Hauptseminar Machine Learning: Support Vector Machines, Kernels

Katja Kunze

13.01.2004

Inhaltsverzeichnis

Kapitel 1

Einführung

Support Vector Machines (**SVM**) ist eine universelle Lern-Prozedur, die auf der statistischen Lern-Theorie basiert. Sie wird benutzt, um verschiedene Arten von Objekten zu klassifizieren. Die Anwendungsgebiete sind vielfältig, man benutzt SVM zum Beispiel in der Optical Character Recognition (**OCR**). SVM weist Ähnlichkeiten zur Klassifizierung mit neuronalen Netzen auf. Dabei verwendet SVM allerdings einfachere und somit auch leicht verständlichere mathematische Methoden. Entwickelt wurde dieser Lern-Algorithmus von V. Vapnik zur Klassifizierung von Daten. SVMs zeichnen sich aus, da sie eine sehr hohe Fähigkeit zur Anwendung auf reale Probleme haben (**Generalisierungsfähigkeit**).

1.1 Grundlagen

Wir verwenden folgende Konzepte für SVMs:

1. Daten mit niedriger Dimension werden abgebildet auf einen neuen **Featurespace** mit höherer Dimension. Unsere Maschine lernt dann mit diesen neuen hochdimensionalen Daten, nicht mit den Orginaldaten. Das erscheint zunächst kompliziert, doch man kann im neuen Raum die Daten mit viel einfacheren, meist linearen Methoden optimal klassifizieren.

2. Man wendet **lineare Schätzer/Näherungsfunktionen** auf den neuen Featureraum an.

3. Das Fehlerrisiko, Daten falsch zu klassifizieren, wird minimiert, da man einfache mathematische Formeln zur Berechnung der optimalen Trennung benutzt.

4. Eine Lösung bietet das **Duale Optimierungsproblem**.

Wir wollen Daten anhand verschiedener Merkmale zu bestimmten Klassen zuteilen. Dabei benutzen wir gegebene Trainingsdaten, anhand derer wir einen Lernalgorithmus entwickeln wollen. Wir beschreiben die zu berücksichtigenden Merkmale als n Parameter und stellen alle Daten als Vektor der Form $\vec{x} = (x_1, ..., x_n)$ dar.
In den Bildern aus **Abbildung 1** wurden Daten, die zu zwei Klassen gehören, in einen Vektorspace transformiert und können dort linear getrennt werden. Der Darstellungsraum unserer Daten heißt **Feature Space (FS)**. Die Vorgehensweise wird an 2 Klassen erklärt (rote Punkte und grüne Quadrate), durch dieselben Schritte können aber später auch mehr Klassen voneinander getrennt werden.

Es lässt sich bei **Abbildung 1** leicht erkennen, dass es im ersten Fall nicht genügt, durch eine einfache Gerade die Daten zu trennen. Diesen Fall nennt man **Underfitting**. Die Lösung im zweiten Bild trennt zwar die gegebenen Daten exakt, doch das Verfahren kann neue Daten schlecht zuordnen, das heißt es läßt sich schlecht generalisieren, was wir mit **Overfitting** bezeichnen. Man braucht ein Verfahren das eine guten Mittelweg findet, so dass man in etwa eine Klassifizierung wie im letzten Beispielbild erhält. Hier wird ein neuer Datenpunkt mit hoher Wahrscheinlichkeit auch zur richtigen Klasse zugeordnet.

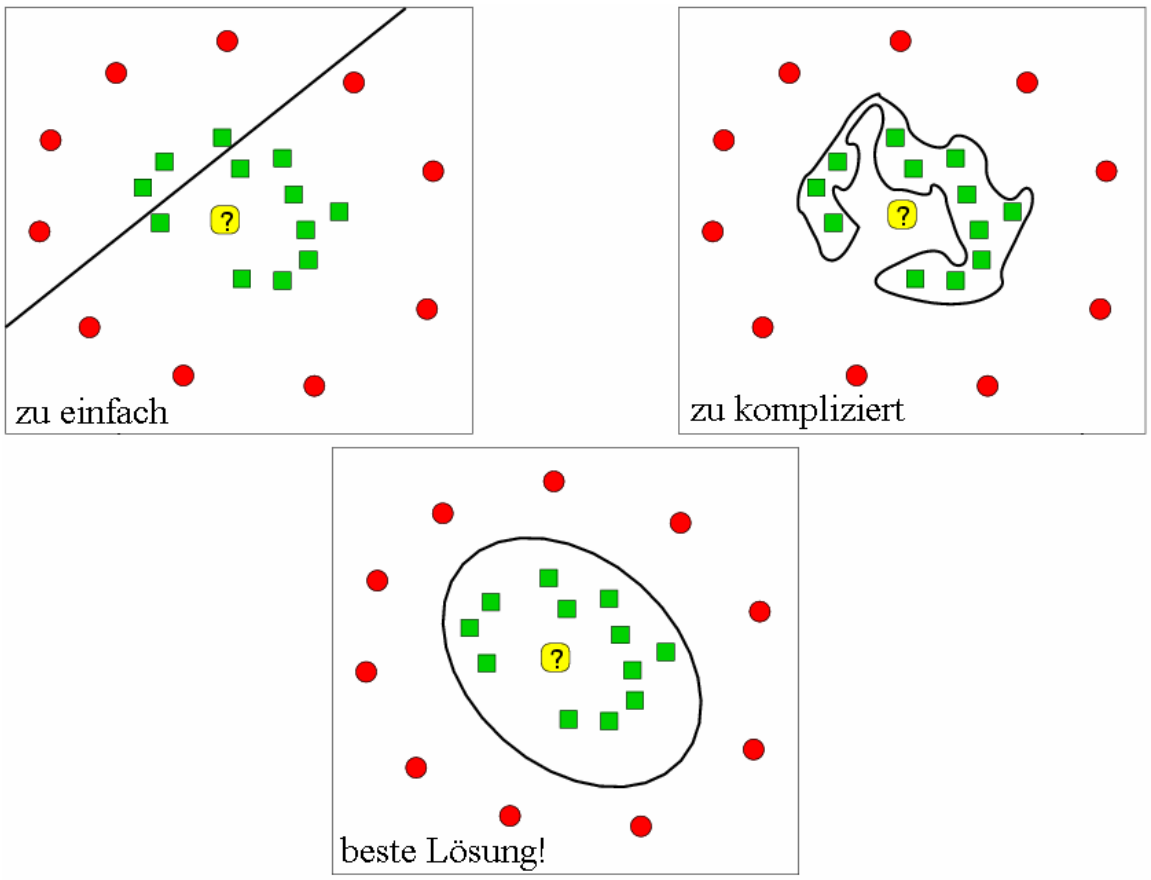


Abbildung 1:
Underfitting / Overfitting /
Optimale Lösung

Kapitel 2

Lineare Seperation

Dieser Abschnitt beschreibt die Trennung der Daten durch Lineare Separation. Dabei geschieht die Trennung im zweidimensionalen Raum durch eine einfache Linie zwischen zwei Gruppen, im Allgemeinen durch eine sogenannte **Hyperebene**.
Wir haben also n Datenpunkte gegeben, die wir folgendermaßen darstellen: $(\vec{x_1}, y_1), ..., (\vec{x_n}, y_n)$. Dabei ist $y_i \in -1, 1$.

Wir wollen die Punkte nun also durch eine Hyperebene trennen. Durch y_i wird dann beschrieben, zu welchen Klasse der Punkt gehört, entweder 1 oder -1. Es gibt allerdings sehr viele Möglichkeiten, eine solche Ebene zu wählen, wie man in **Abbildung 2** sehen kann.
Intuitiv würden wir uns schon für die Ebene in der letzten Abbildung entscheiden. Und zwar aus dem Grund, weil diese Ebene einen Mindestabstand zu jedem Punkt der beiden Klassen hat. Das ist also die Hyperebene, die den größten Spalt der beiden Klassen teilt. Diese Ebene nennen wir auch **Maximum Margin Hyperplane**, Margin = Rand). Erhalten wir einen neuen Datenpunkt, wird dieser mit hoher Wahrscheinlichkeit der richtigen Klasse zugeordnet, je nachdem ob er näher an der einen oder anderen Gruppe liegt. Je größer also der Spalt (Margin), desto bessere Ergebnisse liefert uns die SVM..

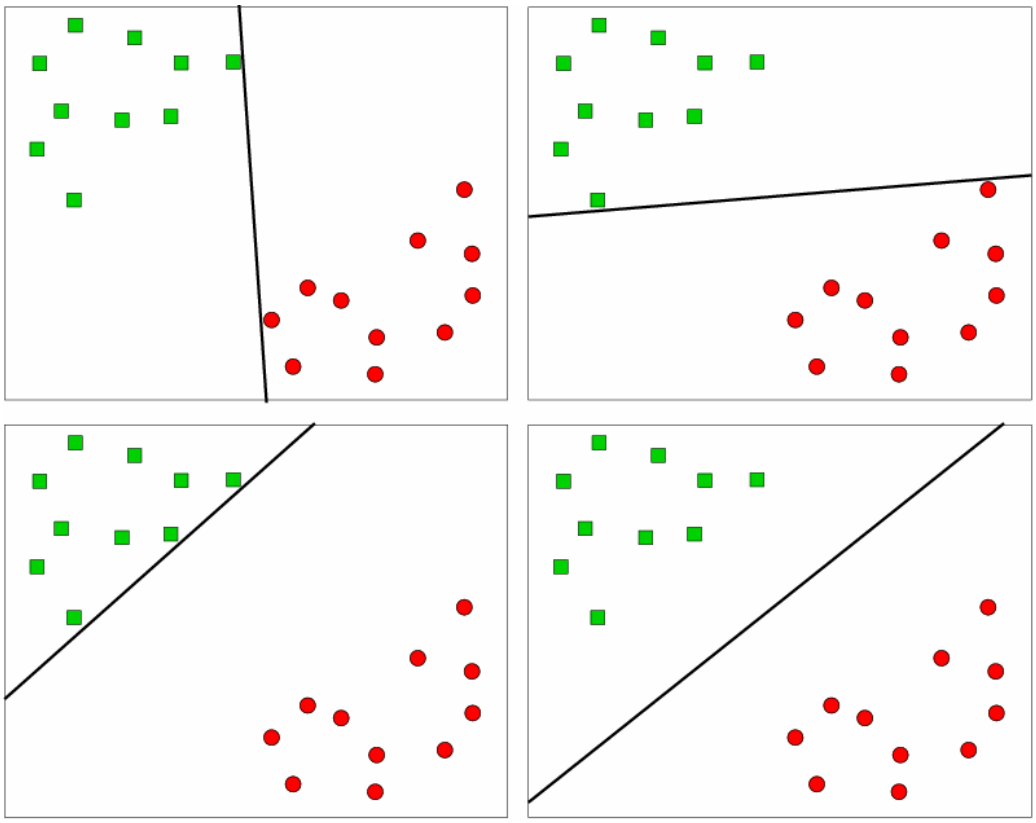


Abbildung 2: Möglichkeiten der Hyperebene

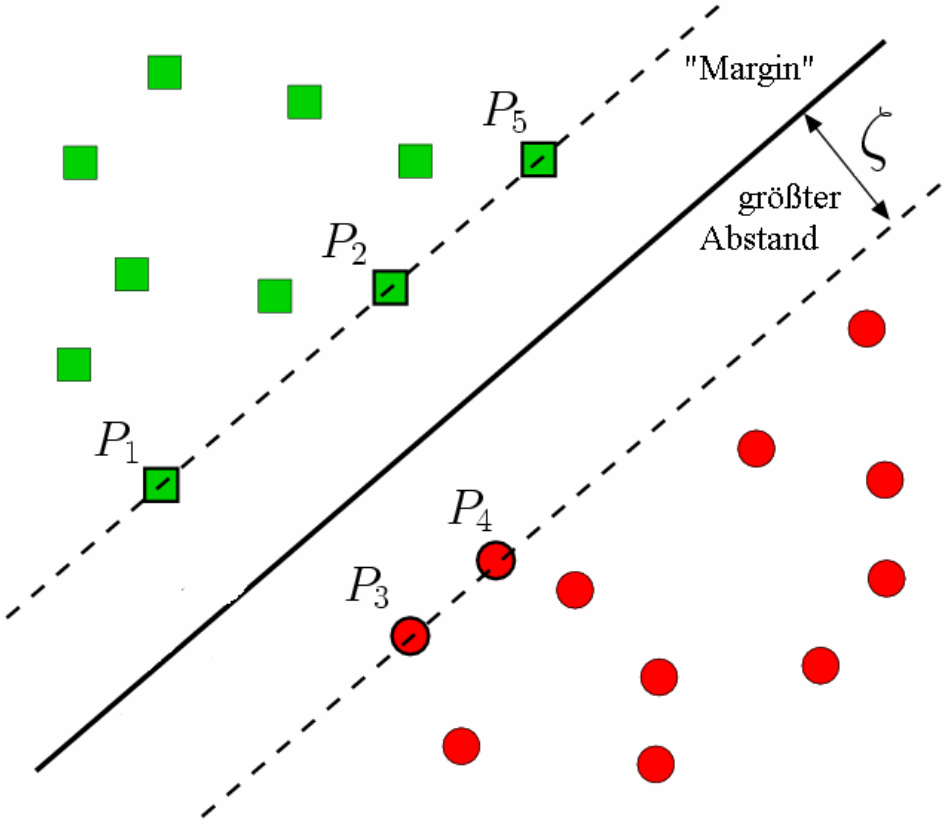


Abbildung 3: Maximum Margin Hyperplane

2.1 Maximum Margin Hyperplane

Jetzt wollen wir herausfinden, wie man die Maximum Margin Hyperplane (**MMH**) am besten findet. Dazu führen wir hier noch ein paar mathematische Symbole ein:

ζ: Der minimalste Abstand eines Datenpunktes zur Hyperebene

P_i: Alle Punkte, für die ζ exakt der geringste Abstand zur Hypereben ist.

Die Objekte P_i heißen **Support Vectors**. Die Support Vectors bestimmen allein die optimale Lage der Ebene, die weiteren Objekte werden dabei nicht mehr berücksichtigt (**Abbildung 3).**

Wir definieren die Ebene wie in der Geometrie üblich:

$$H = H(\vec{w}, b) := \{\vec{x} \in FS | \langle \vec{x}, \vec{w} \rangle + b = 0\} \tag{2.1}$$

Dabei wird das Vektorprodukt verwendet:

$$\langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{d} x_i \cdot y_i \tag{2.2}$$

Der Abstand zur Ebene ist folgendermaßen definiert:

$$dist(\vec{x}, H) = \left| \frac{1}{||\vec{w}||} (\langle \vec{x}, \vec{w} \rangle + b = 0) \right| \tag{2.3}$$

Da die Daten in zwei Klassen klassifiziert sind, beschreiben wir sie mathematisch so:

$$y_i = sgn(\langle \vec{x}, \vec{w} \rangle + b) \tag{2.4}$$

Dabei ist:

$$y_i = \begin{cases} 1 & = \textbf{Objekt oberhalb der Hyperebene} \\ -1 & = \textbf{Objekt unterhalb der Hyperebene} \end{cases} \tag{2.5}$$

Wir müssen nun den Rand beschreiben. Dafür muss der der minimale Abstand $dist(\vec{x}, H)$ so groß wir möglich werden (Maximierungsproblem):

$$\zeta = \min_{\vec{x_i}} \left| \frac{1}{||\vec{w}||} (\langle \vec{x}, \vec{w} \rangle + b = 0) \right| \tag{2.6}$$

Daraus lässt sich mit Hilfe der y_i folgende Ungleichung ableiten:

$$\zeta \leq y_i \frac{1}{||\vec{w}||} (\langle \vec{x}, \vec{w} \rangle + b = 0) \tag{2.7}$$

Diese Ungleichung hängt von ζ und $||\vec{w}||$ ab. Wir können diese Variablen glücklicherweise in Beziehung miteinander setzen, um das Problem der Minimierung zu lösen. Da die Ebene H kanonisch ist, gilt:

$$\zeta \cdot ||\vec{w}|| = 1 \Rightarrow \zeta = \frac{1}{||\vec{w}||} \tag{2.8}$$

Wir erhalten also die größte Distanz, wenn wir ζ, also $\frac{1}{||\vec{w}||}$ maximieren, wobei die Ungleichung

$$y_i(\langle \vec{x}, \vec{w} \rangle + b) \geq 1 \tag{2.9}$$

bzw.

$$y_i(\langle \vec{x}, \vec{w} \rangle + b) - 1 < 0 \tag{2.10}$$

berücksichtig werden muss.
Wir können auch $\frac{1}{2}||\vec{w}||^2$ minimieren. Dies wird beschränktes Optimierungsproblem genannt.

Um dieses Problem zu lösen, führen wir hier die Lagrangefunktion ein. Durch die Einführung von Lagrangemultiplikatoren $\alpha_i > 0$ können wir die Beschränkung in die Funktion mit einfließen lassen.

$$L(\vec{w}, b, \alpha) = \frac{1}{2}||\vec{w}||^2 - \sum_{i=1}^{d} \alpha_i (y_i(\langle \vec{x}, \vec{w} \rangle + b) - 1) \tag{2.11}$$

Die Funktion kann maximiert werden, indem man sie nach b und $\vec{w}$ ableitet:

$$\frac{\partial}{\partial b} L(\vec{w}, b, \alpha) = 0 \frac{\partial}{\partial \vec{w}} L(\vec{w}, b, \alpha) = 0 \tag{2.12}$$

Das führt schließlich zu:

$$\sum_{i=1}^{d} \alpha_i y_i = 0 \tag{2.13}$$

und

$$\vec{w} = \sum_{i=1}^{d} \alpha_i y_i x_i. \tag{2.14}$$

Wenn man die letzten beiden Terme in der Langrangefunktion ersetzt, bekommt man das sogenannte Duale Optimierungsproblem:

$$maximize_{\alpha \in R^n} \; L(\vec{\alpha}) = \sum_{i=1}^{d} \alpha_i - \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \alpha_i \alpha_j y_i y_j \langle \vec{x_i}, \vec{x_j} \rangle \tag{2.15}$$

Dieses Problem kann gelöst werden mit verschiedenen Algorithmen aus der Optimierungstheorie. Die Lösungen sind zum Teil nicht ganz einfach, deshalb wird an dieser Stelle auf eine genaue Erläuterung verzichtet.

Falls die Daten gut linear trennbar sind, liefert dieser Ansatz gute Ergebnisse. Gibt es Daten, die nicht optimal bei den anderen Punkten der Klasse liegen, also 'Ausreiser', gibt es auch andere Methoden das Problem lösen. Eine Möglichkeit ist eine Erweiterung der Maximum Margin Hyperplane, die Soft Margin Hyperplane.

2.2 Soft Margin Hyperplane

In vielen Fällen können die Daten nicht linear getrennt werden. Abbildung 4 zeigt, das trotz einiger Ausreiser trotzdem eine Ebene gefunden werden kann, die die beiden Klassen gut trennt, wobei einige Fehler zugelassen werden. Für diesen Fall definieren wir Variablen ζ_i, die genau den Abstand eines falsch positionierten Punktes P_i messen.
Jetzt müssen wir nicht nur $\frac{1}{2}||\vec{w}||^2$, sondern auch die Fehler, der ja bei linearer Seperation auftritt, also ζ_i minimieren. Wir lassen

dabei das Fehlergewicht C zu, mit dem wir den Abstand zur Hyperebene multiplizieren:

$$minimize \ \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1} n\zeta_i \tag{2.16}$$

Natürlich muss dabei die Bedingung

$$y_i(\langle\vec{x},\vec{w}\rangle + b) \geq 1 - \zeta_i, \quad \zeta \geq 0 \ \forall \ i \tag{2.17}$$

berücksichtigt werden. Das Fehlergewicht C liegt dabei zwischen einem großen Spalt mit großen Fehlern gegenüber einem kleinem Spalt und kleinen Fehlern.

Das Verfahren der Soft Margin Hyperplane beschränkt sich auf den Fall, das bei einem bestimmten Trainingsdatensatz nur ein paar Punkte falsch oder für unsere Methode ungünstig positioniert sind und der übrige Datensatz annähernd linear separierbar ist. Manchmal ist die Grenze zwischen zwei Klassen allerdings sehr schwer linear aufzuzeigen und zu berechnen. Für solch einen komplizierten Fall, und bei den meisten realen Anwendungen ist dies der Fall, wird auf eine andere Methode zurückgegriffen, die im nächsten Kapitel beschrieben wird..

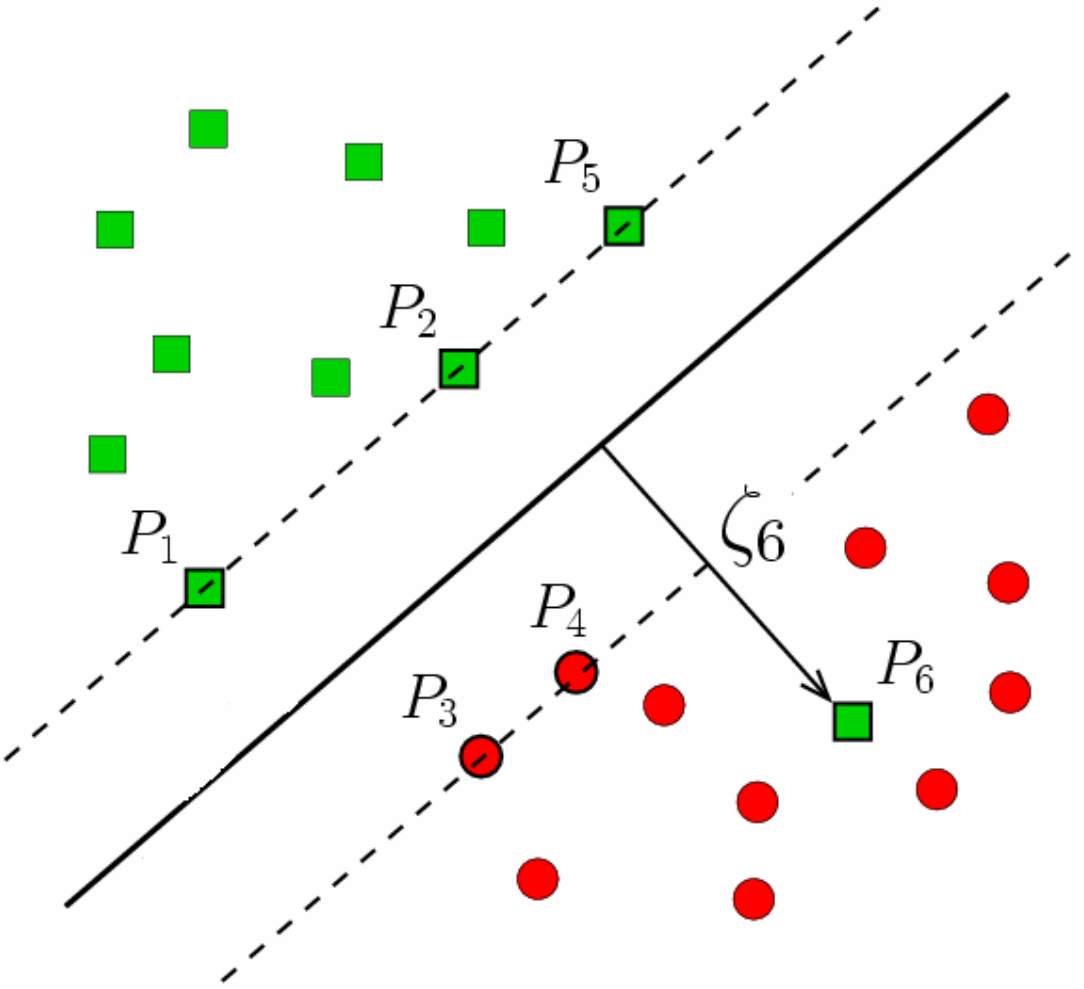


Abbildung 4: Soft Margin Hyperplane

Kapitel 3

Kernels

Die grundsätzliche Idee von Kernels ist es, den Datensatz, bei dem es nicht gelingt, ihn direkt linear zu trennen, in einen Raum zu transformieren, in dem Lineare Seperation möglich ist. Das heißt also wir transformieren die Daten vom Eingaberaum, den sogenannten Input Space(IS), in einen neuen Raum mit höherer Dimension, den Feature Space (FS), um zum Schluss wesentlich einfachere, nämlich lineare Formeln darauf anzuwenden und somit optimal zu trennen.

Dies wird durch die Funktion Φ ermöglicht, die auf jeden Datenpunkt angewendet wird. im FS rechnen wir dann mit $\Phi(\vec{x})$ anstelle von $\vec{x}$ (Abbildung 5).

Das nächste Beispiel soll veranschaulicht etwas besser, wie Kernels arbeiten. Der IS ist hierbei der R^2. Wir transformieren den Satz auf den R^3 durch folgende Abbildung:

$$\Phi : (x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2) \tag{3.1}$$

Abbildung 6 zeigt, das die Konstruktion einer Hyperebene jetzt ganz einfach ist..

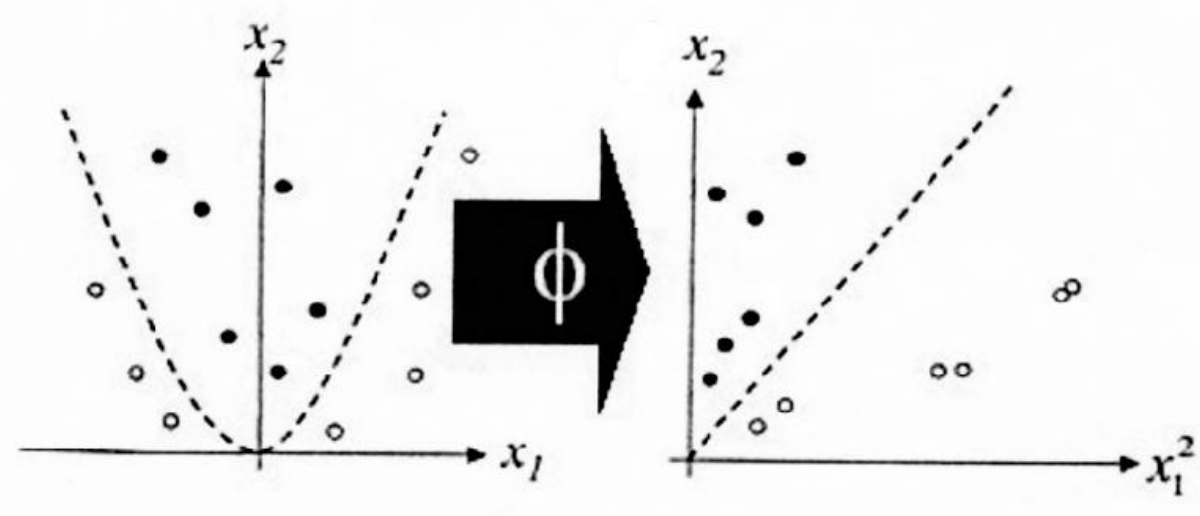


Abbildung 5: Transformation durch Φ

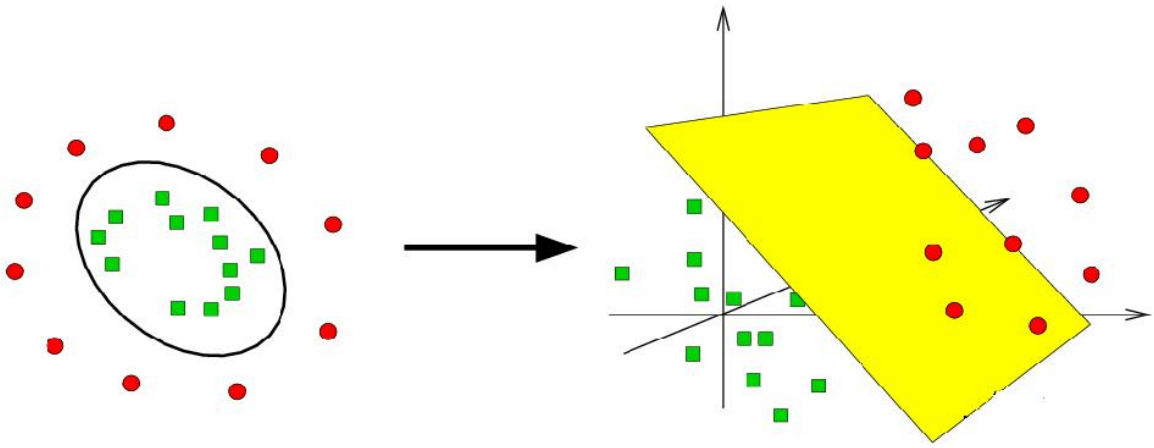


Abbildung 6: Hyperebene im 3 dimensionalen Raum

Mathematisch bedeutet das, daß das Verfahren nach der Transformation genauso arbeitet wie bei den Maximum Margin Hyperplanes. Es wird die Lagrangefunktion maximiert und man erhält das Duale Optimierungsproblem:

$$maximize_{\alpha \in R^n} \; L(\vec{\alpha}) = \sum_{i=1}^{d} \alpha_i - \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \alpha_i \alpha_j y_i y_j \langle \Phi(\vec{x_i}), \Phi(\vec{x_j}) \rangle \tag{3.2}$$

Man sieht, dass hier die Funktion Φ nur Auswirkungen auf das Skalarprodukt hat. Nun kann leider die Berechnung des Produktes $\langle \Phi(\vec{x_i}), \Phi(\vec{x_j}) \rangle$ sehr schwierig sein, wenn die Dimension sehr hoch ist. Hier kommt der eigentliche Kernel-Trick ins Spiel. Wir berechnen anstelle des Skalarproduktes eine Funktion K, die folgendermaßen definiert ist:

$$K_\Phi(\vec{x}, \vec{y}) := \langle \Phi(\vec{x}), \Phi(\vec{y}) \rangle \tag{3.3}$$

Das folgende Rechenbeispiel soll dies klarmachen, das die Berechnung der Funktion genau das Skalarprodukt ergibt:

Gegeben sind zwei Objekte in Vektorform im IS R^2:

$$\vec{x} = (x_1, x_2) \quad \vec{y} = (y_1, y_2) \tag{3.4}$$

Darauf wenden wir die selbe Abbildung wie im Bild oben an:

$$\Phi : (x_1, x_2) \mapsto (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \tag{3.5}$$

Daraus lässt sich das Skalarprodukt einfach berechnen:

$$\begin{aligned} \langle \Phi(\vec{x}), \Phi(\vec{x}) \rangle &= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(y_1^2, \sqrt{2} y_1 y_2, y_2^2) \\ &= x_1^2 y_1^2 + 2 x_1 y_1 x_2 y_2 + x_2^2 y_2^2 \\ &= (x_1 y_1 + x_2 y_2)^2 \\ &= \langle \vec{x}, \vec{y} \rangle^2 =: K(\vec{x}, \vec{y}) \end{aligned} \tag{3.6}$$

Es reicht also hier aus, nur das Quadrat von x und y im R^2 zu berechnen. Der eigentliche Trick dabei ist, dass sich die Kernelfunktionen sehr leicht bestimmt werden können, ohne die Funktion selbst auszurechnen und dass sie sich genauso wie ein Skalarprodukt verhalten. Es spart also gerade in hochdimensionalen Vektorräumen Rechenaufwand und Zeit, da man die Funktion keineswegs für jeden Vektor berechnen muss.
Es liegt nahe, dass man nicht jede Funktion als Kernelfunktion benutzen kann. Eine geeignete Funktion K_Φ muss dazu drei Kriterien erfüllen:

1. Symmetrie

$$K_\Phi(\vec{x}, \vec{y}) \quad = \quad K_\Phi(\vec{y}, \vec{x}) \tag{3.7}$$

2. Satz von Cauchy-Schwarz

$$K_\Phi(\vec{x}, \vec{y})^2 \quad \leq \quad K_\Phi(\vec{x}, \vec{x}) \cdot K_\Phi(\vec{y}, \vec{y}) \tag{3.8}$$

3. Die Matrix mus positiv (semi-definit) sein.

$$G(K) = \begin{pmatrix} K(\vec{x_1}, \vec{x_1}) & \cdots & K(\vec{x_1}, \vec{x_n}) \\ \vdots & & \vdots \\ K(\vec{x_n}, \vec{x_1}) & \cdots & K(\vec{x_n}, \vec{x_n}) \end{pmatrix} \tag{3.9}$$

Grundsätzlich könnte jede Funktion, die diese Kriterien erfüllt, als Kernelfunktion benutzt werden, doch es kommt natürlich auf die Art der Daten und ihre Verteilung an, welcher Kernel am sinnvollsten ist und am einfachsten berechnet werden kann.
Ein paar Beispiele für sehr häufig benutzte Kernelfunktionen sind im Folgenden aufgelistet:
linear:

$$K(\vec{x}, \vec{y}) := \langle \vec{x}, \vec{y} \rangle \tag{3.10}$$

Radial-Basis-Kernel:

$$K(\vec{x}, \vec{y}) := exp(-\gamma \cdot |\vec{x} - \vec{y}|^2) \tag{3.11}$$

poynomiell:

$$K(\vec{x}, \vec{y}) := (\langle \vec{x}, \vec{y} \rangle + 1)^d \tag{3.12}$$

sigmoid:

$$K(\vec{x}, \vec{y}) := tanh(\gamma \cdot (\vec{x} - \vec{y}) + c) \tag{3.13}$$

Oft werden in der Realität auch Kombinationen von Funktionen gewählt. Man hat hierbei eine Fülle von Möglichkeiten, sich eine geeignete Funktion zu wählen.

Kapitel 4

Anwendungen

4.1 Beispiel1: Gesichtserkennung

In der Gesichtserkennung wurden SVMs eingesetzt. Es handelt sich dabei nicht um das Auffinden von Personen z.B. in einer Datenbank (engl. Face-Recognition), sondern um die grundsätzliche Erkennung von Gesichtern auf Bildern (Face-Detection), im Vergleich zum Hintergrund.

Dabei wurden als Eingabe und zum Lernen fertige Bilder genommen, die allerdings vorverarbeitet werden mussten. Man musste also folgende Schritte durchlaufen, um zu einem guten Ergebnis zu kommen:

1. Ausschneiden von gleichgroen Fenster (z.B 20x20 Pixel) aus dem Bild

2. Bild skalieren (Verkleinern/Vergrern) und wieder Fenster zurechtschneiden

3. Verschiedene Fenster vorverarbeiten (Kontraste unter Graustufen hervorheben, Schrfe...)

4. SVM klassifiziert 'Gesicht' oder 'Nicht-Gesicht'

5. Falls 'Gesicht': Markierung mit einem Rahmen

6. 'Bootstrapping': Die fälschlich als 'Gesicht' klassifizierten Objekte wurden wiederum zum Lernen als Beispiele für 'Nicht-Gesicht' aufgegriffen

Die Schwierigkeit besteht hierbei, das es eine starke Varibilität unter den zu findenden Mustern gibt, aufgrund vieler unterschiedlicher Merkmale wie Gesichtsform, Hautfarbe, Winkel der Aufnahme etc. Außerdem gibt es natülich wesentlich mehr Möglichkeiten, Objekte auf einem Bild zu finden, die zur Kategorie 'Nicht-Gesicht' gehören als 'Gesicht'.
Die Bilder zeigen, dass schon sehr gute Ergebnisse mit SVMs erzielt werden konnten, vor allem auch wenn man unscharfe und sehr unterschiedlich 'gute' Bilder klassifizieren sollte.
.

Abbildung 7:
Face Recognition

4.2 Beispiel2: Handschrifterkennung

SVMs werden auch zur Handschrifterkennung eingesetzt.

Der verwendete Kernel war dabei ein 'Gaussian-Dynamic-Time-Warping-Kernel'. Dies ist ein Beispiel, in dem geschriebene Zahlen maschiniell erkannt werden sollen. Hier ist zwar die Eingabe nicht so verschiedenartig wie bei Bildern, allerdings muss in 10 verschiedene Klassen eingeteilt werden. Hat man also mehrere Klassen, liefert die SVM ebenfalls gute Ergebnisse..

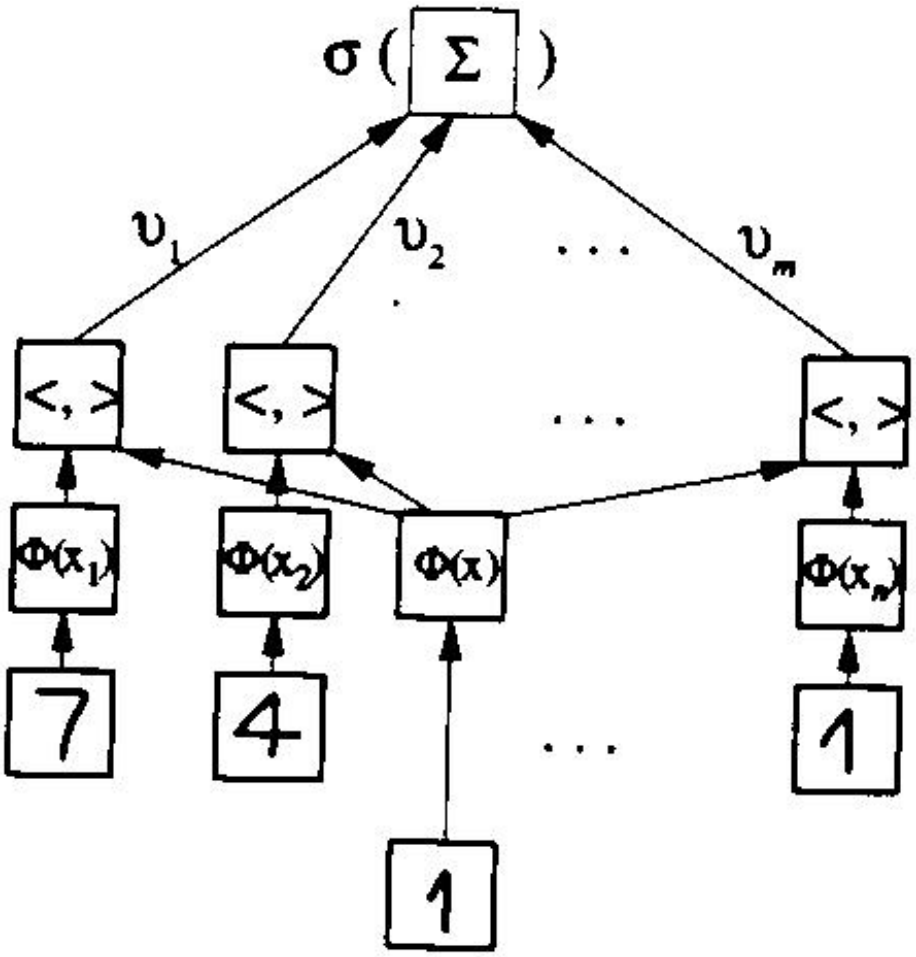


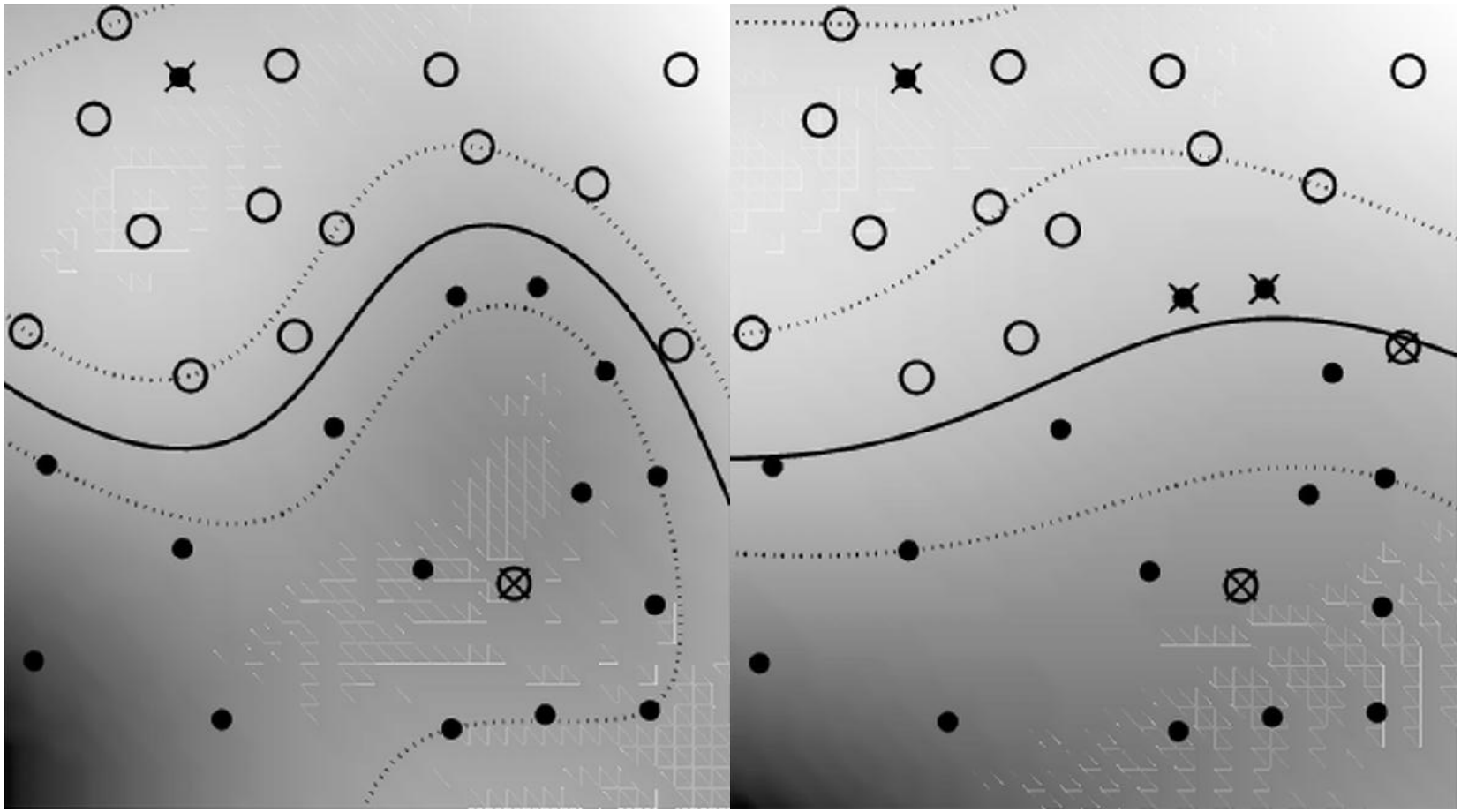
Abbildung 8:
Handschrifterkennung

4.3 Beispiel3: Spamfilterung

Weitere Beispiele sind die Spamfilterung. Dabei bekommt die SVM als Eingabe zum Beispiel Worte wie 'FREE' und viele weitere, von denen bekannt ist, dass sie sehr häufig in den unerwünschten Spam-Mails auftauchen. Eine eintreffende Email wird dann auf die Anzahl solcher Worte durchsucht und schließlich als 'Spam' oder 'Nicht-Spam' klassifiziert. Dabei kann auch Merkmale wie kritische Absendeadressen oder bestimmte Zeichen im Header mitbestimmen lassen.

4.4 Beispiel4: Bioinformatik

In der Bioinformatik arbeitet man häufig mit einem eher kleinen Datensatz, aber dabei handelt sich es um hochdimensionale Daten. In der Genetik ist bekanntlich ein großes Forschungsgebiet die Entschlüsselung von Gensequenzen. Um ein Protein beispielsweise zu klassifizieren, wird die Sequenz in einen 20-dimensionalen Vektor konvertiert, um die Aminosäuresequenz, aus der es besteht, darzustellen. Folgende Bilder sind Aufnahmen aus der Medizin, bei verschiedene Punkte klassifiziert werden sollen. Dabei ist die Trennung im ersten Bild sehr einfach, im dritten allerdings wesentlich komplizierter..



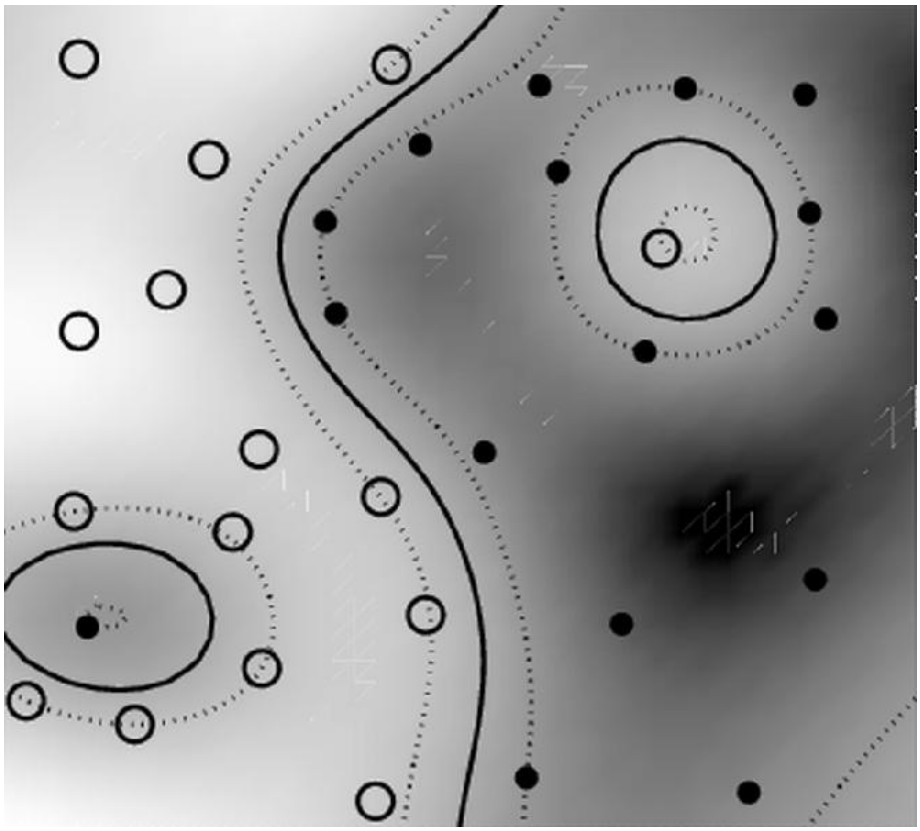


Abbildung 9:
Bioinformatik:
Ergebnisse mit einfacher, mittlerer und hoher Komplexität

Kapitel 5

SVM - Zusammenfassung

Was ist eine Support Vector Machine?
Es ist eine *Hyperebene in einem Feature Space mit maximaler Trennspannex*, **die im hochdimensionalen Raum durch eine** *Kernelfunktion* **konstruiert wird.**

5.1 Pros

-Klassifizierung sehr schnell mglich
-Hohe Przision und geringe Fehlerwahrscheinlichkeit
-Benutzung von relativ einfachen und gut verstndlichen Funktionen und Rechentechniken
-Effektiv auf Daten mit vielen Merkmalen
-Gute Performance in Real-World-Anwendungen

5.2 Contras

-Bentigt sehr viel Zeit frs Lernen (Entscheidung fr die richtige Hyperebene)
-Es muss empirisch nach der geeigneten Kernelfunktion gesucht werden, was evtl. bei komplexen Problemen doch etwas schwerer sein kann.
-Neues Training erforderlich, falls neue Eingabedaten verschieden sind.

-Richtige Vorverarbeitung der Daten (z.B. Skalierung, welche Merkmale sind wichtig) ist essentiell

Zusammenfassend kann man sagen, das SVMs ein sehr interessantes Forschungsgebiet bieten und es viele Möglichkeiten zur Anwendung gibt.