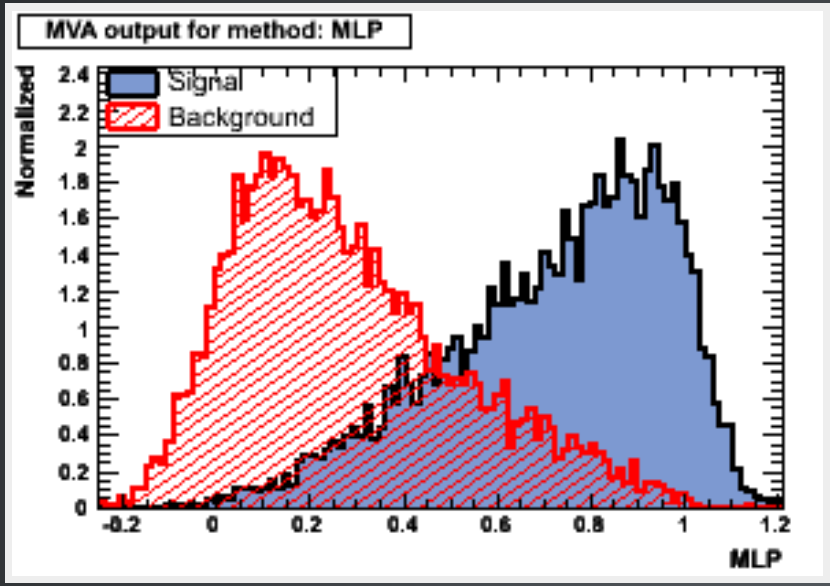
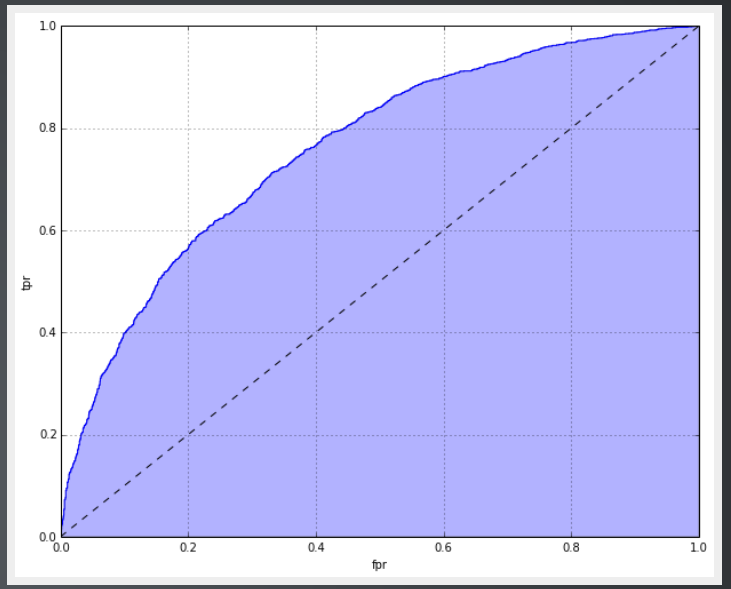
MACHINE LEARNING IN HIGH ENERGY PHYSICS

PRACTICAL CLASS #4

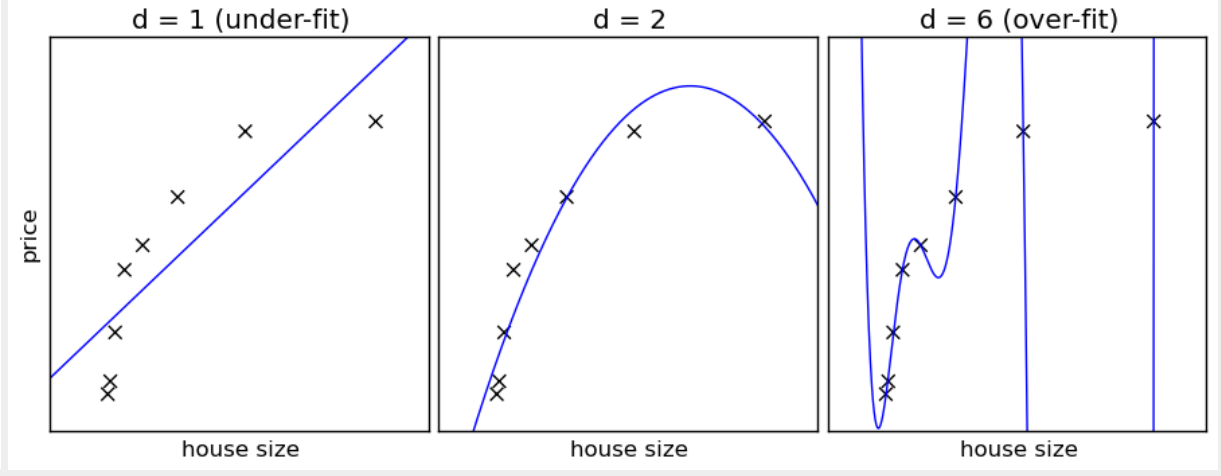
Alex Rogozhnikov, 2015

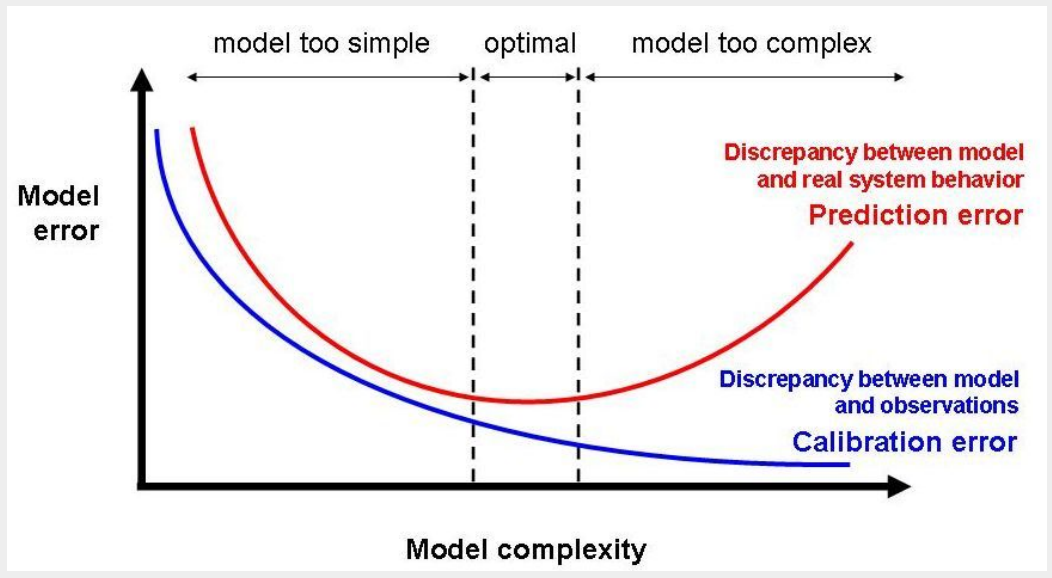
RECAP: ROC-CURVE



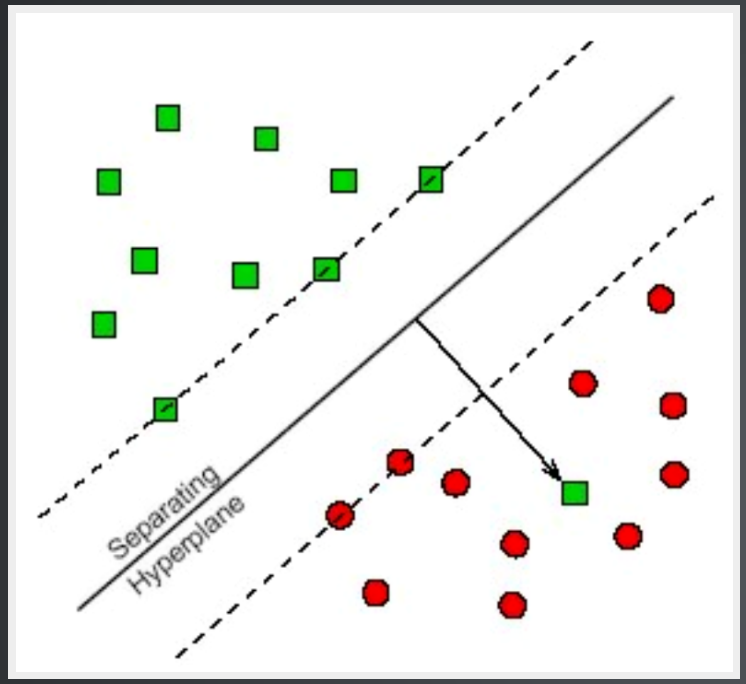


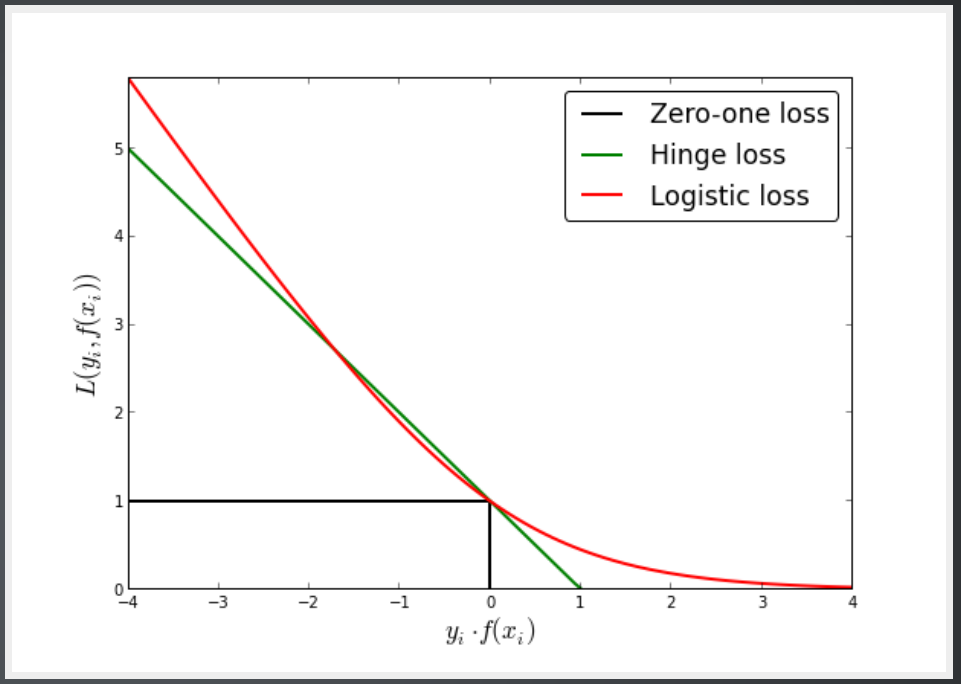
RECAP: OVERFITTING VS OVERFITTING



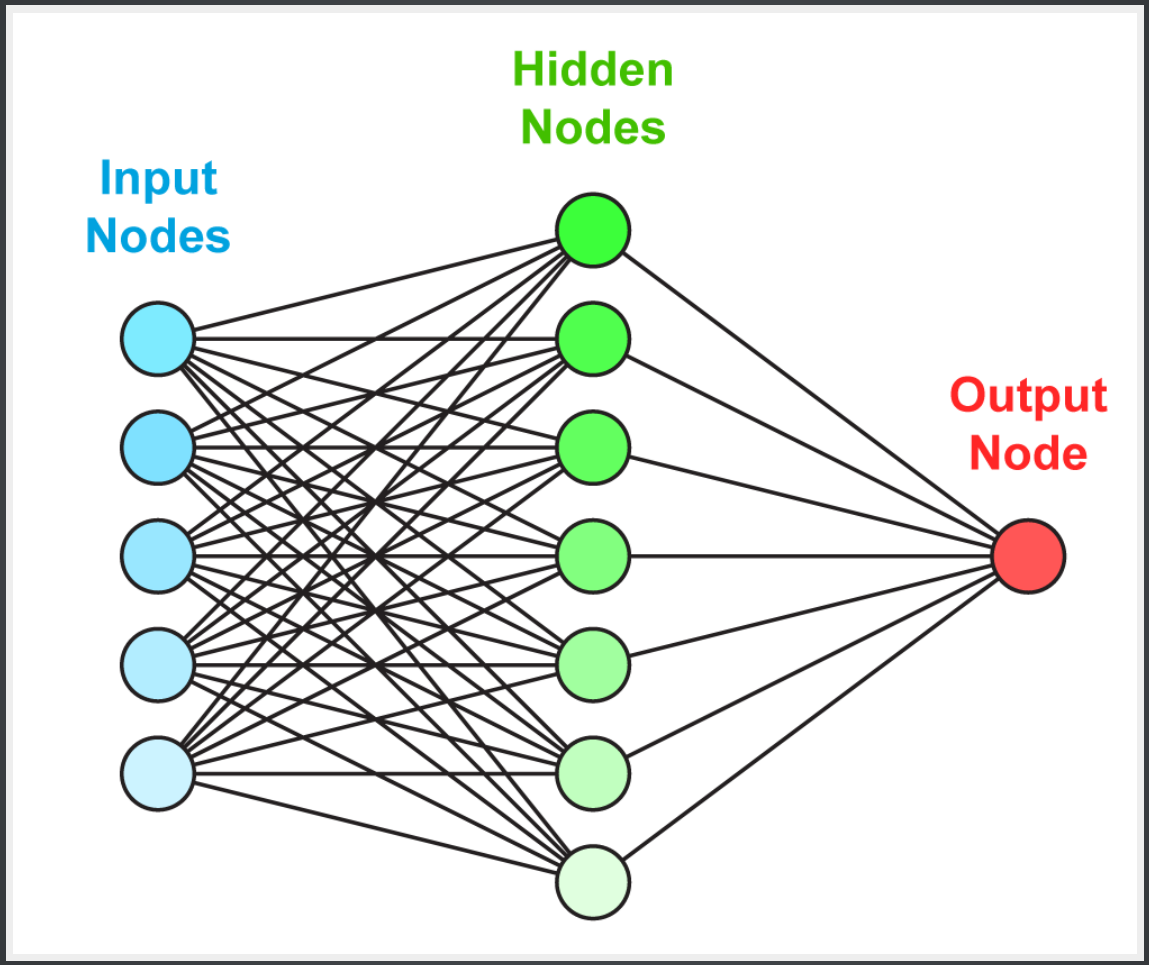


RECAP: LOGISTIC REGRESSION AND SVM

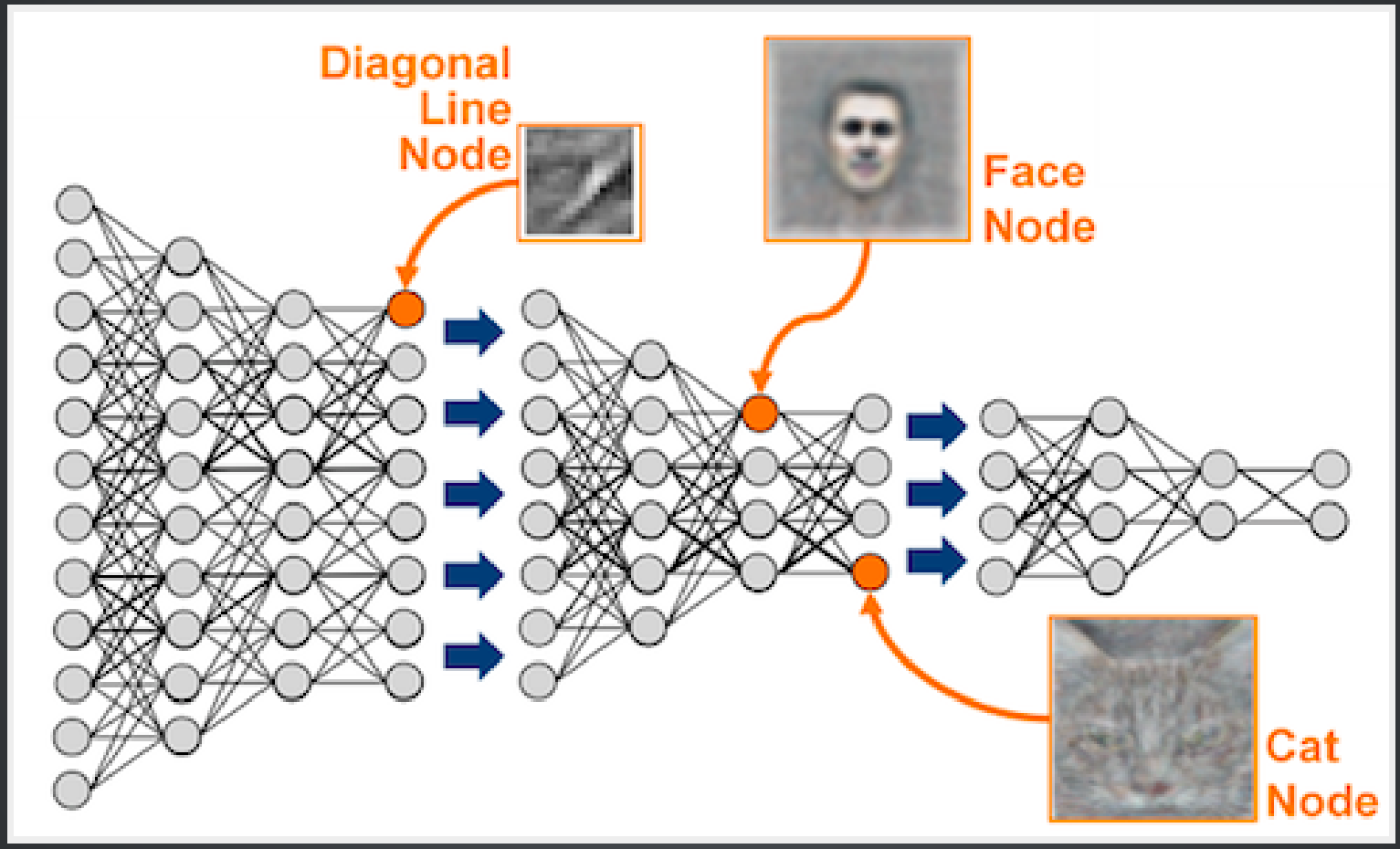




RECAP: NEURAL NETWORKS

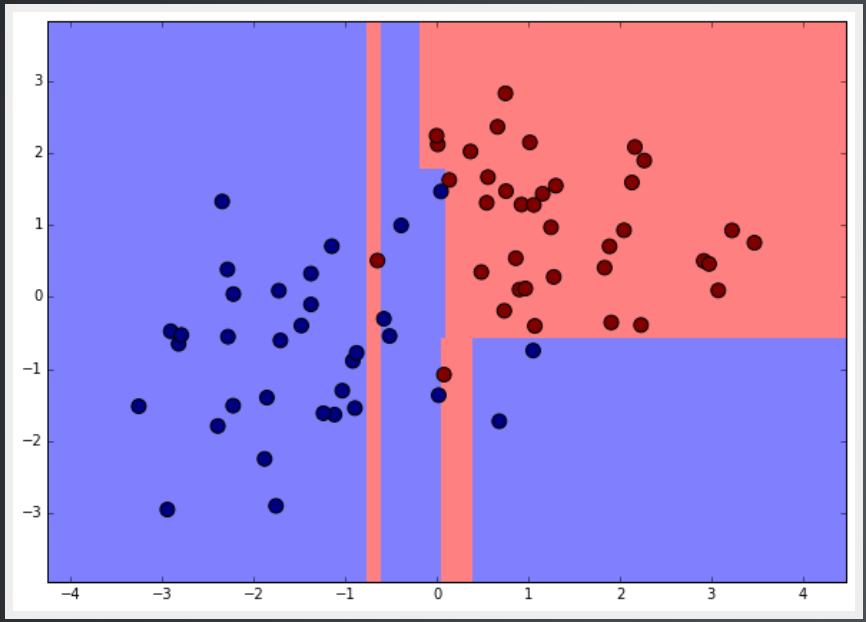


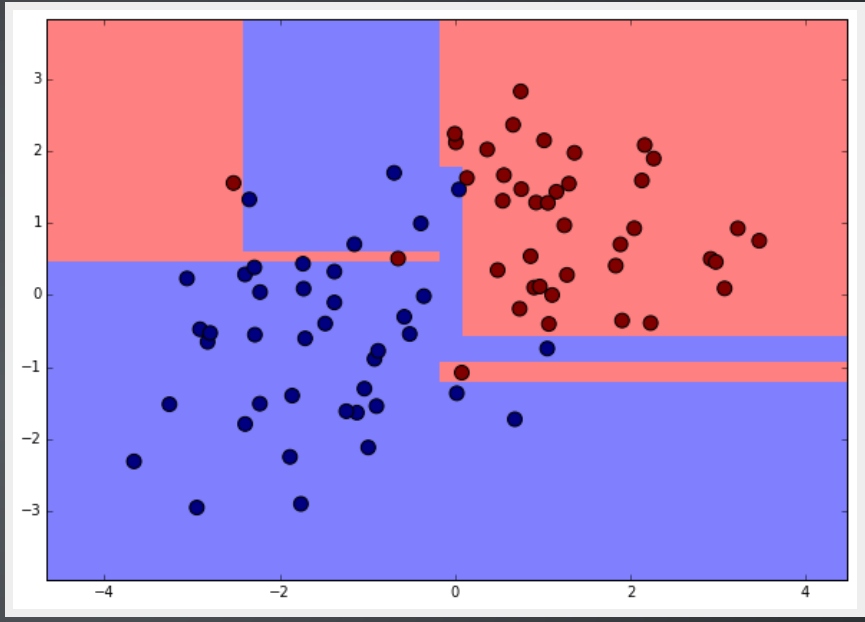
RECAP: DEEP NEURAL NETWORKS



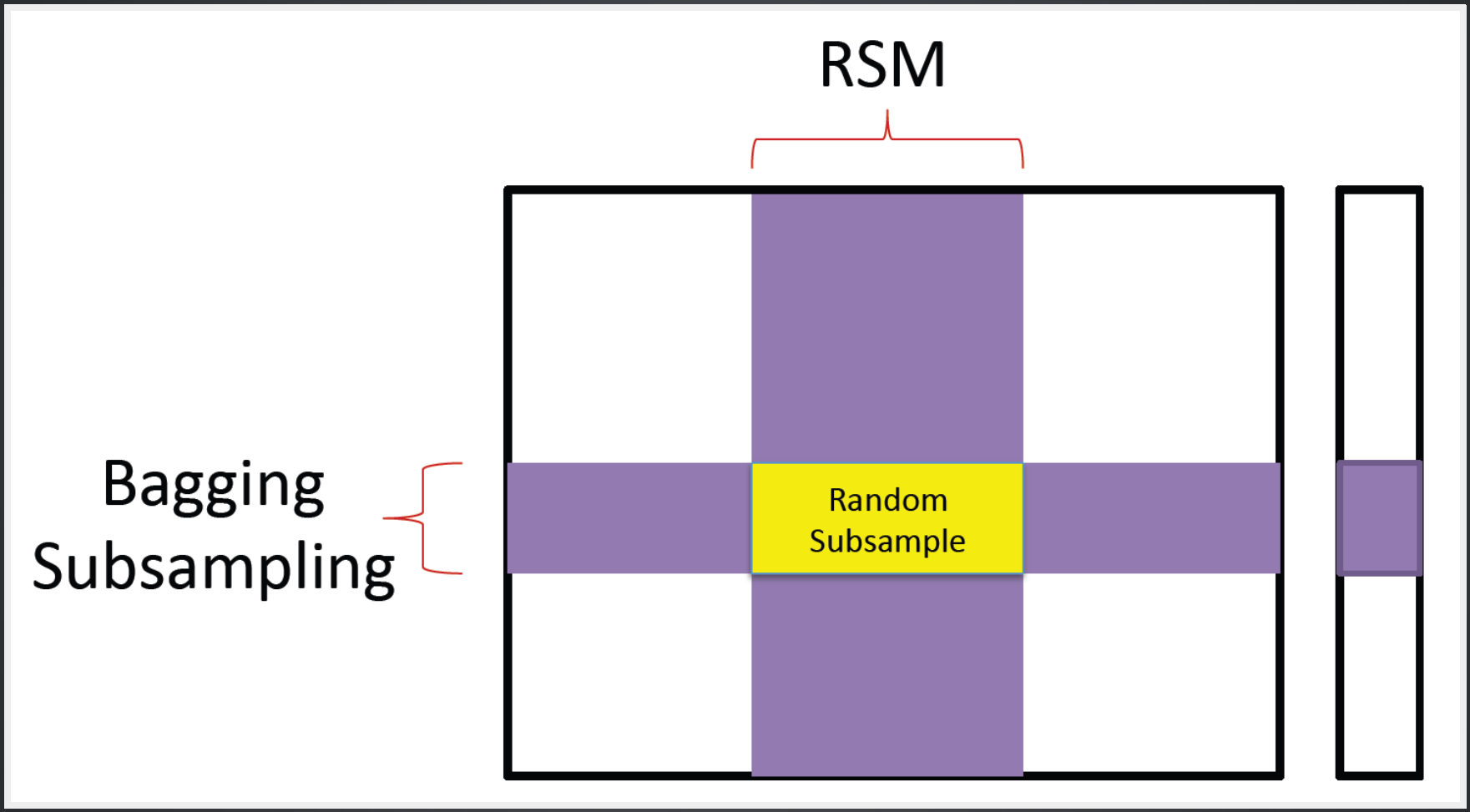
DECISION TREE

Tree is unstable!





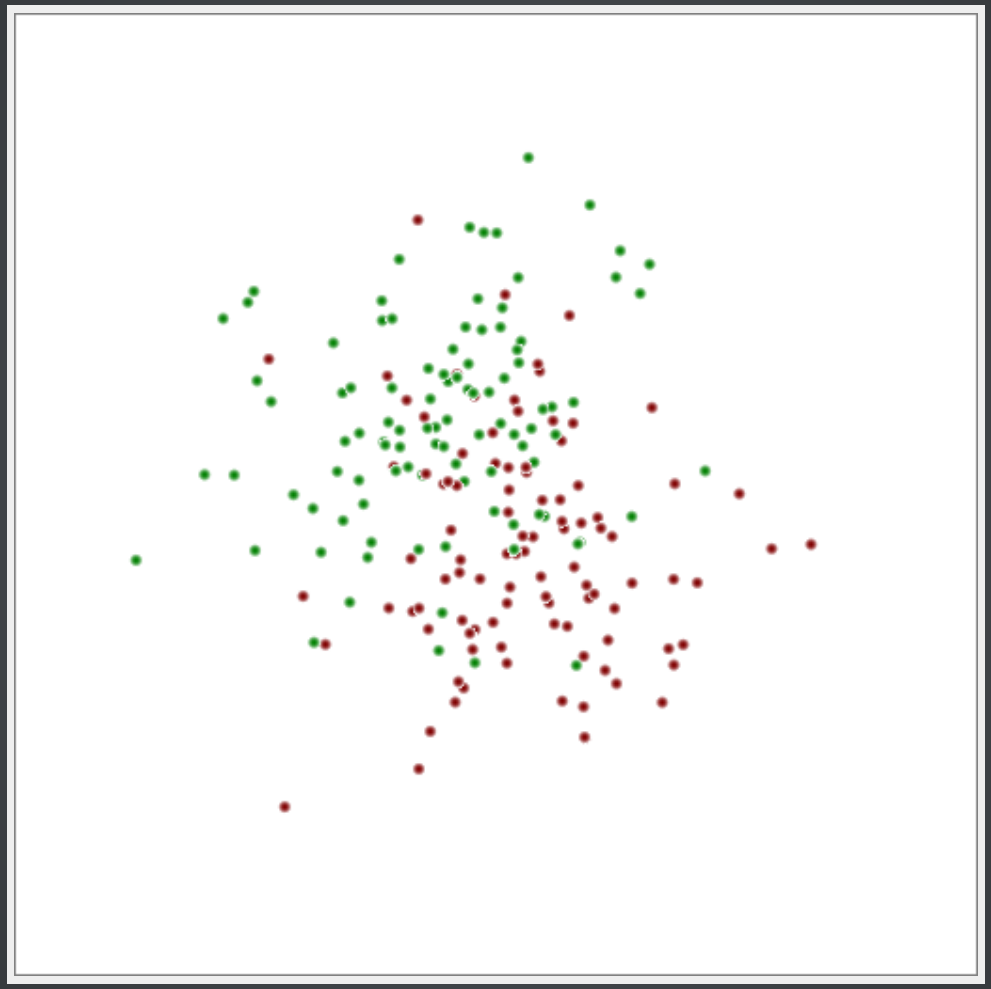
RANDOM FOREST



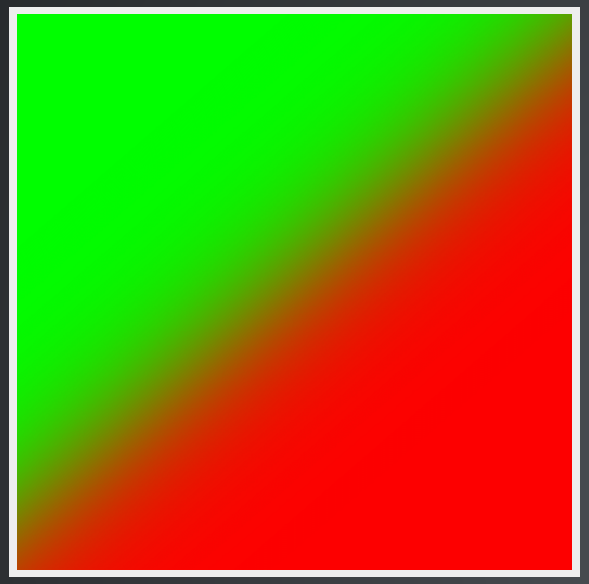
RANDOM FOREST

- Random forest is ensembling over trees which are built independently.
- Each tree is trained on different features and different parts of training sample.
- Simple averaging is used to compute prediction of forest.
- Usually deep trees are used.

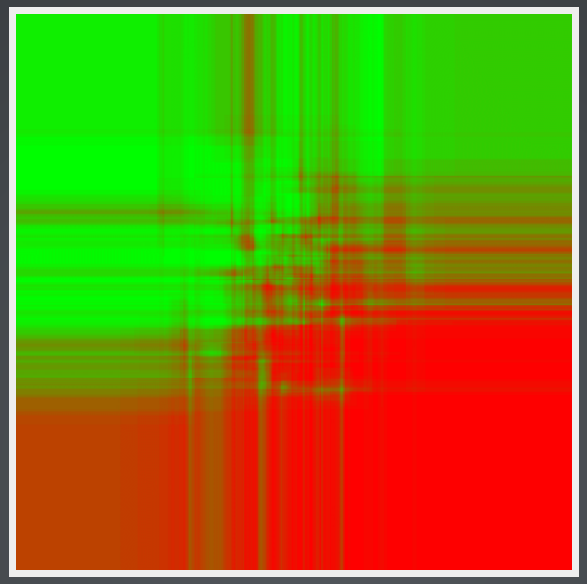
RF: EXAMPLE DATASET



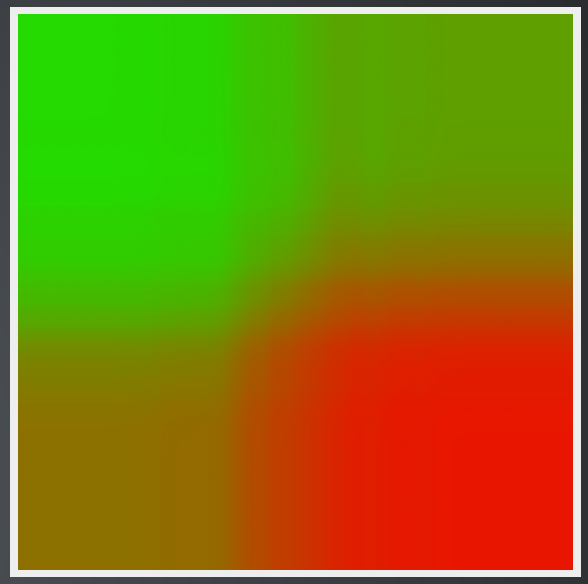
RF: EXAMPLE DATASET



Optimal boundary

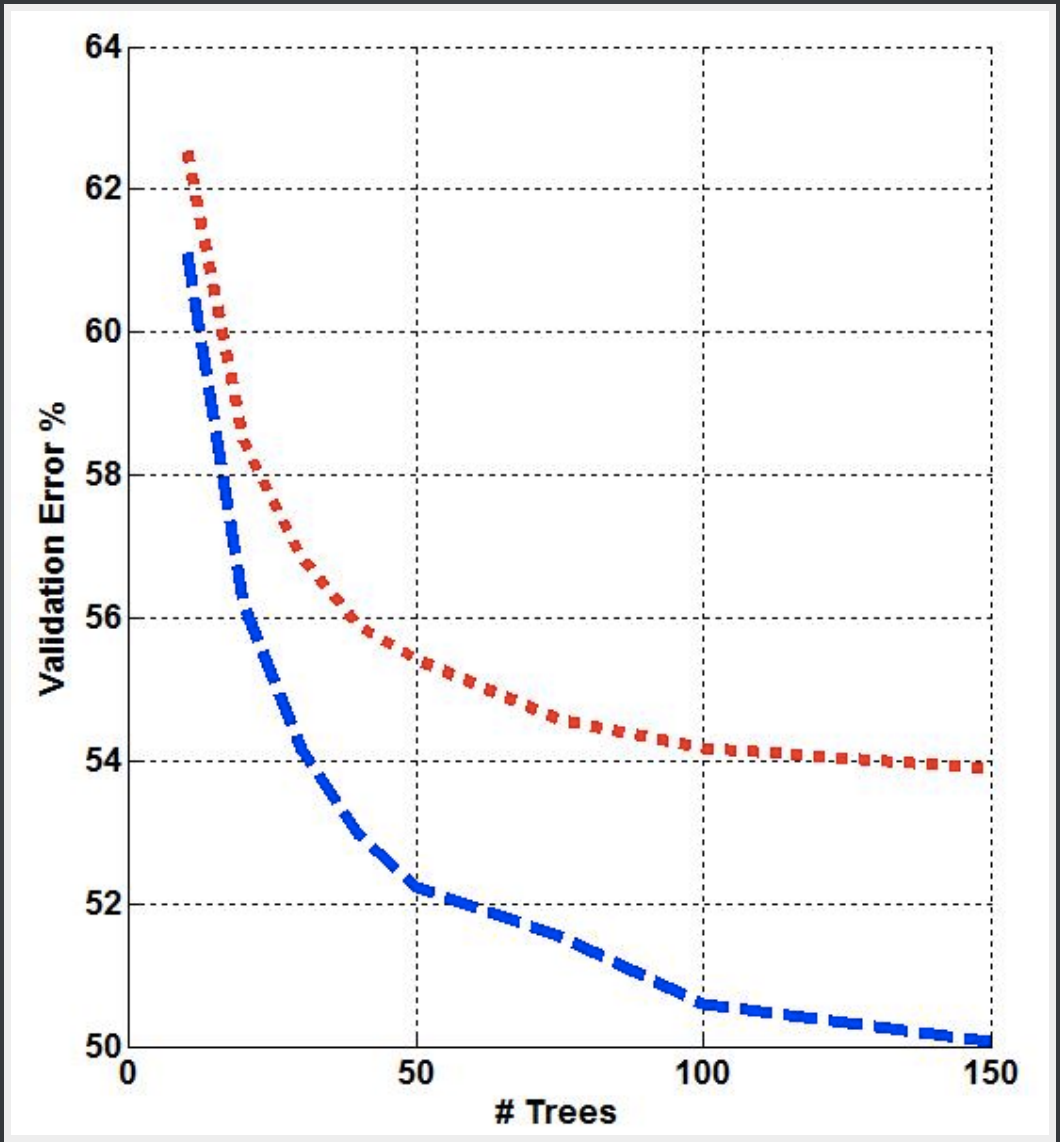


50 trees



2000 trees

Random Forest doesn't overfit!



- Doesn't overfit
- Impressively simple
- Effectively only one parameter:
  number of features used in each tree
- Recommendation: $N_{\text{used}} = \sqrt{N_{\text{features}}}$
- Extremely randomized (extra-) trees

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.

GRADIENT BOOSTING

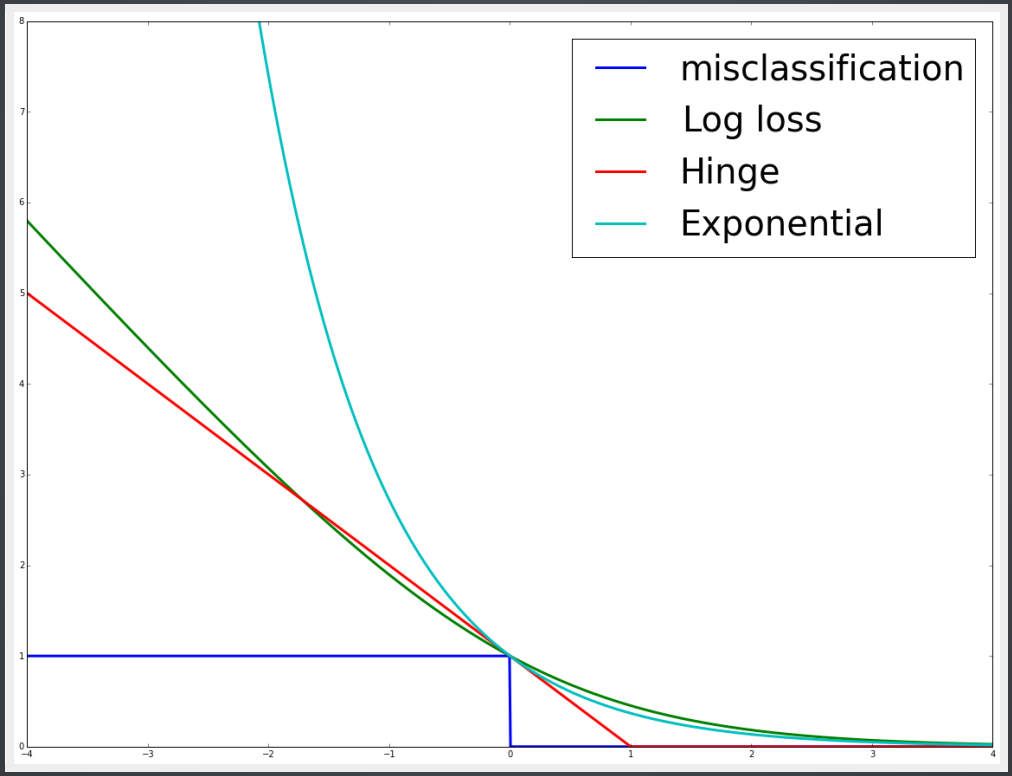
One of the most powerful and flexible methods in ML.

In GB regressors are trained sequentially

LOSS FUNCTIONS

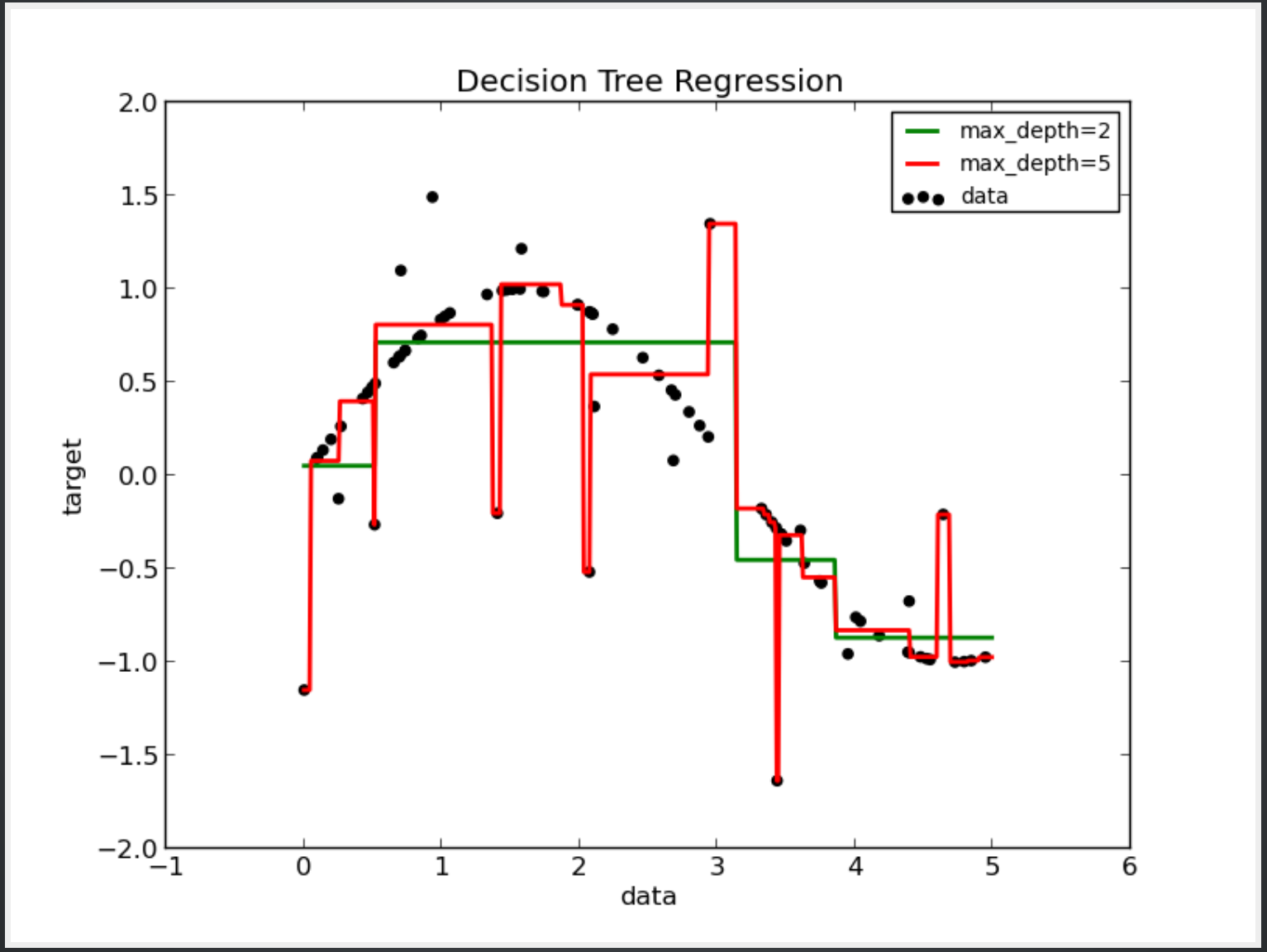
the argument is $y_i \cdot g(x_i)$

the total loss is $Q = \sum_{i=1}^{n} \mathcal{L}(y_i \cdot g(x_i))$



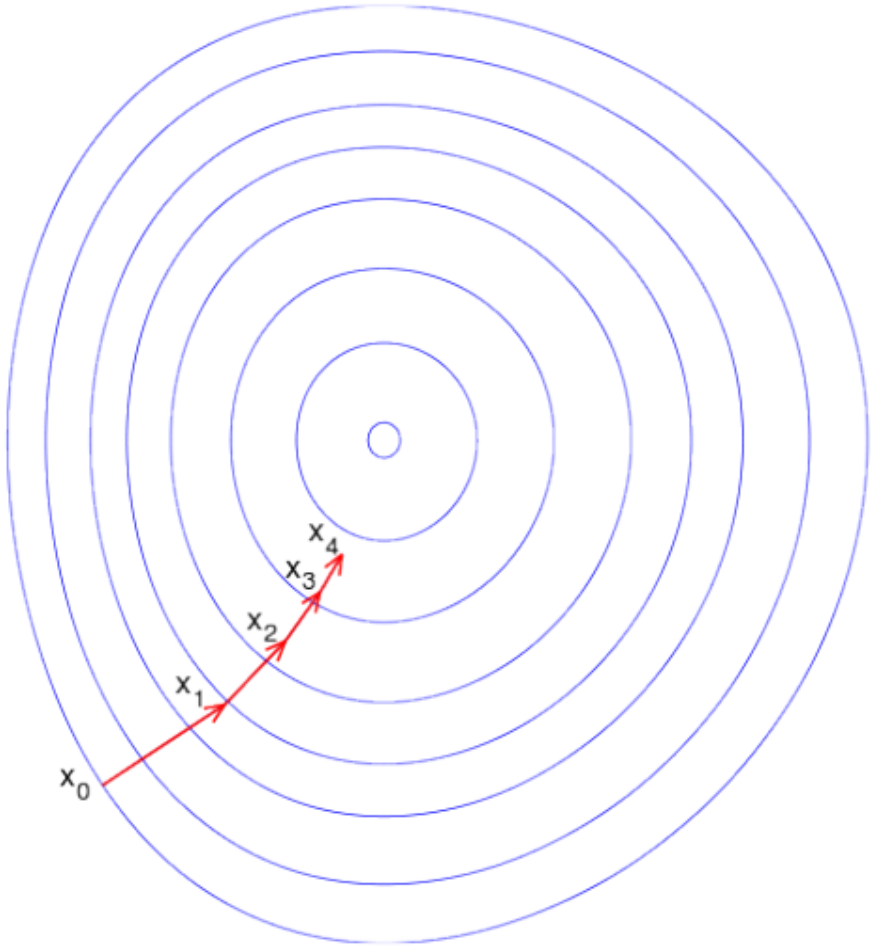
$$g(x_i) = \sum_m c_m \cdot g_m(x_i)$$

REGRESSION WITH TREES



GRADIENT DESCENT

$$\tilde{x}^m = \tilde{x}^{m-1} - \lambda \nabla f(\tilde{x}^{m-1})$$



GRADIENT BOOSTING IDEA

Let $\tilde{y}^m$ be prediction of composition after m stages.

We minimize loss: $Q(\tilde{y}^m, y) = \sum_{i=1}^{N} \mathcal{L}(\tilde{y}_i^m)$

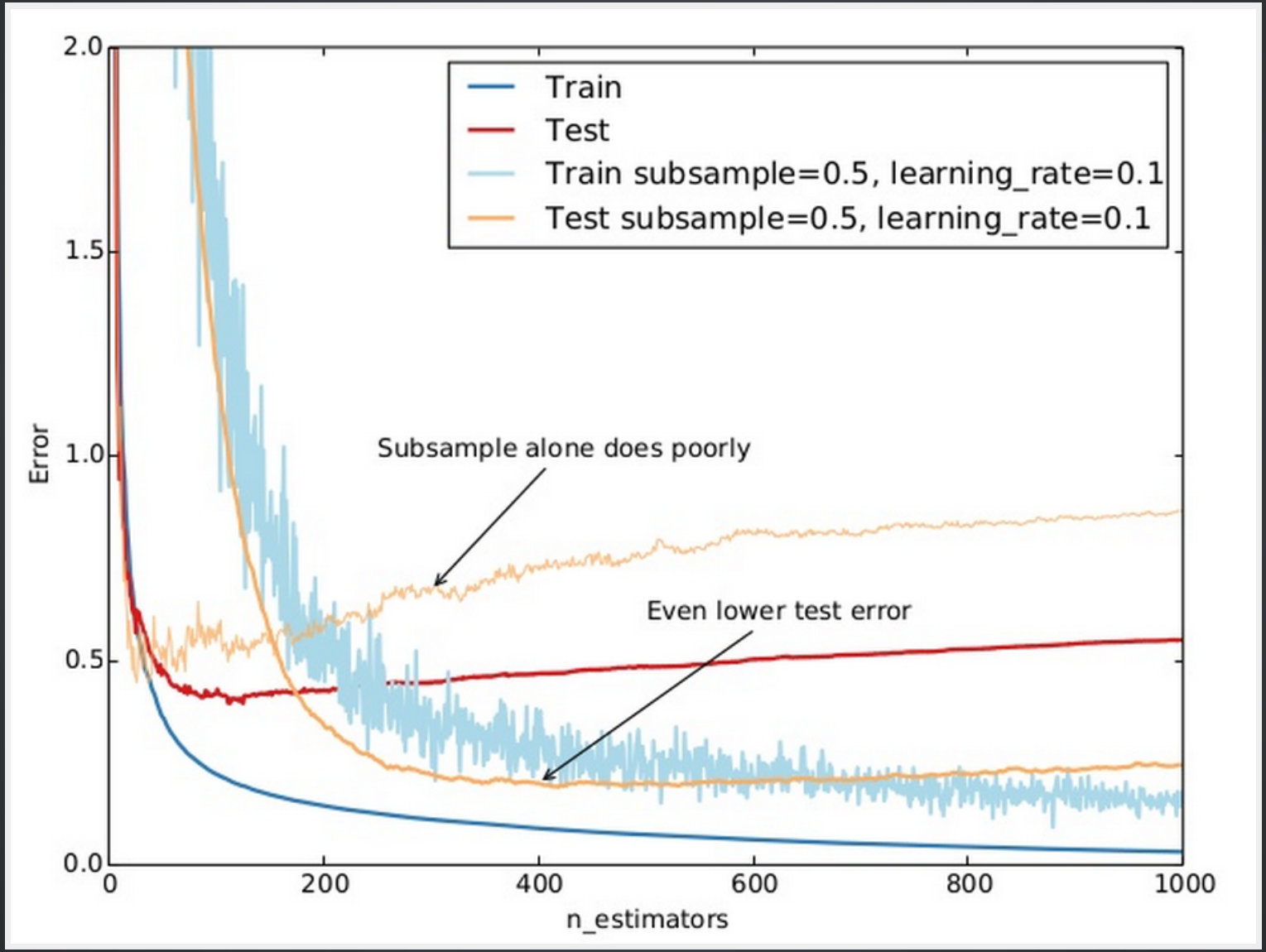
Antigradient: $\left[-\nabla Q(\tilde{y}^m, y)\right]_i = -\frac{\partial \mathcal{L}(M_i^m)}{\partial M_i^m} \cdot y_i$

Build regressor to reproduce

$$x_i \rightarrow \left[-\nabla Q(\tilde{y}^m, y)\right]_i$$

Predictions of base regressors are summed

SUBSAMPLING & SHRINKAGE



GRADIENT BOOSTING

- State-of-art results in many areas
- Can overfit
- GBDT: needs tuning to prevent overfitting
- Tuning: choose shrinkage (=learning rate) last
- Second-order methods of optimization can be applied with trees!

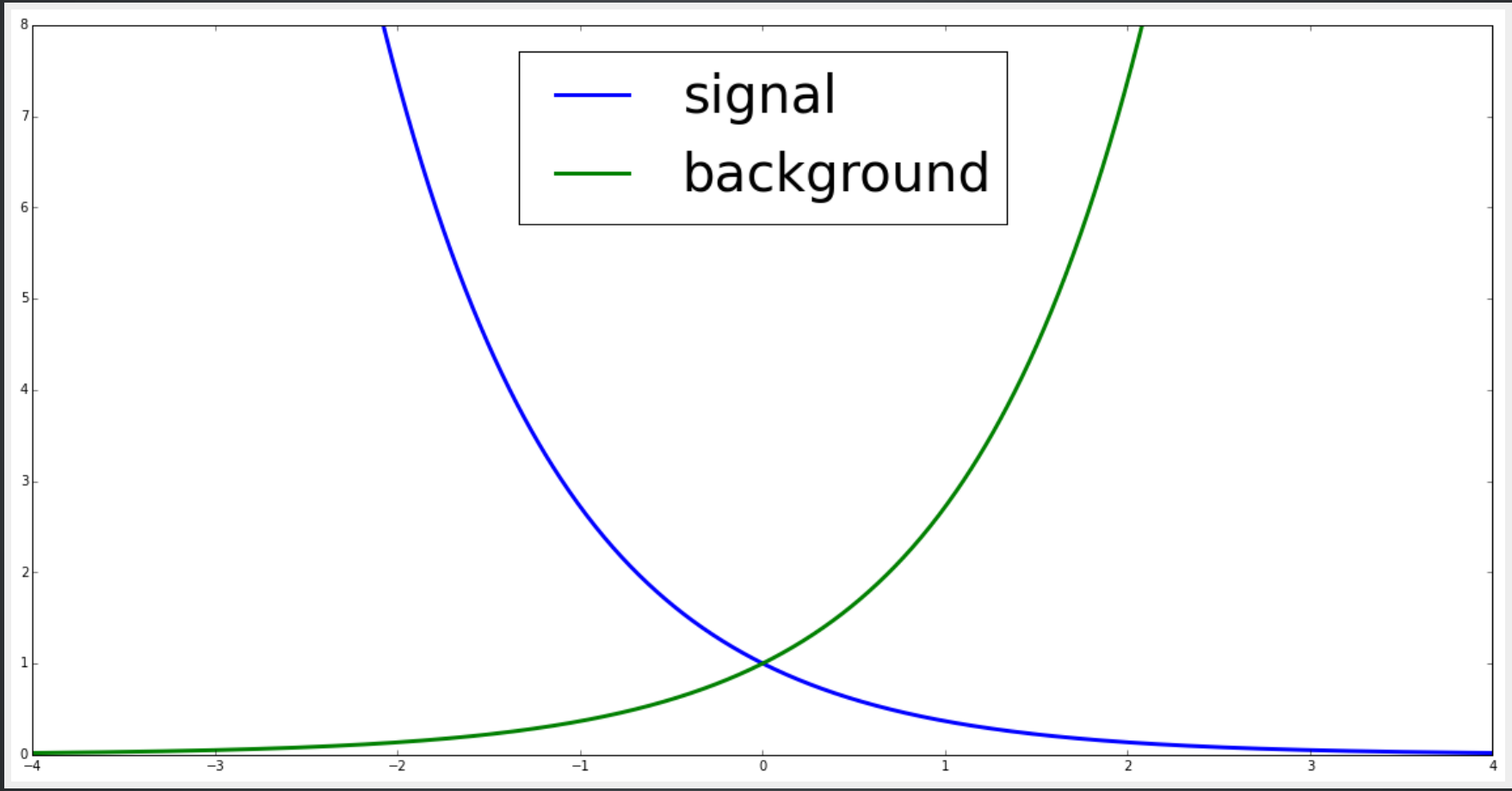
FEATURE IMPORTANCES

There are different approaches

- how many times feature was used
- gain of purity (Gini, Entropy)
- Common recipe (not only for tree-based classifiers): Shuffling column, looking at difference in quality.

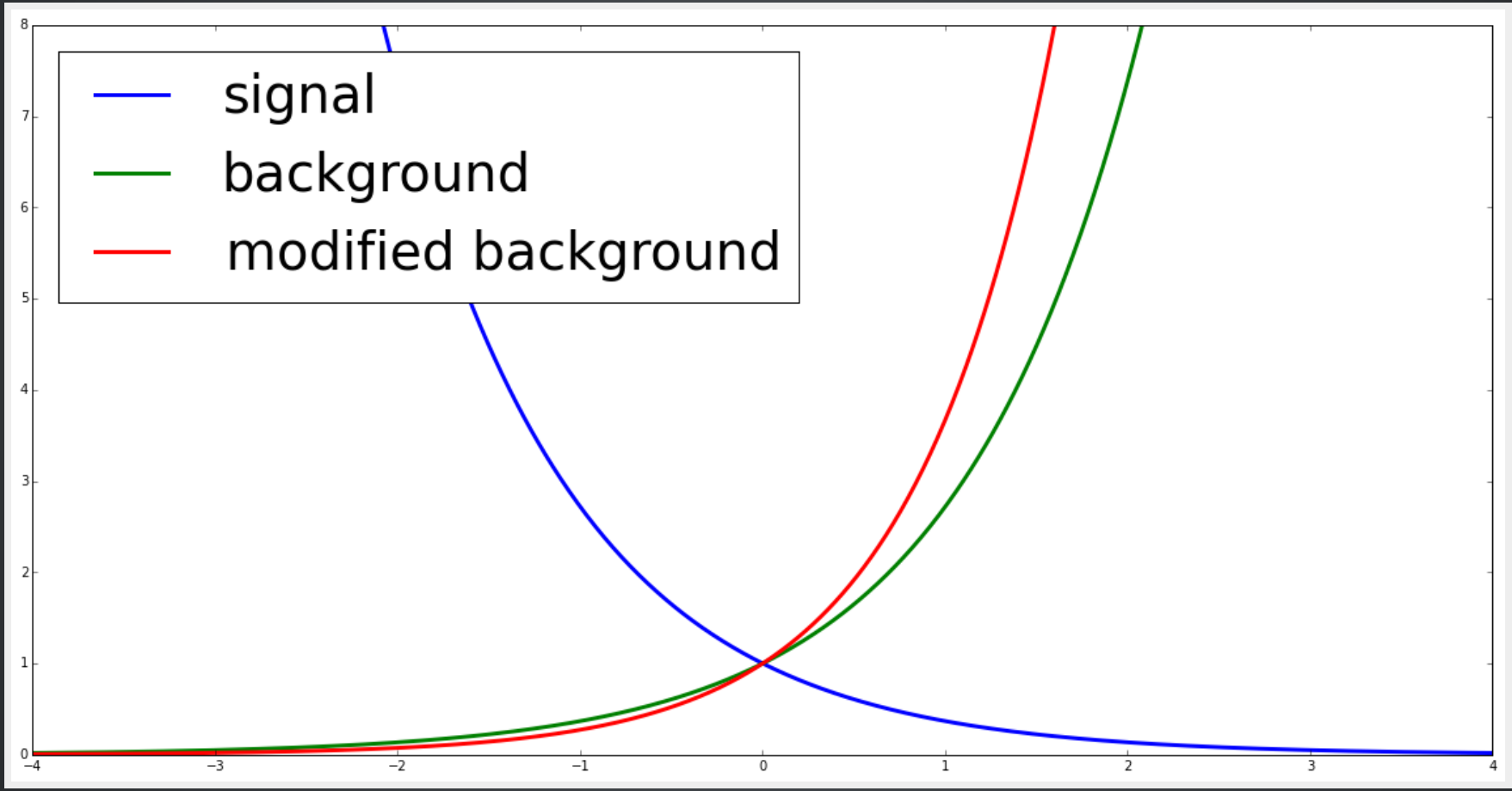
TRICK WITH EXP LOSS, CLEARING THE TOP

Usually we need the signal region to be clear



TRICK WITH EXP LOSS, CLEARING THE TOP

This can be reflected in loss function



LOSS FUNCTIONS

- Flexible tool
- Different variations: i.e. pairwise loss

$$Q = \sum_{i,j} \mathcal{L}_{ij}(\tilde{y}_i, \tilde{y}_j)$$


THE END