FH Vorarlberg
University of Applied Sciences

# Support Vector Machines (SVM)

**Computational Intelligence II**

Informatik - Software and Information Engineering
Fachhochschule Vorarlberg

Erstellt von
André Hopfgartner & Matthias Rupp

Dornbirn, am 10. März 2021

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

**SVM** Support Vector Machine

**QP** Quadratic Programming

# 1 Einführung

## 1.1 Intuition

Ziel: Ebene so wählen dass möglichst breites Band zwischen den 2 verschiedenen Klassen entsteht.

Hard Margin vs Soft Margin

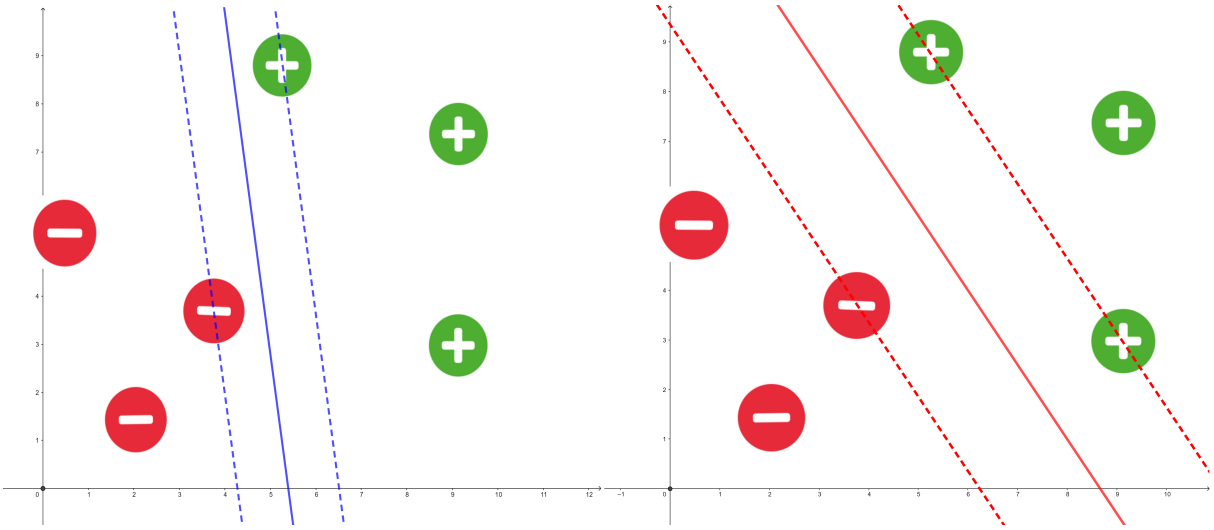

Abbildung 1.1: Abhängig von der Lage der Trennebene entstehen schmale (blau) oder breite (rot) Trennbänder. Das Ziel der SVM ist die Maximierung der Breite des Trennbands durch die Ermittlung der optimalen Lage der Trennebene.

## 1.2 Mathematische Herleitung

TODO ZUSAMMENFASSUNG HERE

### 1.2.1 Problemdefinition

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:

$$w^T x_n + b = 0 \tag{1.1}$$

Mit einer bereits trainierten SVM können neue, noch unklassifizierte Eingabevektoren $x$ wie folgt klassifiziert werden:

$$y = sign(w^T x + b) \qquad \text{gleichbedeutend mit} \tag{1.2a}$$
$$w^T x + b > 0 \qquad \text{für } y = +1 \tag{1.2b}$$
$$w^T x + b < 0 \qquad \text{für } y = -1 \tag{1.2c}$$

Für die Herleitung führen wir eine noch striktere Regel ein. So soll für eine richtige Klassifikation eines gegebenen Eingabevektors $x_n$ mit dem Label $y_n$ gelten:

$$w^T x_n + b \geq +1 \qquad \text{für } y_n = +1 \tag{1.3a}$$
$$w^T x_n + b \leq -1 \qquad \text{für } y_n = -1 \tag{1.3b}$$

Durch Gleichung 1.3 wird sichergestellt, dass Werte aus dem Intervall $]-1, +1[$ als Ergebnis nicht vorkommen können. Wie in Abbildung 1.2 dargestellt entsteht dadurch ein symmetrisches Trennband um die Trennebene in dem keine Punkte liegen. Ziel der SVM ist die Maximierung der Breite dieses Bandes.

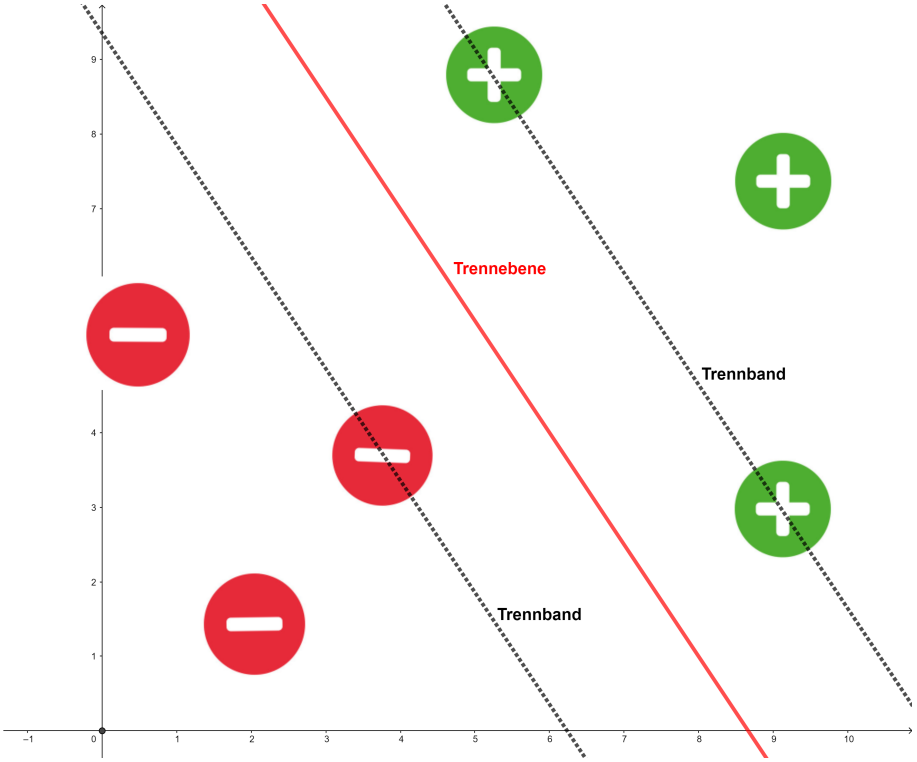


Abbildung 1.2: Um die Trennebene entsteht ein Trennband. Die am nächsten zu der Ebene liegenden Eingabevektoren liegen genau auf den Grenzen des Trennbands.

Gleichung 1.3 kann weiter verallgemeinert werden durch beidseitige Multiplikation mit $y_k$:

$$y_k(w^T x_n + b) \geq 1 \qquad \text{für } y_k = +1 \tag{1.4a}$$

$$y_k(w^T x_n + b) \geq 1 \qquad \text{für } y_k = -1 \tag{1.4b}$$

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

$$y_k(w^T \hat{x} + b) = 1 \tag{1.5}$$

Als nächsten Schritt bestimmen wir den euklidischen Normalabstand $D$ eines beliebigen Punkts $x_n \in \mathbb{R}^K$ zu der Ebene. Hierfür ist zuerst zu bemerken, dass $w$ normal zur definierten Ebene steht.

**Lemma 1.2.1.** *Eine Ebene sei definiert durch $w^T x + b = 0$. Der Vektor $w$ steht normal zu der definierten Ebene.*

*Beweis.* Man wähle zwei Punkte $x_1, x_2 \in \mathbb{R}^K$ die auf der Ebene liegen. Somit muss gelten:

$$\begin{aligned} w^T x_1 + b &= 0 \\ w^T x_2 + b &= 0 \\ w^T(x_1 - x_2) = 0 \leftrightarrow \|w^T\|\|x_1 - x_2\| \cos(\alpha) &= 0 \leftrightarrow \alpha = 90° \end{aligned} \tag{1.6}$$

□

Um den Normalabstand $d$ eines beliebigen Punkts $x_n$ zu ermitteln wählt man einen Punkt $x$, der auf der Ebene liegt, und projiziert den Vektor $(x_n - x)$ auf den Einheitsvektor von $w$. Weil nur der tatsächliche Abstand zur Ebene relevant ist und nicht die Richtung nimmt man den Betrag.

$$\begin{aligned} d &= |\frac{w^T}{\|w\|}(x_n - x)| = \\ &= \frac{1}{\|w\|}|(w^T x_n - w^T x)| = \\ &= \frac{1}{\|w\|}|(w^T x_n + b - (w^T x + b))| \end{aligned} \tag{1.7}$$

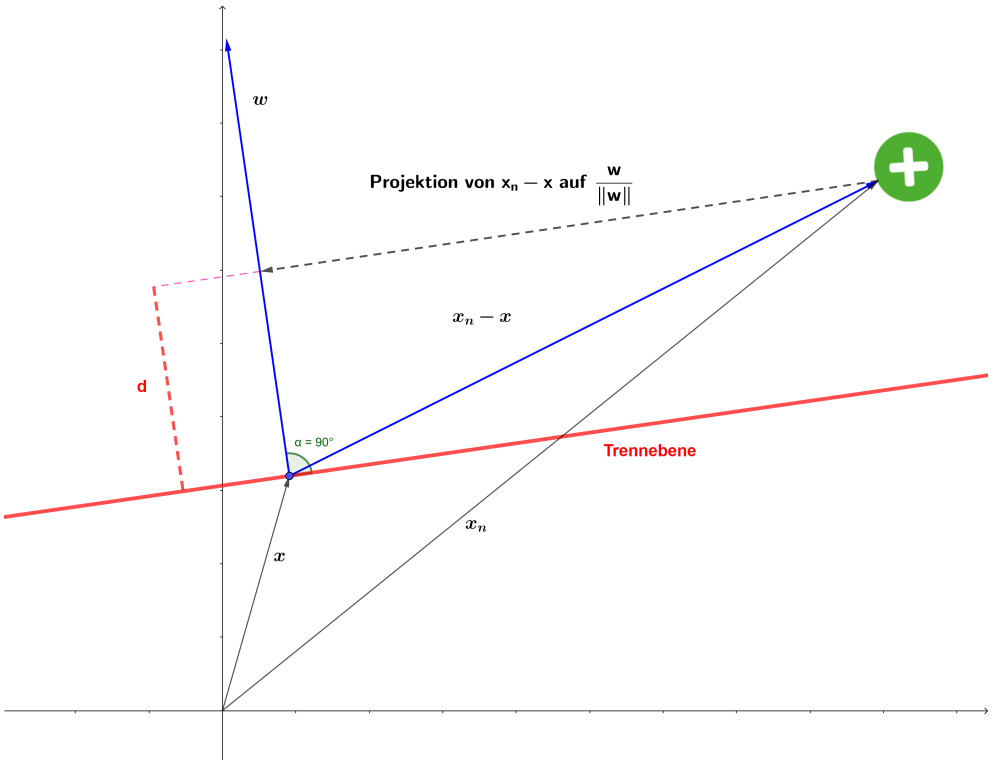


Abbildung 1.3: Durch die Projektion von $(x_n - x)$ auf den Einheitsvektor von $w$ kann der Normalabstand $d$ von $x_n$ zu der Ebene bestimmt werden.

Weil der Punkt $x$ auf der Ebene liegt gilt $w^T x + b = 0$ (Gleichung 1.1):

$$d = \frac{1}{\|w\|} |(w^T x_n + b)| \tag{1.8}$$

Trifft man nun die Annahme, dass $x_n = \hat{x}$ der am nächsten zu der Trenngrenze liegende Punkt ist, so gilt aus Gleichung 1.5 gilt $y_k(w^T \hat{x} + b) = 1 = |w^T \hat{x} + b|$ unter der Annahme, dass der Punkt richtig klassifiziert wurde. Somit ergibt sich der kleinste Abstand zur Trennebene $D = d(\hat{x})$, welcher zugleich der halben Breite des Trennbands entspricht, als:

$$D = \frac{1}{\|w\|} \tag{1.9}$$

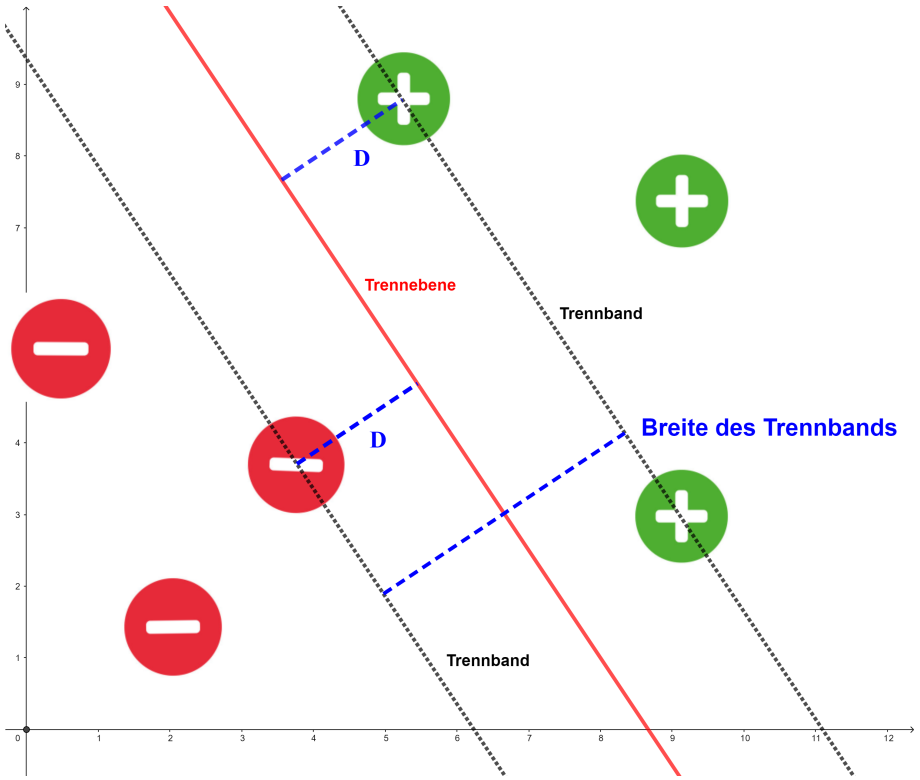


Abbildung 1.4: Der Normalabstand $D$ von der Ebene zu den am nächsten liegenden Eingabevektoren entspricht genau der Hälfte der Breite des Trennbands.

### 1.2.2 Optimierungsproblem

Gleichung 1.9 beschreibt den Normalabstand zu dem am nächsten an der Ebene liegenden Punkt $\hat{x}$. Der Normalabstand ist gleichbedeutend mit der Hälfte der Breite des Trennbands, wobei für die Optimierung der Faktor 2 keine Rolle spielt und vernachlässigt werden kann.

Das Ziel einer SVM ist die Maximierung der Breite des Trennbands für $N$ Eingabevektoren $\{x_1..x_N\}, x_n \in \mathbb{R}^K$. Bei der in diesem Kapitel hergeleiteten Version der SVM müssen die Eingabevektoren linear separierbar sein. Diese Art der SVM wird auch Hard-Margin SVM genannt. Die zweite Version, bei der nicht alle Eingabevektoren linear trennbar sind, heißt Soft-Margin SVM und wird in Unterabschnitt 1.2.5 beschrieben.

Bei dem beschriebenen Problem handelt es sich um ein Optimierungsproblem mit Nebenbedingungen:

$$\max_w \quad \frac{1}{\|w\|} \tag{1.10a}$$

$$\text{mit} \quad \min_{n=1..N} |w^T x_n + b| = 1 \tag{1.10b}$$

Gleichung 1.10b beschreibt hier den am nächsten zur Ebene gelegenen Punkt $\hat{x}$ aus der gebenenen Menge von Eingabevektoren in allgemeiner Form. Der Betrag lässt sich vermeiden durch die Anwendung von Gleichung 1.4:

$$|w^T x_n + b| = y_n(w^T x_n + b) \tag{1.11}$$

Durch Anwendung von Gleichung 1.11 in Gleichung 1.10b, Umformulierung der Maximierung in eine Minimierung und der Verallgemeinerung von $\hat{x}$ auf beliebige Punkte $x_n$ erhält man:

$$\min_w \quad \frac{1}{2} w^T w \tag{1.12a}$$

$$\text{mit} \quad y_n(w^T x_n + b) \geq 1 \text{ für } n = 1..N \tag{1.12b}$$

Die Verallgemeinerung von Gleichung 1.10b auf Gleichung 1.12b auf beliebige Punkte ist so möglich, weil durch Gleichung 1.5 sichergestellt ist, dass für beliebige Punkte $y_n(w^T x_n + b) \geq 1$ gilt.

### 1.2.3 Lagrange Optimierung

Das beschriebene Optimierungsproblem beinhaltet eine Ungleichung in Gleichung 1.12b. Um die Lagrangegleichung aufstellen zu können wird zuerst die Nebenbedingung umgeformt:

$$\min_w \quad \frac{1}{2} w^T w \tag{1.13a}$$

$$\text{mit} \quad y_n(w^T x_n + b) - 1 \geq 0 \text{ für } n = 1..N \tag{1.13b}$$

Das Problem kann nun als Lagrangegleichung dargestellt werden:

$$\min_{w,b} \quad \mathcal{L}(w,b,\alpha) = \frac{1}{2}w^T w - \sum_{n=1}^{N} \alpha_n (y_n(w^T x_n + b) - 1) \tag{1.14a}$$

$$\max_{\alpha_n} \quad \alpha_n \geq 0 \text{ für } n = 1..N \tag{1.14b}$$

Weil Gleichung 1.13b eine Ungleichung der Form $\geq 0$ beinhaltet werden diese Terme von der zu maximierenden Funktion subtrahiert. Nun kann die uneingeschränkte Optimierung von Gleichung 1.14a nach $w$ und $b$ durchgeführt werden indem die Ableitungen bestimmt und 0 gesetzt werden:

$$\begin{aligned} \nabla_w \mathcal{L} = w - \sum_{n=1}^{N} \alpha_n y_n x_n &\stackrel{!}{=} \vec{0} \\ w = \sum_{n=1}^{N} \alpha_n y_n x_n & \end{aligned} \tag{1.15}$$

$$\begin{aligned} \frac{\partial}{\partial b}\mathcal{L} = -\sum_{n=1}^{N} \alpha_n y_n &\stackrel{!}{=} 0 \\ \sum_{n=1}^{N} \alpha_n y_n = 0 & \end{aligned} \tag{1.16}$$

Die Ergebnisse von Gleichung 1.15 und Gleichung 1.16 können in Gleichung 1.14a eingesetzt werden. Hierfür wird zuerst die Summe in Gleichung 1.14a in Teilsummen aufgeteilt:

$$\begin{aligned} \mathcal{L}(w,b,\alpha) &= \frac{1}{2}w^T w - \sum_{n=1}^{N} \alpha_n (y_n(w^T x_n + b) - 1) = \\ &= \frac{1}{2}w^T w - [\sum_{n=1}^{N} \alpha_n y_n b - \sum_{n=1}^{N} \alpha_n + \sum_{n=1}^{N} \alpha_n y_n w^T x_n] \end{aligned} \tag{1.17}$$

Weil $\sum_{n=1}^{N} \alpha_n y_n = 0$ (Gleichung 1.16) gilt fällt der Term $\sum_{n=1}^{N} \alpha_n y_n b$ weg:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}w^T w - [-\sum_{n=1}^{N} \alpha_n + \sum_{n=1}^{N} \alpha_n y_n w^T x_n] \tag{1.18}$$

Vergleicht man den Term $\sum_{n=1}^{N} \alpha_n y_n w^T x_n$ mit dem Ergebnis von Gleichung 1.15 erkennt man, dass $\sum_{n=1}^{N} \alpha_n y_n w^T x_n = w^T w$ gilt. Dies kann ausgeschrieben werden als:

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m \tag{1.19}$$

Gleichung 1.19 beschreibt das Optimierungsproblem ohne Abhängigkeit von $w$ und $b$, wir haben jetzt also eine Maximierung für $\alpha$ mit Nebenbedingungen:

$$\max_{\alpha} \quad \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m \tag{1.20a}$$

$$\text{mit} \quad \alpha_n \geq 0 \text{ für } n = 1..N \tag{1.20b}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N \tag{1.20c}$$

Das in Gleichung 1.20 beschriebene Problem kann beispielsweise mittels eines Quadratic Programming Solvers gelöst werden. Als Ergebnis erhält man einen Vektor $\alpha$ mit allen $\alpha_n$. Durch Einsetzen in $w = \sum_{n=1}^{N} \alpha_n y_n x_n$ kann $w$ bestimmt werden.

Betrachtet man den Ergebnisvektor $\alpha$ so wird man feststellen, dass sehr viele Werte 0 ergeben. In Gleichung 1.14a befindet sich der Term $\alpha_n(y_n(w^T x_n + b) - 1)$. Der Term $(y_n(w^T x_n + b) - 1)$ kann als Schlupf bezeichnet werden. Das Produkt von Schlupf und $\alpha_n$ kann nur 0 werden wenn entweder der Schlupf 0 ist oder $\alpha_n$. Umgekehrt bedeutet dies, dass alle Vektoren, die einen minimalen Abstand zu der Trennebene haben, ein $\alpha_n \neq 0$ aufweisen. Vektoren, die diese Bedingung erfüllen, werden Stützvektoren genannt.

Mit dieser Erkenntnis kann Gleichung 1.15 erneut analysiert werden:

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n \tag{1.21}$$

Weil nur Stützvektoren ein $\alpha_n \neq 0$ aufweisen und somit auch nur Stützvektoren einen Beitrag zu $w$ leisten, kann Gleichung 1.21 stark vereinfacht werden:

$$w = \sum_{n \text{ ist Stützvektor}} \alpha_n y_n x_n \tag{1.22}$$

Der Gewichtsvektor $w$ hängt also lediglich von den Stützvektoren ab, deren Anzahl in der Regel gering ist.

Noch offen ist die Bestimmung des Bias $b$. Weil für Stützvektoren $y_n(w^T x_n + b) = 1$ gilt (Gleichung 1.5) kann der Bias $b$ aus jedem beliebigen Stützvektor bestimmt werden:

$$b = \frac{1}{y_n} - w^T x_n \tag{1.23}$$

### 1.2.4 Lösung mittels Quadratic Programming Solver

Das in Gleichung 1.20 beschriebene Optimierungsproblem kann mittels eines Quadratic Programming Solvers gelöst werden. Hierfür muss das Problem aber umformuliert werden, da die Standardform von QP-Problemen wie folgt ist:

$$\min_x = \frac{1}{2}x^T Q x + cx + d \tag{1.24}$$

$Q \in \mathbb{R}^{n \times n}$ ist eine symmetrische reelle Matrix mit Koeffizienten, die einen quadratischen Beitrag leisten. $c \in \mathbb{R}$ ist ein Faktor für den linearen Beitrag und $d \in \mathbb{R}$ ist ein fixer Anteil.

Unter der Voraussetzung, dass $\max_x f(x) = \min_x(-f(x))$ gilt können wir unser Maximierungsproblem aus Gleichung 1.20 in ein Minimierungsproblem umformen:

$$\min_\alpha \mathcal{L}(\alpha) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m - \sum_{n=1}^{N}\alpha_n \tag{1.25}$$

Als nächsten Schritt müssen wir die Koeffizienten aus den Summen extrahieren, $\alpha$ und $y$ als Vektor darstellen, das Problem in die Standard-QP-Form bringen und die Nebenbedingungen als Matrizenmultiplikation darstellen:

$$\min_\alpha \quad \mathcal{L}(\alpha) = \frac{1}{2}\alpha^T Q \alpha + (-1^T)\alpha \tag{1.26a}$$

$$\text{mit} \quad Q = \begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \dots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \dots & y_2 y_N x_2^T x_N \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \dots & y_N y_N x_N^T x_N \end{bmatrix} \tag{1.26b}$$

$$\text{für} \quad y^T \alpha = 0 \tag{1.26c}$$

$$0 \leq \alpha \leq \infty \tag{1.26d}$$

An dieser Stelle ist zu bemerken dass es sich bei der Matrix $Q$ um eine $N \times N$ Matrix handelt. Für eine große Anzahl an Trainingsdaten ist dies sehr problematisch beziehungsweise nicht mehr lösbar. Hier können andere Optimierungsverfahren wie beispielsweise in Platt (1998) beschrieben verwendet werden.

Das Problem in dieser Darstellung können wir direkt an ein Quadratic-Programming Solver Framework übergeben und erhalten einen Vektor $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ als Ergebnis.

Daraus können wir, wie bereits zuvor erwähnt, $w$ und $b$ bestimmen:

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n \tag{1.27}$$

Für die Biasberechnung benötigen wir einen beliebigen Stützvektor $x_k$. Wir finden einen Stützvektor, indem wir den Ergebnisvektor $\alpha$ betrachten. Ein Stützvektor $x_k$ muss $\alpha_k \neq 0$ aufweisen.

$$b = \frac{1}{y_k} - w^T x_k \tag{1.28}$$

Die SVM ist nun vollständig definiert, neue Eingabevektoren $x$ können wie folgt klassifiziert werden:

$$y = sign(w^T x + b) \tag{1.29}$$

### 1.2.5 Soft Margin SVM

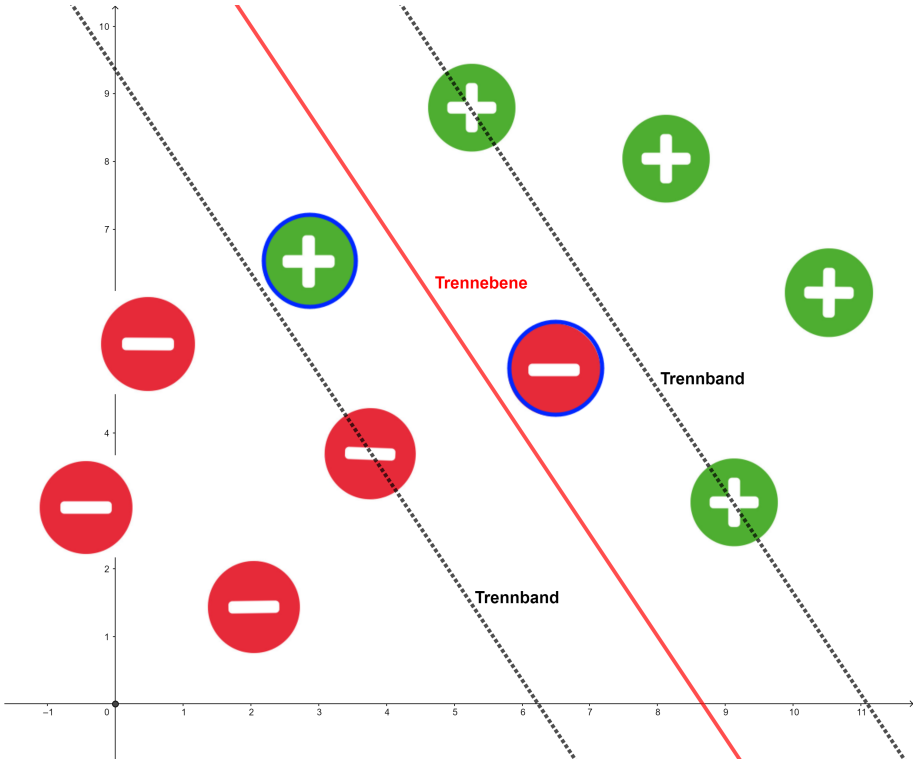


Abbildung 1.5: Die Eingabevektoren können nicht ohne einzelne Fehlklassifizierungen (blau markiert) linear getrennt werden.

Bisher wurde angenommen, dass alle Eingabevektoren linear separierbar sind ohne dass Fehlklassifikationen aufreten. Dieses Problem kann durch eine Hard-Margin SVM, wie in den Kapiteln zuvor gezeigt, gelöst werden. Trifft diese Annahme für eine Menge von Eingabevektoren nicht mehr zu, wie in Abbildung 1.5 dargestellt, wird der bisher beschriebene Algorithmus keine lineare Trennebene finden. Durch die Einführung von positiven Fehlervariablen $\xi_n \in \mathbb{R}^K, \xi_n \geq 0$ in Gleichung 1.3 kann dieses

Problem umgangen werden und trotzdem eine fehlerbehaftete, lineare Trennebene gefunden werden:

$$w^T x_n + b \geq +1 - \xi_n \qquad \text{für } y_n = +1 \tag{1.30a}$$

$$w^T x_n + b \leq -1 + \xi_n \qquad \text{für } y_n = -1 \tag{1.30b}$$

Damit ein Eingabevektor $x_n$ nun, gemäß Gleichung 1.30, falsch klassifiziert werden kann muss der zu dem Vektor gehörende Fehler $\xi_n$ über 1 steigen. Eine obere Grenze für die Anzahl der aufgetretenen Fehler kann somit beschrieben werden durch $\sum_{n=1}^{N} \xi_n$.

Basierend auf dieser Grundlage kann eine Kostenfunktion $E$ basierend auf der Anzahl der Fehler formuliert werden:

$$E = C(\sum_{n=1}^{N} \xi_n) \tag{1.31}$$

Der Parameter $C \in \mathbb{R}, C \geq 0$ bestimmt die Höhe der Bestrafung von Fehlern und kann frei gewählt werden.

Erweitert man die zu minimierende Funktion $\frac{1}{2} w^T w$ aus Gleichung 1.13 um die Kostenfunktion $C(\sum_{n=1}^{N} \xi_n)$ so erhält man ein neues Optimierungsproblem:

$$\min_{w} \qquad \frac{1}{2} w^T w + C(\sum_{n=1}^{N} \xi_n) \tag{1.32a}$$

$$\text{mit} \qquad y_n(w^T x_n + b) - 1 \geq 0 \text{ für } n = 1..N \tag{1.32b}$$

Wendet man die in Unterabschnitt 1.2.3 beschriebenen Schritte auf das in Gleichung 1.32 beschriebene Problem an erhält man folgendes Optimierungsproblem:

$$\max_{\alpha} \qquad \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n^T x_m \tag{1.33a}$$

$$\text{mit} \qquad 0 \leq \alpha_n \leq C \text{ für } n = 1..N \tag{1.33b}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \text{ für } n = 1..N \tag{1.33c}$$

Sehr bemerkenswert hier ist, dass der einzige Unterschied zu Gleichung 1.20 die Beschränkung der Lagrange Multiplikatoren durch $C$ ist. Umgekehrt bedeutet dies, dass eine Hard-Margin SVM durch Gleichung 1.33 beschrieben werden kann wenn der Parameter $C$ sehr groß gewählt wird.

### 1.2.6 Transformation der Problemstellung in eine höhere Dimension

Idee: nichtlineare Transformation der Eingabevektoren in beliebig hohe Dimension

Vorteil: Einzige Zusatzkosten sind Skalarprodukt in Optimierung, Anzahl Alphas und Dimension der Gram-Matrix bleibt gleich weil abhängig von Anzahl Testdaten und nicht der Dimension

Generalisierungsverhalten: Anzahl SV dividiert durch Anzahl Testdaten -> möglichst kleine Zahl

Bei Kernel-Trick: sogar kosten für Skalarprodukt erspart

### 1.2.7 Kernel Methode

# Literatur

Platt, John (Apr. 1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. MSR-TR-98-14, S. 21. URL: https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/.