Lesson 07

Classification, Boosting, Support Vector Machine, Decision Trees

Ing. Marek Hrúz, Ph.D.

Katedra Kybernetiky
Fakulta aplikovaných věd
Západočeská univerzita v Plzni

DEPARTMENT OF
CYBERNETICS

DEPARTMENT OF CYBERNETICS

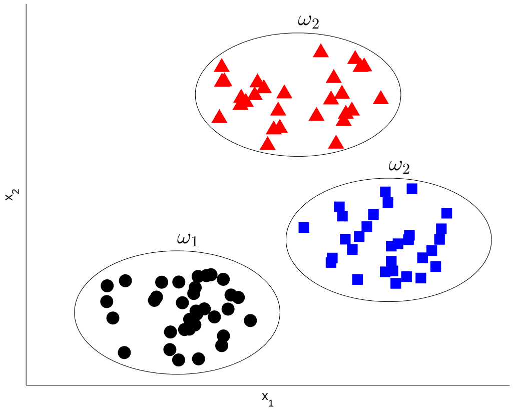
Principles of Classification

- classification vs. clustering - with/without teacher
- **Feature vector**
- is an *n*-dimensional vector describing attributes of the classified object/event
- for the purpose of generality lets assume that a feature vector $x \in \mathcal{R}^n$
- the task of a binary classificator is to divide the $\mathcal{R}^n$ space into two parts so that (ideally) all vectors from one class lie in one part of the space and vice versa
- generally a hyperplane is used as a solution of this problem

DEPARTMENT OF CYBERNETICS

- ω_i is the i^{th} class, $X \in \mathcal{R}^n$ is the space of all classes

$$\begin{aligned} \omega_{1,\dots,N} &\subset X, \\ \bigcup_{i=1}^{N} \omega_i &= X, \\ \omega_i \cap \omega_j &= \emptyset, \text{pro} \quad i,j = 1,\dots,N, \quad i \neq j, \end{aligned}$$



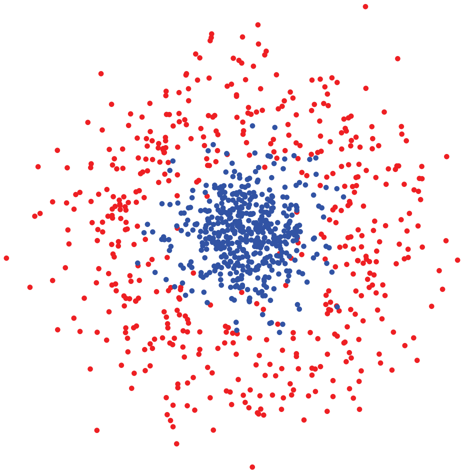
Ada-Boost

- Ada-Boost (short for Adaptive Boosting) is an algorithm creating a strong classifier as a combination of weak classifiers
- a **weak classifier** is such classifier that performs at better than a random choice, i.e. the error $\epsilon < 0.5$ for a binary classification problem
- lets denote a weak classifier as $h(x) \rightarrow \{-1; 1\}$
- a strong classifier is a linear combination of weak classifiers, lets denote it as $H(x) = sign \sum_{t=1}^{T} \alpha_t h_t(x)$

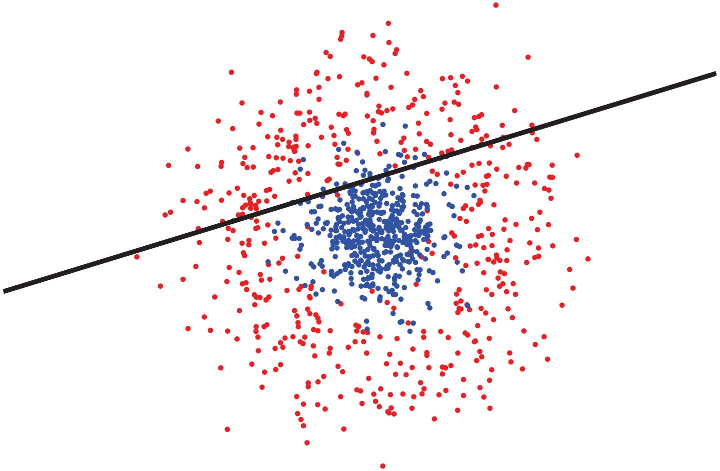
Algorithm

- we have training data available $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_1^N, y \rightarrow \{-1; 1\}$
- initialize weights corresponding to individual feature vectors as $\omega_0(i) = 1/N$
- for $t = 1, \dots, T$:
- compute $h_t = \operatorname{argmin}_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^N \omega_i [y_i \neq h_j(x_i)]$
- if $\epsilon_t \geq 0.5$ then stop - the classifier failed to train
- set $\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- update $\omega_{t+1}(i) = \omega_t(i) \exp(-\alpha_t y_i h_t(x_i))/Z_t$
- iterate until $\epsilon_t = 0$
- the final strong classifier $H(x) = sign \sum_{t=1}^T \alpha_t h_t(x)$

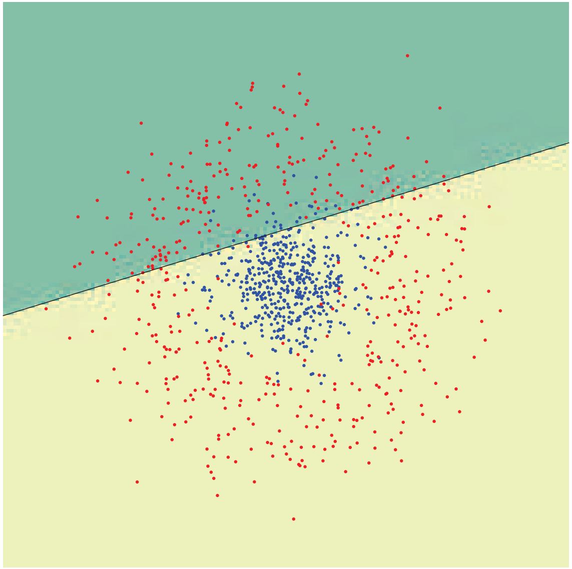
DEPARTMENT OF CYBERNETICS



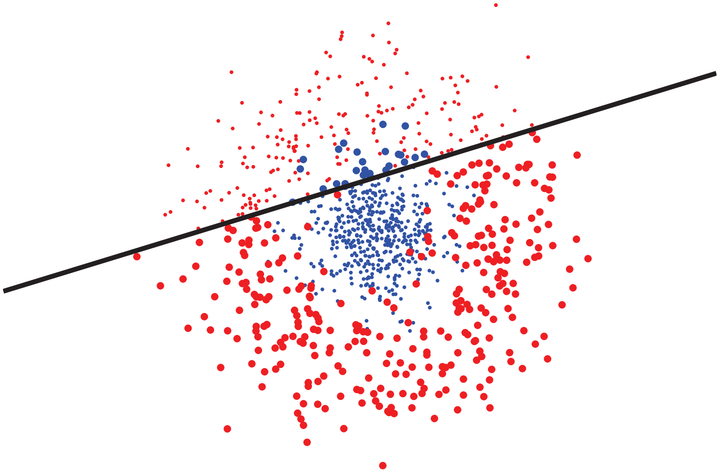
DEPARTMENT OF
CYBERNETICS



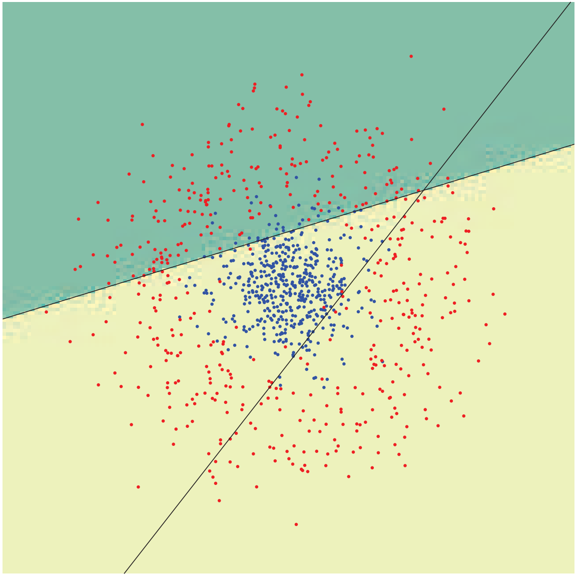
DEPARTMENT OF CYBERNETICS



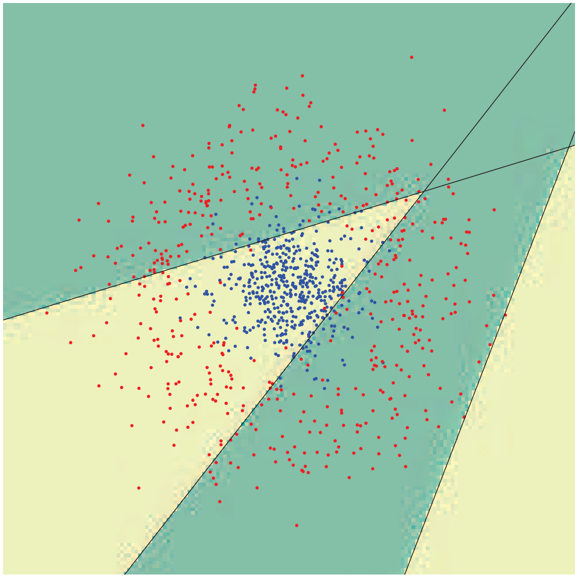
DEPARTMENT OF CYBERNETICS



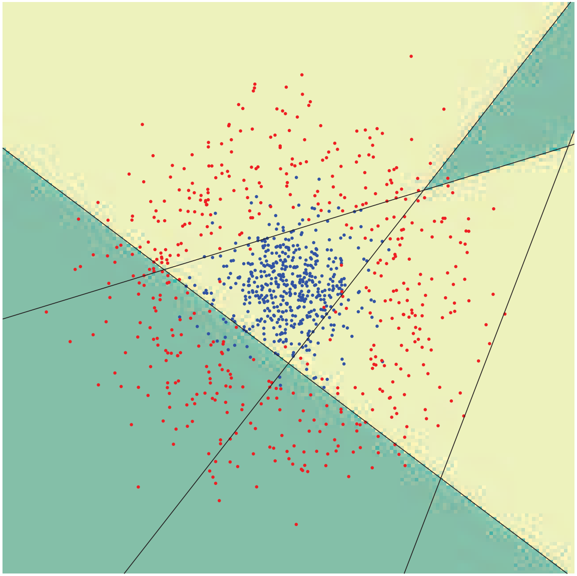
DEPARTMENT OF CYBERNETICS

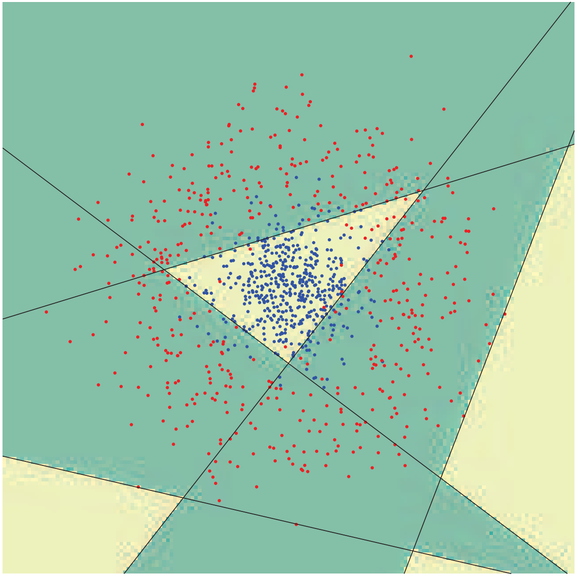


DEPARTMENT OF CYBERNETICS

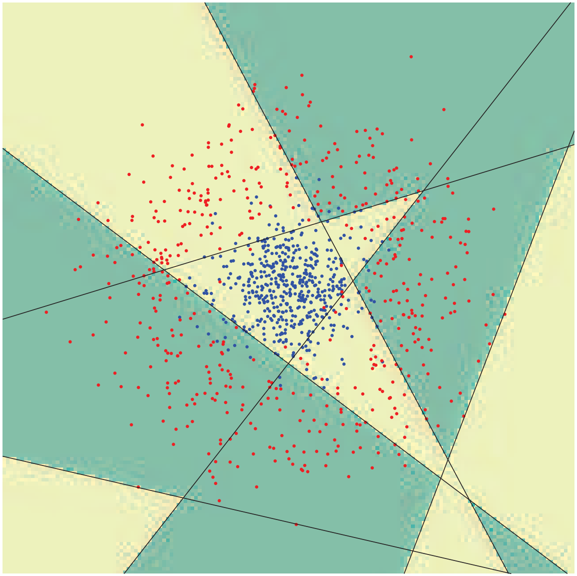


DEPARTMENT OF
CYBERNETICS

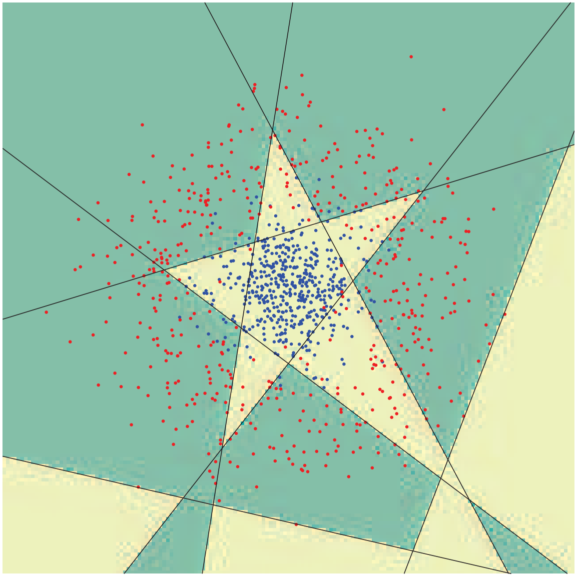




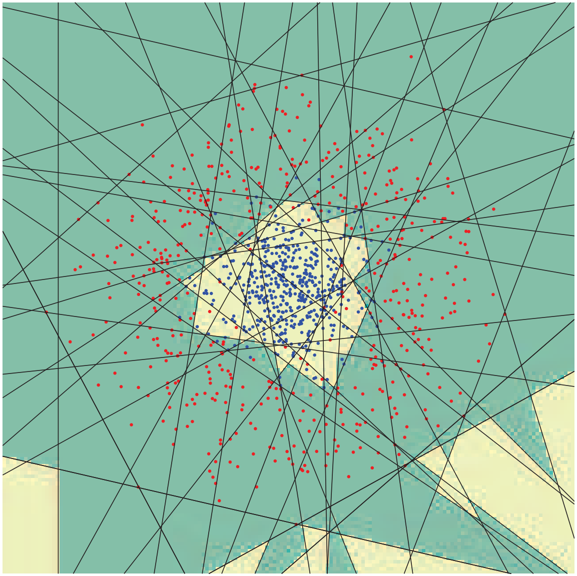
DEPARTMENT OF CYBERNETICS



DEPARTMENT OF CYBERNETICS



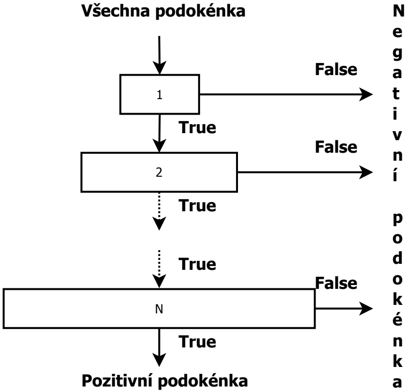
DEPARTMENT OF
CYBERNETICS



DEPARTMENT OF CYBERNETICS

Cascade Ada-Boost

- is a special framework for ada-boost
- the goal is to make the recognition faster but still efficient
- the decision is made sequentially - this allows to refuse some features in very early stages



Support Vector Machine

- in previous sections we have shown how to compute a decision boundary
- in the case of linearly separable classes there exist a lot of boundaries that will classify the training set with 100% precision
- the question is: Is there (in some sense) an optimal decision boundary?

Support Vector Machine

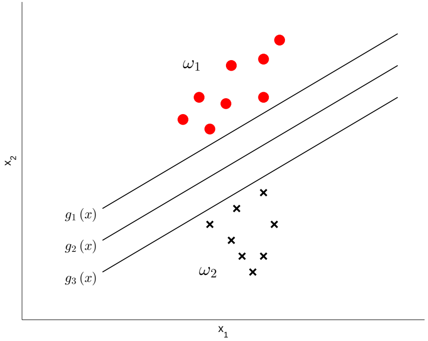
- in previous sections we have shown how to compute a decision boundary
- in the case of linearly separable classes there exist a lot of boundaries that will classify the training set with 100% precision
- the question is: Is there (in some sense) an optimal decision boundary?
- The criterion: The distance between the boundary and the nearest training vector is maximized

- we have a training set $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_1^N, y \rightarrow \{-1; 1\}$
- we have to find the parameters of a decision boundary $\boldsymbol{\omega}$ (previously $\boldsymbol{\Theta}$)

$$\boldsymbol{\omega}^\top \mathbf{x} > 0, \quad \text{pro } \forall \mathbf{x} \in \omega_1,$$
$$\boldsymbol{\omega}^\top \mathbf{x} < 0, \quad \text{pro } \forall \mathbf{x} \in \omega_2.$$

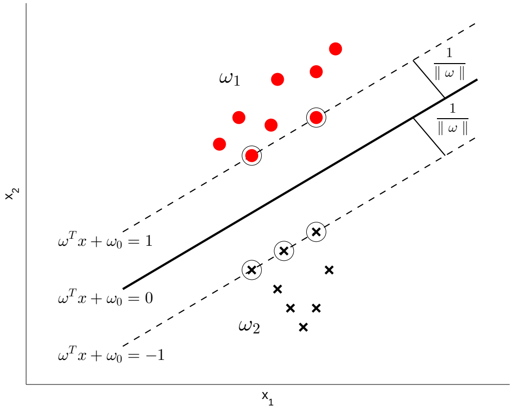
- the decision boundary is then defined as:

$$g(\mathbf{x}) = \boldsymbol{\omega}^\top \mathbf{x} + \omega_0 = 0, \tag{1}$$



DEPARTMENT OF CYBERNETICS

- as said, SVM tries to find the optimal boundary based on the distances from the training data
- with some normalization and math this can be achieved relatively easily



ω_1
ω_2
$\frac{1}{\| \omega \|}$
$\frac{1}{\| \omega \|}$
$\omega^T x + \omega_0 = 1$
$\omega^T x + \omega_0 = 0$
$\omega^T x + \omega_0 = -1$
x_2
x_1

DEPARTMENT OF CYBERNETICS

- we want to find such parameters $\boldsymbol{\omega}$ that will satisfy:

$$\boldsymbol{\omega}^\top \mathbf{x} + \omega_0 \geq +1, \quad \text{pro } \forall \mathbf{x} \in \omega_1,$$
$$\boldsymbol{\omega}^\top \mathbf{x} + \omega_0 \leq -1, \quad \text{pro } \forall \mathbf{x} \in \omega_2.$$

- and we know that the distance between the hyperplanes satisfying the equality in the equations above will be $\frac{2}{\|\boldsymbol{\omega}\|}$
- we want this distance to be maximized
- this leads to the criterion $J = \min \|\boldsymbol{\omega}\|$ which for the math sake will be changed to $J = \min \frac{1}{2}\|\boldsymbol{\omega}\|^{\mathbf{2}}$
- but with the condition of good classification

$$y_i \left(\boldsymbol{\omega}^\top \mathbf{x}_i + \omega_0\right) \geq 1, \quad i = 1, 2, \dots, N. \tag{2}$$

- the vectors $\mathbf{x_i}$ that satisfy $y_i \left(\boldsymbol{\omega}^\top \mathbf{x}_i + \omega_0\right) = 1$ are called support vectors

Optimization of the SVM criterion

- to optimize a criterion with conditions we make use of the Lagrangian multiplicator

$$\mathcal{L}\left(\boldsymbol{\omega}, \omega_0, \boldsymbol{\lambda}\right) = \frac{1}{2}\boldsymbol{\omega}^\top\boldsymbol{\omega} - \sum_{i=1}^{N} \lambda_i \left[y_i \left(\boldsymbol{\omega}^\top \mathbf{x}_i + \omega_0\right) - 1\right] \tag{3}$$

- we need to find the minimum of $\mathcal{L}$
- we use partial derivations

$$\frac{\partial}{\partial \boldsymbol{\omega}}\mathcal{L}\left(\boldsymbol{\omega}, \omega_0, \boldsymbol{\lambda}\right) = 0 \tag{4}$$

$$\frac{\partial}{\partial \omega_0}\mathcal{L}\left(\boldsymbol{\omega}, \omega_0, \boldsymbol{\lambda}\right) = 0 \tag{5}$$

- to optimize a criterion with conditions we make use of the Lagrangian multiplicator

$$\mathcal{L}\left(\boldsymbol{\omega}, \omega_0, \boldsymbol{\lambda}\right) = \frac{1}{2}\boldsymbol{\omega}^\top\boldsymbol{\omega} - \sum_{i=1}^{N} \lambda_i \left[y_i \left(\boldsymbol{\omega}^\top \mathbf{x}_i + \omega_0\right) - 1\right] \tag{6}$$

- this leads to the solution

$$\boldsymbol{\omega} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i, \tag{7}$$

$$0 = \sum_{i=1}^{N} \lambda_i y_i. \tag{8}$$

Dual form

- we make use of the dual form of the problem
- we take the primal solution and substitute it to the primal problem and find the maximum

$$\min_{\boldsymbol{\omega},\omega_0} \left(\frac{1}{2}\boldsymbol{\omega}^\top \boldsymbol{\omega} - \sum_{i=1}^{N} \lambda_i \left[y_i \left(\boldsymbol{\omega}^\top \mathbf{x}_i + \omega_0 \right) - 1 \right] \right) \tag{9}$$

- becomes

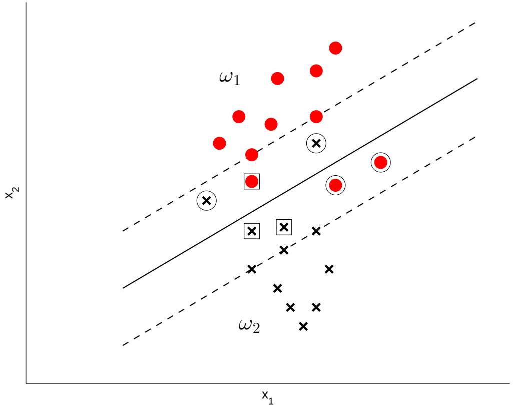
$$\max_{\lambda} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \right) \tag{10}$$

- maximizing this equation yields the solution for λ_i which when substituted to the equation $\boldsymbol{\omega} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ give us the solution for ω

DEPARTMENT OF CYBERNETICS

Soft-margin

- when the classes are linearly non-separable



DEPARTMENT OF CYBERNETICS

- vectors that are correctly classified: $y_i \left(\boldsymbol{\omega}^\top \mathbf{x}_i + \omega_0\right) \geq 1$
- vectors that are correctly classified but lie in the margin: $0 \leq y_i \left(\boldsymbol{\omega}^\top \mathbf{x} + \omega_0\right) < 1$
- vectors that are misclassified: $y_i \left(\boldsymbol{\omega}^\top \mathbf{x} + \omega_0\right) < 0$
- this can be written as:

$$y_i \left(\boldsymbol{\omega}^\top \mathbf{x} + \omega_0\right) \geq 1 - \xi_i \tag{11}$$

- the goal is to find the hyperplane that maximizes the margin and minimizes the number of points for which $\xi > 1$
- this leads to a new formulation of the problem:

$$J\left(\boldsymbol{\omega}, \omega_0, \boldsymbol{\xi}\right) = \frac{1}{2} \parallel \boldsymbol{\omega} \parallel^2 + C \sum_{i=1}^{N} I\left(\xi_i\right), \tag{12}$$

$$I\left(\xi_i\right) = \begin{cases} 1, & \xi_i > 0, \\ 0, & \xi_i = 0. \end{cases} \tag{13}$$

Kernel Trick

- in the solution of the SVM:

$$\max_{\lambda} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \right) \tag{14}$$

- we can see the dot product of x_i, x_j
- this can be efficiently written with the kernel trick as

$$\max_{\lambda} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \right) \tag{15}$$

- this represents a transformation of the vectors into a higher dimension
- in this higher dimension the vectors can be linearly separable

Kernel Types

Type of kernel	Formula	Note
Polynomial	$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\mathbf{x}_i \mathbf{x}_j + \theta\right)^d$	Parameter d and threshold θ is chosen by user.
Sigmoid kernel	$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = tahh\left(\eta \mathbf{x}_i \mathbf{x}_j + \theta\right)$	Parameter η and threshold θ is chosen by user.
Gauss kernel *Radial Basis Function* (RBF)	$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = exp\left(-\frac{1}{2\sigma^2} \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2\right)$	Parameter σ is is chosen by user.

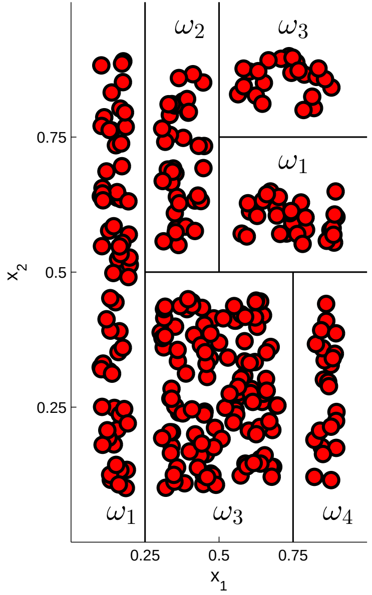
DEPARTMENT OF CYBERNETICS

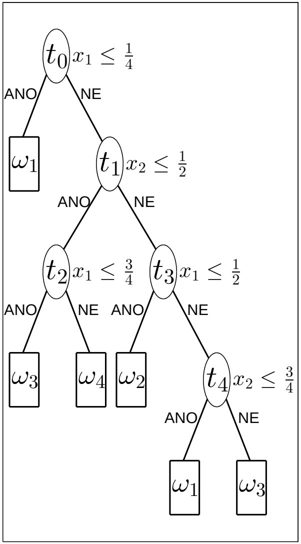
Decision Tree

- non-linear classification method, the model is based on oriented graph $\rightarrow$ tree

Decision Tree

- non-linear classification method, the model is based on oriented graph $\rightarrow$ tree
- belongs to a family of models - Classification And Regression Tree (CART)

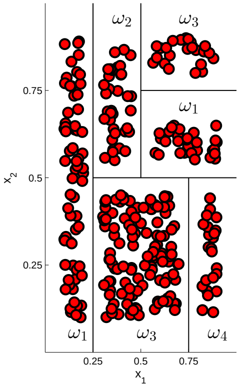


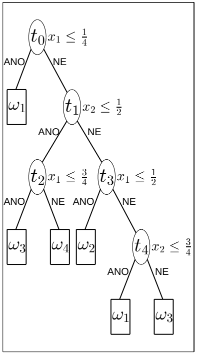


t_0 $x_1 \leq \frac{1}{4}$
ANO
NE
ω_1
t_1 $x_2 \leq \frac{1}{2}$
ANO
NE
t_2 $x_1 \leq \frac{3}{4}$
t_3 $x_1 \leq \frac{1}{2}$
ANO
NE
ANO
NE
ω_3
ω_4
ω_2
t_4 $x_2 \leq \frac{3}{4}$
ANO
NE
ω_1
ω_3

DEPARTMENT OF CYBERNETICS

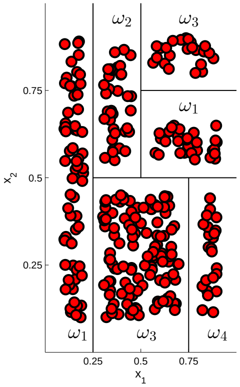
- method uses binary decision tree T consisting of nodes $\rightarrow$ elements of feature vector $\mathbf{x} \in X$ are evaluated via a condition

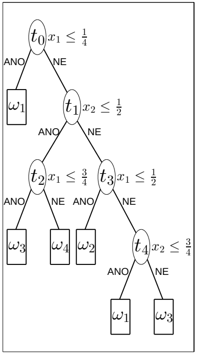




t_0 $x_1 \leq \frac{1}{4}$
ANO
NE
ω_1
t_1 $x_2 \leq \frac{1}{2}$
ANO
NE
t_2 $x_1 \leq \frac{3}{4}$
t_3 $x_1 \leq \frac{1}{2}$
ANO
NE
ANO
NE
ω_3
ω_4
ω_2
t_4 $x_2 \leq \frac{3}{4}$
ANO
NE
ω_1
ω_3

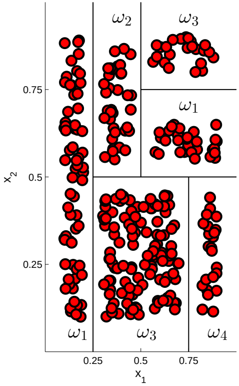
- method uses binary decision tree T consisting of nodes $\rightarrow$ elements of feature vector $\mathbf{x} \in X$ are evaluated via a condition
- the tree then represents a gradual segmentation of the feature space X into disjunct regions

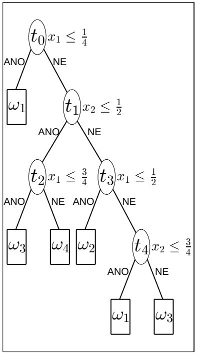




t_0 $x_1 \leq \frac{1}{4}$
ANO
NE
ω_1
t_1 $x_2 \leq \frac{1}{2}$
ANO
NE
t_2 $x_1 \leq \frac{3}{4}$
t_3 $x_1 \leq \frac{1}{2}$
ANO
NE
ANO
NE
ω_3
ω_4
ω_2
t_4 $x_2 \leq \frac{3}{4}$
ANO
NE
ω_1
ω_3

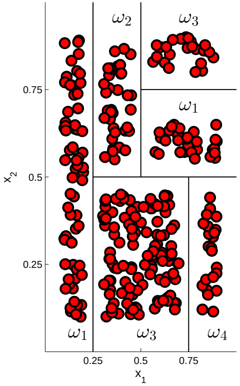
- each region represents one and only one class

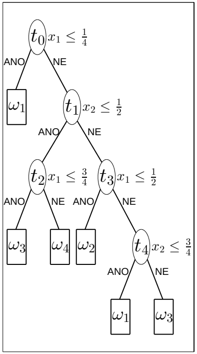




t_0 $x_1 \leq \frac{1}{4}$
ANO
NE
ω_1
t_1 $x_2 \leq \frac{1}{2}$
ANO
NE
t_2 $x_1 \leq \frac{3}{4}$
t_3 $x_1 \leq \frac{1}{2}$
ANO
NE
ANO
NE
ω_3
ω_4
ω_2
t_4 $x_2 \leq \frac{3}{4}$
ANO
NE
ω_1
ω_3

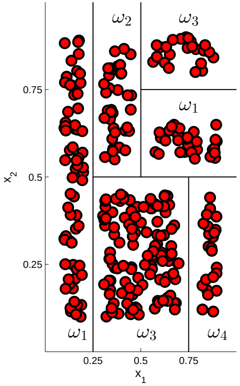
- each region represents one and only one class
- the feature space is divided into rectangular regions (the region boundaries are parallel to axis of feature space)

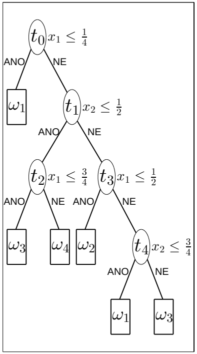




t_0 $x_1 \le \frac{1}{4}$
ANO
NE
ω_1
t_1 $x_2 \le \frac{1}{2}$
ANO
NE
t_2 $x_1 \le \frac{3}{4}$
t_3 $x_1 \le \frac{1}{2}$
ANO
NE
ANO
NE
ω_3
ω_4
ω_2
t_4 $x_2 \le \frac{3}{4}$
ANO
NE
ω_1
ω_3

- each region represents one and only one class
- the feature space is divided into rectangular regions (the region boundaries are parallel to axis of feature space)
- the inequations in nodes $x_i \leq \alpha$ is known as the decision rule





t_0 $x_1 \leq \frac{1}{4}$
ANO
NE
ω_1
t_1 $x_2 \leq \frac{1}{2}$
ANO
NE
t_2 $x_1 \leq \frac{3}{4}$
t_3 $x_1 \leq \frac{1}{2}$
ANO
NE
ANO
NE
ω_3
ω_4
ω_2
t_4 $x_2 \leq \frac{3}{4}$
ANO
NE
ω_1
ω_3

Classification and Learning:

- classification proceeds by comparing an unknown vector in the nodes of the tree
- the unknown vector then falls into one of the leafs which represents a class
- usually the learning is supervised (learning with teacher)
- straightforward way of training $\rightarrow$ the regions are constructed by comparing values in individual dimensions of the vector $\mathbf{x}$ with a threshold, $x_i \leq \alpha$, x_i is the i^{th} element of the feature vector $\mathbf{x}$ and α is a threshold

Rules of the construction of the decision tree:

- the first node (root) of the tree contains the whole training set, $X_s = X$
- every next node s contains the subset $X_s \subset X$ given by the decision rule of the previous node
- the decision rule divides X_s into two subsets X_{sT} (TRUE) and X_{sF} (FALSE)
- the division must fulfill:

$$X_{sT} \cap X_{sF} = \emptyset,$$
$$X_{sT} \cup X_{sF} = X_s.$$

- from all the possible divisions of X_s we pick just one, which is optimal given a **division criterion**

On choosing the decision rule

- the decision rule in the form $x_i \leq \alpha_i$, where α_i is a threshold $\alpha_i \in \mathrm{R}$ divides the feature vectors based on the comparison of the i^{th} dimension of the feature vector
- thanks to the train set X it is possible to enumerate a finite set of values for computing α_i
- for the i^{th} dimension of feature space the values of all feature vectors on this dimension are ordered ascending $\rightarrow$ we have a finite set of values for computing the threshold
- in a given node we can enumerate all the possible values from all the dimensions x_i
- from this set of possible divisions (values of the threshold) we need to choose such that will divide the given set of feature vectors "the best"$\rightarrow$ we need a metric (eg. Gini impurity, variance reduction, information gain, ...)

Information gain approach

- let $P(\omega_i \mid s)$ be the probability of vectors in the set X_s belonging to the class ω_i
- the information gain is based around the entropy:

$$I(s) = -\sum_{i=1}^{M} P(\omega_i \mid s) \log_2 P(\omega_i \mid s). \qquad (16)$$

- this equation represents the rate of *entropy* of the node s
- the probabilities $P(\omega_i \mid s)$ are estimated by $\frac{N_s^i}{N_s}$, where N_s^i is the number of vectors in X_s belonging to class ω_i and N_s is the total number of vectors in the subset X_s

- after dividing X_s into two subsets X_{sT} a X_{sF}, where X_{sT} is composed of N_{sT} vectors and X_{sF} is composed of N_{sF} vectors, the information gain (of this division) is:

$$\Delta I(s) = I(s) - \frac{N_{sA}}{N_s} I(s_A) - \frac{N_{sN}}{N_s} I(s_N), \qquad (17)$$

  where $I(s_A)$, $I(s_N)$ are the rates of entropy of nodes s_A and s_N
- the goal of the training is to find for each node s such division for which the information gain $\Delta I(s)$ is maximized

DEPARTMENT OF CYBERNETICS

Stopping criterion

- is used to stop the process of division and thus creating a leaf node
- one option is to set the minimal number of training vectors in the node
- another option is to set a minimal information gain that is needed for the division

Classification

- a leaf node s represents the class for which there are the most training vectors in the leaf node
- each leaf node represents one class ω_j, where j is

$$j = \operatorname*{argmax}_i P\left(\omega_i \mid s\right). \tag{18}$$

Other options of constructing the tree:

- the decision rule can have the form of $\sum_{i=1}^{l} c_i x_i \leq \alpha$
- we are not looking for thresholds but for parameters of a hyperplane that divides the feature space into two subsets
- when considering two dimensional feature space and by rearranging the expression we obtain: $c_1 x + c_2 y - \alpha \leq 0$
- which is a general form of equation of a half-plane
- can be more suitable in some cases, but the construction of the tree is more complex

Decision forest

- a disadvantage of the decision tree is the sensitivity to the training set, so called bad generalization
- a small change in training set X results in change of topology of the whole decision tree T
- this drawback is compensated by using more trees in the training/testing phase
- principle: for one training set we construct several different trees

Training:

- the training set X is divided into several training sets $X_{(t)}$ by utilizing the *bootstrap aggregating* algorithm

Classification:

Training:

- the training set X is divided into several training sets $X_{(t)}$ by utilizing the *bootstrap aggregating* algorithm
- each set contains $N_{(t)}$ unique feature vectors, with $N_{(t)} \leq N$, but the cardinality of the set remains N (the elements may repeat)

Classification:

DEPARTMENT OF CYBERNETICS

Training:

- the training set X is divided into several training sets $X_{(t)}$ by utilizing the *bootstrap aggregating* algorithm
- each set contains $N_{(t)}$ unique feature vectors, with $N_{(t)} \leq N$, but the cardinality of the set remains N (the elements may repeat)
- for each set $X_{(n)}$ a decision tree is constructed

Classification:

Training:

- the training set X is divided into several training sets $X_{(t)}$ by utilizing the *bootstrap aggregating* algorithm
- each set contains $N_{(t)}$ unique feature vectors, with $N_{(t)} \leq N$, but the cardinality of the set remains N (the elements may repeat)
- for each set $X_{(n)}$ a decision tree is constructed

Classification:

- unknown vector $\mathbf{y}$ is inputed into all decision trees

Training:

- the training set X is divided into several training sets $X_{(t)}$ by utilizing the *bootstrap aggregating* algorithm
- each set contains $N_{(t)}$ unique feature vectors, with $N_{(t)} \leq N$, but the cardinality of the set remains N (the elements may repeat)
- for each set $X_{(n)}$ a decision tree is constructed

Classification:

- unknown vector $\mathbf{y}$ is inputed into all decision trees
- each decision tree outputs the class ω_i for the unknown vector $\mathbf{y}$

DEPARTMENT OF CYBERNETICS

Training:

- the training set X is divided into several training sets $X_{(t)}$ by utilizing the *bootstrap aggregating* algorithm
- each set contains $N_{(t)}$ unique feature vectors, with $N_{(t)} \leq N$, but the cardinality of the set remains N (the elements may repeat)
- for each set $X_{(n)}$ a decision tree is constructed

Classification:

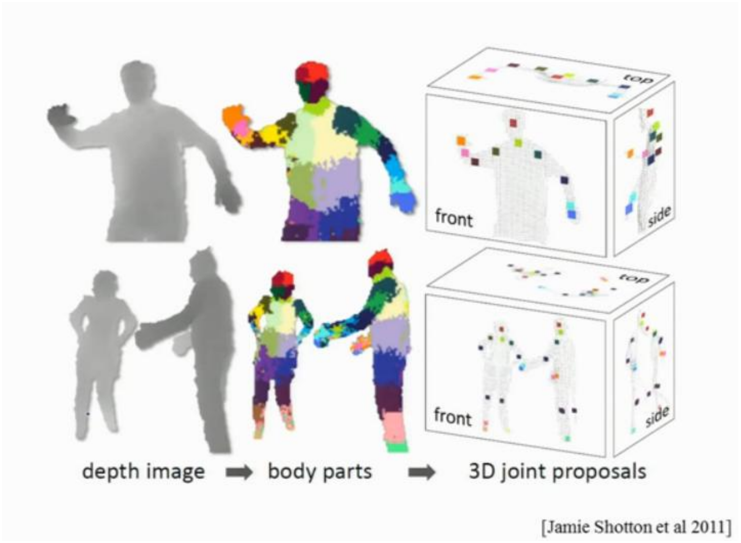
- unknown vector $\mathbf{y}$ is inputed into all decision trees
- each decision tree outputs the class ω_i for the unknown vector $\mathbf{y}$
- index i of the final class is chosen as the most frequent result, alternatively we may compute the probability for each class as $P(\omega_i|\mathbf{y}) = \frac{1}{T}\sum_{t=1}^{T} P_t(\omega_i|\mathbf{y})$

Náhodný rozhodovací les (Random Decision Forest)

- the same principle as the decision forest $\rightarrow$ lowering the sensitivity of classification on the training set
- . . . but also
- goal 1: lowering the correlation of the trees in the forest
- goal 2: make the training faster (especially for higher dimensions)

DEPARTMENT OF CYBERNETICS

Real-time classification of depth data from MS Kinect into individual parts of human body (Microsoft Research, 2011):



DEPARTMENT OF CYBERNETICS

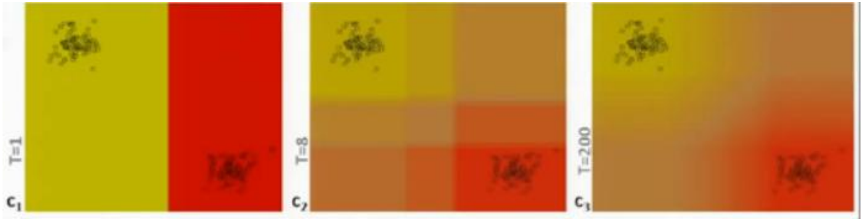
Training

1. **division of the training set** X into T sets $X_{(t)}$ using *bootstrap aggregating* (the same)
2. we choose a parameter m ($m \ll l$, where l is the dimensionality of $\mathbf{x} \in X$)
3. for one tree in a given node $\leftarrow$ the decision rule is determined based only on randomly chosen m dimensions
4. after the tree is trained, choose another m dimensions and train another tree, and so on

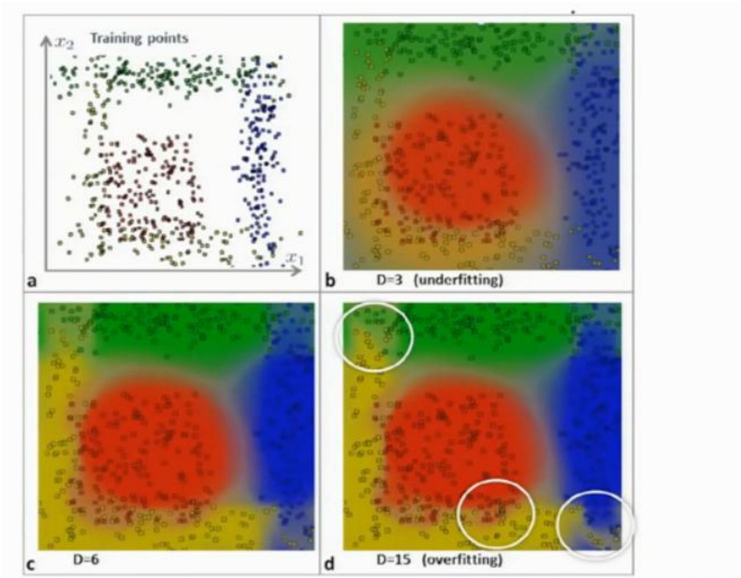
Classification

- an unknown vector $\mathbf{y}$ is inputed into all the trees
- index i of the final class is chosen as the most frequent result, alternatively we may compute the probability for each class as $P(\omega_i|\mathbf{y}) = \frac{1}{T}\sum_{t=1}^{T} P_t(\omega_i|\mathbf{y})$

Effect of the size of the forest:



Effect of the depth of the trees:



[Criminisi et al. 2011]