MACHINE LEARNING IN HIGH ENERGY PHYSICS

LECTURE #2

Alex Rogozhnikov, 2015

RECAPITULATION

- classification, regression
- kNN classifier and regressor
- ROC curve, ROC AUC

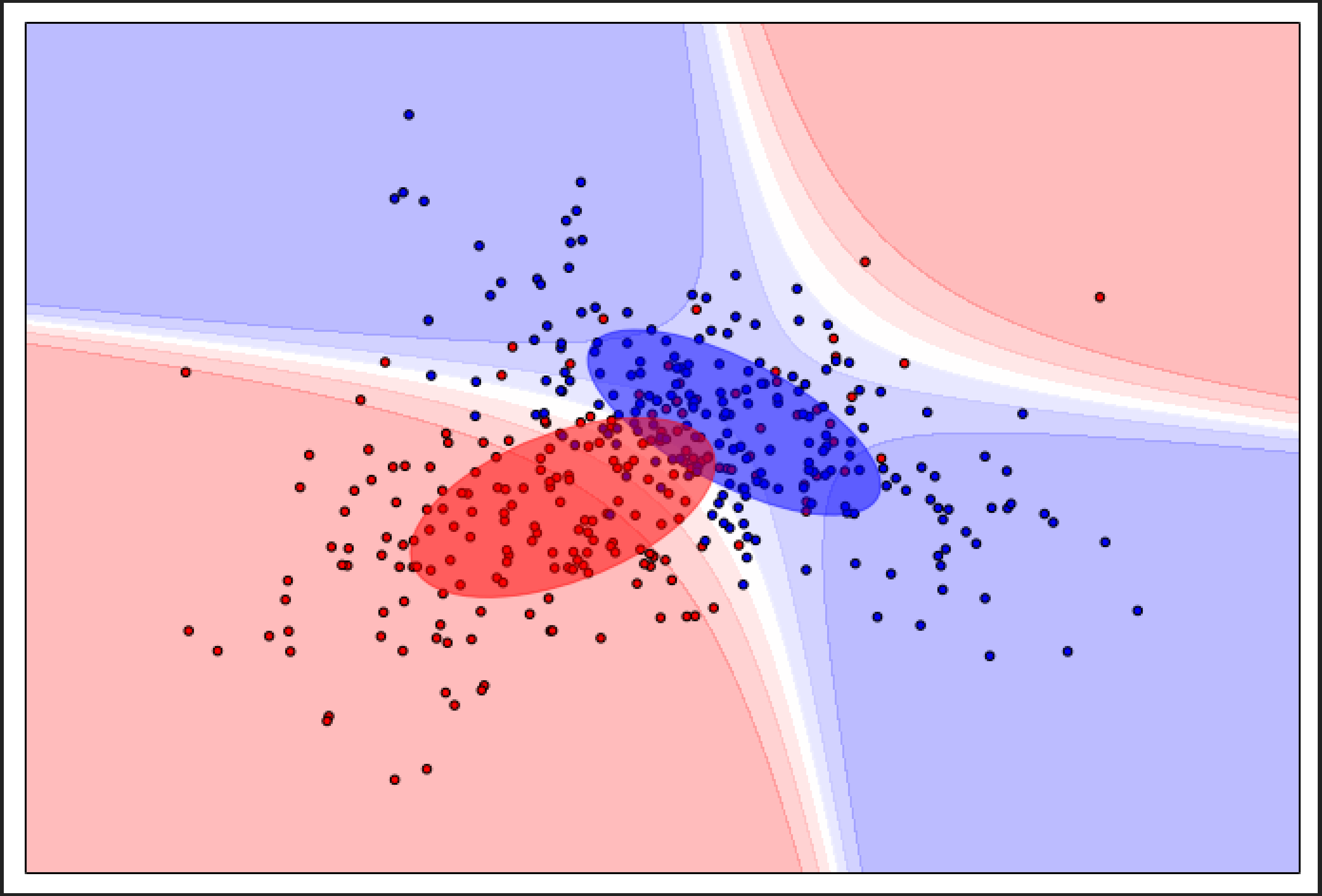
OPTIMAL BAYESIAN CLASSIFIER

Given knowledge about distributions, we can build optimal classifier

$$\frac{p(y=1\,|\,x)}{p(y=0\,|\,x)} = \frac{p(y=1)\,p(x\,|\,y=1)}{p(y=0)\,p(x\,|\,y=0)}$$

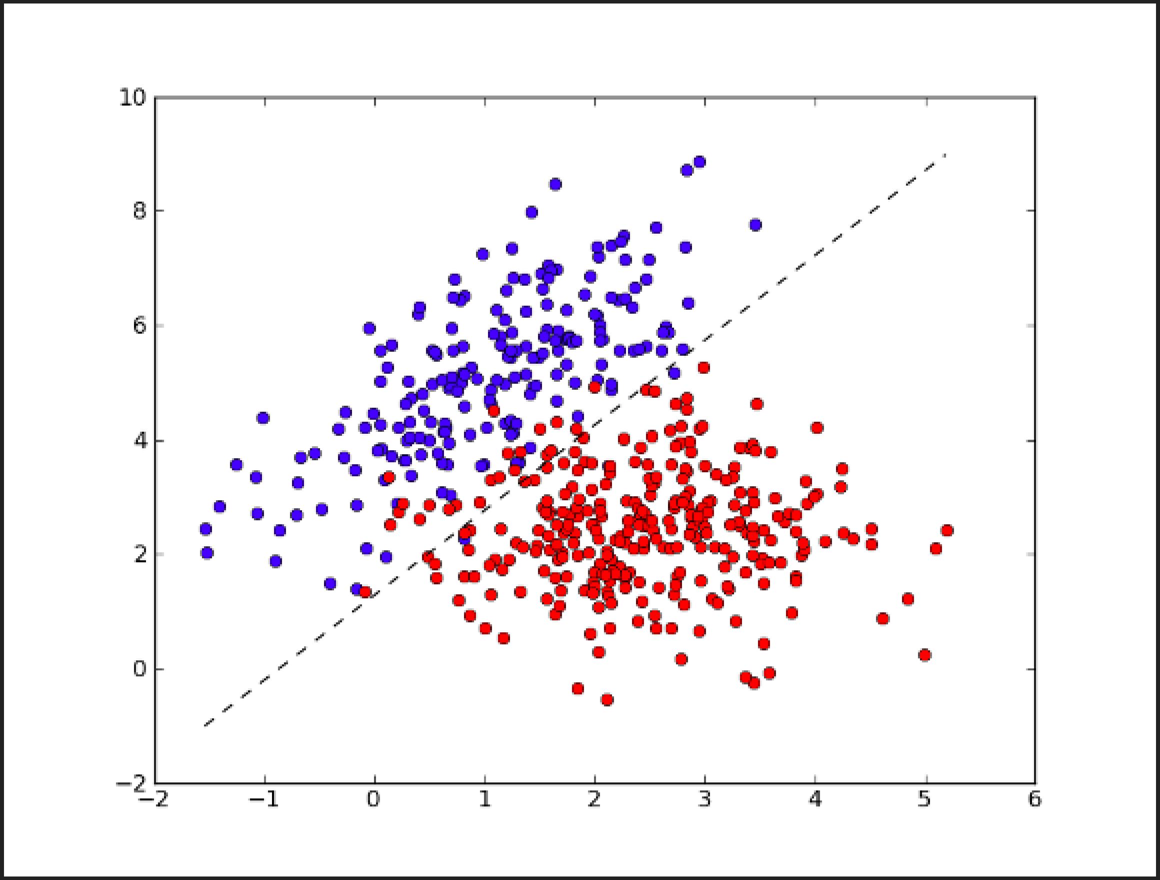
But distributions are complex, contain many parameters.

QDA



QDA follows generative approach.

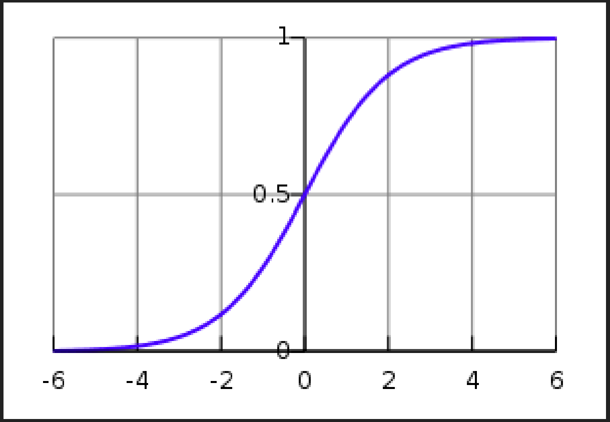
LOGISTIC REGRESSION



Decision function $d(x) =< w, x > +w_0$

Sharp rule: $\hat{y} = \operatorname{sgn} d(x)$

LOGISTIC REGRESSION



$$d(x) =< w, x > +w_0$$

Smooth rule:

$$\begin{aligned} p_{+1}(x) &= \sigma(d(x)) \\ p_{-1}(x) &= \sigma(-d(x)) \end{aligned}$$

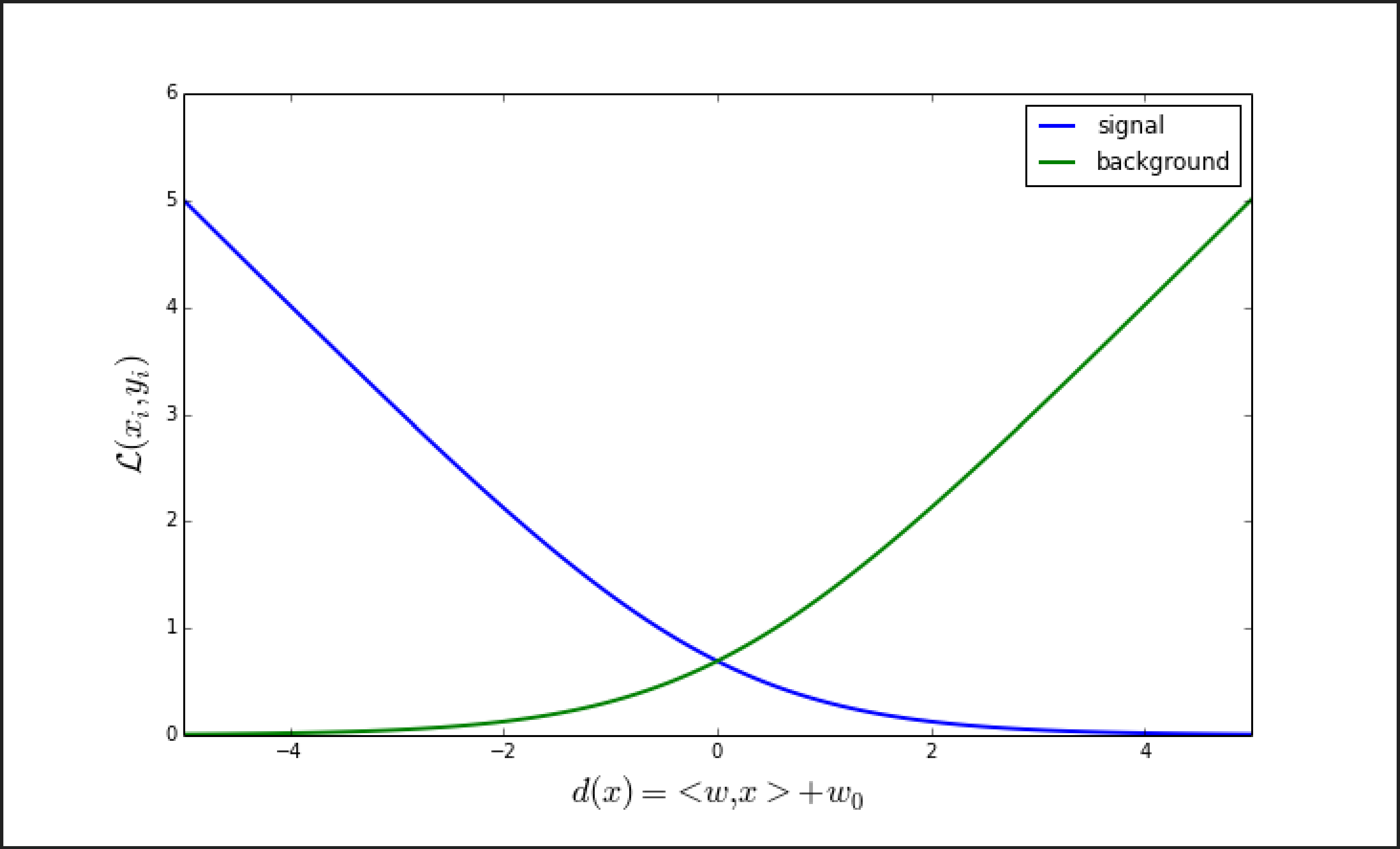
Optimizing weights w, w_0 to maximize log-likelihood

$$\mathcal{L} = \frac{1}{N} \sum_{i \in \text{events}} - \ln(p_{y_i}(x_i)) = \frac{1}{N} \sum_i L(x_i, y_i) \to \min$$

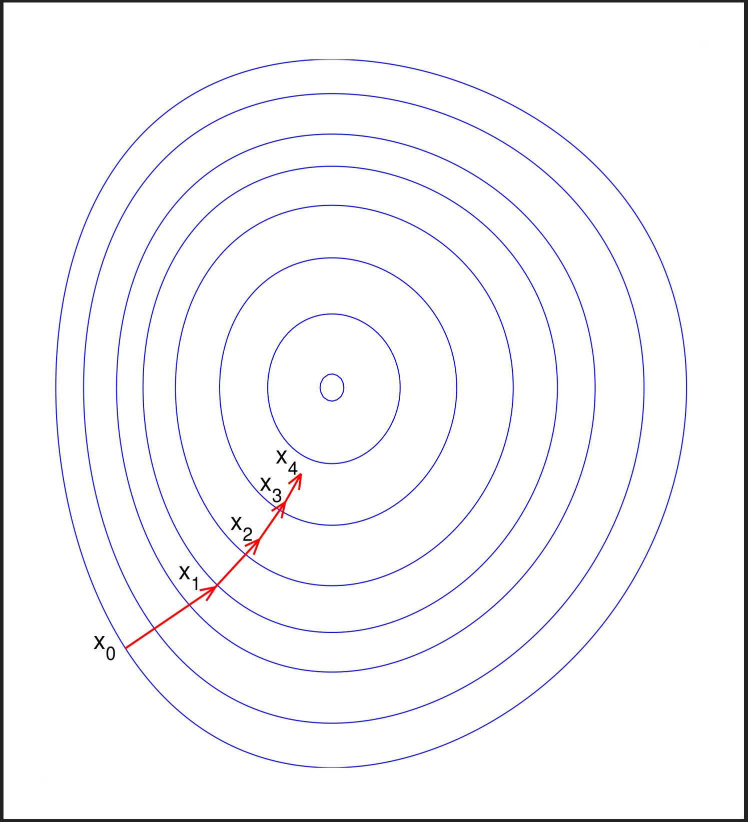
LOGISTIC LOSS

Loss penalty for single observation

$$L(x_i, y_i) = -\ln(p_{y_i}(x_i)) = \begin{cases} \ln(1 + e^{-d(x_i)}), & y_i = +1 \\ \ln(1 + e^{d(x_i)}), & y_i = -1 \end{cases}$$



GRADIENT DESCENT & STOCHASTIC OPTIMIZATION



Problem:
finding w to minimize $\mathcal{L}$

$$w \leftarrow w - \eta \frac{\partial \mathcal{L}}{\partial w}$$

η is step size
(also `shrinkage`, `learning rate`)

STOCHASTIC GRADIENT DESCENT

$$\mathcal{L} = \frac{1}{N} \sum_i L(x_i, y_i) \to \min$$

On each iteration make a step with respect to only one event:

1. take i — random event from training data
2. $w \leftarrow w - \eta \frac{\partial \mathcal{L}(x_i, y_i)}{\partial w}$

Each iteration is done much faster, but training process is less stable.

POLYNOMIAL DECISION RULE

$$d(x) = w_0 + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j$$

is again linear model, introduce new features:

$$z = \{1\} \cup \{x_i\}_i \cup \{x_i x_j\}_{ij}$$

$$d(x) = \sum_i w_i z_i$$

and reusing logistic regression.

We can add $x_0 = 1$ as one more variable to dataset and forget about intercept

$d(x) = w_0 + \sum_{i=1}^N w_i x_i = \sum_{i=0}^N w_i x_i$

PROJECTING IN HIGHER DIMENSION SPACE

After adding new features, classes may become separable.

KERNEL TRICK

P is projection operator (which adds new features).

$$d(x) = < w, P(x) >$$

Assume

$$w = \sum_i \alpha_i P(x_i)$$

and look for optimal α_i

$$d(x) = \sum_i \alpha_i < P(x_i), P(x) >= \sum_i \alpha_i K(x_i, x)$$

We need only kernel: $K(x, y) =< P(x), P(y) >$

KERNEL TRICK

Popular kernel is gaussian Radial Basis Function:

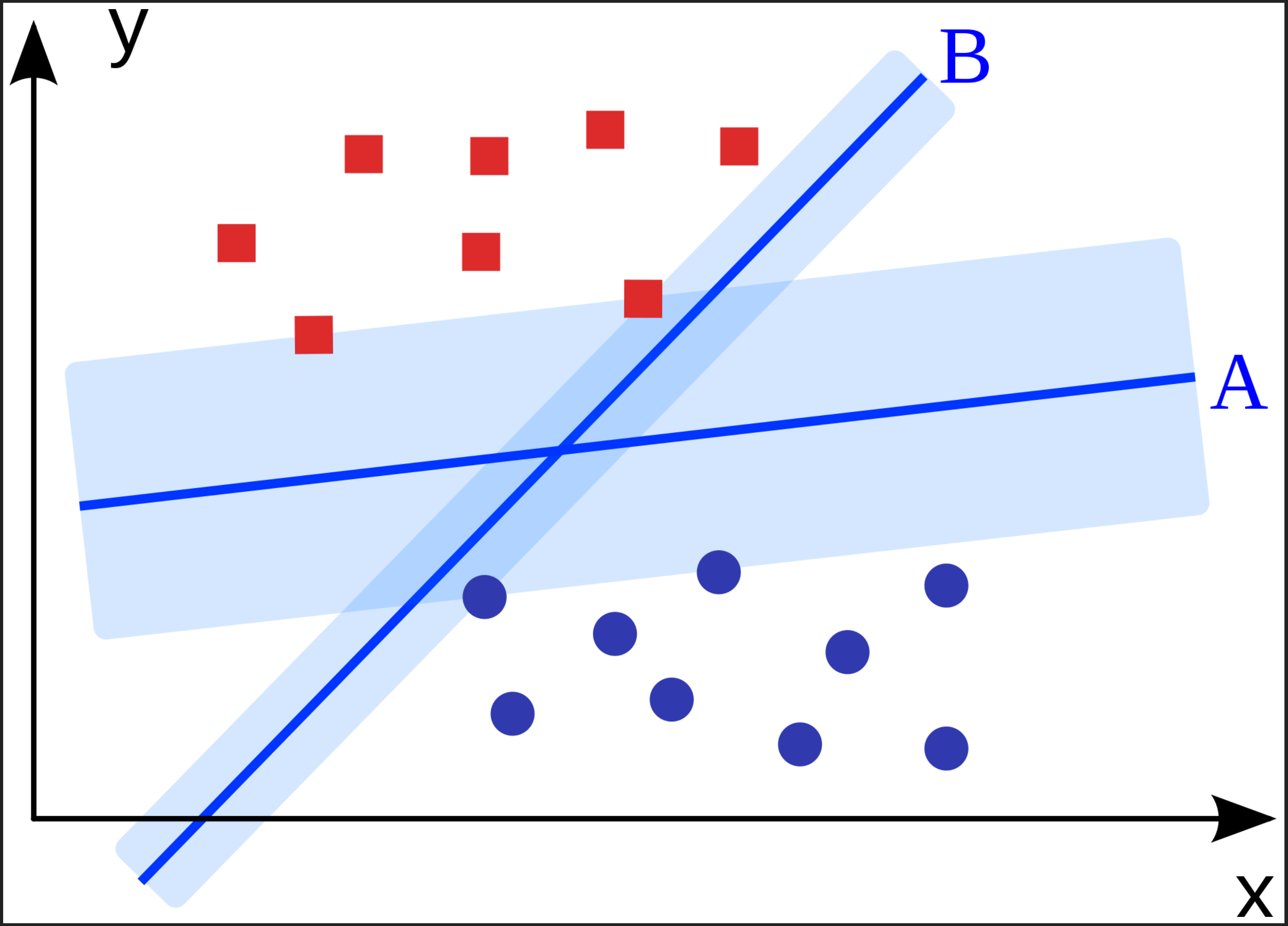
$$K(x, y) = e^{-c||x-y||^2}$$

Corresponds to projection to Hilbert space.

Exercise: find a correspong projection.

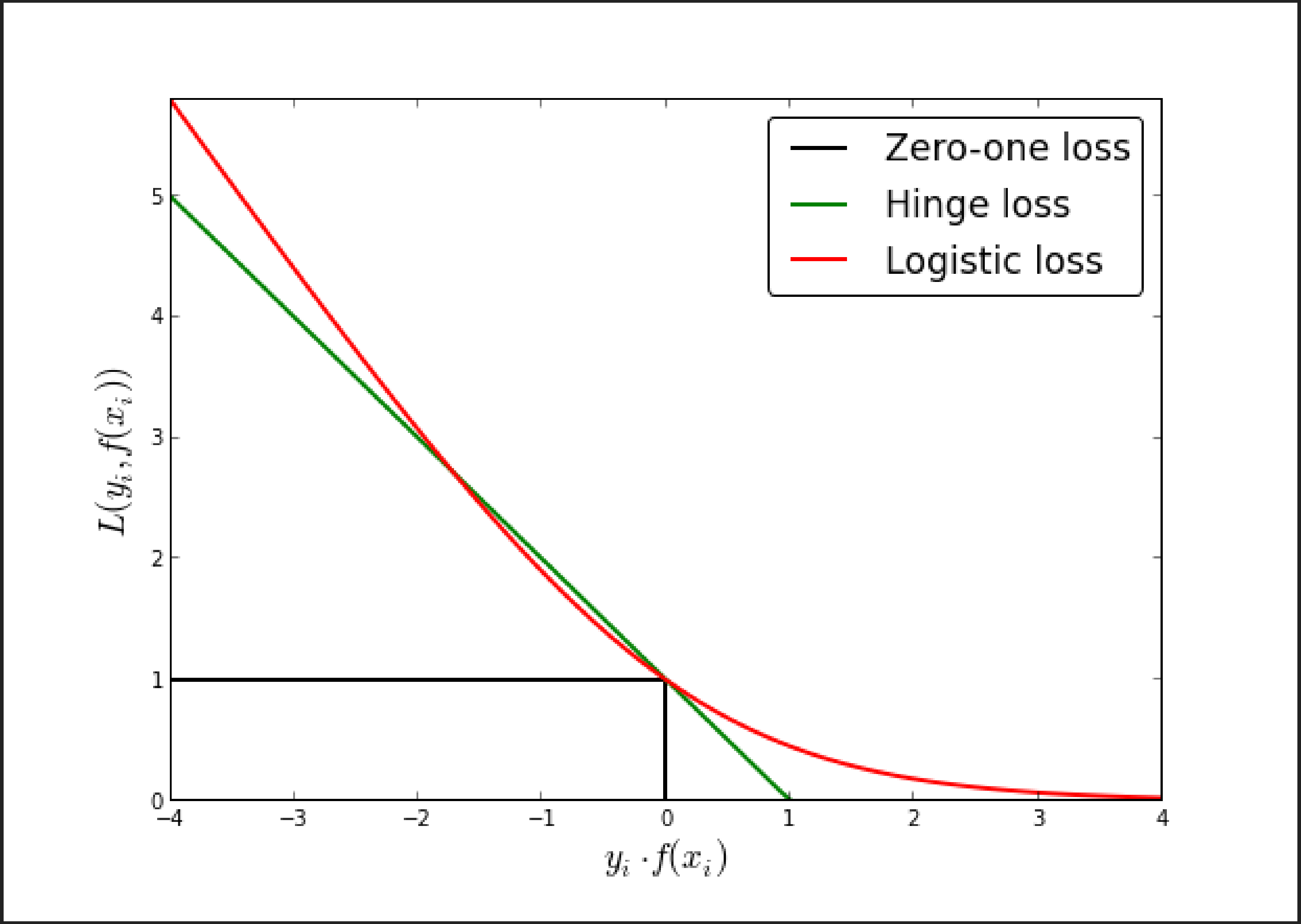
SUPPORT VECTOR MACHINE

SVM selects decision rule with maximal possible margin.

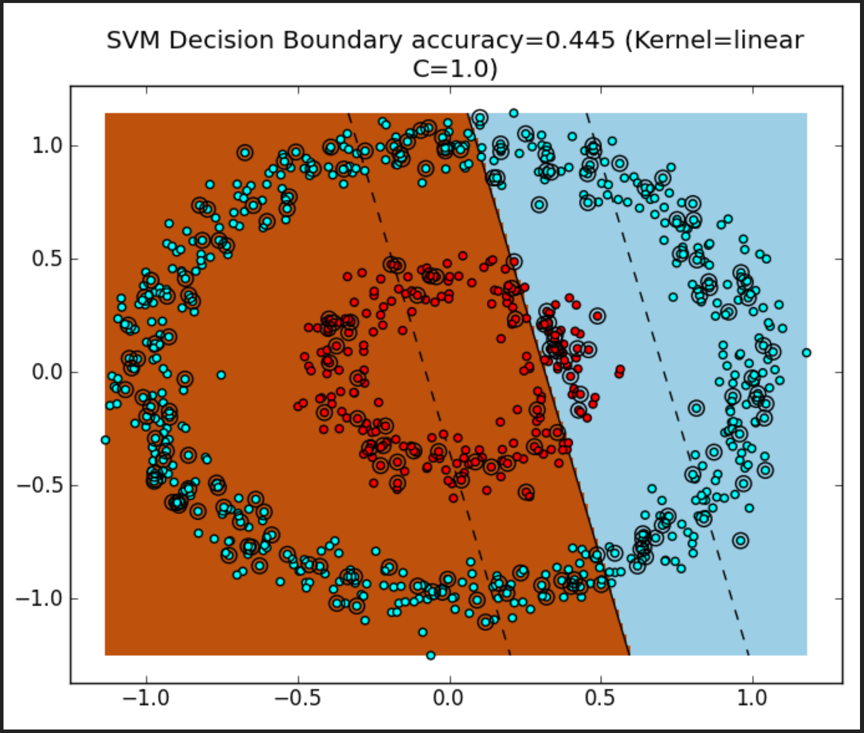


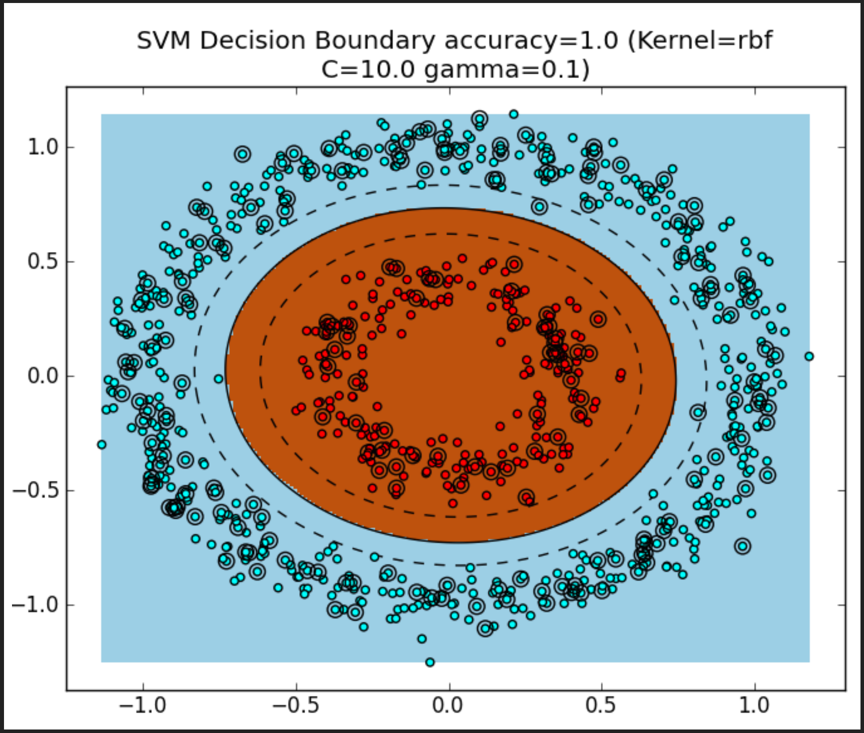
HINGE LOSS FUNCTION

SVM uses different loss function (only signal losses compared):

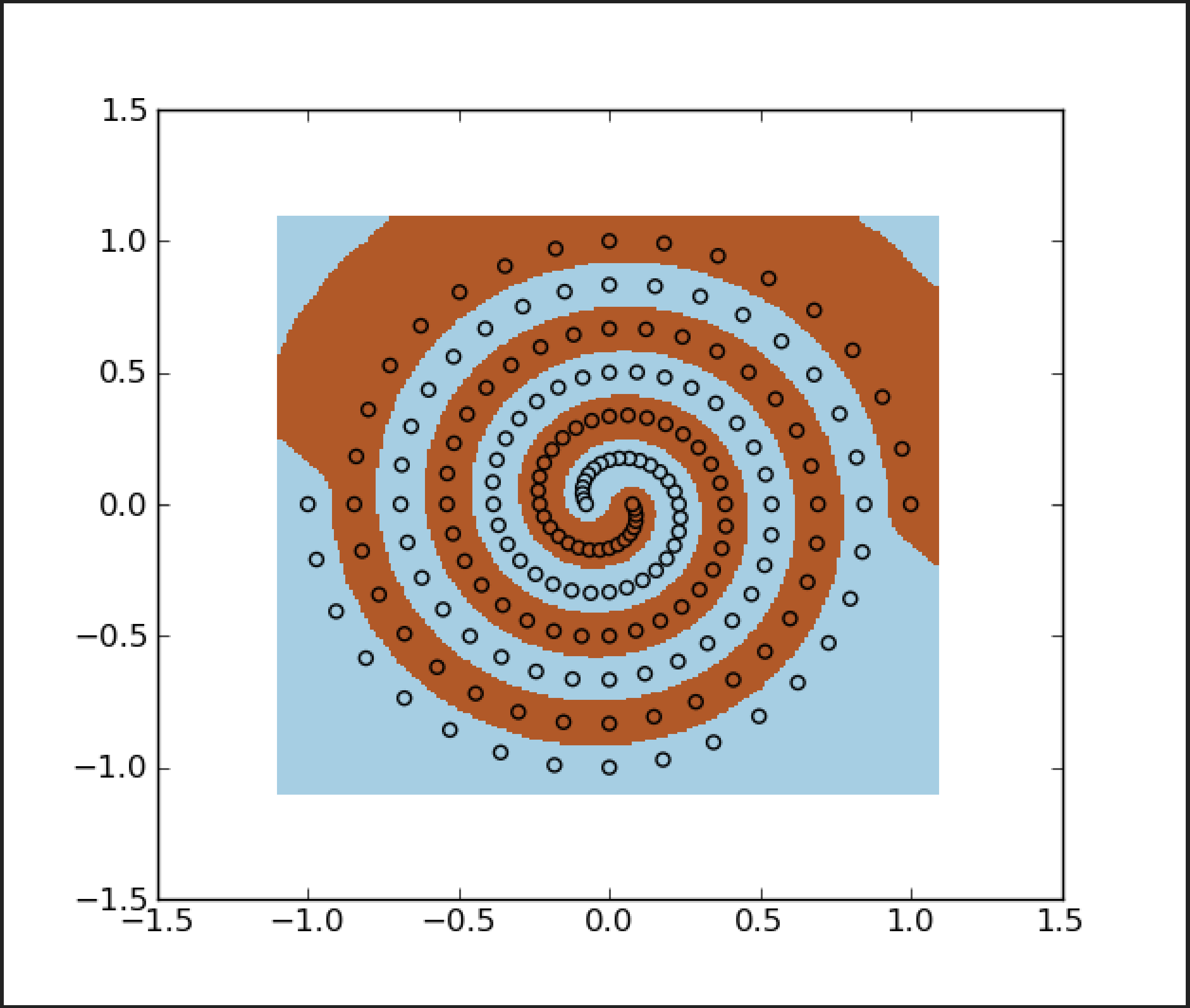


SVM + RBF KERNEL

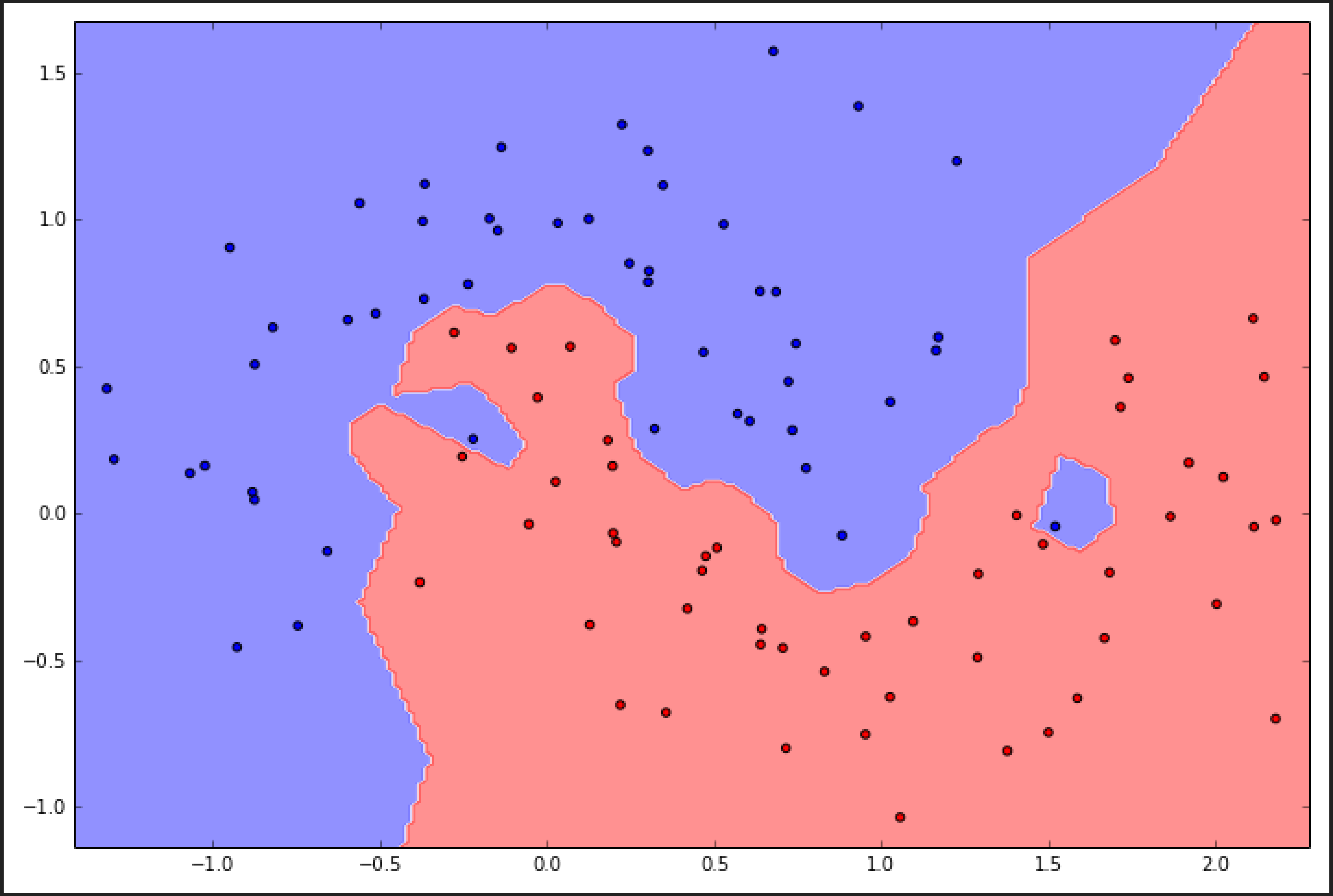




SVM + RBF KERNEL

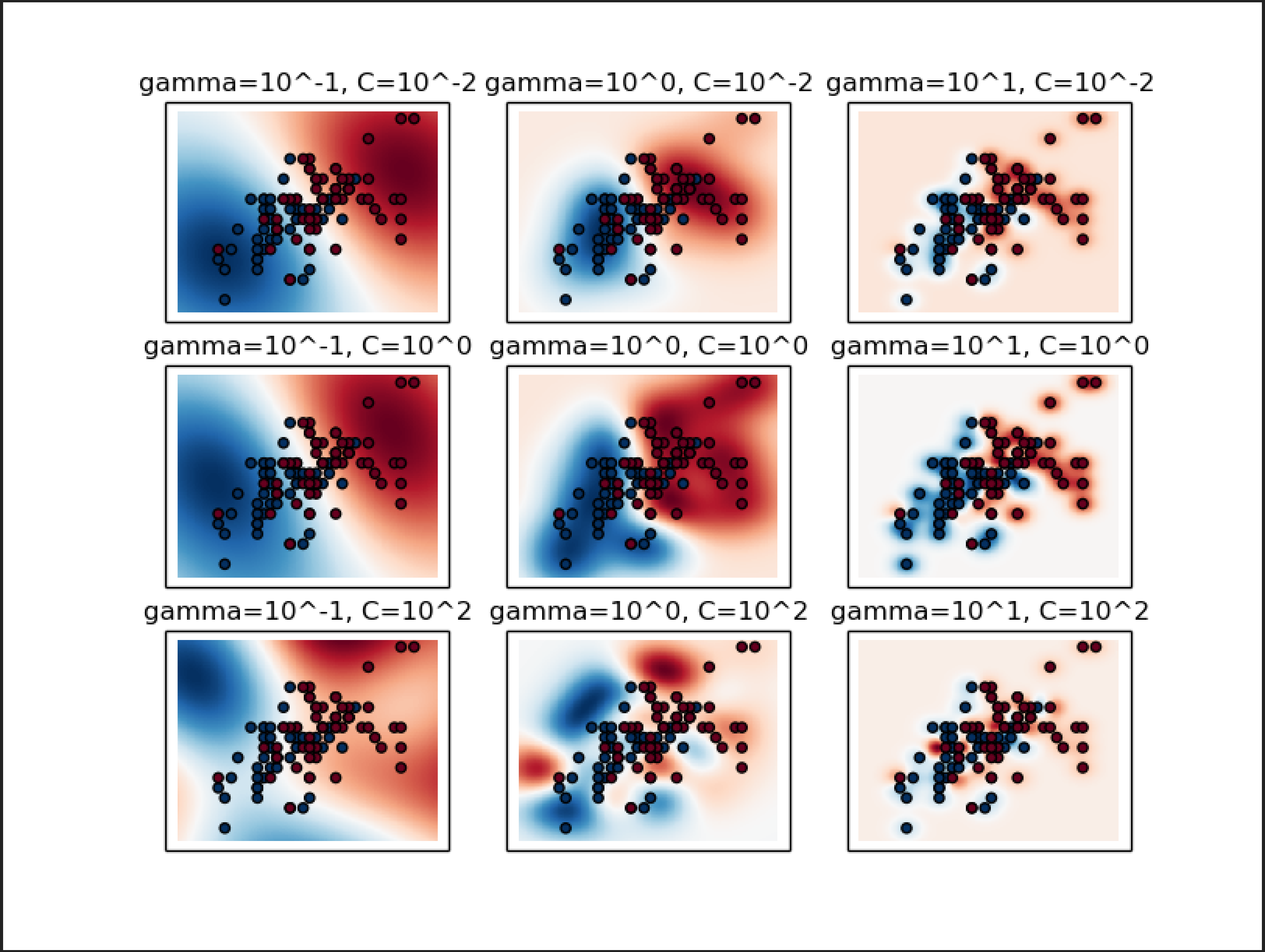


OVERFITTING



Knn with k=1 gives ideal classification of training data.

OVERFITTING



There are two definitions of overfitting, which often coincide.

DIFFERENCE-OVERFITTING

There is significant difference in quality of predictions between train and test.

COMPLEXITY-OVERFITTING

Formula has too high complexity (e.g. too many parameters), increasing the number of parameters drives to lower quality.

MEASURING QUALITY

To get unbiased estimate, one should test formula on independent samples (and be sure that no train information was given to algorithm during training)

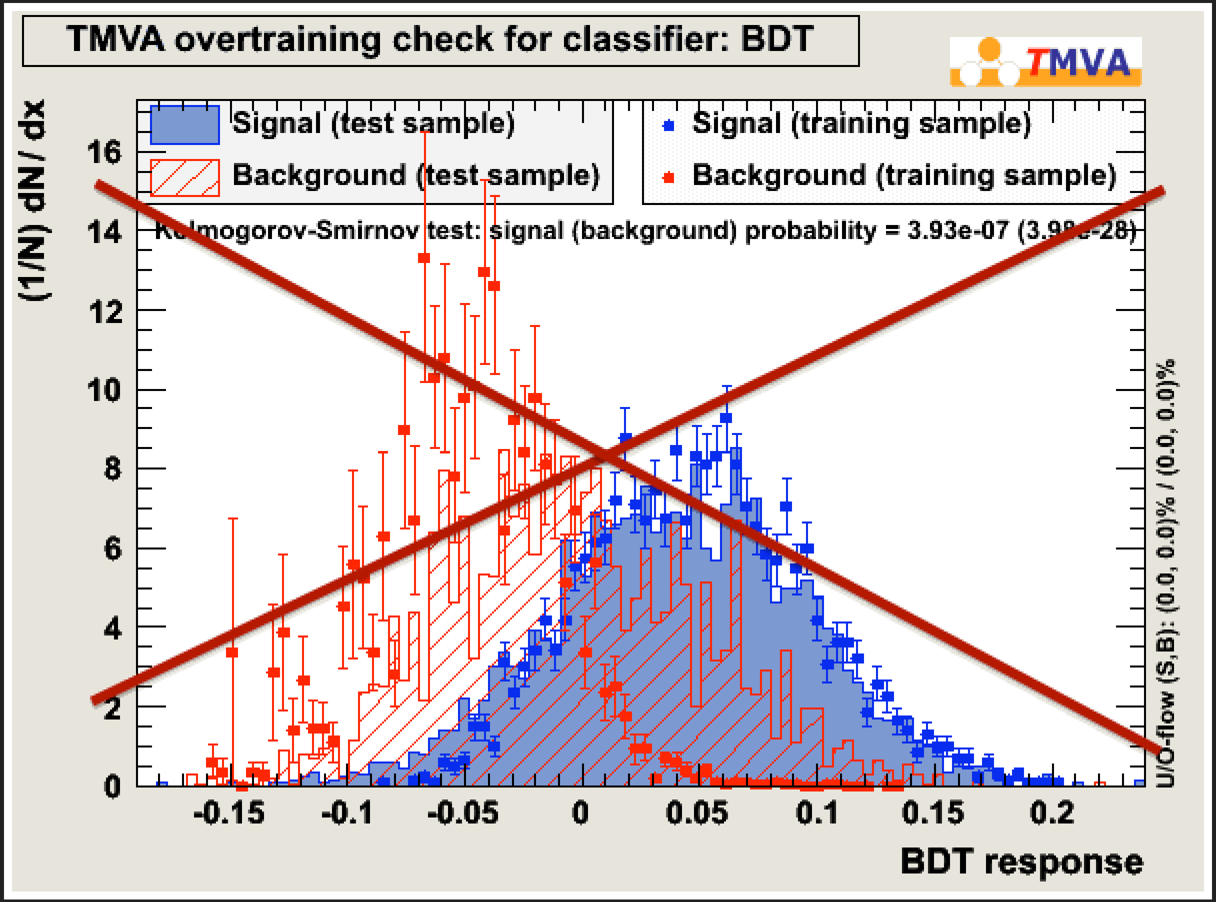
In most cases, simply splitting data into train and holdout is enough.

More approaches in seminar.

Difference-overfitting is inessential, provided that we measure quality on holdout (though easy to check).

Complexity-overfitting is problem — we need to test different parameters for optimality (more examples through the course).

Don't use distribution comparison to detect overfitting


TMVA overtraining check for classifier: BDT
TMVA
Signal (test sample)
Background (test sample)
Signal (training sample)
Background (training sample)
Kolmogorov-Smirnov test: signal (background) probability = 3.93e-07 (3.98e-28)
(1/N) dN/ dx
BDT response
U/O-flow (S,B): (0.0, 0.0)% / (0.0, 0.0)%

REGULARIZATION

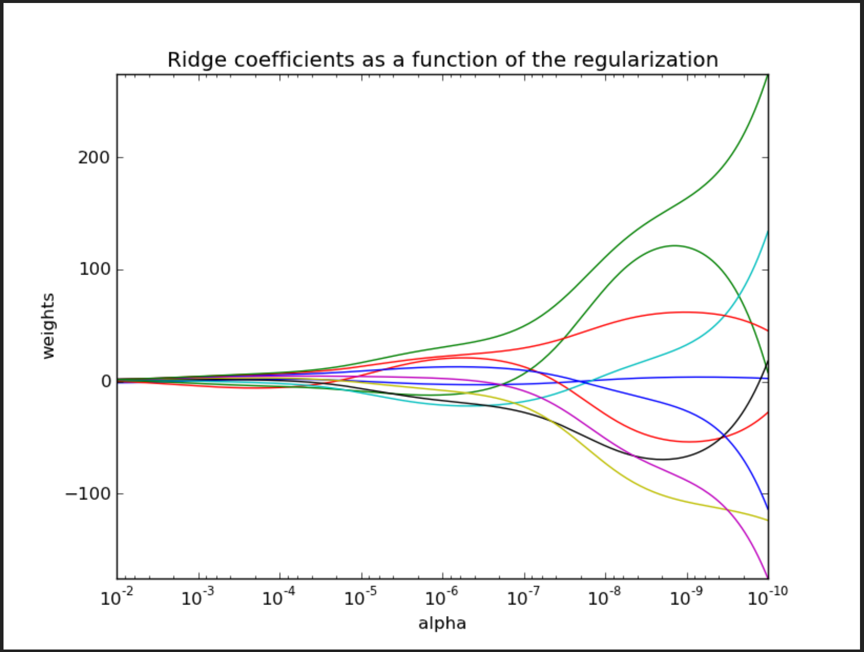
When number of weights is high, overfitting is very probable

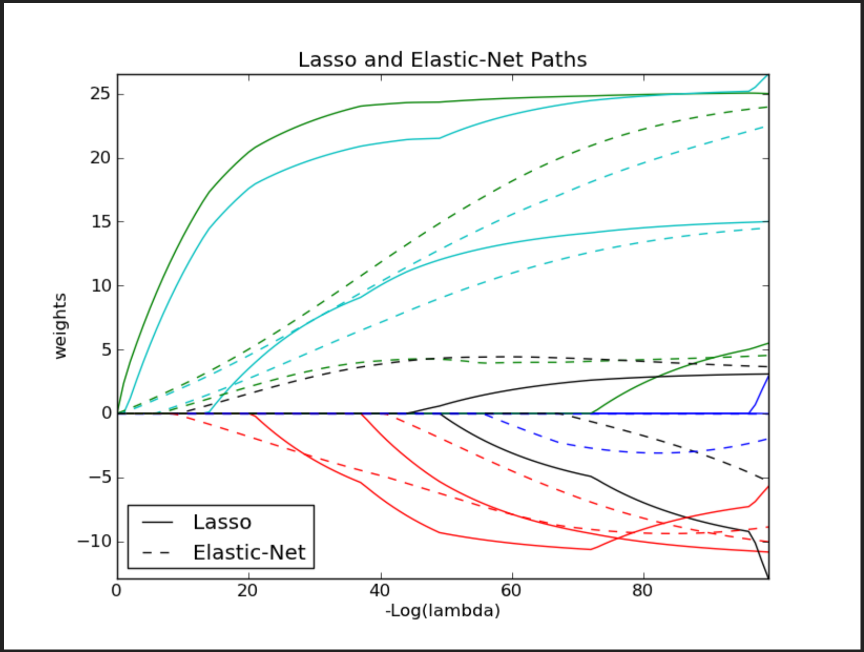
Adding regularization term to loss function:

$$\mathcal{L} = \frac{1}{N} \sum_i L(x_i, y_i) + \mathcal{L}_{\text{reg}} \rightarrow \min$$

- L_2 regularization : $\mathcal{L}_{\text{reg}} = \alpha \sum_j |w_j|^2$
- L_1 regularization: $\mathcal{L}_{\text{reg}} = \beta \sum_j |w_j|$
- $L_1 + L_2$ regularization: $\mathcal{L}_{\text{reg}} = \alpha \sum_j |w_j|^2 + \beta \sum_j |w_j|$

L_2, L_1 — REGULARIZATIONS



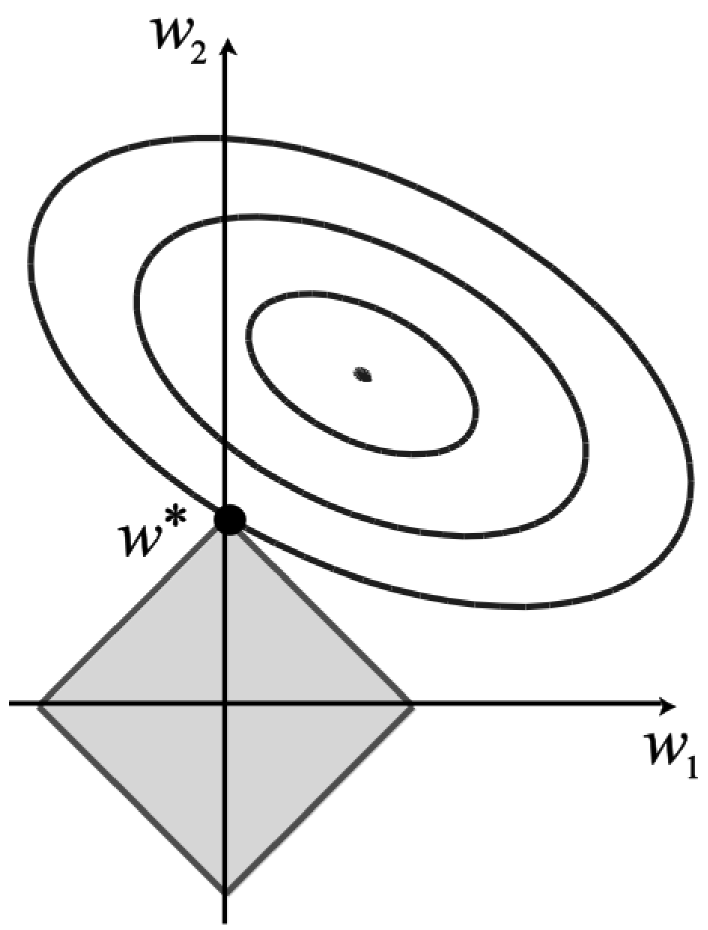


L2 regularization

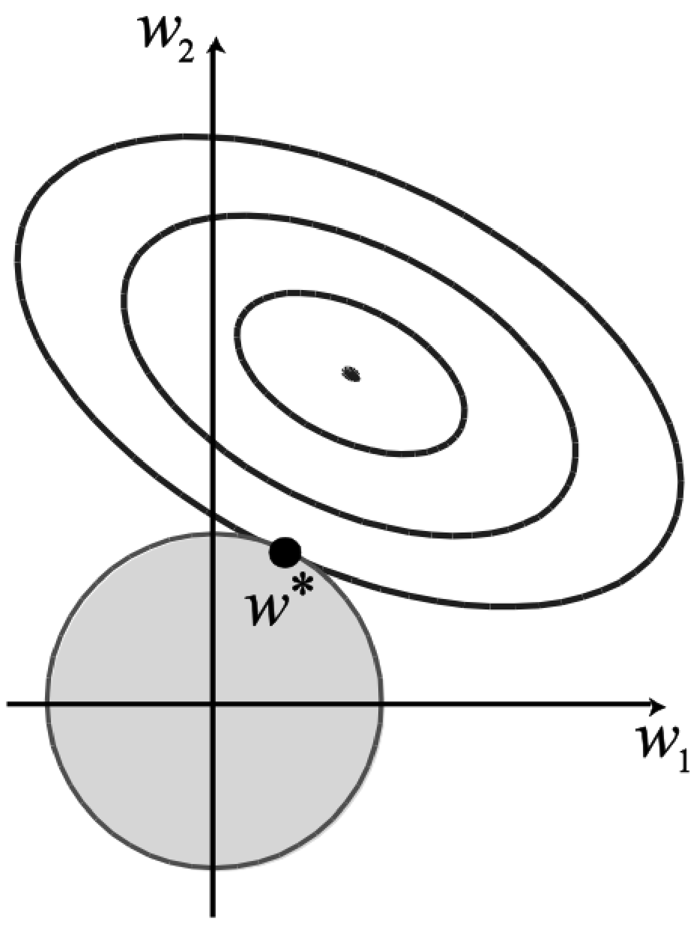
L1 (solid), L1 + L2 (dashed)

REGULARIZATIONS

L_1 regularization encourages sparsity

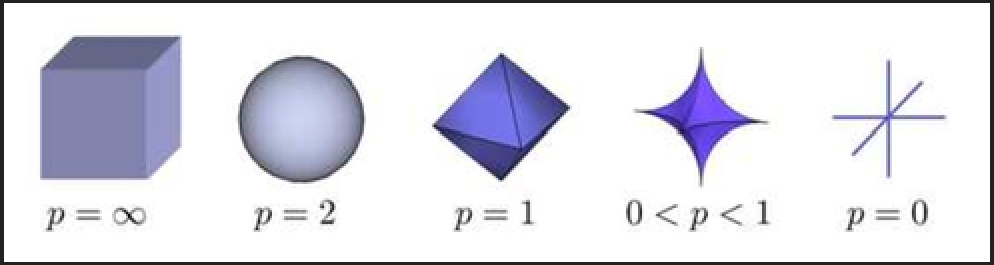


L1



L2

$\mathbf{L_p}$ REGULARIZATIONS



$$L_p = \sum_i w_i^p$$

- What is the expression for L_0?
- $L_0 = \sum_i [w_i \neq 0]$
  But nobody uses it, even $L_p,\ 0 < p < 1$. Why?
- Because it is not convex

LOGISTIC REGRESSION

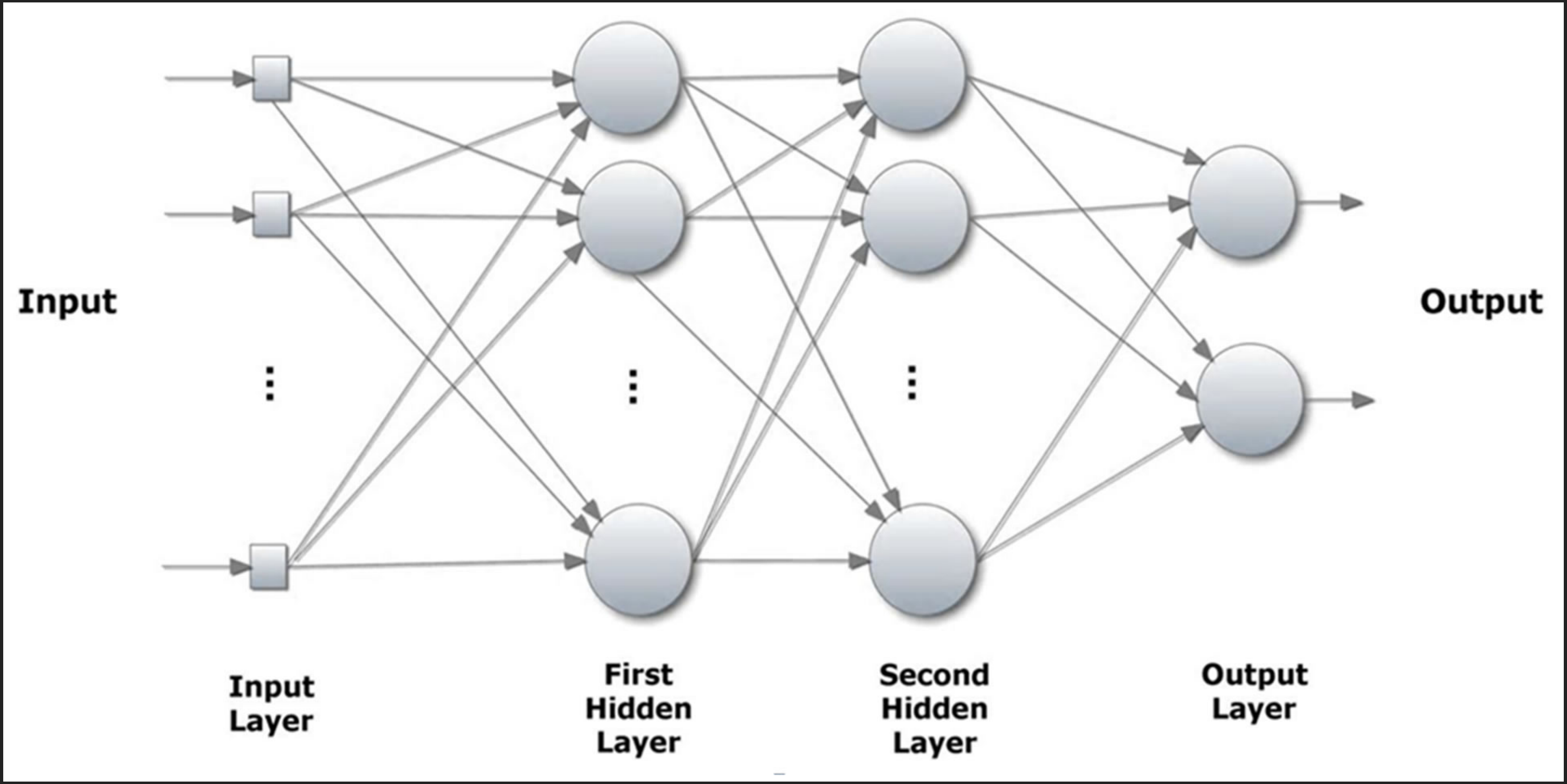
- classifier based on linear decision rule
- training is reduced to convex optimization
- other decision rules are achieved by adding new features
- stochastic optimization is used
- can handle > 1000 features, requires regularization
- no iteraction between features

[ARTIFICIAL] NEURAL NETWORKS

Based on our understanding of natural neural networks

- neurons are organized in networks
- receptors activate some neurons, neurons are activating other neurons, etc.
- connection is via synapses

STRUCTURE OF ARTIFICIAL FEED-FORWARD NETWORK



ACTIVATION OF NEURON

Neuron states: $n = \begin{cases} 1, & \text{activated} \\ 0, & \text{not activated} \end{cases}$

Let n_i to be state of w_i to be weight of connection between i-th neuron and output neuron:

$$n = \begin{cases} 1, & \sum_i w_i n_i > 0 \\ 0, & \sum_i otherwise \end{cases}$$

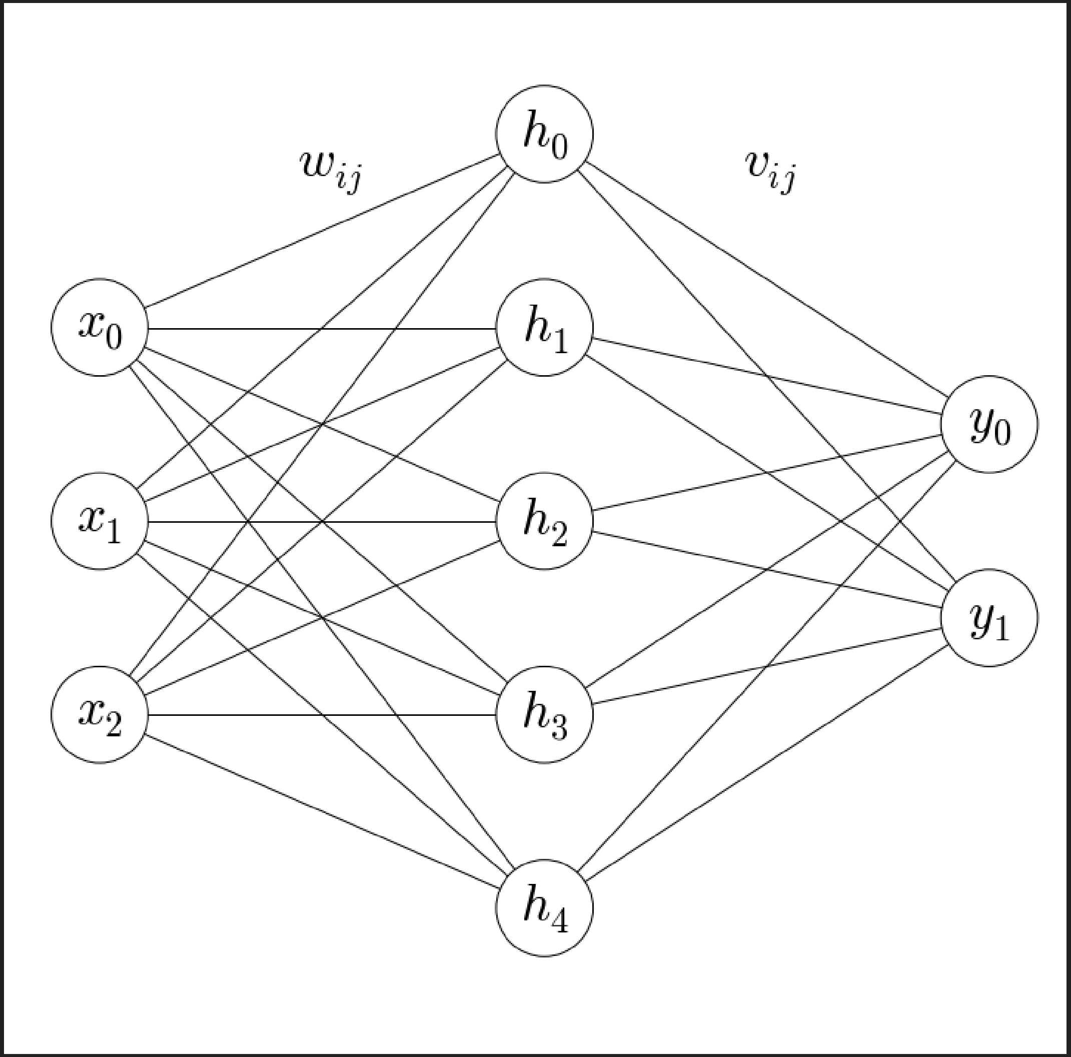
Problem: find set of weights, that minimizes error on train dataset. (discrete optimization)

SMOOTH ACTIVATIONS:

ONE HIDDEN LAYER

$$h_i = \sigma(\sum_j w_{ij} x_j)$$

$$y_i = \sigma(\sum_i v_{ij} h_j)$$



VISUALIZATION OF NN

NEURAL NETWORKS

- Powerful general purpose algorithm for classification and regression
- Non-interpretable formula
- Optimization problem is non-convex with local optimums and has many parameters
  Stochastic optimization speeds up process and helps not to be caught in local minimum.
- Overfitting due to large amount of parameters
  L_1, L_2 — regularizations (and other tricks)

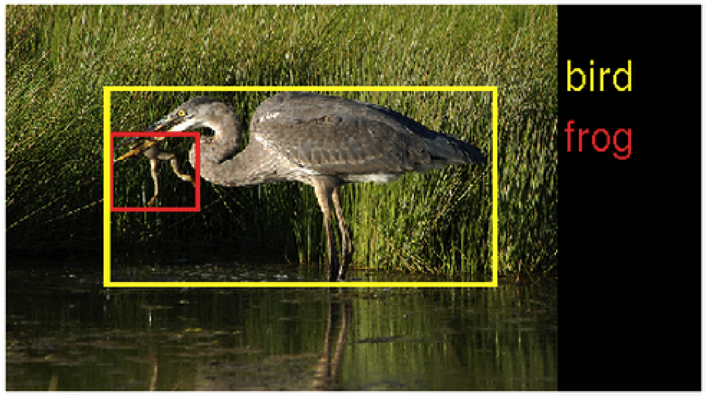
x MINUTES BREAK

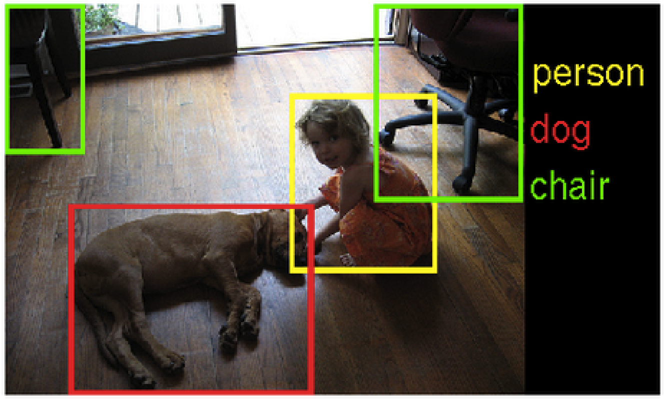
DEEP LEARNING

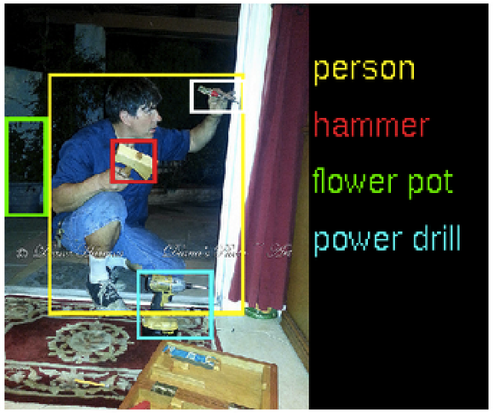
Gradient diminishes as number of hidden layers grows.
Usually 1-2 hidden layers are used.

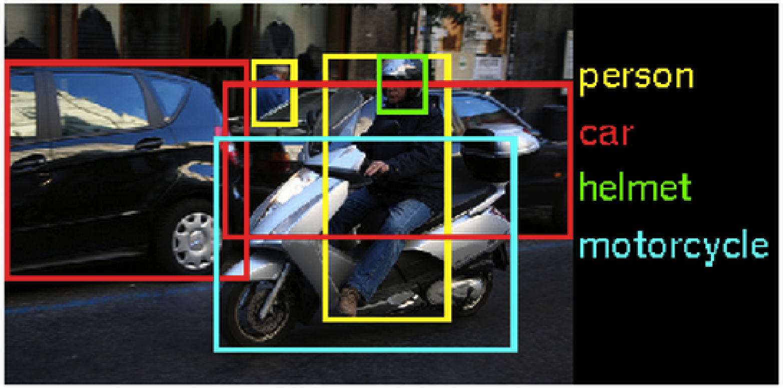
But modern ANN for image recognition have 7-15 layers.

Example ILSVRC2014 images:

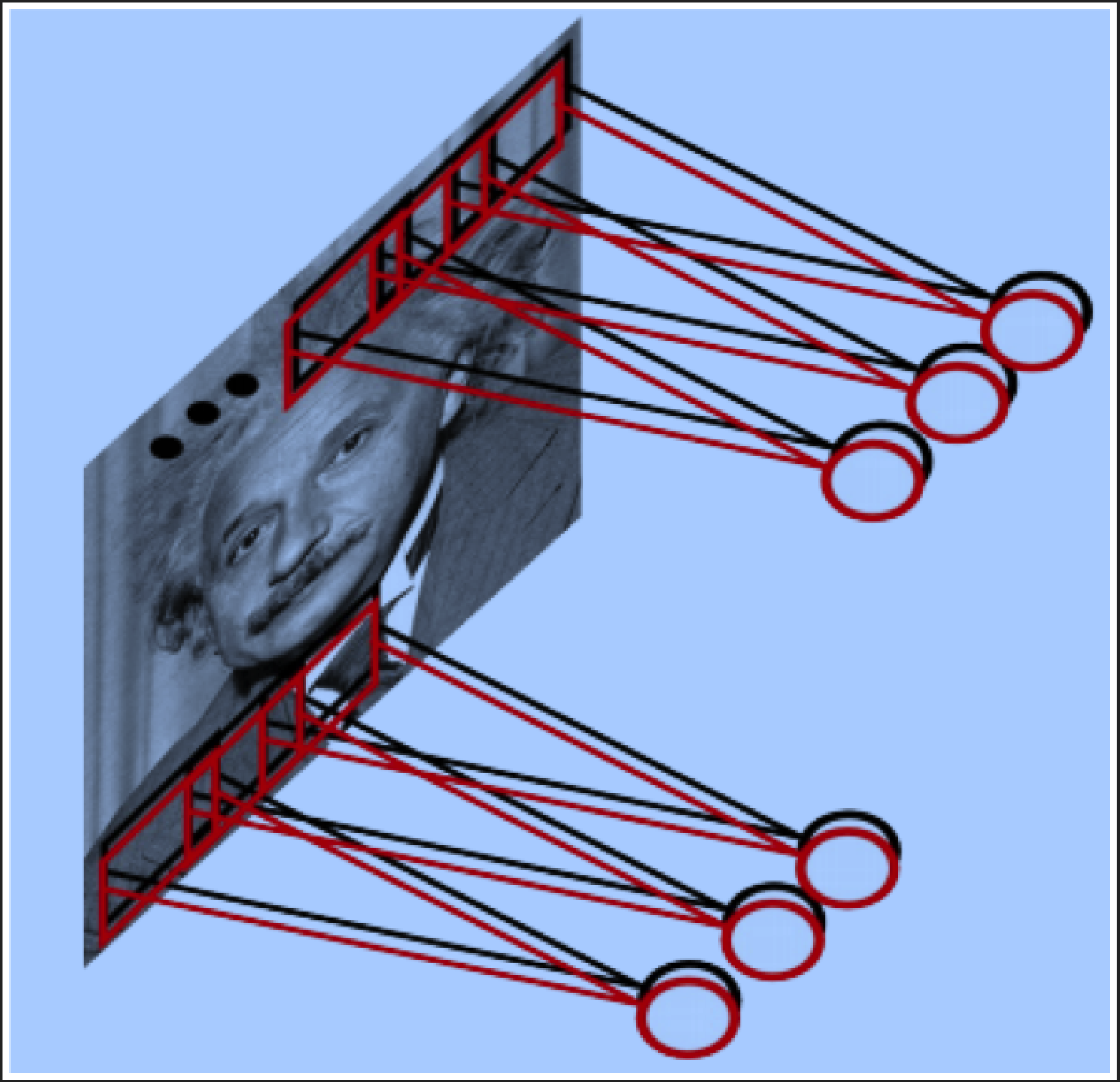




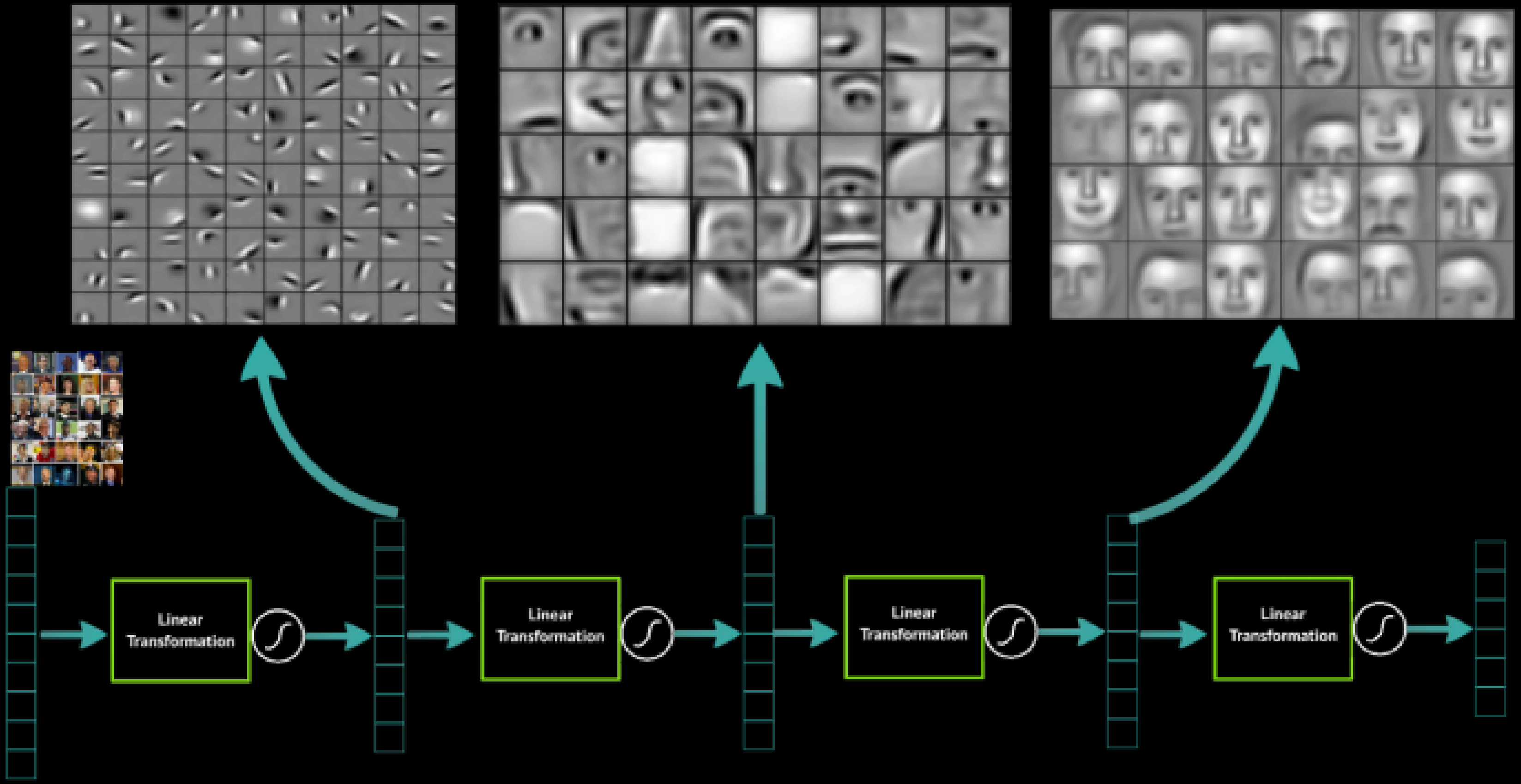




CONVOLUTIONAL NEURAL NETWORK

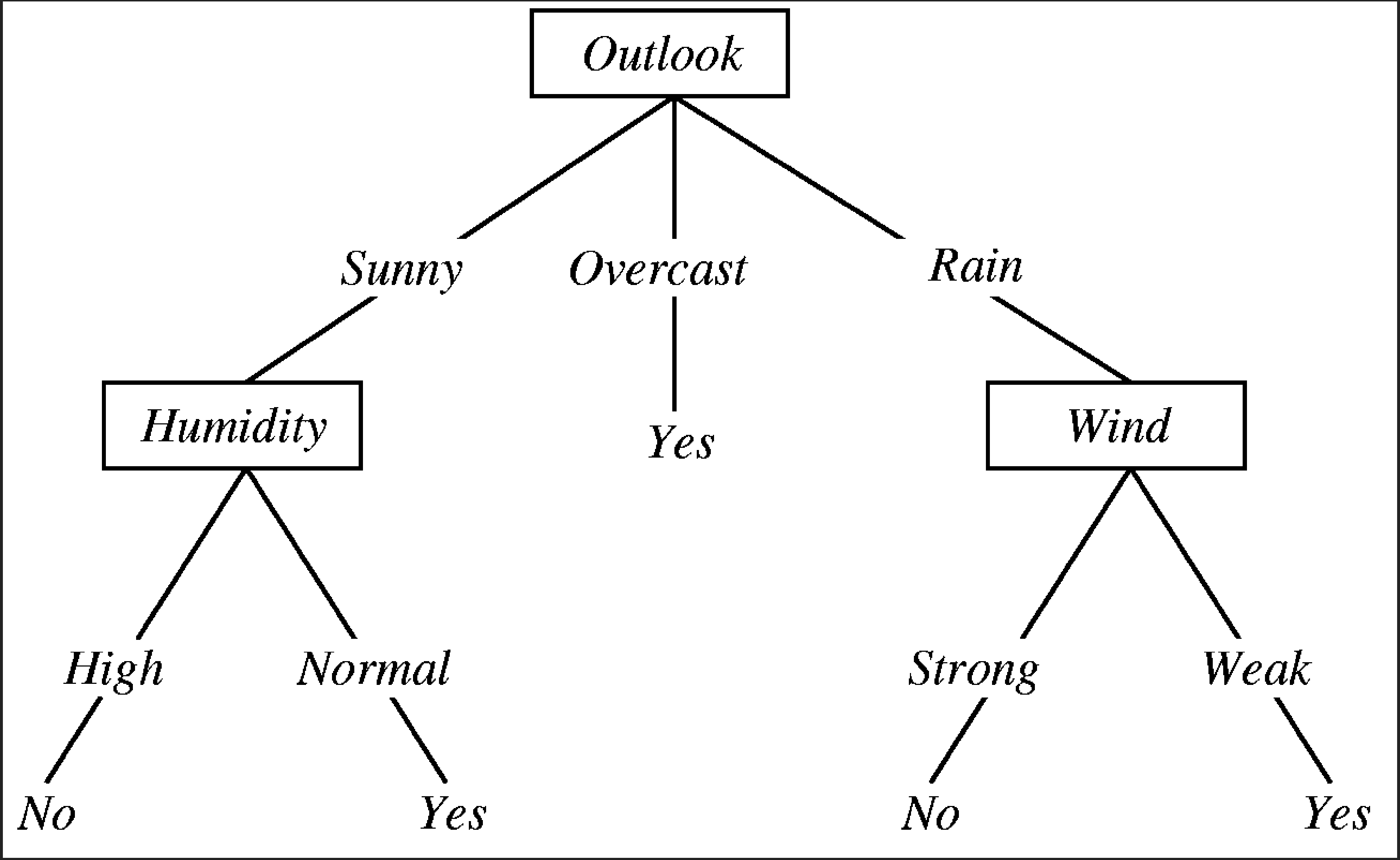


Deep Learning learns layers of features

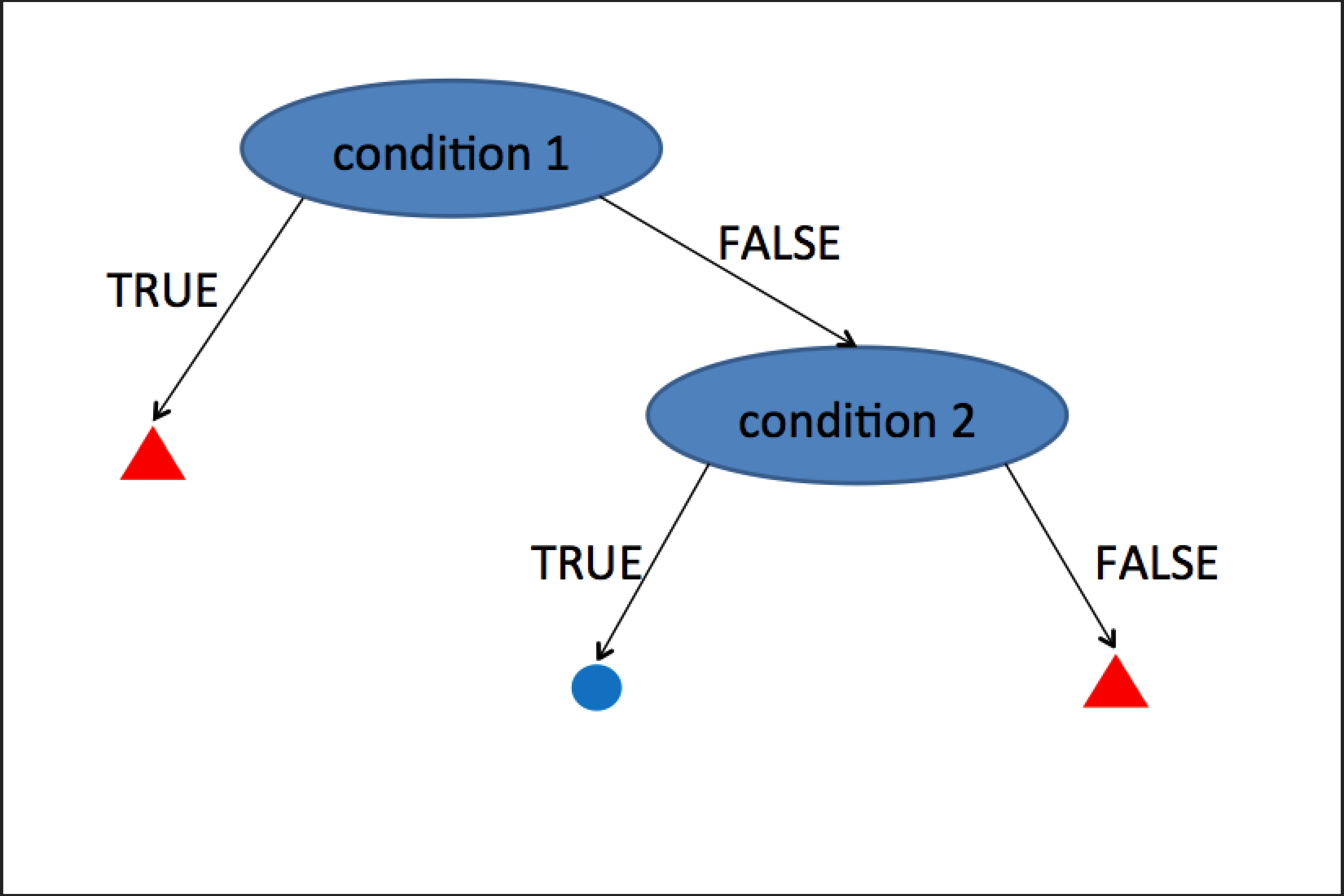


DECISION TREES

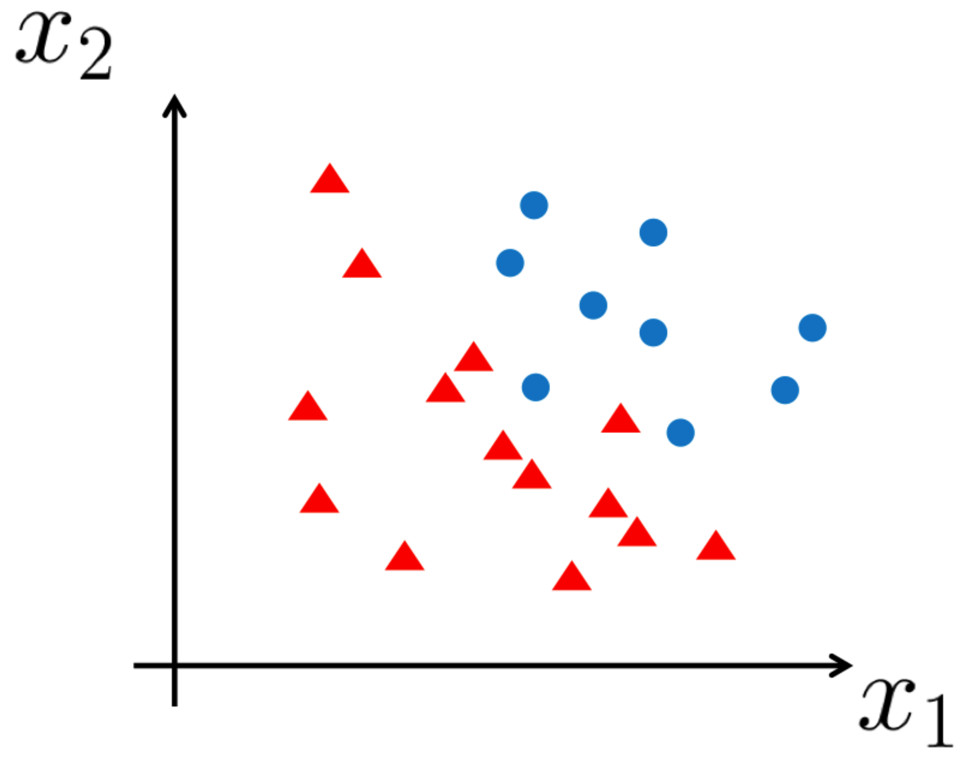
Example: predict outside play based on weather conditions.


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

DECISION TREES: IDEA

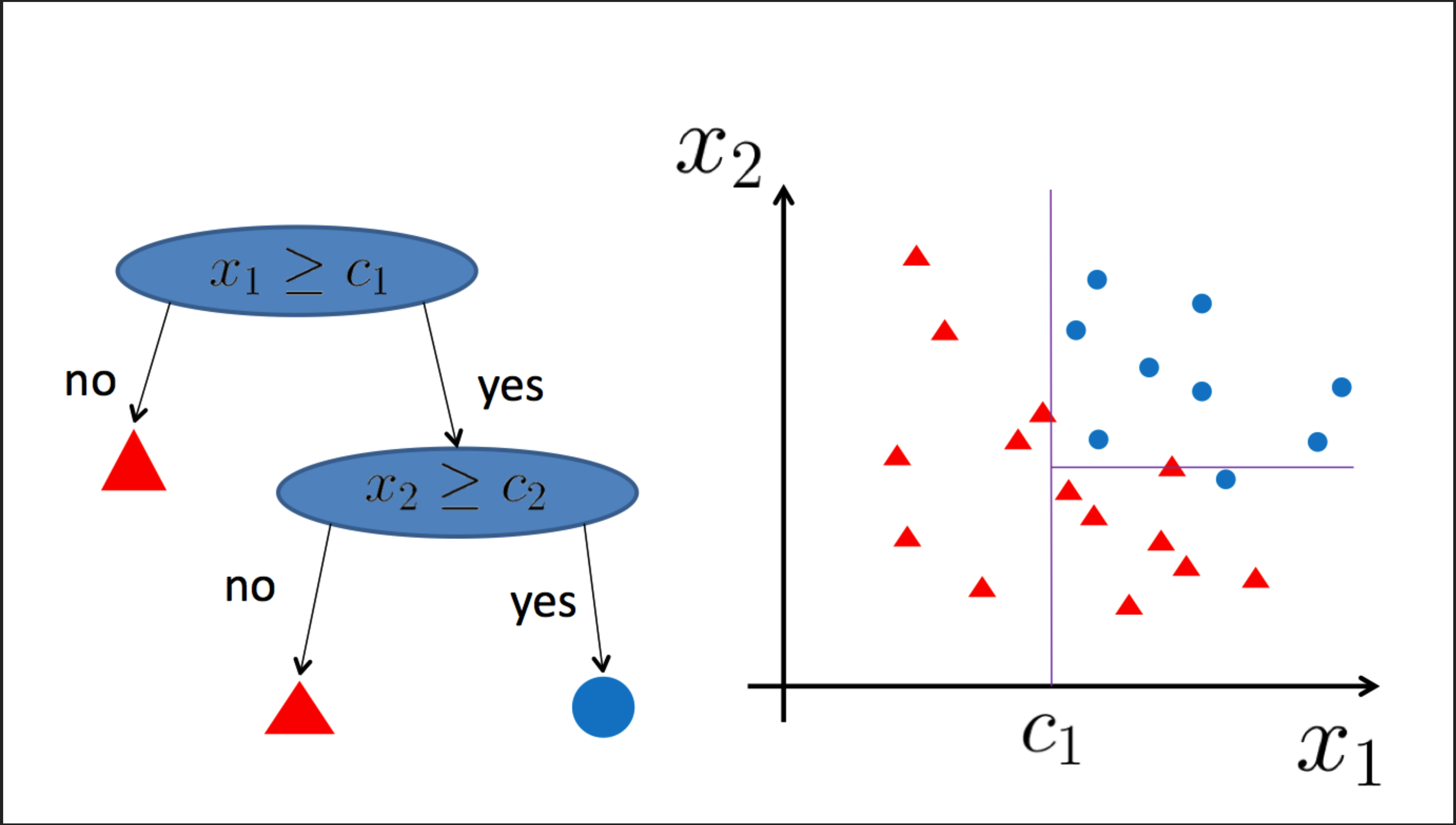


DECISION TREES



“Stump” conditions: x > c

DECISION TREES



DECISION TREE

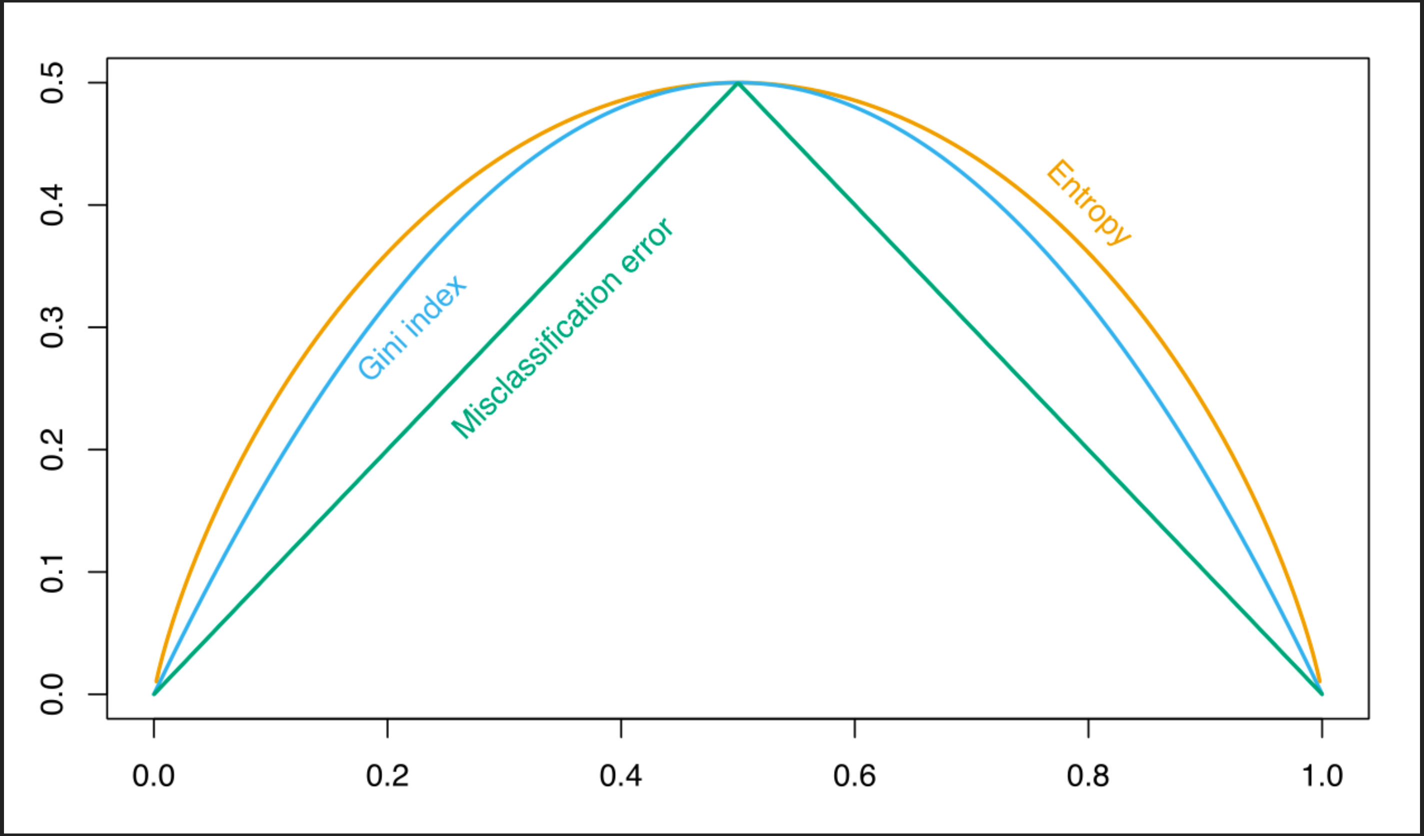
- fast & intuitive prediction
- building optimal decision tree is NP complete
- building tree from root using greedy optimization

  each time we split one leaf, finding optimal feature and threshold
- need criterion to select best splitting (feature, threshold)

SPLITTING CRITERIONS

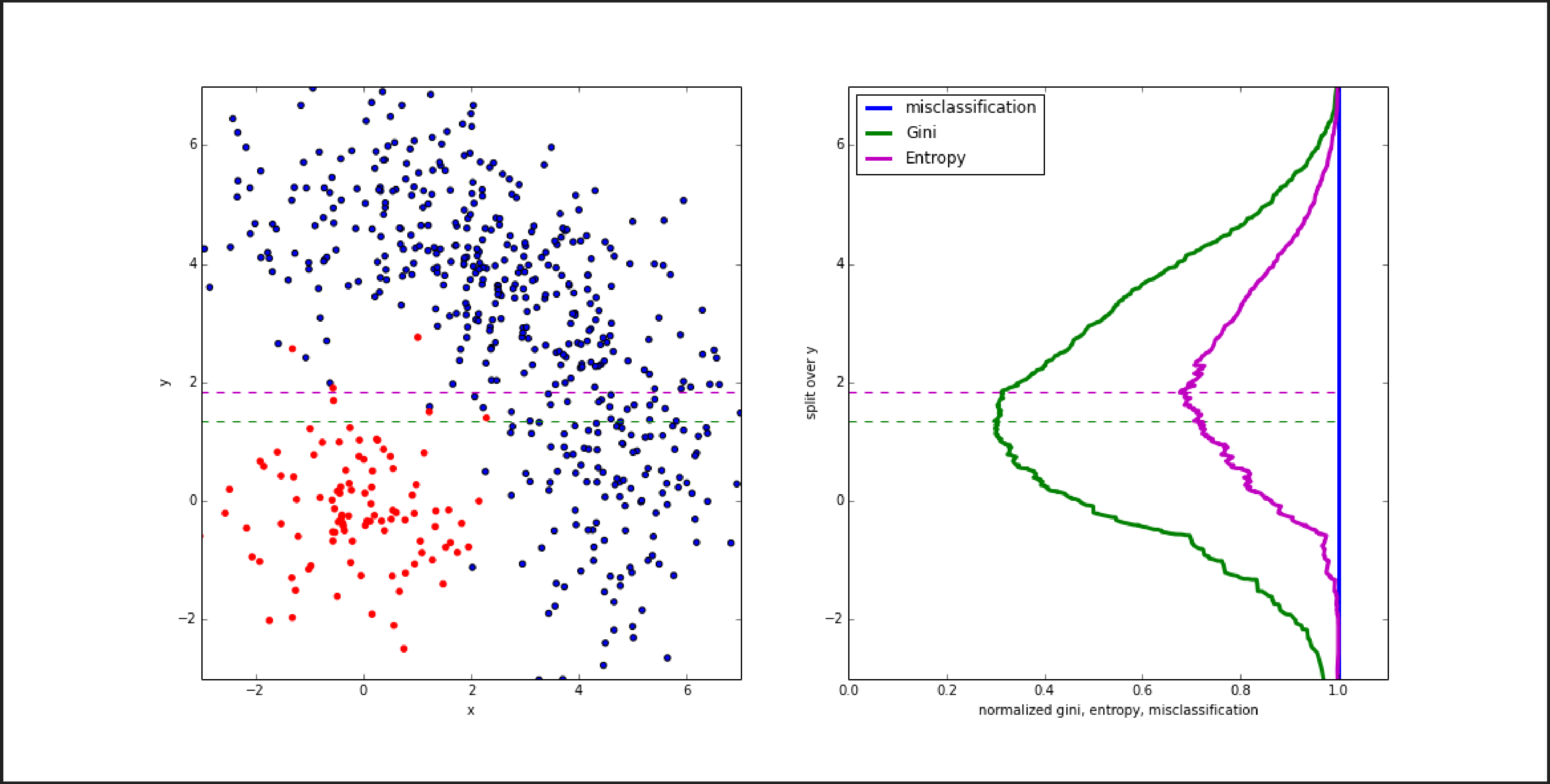
$$\text{TotalImpurity} = \sum_{\text{leaf}} \text{impurity}(leaf) \times \text{size}(leaf)$$

$$\begin{aligned} \text{Misclass.} &= \min(p, 1-p) \\ Gini &= p(1-p) \\ Entropy &= -p\log p - (1-p)\log(1-p) \end{aligned}$$



SPLITTING CRITERIONS

Why using Gini or Entropy not misclassification?



REGRESSION TREE

Greedy optimization (minimizing MSE):

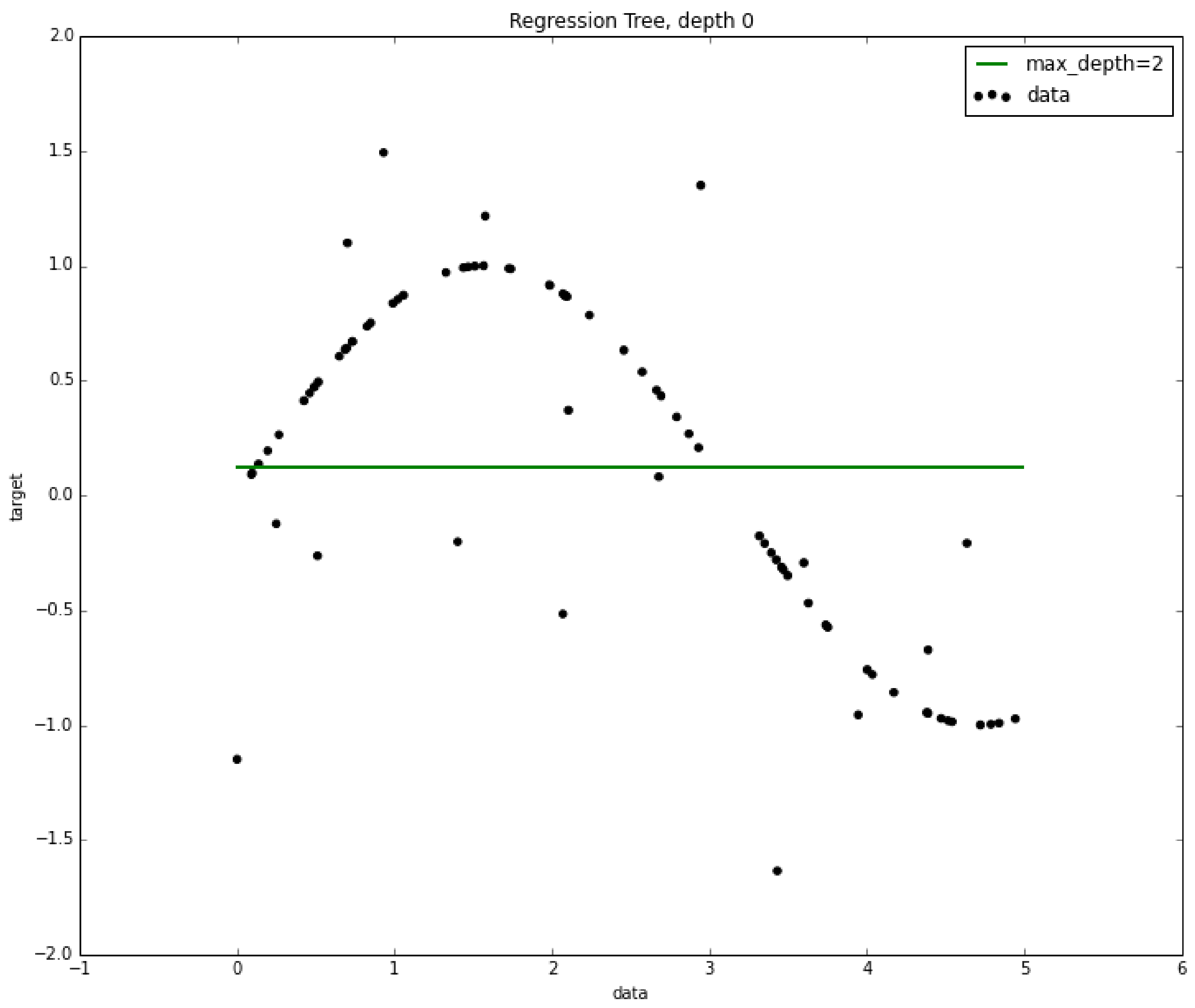
$$\text{GlobalMSE} \sim \sum_i (y_i - \hat{y}_i)^2$$

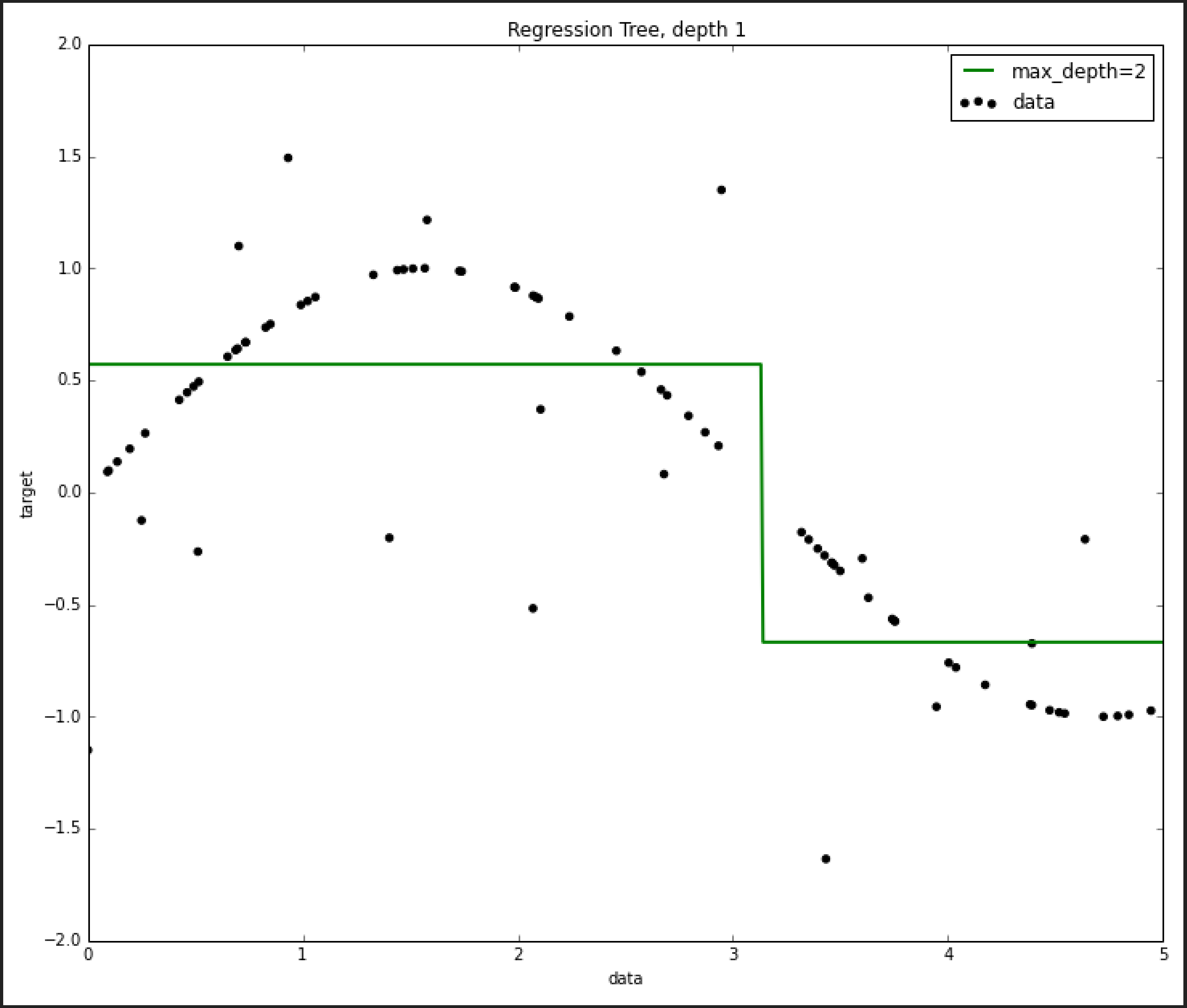
Can be rewritten as:

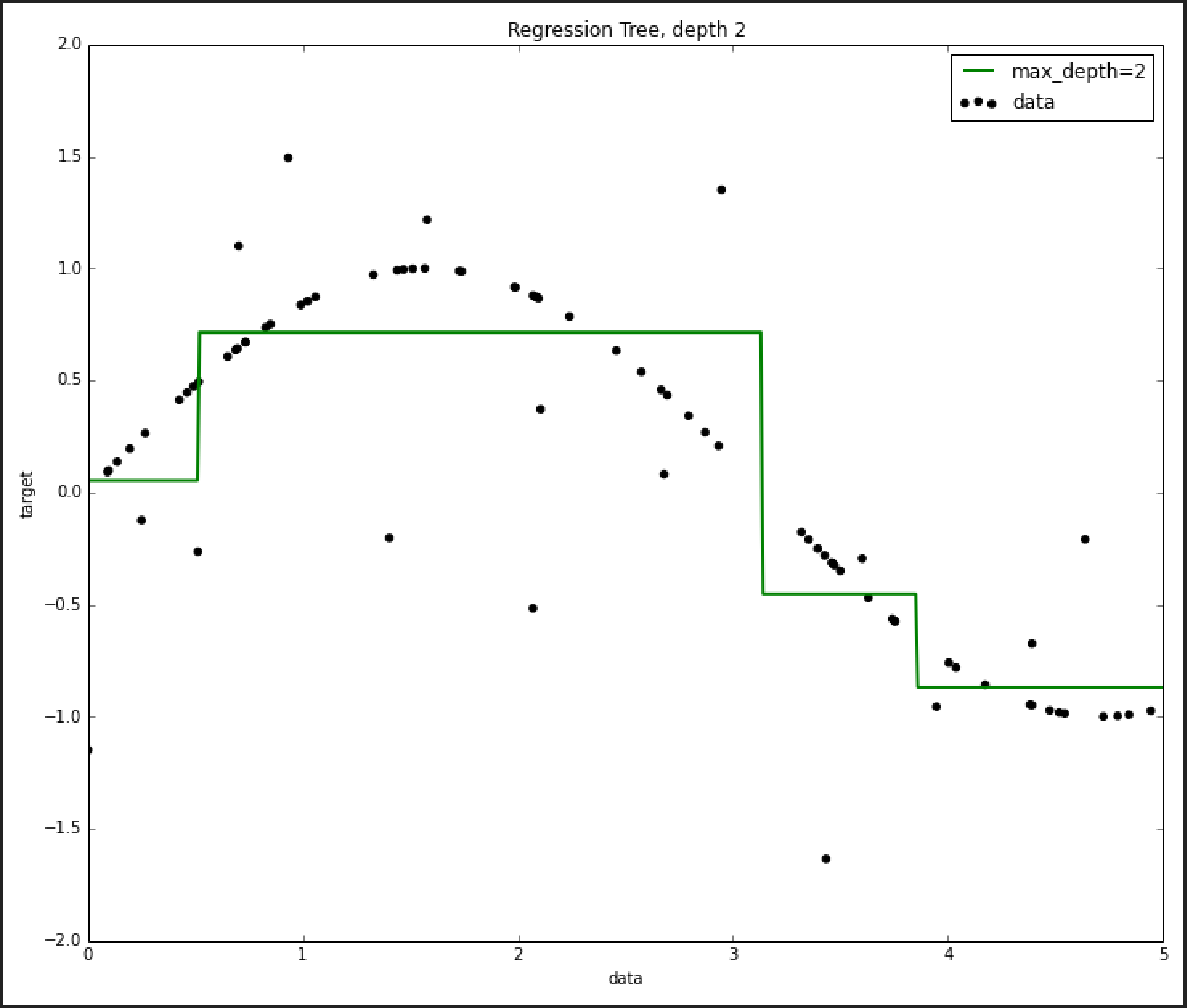
$$\text{GlobalMSE} \sim \sum_{\text{leaf}} \text{MSE(leaf)} \times \text{size(leaf)}$$

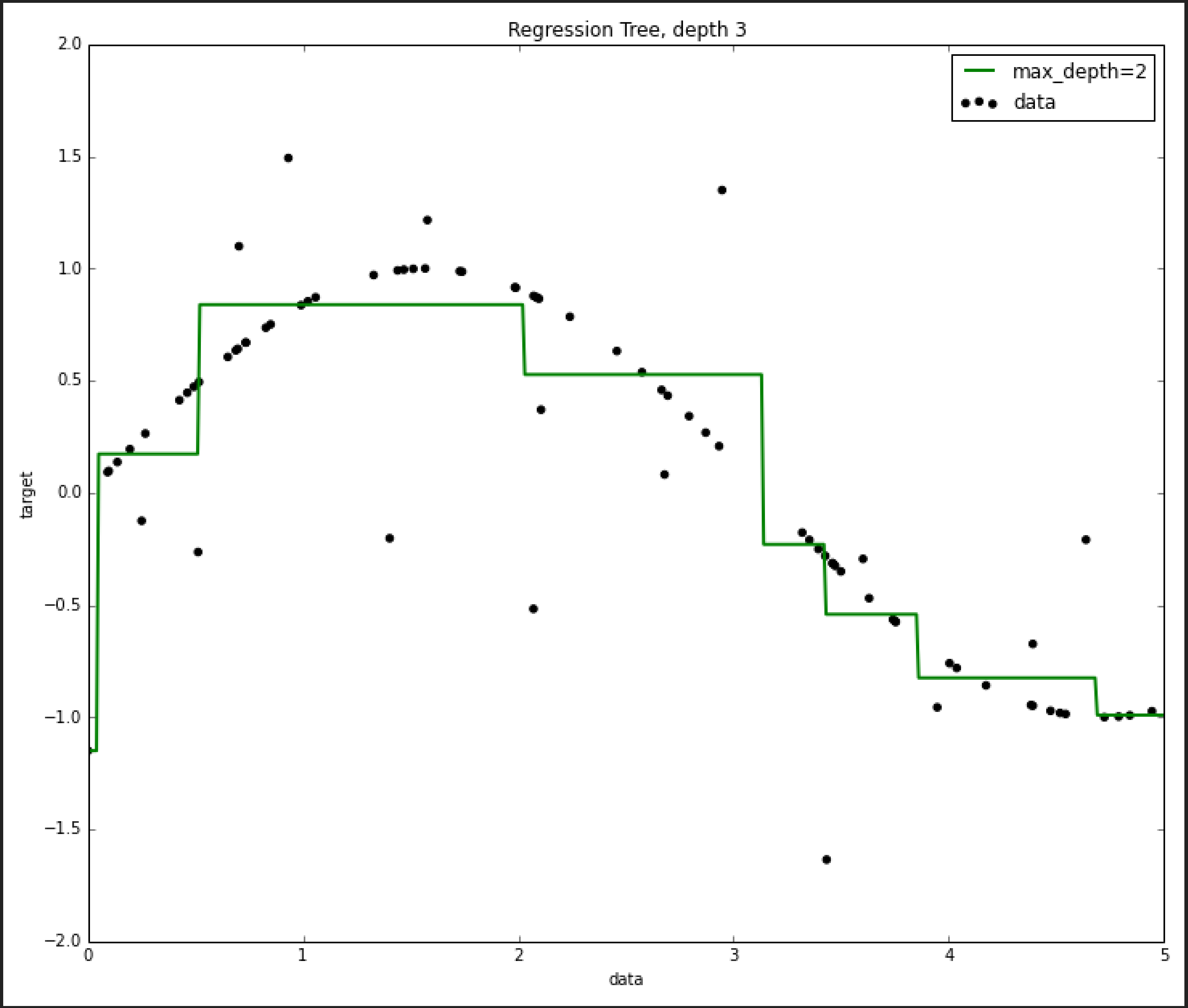
MSE(leaf) is like 'impurity' of leaf

$$\text{MSE(leaf)} = \frac{1}{\text{size(leaf)}} \sum_{i \in \text{leaf}} (y_i - \hat{y}_i)^2$$

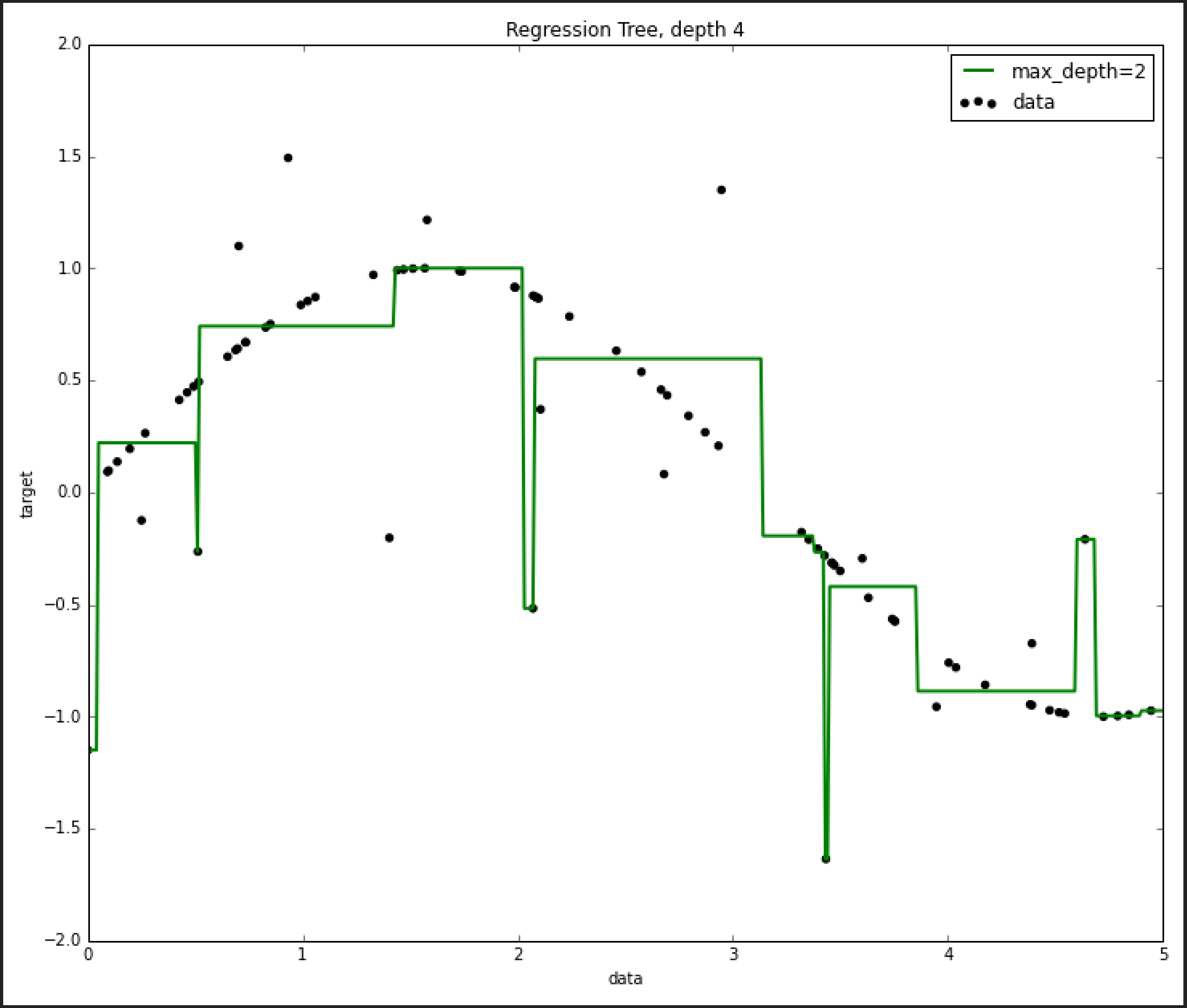

Regression Tree, depth 0
max_depth=2
data
target
data
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1
0
1
2
3
4
5
6


Regression Tree, depth 1
max_depth=2
data
target
data
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5


Regression Tree, depth 2
max_depth=2
data
target
data
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5



Regression Tree, depth 3
max_depth=2
data
target
data
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5



Regression Tree, depth 4
max_depth=2
data
target
data
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5

In most cases, regression trees are optimizing MSE:

$$\text{GlobalMSE} \sim \sum_i (y_i - \hat{y}_i)^2$$

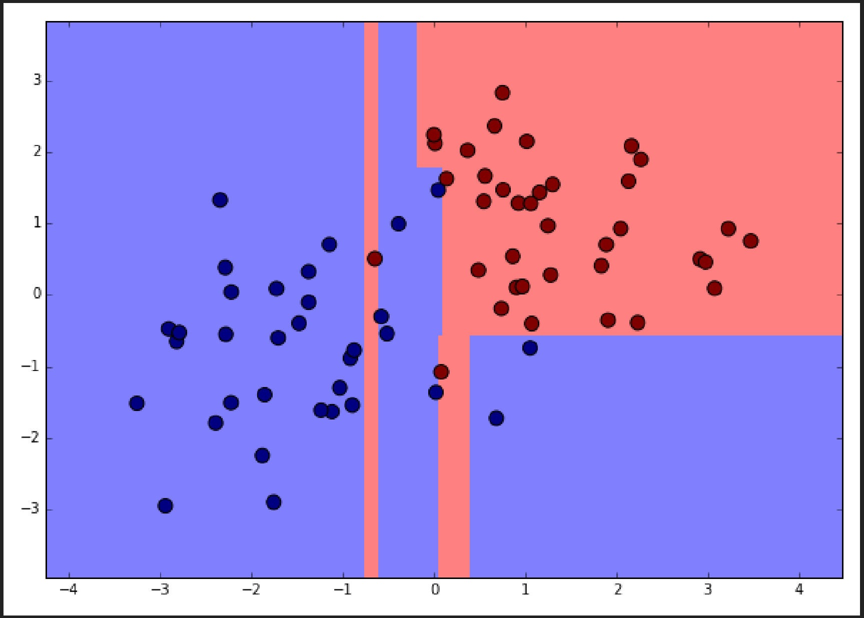
But other options also exist, i.e. MAE:

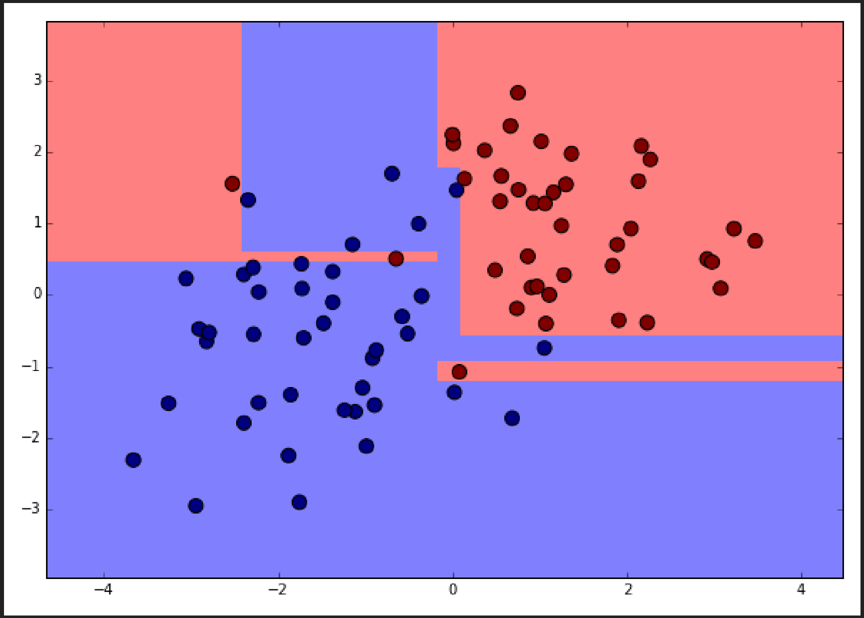
$$\text{GlobalMAE} \sim \sum_i |y_i - \hat{y}_i|$$

For MAE optimal value of leaf is median, not mean.

DECISION TREES INSTABILITY

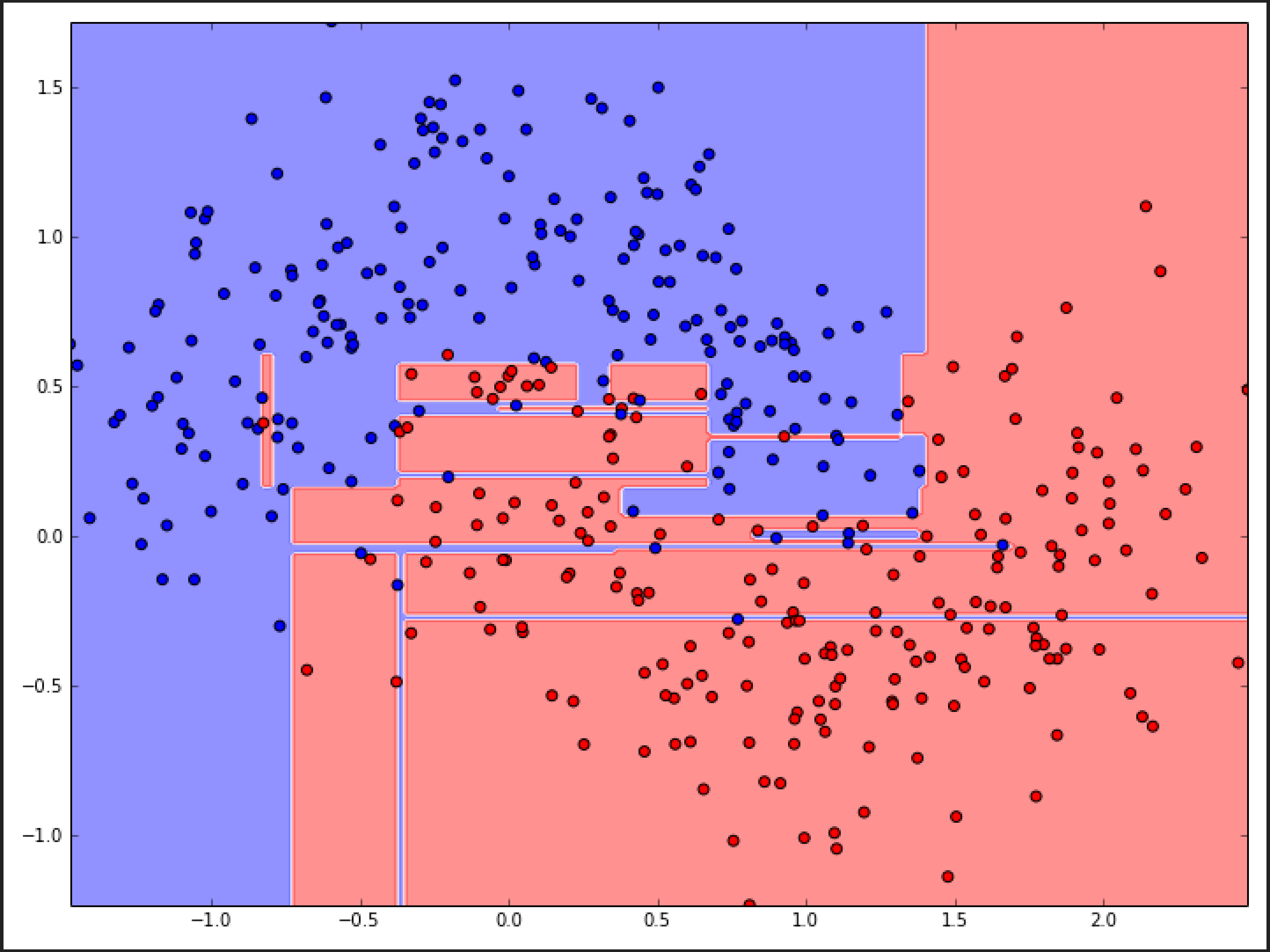
Little variation in training dataset produce different classification rule.





PRE-STOPPING OF DECISION TREE

Tree keeps splitting until each event is correctly classified.

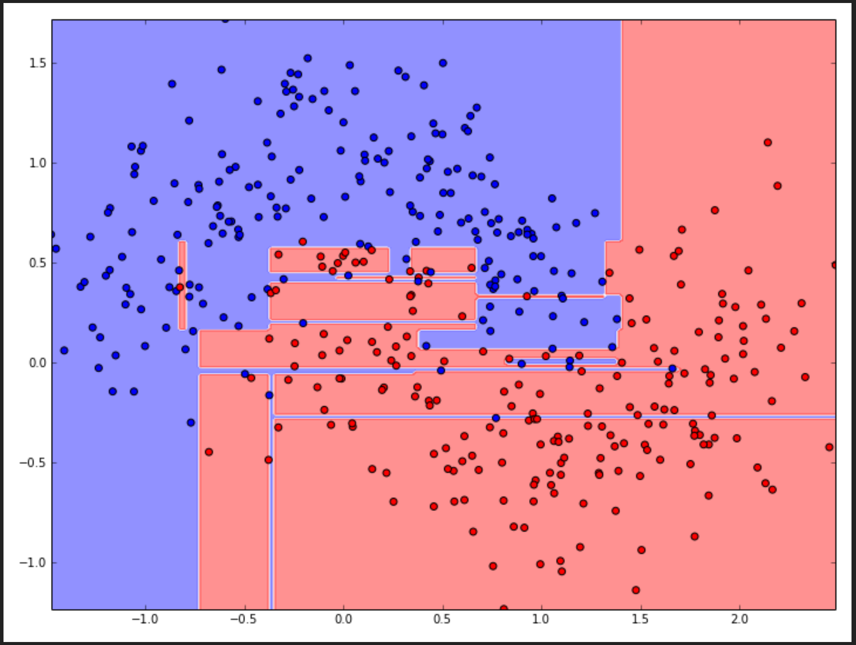


PRE-STOPPING

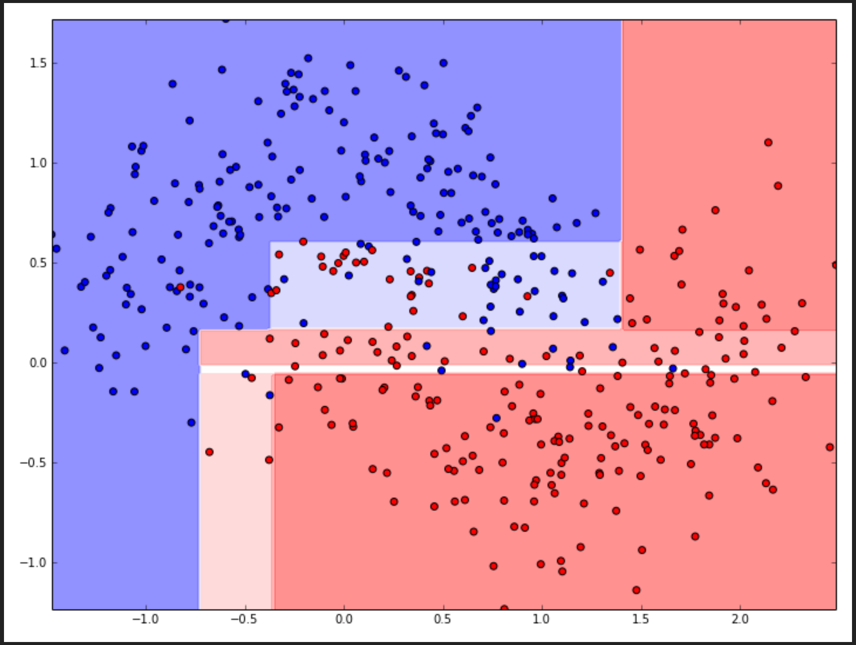
We can stop the process of splitting by imposing different restrictions.

- limit the depth of tree
- set minimal number of samples needed to split the leaf
- limit the minimal number of samples in leaf
- more advanced: maximal number of leaves in tree

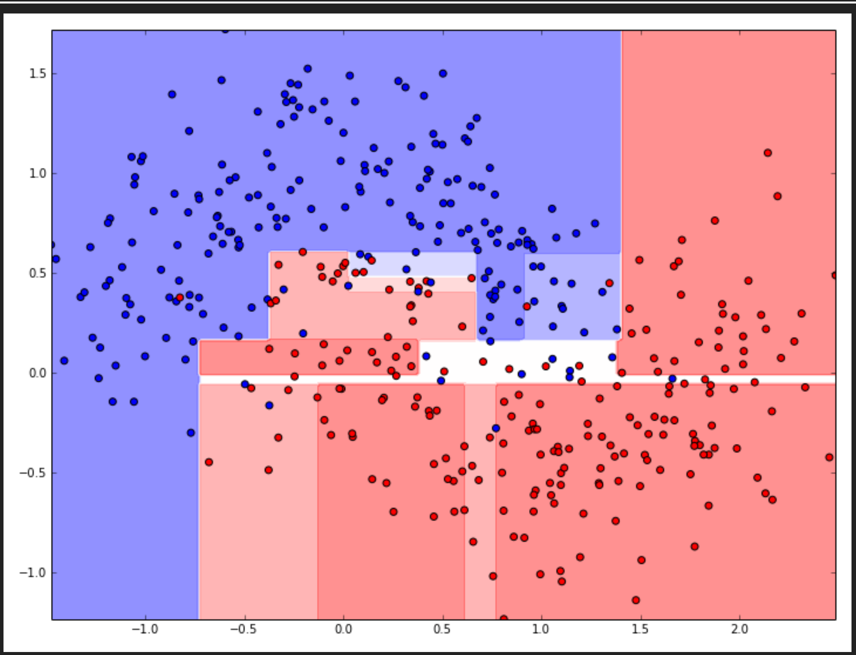
Any combinations of rules above is possible.



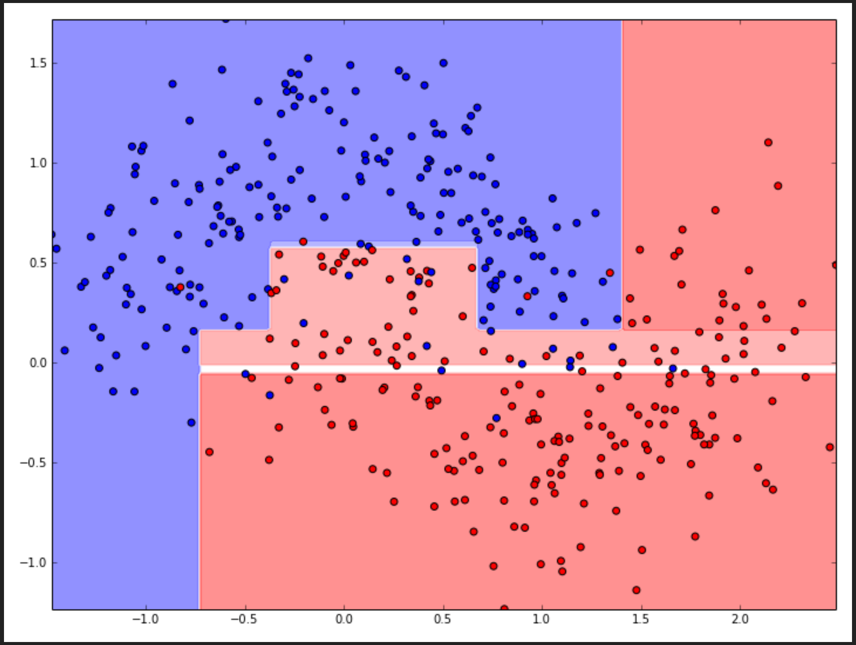
no prepruning



max_depth



min # of samples in leaf



maximal number of leaves

POST-PRUNING

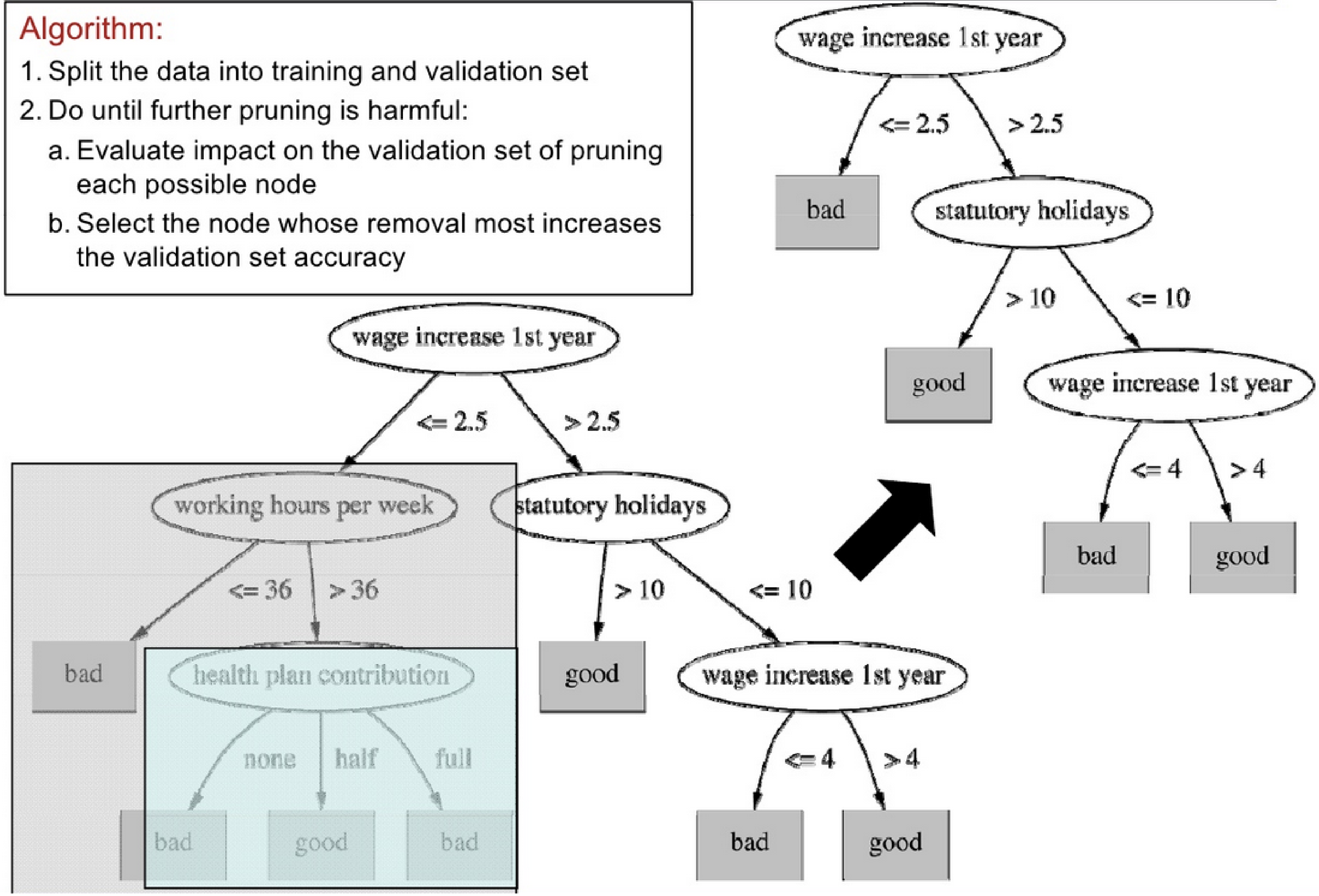
When tree tree is already built we can try optimize it to simplify formula.

Generally, much slower than pre-stopping.

Subtree Replacement

Algorithm:

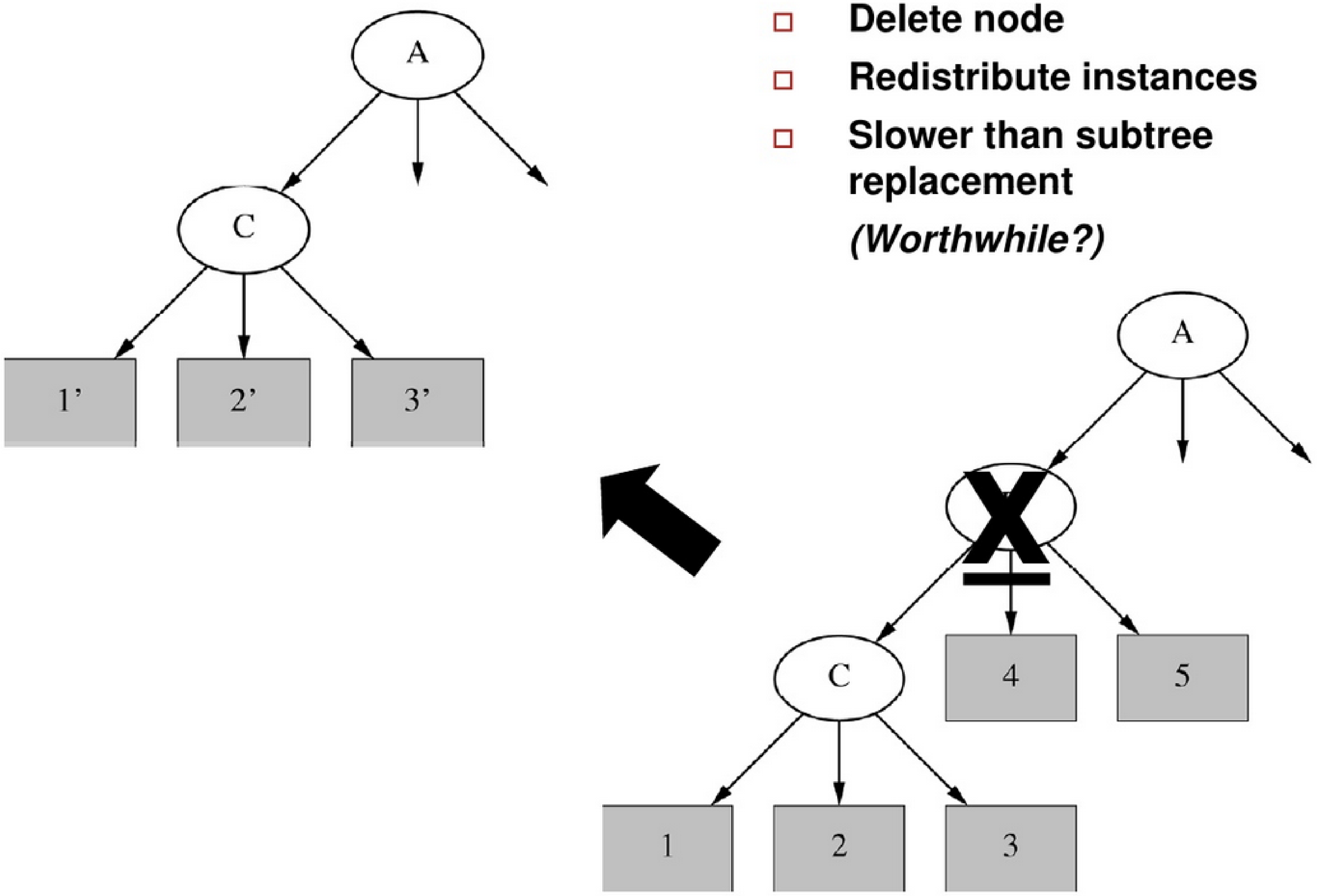
1. Split the data into training and validation set
2. Do until further pruning is harmful:
   a. Evaluate impact on the validation set of pruning each possible node
   b. Select the node whose removal most increases the validation set accuracy



Subtree Raising

- Delete node
- Redistribute instances
- Slower than subtree replacement

  (Worthwhile?)



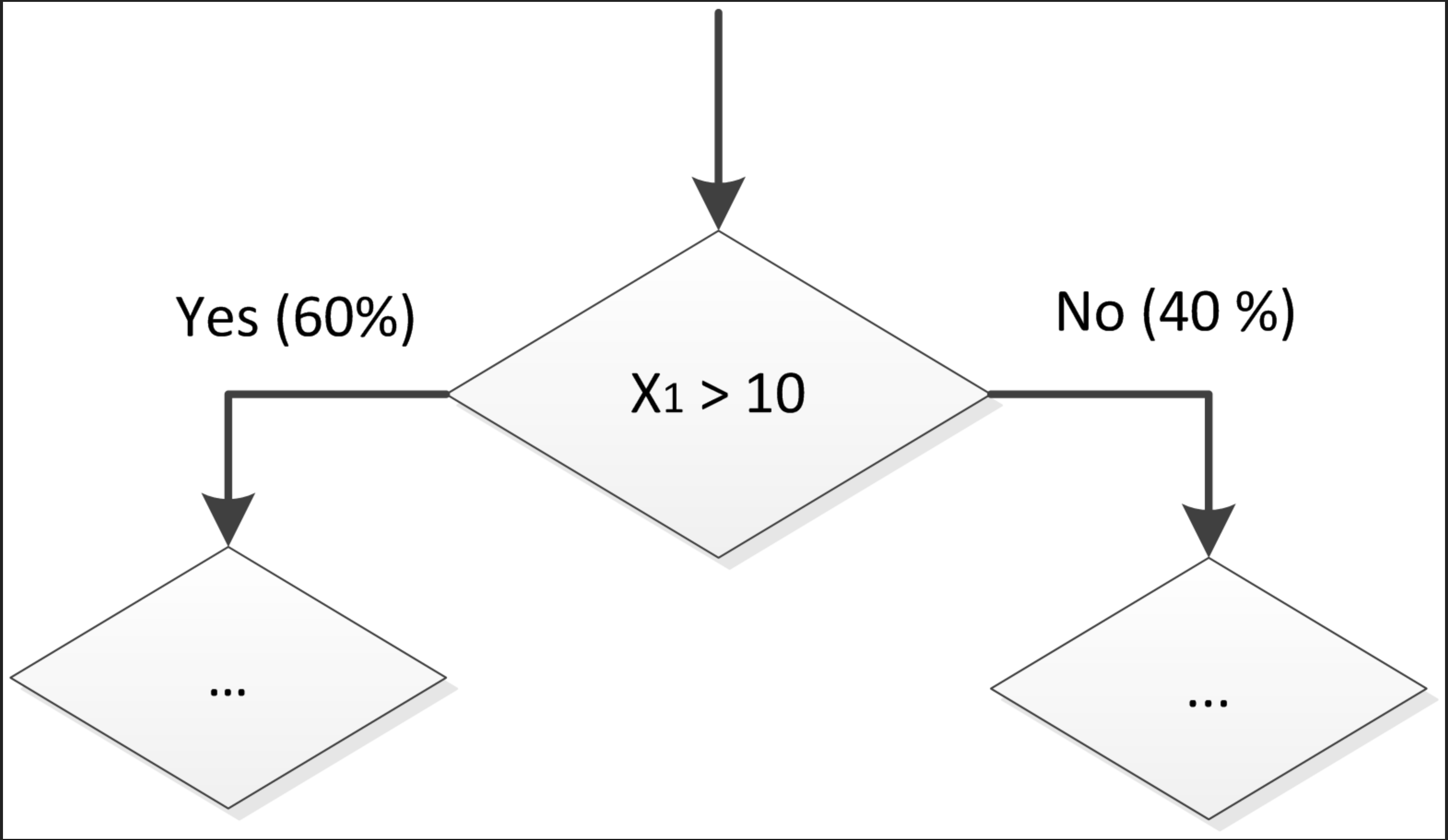
SUMMARY OF DECISION TREE

1. Very intuitive algorithm for regression and classification
2. Fast prediction
3. Scale-independent
4. Supports multiclassification

But

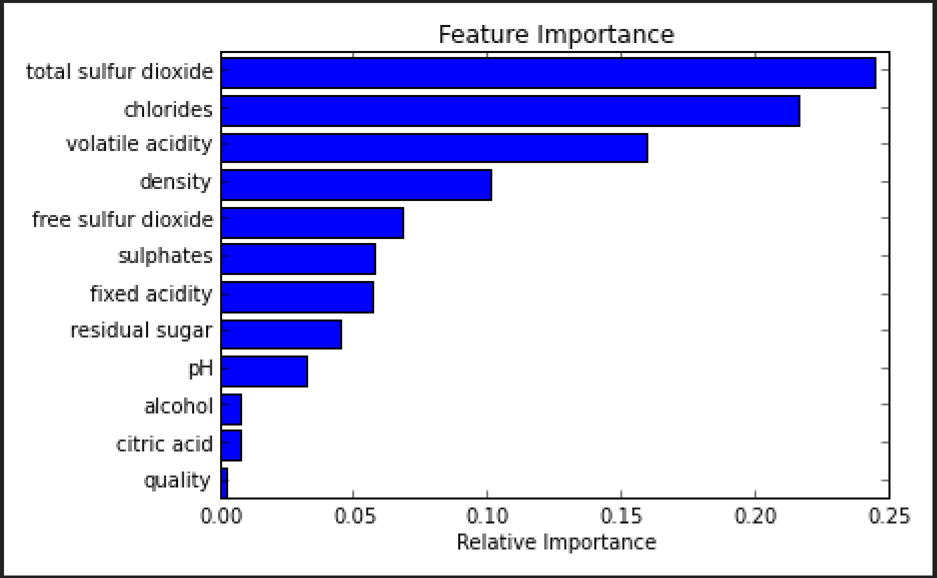
1. Training optimal tree is NP-complex
2. Trained greedily by optimizing Gini index or entropy (fast!)
3. Non-stable
4. Uses only trivial conditions

MISSING VALUES IN DECISION TREES



If event being predicted lacks x_1, we use prior probabilities.

FEATURE IMPORTANCES



Different approaches exist to measure importance of feature in final model

Importance of feature $\neq$ quality provided by one feature

FEATURE IMPORTANCES

- tree: counting number of splits made over this feature
- tree: counting gain in purity (e.g. Gini)
  fast and adequate
- common recipe: train without one feature,
  compare quality on test with/without one feature

  requires many evaluations

- common recipe: feature shuffling

  take one column in test dataset and shuffle them. Compare quality with/without shuffling.

THE END

Tomorrow: ensembles and boosting