Support Vektor Maschinen & Sequential Minimal Optimization

Manuela Begic

Bachelorarbeit
an der Fakultät für Mathematik, Informatik und Statistik
Department Mathematik
Ludwig–Maximilians–Universität
München

vorgelegt von
Manuela Begic
Matrikelnummer: 11338597
Studiengang: B.Sc. Wirtschaftsmathematik
Betreuer: Dr. Dirk Andre Deckert

München, den 06.02.2022

Inhaltsverzeichnis

1 Einleitung **3**
1.1 Was ist maschinelles Lernen? . . . 3
1.2 Grundlegende Begriffe . . . 3

2 Support Vektor Maschinen **4**
2.1 Lineare Klassifikation . . . 4
2.2 Margen . . . 5
2.3 Trennbarer Fall . . . 7
2.3.1 Das primale Optimierungsproblem . . . 9
2.3.2 Das duale Optimierungsproblem . . . 11
2.4 Nicht-trennbarer Fall . . . 15
2.4.1 Das primale Optimierungsproblem . . . 16
2.4.2 Das duale Optimierungsproblem . . . 18

3 SMO-Algorithmus **20**
3.1 Geschichtlicher Hintergrund & Konvergenzgewährleistung . . . 20
3.2 Sequential Minimal-Optimization . . . 23
3.2.1 Lösen der zwei Lagrange-Multiplikatoren . . . 24
3.2.2 Heuristik für die Wahl der zu optimierenden Multiplikatoren 30
3.2.3 Aktualisierung von w . . . 32
3.2.4 Aktualisierung von b . . . 32
3.2.5 Aktualisierung von E . . . 32
3.3 Zusammenfassung . . . 33
3.3.1 Beispiele . . . 34

4 Anhang **36**
4.0.1 Anhang A . . . 36
4.0.2 Anhang B . . . 38

1 Einleitung

1.1 Was ist maschinelles Lernen?

Maschinelles Lernen kann im Allgemeinen als computergestützte Methode bezeichnet werden, die Erfahrung nutzt, um eine Leistung zu verbessern oder eine Vorhersage treffen zu können. Hierbei bezieht sich der Begriff Erfahrung auf die Information aus der Vergangenheit, die dem Lernenden zur Verfügung steht. In der Regel handelt es sich dabei um Information in Form von elektronischen Daten, die gesammelt und für die Analyse zur Verfügung gestellt werden.
Man definiert im maschinellen Lernen sogenannte Standardaufgaben zur Lösung praxisbezogener Probleme. Die *Klassifizierung* ist eine von den Standardaufgaben. Unter diesem Begriff versteht man das Problem ein Objekt an Hand bestimmter Merkmale oder Kriterien einer Kategorie bzw. Klasse zuzuordnen. Ein kanonisches Beispiel aus der Praxis stellt die Erkennung von Spam dar. Bei der Spam-Erkennung geht es um die automatische Klassifizierung von E-Mails in Spam oder Nicht-Spam. Wir nutzen dieses Beispiel, um die folgenden Standardbegriffe aus dem maschinellen Lernen zu definieren.

1.2 Grundlegende Begriffe

Wie bereits erwähnt, erläutern wir in diesem Abschnitt, anhand des Beispiels der Spam-Erkennung, grundlegende Begrifflichkeiten, die üblicherweise im maschinellen Lernen genutzt werden.

Elemente oder Instanzen von Daten, die für das Lernen oder für die Bewertung genutzt werden, werden als *Beispiele* bezeichnet. Offensichtlich sind diese Beispiele in der Spam-Erkennung die E-Mails. Beispiele lassen sich unterteilen in *Trainingsbeispiele* ,*Validierungsbeispiele* und die *Teststichprobe*. Dabei sind Trainingsbeispiele, diejenigen, die zum Trainieren eines Lernalgorithmus verwendet werden. Validierungsbeispiele sind zugeordnete Beispiele, die der Abstimmung der Parameter für den Lernalgorithmus dienen und die *Teststichsprobe* wird genutzt, um die Leistung des Lernalgorithmus zu bewerten. Beim Spam-Problem besteht die Teststichprobe aus einer Sammlung von E-Mails, für die der Lernalgorithmus auf Grundlage derer Eigenschaften, vorhersagen muss, ob es sich um Spam handelt oder nicht. Im Anschluss wird diese Vorhersage mit der richtigen Zuordnung der E-Mails verglichen, um die Leistung des Algorithmus zu messen.

Woran erkennen wir, ob es sich bei einer E-Mail um Spam handelt? Intuitiv achten wir zum Beispiel auf den Absender der E-Mail, die Länge der E-Mail oder auf bestimmte Schlüsselwörter im Text. Die Menge der Merkmale einer E-Mail fassen

wir als *Eigenschaften* zusammen, die oft als Vektor dargestellt werden. Bei der Spam-Erkennung können wir die E-Mails in zwei Klassen einteilen, nämlich Spam und Nicht-Spam. Werte oder Klassen, die den Beispielen zugeordnet werden, nennt man *Label*.

Die *Hypothesenmenge* bildet die Zuordnung der jeweiligen Eigenschaft auf die Menge der Labels ab. In der Spam-Erkennung könnte das eine Menge an Funktionen sein, die die E-Mail Eigenschaften auf $\{Spam, Nicht - Spam\}$ abbildet. Die Hypothesen können Funktionen sein, die die E-Mail-Eigenschaftsvektoren auf die reellen Zahlen abbilden, die als Punktwerte interpretiert werden können.

2 Support Vektor Maschinen

Ziel dieser Arbeit ist es das Optimierungsproblem zu den Support Vektor Maschinen herzuleiten und die Lösung des Optmierungsproblems mittels des Sequential-Minimum-Optimization Algorithmus, auch bekannt unter SMO, aufzuzeigen. Was die sogenannten Support-Vektoren darstellen und wie sie definiert sind, werden wir in einem späteren Abschnitt aufgreifen. Zunächst zeigen wir auf, wie man Praxisprobleme, wie oben genannte E-Mail-Spamerkennung, mathematisch abbilden kann. Wir beschränken uns dabei, wie im Beispiel der Spamerkennung, auf zwei Labels und führen dazu den Begriff der linearen Klassifikation ein.

Kapitel 2 stellt die Basis für den SMO - Algorithmus dar und wurde, wenn nicht anders gekennzeichnet, aus [Tal18] entnommen.

2.1 Lineare Klassifikation

Sei $n \in \mathbb{N}$. Betrachten wir eine Inputmenge $X \in \mathbb{R}^n$ und eine Outputmenge $Y \in \{-1, +1\}$. Sei $f : X \mapsto Y$ eine Funktion, die jedem Element der Inputmenge einen Wert der Outputmenge zuweist. Um alle Kombinationen an solchen möglichen Funktionen festzuhalten, definieren wir $\mathcal{H} = Y^X$ als unsere Hypothesenmenge.

Sei $S = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\} \in \{X \times Y\}^m$ für $m \in \mathbb{N}$ mit $y_i = f(x_i)$ für alle $i \in \{1, ..., m\} \coloneqq [m]$ definiert als die Menge aller Trainingsbeispiele und sei S unter einer möglicherweise unbekannten Verteilung $\mathcal{D}$ unabhängig und identisch verteilt. Es sei weiter ein Wahrscheinlichkeitsmaß $\mathbb{P}$ gegeben.
Gesucht ist nun ein binärer Klassifizierer $h \in \mathcal{H}$ mit einem möglichst geringen Generalisierungsfehler $\mathcal{R}_\mathcal{D}$:

$$\mathcal{R}_\mathcal{D} := \mathop{\mathbb{P}}_{x \sim \mathcal{D}}[h(x) \neq f(x)] \tag{2.1}$$

Offensichtlich ist die Wahl der Hypothesenmenge $\mathcal{H}$ hierfür entscheidend. Eine erfolgreiche Klassifizierung ergibt sich aus einer einfach gewählten Hypothesenmenge, die nicht zu viele, aber genügend komplexe Hypothesen enthält. Natürlich hängt diese Wahl auch mit den Testdaten und dem Konzept f, das man erkennen will, zusammen. Beispiele für einfache Hypothesenmengen stellen lineare Klassifizierer oder so genannte Hyperebenen dar.
Eine mögliche Parametrisierung solch einer Hyperebene ist gegeben durch

$$\mathcal{H} = \{x \mapsto \text{sgn}(w^T \cdot x + b) \quad | \quad w \in \mathbb{R}^n, b \in \mathbb{R}\}, \tag{2.2}$$

welche das Klassifizierungsproblem auf ein sogenanntes lineares Klassifizierungsproblem beschränkt. Die übliche Darstellung solcher Hyperebenen im $\mathbb{R}^n$ ist mit der Normalengleichung $w^T \cdot x + b = 0$ gegeben, wobei $w \in \mathbb{R}^n$, ein zu der Hyperebene othogonaler Vektor und $b \in \mathbb{R}$ ein Skalar ist.
Damit kennzeichnet eine Hypothese der Form $x \mapsto \text{sgn}(w^T \cdot x + b)$ alle Punkte, die auf einer Seite der Hyperbene liegen, positiv, und entsprechend alle Punkte auf der anderen Seite der Hyperbene, negativ.

2.2 Margen

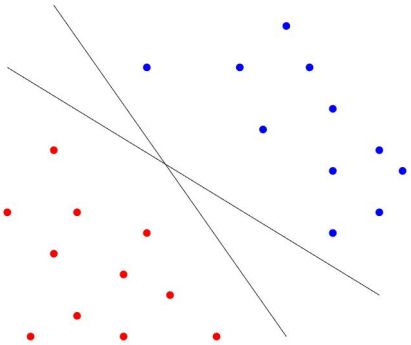

Abbildung 1: Beispiele für mögliche Hyperebenen

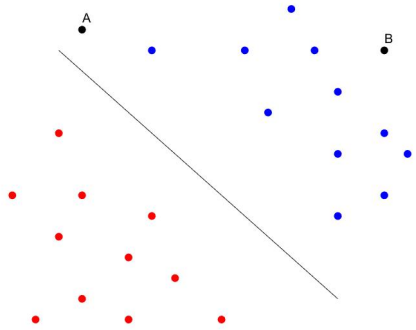

Abbildung 2: Warum wird bei gewählter Hyperbene B mit höherer Sicherheit als A klassifiziert?

Wie in Abbildung 1 zu erkennen ist, gibt es unendlich viele Hyperebenen, die die blaue und rote Klasse voneinander trennen. Dabei seien die blauen Punkte die Beispiele, die $f(x_i) = +1$ und die roten Punkte die Beispiele, die $f(x_i) = -1$ für $i \in \{1, ..., m\}$ erfüllen. Man bedenke, dass die Wahl der Hyperbene auf der Grundlage dessen erstellt wird, was unser Modell, basierend auf unseren Trainingsdaten,

lernen kann. Letztendlich möchten wir aber, dass das Modell auch für neue Daten oder zumindest für die Daten in unserem Testdatensatz gut funktioniert. Es macht intuitiv Sinn, dass wir Beispiele, die weit von der Hyperebene entfernt liegen, sicherer vorhersagen können als solche, die nahe an ihr liegen. In diesem Fall bezieht sich der Begriff „Sicherheit“ auf die Zuweisung der Beispiele und ist nicht im Sinne des Wahrscheinlichkeitsmaßes gemeint.
In Abbilung 2 sind zwei neue Beobachtungen mit A und B gekennzeichnet. Bei der gewählten Hyperebene würde man mit größerer Sicherheit B klassifizieren, als A. Die optimierte Hyperebene ändert sich geringfügig, wenn sie auf verschiedenen Teilmengen der Daten trainiert wird, sodass Beobachtungen, die nahe an der Hyperebene liegen, anfällig für Änderungen in der Klassifizierungsausgabe sind. Beobachtungen, die weit von der Hyperebene entfernt liegen, sind viel stabiler im Bezug auf die Vorhersage ihrer Klasse. Mit anderen Worten, wir möchten eine Hyperebene konstruieren, die mit den Daten konsistent ist, während wir uns so wenig wie möglich an die Trainingsbeispiele binden - Modelle, die zu sehr von den Trainingsbeispielen abhängen, sind anfällig für Overfitting. Wir möchten so viel Raum wie möglich zwischen der Hyperebene und den Punkten auf jeder Seite der Hyperbene haben, um das Gesamtvertrauen unserer Vorhersagen zu erhöhen.
In der Welt der Support Vektor Maschinen wird dieser Raum als Marge bezeichnet. Durch Maximierung der Marge können wir einen optimalen Klassifikator erstellen.

Die folgende Definition wurde aus [Ng] entnommen.

Definition 2.1 (**funktionelle Marge**). *Sei $X \in \mathbb{R}^n$ und $Y \in \{+1, -1\}$. Sei weiterhin $S = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\} \in \{X \times Y\}^m$ eine Trainingsmenge. Die funktionelle Marge eines linearen Klassifizierers $w^T x + b$ in einem Punkt $x_i \in X$ wird definiert als*

$$y_i \cdot (w^T x_i + b) \tag{2.3}$$

Man kann sich die funktionelle Marge als eine Funktion vorstellen, die jeden Punkt auf die Korrektheit der Klassifikation prüft. Gilt $y_i = \text{sgn}(w^T x_i + b)$, so ist der Punkt der richtigen Klasse zugeordnet und es lässt sich für die funktionelle Marge folgern:

- $y_i = 1 \Rightarrow w^T x_i + b > 0$
- $y_i = -1 \Rightarrow w^T x_i + b < 0$

Des Weiteren, kann man dem Wert der funktionellen Marge entnehmen, ob sich der Punkt mit hoher oder geringer Sicherheit zuordnen lässt:

- Für $y_i = 1$: Je größer $w^T x_i + b$, desto sicherer die Zuweisung

- Für $y_i = -1$: Je kleiner $w^T x_i + b$, desto sicherer die Zuweisung

Definition 2.2 (**geometrische Marge**). *Die geometrische Marge $p_h(x)$ eines linearen Klassifizierers $h : x \mapsto w^T x + b$ an einem Punkt x ist definiert als sein euklidischer Abstand zu der Hyperebene $w^T x + b = 0$.*

$$p_h(x) = \frac{|w^T x + b|}{||w||} \tag{2.4}$$

Die geometrische Marge p_h eines linearen Klassifizierers h für eine Trainingsmenge $S = (x_1, ..., x_m)$ ist das Minimum aller geometrischen Margen der Punkte in S.

$$p_h = \min_{i \in \{1,..,m\}} p_h(x_i) \tag{2.5}$$

Die Marge entspricht damit dem Abstand der Hyperebene zu den nähesten Punkten.

Mit dieser Definition können wir nun das Optimierungsproblem zu den Support Vektor Maschinen aufstellen, denn die Lösung des SVM ist eine trennende Hyperebene mit maximaler geometrischer Marge.

2.3 Trennbarer Fall

Für dieses Kapitel legen wir die Annahme fest, es existiere eine Hyperebene, die unsere Trainingsmenge S linear in die zwei Klassen $+1$ und -1 trennt, d.h,

$$\exists (w, b) \in \mathbb{R}^n \backslash \{0\} \times \mathbb{R}, \tag{2.6}$$

$$\text{so dass} \quad \forall i \in \{1, ..., m\} : y_i(w^T x + b) \geq 0. \tag{2.7}$$

Man beachte, dass die Bedingung (2.7) die Eigenschaften der funktionellen Marge für richtig klassifizierte Punkte aufgreift.
Nach Definition 2.1 und 2.2 ist die maximale geometrische Marge p der Hyperebene h gegeben durch

$$p = \max_{w,b} \quad p_h = \max_{\substack{w,b: \\ y_i(w^T x_i + b) \geq 0}} \quad \min_{i \in [m]} \quad p_h(x_i)$$

$$= \max_{\substack{w,b: \\ y_i(w^T x_i + b) \geq 0}} \quad \min_{i \in [m]} \frac{|w^T x_i + b|}{||w||} = \max_{w,b} \quad \min_{i \in [m]} \frac{y_i(w^T x_i + b)}{||w||}$$

$$= \max_{\substack{w,b: \\ \min_{i \in [m]} y_i(w^T x_i + b) = 1}} \frac{1}{||w||} = \max_{\substack{w,b: \\ \forall i \in [m] : y_i(w^T x_i + b) \geq 1}} \frac{1}{||w||}$$

Die zweite Zeile ergibt sich aus der Voraussetzung, dass die Daten trennbar sind. Denn, für ein Paar (w, b) gilt in diesem Fall $y_i(w^T x_i + b) \geq 0 \quad \forall i \in \{1, ..., m\}$. Mit der Beobachtung, dass der rechte Ausdruck in der zweiten Zeile invariant unter der Multiplikation von (w, b) mit einem positiven Skalar ist, können wir uns auf Paare (w, b) mit der Eigenschaft $\min_{i \in [m]} y_i(w^T x_i + b) = 1$ beschränken.

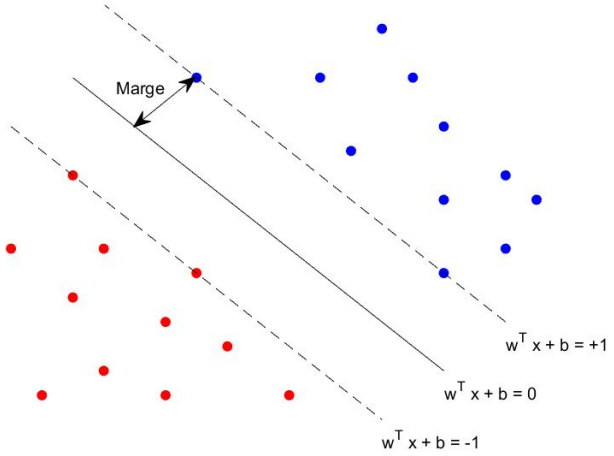

Abbildung 3: Maximale geometrische Marge der gegebenen Daten

Neben der separierenden Hyperbene $w^T x + b = 0$ mit maximaler geometrischer Marge, sind in in Abbildung 3 parallel zwei Hyperebenen $w^T x + b = \pm 1$ eingezeichnet, die man als Randhyperebenen bezeichnet. Die Definition dieser Hyperebenen beruht auf der Tatsache, dass die Punkte, die am nähesten an der Hyperebene liegen $|w^T x + b| = 1$ erfüllen und den sogenannten Rand bzw. die Marge definieren.

Unser primäres Ziel besteht darin, den Abstand zwischen den Punkte, die auf den Randhyperebenen liegen, zu erhöhen und damit die Hyperbene mit maximaler geometrischer Marge zu definieren. Dazu geben wir im folgenden Kapitel das Optimierungsproblem, das zur Lösungsfindung genutzt wird und Eigenschaften, die wir im Laufe der Arbeit nutzen werden, an.

2.3.1 Das primale Optimierungsproblem

Da die Maximierung von $\frac{1}{||w||}$, also die Maximierung der geometrischen Marge, in Äquivalenz zur Minimierung von $\frac{1}{2}||w||^2$ steht, wird folgendes konvexes Optimierungsproblem konventionellerweise zur Lösungsfindung genutzt.

$$\min_{w,b} \frac{1}{2}||w||^2 \tag{2.8}$$

$$\text{so dass} \quad y_i(w^T x_i + b) \geq 1 \quad \forall i \in \{1, ..., m\}$$

- Die Funktion $f : w \mapsto \frac{1}{2}||w||^2$ ist unendlich oft in w differenzierbar.
- Die Hesse-Matrix $\nabla^2 f(w) = I$ der Funktion f, wobei mit I die Identität gemeint ist, ist strikt positiv definit und f damit eine konvexe Funktion.
- Die Nebenbedinungen, gegeben mit $g_i : (w, b) \mapsto 1 - y_i(w^T x_i + b)$, sind affine Funktionen.

Das Programm (2.8) definiert ein konvex quadratisches Optimierungsproblem(KQP) mit affin-linearen Nebenbedingungen. Aufgrund der Konvexität, ist die Lösung von (2.8) eindeutig und erfüllt die Karush-Kuhn-Tucker Bedingungen, wie in [Tal18] Appendix B.3 nachzulesen ist.

Theorem 2.3 (**KKT-Bedingungen**). *Es sei ein konvexes Programm der Form*

$$\min_{x \in X} f(x)$$
$$\textit{so dass} \quad g_i(x) \leq 0 \quad i = 1, .., k$$

wobei $f : X \mapsto \mathbb{R}$ konvex, $g_i : X \mapsto \mathbb{R}$ affin und beide Funktionen stetig differenzierbar sind. Ein zulässiger[1] Punkt x^* des konvexen OPs erfüllt die KKT-Bedingungen genau dann wenn ein $\alpha \in \mathbb{R}^k$ existiert, so dass

$$\begin{aligned} \nabla_{x^*} \mathcal{L}(x^*, \alpha) = \nabla f(x^*) + \sum_{i=1}^{k} \alpha_i \nabla g(x^*) = 0, \\ g_i(x^*) \leq 0, \\ \alpha_i \geq 0, \\ \alpha_i g_i(x^*) = 0, \\ \forall i \in \{1, ..., k\}. \end{aligned}$$

[1] Ein Punkt $x \in X$ heißt zulässig für obiges Programm, genau dann wenn x alle Nebenbedingungen $g_i(x)$ für $i = \{1, ..., k\}$ erfüllt.

wobei $\mathcal{L} = \mathcal{L}(x^*, \alpha) = f(x^*) + \sum_{i=1}^{k} \alpha_i g(x^*)$ die Langrange-Funktion bezeichnet. Ein Punkt (x^*, α), der die obigen Bedingungen erfüllt, heißt dann KKT-Punkt und ist optimal für das gegebene Optimierungsproblem.
Man bezeichnet α als Lagrange-Multiplikator oder duale Variable.

Die Definition der dualen Variable α verweist auf das duale Optimierungsproblem, welches wir später noch aufgreifen werden.

Theorem 2.4. *Ein zulässiger Punkt (w, b) ist optimal für das Optimierungsproblem (2.8) genau dann wenn $w \in \mathbb{R}^n, b \in \mathbb{R}$ und $\alpha = (\alpha_1, ..., \alpha_m) \in \mathbb{R}^m$ mit $\alpha \geq 0$ existieren, so dass*

$$\begin{aligned} \sum_{i=1}^{m} \alpha_i y_i x_i &= w, \\ \sum_{i=1}^{m} \alpha_i y_i &= 0, \\ \alpha_i(y_i(w^T x_i + b) + 1) &= 0 \end{aligned}$$

für alle $i \in \{1, ..., m\}$ erfüllt ist.

Beweis. Wir definieren $f(w) = \frac{1}{2}||w||^2$ und $g_i(w, b) = -y_i(w^T x_i + b) + 1 \leq 0$ für alle $i \in \{1, ..., m\}$, dann ist für $w \in \mathbb{R}^n$,$b \in \mathbb{R}$ und $\alpha \in \mathbb{R}^m$ die Lagrange-Funktion gegeben mit

$$\begin{aligned} \mathcal{L}(w, b, \alpha) &= \frac{1}{2}||w||^2 + \sum_{i=1}^{m} \alpha_i(-y_i(w^T x_i + b) + 1) \\ &= \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i(y_i(w^T x_i + b) - 1) \end{aligned}$$

und damit erfüllt ein zulässiger Punkt (w, b) die KKT-Bedinugungen, genau dann wenn für alle $i \in \{1, ...m\}$:

$$\nabla_w \mathcal{L}(w, b, \alpha) = 0 \quad \Leftrightarrow \quad \nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \tag{2.9}$$

$$\nabla_b \mathcal{L}(w, b, \alpha) = 0 \quad \Leftrightarrow \quad \nabla_b \mathcal{L}(w, b, \alpha) = -\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.10}$$

$$g_i(w, b) \leq 0 \quad \Leftrightarrow \quad -y_i(w^T x_i + b) + 1 \leq 0 \tag{2.11}$$

$$\alpha_i g_i(w, b) = 0 \quad \Leftrightarrow \quad -\alpha_i(y_i(w^T x_i + b) + 1) = 0 \tag{2.12}$$

$$\alpha_i \geq 0 \tag{2.13}$$

Da ein zulässiger Punkt (2.11) erfüllt und $\alpha \geq 0$ vorausgesetzt wurde, folgt die Behauptung aus (2.9), (2.10) und (2.12).

□

Wir stellen also die Beobachtung fest, dass $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ und damit eine Linearkombination der $x_1, ..., x_m$ darstellt. Außerdem muss entweder $\alpha_i = 0$ oder $y_i(w^T x_i + b) = 1$ gelten, damit die Bedingung (2.12) erfüllt wird. Das bedeutet, ein Vektor x_i beeinflusst die Lösung w nur dann, falls $\alpha_i \neq 0$. Solche Vektoren werden Support Vektoren genannt, denn sie liegen mit der Eigenschaft $y_i(w^T x_i + b) = 1$ auf den Randhyperbenen, die durch $w^T x_i + b = \pm 1$ parametrisiert sind.
Vektoren, die nicht auf den Randhyperebenen liegen, haben somit keine Auswirkungen auf die Lösung des SVM, und damit auf die gesuchte Hyperbene.

2.3.2 Das duale Optimierungsproblem

In diesem Abschnitt nutzen wir die Definition der *Lagrange-Dualität*, um ein zu (2.8) duales Optimierungsproblem aufzusetzen, welches in Abhängigkeit der Lagrange-Multiplikatoren definiert ist.

Die folgenden Definitionen wurden aus [Tal18] Appendix B.2 entnommen.

Definition 2.5 (**Lagrange-Dualität**). *Es sei ein Optimierungsproblem der Form*

$$\begin{aligned} & \min_{x \in X} f(x) \\ \textit{so dass} \quad & g_i(x) \leq 0 \quad i = 1, .., k \end{aligned}$$

mit $X \in \mathbb{R}^n$ und $f, g_i : X \mapsto \mathbb{R} \quad \forall i \in \{1, ..., k\}$ gegeben. Es bezeichne p^* die optimale Lösung des angegeben Problems.
*Die zugehörige **Lagrange-Funktion** des obigen Optimierungsproblems ist definiert wie folgt.*

$$\forall x \in X, \forall \alpha \geq 0 \quad \mathcal{L} = \mathcal{L}(x, \alpha) = f(x) + \sum_{i=1}^{k} \alpha_i g_i(x)$$

Die Funktion

$$q(\alpha) = \inf_{x \in X} \mathcal{L}(x, \alpha) \quad \forall \alpha \geq 0$$

*heißt **duale Funktion** zu dem obigen Optimierungsproblem und ist immer konkav, da die Lagrange-Funktion in Abhängigkeit von α linear ist und über das Infimum*

Konkavität bewahrt wird. Das ***duale Optimierungsproblem*** *ist dann mit*

$$\max_{\alpha} q(\alpha)$$
$$so\ dass\ \ \alpha \geq 0$$

gegeben und ist als Maximierung eines konkaven Problems immer ein konvexes Optimierungsproblem. Es bezeichne d^* die optimale Lösung des dualen OPs. Für $x \in X$, die die KKT-Bedingungen erfüllen, gilt $f(x)+\sum_{i=1}^{k} \alpha_i g_i(x) \leq f(x)$, woraus

$$\forall \alpha \geq 0 \quad q(\alpha) \leq p^*$$

folgt. Die Ungleichung $d^* \leq p^*$, bekannt als ***schwache Dualität****, ist immer erfüllt. Die Gleichung $d^* = p^*$, die man als* ***starke Dualität*** *bezeichnet, ist im Allgemeinen nicht gültig. Die starke Dualität gilt jedoch, wenn für obiges Optimierungsproblem $f : X \mapsto \mathbb{R}$ konvex und $g_i : X \mapsto \mathbb{R}$ für alle $i \in \{1, ..., k\}$ affin ist.*

Die zugehörige Theorie, sowie die Beweise zu den aufgeführten Behauptungen in der Definition der Lagrange-Dualität sind zum Beispiel in [Sun06] und [Jun15] nachzulesen.

Bei der dualen Form solcher Optimierungsprobleme, werden damit die Nebenbedingungen aus dem primalen Optimierungsproblem entfernt und dafür mit Multiplikatoren in die Zielfunktion des dualen Optimierungsproblems eingesetzt, um den optimalen Zielwert der Funktion in Abhängigkeit des Lagrange-Multiplikators zu erhalten.

Theorem 2.6. *Die duale Form des Optimierungsproblems in (2.8) ist gegeben mit*

$$\max_{\alpha}\Big(\sum_{i=1}^{m} \alpha_i - \frac{1}{2}||\sum_{i=1}^{m} \alpha_i y_i x_i||^2\Big) \tag{2.14}$$

$$so\ dass\ \ \alpha_i \geq 0$$
$$und\ \ \sum_{i=1}^{m} \alpha_i y_i = 0$$
$$für\ \ \forall i \in \{1, ..., m\}.$$

Beweis. Mit

$$f(w) + \sum_{i=1}^{m} \alpha_i g_i(w, b) \leq f(w)$$

folgt

$$\frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i (y_i(w^T x_i + b) - 1) \leq \frac{1}{2}||w||^2$$

für alle (w, b), die die KKT-Bedingungen erfüllen. Sei p^* die optimale Lösung des primalen OPs. Dann gilt

$$q(\alpha) := \inf_{w,b}(\frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i (y_i(w^T x_i + b) - 1)) \leq p^*$$

Durch Einsetzen von $\nabla_w \mathcal{L}(w, b, \alpha) = 0 \Leftrightarrow w = \sum_{i=1}^{m} \alpha_i y_i x_i$ erhalten wir

$$\inf_{w,b} \mathcal{L}(w, b, \alpha) = -\frac{1}{2}||\sum_{i=1}^{m} \alpha_i y_i x_i||^2 - b\sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i$$

Mit $\nabla_b \mathcal{L}(w, b, \alpha) = 0 \Leftrightarrow \sum_{i=1}^{m} \alpha_i y_i = 0$ erhalten wir

$$\inf_{w,b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}||\sum_{i=1}^{m} \alpha_i y_i x_i||^2$$

und das duale Optimierungsproblem zu (2.8)

$$\begin{aligned} &\max_{\alpha} \Big(\sum_{i=1}^{m} \alpha_i - \frac{1}{2}||\sum_{i=1}^{m} \alpha_i y_i x_i||^2 \Big) \\ &\text{so dass} \quad \alpha_i \geq 0 \\ &\text{und} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \\ &\forall i \in \{1, ..., m\} \end{aligned}$$

□

Sei $\max_\alpha q(\alpha) := d^*$ die optimale Lösung des dualen Optimierungsproblems. Mit den gegebenen Eigenschaften von f und g_i für $i \in \{1, ..., m\}$ gilt $d^* = p^*$. Außerdem

- ist die Funktion $q : \alpha \mapsto \sum_{i=1}^{m} \alpha_i - \frac{1}{2}||\sum_{i=1}^{m} \alpha_i y_i x_i||^2$ unendlich oft in α differenzierbar.
- ist die Hesse-Matrix $\nabla^2 q = (y_i x_i \cdot y_j x_j)_{ij}$ positiv semidefinit und q damit konkav.
- sind die Nebenbedinungen $l_i(\alpha) =: \alpha_i \quad \forall i \in \{1, ..., m\}$ und $v(\alpha) =: \sum_{i=1}^{m} \alpha_i y_i$ affin und konvex.

Da die die Zielfunktion q quadratisch ist, handelt es sich bei dem dualen Optimierungsproblem auch wieder um ein konvex quadratisches Optimierungsproblem. Da f konvex und die Nebenbedingungen g_i affin sind folgt, dass die Lösung des primalen und dualen Systems identisch sind. D.h. die gesuchte Hypereben kann mit der Lösung des dualen Systems und $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ in der Form

$$h(x) = sgn(w^T x + b) = sgn(\sum_{i=1}^{m} \alpha_i y_i x_i^T x + b)$$

dargestellt werden.
Des Weiteren können wir b mittels der Support Vektoren ausdrücken. Wir wissen bereits, dass ausschließlich Support Vektoren einen Einfluss auf die Lösung haben. Sie liegen, wie wir bereits festgestellt haben, auf den Randhyperebenen und erfüllen $y_i(w^T x_i + b) = 1$. Das bedeutet, für jeden Support Vektor x_i gilt $w^T x_i + b = y_i$, da $y_i \in \{+1, -1\}$ und damit lässt sich b wie folgt ermitteln

$$b = y_i - w^T x_i = y_i - \sum_{j=1}^{m} \alpha_j y_j x_j^T x_i \tag{2.15}$$

Erinnern wir uns daran zurück, wie die maximale geometrische Marge p der gesuchten Hyperebene definiert war, können wir diese mit (2.15) nun auch in Abhängigkeit von α ausdrücken:

$$\begin{aligned}
b &= y_i - \sum_{j=1}^{m} \alpha_j y_j x_j^T x_i \\
\Leftrightarrow \quad \sum_{i=1}^{m} \alpha_i y_i b &= \sum_{i=1}^{m} \alpha_i y_i^2 - \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\Leftrightarrow \quad b \underbrace{\sum_{i=1}^{m} \alpha_i y_i}_{=0} &= \sum_{i=1}^{m} \alpha_i - ||w||_2^2 \\
\Leftrightarrow \quad ||w||_2^2 &= \sum_{i=1}^{m} \alpha_i = ||\alpha||_1
\end{aligned}$$

Da die maximale geometrische Marge

$$p = \frac{1}{||w||_2} \quad \Leftrightarrow \quad p^2 = \frac{1}{||w||_2^2}$$

folgt

$$p^2 = \frac{1}{||\alpha||_1}.$$

Wie das Optimierungsproblem aus Theorem 2.6 rechnerisch gelöst werden kann, zeigen wir im letzten Kapitel mithilfe des SMO-Algorithmus. Bevor wir uns mit dem Algorithmus befassen, analysieren wir in Verbindung mit den bisherigen Resultaten noch den Fall einer nicht-trennbaren Trainingsmenge.

2.4 Nicht-trennbarer Fall

In der Praxis liegt meistens der Fall vor, dass sich die gegebenen Trainingsdaten nicht linear trennen lassen. Das impliziert für jede Hyperebene $w^T x + b = 0$, dass ein $(x_i, y_i) \in S$ existiert, so dass

$$y_i(w^T x_i + b) \ngeq 1 \tag{2.16}$$

d.h. die Nebenbedinungen in (2.8) aus dem trennbaren Fall, können nicht für alle Paare (x_i, y_i) für $i \in \{1, ..., m\}$ erfüllt werden. Um eine Hyperebene definieren zu können, müssen wir demnach Missklassifikationen und Verletzungen der Marge zulassen. Der einfachheithalber nennen wir diese Gruppe an Beispielen *Verletzer*. Wir suchen also für dieses Model eine Hyperebene mit maximaler Marge und minimaler Anzahl an Verletzern. Dafür führen wir die sogenannten Schlupfvariablen ein, die als Pramater dienen, um den Abstand jeden Paares (x_i, y_i) und seiner zugehörigen Klasse zu messen.

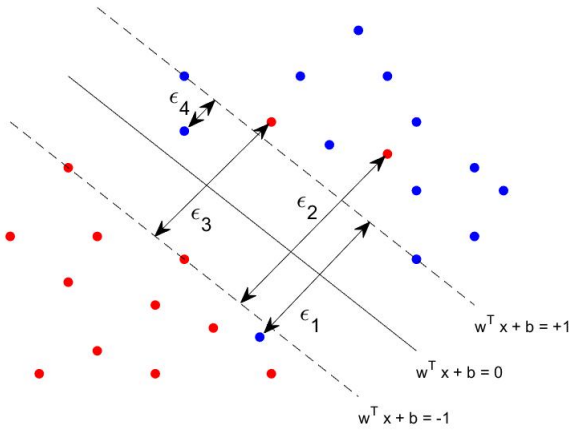

Abbildung 4: Abstände falsch klassifizierter Proben

Abbildung 4 veranschaulicht diese Situation. Die Schlupfvariablen ε_i für $i = \{1, .., 4\}$ kennzeichnen die Abstände der falsch klassifizierten Proben. Die Abstände der restlichen Proben sind Null, da sie bereits in ihrer richtigen Klasse liegen.

Man erkennt sofort, dass es sich bei den Punkten x_1, x_2 und x_3 um Missklassifikationen handelt. Bei x_4 hingegen ist die Zuordnung korrekt, allerdings ist die Marge kleiner als 1.

Mit Hilfe der Schlupfvariablen können wir nun eine abgeschwächte Version der Nebenbedingungen aus (2.8) definieren.

$$\forall i \in \{1, ..., m\} \quad \exists \varepsilon_i \geq 0 : y_i(w^T x_i + b) \geq 1 - \varepsilon_i \tag{2.17}$$

Mit dieser Formalisierung der Nebenbedingung lassen wir im Allgemeinen Verletzungen der Marge und Missklassifikationen zu. Die Marge, die aus der trennenden Hyperbene resultiert, wird in diesem Fall als weiche Marge bezeichnet. Im trennbaren Fall spricht man dann von einer harten Marge.

2.4.1 Das primale Optimierungsproblem

Wie bereits erwähnt, suchen wir eine Hyperbene mit maximaler geometrischer Marge und minimaler Anzahl an Verletzern. Letztere können durch $\sum_{i=1}^{m} \varepsilon_i$ oder allgemeiner durch $\sum_{i=1}^{m} \varepsilon_i^p$ für ein $p \geq 1$ gemessen werden. Sei $C \geq 0$ ein Parameter, der eine Gewichtung zwischen der geometrischen Marge und den Verletzern herstellt, dann ist das primale Optimierungsproblem gegeben durch

$$\min_{w,b,\varepsilon} \left(\frac{1}{2} ||w||^2 + C \sum_{i=1}^{m} \varepsilon_i^p \right), \tag{2.18}$$

$$\begin{aligned} \text{so dass} \quad & y_i(w^T x_i + b) \geq 1 - \varepsilon_i, \\ & \varepsilon_i \geq 0, \\ & \forall i \in \{1, ..., m\}, \end{aligned}$$

wobei $\varepsilon = (\varepsilon_1, ..., \varepsilon_m)^T$. Abhängig von der Wahl des Parameters C, ergeben sich unterschiedliche Prioritäten bei der Suche einer Hyperbene. Für große C erhält man weniger Verletzer, aber eine geringere Marge. Analog für kleine C, eine große Marge, aber eine höhere Anzahl an Verletzern. Typischerweise wird C mit Hilfe der n-fache Kreuzvalidierung bestimmt.

Analog zum trennbaren Fall, definiert das primale Optimierungsproblem (2.18) ein konvexes Optimierungsproblem, da die Nebenbedingungen affin und konvex sind und die Zielfunktion ebenfalls konvex ist. Insbesondere ist $\varepsilon \mapsto \sum_{i=1}^{m} \varepsilon_i^p$ auf Grund der Konvexität der p-Norm $||\cdot||_p$ auch konvex. Des Weiteren sind sowohl die

Zielfunktion als auch die Nebenbedingungen unendlich oft differenzierbar. Damit existiert wieder genau eine Lösung, die die KKT-Bedinungen erfüllt.

Theorem 2.7. *Ein zulässiger Punkt ist optimal für das Optimierungsproblem (2.18) mit $p = 1$, genau dann wenn $w \in \mathbb{R}^n, b \in \mathbb{R}, \alpha, \beta, \varepsilon \in \mathbb{R}^m_{0,+}$ und $C = \alpha + \beta$ existieren, so dass*

$$\sum_{i=1}^{m} \alpha_i y_i x_i = w,$$
$$\sum_{i=1}^{m} \alpha_i y_i = 0,$$
$$\forall i \in \{1, ..., m\} : -\alpha(y_i(w^T x_i + b) + 1 - \varepsilon_i) = 0.$$

Beweis. Wir definieren $f(w, \varepsilon) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \varepsilon_i^p$, $g_i(w, b, \varepsilon) = -y_i(w^T x_i + b) + 1 - \varepsilon_i \leq 0$ und $h_i(\varepsilon) = -\varepsilon_i \leq 0$ für $i \in \{1, ..., m\}$. Wir setzen $p = 1$ fest. Für $w \in \mathbb{R}^n$,$b \in \mathbb{R}$ und $\varepsilon, \alpha, \beta \in \mathbb{R}^m$ ist die Langrange Funktion definiert wie folgt.

$$\mathcal{L}(w, b, \varepsilon, \alpha, \beta) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \varepsilon_i - \sum_{i=1}^{m} \alpha_i(y_i(w^T x_i + b) - 1 + \varepsilon_i) - \sum_{i=1}^{m} \beta_i \varepsilon_i \tag{2.19}$$

Die Lösung des primalen Optimierungsproblems erfüllt die KKT-Bedingungen genau dann wenn für $\forall i \in \{1, ..., m\}$

$$\nabla_w \mathcal{L}(w, b, \varepsilon, \alpha, \beta) = 0 \quad \Leftrightarrow \quad w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \tag{2.20}$$
$$\nabla_b \mathcal{L}(w, b, \varepsilon, \alpha, \beta) = 0 \quad \Leftrightarrow \quad -\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.21}$$
$$\nabla_\varepsilon \mathcal{L}(w, b, \varepsilon, \alpha, \beta) = 0 \quad \Leftrightarrow \quad C - \alpha_i - \beta_i = 0 \tag{2.22}$$
$$g_i(w, b, \varepsilon) \leq 0 \quad \Leftrightarrow \quad -y_i(w^T x_i + b) + 1 - \varepsilon_i \leq 0 \tag{2.23}$$
$$\alpha_i g_i(w, b, \varepsilon) = 0 \quad \Leftrightarrow \quad -\alpha_i(y_i(w^T x_i + b) + 1 - \varepsilon_i) = 0 \tag{2.24}$$
$$h_i(\varepsilon) \leq 0 \quad \Leftrightarrow \quad -\varepsilon_i \leq 0 \tag{2.25}$$
$$\beta_i h_i(\varepsilon) = 0 \quad \Leftrightarrow \quad -\beta_i \varepsilon_i = 0 \tag{2.26}$$
$$\alpha_i \geq 0 \tag{2.27}$$
$$\beta_i \geq 0 \tag{2.28}$$

□

Analog zum trennbaren Fall, ergibt sich aus der Gleichung (2.20), dass w eine Linearkombination der Vektoren $x_1, ..., x_m$ ist. Für $\alpha_i \neq 0$ ist x_i ein Support-Vektor.

Im nicht-trennbaren Fall unterscheidet man zwei Arten von Support-Vektoren. Für $\alpha_i \neq 0$ folgt aus (2.24), dass $y_i(w^T x_i + b) = 1 - \varepsilon_i$. Falls $\varepsilon_i = 0$, liegt x_i als Support-Vektor auf einer der Randhyperebenen, denn dann gilt $y_i(w^T x_i + b) = 1$. Andernfalls ist $\varepsilon_i \neq 0$ und x_i eine Missklassifikation. In diesem Fall impliziert (2.26) $\beta_i = 0$ und (2.22) $\alpha_i = C$. Damit können wir aus den KKT-Bedingungen folgern, dass ein optimaler Wert für das QP gegeben ist wenn für $\forall i \in \{1, ..., m\}$

$$\begin{aligned} \alpha_i = 0 &\Leftrightarrow y_i(w^T x_i + b) > 1 \\ 0 < \alpha_i < C &\Leftrightarrow y_i(w^T x_i + b) = 1 \\ \alpha_i = C &\Leftrightarrow y_i(w^T x_i + b) < 1. \end{aligned} \tag{2.29}$$

2.4.2 Das duale Optimierungsproblem

Theorem 2.8. *Das duale Optimierungsproblem zu (2.18) ist*

$$\max_{\alpha}\Big(\sum_{i=1}^{m}\alpha_i - \frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2\Big), \tag{2.30}$$

$$\begin{aligned} \textit{so dass} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \\ & \forall i \in \{1, ...m\}. \end{aligned}$$

Beweis. Analog zu dem trennbaren Fall setzen wir die KKT-Bedingungen in (2.19) ein und erhalten

$$\begin{aligned} \inf_{w,b,\epsilon} \mathcal{L}(w, b, \alpha) &= \frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2 - \sum_{i=1}^{m}\alpha_i(y_i(\sum_{j=1}^{m}\alpha_j y_j x_j)x_i + b) - 1 + \varepsilon_i) \\ &\quad - \underbrace{\sum_{i=1}^{m}\beta_i\varepsilon_i}_{\underset{(2.26)}{=0}} + C\sum_{i=1}^{m}\varepsilon_i \\ &= \frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2 - ||\sum_{i=1}^{m}\alpha_i y_i x_i||^2 - b\underbrace{\sum_{i=1}^{m}\alpha_i y_i}_{\underset{(2.21)}{=0}} + \sum_{i=1}^{m}\alpha_i \\ &\quad - \sum_{i=1}^{m}\alpha_i\varepsilon_i + C\sum_{i=1}^{m}\varepsilon_i \end{aligned}$$

$$= \quad -\frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2 + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i\varepsilon_i + C\sum_{i=1}^{m}\varepsilon_i$$

Aus der Bedingung (2.22) erhalten wir $C = \alpha_i + \beta_i$.

$$\begin{aligned}\inf_{w,b,\varepsilon}\mathcal{L} &= -\frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2 + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i\varepsilon_i + (\alpha_i+\beta_i)\sum_{i=1}^{m}\varepsilon_i \\ &= -\frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2 + \sum_{i=1}^{m}\alpha_i \underbrace{-\sum_{i=1}^{m}\alpha_i\varepsilon_i + \sum_{i=1}^{m}\alpha_i\varepsilon_i}_{=\,0} + \underbrace{\sum_{i=1}^{m}\beta_i\varepsilon_i}_{=\,0}\end{aligned}$$

Damit ist

$$\inf_{w,b,\varepsilon}\mathcal{L} = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2$$

und das duale Optimierungsproblem zu (2.18) gegeben mit

$$\max_{\alpha}\Big(\sum_{i=1}^{m}\alpha_i - \frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2\Big),$$

$$\begin{aligned}\text{so dass} \quad &\sum_{i=1}^{m}\alpha_i y_i = 0, \\ &0 \le \alpha_i \le C, \\ &\forall i \in \{1, ...m\}.\end{aligned}$$

□

Das duale Optimierungsproblem unterscheidet sich nur durch die Nebenbedingung $0 \le \alpha_i \le C$, die sich aus $\beta_i \ge 0$ und $C = \alpha_i + \beta_i$ ergibt, von dem Optimierungsproblem im trennbaren Fall.
Auch in diesem Fall können wir die Hyperebene mit α wie folgt ausdrücken.

$$h(x) = sgn(w^T x + b) = sgn(\sum_{i=1}^{m}\alpha_i y_i x_i^T x + b)$$

Analog zum trennbaren Fall, können wir b mittels der Support-Vektoren x_i definieren, welche $0 < \alpha < C$ erfüllen.

$$b = y_i - \sum_{j=1}^{m}\alpha_j y_j x_j^T x_i$$

3 SMO-Algorithmus

Für eine Trainingsmenge $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ mit $x_i \in \mathbb{R}^n$ Beispielen und ihren zugehörigen Labels $y_i \in \{+1, -1\}, i \in [m]$ haben wir eine Hyperbene gesucht, die S linear und mit möglichst großem Abstand in die beiden Klassen unterteilt. Dafür haben wir über die Definition der Marge das primale Optimierungsproblem aufgesetzt und die Bedeutung der sogenannten Support-Vektoren kennengelernt. Über die Lagrange-Dualität haben wir dann die duale Form des Optimierungsproblems hergeleitet. Das Ziel des SMO-Algorithmus ist es nun eine schnelle approximierende Lösung für dieses duale Optimierungsproblem zu liefern.

3.1 Geschichtlicher Hintergrund & Konvergenzgewährleistung

1998 führte John C. Platt den SMO-Algorithmus ein, welcher der Lösung quadratischer Optimierungsprobleme dient. Das Training von Support Vektor Maschinen kann die Lösung eines sehr komplexen und hochdimensionalen quadratischen Optimierungsproblems fordern. Der SMO-Algorithmus zerlegt ein solches größere quadratische Optimierungsproblem in eine Reihe kleinstmöglicher quadratischer Optimierungsprobleme. Diese QP's werden exakt gelöst, wodurch eine zeitaufwändige numerische QP-Optimierung als innere Schleife vermieden wird.

Vladimir Vapnik beschrieb bereits eine Methode, wie dieses große QP in eine Reihe kleiner QP's zerlegt werden kann, die seitdem als "Chunking"-Methode bekannt ist. Der Chunking-Algorithmus nutzt die Tatsache, dass der Wert des QP's unverändert bleibt, wenn man diejenigen Zeilen und Spalten der Matrix entfernt, die den Lagrange-Multiplikatoren Null entsprechen. Daher wird das große QP-Problem in eine Reihe kleiner QP-Probleme zerlegt, deren Ziel es ist, alle Nicht-Null Lagrange-Multiplikatoren zu identifizieren und alle Null Lagrange-Multiplikatoren zu verwerfen. In jedem Schritt löst der "Chunking"-Algorithmus ein QP-Proplem, das aus jedem Nicht-Null Lagrange Multiplikator aus dem letzten Schritt und einer bestimmten Anzahl an schlechten Beispielen, die die KKT-Bedingungen verletzten, besteht.
Wie Platt in [Pla98] beschreibt, beinhaltet (2.30) eine Matrix mit einer Anzahl von Elementen, die dem Quadrat der Anzahl der Trainingsbeispiele entspricht. Diese Matrix kann nicht in 128 Megabytes untergebracht werden, wenn es mehr als 4000 Trainingsbeispiele gibt. Durch Chunking wird die Größe der Matrix von der Anzahl der Trainingsbeispiele zum Quadrat auf ungefähr die Anzahl der Lagrange-Multiplikatoren ungleich Null zum Quadrat. Allerdings kann Chunking immer noch keine großen Trainingsprobleme lösen, da selbst diese reduzierte Matrix möglicherweise nicht in den Speicher passt.

1997 bewiesen Osuna, Freund und Girosi in [Gir97] ein Theorem, das eine neue Reihe von QP-Algorithmen für SVM's vorschlägt. Dieses Theorem beweist, dass das große QP-Problem in eine Reihe von kleineren QP-Unterproblemen zerlegt werden kann, solange in jedem Schritt mindestens ein Beispiel, das die KKT-Bedingungen verletzt, hinzugefügt wird. Mit jedem Schritt wird die Zielfunktion reduziert und ein Punkt beibehalten, der alle Nebenbedingungen erfüllt. Dabei wird vorgeschlagen, dass eine konstante Größe für jedes QP-Teilproblem beizubehalten ist. Dies impliziert, dass in gleichem Maße Beispiele hinzugefügt und entfernt werden. Das ermöglicht das Training von beliebig großen Datensätzen. Genauer, der Algorthmus schlägt das Hinzufügen bzw. Entfernen von genau einem Datensatz pro Schritt vor. Offensichtlich wäre das ineffizient, weshalb in der Praxis mehrere Beispiele addiert und subtrahiert werden. Allerdings wird in jedem Fall für diese Methode ein numerischer QP-Löser benötigt und die numerische QP ist bekanntermaßen schwieriger zu lösen.

Sequential-Minimum-Optimization ist ein einfacher Algorithmus, der das SVM QP-Problem schnell lösen kann, ohne zusätzliche Matrixspeicherkapazität zu benötigen und vor allem ohne numerische QP-Optimierungsschritte zu verwenden. SMO zerlegt das gesamte QP Problem in QP-Unterprobleme und nutzt das Theorem von Osuna, um Konvergenz zu gewährleisten.

Wir stellen die erwähnten Konvergenztheoreme aus [Gir97] im Folgenden kurz vor. Dazu formulieren wir das Optimierungsproblem (2.30) wie folgt um: Es bezeichne e den Einheitsvektor und $Q = (q_{i,j})_{i,j=1,..,m}$ eine $m \times m$-Matrix mit den Einträgen $q_{ij} = y_i y_j x_i^T x_j$. Sei weiter $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_m) \in \mathbb{R}^m$, $\boldsymbol{y} = (y_1, ..., y_m) \in \{+1, -1\}^m$ und $\Psi(\boldsymbol{\alpha}) = e^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}$. Dann lässt sich das duale Optimierungsproblem (2.30) wie folgt umformulieren.

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & e^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} \\ & 0 \leq \boldsymbol{\alpha} \leq C \\ & y^T\boldsymbol{\alpha} = 0 \end{aligned} \tag{3.1}$$

Wir partitionieren die Indexmenge in zwei Teilmengen $B \subset \{1, ..., m\}$ und $N = \{1, ..., m\} \setminus B$ und zerlegen $\boldsymbol{\alpha}$ in $\boldsymbol{\alpha}_B$ und $\boldsymbol{\alpha}_N$. Wir erlauben eine Veränderung nur für $\boldsymbol{\alpha}_B$ und lassen damit $\boldsymbol{\alpha}_N$ konstant.

Mit

$$\begin{aligned} \Psi(\boldsymbol{\alpha}) &= \Psi(\boldsymbol{\alpha_B}, \boldsymbol{\alpha_N}) \\ &= e^T\boldsymbol{\alpha} + e^T\boldsymbol{\alpha}_N - \frac{1}{2}[\boldsymbol{\alpha}_B^T Q_{BB}\boldsymbol{\alpha}_B + \boldsymbol{\alpha}_B^T Q_{BN}\boldsymbol{\alpha}_N + \boldsymbol{\alpha}_N^T Q_{NB}\boldsymbol{\alpha}_B + \boldsymbol{\alpha}_N^T Q_{NN}\boldsymbol{\alpha}_N] \end{aligned}$$

ergibt sich folgendes QP-Unterproblem zu (3.1).

$$\begin{aligned} \max_{\boldsymbol{\alpha}_B} \quad & e^T\boldsymbol{\alpha} + e^T\boldsymbol{\alpha}_N - \frac{1}{2}[\boldsymbol{\alpha}_B^T Q_{BB}\boldsymbol{\alpha}_B + \boldsymbol{\alpha}_B^T Q_{BN}\boldsymbol{\alpha}_N + \boldsymbol{\alpha}_N^T Q_{NB}\boldsymbol{\alpha}_B + \boldsymbol{\alpha}_N^T Q_{NN}\boldsymbol{\alpha}_N], \\ \textbf{so dass} \quad & \boldsymbol{\alpha}_B^T\boldsymbol{y}_B + \boldsymbol{\alpha}_N^T\boldsymbol{y}_N = 0, \\ & 0 \le \boldsymbol{\alpha}_B \le C. \end{aligned}$$

In dem gegebenen QP-Unterproblem stellt $e^T\boldsymbol{\alpha}_N - \frac{1}{2}\boldsymbol{\alpha}_N^T Q_{NN}\boldsymbol{\alpha}_N$ eine Konstante dar.

Da $\langle x_i, x_j\rangle := x_i^T x_j$ symmetrisch ist, folgt

$$\boldsymbol{\alpha}_B^T Q_{BN}\boldsymbol{\alpha}_N + \boldsymbol{\alpha}_N^T Q_{NB}\boldsymbol{\alpha}_B = 2\boldsymbol{\alpha}_B^T Q_{BN}\boldsymbol{\alpha}_N.$$

Dies ist eine sehr wichtige Vereinfachung, da sie es uns ermöglicht, die Größe des Teilproblems unabhängig von der Anzahl der festen Variablen N zu halten, d.h. sie ist auch unabhängig von der Anzahl der Support-Vektoren.

Theorem 3.1. *Es sei oben angegebenes QP-Unterproblem zu in (3.1) angegebenem QP-Problem gegeben. Das Verschieben einer Indexvariable von B nach N lässt die Zielfunktion $\Psi(\boldsymbol{\alpha})$ unverändert und die Zulässigkeit für beide QPs, das Teilproblem und das ursprünglichen Problem, wird dabei nicht verletzt.*

Beweis. Sei $B' = B \setminus \{k\}$ und $N' = N \cup \{k\}$, wobei $k \in \{1, ...m\}$ beliebig gewählt werden kann. Dann gilt:

$$\begin{aligned} \Psi(\boldsymbol{\alpha}_B, \boldsymbol{\alpha}_N) &= \sum_{i\in B}\alpha_i + \sum_{i\in N}\alpha_i - \frac{1}{2}[\sum_{i,j\in B}\alpha_i\alpha_j Q_{ij} + 2\sum_{i\in B}\alpha_i\sum_{j\in N}\alpha_j Q_{ij} + \sum_{i,j\in N}\alpha_i\alpha_j Q_{ij}] \\ &= \sum_{i\in B'}\alpha_i + \alpha_k + \sum_{i\in N}\alpha_i - \frac{1}{2}[\sum_{i,j\in B'}\alpha_i\alpha_j Q_{ij} + 2\alpha_k\sum_{i\in B'}\alpha_i Q_{ik} \\ &\quad + 2\alpha_k\sum_{j\in N}\alpha_j Q_{jk} + 2\sum_{i\in B'}\alpha_i\sum_{j\in N}\alpha_j Q_{ij} + \alpha_k^2 Q_{kk} + \sum_{i,j\in N}\alpha_i\alpha_j Q_{ij}] \\ &= \sum_{i\in B'}\alpha_i + \sum_{i\in N'}\alpha_i - \frac{1}{2}[\sum_{i,j\in B'}\alpha_i\alpha_j Q_{ij} + 2\sum_{i\in B'}\alpha_i\sum_{j\in N'}\alpha_j Q_{ij} + \sum_{i,j\in N'}\alpha_i\alpha_j Q_{ij}] \\ &= \Psi(\boldsymbol{\alpha}_{B'}, \boldsymbol{\alpha}_{N'}) \end{aligned}$$

Die Lösung $\Psi(\boldsymbol{\alpha}_{B'}, \boldsymbol{\alpha}_{N'})$ ist zulässig für das QP-Unterproblem, da

$$\begin{aligned} 0 &= \boldsymbol{\alpha}_B^T\boldsymbol{y}_B + \boldsymbol{\alpha}_N^T\boldsymbol{y}_N \\ &= \boldsymbol{\alpha}_{B'}^T\boldsymbol{y}_{B'} + \alpha_k y_k + \boldsymbol{\alpha}_N^T\boldsymbol{y}_N \\ &= \boldsymbol{\alpha}_{B'}^T\boldsymbol{y}_{B'} + \boldsymbol{\alpha}_{N'}^T\boldsymbol{y}_{N'} \end{aligned}$$

gilt und die Nebenbedingung $0 \le \boldsymbol{\alpha} \le C$ unberührt bleibt. □

Bemerkung 3.1.1. *Wenn das Teilproblem vor einer solchen Ersetzung optimal ist, ist das neue Teilproblem dann und nur dann optimal, wenn j die Optimalitätsbedingungen aus (2.29) für den entsprechenden Fall erfüllt.*

Theorem 3.2. *Sei obiges QP-Unterproblem gegeben. Das Verschieben einer Variablen von N nach B, welche die Optimalitätsbedingungen verletzt, führt zu einer strikten Verbesserung der Zielfunktion* $\Psi(\boldsymbol{\alpha})$ *wenn das QP-Unterproblem erneut optimiert wird.*

Beweis. Der Beweis folgt direkt aus Thorem 3.1 und der Tatsache, dass die KKT-Bedigungen ein notweniges und hinreichendes Optimalitätskriterium für das angegebene QP sind. □

Gemäß Theorem 3.2 wird die Zielfunktion bei jeder Iteration strikt verbessert. Da die Zielfunktion $\Psi(\alpha)$ beschränkt ist ($\Psi(\alpha)$ ist konvex quadratisch und die zulässige Menge ist beschränkt), muss der Algorithmus in einer endlichen Anzahl von Iterationen zur globalen optimalen Lösung konvergieren.

Der SMO-Algorithmus kann als Spezialfall des Algorithmus in [Gir97] betrachtet werden, bei dem die Größe der Optimierung zwei beträgt und beide Lagrange-Multiplikatoren bei jedem Schritt durch neue Multiplikatoren ersetzt werden, die durch gute Heuristiken ausgewählt werden.

3.2 Sequential Minimal-Optimization

SMO wählt in jedem Schritt das kleinstmögliche Optimierungsproblem. Für unser SVM-QP beinhaltet das kleinstmögliche Optimierungsproblem zwei Lagrange-Multiplikatoren. In jedem Schritt wählt der SMO-Algorithmus zwei Lagrange-Multiplikatoren zur gemeinsamen Optimierung, findet die optimalen Werte für diese Multiplikatoren und aktualisiert das SVM, um die neuen optimalen Werte zu berücksichtigen.

Der Vorteil des SVM liegt darin, dass das Lösen für zwei Lagrange-Multiplikatoren explizit erfolgen kann. Dadurch wird eine numerische QP-Optimierung vollständig vermieden. Auch wenn im Laufe des Algorithmus mehrere QP-Unterprobleme gelöst werden müssen, ist jedes Teilproblem so schnell, dass das gesamte QP-Problem schnell gelöst werden kann.

Der SMO-Algorithmus besteht aus zwei Komponenten: einer expliziten Methode zum Lösen der beiden Lagrange-Multiplikatoren und einer Heuristik, zur möglichst optimalen Auswahl der zu optimierenden Multiplikatoren.

3.2.1 Lösen der zwei Lagrange-Multiplikatoren

Um die beiden Lagrange-Multiplikatoren zu lösen, berechnet SMO zunächst die Nebenbedingungen für diese Multiplikatoren und löst dann das eingeschränkte Minimum. Wir kennzeichnen der Einfachheit-halber alle Größen, die sich auf den ersten Multiplikator beziehen, mit einer tiefgestellten 1 und die Größen, die sich auf den Zweiten beziehen mit einer 2. Zudem steht α^{alt} für den Wert von α vor der Optimierung und α^{neu} für den Wert nach der Optimierung.

Wir betrachten das Optimierungsproblem (2.30) aus Kapitel 2.4.2 und maximieren über die zwei Lagrange-Multiplikatoren α_1 und α_2.

$$\max_{\alpha_1,\alpha_2}\Big(\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j\Big), \tag{3.2}$$
$$\text{so dass} \quad \sum_{i=1}^{m}\alpha_i y_i = 0,$$
$$0 \le \alpha_i \le C,$$
$$\forall i \in \{1, ...m\}.$$

O.B.d.A. wird zuerst eine Lösung für α_2 berechnet. Mit den gegebenen Nebenbedingungen lassen sich eine untere Schranke L und eine obere Schranke H bestimmen, so dass $L \le \alpha_2 \le H$.

Für α_1 und α_2 gelten die folgenden Beschränkungen.

(1) $0 \le \alpha_1 \le C \quad \Leftrightarrow \quad 0 \le \alpha_1 \wedge \alpha_1 \le C \quad \Leftrightarrow \quad 0 \ge -\alpha_1 \wedge -\alpha_1 \ge C$

(2) $0 \le \alpha_2 \le C \quad \Leftrightarrow \quad 0 \le \alpha_2 \wedge \alpha_2 \le C \quad \Leftrightarrow \quad 0 \ge -\alpha_1 \wedge -\alpha_2 \ge C$

Mit Hilfe von (1) und (2) können wir nun $y_1\alpha_1 + y_2\alpha_2$ für α_2 über eine Fallunterscheidung abschätzen.

Theorem 3.3. *Für das Optimierungsproblem in (3.2) ergeben sich die folgenden Schranken L und H, so dass $L < \alpha_2 < H$.*

$$L = \max(0, \alpha_2 - \alpha_1), \quad H = \min(C, C - \alpha_1 + \alpha_2) \quad \textit{für} \quad y_1 = y_2$$
$$L = \max(0, \alpha_1 + \alpha_2 - C), \quad H = \min(C, \alpha_1 + \alpha_2) \quad \textit{für} \quad y_1 \ne y_2$$

Beweis. Fall 1: $y_1 = y_2$
Aus den Ungleichungen (1) und (2) folgt für $y_1 = y_2 = 1$

$$y_1\alpha_1 + y_2\alpha_2 = \alpha_1 + \alpha_2 \le C + \alpha_2$$
$$\alpha_2 \le C + \alpha_2 - \alpha_1 = C - \alpha_1 + \alpha_2$$

und

$$\begin{aligned} 0 + \alpha_2 &\leq \alpha_1 + \alpha_2 \\ \alpha_2 &\leq \alpha_1 + \alpha_2 \\ \alpha_2 - \alpha_1 &\leq \alpha_2. \end{aligned}$$

Analog folgt für $y_1 = y_2 = -1$

$$\begin{aligned} y_1\alpha_1 + y_2\alpha_2 = -\alpha_1 - \alpha_2 &\geq -C - \alpha_2 \\ -\alpha_2 &\geq -C - \alpha_2 + \alpha_1 \\ \alpha_2 &\leq C + \alpha_2 - \alpha_1 = C - \alpha_1 + \alpha_2 \end{aligned}$$

und

$$\begin{aligned} 0 - \alpha_2 &\geq -\alpha_1 - \alpha_2 \\ \alpha_2 &\leq \alpha_1 + \alpha_2 \\ \alpha_2 - \alpha_1 &\leq \alpha_2. \end{aligned}$$

Fall 2: $y_1 \neq y_2$
Aus den Ungleichungen (1) und (2) folgt für $y_1 = 1, y_2 = -1$

$$\begin{aligned} y_1\alpha_1 + y_1\alpha_2 = \alpha_1 - \alpha_2 &\leq C - \alpha_2 \\ -\alpha_2 &\leq C - \alpha_2 - \alpha_1 \\ \alpha_2 &\geq -C + \alpha_2 + \alpha_1 = \alpha_1 + \alpha_2 - C \end{aligned}$$

und

$$\begin{aligned} 0 - \alpha_2 &\leq \alpha_1 - \alpha_2 \\ -\alpha_2 - \alpha_1 &\leq -\alpha_2 \\ \alpha_2 + \alpha_1 &\geq \alpha_2. \end{aligned}$$

Analog folgt für $y_1 = -1, y_2 = 1$

$$\begin{aligned} y_1\alpha_1 + y_1\alpha_2 = -\alpha_1 + \alpha_2 &\geq -C + \alpha_2 \\ \alpha_2 &\geq -C + \alpha_2 + \alpha_1 = \alpha_1 + \alpha_2 - C \end{aligned}$$

und

$$\begin{aligned} 0 + \alpha_2 &\geq -\alpha_1 + \alpha_2 \\ \alpha_2 + \alpha_1 &\geq \alpha_2. \end{aligned}$$

Damit erhalten wir für $y_1 = y_2$ die Schranken

$$L = \max(0, \alpha_2 - \alpha_1), \quad H = \min(C, C - \alpha_1 + \alpha_2)$$

und für $y_1 \neq y_2$ die Schranken

$$L = \max(0, \alpha_1 + \alpha_2 - C), \quad H = \min(C, \alpha_1 + \alpha_2)$$

□

Theorem 3.4. *Sei* $\Psi(\alpha_2) := \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$ *die Zielfunktion aus (3.2) in Abhängigkeit von* α_2 *mit* $(\alpha_1, \alpha_3, ..., \alpha_m) \in \mathbb{R}^{m-1}, x_i \in \mathbb{R}^m, y_i \in \{+1, -1\}$ *und* $i, j \in \{1, .., m\}$. *Dann ist*

$$\eta := \frac{\partial^2 \Psi(\alpha_2)}{\partial^2 \alpha_2} = 2x_1^T x_2 - x_1^T x_1 - x_2^T x_2$$

Beweis. Wir schreiben zuerst $\Psi(\alpha_2)$ geeignet um und bezeichnen mit Ψ_{const} den Teil aus der Zielfunktion, der nicht von α_2 und α_1 abhängt.

$$\begin{aligned}
\Psi(\alpha_2) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
&= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} (\sum_{i=1}^{m} (\alpha_i \alpha_1 y_i y_1 x_i^T x_1 + \alpha_i \alpha_2 y_1 y_2 x_i^T x_2 + ... + \alpha_i \alpha_m y_i y_m x_i^T x_m)) \\
&= \alpha_1 + \alpha_2 - \frac{1}{2} \alpha_1^2 x_1^T x_1 - \frac{1}{2} \alpha_2^2 x_2^T x_2 - \alpha_1 \alpha_2 y_1 y_2 x_1^T x_2 - \sum_{j=3}^{m} \alpha_1 \alpha_j y_1 y_j x_1^T x_j \\
&\quad - \sum_{j=3}^{m} \alpha_2 \alpha_j y_2 y_j x_2^T x_j + \Psi_{const}
\end{aligned}$$

Sei des Weiteren $\alpha_1 + y_1 y_2 \alpha_2 := k(\alpha_2)$ und $y_1 y_2 := s$, so erhalten wir nun eine Zielfunktion nur in Abhängigkeit von α_2:

$$\begin{aligned}
\Psi(\alpha_2) = k(\alpha_2) - s\alpha_2 + \alpha_2 - \frac{1}{2}(k(\alpha_2) - s\alpha_2) x_1^T x_1 - \frac{1}{2} \alpha_2^2 x_2^T x_2 \\
- s(k(\alpha_2) - s\alpha_2) \alpha_2 x_1^T x_2 - \sum_{j=3}^{m} (k(\alpha_2) - s\alpha_2) \alpha_j y_1 y_j x_1^T x_j \\
- \sum_{j=3}^{m} \alpha_2 \alpha_j y_2 y_j x_2^T x_j + \Psi_{const}
\end{aligned} \tag{3.3}$$

Mit (3.3) und unter Verwendung von $s^2 = 1$ erhalten wir

$$\begin{aligned}
\frac{\partial \Psi(\alpha_2)}{\partial \alpha_2} = &- s + 1 + s(k - s\alpha_2) x_1^T x_1 - \alpha_2 x_2^T x_2 - sk x_1^T x_2 + 2\alpha_2 x_1^T x_2 \\
&+ s \sum_{j=3}^{m} \alpha_j y_1 y_j x_1^T x_j - \sum_{j=3}^{m} \alpha_j y_2 y_j x_2^T x_j
\end{aligned} \tag{3.4}$$

und damit ist die zweite Ableitung nach α_2 gegeben mit

$$\frac{\partial^2 \Psi(\alpha_2)}{\partial^2 \alpha_2} = 2x_1^T x_2 - x_1^T x_1 - x_2^T x_2$$

□

Mit Hilfe der ersten und zweiten Ableitung können wir die optimierten Werte für α_1 und α_2 herleiten.

Bemerkung 3.4.1. *Da Theorem 3.4 $\eta < 0$ garantiert und $\eta \neq 0$ für $i \neq j, i, j \in \{1, ..., m\}$, betrachten wir nur den Fall $\eta < 0$.*

Theorem 3.5. *Unter der Bedingung, dass nur die Werte für α_1 und α_2 geändert werden dürfen, wird der maximale Wert der Zielfunktion in (3.2) für $\eta < 0$ unter*

$$\alpha_2^{neu} = \alpha_2 - \frac{y_2(E_1 - E_2)}{\eta} \tag{3.5}$$

und

$$\alpha_1^{neu} = \alpha_1 + s(\alpha_2 - \alpha_2^{neu,clipped})$$

angenommen, wobei

$$\alpha_2^{neu,clipped} = \begin{cases} H & für \quad \alpha_2^{neu} \geq H \\ \alpha_2^{neu} & für \quad H < \alpha_2^{neu} < L \\ L & für \quad \alpha_2^{neu} \leq L \end{cases}$$

und

$$E_i := u(x_i) - y_i,$$
$$u(x_i) := \sum_{j=1}^{m} \alpha_j y_j x_i^T x_j + b.$$

Beweis. Damit α_2^{neu} ein kritischer Punkt ist, muss demnach nur

$$\frac{\partial \Psi(\alpha_2^{neu})}{\partial \alpha_2^{neu}} = 0$$

gelten. Gegeben (3.4), mit $k(\alpha_2) = \alpha_1 + s$ und $\alpha_2, s = y_1 y_2$ berechnen wir

$$\begin{aligned}
0 \quad = \frac{\partial \Psi(\alpha_2)}{\partial \alpha_2} = \quad & -s + 1 + s(k(\alpha_2) - s\alpha_2)x_1^T x_1 - \alpha_2 x_2^T x_2 - sk(\alpha_2)x_1^T x_2 \\
& + 2\alpha_2 x_1^T x_2 + s\sum_{j=3}^{m} \alpha_j y_1 y_j x_1^T x_j - \sum_{j=3}^{m} \alpha_j y_2 y_j x_2^T x_j \\
= \quad & -s + 1 + sk(\alpha_2)x_1^T x_1 - \alpha_2^{neu} x_1^T x_1 - \alpha_2 x_2^T x_2 - sk(\alpha_2)x_1^T x_2 \\
& + 2\alpha_2 x_1^T x_2 + s\sum_{j=3}^{m} \frac{y_2}{s}\alpha_j y_j x_1^T x_j - \sum_{j=3}^{m} \alpha_j y_2 y_j x_2^T x_j \\
= \quad & -s + 1 + y_2(\sum_{j=3}^{m} \alpha_j y_j x_1^T x_j - \sum_{j=3}^{m} \alpha_j y_j x_2^T x_j) \\
& + sk(\alpha_2)(x_1^T x_1 - x_1^T x_2) + \alpha_2 \underbrace{(-x_1^T x_1 - x_2^T x_2 + 2x_1^T x_2)}_{=\eta} \\
\Leftrightarrow \quad -\alpha_2\eta \quad = \quad & -s + 1 + y_2(\sum_{j=3}^{m} \alpha_j y_j x_1^T x_j - \sum_{j=3}^{m} \alpha_j y_j x_2^T x_j) \\
& + sk(\alpha_2)(x_1^T x_1 - x_1^T x_2) \\
\Leftrightarrow \quad -\alpha_2\eta y_2 \quad = \quad & y_2 - y_1 + y_1 k(\alpha_2)(x_1^T x_1 - x_1^T x_2) + \sum_{j=3}^{m} \alpha_j y_j x_1^T x_j \\
& - \sum_{j=3}^{m} \alpha_j y_j x_2^T x_j
\end{aligned}$$

Mit

$$\sum_{j=1}^{m} \alpha_j y_j x_i^T x_j = \alpha_1 y_1 x_i^T x_1 + \alpha_2 y_2 x_i^T x_2 + \sum_{j=3}^{m} \alpha_j y_j x_i^T x_j$$

für $i \in \{1, 2\}$ erhalten wir

$$\begin{aligned}
-\alpha_2\eta y_2 \quad = \quad & y_2 - y_1 + y_1 k(x_1^T x_1 - x_1^T x_2) + \underbrace{\sum_{j=1}^{m} \alpha_j y_j x_1^T x_j}_{=u(x_1)-b} - \alpha_1 y_1 x_1^T x_1 \\
& - \alpha_2 y_2 x_1^T x_2 - \underbrace{\sum_{j=1}^{m} \alpha_j y_j x_2^T x_j}_{=u(x_2)-b} + \alpha_1 y_1 x_2^T x_1 + \alpha_2 y_2 x_2^T x_2
\end{aligned}$$

Einsetzen von $k = \alpha_1 + y_1 y_2 \alpha_2$ und $E_i := u(x_i) - y_i$ ergibt

$$\begin{aligned}
-\alpha_2 \eta y_2 =& E_1 - E_2 + y_1(\alpha_1 + y_1 y_2 \alpha_2) x_1^T x_1 - y_1(\alpha_1 + y_1 y_2 \alpha_2) x_1^T x_2 \\
& - \alpha_1 y_1 x_1^T x_1 - \alpha_2 y_2 x_1^T x_2 + \alpha_1 y_1 x_2^T x_1 + \alpha_2 y_2 x_2^T x_2 \\
=& E_1 - E_2 + y_1 \alpha_1 x_1^T x_1 + y_2 \alpha_2 x_1^T x_1 - y_1 \alpha_1 x_1^T x_2 - y_2 \alpha_2 x_1^T x_2 \\
& - \alpha_1 y_1 x_1^T x_1 - \alpha_2 y_2 x_1^T x_2 + \alpha_1 y_1 x_2^T x_1 + \alpha_2 y_2 x_2^T x_2 \\
=& E_1 - E_2 + y_2 \alpha_2 x_1^T x_1 - y_2 \alpha_2 x_1^T x_2 - \alpha_2 x_1^T x_2 + \alpha_2 x_2^T x_2 \\
=& E_1 - E_2 + y_2 \alpha_2 \underbrace{(x_1^T x_1 + x_2^T x_2 - 2 x_1^T x_2)}_{=-\eta}
\end{aligned}$$

Daraus erhalten wir

$$\begin{aligned}
-\alpha_2 \eta y_2 &= E_1 - E_2 - \alpha_2 \eta y_2 \\
\alpha_2 \eta y_2 &= \alpha_2 \eta y_2 - (E_1 - E_2) \\
\Rightarrow \alpha_2^{neu} &= \alpha_2^{alt} - \frac{y_2 (E_1 - E_2)}{\eta}
\end{aligned}$$

E_i wird auch als Fehlerrate des i-ten Trainingsbeispiels bezeichnet.
Falls das Maximum außerhalb der Grenzen liegt, wird α_2^{neu} entsprechend angepasst:

$$\alpha_2^{neu,clipped} = \begin{cases} H & \text{für} \quad \alpha_2^{neu} \geq H \\ \alpha_2^{neu} & \text{für} \quad H < \alpha_2^{neu} < L \\ L & \text{für} \quad \alpha_2^{neu} \leq L \end{cases}$$

Für $s = y_1 y_2$, bestimmt man den Wert von α_1^{neu} trivialerweise wie folgt.

$$\alpha_1^{neu} = \alpha_1 + s(\alpha_2 - \alpha_2^{neu,clipped})$$

□

Des Weiteren ergeben sich dann für α_1^{neu} die untere Schranke L_1 und die obere Schranke H_1 mit:

$$L_1 = \alpha_1 + s(\alpha_2 - L) \quad H_1 = \alpha_1 + s(\alpha_2 - H)$$

3.2.2 Heuristik für die Wahl der zu optimierenden Multiplikatoren

Bisher haben wir hergeleitet, wie die zwei ausgewählten Lagrange-Multiplikatoren zu optimieren sind. In diesem Abschnitt klären wir, wie die passenden Indizes in jedem Iterationsschritt gewählt werden und welche Größen zusätzlich angepasst werden müssen.

Solange SMO bei jedem Schritt zwei Lagrange-Multiplikatoren optimiert und mindestens einer der beiden die KKT-Bedingungen vor der Optimierung verletzt, wird mit jedem Schritt die Zielfunktion laut Osunas Theorem vergrößert bzw. verringert. Je nachdem ob man das Maximierungs- oder Minimierungsproblem betrachtet. Daher wird Konvergenz garantiert. Um die Konvergenz zu beschleunigen, verwendet SMO Heuristiken für die Wahl der zu optimierenden Lagrange-Multiplikatoren. Man unterscheidet zwischen zwei Auswahlheuristiken. Die Wahl der ersten Heuristik liefert die äußere Schleife und bezieht sich auf den ersten Lagrange-Multiplikator und die Wahl der zweiten Heuristik auf den zweiten La - grange-Multiplikator.
Die äußere Schleife iteriert zunächst über die gesamte Trainingsmenge und bestimmt, ob jedes Beispiel die KKT-Bedingungen verletzt. Wenn ein Beispiel die KKT-Bedingungen verletzt, kommt es für eine sofortige Optimierung in Frage. Anschließend wird der zweite Multiplikator mit Hilfe der zweiten Heuristik ausgewählt und die beiden Multiplikatoren werden gemeinsam optimiert.

Um das Training zu beschleunigen, iteriert die äußere Schleife nicht immer durch die gesamte Trainingsmenge. Nach einem Durchlauf durch die Trainingsmenge iteriert die äußere Schleife nur über die Beispiele, deren Lagrange-Multiplikatoren weder 0 noch C sind, die nicht-gebundenen Beispiele. Wie wir in Kapitel 2.4.1 feststellen konnten, sind die Support-Vektoren, diejenigen die $0 < \alpha_i < C$ erfüllen und damit sind die Trainingsbeispiele mit $\alpha_i \in \{0, C\}$ nicht-bindend für das Optimierungsproblem. Auch hier wird jedes Beispiel anhand der KKT-Bedingungen überprüft und Beispiele, die gegen die Bedingungen verstoßen, kommen für eine sofortige Optimierung und Aktualisierung in Frage. Die äußere Schleife macht wiederholte Durchläufe über die nicht-gebundenen Beispiele, bis alle diese Beispiele den KKT-Bedingungen gehorchen. Die äußere Schleife geht dann zurück und iteriert über die gesamte Trainingsmenge. Damit wechselt die äußere Schleife zwischen einzelnen Durchläufen über die gesamte Trainingsmenge und mehreren Durchläufen über die nicht-gebundene Teilmenge, bis die gesamte Trainingsmenge den KKT-Bedingungen innerhalb von ε gehorcht, woraufhin der Algorithmus beendet wird.

Bei der ersten Auswahlheuristik konzentriert sich der Algorithmus im Wesent-

lichen auf die Prozessorzeit der Beispiele, die am wahrscheinlichsten die KKT-Bedingungen verletzen und das sind die nicht-gebundenen Beispiele. Während der Laufzeit des SMO-Algorithmus werden Beispiele, die an den Grenzen liegen, sehr wahrscheinlich auch an den Grenzen bleiben. Wohingegegen Beispiele, die sich weiter weg von der Grenze befinden, durch die Optimierung anderer Beispiele, verschoben werden. Daher iteriert SMO solange über die nicht-gebundene Teilmenge, bis diese selbstkonsistent ist. Im Anschluss wird der gesamte Datensatz gescannt, um nach allen gebundenen Beispielen zu suchen, die auf Grund der Optimierung der nicht-gebundenen Teilmenge, die KKT-Bedingungen verletzen.

Wie bereits erwähnt, gewährleistet SMO Konvergenz. Die Schnelligkeit der Konvergenz hängt von der Wahl von ε ab. Typischerweise wird z.B. $\varepsilon = 10^{-3}$ gesetzt, da man in der Regel mit keiner hohen Genauigkeit die Erfüllung der KKT-Bedingungen voraussetzt. Deswegen sollte man bedenken, dass der Algorithmus bei Erwartung hoher Genauigkeit langsamer konvergieren wird.

Bei der zweiten Auswahlheuristik geht es um die optimale Wahl des zweiten Lagrange-Multiplikators. Sobald der erste Lagrange-Multiplikator gewählt wurde, sucht SMO nach einem Lagrange-Multiplikator, der in gemeinsamer Optimierung mit dem ersten, den größten durchgeführten Schritt in der Optimierung macht. Betrachten wir die Darstellung (3.5), so können wir erkennen, dass die Änderung des zweiten Lagrange-Multiplikators von η und $E_1 - E_2$ abhängt. Da die Berechnung von η zu zeitaufwendig ist, wählt man den zweiten Lagrange-Multiplikator so, dass $E_1 - E_2$ maximiert wird.

Unter ungünstigen Umständen kann SMO mit der oben beschriebenen zweiten Auswahlheuristik keinen Fortschritt erzielen. In diesem Fall beginnt SMO mit der Iteration durch die nicht-gebundenen Beispiele und sucht nach einem zweiten Beispiel, das einen positiven Fortschritt erzielen kann. Wenn keines der nicht-gebundenen Beispiele einen positiven Fortschritt macht, beginnt SMO, durch den gesamten Trainingssatz zu iterieren, bis ein Beispiel gefunden wird, das einen Fortschritt erzielt. Sowohl die Iteration durch die nicht-gebundenen Beispiele als auch die Iteration durch die gesamte Trainingsmenge werden an zufälligen Stellen gestartet. Unter extrem degenerierten Umständen wird keines der Beispiele ein adäquates zweites Beispiel ergeben. Wenn dies geschieht, wird das erste Beispiel übersprungen und SMO fährt mit einem anderen ausgewählten ersten Beispiel fort.

3.2.3 Aktualisierung von w

Nach jedem Durchlauf muss der Vektor w aktualisiert werden. w^{neu} ergibt sich aus der Differenz der beiden optimierten Lagrange-Multiplikatoren und deren ursprünglichem Wert und lässt sich wie folgt ermitteln.

$$w^{neu} = w^{alt} + y_1(\alpha_1^{neu} - \alpha_1)x_1 + y_2(\alpha_2^{neu,clipped} - \alpha_2)x_2$$

3.2.4 Aktualisierung von b

Wir haben bereits in den vorherigen Kapiteln festgestellt, dass b mittels der Support-Vektoren ausgedrückt werden kann.

$$b = y_i - w^T x_i = y_i - \sum_{i=1}^{m} \alpha_j y_j x_j^T x_i,$$

wobei x_i ein Support-Vektor ist.

Ist $\alpha_1^{neu} \in (0, C)$ folgt $y_1 \stackrel{!}{=} u(x_i)$ und damit wird b_1 für α_1^{neu} wie folgt aktualisiert.

$$\begin{aligned} b_1 &= y_1 - \sum_{j=1}^{m} \alpha_j^{alt} y_j x_1^T x_j - y_1\alpha_1^{neu} x_1^T x_1 + y_1\alpha_1^{alt} x_1^T x_1 - y_2\alpha_2^{neu,clipped} x_1^T x_2 + y_2\alpha_2^{alt} x_1^T x_2 \\ &= -E_1 - y_1(\alpha_1^{neu} - \alpha_1^{alt})x_1^T x_1 - y_2(\alpha_2^{neu,clipped} - \alpha_2^{alt})x_1^T x_2 + b^{alt} \end{aligned}$$

Analog erhalten wir b_2 für $\alpha_2^{neu} \in (0, C)$:

$$b_2 = -E_2 - y_1(\alpha_1^{neu} - \alpha_1^{alt})x_1^T x_2 - y_2(\alpha_2^{neu,clipped} - \alpha_2^{alt})x_2^T x_2 + b^{alt}$$

Wenn sowohl b_1, als auch b_2 gültig sind, dann sind sie gleich. Wenn die beiden optimierten Lagrange-Multiplikatoren an den Grenzen liegen und $L \neq H$ gilt, dann gibt das Intervall $[b_1, b_2]$ die Werte an, die mit den KKT-Bedingungen übereinstimmen und SMO wählt die Mitte des Intervalls.

3.2.5 Aktualisierung von E

Für alle α_i, die $0 < \alpha_i < C$ erfüllen, wird die Fehlerrate E_i wie in Theorem 3.5 definiert, konstant angepasst und mitgeführt. E_i wird dabei wie folgt aktualisiert.

$$\begin{aligned} E_i^{neu} = \; & E_i^{alt} - \alpha_1^{alt} y_1 x_1^T x_i + \alpha_1^{neu} y_1 x_1^T x_i - \alpha_2^{alt} y_2 x_2^T x_i + \alpha_2^{neu,clipped} y_2 x_2^T x_i - b^{alt} \\ & + b^{neu} \\ = \; & E_i^{alt} + y_1(\alpha_1^{neu} - \alpha_1^{alt})x_1^T x_i + y_2(\alpha_2^{neu,clipped} - \alpha_2^{alt})x_2^T x_i - b^{alt} + b^{neu} \end{aligned}$$

Liegt α_i an den Grenzen, also für $\alpha_i \in \{0, C\}$, so wird E_i gleich Null gesetzt.

3.3 Zusammenfassung

Wir setzen eine Trainingsmenge $S = \{(x_1, y_1)..., (x_m, y_m)\} \in \{X \times Y\}^m$ für $m \in \mathbb{N}, X \in \mathbb{R}^n, Y \in \{+1, -1\}$ mit $y_i = f(x_i) \quad \forall i \in \{1, .., m\}$ voraus. Gesucht ist die Lösung des folgenden Optimierungsproblems.

$$\begin{aligned} \max_{\alpha}\Big(\sum_{i=1}^{m}\alpha_i - \frac{1}{2}||\sum_{i=1}^{m}\alpha_i y_i x_i||^2\Big), \\ \text{so dass} \quad \sum_{i=1}^{m}\alpha_i y_i = 0, \\ 0 \leq \alpha_i \leq C, \\ \forall i \in \{1, ...m\}. \end{aligned}$$

1. Sei $k \in \mathbb{N}$. Setze $k = 1$ und setze $\alpha^k = (\alpha_1^k, ..., \alpha_m^k) = (0, ..., 0) \in \mathbb{R}^m$ als Startwert und Lösung des QPs.

2. Ist α^k optimal für das QP, STOP. Andernfalls finde eine zwei-elementige Teilmenge $B = \{i, j\} \subset \{1, ..., m\}$ wie in 3.2.2 beschrieben, um passende α_i und α_j zu optimieren. Definiere $N = \{1, ..., m\}\backslash B$ und α_B^k und α_N^k als Untervektoren von α^k.

3. Bestimme die Schranken für α_j, so dass $L < \alpha_j < H$:

   Für $y_i = y_j : L = \max(0, \alpha_j - \alpha_i)$ und $H = \min(C, C - \alpha_i + \alpha_j)$
   Für $y_i \neq y_j : L = \max(0, \alpha_i + \alpha_j - C)$ und $H = \min(C, \alpha_i + \alpha_j)$

4. Berechne $\eta = 2x_i^T x_j - x_i^T x_i - x_j^T x_j$ und setze

   $$\alpha_j^{k+1} = \alpha_j^k - \frac{y_j(E_i - E_j)}{\eta}$$

   mit $E_i = (\sum_{j=1}^{m} \alpha_j^k y_j x_i^T x_j + b) - y_i$
   Passe α_j^{k+1} entsprechend den Schranken an:

   $$\alpha_j^{k+1} = \begin{cases} L & \text{für} \quad \alpha_j^{k+1} \leq L \\ \alpha_j^{k+1} & \text{für} \quad H < \alpha_j^{k+1} < L \\ H & \text{für} \quad \alpha_j^{k+1} \geq H \end{cases}$$

6. Setze

   $$\alpha_i^{k+1} = \alpha_i^k + y_i y_j(\alpha_j^k - \alpha_j^{k+1})$$

7. Aktualisiere w, b und E wie in (3.2.3), (3.2.4) und (3.2.5) beschrieben.

8. Setze $\alpha_N^{k+1}{=}\alpha_N^k$ und α^{k+1} als neue optimale Lösung des OPs.
   Gehe zu Schritt 2.

3.3.1 Beispiele

Wir zeigen an Hand von zwei einfach gewählten Beispielen zuerst die Lösung des SMO-Algorithmus auf und veranschaulichen das Resultat im Anschluss geometrisch. Wir beschränken uns dabei auf eine Trainingsmenge mit zwei Datensätzen, so dass die Schleifen zur Ermittlung der zu optimierenden Multiplikatoren wegfällt.

Beispiel 3.6. *Sei* $n = 2$ *und* $S = \{(x_1, y_1), (x_2, y_2)\} = \{((1,1), -1), ((2,2), 1)\}$ *eine gegebene Trainingsmenge mit* $m = 2$ *Daten, wobei* $x \in \mathbb{R}^n$ *und* $y \in \{+1, -1\}$.

Greifen wir die Spam-Erkennung aus der Einleitung auf, so könnten $x_1 = (1,1)$ *und* $x_2 = (2,2)$ *zwei E-Mails mit ihren jeweiligen Eigenschaften, projiziert auf die reellen Zahlen, sein. Dabei könnte* $y_1 = -1$ *für „Nicht-Spam “ und* $y_2 = 1$ *für „Spam“ stehen.*

Wir setzen $\alpha = (0,0)^T, w = (0,0)^T, b = 0$ *als Startwerte.*
Mit Hilfe des im Anhang A angegebenen Python-Codes simulieren wir die Zusammenfassung in 3.3 für das gegebene Beispiel und bilden die gesuchte Hyperbene mit ihren Randhyperbenen ab.

```
w= [1. 1.]
b= -3.0
alpha= [1. 1.]
Fehlerratenvektor= [0. 0.]
Die gesuchte Hyperebene ist: [1. 1.] x -3.0 =0
Die Maximale Marge ist: 0.71
```

Abbildung 5: Ausgabe des Codes für 3.6

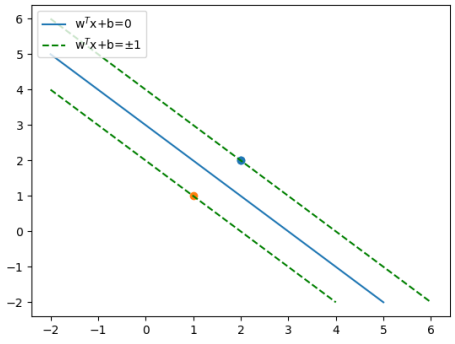

Abbildung 6: Hyperebene mit maximaler Marge für Bsp.3.6

Beispiel 3.7. *Sei* $n = 3$ *und* $S = \{(x_1, y_1), (x_2, y_2)\} = \{((1, 1, 1.5), -1), ((5, 3, 5), 1)\}$ *eine gegebene Trainingsmenge mit* $m = 2$ *Daten, wobei* $x \in \mathbb{R}^n$ *und* $y \in \{+1, -1\}$.

Wir setzen $\alpha = (0, 0, 0)^T, w = (0, 0, 0)^T, b = 0$ *und bestimmen mittels des in Anhang B angegebenen Python-Codes die Lösung des SMO-Algorithmus und simulieren die Lösung im Anschluss graphisch.*

```
w= [0.25 0.12 0.22]
b= -1.7
alpha= [0.06 0.06]
Fehlerratenvektor= [0. 0.]
Die gesuchte Hyperebene ist: [0.25 0.12 0.22] ^Tx -1.7 =0
Die Maximale Marge ist: 2.84
```

Abbildung 7: Ausgabe des Codes für Beispiel 3.7

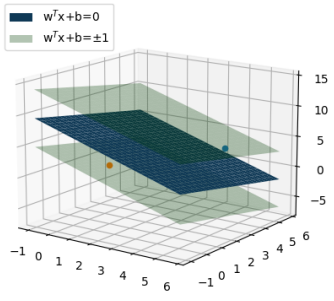

Abbildung 8: Hyperebene mit maximaler Marge für Beispiel 3.7

4 Anhang

4.0.1 Anhang A

Zur Berechnung des Beispiels 3.6 wurde folgender Python Code verwendet.

```
import numpy as np
import matplotlib.pyplot as plt
import math

#Anfangswerte
x = np.array([[1, 1], [2, 2]])
y = np.array([-1, 1])
m = len(x)
a = np.zeros(m)
i = 0
j = 1
b = 0
w = 0
C = 1

def LinKer(x, y):
    return np.dot(x, y)

eta = 2*LinKer(x[i],x[j]) - LinKer(x[i],x[i]) - LinKer(x[j],x[j])

if y[i] == y[j]:
    L = max(0, a[j] - a[i])
else:
    L = max(0, a[i] + a[j] - C)
if y[i] != y[j]:
    H = min(C, C - a[i] + a[j])
else:
    H = min(C, a[i] - a[j])

def E_FKT(i, x, y, a, m):
    E = 0
    j = 0
    while j in range(m):
        E = E + a[j] * y[j] * LinKer(x[i], x[j])
        j = j + 1
    return E+b-y[i]

M = np.arange(m)

#Vektor E
E = np.array([E_FKT(k, x, y, a, m) for k in M])

#Zwischenwerte
```

```
w = w-y[i]*a[i]*x[i]-y[j]*a[j]*x[j]
b1 = y[i]*a[i]*LinKer(x[i],x[i])+y[j]*a[j]*LinKer(x[i],x[j])
b2 = y[j]*a[j]*LinKer(x[j],x[j])+y[i]*a[i]*LinKer(x[j],x[i])

Eneu = np.array([E_FKT(k, x, y, a, m) for k in M])

for k in range(m):
    Eneu[k]=E[k]-y[i]*a[i]*LinKer(x[i],x[k])-y[j]*a[j]*LinKer(x[j
   ],x[k])-b
a[i]=a[i]+y[i]*y[j]*a[j]

a[j]=a[j]-(y[j]*(E[i]-E[j]))/eta

if a[j] <= L:
      a[j] = L
elif a[j] >= H:
    a[j] = H
else: a[j]=a[j]

a[i]=a[i]-y[i]*y[j]*a[j]
w=w+y[i]*a[i]*x[i]+y[j]*a[j]*x[j]
b1=b1-E[i]-y[i]*a[i]*LinKer(x[i],x[i])-y[j]*a[j]*LinKer(x[i],x[j])
   +b
b2=b2-E[j]-y[i]*a[i]*LinKer(x[j],x[i])-y[j]*a[j]*LinKer(x[j],x[j])
   +b

if a[j]>0 and a[j]<C and a[i]>0 and a[i]<C:
        if b1!=b2: exit()
        else: b=b1
elif ((a[j]==0 or a[j]==C) and (a[j]==0 or a[j]==C) and H!=L):
         b=(b1-b2)/2
else: b = b

for k in range(m):
    Eneu[k] = E[k]+y[i]*a[i]*LinKer(x[i],x[k])+y[j]*a[j]*LinKer(x[
   j],x[k])+b

for k in range(m):
    E[k] = Eneu[k]

def h(w,x,b):
    return LinKer(w,x) +b

#Plot
##Wertespanne
x1_arr=np.arange(-2,21,0.5)
x2_arr=np.arange(-2,21,0.5)

plt.plot(x_values0[:,0],x_values0[:,1],label='w$^T$x+b=0')
```

```
89 plt.plot(x_valuesplus1[:,0],x_valuesplus1[:,1],linestyle='--',
       color='g')
90 plt.plot(x_valuesminus1[:,0],x_valuesminus1[:,1],linestyle='--',
       color='g',label='w$^T$x+b=$\pm$1')
91 plt.legend(loc='upper left')
92
93 Klasse1=np.array([x[i] for i in M if y[i]==1])
94 Klasse2=np.array([x[i] for i in M if y[i]==-1])
95
96 plt.scatter(Klasse1[:,0],Klasse1[:,1])
97 plt.scatter(Klasse2[:,0],Klasse2[:,1])
98
99 maxMarge=1/math.sqrt(LinKer(w,w))
100 print('w=',w)
101 print('b=',b)
102 print('alpha=',a)
103 print('Fehlerratenvektor=',E)
104 print('Die gesuchte Hyperebene ist:',w,'x',+b,'=0')
105 print('Die Maximale Marge ist:',round(maxMarge,2))
106 plt.show()
```

4.0.2 Anhang B

Zur Berechnung des Beispiels 3.7 wurde folgender Python Code verwendet.

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits import mplot3d
3 from mpl_toolkits.mplot3d import Axes3D
4
5 import numpy as np
6 import math
7
8 #Anfangswerte
9 x = np.array([[1, 1, 1.5], [5, 3, 5]])
10 y = np.array([-1, 1])
11 m = len(x)
12 a = np.zeros(m)
13 i = 0
14 j = 1
15 C = 1
16 b = 0
17 w = 0
18
19 def LinKer(x, y):
20     return np.dot(x, y)
21
22 eta = 2 * LinKer(x[i], x[j]) - LinKer(x[i], x[i]) - LinKer(x[j], x
       [j])
23
```

```
if y[i] == y[j]:
    L = max(0, a[j] - a[i])
else:
    L = max(0, a[i] + a[j] - C)
if y[i] != y[j]:
    H = min(C, C - a[i] + a[j])
else:
    H = min(C, a[i] - a[j])

def E_FKT(i, x, y, a, m):
    E = 0
    j = 0
    while j in range(m):
        E = E + a[j] * y[j] * LinKer(x[i], x[j])
        j = j + 1
    return E+b-y[i]

M=np.arange(m)
#Vektor E
E=np.array([E_FKT(k, x, y, a, m) for k in M])

#Zwischenwerte
w = w-y[i]*a[i]*x[i]-y[j]*a[j]*x[j]
b1 = y[i]*a[i]*LinKer(x[i],x[i])+y[j]*a[j]*LinKer(x[i],x[j])
b2 = y[j]*a[j]*LinKer(x[j],x[j])+y[i]*a[i]*LinKer(x[j],x[i])

Eneu=np.array([E_FKT(k, x, y, a, m) for k in M])
for k in range(m):
    Eneu[k]=E[k]-y[i]*a[i]*LinKer(x[i],x[k]) y[j]*a[j]*LinKer(x[j
   ],x[k])-b

a[i]=a[i]+y[i]*y[j]*a[j]
a[j]=a[j]-(y[j]*(E[i]-E[j]))/eta
if a[j] < L:
      a[j] = L
elif a[j] > H:
    a[j] = H
else: a[j]=a[j]

a[i]=a[i]-y[i]*y[j]*a[j]
w=w+y[i]*a[i]*x[i]+y[j]*a[j]*x[j]
b1=b1-E[i]-y[i]*a[i]*LinKer(x[i],x[i])-y[j]*a[j]*LinKer(x[i],x[j])
   +b
b2=b2-E[j]-y[i]*a[i]*LinKer(x[j],x[i])-y[j]*a[j]*LinKer(x[j],x[j])
   +b

if a[j]>0 and a[j]<C and a[i]>0 and a[i]<C:
        if b1!=b2: exit()
        else: b=b1
```

```
70 elif ((a[j]==0 or a[j]==C) and (a[j]==0 or a[j]==C) and H!=L):
71          b=(b1-b2)/2
72 else: b=b
73
74 for k in range(m):
75     Eneu[k]=E[k]+y[i]*a[i]*LinKer(x[i],x[k])+y[j]*a[j]*LinKer(x[j
   ],x[k])+b
76 for k in range(m):
77     E[k]=Eneu[k]
78
79
80 def h(w,x,b):
81     return LinKer(w,x) +b
82
83 #Plot
84 ##Wertespanne
85 x1_arr=np.arange(-1,6,0.25)
86 x2_arr=np.arange(-1,6,0.25)
87 x3_arr=np.arange(-1,6,0.25)
88
89 ##Werte die auf der Hyperebene liegen
90 x_values0=np.array([[x1,x2,x3] for x1 in x1_arr for x2 in x2_arr
   for x3 in x3_arr if h(w,np.array([x1,x2,x3]),b)==0])
91
92 ##Werte die auf den Randhyperbenen liegen
93 x_valuesplus1=np.array([[x1,x2,x3] for x1 in x1_arr for x2 in
   x2_arr for x3 in x3_arr if h(w,np.array([x1,x2,x3]),b)-1==0])
94 x_valuesminus1=np.array([[x1,x2,x3] for x1 in x1_arr for x2 in
   x2_arr for x3 in x3_arr if h(w,np.array([x1,x2,x3]),b)+1==0])
95
96 fig = plt.figure()
97 ax = fig.add_subplot(111,projection='3d')
98 xx, yy = np.meshgrid(x1_arr, x2_arr)
99 z=(-w[0]*xx-w[1]*yy-b)/w[2]
100
101 z_plus1=(-w[0]*xx-w[1]*yy-b-1)/w[2]
102 z_minus1=(-w[0]*xx-w[1]*yy-b+1)/w[2]
103
104 Klasse1=np.array([x[i] for i in M if y[i]==1])
105 Klasse2=np.array([x[i] for i in M if y[i]==-1])
106
107 ax.scatter(Klasse1[:,0],Klasse1[:,1],Klasse1[:,2])
108 ax.scatter(Klasse2[:,0],Klasse2[:,1],Klasse2[:,2])
109
110 hyperbene=ax.plot_surface(xx, yy, z,label='w$^T$x+b=0')
111 randhyperbene1=ax.plot_surface(xx, yy, z_minus1, alpha=0.3, color
   ='g')
112 randhyperbene2=ax.plot_surface(xx, yy, z_plus1, alpha=0.3, color='
   g',label='w$^T$x+b=$\pm$1')
```

```
113
114 hyperbene._facecolors2d=hyperbene._facecolor3d
115 hyperbene._edgecolors2d=hyperbene._edgecolor3d
116 randhyperbene2._facecolors2d=randhyperbene2._facecolor3d
117 randhyperbene2._edgecolors2d=randhyperbene2._edgecolor3d
118 ax.legend(loc='upper left')
119
120 maxMarge=1/math.sqrt(LinKer(w,w))
121 print('w=',w )
122 print('b=',b )
123 print('alpha=',a)
124 print('Fehlerratenvektor=',E)
125 print('Die gesuchte Hyperebene ist:',w,'^Tx',+b,'=0')
126 print('Die Maximale Marge ist:',round(maxMarge,2))
127 plt.show()
```

Literatur

[Gir97] Edgar Osuna, Robert Freund,Federico Girosi. Improved training algorithm for support vector machines. Technical report, MIT, 1997.

[Jun15] Dieter Jungnickel. *Optmierungsmethoden-Eine Einführung.* Springer, 2015.

[Ng] Andrew Ng. Cs229 lecture notes - supervised learning. `https://sgfin.github.io/files/notes/CS229_Lecture_Notes.pdf`.

[Pla98] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research, 1998.

[Sun06] Duan Li, Xiaoling Sun. *Nonlinear Integer Programming.* Springer, 2006.

[Tal18] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar. *Foundations of Machine Learning.* The MIT Press, 2018.

Abbildungsverzeichnis

1 Beispiele für mögliche Hyperebenen 5
2 Warum wird bei gewählter Hyperbene B mit höherer Sicherheit als A klassifiziert? 5
3 Maximale geometrische Marge der gegebenen Daten 8
4 Abstände falsch klassifizierter Proben 15
5 Ausgabe des Codes für 3.6 34
6 Hyperebene mit maximaler Marge für Bsp.3.6 34
7 Ausgabe des Codes für Beispiel 3.7 35
8 Hyperebene mit maximaler Marge für Beispiel 3.7 35

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit eigenständig und ohne fremde Hilfe angefertigt habe. Ich versichere, dass ich keine anderen als die angegebenen Quellen verwendet habe, und, dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahren gewesen ist.

Datum, Ort	Unterschrift