Seminararbeit

Kurze Einführung in das maschinelle Lernen mit einem Fokus auf Support Vector Maschinen

für das Mathematische Seminar im WS2014

Harald Burgsteiner

28. Januar 2015

In dieser Seminararbeit geht es um eine kurze Einführung in die Grundlagen des maschinellen Lernens. Anfangs werden die Grundprinzipien von Klassifikatoren und deren Fehler dargestellt. Es wird besprochen, wie Hyperebenen zur Klassifikation von Daten verwendet werden und was unter einer optimalen Hyperebene verstanden wird, wie sie unter anderem von Support Vektor Machines (SVM), einer State-of-the-Art Methode zur Mustererkennung, verwendet wird. Es werden lineare und nicht-lineare SVMs vorgestellt und besprochen, wie diese linear trennbare und auch nicht linear trennbare Daten separieren können.

1 Einleitung und Einführung in die Lerntheorie

Der Forschungsbereich des „maschinellen Lernens“ beschäftigt sich laut [Mitchell, 1997] mit der Frage, wie man Computerprogramme oder genereller Maschinen konstruieren kann, die sich automatisch durch ihre eigene gewonnene Erfahrung verbessern. Die Auswirkungen davon wären immens, und es zeigen sich vielfältige Einsatzgebiete: Systeme, die anhand von Patientendaten lernen, welche Therapien die wirkungsvollsten sind, Fahrzeuge, die selbständig navigieren und sich situationserfassend verhalten können, eBook Lesegeräte,

die nach beobachteten persönlichen Bedürfnissen bestimmte Artikel aus der Zeitung vorschlagen oder hervorheben etc. Dazu gibt es viele verschiedene Ansätze, die z.B. in [Mitchell, 1997] oder [Russel and Norvig, 2012] beschrieben werden.

Eine wichtige Kategorie von Algorithmen aus dem Bereich des maschinellen Lernens bzw. der Computational Intelligence sind sogenannte Klassifizierungsmethoden. Dabei geht es darum, Beispiele anhand einer beliebigen Anzahl von Parametern einer bestimmten zugehörigen Klasse zuzuordnen. Ein einfaches Beispiel aus dem Bereich der Medizin ist, aufgrund von Blutlaborwerten das vorhanden sein einer bestimmten Krankheit zu diagnostizieren. Mathematisch formuliert besteht jedes Beispiel aus einem Paar $\langle \mathbf{a}, c \rangle \in A \times C$ wobei A die Menge der möglichen Attributwerte und C die Menge der möglichen Klassen ist, womit $A \times C := \{\langle \mathbf{a}, c \rangle | \mathbf{a} \in A \wedge c \in C\}$. Diese Paare werden hypothetisch von einer Funktion gebildet, die jedem $\mathbf{a} \in A$ ein $c \in C$ zuweist. Also

$$f := \begin{cases} A \to C \\ \mathbf{a} \mapsto c := f(\mathbf{a}) \end{cases}$$

In der Trainings- oder Lernphase steht eine Menge L mit l sogenannten Trainingsbeispielen aus $A \times C$ zur Verfügung: $L = \{\langle \mathbf{a}_1, c_1 \rangle, \langle \mathbf{a}_2, c_2 \rangle, \langle \mathbf{a}_3, c_3 \rangle, \ldots, \langle \mathbf{a}_l, c_l \rangle\}$. Eine Lernaufgabe besteht nun darin, eine Hypothese $h : A \to C$ zu konstruieren, welche die ursprüngliche Funktion f möglichst gut approximiert (oder im besten Fall mit dieser identisch ist).

Die aktuelle Qualität einer Hypothese wird häufig mit dem empirischen Fehler

$$error_{T_k}(h) := \frac{\#\{i \in 1, \ldots, k : h(\mathbf{a}_i) \neq c_i\}}{k}$$

über einer Menge T_k gemessen. Wobei

$$T_k := \{\langle \mathbf{a}_i, c_i \rangle \in A \times C : 1 \leq i \leq k\}$$

eine dem lernenden System bisher unbekannte Menge von Test- oder Validierungsdatengemessen darstellt. Dies bedeutet auch umgekehrt, dass diese Test- oder Validierungsdaten während des Trainingsprozesses nicht verwendet werden dürfen.

2 Hyperebenen und optimale Hyperebenen

2.1 Lineare Separierbarkeit

Die einfachste Form eines Klassifikators bilden die Systeme, die zur Trennung von Beispielen mit n verschiedenen Parametern (i.e. $\mathbf{a} \in A \subseteq (\mathbb{M}_1 \times \mathbb{M}_2 \times \ldots \times \mathbb{M}_n)$), Hyperebenen

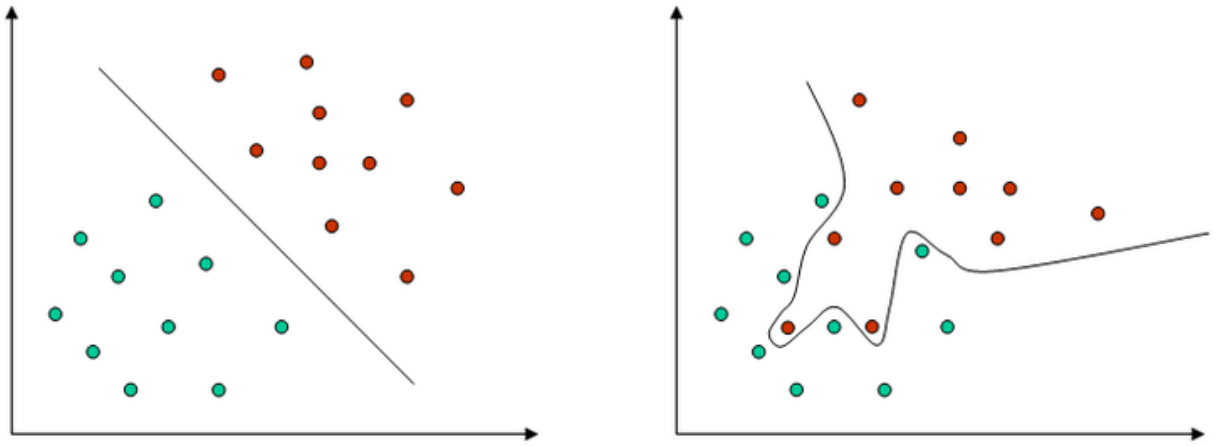

Abbildung 1: Beispiel einer linear trennbaren (links) und einer nicht linear trennbaren (rechts) Datenmenge. Die Datenpunkte auf jeder Seite der trennenden Funktion gehören zur jeweils identischen Klasse, hier mit roten und grünen Punkten gekennzeichnet.

(der Dimension $n-1$) einsetzen. Werden die Parameter $\mathbf{x} \in A = \mathbb{R}^n$ gewählt, so lässt sich eine Hyperebene geometrisch im 2-dimensionalen Fall als Gerade veranschaulichen, im 3-dimensionalen Fall tatsächlich als Ebene, oder allgemein definieren als

$$h_{\mathbf{w}} := \left\{ \mathbf{x} \mid 0 = \mathbf{x}^T \cdot \mathbf{w} + w_0 = \sum_{i=1}^{n} x_i \cdot w_i + w_0 \overset{x_0:=1}{\underset{\downarrow}{=}} \sum_{i=0}^{n} x_i \cdot w_i = \mathbf{x}^{*T} \cdot \mathbf{w}^* \right\}$$

Im Rest der Seminararbeit ist bei jeder Multiplikation von Vektoren wie z.B. $\mathbf{x} \cdot \mathbf{y}$ das Skalarprodukt gemeint, wird jedoch ohne die Transponierung des ersten Vektors geschrieben und bedeutet mathematisch somit $\mathbf{x}^T \cdot \mathbf{y}$. Für grafische Interpretationen wird w_0 auf häufig mit b, als Verschiebewert oder sogenannten Bias bezeichnet und die Hyperebene damit als $0 = \mathbf{x} \cdot \mathbf{w} + b$ definiert.

Eine Klassifikation wird dadurch erreicht, dass diese Hyperebenen eine Orientierung erhalten und (im Fall der binären Klassifikation) jede Seite der Hyperebene einer Klasse entspricht, sodass für alle Beispiele der einen Klasse $\mathbf{x} \cdot \mathbf{w} + b < 0$ und für alle Beispiele der anderen Klasse $\mathbf{x} \cdot \mathbf{w} + b > 0$ gilt. „Lernen“ bedeutet auf dieses formale Konstrukt reduziert, nun das Finden einer geeigneten Hyperbene und damit der Parameter $w_i, i = 0 \ldots n$. Eine Klassifikation kann natürlich auch mit beliebigen, nicht-linearen Funktionen erreicht werden. Das Finden geeigneter Parameter wird dabei jedoch wesentlich komplexer und aufwändiger und ist nicht mehr, wie im linearen Fall, direkt berechenbar, sondern macht den Einsatz von Näherungsverfahren wie Gradientenabstieg oder ähnliches (siehe [Haykin, 1999] oder [Bishop, 2005]) notwendig. Abbildung 1 zeigt eine grafische Veranschaulichung eines linear trennbaren und eines nicht-linear trennbaren Beispiels.

Lineare Trennbarkeit von Klassen ist eine wichtige Eigenschaft im Bereich des maschinellen Lernens, da davon die Wahl der Hypothese bzw. der angewandten (Lern-)Methode abhängt. Sind linear trennbare Klassen mit einfachen Methoden wie Perzeptronen (d.h. simplen Modellen von biologischen Neuronen, siehe z.B. [Haykin, 1999] für mehr Details) lösbar, so verlangen nicht-linear trennbare Klassen komplexere Verfahren. Im Abschnitt 5 werden wir uns mit Alternativen zu nicht-linearen Lernalgorithmen beschäftigen. Im folgenden Satz aus [Burges, 1998] wird gezeigt, dass die lineare Trennbarkeit zweier Punktemengen von der Beziehung ihrer konvexen Hüllen abhängt, was eine wichtige Grundlage für die Konstruktion von optimalen Hyperebenen darstellt.

Satz 1 (Lineare Separierbarkeit von Mengen)**.** *Zwei Mengen von Punkten aus dem $\mathbb{R}^n$ können genau dann von einer Hyperebene getrennt werden, wenn die Schnittmenge von deren konvexen Hüllen leer ist.*

Beweis. Elemente und damit Punkte aus dem $\mathbb{R}^n$ werden in diesem Beweis sowohl als Punkt als auch als Ortsvektor behandelt. Seien $\mathcal{C}_A$ und $\mathcal{C}_B$ die konvexen Hüllen der beiden disjunkten Mengen $A, B \subseteq \mathbb{R}^n$ und $A - B$ die Menge der Punkte deren Positionsvektoren durch die Differenz aller Vektoren aus A und B berechnet wird: $A - B := \{\mathbf{a} - \mathbf{b} | \mathbf{a} \in A \wedge \mathbf{b} \in B\}$. $A - B$ enthält damit nicht den Ursprung, d.h. keinen Nullvektor $\mathcal{O}$, da die beiden Mengen A und B disjunkt sind (kein Beispiel kann beiden Klassen zur gleichen Zeit angehören). $\mathcal{C}_A - \mathcal{C}_B$ hat die gleiche Bedeutung für die Elemente der konvexen Hüllen. Damit wird der Beweis, dass A und B linear trennbar (d.h. durch eine Hyperebene trennbar) sind, reduziert darauf zu zeigen, dass die Menge $A - B$ linear trennbar vom Nullpunkt bzw. Nullvektor $\mathcal{O}$ ist[1].

Wir zeigen zuerst: $\mathcal{C}_A \cap \mathcal{C}_B = \emptyset \Rightarrow \mathcal{C}_A - \mathcal{C}_B$ ist linear trennbar von $\mathcal{O}$. $\mathcal{C}_A - \mathcal{C}_B$ klarerweise nicht den Nullvektor, da die Schnittmenge von $\mathcal{C}_A$ und $\mathcal{C}_B$ laut Voraussetzung leer ist. Weiters ist $\mathcal{C}_A - \mathcal{C}_B$ konvex, denn $\forall \mathbf{x}_1 = \mathbf{a}_1 - \mathbf{b}_1, \mathbf{x}_2 = \mathbf{a}_2 - \mathbf{b}_2, \lambda \in [0,1], \mathbf{a}_1, \mathbf{a}_2 \in \mathcal{C}_A, \mathbf{b}_1, \mathbf{b}_2 \in \mathcal{C}_B$ ist $(1-\lambda)\mathbf{x}_1 + \lambda\mathbf{x}_2 = ((1-\lambda)\mathbf{a}_1 + \lambda\mathbf{a}_2) - ((1-\lambda)\mathbf{b}_1 + \lambda\mathbf{b}_2) \in \mathcal{C}_A - \mathcal{C}_B$. Darum reicht es zu zeigen, dass jede konvexe Menge S, die den Nullvektor nicht enthält, vom Nullvektor linear trennbar ist. Sei deshalb $\mathbf{x}_{min} \in S$ derjenige Punkt aus S, dessen euklidischer Abstand $\|\mathbf{x}_{min}\|$ zum Nullvektor minimal ist[2]. Wir zeigen, dass $\forall \mathbf{x} \in S, \mathbf{x} \cdot \mathbf{x}_{min} > 0$. Nehmen wir an, $\exists \mathbf{x} \in S, \mathbf{x} \cdot \mathbf{x}_{min} \leq 0$. Sei dann L die Gerade, die $\mathbf{x}$ und $\mathbf{x}_{min}$ verbindet. Die Konvexität von

[1] Denn nehmen wir an, dass $A - B$ von $\mathcal{O}$ durch eine Hyperebene trennbar ist. Dann $\exists \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}, b < 0$, so dass $\forall x \in A - B : \mathbf{x} \cdot \mathbf{w} + b > 0$. Wählt man nun ein beliebiges $\mathbf{y} \in B$ und nennt $A - B + \mathbf{y}$ die Menge aller Punkte $\mathbf{a} - \mathbf{b} + \mathbf{y}, \mathbf{a} \in A, \mathbf{b} \in B$, dann ist $\forall \mathbf{x} \in A - B + \mathbf{y} : \mathbf{x} \cdot \mathbf{w} + b > \mathbf{y} \cdot \mathbf{w}$ und natürlich auch $\mathbf{y} \cdot \mathbf{w} + b < \mathbf{y} \cdot \mathbf{w}$. Daher ist die Menge $A - B + \mathbf{y}$ und $\mathbf{y}$ linear trennbar. Wiederholt man diesen Vorgang nun für alle $\mathbf{y} \in B$, zeigt sich, dass $A - B$ genau dann linear trennbar vom Nullvektor $\mathcal{O}$ ist, wenn A und B linear trennbar sind.

[2] Es kann nur einen solchen Punkt geben, denn gäbe es zwei, so würde jeder Punkt der Verbindungsgeraden zwischen diesen beiden Punkten noch näher zum Nullvektor liegen. Die Verbindungsgerade ist ja ebenfalls Teil der Menge S, da diese konvex ist.

S verlangt, dass natürlich auch $L \subset S$. Daher ist $\mathcal{O} \notin L$, weil laut Voraussetzung $\mathcal{O} \notin S$. Weil das Skalarprodukt negativ ist, bilden die drei Punkte $\mathcal{O}, \mathbf{x}$ und $\mathbf{x}_{min}$ ein stumpfes oder höchstens rechtwinkeliges Dreieck, bei dem dieser stumpfe bzw. rechte Winkel im Ursprung $\mathcal{O}$ liegt[3]. Sei nun $\mathbf{n} := (\mathbf{x} - \mathbf{x}_{min}) / \|\mathbf{x} - \mathbf{x}_{min}\|$. Dann ist der Abstand vom nächstgelegenen Punkt von L zu $\mathcal{O}$ gleich $\|\mathbf{x}_{min}\|^2 - (\mathbf{x}_{min} \cdot \mathbf{n})^2$, was kleiner wäre als $\|\mathbf{x}_{min}\|^2$. Daher ist $\mathbf{x} \cdot \mathbf{x}_{min} > 0$ und S linear trennbar von $\mathcal{O}$. Darum ist auch schließlich auch $\mathcal{C}_A - \mathcal{C}_B$ und ebenso $A - B$ linear trennbar von $\mathcal{O}$ und daher ist A linear trennbar von B.

Es bleibt noch die andere Richtung zu zeigen, dass, wenn zwei Punktemengen A und B linear separierbar sind, dass dann auch die Schnittmenge der konvexen Hüllen leer ist. Unter dieser Prämisse $\exists \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}$, so dass $\forall \mathbf{a}_i \in A, 1 \leq i \leq |A| : \mathbf{w} \cdot \mathbf{a}_i + b > 0$ und $\forall \mathbf{b}_i \in B, 1 \leq i \leq |B| : \mathbf{w} \cdot \mathbf{b}_i + b < 0$. Betrachtet man einen beliebigen Punkt $\mathbf{x} \in \mathcal{C}_A$, so kann dieser auch geschrieben werden als $\mathbf{x} = \sum_i \lambda_i \mathbf{a}_i, \sum \lambda_i = 1, 0 \leq \lambda_i \leq 1$. Dann ist $\mathbf{w} \cdot \mathbf{x} + b = \mathbf{w} \cdot (\sum_i \lambda_i \mathbf{a}_i) + b = \sum_i (\lambda_i \mathbf{w} \cdot \mathbf{a}_i + b) = \sum_i \lambda_i (\mathbf{w} \cdot \mathbf{a}_i + b) > 0$. Ebenso folgt für alle Punkte $\mathbf{y} \in \mathcal{C}_B$, dass $\mathbf{w} \cdot \mathbf{y} + b < 0$. Daraus folgt, dass $\mathcal{C}_A \cap \mathcal{C}_B = \emptyset$, nachdem wir sonst einen Punkt $\mathbf{x} = \mathbf{y}$ finden könnten, der beide Ungleichungen erfüllt.

□

2.2 Optimale Hyperebene

Sind zwei Punktemenge linear separierbar, so kann sehr leicht eine (beliebige) trennende Hyperebene direkt als Lösung eines linearen Optimierungsproblems berechnet werden. Schwieriger wird dieses Problem, wenn man nicht eine beliebige, sondern eine optimale Hyperebene sucht. Optimal bedeutet hier, dass die Hyperebene nicht nur die beiden Mengen trennen soll, sondern gleichzeitig der beiderseitige Abstand von der Hyperebene zu den nächstgelegenen Punkten aus den Mengen A und B maximiert werden soll. Abbildung 2 veranschaulicht dieses Kriterium. Das Finden einer optimalen Hyperebene ist das eigentliche Prinzip einer Support Vector Machine (SVM). Im nächsten Abschnitt betrachten wir zunächst den Fall, eine optimale lineare Hyperebene zu finden.

3 Lineare Support Vector Machines

Die Trainingsbeispiele sind eine Menge mit l verschiedenen Paaren $\langle \mathbf{x}_i, y_i \rangle, i = 1, \ldots, l$, mit $y_i \in \{-1, 1\}, \mathbf{x}_i \in \mathbb{R}^n$. Wir beschränken uns hier wieder auf ein binäres Klassifikationsproblem mit den Klassenbezeichnungen „-1“ und „1“. Ein Verallgemeinerung auf beliebig viele Klassen, ist ohne weiteres durch Hinzunahme weiterer Hyperebenen möglich. Eine trennende Hyperebene $\mathbf{x} \cdot \mathbf{w} + b = 0$ separiert die Datenpunkte beider Klassen. Punkte

[3] Wegen $cos\alpha = \frac{\mathbf{x} \cdot \mathbf{x}_{min}}{\|\mathbf{x}\| \cdot \|\mathbf{x}_{min}\|}$

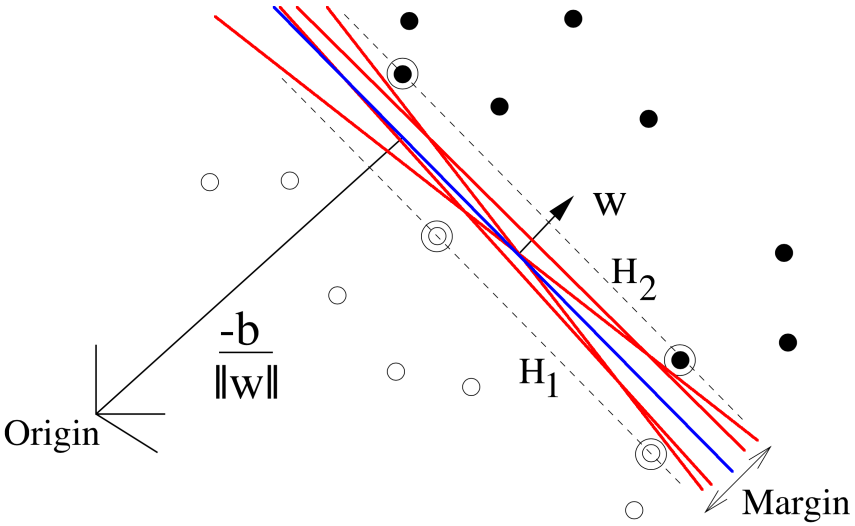

Abbildung 2: Beispiel zweier linear trennbarer Datenmengen (in Anlehnung an [Burges, 1998]). Die roten Linien zeigen beliebige korrekte Hyperebenen, die blauen Linie zeigt die optimale Hyperebene, für die der Abstand („Margin“) zu beiden Seiten der Hyperebene bezüglich der beiden Datenmengen maximiert wurde.

$x \in \mathbb{R}^n$, die diese Gleichung erfüllen, liegen genau auf der Hyperebene, der Vektor $\mathbf{w}$ steht normal auf die Ebene und $|b|/\|w\|$ beschreibt den Normalabstand der Hyperebene zum Ursprung. Seien nun d_+ und d_- die kürzesten Normalabstände eines positiven ($y_i = 1$) bzw. negativen ($y_i = -1$) Beispiels zur Hyperebene. Der minimale Abstand zwischen positiven und negativen Beispielen und damit der sogenannte „Margin“ ist damit $d_+ + d_-$. Im linearen Fall, sucht der SVM-Algorithmus nun einfach nach den Beispielen aus der Trainingsdatenmenge, für die der Margin maximiert wird. Das kann mathematisch wie folgt definiert werden. Nehmen wir an, für alle Beispiele der Trainingsmenge gilt:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \forall y_i = +1 \tag{1}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \forall y_i = -1 \tag{2}$$

Die beiden Ungleichungen können auch zusammengefasst werden zu

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \tag{3}$$

Beispiele, die die Gleichheit erfüllen, liegen exakt auf den parallelen Ebenen H_1 oder H_2 wie in Abbildung 2 dargestellt. Der Normalabstand dieser Punkte auf H_1 und H_2 zum Ursprung, ist $|1 - b|/\|w\|$ bzw. $|-1 - b|/\|w\|$. Der Margin um die Hyperebene beträgt somit $d_+ + d_- = 1/\|w\| + 1/\|w\| = 2/\|w\|$. Die Hyperebene mit maximalem Margin kann

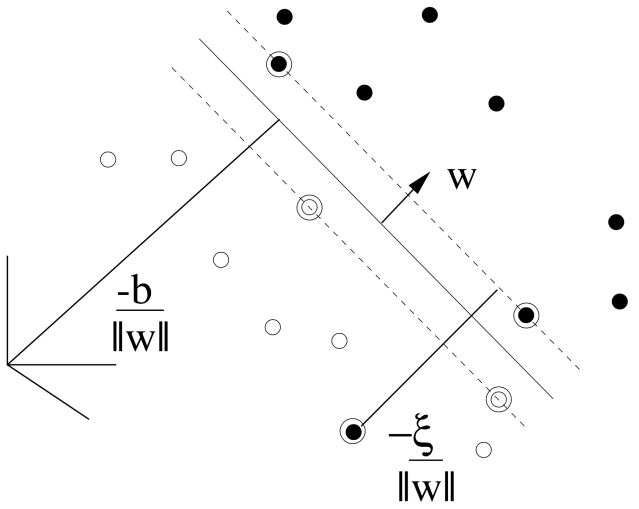

Abbildung 3: Erweiterung der obigen Abbildung um einen Datenpunkt der verhindert, dass eine Menge linear separierbar ist. Für dieses Beispiel ist der Strafterm $\xi_i > 0$, für alle anderen ist $\xi_i = 0$.

somit gefunden werden, in dem $\|w\|$ minimiert wird. In Abbildung 2 sind auch spezielle Beispiele markiert, die exakt auf H_1 bzw. H_2 liegen. Dies sind die sogenannten „Support Vektoren“, die, genauer betrachtet, die einzig wichtigen und notwendigen Beispiele aus der Trainingsdatenmenge darstellen. Diese Support Vektoren werden vom SVM-Algorithmus damit implizit gefunden.

In der Praxis wird nicht versucht, $\|w\|$ direkt unter Berücksichtigung von Ungleichung 3 zu minimieren. Vielmehr wird die äquivalente Formulierung mit Lagrange-Multiplikatoren verwendet. Das bietet zwei Vorteile. Zum einen kann dadurch eine bekannte Standardmethode zur Lösung von Optimierungsproblemen mit Randbedingungen eingesetzt werden. Zum anderen führt dies zu einer sehr ähnlichen Formulierung für den nicht-linearen Fall in Abschnitt 6. Man erhält

$$L_P = \frac{\|w\|^2}{2} - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{4}$$

was nun minimiert werden soll. Details dazu würden den Rahmen dieser Seminararbeit sprengen und können bei Bedarf in [Haykin, 1999] oder [Burges, 1998] nachgelesen werden.

4 Support Vector Machines bei nicht separierbaren Daten

Im Fall, wenn die Beispiele nicht linear trennbar sind, scheitert das Optimierungsverfahren aus Abschnitt 3. Es wird keine Lösung gefunden, da die Randbedingungen zu scharf

formuliert sind. Die beiden Bedingungen in den Ungleichungen 1 und 2 müssen deshalb erweitert werden, um für die Beispiele, die auf der falschen Seite der Hyperebene liegen, einen zusätzlichen positiven Strafterm $\xi_i, i = 1, \ldots, l$ verwenden zu können (für alle korrekten sind die $\xi_i = 0$). Die Bedingungen können damit geschrieben werden als

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \xi_i \quad \forall y_i = +1 \tag{5}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + \xi_i \quad \forall y_i = -1 \tag{6}$$

$$\xi_i \geq 0 \quad \forall i \tag{7}$$

Abbildung 3 zeigt den Fall, dass ein Beispiel auf der falschen Seite der Hyperebene liegt. In diesem Fall sind alle Strafterme $\xi_i = 0$, bis auf den des falsch gelegenen Beispiels. Damit kann der Großteil des Optimierungsverfahrens ähnlich definiert werden, wie im linearen Fall. Eine notwendige Erweiterung ist, statt lediglich $\|w\|$ zu minimieren, den Term zu ergänzen auf $\|w\| + C \cdot (\sum_{i=1}^{l} \xi_i)$, wobei C ein vom Benutzer gewählter Parameter ist. C bestimmt, wie stark eine falsche Klassifizierung im Verhältnis zur Optimalität der Hyperebene gewichtet wird. Mit Lagrange-Multiplikatoren kann die Optimierung geschrieben werden als

$$L_P = \frac{\|w\|^2}{2} + C \cdot \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i \{y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i\} - \sum_{i=1}^{l} \mu_i \xi_i \tag{8}$$

wobei die μ_i zusätzliche Lagrange-Multiplikatoren sind, um die Positivität der ξ_i zu erzwingen. Für weitere Details sei wieder auf [Haykin, 1999] oder [Burges, 1998] verwiesen. Als Lösung erhält man wie im linearen Fall den Vektor

$$\mathbf{w} = \sum_{i=1}^{N_S} \alpha_i y_i \mathbf{x}_i$$

wobei N_S die Anzahl der Support-Vektoren ist, die gefunden wurden. Alle anderen Vektoren $\mathbf{x}_i$ aus den Trainingsdaten spielen keine Rolle. Für neue, unbekannte Daten kann diese Hyperebene nun verwendet werden, um die Klasse dieser Daten zu bestimmen. Dazu wird berechnet, auf welcher Seite der Hyperebene, die von $\mathbf{w}$ und b beschrieben wird, $\mathbf{x}$ liegt. Die Klassifikation unbekannter Daten $\mathbf{x}$ wird damit reduziert auf die Berechnung von

$$y = sgn(\mathbf{w} \cdot \mathbf{x} + b).$$

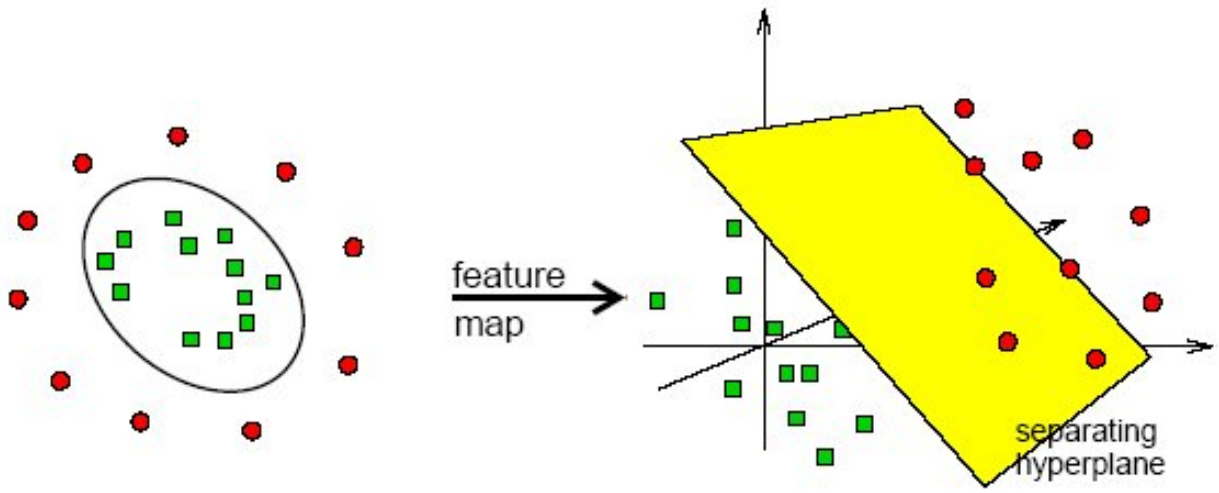

Abbildung 4: Die Idee hinter Cover's Theorem: sollte eine Datenmenge in ihrem ursprünglichen Raum nicht-linear separierbar sein, kann ein nicht-lineares, sogenanntes Feature-Mapping in einen höher dimensionalen Raum gefunden werden, in dem die Daten linear trennbar sind.

5 Cover's Theorem

In vielen Bereichen des maschinellen Lernens wird versucht zu vermeiden, nicht-lineare Funktionen einzusetzen, um nicht-linear trennbare Daten trennen zu können. Der Hauptgrund dafür ist, das nicht-lineare Optimierungsprobleme inhärent schwieriger zu lösen sind, als lineare. Während für lineare direkte Berechnungsmodelle (z.B. quadratische Programmierung, Newton-Verfahren etc.) verwendet werden können, stehen für nicht-lineare Optimierungen lediglich Näherungsverfahren zur Verfügung, bei denen nicht garantiert werden kann, auch tatsächlich ein globales Minimum zu finden. Statt dessen wird häufig nicht-lineares Mapping der Daten in einen höher dimensionalen Raum durchgeführt, in dem anschließend versucht wird, wiederum eine lineare Hyperebene zu finden. Cover's Theorem besagt:

> »A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.«
>
> — Cover, T.M., 1965

Abbildung 4 veranschaulicht die Idee dahinter. Sind die Beispiele ursprünglich beispielsweise Paare $\langle \mathbf{x}, y \rangle$ mit $\mathbf{x} \in \mathbb{R}^n$ und $y \in \mathbb{R}$, so kann eine nicht-lineare Mappingfunktion φ

in den $\mathbb{R}^m, m > n$ wie folgt definiert werden:

$$\varphi := \begin{cases} \mathbb{R}^n \to \mathbb{R}^m \\ \mathbf{x} \mapsto \begin{pmatrix} \varphi_1(\mathbf{x}) \\ \varphi_2(\mathbf{x}) \\ \vdots \\ \varphi_m(\mathbf{x}) \end{pmatrix} \end{cases}$$

Für die Wahl der Funktionen φ_i gibt es prinzipiell keine Einschränkungen, außer der Forderung nach Nicht-Linearität. Ein simples Beispiel wäre

$$\varphi := \begin{cases} \mathbb{R}^3 \to \mathbb{R}^4 \\ \mathbf{x} \mapsto \begin{pmatrix} e^{x_1+x_2+x_3} \\ sinh(x_3 x_2) \\ (x_1^2 - x_2^2) \cdot x_3 \\ x_1^{x_2} \end{pmatrix} \end{cases}$$

In der Praxis werden für alle i meist Funktionen gewählt, die sich lediglich um einen bestimmten Parameter $\mathbf{t}_i$ voneinander unterscheiden. Häufig wird dafür z.B. die Gaußsche Funktion mit Zentren an paarweise verschiedenen $\mathbf{t}_i$ eingesetzt:

$$\varphi_i(\mathbf{x}) = exp(-\frac{\|\mathbf{x} - \mathbf{t}_i\|}{2\sigma^2})$$

6 Nicht-lineare Support Vector Machines

Die Optimierung von tatsächlich optimalen, nicht-linearen Funktionen, würde sich äußerst schwierig gestalten. Statt dessen verwendet man die Idee von Cover's Theorem und projiziert die Attribute der Beispiele wieder mit einer Funktion $\varphi : \mathbb{R}^n \to \mathcal{H}$ in einen beliebig hochdimensionalen (auch unendlich dimensionalen) Hilbertraum $\mathcal{H}$ und optimiert dort wieder eine (lineare) Hyperebene.

Im Rahmen der Berechnung der Lagrange-Multiplikatoren einer gewöhnlichen, linearen SVM, ist es an einer Stelle notwendig, den Ausdruck

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

über alle Beispiele i, j zu maximieren. Dies ist die einzige Stelle im Algorithmus, an der direkt die Vektoren $\mathbf{x_i}$ und $\mathbf{x}_j$ vorkommen. Hier werden nun diese Vektoren durch deren

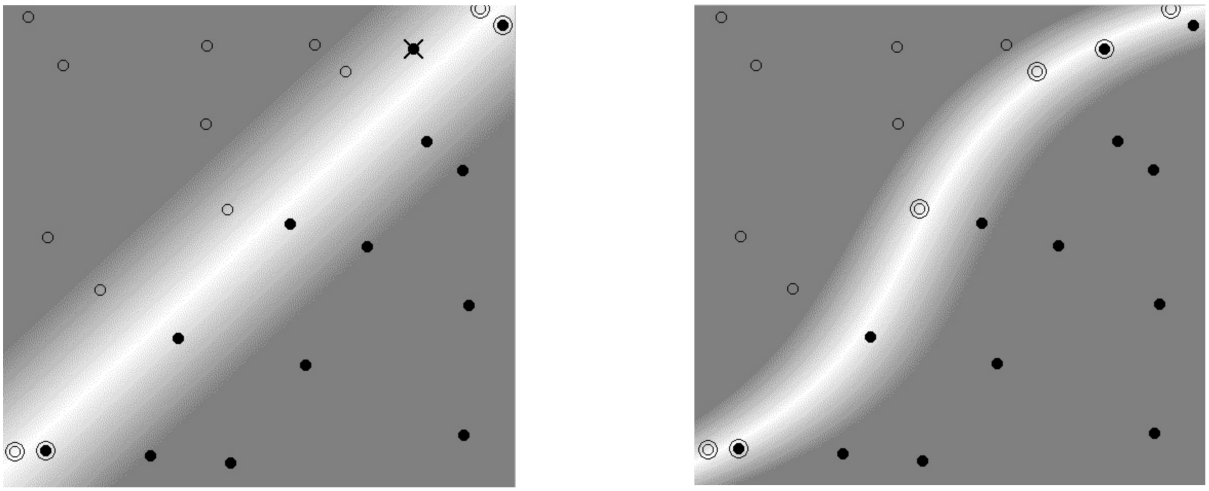

Abbildung 5: Vergleich des Ergebnisses der Berechnung zweier SVMs für linear nicht trennbare Daten. Links, eine gefundene Hyperebene im ursprünglichen Datenraum unter Zulassung von Fehlern durch Strafterme. Rechts, eine das Ergebnis einer nicht-linearen SVM unter Verwendung eines polynomiellen Kernels vom Grad 3 (aus [Burges, 1998]).

Projektion $\varphi(\mathbf{x_i})$ und $\varphi(\mathbf{x}_j)$ ersetzt:

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \varphi(\mathbf{x_i}) \varphi(\mathbf{x}_j)$$

Mathematisch betrachtet ist dies jedoch eine höchst aufwändige Berechnung: Zwei Vektoren werden in (unendlich) hoch-dimensionale Räume projiziert und dort das Ergebnis skalar miteinander multipliziert. Das Ergebnis dieser gesamten Berechnung ist jedoch „nur“ ein Skalar. Wenn es nun eine „Kernel-Funktion“ $K(\mathbf{x_i}, \mathbf{x}_j) := \varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x}_j)$ gäbe, könnte man diese im Berechnungsalgorithmus einbauen und müsste nie explizit die Projektionen oder das Skalarprodukt berechnen. Man müsste nicht einmal die Projektionsfunktionen selbst kennen. Tatsächlich können sehr viele Funktionen als Kernel verwendet werden, die bestimmte Konditionen erfüllen („Mercer's Condition“, siehe [Burges, 1998]). Typische Kernelfunktionen, wie sie im Bereich der Pattern-Recognition häufig verwendet werden, sind z.B.

$$\begin{aligned}
K(\mathbf{x}, \mathbf{y}) &= (\mathbf{x} \cdot \mathbf{y} + 1)^p && \ldots \text{ polynomielle Funktion vom Grad } p \\
K(\mathbf{x}, \mathbf{y}) &= e^{\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} && \ldots \text{ Gaußscher Kernel} \\
K(\mathbf{x}, \mathbf{y}) &= tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta) && \ldots \text{ spezielles Multi-Layer Perzeptron}
\end{aligned}$$

Wird nun eine SVM unter Verwendung einer Kernelfunktion erzeugt, erhält man eine Hyperebene – nur ist diese im hochdimensionalen Raum $\mathcal{H}$. Auch in diesem Fall ergibt

sich ähnlich wie im linearen Fall als Lösung für die Ebene

$$\mathbf{w} = \sum_{i=1}^{N_S} \alpha_i y_i \varphi(\mathbf{s}_i) \tag{9}$$

wobei N_S die Anzahl der Support-Vektoren ist und die $\mathbf{s}_i$ die dazugehörigen Support-Vektoren sind, die beim Training gefunden wurden. Der Unterschied bei der Klassifikation von neuen, unbekannten Daten ist lediglich, dass hier ebenfalls die Projektionen $\varphi(\mathbf{x})$ der Vektoren $\mathbf{x}$ verwendet werden. Wird die Lösung für $\mathbf{w}$ aus Gleichung 9 nun in die Funktion der Hyperebene eingesetzt, so wird die Klasse y von neuen, bislang unbekannten Daten $\mathbf{x}$, berechnet mit

$$y = sgn(\mathbf{w} \cdot \varphi(\mathbf{x}) + b) = sgn\left(\sum_{i=1}^{N_S} \alpha_i y_i \varphi(\mathbf{s}_i) \cdot \varphi(\mathbf{x}) + b\right) = sgn\left(\sum_{i=1}^{N_S} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b\right)$$

Also auch hier werden die Mappingfunktionen φ nicht in expliziter Form benötigt, sondern es kann statt dessen wieder die Kernelfunktion verwendet werden. Abbildung 5 zeigt einen Vergleich zwischen einer gefundenen Hyperebene bei nicht-separierbaren Daten (im ursprünglichen Raum) und einer rückprojizierten Hyperebene aus dem hochdimensionalen Raum unter Verwendung eines polynomiellen Kernels vom Grad 3.

7 Literatur

Literatur

Bishop, C. M. (2005). *Neural Networks for Pattern Recognition.* Oxford University Press.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167.

Haykin, S. (1999). *Neural Networks - A Comprehensive Foundation.* Prentice-Hall, 2nd edition.

Mitchell, T. (1997). *Machine Learning.* McGraw-Hill.

Russel, S. and Norvig, P. (2012). *Artificial Intelligence - A Modern Approach.* Prentice-Hall, 3rd edition.