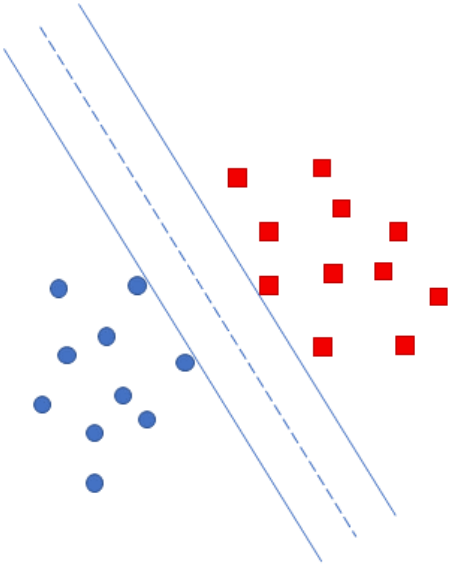
Support Vector Machines



Kurs: Intelligente Agenten und Multiagentensysteme
Dozent: Prof. Dr. Homberger

Ausarbeitung von:
Ernesto Elsäßer
Markus Lippert
Matthias Hofmann

DHBW CAS
Duale Hochschule
Baden-Württemberg
Center for Advanced Studies

Inhaltsverzeichnis

1. Einleitung

Die Support Vector Machines sind ein Teil des Forschungsgebietes der künstlichen Intelligenz, genauer des Maschinellen Lernens. Das Ziel einer Support Vector Machine ist die Klassifizierung einer Menge von Objekten. Dies wird erreicht, indem man mit Hilfe eines mathematischen Verfahrens der Mustererkennung und sogenannter Trainingsdaten (Datensatz mit Eigenschaften und einer Klassifizierung) eine Klassengrenze festlegt. Eine optimale Trennung der Klassen ist gegeben, wenn der Bereich zwischen den Klassen maximal ist.

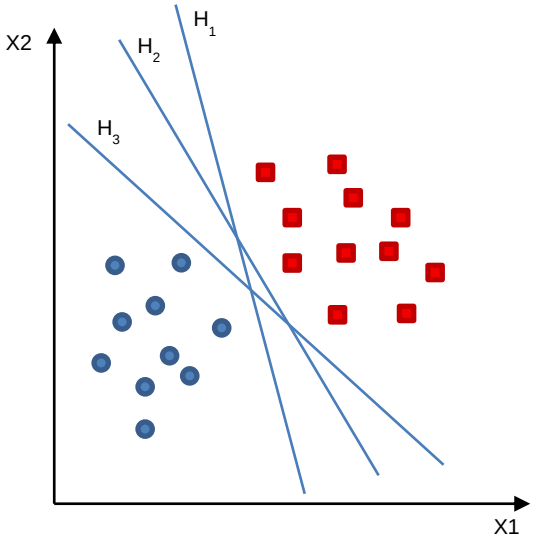


Abb. 1: Mögliche Klassengrenzen

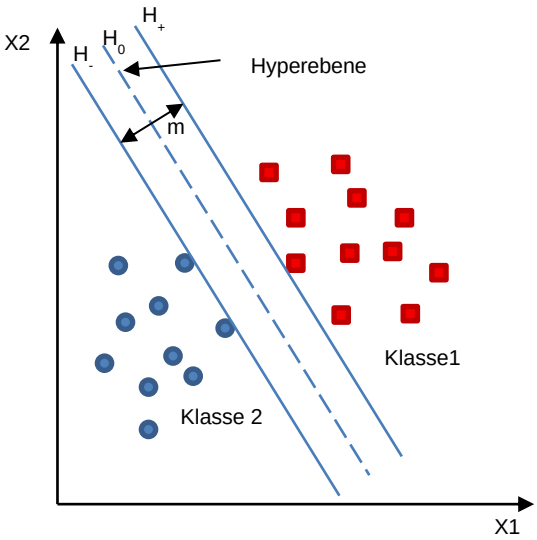


Abb. 2: Optimale Hyperebene

2. Grundlagen

Optimale Hyperebene

Projiziert man den zu klassifizierenden Vektor (Datenpunkt) $\vec{u}$, mit Hilfe des Skalarprodukts, auf den zu einer optimalen Hyperebene orthogonalen Vektor $\vec{\omega}$, erhält man den Betrag (Länge) von $\vec{\omega}$. Ist diese Länge größer als eine Konstante C wird dieser Datenpunkt der Klasse 1 (Positives Objekt/Roter Datenpunkt) zugeordnet.

$$\vec{\omega} * \vec{u} \geq C \qquad 1.0$$

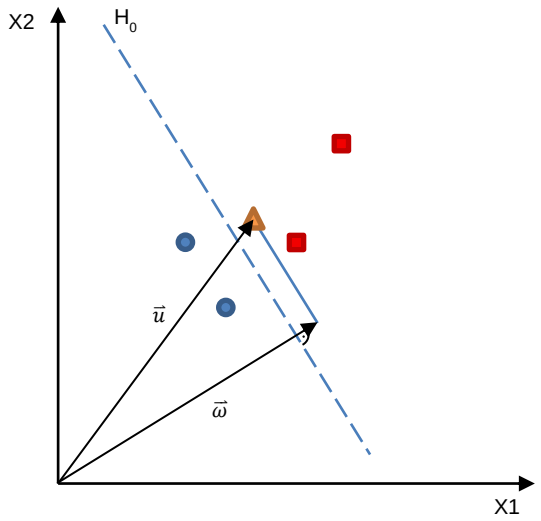


Abb. 3: Datenpunkt klassifizieren

Kennt man nun die Verschiebung b der Hyperebene zum Nullpunkt, lässt sich daraus folgende Entscheidungsregel für die Klasse 1 ableiten:

$$\vec{\omega} * \vec{u} + b \geq 0 \qquad für\ C = -b \qquad 1.1$$

Legt man nun die Regeln fest, dass positive Datenpunkte größer oder gleich eins und negative Datenpunkte kleiner oder gleich minus eins sein müssen

$$\vec{\omega} * \overrightarrow{x_+} + b \geq +1 \qquad 1.2$$
$$\vec{\omega} * \overrightarrow{x_-} + b \leq -1 \qquad 1.3$$

und führt eine Variable y_i ein,

$$y_i = +1 \qquad für\ pos. Datenpunkte \qquad 1.4$$
$$y_i = -1 \qquad für\ neg. Datenpunkte \qquad 1.5$$

lassen sich die Gleichung durch Multiplikation von y_i zusammenfassen:

$$y_i(\vec{\omega} * \vec{x}_i + b) - 1 \geq 0 \qquad 1.6$$

Weiterhin wird festgelegt, dass für x_i im Bereich der „Straße“ (zwischen H_+ und H_-) gelten muss:

$$y_i(\vec{\omega} * \vec{x}_i + b) - 1 = 0 \qquad 1.7$$

Die Entfernung m zwischen den Hilfsebenen H_+ und H_- erhält man, indem das Skalarprodukt aus der Entfernung der Support Vektoren (x_+ und x_-) und dem normierten Vektor $\vec{\omega}$ bildet.

$$m = (\overrightarrow{x_+} - \overrightarrow{x_-}) * \frac{\vec{\omega}}{|\omega|} \qquad 1.8$$

Setz man 1.4 bzw. 1.5 in 1.7 ein wird ersichtlich,

$$x_+ = 1 - b \qquad f\ddot{u}r\ y_i = +1 \qquad 1.9$$
$$x_- = -(1 + b) \qquad f\ddot{u}r\ y_i = -1 \qquad 1.10$$

Aus 1.8 folgt somit mit 1.9 und 1.10

$$m = \frac{2}{|\omega|} \qquad 1.11$$

Um die bestmögliche Hyperebene zu erhalten muss die Entfernung m maximiert werden, was bedeutet, dass ω minimiert werden muss.

$$MAX\frac{2}{|\omega|} = MIN\ |\omega| = MIN\frac{1}{2}\ |\omega|^2 \qquad 1.12$$

Unter Verwendung der Lagrange-Methode kann das Minimierungsproblem wie folgt gelöst werden:

$$L = \frac{1}{2}\ |\omega|^2 - \sum \alpha_i[y_i(\vec{\omega} * \vec{x}_i + b) - 1] \qquad 1.13$$

$$\frac{\partial L}{\partial \vec{\omega}} = \vec{\omega} - \sum_i \alpha_i y_i x_i = 0 \rightarrow \vec{\omega} = \sum_i \alpha_i y_i x_i \qquad 1.14$$

$$\frac{\partial L}{\partial \vec{b}} = -\sum_i \alpha_i y_i = 0 \rightarrow \sum_i \alpha_i y_i = 0 \qquad 1.15$$

Setzt man 1.14 und 1.15 in 1.13 ein, erhält man nach Vereinfachungen, eine quadratische Gleichung (duales Problem):

$$L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j (\vec{x}_i * \vec{x}_j) \qquad 1.16$$

$$0 \leq \alpha_i \leq C, \forall i$$

$$\sum_i \alpha_i y_i = 0$$

Auch die Entscheidungsregel lässt sich jetzt folgendermaßen darstellen:

$$\sum_i \alpha_i y_i (\vec{x}_i * \vec{u}) + b \geq 0 \qquad 1.17$$

Das duale Problem lässt sich mit Hilfe numerischer Methoden lösen, indem alle $\alpha_i's$ maximiert werden.

Kernelfunktionen

In den meisten Fällen sind reelle Trainingsdaten nicht linear trennbar. Aus diesem Grund muss eine neue, nichtlineare, Klassifizierungsregel gefunden werden. Die Merkmalsvektoren $\vec{x}_i$ müssen mit einer Transformationsfunktion $\Phi(\vec{x}_i)$ in einen hochdimensionalen Raum abgebildet werden. In diesem hochdimensionalen Raum sind die Trainingsdaten dann wieder linear separierbar.

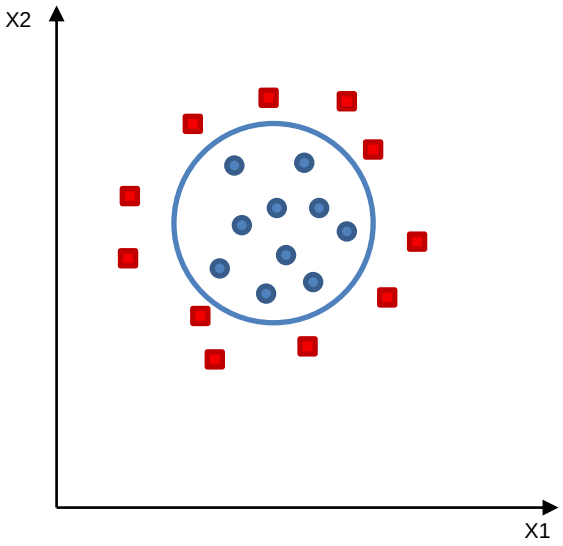


Abb. 4: Daten im Eingaberaum

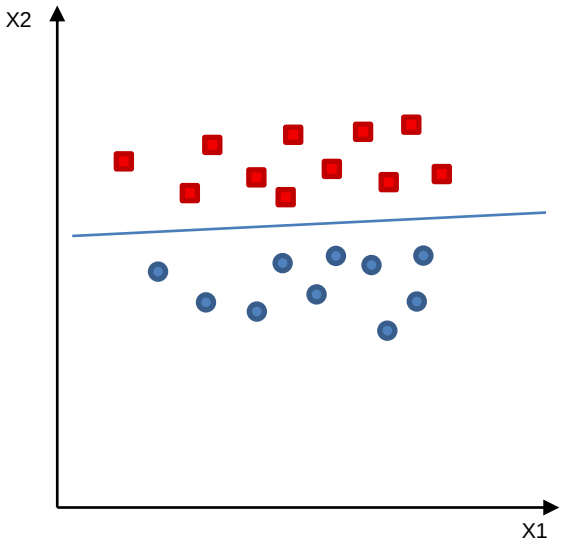


Abb. 5: Transformierte Daten

Gibt es eine Kernelfunktion $K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i) * \Phi(\vec{x}_j)$, ist es nicht nötig die Transformationsfunktion $\Phi(\vec{x}_i)$ zu berechnen. Das duale Problem lässt sich dann wie folgt darstellen:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j) \qquad 1.18$$

Gängige Kernelfunktionen:

- Linear: $K(\vec{x}_i, \vec{x}_j) = \vec{x}_i * \vec{x}_j$
- Polynomial: $K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i * \vec{x}_j + 1)^d$
- RBF (Gauß): $K(\vec{x}_i, \vec{x}_j) = e^{(-\frac{|x-y|^2}{2\sigma^2})}$

Sequential minimal optimization (SMO)

Zur Lösung des dualen Problems (1.16) wird die Sequential minimal optimization verwendet. Der Algorithmus optimiert immer zwei $\alpha_i's$, während die restlichen $\alpha_i's$ konstant bleiben. Die $\alpha_i's$, welche optimiert werden, werden mit einer Heuristik (vgl. [1]) gewählt. Dieser Prozess wird wiederholt bis die $\alpha_i's$ konvergieren.

Aus der Bedingung $\sum \alpha_i y_i = 0$ kann die Bedingung abgeleitet werden, dass die Summe der optimierten $\alpha_i's$ gleich der Summe der alten $\alpha_i's$ sein muss. Weiter muss gelten $0 \leq \alpha_i \leq C$. Dies lässt sich für den zweidimensionalen Fall visuell darstellen:

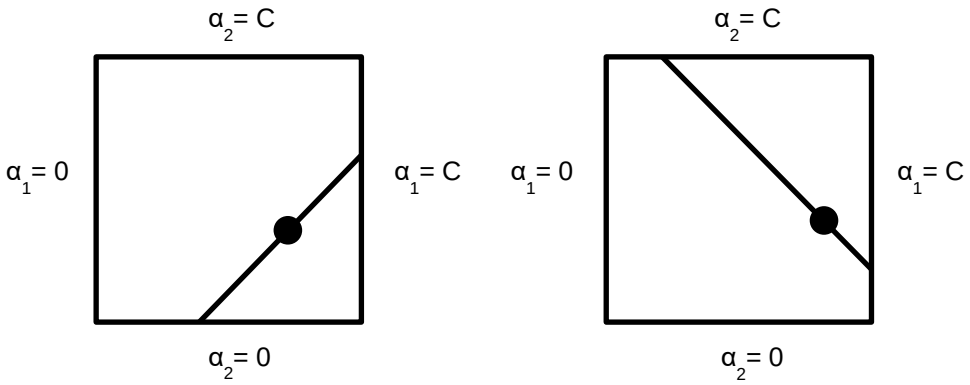


Abb. 6: SMO Optimierung [1]

Aus Abb. 6 lässt sich ablesen, dass

für $y_1 \neq y_2$

$$L = \max(0, \alpha_2 - \alpha_1), H = \min(C, C + \alpha_2 - \alpha_1) \quad 1.19$$

und für $y_1 = y_2$

$$L = \max(0, \alpha_2 + \alpha_1 - C), H = \min(C, \alpha_2 + \alpha_1) \quad 1.20$$

gelten muss.

Zur Bestimmung des Minimums von α_2 wird die zweite Ableitung von (1.16) benötigt. Nach algebraischen Umformungen (vgl. [1]) und unter Berücksichtigung des Fehlers $E_i = u_i - y_i$ ergibt sich

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{K(\vec{x}_1, \vec{x}_1) + K(\vec{x}_2, \vec{x}_2) - 2K(\vec{x}_1, \vec{x}_2)} \quad 1.21$$

Dieses α_2^{new} berücksichtigt die Nebenbedingungen (siehe oben) nicht. Deshalb muss α_2^{new} wie folgt begrenzt wurden:

$$\alpha_2^{new,beg.} = \begin{cases} H & \text{für } \alpha_2^{new} \geq H \\ \alpha_2^{new} & \text{für } L < \alpha_2^{new} < H \\ L & \text{für } \alpha_2^{new} \leq L \end{cases} \quad 1.22$$

Danach lässt sich α_1^{new} berechnen:

$$\alpha_1^{new} = \alpha_1 + y_1 y_2 (\alpha_2 - \alpha_2^{new,beg.}) \quad 1.22$$

[1] John C. Platt - Sequential Minimal Optimization

Evolutionärer Algorithmus

Ein weiterer Ansatz zur Lösung des dualen Problems (1.16) sind evolutionäre Algorithmen. Im Gegensatz zur Sequential minimal optimization werden hier alle $\alpha_i's$ auf einmal optimiert. Nachfolgend soll eine einfache Form, eines evolutionären Algorithmus, näher beschrieben werden.

Für den einfachen Algorithmus werden im ersten Schritt alle $\alpha_i's$, bis auf eines, zufällig zwischen null und eins gewählt. Das verbleibende α_i muss dann so gewählt werden, dass $\sum \alpha_i y_i = 0$ gilt. Mit den gefundenen $\alpha_i's$ kann die Gleichung (1.16) gelöst werden.

Im nächsten Schritt werden alle $\alpha_i's$ um ein zufälliges Delta geändert, wiederum das Letzte so gewählt, dass $\sum \alpha_i y_i = 0$ gilt und mit diesen geänderten $\alpha_i's$ die Gleichung (1.16) gelöst. Ist das Ergebnis besser (größer) als das Vorherige, werden die optimierten $\alpha_i's$ übernommen, ansonsten werden die Werte verworfen. Dieses Vorgehen wird für eine große Anzahl an Iterationen wiederholt. Da es sich um ein quadratisches Problem handelt, wird das globale Maximum und somit die optimale Hyperebene am Ende der Iterationen sicher gefunden.

3. Beispiele

Zur Verdeutlichung der Theorie sind nachfolgend verschiedene Beispiele für linear separierbare und nicht linear separierbare Daten aufgeführt. Zur Lösung des dualen Problems wurde der evolutionäre Algorithmus verwendet.

Linear separierbar

Datenpunkt	Klassifizierung	X1	X2
1	-1	1.0	4.0
2	-1	1.2	2.2
3	-1	2.0	4.0
4	-1	2.5	1.5
5	-1	4.0	1.0
6	+1	3.0	6.0
7	+1	5.0	5.5
8	+1	6.0	3.0
9	+1	6.0	4.5
10	+1	7.0	5.0

Der evolutionäre Algorithmus wurde bei diesen Beispiele 10 Millionen Mal mit einer Schrittweite von 10^{-5} durchlaufen.

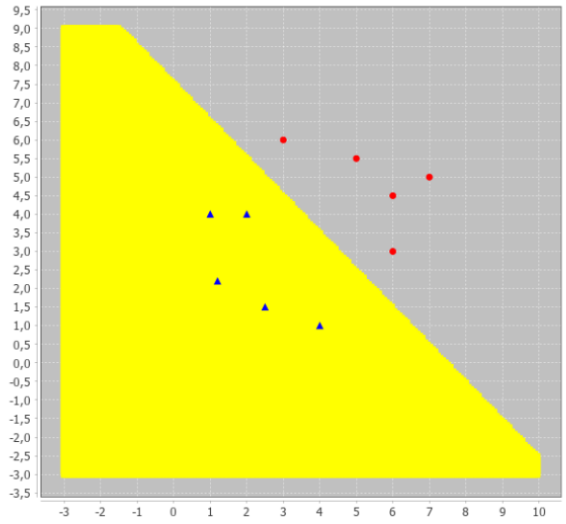


Abb. 7: Linearer Kernel

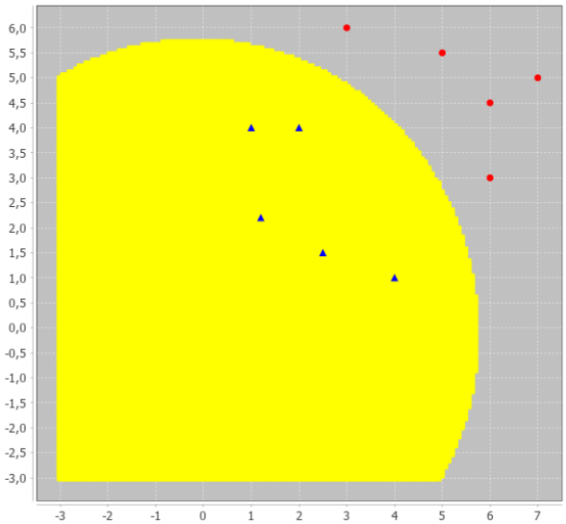


Abb. 8: Polynomialer Kernel

In Abb. 7 ist ersichtlich, dass die Daten linear trennbar und somit mit der linearen Kernelfunktion trainiert werden kann. Auch die Daten in Abb. 8 sind linear separierbar. Hier wurde jedoch testweise eine polynomiale Kernelfunktion zweiten Grades verwendet. Auch hier konnte ein optimales Ergebnis erzielt werden.

Nicht linear separierbar

Datenpunkt	Klassifizierung	X1	X2
1	-1	30	40
2	-1	40	50
3	-1	50	40
4	-1	40	30
5	+1	20	40
6	+1	40	60
7	+1	60	40
8	+1	40	20

Der evolutionäre Algorithmus wurde bei diesen Beispiele 100 Millionen Mal mit einer Schrittweite von 10^{-7} durchlaufen.

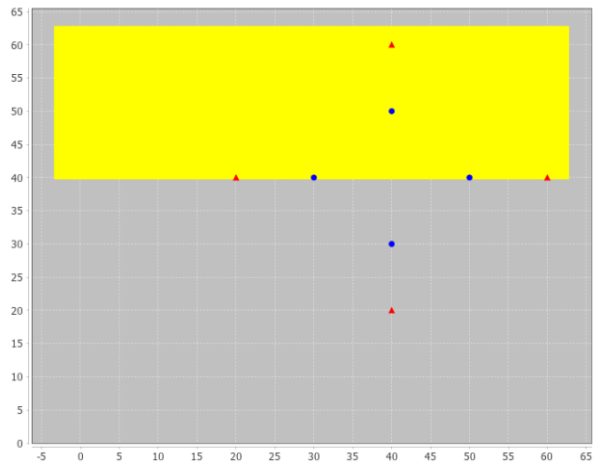


Abb. 9. Linearer Kernel

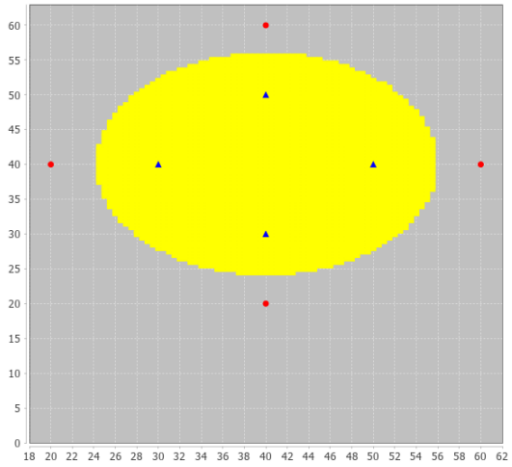


Abb. 10: Polynomialer Kernel

In Abb. 9 ist ersichtlich, dass die Daten nicht linear trennbar und darum nur mit der linearen Kernelfunktion trainiert werden kann, wenn ein gewisser Fehler zugelassen wird. In Abb. 10 wurde für die gleichen Daten eine polynomiale Kernelfunktion zweiten Grades verwendet. Mit dieser Kernelfunktion ist es wieder möglich die Klassen optimal zu trennen.

4. Evaluierung

Der in den Beispielen verwendete evolutionäre Algorithmus zur Lösung des dualen Problems konnte mit einer genügend großen Anzahl an Iterationen und einer kleinen Schrittweite immer eine Lösung finde. Da der Algorithmus sehr einfach gehalten ist findet keine Veränderung der Schrittweite statt. Ebenso wird die komplette Anzahl der Iterationen durchlaufen. Dies führt dazu, dass die Laufzeit für große Datensätze sehr lang ist.
Auch der der SMO Algorithmus konnte bei den Trainingsdaten eine optimale Lösung finden. Da dieser Algorithmus deutlich ausgereifter ist, ist die Optimierung sehr schnell.

Das duale Problem kann also über verschiedene Algorithmen gelöst werden. Die Schwierigkeit bei der Verwendung von Support Vector Machines ist das Finden einer geeigneten Kernelfunktion für Daten die nicht im zweidimensionalen Raum separierbar sind.