Support Vector Machines
SVM

Prof. Dr. Stephan Trahasch

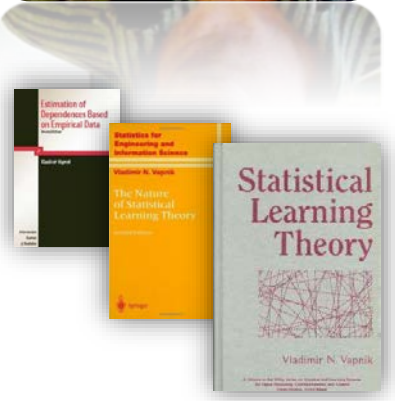
Offenburg University of Applied Sciences

Outline

- Motivation
- Classification with hyperplanes
- Mathematical formulation
- Soft margin
- Kernel trick
- One-Class SVM
- Summary

History

Inventor Vladimir Naumovich Vapnik



1936 Soviet-American mathematician

1990 AT&T Bell Laboratories, New Jersey

Researches in the field of adaptive systems

2002 NEC in Princeton, New Jersey

Researches in the field of machine learning

2014 Vapnik joined Facebook's Artificial Intelligence lab

1995 Professor of Computer Science and Statistics position at Royal Holloway, University of London

2003 Professor of Computer Science at Columbia University, New York City

25.11.2014

Facebook AI Research
We are delighted to announce that Vladimir Vapnik has joined Facebook AI Research.

Vladimir is universally known in the machine learning and statistics communities as the father of statistical learning theory and the co-inventor of the Support Vector Machine method. One of the key concepts of learning theory bears his name: the Vapnik-Chervonenkis Dimension, which measures the capacity of a learning machine.

Vladimir is rejoining some of his long-time collaborators Jason Weston, Ronan Collobert, and Yann LeCun.

He is working on new book, and will be collaborating with FAIR research scientists to develop some of his new ideas on conditional density estimation, learning with privileged information, and other topics. — mit Yann LeCun.

25. November

History of SVMs

1963	Large Margin Classifier	classic, linear and binary Classification
1992	Non-Linear Classifier	Kernel Machine
1995	Soft Margin Classifier	Classification with Outliers
1996	Regression (Unsupervised Learning)	Statistical Correlation: Input and Output
~2000	Multi-Class (Meta)	Multi-Classifier

Journey to the Support Vector Machines

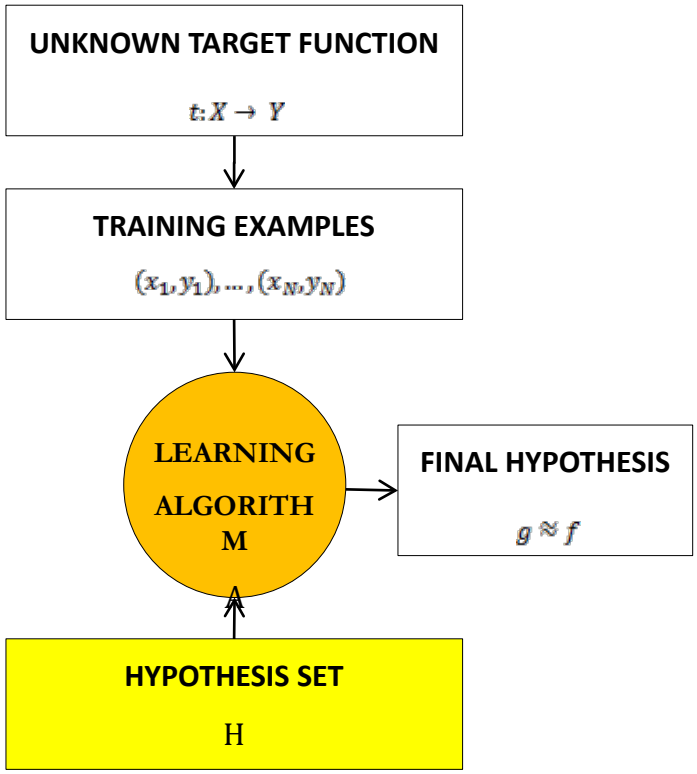
1. Motivation
2. Classification with hyperplanes
3. Finding an optimal hyperplane
    1. Formulation as a primary minimization problem
    2. Converting to a dual problem
    3. Solving with Lagrange

Second section:

1. soft margin
2. kernel trick

Support Vector Machine as „linear classifier“

- Learn the optimal linear separation of data elements
- Allows later classification of unknown test data based on the learned linear hyperplane
- No machine in the true sense of the word



Error minimization

Up to now we have optimized linear models

$$y = f(\vec{x}) = \sum_{i=0}^{p} \beta_i x_i = \vec{x}^T \vec{\beta}$$

for error minimization:

$$\mathrm{RSS}(\vec{\beta}) = \sum_{i=1}^{N} \left(y_i - \vec{x}_i^{\,T} \vec{\beta} \right)^2$$

Problem

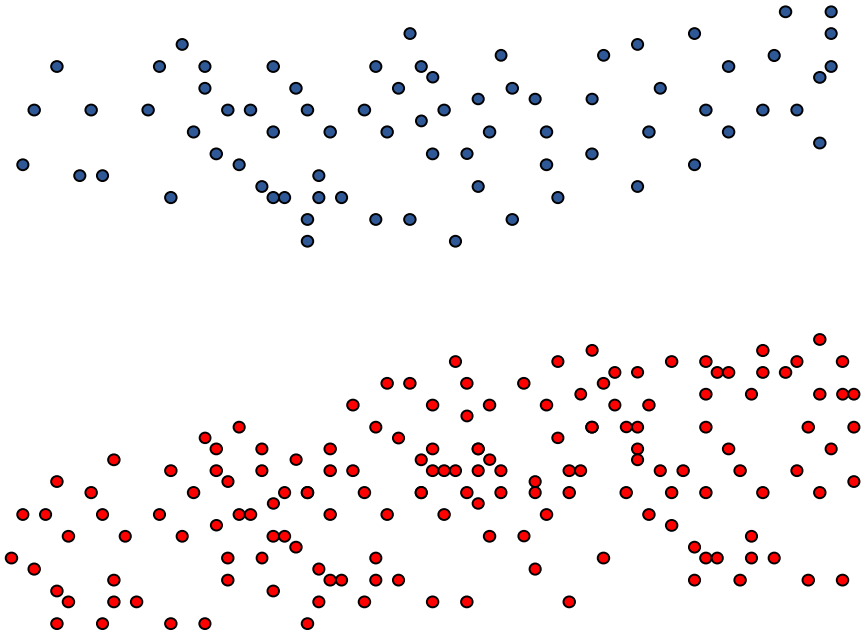
There are several functions with minimal errors.

Which one should you choose?

Idea:

Keep as wide a margin as possible around the class boundaries.

→ large-margin classification



Basis notation and concepts

Scalar product <x,y>: Given vectors $x, y \in \mathbb{R}^p$

$$\langle \vec{x}, \vec{y} \rangle = \vec{x} * \vec{y} = \sum_{i=1}^{p} x_i \, y_i$$

$$\langle \vec{x}, \vec{y} \rangle = \|\vec{x}\| * \|\vec{y}\| * \cos\left(\measuredangle(\vec{x}, \vec{y})\right)$$

Euclidian distance of vectors $\vec{x}$ $\|\vec{x}\| = \sqrt{\vec{x} * \vec{x}} = \left(\sum_{i=1}^{p} {x_i}^2 \right)^{\frac{1}{2}}$

Hyperplane H: Let $\vec{\beta}$ be the normal vector and $\beta_0 \in \mathbb{R}$

$$H\left(\vec{\beta}, \beta_0\right) = \left\{ \vec{x} \middle| \; \vec{x}^T \vec{\beta} + \beta_0 = 0 \right\}$$

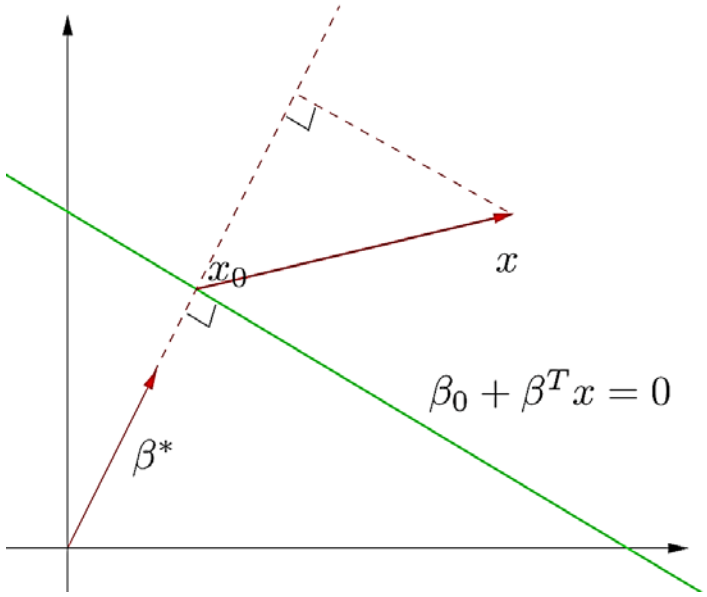
Properties of a hyperplane

Hyperlane $H = \{x \mid x^T\beta + \beta_0 = 0\}$

1. For an two points x_1 and x_2 lying in H, $(x_1 - x_2)^T\beta = 0$, and hence $\beta^* = \frac{\beta}{\|\beta\|}$ is the vector normal to the surface of H.

2. For any point x_0 in H $x_0^T\beta = -\beta_0$

3. The signed distance of any point x to H is given by $(x - x_0)^T\beta^*$



Basis notation and concepts

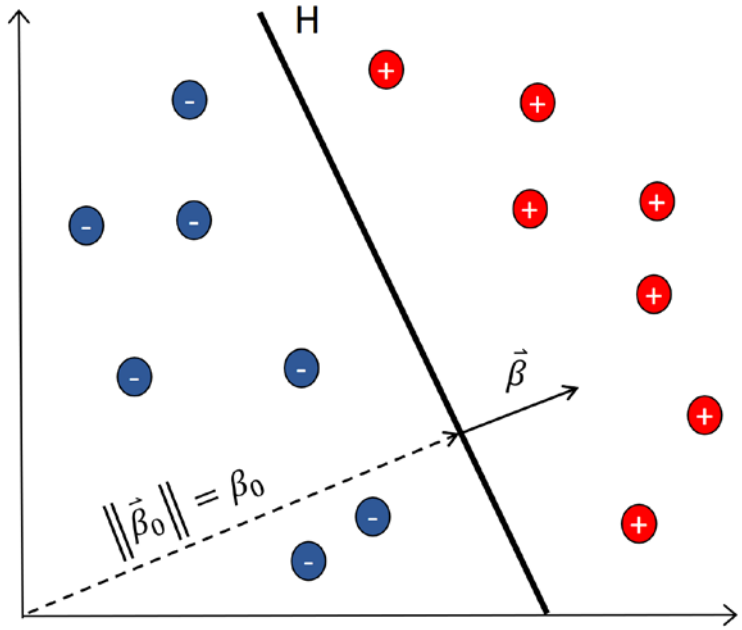
$\vec{\beta}$ is perpendicular to all vectors of $H' = \{x \mid x^T\beta + \beta_0 = 0\}$

$$x^T\beta^* + \beta_0 = \begin{cases} > 0 & \text{if x is on the positive side} \\ = 0 & \text{if x is on H} \\ < 0 & \text{if x is on the negative side} \end{cases}$$

Transform to Hesse normal form:

$$H = \{x \mid x^T\beta^* + \beta_0 = 0\}$$

$$\beta^* = \frac{\beta}{\|\beta\|}$$



Overview

- Motivation
- Classification with hyperplanes
- Mathematical formulation
- Soft margin
- Kernel trick
- One-Class SVM
- Summary

(Classic) SVM is a binary Classifier

Trainings data $(x_1, y_1), (x_2, y_2) \ldots, (x_m, y_m)$ $x_i \in X, y_i \in \{-1, +1\}$

Classification of a new x_k with $y_i \in \{-1, +1\}$

Let $X = C_+ \cup C_-$ be the set of trainings data with

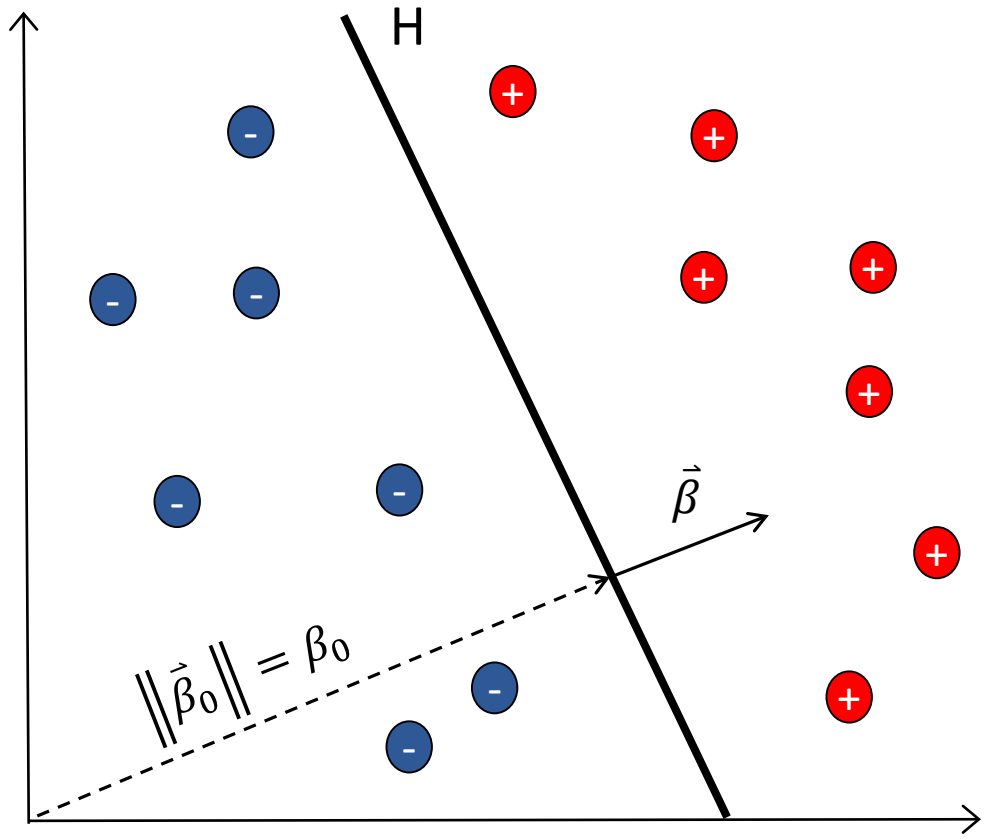
$$C_+ = \{(\vec{x}, y) | y = +1\} \text{ and } C_- = \{(\vec{x}, y) | y = -1\}$$

Find a hyperplane $H = \{x | x^T \beta + \beta_0 = 0\}$

that separates C+ and C- optimal.

For a given hyperplane you can classify a data point x with

$$\hat{y} = sign(x^T \beta + \beta_0)$$


H
$\vec{\beta}$
$\|\vec{\beta}_0\| = \beta_0$

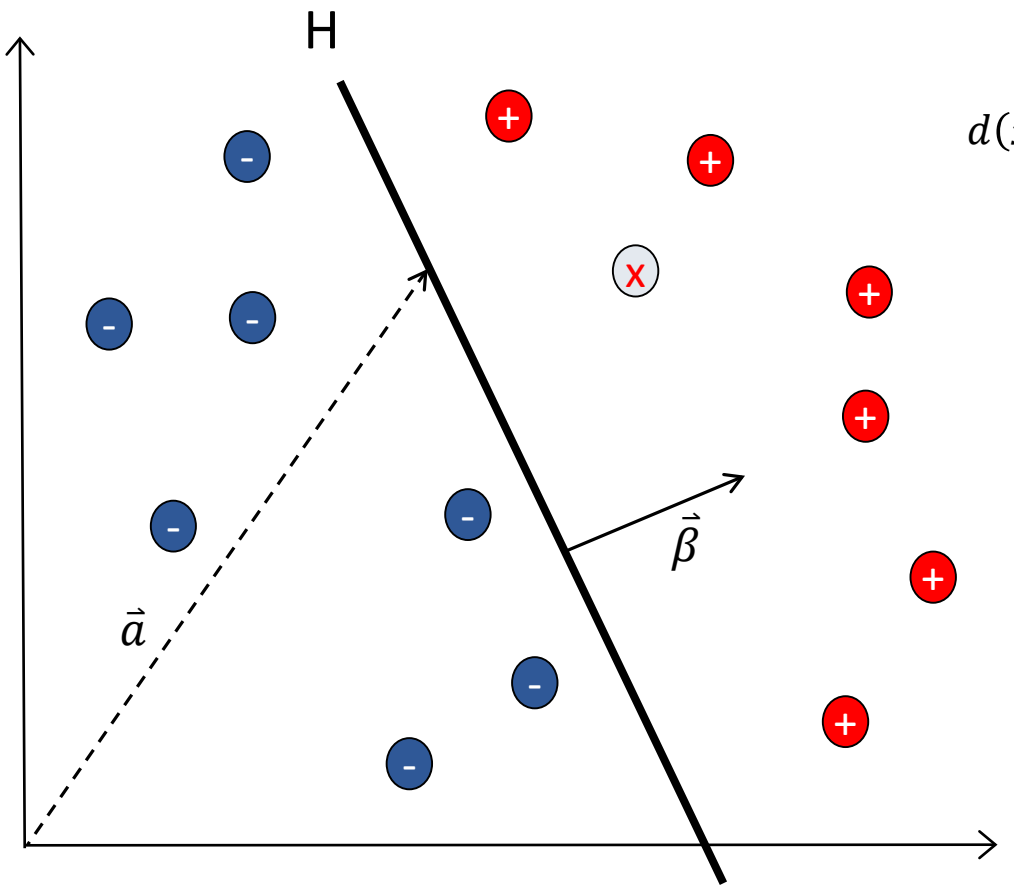
Cosinus determines the sign for the classification

The signed distance of a point x to a hyperplane H with support vector $\vec{a}$ and normal vector $\vec{\beta}$ is calculated as

$$
\begin{aligned}
d(\vec{x}, H) &= \left\langle \vec{x}, \vec{\beta} \right\rangle - \beta_0 \\
&= \left\langle \vec{x}, \vec{\beta} \right\rangle - \left\langle \vec{a}, \vec{\beta} \right\rangle \\
&= \left\langle \vec{x} - \vec{a}, \vec{\beta} \right\rangle \\
&= \|\vec{x} - \vec{a}\| \times \left\|\vec{\beta}\right\| \times \cos\left(\measuredangle\left(\vec{x} - \vec{a}, \vec{\beta}\right)\right)
\end{aligned}
$$

Only $\cos\left(\measuredangle\left(\vec{x} - \vec{a}, \vec{\beta}\right)\right)$ can be negativ!

Example



$$H = \left\{ \vec{x} \middle| \left\langle \vec{x}, \vec{\beta} \right\rangle + \beta_0 = 0 \right\}$$

$$\begin{aligned} d(\vec{x}, H) &= \left\langle \vec{x}, \vec{\beta} \right\rangle - \beta_0 \\ &= \left\langle \vec{x}, \vec{\beta} \right\rangle - \left\langle \vec{a}, \vec{\beta} \right\rangle \\ &= \left\langle \vec{x} - \vec{a}, \vec{\beta} \right\rangle \\ &= \|\vec{x} - \vec{a}\| \times \left\|\vec{\beta}\right\| \times \cos\left(\measuredangle \left(\vec{x} - \vec{a}, \vec{\beta} \right) \right) \end{aligned}$$

$\vec{a}$ = Support Vector $\left\langle \vec{a}, \vec{\beta} \right\rangle = \beta_0$

Sign of Cosinus

Signed distance $d(\vec{x}, H)$

1. distance $|d(\vec{x}, H)|$ as distance from $\vec{x}$ to plane H
2. position of $\vec{x}$ relativ to the orientation $\vec{\beta}$ of H

$$sign(\mathrm{d}(\vec{\mathrm{x}}, \mathrm{H})) = \begin{cases} +1 \text{ if } \mathrm{d}(\vec{\mathrm{x}}, \mathrm{H}) > 0, \cos(\angle(\vec{x}, \vec{\beta})) > 0 \\ -1 \text{ if } \mathrm{d}(\vec{\mathrm{x}}, \mathrm{H}) < 0, \cos(\angle(\vec{x}, \vec{\beta})) < 0 \end{cases}$$

In this way, the points can be classified $\hat{y} = sign\left(\left\langle \vec{x}, \vec{\beta} \right\rangle - \beta_0\right)$

y = -1 → point x is located in the half space containing the origin.

Naive learning algorithm by Schölkopf/Smola

Mean of a class:

$$c_+ = \frac{1}{m_+} \sum_{\{i|y_i=+1\}} x_i$$

$$c_- = \frac{1}{m_-} \sum_{\{i|y_i=-1\}} x_i$$

m_+ = Anzahl positiver Beispiele

m_- = Anzahl negativer Beispiele

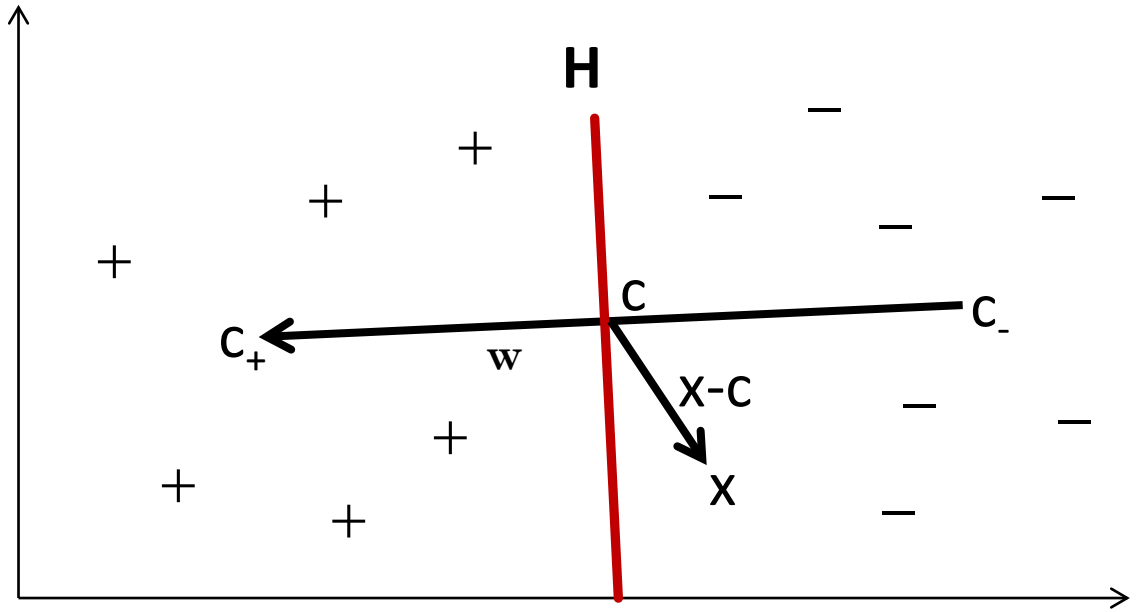
In the middle is the point $\vec{c} = \frac{\vec{c}_+ + \vec{c}_-}{2}$

Vector $\vec{x} - \vec{c}$ connects new observation x and c

Similarity to the average of a class:
Angle between $\vec{w} = \vec{c}_+ - \vec{c}_-$ und $\vec{x} - \vec{c}$

Calculate using a scalar product.

Learning algorithm by Schölkopf/Smola



$$
\begin{aligned}
y &= sign\big((x-c)*w\big) \\
&= sign\big((x-(c_+ + c_-)/2)*(c_+ - c_-)\big) \\
&= sign\left((x*c_+)-(x*c_-)-\frac{1}{2}c_+^2-\frac{1}{2}c_+*c_-+\frac{1}{2}c_-^2+\frac{1}{2}c_+*c_-\right) \\
&= sign\left((x*c_+)-(x*c_-)+\frac{1}{2}\left(\|c_-\|^2-\|c_+\|^2\right)\right) \\
&= sign\big((x*c_+)-(x*c_-)+b\big)
\end{aligned}
$$

That would almost be the support vector method.

But:

Simply calculating the center of the examples of a class is too easy to get a decent w.

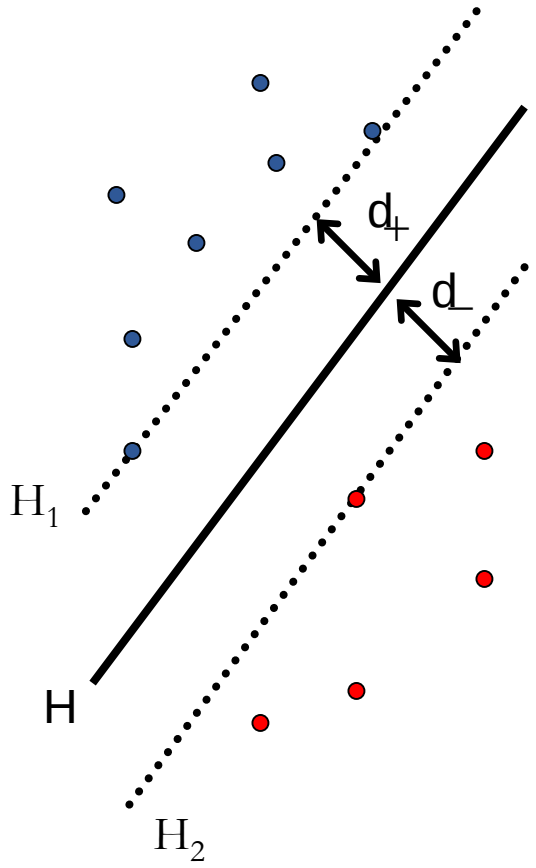
You don't get the optimal hyperplane that way.

Optimal hyperplane

Observations are called linearly separable if there is a hyperplane H that separates the positive and negative examples.

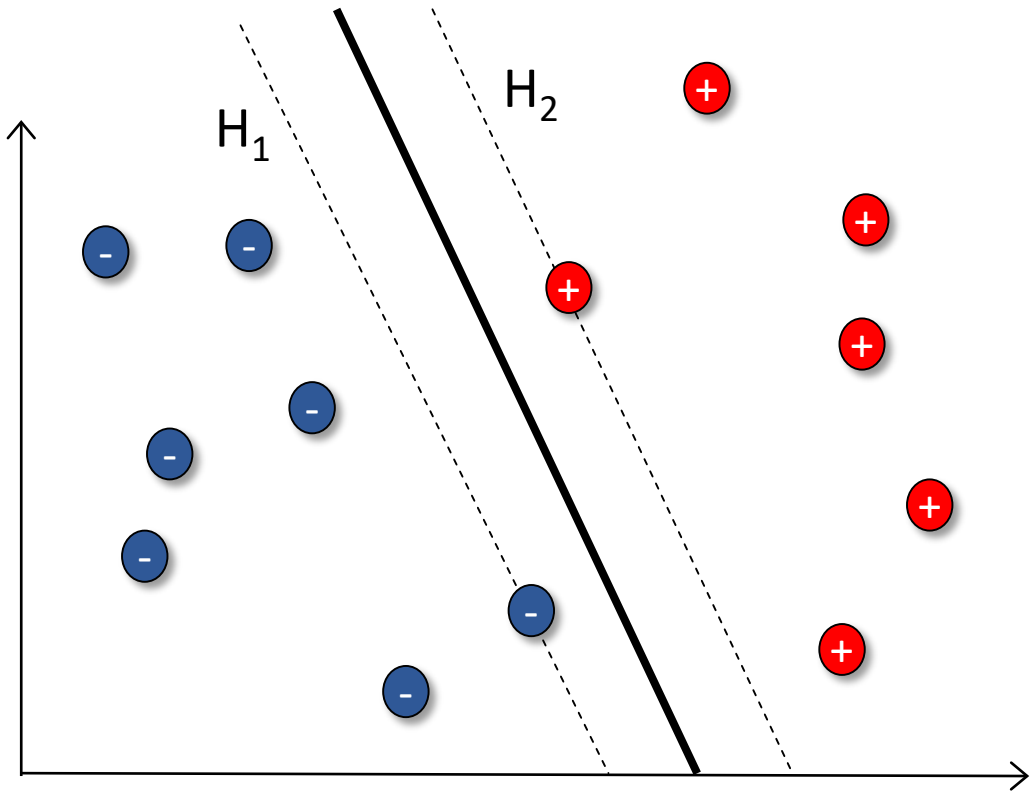
H is a **optimal hyperplane**, if its distance d to the next positive and to the next negative example is maximum.

There is a clearly defined optimal hyperplane.



Margin

$$H = \{x | x^T \beta + \beta_0 = 0\}$$



$$x_+^T \beta + \beta_0 \geq +1$$

$$x_-^T \beta + \beta_0 \leq -1$$

$$y_i = \begin{cases} +1 & \text{for } + \text{ samples} \\ -1 & \text{for } - \text{ samples} \end{cases}$$

$$y_i(x_+^T \beta + \beta_0) \geq +1$$

$$y_i(x_-^T \beta + \beta_0) \geq +1$$

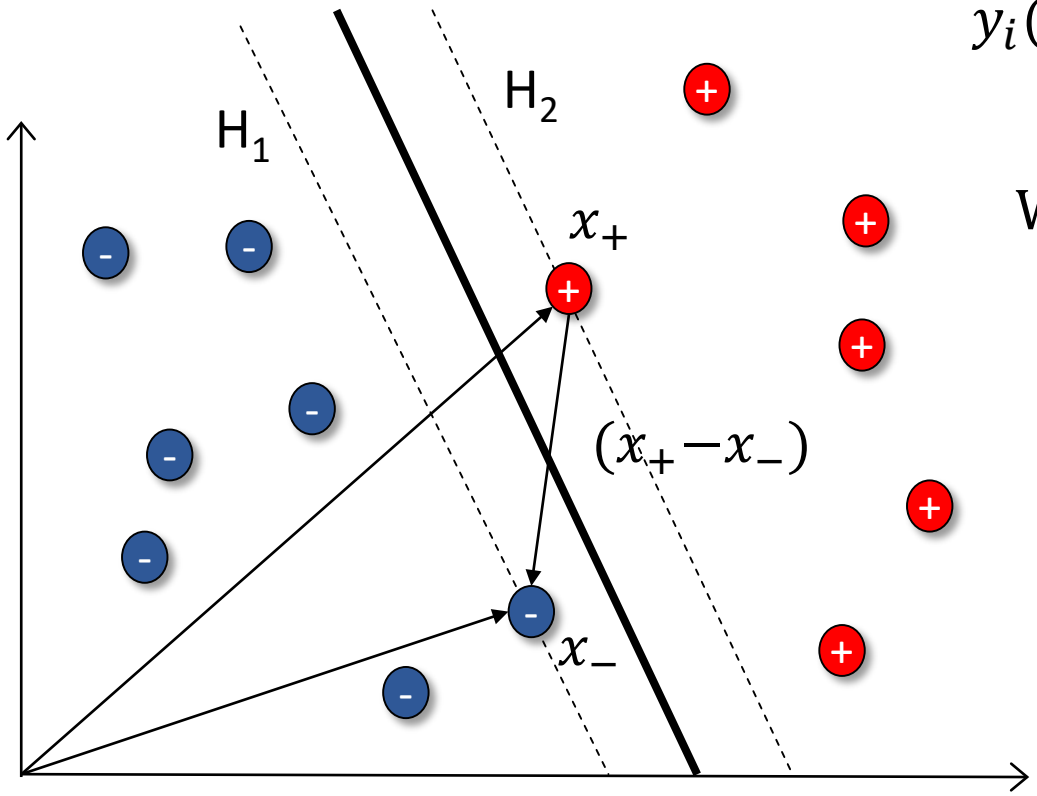
$$y_i(x_{+/-}^T \beta + \beta_0) - 1 \geq 0$$

$$y_i(x_i^T \beta + \beta_0) - 1 = 0$$

For x_i in gutter.

Margin

$$H = \{x | x^T \beta + \beta_0 = 0\}$$



$$y_i(x_{+/-}^T \beta + \beta_0) - 1 \geq 0$$

$$y_i(x_i^T \beta + \beta_0) - 1 = 0 \quad \text{For } x_i \text{ in gutter.}$$

$$\text{Width} = (x_+ - x_-) * \frac{\beta}{\|\beta\|} = \frac{2}{\|\beta\|}$$

The optimal hyperplane is determined by the nearest points of C+ and C-.

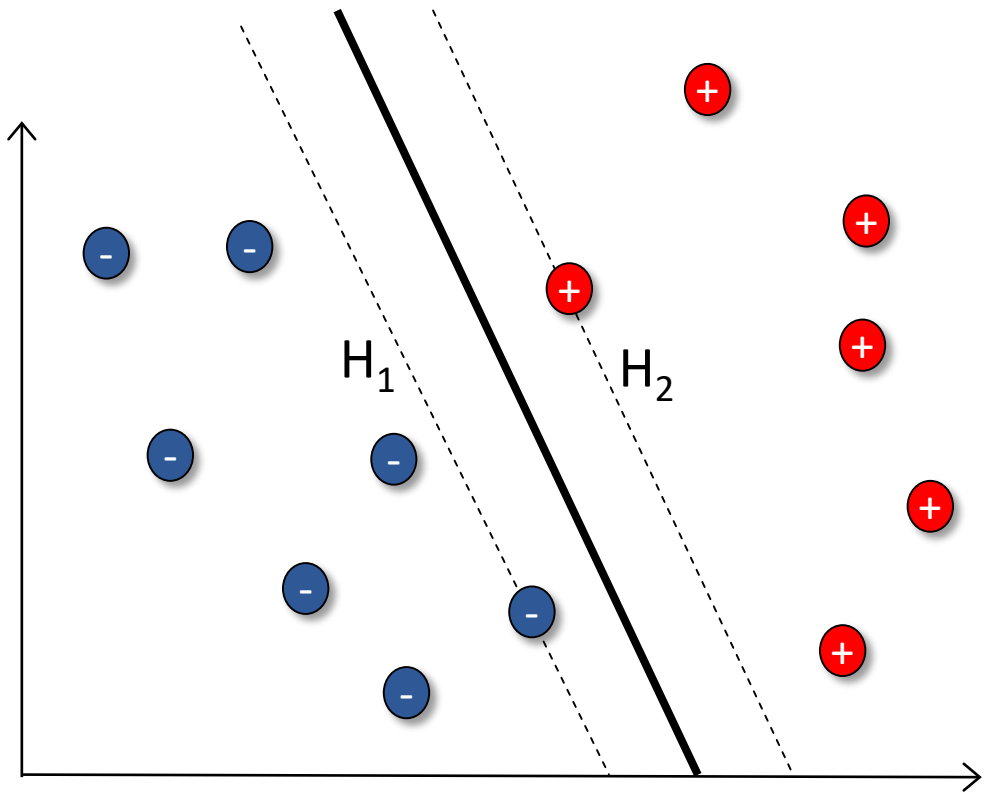
Margin

After construction there is no example between H_1 and H_2, i.e.

$$x^T\beta + \beta_0 \geq +1 \quad \forall \vec{x} \in C_+$$
$$x^T\beta + \beta_0 \leq -1 \quad \forall \vec{x} \in C_-$$

Distance $d(H_1, H_2) = \frac{2}{\left\|\vec{\beta}\right\|}$

is called Margin and should be maximized.



Maximum Margin

By maximizing the margin we find an optimal hyperplane within the set of possible separating hyperplanes.

To maximize $\frac{2}{\left\|\vec{\beta}\right\|}$ we can also minimize $\left\|\vec{\beta}\right\|^2$.

Minimization of $\frac{1}{2}\left\|\vec{\beta}\right\|^2$ is a convex quadratic optimization problem.

There is a clearly defined, optimal hyperplane

$$H = \{x \mid x\beta + \beta_0 = 0\}$$

The quadratic optimization problem can be solved in $O(n^3)$.

Optimization Task

The following optimization task must therefore be solved:

Minimize $\frac{1}{2}\left\|\vec{\beta}\right\|^2$

With constraints

$$\left\langle \vec{x}, \vec{\beta} \right\rangle + \beta_0 \geq +1 \quad \forall \vec{x} \in C_+$$

$$\left\langle \vec{x}, \vec{\beta} \right\rangle + \beta_0 \leq -1 \quad \forall \vec{x} \in C_-$$

The constraints can be summarized as follows

$$y\left(\left\langle \vec{x}, \vec{\beta} \right\rangle + \beta_0\right) - 1 \geq 0 \quad \forall (\vec{x}, y) \in X$$

Overview

- Motivation
- Classification with hyperplanes
- Mathematical formulation
- Soft margin
- Kernel trick
- One-Class SVM
- Summary

Optimierung mit Lagrange

Die Optimierung nach Lagrange ermöglicht die Optimierung einer Funktion f(x) unter Nebenbedingungen durch Relaxation.

Mit der Lagrange-Methode lassen sich Nebenbedingungen g_i und h_j der Art

$$g_i(x) \leq 0 \text{ und } h_j(x) = 0$$

behandeln, indem diese zur optimierenden Funktion f hinzugefügt werden. In dem Fall eines Minimierungsproblems als

$$\min f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{j=1}^{m} \mu_j h_j(x)$$

α_i und μ_j heißen Lagrange-Multiplikatoren.

Lagrange Function

The transformation of the constraints now allows the application of Lagrange:

Given the optimization task minimize $f(\vec{\beta})$

subject to $g_i\left(\vec{\beta}\right) \geq 0 \quad i = 1, \ldots, m$

we can formulate the Lagrange function

$$L(\vec{\beta}, \vec{\alpha}) = f(\vec{\beta}) - \sum_{i=1}^{m} \alpha_i g_i(\vec{\beta})$$

Here $\alpha_i \geq 0$ must apply, equality conditions are not given.

Lagrange Optimization

The constraints g_i are given by

$$g_i\left(\vec{\beta},\beta_0\right) = y_i\left(\left\langle \overrightarrow{x_i},\vec{\beta}\right\rangle + \beta_0\right) - 1 \geq 0 \quad \forall \overrightarrow{x_i} \in X$$

Lagrange's formulation of the optimization problem is also referred to as the Primal Problem:

Primal Problem

Function

$$L_P(\vec{\beta},\beta_0,\vec{\alpha}) = \frac{1}{2}\left\|\vec{\beta}\right\|^2 - \sum_{i=1}^{N}\alpha_i\left(y_i\left(\left\langle x_i,\vec{\beta}\right\rangle + \beta_0\right) - 1\right)$$

L bezüglich $\vec{\beta},\beta_0$ minimiert, bezüglich $\vec{\alpha}$ maximiert werden.

Karush-Kuhn-Tucker Conditions

The partial derivatives $\vec{\beta}$ and β_0

$$\frac{\partial}{\partial \vec{\beta}} L_P(\vec{\beta}, \beta_0, \vec{\alpha}) = \vec{\beta} - \sum_i \alpha_i y_i \vec{x}_i$$

$$\frac{\partial}{\partial \beta_0} L_P(\vec{\beta}, \beta_0, \vec{\alpha}) = -\sum_i \alpha_i y_i$$

Zeroing the derivatives and taking the constraints into account leads to the KKT conditions for a solution für L_P

$$\vec{\beta} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i \quad \text{und} \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i = 1, \ldots, N$$

$$\alpha_i \left(y_i \left(\left\langle x_i, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right) = 0 \quad \forall i = 1, \ldots, N$$

Dual Problem

The primal problem should be minimized with respect to $\vec{\beta}$ and β_0 and maximized with respect to $\vec{\alpha}$.

With the conditions $\frac{\partial L_P}{\partial \vec{\beta}}$ and $\frac{\partial L_P}{\partial \vec{\beta}_0}$ we get the dual Lagrange expression

- Dual Langrage expression L($\vec{\alpha}$) is to be maximized.
- The minimum of the original optimization problem occurs exactly at those values of $\vec{\beta}, \beta_0$ and $\vec{\alpha}$ as the maximum of the dual problem.

Transformation of the primal problem into the dual problem

$$
\begin{aligned}
& \frac{1}{2}\left\|\vec{\beta}\right\|^2 - \sum_{i=1}^{N}\alpha_i \left[y_i\left(\left\langle \vec{x}_i, \vec{\beta}\right\rangle + \beta_0\right) - 1\right] \\
= \; & \frac{1}{2}\left\|\vec{\beta}\right\|^2 - \sum_{i=1}^{N}\alpha_i y_i\left(\left\langle \vec{x}_i, \vec{\beta}\right\rangle + \beta_0\right) + \sum_{i=1}^{N}\alpha_i \\
= \; & \frac{1}{2}\left\|\vec{\beta}\right\|^2 - \sum_{i=1}^{N}\alpha_i y_i\left\langle \vec{x}_i, \vec{\beta}\right\rangle - \sum_{i=1}^{m}\alpha_i y_i \beta_0 + \sum_{i=1}^{N}\alpha_i \\
= \; & \frac{1}{2}\left\|\vec{\beta}\right\|^2 - \sum_{i=1}^{N}\alpha_i y_i\left\langle \vec{x}_i, \vec{\beta}\right\rangle + \sum_{i=1}^{N}\alpha_i
\end{aligned}
$$

Transformation II

Inserting of $\vec{\beta} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$

$$\frac{1}{2}\left\|\vec{\beta}\right\|^2 - \sum_{i=1}^{N} \alpha_i y_i \left\langle \vec{x}_i, \vec{\beta} \right\rangle + \sum_{i=1}^{N} \alpha_i$$

$$= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \vec{x}_i, \vec{x}_j \right\rangle - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \vec{x}_i, \vec{x}_j \right\rangle + \sum_{i=1}^{N} \alpha_i$$

$$= + \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \vec{x}_i, \vec{x}_j \right\rangle$$

Under the constraints: $0 = \sum_{i=1}^{N} \alpha_i y_i$ and $\alpha_i \geq 0$

SVM Optimization Task (Dual Problem)

The transformations lead to the dual problem after the introduction of the KKT conditions:

Dual Problem

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i, \vec{x}_j \right\rangle$$

w.r.t. $\alpha_i \geq 0$ for all i and $\sum \alpha_i y_i = 0$

Support Vectors

Solution $\vec{\alpha}^*$ of Dual Problem

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i, \vec{x}_j \right\rangle$$

must meet the CCT conditions, i.e. among other things

$$\alpha_i \left(y_i \left(\left\langle \vec{x}_i, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right) = 0 \quad \forall i = 1, \ldots, N$$

$\vec{\alpha}^*$ contains for each observation $\overrightarrow{x_i}$ exactly one α_i .

- $0 = \alpha_i$ if $\overrightarrow{x_i}$ in matching half space.
- $0 < \alpha_i$ if $\overrightarrow{x_i}$ on hyperplane H_1 or H_2 (Support Vector).

Optimal Hyperplane

If we have determined the optimal $\vec{\alpha}^*$, we get our optimal hyperplane. The following applies

$$\vec{\beta} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$$

This means the optimal normal vector $\vec{\beta}$ is a linear combination of support vectors.

To determine β_0 we can use

$$\alpha_i \left(y_i \left(\left\langle \vec{x}_i, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right) = 0$$

for any i and our calculated $\vec{\beta}$.

Calculation of α

The basic procedure for SVM is the same as for other learning methods:

1. Parameterization of the models, here via detours through $\overrightarrow{\alpha}$
2. Determination of an optimality criterion, here: Maximum Margin
3. Formulation as an optimization problem

The optimization problem can be solved with different methods, e.g.

- Numerical methods(quadrat. Optimierung)
- Evolutionary Algorithms (Mierswa, 2006)

Summary of Lagrange Optimization for SVM

Lagrange Optimization is defined as

$$L_P(\vec{\beta}, \beta_0, \vec{\alpha}) = \frac{1}{2}\left\|\vec{\beta}\right\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \left(\left\langle x_i, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right)$$

with Lagrange mulitplier $\alpha_i \geq 0$

The conditions required for a minimum are given by the derivation to $\vec{\beta}$ and β_0

$$\frac{\partial}{\partial \vec{\beta}} L_P(\vec{\beta}, \beta_0, \vec{\alpha}) = \vec{\beta} - \sum_i \alpha_i y_i \vec{x}_i \qquad \frac{\partial}{\partial \beta_0} L_P(\vec{\beta}, \beta_0, \vec{\alpha}) = -\sum_i \alpha_i y_i$$

These lead to the dual problem:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i, \vec{x}_j \right\rangle$$

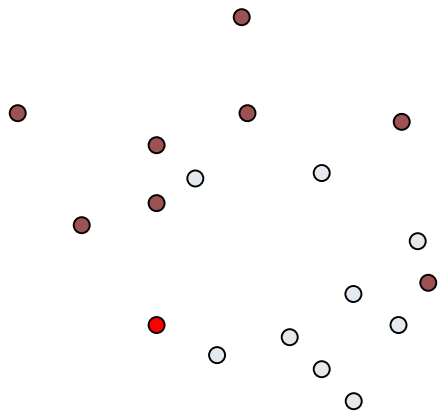
Overview

- Motivation
- Classification with hyperplanes
- Mathematical formulation
- Soft margin
- Kernel trick
- One-Class SVM
- Summary

What do we know now?

- Maximizing the margin of a hyperplane results in a clear definition of the optimal separating hyperplane.
- We have to minimize the length of the normal vector w.
  - Formulation as Lagrange function
  - Formulation as a dual optimization problem
- The learning outcome is a linear combination of support vectors.
- We only have to calculate the scalar product with the examples.

In practice, linearly separable data is rare.



Naive approach:

Remove a minimal amount of data points so that the data becomes linearly separable (minimal misclassification).

Problem: Algorithm becomes exponential.

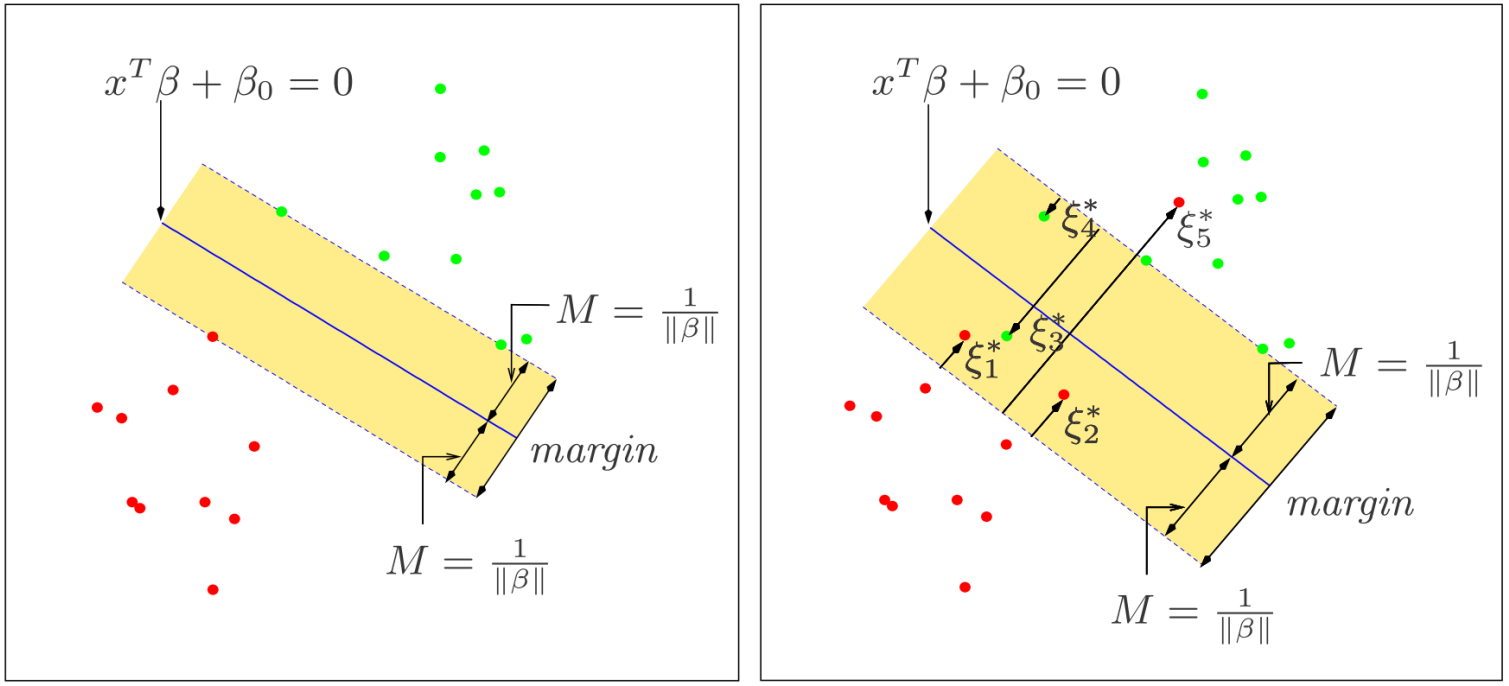


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

Hastie, T., Tibshirani, R., & Friedman, J. H. (2017). *The elements of statistical learning: Data mining, inference, and prediction* (12th ed). *Springer series in statistics*. New York: Springer.

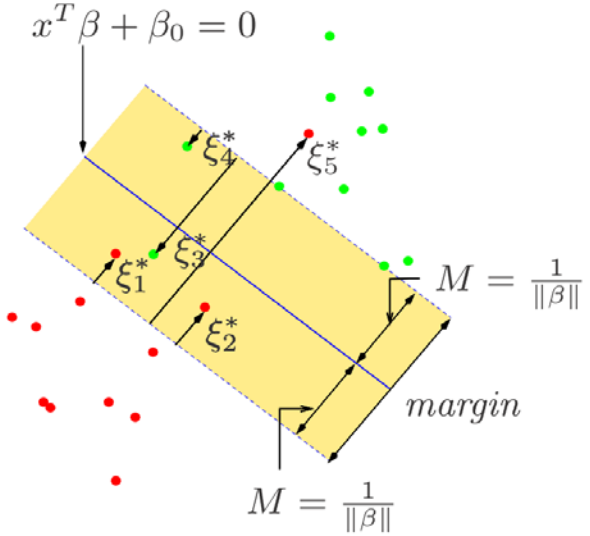
Relaxation

We allow for some points to be on the wrong side of the margin.

Define the slack variables $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$

$$y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i$$

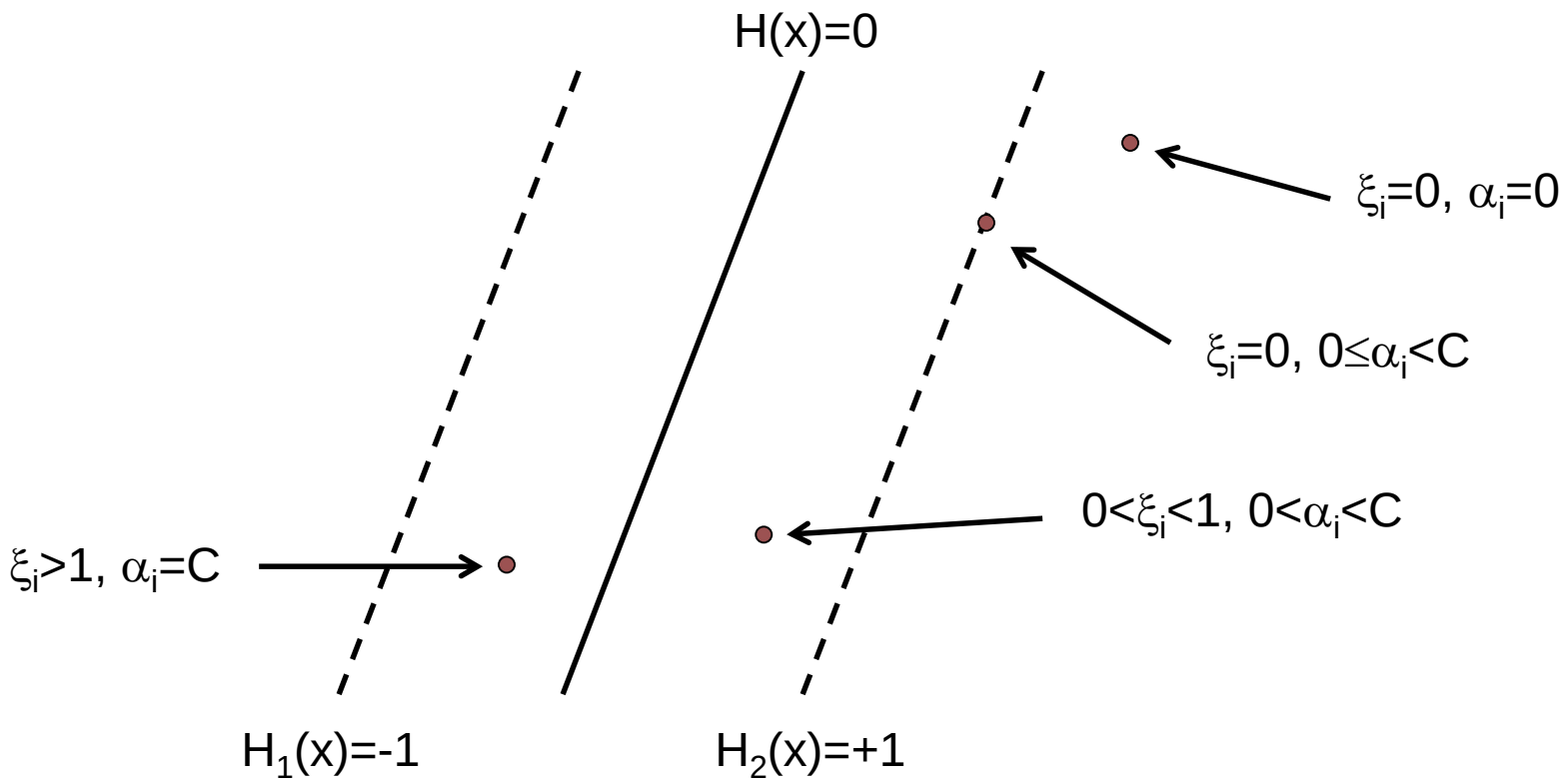
with $\xi_i \geq 0 \;\; and \; \sum_i \xi_i \leq Constant \; C$



Optimization problem: $\frac{1}{2}\left\|\vec{\beta}\right\|^2 + C\sum_{i=1}^{n}\xi_i$

The solution for ß has the form $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$

Meaning of ξ and α

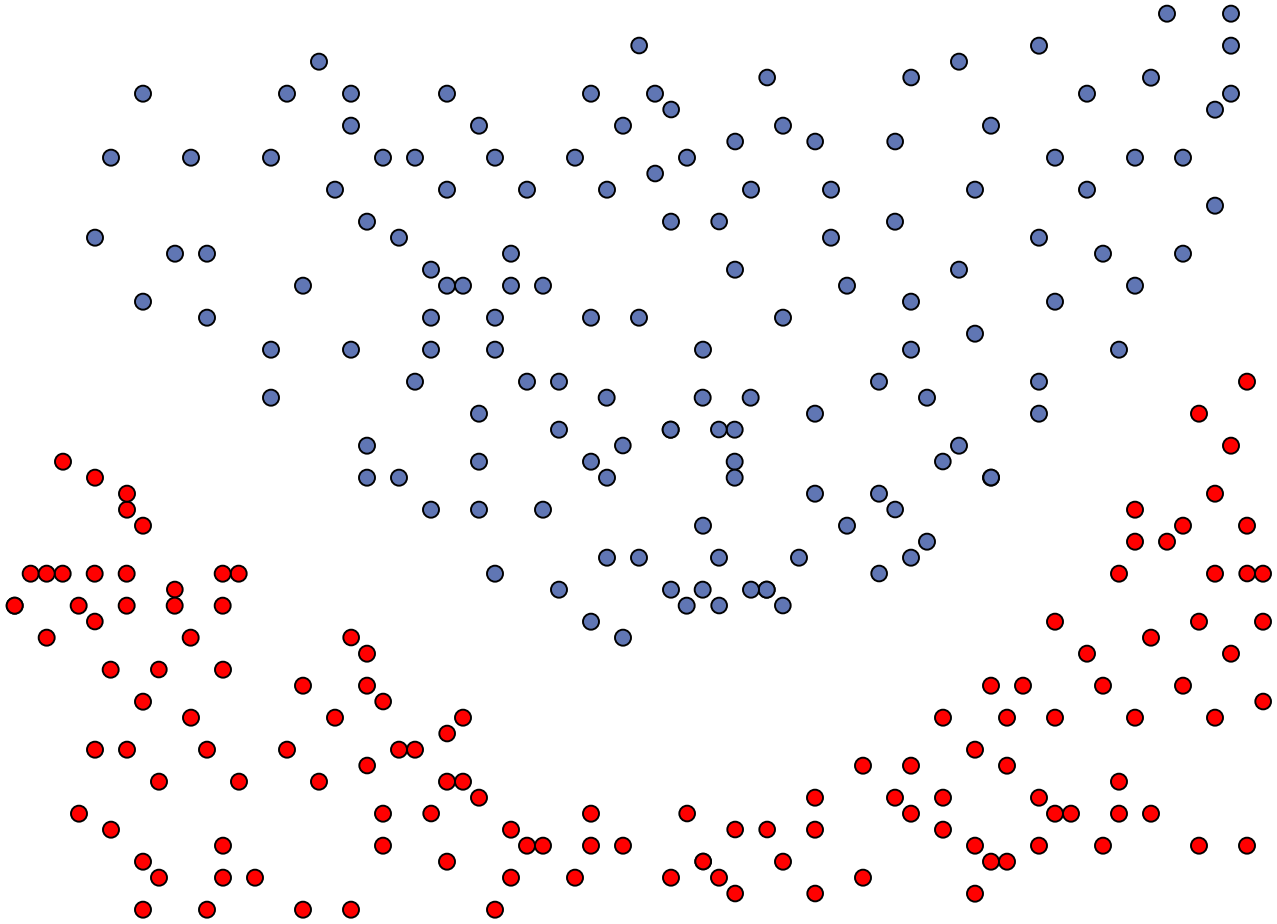


Data points x_i with $\alpha_i>0$ are called Support Vectors → SVM

Overview

- Motivation
- Classification with hyperplanes
- Mathematical formulation
- Soft margin
- Kernel trick
- One-Class SVM
- Summary

Nonseparable Classes



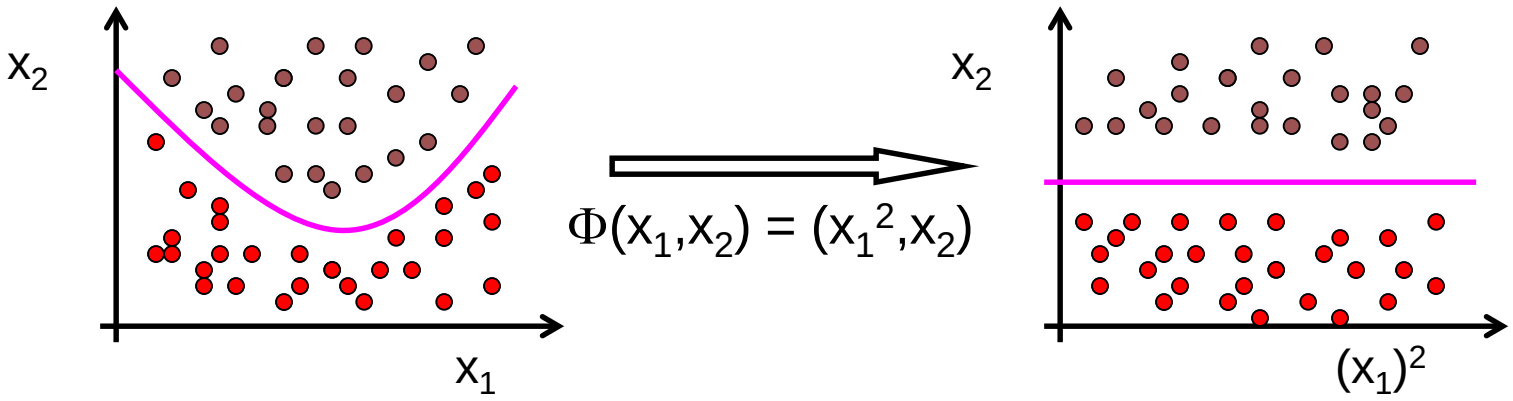
Nonseparable Classes – What can we da=

Develop new SVM theory?

Use linear SVM?

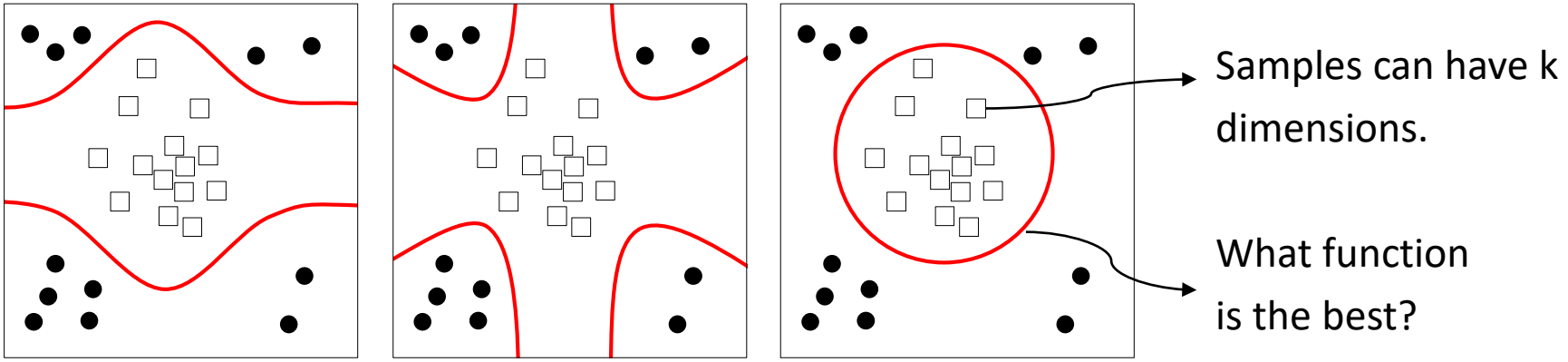
„If all you‘ve got is a hammer, every problem looks like a nail.“

Transformation in a lineare problem!



Higher dimensional Feature Space

Non-linear hyperplane separates data „seriously inseparable“

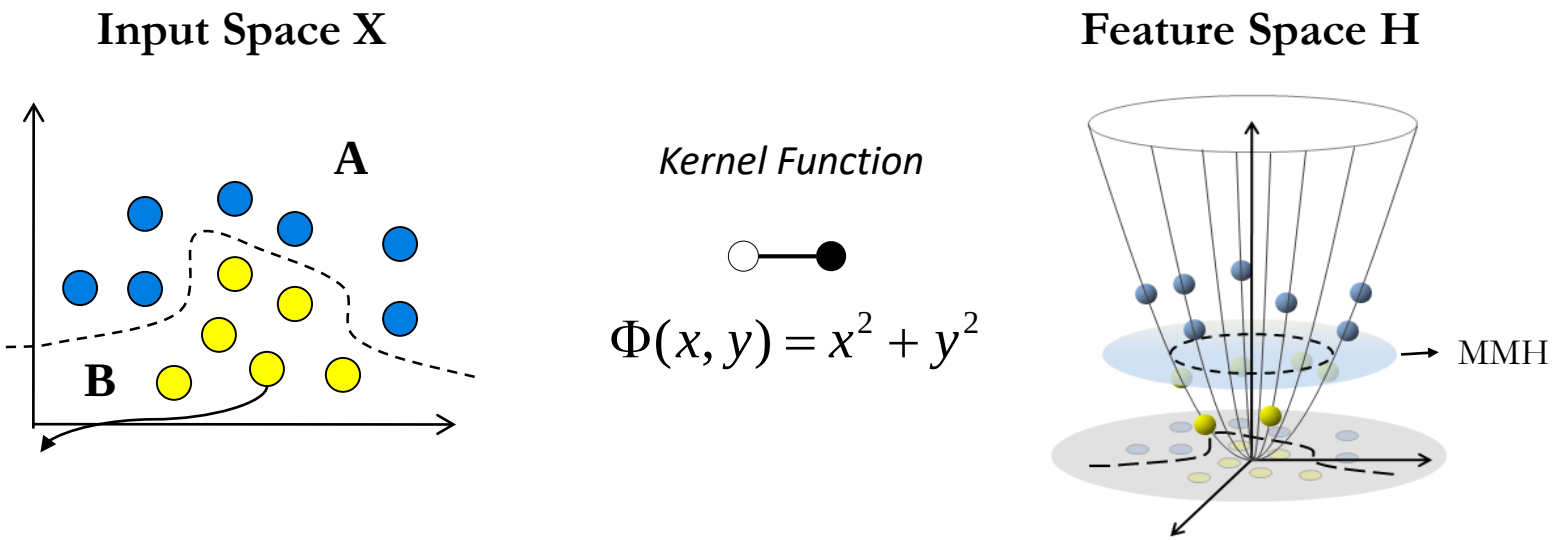


Two naive solutions:

1. Remain in the dimensions of the training set and take a non-linear complex function.
2. Transformation into a higher dimension "feature space" that makes the data set linearly separable.

Transformation into Feature Space H

Example: $\Phi(x, y) = x^2 + y^2$



Nonr separable in low dimensions

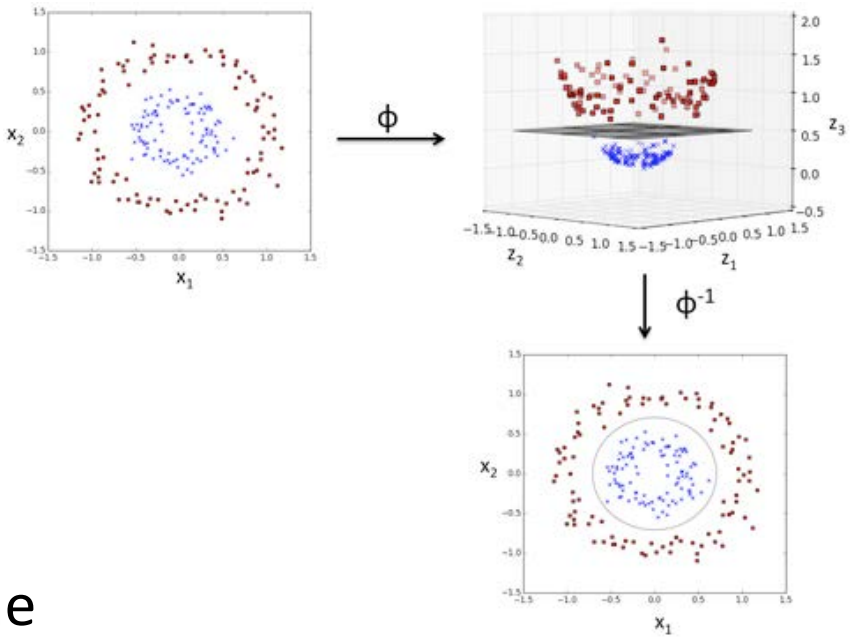
linear separable in higher dimensions

Feature Space

Data points are transformed with a function ("feature map"):

$$\Phi : X \to H$$

$$x \mapsto \Phi(x)$$



H is a space ("feature space") in which a scalar product is explained.

In H, $\Phi(x)$ are separated with SVM.

Dimension of H is much larger than the original space with the attributes (and most of these are already very many!).

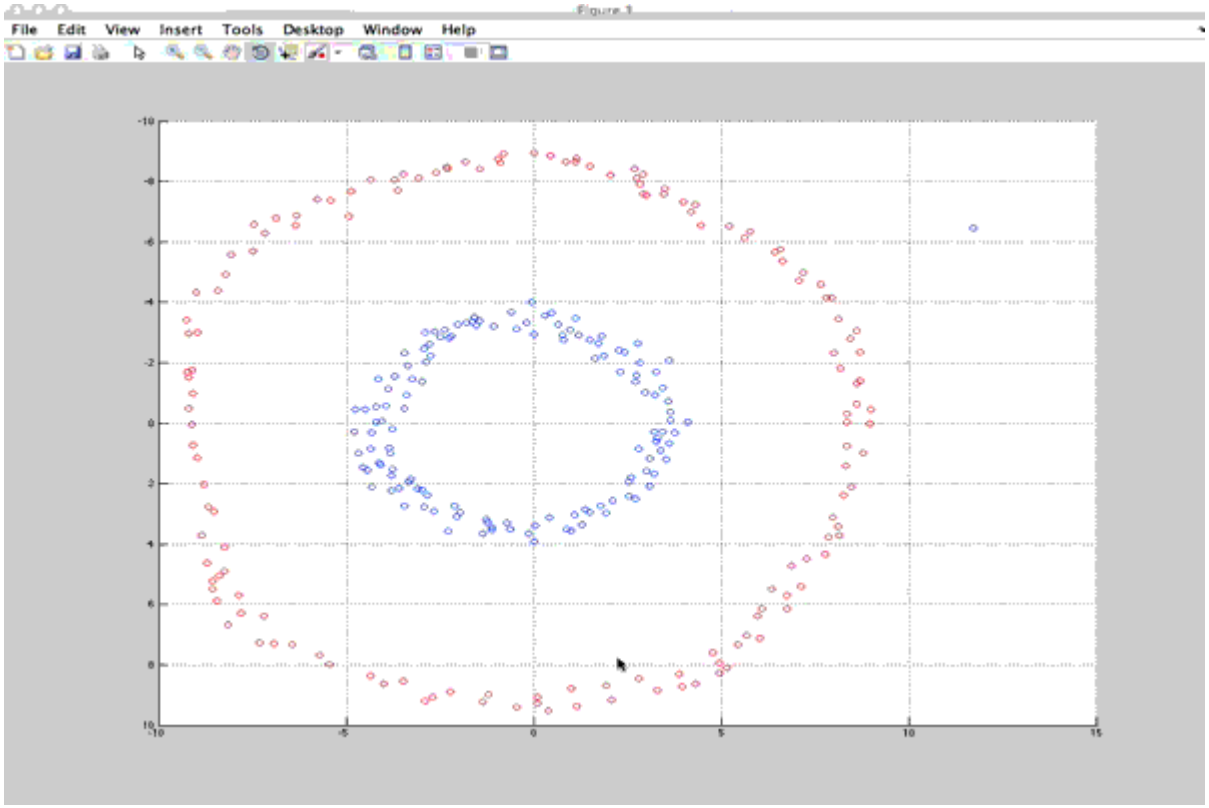
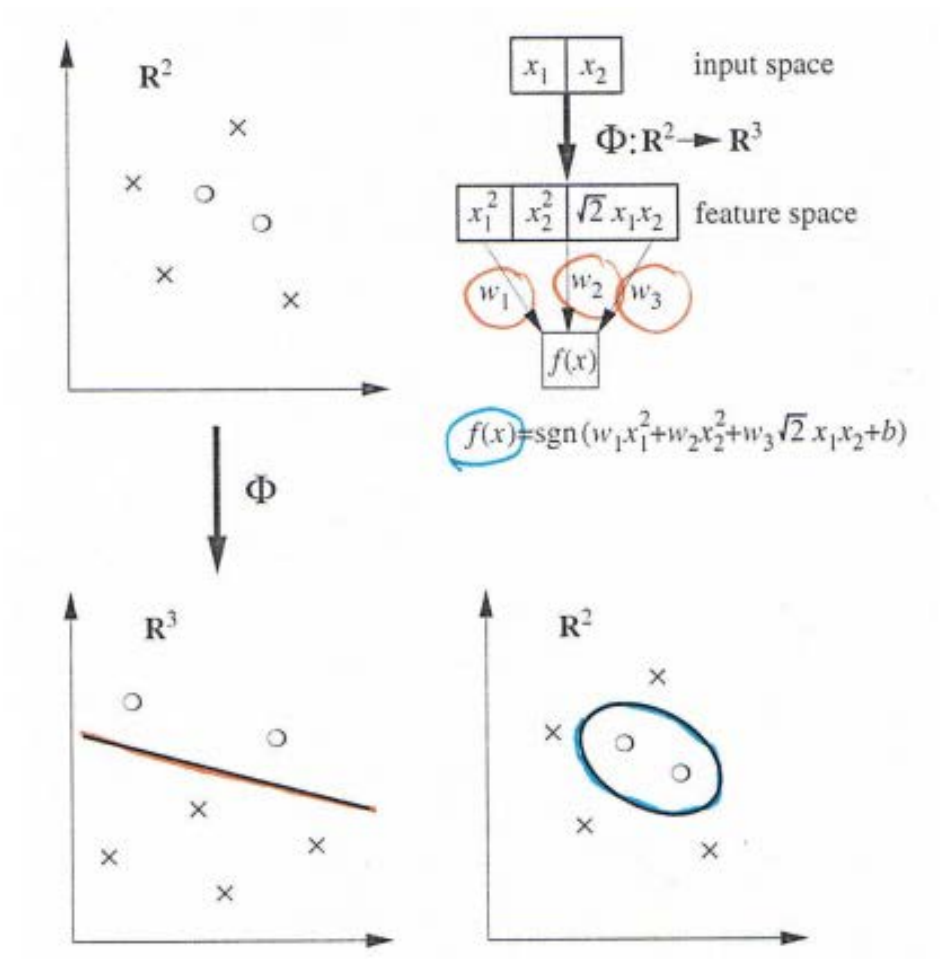


Figure 1
File Edit View Insert Tools Desktop Window Help

Example

ursprüngliche Daten



höher dimensionaler Raum $\mathcal{M}$ *(hier:* $\mathcal{M} = \mathbb{R}^3$*)*

nichtlineare Entscheidungs-fläche im ursprünglichen Raum.

Hyperplane with maximum separation in Feature Space

If classification in high-dimensional spaces is really easier, we want to construct the separating hyperplane there.

Then we have to calculate scalar products $\left\langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x'}) \right\rangle$.

This can be very difficult / impossible if the dimension of H becomes too large.

Solution

Instead of the scalar product, we use a kernel function K, which lives in X but behaves like a scalar product in H.

$$\mathrm{K}\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left\langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x'}) \right\rangle$$

Kernel function

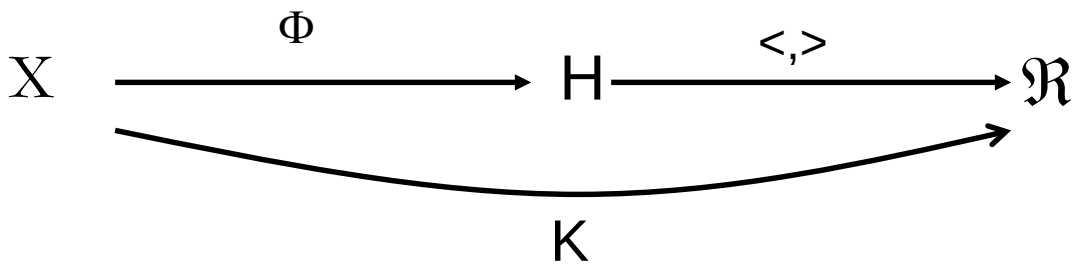
Recall:

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i, \vec{x}_j \right\rangle$$

$$f(\vec{x}) = \sum_{i=1}^{m} \alpha_i y_i \left\langle \vec{x}_i, \vec{x} \right\rangle + \beta_0$$

SVM depends on x only via scalar product $\left\langle \vec{x}, \overrightarrow{x'} \right\rangle$.

Replace transformation Φ and scalar product with kernel function $K\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left\langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x'}) \right\rangle$, because transformation is very complex and computational.



Kernel Trick

The feature space is very large. Mapping the examples into the characteristic space and then calculating the scalar products between the transformed examples is very inefficient.

A core function that delivers the same result directly applied to the examples would be efficient!

$$K\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left\langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x'}) \right\rangle$$

$$\Phi: X \rightarrow \mathcal{H}$$

The scalar product of the vectors in characteristic space $\mathcal{H}$ corresponds to the value of the core function above the examples.

What functions make $K\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left\langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x'}) \right\rangle$ true?

Kernel functions

Linear $$\mathrm{K}\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left\langle \vec{x}, \overrightarrow{x'} \right\rangle$$

Polynomiell $$\mathrm{K}\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left(\gamma \left\langle \vec{x}, \overrightarrow{x'} \right\rangle + c_0\right)^d$$

Radial basic function $$\mathrm{K}\left(\overrightarrow{x}, \overrightarrow{x'}\right) = exp\left(-\gamma \left\| \vec{x} - \overrightarrow{x'} \right\|^2\right)$$

Construction of special cores by sums and products of kernel functions, multiplication by positive numbers, omission of attributes.

Skalar product

$\Phi_2: \Re^2 \rightarrow \mathcal{H}^4$

Skalar product:

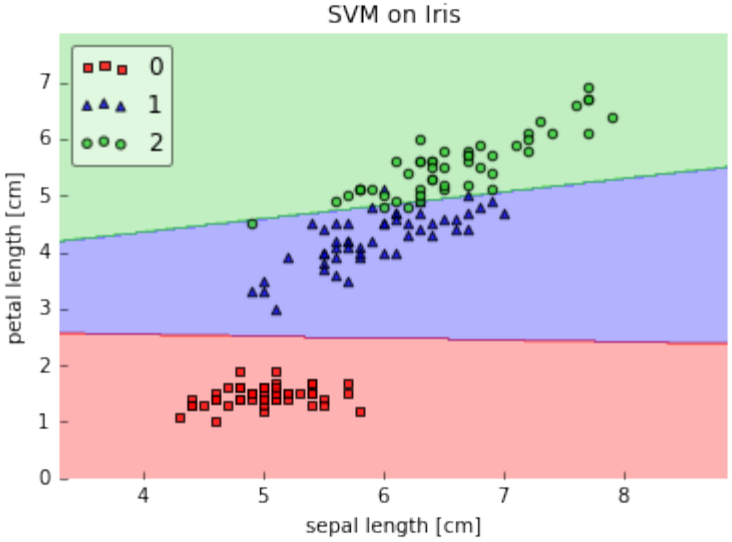
$$\Phi_2(x):(x_1,x_2)\rightarrow(x_1^2,x_2^2,x_1x_2,x_2x_1)$$

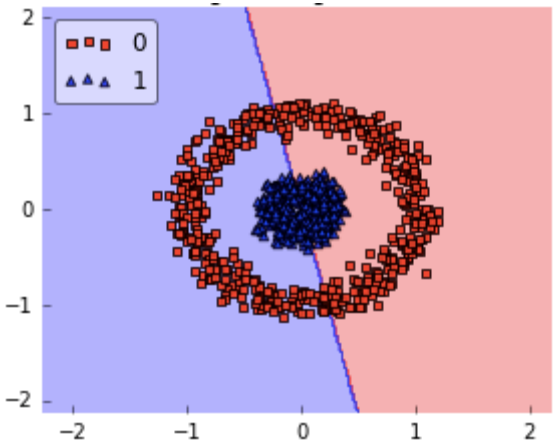
$$\begin{aligned}\langle\Phi_2(x),\Phi_2(x')\rangle &= x_1^2x'^2_1+x_2^2x'^2_2+2x_1x_2x'_1\,x'_2\\ &= \langle x,x'\rangle^2\end{aligned}$$

This generally applies to all ordered products of d-th degree of the components of x: $\left\langle\Phi(\overrightarrow{x}),\Phi(\overrightarrow{x'})\right\rangle = \langle x,x'\rangle^d$

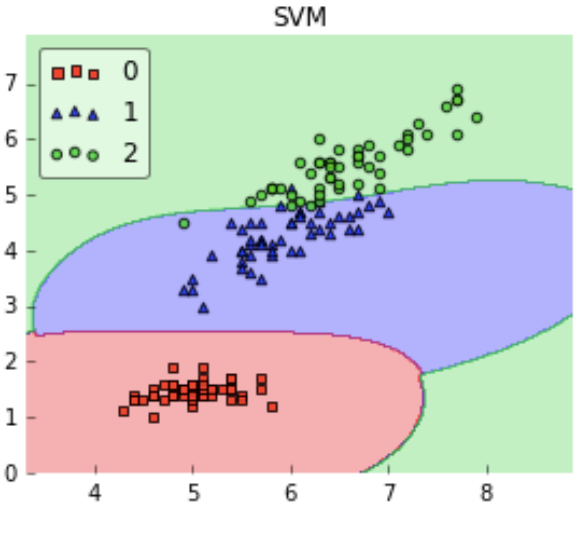
Examples

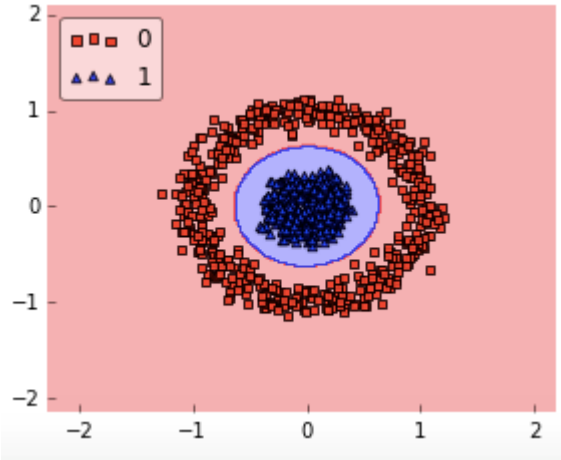
SVM with linear function





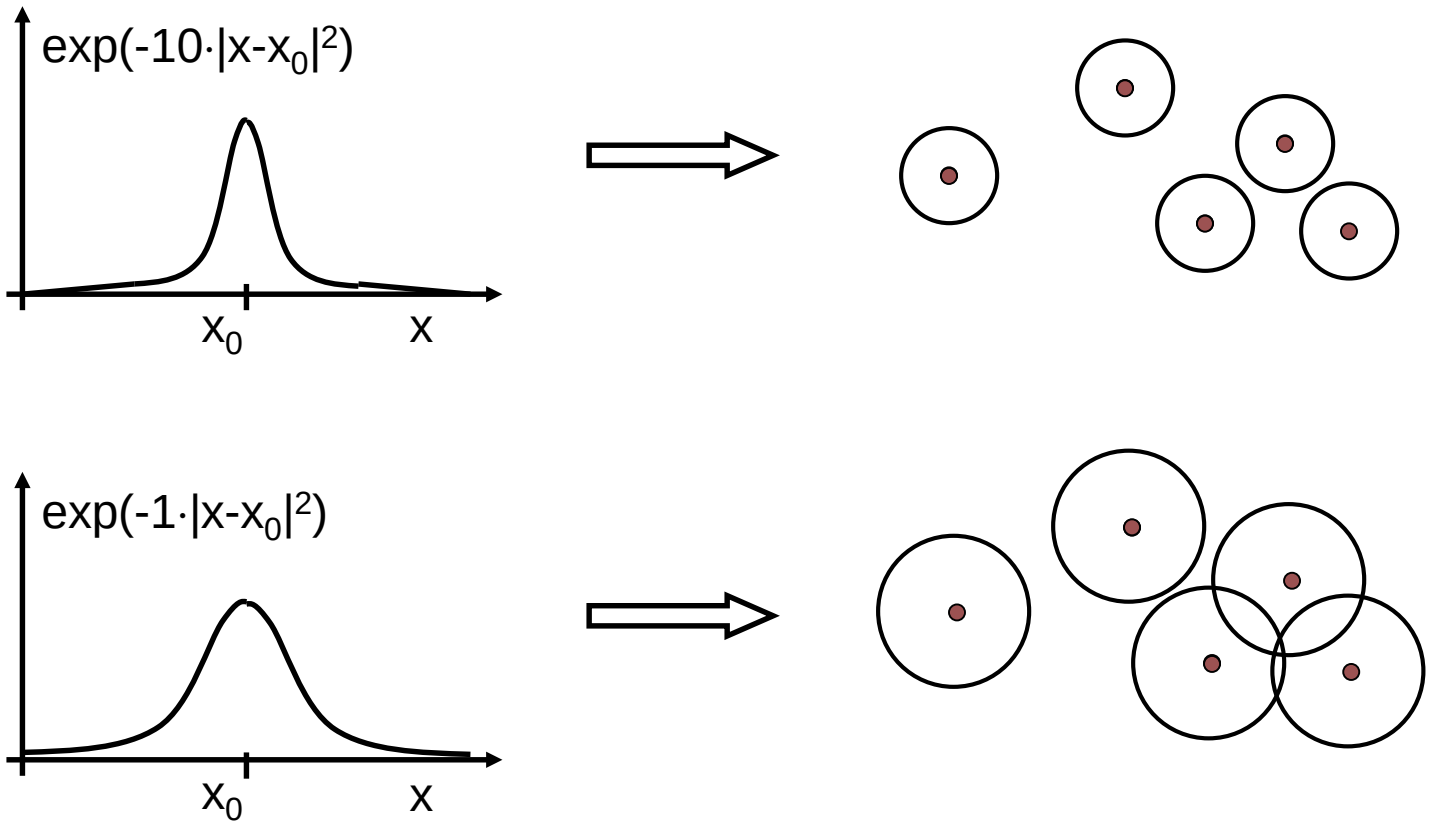
SVM with RBF





Images from http://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html

RBF kern function



Kernel function

- The kernel functions are not performed as a preprocessing step.
- You only have to consider the kernel function when calculating the scalar product.
- However, ß can now no longer be interpreted as the meaning of the variables (characteristics) X_i.

Why is this a trick?

We don't need to know what the feature space really looks like.

We just need the kernel function as a measure of similarity.

Another black box:

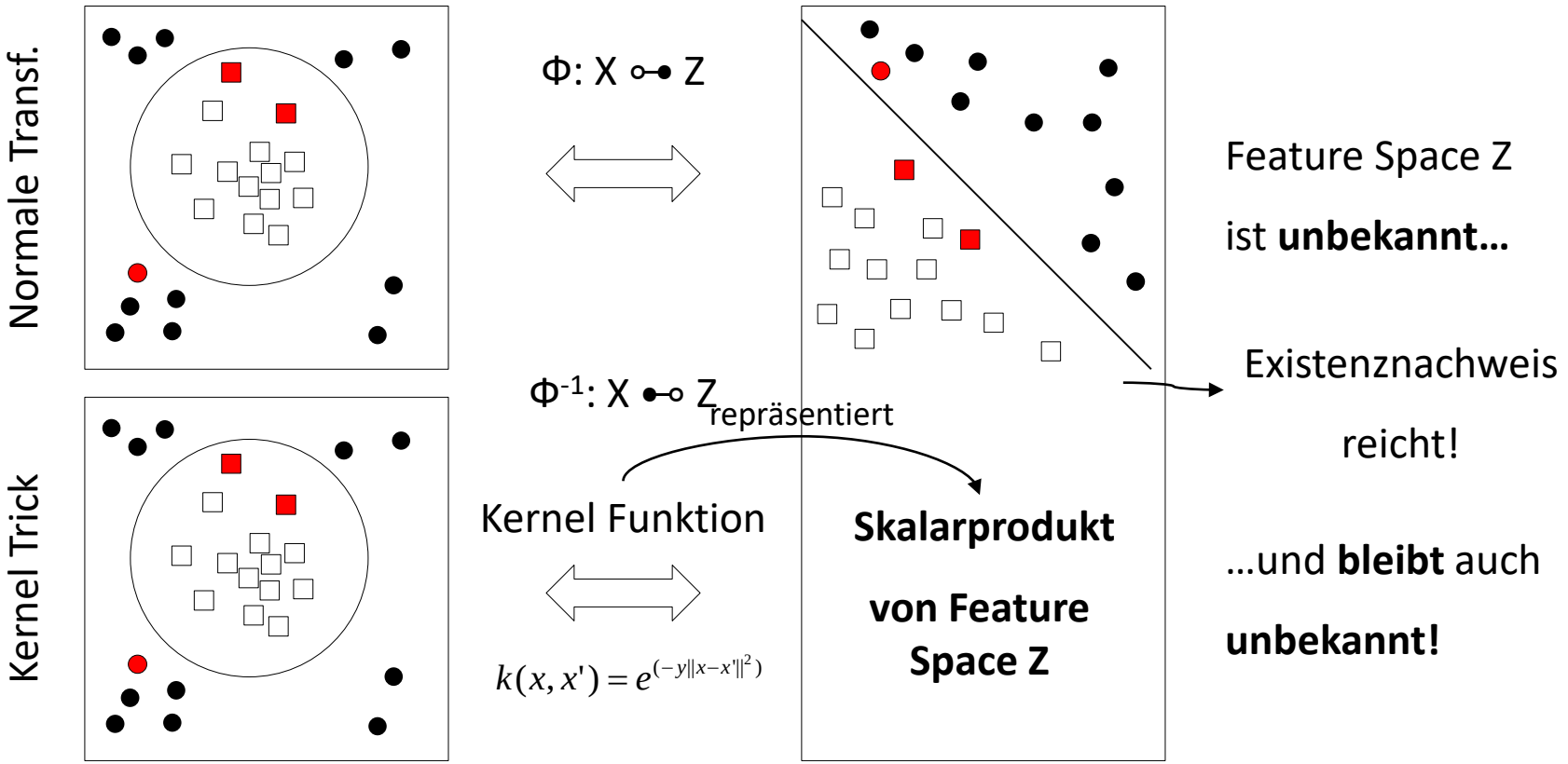
We don't know what happens in the kernel, we are only interested in the result.

Nevertheless, we have the geometric interpretation of the separating hyperplane:

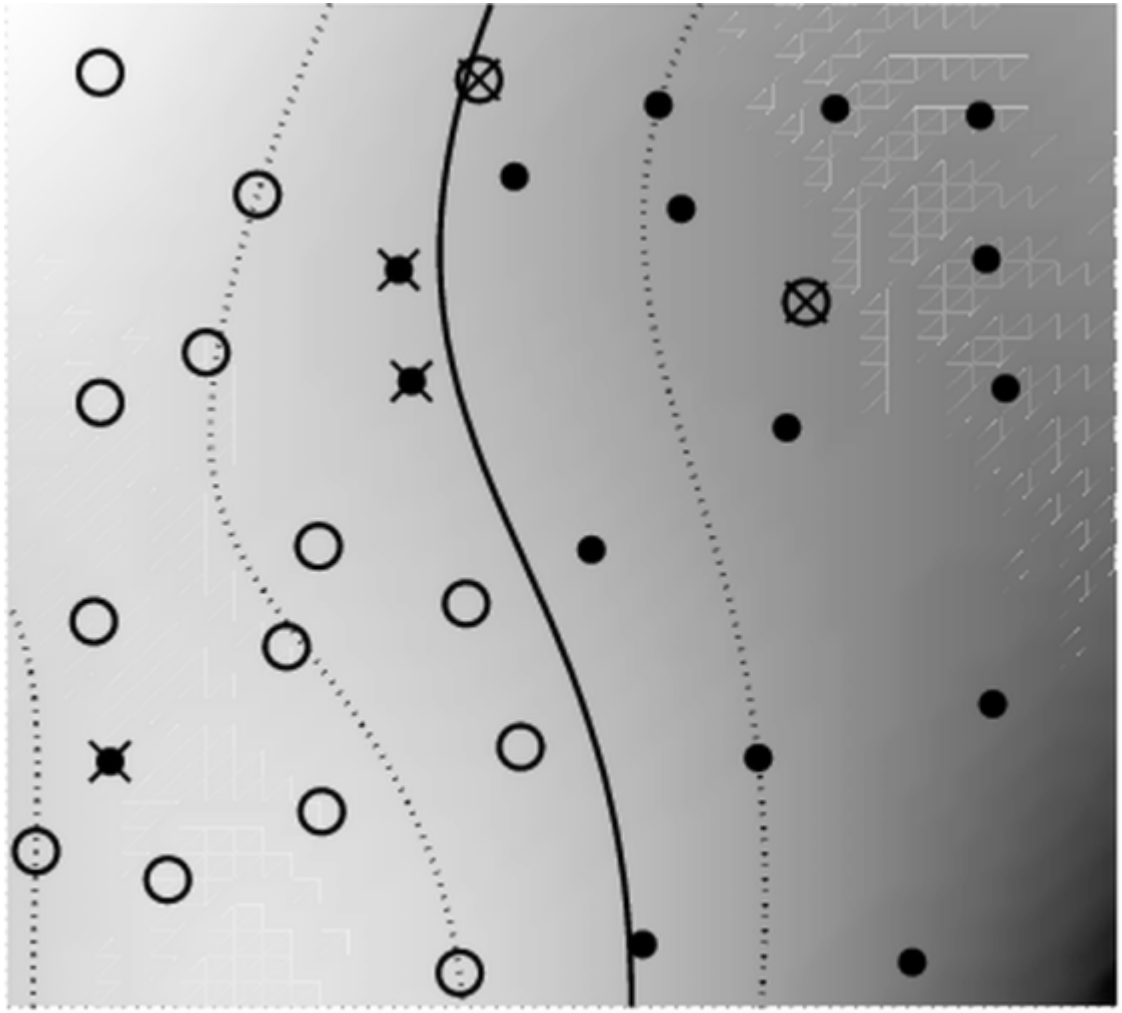
SVMs are more transparent than artificial neural networks, for example.

Der Kernel-Trick

Transformation ohne zu transformieren

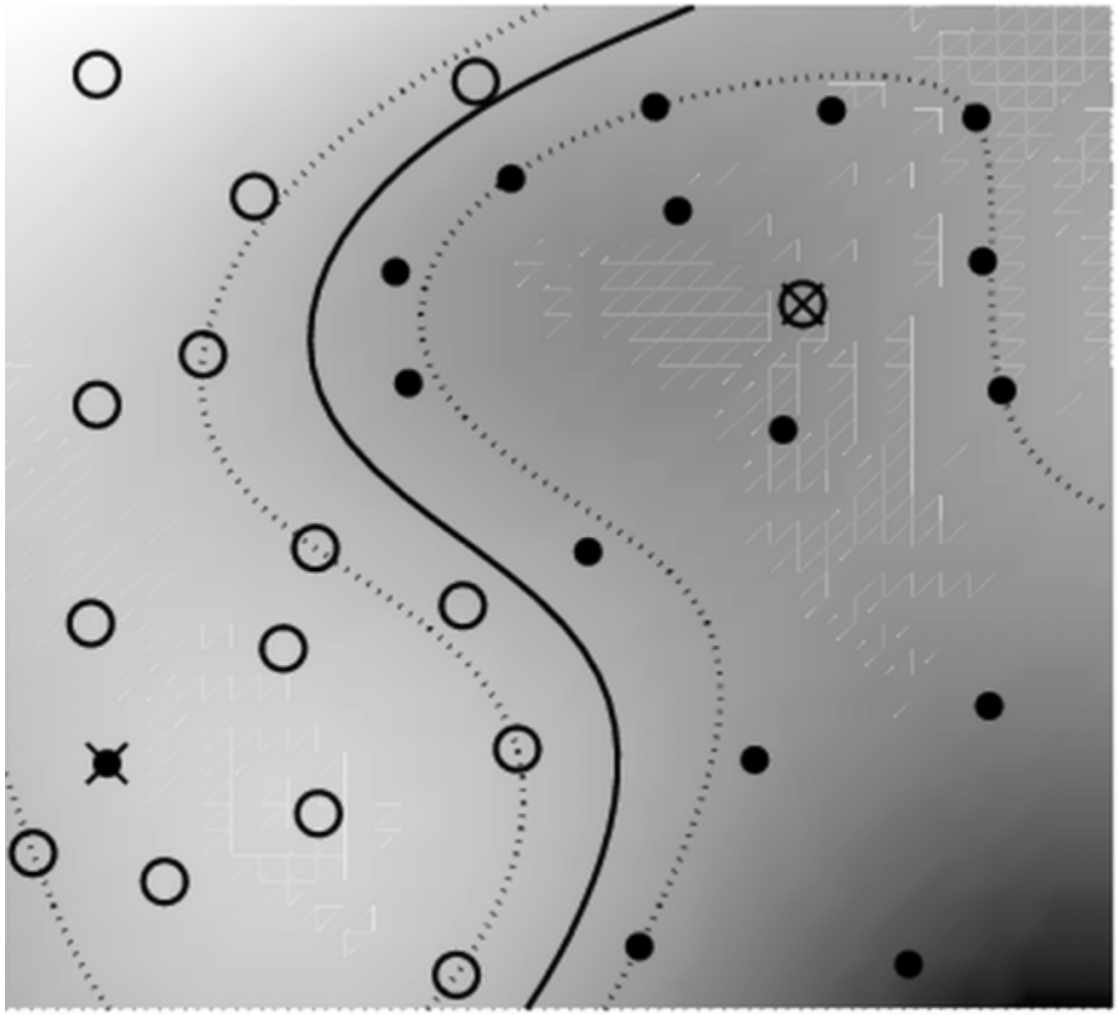


Beispiel: niedrige Komplexität



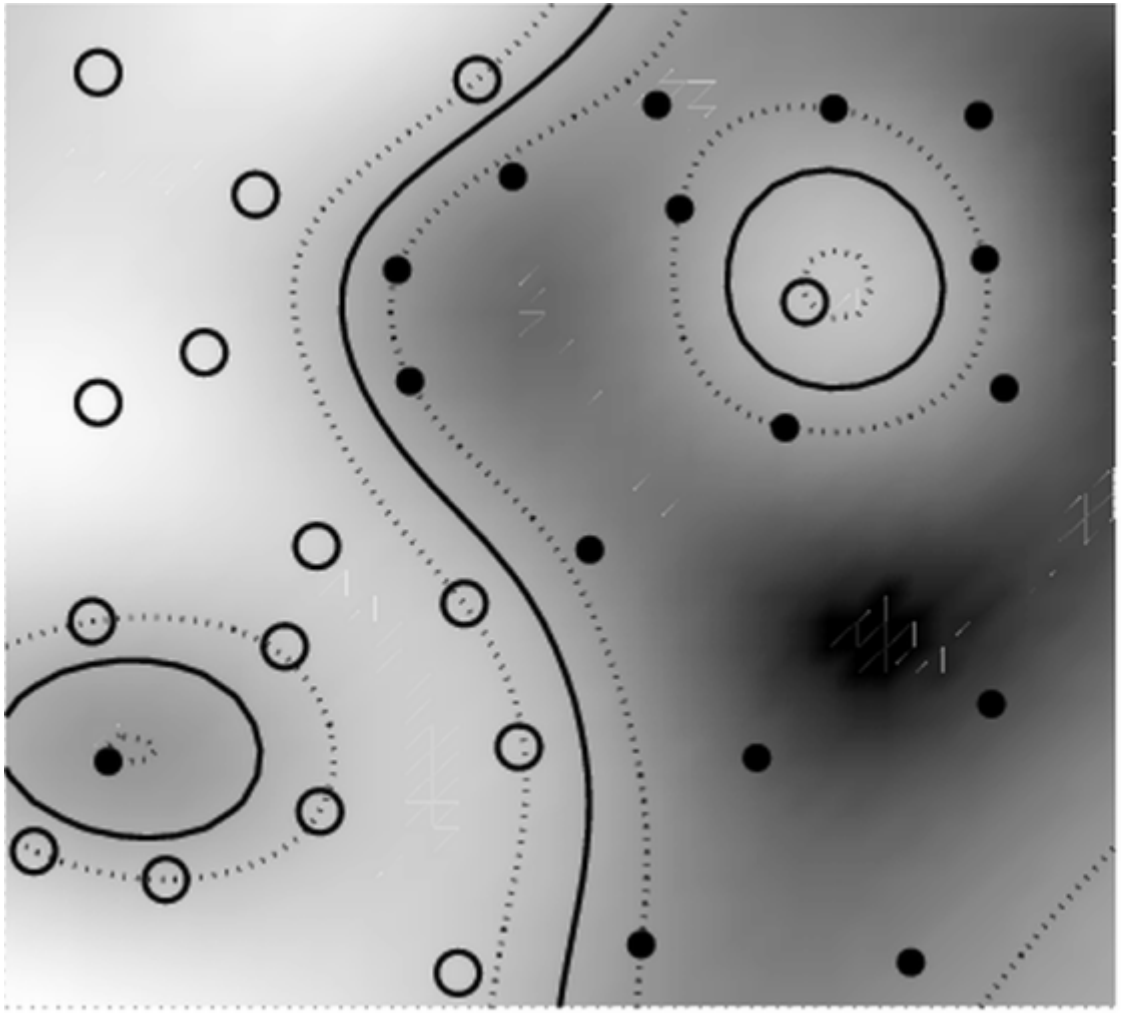
aus: SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217

Mittlere Komplexität



aus: SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217

Hohe Komplexität



aus: SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217

Summary: Kernel

- Kernel functions calculate the scalar product of the observations in a characteristic space without actually mapping it into the characteristic space. $k(x,x') = \Phi(x) * \Phi(x')$
- Polycore and RBF core as examples.
- The kernel trick: k(x,x') can only be calculated from x*x'.
- A function $X \times X$, which can be represented for all x_i in X as a positively definite gram matrix with symmetrical function k, is called kernel function.
- The mercer condition checks if the function is a kernel function, i.e. if the matrix is positive.

Overview

- Motivation
- Classification with hyperplanes
- Mathematical formulation
- Soft margin
- Kernel trick
- One-Class SVM
- Summary

One-Class SVM

- In one-class SVM, the support vector model is trained on data that has only one class, which is the “normal” class.
- We have a collection of positive examples only.
- It infers the properties of normal cases and from these properties can predict which examples are unlike the normal examples.
- This is useful for anomaly detection because the scarcity of training examples is what defines anomalies:
  that is, typically there are very few examples of the network intrusion, fraud, or other anomalous behavior.

Proposed Approach

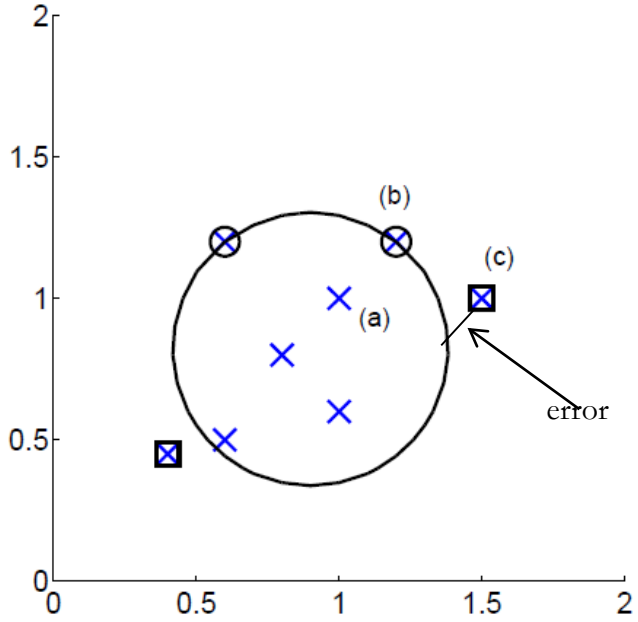
- Training data $\mathrm{X} = \{x_1, x_2, \ldots, x_m\}$
- Try to fit a tight hyper-sphere (in a transformed space) to include most positive training examples.
- Such hyper-sphere tries to capture the support within which the positive examples are clustered, in an effort to separate them from “the rest of the world”.
- The hyper-sphere will include most, but not all, training data to avoid overfitting.
- Consider a sphere with center a and radius R
- Minimize R and the error resulting from points outside the sphere—their error is their distance to the sphere.

- Consider a sphere with center a and radius R
- Minimize R and the error resulting from points outside the sphere—their error is their distance to the sphere.

$$\min R^2 + \mathrm{C}\sum_t \xi^t$$

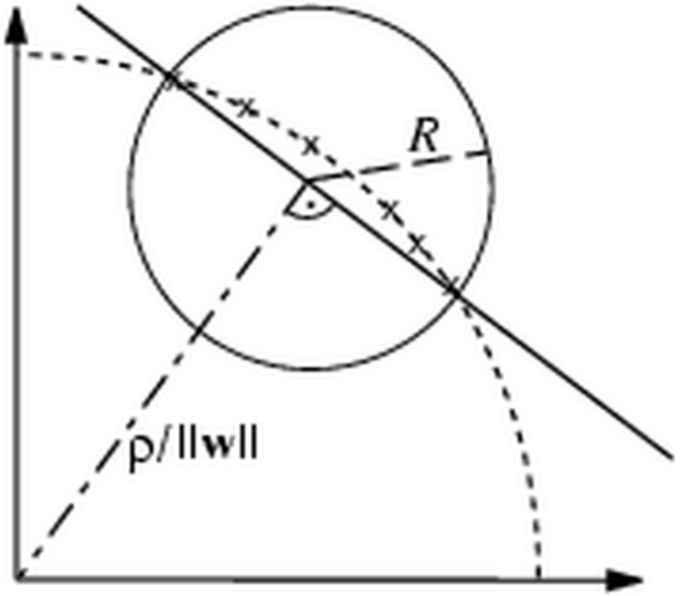
subject to

$$\left\|\mathbf{x}^t - a\right\| \le R^2 + \xi^t, \xi^t \ge 0$$



Finding the smallest sphere enclosing the data is equivalent to maximizing the margin of separation from the origin.

In contrast to traditional SVMs, one-class SVMs attempt to learn a decision boundary that achieves the maximum separation between the points and the origin



Overview

- Motivation
- Classification with hyperplanes
- Mathematical formulation
- Soft margin
- Kernel trick
- One-Class SVM
- Summary

Summary

Advantages of SVMs:

- have solid mathematical foundations
- are robust and fast
- hardly have overfitting (because VC dimension is reduced)
- can classify linearly or non-linearly
- are not based on probabilities
- are extendable by own kernel functions
- are very well supported (Rapid Miner 5, R, Matlab etc.)
- are very well documented
- are intuitive geometrically understandable

Zusammenfassung

Disadvantages of SVMs:

- are to be understood as a trial-and-error approach
- Previous knowledge of data volume can be poorly specified
- can have immense runtimes with wrong choice of kernel
- can only handle numeric attributes
- assume that you are familiar with SVMs

Applications with SVMs

- (Social-)Marketing, Recommender Systeme, Opinion Mining
- Fraud Dedection
- Classification
- Astro Physics, Biologie (microarray expression profiles), Massenspektren
- Shipping (Sonar)
- …

e1071 Packages in R mit SVM

model <- svm(Species ~ ., data = iris)

`kernel`	the kernel used in training and predicting. You might consider changing some of the following parameters, depending on the kernel type. **linear:** $u'v$ **polynomial:** $(\gamma u'v + coef0)^{degree}$ **radial basis:** $e^{(} - \gamma \lvert u - v \rvert^2)$ **sigmoid:** $tanh(\gamma u'v + coef0)$
`degree`	parameter needed for kernel of type `polynomial` (default: 3)
`gamma`	parameter needed for all kernels except `linear` (default: 1/(data dimension))
`coef0`	parameter needed for kernels of type `polynomial` and `sigmoid` (default: 0)
`cost`	cost of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation.

Zusammenfassung