Support Vector Machines

André Hopfgartner & Matthias Rupp

08.06.2021

Vorarlberg University of Applied Sciences

FH Vorarlberg
University of Applied Sciences

Agenda

1. Einführung
2. Hard-Margin Support Vector Machine
3. Lösung mittels QP-Solver
4. Soft-Margin Support Vector Machine
5. Vergleich Hard- & Soft-Margin Support Vector Machine
6. Nichtlineare Trennung
7. Pseudocode und Beispiele

Einführung

Intuition

Ziel: lineare Trennung zweier Klassen

Intuition

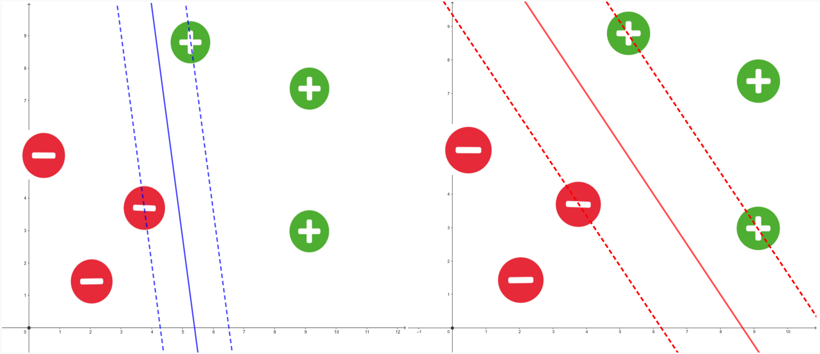
Ziel: lineare Trennung zweier Klassen
Wie?: Definition einer (Hyper-) Ebene

Intuition

Ziel: lineare Trennung zweier Klassen
Wie?: Definition einer (Hyper-) Ebene
Nebenbedingung: Möglichst großer freier Bereich

Intuition

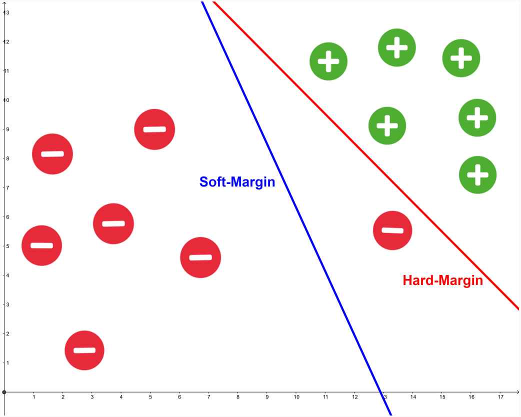
Ziel: lineare Trennung zweier Klassen
Wie?: Definition einer (Hyper-) Ebene
Nebenbedingung: Möglichst großer freier Bereich



Arten von SVM

Arten von SVM:

- *Hard-Margin SVM*: Daten werden 100% korrekt getrennt
- *Soft-Margin SVM*: Einzelne Datenpunkte können falsch klassifiziert werden um insgesamt bessere Trennung zu erhalten



Hard-Margin Support Vector Machine

Gegeben sei ein Gewichtsvektor $w \in \mathbb{R}^K$, ein Bias $b \in \mathbb{R}$, ein beliebiger Punkt $x_n \in \mathbb{R}^K$ und ein zugehöriges Label $y_n \in \{-1, +1\}$. Eine Ebene im Raum kann allgemein definiert werden durch:

$$w^T x_n + b = 0$$

Ziel der SVM: w und b bestimmen für optimale Trennung

Klassifikation

Annahme: w und b bereits bekannt

Wie klassifiziert man einen Punkt x_n?

Annahme: w und b bereits bekannt
Wie klassifiziert man einen Punkt x_n?
Liegt x_n über oder unter Ebene = Vorzeichen:

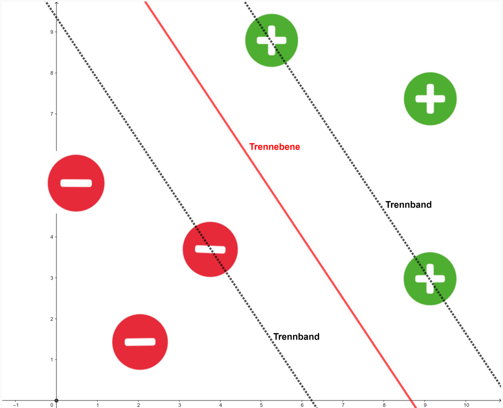
$$
\begin{aligned}
y = sign(w^T x_n + b) &\qquad \text{ist gleichbedeutend mit} \\
w^T x_n + b > 0 &\qquad \text{für } y_n = +1 \\
w^T x_n + b < 0 &\qquad \text{für } y_n = -1
\end{aligned}
$$

Bisher: Punkte können genau auf der Grenze liegen wenn $w^T x_n + b = 0$

Einführung eines Trennbandes

Striktere Regel: Um Ebene soll Band frei bleiben

$$
\begin{aligned}
w^T x_n + b &\geq +1 \qquad \text{für } y_n = +1 \\
w^T x_n + b &\leq -1 \qquad \text{für } y_n = -1
\end{aligned}
$$



Einführung eines Trennbandes

Beidseitige Multiplikation mit y_n

$$\begin{aligned} y_n(w^T x_n + b) &\geq 1 \qquad &&\text{für } y_n = +1 \\ y_n(w^T x_n + b) &\geq 1 \qquad &&\text{für } y_n = -1 \end{aligned}$$

Einführung eines Trennbandes

Beidseitige Multiplikation mit y_n

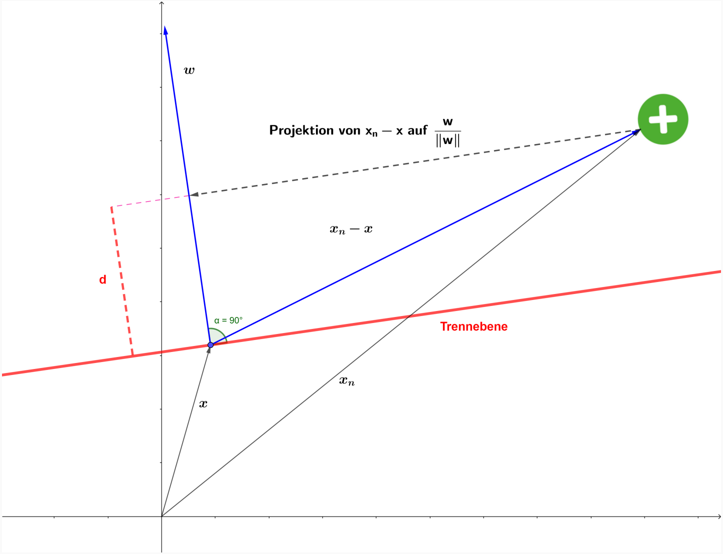
$$\begin{aligned} y_n(w^T x_n + b) &\geq 1 \qquad \text{für } y_n = +1 \\ y_n(w^T x_n + b) &\geq 1 \qquad \text{für } y_n = -1 \end{aligned}$$

Für den Fall, dass $x_n = \hat{x}$ genau an der Grenze des Trennbands liegt, gilt somit:

$$y_n(w^T \hat{x} + b) = 1$$

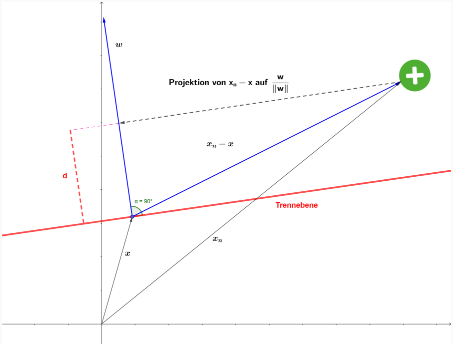
Normalabstand eines Punktes zur Ebene

Gesucht: Normalabstand d eines Punktes $x_n \in \mathbb{R}^K$ zur Ebene



Normalabstand eines Punktes zur Ebene

$$
\begin{aligned}
d &= |\frac{w^T}{\|w\|}(x_n - x)| = \\
&= \frac{1}{\|w\|}|(w^T x_n - w^T x)| = \\
&= \frac{1}{\|w\|}|(w^T x_n + b - (w^T x + b))|
\end{aligned}
$$



Normalabstand eines Punktes zur Ebene

$$d = \frac{1}{\|w\|} |(w^T x_n + b - (w^T x + b))|$$

Weil der Punkt x auf der Ebene liegt gilt $w^T x + b = 0$ und somit für den Normalabstand eines beliebigen Punktes x_n:

$$d = \frac{1}{\|w\|} |(w^T x_n + b)|$$

Breite des Trennbands

$$d = \frac{1}{\|w\|}|(w^T x_n + b)|$$

Annahme: $x_n = \hat{x}$ ist der am nächsten zur Ebene liegende Punkt auf der Grenze des Trennbands
Weil $y_n(w^T\hat{x} + b) = 1 = |w^T\hat{x} + b|$ gilt ergibt sich der minimale Normalabstand D:

$$D = \frac{1}{\|w\|}$$

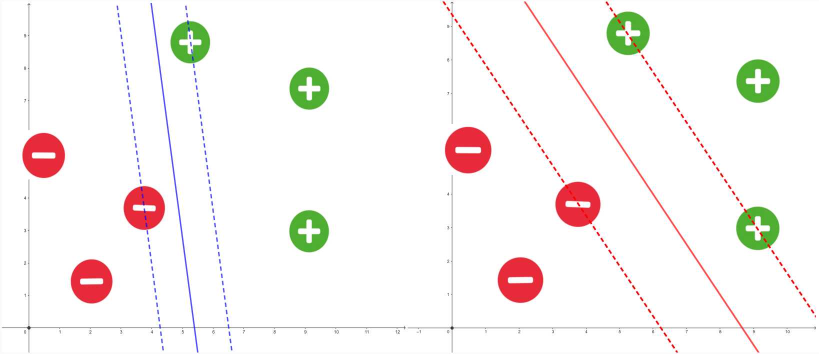
Weil D der minimale Normalabstand zur Ebene ist, ist $2D$ die Breite des freien Trennbands.

Reminder

Ziel: lineare Trennung mit möglichst breitem, freien Trennband

Entspricht Maximierung:

$$\max_w(2D) = \max_w \frac{2}{\|w\|} = \max_w \frac{1}{\|w\|}$$



Optimierungsproblem

$$\max_{w} \quad \frac{1}{\|w\|}$$

$$\text{mit} \quad \min_{n=1..N} |w^T x_n + b| = 1$$

$\min_{n=1..N} |w^T x_n + b| = 1$ ist der am nächsten zur Ebene liegende Punkt $\hat{x}$

Beidseitige Multiplikation mit y_n zur Vermeidung des Betrags:

$$|w^T x_n + b| = y_n(w^T x_n + b)$$

Optimierungsproblem

Nach Umformung (Maximierung in Minimierung) und Verallgemeinerung der Nebenbedingung auf beliebige Punkte x_n:

$$\begin{array}{ll} \min\limits_{w} & \frac{1}{2} w^T w \\ \text{mit} & y_n(w^T x_n + b) \geq 1 \text{ für } n = 1..N \end{array}$$

Bemerkungen:

- Faktor $\frac{1}{2}$ wird so gewählt weil dieser später wegfällt
- $w^T w$ und $\|w\|$ sind aus Optimierungssicht gleichbedeutend, Problem ist in dieser Form aber besser optimierbar

Lagrange Optimierung

Optimierungsproblem mit Ungleichung als Nebenbedingung

Umformen der Nebenbedingung:

$$\begin{array}{ll} \min\limits_{w} & \frac{1}{2} w^T w \\ \text{mit} & y_n(w^T x_n + b) - 1 \geq 0 \text{ für } n = 1..N \end{array}$$

Aufstellen der Lagrange Gleichung

Ungleichung wird von zu optimierender Funktion abgezogen und Lagrange Multiplikatoren eingeführt:

$$\min_{w,b} \qquad \mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{n=1}^{N} \alpha_n (y_n (w^T x_n + b) - 1)$$

$$\max_{\alpha_n} \qquad \alpha_n \geq 0 \text{ für } n = 1..N$$

Lösung durch 0 setzen der partiellen Ableitungen:

$$\nabla_w \mathcal{L} \stackrel{!}{=} \vec{0}$$

$$\frac{\partial}{\partial b} \mathcal{L} \stackrel{!}{=} 0$$

Lösen der Lagrange Gleichung

Nach w:

$$\nabla_w \mathcal{L} = w - \sum_{n=1}^{N} \alpha_n y_n x_n \stackrel{!}{=} \vec{0}$$

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

Nach b:

$$\frac{\partial}{\partial b} \mathcal{L} = - \sum_{n=1}^{N} \alpha_n y_n \stackrel{!}{=} 0$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0$$

Rücksubstitution in Lagrange Gleichung

Aufteilen der Summe:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}w^T w - \sum_{n=1}^{N} \alpha_n (y_n(w^T x_n + b) - 1) =$$
$$= \frac{1}{2}w^T w - [\sum_{n=1}^{N} \alpha_n y_n b - \sum_{n=1}^{N} \alpha_n + \sum_{n=1}^{N} \alpha_n y_n w^T x_n]$$

Aus Ableitung nach b wissen wir $\sum_{n=1}^{N} \alpha_n y_n = 0$:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}w^T w - [-\sum_{n=1}^{N} \alpha_n + \sum_{n=1}^{N} \alpha_n y_n w^T x_n]$$

Rücksubstitution in Lagrange Gleichung

Vergleicht man den Term $\sum_{n=1}^{N} \alpha_n y_n w^T x_n$ mit dem Ergebnis der partiellen Ableitung nach w ($w = \sum_{n=1}^{N} \alpha_n y_n x_n$) erkennt man, dass gilt:

$$\begin{aligned}\sum_{n=1}^{N} \alpha_n y_n w^T x_n = w^T w = \\ &= \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m\end{aligned}$$

Eingesetzt in Lagrange Gleichung:

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m$$

Maximierung ohne Nebenbedingung

Quadratic Programming Problem ($x_n^T x_m$):

$$
\begin{aligned}
\max_{\alpha} \quad & \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m \\
\text{mit} \quad & \alpha_n \geq 0 \text{ für } n = 1..N \\
& \sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N
\end{aligned}
$$

Lösung mittels QP-Solver

Ergebnis: α Vektor mit α_n Lagrange-Multiplikatoren

Schlupfterm

Reminder Ausgangsproblem:

$$\min_{w,b} \qquad \mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{n=1}^{N} \alpha_n (y_n(w^T x_n + b) - 1)$$

$$\max_{\alpha_n} \qquad \alpha_n \geq 0 \text{ für } n = 1..N$$

$\alpha_n(y_n(w^T x_n + b) - 1)$ („Schlupf“) wird 0 wenn:

- $\alpha_n = 0$ oder
- $(y_n(w^T x_n + b) - 1) = 0$

Umgekehrt: Alle x_n mit $\alpha_n \neq 0$ haben Schlupf 0, liegen also am nächsten zur Trennebene.
Diese Vektoren werden **Stützvektoren** genannt.

Bestimmung Gewichtsvektor

α Vektor mit α_n Faktoren ist bekannt aus QP-Solver

Viele α_i werden 0 sein, die $\alpha_i \neq 0$ gehören zu den Stützvektoren x_i.

Damit kann Formel für w

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

vereinfacht werden:

$$w = \sum_{n \text{ ist Stützvektor}} \alpha_n y_n x_n$$

Die Bezeichnung Stützvektor ergibt sich, weil die Ebene durch diese Vektoren „gestützt "wird. Alle Vektoren mit $\alpha_n = 0$ haben keinen Einfluss!

Bestimmung Bias

$y_n(w^T x_n + b) = 1$ gilt für Stützvektoren, daher kann mit beliebigem Stützvektor x_n der Bias bestimmt werden:

$$
\begin{aligned}
b &= \frac{1}{y_n} - w^T x_n = \\
&= y_n - w^T x_n
\end{aligned}
$$

Lösung mittels QP-Solver

Lösung mittels QP-Solver

Standardform von QP-Problemen:

$$\min_{x} = \frac{1}{2} x^T Q x + cx + d$$

Umformung Maximierung in Minimierung weil $\max -f(x) = \min f(x)$:

$$\min_{\alpha} \mathcal{L}(\alpha) = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n$$

Problem in QP-Standardform

$$\min_{\alpha} \mathcal{L}(\alpha) = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n$$

In QP-Standardform $\rightarrow$ Lösungs-Frameworks:

$$\min_{\alpha} \qquad \mathcal{L}(\alpha) = \frac{1}{2} \alpha^T Q \alpha + (-1^T)\alpha$$

$$\text{mit} \qquad Q = \begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \dots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \dots & y_2 y_N x_2^T x_N \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \dots & y_N y_N x_N^T x_N \end{bmatrix}$$

Problem in QP-Standardform

$$\min_{\alpha} \mathcal{L}(\alpha) = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n$$

In QP-Standardform $\rightarrow$ Lösungs-Frameworks:

$$\min_{\alpha} \quad \mathcal{L}(\alpha) = \frac{1}{2} \alpha^T Q \alpha + (-1^T)\alpha$$

$$\text{mit} \quad Q = \begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \dots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \dots & y_2 y_N x_2^T x_N \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \dots & y_N y_N x_N^T x_N \end{bmatrix}$$

Q ist $N \times N$ Matrix. Problematisch bei großen Trainingssets!

Lösung mittels QP-Solver

Ergebnis des QP-Solvers: $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$
Berechnung von w und b wie zuvor gezeigt:

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

Mit beliebigem Stützvektor x_k:

$$b = \frac{1}{y_k} - w^T x_k$$

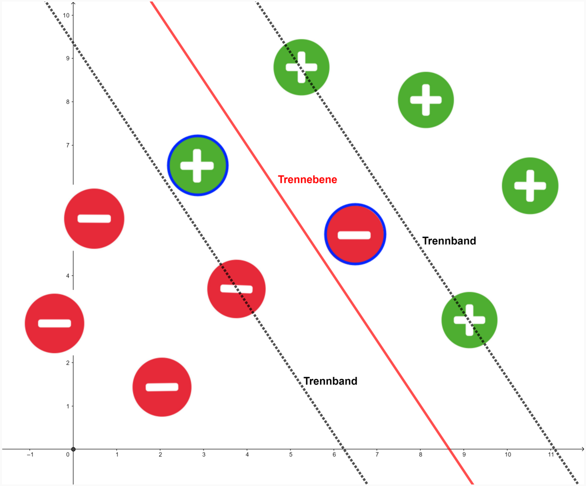
Klassifikation neuer Eingaben x:

$$y = sign(w^T x + b)$$

Soft-Margin Support Vector Machine

Einführung Soft-Margin SVM

Annahme bisher: Daten linear trennbar ohne Fehler



Einführung von Fehlervariablen

Problem: bisheriger Algorithmus terminiert nicht bei Fehlern
Lösung: Einführung von positiven Fehlervariablen $\xi_n \in \mathbb{R}^K, \xi_n \geq 0$:

$$
\begin{aligned}
w^T x_n + b &\geq +1 - \xi_n \qquad \text{für } y_n = +1 \\
w^T x_n + b &\leq -1 + \xi_n \qquad \text{für } y_n = -1
\end{aligned}
$$

Wann kann einzelne Fehlklassifikation auftreten? Wenn $\xi_n > 1$
Obere Grenze Anzahl Fehler:

$$E = C(\sum_{n=1}^{N} \xi_n)$$

$C \in \mathbb{R}, C \geq 0$: „Straffaktor“für Fehler

Erweiterung Optimierungsproblem um Fehlerterm

Ziel: Optimales w mit möglichst wenig Fehlern:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2} w^T w + C(\sum_{n=1}^{N} \xi_n) \\ \text{mit} \quad & y_n(w^T x_n + b) - 1 \geq 0 \text{ für } n = 1..N \end{aligned}$$

Ableiten, 0 setzen und lösen wie zuvor...

Soft-Margin SVM Optimierungsproblem

Soft-Margin Optimierungsproblem:

$$
\begin{aligned}
\max_{\alpha} \quad & \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m \\
\text{mit} \quad & 0 \leq \alpha_n \leq C \text{ für } n = 1..N \\
& \sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N
\end{aligned}
$$

Einziger Unterschied zu Hard-Margin: Beschränkung $\alpha_n \leq C$
(Hard-Margin: $\alpha_n \leq \infty$)

Soft-Margin SVM Optimierungsproblem

Soft-Margin Optimierungsproblem:

$$\max_{\alpha} \quad \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m$$

$$\text{mit} \quad 0 \leq \alpha_n \leq C \text{ für } n = 1..N$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N$$

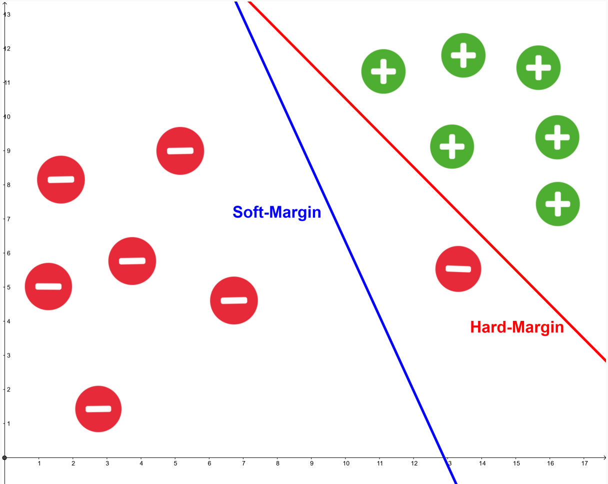
Einziger Unterschied zu Hard-Margin: Beschränkung $\alpha_n \leq C$
(Hard-Margin: $\alpha_n \leq \infty$)
Umgekehrt: Soft-Margin mit $C \to \infty$ entspricht Hard-Margin
Lösung: Wie zuvor gezeigt mit QP-Solver

Vergleich Hard- & Soft-Margin Support Vector Machine

Vergleich Hard- & Soft-Margin SVM

Hard-Margin: einzelne Ausreißer bestimmen Lage der Ebene

Soft-Margin: Fehlklassifikationen zugunsten besserer Gesamt-Trennung



Nichtlineare Trennung

Einleitung

Ziel: nichtlineare Trennung

Problem: SVM trennt ausschließlich linear

Ziel: nichtlineare Trennung

Problem: SVM trennt ausschließlich linear

Lösung: Transformation Eingabevektoren in linear trennbaren Raum

Transformationsfunktion $\Phi(x) : \mathbb{R}^K \rightarrow \mathbb{R}^L$

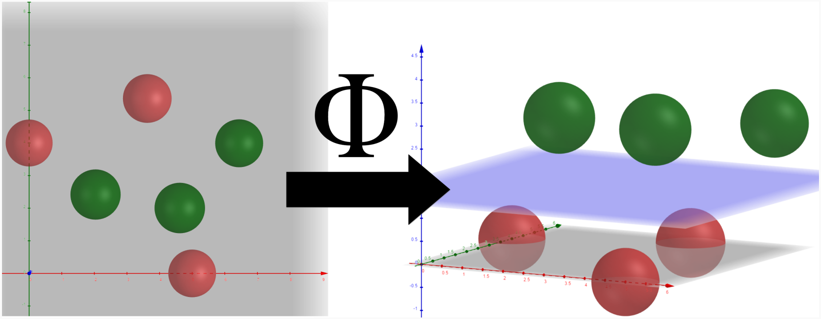


Abbildung 1: Transformation Eingabevektoren macht linear trennbar

Optimierungsproblem transformiert

Optimierungsproblem mit transformierten Eingabevektoren:

Optimierungsproblem transformiert

Optimierungsproblem mit transformierten Eingabevektoren:

$$
\begin{aligned}
\max_{\alpha} \quad & \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m \Phi(x_n)^T \Phi(x_m) \\
\text{mit} \quad & 0 \leq \alpha_n \leq C \text{ für } n = 1..N \\
& \sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N
\end{aligned}
$$

Optimierungsproblem transformiert

Optimierungsproblem mit transformierten Eingabevektoren:

$$\begin{aligned}
\max_{\alpha} \quad & \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m \Phi(x_n)^T \Phi(x_m) \\
\text{mit} \quad & 0 \leq \alpha_n \leq C \text{ für } n = 1..N \\
& \sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N
\end{aligned}$$

Anzahl Lagrangefaktoren α und Dimension der Q-Matrix hängen von Anzahl Eingabevektoren ab, nicht von der Dimension

Optimierungsproblem transformiert

Optimierungsproblem mit transformierten Eingabevektoren:

$$\max_{\alpha} \qquad \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m \Phi(x_n)^T \Phi(x_m)$$

$$\text{mit} \qquad 0 \leq \alpha_n \leq C \text{ für } n = 1..N$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N$$

Anzahl Lagrangefaktoren α und Dimension der Q-Matrix hängen von Anzahl Eingabevektoren ab, nicht von der Dimension
=> Zusatzkosten: Berechnung höherdimensionaler Skalarprodukte $\Phi(x)^T \Phi(x)$

Probleme und Erkenntnisse

Problem 1: Wahl der Transformationsfunktion $\Phi(x)$

Probleme und Erkenntnisse

Problem 1: Wahl der Transformationsfunktion $\Phi(x)$
Problem 2: Eingabevektoren in sehr hochdimensionalen/unendlichen Raum transformiert -> Berechnung Skalarprodukt sehr aufwändig/unmöglich

Probleme und Erkenntnisse

Problem 1: Wahl der Transformationsfunktion $\Phi(x)$
Problem 2: Eingabevektoren in sehr hochdimensionalen/unendlichen Raum transformiert -> Berechnung Skalarprodukt sehr aufwändig/unmöglich
Erkenntnis 1: Transformation erlaubt Bestimmung nichtlinearer Trenngrenzen

Probleme und Erkenntnisse

Problem 1: Wahl der Transformationsfunktion $\Phi(x)$
Problem 2: Eingabevektoren in sehr hochdimensionalen/unendlichen Raum transformiert -> Berechnung Skalarprodukt sehr aufwändig/unmöglich
Erkenntnis 1: Transformation erlaubt Bestimmung nichtlinearer Trenngrenzen
Erkenntnis 2: Dimension Vektoren beeinflusst Optimierungsproblem nicht stark

Probleme und Erkenntnisse

Problem 1: Wahl der Transformationsfunktion $\Phi(x)$
Problem 2: Eingabevektoren in sehr hochdimensionalen/unendlichen Raum transformiert -> Berechnung Skalarprodukt sehr aufwändig/unmöglich
Erkenntnis 1: Transformation erlaubt Bestimmung nichtlinearer Trenngrenzen
Erkenntnis 2: Dimension Vektoren beeinflusst Optimierungsproblem nicht stark
Verbesserung: Umgehung Zusatzkosten der transformierten Skalarprodukte => Kernel Trick

Kernel Trick

Es gilt: $z = \Phi(x)$

$$\max_{\alpha} \qquad \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m z_n^T z_m$$

$$\text{mit} \qquad 0 \leq \alpha_n \leq C \text{ für } n = 1..N$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N$$

Kernel Trick

Es gilt: $z = \Phi(x)$

$$\begin{aligned}
\max_{\alpha} \quad & \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m z_n^T z_m \\
\text{mit} \quad & 0 \leq \alpha_n \leq C \text{ für } n = 1..N \\
& \sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N
\end{aligned}$$

Berechnung von w und b:

$$w = \sum_{n=1}^{N} \alpha_n y_n z_n$$

$$b = \frac{1}{y_k} - w^T z_k$$

Kernel Trick

Einführung einer Kernel-Funktion $K(x, x') = z_1^T z_2 = \Phi(x)^T \Phi(x')$

Kernel Trick

Einführung einer Kernel-Funktion $K(x, x') = z_1^T z_2 = \Phi(x)^T \Phi(x')$

Berechnet Skalarprodukt der transformierten Eingabevektoren

Transformiert Eingabevektoren aber nicht tatsächlich in den neuen Raum

Kernel Trick

Einführung einer Kernel-Funktion $K(x, x') = z_1^T z_2 = \Phi(x)^T \Phi(x')$

Berechnet Skalarprodukt der transformierten Eingabevektoren

Transformiert Eingabevektoren aber nicht tatsächlich in den neuen Raum

Berechnung hochdimensionaler Skalarprodukte wird umgangen

Beispiel Kernel-Funktion

Kernel-Funktion für $x, x' \in \mathbb{R}^2$:

$$K(x, x') = (1 + x^T x')^2 =$$

Beispiel Kernel-Funktion

Kernel-Funktion für $x, x' \in \mathbb{R}^2$:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^2 = \\ &= (1 + x_1 x'_1 + x_2 x'_2)^2 = \end{aligned}$$

Beispiel Kernel-Funktion

Kernel-Funktion für $x, x' \in \mathbb{R}^2$:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^2 = \\ &= (1 + x_1 x_1' + x_2 x_2')^2 = \\ &= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2' \end{aligned}$$

Beweis Kernel-Funktion entspricht Skalarprodukt

Annahme für verwendete Transformationsfunktion Φ:

$$\Phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2)$$

Anwendung auf Vektoren x und x':

Beweis Kernel-Funktion entspricht Skalarprodukt

Annahme für verwendete Transformationsfunktion Φ:

$$\Phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2)$$

Anwendung auf Vektoren x und x':

$$\Phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2)$$

Annahme für verwendete Transformationsfunktion Φ:

$$\Phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

Anwendung auf Vektoren x und x':

$$\begin{aligned}\Phi(x) &= (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)\\ \Phi(x') &= (1, x_1'^2, x_2'^2, \sqrt{2}x_1', \sqrt{2}x_2', \sqrt{2}x_1'x_2')\end{aligned}$$

Beweis Kernel-Funktion entspricht Skalarprodukt

Annahme für verwendete Transformationsfunktion Φ:

$$\Phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

Anwendung auf Vektoren x und x':

$$\begin{aligned}
\Phi(x) &= (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2) \\
\Phi(x') &= (1, x_1'^2, x_2'^2, \sqrt{2}x_1', \sqrt{2}x_2', \sqrt{2}x_1'x_2') \\
\Phi(x)^T\Phi(x') &= 1 + x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_1' + 2x_2x_2' + 2x_1x_1'x_2x_2'
\end{aligned}$$

Beweis Kernel-Funktion entspricht Skalarprodukt

Annahme für verwendete Transformationsfunktion Φ:

$$\Phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2)$$

Anwendung auf Vektoren x und x':

$$\begin{aligned}
\Phi(x) &= (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2) \\
\Phi(x') &= (1, x_1'^2, x_2'^2, \sqrt{2}x_1', \sqrt{2}x_2', \sqrt{2}x_1' x_2') \\
\Phi(x)^T \Phi(x') &= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2'
\end{aligned}$$

Kernel-Funktion $K(x, x') = (1 + x^T x')^2$ entspricht Skalarprodukt der mit Φ transformierten Vektoren x, x'

Polynomieller Kernel

Verallgemeinerung des Beispiels

Seien Eingabevektoren $x \in \mathbb{R}^d$ und Transformationsfunktion $\Phi : \mathbb{R}^d \to \mathbb{Z}$ ein Polynom der Ordnung Q

Kernel-Funktion:

$$K(x, x') = (1 + x^T x')^Q =$$

Polynomieller Kernel

Verallgemeinerung des Beispiels

Seien Eingabevektoren $x \in \mathbb{R}^d$ und Transformationsfunktion $\Phi : \mathbb{R}^d \to \mathbb{Z}$ ein Polynom der Ordnung Q

Kernel-Funktion:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^Q = \\ &= (1 + x_1 x_1' + x_2 x_2' + \cdots + x_d x_d')^Q \end{aligned}$$

Polynomieller Kernel

Verallgemeinerung des Beispiels

Seien Eingabevektoren $x \in \mathbb{R}^d$ und Transformationsfunktion $\Phi : \mathbb{R}^d \to \mathbb{Z}$ ein Polynom der Ordnung Q

Kernel-Funktion:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^Q = \\ &= (1 + x_1 x'_1 + x_2 x'_2 + \cdots + x_d x'_d)^Q \end{aligned}$$

Skalierungsfaktoren a und b für Kompensation Faktoren:

$$K(x, x') = (a x^T x' + b)^Q$$

Polynomieller Kernel

Verallgemeinerung des Beispiels

Seien Eingabevektoren $x \in \mathbb{R}^d$ und Transformationsfunktion $\Phi : \mathbb{R}^d \to \mathbb{Z}$ ein Polynom der Ordnung Q

Kernel-Funktion:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^Q = \\ &= (1 + x_1 x'_1 + x_2 x'_2 + \dots + x_d x'_d)^Q \end{aligned}$$

Skalierungsfaktoren a und b für Kompensation Faktoren:

$$K(x, x') = (a x^T x' + b)^Q$$

Berechnung des Skalarprodukts eines Polynoms vom Grad Q ohne Transformation

Polynomieller Kernel

Verallgemeinerung des Beispiels

Seien Eingabevektoren $x \in \mathbb{R}^d$ und Transformationsfunktion $\Phi : \mathbb{R}^d \to \mathbb{Z}$ ein Polynom der Ordnung Q

Kernel-Funktion:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^Q = \\ &= (1 + x_1 x_1' + x_2 x_2' + \cdots + x_d x_d')^Q \end{aligned}$$

Skalierungsfaktoren a und b für Kompensation Faktoren:

$$K(x, x') = (a x^T x' + b)^Q$$

Berechnung des Skalarprodukts eines Polynoms vom Grad Q ohne Transformation

Polynomieller Kernel

Radial Basis Function Kernel

Weitere Kernel-Funktion: Radial Basis Function (RBF) Kernel:

Radial Basis Function Kernel

Weitere Kernel-Funktion: Radial Basis Function (RBF) Kernel:

$$K(x, x') = \exp \gamma \|x - x'\|^2$$

Radial Basis Function Kernel

Weitere Kernel-Funktion: Radial Basis Function (RBF) Kernel:

$$K(x, x') = \exp \gamma \|x - x'\|^2$$

γ ist wählbarer Parameter

Radial Basis Function Kernel

Weitere Kernel-Funktion: Radial Basis Function (RBF) Kernel:

$$K(x, x') = \exp \gamma \|x - x'\|^2$$

γ ist wählbarer Parameter

Dem Kernel zugehörige Transformationsfunktion Φ bildet in unendlich dimensionalen Raum ab

Beweis unendliche Dimensionalität

Beweis für einfachsten Fall:

Beweis unendliche Dimensionalität

Beweis für einfachsten Fall:

$$K(x, x') = \exp\left(-(x - x')^2\right) =$$

Beweis unendliche Dimensionalität

Beweis für einfachsten Fall:

$$\begin{aligned} K(x, x') &= \exp\left(-(x - x')^2\right) = \\ &= \exp\left(-x^2 + 2xx' - x'^2\right) = \end{aligned}$$

Beweis unendliche Dimensionalität

Beweis für einfachsten Fall:

$$\begin{aligned} K(x, x') &= \exp\left(-(x - x')^2\right) = \\ &= \exp\left(-x^2 + 2xx' - x'^2\right) = \\ &= \exp\left(-x^2\right) \exp\left(2xx'\right) \exp\left(-x'^2\right) = \end{aligned}$$

Beweis unendliche Dimensionalität

Beweis für einfachsten Fall:

$$
\begin{aligned}
K(x, x') &= \exp\left(-(x - x')^2\right) = \\
&= \exp\left(-x^2 + 2xx' - x'^2\right) = \\
&= \exp\left(-x^2\right) \exp\left(2xx'\right) \exp\left(-x'^2\right) = \\
&= \exp\left(-x^2\right) \sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!} \exp\left(-x'^2\right)
\end{aligned}
$$

Beweis unendliche Dimensionalität

Beweis für einfachsten Fall:

$$
\begin{aligned}
K(x, x') &= \exp\left(-(x - x')^2\right) = \\
&= \exp\left(-x^2 + 2xx' - x'^2\right) = \\
&= \exp\left(-x^2\right) \exp\left(2xx'\right) \exp\left(-x'^2\right) = \\
&= \exp\left(-x^2\right) \sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!} \exp\left(-x'^2\right)
\end{aligned}
$$

Taylorexpansion von $\exp(2xx')$ macht Unendlichkeit Raum sichtbar

Beweis unendliche Dimensionalität

Beweis für einfachsten Fall:

$$
\begin{aligned}
K(x, x') &= \exp\left(-(x - x')^2\right) = \\
&= \exp\left(-x^2 + 2xx' - x'^2\right) = \\
&= \exp\left(-x^2\right) \exp\left(2xx'\right) \exp\left(-x'^2\right) = \\
&= \exp\left(-x^2\right) \sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!} \exp\left(-x'^2\right)
\end{aligned}
$$

Taylorexpansion von $\exp(2xx')$ macht Unendlichkeit Raum sichtbar
Hat für Skalarprodukt benötigte Symmetrie $\exp(-x^2)$ - $\exp(-x'^2)$ und $(x)^k$ - $(x')^k$

Beweis unendliche Dimensionalität

Beweis für einfachsten Fall:

$$
\begin{aligned}
K(x, x') &= \exp\left(-(x - x')^2\right) = \\
&= \exp\left(-x^2 + 2xx' - x'^2\right) = \\
&= \exp\left(-x^2\right) \exp\left(2xx'\right) \exp\left(-x'^2\right) = \\
&= \exp\left(-x^2\right) \sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!} \exp\left(-x'^2\right)
\end{aligned}
$$

Taylorexpansion von $\exp(2xx')$ macht Unendlichkeit Raum sichtbar
Hat für Skalarprodukt benötigte Symmetrie $\exp(-x^2)$ - $\exp(-x'^2)$ und $(x)^k$ - $(x')^k$
Anteile $\frac{2^k}{k!}$ gleichmäßig auf x und x' aufteilbar (Wurzel der Anteile zu x und x' multiplizieren)

SVM mit Kernel

Eingabe x, Transformation mit $z = \Phi(x)$, Klassifikation mit:

SVM mit Kernel

Eingabe x, Transformation mit $z = \Phi(x)$, Klassifikation mit:

$$y(x) = sign(w^T z + b) \tag{16}$$

SVM mit Kernel

Eingabe x, Transformation mit $z = \Phi(x)$, Klassifikation mit:

$$y(x) = sign(w^T z + b) \tag{16}$$

Funktion Φ muss bekannt sein

SVM mit Kernel

Eingabe x, Transformation mit $z = \Phi(x)$, Klassifikation mit:

$$y(x) = sign(w^T z + b) \tag{16}$$

Funktion Φ muss bekannt sein
Transformation nötig

SVM mit Kernel

Eingabe x, Transformation mit $z = \Phi(x)$, Klassifikation mit:

$$y(x) = sign(w^T z + b) \tag{16}$$

Funktion Φ muss bekannt sein

Transformation nötig

Ziel: Problem mittels Kernel-Funktion $K(x, x')$ ausdrücken, transformierte Vektoren vermeiden

SVM mit Kernel

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \tag{17}$$

SVM mit Kernel

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \tag{17}$$

Einsetzen Gleichung (17) in Gleichung (16):

SVM mit Kernel

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \tag{17}$$

Einsetzen Gleichung (17) in Gleichung (16):

$$y(x) = sign(\sum_{\alpha_n > 0} \alpha_n y_n z_n^T z + b) =$$

SVM mit Kernel

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \tag{17}$$

Einsetzen Gleichung (17) in Gleichung (16):

$$\begin{aligned} y(x) &= sign(\sum_{\alpha_n > 0} \alpha_n y_n z_n^T z + b) = \\ &= sign(\sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x) + b) \end{aligned}$$

SVM mit Kernel

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \tag{17}$$

Einsetzen Gleichung (17) in Gleichung (16):

$$\begin{aligned} y(x) &= sign(\sum_{\alpha_n > 0} \alpha_n y_n z_n^T z + b) = \\ &= sign(\sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x) + b) \end{aligned}$$

Einsetzen von Gleichung (17) für beliebigen Stützvektor für b:

SVM mit Kernel

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \tag{17}$$

Einsetzen Gleichung (17) in Gleichung (16):

$$\begin{aligned} y(x) &= sign(\sum_{\alpha_n > 0} \alpha_n y_n z_n^T z + b) = \\ &= sign(\sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x) + b) \end{aligned}$$

Einsetzen von Gleichung (17) für beliebigen Stützvektor für b:

$$b = \frac{1}{y_k} - w^T z_k =$$

SVM mit Kernel

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \tag{17}$$

Einsetzen Gleichung (17) in Gleichung (16):

$$\begin{aligned} y(x) &= sign(\sum_{\alpha_n > 0} \alpha_n y_n z_n^T z + b) = \\ &= sign(\sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x) + b) \end{aligned}$$

Einsetzen von Gleichung (17) für beliebigen Stützvektor für b:

$$\begin{aligned} b &= \frac{1}{y_k} - w^T z_k = \\ &= \frac{1}{y_k} - \sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x_k) = \end{aligned}$$

SVM mit Kernel

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \tag{17}$$

Einsetzen Gleichung (17) in Gleichung (16):

$$\begin{aligned} y(x) &= sign(\sum_{\alpha_n > 0} \alpha_n y_n z_n^T z + b) = \\ &= sign(\sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x) + b) \end{aligned}$$

Einsetzen von Gleichung (17) für beliebigen Stützvektor für b:

$$\begin{aligned} b &= \frac{1}{y_k} - w^T z_k = \\ &= \frac{1}{y_k} - \sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x_k) = \\ &= y_k - \sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x_k) \end{aligned}$$

SVM mit Kernel

- SVM vollständig definiert

SVM mit Kernel

- SVM vollständig definiert
- Transformationsfunktion Φ muss nicht bekannt sein

SVM mit Kernel

- SVM vollständig definiert
- Transformationsfunktion Φ muss nicht bekannt sein
- Keine einzige tatsächliche Transformation wird durchgeführt

SVM mit Kernel

- SVM vollständig definiert
- Transformationsfunktion Φ muss nicht bekannt sein
- Keine einzige tatsächliche Transformation wird durchgeführt
- Beliebige dimensionale Räume durch entsprechende Kernel-Funktionen verwendbar

SVM mit Kernel

- SVM vollständig definiert
- Transformationsfunktion Φ muss nicht bekannt sein
- Keine einzige tatsächliche Transformation wird durchgeführt
- Beliebige dimensionale Räume durch entsprechende Kernel-Funktionen verwendbar
- Beliebige Kernel-Funktion verwendbar, solange bestimmte Bedingungen erfüllt werden

Bedingungen für Kernel-Funktion

Kernel-Funktion muss Skalarprodukt in Raum entsprechen

Zwei verschiedene Ansätze, um das zu zeigen:

Bedingungen für Kernel-Funktion

Kernel-Funktion muss Skalarprodukt in Raum entsprechen

Zwei verschiedene Ansätze, um das zu zeigen:

- Für vermutlich richtige Kernel-Funktion wird konstruktiv versucht, die zugehörige Transformationsfunktion Φ zu bestimmen

Bedingungen für Kernel-Funktion

Kernel-Funktion muss Skalarprodukt in Raum entsprechen

Zwei verschiedene Ansätze, um das zu zeigen:

- Für vermutlich richtige Kernel-Funktion wird konstruktiv versucht, die zugehörige Transformationsfunktion Φ zu bestimmen
- Kernel ist gültig, wenn $K(x, x')$ symmetrisch und die Matrix

$$K = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \dots & K(x_1, x_N) \\ K(x_2, x_1) & K(x_2, x_2) & \dots & K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ K(x_N, x_1) & K(x_N, x_2) & \dots & K(x_N, x_N) \end{bmatrix}$$

  positiv semi-definit ist für jedes beliebige $x_1..x_N$.

Bedingungen für Kernel-Funktion

Kernel-Funktion muss Skalarprodukt in Raum entsprechen

Zwei verschiedene Ansätze, um das zu zeigen:

- Für vermutlich richtige Kernel-Funktion wird konstruktiv versucht, die zugehörige Transformationsfunktion Φ zu bestimmen
- Kernel ist gültig, wenn $K(x, x')$ symmetrisch und die Matrix

$$K = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \dots & K(x_1, x_N) \\ K(x_2, x_1) & K(x_2, x_2) & \dots & K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ K(x_N, x_1) & K(x_N, x_2) & \dots & K(x_N, x_N) \end{bmatrix}$$

  positiv semi-definit ist für jedes beliebige $x_1..x_N$. Auch Satz von Mercer genannt, garantiert, dass Funktion Φ existiert, die in Raum abbildet, dessen Skalarprodukte durch Kernel-Funktion beschrieben werden können

Anpassung QP-Solver

Lösung mittels Quadratic Programming Solver weiter möglich

Änderung: In $Q - Matrix\ K(x_n, x_m)$ statt $x_n^T x_m$

Anpassung QP-Solver

Lösung mittels Quadratic Programming Solver weiter möglich

Änderung: In $Q - Matrix\ K(x_n, x_m)$ statt $x_n^T x_m$

$$
\begin{aligned}
\min_{\alpha} \quad & \mathcal{L}(\alpha) = \frac{1}{2}\alpha^T Q\alpha + (-1^T)\alpha \\
\text{mit} \quad & Q = \begin{bmatrix} y_1 y_1 K(x_1, x_1) & y_1 y_2 K(x_1, x_2) & \dots & y_1 y_N K(x_1, x_N) \\ y_2 y_1 K(x_2, x_1) & y_2 y_2 K(x_2, x_2) & \dots & y_2 y_N K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 K(x_N, x_1) & y_N y_2 K(x_N, x_2) & \dots & y_N y_N K(x_N, x_N) \end{bmatrix} \\
\text{für} \quad & y^T \alpha = 0 \\
& 0 \leq \alpha \leq \infty
\end{aligned}
$$

Anpassung QP-Solver

Lösung mittels Quadratic Programming Solver weiter möglich

Änderung: In $Q - Matrix\ K(x_n, x_m)$ statt $x_n^T x_m$

$$\min_{\alpha} \qquad \mathcal{L}(\alpha) = \frac{1}{2}\alpha^T Q \alpha + (-1^T)\alpha$$

$$\text{mit} \qquad Q = \begin{bmatrix} y_1 y_1 K(x_1, x_1) & y_1 y_2 K(x_1, x_2) & \dots & y_1 y_N K(x_1, x_N) \\ y_2 y_1 K(x_2, x_1) & y_2 y_2 K(x_2, x_2) & \dots & y_2 y_N K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 K(x_N, x_1) & y_N y_2 K(x_N, x_2) & \dots & y_N y_N K(x_N, x_N) \end{bmatrix}$$

$$\text{für} \qquad y^T \alpha = 0$$

$$0 \leq \alpha \leq \infty$$

An QP-Solver übergeben und α erhalten

Anpassung QP-Solver

Lösung mittels Quadratic Programming Solver weiter möglich

Änderung: In $Q - Matrix\ K(x_n, x_m)$ statt $x_n^T x_m$

$$
\begin{aligned}
\min_{\alpha} \quad & \mathcal{L}(\alpha) = \frac{1}{2}\alpha^T Q \alpha + (-1^T)\alpha \\
\text{mit} \quad & Q = \begin{bmatrix} y_1 y_1 K(x_1, x_1) & y_1 y_2 K(x_1, x_2) & \dots & y_1 y_N K(x_1, x_N) \\ y_2 y_1 K(x_2, x_1) & y_2 y_2 K(x_2, x_2) & \dots & y_2 y_N K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 K(x_N, x_1) & y_N y_2 K(x_N, x_2) & \dots & y_N y_N K(x_N, x_N) \end{bmatrix} \\
\text{für} \quad & y^T \alpha = 0 \\
& 0 \leq \alpha \leq \infty
\end{aligned}
$$

An QP-Solver übergeben und α erhalten

SVM in Kombination mit Kernel-Funktionen auf beliebige binäre Klassifikationsprobleme anwendbar

Pseudocode und Beispiele

Pseudocode Hard-Margin SVM

Hard-Margin SVM	**Zeile**
Initialisiere x, y	1

Pseudocode Hard-Margin SVM

Hard-Margin SVM	**Zeile**
Initialisiere x, y	1
$Q = (yy^T)K$	2

Pseudocode Hard-Margin SVM

Hard-Margin SVM	**Zeile**
Initialisiere x, y	1
$Q = (yy^T)K$	2
$c = (-1, -1, \dots, -1)^T$	3

Pseudocode Hard-Margin SVM

Hard-Margin SVM	**Zeile**
Initialisiere x, y	1
$Q = (yy^T)K$	2
$c = (-1, -1, \ldots, -1)^T$	3
$A = \mathit{diag}\,(-1, -1, \ldots, -1)$	4
$b = (0, 0, \ldots, 0)^T$	5

Hard-Margin SVM	**Zeile**
Initialisiere x, y	1
$\mathrm{Q} = (\mathrm{yy}^{\mathsf{T}})\mathrm{K}$	2
$\mathrm{c} = (-1, -1, \ldots, -1)^{\mathsf{T}}$	3
$\mathrm{A} = \mathit{diag}\,(-1, -1, \ldots, -1)$	4
$\mathrm{b} = (0, 0, \ldots, 0)^{\mathsf{T}}$	5
$\mathrm{A}_{\mathrm{eq}} = y^{\mathsf{T}}$	6
$b_{eq} = 0$	7

Pseudocode Hard-Margin SVM

Hard-Margin SVM	**Zeile**
Initialisiere x, y	1
$Q = (yy^T)K$	2
$c = (-1, -1, \ldots, -1)^T$	3
$A = diag\,(-1, -1, \ldots, -1)$	4
$b = (0, 0, \ldots, 0)^T$	5
$A_{eq} = y^T$	6
$b_{eq} = 0$	7
$\alpha = QPSolver\,(Q, c, A, b, A_{eq}, b_{eq})$	8

Pseudocode Hard-Margin SVM

Hard-Margin SVM	**Zeile**
Initialisiere x, y	1
$\mathrm{Q} = (\mathrm{yy}^{\mathsf{T}})\mathrm{K}$	2
$\mathrm{c} = (-1, -1, \ldots, -1)^{\mathsf{T}}$	3
$\mathrm{A} = \mathit{diag}\,(-1, -1, \ldots, -1)$	4
$\mathrm{b} = (0, 0, \ldots, 0)^{\mathsf{T}}$	5
$\mathrm{A}_{\mathrm{eq}} = y^{\mathsf{T}}$	6
$b_{eq} = 0$	7
$\alpha = \mathit{QPSolver}\,(\mathrm{Q}, \mathrm{c}, \mathrm{A}, b, \mathrm{A}_{\mathrm{eq}}, \mathrm{b}_{\mathrm{eq}})$	8
$\mathrm{w} = \sum_{n=SV} \alpha_n y_n x_n$	9

Pseudocode Hard-Margin SVM

Hard-Margin SVM
Initialisiere x, y
$\mathrm{Q} = (\mathrm{yy}^{\mathrm{T}})\mathrm{K}$
$\mathrm{c} = (-1, -1, \ldots, -1)^{\mathrm{T}}$
$\mathrm{A} = diag\,(-1, -1, \ldots, -1)$
$\mathrm{b} = (0, 0, \ldots, 0)^{\mathrm{T}}$
$\mathrm{A}_{\mathrm{eq}} = y^{\mathrm{T}}$
$b_{eq} = 0$
$\alpha = QPSolver\,(\mathrm{Q}, \mathrm{c}, \mathrm{A}, b, \mathrm{A}_{\mathrm{eq}}, \mathrm{b}_{\mathrm{eq}})$
$\mathrm{w} = \sum_{n=SV} \alpha_n y_n x_n$
$bias = \frac{1}{y_n} - \mathrm{w}^{\mathrm{T}} x_n$

QP Parameter Hard Margin

$$\min_x = \frac{1}{2}x^T Qx + cx + d$$
$$Ax \leq b$$
$$A_{eq}x = b_{eq}$$

Anmerkungen zu Pseudocode Hard-Margin SVM

Anmerkungen

- Zeile 1: Initialisieren von Werte- und Klassenvektoren
- Zeile 2: Berechnen der Matrix Q
- Zeile 3: Berechnen von c
- Zeile 4, 5: Berechnen der Ungleichheitsbedingungen
- Zeile 6, 7: Berechnen der Gleichheitsbedingungen
- Zeile 8: Lösen mittels Quadratic Programming
- Zeile 9: Berechnung Gewichte mit Stützvektoren
- Zeile 10: Berechnung bias mit beliebigem Stützvektor

Pseudocode Berechnung K-Matrix

K **Berechnung**	**Zeile**
Initialisiere x	1

Pseudocode Berechnung K-Matrix

K **Berechnung**	**Zeile**
Initialisiere x	1
For $i = 1$ To N	2
For $j = 1$ To N	3

Pseudocode Berechnung K-Matrix

K **Berechnung**	**Zeile**
Initialisiere x	1
For $i = 1$ To N	2
For $j = 1$ To N	3
$\mathrm{K}(i,j) = x_i \cdot x_j$	4
Ende For	5
Ende For	6

Beispiel Hard-Margin SVM

Trennung linear separierbarer Punkte mit Hard-Margin SVM

Beispiel Hard-Margin SVM

Trennung linear separierbarer Punkte mit Hard-Margin SVM

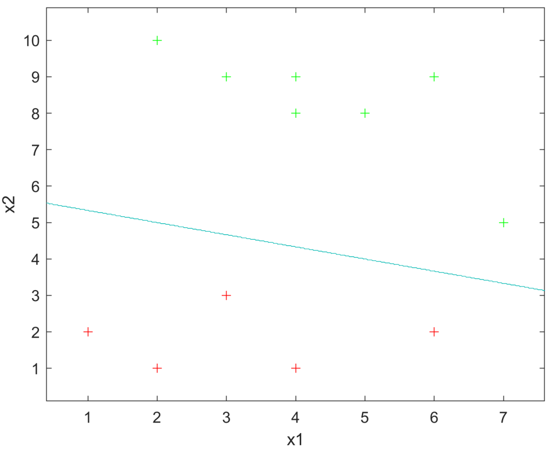


Abbildung 2: Beispiel Trenngrenze lineares Problem mit Hard-Margin SVM

Vergleich Hard-Margin SVM

Vergleich mit α-LMS, Algorithmus mit Pseudoinverse

Vergleich Hard-Margin SVM

Vergleich mit α-LMS, Algorithmus mit Pseudoinverse

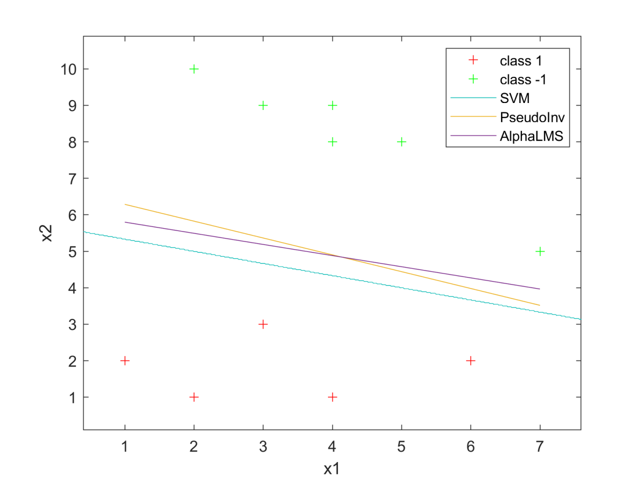


Abbildung 3: Vergleich Hard-Margin SVM, α-LMS, Algo mit Pseudoinverse

Pseudocode Soft-Margin SVM

Soft-Margin SVM	Zeile
Initialisiere x, y, C	1

Pseudocode Soft-Margin SVM

Soft-Margin SVM	**Zeile**
Initialisiere x, y, C	1
$\mathrm{Q} = (\mathrm{y}\mathrm{y}^{\mathrm{T}})\mathrm{K}$	2
$\mathrm{c} = (-1, -1, \ldots, -1)^{\mathrm{T}}$	3
$\mathrm{A} = diag\,(-1, -1, \ldots, -1)$	4
$\mathrm{b} = (0, 0, \ldots, 0)^{\mathrm{T}}$	5
$\mathrm{A}_{\mathrm{eq}} = \mathrm{y}^{\mathrm{T}}$	6
$b_{eq} = 0$	7

Pseudocode Soft-Margin SVM

Soft-Margin SVM
Initialisiere x, y, C
$\mathrm{Q} = (\mathrm{y}\mathrm{y}^{\mathrm{T}})\mathrm{K}$
$\mathrm{c} = (-1, -1, \dots, -1)^{\mathrm{T}}$
$\mathrm{A} = \mathit{diag}\,(-1, -1, \dots, -1)$
$\mathrm{b} = (0, 0, \dots, 0)^{\mathrm{T}}$
$\mathrm{A}_{\mathrm{eq}} = \mathrm{y}^{\mathrm{T}}$
$b_{eq} = 0$
$\mathrm{lb} = (0, 0, \dots, 0)^{\mathrm{T}}$
$\mathrm{ub} = C * (1, 1, \dots, 1)^{\mathrm{T}}$

Pseudocode Soft-Margin SVM

Soft-Margin SVM	Zeile
Initialisiere $\mathrm{x}, \mathrm{y}, C$	1
$\mathrm{Q} = (\mathrm{y}\mathrm{y}^{\mathrm{T}})\mathrm{K}$	2
$\mathrm{c} = (-1, -1, \ldots, -1)^{\mathrm{T}}$	3
$\mathrm{A} = \mathit{diag}\,(-1, -1, \ldots, -1)$	4
$\mathrm{b} = (0, 0, \ldots, 0)^{\mathrm{T}}$	5
$\mathrm{A}_{\mathrm{eq}} = \mathrm{y}^{\mathrm{T}}$	6
$b_{eq} = 0$	7
$\mathrm{lb} = (0, 0, \ldots, 0)^{\mathrm{T}}$	8
$\mathrm{ub} = C * (1, 1, \ldots, 1)^{\mathrm{T}}$	9
$\alpha = \mathit{QPSolver}\,(\mathrm{Q}, \mathrm{c}, \mathrm{A}, b, \mathrm{A}_{\mathrm{eq}}, \mathrm{b}_{\mathrm{eq}}, \mathrm{lb}, \mathrm{ub})$	10

Soft-Margin SVM	Zeile
Initialisiere x, y, C	1
$\mathrm{Q} = (\mathrm{y}\mathrm{y}^{\mathrm{T}})\mathrm{K}$	2
$\mathrm{c} = (-1, -1, \ldots, -1)^{\mathrm{T}}$	3
$\mathrm{A} = \mathit{diag}\,(-1, -1, \ldots, -1)$	4
$\mathrm{b} = (0, 0, \ldots, 0)^{\mathrm{T}}$	5
$\mathrm{A}_{\mathrm{eq}} = \mathrm{y}^{\mathrm{T}}$	6
$b_{eq} = 0$	7
$\mathrm{lb} = (0, 0, \ldots, 0)^{\mathrm{T}}$	8
$\mathrm{ub} = C * (1, 1, \ldots, 1)^{\mathrm{T}}$	9
$\alpha = \mathit{QPSolver}\,(\mathrm{Q}, \mathrm{c}, \mathrm{A}, b, \mathrm{A}_{\mathrm{eq}}, \mathrm{b}_{\mathrm{eq}}, \mathrm{lb}, \mathrm{ub})$	10
$\mathrm{w} = \sum_{n=SV} \alpha_n y_n x_n$	11
$\mathit{bias} = \frac{1}{y_n} - \mathrm{w}^{\mathrm{T}} x_n$	12

QP Parameter Soft Margin

$$\min_x = \frac{1}{2}x^T Qx + cx + d$$
$$Ax \leq b$$
$$A_{eq}x = b_{eq}$$
$$lb \leq x \leq ub$$

Anmerkungen

- Zeile 1: Initialisieren von Werte- und Klassenvektoren und Bestrafungsparameter C
- Zeile 2: Berechnen der Matrix Q
- Zeile 3: Berechnen von c
- Zeile 4, 5: Berechnen der Ungleichheitsbedingungen
- Zeile 6, 7: Berechnen der Gleichheitsbedingungen
- Zeile 8: untere Grenze (kann auch $(-\infty, -\infty, \ldots, -\infty)$ gewählt werden)
- Zeile 9: Berechnung obere Grenze mit C
- Zeile 10: Lösen mittels Quadratic Programming
- Zeile 11: Berechnung Gewichte mit Stützvektoren
- Zeile 12: Berechnung bias mit beliebigem Stützvektor

Beispiel Soft-Margin SVM

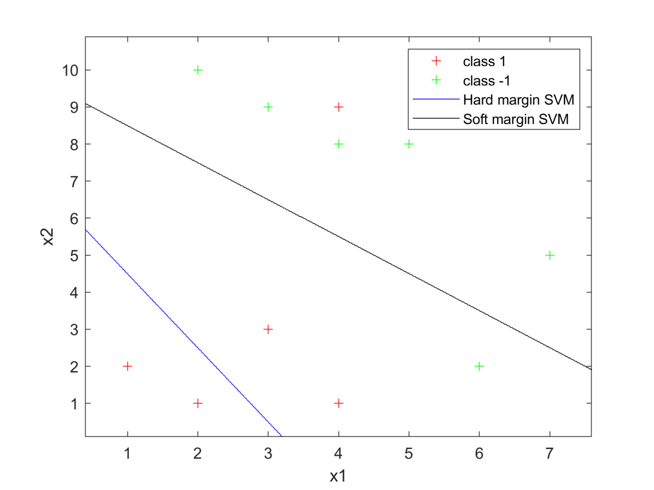


Abbildung 4: Trenngrenze nichtlineares Problem mit Hard-Margin und Soft-Margin SVM

Beispiel Soft-Margin SVM

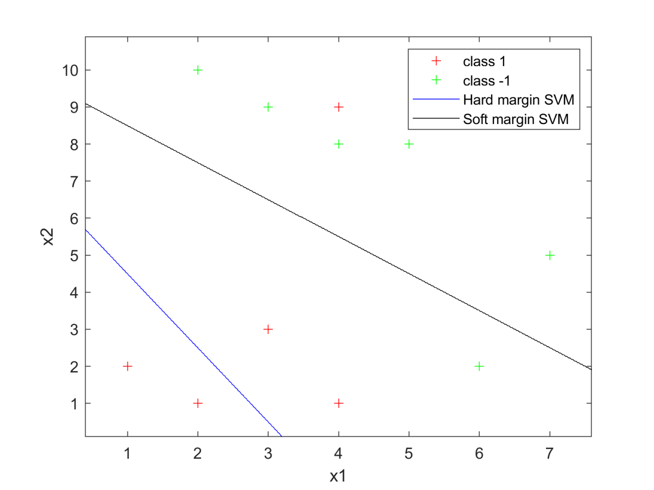


Abbildung 4: Trenngrenze nichtlineares Problem mit Hard-Margin und Soft-Margin SVM

Gute Ergebnisse auch bei Ausreißern

Soft-Margin SVM mit Kernel-Trick	**Zeile**
Initialisiere x, y, C	1

Pseudocode Kernel-Trick

Soft-Margin SVM mit Kernel-Trick	**Zeile**
Initialisiere x, y, C	1
$\mathrm{Q} = (\mathrm{y}\mathrm{y}^{\mathrm{T}})\mathrm{K}$	2

Pseudocode Kernel-Trick

Soft-Margin SVM mit Kernel-Trick	**Zeile**
Initialisiere $\mathrm{x}, \mathrm{y}, C$	1
$\mathrm{Q} = (\mathrm{y}\mathrm{y}^{\mathrm{T}})\mathrm{K}$	2
$\mathrm{c} = (-1, -1, \ldots, -1)^{\mathrm{T}}$	3
$\mathrm{A} = \mathit{diag}\,(-1, -1, \ldots, -1)$	4
$\mathrm{b} = (0, 0, \ldots, 0)^{\mathrm{T}}$	5
$\mathrm{A}_{\mathrm{eq}} = \mathrm{y}^{\mathrm{T}}$	6
$b_{eq} = 0$	7
$\mathrm{lb} = (0, 0, \ldots, 0)^{\mathrm{T}}$	8
$\mathrm{ub} = C * (1, 1, \ldots, 1)^{\mathrm{T}}$	9
$\alpha = \mathit{QPSolver}\,(\mathrm{Q}, \mathrm{c}, \mathrm{A}, b, \mathrm{A}_{\mathrm{eq}}, \mathrm{b}_{\mathrm{eq}}, \mathrm{lb}, \mathrm{ub})$	10

Pseudocode Kernel-Trick

Soft-Margin SVM mit Kernel-Trick	**Zeile**
Initialisiere $\mathrm{x}, \mathrm{y}, C$	1
$\color{red}{\mathrm{Q} = (\mathrm{y}\mathrm{y}^{\mathrm{T}})\mathrm{K}}$	2
$\mathrm{c} = (-1, -1, \ldots, -1)^{\mathrm{T}}$	3
$\mathrm{A} = diag\,(-1, -1, \ldots, -1)$	4
$\mathrm{b} = (0, 0, \ldots, 0)^{\mathrm{T}}$	5
$\mathrm{A}_{\mathrm{eq}} = \mathrm{y}^{\mathrm{T}}$	6
$b_{eq} = 0$	7
$\mathrm{lb} = (0, 0, \ldots, 0)^{\mathrm{T}}$	8
$\mathrm{ub} = C * (1, 1, \ldots, 1)^{\mathrm{T}}$	9
$\alpha = QPSolver\,(\mathrm{Q}, \mathrm{c}, \mathrm{A}, b, \mathrm{A}_{\mathrm{eq}}, \mathrm{b}_{\mathrm{eq}}, \mathrm{lb}, \mathrm{ub})$	10
$\color{red}{bias = y_k - \sum_{\alpha_n > 0} \alpha_n y_n KF\,(x_n, x_k)}$	11

Anmerkungen zu Pseudocode Kernel-Trick

Anmerkungen

- *KF* ist die Kernel-Funktion.
- Algorithmus ist für alle Kernel (RBF, polynomiell, ...) gleich.
- Klassifizierung muss auch angepasst werden.
- Zeile 1: Initialisieren von Werte- und Klassenvektoren und Bestrafungsparameter C
- Zeile 2: Berechnen der Matrix Q
- Zeile 3: Berechnen von c
- Zeile 4, 5: Berechnen der Ungleichheitsbedingungen
- Zeile 6, 7: Berechnen der Gleichheitsbedingungen
- Zeile 8: untere Grenze (kann auch $(-\infty, -\infty, \ldots, -\infty)$ gewählt werden)
- Zeile 9: Berechnung obere Grenze mit C
- Zeile 10: Lösen mittels Quadratic Programming
- Zeile 11: Berechnung bias, x_k ist ein beliebiger Stützvektor, y_k

Pseudocode für K-Matrix bei Kernel

K **Berechnung**	**Zeile**
Initialisiere x	1
For $i = 1$ To N	2
For $j = 1$ To N	3

Pseudocode für K-Matrix bei Kernel

K **Berechnung**	**Zeile**
Initialisiere x	1
For $i = 1$ To N	2
For $j = 1$ To N	3
$\mathrm{K}(i, j) = KF(x_i, x_j)$	4
Ende For	5
Ende For	6

Pseudocode für K-Matrix bei Kernel

K **Berechnung**	**Zeile**
Initialisiere x	1
For $i = 1$ To N	2
For $j = 1$ To N	3
$\mathrm{K}\,(i, j) = KF\,(x_i, x_j)$	4
Ende For	5
Ende For	6

N Anzahl Eingabevektoren, KF Kernel-Funktion

Anstatt Skalarprodukt Kernel-Funktion

Spart Transformation in höhere Dimensionen

Beispiel Polynomieller Kernel

Kernel: $(ax^T x' + b)^Q$, $b = 1, a = 1$, Exponent Q wird variiert

Beispiel Polynomieller Kernel

Kernel: $(ax^T x' + b)^Q$, $b = 1, a = 1$, Exponent Q wird variiert

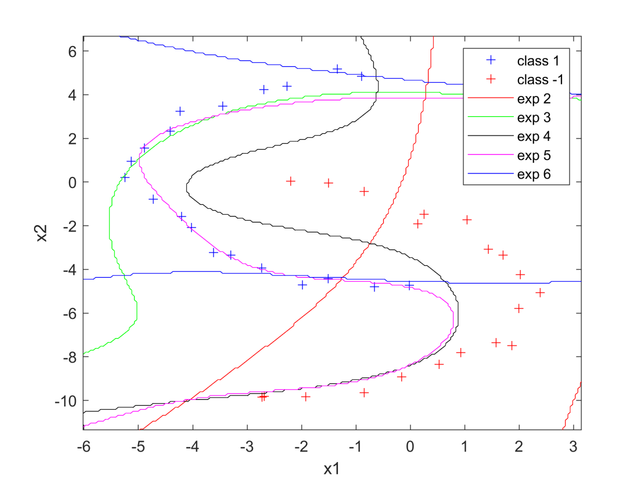


Abbildung 5: Trenngrenzen für polynomiellen Kernel mit verschiedenen Exponenten Q

Beispiel Polynomieller Kernel mit $Q = 4$

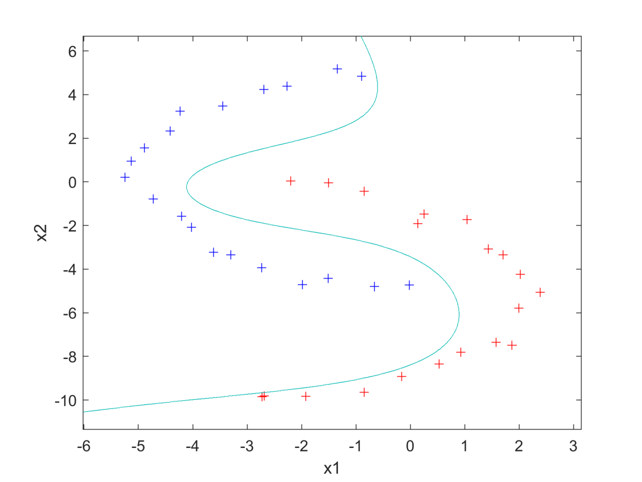


Abbildung 6: Trenngrenzen für polynomiellen Kernel $(ax^T x' + b)^Q$ mit $Q = 4$

Beispiel RBF Kernel, verschiedene γ

Parameter γ muss richtig eingestellt werden

Beispiel RBF Kernel, verschiedene γ

Parameter γ muss richtig eingestellt werden

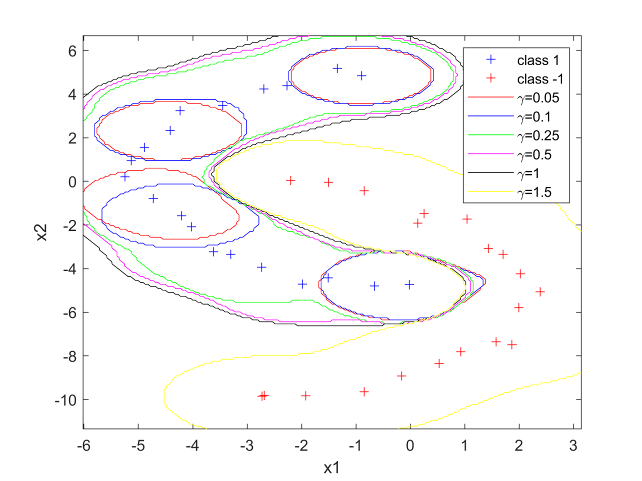


Abbildung 7: Trenngrenzen für RBF Kernel mit verschiedenen γ

Beispiel RBF Kernel, verschiedene γ

Parameter γ muss richtig eingestellt werden

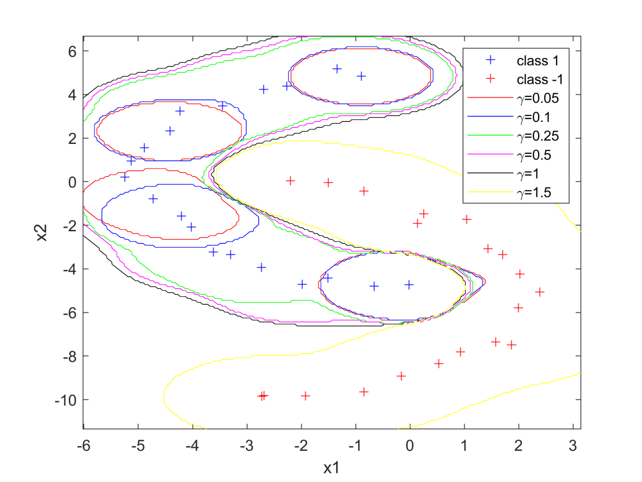


Abbildung 7: Trenngrenzen für RBF Kernel mit verschiedenen γ

Bei falschem γ zu kleine Gebiete

Beispiel RBF Kernel, verschiedene γ

Parameter γ muss richtig eingestellt werden

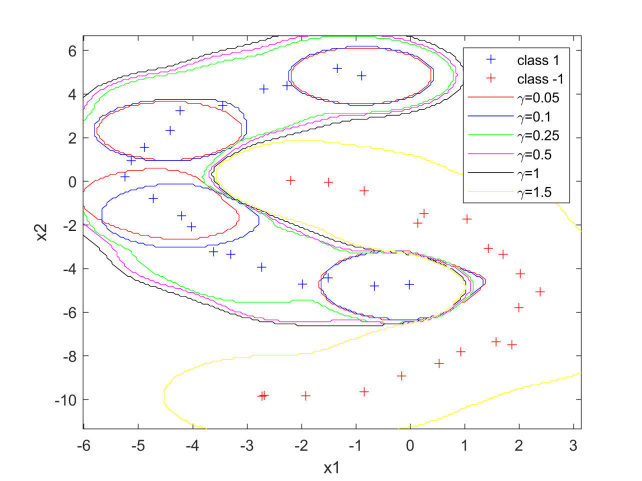


Abbildung 7: Trenngrenzen für RBF Kernel mit verschiedenen γ

Bei falschem γ zu kleine Gebiete

γ gut -> eine Klasse umschlossen

Beispiel RBF Kernel, $\gamma = 1$

Parameter $\gamma = 1$

Beispiel RBF Kernel, $\gamma = 1$

Parameter $\gamma = 1$

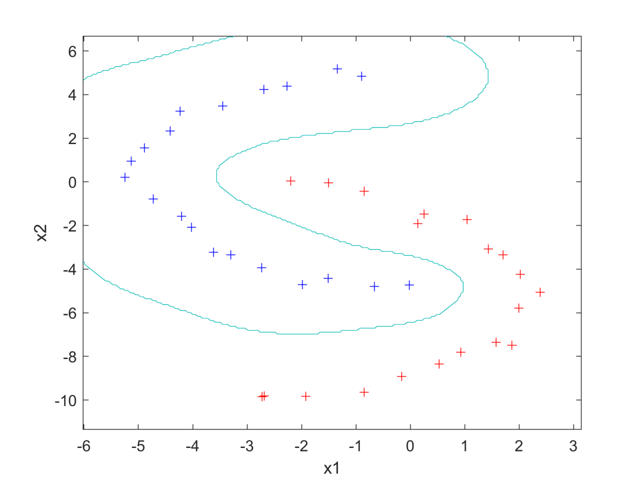
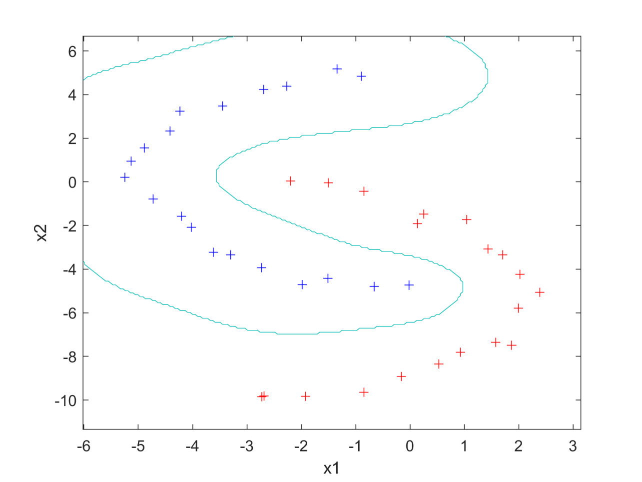


Abbildung 8: Trenngrenzen für RBF Kernel mit $\gamma = 1$

Beispiel RBF Kernel, $\gamma = 1$

Parameter $\gamma = 1$



Blaue Punkte umschlossen

Abbildung 8: Trenngrenzen für RBF Kernel mit $\gamma = 1$

Zusammenfassung

SVM für lineare, binäre Klassifikationsprobleme

Zusammenfassung

SVM für lineare, binäre Klassifikationsprobleme
Hard und Soft Margin (Soft vermindert Einfluss Ausreißer)

Zusammenfassung

SVM für lineare, binäre Klassifikationsprobleme

Hard und Soft Margin (Soft vermindert Einfluss Ausreißer)

Optimierungsproblem -> Lösen mit QP-Solver

Zusammenfassung

SVM für lineare, binäre Klassifikationsprobleme

Hard und Soft Margin (Soft vermindert Einfluss Ausreißer)

Optimierungsproblem -> Lösen mit QP-Solver

Nichtlinear nur durch Transformation Eingabevektoren

Zusammenfassung

SVM für lineare, binäre Klassifikationsprobleme

Hard und Soft Margin (Soft vermindert Einfluss Ausreißer)

Optimierungsproblem -> Lösen mit QP-Solver

Nichtlinear nur durch Transformation Eingabevektoren

Kernel Trick zum Umgehen der Transformation

Fragen?