Lecture 4:
Decision Tree and Neural Networks

http://cs.nju.edu.cn/yuy/course_dm12.ashx

Learning algorithms

Decision tree

Neural networks

Linear classifiers

Bayesian classifiers

Lazy classifiers

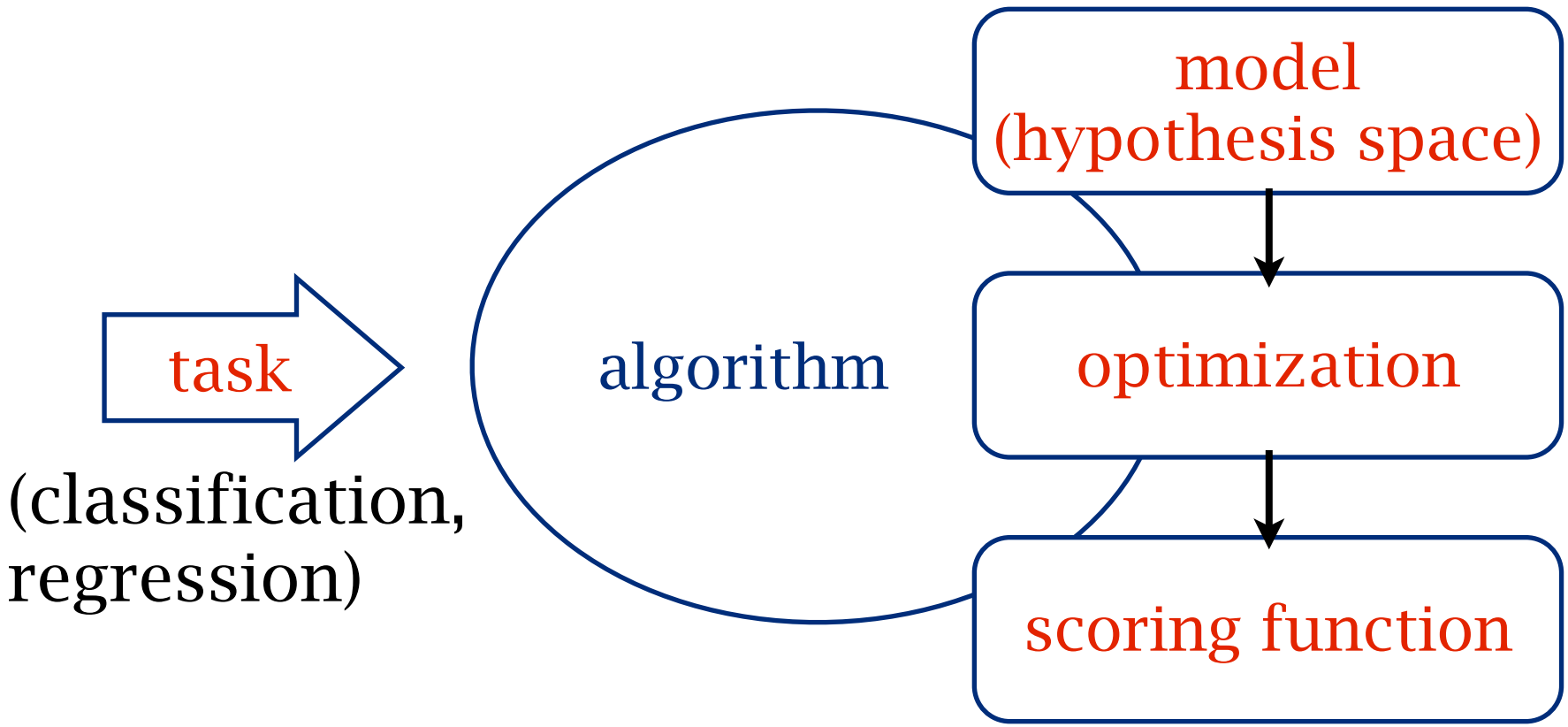
Why different classifiers?

- heuristics
- viewpoint
- performance

Ensemble methods

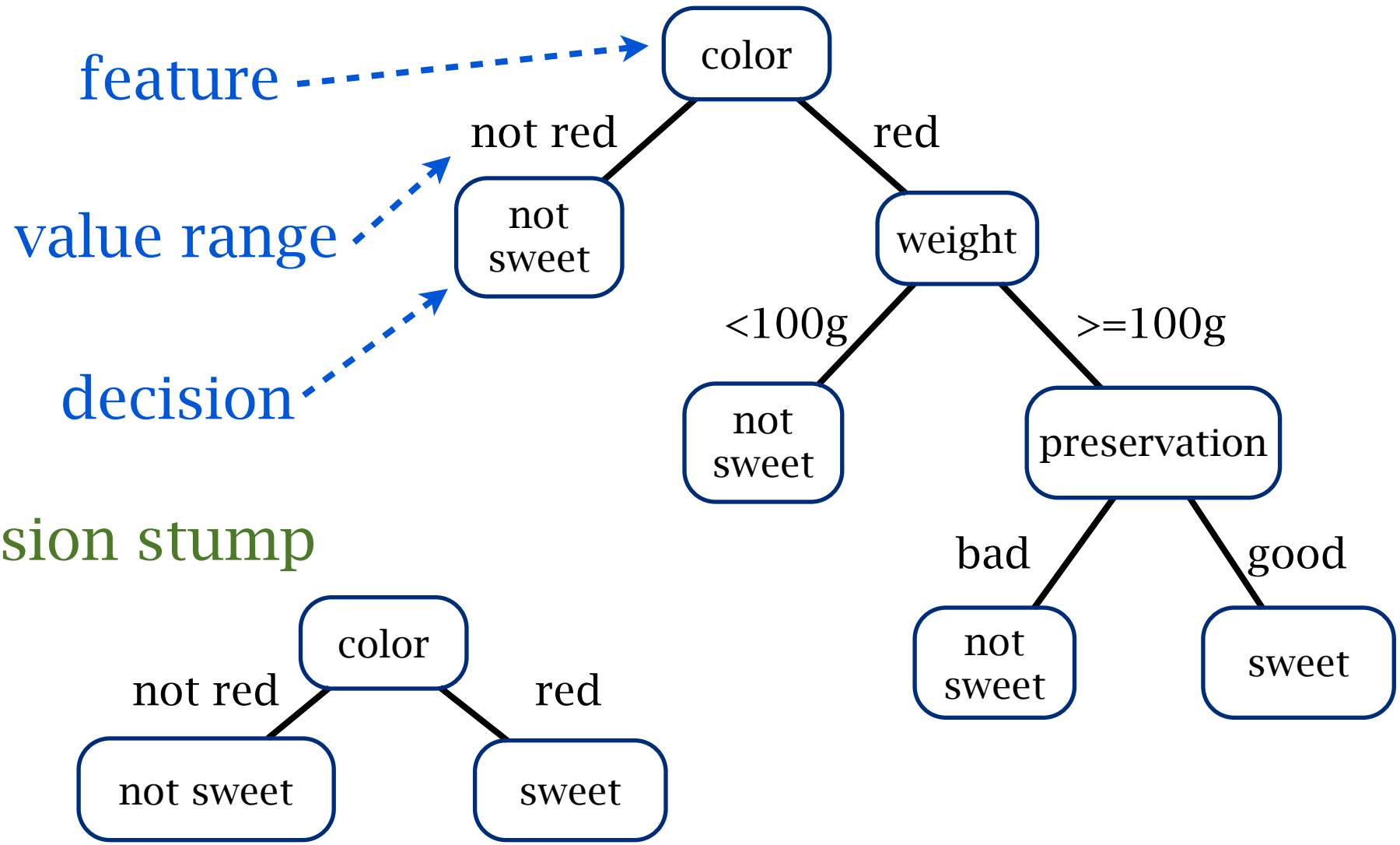
Handling big data

Learning algorithm components



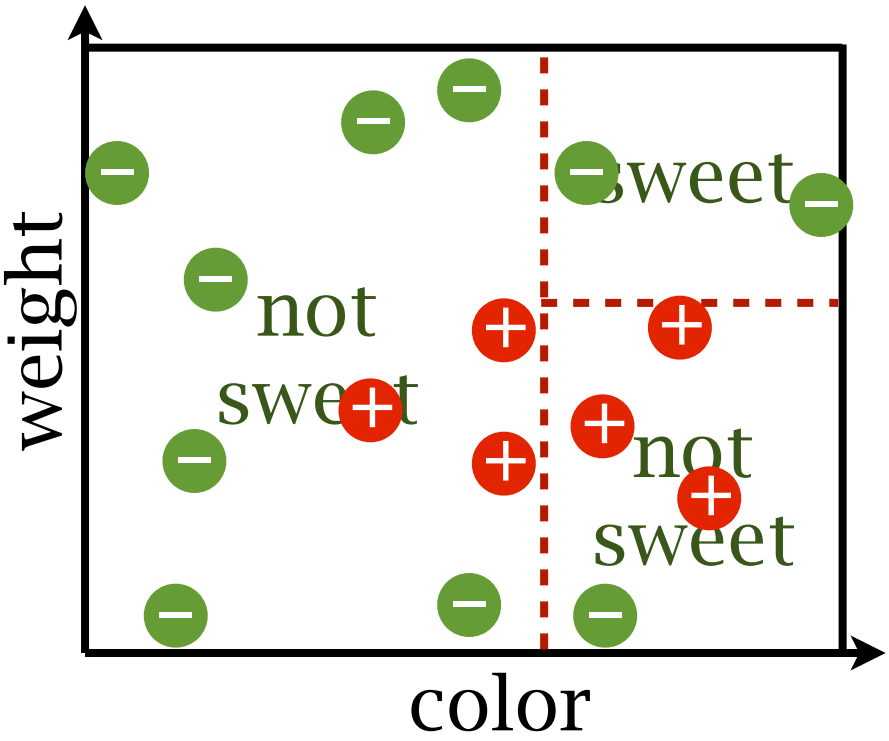
Decision tree model

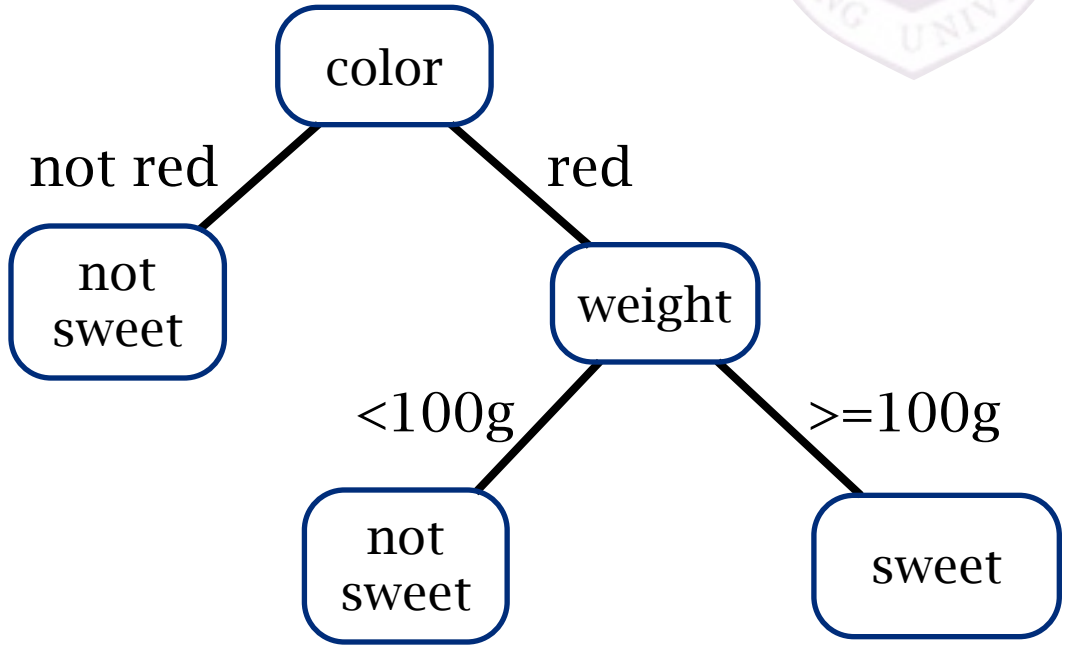
decision process
with a tree structure



decision stump

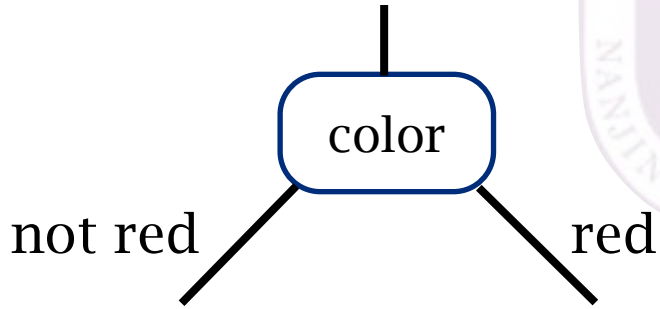
Decision tree model





find a decision tree that matches the data?

Top-down induction



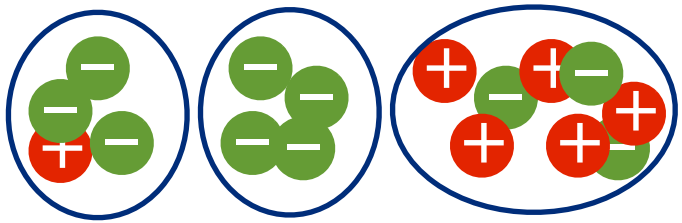
function construct-node(*data*) :

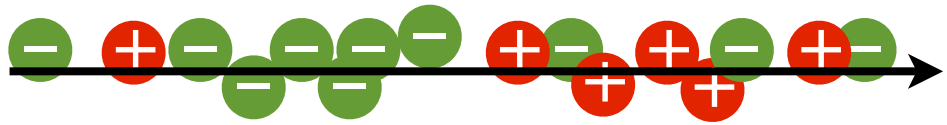
1. *feature*, *value* ←**split-criterion** (*data*)
2. if feature is valid
3. *subdata*[] ← split(*data*, *feature*, *value*)
4. for each branch *i*
5. construct-node (*subdata*[*i*])
6. else
7. **make a leaf**
8. return

divide and conquer

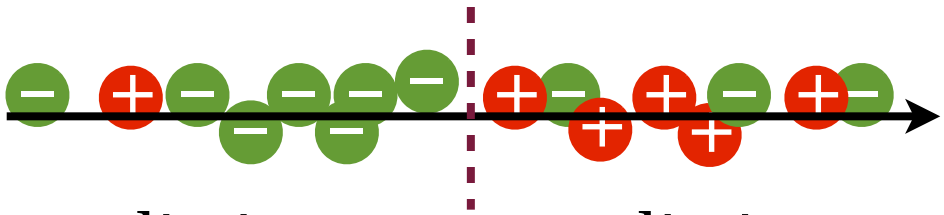
Split-criterion: classification

for every possible split of every feature:





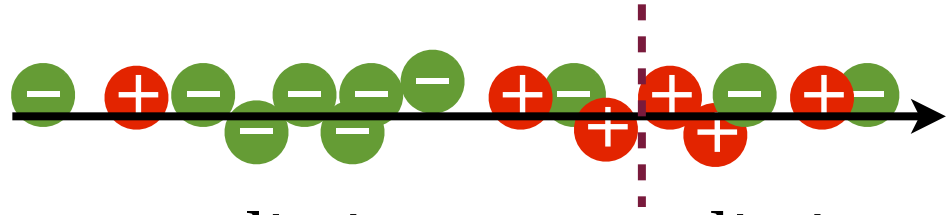
Training error:



prediction: –
error: 1

prediction: +
error: 3

total error: 4

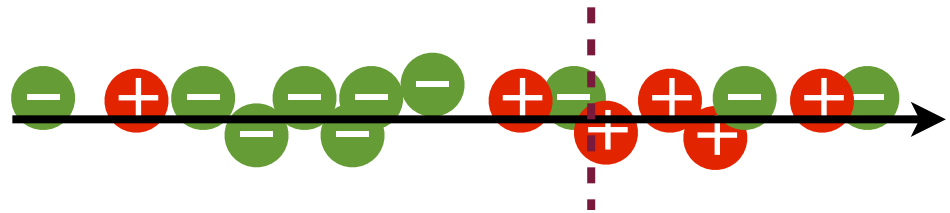


prediction: –
error: 3

prediction: +
error: 2

total error: 5

total error: 4



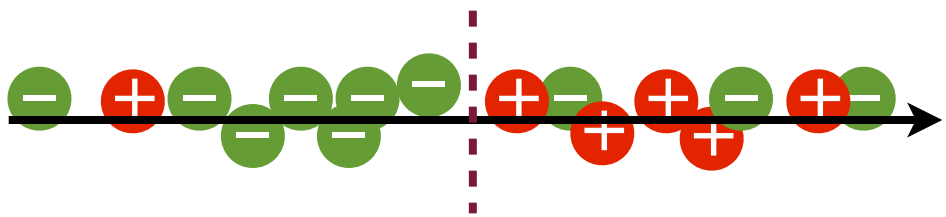
Split-criterion: classification

Information gain (ID3):

Entropy: $H(X) = -\sum_i p_i \ln(p_i)$

Entropy after split: $I(X; \text{split}) = \frac{\#\text{left}}{\#\text{all}} H(\text{left}) + \frac{\#\text{right}}{\#\text{all}} H(\text{right})$

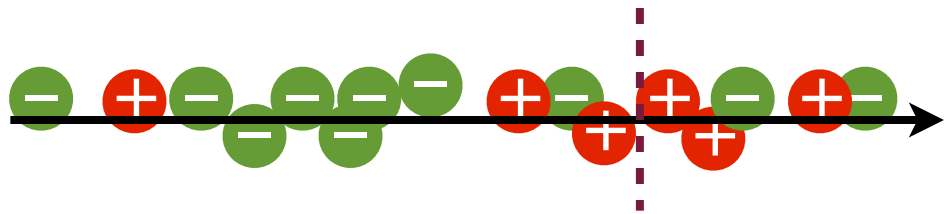
Information gain: $H(X)$-$I(X;\text{split})$



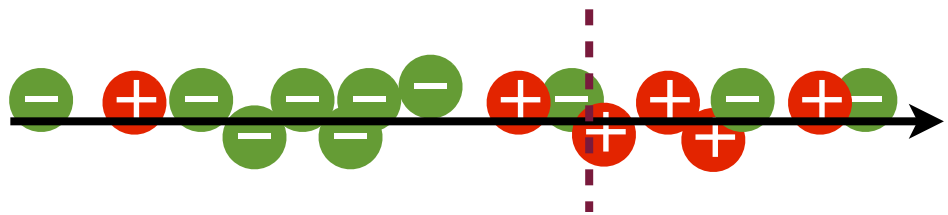
$$H(\text{left}) = -\frac{1}{8}\ln\frac{1}{8} - \frac{7}{8}\ln\frac{7}{8} = 0.3768$$

$$H(\text{right}) = -\frac{5}{8}\ln\frac{5}{8} - \frac{3}{8}\ln\frac{3}{8} = 0.6616$$

$$\begin{aligned}\text{IG} &= H(X) - (0.5 \times 0.3768 + 0.5 \times 0.6616) \\ &= H(X) - 0.5192\end{aligned}$$



$$\text{IG} = H(X) - 0.6132$$



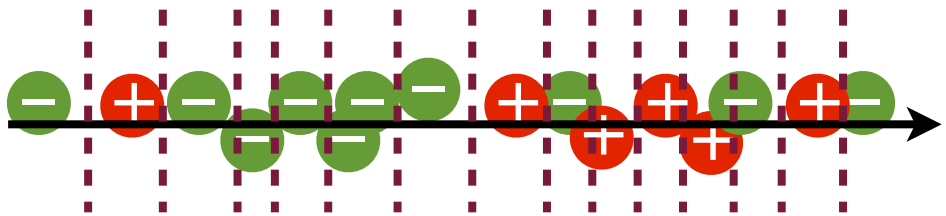
$$\text{IG} = H(X) - 0.5514$$

Split-criterion: classification

Gain ratio (C4.5):

$$\text{Gain ratio}(X) = \frac{H(X) - I(X;\text{split})}{IV(\text{split})}$$

$$IV(\text{split}) = H(\text{split})$$



e.g. student ID

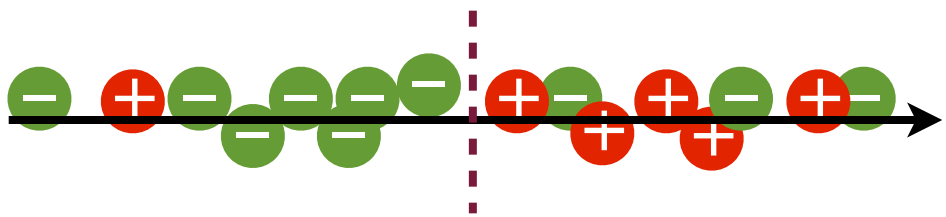
$$\text{IG} = H(X) - 0$$

Split-criterion: classification

Gini index (CART):

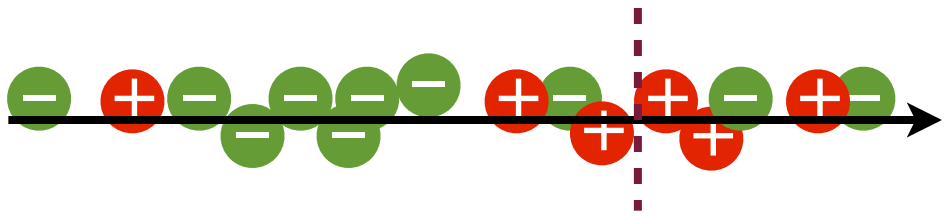
Gini: $Gini(X) = 1 - \sum_i p_i^2$

Gini after split: $\frac{\#\text{left}}{\#\text{all}} Gini(\text{left}) + \frac{\#\text{right}}{\#\text{all}} Gini(\text{right})$



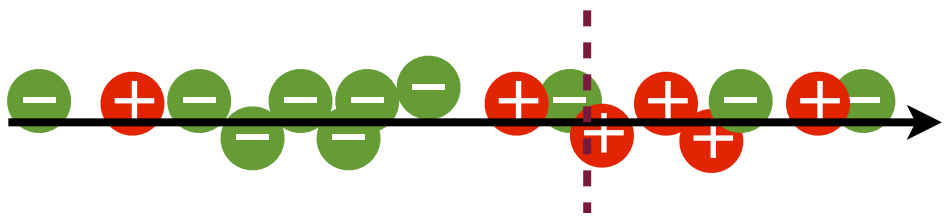
$\text{IG} = H(X) - 0.5192$

$Gini = 0.3438$



$\text{IG} = H(X) - 0.6132$

$Gini = 0.4427$

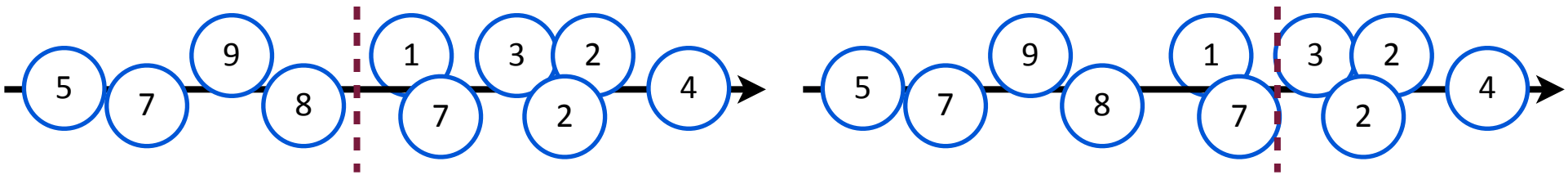


$\text{IG} = H(X) - 0.5514$

$Gini = 0.3667$

Split-criterion: regression

Training error:



mean: 7.25 mean: 3.1667

MSE: 8.75+22.83=31.583

mean: 6.1667 mean: 2.75

MSE: 43.5833

Split-criterion: stop

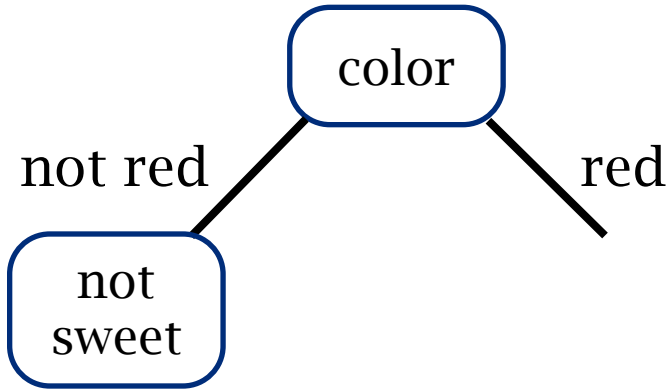
Stop criterion:

no feature to use

Classification: examples are pure of class

Regression: variance small enough

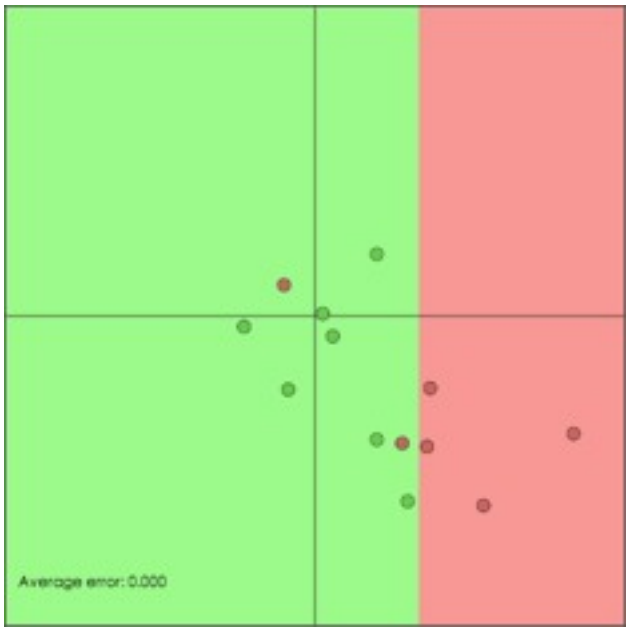
Make-leaf



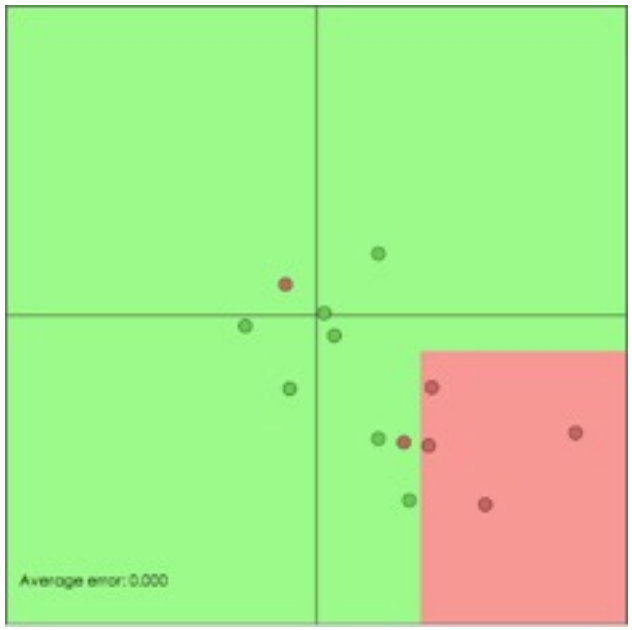
Classification: major class

Regression: mean value

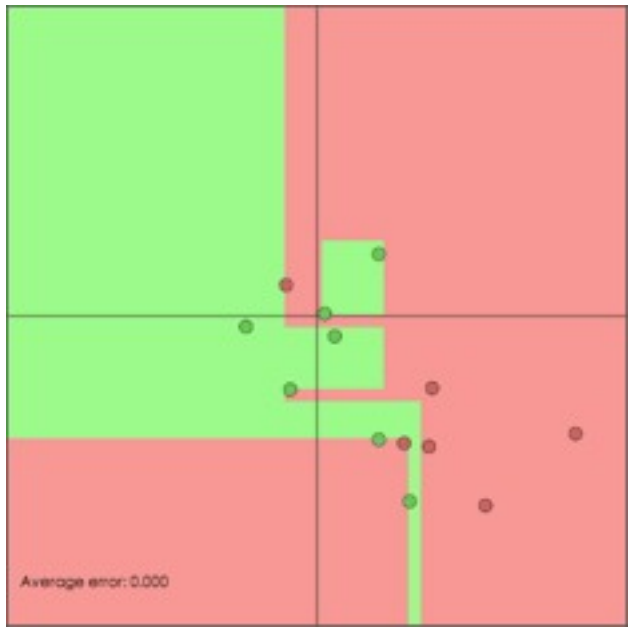
DT boundary visualization



decision stump



max depth=2



max depth=12

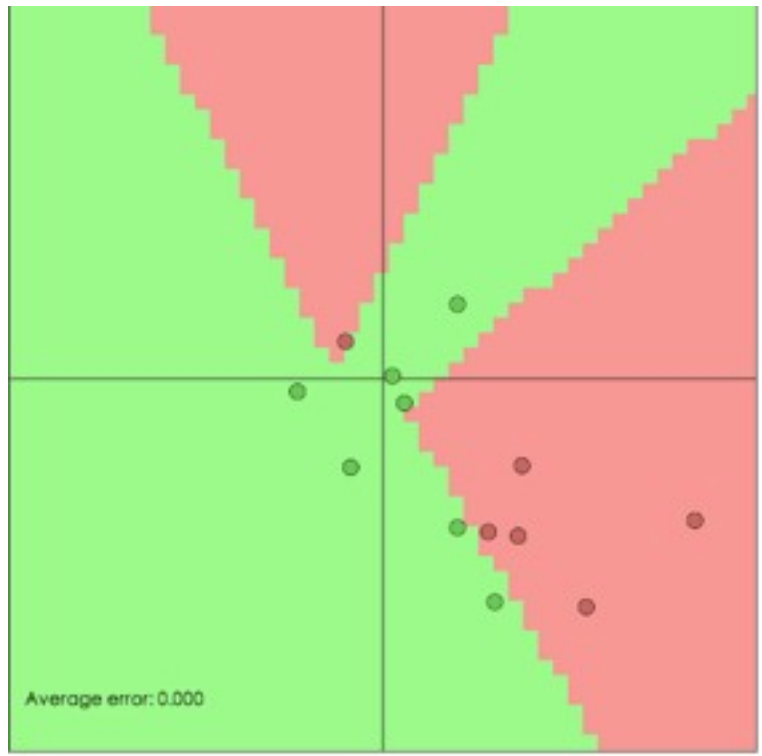
Oblique decision tree

choose a linear combination in each node:

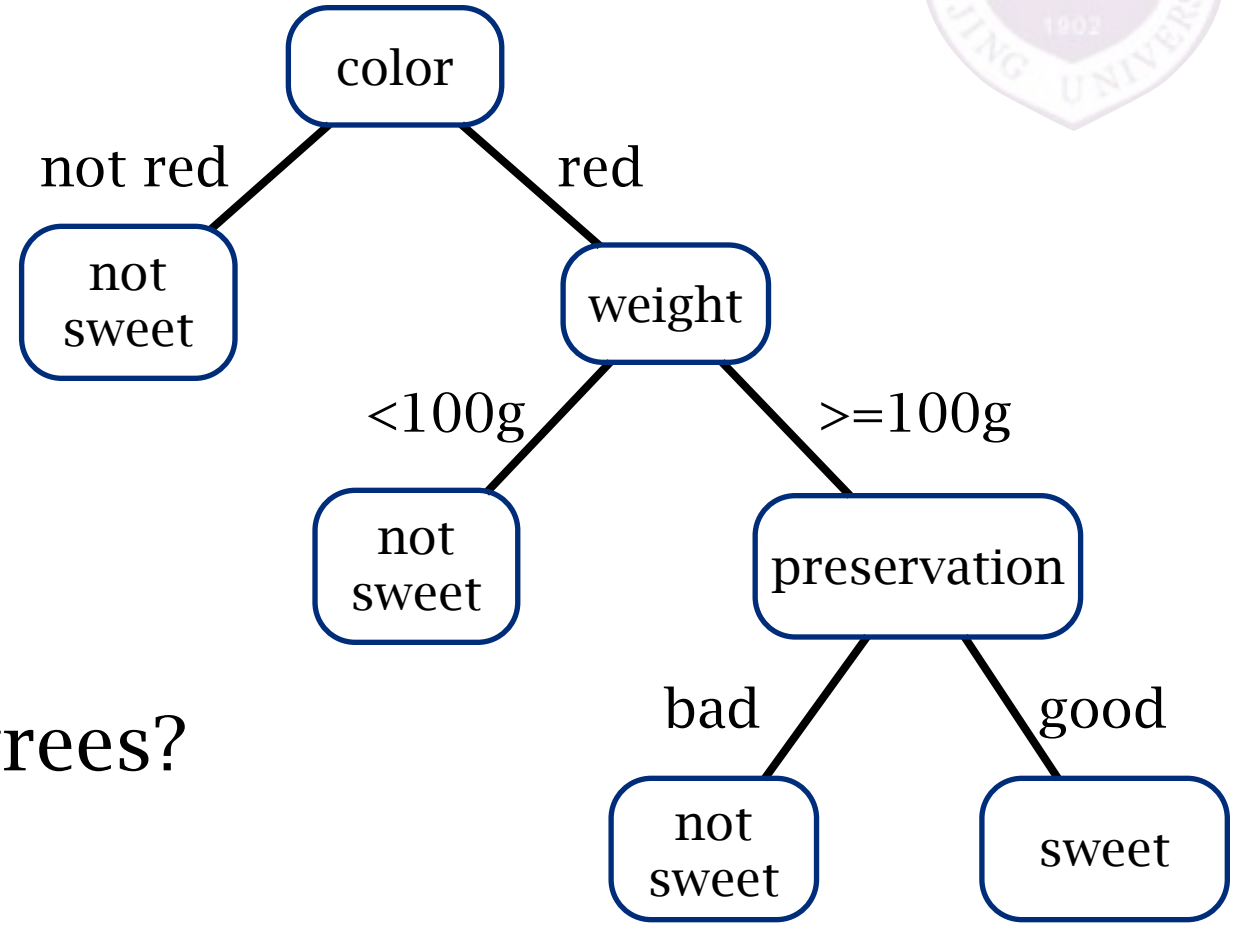
axis parallel:
$X_1 > 0.5$

oblique:
$0.2\ X_1 + 0.7\ X_2 + 0.1\ X_3 > 0.5$

hard to train



Hypothesis space complexity



features: n
feature type: binary
depth: $d<n$

How many different trees?

$$< n^{2^d-1}$$

$$> \frac{n!}{(n-d)!} > (\frac{n}{n-d})^n (\frac{n-d}{e})^d$$

hypothesis space complexity grows very fast with d

Pruning

To make decision tree less complex

Pre-pruning: early stop

- minimum data in leaf
- maximum depth
- maximum accuracy

Post-pruning: prune full grown DT

1. divide a validation set from training set
2. cut a leaf if the validation error does not grow

Advantages

Fast to train

samples: m
features: n
feature splits: k
depth: $d<n$

training time:

one node: $O(mkn)$

d depth tree: $O(2^d mkn)$

full tree: $O(m^2 kn)$

Fast to test

not all features are tested

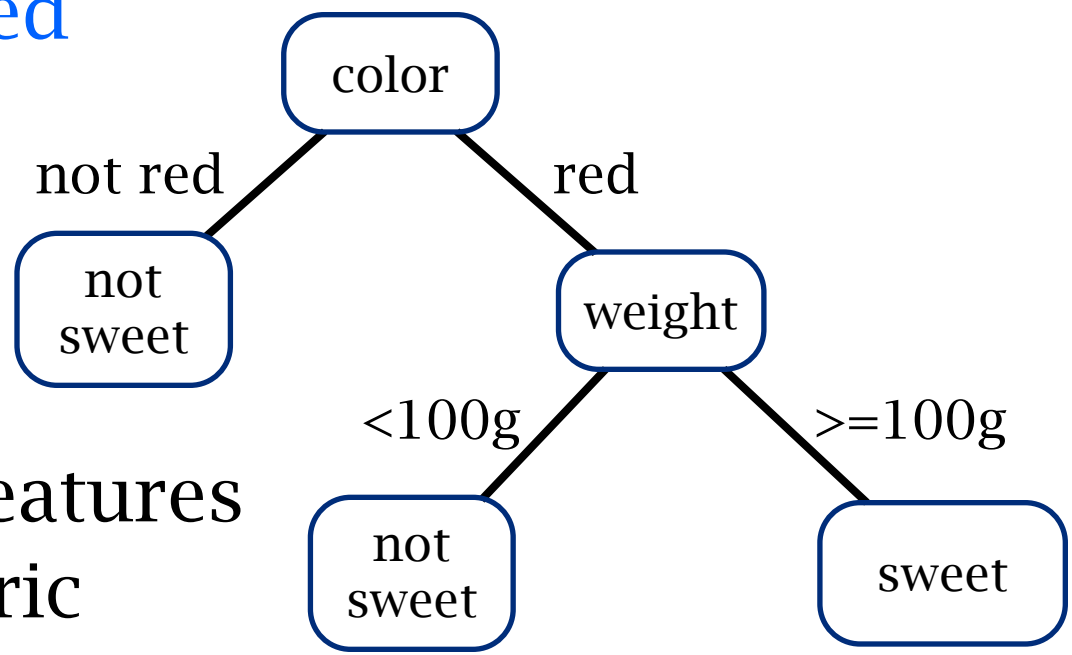
Regression/Classification

Multi-class

Comprehensibility

Nominal and numerical features

Non-parametric, non-metric



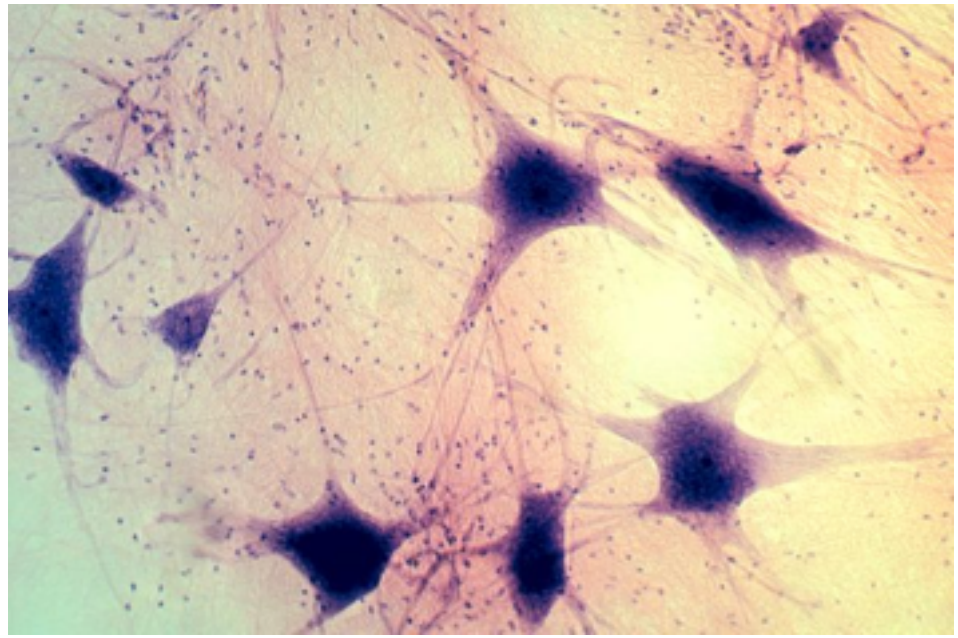
Summary: decision tree

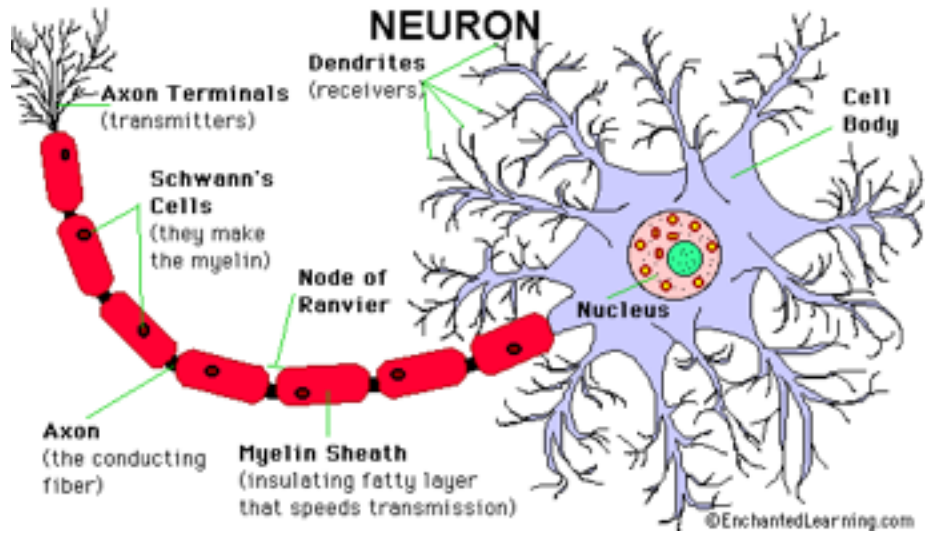
model: tree structure

scoring function: 0-1 loss / approximate criteria

optimization: greedy search

Neural networks





Neuron / perceptron

output a function of sum of input

linear function:

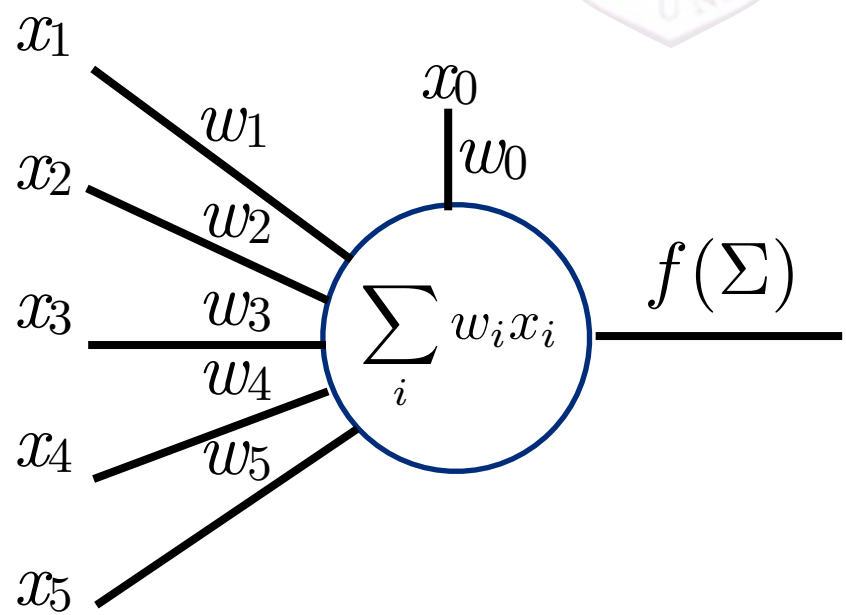
$$f(\sum_i w_i x_i) = \sum_i w_i x_i$$

threshold function:

$$f(\sum_i w_i x_i) = I(\sum_i w_i x_i > 0)$$

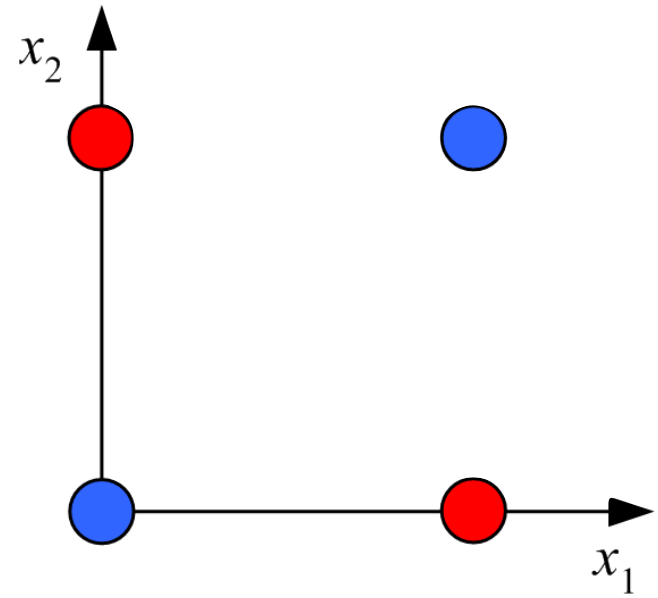
sigmoid function:

$$f(\sum_i w_i x_i) = \frac{1}{1 + e^{-\Sigma}}$$



Limitation of single neuron

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0



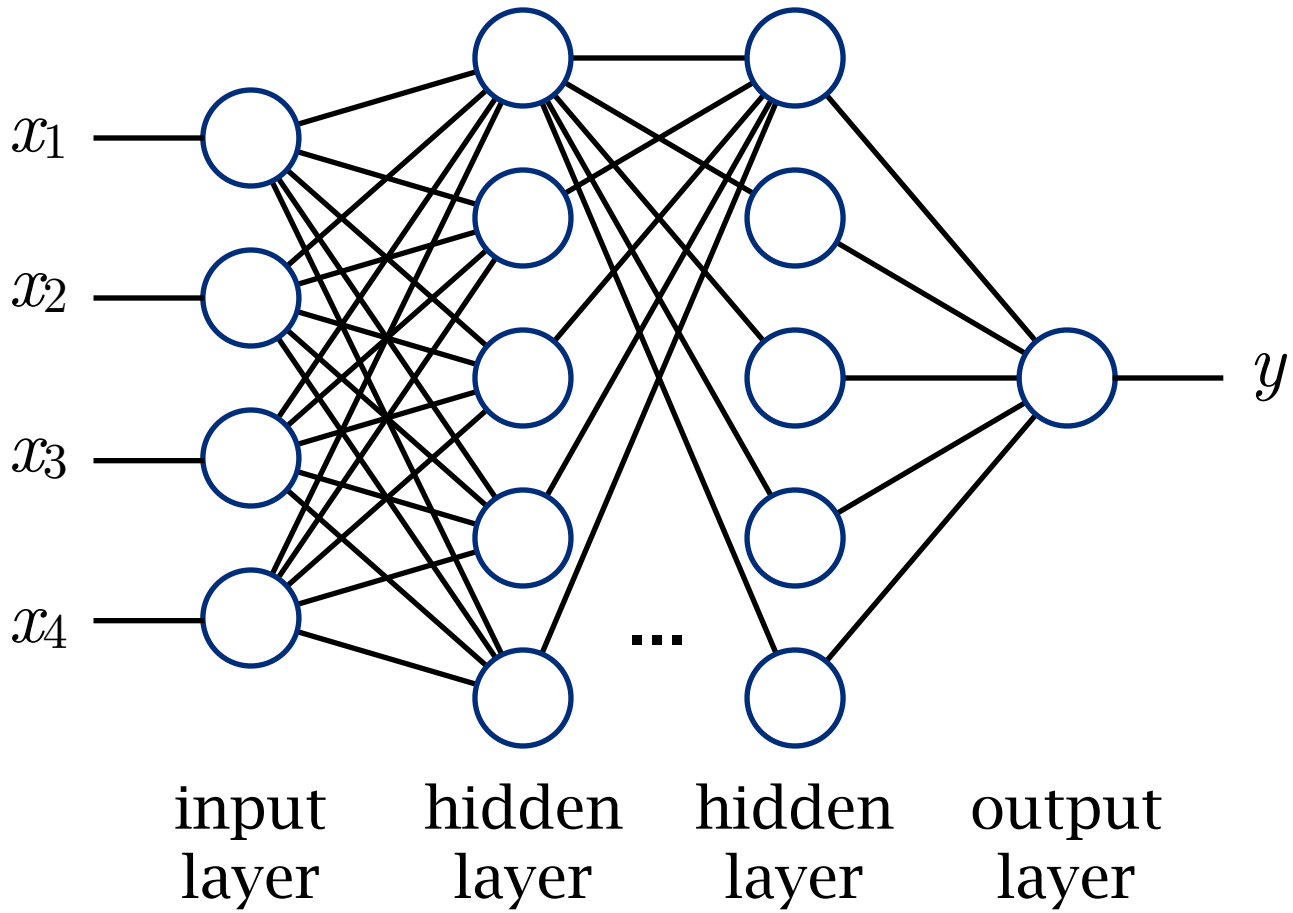
[Minsky and Papert, *Perceptrons,* 1969]

Marvin Minsky
Turing Award 1969

AI Winter

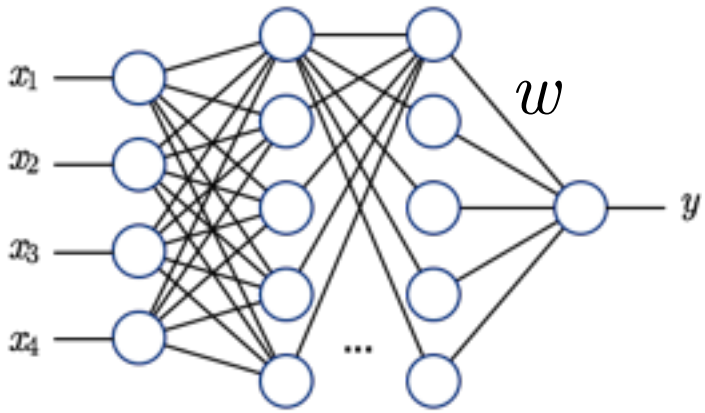
Multi-layer perceptrons

feed-forward network



sigmoid network with one hidden layer can approximate arbitrary function [Cybenko 1989]

Back-propagation algorithm



$$\hat{y} = F(\boldsymbol{x})$$

gradient descent

error: $E(\boldsymbol{w}) = (F(\boldsymbol{x}) - y)^2$

update one weight: $\Delta w_{i,j} = -\eta \dfrac{\partial E(\boldsymbol{w})}{\partial w_{i,j}}$

weight of the laster layer

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}} = \frac{\partial E(\boldsymbol{w})}{\partial F(\boldsymbol{x})} \frac{\partial F(\boldsymbol{x})}{\partial w_{i,j}}$$

weight of the first layer

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}} = \frac{\partial E(\boldsymbol{w})}{\partial F(\boldsymbol{x})} \frac{\partial F(\boldsymbol{x})}{\partial \mathrm{HL2}} \frac{\partial \mathrm{HL2}}{\partial \mathrm{HL1}} \frac{\partial \mathrm{HL1}}{\partial w_{i,j}}$$

[Rumelhart, Hinton, Williams, Nature 1986]

Back-propagation algorithm

For each given training example (**x**, **y**), **do**

1. Input the instance **x** to the NN and compute the output value o_u of every output unit u of the network

2. **For each** network output unit k, calculate its error term δ_k

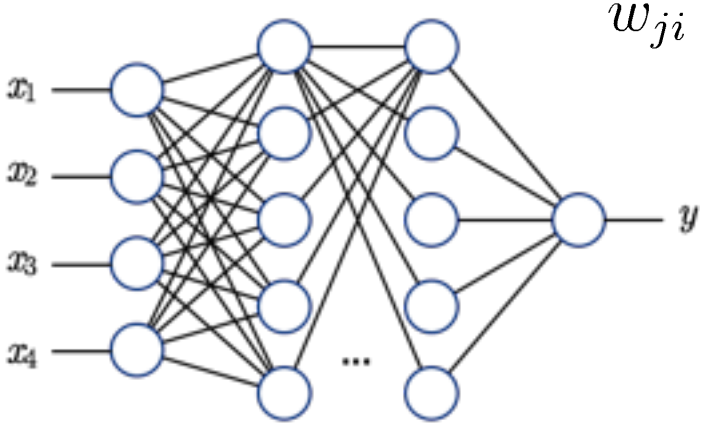
$$\delta_k \leftarrow o_k(1 - o_k)(y_k - o_k)$$

3. **For each** hidden unit k, calculate its error term δ_h

$$\delta_h \leftarrow o_k(1 - o_k) \sum_{k \in outputs} w_{kh}\delta_k$$

4. Update each network weight w_{ji} which is the weight associated with the i-th input value to the unit j

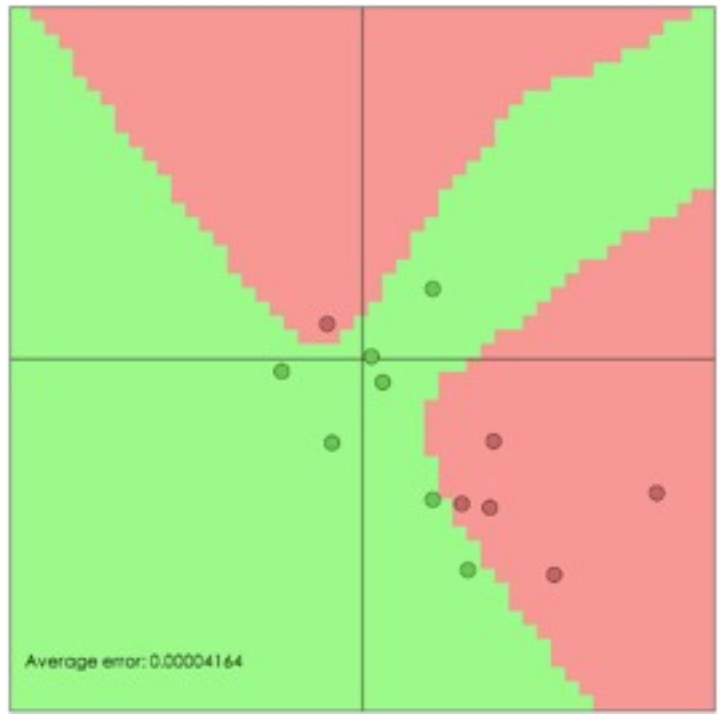
$$w_{ji} \leftarrow w_{ji} + \eta\delta_j x_{ji}$$



[Rumelhart, Hinton, Williams, Nature 1986]

Advantage and disadvantages

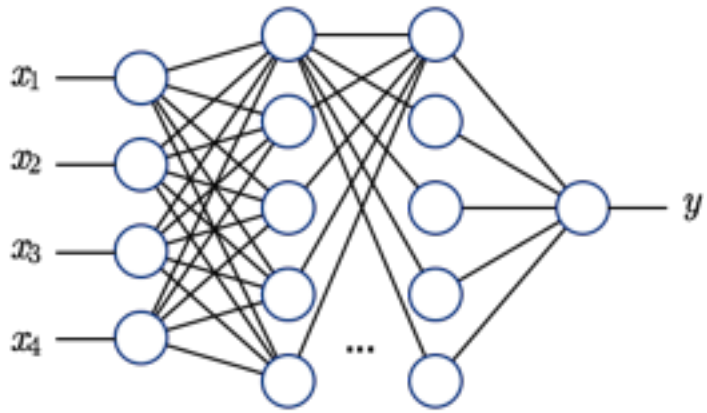
Smooth and nonlinear decision boundary



Slow convergence

Many local optima

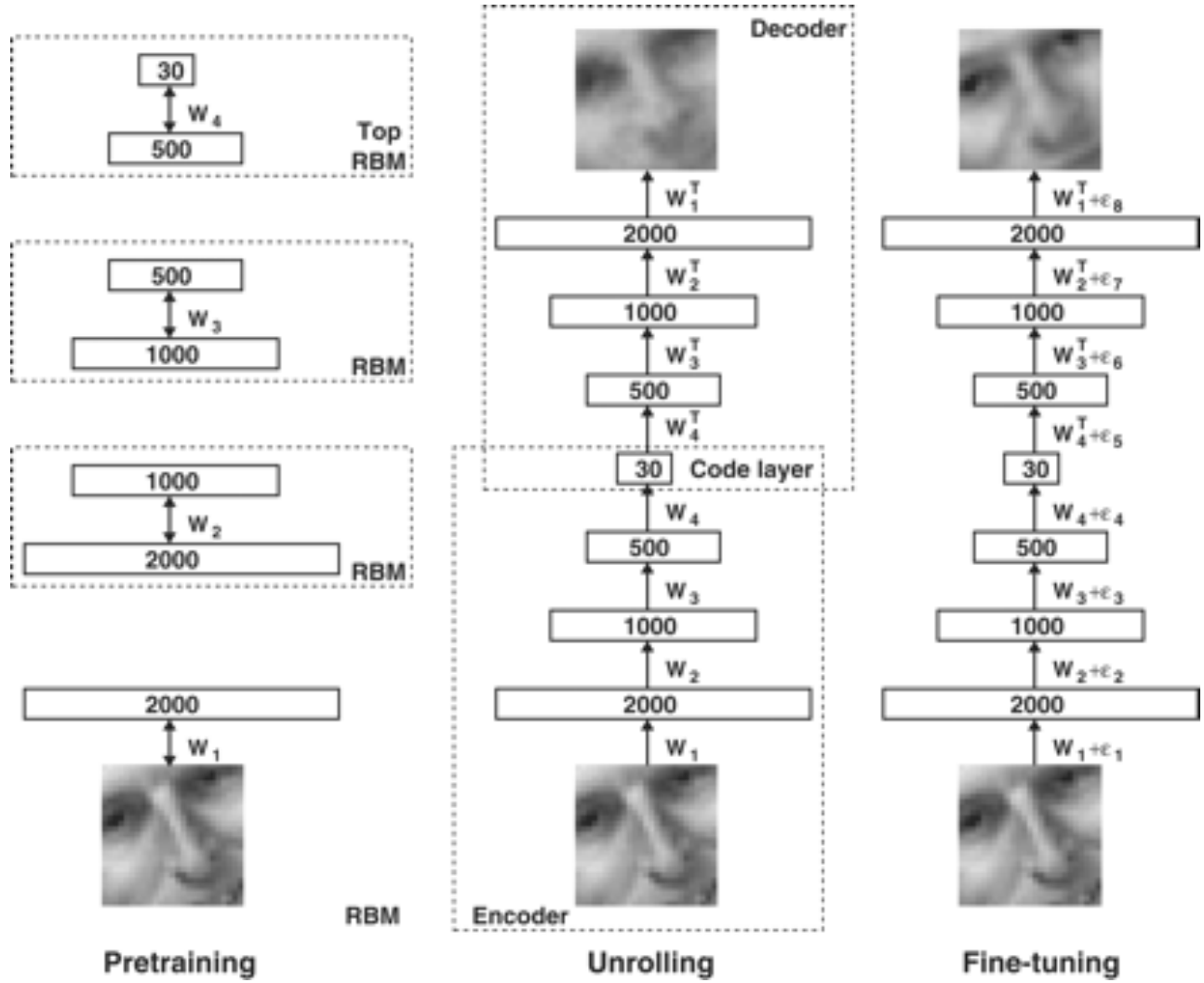
Best network structure unknown



Hard to handle nominal features

Deep network

autoencoder:



[Hinton and Salakhutdinov, Science 2006]

Summary: neural networks

model: neural network

scoring function: least square loss

optimization: gradient descent

习题

对于分类问题，当训练数据没有冲突时，决策树学习算法是否一定能取得0训练错误率？（冲突样本：两个完全相同的样本却被标记为不同类别）

决策树学习算法是否需要训练样本规范化(normalization)?

BP算法能否收敛到全局最优解?