MACHINE LEARNING IN HIGH ENERGY PHYSICS

LECTURE #2

Alex Rogozhnikov, 2015

RECAPITULATION

- classification, regression
- kNN classifier and regressor
- ROC curve, ROC AUC
- QDA
- logistic function and logistic regression

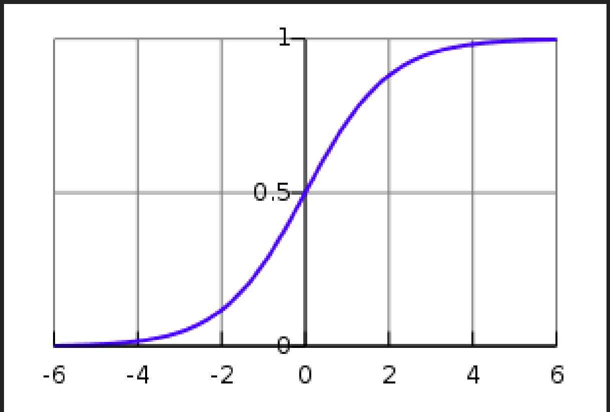
OPTIMAL BAYESIAN CLASSIFIER

Given knowledge about distributions, we can build optimal classifier

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

But distributions are complex, contain many parameters.

LOGISTIC REGRESSION



$$d(x) = \langle w, x \rangle + w_0$$

$$p_1(x) = \sigma(d(x))$$

$$p_0(x) = 1 - p_1(x)$$

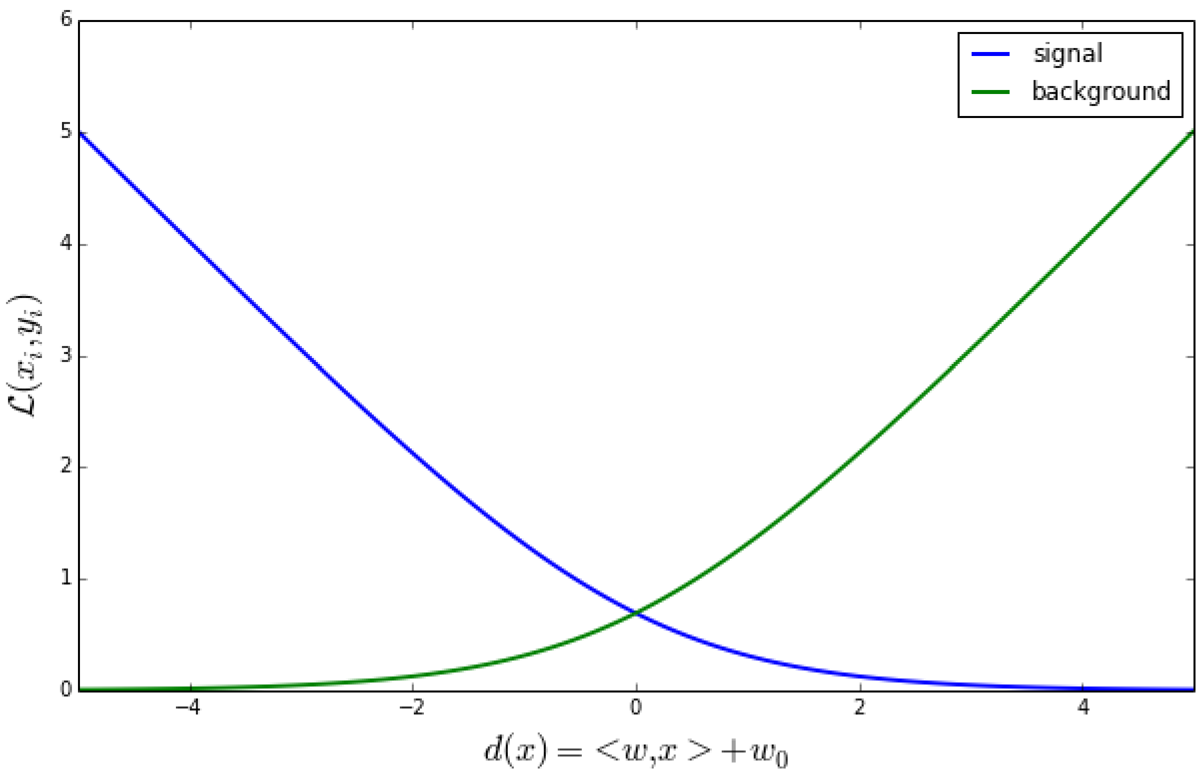
Optimizing weights w, w_0 to maximize log-likelihood

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

LOGISTIC LOSS

Loss penalty for single observation

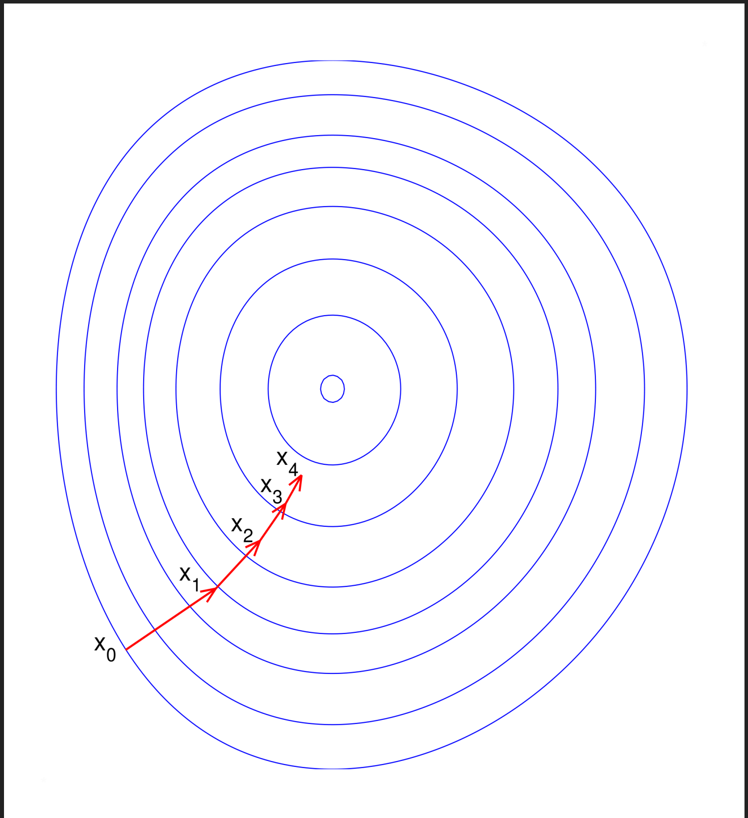
$$L(x_i, y_i) = -\ln(p_{y_i}(x_i)) = \begin{cases} \ln(1 + e^{-d(x_i)}), & y_i = 1 \\ \ln(1 + e^{d(x_i)}), & y_i = 0 \end{cases}$$



signal
background
$\mathcal{L}(x_i, y_i)$
$d(x) = <w,x> + w_0$
0
1
2
3
4
5
6
−4
−2
0
2
4

Visualization of logistic regression

GRADIENT DESCENT & STOCHASTIC OPTIMIZATION



Problem:
finding w to minimize L

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

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

STOCHASTIC GRADIENT DESCENT

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

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

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

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

SVM with polynomial kernel visualization

After adding new features, classes may become separable.

KERNEL TRICK

P is projection operator (which adds new features).

$$d(x) = \langle w, P(x) \rangle$$

Assume

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

and look for optimal α_i

$$d(x) = \sum_i \alpha_i \langle P(x_i), P(x) \rangle = \sum_i \alpha_i K(x_i, x)$$

We need only kernel: $K(x, y) = \langle P(x), P(y) \rangle$

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

KERNEL TRICK

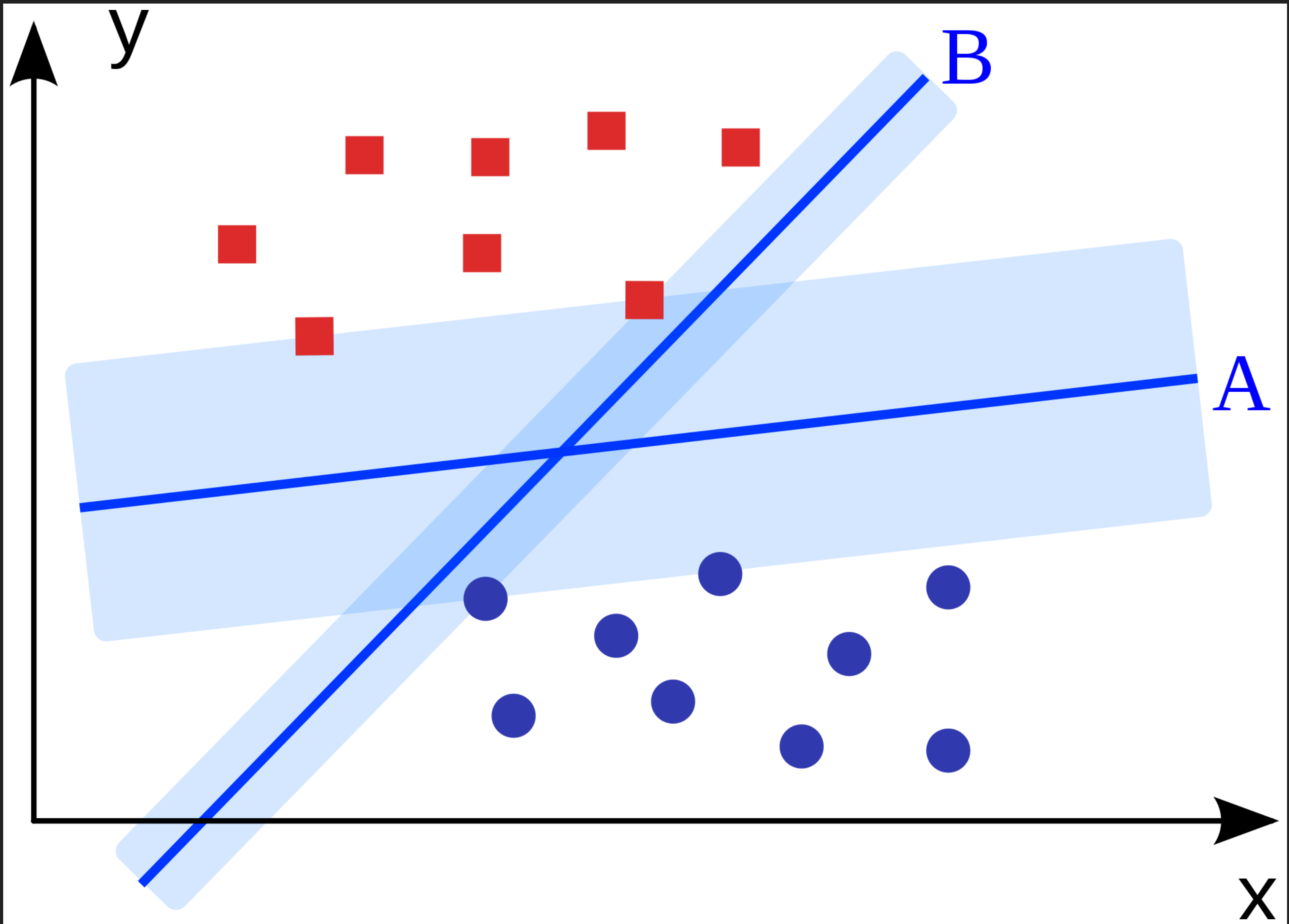
Popular kernel is gaussian Radial Basis Function:

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

Corresponds to projection to Hilbert space.

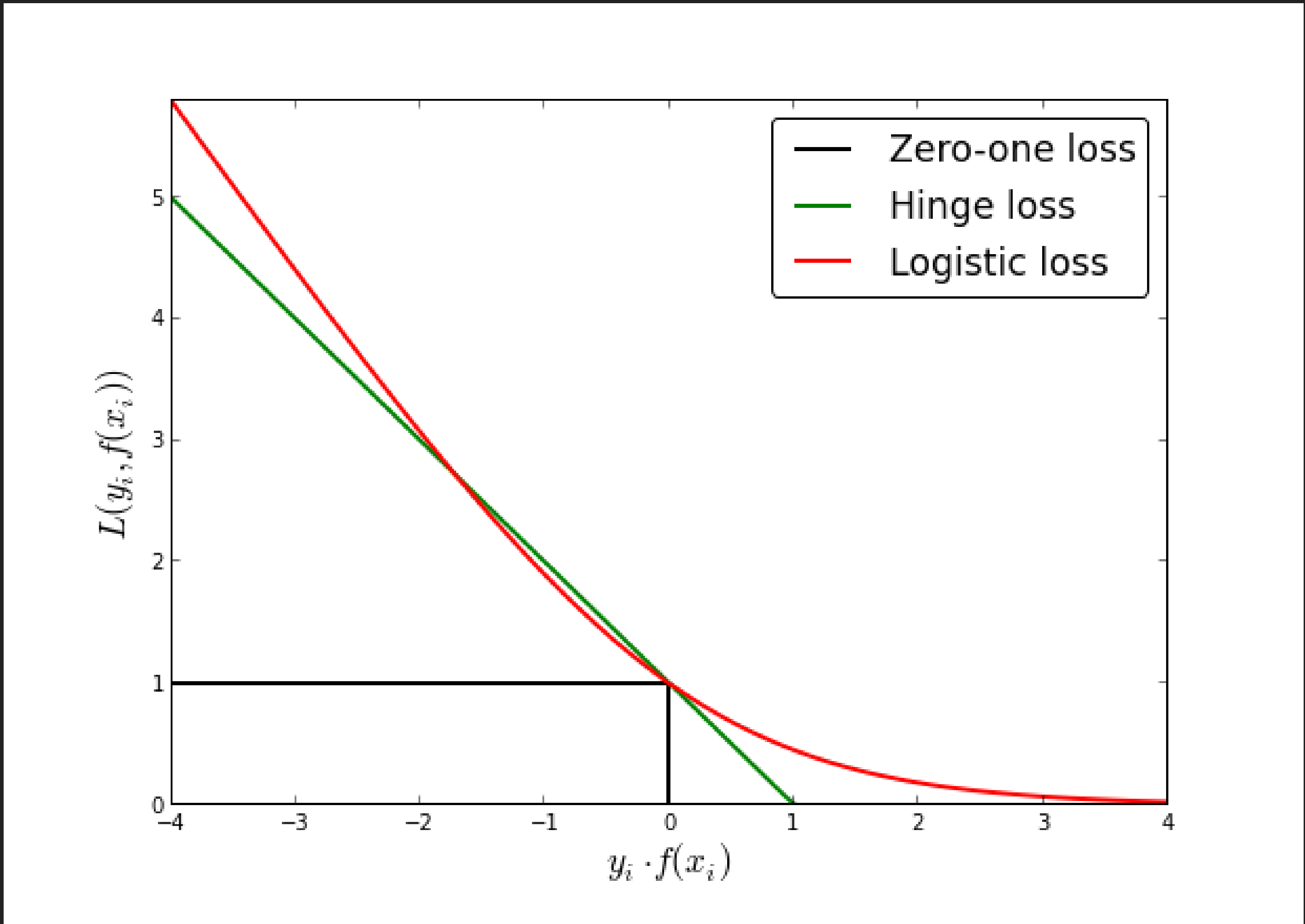
SUPPORT VECTOR MACHINE

SVM selects decision rule with maximal possible margin.



HINGE LOSS FUNCTION

SVM uses different loss function:



ESTIMATING QUALITY, OVERFITTING

1. knn example: 2. dimensionality in kernel: Solution: holdout! (more details in seminars)

REGULARIZATION

When number of weights is high, overfitting is very probable

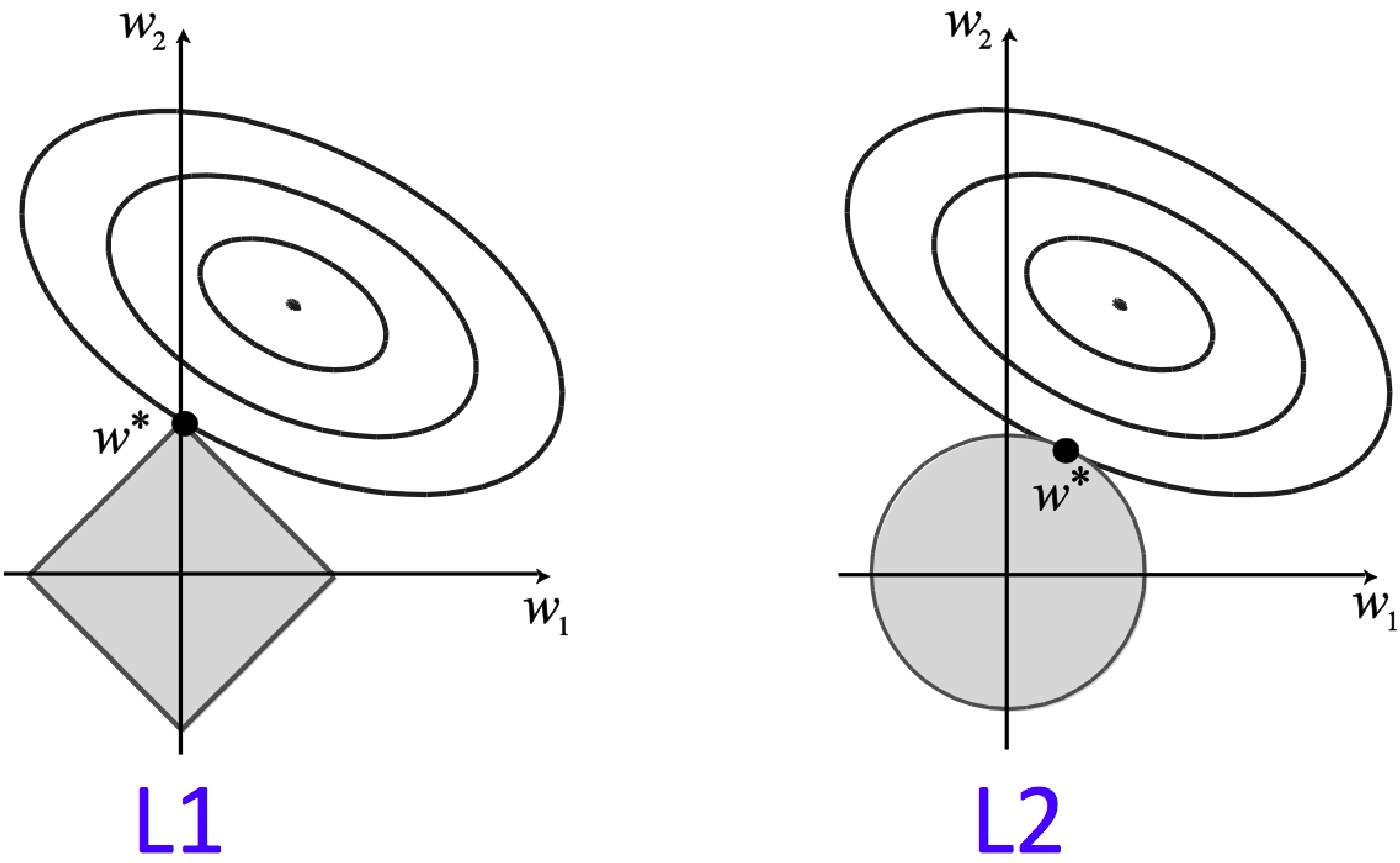
Adding regularization term to loss function:

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

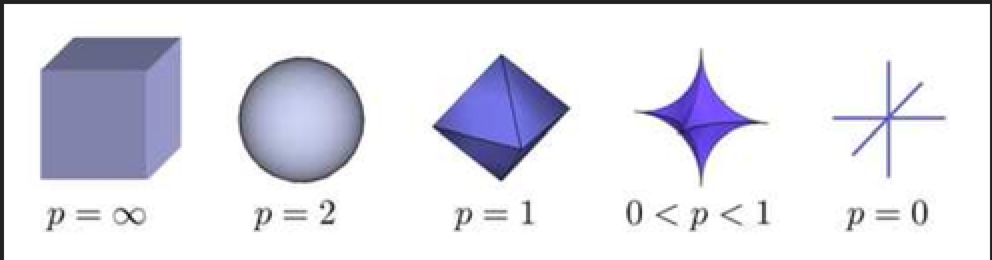
- L_2 regularization : $L_{reg} = \alpha \sum_j |w_j|^2$
- L_1 regularization: $L_{reg} = \beta \sum_j |w_j|$
- $L_1 + L_2$ regularization: $L_{reg} = \alpha \sum_j |w_j|^2 + \beta \sum_j |w_j|$

REGULARIZATIONS

L_1 regularization encourages sparsity


w_2
w^*
w_1
L1
w_2
w^*
w_1
L2

L_P REGULARIZATIONS



- 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

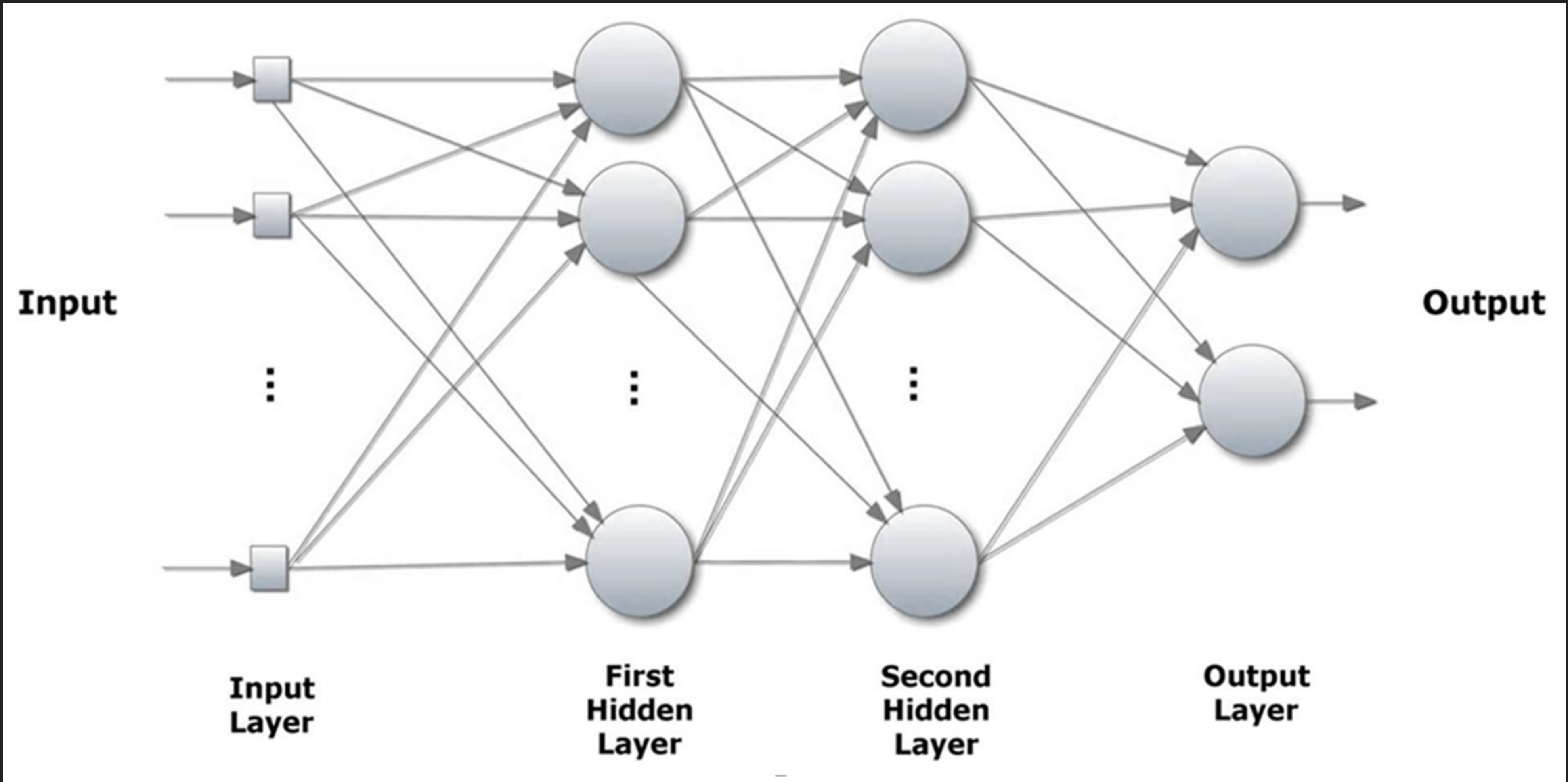
- classifier based on linear decision rule
- training is reduced to convex optimization
- other decision rules are achieved by adding new features (generalized linear models)
- stochastic optimization is used
- can handle > 1000 features, requires regularization
- no iteraction between features

[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:

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

Problem: find set of weights, that minimizes error on train

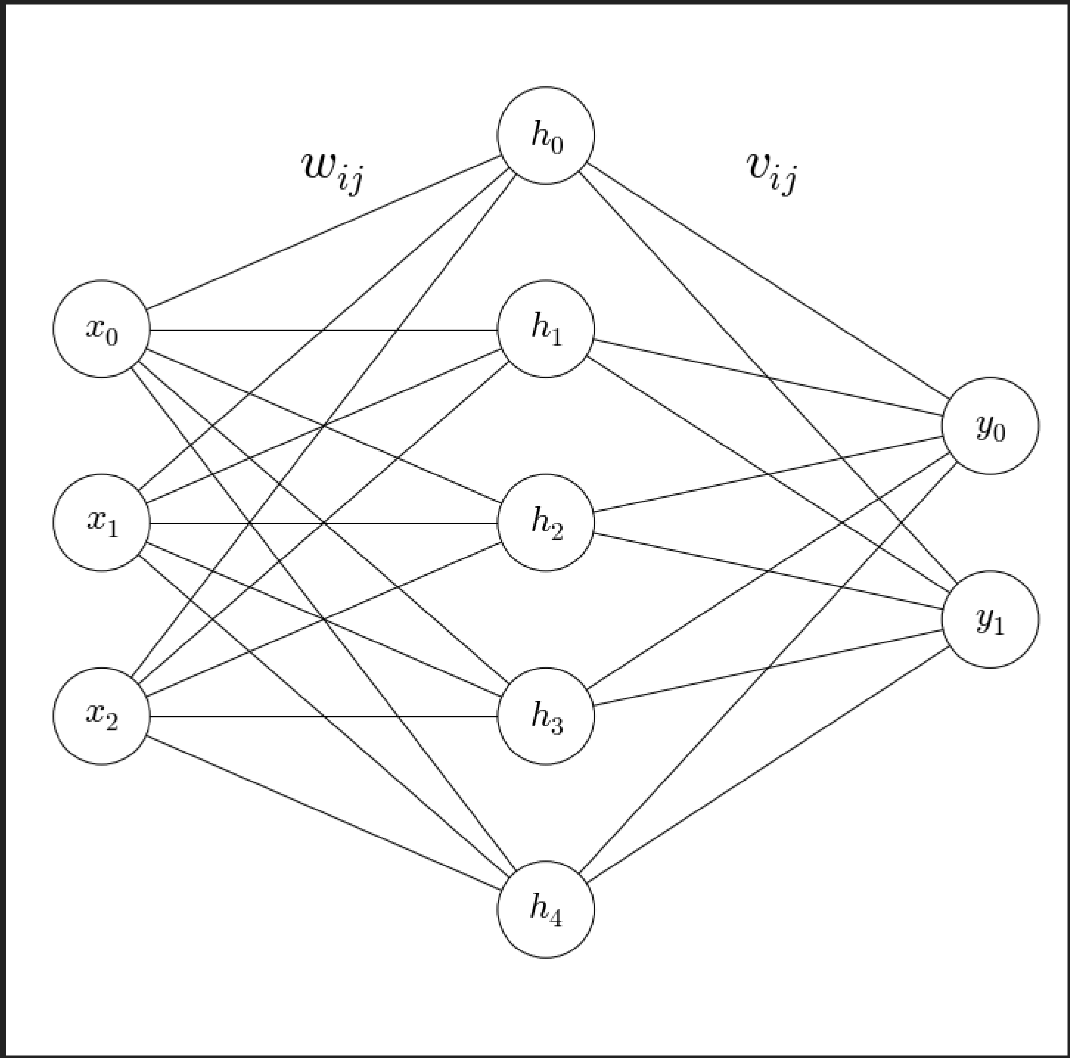
Problem: find set of weights, that minimizes error on train dataset. (discrete optimization)

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

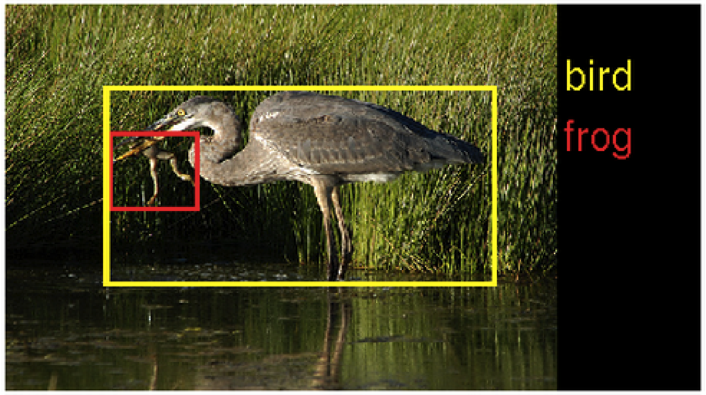
- Powerful general purpose algorithm for classification and regression
- Non-interpretable formula
- Optimization problem is non-convex with local optimums and has many parameters
  Stochastic optimization speeds up process and helps not to be caught in local minimum.
- Overfitting due to large amount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

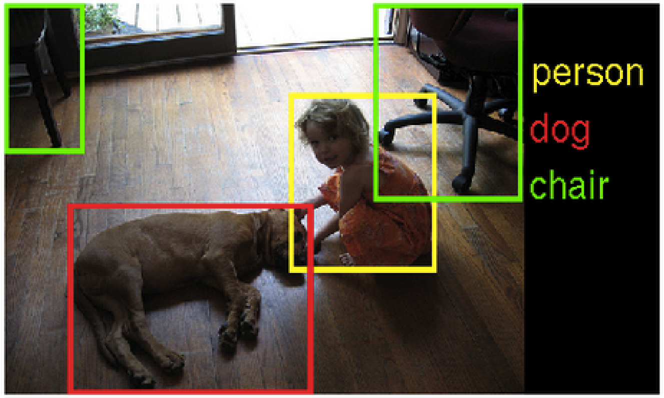
DEEP LEARNING

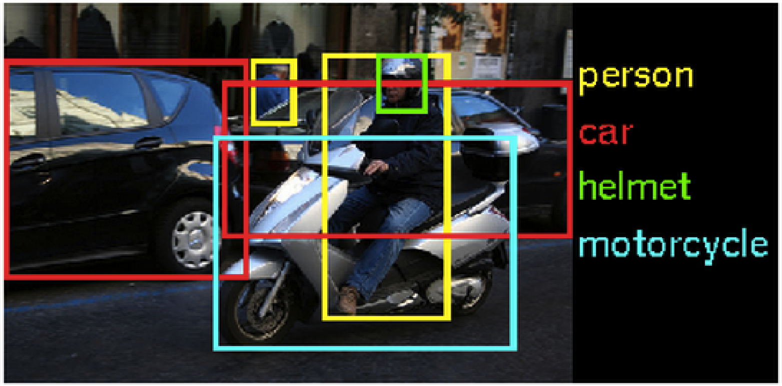
Gradient diminishes as number of hidden layers grows.
Usually 1-2 hidden layers are used.

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

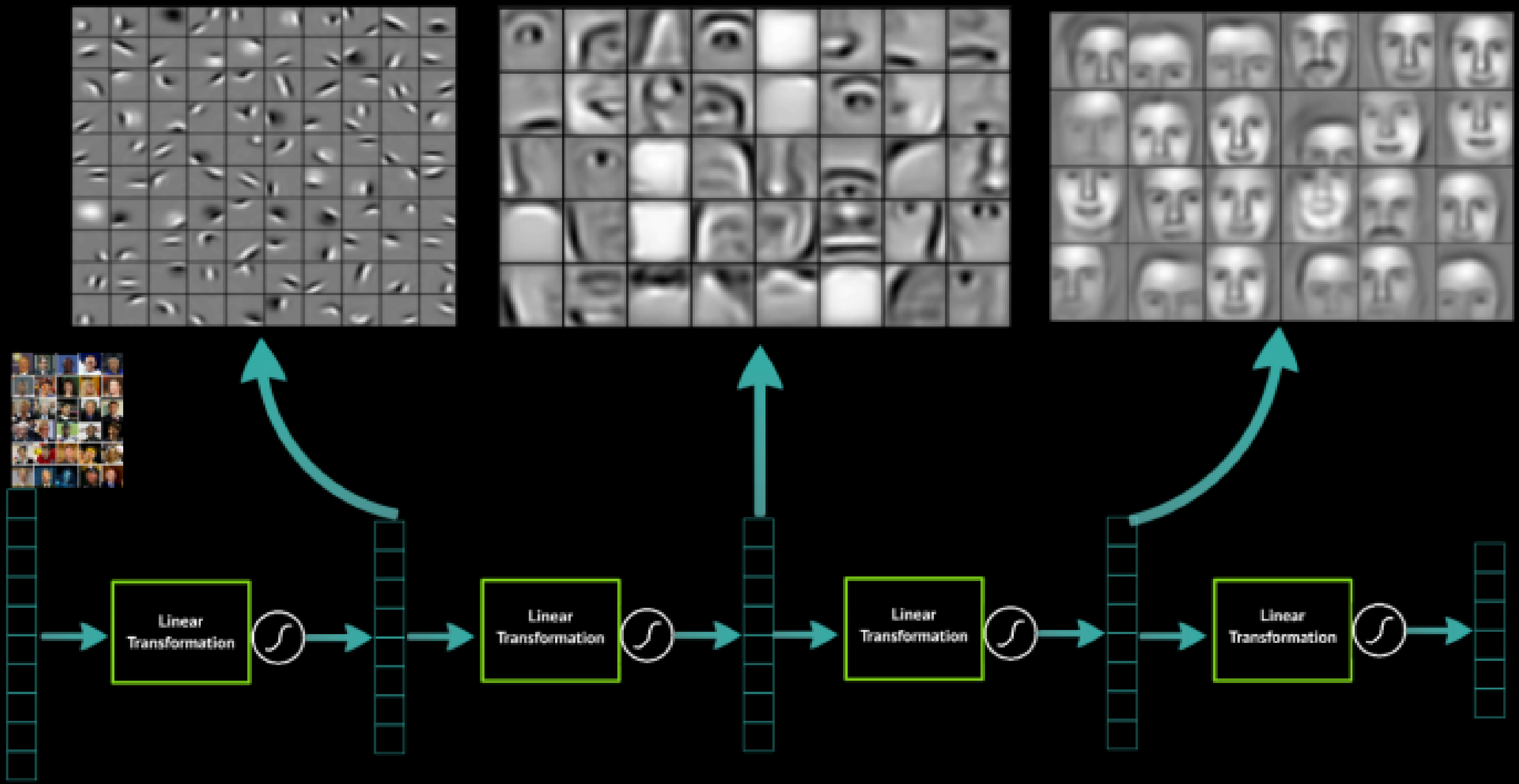
Example ILSVRC2014 images:





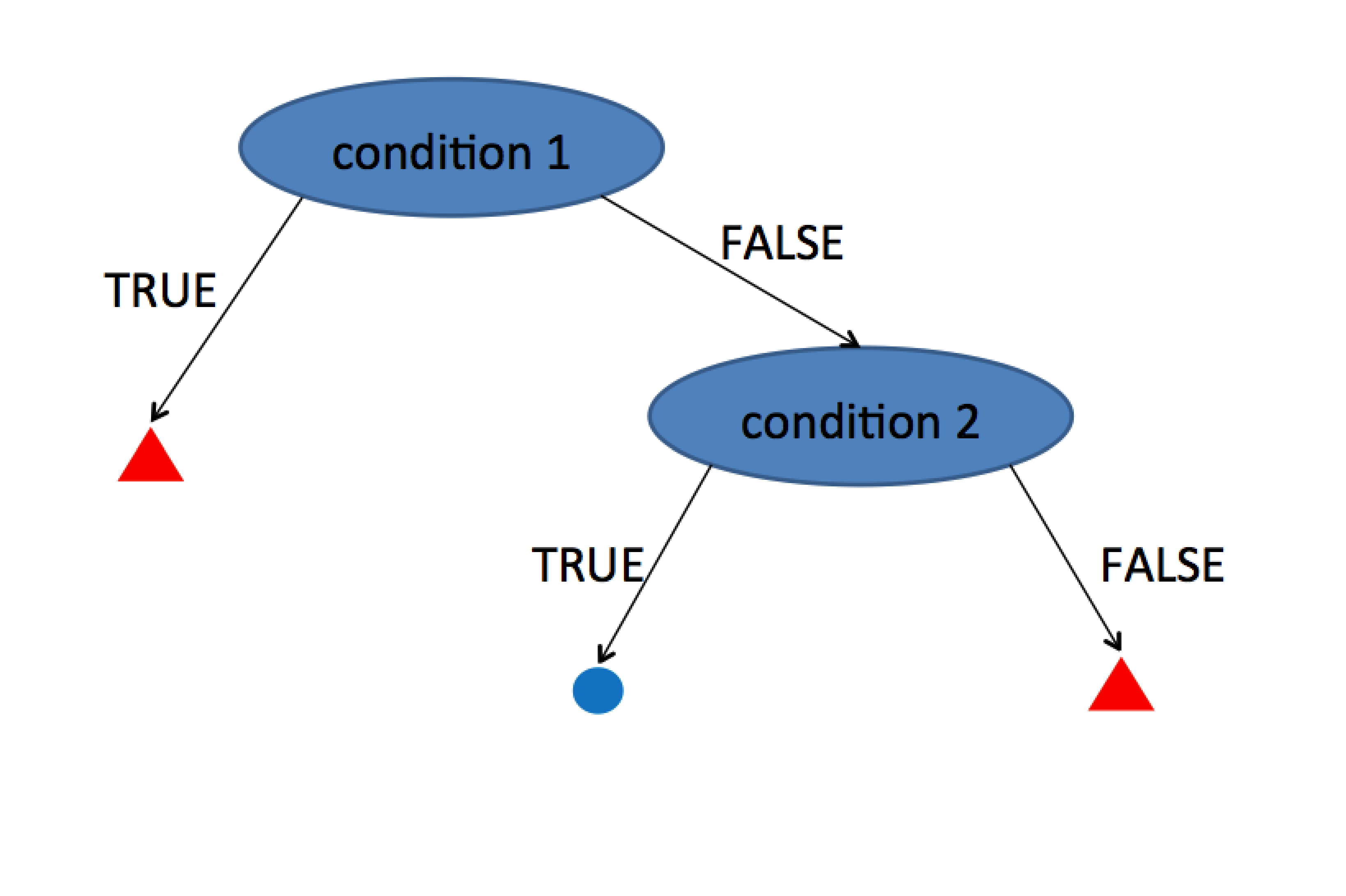


Deep Learning learns layers of features

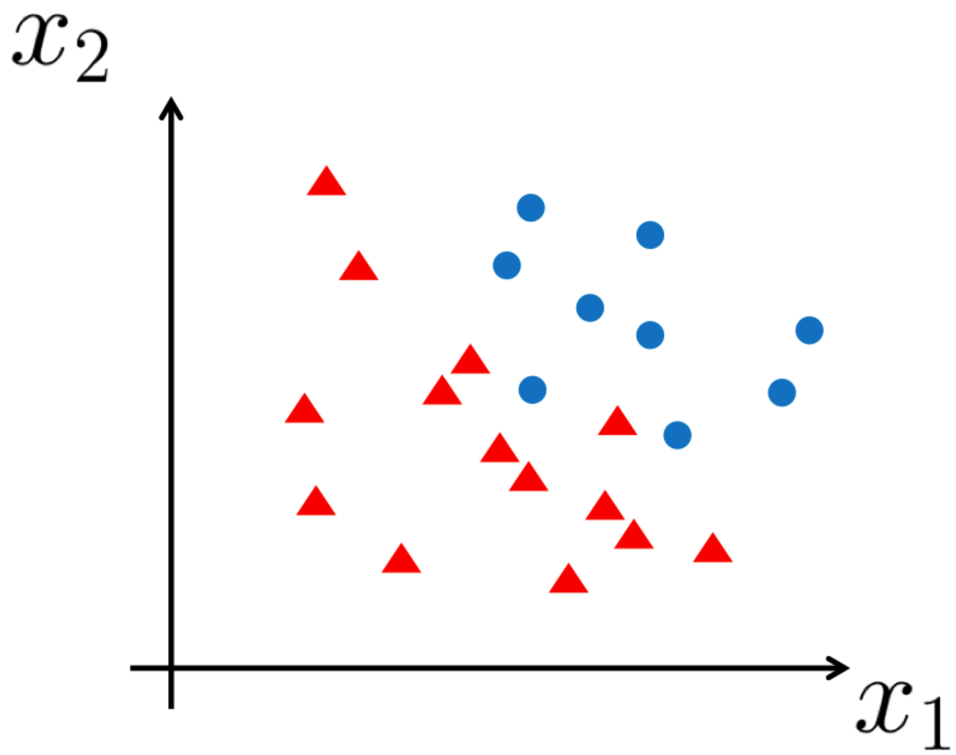


X MINUTES BREAK

DECISION TREES: IDEA

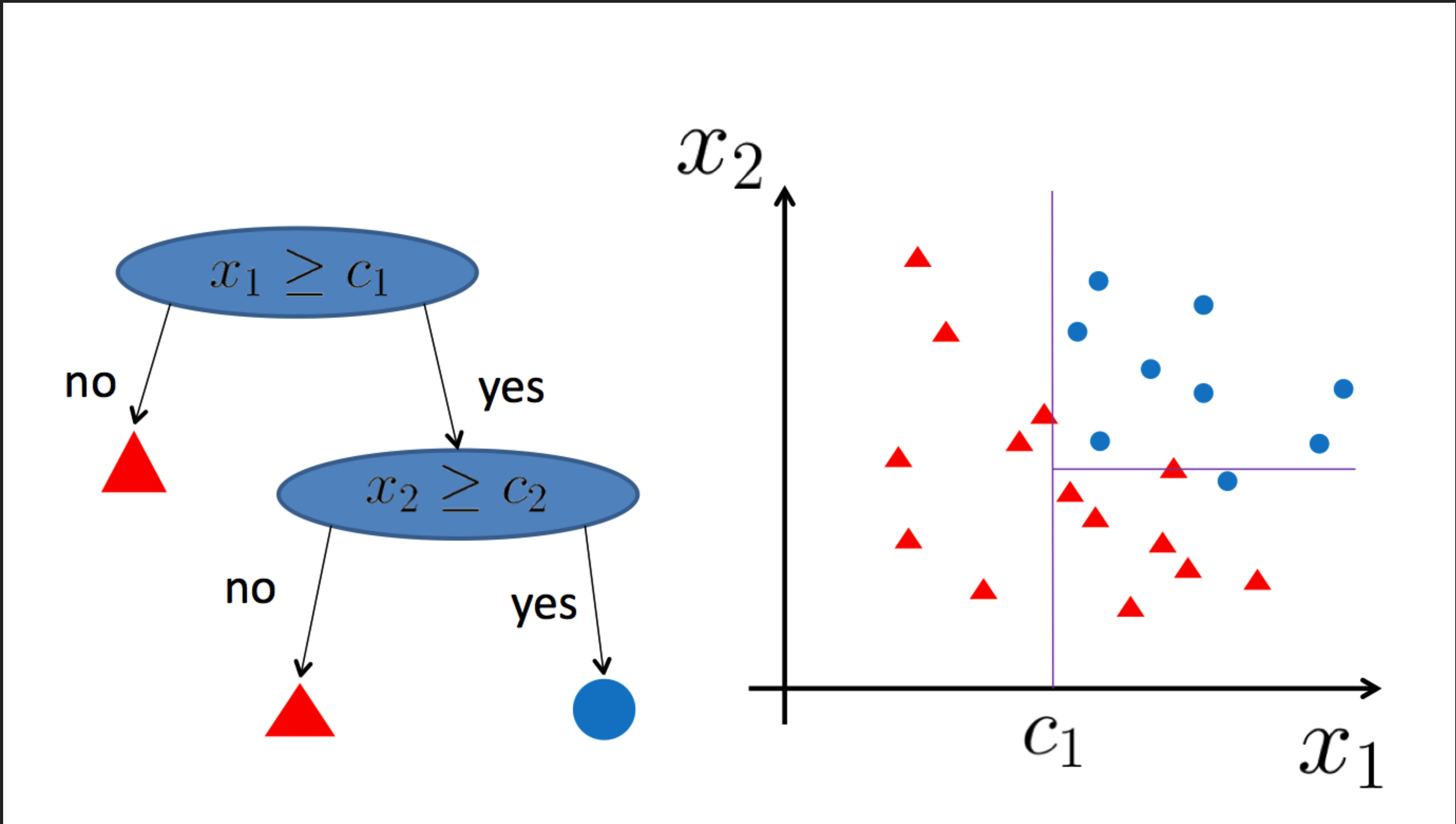


DECISION TREES



“Stump” conditions: x > c

DECISION TREES



DECISION TREE

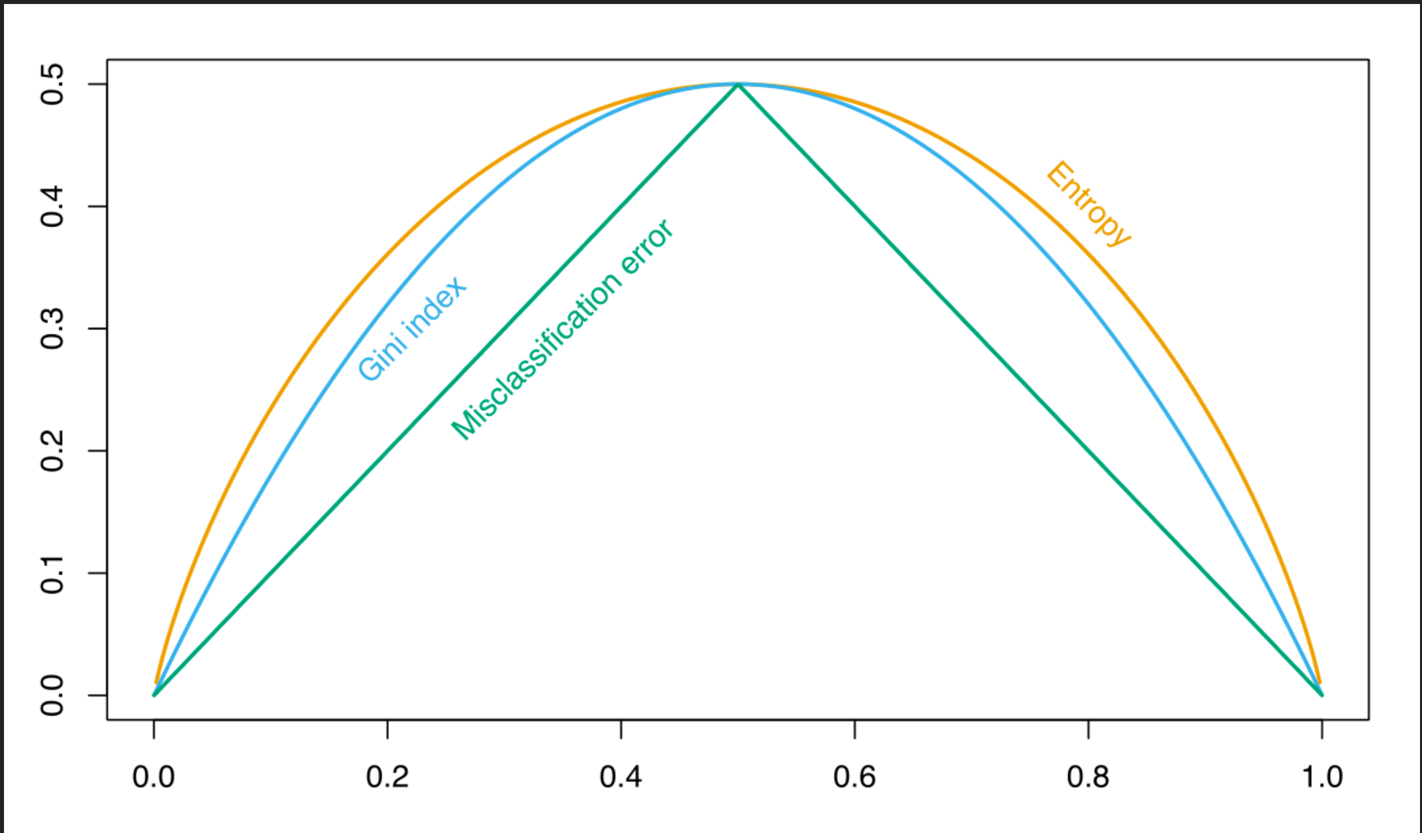
- fast & intuitive prediction
- building optimal decision tree is NP complete
- building tree from root using greedy optimization

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

SPLITTING CRITERIONS

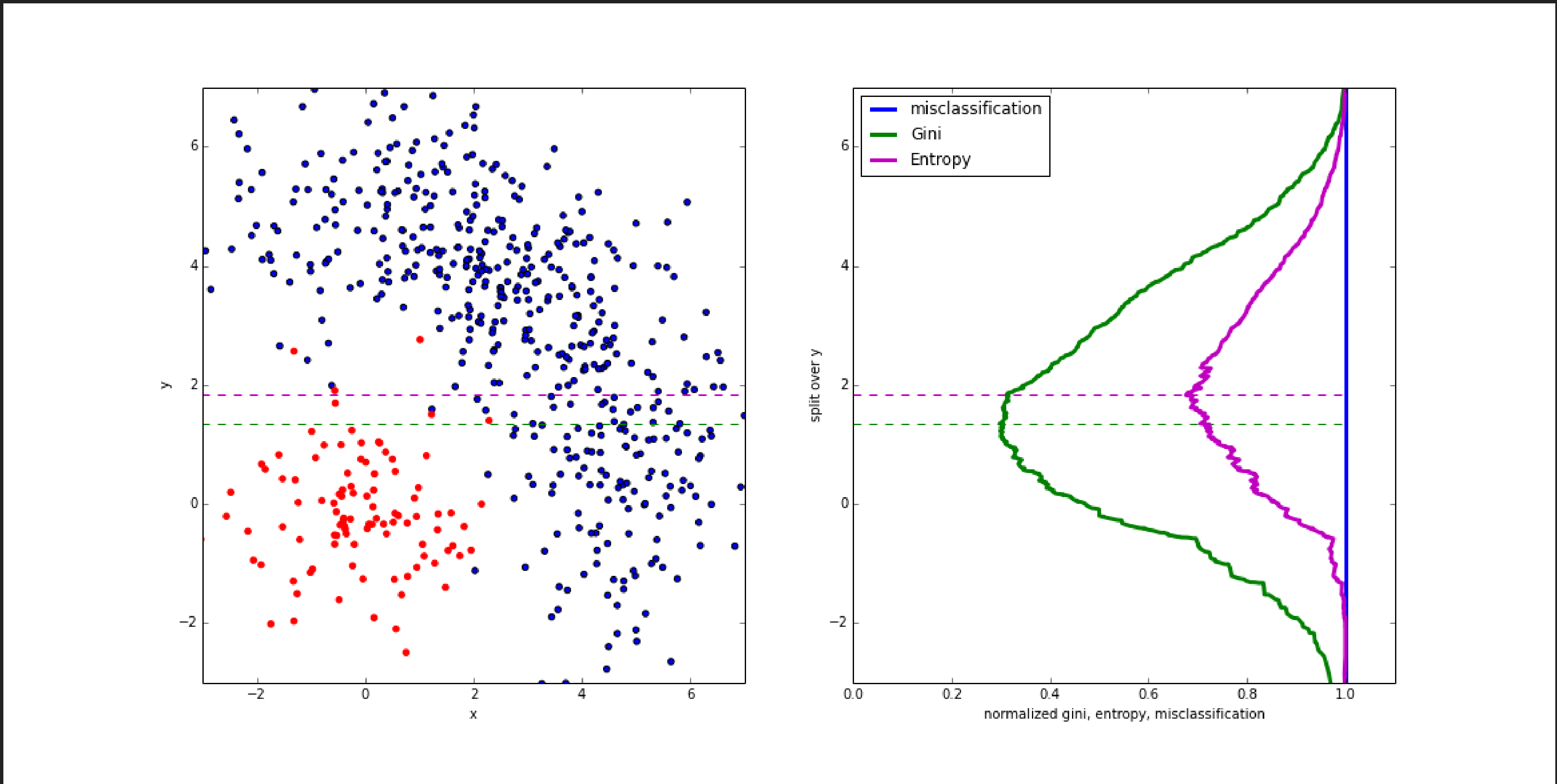
$$\text{TotalImpurity} = \sum_{\text{leaf}} \text{impurity}(\text{leaf}) \times \text{size}(\text{leaf})$$

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



SPLITTING CRITERIONS

Why using Gini or Entropy not misclassification?



REGRESSION TREE

Greedy optimization (minimizing MSE):

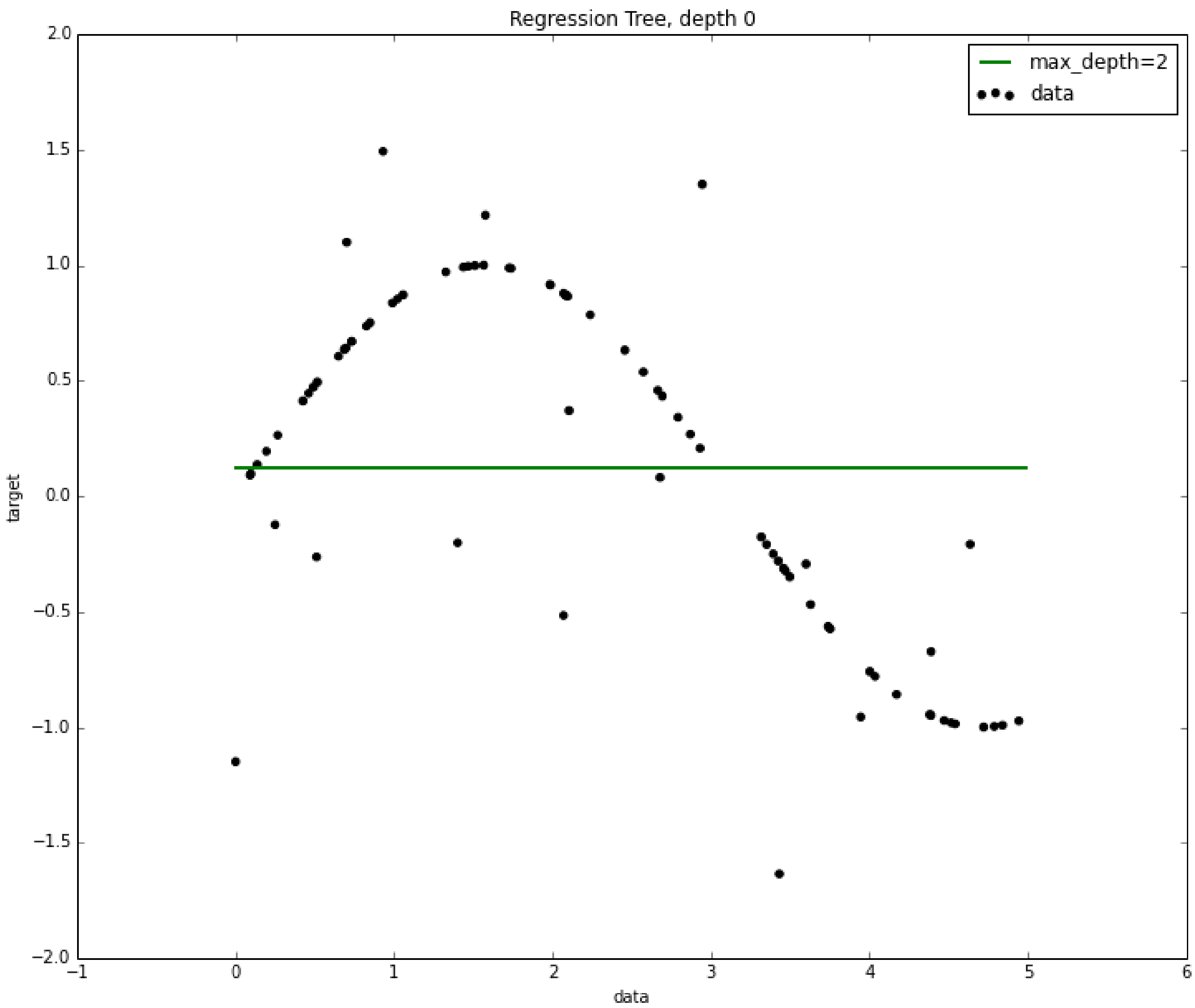
$$\text{GlobalMSE} \sim \sum_i (y_i - \hat{y}_i)^2$$

Can be rewritten as:

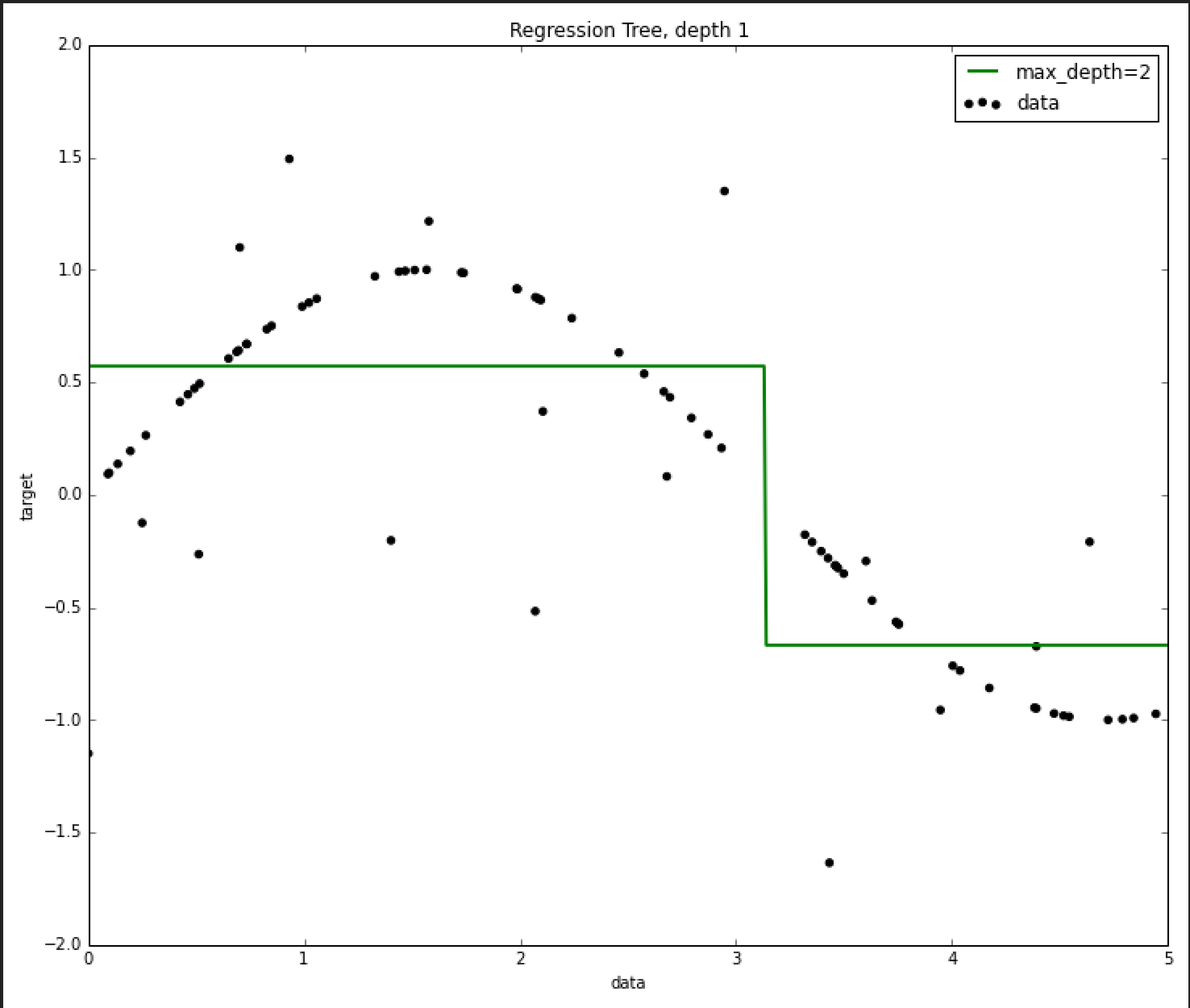
$$\text{GlobalMSE} \sim \sum_{\text{leaf}} \text{MSE}(\text{leaf}) \times \text{size}(\text{leaf})$$

MSE(leaf) is like 'impurity' of leaf

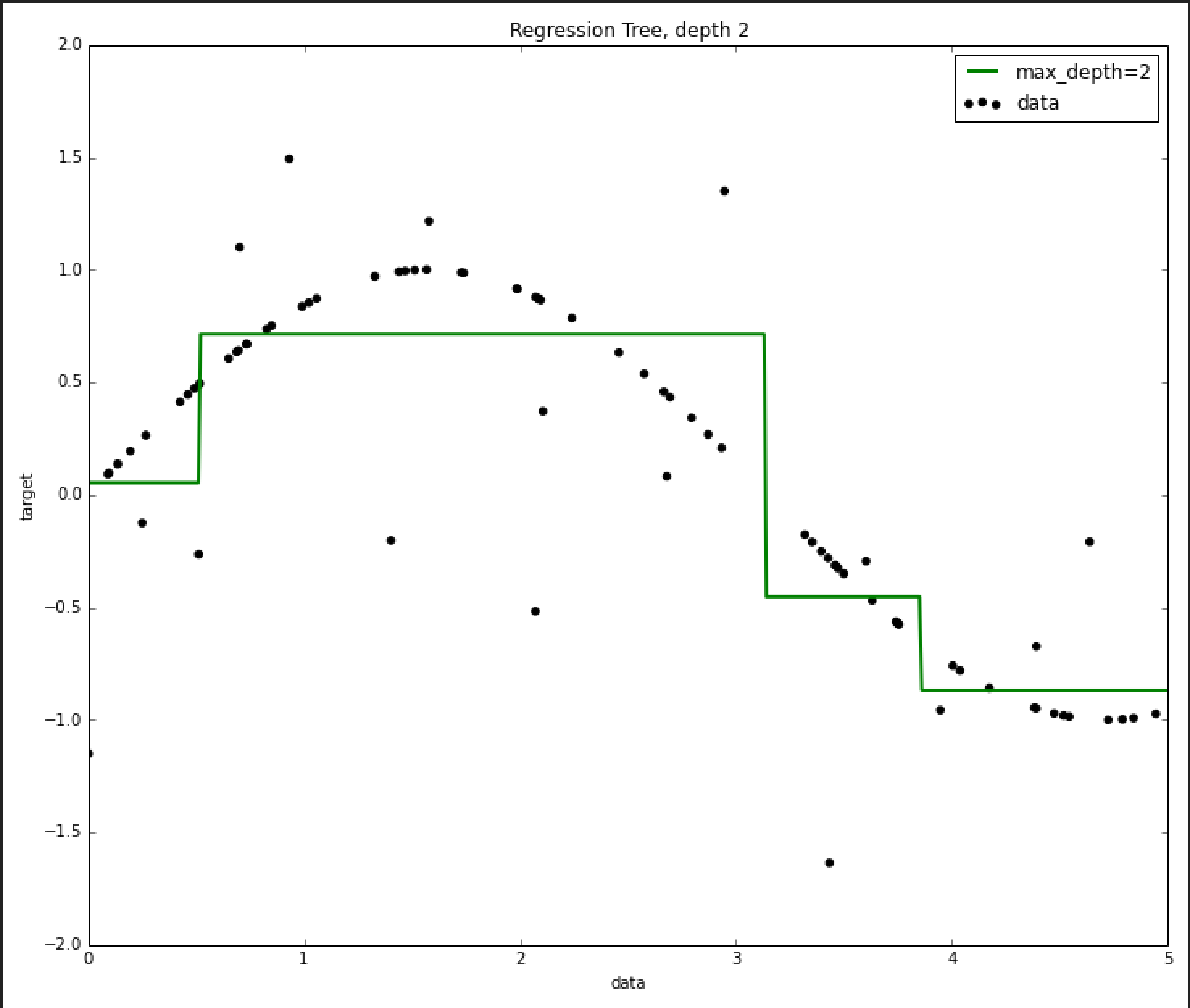
$$\text{MSE}(\text{leaf}) = \frac{1}{\text{size}(\text{leaf})} \sum_{i \in \text{leaf}} (y_i - \hat{y}_i)^2$$

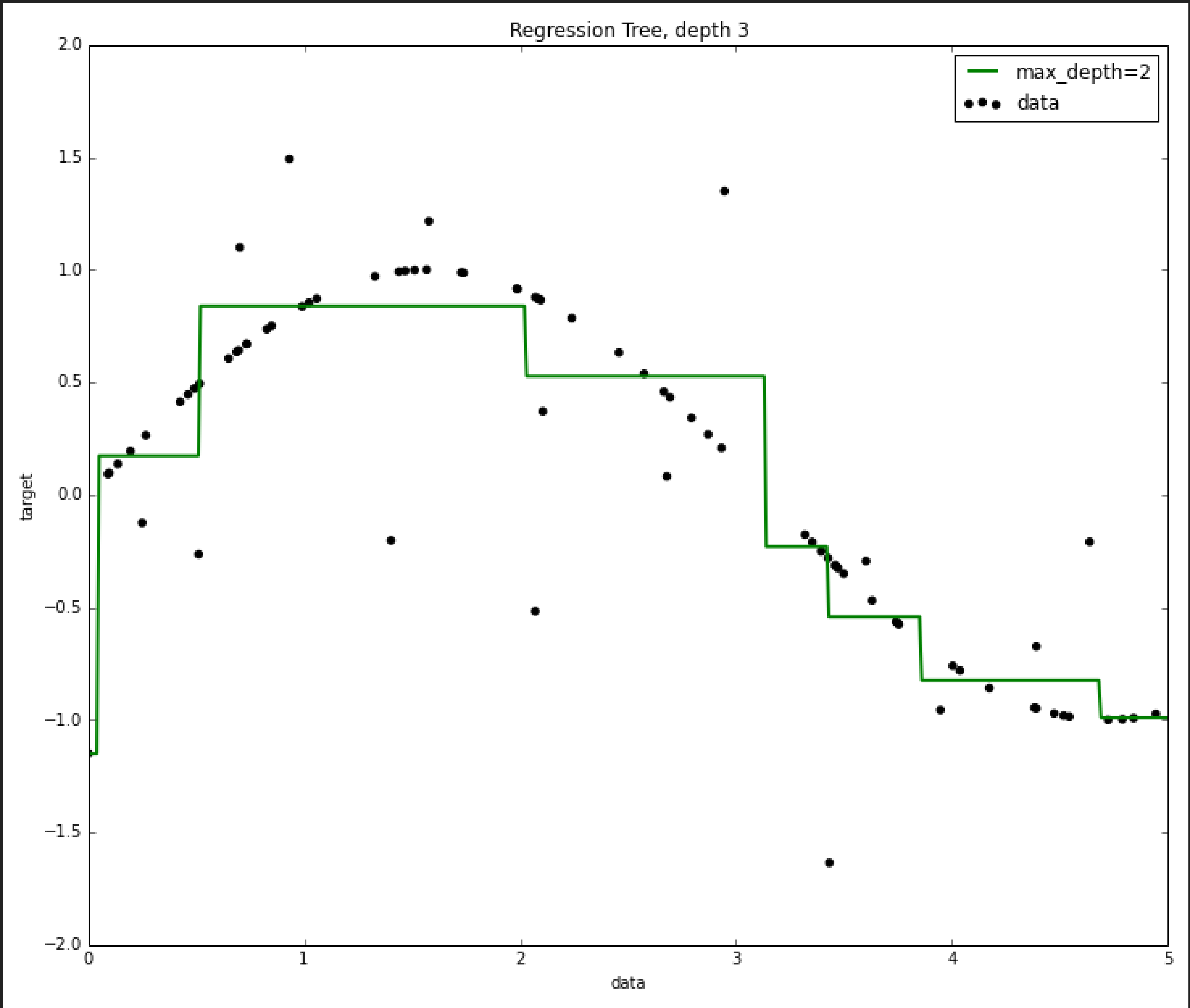


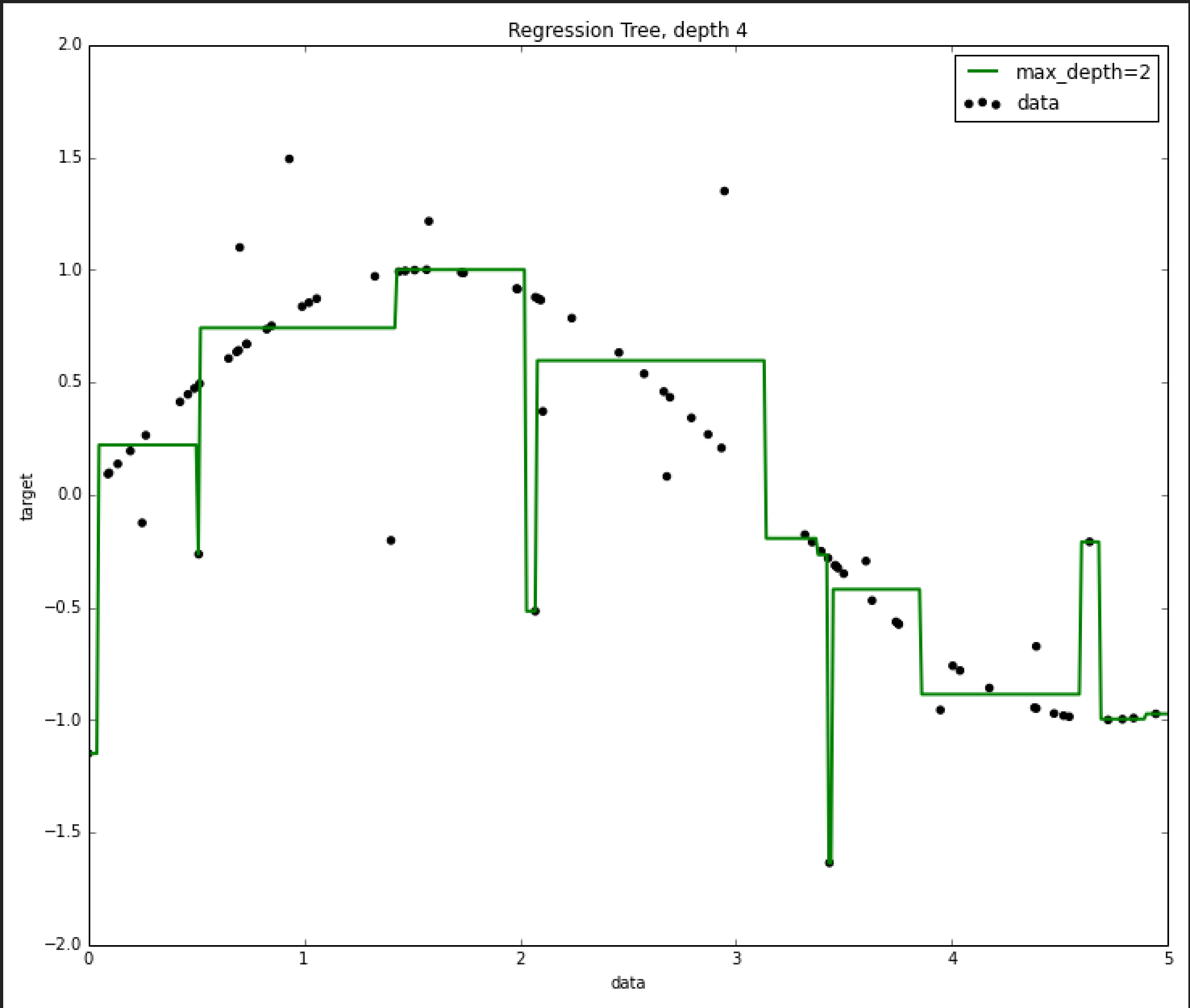
Regression Tree, depth 0
max_depth=2
data
target
data
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1
0
1
2
3
4
5
6



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
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
target
0
1
2
3
4
5
data


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:

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

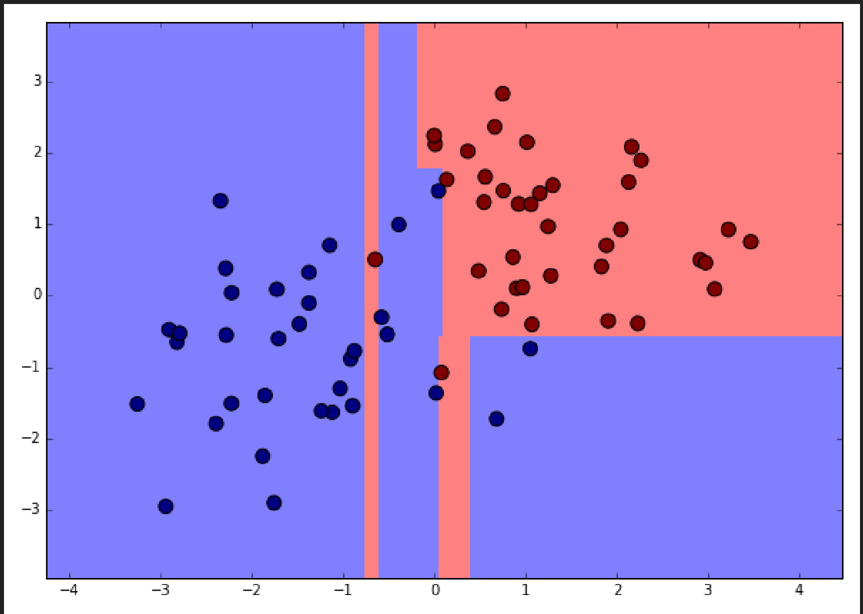
But other options also exist, i.e. MAE:

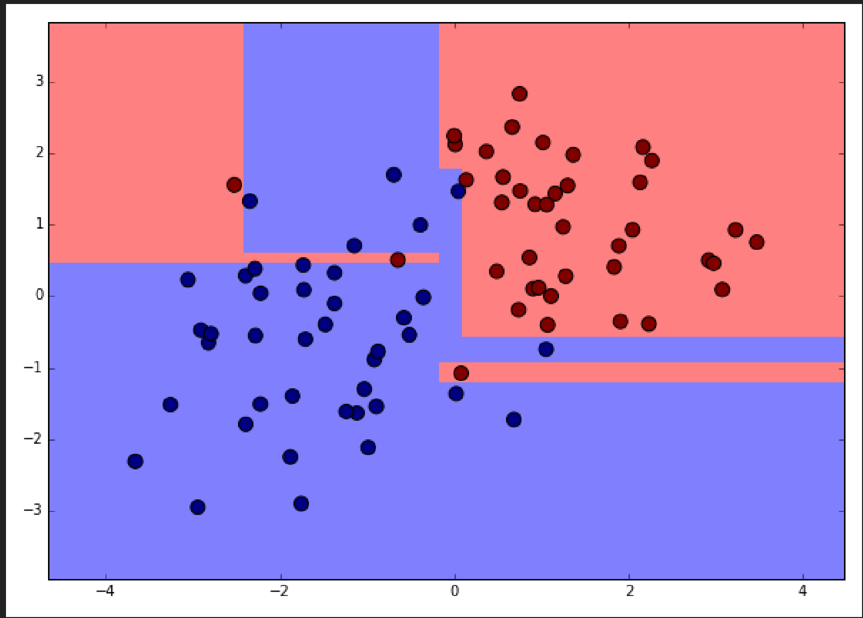
$$\text{GlobalMAE} \sim \sum_i |y_i - \hat{y}_i|$$

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

DECISION TREES INSTABILITY

Little variation in training dataset produce different classification rule.





Pre-Pruning of tree

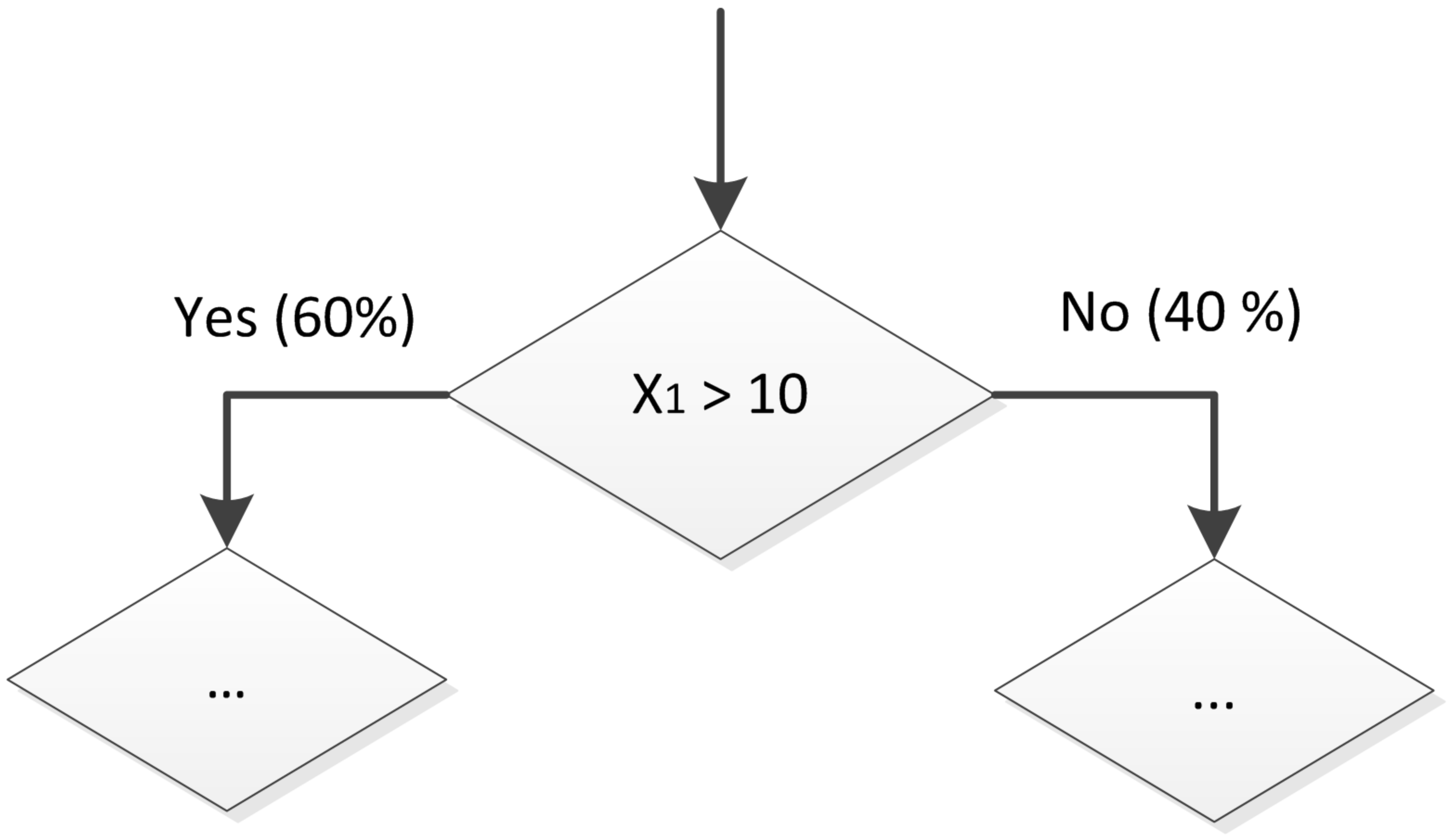
SUMMARY OF DECISION TREE

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



TODO Good code and reproducibility

THE END