

Todo list

multiclass separation summary text	28
one vs all section	28
one vs one section	28

Support Vector Machines (SVM)

Computational Intelligence II

Informatik - Software and Information Engineering
Fachhochschule Vorarlberg

Erstellt von
André Hopfgartner & Matthias Rupp

Dornbirn, am 14. März 2021

Inhaltsverzeichnis

Abkürzungsverzeichnis	5
1 Intuition	6
2 Hard-Margin Support Vector Machine	7
2.1 Problemdefinition	7
2.2 Optimierungsproblem	10
2.3 Lagrange Optimierung	11
2.4 Lösung mittels Quadratic Programming Solver	14
3 Soft-Margin Support Vector Machine	16
4 Vergleich Soft-Margin und Hard-Margin Support Vector Machine	18
5 Nichtlineare Trennung	20
5.1 Transformation der Problemstellung	20
5.2 Kernel Trick	22
5.2.1 Beispiel Kernel-Funktion	22
5.2.2 Polynomieller Kernel	23
5.2.3 Radial Basis Function Kernel	24
5.2.4 SVM Definition mit Kernel	24
5.2.5 Anforderungen an eine Kernel-Funktion	25
6 Generalisierungsmetrik	27
7 Multiclass Separation	28
7.1 One vs All	28
7.2 One vs One	28
Weiterführende Ressourcen	29

Abkürzungsverzeichnis

SVM Support Vector Machine

QP Quadratic Programming

RBF Radial Basis Function

1 Intuition

Das Ziel einer Support Vector Machine (SVM) ist die lineare Separation von zwei verschiedenen Klassen. Die Separation wird durch eine Ebene durchgeführt. Die Lage der Ebene wird so gewählt, dass um die Ebene ein möglichst breites Band entsteht. Abbildung 1.1 zeigt Beispiele für solche Trennungen. Weiters gibt es zwei Arten von SVMs:

- Eine Hard-Margin SVM trennt die Klassen so, dass keine Fehlklassifikationen entstehen.
- Eine Soft-Margin SVM erlaubt einzelne Fehlklassifikationen damit eine mitunter bessere Trennebene gefunden werden kann.

Diese beiden Arten von SVMs werden in den folgenden Kapiteln genauer betrachtet. Die Qualität der Trennung wird bei einer SVM durch die Breite des Trennbands beurteilt.

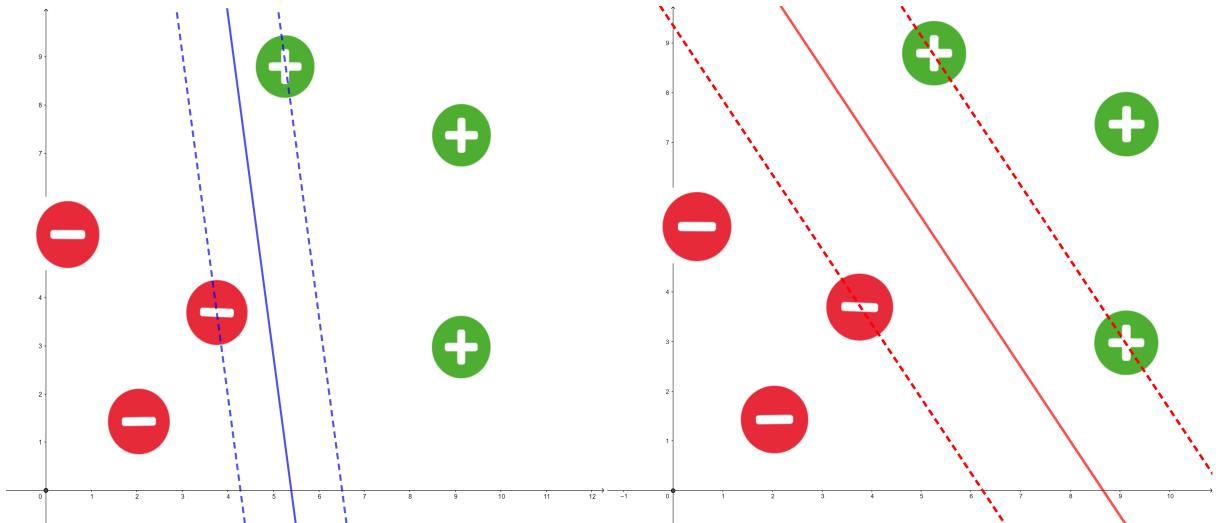


Abbildung 1.1: Abhängig von der Lage der Trennebene entstehen schmale (blau) oder breite (rot) Trennbänder. Das Ziel der SVM ist die Maximierung der Breite des Trennbands durch die Ermittlung der optimalen Lage der Trennebene.

2 Hard-Margin Support Vector Machine

Wie bereits in der Einführung erwähnt ist das Ziel einer Hard-Margin SVM ist die Trennung zweier Klassen ohne Fehler mit einem möglichst breiten Trennband. Diese Aussage wird in diesem Kapitel mathematisch formuliert und als Optimierungsproblem dargestellt. Weiters wird beschrieben wie dieses Problem in die Standardform von Quadratic Programming (QP) Problemen transformiert werden kann was für die Lösung nötig ist.

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:

$$w^T x_n + b = 0 \quad (2.1)$$

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

$$y = \text{sign}(w^T x + b) \quad \text{gleichbedeutend mit} \quad (2.2a)$$

$$w^T x + b > 0 \quad \text{für } y = +1 \quad (2.2b)$$

$$w^T x + b < 0 \quad \text{für } y = -1 \quad (2.2c)$$

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:

$$w^T x_n + b \geq +1 \quad \text{für } y_n = +1 \quad (2.3a)$$

$$w^T x_n + b \leq -1 \quad \text{für } y_n = -1 \quad (2.3b)$$

Durch Gleichung 2.3 wird sichergestellt, dass Werte aus dem Intervall $]-1, +1[$ als Ergebnis nicht vorkommen können. Wie in Abbildung 2.1 dargestellt entsteht dadurch ein symmetrisches Trennband um die Trennebene in dem keine Punkte liegen. Ziel der SVM ist die Maximierung der Breite dieses Bandes.

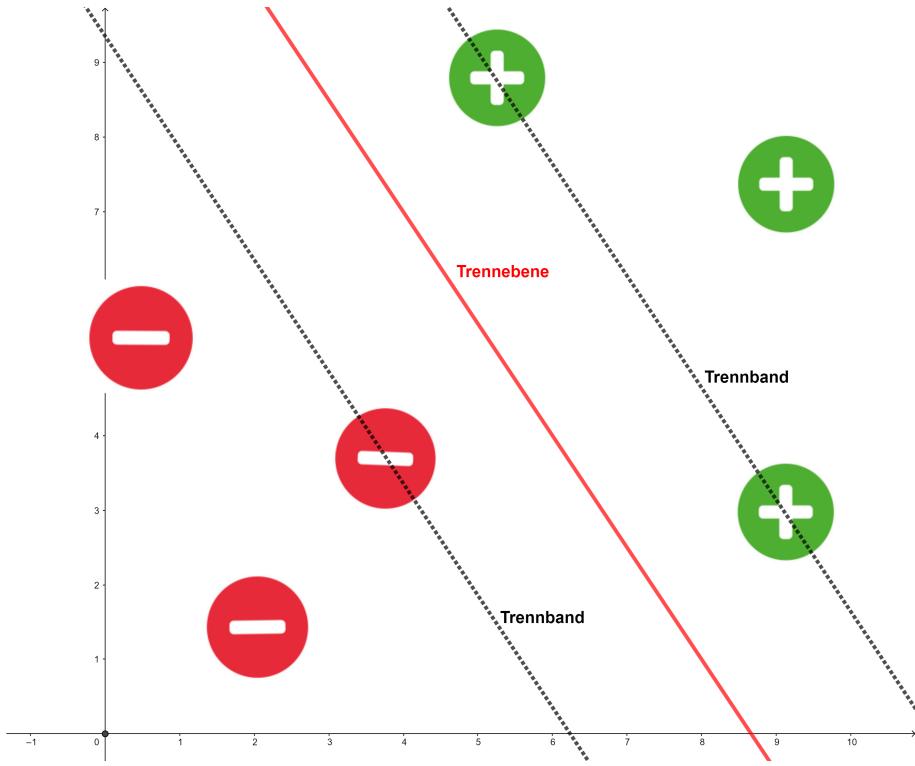


Abbildung 2.1: Um die Trennebene entsteht ein Trennband. Die am nächsten zu der Ebene liegenden Eingabevektoren liegen genau auf den Grenzen des Trennbands.

Gleichung 2.3 kann weiter verallgemeinert werden durch beidseitige Multiplikation mit y_k :

$$y_k(w^T x_n + b) \geq 1 \quad \text{für } y_k = +1 \quad (2.4a)$$

$$y_k(w^T x_n + b) \geq 1 \quad \text{für } y_k = -1 \quad (2.4b)$$

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

$$y_k(w^T \hat{x} + b) = 1 \quad (2.5)$$

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 2.1.1. 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:

$$\begin{aligned} w^T x_1 + b &= 0 \\ w^T x_2 + b &= 0 \\ w^T(x_1 - x_2) = 0 &\Leftrightarrow \|w^T\| \|x_1 - x_2\| \cos(\alpha) = 0 \Leftrightarrow \alpha = 90^\circ \end{aligned} \tag{2.6}$$

□

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.

$$\begin{aligned} d &= \left| \frac{w^T}{\|w\|} (x_n - x) \right| = \\ &= \frac{1}{\|w\|} |(w^T x_n - w^T x)| = \\ &= \frac{1}{\|w\|} |(w^T x_n + b - (w^T x + b))| \end{aligned} \tag{2.7}$$

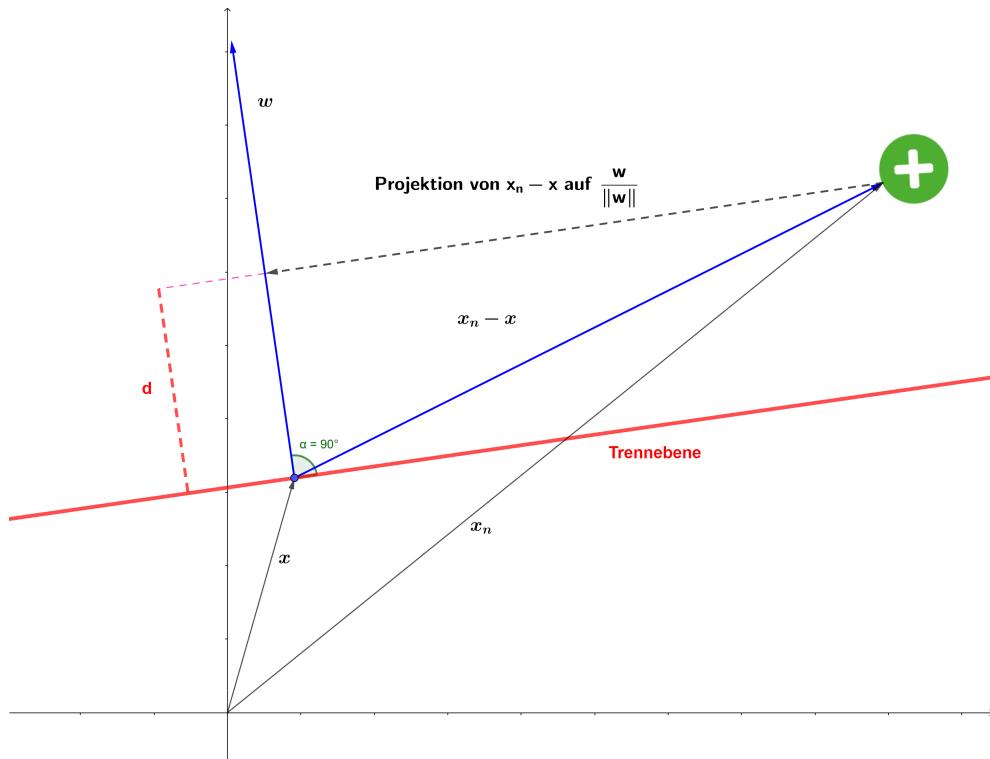


Abbildung 2.2: Durch die Projektion von $(x_n - x)$ auf den Einheitsvektor von w kann der Normalabstand d von x_n zu der Ebene bestimmt werden.

Weil der Punkt x auf der Ebene liegt gilt $w^T x + b = 0$ (Gleichung 2.1):

$$d = \frac{1}{\|w\|} |(w^T x_n + b)| \quad (2.8)$$

Trifft man nun die Annahme, dass $x_n = \hat{x}$ der am nächsten zu der Trenngrenze liegende Punkt ist, so gilt aus Gleichung 2.5 $y_k(w^T \hat{x} + b) = 1 = |w^T \hat{x} + b|$ 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:

$$D = \frac{1}{\|w\|} \quad (2.9)$$

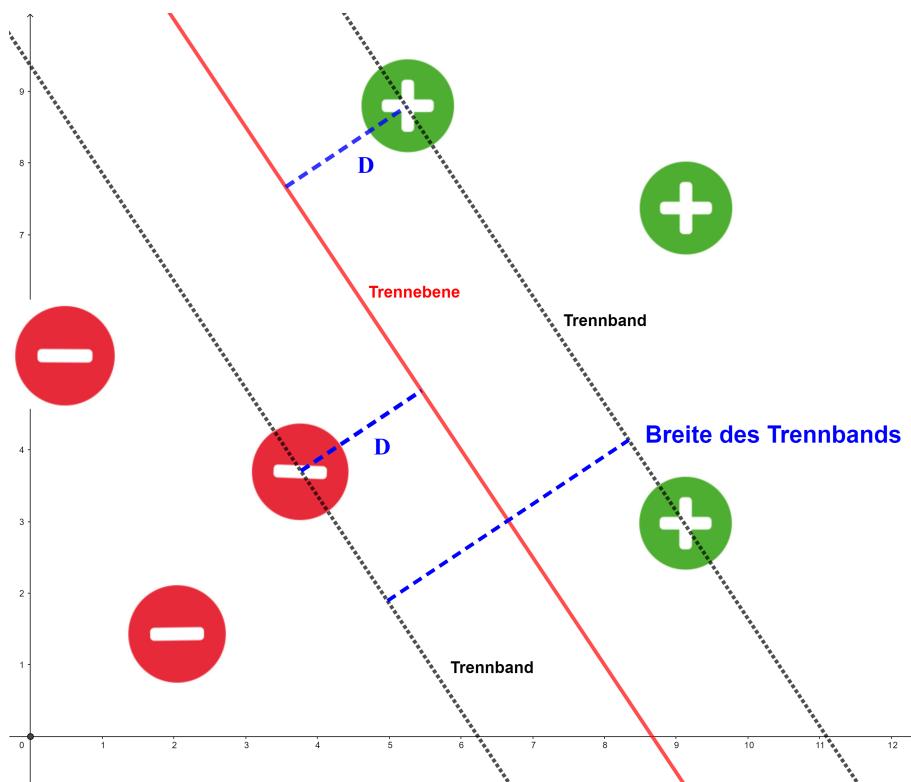


Abbildung 2.3: Der Normalabstand D von der Ebene zu den am nächsten liegenden Eingabevektoren entspricht genau der Hälfte der Breite des Trennbands.

2.2 Optimierungsproblem

Gleichung 2.9 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 dem beschriebenen Problem handelt es sich um ein Optimierungsproblem mit Nebenbedingungen:

$$\max_w \quad \frac{1}{\|w\|} \quad (2.10a)$$

$$\text{mit} \quad \min_{n=1..N} |w^T x_n + b| = 1 \quad (2.10b)$$

Gleichung 2.10b beschreibt hier den am nächsten zur Ebene gelegenen Punkt \hat{x} aus der gebenen Menge von Eingabevektoren in allgemeiner Form. Der Betrag lässt sich vermeiden durch die Anwendung von Gleichung 2.4:

$$|w^T x_n + b| = y_n(w^T x_n + b) \quad (2.11)$$

Durch Anwendung von Gleichung 2.11 in Gleichung 2.10b, Umformulierung der Maximierung in eine Minimierung und der Verallgemeinerung von \hat{x} auf beliebige Punkte x_n erhält man:

$$\min_w \quad \frac{1}{2} w^T w \quad (2.12a)$$

$$\text{mit} \quad y_n(w^T x_n + b) \geq 1 \text{ für } n = 1..N \quad (2.12b)$$

Die Verallgemeinerung von Gleichung 2.10b auf Gleichung 2.12b auf beliebige Punkte ist so möglich, weil durch Gleichung 2.5 sichergestellt ist, dass für beliebige Punkte $y_n(w^T x_n + b) \geq 1$ gilt.

2.3 Lagrange Optimierung

Das beschriebene Optimierungsproblem beinhaltet eine Ungleichung in Gleichung 2.12b. Um die Lagrangegleichung aufzustellen zu können wird zuerst die Nebenbedingung umgeformt:

$$\min_w \quad \frac{1}{2} w^T w \quad (2.13a)$$

$$\text{mit} \quad y_n(w^T x_n + b) - 1 \geq 0 \text{ für } n = 1..N \quad (2.13b)$$

Das Problem kann nun als Lagrangegleichung dargestellt werden:

$$\min_{w,b} \quad \mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{n=1}^N \alpha_n (y_n(w^T x_n + b) - 1) \quad (2.14a)$$

$$\max_{\alpha_n} \quad \alpha_n \geq 0 \text{ für } n = 1..N \quad (2.14b)$$

Weil Gleichung 2.13b eine Ungleichung der Form ≥ 0 beinhaltet werden diese Terme von der zu maximierenden Funktion subtrahiert. Nun kann die uneingeschränkte Optimierung von Gleichung 2.14a nach w und b durchgeführt werden indem die Ableitungen bestimmt und 0 gesetzt werden:

$$\begin{aligned}\nabla_w \mathcal{L} &= w - \sum_{n=1}^N \alpha_n y_n x_n \stackrel{!}{=} \vec{0} \\ w &= \sum_{n=1}^N \alpha_n y_n x_n\end{aligned}\tag{2.15}$$

$$\begin{aligned}\frac{\partial}{\partial b} \mathcal{L} &= - \sum_{n=1}^N \alpha_n y_n \stackrel{!}{=} 0 \\ \sum_{n=1}^N \alpha_n y_n &= 0\end{aligned}\tag{2.16}$$

Die Ergebnisse von Gleichung 2.15 und Gleichung 2.16 können in Gleichung 2.14a eingesetzt werden. Hierfür wird zuerst die Summe in Gleichung 2.14a in Teilsummen aufgeteilt:

$$\begin{aligned}\mathcal{L}(w, b, \alpha) &= \frac{1}{2} w^T w - \sum_{n=1}^N \alpha_n (y_n (w^T x_n + b) - 1) = \\ &= \frac{1}{2} w^T w - \left[\sum_{n=1}^N \alpha_n y_n b - \sum_{n=1}^N \alpha_n + \sum_{n=1}^N \alpha_n y_n w^T x_n \right]\end{aligned}\tag{2.17}$$

Weil $\sum_{n=1}^N \alpha_n y_n = 0$ (Gleichung 2.16) gilt fällt der Term $\sum_{n=1}^N \alpha_n y_n b$ weg:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \left[- \sum_{n=1}^N \alpha_n + \sum_{n=1}^N \alpha_n y_n w^T x_n \right]\tag{2.18}$$

Vergleicht man den Term $\sum_{n=1}^N \alpha_n y_n w^T x_n$ mit dem Ergebnis von Gleichung 2.15 erkennt man, dass $\sum_{n=1}^N \alpha_n y_n w^T x_n = w^T w$ gilt. Dies kann ausgeschrieben werden als:

$$\mathcal{L}(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^M y_n y_m \alpha_n \alpha_m x_n^T x_m\tag{2.19}$$

Gleichung 2.19 beschreibt das Optimierungsproblem ohne Abhängigkeit von w und

b , wir haben jetzt also eine Maximierung für α mit Nebenbedingungen:

$$\max_{\alpha} \quad \mathcal{L}(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^M y_n y_m \alpha_n \alpha_m x_n^T x_m \quad (2.20a)$$

$$\text{mit} \quad \alpha_n \geq 0 \text{ für } n = 1..N \quad (2.20b)$$

$$\sum_{n=1}^N \alpha_n y_n = 0 \text{ für } n = 1..N \quad (2.20c)$$

Bei dem in Gleichung 2.20 beschriebene Problem handelt es sich um ein Quadratic Programming Problem, ersichtlich an dem Term $x_n^T x_m$, welches beispielsweise mittels eines Quadratic Programming Solvers gelöst werden kann. Als Ergebnis erhält man einen Vektor α mit allen α_n . Durch Einsetzen in $w = \sum_{n=1}^N \alpha_n y_n x_n$ kann w bestimmt werden.

Betrachtet man den Ergebnisvektor α so wird man feststellen, dass sehr viele Werte 0 ergeben. In Gleichung 2.14a 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 α_n kann nur 0 werden wenn entweder der Schlupf 0 ist oder α_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 2.15 erneut analysiert werden:

$$w = \sum_{n=1}^N \alpha_n y_n x_n \quad (2.21)$$

Weil nur Stützvektoren ein $\alpha_n \neq 0$ aufweisen und somit auch nur Stützvektoren einen Beitrag zu w leisten, kann Gleichung 2.21 stark vereinfacht werden:

$$w = \sum_{n \text{ ist Stützvektor}} \alpha_n y_n x_n \quad (2.22)$$

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 2.5) kann der Bias b aus jedem beliebigen Stützvektor bestimmt werden:

$$b = \frac{1}{y_n} - w^T x_n \quad (2.23)$$

2.4 Lösung mittels Quadratic Programming Solver

Weil es sich bei dem in Gleichung 2.20 beschriebene Optimierungsproblem um ein Quadratic Programming Problem handelt kann dieses auch mittels eines Quadratic Programming Solvers gelöst werden. Die Implementationsdetails für diese Lösungsverfahren werden an dieser Stelle nicht weiter ausgeführt. Für die Anwendung eines Quadratic Programming Solvers muss das Problem in die Standardform von QP Problemen umformuliert werden.

$$\min_x = \frac{1}{2}x^T Qx + cx + d \quad (2.24)$$

$Q \in \mathbb{R}^{n \times n}$ 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 2.20 in ein Minimierungsproblem umformen:

$$\min_{\alpha} \mathcal{L}(\alpha) = \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^M y_n y_m \alpha_n \alpha_m x_n^T x_m - \sum_{n=1}^N \alpha_n \quad (2.25)$$

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

$$\min_{\alpha} \mathcal{L}(\alpha) = \frac{1}{2} \alpha^T Q \alpha + (-1^T) \alpha \quad (2.26a)$$

$$\text{mit } Q = \begin{bmatrix} y_1 y_1 x_1^T & y_1 y_2 x_1^T & \dots & y_1 y_N x_1^T \\ y_2 y_1 x_2^T & y_2 y_2 x_2^T & \dots & y_2 y_N x_2^T \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 x_N^T & y_N y_2 x_N^T & \dots & y_N y_N x_N^T \end{bmatrix} \quad (2.26b)$$

$$\text{für } y^T \alpha = 0 \quad (2.26c)$$

$$0 \leq \alpha \leq \infty \quad (2.26d)$$

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 $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ als Ergebnis.

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

$$w = \sum_{n=1}^N \alpha_n y_n x_n \quad (2.27)$$

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

$$b = \frac{1}{y_k} - w^T x_k \quad (2.28)$$

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

$$y = \text{sign}(w^T x + b) \quad (2.29)$$

3 Soft-Margin Support Vector Machine

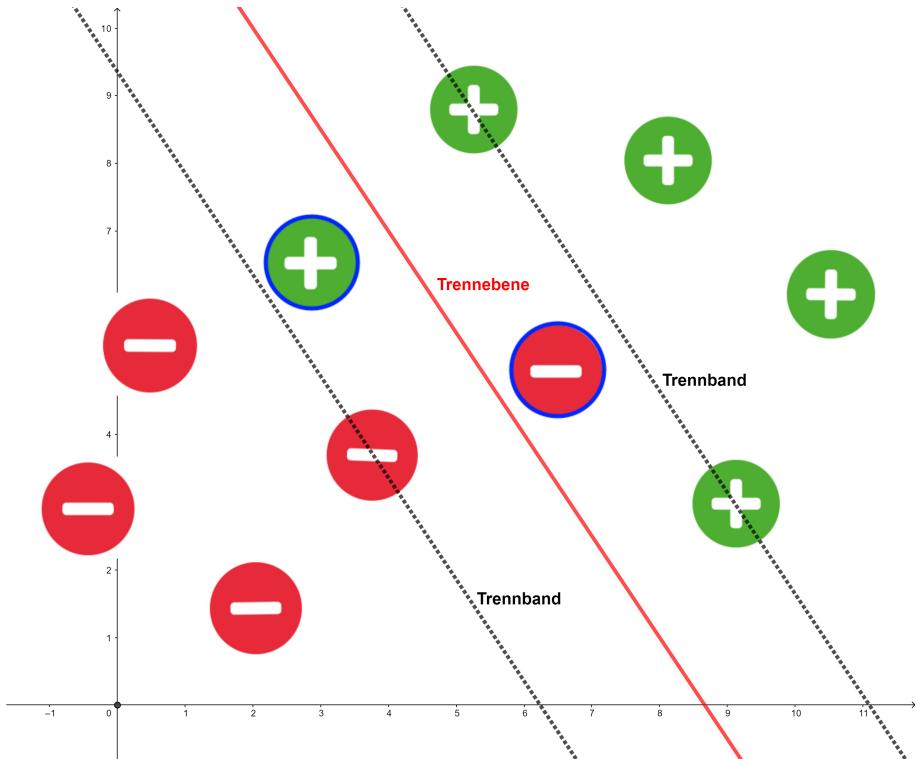


Abbildung 3.1: Die Eingabevektoren können nicht ohne einzelne Fehlklassifizierungen (blau markiert) linear getrennt werden.

In den Überlegungen in Kapitel 2 wurde angenommen, dass alle Eingabevektoren linear separierbar sind ohne dass Fehlklassifikationen auftreten. 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 3.1 dargestellt, kann 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 2.3 kann dieses Problem umgangen werden und trotzdem eine fehlerbehaftete, lineare Trennebene gefunden werden:

$$w^T x_n + b \geq +1 - \xi_n \quad \text{für } y_n = +1 \quad (3.1a)$$

$$w^T x_n + b \leq -1 + \xi_n \quad \text{für } y_n = -1 \quad (3.1b)$$

Damit ein Eingabevektor x_n nun, gemäß Gleichung 3.1, falsch klassifiziert werden kann muss der zu dem Vektor gehörende Fehler ξ_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:

$$E = C \left(\sum_{n=1}^N \xi_n \right) \quad (3.2)$$

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 2.13 um die Kostenfunktion $C(\sum_{n=1}^N \xi_n)$ so erhält man ein neues Optimierungsproblem:

$$\min_w \quad \frac{1}{2} w^T w + C \left(\sum_{n=1}^N \xi_n \right) \quad (3.3a)$$

$$\text{mit} \quad y_n(w^T x_n + b) - 1 \geq 0 \quad \text{für } n = 1..N \quad (3.3b)$$

Wendet man die in Abschnitt 2.3 beschriebenen Schritte auf das in Gleichung 3.3 beschriebene Problem an erhält man folgendes Optimierungsproblem:

$$\max_{\alpha} \quad \mathcal{L}(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^M y_n y_m \alpha_n \alpha_m x_n^T x_m \quad (3.4a)$$

$$\text{mit} \quad 0 \leq \alpha_n \leq C \quad \text{für } n = 1..N \quad (3.4b)$$

$$\sum_{n=1}^N \alpha_n y_n = 0 \quad \text{für } n = 1..N \quad (3.4c)$$

Sehr bemerkenswert hier ist, dass der einzige Unterschied zu Gleichung 2.20 die Beschränkung der Lagrange Multiplikatoren durch C ist. Umgekehrt bedeutet dies, dass eine Hard-Margin SVM durch Gleichung 3.4 beschrieben werden kann wenn der Parameter C sehr groß gewählt wird.

4 Vergleich Soft-Margin und Hard-Margin Support Vector Machine

Ein großer Vorteil der Soft-Margin SVM ist die Verminderung des Einflusses von Ausreißern auf die Trenngrenze. Bei der Hard-Margin SVM kann mitunter ein Ausreißer, der näher als alle anderen Punkte bei der anderen Klasse liegt, die Lage der Trenngrenze bestimmen. Die so bestimmte Trenngrenze liegt mitunter viel näher bei der abzugrenzenden Klasse als eine optimale Trenngrenze liegen würde. Ein Beispiel für eine solche Situation ist in Abbildung 4.1 dargestellt. Die Soft-Margin SVM hat die Möglichkeit Fehlklassifikationen zuzulassen und kann dadurch bessere liegende Trennebenen finden die zu stabileren Klassifikationsergebnissen für neue Daten führt.

Aufgrund des zuvor genannten Verhaltens tendiert die Hard-Margin SVM im Allgemeinen zu Overfitting. Aus diesem Grund sollte die Soft-Margin SVM bevorzugt werden und der Parameter C abhängig von den gegebenen Trainingsdaten gewählt werden.

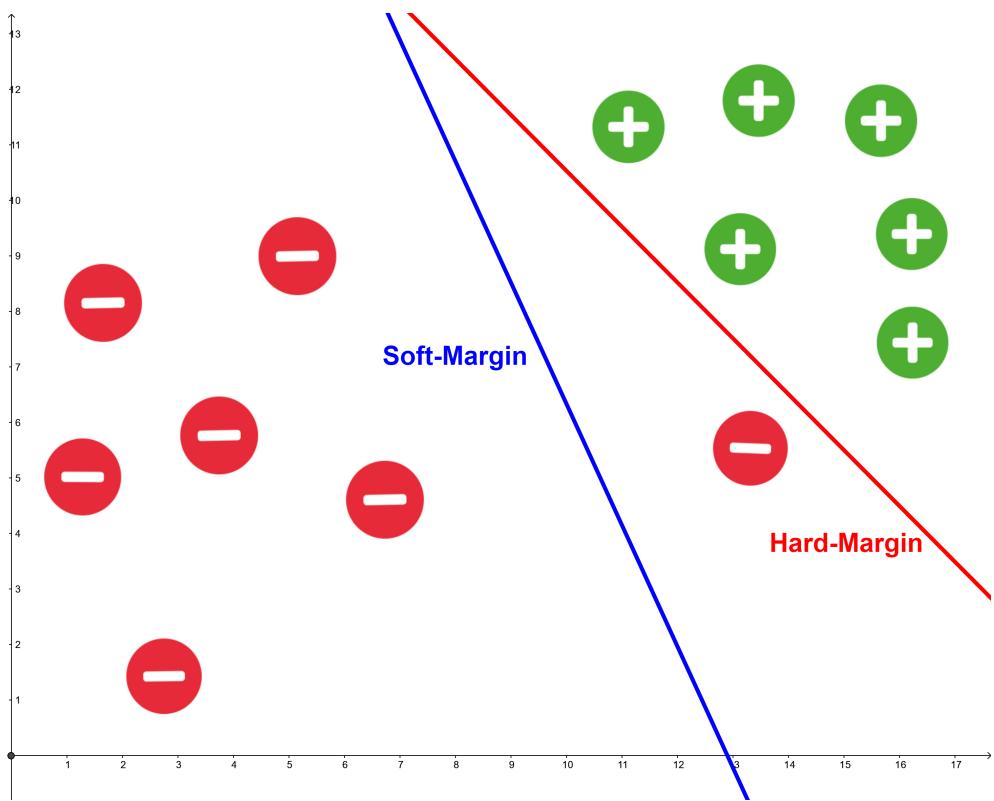


Abbildung 4.1: Die Hard-Margin SVM trennt die Klassen zwar ohne Fehler, die Lage der Trennebene ist allerdings sehr nahe bei den grünen Punkten. Dadurch werden neue Eingabevektoren, die nahe bei der Trenngrenze liegen, mit großer Wahrscheinlichkeit falsch klassifiziert. Die Soft-Margin SVM „opfert“ eine Fehlklassifikation für eine viel bessere, stabilere Lage der Trenngrenze.

5 Nichtlineare Trennung

Wird eine nichtlineare Trennung von Eingabevektoren gewünscht lässt sich dies nicht direkt durch eine SVM realisieren, da eine SVM in ihrer Grundform ausschließlich linear trennen kann. Werden die Eingabevektoren allerdings mittels einer Funktion so transformiert, dass diese linear trennbar sind, kann eine SVM auch in ihrer Grundform nicht linear trennbare Eingabevektoren trennen. In Abschnitt 5.1 wird zuerst eine Intuition für die Transformation aufgebaut. Diese Intuition wird anschließend in Abschnitt 5.2 weiter formalisiert und verallgemeinert.

5.1 Transformation der Problemstellung

Sind die Eingabevektoren nicht linear trennbar können diese mittels einer Transformation $\Phi(x) : \mathbb{R}^K \rightarrow \mathbb{R}^L$ in einen Raum, in dem diese trennbar sind, transformiert werden. Beispiele für solche Transformationen sind in Abbildung 5.1 und Abbildung 5.2 dargestellt.

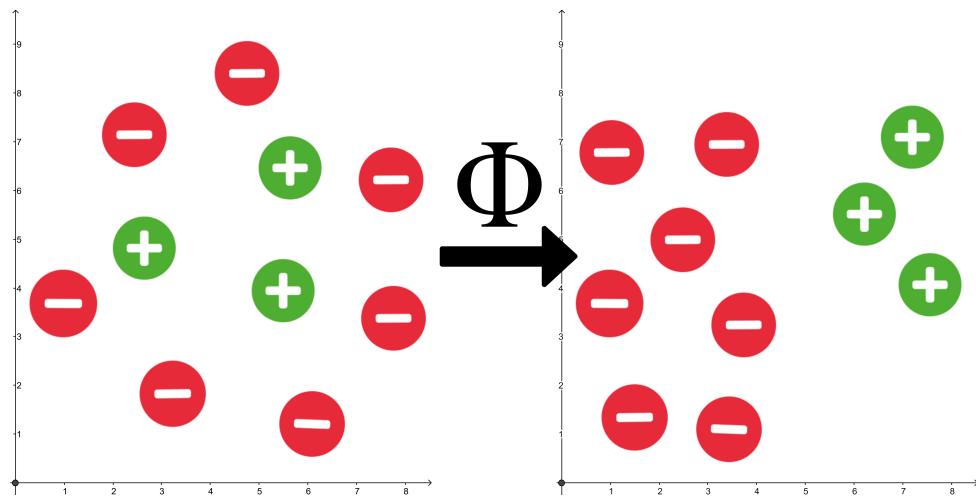


Abbildung 5.1: Die Eingabevektoren werden mittels einer Funktion $\Phi(x)$ transformiert. Die transformierten Vektoren sind linear trennbar.

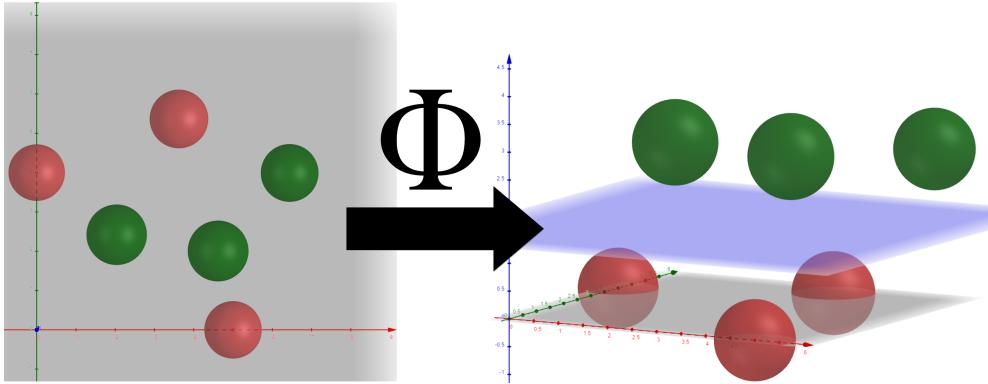


Abbildung 5.2: Die Eingabevektoren werden mittels einer Funktion $\Phi(x)$ in einen höherdimensionalen Raum transformiert. Die transformierten Vektoren sind durch eine Ebene linear trennbar.

Wird statt allen Eingabevektoren x deren transformierte Vektoren $\Phi(x)$ in Gleichung 3.4 eingesetzt erhält man:

$$\max_{\alpha} \quad \mathcal{L}(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^M y_n y_m \alpha_n \alpha_m \Phi(x_n)^T \Phi(x_m) \quad (5.1a)$$

$$\text{mit} \quad 0 \leq \alpha_n \leq C \text{ für } n = 1..N \quad (5.1b)$$

$$\sum_{n=1}^N \alpha_n y_n = 0 \text{ für } n = 1..N \quad (5.1c)$$

Wie in Gleichung 5.1 ersichtlich sind die einzigen Zusatzkosten gegenüber einer Soft-Margin SVM die Berechnungen der mitunter höherdimensionalen Skalarprodukte $\Phi(x)^T \Phi(x)$. Die Anzahl der Lagrangeefaktoren α und die Dimension der Q-Matrix aus Gleichung 5.16 bleibt gleich weil diese nur von der Anzahl der Eingabevektoren und nicht von der Dimension derer abhängig sind.

Die Stützvektoren dieser Methode befinden sich in dem neuen Raum \mathbb{R}^L , weil auch nur die transformierten Eingabevektoren betrachtet werden. Durch die, mitunter nichtlineare, Transformation Φ ist es so möglich mittels einer SVM nichtlineare Trenngrenzen durchführen zu können.

Das Problem dieses Ansatzes ist die Wahl der Transformationsfunktion $\Phi(x)$. Im Normalfall ist eine solche Funktion nicht bekannt. Trotzdem ist die Erkenntnis, dass die Dimension der Vektoren das Optimierungsproblem nicht groß beeinflusst und dass durch eine Transformation nichtlineare Trenngrenzen bestimmt werden können, äußerst relevant für die weitere Betrachtung. In Abschnitt 5.2 wird beschrieben wie die Zusatzkosten der Skalarprodukte auch umgangen werden können.

5.2 Kernel Trick

Wie in Abschnitt 5.1 beschrieben können die Eingabevektoren mittels einer Funktion Φ in einen anderen, beliebigen Raum transformiert werden. In diesem Kapitel wird diese Idee der Transformation verallgemeinert betrachtet.

Für die Betrachtung wird das Optimierungsproblem und die Formeln für w und b für die Soft-Margin SVM so abgeändert, dass statt allen Eingabevektoren x deren transformierte Form $z = \Phi(x)$ eingesetzt wird:

$$\max_{\alpha} \quad \mathcal{L}(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^M y_n y_m \alpha_n \alpha_m z_n^T z_m \quad (5.2a)$$

$$\text{mit} \quad 0 \leq \alpha_n \leq C \text{ für } n = 1..N \quad (5.2b)$$

$$\sum_{n=1}^N \alpha_n y_n = 0 \text{ für } n = 1..N \quad (5.2c)$$

Für die Berechnung von w und b ergibt sich:

$$w = \sum_{n=1}^N \alpha_n y_n z_n \quad (5.3)$$

$$b = \frac{1}{y_k} - w^T z_k \quad (5.4)$$

Wie bereits zuvor erwähnt hängt das Problem ausschließlich von den Skalarprodukten der Eingabevektoren, die jetzt in ihrer transformierten Form verwendet werden, ab. Werden die Eingabevektoren mittels Φ in einen sehr hochdimensionalen, mitunter unendlich dimensionalen, Raum transformiert kann die Berechnung des Skalarprodukts sehr aufwändig bis gar nicht berechenbar werden.

Durch die Einführung einer Kernel-Funktion $K(x, x') = z_1^T z_2 = \Phi(x)^T \Phi(x')$, die das Skalarprodukt der transformierten Eingabevektoren berechnet ohne die Eingabevektoren tatsächlich in den neuen Raum zu transformieren, kann das Problem der Berechnung von hochdimensionalen Skalarprodukten umgangen werden.

5.2.1 Beispiel Kernel-Funktion

Die Idee einer Kernel-Funktion wird nun anhand eines einfachen Beispiels gezeigt. Folgende Kernel-Funktion für $x, x' \in \mathbb{R}^2$ sei gegeben:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^2 = \\ &= (1 + x_1 x'_1 + x_2 x'_2)^2 = \\ &= 1 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1 x'_1 + 2x_2 x'_2 + 2x_1 x'_1 x_2 x'_2 \end{aligned} \quad (5.5)$$

Diese Kernel-Funktion scheint auf den ersten Blick nicht, wie zuvor definiert, einem Skalarprodukt der transformierten Vektoren $\Phi(x)$ und $\Phi(x')$ zu entsprechen.

Angenommen die verwendete Transformationsfunktion Φ entspräche:

$$\Phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2) \quad (5.6)$$

Angewandt auf die Vektoren x und x' :

$$\begin{aligned} \Phi(x) &= (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2) \\ \Phi(x') &= (1, x_1'^2, x_2'^2, \sqrt{2}x_1', \sqrt{2}x_2', \sqrt{2}x_1'x_2') \\ \Phi(x)^T \Phi(x') &= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2' \end{aligned} \quad (5.7)$$

Vergleicht man nun das Ergebnisse von Gleichung 5.5 und Gleichung 5.7 erkennt man, dass die Kernel-Funktion $(1 + x^T x')^2$ tatsächlich dem Skalarprodukt der mit Φ transformierten Vektoren x und x' entspricht.

Φ transformiert in diesem Beispiel 2-dimensionale Vektoren nach \mathbb{R}^6 . Mittels der Kernel-Funktion kann das Skalarprodukt der transformierten Vektoren durch $K(x, x') = (1 + x^T x')^2$ berechnet werden ohne dass die Vektoren x und x' tatsächlich in den neuen Raum transformiert werden müssen.

In dem Beispiel wurde die Berechnung von 6-dimensionalen Skalarprodukten durchgeführt durch die Berechnung mittels der Kernel-Funktion, die lediglich 2-dimensionale Skalarprodukte berechnen muss. Führt die Funktion Φ eine Transformation in eine viel höhere oder unendlich hohe Dimension aus ist der Vorteil einer solchen Kernel-Funktion sofort ersichtlich.

5.2.2 Polynomieller Kernel

Die in Unterabschnitt 5.2.1 verwendete Kernel-Funktion kann verallgemeinert werden. Seien die Eingabevektoren $x \in \mathbb{R}^d$ und die Transformationsfunktion $\Phi : \mathbb{R}^d \rightarrow \mathbb{Z}$ ein Polynom der Ordnung Q , dann kann eine Kernel-Funktion wie folgt definiert werden:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^Q = \\ &= (1 + x_1 x_1' + x_2 x_2' + \dots + x_d x_d')^Q \end{aligned} \quad (5.8)$$

Die in Gleichung 5.8 beschriebene Kernel-Funktion wird polynomieller Kernel genannt. Weil durch das Polynom eine Vielzahl an Faktoren entsteht kann zur Kompenstation dieser Faktoren die Kernel-Funktion wie folgt adaptiert werden:

$$K(x, x') = (ax^T x' + b)^Q \quad (5.9)$$

Mittels dieser Kernel-Funktion kann die Berechnung eines Skalarprodukts eines beliebig großen Polynoms des Grades Q durch die Berechnung eines Skalarprodukts

in \mathbb{R}^d und der Exponentiation mit Q durchgeführt werden ohne dass die Vektoren tatsächlich in den Polynomraum transformiert werden müssen.

5.2.3 Radial Basis Function Kernel

Eine weitere mögliche Kernel-Funktion ist der Radial Basis Function (RBF) Kernel:

$$K(x, x') = \exp \gamma \|x - x'\|^2 \quad (5.10)$$

Die zu diesem Kernel zugehörige Transformationsfunktion Φ bildet in einen unendlich dimensionalen Raum ab. Dies kann wie folgt gezeigt werden (für den einfachsten Fall):

$$\begin{aligned} K(x, x') &= \exp(-(x - x')^2) = \\ &= \exp(-x^2 + 2xx' - x'^2) = \\ &= \exp(-x^2) \exp(2xx') \exp(-x'^2) = \\ &= \exp(-x^2) \sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!} \exp(-x'^2) \end{aligned} \quad (5.11)$$

Durch die Taylorexpansion von $\exp(2xx')$ wird die Unendlichkeit dieses Raums sichtbar. Das Ergebnis von Gleichung 5.11 hat bereits die für ein Skalarprodukt benötigte Symmetrie von $\exp(-x^2) \cdot \exp(-x'^2)$ und $(x)^k \cdot (x')^k$. Die Anteile $\frac{2^k}{k!}$ können gleichmäßig auf x und x' aufgeteilt werden indem je die Wurzel der Anteile zu x und x' multipliziert werden. Damit wurde gezeigt dass der RBF Kernel das Skalarprodukt in einem unendlich dimensionalen Raum berechnet.

5.2.4 SVM Definition mit Kernel

Ein Input x kann mit einer SVM wie folgt klassifiziert werden:

$$y(x) = \text{sign}(w^T z + b) \quad (5.12)$$

Das Ziel der neuen Definition ist die Vermeidung von einzeln auftretenden transformierten Eingabevektoren z in den Formeln um das gesamte Problem mittels einer Kernel-Funktion $K(x, x')$ ausdrücken zu können. Dadurch muss die Transformation Φ nie direkt auf einen Vektor angewandt werden und lediglich die Skalarprodukte mittels der Kernel-Funktion berechnet werden.

$$w = \sum_{z_n \text{ ist SV}} \alpha_n y_n z_n \quad (5.13)$$

Setzt man Gleichung 5.13 in Gleichung 5.12 ein erhält man:

$$\begin{aligned} y(x) &= \text{sign}\left(\sum_{\alpha_n > 0} \alpha_n y_n z_n^T z + b\right) = \\ &= \text{sign}\left(\sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x) + b\right) \end{aligned} \quad (5.14)$$

Für b ergibt sich durch Einsetzen von Gleichung 5.13 für einen beliebigen Stützvektor x_k :

$$\begin{aligned} b &= \frac{1}{y_k} - w^T z_k = \\ &= \frac{1}{y_k} - \sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x_k) = \\ &= y_k - \sum_{\alpha_n > 0} \alpha_n y_n K(x_n, x_k) = \end{aligned} \quad (5.15)$$

Die SVM ist nun vollständig definiert durch die Verwendung der Kernel-Funktion ohne dass die Transformationsfunktion Φ bekannt sein muss. Es wird keine einzige tatsächliche Transformation eines Eingabevektors mittels Φ benötigt und es können beliebig dimensionale Räume durch die Verwendung von entsprechenden Kernel-Funktionen verwendet werden ohne je einen Vektor in diesen Raum transformieren zu müssen.

Da die Funktion Φ überhaupt nicht bekannt sein muss impliziert das, dass beliebige Kernel-Funktionen verwendet werden können. Das ist in der Tat möglich, allerdings müssen die Kernel-Funktionen bestimmte Eigenschaften erfüllen. Diese werden in Unterabschnitt 5.2.5 erläutert.

Das in Abschnitt 2.4 beschriebene Lösungsverfahren ist nach wie vor gültig, lediglich die Q -Matrix muss entsprechend angepasst werden. Es muss nur statt $x_n^T x_m$ $K(x_n, x_m)$ eingesetzt werden:

$$\min_{\alpha} \quad \mathcal{L}(\alpha) = \frac{1}{2} \alpha^T Q \alpha + (-1^T) \alpha \quad (5.16a)$$

$$\text{mit} \quad Q = \begin{bmatrix} y_1 y_1 K(x_1, x_1) & y_1 y_2 K(x_1, x_2) & \dots & y_1 y_N K(x_1, x_N) \\ y_2 y_1 K(x_2, x_1) & y_2 y_2 K(x_2, x_2) & \dots & y_2 y_N K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 K(x_N, x_1) & y_N y_2 K(x_N, x_2) & \dots & y_N y_N K(x_N, x_N) \end{bmatrix} \quad (5.16b)$$

$$\text{für} \quad y^T \alpha = 0 \quad (5.16c)$$

$$0 \leq \alpha \leq \infty \quad (5.16d)$$

Mit dieser Änderung kann das Problem wieder an ein Quadratic Programming Framework übergeben werden um die α Werte zu bestimmen. Die SVM lässt sich somit in Kombination mit Kernel-Funktionen auf beliebige binäre Klassifikationsprobleme anwenden.

5.2.5 Anforderungen an eine Kernel-Funktion

Wie in den Kapiteln zuvor gezeigt wurde gibt es eine Vielzahl von verschiedenen Kernel-Funktionen. Wenn eine neue Funktion als Kernel-Funktion verwendet werden

soll muss diese einem Skalarprodukt in einem Raum entsprechen. Um dies überprüfen zu können gibt es zwei verschiedene Ansätze:

- Für eine vermutlich richtige Kernel-Funktion wird konstruktiv versucht die zugehörige Transformationsfunktion Φ zu bestimmen. Diese Vorgehensweise wurde in Unterabschnitt 5.2.1 durchgeführt.
- Der Kernel $K(x, x')$ ist ein gültiger Kernel wenn die Funktion $K(x, x')$ symmetrisch ist 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 $x_1..x_N$. Die Forderung der positiv semi-definiten Matrix wird auch Satz von Mercer genannt und garantiert bei Erfüllung dass eine Funktion Φ existiert deren Skalarprodukte durch die Kernel-Funktion beschrieben werden können.

6 Generalisierungsmetrik

Um das Generalisierungsverhalten einer SVM für N Eingabevektoren beurteilen zu können kann folgendes Verhältnis betrachtet werden:

$$\mathbb{E} = \frac{\#\text{Stützvektoren}}{N - 1} \quad (6.1)$$

Je kleiner dieses Verhältnis ist, desto weniger Vektoren wurden für die Bestimmung der Trenngrenze verwendet. Ein großes \mathbb{E} hat mit hoher Wahrscheinlichkeit Overfitting zur Folge.

7 Multiclass Separation

<https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/>

multiclass separation summary text

7.1 One vs All

one vs all section

7.2 One vs One

one vs one section

Weiterführende Ressourcen

- Abu-Mostafa, Yaser S., Malik Magdon-Ismail und Hsuan-Tien Lin (2012). *Learning From Data*. AMLBook. 213 S. ISBN: 978-1-60049-006-4.
- Bishop, Christopher M. (23. Aug. 2016). *Pattern Recognition and Machine Learning*. Softcover reprint of the original 1st ed. 2006 Edition. New York, NY: Springer. 758 S. ISBN: 978-1-4939-3843-8.
- Boyd, Stephen und Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge, UK ; New York: Cambridge University Press. 727 S. ISBN: 978-0-521-83378-3. URL: <https://web.stanford.edu/~boyd/cvxbook/>.
- Burges, Christopher J.C. (1. Juni 1998). „A Tutorial on Support Vector Machines for Pattern Recognition“. In: *Data Mining and Knowledge Discovery* 2.2, S. 121–167. ISSN: 1573-756X. DOI: 10.1023/A:1009715923555. URL: <https://doi.org/10.1023/A:1009715923555> (besucht am 06.03.2021).
- Kecman, Vojislav (8. Juni 2001). *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. Reprint Edition. MIT Press. 576 S. ISBN: 978-0-262-52790-3.
- 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/>.
- Winston, Patrick (2010). „6.034 Artificial Intelligence“. Massachusetts Institute of Technology: MIT OpenCourseWare. URL: <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010>.