is a problem

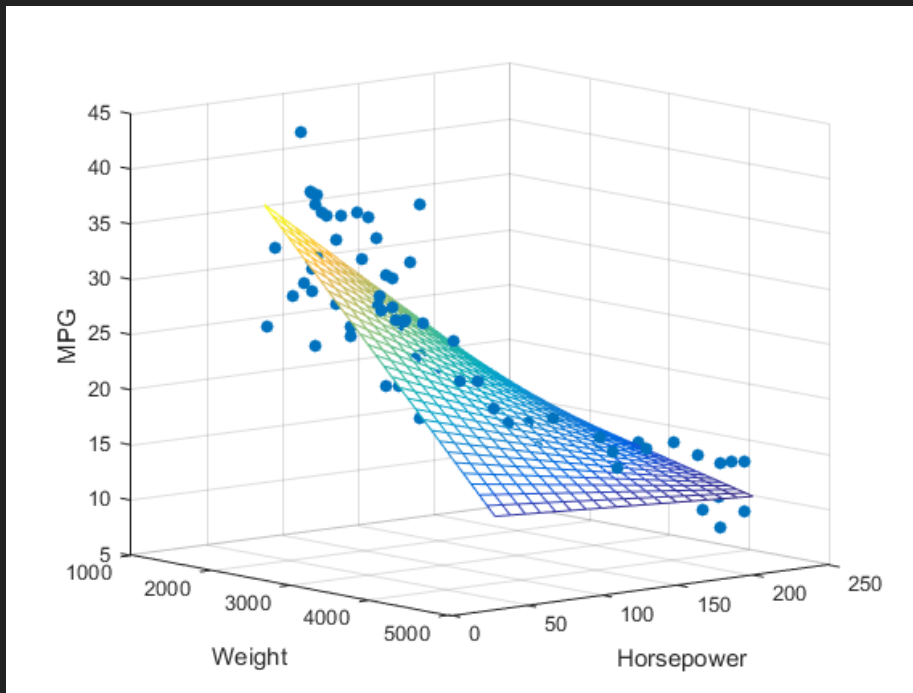
Before running grid optimization make sure your metric is stable (i.e. by train/testing on different subsets).

Overfitting by using many attempts is real issue.



# OPTIMAL GRID SEARCH

- stochastic optimization (Metropolis-Hastings, annealing)
- regression techniques, reusing all known information (ML to optimize ML!)



# OPTIMAL GRID SEARCH USING REGRESSION

General algorithm (point of grid = set of parameters):

1. evaluations at random points
2. build regression model based on known results
3. select the point with best expected quality according to trained model
4. evaluate quality at this points
5. Go to 2 if not enough evaluations

Why not using linear regression?

# GAUSSIAN PROCESSES FOR REGRESSION

Some definitions:  $Y \sim GP(m, K)$ , where  $m$  and  $K$  are functions of mean and covariance:  $m(x)$ ,  $K(x_1, x_2)$

- $m(x) = \mathbb{E}Y(x)$  represents our prior expectation of quality (may taken constant)
- $K(x_1, x_2) = \mathbb{E}Y(x_1)Y(x_2)$  represents influence of known results on expectation of new
- RBF kernel is the most useful:  
$$K(x_1, x_2) = \exp(-|x_1 - x_2|^2)$$

We can model the posterior distribution of results in each point

point.

Gaussian Process Demo on Mathematica



Also see: <http://www.tmpl.fi/gp/>.

**X** MINUTES BREAK

# PREDICTION SPEED

## Cuts vs classifier

- cuts are interpretable
- cuts are applied really fast
- ML classifiers are applied much slower

# SPEED UP WAYS

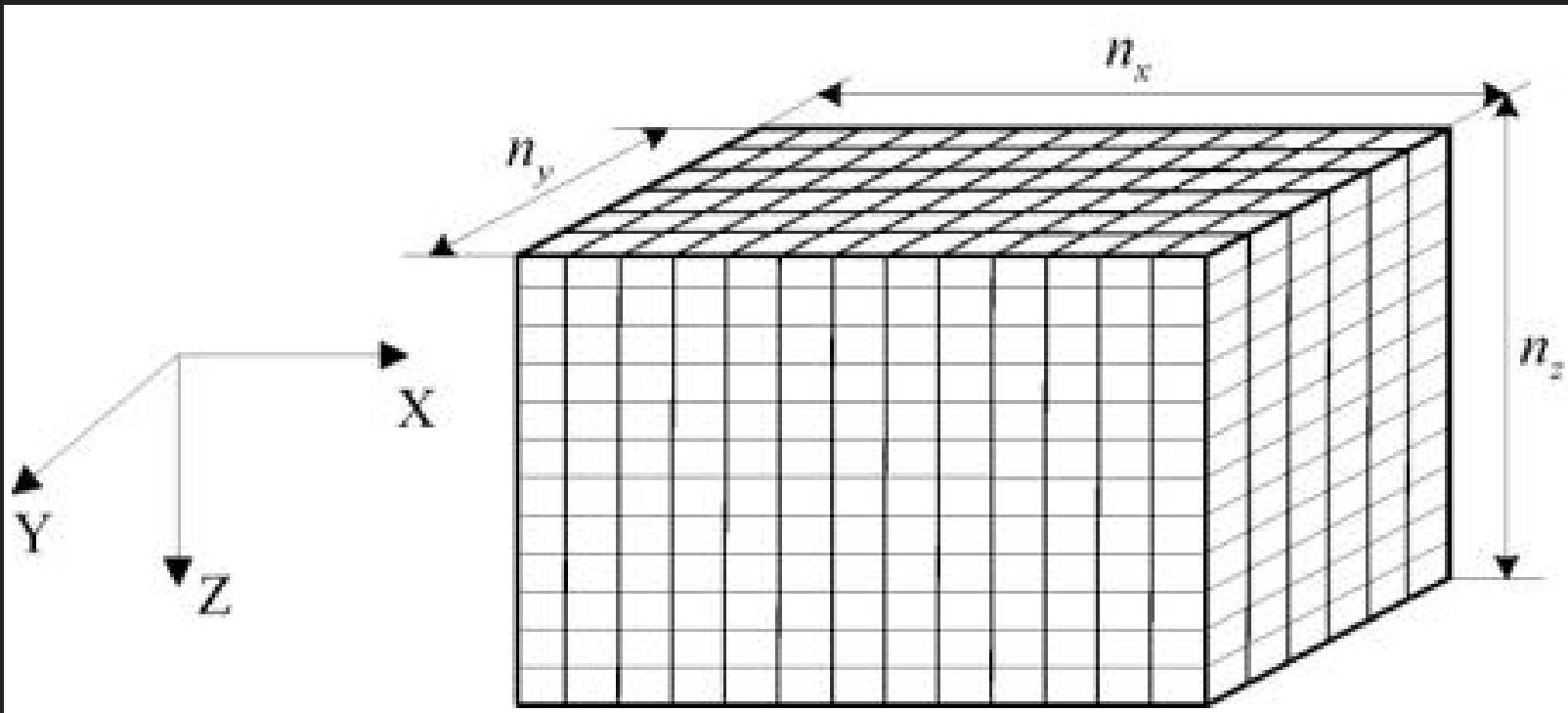
## Method-specific

- Logistic regression: sparsity
- Neural networks: removing neurons
- GBDT: pruning by selecting best trees

## Staging of classifiers (a-la triggers)

# SPEED UP: LOOKUP TABLES

Split space of features into bins, simple piecewise-constant function





# SPEED UP: LOOKUP TABLES

## Training:

1. split each variable into bins
2. replace values with index of bin
3. train any classifier on indices of bins
4. create lookup table (evaluate answer for each combination of bins)

## Prediction:

1. convert features to bins' indices
2. take answer from lookup table

# LOOKUP TABLES

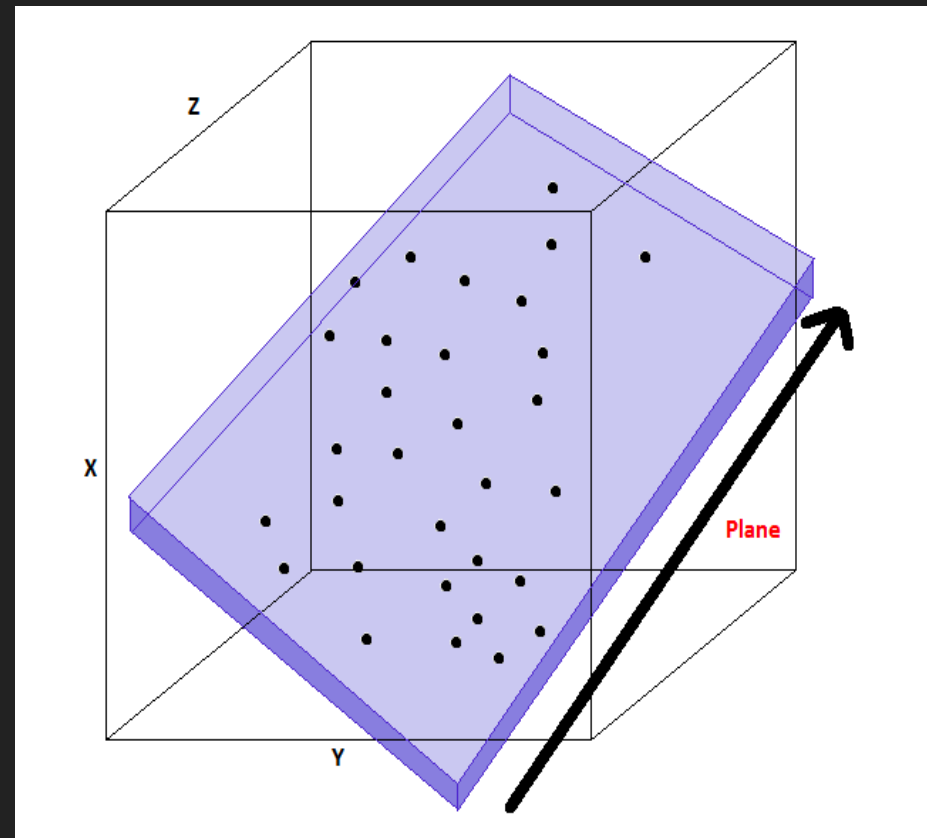
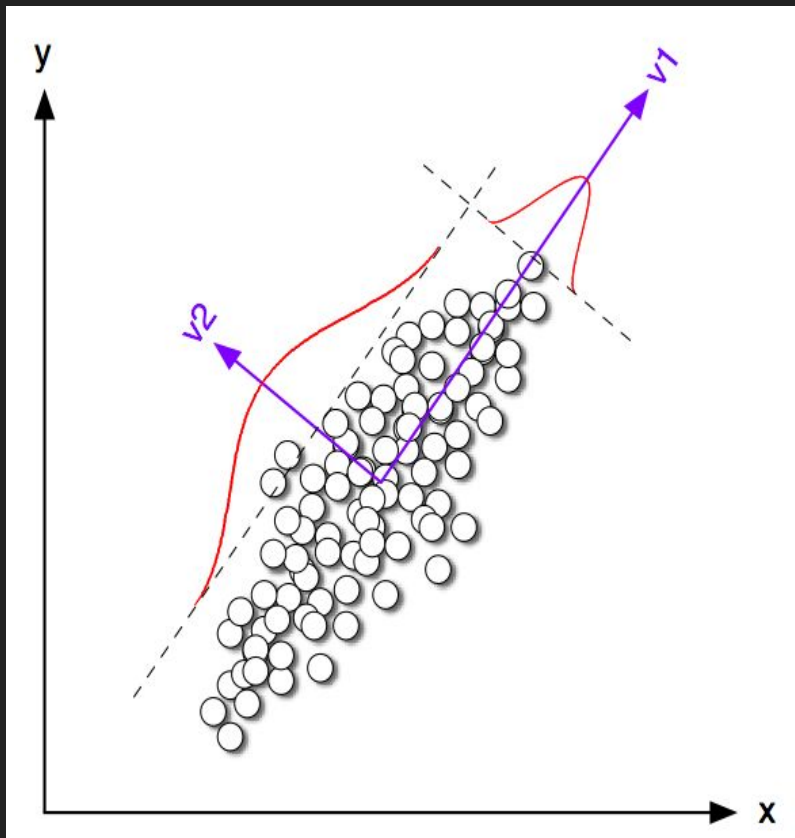
- speed is comparable to cuts
- allows to use any ML model behind
- used in LHCb topological trigger ('Bonsai BDT')

## Problems:

- too many combination when number of features  $N > 10$
- $8^{10} \sim 1Gb$  (8 bins in 10 features),
- finding optimal thresholds of bins

# UNSUPERVISED LEARNING: PCA [PEARSON, 1901]

PCA is finding axes along which variation is maximal  
(based on principal axis theorem)



# PCA DESCRIPTION

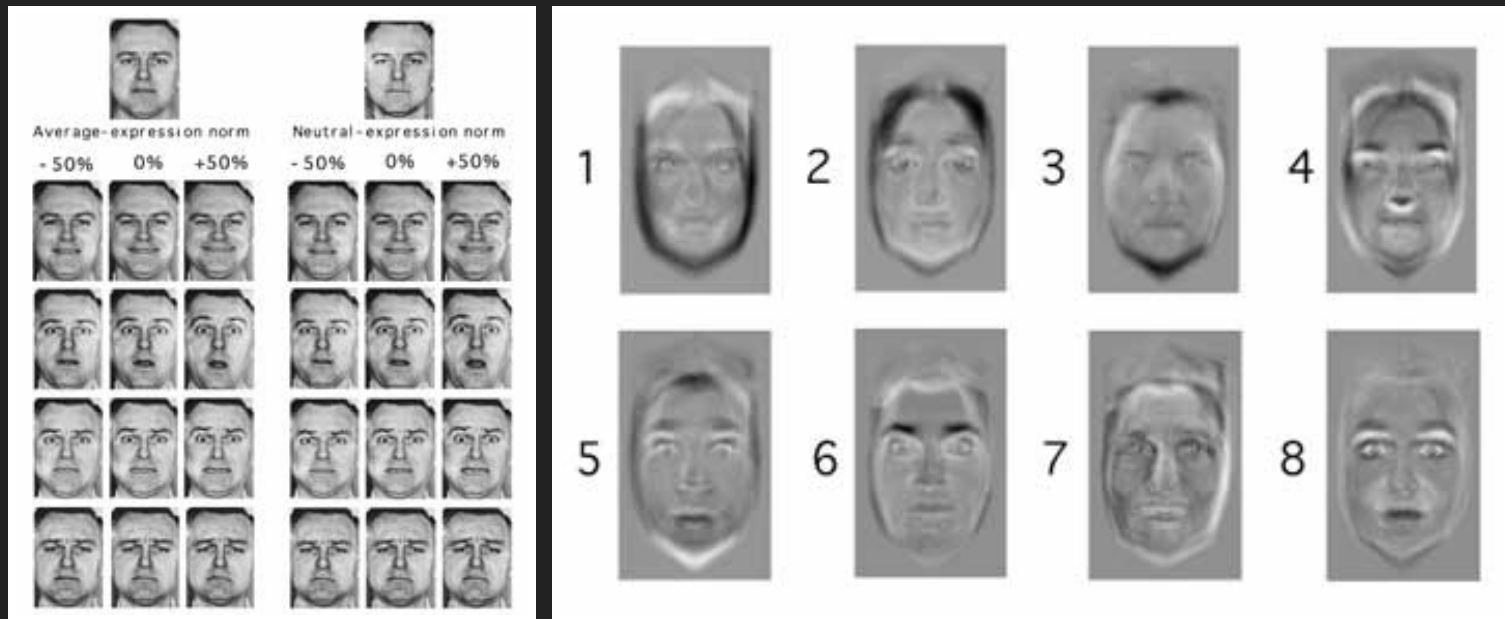
PCA is based on the principal axes

$$Q = U^T \Lambda U$$

$U$  is orthogonal matrix,  $\Lambda$  is diagonal matrix.

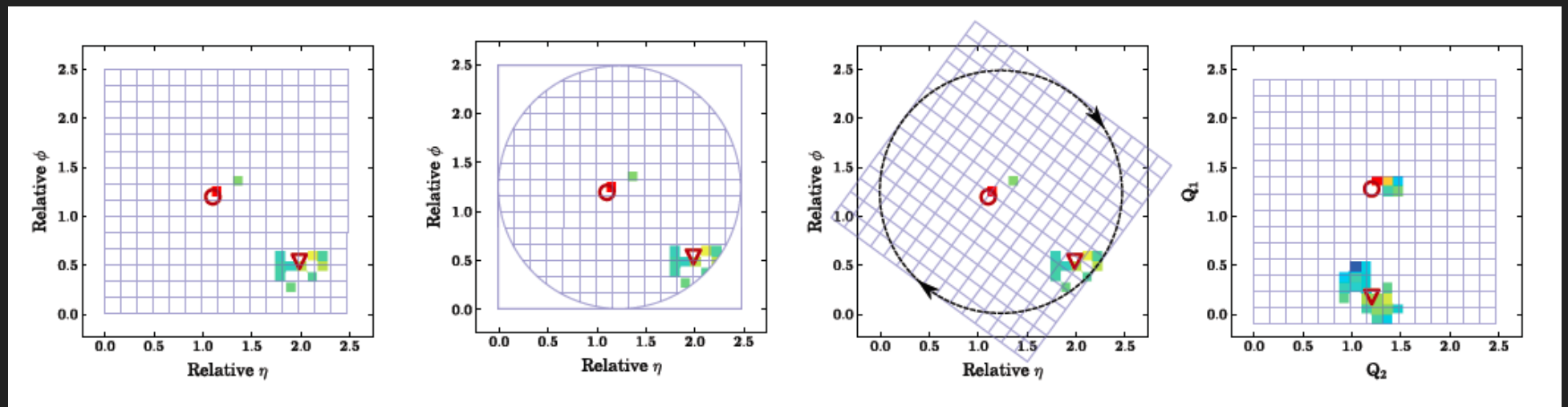
$$\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n), \quad \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$$

# PCA: EIGENFACES



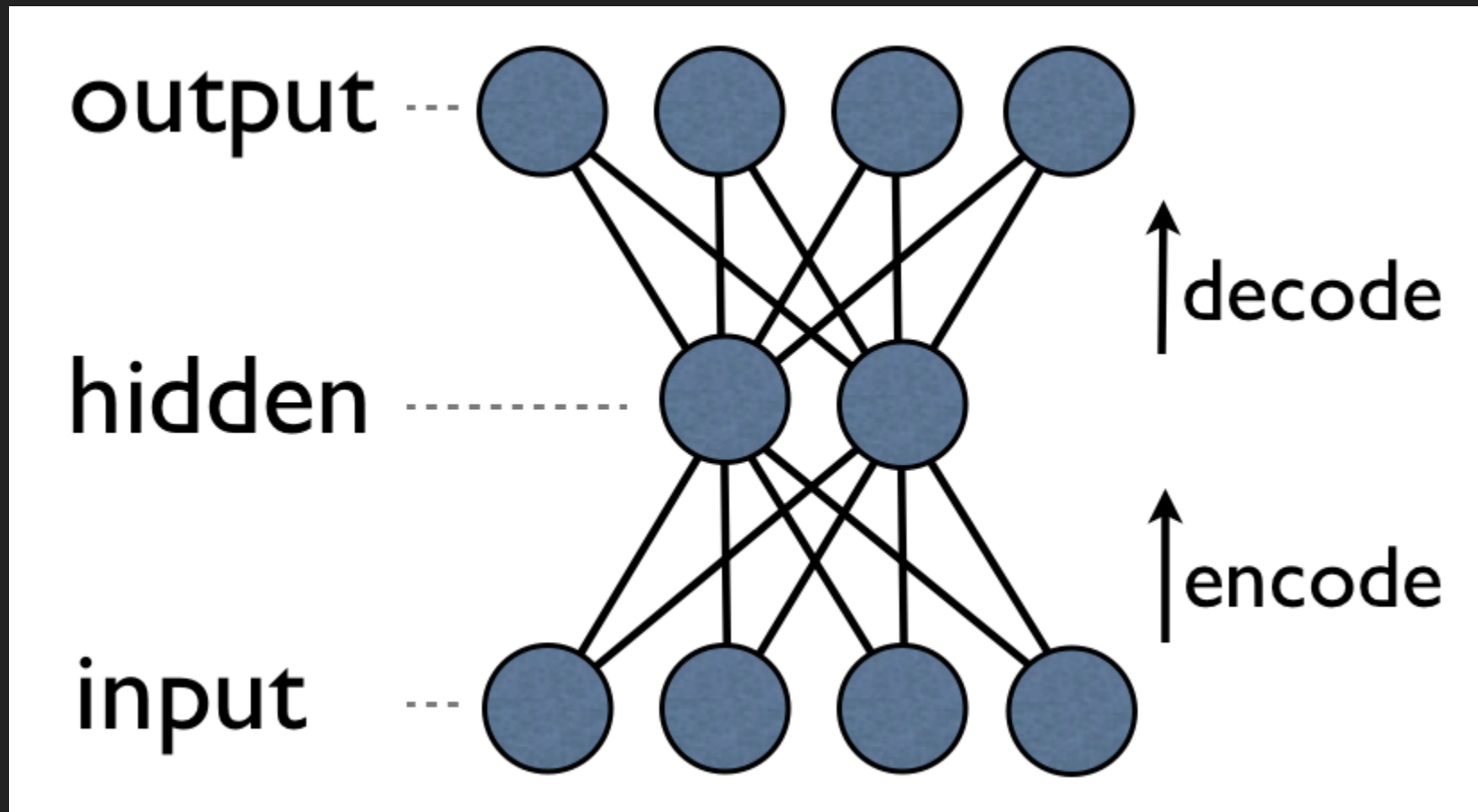
$$\text{Emotion} = \alpha[\text{scared}] + \beta[\text{laughs}] + \gamma[\text{angry}] + \dots$$

# PCA: CALORIMETER



Electron/jet recognition in ATLAS

# AUTOENCODERS



autoencoding:  $y = x$ , error  $|\hat{y} - y| \rightarrow \min$  hidden layer is smaller

# CONNECTION TO PCA

When optimizing MSE:

$$\sum_i (\hat{y}_i - x_i)^2$$

And activation is linear:

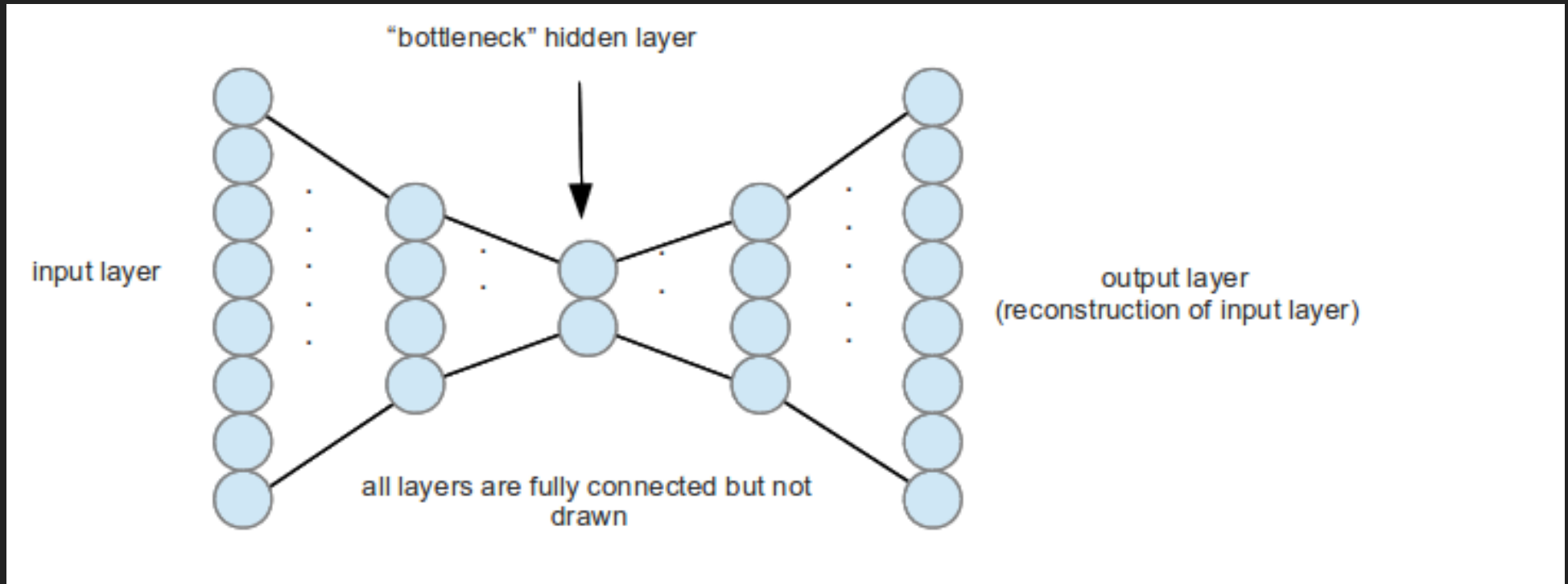
$$h_i = \sum_j w_{ij} x_j$$

$$\hat{y}_i = \sum_j a_{ij} h_j$$



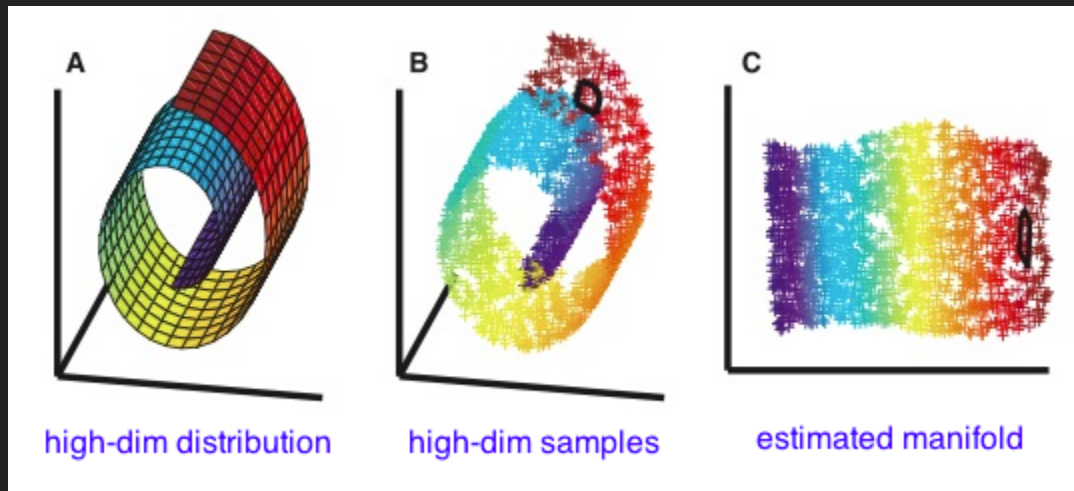
Optimal solution of optimization problem is PCA.

# BOTTLENECK

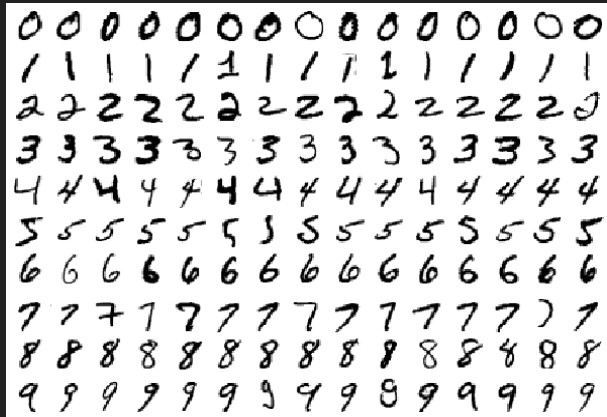


There is an informational bottleneck  $\Rightarrow$  algorithm will eager to pass the most precious information.

# MANIFOLD LEARNING TECHNIQUES

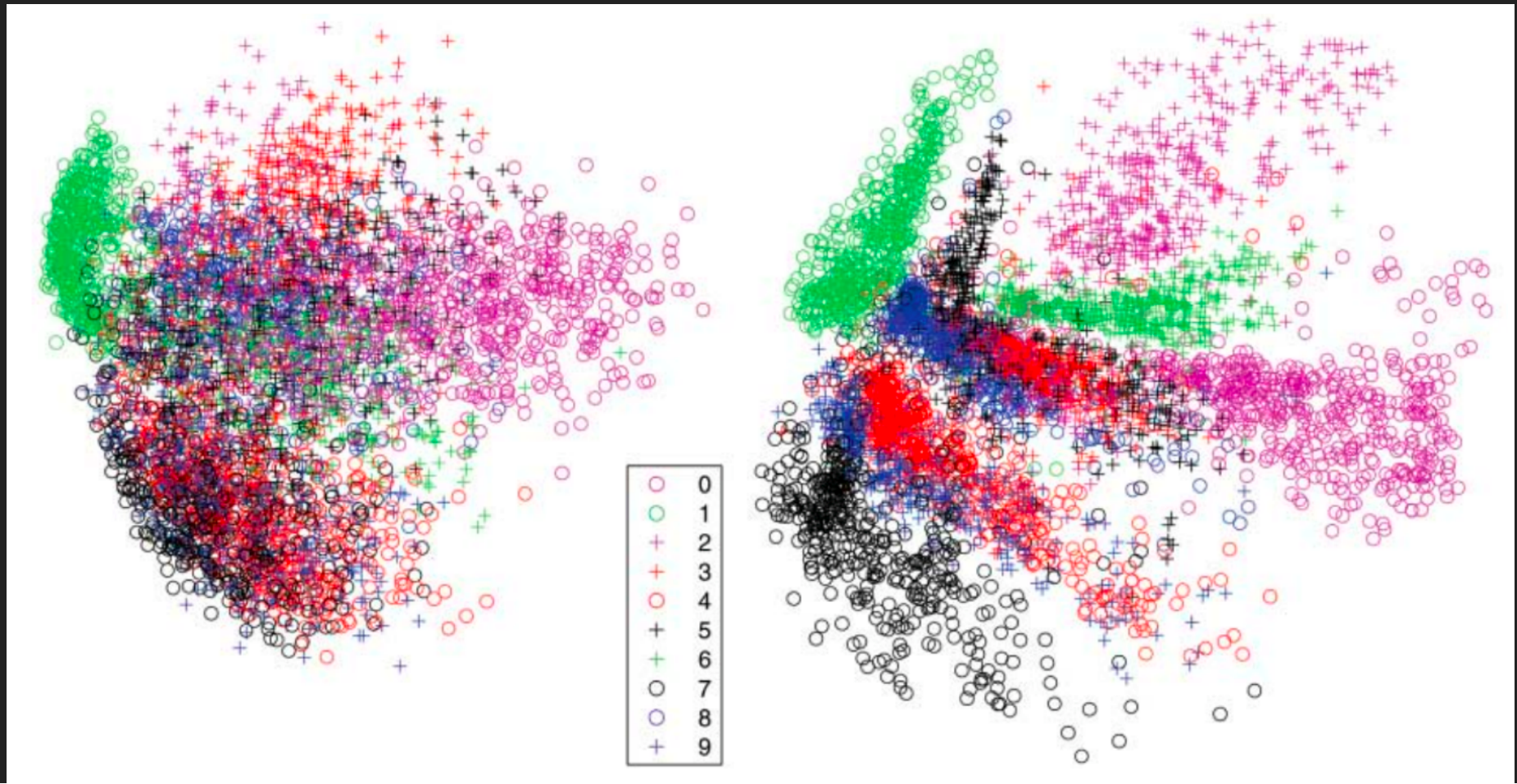


Preserve neighbourhood/distances.



Digits samples from MNIST

PCA and autoencoder:

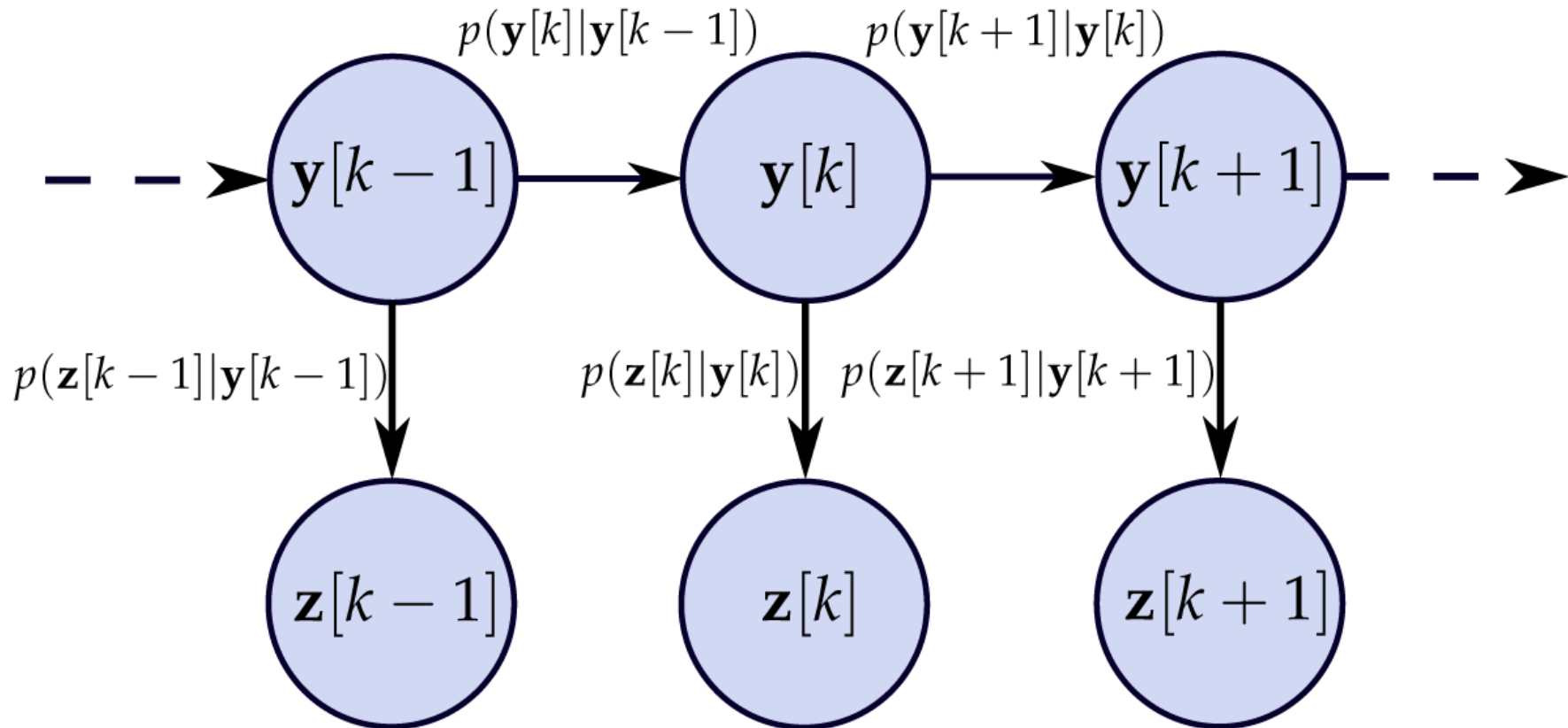


# PART-OF-SPEECH TAGGING

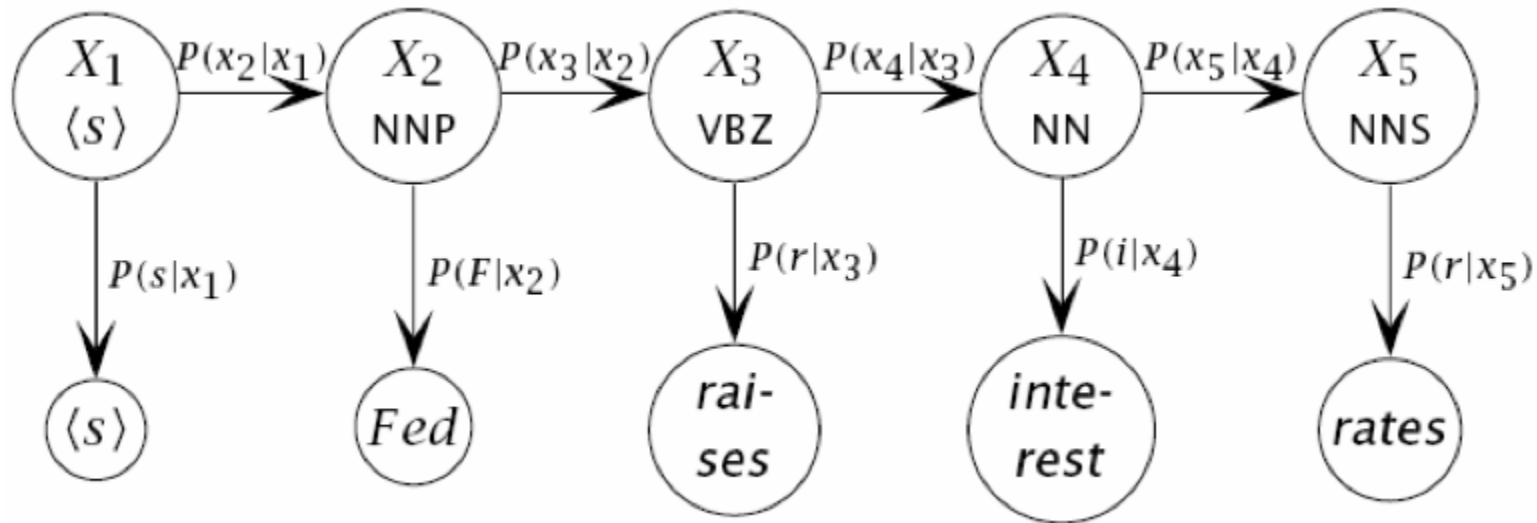
## Ambiguity

- “**Plants**/N need light and water.”
- “Each one **plant**/V one.”
- “Flies like a flower”
  - *Flies*: noun or verb?
  - *like*: preposition, adverb, conjunction, noun, or verb?
  - *a*: article, noun, or preposition?
  - *flower*: noun or verb?

# MARKOV CHAIN

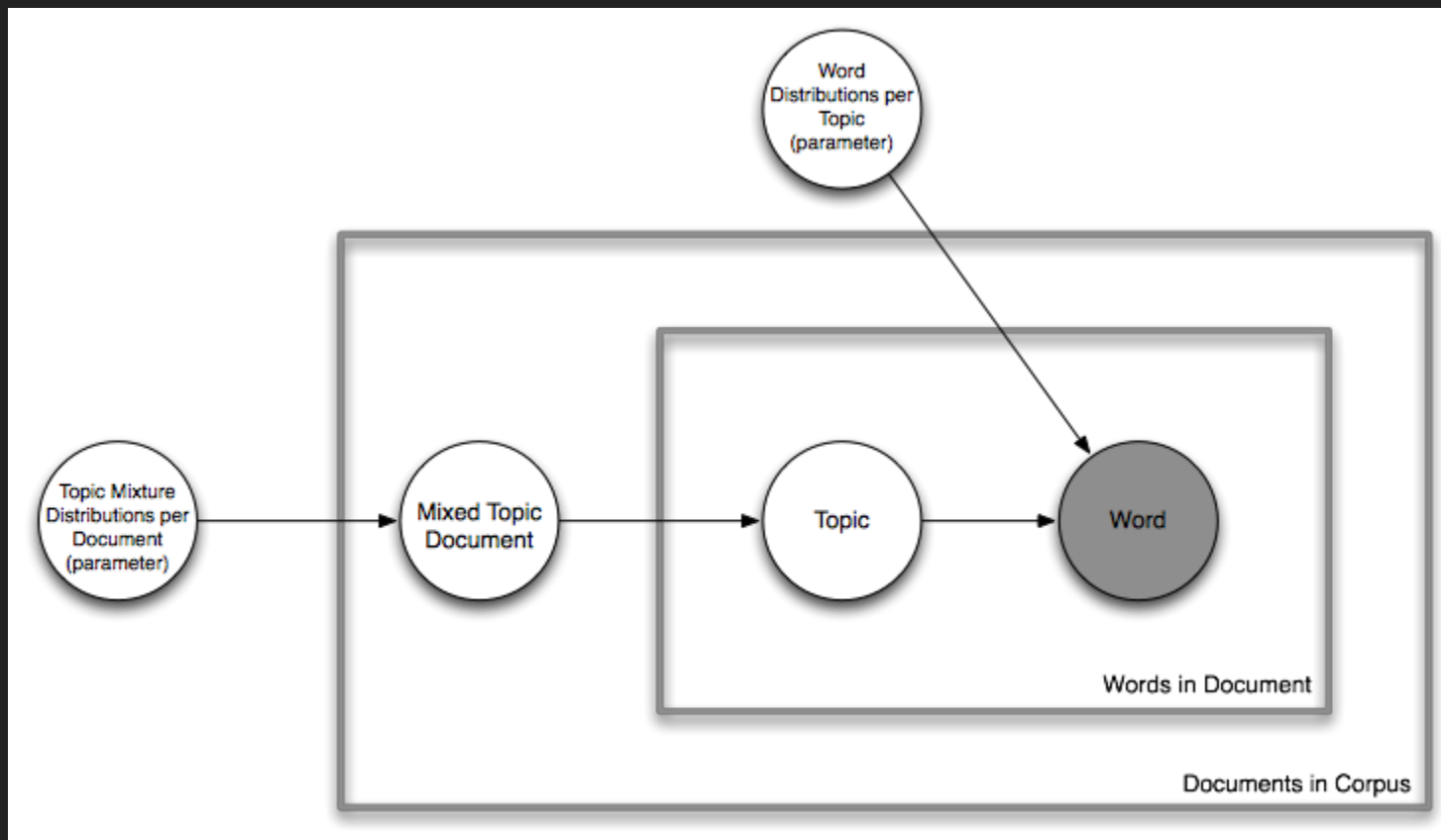


# MARKOV CHAIN IN TEXT



# TOPIC MODELLING (LDA)

- bag of words
- generative model  $p(x)$  that describe different probabilities
- fitting by maximizing log-likelihood
- parameters of this model are useful



# COLLABORATIVE RESEARCH

- problem: different environments
- problem: data ping-pong
- solution: dedicated machine (server) for a team
- restart the research = run script or notebook

```
!ping www.bbc.co.uk
```



# REPRODUCIBLE RESEARCH

- readable code
- code + description + plots together
- keep versions + reviews

Solution: IPython notebook + github

Example notebook

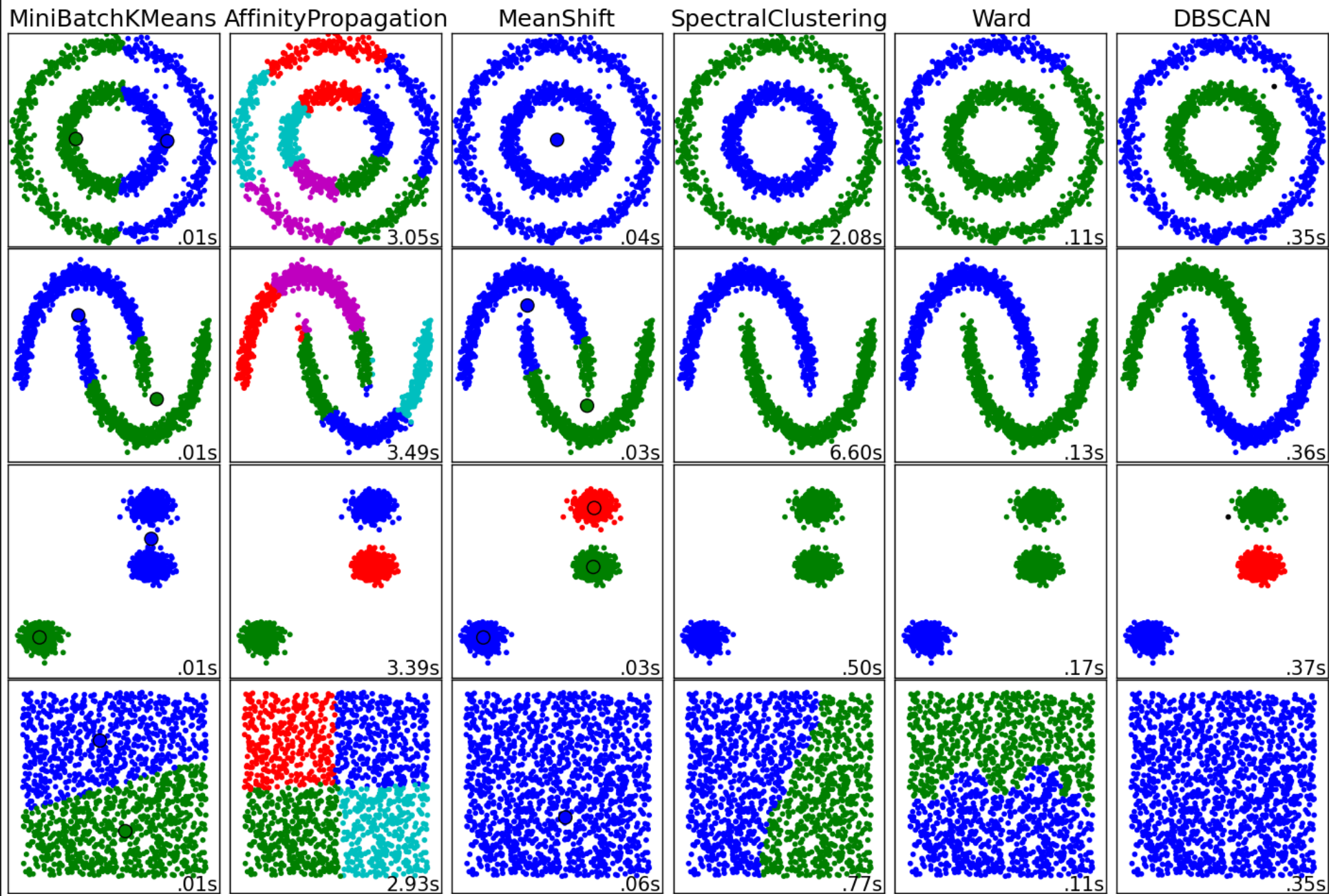
## EVEN MORE REPRODUCIBILITY?

- Analysis preservation: VM with all needed stuff.
- Docker

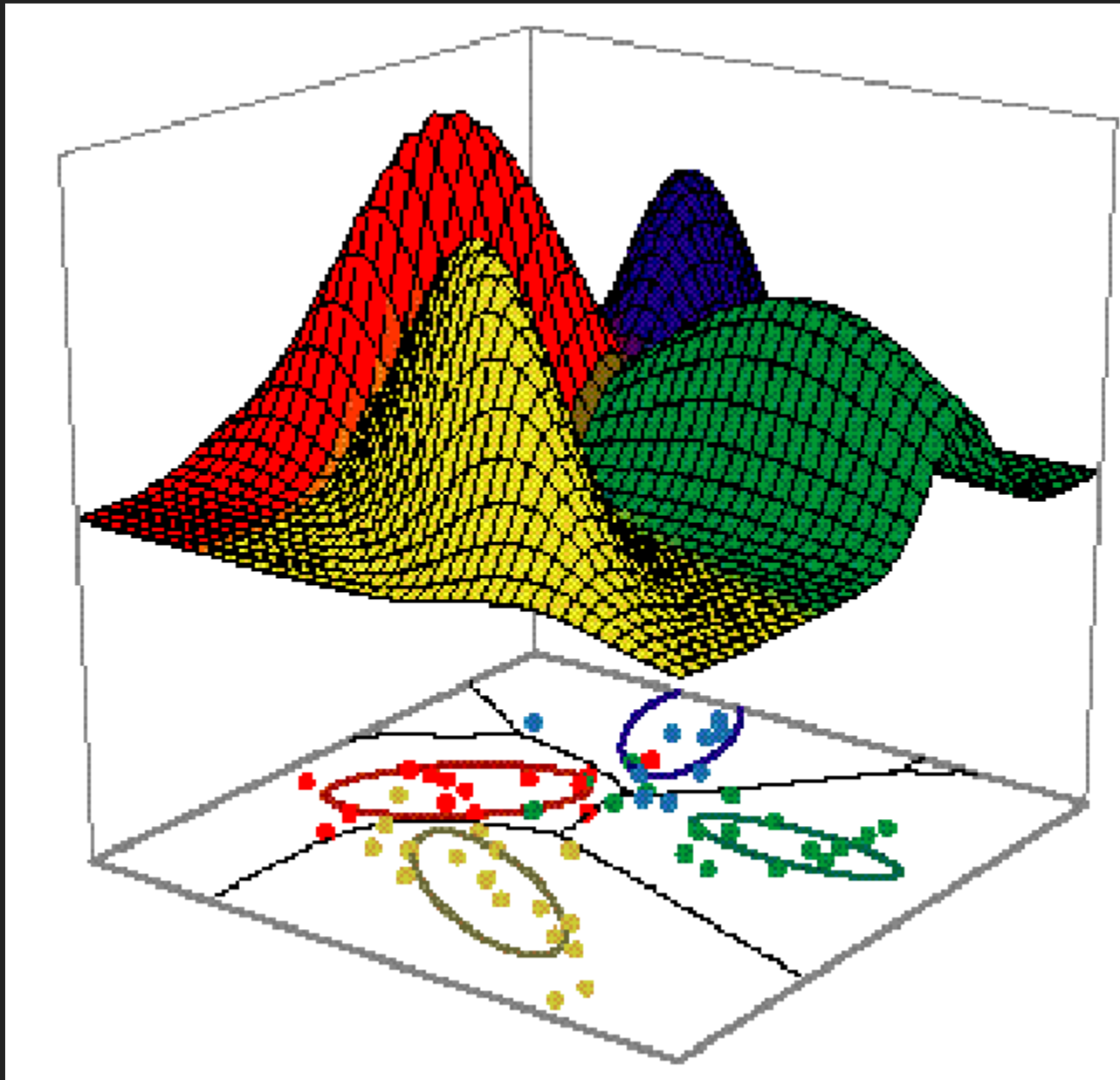
# BIRD'S EYE VIEW TO ML:

- Classification
- Regression
- Ranking
- Dimensionality reduction

# CLUSTERING



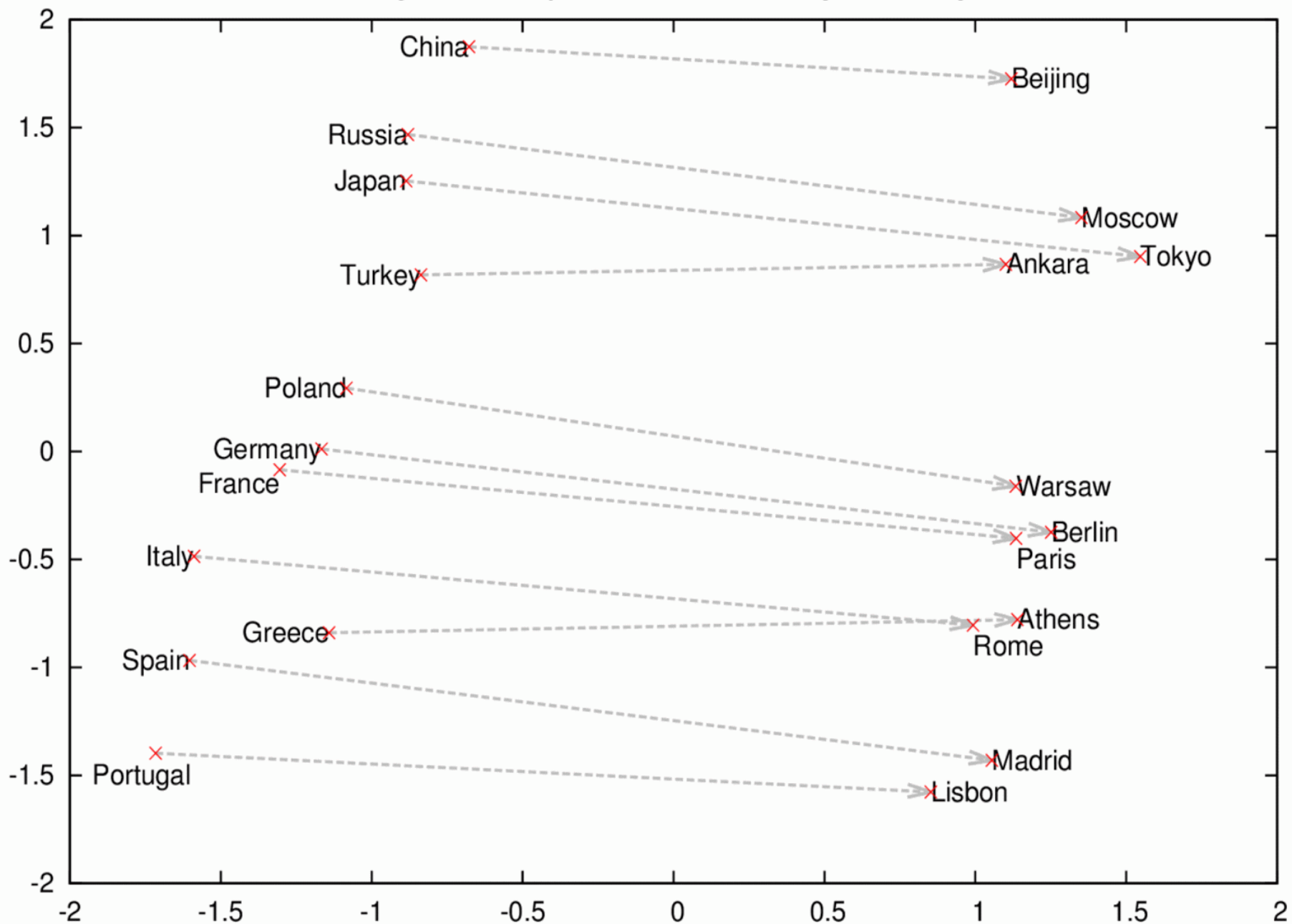
# DENSITY ESTIMATION



# REPRESENTATION LEARNING

Word2vec finds representations of words.

# Country and Capital Vectors Projected by PCA



**THE END**